Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Удосконалений метод планування процесами реального часу
в кіберфізичних системах
Назва теми

Рівень вищої освіти *другий (магістерський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 240233.24.02.22 ПЗ

Виконав здобувач IV курсу, група КІ2м-22-1 _____ Владислав КОВАЛЬ
Підпис Ініціали, прізвище

Керівник канд. екон. наук, доцент _____ Світлана САЧЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер канд.ф-м. наук, доцент _____ Тетяна КИСІЛЬ
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«_1_» _травня_ 2026 р. Підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ДРУГИЙ (МАГІСТЕРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА
" 12 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ковалю Владислав Сергійович
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Удосконалений метод планування процесами реального часу в кіберфізичних системах
Керівник проекту (роботи) Світлана САЧЕНКО, канд.екон.наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 12.01.2026 р. № 6

2. Термін подання здобувачем роботи на кафедру 01.05.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Архітектура та принципи синтезу кіберфізичних систем і постановка задачі щодо її удосконалення

Моделі нульового копіювання в КФС

Метод планування процесів реального часу у кіберфізичних системах

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, КПІСдоцент кафедри КПІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КПІС		

7. Дата видачі завдання «_12_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проєкту (роботи)	Термін виконання етапів проєкту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	12.01.2026	виконано
2	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	25.02.2026	виконано
3	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	15.03.2026	виконано
4	Робота над науковою статтею	01.04.2026	виконано
5	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	10.04.2026	виконано
6	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	25.04.2026	виконано
7	Оформлення пояснювальної записки згідно вимог	25.04.2026	виконано
8	Попередній захист ВКР	30.04.2025	виконано
9	Захист ВКР на засіданні ЕК	травень 2026 року	

Здобувач _____ Владислав КОВАЛЬ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Світлана САЧЕНКО

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: «Удосконалений метод планування процесами реального часу в кіберфізичних системах»

Автор роботи: Коваль Владислав Сергійович

Керівник роботи: Саченко Світлана Іванівна

Пояснювальна записка: 79 с., 9 рис., 10 табл., 4 дод., 81 джерело.

ДЕЦЕНТРАЛІЗАЦІЯ, КІБЕРФІЗИЧНА СИСТЕМА, ПРИСТРОЇ ІОТ, ЦІЛЬОВА ФУНКЦІЯ, ХМАРНЕ СЕРЕДОВИЩЕ.

Об’єктом дослідження є планування процесів реального часу в кіберфізичних системах.

Предметом дослідження є методи та засоби планування процесів реального часу в кіберфізичних системах.

Метою кваліфікаційної роботи магістра є покращення планування процесами реального часу в кіберфізичних системах за рахунок розроблення нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–прискорених пристроїв, забезпеченні планування реального часу та спільного проєктування керування і планування.

Для розв’язання поставлених задач використовувалися методи планування в операційних системах реального часу, методи синтезу кіберфізичних систем, теорія множин.

Наукова новизна отриманих результатів:

– удосконалено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування, що дало змогу покращити використання пам’яті КФС.

На основі проведених досліджень розроблена архітектура і алгоритми реалізації методу планування процесами реального часу в кіберфізичних системах.

Практична значимість отриманих результатів полягає у розроблені механізму нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС.

У вступі подано об’єкт та предмет дослідження, мету, наукову новизну та практичну цінність роботи, а також характеристику структури роботи.

У першому розділі проведено аналіз відомих рішень щодо кіберфізичних систем і планування реального часу.

У другому розділі здійснено дослідження предметної області, запропоновано архітектуру та компоненти кіберфізичних систем, включно з актуаторами, а також досліджено коректність запропонованих рішень щодо дослідження типового класу КФС.

У третьому розділі розроблено удосконалено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування, що дало змогу покращити використання пам’яті КФС.

У четвертому розділі здійснено розроблення алгоритмів реалізації методу планування процесами реального часу в кіберфізичних системах, засоби реалізації та подано результати експериментальних досліджень з розробленою КФС, включно з обгрунтуванням постановки експмериментів.

У висновках підведено підсумки досягнення результатів з розв’язання завдань дослідження.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ІТ – інформаційні технології

КФС – кіберфізична система

ОС – операційна система

ПЗ – програмне забезпечення

ROS2 – Robot Operating System 2 (), проміжне програмне забезпечення для робототехніки

ВСТУП

Проєктування на основі компонентів є парадигмою розробки програмних застосунків, що передбачає модульну реалізацію функцій системи. Такий підхід широко застосовується в кіберфізичних і роботизованих системах, де типові рішення (вузли сприйняття, оцінювачі стану, планувальники руху) можуть бути розроблені, перевірені та повторно використані як готові модулі. Ці компоненти інтегруються на гетерогенних апаратних платформах із розподілом обчислень між CPU, GPU та FPGA.

Однак інтеграція компонентів на різних обчислювальних пристроях може спричиняти часову невизначеність через передавання даних між доменами пам’яті та додаткове копіювання, що порушує обмеження реального часу. Додаткові складнощі створює Robot Operating System 2 (ROS2) – проміжне програмне забезпечення для робототехніки, яке, попри свою гнучкість і підтримку складних та багатороботних систем, лише нещодавно почало отримувати гарантії таймінгу. Основною проблемою її застосування залишається передбачуваність виконання.

Ключовими завданнями є управління пам’яттю апаратно–прискорених пристроїв, планування реального часу та спільне проєктування керування і планування. У сукупності ці аспекти мають забезпечити модульність, високу швидкодію та відповідність кіберфізичним часовим обмеженням. Отже, проєктування передбачуваних компонентних систем реального часу є критично важливим і повинно охоплювати три рівні системного стеку: управління пам’яттю, планування та контроль.

Актуальність роботи полягає в покращенні управління пам’яттю апаратно–прискорених пристроїв, планування реального часу та спільне проєктування керування і планування.

Метою кваліфікаційної роботи магістра є покращення планування процесами реального часу в кіберфізичних системах за рахунок розроблення нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–

прискорених пристроїв, забезпеченні планування реального часу та спільного проєктування керування і планування.

Поставлена мета досягається розв’язанням таких основних завдань:

– проаналізувати відомі методи планування процесами реального часу в кіберфізичних системах та спільного проєктування керування і планування;

– розробити метод планування процесами реального часу в кіберфізичних системах на основі механізму нульового копіювання;

– розробити актуатор та алгоритми для забезпечення реалізації механізму нульового копіювання;

– розробити КФС з механізмом нульового копіювання даних в різних компонентах для управління пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС та провести з нею експериментальні дослідження.

Об’єктом дослідження є планування процесів реального часу в кіберфізичних системах.

Предметом дослідження є методи та засоби планування процесів реального часу в кіберфізичних системах.

Наукова новизна отриманих результатів:

– удосконалено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування, що дає змогу покращити використання пам’яті КФС.

На основі проведених досліджень розроблено метод планування процесами реального часу в кіберфізичних системах.

Практична значимість отриманих результатів полягає у розроблені механізму нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС.

Для розв’язання поставлених задач використовувалися методи планування в

операційних системах реального часу, методи синтезу кіберфізичних систем, теорія множин.

За темою кваліфікаційної роботи опубліковано одну публікацію [81] у фаховому науковому журналі «*Herald of Khmelnytskyi National University. Technical Sciences»* (м. Хмельницький, 2026, № 361(1), С. 568–574. DOI: https://doi.org/10.31891/2307–5732–2026–361–78).

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ ТА ЗАСОБІВ ПЛАНУВАННЯ ПРОЦЕСАМИ РЕАЛЬНОГО ЧАСУ В КІБЕРФІЗИЧНИХ СИСТЕМАХ

1.1 Огляд та поняття планування процесів операційних систем реального часу в кіберфізичних системах

Проєктування на основі компонентів є однією з ключових парадигм сучасної розробки програмних застосунків, яка спрямована на зменшення складності систем шляхом декомпозиції функціональності на окремі, логічно завершені модулі. Такий підхід дозволяє ізолювати відповідальність компонентів, спростити тестування, супровід і масштабування програмного забезпечення. Концепція «будівельних блоків» особливо широко використовується в кіберфізичних та роботизованих системах, де складні поведінкові сценарії формуються шляхом композиції типових функціональних елементів. Поширені та усталені рішення для конкретних задач, зокрема вузли сприйняття навколишнього середовища, оцінювачі стану системи, алгоритми локалізації та планувальники руху, можуть бути незалежно розроблені, формально перевірені, оптимізовані та повторно використані як стандартизовані програмні модулі в різних застосуваннях.

Такі модулі зазвичай інтегруються поверх сучасних гетерогенних апаратних платформ, де виконання застосунку розподіляється між різними обчислювальними ресурсами, зокрема ядрами центрального процесора (CPU), графічними процесорами (GPU) та програмованими логічними інтегральними схемами (FPGA). Гетерогенність обчислювального середовища дозволяє досягти високої продуктивності та енергоефективності, однак водночас суттєво ускладнює інтеграцію компонентів. Розподіл навантаження між різними пристроями може призводити до часової невизначеності, оскільки дані часто передаються між різними доменами пам’яті, потребують копіювання або синхронізації, що негативно впливає на дотримання жорстких часових обмежень систем реального часу.

Додатковим фактором складності є використання Robot Operating System 2 (ROS2), яка фактично виступає як проміжне програмне забезпечення для

робототехнічних систем. ROS2 забезпечує механізми обміну повідомленнями між компонентами, надає бібліотеки, драйвери пристроїв, інструменти візуалізації та засоби налагодження, значно спрощуючи розробку складних робототехнічних застосунків. Водночас архітектура ROS2 історично орієнтувалася на гнучкість і масштабованість, а не на суворе дотримання принципів реального часу. Її планувальник відходить від класичних підходів до систем реального часу, і лише в останні роки розпочалися активні дослідження та розробки, спрямовані на впровадження формальних гарантій таймінгу та передбачуваності виконання.

Основною проблемою застосування ROS2 та подібних платформ у кіберфізичних системах є недостатня передбачуваність поведінки. Для її подолання необхідно вирішити низку важливих завдань, серед яких ключовими є організація ефективного управління пам’яттю для апаратно–прискорених пристроїв, розробка та аналіз планувальників реального часу, а також дослідження підходів до спільного проєктування керування та планування з урахуванням часових обмежень. У сукупності ці завдання мають забезпечити підтримку модульності, можливість композиційної верифікації компонентів і високу швидкодію системи, водночас гарантуючи відповідність жорстким кіберфізичним вимогам щодо часу виконання та інших критичних властивостей від початку до завершення задач.

