УДК 004.94:681.5
DOI https://doi.org/10.32782/2663-5941/2025.5.2/23

Манько Д.В.
Національний університет «Запорізька політехніка»

Миронова Н.О.
Національний університет «Запорізька політехніка»

Шаптала С.В.
Національний університет «Запорізька політехніка»

Куляба-Харитонова Т.І.
Національний університет «Запорізька політехніка»

СИСТЕМА СИНХРОНІЗАЦІЇ ТРИВИМІРНОЇ МОДЕЛІ ЦИФРОВОГО ДВІЙНИКА РОБОТОТЕХНІЧНОГО ПРИСТРОЮ З ПОТОКОВИМИ IOT-ДАНИМИ

У статті представлено дослідження та реалізацію системи синхронізації тривимірної моделі цифрового двійника робототехнічного пристрою з потоковими IoT-даними у середовищі Webots. Актуальність роботи зумовлена зростаючою потребою у побудові цифрових двійників для робототехніки, здатних працювати в режимі, наближеному до реального часу, для завдань моделювання, тестування та оптимізації керування. Об'єктом дослідження є процес оновлення параметрів тривимірної моделі робототехнічного пристрою на основі зовнішніх потокових IoT-даних. Предметом дослідження є алгоритми та програмні засоби для реалізації системи синхронізації цифрового двійника. У роботі обґрунтовано вибір середовища Webots як оптимальної платформи для моделювання та візуалізації робототехнічних систем, що забезпечує реалістичну фізику, підтримку API та можливість інтеграції з зовнішніми сервісами. Запропоновано архітектуру системи, що складається з трьох основних модулів: збору та попередньої обробки IoT-даних, серверної обробки і надання даних через API, а також візуалізації та керування моделлю у Webots. Розроблено наступні алгоритми: зчитування і передачі даних з таблиці CSV у форматі JSON, обробки даних і відповіді на запити клієнтів, та періодичного оновлення стану моделі на основі отриманих параметрів. Алгоритми забезпечують обмін інформацією у режимі реального часу з мінімальною затримкою та високою стабільністю. Проведено експериментальне дослідження, під час якого оцінювалась швидкість оновлення моделі, точність синхронізації та стійкість системи до збоїв мережі. Результати показали, що затримка між надходженням даних та їх відображенням у тривимірній моделі не перевищує 200–300 мс, система коректно обробляє втрачені пакети та забезпечує реалістичну візуалізацію поведінки цифрового двійника. Аналіз впливу параметрів продемонстрував критичну роль оптимізації інтервалів запитів, формату переданих даних та обробки помилок на стороні API. Отримані результати підтверджують ефективність запропонованих алгоритмів синхронізації і доцільність використання розробленої архітектури для побудови цифрових двійників у робототехніці. Розроблений підхід може бути розширений шляхом інтеграції з реальними IoT-пристроями, застосування протоколів WebSocket для зменшення затримок, а також впровадження алгоритмів прогнозування на основі машинного навчання. Подальші дослідження планується спрямувати на масштабування системи, адаптацію до промислових умов і створення гібридних цифрових двійників з підтримкою віддаленого управління.

Ключові слова: *цифровий двійник, алгоритми синхронізації, IoT, 3D-модель, Webots, REST API, Node.js, Python, робототехніка, потокові дані.*

Постановка проблеми. Сучасні тенденції розвитку Індустрії 4.0, зокрема активне впровадження концепції Інтернету речей (IoT), зумовлюють зростання значущості цифрових двійників у галузі робототехніки. Цифровий двійник являє собою віртуальну копію фізичного об'єкта, що здатна динамічно оновлюватися на основі даних, отриманих від сенсорів у режимі, наближеному до реального часу. Такий підхід дозволяє здійснювати моніторинг стану системи, прогнозувати її поведінку та оптимізувати процеси керування.

Разом із тим, розробка сучасних систем цифрових двійників стикається з низкою проблем. Найбільш суттєвим завданням є забезпечення стабільної та своєчасної синхронізації між зовнішніми потоковими даними та віртуальною тривимірною моделлю об'єкта. Невідповідність у часових характеристиках або втрата даних призводять до зниження достовірності візуалізації та обмежують можливості використання цифрового двійника у реальних умовах. До основних проблем належать: затримки в обміні інформацією між джерелами IoT-даних і тривимірною моделлю, що зумовлюють спотворення відображення фізичного стану об'єкта; нестабільність мережі та втрата пакетів, які знижують точність синхронізації; необхідність масштабування системи при збільшенні обсягу даних або частоти оновлень; потреба у реалізації гнучких алгоритмів, здатних адаптуватися до зміни умов роботи. Отже, постає проблема створення системи, що забезпечує узгодження станів фізичного та цифрового об'єктів. Її вирішення можливе шляхом розробки методів обробки даних, оптимізації частоти обміну та впровадження механізмів контролю помилок. Тому робота спрямована на дослідження та реалізацію системи синхронізації цифрового двійника робототехнічного пристрою з потоковими IoT-даними.

Аналіз останніх досліджень і публікацій. Проблема синхронізації цифрових двійників є актуальною у сучасних дослідженнях, що охоплюють питання оптимізації передачі даних, мінімізації затримок та забезпечення узгодженості станів між фізичними і віртуальними об'єктами. У роботі [1] сформульовано математичну модель проблеми синхронізації цифрових двійників у виробничих середовищах. Автори пропонують оптимізаційний підхід для визначення моментів оновлення станів, що дозволяє зменшити похибку та час відгуку системи. Дослідження доводить, що адаптивні алгоритми синхронізації є ефективнішими за методи з фіксованими інтервалами. Дослідження [2] спрямоване на аналіз впливу мережевих параметрів на якість синхронізації цифрових двійників. Запропоновано показник Twin Alignment Ratio (τ), який дозволяє кількісно оцінити ступінь узгодженості фізичного та цифрового об'єктів. Встановлено, що вибір протоколу (TCP/UDP) та налаштування мережі мають вирішальне значення для зменшення затримок. У роботі [3] здійснено систематизацію 11 шаблонів синхронізації (time-driven, event-driven, hybrid, adaptive). Зроблено висновок, що гібридні та адаптивні алгоритми є найбільш придатними для кіберфізичних систем, зокрема робототехніки, завдяки їх здатності балансувати між частотою оновлень та ефективністю використання ресурсів. Робота [4] присвячена оцінюванню синхронізації цифрових двійників у контексті робототехнічних систем. Автори проводять експерименти з роботами, що взаємодіють із цифровими копіями, та показують, що алгоритми, здатні реагувати на контекстні зміни, забезпечують значно вищу точність відображення динаміки об'єкта. Дослідження [5] пропонує багаторівневу архітектуру для синхронізації цифрових двійників у системах IoT. Застосування розподілених мережевих структур дає змогу знизити затримки та підвищити масштабованість, що особливо важливо для реальних промислових застосувань. У роботі [6] розглядається інтеграція цифрових двійників з безпілотними літальними апаратами (UAV). Запропоновано метрику Age of Digital Twin (AoDT) для оцінки актуальності даних, що використовується як показник ефективності синхронізації. Доведено, що оптимізація топології мережі та вибір частоти оновлень дозволяє суттєво покращити якість взаємодії між фізичним і цифровим об'єктом.

Результати проведеного аналізу свідчать, що для забезпечення високої точності синхронізації цифрових двійників необхідне застосування адаптивних алгоритмів, здатних враховувати зміну умов мережі та варіативність обсягів даних. Впровадження спеціалізованих метрик (τ, AoDT) дозволяє кількісно оцінювати ефективність обраних рішень, а поєднання гібридних схем обміну з оптимізованими протоколами забезпечує стабільність і масштабованість системи.

Метою статті є дослідження та розробка системи синхронізації тривимірної моделі цифрового двійника робототехнічного пристрою з потоковими IoT-даними для забезпечення її оновлення в режимі, наближеному до реального часу.

Виклад основного матеріалу. Для реалізації цифрового двійника обрано чотириколісного

мобільного робота, знайденого на платформі GrabCAD у збірці під назвою Arduino 4WD Buggy Kit. Ця модель містить основні елементи, які потрібні для симуляції: колеса, шасі, двигуни, а також дозволяє реалізувати просту кінематику пересування, що ідеально підходить для тестування обміну даними з зовнішніми джерелами.

Після завантаження архіву з моделлю у форматі .f3d, її відкрито у середовищі Autodesk Fusion 360. На етапі підготовки моделі виконано розділення всієї збірки на окремі логічні компоненти: нижня платформа, верхня платформа, колеса, двигуни та монтажні елементи. Для кожного з компонентів створено окремі тіла, що дозволило точніше контролювати поведінку моделі під час симуляції у Webots.