Поєднання обізнаності про ієрархію та поведінку пам’яті з формально аналізованим плануванням і часово–орієнтованим керуванням дозволяє досягти покращених характеристик передбачуваності та продуктивності кіберфізичних систем. Таким чином, актуальним завданням є забезпечення проєктування передбачуваних компонентних систем реального часу для кіберфізичних застосувань. Критерій передбачуваності є визначальним для таких систем і вимагає комплексного охоплення трьох критичних рівнів системного стеку: управління пам’яттю, планування виконання та контроль, що в сукупності формують основу надійної та безпечної роботи кіберфізичних систем.

ROS2 – це широко прийнятий програмний фреймворк для розробки робототехнічних застосунків, який застосовується для систем реального часу. Хоча ROS2 забезпечує механізми модульної розробки, але її стандартні можливості

планування створюють виклики для передбачуваної продуктивності в реальному часі, особливо для складних застосувань, що включають розгалуження або об'єднання потоків даних, тобто довільні графи. ROS2 надає компонентний фреймворк для створення КФС з акцентом на робототехніку. Ключові поняття включають вузли і теми. Вузли інкапсулюють пов'язані функції, наприклад, драйвер сенсора, планувальник шляху. Теми створюють іменовані канали комунікації, що полегшують обмін даними через парадигму публікації–підписки. Вузли, які надають дані, публікують інформацію на тему, а вузли, які цікавляться цими даними, підписуються на них. Проміжним програмним забезпеченням, відповідальним за полегшення передачі даних, зазвичай є сервіс розподілу даних [1, 2], хоча можуть використовуватися й інші протоколи зв'язку [3, 4]. Зворотні виклики – це функції всередині вузлів, які запускаються конкретними подіями, головним чином: Зворотні виклики слугують атомарними виконуваними одиницями обчислень у ROS2. Через публікацію та підписку вони також можуть спровокувати виконання одне одного. Послідовність зворотних викликів, що запускаються даними, які проходять через теми, формує логіку обробки застосунку, часто створюючи неявні залежності або ланцюги виконання. Такі ланцюги спочатку були визначені у [5], а подальші дослідження [6, 7] продовжували використовувати цю абстракцію.

ROS2 спроектована у ієрархічному стеку. Верхній шар використовується безпосередньо кінцевим розробником і забезпечує специфічне мовне зв'язування для використання ROS2. Саме тут приймаються рішення щодо розкладу. Базова функціональність ROS 2 знаходиться на рівні rcl, який надає такі функції, як вузли, простори імен та логування. Найнижчий шар стеку ROS2, який називається ROS Middleware (RMW) слугує обгорткою для комунікаційної платформи [8], яка безпосередньо полегшує комунікацію між вузлами.

У статті [9] було використано застосунок як платформу для розробки дрона оператора, який інтегрував автоматизовані рішення у планування траєкторії. Він був розроблений для слідування за рухомою ціллю, наприклад, за реальним об’єктом. Сенсорний конвеєр складається з 3 етапів. Виявлення та відстеження

складалися з моделі YOLOv4, яка виявляла об’кт в кадрі і те, наскільки він зміщувався у кадрі. Оцінка заголовків отримувала чітко обрізане та центроване зображення об’єкту, за допомогою якого модель зору визначала, у якому напрямку орієнтований об’єкт. Після того, як статус і напрямок об’єкту були відомі, це передавалося на фінальний етап виявлення. Оцінка руху використовувала оцінювач стану для оцінки положення та швидкості об’єкту. Припускаючи, що положення дрона відоме з високою впевненістю, відносні вимірювання положення та орієнтації об’єкта з попереднього рівня проектуються на площину землі, щоб обчислити позицію та орієнтацію об’єкта у системі координат. Це подається у фільтр Калмана для оцінки позиції та швидкості об’єкта, а також майбутньої траєкторії. Цю оцінку можна використовувати для продовження прогнозування траєкторії об’єкта навіть тоді, коли вимірювання недоступні, наприклад, коли об’єкт закритий. Оптимізація траєкторії використовувала планувальник шляху для планування траєкторії дрона, який безпечно уникає перешкод. Стандартний шлях – це траєкторія, паралельна об’єкту, з фіксованою відстанню та положенням дрона. Обидва параметри можна змінювати під час виконання, якщо пізніше потрібен інший кут. Етап оптимізації бере цей стандартний шлях і коригує його, щоб уникнути перешкод. Це досягається шляхом максимізації функції вартості, яка серед інших компонентів, включає лінійний інтеграл траєкторії із конвеєра відображення. Оскільки він позитивний у зонах, де уникають перешкод, то оптимізація цієї функції витрат природно відштовхне траєкторію у вільний простір. У роботі [10] доповнено цю функцію витрат, щоб також уникнути закриття об’єкту, плавності траєкторії та послідовності перспективи руху до об’єкту. Відстеження траєкторії – це етап спрацьовування, коли позиція дрона порівнюється з запланованою траєкторією, а закрите керування використовується для коригування положення дрона відповідно до траєкторії. Цей застосунок є привабливим кандидатом для тестування майбутніх досліджень, оскільки містить різноманітний набір апаратно–прискорюваних компонентів і ставить завдання у системному проєктуванні. При цьому, дані карти дуже великі і при переході з одного компонента в інший, вони спричиняли великі витрати на пам'ять.

В роботі [11] розглянуто планувальник ROS2, який складається з двох ланцюгів із об'єднуючим потоком даних, тому виникла потреба надійно планувати складні графи на ROS2. Управління пам’яттю з нульовим копіюванням (Zero–copy memory management) для гетерогенних обчислень є сучасним підходом [12, 13] до організації обміну даними між різними обчислювальними пристроями, зокрема центральним процесором (CPU), графічним процесором (GPU), прискорювачами штучного інтелекту або іншими спеціалізованими обчислювальними блоками, за якого фізичні дані не дублюються в різних областях пам’яті, а використовуються спільно. Ключова ідея [14, 15] полягає в тому, що різні пристрої отримують доступ до одних і тих самих фізичних ділянок пам’яті через спільний або уніфікований віртуальний адресний простір, що дозволяє оперувати єдиними покажчиками без необхідності явного копіювання даних між RAM і VRAM. У традиційних моделях гетерогенних обчислень передача даних відбувається через шину PCI Express [16], що супроводжується значними затримками, додатковими витратами енергії та потребою в подвоєнні або навіть потроєнні обсягів пам’яті для зберігання одних і тих самих масивів, тоді як zero–copy підхід дозволяє обчислювальному пристрою безпосередньо звертатися до даних у пам’яті хоста або в спільній фізичній пам’яті системи. Такий механізм особливо ефективний для задач реального часу [17, 18], потокової обробки даних, високопродуктивних обчислень, машинного навчання та обробки великих масивів інформації, де додаткові витрати на копіювання можуть стати критичним вузьким місцем. Реалізація zero–copy зазвичай вимагає апаратної та програмної підтримки [19], зокрема використання технологій на кшталт NVIDIA Unified Memory у CUDA, Shared Virtual Memory в OpenCL або архітектур з інтегрованою графікою, де CPU та GPU фізично використовують одну оперативну пам’ять, що значно спрощує модель програмування і знижує бар’єр входу для розробників [20]. Відсутність необхідності вручну керувати передачею даних між хостом і пристроєм зменшує складність коду, робить його більш читабельним і менш схильним до помилок, пов’язаних із неправильним порядком копіювання чи звільненням ресурсів. Разом із тим, ключовим викликом zero–copy підходу залишається забезпечення коректної синхронізації доступу до спільних даних [21,

22] і підтримання когерентності кешів [23, 24], оскільки одночасний доступ CPU та GPU до одних і тих самих змінюваних структур може призвести до гонок даних, неконсистентних результатів або деградації продуктивності. Для розв’язання цих проблем застосовуються механізми бар’єрів, атомарних операцій, керування життєвим циклом даних та спеціальні протоколи когерентності, що вимагає від розробника глибокого розуміння архітектури системи [25]. Попри ці складнощі, zero–copy memory management вважається одним із ключових напрямів розвитку гетерогенних обчислень, оскільки він дозволяє ефективніше використовувати апаратні ресурси, зменшувати затримки та будувати більш масштабовані й енергоефективні обчислювальні системи [26, 27].

Вбудовані системи в реальному часі підтримують критично важливі для місії безпеку застосування [27] та гібридне моделювання в реальному часі [28]. Останніми роками поєднання модульності на основі компонентів та налаштовуваного керування потоками та пам'яттю в ROS2 зробило його особливо привабливим для розробки вбудованих робототехнічних застосувань у реальному часі [29], включно з відкритою робототехнічною платформою для досліджень [30]. Такі системи часто мають обмеження за часом. Розробники повинні ретельно керувати апаратними та програмними витратами, варіаціями часу та розкладом, щоб завдання в реальному часі встигали до дедлайнів. Навіть на традиційних багатоядерних платформах операції з пам'яттю можуть створювати значні додаткові витрати, наприклад, коли кількість ядер перевищує один сокет чіпа, так що вартість операцій пам'яті між потоками на ядрах у різних сокетах чіпа обмежує кількість ядер, які можна експлуатувати на дрібних часових масштабах [31], або коли частий зв'язок між компонентами призводить до того, що загальна вартість операцій з пам'яттю наближається до інших менш частих, але дорожчих додаткових витрат. Наприклад, перемикання контексту потоку. Отже, зменшення частоти операцій з пам'яттю або підвищення їх ефективності [32] є важливим питанням для цих систем.

Таким чином, при проєктуванні операційних систем реального часу в кіберфізичних системах на основі компонентного підходу важливими завданнями

виступають завдання з планування реального часу, забезпечення зменшення складності процесу, покращенні управління пам’яттю апаратно–прискорених пристроїв та спільне проєктування керування і планування.

1.2 Операції з пам’яттю в операційних системах реального часу в кіберфізичних системах

Для гетерогенних обчислювальних платформ, які підтримують апаратне прискорення обчислень (наприклад, NVIDIA, Jetson і Xilinx Kria), управління пам'яттю з урахуванням затримок є дуже важливим. Наприклад, було показано в роботі [33], що GPU можуть бути причиною недетермінованої поведінки в системах реального часу. Операції з пам'яттю можуть спричиняти синхронізацію по всьому пристрою [34]. З апаратним прискоренням чи без нього, багатокомпонентний застосунок може зменшити пропускну здатність пам'яті, використовуючи спільні структури пам'яті та передаваючи посилання на дані замість копіювання самих даних [35], коли це можливо. Такі підходи «нульового копіювання» є інтуїтивними за принципом, але повинні забезпечувати володіння даними, щоб уникнути гоночних умов, які можуть створити застосункові інженерні виклики на практиці. Нульове копіювання у ROS2 передбачає, що якщо абоненти отримують незмінний вказівник на ті ж дані, і коли всі абоненти повернули свої вказівники, то пам'ять повідомлення звільняється або переробляється, залежно від реалізації [36, 37].