Оскільки повна геометрія моделі містила велику кількість дрібних елементів (гвинти, болти, фаски), що негативно впливали на продуктивність симуляції, було здійснено оптимізацію моделі (рис. 1). Для цього обрані об'єкти (chasis+motors, wheels) експортовані у форматі .obj з додатковим файлом .mtl, після чого зменшено кількість полігонів. Процедура редукції мешу дозволила зберегти зовнішній вигляд моделі, одночасно значно знизивши навантаження на рушій Webots при візуалізації фізики руху.

Після завершення редагування модель протестовано на сумісність із середовищем Webots, зокрема на правильність орієнтації компонентів та коректність масштабування. Підтверджено, що модель придатна для подальшої інтеграції у симуляційне середовище та підключення до системи обміну даними.

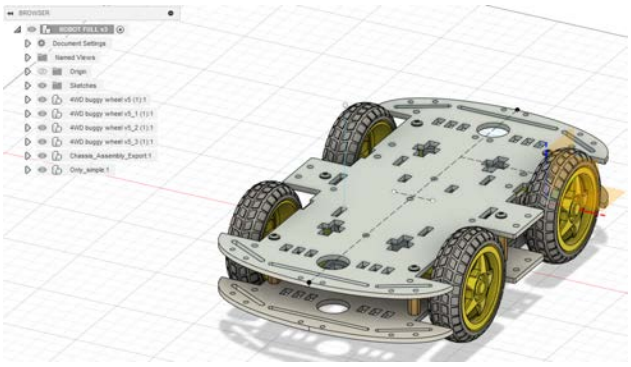

Рис. 1. Оптимізована модель робота у Fusion 360

Для візуалізації роботи цифрового двійника та перевірки коректності передачі даних у режимі, наближеному до реального часу, створено симуляційне середовище у Webots. На першому етапі проєкту було підготовлено окрему сцену, яка містить лише необхідні об’єкти – тривимірну модель робота та плоску поверхню (Ground) як основу (рис. 2). Такий підхід дозволив зосередитись на налагодженні обміну даними без зайвого навантаження на рушій візуалізації.

Імпортована модель робота підготовлена попередньо у Fusion 360, експортована у форматі .obj з відповідним .mtl.

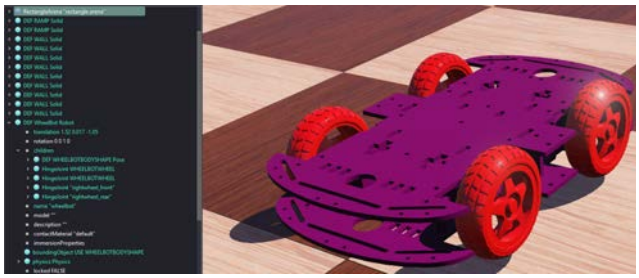

Рис. 2. Модель робота у Webots

Спочатку створено новий об’єкт Robot, з яким в подальшому і буде вестись робота. У ньому створені 5 Children:

– WHEELBOTBODYSHAPE Pose – сама основа;

– HingeJoint WHEELBOTWHEEL – переднє ліве колесо;

– HingeJoint WHEELBOTWHEEL – заднє ліве колесо;

– HingeJoint rightwheel_front – переднє праве колесо;

– HingeJoint rightwheel_rear – заднє праве колесо;

Сама основа фіолетового кольору, щоб добре виділялась на фоні полу. У основі створено geometry Mesh, у який закинуто mesh зі Fusion 360. Bounding object для основи був використаний той же mesh (рис. 3). Для кожного з HingeJoint додано device Brake та Rotational motor. Колеса червоного кольору, щоб видно було як вони крутяться та можна було б одразу дізнатися якщо десь помилка. У колес так само geometry Mesh з Fusion 360, але Bounding object у них – це циліндр, щоб при симуляції не було занадто багато полігонів для опрацювання. Для кожного колеса використано один і той же mesh, але розвернутий для кожного у свою сторону. Також для усього робота призначено його фізику: вагу, інерцію та інші фізичні параметри.

Архітектура системи синхронізації. Розроблено трирівневу архітектуру систему синхронізації (рис. 4), яка дозволяє передавати симульовані дані від пристрою до тривимірної моделі у Webots. Кожен компонент виконує окрему роль в обробці, передачі та візуалізації даних.