Популярною реалізацією семантики нульового копіювання для ROS 2 є проєкт поданий в роботі [38], який надає стороння службова програма, що керує спільним пулом пам'яті. Видавці та підписники, які бажають використовувати нульову копію, зобов'язані використовувати семантику та повернення для доступу до блоків пам'яті [39].

Показники продуктивності багатоядерних пристроїв показують мінімальні додаткові витрати [40, 41], але така схема не підтримує семантику нульового копіювання для пам'яті інших пристроїв. Розробник, який бажає використати

апаратне прискорення, повинен вручну копіювати повідомлення в пам'ять пристрою та з неї. Крім того, пули пам'яті попередньо виділяються стороннім демоном, який кінцевий розробник повинен налаштувати для оптимізації використання пам'яті. За стандартними налаштуваннями блоки пам'яті, ймовірно, будуть надмірними і можуть бути недостатньо великими, щоб вмістити накопичення повідомлень у найгіршому випадку. Вона підтримує нульове копіювання лише звичайних типів даних фіксованого розміру. Будь–які повідомлення, які потребують динамічного виділення пам'яті, можуть частково виділятися на купу, що потенційно може призвести до недійсних вказівників і помилок сегментації при обміні повідомленнями між процесами. Це обмеження, яке також має реалізація, і для виправлення, ймовірно, знадобиться оновлення API ROS2 у майбутньому.

Ще один механізм нульового копіювання доступний в програмному стеку ROS2 [42, 43], але підтримує лише внутрішньопроцесну семантику нульового копіювання.

Хоча додавання нульового копіювання до ROS 2 може допомогти зменшити недетермінованість, спричинену надмірними операціями з пам'яттю, але цього недостатньо [44]. Багато робототехнічних застосувань використовують апаратне прискорення з міркувань продуктивності, але функції ROS2 з нульовим копіюванням поки що не знають про межі між пристроєм і пам'яттю, тому компоненти, що використовують апаратне прискорення, повинні брати на себе відповідальність за копіювання даних всередині та поза пам'яттю пристрою. Це може нейтралізувати переваги нульового копіювання та додати застосункову недетермінованість [45, 46].

Для компонентів, що передають дані один одному, потрібно мінімізувати затримку, спричинену трьома типами операцій [47, 48]. Внутрішньокомпонентні копії пам'яті, обчислювальні операції та міжкомпонентні копії пам'яті є їх складовими частинами [49]. При апаратному прискоренні, незалежно від того, використовує GPU чи FPGA, складається з трьох етапів: копіювання в пам'ять пристрою; запуск ядра прискорення; копіювання з пам'яті пристрою [50, 51]. Ці три

операції відбуваються всередині компонента. Головним завданням в цьому процесі є мета щодо повного видалення внутрішньокомпонентних копій пам'яті, доручивши проміжному програмному забезпечення відповідальність за операції з пам'яттю пристрою [52, 53]. Міжкомпонентні копії пам'яті також можуть бути усунені, коли умови сприятливі для нульового копіювання, тобто коли два послідовні вузли працюють в межах однієї області пам'яті [54, 55].

Традиційне програмне забезпечення базувалося на сигнально–орієнтованому зв'язку між численними електронними блоками керування, з'єднаними через мережі. Хоча цей підхід історично гарантував надійність, і він має добре задокументовані обмеження у масштабованості, гнучкості та обслуговуванні, оскільки сучасні автомобілі інтегрують велику кількість даних та обмінюються зростаючими обсягами даних [56, 57]. Ці обмеження стимулювали поступовий перехід до систем, у яких функції інкапсулюються як сумісні сервіси, що спілкуються через стандартизовані протоколи [58]. ROS2 сприяє модульності, повторному використанню та обміну даними з високою пропускною здатністю між гетерогенними платформами [59,60].

Ця апаратна еволюція має глибокі наслідки для проєктування архітектури програмного забезпечення. Сучасні системи тепер повинні підтримувати розподілені обчислення між кількома процесорними блоками, гнучке розгортання сервісів і безшовну сумісність між компонентами, розробленими різними виробниками [61,62]. Відповідно, програмний рівень повинен приймати модульні, сервісно–орієнтовані принципи, які відокремлюють функціональність від конкретних апаратних вузлів, забезпечуючи масштабованість і легкість інтеграції. У цьому контексті проміжне програмне забезпечення, таке як ROS2 [1, 5], зі своєю моделлю комунікації та деталізованими політиками якості обслуговування, забезпечує ефективну основу для впровадження сервісно–орієнтованих парадигм у архітектурах автономних засобів [63,64]. Вона зарекомендував себе як фактична еталонна рамка для програмного забезпечення. Її мета полягає в забезпеченні спільної архітектури, яка забезпечить сумісність, багаторазове використання та безпеку між гетерогенними електронними блоками керування, розробленими

різними постачальниками. Ця конструкція забезпечує детермінований таймінг, високу надійність і відповідність стандартам функціональної безпеки [65], але обмежує гнучкість. Усі завдання, патерни комунікації та параметри планування фіксуються під час компіляції.

Зі зростанням обчислювального навантаження на функції, жорсткість платформи стала обмежуючим фактором [66, 67]. Щоб подолати ці обмеження, була впроваджена адаптивна платформа як її природна еволюція, орієнтована на високопродуктивні обчислювальні пристрої, такі як багатоядерні процесори та GPU [68, 69]. Адаптивний система переходить від сигнал–орієнтованої моделі до сервісно–орієнтованої, де застосунки динамічно експонують і споживають сервіси через стандартизовані комунікаційні рамки. Це дозволяє виявляти програмне забезпечення під час виконання, здійснювати динамічне розгортання та оновлення програмного забезпечення [70, 71].

Відкриті платформи повного стеку суттєво збільшили доступність академічних і промислових досліджень, надаючи багаторазові модулі, інструменти та інфраструктуру валідації [72]. З боку сприйняття вони поєднують класичні та методи, засновані на навчанні, такі як локалізація, кластеризація та модельне виявлення, а також глибокі архітектури, такі як сімейства YOLO [73]. Вони також інтегрують фільтрацію сенсорного синтезу для надійного відстеження та асоціації об'єктів між гетерогенними сенсорами.

Підсистема планування охоплює генерацію глобальних маршрутів і локальне планування руху, тоді як стек керування надає кілька алгоритмів слідування траєкторій [74]. Окрім самих програмних модулів, внесок ROS2, задокументовані інтерфейси та широке впровадження на різних засобах відіграли ключову роль у поширенні найкращих практик і прискоренні передачі технологій від досліджень до прототипування. Незважаючи на зрілість і комплексний дизайн, вона орієнтована на високопродуктивні обчислювальні платформи та виробничі апаратні ресурси та інфраструктуру розгортання [75, 76]. Її модульний стек складається з великої кількості взаємозалежних вузлів, конвеєрів сприйняття, прискорених GPU, та складних конфігураційних шарів, що робить її менш

придатною для експериментальних установок з обмеженими ресурсами у малому масштабі. Натомість архітектура на базі ROS2 орієнтована на дослідження фреймворку, що дозволяє швидко прототипувати, валідацію в реальному часі та розгортати на вбудованих платформах, таких як NVIDIA Jetson або мікроконтролери STM32. ROS2 став ключовим проміжним програмним забезпеченням для створення складних і розподілених автономних систем у робототехніці [5, 77].

Ландшафт автономних архітектур охоплює від промислових систем виробничого рівня до орієнтованих на дослідження експериментальних платформ. Виробничі системи є надзвичайно зрілими рамками, призначеними для впровадження на повномасштабних засобах із широкими наборами сенсорів, кластерами високопродуктивних обчислень і значними інженерними ресурсами. Хоча ці архітектури демонструють найсучаснішу продуктивність у реальних умовах, їхня складність, апаратні вимоги та часто власність обмежують доступність академічних досліджень і швидкого прототипування [78].

Натомість архітектура на базі ROS2, яка орієнтована на дослідницьке прототипування та валідацію на вбудованих платформах з обмеженими ресурсами, спеціально розроблена для роботи на доступних обчислювальних пристроях, що дозволяє швидко ітерувати без інфраструктурних витрат, типових для виробничих систем. Легка модульність архітектури дозволяє безшовно інтегрувати нові алгоритми сприйняття, стратегії планування або сенсорні модальності з мінімальним рефакторингом, підтримуючи поступові цикли розробки, необхідні для експериментальних досліджень. Крім того, платформа засобів дозволяє контрольовано експериментувати з використанням передових технологій у безпечному, повторюваному внутрішньому середовищі, дозволяючи дослідникам перевіряти нові алгоритми перед тим, як перейти на повномасштабне впровадження. Архітектура таким чином доповнює фреймворки виробничого рівня, забезпечуючи платформу для досліджень, оптимізовану для гнучкості, прозорості та ефективності ресурсів [79].

ROS2 використовує децентралізовану парадигму комунікації публікації / підписки, в якій незалежні програмні компоненти, які називаються вузлами, обмінюються повідомленнями по типізованих темах. Ця модульна архітектура спрощує розробку, тестування, інтеграцію та прискорює прототипування на розподілених апаратних платформах. Впровадження розподілених сервісів як базового проміжного програмного забезпечення дозволяє автоматично виявляти видавців і абонентів під час виконання, що дозволяє динамічно встановлювати комунікаційні лінії без попередньої конфігурації. На відміну від традиційних статичних архітектур, де обмін даними має бути чітко визначений заздалегідь, ця модель забезпечує самоналаштовуване підключення, яке адаптується при приєднанні або виході вузлів із мережі, значно підвищуючи гнучкість і стійкість розподілених систем [1].

Автономна система реального часу – це обчислювальна система, коректність якої визначається спільно функціональними результатами та дотриманням обмежених часових обмежень уздовж ланцюга сприйняття, прогнозування/планування та керування [80]. Ці обмеження визначаються як наскрізні властивості таймінгу та максимальний вік даних і мають бути передбачувано виконані в умовах роботи системи [34].