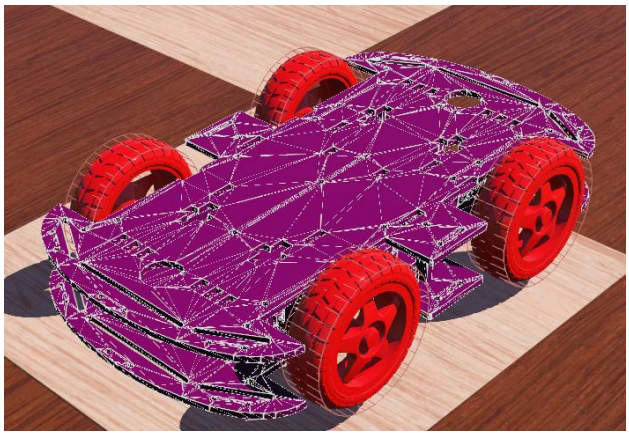

Рис. 3. Bounding object для основи

Перший рівень – генератор даних, розроблений на Python. Він зчитує значення з таблиці формату CSV, що містить часові мітки та числові параметри (наприклад, швидкість обертання коліс). Після зчитування дані формуються у форматі JSON і надсилаються на сервер через HTTP POST-запит. Передача здійснюється з певним інтервалом часу, щоб симулювати реальний потік значень.

Другий рівень – API-сервер, реалізований на платформі Node.js. Цей сервер приймає запити від Python-скрипта, зберігає останні отримані дані в оперативній пам'яті та надає доступ до них через HTTP GET-запити. У відповідь на GET-запит сервер надсилає JSON-об'єкт із актуальними значеннями параметрів, які використовуються для оновлення стану 3D-моделі.

Третій рівень – Webots-контролер, написаний на C. У межах симуляції контролер із фіксованою частотою звертається до API-сервера, отримує поточні параметри та оновлює стан моделі. Зокрема, змінюється швидкість обертання коліс або координати руху. Усі оновлення відображаються у візуальному вікні Webots у реальному часі.

Загальна архітектура системи забезпечує чітку взаємодію між усіма рівнями. Потік даних має односторонній напрям: від таблиці до тривимірної моделі, що дозволяє мінімізувати затримки та забезпечити стабільну симуляцію поведінки робота. Такий підхід легко масштабувати, підключаючи реальні сенсори замість таблиці або замінюючи Webots на інші рушії з підтримкою HTTP-запитів.

Розробка клієнтського Python-скрипта для надсилання даних. Для передачі параметрів руху у Webots розроблено клієнтський Python-скрипт, який імітує надходження даних від зовнішнього джерела. Джерелом виступає попередньо підготовлений файл формату CSV, у якому записано послідовність координат або інших характеристик, що впливають на поведінку моделі.

На першому етапі реалізовано зчитування даних з таблиці. Для цього використано стандартну бібліотеку csv, що дозволила построково обробляти значення з кожного рядка. Після зчитування дані перетворюються у формат JSON, придатний для передачі через HTTP-запит. Структура JSON включає необхідні параметри, такі як положення об'єкта або кути повороту компонентів.

Передача даних відбувається за допомогою бібліотеки requests. Для кожного рядка таблиці формується окремий POST-запит до API-сервера. Частота надсилання регулюється вручну через параметр затримки між запитами, що задається на початку скрипта. Це дозволяє змінювати швидкість відтворення поведінки в 3D-середовищі без зміни структури скрипта.

Для контролю роботи передбачено базове логування. У консоль виводиться інформація про успішність кожного запиту, включно з кодом відповіді сервера. У випадку помилки відображається повідомлення про її тип, що спрощує налагодження під час тестування системи.