У сучасній практиці сприйняття/прогнозування зазвичай моделюється як м'який режим реального часу, тоді як механізм у замкненому циклі засобу залишається жорстким у реальному часі. У жорсткому режимі реального часу будь–який пропущений дедлайн вважається збоєм системи, оскільки обмеження часу є частиною вимог системи. На відміну від цього, м'який режим реального часу дозволяє час від часу допускати промахи з плавним погіршенням якості обслуговування.

Основною перевагою ROS2 є її можливість точно налаштовувати характеристики комунікації, такі як затримка, надійність, довговічність, глибина історії та живість. Детальне керування цими параметрами підтримує гетерогенні потоки даних, наприклад канали з низькою затримкою для телеметрії сенсорів і дуже надійні канали для критично важливих для безпеки команд дії. Коли ці

механізми поєднуються з ядром Linux у реальному часі, ROS2 може підтримувати контрольні цикли, які задовольняють суворі часові обмеження. Емпіричні результати повідомляють про обмежені затримки від кінця до кінця навіть при розподілених розгортаннях, що підкреслює доцільність ROS2 для критично важливого для безпеки керування засобами [57]. Крім того, інтеграція перетворює стандартний планувальник Linux на повністю преемптивний, зменшуючи найгірші затримки та кінцевий джитер у порядок і дозволяючи ROS2 досягати детермінованих показників продуктивності, типових для сучасних контролерів. Ті ж самі проміжні абстракції, що сприяють детермінізму, також відокремлюють логіку застосування від апаратної специфіки. Стандартизовані типи повідомлень і інтерфейси ROS2 ізолюють програмні компоненти від характеристик датчиків і виконавців, що залежить від виробника, і підвищує портативність між гетерогенними обчислювальними вузлами. ROS2 додатково надає модель управління життєвим циклом і контейнери компонентів, які дозволяють контрольовано запускати/вимикати, локалізувати несправності та комбінувати в процесі для мінімізації додаткових витрат та ключових факторів стабільної роботи в умовах реального часу з обмеженням ресурсів

Загалом, функції ROS2 позиціонують її як надійну, модульну та масштабовану основу для автономних засобів. Розробники можуть розподіляти конвеєри сприйняття та планування між кількома обчислювальними вузлами, адаптувати профілі якості до потреб затримки та надійності різних датчиків і виконавців, повторно використовувати компоненти у різних апаратних версіях та інтегрувати безпеку за дизайном. Відповідно, ROS2 пропонує великий вибір проміжного програмного забезпечення, який підтримує як швидке прототипування, так і орієнтовані на виробництво реалізацію архітектури автономних засобів [81].

1.3 Постановка задачі

Поставлена мета досягається розв’язанням таких основних завдань:

– проаналізувати відомі методи планування процесами реального часу в кіберфізичних системах та спільного проєктування керування і планування;

– розробити метод планування процесами реального часу в кіберфізичних системах на основі механізму нульового копіювання;

– розробити актуатор та алгоритми для забезпечення реалізації механізму нульового копіювання;

– розробити КФС з механізмом нульового копіювання даних в різних компонентах для управління пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС та провести з нею експериментальні дослідження.

1.4 Висновки до першого розділу

Проаналізовано відомі методи та засоби забезпечення планування процесами реального часу в кіберфізичних системах на основі механізму нульового копіювання. Було встановлено недоліки існуючих схем класичного та традиційного планування реального часу та визначено напрями подальших досліджень з удосконалення керування пам’яттю в КФС.

2 АРХІТЕКТУРА ТА МЕТОД СИНТЕЗУ КІБЕРФІЗИЧНИХ СИСТЕМ

2.1 Архітектура та компоненти кіберфізичних систем

Для забезпечення ефективного керування пам’яттю в системах реального часу з метою вивільнення пам’яті різних пристроїв КФС розробимо комунікаційне проміжне програмне забезпечення з нульовим копіюванням, яке підтримує обізнаність про домен пам'яті, виконання контекстно необхідних операцій пам'яті між різними доменами пам'яті та забезпечення нульового копіювання між двома компонентами в одному й тому ж домені пам'яті.

Спочатку розглянемо архітектуру всієї КФС для типів якої будуть розроблені рішення. Зобразимо на рис. 2.1 архітектуру типового класу КФС так, щоб відобразити місце основних компонентів систем та місце ROS2 і зв’язки між ними.

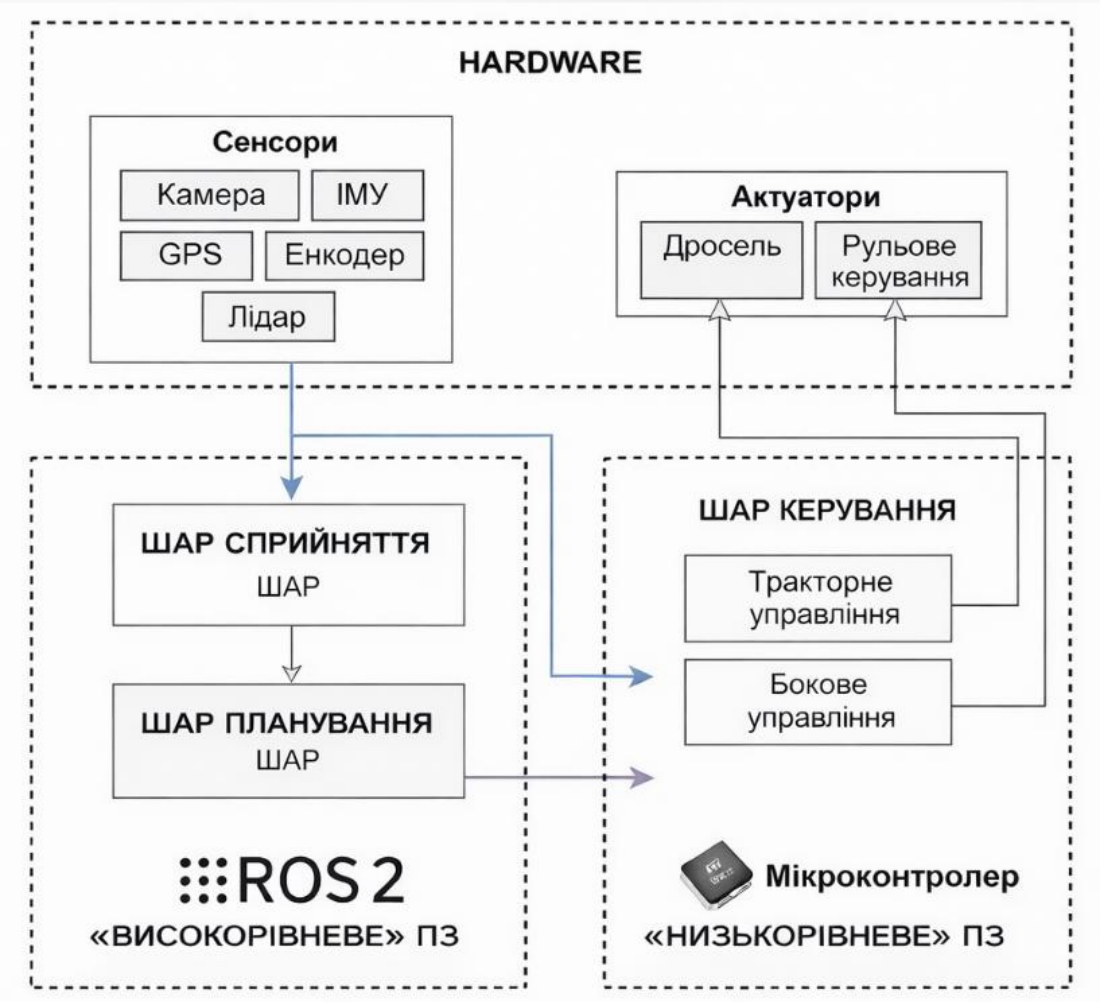


Рисунок 2.1 – Місце ROS2 в архітектурі КФС

Подана схема на рис. 2.1 відображає типову архітектуру класу КФС, у якій тісно поєднані фізичні компоненти, обчислювальні засоби та програмне забезпечення різних рівнів, а всі елементи взаємодіють через чітко визначені інформаційні та керуючі зв’язки. У верхній частині архітектури розташований фізичний рівень апаратного забезпечення, який включає сенсори та актуатори і є безпосередньо пов’язаним з реальним середовищем функціонування системи. Сенсори забезпечують первинний збір інформації про стан об’єкта керування та навколишнє середовище. Камера формує візуальні дані, що містять просторову та семантичну інформацію; інерціальний вимірювальний модуль (ІМУ) надає дані про прискорення та кутові швидкості; GPS забезпечує глобальну просторову прив’язку та оцінку положення; енкодери вимірюють параметри обертального або лінійного руху виконавчих механізмів; лідар формує просторову картину оточення у вигляді хмар точок або відстаней до об’єктів. Усі ці сенсорні дані надходять у цифровому вигляді до обчислювальної частини системи, формуючи багатовимірний потік інформації, що характеризує поточний стан КФС.

Зібрані сенсорні дані передаються до високорівневого програмного забезпечення, реалізованого на основі ROS 2, де вони спочатку обробляються на рівні сприйняття. На цьому рівні здійснюється фільтрація, синхронізація та злиття даних з різних джерел, а також побудова внутрішніх представлень середовища і стану системи, таких як оцінка положення та орієнтації, карта оточення, виявлення перешкод або об’єктів. Результати роботи шару сприйняття передаються далі до шару планування, між якими існує прямий логічний та інформаційний зв’язок. Шар планування використовує узагальнену та інтерпретовану інформацію для формування рішень щодо майбутньої поведінки системи, зокрема побудови траєкторій руху, вибору цільових станів або визначення послідовності дій, необхідних для досягнення поставленої мети з урахуванням обмежень і поточного контексту.

Сформовані на високому рівні команди та цільові параметри передаються з шару планування до низькорівневого програмного забезпечення, що виконується на мікроконтролері. Цей перехід від високорівневих абстрактних рішень до

конкретних керуючих сигналів є ключовим елементом кіберфізичної взаємодії. На мікроконтролері реалізований шар керування, який відповідає за безпосереднє управління фізичними процесами. У його складі виділяються підсистеми тракторного управління та бокового управління, що відповідають за регулювання поздовжнього руху, тобто швидкість, тяга, та поперечної динаміки для керування напрямком руху відповідно. Ці підсистеми отримують цільові значення або команди від високорівневого ПЗ та, використовуючи зворотний зв’язок від сенсорів, формують сигнали керування у реальному часі.