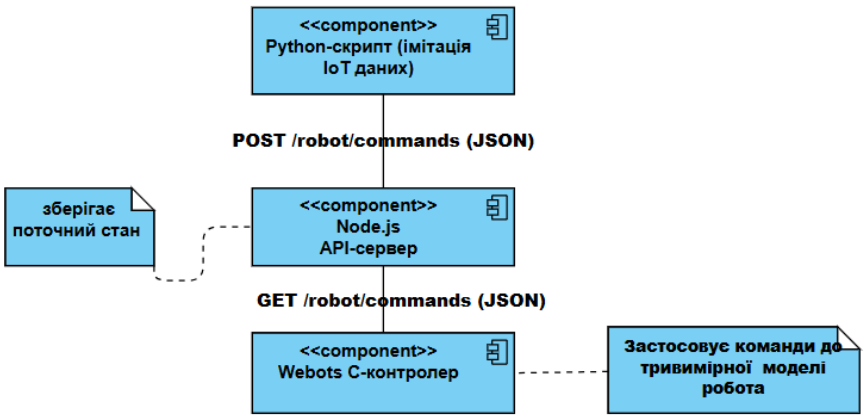

Рис. 4. Архітектура системи синхронізації

Застосування скрипта дало змогу реалізувати послідовну, контрольовану передачу даних з CSV-файлу до Webots через проміжний сервер. Отримана зв'язка стала основою для динамічного оновлення стану цифрового двійника у відповідь на симульовані параметри руху.

Розробка серверної частини системи. Для реалізації обміну даними між Python-скриптом і контролером у Webots створено простий API-сервер за допомогою середовища Node.js. Сервер виконує роль проміжної ланки, яка приймає параметри руху у вигляді HTTP-запитів, зберігає їх у пам'яті та надає доступ до них на запит 3D-рушія.

У першому етапі налаштовано базовий сервер за допомогою бібліотеки Express. Створено маршрут POST /robot/commands, який приймає дані в форматі JSON. Для зберігання отриманої інформації використано локальну змінну у вигляді об'єкта, який зберігає останній стан параметрів.

Реалізовано маршрут GET /robot/commands, що відповідає за передачу цих параметрів контролеру у Webots. Контролер періодично звертається до цього маршруту, щоб отримати актуальні команди для оновлення стану 3D-моделі. Обробка запитів від обох компонентів – Python-скрипта і контролера Webots – виконується асинхронно, без збереження в базі даних.

Для перевірки працездатності серверу використано середовище Postman. Надіслано тестові запити до маршруту POST, після чого успішно отримано збережені дані через GET. Перевірено правильність форматування, стабільність відповіді та відповідність очікуваному JSON-формату.

Реалізоване рішення дозволило забезпечити просту, але стабільну взаємодію між джерелом даних і рушієм візуалізації. Сервер працює локально, без додаткових бібліотек, що спростило процес налаштування та налагодження під час розробки.

Реалізація контролера у Webots. У середовищі Webots реалізовано контролер, який відповідає за періодичне оновлення стану 3D-моделі на основі даних, отриманих із API-сервера. Для написання контролера використано мову програмування C, що є нативно підтримуваною Webots для створення логіки поведінки об'єктів у симуляційному середовищі.

Основна задача контролера – регулярно надсилати GET-запити на сервер для отримання актуального JSON-об'єкта з параметрами руху. Запити виконуються з певною затримкою, що задається таймером Webots, що дозволяє синхронізувати частоту оновлення з частотою надсилання даних із клієнтського скрипта.

Після отримання відповіді від API-сервера JSON-об'єкт розбирається на окремі параметри. Використано вбудовані методи для обробки HTTP-відповідей та бібліотеку json-glib для парсингу структури JSON, хоча в базовому варіанті реалізовано вручну розбір рядка без зовнішніх залежностей.

Отримані координати або інші параметри прив'язуються до властивостей 3D-моделі: положення корпусу, кути обертання коліс, напрямок руху. Для цього використовуються функції Webots API, які дозволяють змінювати положення об'єктів у просторі симуляції в реальному часі.

Для перевірки роботи контролера проведено тестування, у ході якого відстежувалась зміна стану моделі під час надходження нових команд. Візуально зафіксовано правильне оновлення параметрів – положення моделі відповідало очікуваним значенням, а зміна координат відбувалась згідно з даними, що надсилались через API. Реалізований контролер забезпечив базову, але стабільну реакцію моделі на зовнішній потік інформації.

Тестування та експериментальне дослідження розробленої системи. У процесі розробки системи синхронізації цифрового двійника з потоковими даними виявлено наступні технічні складності, які вимагали корекції як на етапі підготовки тривимірної моделі, так і під час налаштування обміну даними між компонентами системи.

Однією з перших проблем стала надмірна складність вихідної тривимірної моделі, завантаженої з відкритої бібліотеки. У вихідному вигляді вона містила велику кількість дрібних деталей і понаднормову кількість полігонів, що створювало навантаження на симулятор та призводило до зниження продуктивності. Для вирішення цієї проблеми проведено спрощення геометрії у Fusion 360: складні компоненти замінено на простіші меші, зменшено загальну кількість трикутників у 20 разів, що дозволило досягти стабільного відтворення фізики у Webots

На етапі тестування моделі виявлено також невідповідність фізичних параметрів деяких елементів. Зокрема, некоректно працювала взаємодія коліс із поверхнею, що проявлялось у відсутності стабільного руху. Для усунення проблеми повторно налаштовано фізичні властивості – масу, тертя та інерцію – безпосередньо у Webots.

Під час налагодження обміну даними виникали тимчасові збої у з'єднанні між Webots і API-

сервером, пов'язані з мережею або повільною обробкою запитів. Для підвищення надійності комунікації додано механізм повторних GET-запитів із затримкою в разі невдачі, що дозволило уникнути збоїв у відображенні даних.

Окремо виникали помилки, пов'язані з форматом JSON, особливо у випадках некоректного формування об'єкта на стороні Python-скрипта. Для виявлення та усунення таких помилок реалізовано перевірку вхідних даних на боці API-сервера, з поверненням відповідного коду помилки і логуванням проблемного запиту. Це забезпечило більшу стабільність та передбачуваність у роботі системи.

Демонстрація роботи системи. При запуску симуляції робота у Webots та серверу, можна побачити, що сервер постійно оновлюється (рис. 5) та контролер робота намагається брати значення, які дорівнюють 0, щоб робот нікуди не рухався до запуску потоку IoT– даних.

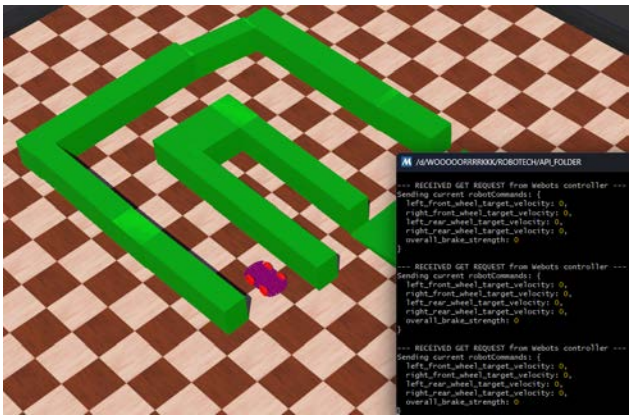

Рис. 5. Значення серверу у консолі

При запуску Python-скрипту для зчитування IoT-даних, через Visual Studio Code, робот починає свій рух та сервер оновлює свої дані. Для більш наглядної демонстрації було побудовано деякі споруди, підйом догори та спуск вниз у кінці шляху робота, і як можна побачити (рис. 6-8) робот спокійно оминає перешкоди та рухається за вказаним шляхом.

Проведено експериментальне дослідження, під час якого оцінювалась швидкість оновлення моделі, точність синхронізації та стійкість системи до збоїв мережі. Результати показали, що затримка між надходженням даних та їх відображенням у тривимірній моделі не перевищує 200–300 мс, система коректно обробляє втрачені пакети та забезпечує реалістичну візуалізацію поведінки цифрового двійника.