Згенеровані керуючі сигнали надходять до актуаторів, які замикають контур взаємодії між кібернетичною та фізичною частинами системи. Дросель реалізує керування тяговим зусиллям або потужністю, а рульове керування забезпечує зміну напряму руху. Дія актуаторів призводить до зміни фізичного стану системи та її положення в просторі, що, у свою чергу, знову фіксується сенсорами. Таким чином формується замкнений цикл «сприйняття – планування – керування – фізична дія – нове сприйняття», який є фундаментальною ознакою кіберфізичних систем.

Актуатори в типовій кіберфізичній системі відіграють ключову роль, оскільки саме вони є ланкою, через яку цифрові рішення та керуючі алгоритми безпосередньо впливають на фізичний об’єкт. На відміну від сенсорів, які лише спостерігають і вимірюють стан системи та середовища, актуатори здійснюють активну дію, змінюючи динаміку, положення або енергетичний стан системи. У поданій схемі актуатори представлені двома основними підсистемами, зокрема дроселем і рульовим керуванням, що є типовим набором для мобільних КФС.

Дросель як актуатор відповідає за реалізацію поздовжнього руху системи та безпосередньо впливає на швидкість і прискорення. З точки зору керування, він виконує перетворення цифрового керуючого сигналу, який сформовано мікроконтролером, у фізичну дію, що полягає у зміні подачі енергії до приводу, наприклад електродвигуна або двигуна внутрішнього згоряння. У кіберфізичному контексті дросель зазвичай працює в замкненому контурі керування. Низькорівневий регулятор порівнює бажане значення швидкості або тяги, отримане від шару планування, з поточним станом, який оцінюється за даними

енкодерів, ІМУ або інших сенсорів, і відповідно коригує сигнал на дросель. Таким чином забезпечується стабільність руху, плавність розгону та гальмування, а також дотримання заданих обмежень безпеки.

Рульове керування як актуатор відповідає за бокову динаміку та зміну напряму руху системи. Воно перетворює команди керування кутом повороту або траєкторією у фізичне відхилення керованих коліс, рульового механізму або іншого виконавчого елемента. У межах КФС цей актуатор також інтегрований у замкнений контур керування, де бокове управління враховує як цільові траєкторії, сформовані шаром планування, так і поточний стан системи, зокрема кутову швидкість, орієнтацію та поперечні відхилення, що оцінюються за допомогою ІМУ, лідару або візуальних сенсорів. Завдяки цьому забезпечується точне слідування заданому маршруту та стійкість руху навіть за наявності зовнішніх збурень.

Важливою особливістю актуаторів у кіберфізичних системах є те, що вони зазвичай не керуються безпосередньо з високорівневого програмного забезпечення. Замість цього високорівневі алгоритми передають узагальнені цільові параметри, тоді як мікроконтролер реалізує детерміноване, часово чутливе керування актуаторами з урахуванням апаратних обмежень і динаміки виконавчих механізмів. Такий підхід дозволяє відокремити складні алгоритми планування і прийняття рішень від задач реального часу, що є критичним для надійної роботи актуаторів.

Загалом актуатори формують завершальний етап кіберфізичного циклу, у якому результати обчислень та керуючих рішень матеріалізуються у фізичних діях. Саме через них система взаємодіє з реальним світом, а ефективність, точність і надійність актуаторів безпосередньо визначають загальну продуктивність КФС, її стійкість та здатність виконувати поставлені завдання в динамічному середовищі.

Уся архітектура демонструє чіткий поділ відповідальності між рівнями, але водночас тісну інтеграцію між ними через спрямовані інформаційні та керуючі зв’язки. Високорівневе програмне забезпечення забезпечує інтелектуальну обробку інформації та прийняття рішень, тоді як низькорівневе ПЗ гарантує

детерміноване, швидке та надійне виконання цих рішень у фізичному світі. Саме така ієрархічна, але взаємопов'язана структура є типовою для сучасних КФС і дозволяє поєднати складні алгоритми з реальними обмеженнями фізичних процесів.

Формально КФС подамо як ієрархічну замкнену динамічну систему з керуванням, яка поєднує фізичний об'єкт, обчислювальні підсистеми та канали взаємодії між ними. Узагальнено КФС задамо кортежем (2.1):

$$M_{KFS} = \langle P, S, A, C, N \rangle, \tag{2.1}$$

де P – фізичний об'єкт (процес);

S – множина сенсорів;

A – множина актуаторів;

C – множина обчислювальних та керуючих підсистем;

N – канали інформаційної та керуючої взаємодії.

Фізичний об'єкт P задамо системою диференціальних або різницевих рівнянь стану як у формулі (2.2):

$$\begin{aligned} x_1(t) &= f\big(x(t), u(t), w(t)\big); \\ y_1(t) &= h\big(y(t), v(t)\big), \end{aligned} \tag{2.2}$$

де x(t) $\epsilon\, R^n$ – вектор стану фізичної системи (положення, швидкість, орієнтація тощо);

u(t) $\epsilon\, R^m$ – вектор керуючих впливів, що реалізуються актуаторами (дросель, рульове керування);

w(t) – зовнішні збурення;

y(t) – вихідні величини, доступні для вимірювання;

v(t) – вимірювальний шум;

$(x_1(t), y_1(t))$ – координати об'єкту P;

t – поточний час.

Сенсорна підсистема S реалізує відображення фізичного стану у цифрові дані і задамо її за формулою (2.3):

$$z_1(t) = S\big(y(t)\big), \tag{2.3}$$

де y(t) – вихідні величини, доступні для вимірювання;

t – поточний час.

У дискретному часі визначення відображення фізичного стану у цифрові дані задамо формулою (2.4):

$$z_k = h_k(y_k) + v_k, \tag{2.4}$$

де y_k – вихідні величини, доступні для вимірювання у визначений час та через певні інтервали часу;

v_k – вимірювальний шум у визначений час та через певні інтервали часу.

Обчислювальна підсистема C має ієрархічну структуру. Подамо її композицією трьох рівнів (2.5):

$$C = C_1 \cdot C_2 \cdot C_3, \tag{2.5}$$

де C_1 – шар сприйняття;

C_2 – шар планування;

C_3 – шар керування.

Шар сприйняття виконує оцінку стану та середовища. Задамо процес його виконання за формулою (2.6):

$$\hat{x}_k = C_1(z_k), \tag{2.6}$$

де $\hat{x}_k$ – оцінений стан системи та її оточення, отриманий шляхом фільтрації, злиття даних і побудови внутрішніх моделей.

Шар планування формує цілі та траєкторії. Задамо його формулою (2.7):

$$r_k = C_2(\hat{x}_k, Q), \tag{2.7}$$

де r_k – вектор цільових значень або траєкторій;

Q – множина глобальних цілей і обмежень.

Шар керування перетворює цільові параметри у сигнали актуаторів. Задамо його згідно формули (2.8):

$$u_k = C_3(r_k, \hat{x}_k), \tag{2.8}$$

де C_3 реалізується на мікроконтролері та працює в реальному часі;

r_k – вектор цільових значень або траєкторій;

$\hat{x}_k$ – оцінений стан системи та її оточення, отриманий шляхом фільтрації, злиття даних і побудови внутрішніх моделей.

Актуаторна підсистема A здійснює відображення керуючих сигналів у фізичні дії як у формулі (2.9):

$$u(t) = A(u_k), \tag{2.9}$$

де u_k – сигнали актуаторів.

Цифрові команди перетворюються на електричні, механічні або гідравлічні впливи. У даній КФС це відповідає керуванню дроселем (поздовжня динаміка) та рульовим механізмом (бокова динаміка).

Взаємодія між усіма компонентами відбувається через мережу зв’язків N, яка забезпечує:

1) передачу сенсорних даних $S \rightarrow C_1$;

2) передачу рішень $C_2 \rightarrow C_3$;

3) передачу керування $C_3 \to A$;

4) зворотній зв’язок $P \to S$.

Таким чином, формальна модель КФС замикається у цикл (2.10):

$$x \to y \to z \to \hat{x} \to r \to u \to x, \tag{2.10}$$

який реалізує безперервну кіберфізичну взаємодію між фізичним процесом і цифровими обчисленнями.

Запропонована модель відображає ключові властивості типових КФС: ієрархічність; наявність замкнених контурів керування; розподіл між високорівневими та низькорівневими алгоритмами; зв’язок між обчислювальними та фізичними компонентами.

Деталізуємо моделі процесів в типовому класі КФС, які задано за формулами (2.1) – (2.10). Для цього розглянемо мобільну роботизовану систему.

Здійснимо конкретизацію моделі фізичного об’єкта. Нехай фізичним об’єктом є мобільна платформа типу «автомобільна кінематика». Тоді, вектор стану задамо формулою (2.11):

$$x(t) = \begin{bmatrix} X(t) \\ Y(t) \\ \psi(t) \\ v(t) \end{bmatrix}, \tag{2.11}$$

де $(X(t), Y(t))$ – координати платформи в поточний момент часу t;

$\psi(t)$ – курс (орієнтація);

$v(t)$ – лінійна швидкість.

Вектор керування (актуатори) задамо формулою (2.12):

$$u(t) = \begin{bmatrix} \alpha(t) \\ \delta(t) \end{bmatrix}, \tag{2.12}$$

де $\alpha(t)$ – поздовжнє прискорення (дросель) в поточний момент часу t;

$\delta(t)$ – кут повороту коліс (рульове керування).

Динаміку, деталізацію функції f (формула (2.2)), задамо формулою (2.13):

$$f(t) = \begin{cases} X(t) = v(t) \cdot \cos\big(\psi(t)\big); \\ Y(t) = \sin\big(\psi(t)\big); \\ \psi(t) = \frac{v(t)}{L} \cdot tg\big(\delta(t)\big); \\ v(t) = \alpha(t), \end{cases}, \tag{2.13}$$

де L – довжина колісної бази;

$(X(t), Y(t))$ – координати платформи в поточний момент часу t;

$\psi(t)$ – курс (орієнтація); $v(t)$ – лінійна швидкість;

$\alpha(t)$ – поздовжнє прискорення (дросель) в поточний момент часу t;

$\delta(t)$ – кут повороту коліс (рульове керування).

Числовий приклад для моделі, яку задано за формулою (2.13) подано в табл. 2.1.