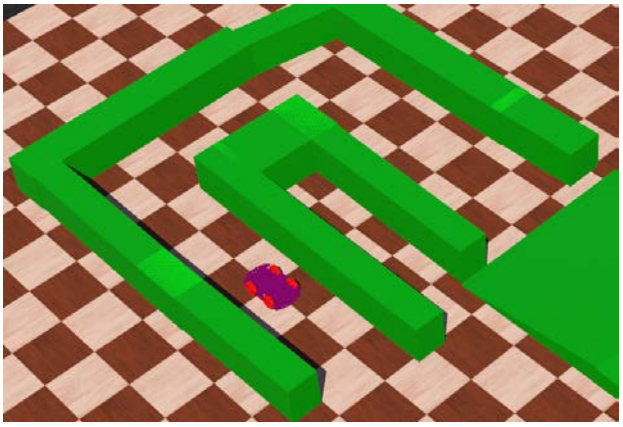

Рис. 6. Початок руху

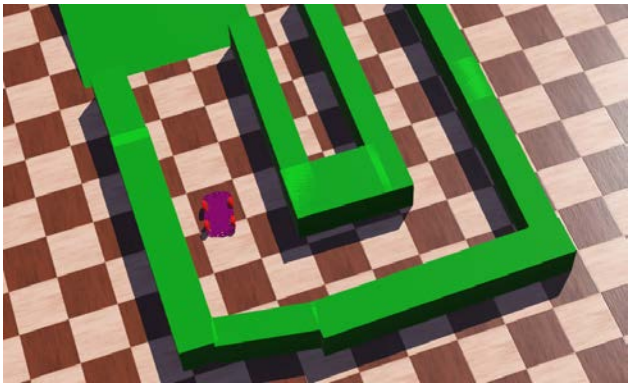

Рис. 7. Оминання перешкоди

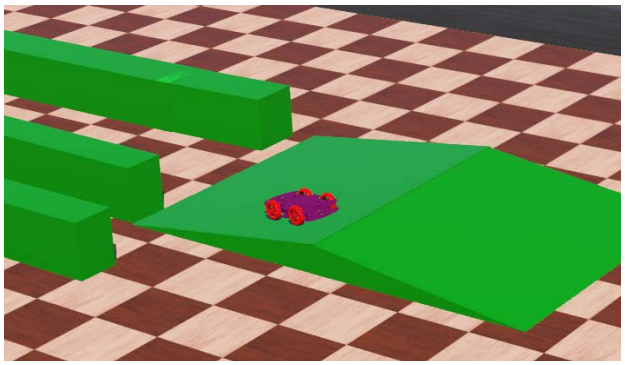

Рис. 8. Підйом догори

Для кількісної оцінки ефективності синхронізації додатково використано дві метрики: Twin Alignment Ratio (τ) та Age of Digital Twin (AoDT).

Twin Alignment Ratio (τ) відображає ступінь узгодженості станів цифрового двійника з фізичним об'єктом у часі. Значення τ, наближене до 1, свідчить про високу точність синхронізації, тоді як його зменшення вказує на зростання розбіжності між моделлю та реальним пристроєм.

AoDT визначає актуальність даних, що використовуються для оновлення моделі, шляхом вимірювання часу, що минув з моменту останнього отримання достовірної інформації. Зростання

AoDT сигналізує про старіння даних і погіршення якості синхронізації.

У ході експериментів було встановлено, що запропонована система забезпечує значення τ у межах 0,92–0,96, що відповідає високому рівню узгодженості станів, а середнє значення AoDT не перевищувало 250 мс, що підтверджує своєчасність оновлення моделі. Таким чином, використані метрики дозволили не лише якісно, а й кількісно підтвердити ефективність розроблених алгоритмів синхронізації.

Таким чином, експериментальні дослідження підтвердили, що розроблена система синхронізації забезпечує роботу цифрового двійника робототехнічного пристрою в умовах наближених до реального часу. Отримані результати можуть бути використані як основа для подальшого вдосконалення систем, зокрема шляхом впровадження протоколів WebSocket та алгоритмів прогнозування на основі машинного навчання.

Висновки. У роботі представлено дослідження та реалізації системи синхронізації цифрового двійника мобільного роботизованого пристрою з потоковими даними в умовах симульованого середовища. Розроблено архітектуру, яка поєднує зовнішній скрипт для генерації даних, серверну частину для маршрутизації запитів і передачі інформації, а також тривимірну модель у Webots, яка реагує на зовнішні зміни параметрів.

В основу системи покладено підхід, що імітує отримання реальних телеметричних даних. З використанням Python-скрипта реалізовано зчитування параметрів з таблиці, формування JSON-структур і відправлення їх на API. Серверна частина на Node.js відповідає за прийом і збереження актуального стану пристрою та обслуговує запити від контролера моделі. Контролер, написаний мовою C, регулярно отримує оновлення, розбирає дані та змінює властивості моделі відповідно до них у режимі, наближеному до реального часу.

В роботі реалізоване віртуальне середовище у Webots, адаптовано імпортовану модель чотириколісного робота з урахуванням вимог до фізичної симуляції, оптимізовано геометрію та спрощено складні компоненти. Виявлені проблеми з продуктивністю та передачею даних були усунені через додавання валідації запитів, повторних підключень та оновлення властивостей моделі на основі буфера даних.

Експериментальне тестування підтвердило коректну синхронізацію параметрів і поведінки цифрового двійника. Спостерігається коректне відображення положення, швидкості та змін стану моделі відповідно до зовнішнього потоку, що свідчить про готовність системи до подальшої інтеграції з реальними сенсорами та контролерами.

Запропоноване рішення може бути використане як основа для побудови складніших цифрових двійників у промислових і дослідницьких задачах. Перспективним є розширення функціоналу системи через підключення до хмарних сервісів та інтеграцію з базами даних.

Список літератури:

1. Tan B., Matta A. The Digital Twin Synchronization Problem: Framework, Formulations, and Analysis. *IISE Transactions*. 2023. DOI: 10.1080/24725854.2023.2253881. URL: https://faculty.ozyegin.edu.tr/baristan/files/2024/05/IISE_DTSynchro2023.pdf (дата звернення: 01.09.2025)

2. Cakir L., et al. How to Synchronize Digital Twins? A Communication Performance Analysis. *arXiv preprint*. 2024. URL: https://arxiv.org/pdf/2402.05587 (дата звернення: 01.09.2025)

3.Alghamdi W., et al. Synchronization Patterns for Digital Twin Systems. *Journal of Applied Data Sciences*. 2024. T. 5, № 3. C. 1026–1037. DOI: 10.47738/jads.v5i3.267

4. Touhid M. T. B., Zhu E., Ehteshamfara M. V., et al. Evaluation of digital twin synchronization in robotic assembly using YOLOv8. *International Journal of Advanced Manufacturing Technology*. 2024. T. 134. C. 871–885. DOI: 10.1007/s00170-024-14182-7

5. Qadir M. A. Digital Twin-Assisted Multi-Layer Networks for Low-Latency Applications. *Computer Communications*. 2025. T. 241. DOI: 10.1016/j.comcom.2025.108219

6. Khalaf G. A UAV-Aided Digital Twin Framework for IoT Networks with High Accuracy and Synchronization. *arXiv preprint*. 2025. URL: https://arxiv.org/abs/2504.15967 (дата звернення: 01.09.2025)

Manko D.V., Myronova N.O., Shaptala S.V., Kulyaba-Kharytonova T.I. SYNCHRONISATION SYSTEM FOR A THREE-DIMENSIONAL MODEL OF A DIGITAL TWIN OF A ROBOTIC DEVICE WITH STREAMING IOT DATA

The article presents research and implementation of a system for synchronising a three-dimensional model of a digital twin of a robotic device with streaming IoT data in the Webots environment. The relevance of the work is due to the growing need to build digital twins for robotics that are capable of operating in near real

time for modelling, testing and control optimisation tasks. The object of the study is the process of updating the parameters of a three-dimensional model of a robotic device based on external streaming IoT data. The subject of the study is algorithms and software tools for implementing a digital twin synchronisation system. The paper justifies the choice of the Webots environment as the optimal platform for modelling and visualising robotic systems, providing realistic physics, API support and the ability to integrate with external services. A system architecture consisting of three main modules is proposed: collection and pre-processing of IoT data, server processing and data provision via API, as well as visualisation and model control in Webots. The following algorithms have been developed: reading and transferring data from a CSV table in JSON format, processing data and responding to client requests, and periodically updating the model status based on the parameters received. The algorithms ensure real-time information exchange with minimal delay and high stability. An experimental study was conducted to evaluate the model update speed, synchronization accuracy, and system stability against network failures. The results showed that the delay between data arrival and its display in the three-dimensional model does not exceed 200–300 ms, the system correctly processes lost packets and provides realistic visualisation of the digital twin's behaviour. The analysis of the impact of parameters demonstrated the critical role of optimising request intervals, the format of transmitted data, and error handling on the API side. The results confirm the effectiveness of the proposed synchronisation algorithms and the feasibility of using the developed architecture for building digital twins in robotics. The developed approach can be extended by integrating with real IoT devices, applying WebSocket protocols to reduce delays, and implementing machine learning-based prediction algorithms. Further research will focus on scaling the system, adapting it to industrial conditions, and creating hybrid digital twins with remote control support.

Key words: *digital twin, synchronisation algorithms, IoT, 3D model, Webots, REST API, Node.js, Python, robotics, streaming data.*

Дата надходження статті: 01.09.2025
Дата прийняття статті: 16.09.2025
Опубліковано: 16.12.2025