Таблиця 2.1 – Числові значення для моделі

Параметр	Значення	Опис
L	2.5 м	колісна база
$X_0(t)$	0 м	початкова координата
$Y_0(t)$	0 м	початкова координата
$\psi_0(t)$	0 рад	напрям руху
$v_0(t)$	2.0 м/с	початкова швидкість
$\alpha(t)$	0.5 м/с²	команда дроселя
$\delta(t)$	0.1 рад	кут керма

Результат обчислення згідно даних з табл. 2.1 і формули (2.13):

$$f(t) \approx 0.08 \text{ рад/с}.$$

Сенсорну модель, функцію $h\big(y(t), v(t)\big)$, визначимо за формулою (2.4) з врахуванням того, що сенсори вимірюють частини процесів, формулою (2.14):

$$z_k = \begin{bmatrix} X(t) \\ Y(t) \\ \psi(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} \mu_{X(t)} \\ \mu_{Y(t)} \\ \mu_{\psi(t)} \\ \mu_{v(t)} \end{bmatrix}, \tag{2.14}$$

де $(X(t), Y(t))$ – координати платформи в поточний момент часу t;

$\psi(t)$ – курс (орієнтація);

$v(t)$ – лінійна швидкість; $\mu_{X(t)}$, $\mu_{Y(t)}$, $\mu_{\psi(t)}$, $\mu_{v(t)}$ – похибки відповідно для величин $X(t)$, $Y(t)$, $\psi(t)$, $v(t)$.

Тоді, наприклад в табл. 2.2 отримано значення за результатами вимірювань.

Таблиця 2.2 –Результати вимірювань з урахуванням похибок для величин

Сенсор	Значення	Похибка
$X(t)$	10.02 м	±0.5 м
$Y(t)$	4.95 м	±0.5 м
$\psi(t)$	0.52 рад	±0.01 рад
$v(t)$	1.95 м/с	±0.05 м/с

Здійснимо оцінювання стану шару сприйняття. Якщо використати простий фільтр, тобто спрощену форму фільтру, то отримаємо такий вираз:

$$\hat{x}_k = \alpha \cdot \hat{x}_{k-1} + (1 - \alpha) \cdot z_k, \tag{2.15}$$

де $\alpha = 0{,}7$;

$\hat{x}_k$ – оцінений стан системи та її оточення, отриманий шляхом фільтрації, злиття даних і побудови внутрішніх моделей;

z_k – результати відображення фізичного стану у цифрові дані.

Приклад оцінювання швидкості подано в табл. 2.3.

Таблиця 2.3 – Приклад оцінювання швидкості

Крок	$v_0(t)$	$v_{k-1}(t)$	$v_k(t)$
k	1.95	2.00	1.985

Планування руху, тобто деталізацію r_k, для шару планування, який формує бажану швидкість та курс, заламо так:

$$r_k(t) = \begin{bmatrix} v_k(t) \\ \psi_k(t) \end{bmatrix}, \tag{2.16}$$

де $\psi_k(t)$ – курс (орієнтація); $v_k(t)$ – лінійна швидкість.

Наприклад, цілю системи є рух по прямій зі швидкістю 2,5 м/с, тоді отримуємо такі значення: $v_k(t) = 2{,}5\,\frac{\text{м}}{\text{с}}$; $\psi_k(t) = 0{,}5$ рад.

Низькорівневе керування, тобто деталізація C_1, потребує визначення поздовжнього та бокового регуляторів. Поздовжній регулятор визначимо так:

$$\alpha(t) = K_\alpha \cdot \big(v_\alpha(t) - v(t)\big), \tag{2.17}$$

де, наприклад, $K_\alpha = 1{,}2$, то тоді $\alpha(t) \approx 0{,}53\ \text{м/с}^2$.

Боковий регулятор задамо так:

$$\delta(t) = K_\psi \cdot \big(\psi_\delta(t) - \psi(t)\big), \tag{2.18}$$

де, наприклад, $K_\psi = 0{,}8$, $\psi_\delta(t) = 0{,}5$ рад, $\psi(t) = 0{,}52$ рад, то тоді $\delta(t) \approx -0{,}016$ рад.

Згідно формул (2.11)–(2.18) встановимо значення актуаторів, використовуючи числову інтерпретацію з наведених прикладів з числовими значеннями параметрів. Дросель в поточний момент часу $\alpha(t) \approx 0{,}63\ \text{м/с}^2$, що

відображає збільшення тяги. Рульовий механізм – $\delta(t) \approx -0{,}016$ рад, що вказує на корегування напряму руху.

Відображення замкненого циклу згідно формули (2.10) подамо векторами у табл. 2.4.

Таблиця 2.4 – Значення векторів для відображення замкненого циклу

Етап	Дані
Сенсори	z_k = (10.02,;4.95,;0.52,;1.95)
Оцінка	$\hat{x}_k$ = (10.01,;4.97,;0.51,;1.99)
Планування	$r_k(t)$ = (2.5,;0.5)
Керування	u_k = (0.62,;–0.016)

Фізичний вплив призводить до зміни швидкості і напряму руху, що підтверджується даними з табл. 2.4.

Таким чином, за формулами (2.11)–(2.18) і в табл. 2.1–2.4 подана деталізована модель класу мобільних роботизованих систем, що є типовим класом КФС, який визначається узагальненою формальною моделлю КФС згідно формул (2.1)–(2.10) та рис. 2.1. При цьому, кожна абстрактна функція замінена конкретними рівняннями, усі змінні мають фізичний сенс, наведені реальні числові приклади, які є типовими для автономних систем.

Розроблено типову архітектуру класу КФС, у якій тісно поєднані фізичні компоненти, обчислювальні засоби та програмне забезпечення різних рівнів, а всі елементи взаємодіють через чітко визначені інформаційні та керуючі зв’язки. Фізичний рівень апаратного забезпечення включає сенсори та актуатори і є безпосередньо пов’язаним з реальним середовищем функціонування системи. Сенсори забезпечують первинний збір інформації про стан об’єкта керування та навколишнє середовище. Камера формує візуальні дані, що містять просторову та семантичну інформацію; інерціальний вимірювальний модуль надає дані про прискорення та кутові швидкості. GPS забезпечує глобальну просторову прив’язку та оцінку положення. Енкодери вимірюють параметри обертального або лінійного

руху виконавчих механізмів; лідар формує просторову картину оточення у вигляді хмар точок або відстаней до об’єктів. Усі ці сенсорні дані надходять у цифровому вигляді до обчислювальної частини системи, формуючи багатовимірний потік інформації, що характеризує поточний стан КФС. Розроблена таким чином архітектура КФС формує типовий клас КФС, який є основою для розроблення підходів до покращення управління пам’яттю апаратно–прискорених пристроїв, планування реального часу та спільного проєктування керування і планування.

2.2 Архітектура алокатора

Для типового класу КФС, який задано моделлю за формулою (2.1) розглянемо управління пам’яттю апаратно–прискорених пристроїв, планування реального часу та спільного проєктування керування і планування. Використаємо термін «домен пам'яті» для позначення будь–якої області пам'яті, пов'язаної з певним набором процесорів і гарантовано читабельною для процесорів у цьому наборі. Пам'ять хоста є окремим доменом від пам'яті GPU, а обидва окремі домени від пам'яті FPGA, і дані в одному домені мають бути скопіювані, щоб бути доступними в іншому домені. Визначимо спеціальні розподільчі алокатори з урахуванням гетерогенності, які керують пам'яттю для певної доменної області пам'яті відповідно до інтерфейсу, незалежного від домену. Вони мають функціональність доступу до пам'яті один одного та виконання операцій копіювання за потреби, коли вміст пам'яті не знаходиться у відповідній області. Визначимо також спільну чергу повідомлень, яка зберігає посилання на повідомлення, виділені розподільником. Видавці поміщали повідомлення в чергу для подальшого споживання підписниками. Коли абонент висловлює бажання отримати повідомлення в іншому домені пам'яті, то копія робиться миттєво у правильному домені пам'яті, а токен до дублікатної копії також зберігається в черзі повідомлень. Це найгірший сценарій. Коли дані переміщуються між доменами пам'яті, то необхідно виконати операцію копіювання. Кожна тема отримує унікальну чергу повідомлень. В зв’язку з такої проблемою і для її усунення

пропонується система, яка обробляє нульове копіювання повністю децентралізовано, і розробникам дозволяють застосовувати власні стратегії розподілу замість того, щоб змушувати їх використовувати статичний кільцевий буфер. Розроблений підхід не є справжньою нульовою копією у всіх застсоуваннях. Якщо застосунок частково апаратно прискорений, то потрібні копії пам'яті між частинами, що працюють у пам'яті хоста, і частинами, що працюють у пам'яті пристрою. Однак, якщо послідовні вузли працюють в одному домені пам'яті, то отримуємо переваги нульового копіювання незалежно від того, що це за домен пам'яті, тобто чи це хост, GPU чи у майбутньому FPGA пам'ять.

Проблема полягає в неефективному управлінні пам’яттю та обміні даними в гетерогенних обчислювальних системах, що поєднують хост–процесори, GPU та FPGA. Пам’ять кожного з цих пристроїв належить до окремого домену, і дані, розміщені в одному домені, не є безпосередньо доступними в іншому. У результаті будь–яка взаємодія між компонентами, що працюють у різних доменах пам’яті, потребує явного копіювання даних. Це призводить до додаткових витрат, збільшення затримок, ускладнення планування реального часу та зниження загальної продуктивності системи. Особливо критичним є найгірший сценарій, коли повідомлення необхідно дублювати для кожного підписника в іншому домені пам’яті, що суперечить ідеї ефективного та масштабованого обміну даними. Тому, запропоновано децентралізований підхід до управління пам’яттю та обміну повідомленнями, орієнтований на мінімізацію копіювань між доменами пам’яті. Вводиться поняття домен–незалежних розподільчих алокаторів, які керують пам’яттю конкретних доменів, але надають уніфікований інтерфейс доступу до даних. Такі алокатори самостійно виконують копіювання лише за необхідності, коли дані відсутні в потрібному домені. Додатково використовується спільна черга повідомлень, що зберігає посилання на дані, а не їхні копії, і підтримує механізми відкладеного копіювання. Це дозволяє реалізувати ефект нульового копіювання для послідовних вузлів, що працюють в одному домені пам’яті, незалежно від типу пристрою. Запропонований підхід надає розробникам можливість у виборі

стратегій розподілу пам’яті та усуває обмеження статичних буферних рішень, підвищуючи гнучкість і ефективність системи.

Тому розглянемо розроблення алокатору, зокрема з усвідомленістю гетерогенності. Кастомні розподілювачі пам’яті вже тривалий час застосовуються в системах реального часу, зокрема для статичного виділення пам’яті або повторного використання раніше створених об’єктів. Такий підхід дозволяє уникнути значних додаткових витрат, притаманних стандартним механізмам динамічного управління пам’яттю операційних систем, а також забезпечити детерміновану поведінку, критичну для часово–чутливих застосунків. З часом були розроблені більш складні абстракції, зокрема контейнери, орієнтовані на розподіл пам’яті, та поліморфні алокатори, які забезпечують гнучке використання різних стратегій розподілу. Завдяки цьому контейнери, які створені з використанням різних алокаторів, залишаються повністю сумісними між собою без порушення типової безпеки та інтерфейсної узгодженості.

Мінімальний алокатор, з точки зору функціональності, повинен підтримувати дві базові операції: виділення пам’яті; звільнення пам’яті. Деякі реалізації також беруть на себе відповідальність за побудову та знищення об’єктів, однак у запропонованому підході ця функція буде відсутня. Припустимо, що алокатори, хоча й можуть керувати пам’яттю периферійних пристроїв, але виконуються на центральному процесорі та не повинні здійснювати спроб прямого доступу до пам’яті пристрою, якою вони управляють.

Управління кількома доменами пам’яті в межах запропонованого підходу додає значно жорсткіші вимоги до архітектури алокатора, ніж проста підтримка операцій allocate та deallocate. Зокрема, необхідно враховувати взаємодію між доменами, механізми спільного використання пам’яті та забезпечення коректності доступу між процесами. У зв’язку з цим деталізуємо новий розширений інтерфейс алокатора, функції збору сміття, які необхідні для коректної роботи зі спільною пам’яттю, структуру самого алокатора та його пулу пам’яті, а також спосіб забезпечення достовірності при реконструкції алокатора в інших процесах.

Пропонується новий інтерфейс алокатора, який розширює традиційні можливості за рахунок підтримки перетворення довільної області пам’яті, виділеної іншим алокатором, у домен, читабельний для поточного розподілювача. Таке перетворення може реалізовуватися як тривіальна операція проходу, якщо обидва алокатори працюють в одному й тому самому домені пам’яті, що відповідає умові нульового копіювання. В іншому випадку виконуються всі необхідні операції копіювання даних між різними доменами.

Для одного домену пам’яті допускається існування кількох алокаторів. Припустимо, що будь–яке виділення може бути доступне будь–якому компоненту в межах того самого домену за умови виконання всіх необхідних операцій відображення пам’яті. Застосунковий інтерфейс включає шість основних функцій:

1) remap відображає алокатор, який раніше створений в іншому процесі, у поточний адресний простір;

2) unmap видаляє алокатор з адресного простору процесу;

3) share збільшує лічильник посилань для конкретного виділення;

4) copy_from копіює дані з алокатора у вказівник у пам’яті процесора;

5) copy_to копіює дані з пам’яті процесора в алокатор;

6) copy копіює дані з довільного алокатора в поточний, за потреби використовуючи пам’ять хоста як проміжний буфер.

Алокатори повертають не абсолютні вказівники, а цілочисельні зсуви. Це зумовлено тим, що один і той самий алокатор може бути відображений у різні області адресного простору різних процесів. Тому всі виділення задаються відносно початку алокатора, а перерахунок абсолютних адрес виконується локально в кожному процесі.

Розміщення об’єктів у спільній пам’яті може ініціюватися будь–яким потоком виконання. Хоча повідомлення зазвичай створюється видавцем, відповідальність за звільнення пам’яті несе той підписник, який останнім утримує посилання на відповідний об’єкт. Для потоків, чутливих до часових обмежень, операція deallocate повинна мати гарантований верхній час виконання, що суттєво обмежує перелік допустимих стратегій розподілу пам’яті. На поточному етапі

реалізації підтримується лише статичний кільцевий буфер, однак можна задати вибір інших стратегій. Незмінною вимогою залишається те, що всі стратегії розподілу повинні забезпечувати звільнення пам’яті з мінімальною часовою складністю. Кожен домен пам’яті може копіювати дані в пам’ять хоста і навпаки. Водночас не всі домени підтримують пряме копіювання між собою. Тому кожен алокатор повинен явно визначати можливість копіювання з власного домену, копіювання з домену в пам’ять хоста та копіювання з пам’яті хоста у власний домен. У більшості випадків функція copy реалізується як послідовність цих двох операцій. Проте за наявності відповідного апаратного забезпечення розробники можуть реалізовувати оптимізовані шляхи, що оминають пам’ять хоста, наприклад, пряме копіювання між GPU.

Алокатор з усвідомленістю гетерогенності зобов’язаний реалізовувати стратегію обчислення кількості посилань. Функція deallocate фактично звільняє пам’ять лише тоді, коли кількість викликів звільнення перевищує кількість викликів share. Такий механізм потенційно створює умову гонки, яка усувається за допомогою черги повідомлень. При реалізації алокаторів для пам’яті пристроїв методи розподілу не мають прямого доступу до виділеної пам’яті, тому класичні підходи, зокрема використання межових тегів, не застосовуються. Алокатори також надають метадані застосунковому рівню. Тип пристрою ідентифікує апаратний компонент, номер пристрою дозволяє розрізняти кілька екземплярів одного типу, а стратегія визначає конкретний підхід до розподілу пам’яті, наприклад кільцевий буфер. Для реалізації нульового копіювання тип і номер пристрою двох алокаторів повинні збігатися. Під час відображення алокатора в новий процес його ідентифікація здійснюється на основі ідентифікатора сегмента спільної пам’яті System V відповідно до стандарту POSIX.

Кожен алокатор складається з трьох суміжних відображень у пам’яті: локальної частини; спільної області, видимої між процесами; пулу пам’яті пристрою. Локальна частина зберігає вказівники на функції й використовується для емуляції об’єктно–орієнтованого поліморфізму. Це дозволяє черзі повідомлень працювати незалежно від конкретного типу алокатора. Справжній поліморфізм у

спільній пам’яті є неможливим через невизначеність структури деяких типів даних і стирання типів під час міжпроцесної взаємодії. Тому всі реалізації алокаторів мають ідентичну структуру в перших байтах пам’яті. Для досягнення цього використовується виключно звичайні структури даних без прихованих внутрішніх залежностей. Оскільки вказівники на функції не є коректними між різними процесами, частина алокатора, що їх зберігає, не відображається як спільна пам’ять і залишається локальною для кожного процесу. На рис. 2.2 зображено структуру алокатора та її зв’язок з інтерфейсом.

Відповідно до інтерфейсу алокатора, зображеного на рис.2.2, структура алокатора спроєктована таким чином, щоб забезпечити підтримку гетерогенних доменів пам’яті, міжпроцесну взаємодію та реалізацію операцій нульового або мінімізованого копіювання. Алокатор складається з трьох логічно та фізично відокремлених частин, кожна з яких безпосередньо пов’язана з функціями інтерфейсу.

Локальна частина алокатора розміщується в адресному просторі конкретного процесу та не є спільною між процесами. Саме в цій частині зберігаються вказівники на функції, що реалізують інтерфейс алокатора. Локальна частина виконує роль диспетчера викликів. Вона визначає, яка конкретна реалізація функції буде використана для даного типу пристрою та стратегії розподілу. Такий підхід імітує об’єктно–орієнтований поліморфізм у середовищі, де використання справжнього поліморфізму неможливе через міжпроцесні обмеження. Оскільки вказівники на функції не мають коректного значення за межами процесу, то ця частина не може бути розміщена у спільній пам’яті.

Спільна частина алокатора розміщується в сегменті спільної пам’яті (System V) і є ідентичною для всіх процесів, які відображають відповідний алокатор за допомогою функції remap. У цій частині зберігаються:

1) метадані алокатора (тип пристрою, номер пристрою, стратегія розподілу);

2) інформація, необхідна для коректної реконструкції алокатора в іншому процесі;

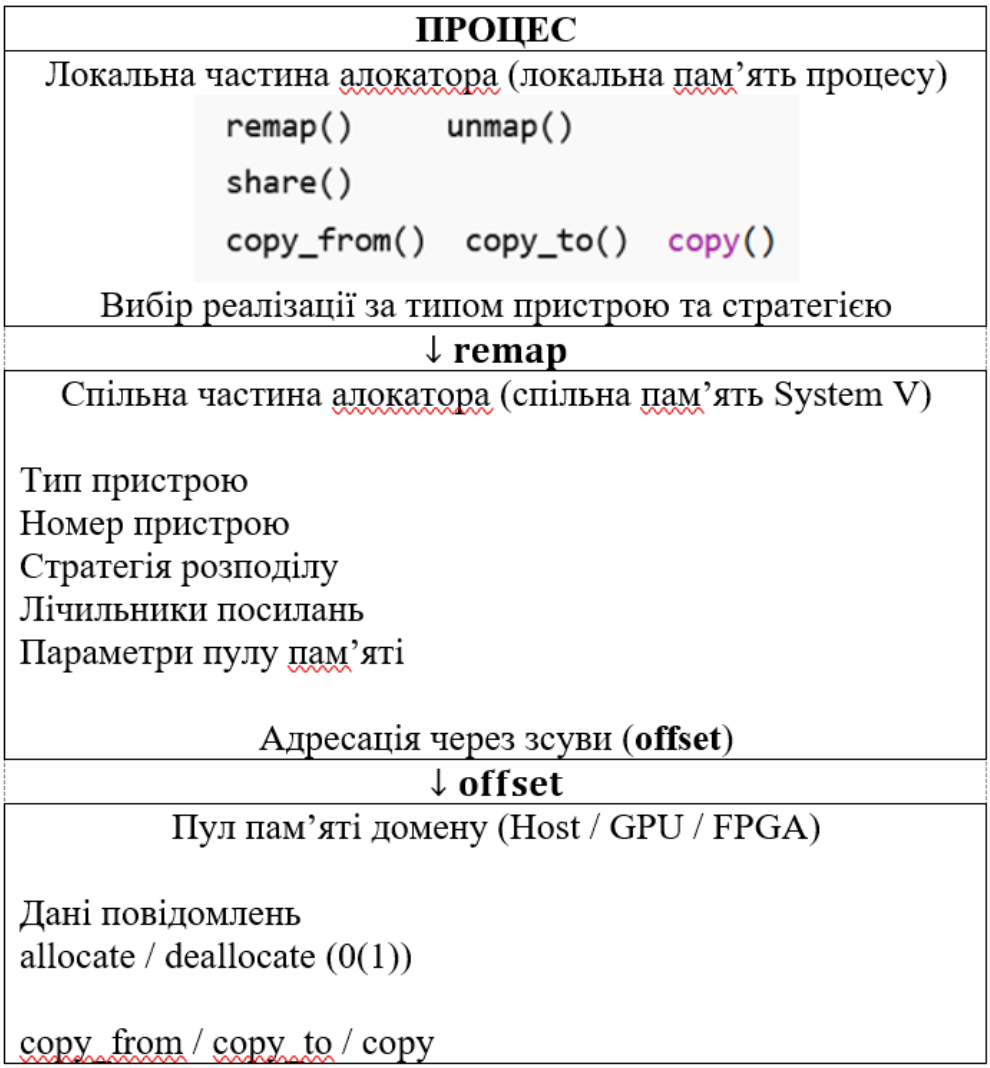


Рисунок 2.2 – Структура алокатора та її зв'язок з інтерфейсом

3) службові структури для обчислення кількості посилань, що використовуються функціями share та deallocate;

4) параметри пулу пам'яті, з яким працюють операції копіювання.

Саме ця частина дозволяє різним процесам коректно інтерпретувати цілочисельні зсуви, які повертаються алокатором замість абсолютних вказівників. Функції copy, copy_from та copy_to використовують спільні метадані для визначення допустимих шляхів копіювання між доменами пам'яті, зокрема для вибору між прямим копіюванням і копіюванням через пам'ять хоста.

Третьою складовою структури алокатора є пул пам'яті, що відповідає конкретному домену пам'яті, тобто пам'яті хоста, GPU або іншого апаратного

прискорювача. Цей пул може бути фізично розміщений у пам’яті пристрою або відображений у віртуальний адресний простір процесу, залежно від типу домену.

Функції copy_from та copy_to забезпечують перенесення даних між цим пулом і пам’яттю хоста, тоді як функція copy реалізує узагальнений механізм передачі даних між двома пулами пам’яті різних алокаторів. У випадку, коли обидва алокатори мають однаковий тип і номер пристрою, функція copy може бути зведена до операції проходу без фактичного копіювання, що відповідає режиму нульового копіювання.

Таким чином, структура алокатора безпосередньо відображає інтерфейс. Локальна частина забезпечує поліморфну поведінку функцій, спільна частина – узгодженість між процесами та коректне обчислення кількості посилань, а пул пам’яті пристрою – фізичне зберігання даних у відповідному домені. Шість основних функцій інтерфейсу охоплюють повний життєвий цикл алокатора, зокрема від відображення та спільного використання до керування копіюванням даних між гетерогенними доменами пам’яті.

Розроблений варіант алокатора відрізняється від відомих варіантів вирішення проблеми управління пам’яттю. Розглянемо відмінності між ними. Найпростішим підходом до розв’язання проблеми обміну даними між різними доменами пам’яті є централізоване копіювання через пам’ять хоста. У цьому випадку всі дані з пам’яті прискорювачів (GPU, FPGA) спочатку копіюються в пам’ять хоста, а вже звідти копіюються до цільового домену. Такий підхід є універсальним і не потребує складних механізмів синхронізації, однак призводить до значних додаткових витрат, збільшення затримок і не підходить для систем реального часу або високопродуктивних застосунків.

Іншим підходом є статичне розміщення даних у певному домені пам’яті з жорстким закріпленням обчислювальних вузлів за цим доменом. Дані створюються та використовуються лише в межах одного домену, що мінімізує кількість копіювань. Недоліком такого рішення є низька гнучкість, складність масштабування та обмежені можливості повторного використання компонентів системи.

Проблему можна вирішити шляхом запровадження уніфікованого інтерфейсу доступу до пам’яті, який приховує від застосунку деталі фізичного розташування даних. У цьому випадку всі операції роботи з пам’яттю виконуються через абстракцію, яка самостійно визначає необхідність копіювання між доменами. Такий підхід підвищує зручність розробки, але може приховувати значні витрати та ускладнювати гарантування часових обмежень.

Ефективнішим рішенням є використання алокатора з усвідомленістю гетерогенності, який явно враховує існування кількох доменів пам’яті. Алокатор керує пам’яттю кожного домену окремо, але надає домен–незалежний інтерфейс. Копіювання між доменами виконується лише за необхідності, а у випадку, коли послідовні обчислювальні вузли працюють в одному домені пам’яті, забезпечується режим нульового копіювання. Такий підхід дозволяє зменшити додаткові витрати, зберегти детермінованість і підвищити масштабованість системи.

Ще одним варіантом є децентралізоване управління пам’яттю, у якому відповідальність за життєвий цикл даних розподіляється між видавцями та підписниками. Використання підрахунку посилань дозволяє безпечно звільняти пам’ять після завершення її використання всіма споживачами. У поєднанні зі спільною чергою повідомлень та домен–орієнтованими алокаторами цей підхід мінімізує кількість копіювань і усуває глобальні точки блокування.

Найбільш практичним є комбінований підхід, який поєднує статичне розміщення критичних даних, гетерогенно–усвідомлені алокатори та явні механізми копіювання лише в критичних точках взаємодії між доменами. Такий підхід дозволяє досягти балансу між продуктивністю, гнучкістю та передбачуваністю часових характеристик, що є особливо важливим для кіберфізичних систем та систем реального часу.

Таким чином, проблема управління пам’яттю в гетерогенних системах може бути вирішена різними способами, однак найбільш ефективним є підхід, що поєднує домен–орієнтоване управління пам’яттю, мінімізацію копіювань і

децентралізований контроль життєвого циклу даних. Саме такий підхід забезпечує масштабованість, детермінованість та ефективне використання апаратних ресурсів.

Основний внесок гетерогенності нульових копій для гетерогенних обчислювальних середовищ полягає в тому, що копії пам'яті виконуються лише тоді, коли дані в одній області пам'яті потрібні в іншій області. У всіх інших випадках пам'ять перероблюється, що суттєво знижує найгіршу наскрізну затримку застосунку, а також усуває джерела недетермінованості. Найбільші переваги продуктивності здебільшого спостерігаються, коли послідовні компоненти в застсоунках працюють у одній області. Навіть у найгіршому випадку, для застосунку, що використовує n доменів, не буде більше ніж n–1 операції копіювання на повідомлення. Крім зростання продуктивності апаратних прискорень робочих навантажень, розроблений алокатор має децентралізовану архітектуру, що додатково зменшує необхідні знання розробника, оскільки немає сторонніх демонів для запуску і немає кільцевих буферів для налаштування. Вбудовані налаштування за замовчуванням та ініціалізація під час виконання дозволяють йому працювати навіть при обмеженій інформації про застосунок.

2.3 Висновки до другого розділу

Розроблено типову архітектуру класу кіберфізичних систем (КФС), у якій інтегровані фізичні компоненти, обчислювальні засоби та програмне забезпечення різних рівнів, що взаємодіють через визначені інформаційні й керуючі зв’язки. Фізичний рівень включає сенсори й актуатори, безпосередньо пов’язані з реальним середовищем. Сенсори забезпечують збір даних про стан об’єкта та оточення: камера формує візуальну інформацію, IMU надає дані про прискорення і кутові швидкості, GPS забезпечує просторову прив’язку, енкодери вимірюють рух виконавчих механізмів, а лідар формує просторову модель середовища. Усі дані надходять до обчислювальної частини у вигляді багатовимірного потоку, що описує поточний стан КФС. Така архітектура є основою для досліджень управління

пам’яттю апаратно–прискорених пристроїв, планування реального часу та спільного проєктування керування і планування.

Основна перевага гетерогенності нульових копій полягає у виконанні копіювання пам’яті лише за потреби доступу до даних в іншій області пам’яті. В інших випадках пам’ять повторно використовується, що зменшує наскрізну затримку та усуває джерела недетермінованості. Найвищий приріст продуктивності досягається, коли послідовні компоненти застосунку працюють в одному домені. Навіть у найгіршому випадку кількість копіювань не перевищує $n-1$ для n доменів. Запропонований алокатор має децентралізовану архітектуру, не потребує сторонніх демонів чи складного налаштування і може ефективно працювати за умов обмеженої інформації про застосунок.

3 МЕТОД ПЛАНУВАННЯ ПРОЦЕСАМИ РЕАЛЬНОГО ЧАСУ В КІБЕРФІЗИЧНИХ СИСТЕМАХ

3.1 Метод планування процесами реального часу

У КФС з апаратно–прискореними пристроями (GPU, FPGA, AI–акселератори) традиційне копіювання даних між центральною пам’яттю та пам’яттю прискорювача призводить до значних затримок доступу до даних, порушення дедлайнів задач реального часу, надмірного використання ресурсів пам’яті і шини передачі даних, ускладнення спільного проєктування керування і планування через непередбачувані затримки. Проблема полягає у забезпеченні нульового копіювання (zero–copy) у КФС, тобто прямого доступу прискорювачів до даних у пам’яті без проміжного копіювання, при цьому зберігаючи детермінованість виконання задач і стабільність керування. Тому, необхідно забезпечити уникнення повторних копіювань, спільне проєктування керування, планування реального часу та алокатора пам’яті, що повинно ефективно використовувати нульове копіювання для мінімізації затримок передачі даних, уникнення фрагментації пам’яті прискорювачів, гарантування виконання критичних задач у межах дедлайнів.

Розглянемо основні кроки методу планування процесами реального часу в КФС, які повинні забезпечити уникнення проблеми з повторним копіюванням.

1. Моделювання КФС та потоків даних. Формалізація задач реального часу і їхніх вимог до обчислень та пам’яті. Ідентифікація даних, що можуть бути використані через zero–copy (прямий доступ апаратного прискорювача без проміжних буферів).

2. Аналіз архітектури пам’яті прискорювача. Визначення областей пам’яті, які підтримують zero–copy (наприклад, pinned memory у GPU або shared memory на FPGA). Моделювання часу доступу до цих областей і обмежень на одночасний доступ.

3. Розробка детермінованого алокатора пам’яті. Статична або напівстатична алокація memory regions для zero–copy, щоб уникнути динамічного копіювання під

час виконання. Визначення верхніх меж часу доступу та забезпечення сумісності з планувальником задач.

4. Планування задач реального часу з урахуванням zero–copy. Інтеграція інформації про доступні zero–copy буфери у планування та забезпечення того, щоб критичні задачі отримували прямий доступ до пам’яті без очікування копіювання. Мінімізація конфліктів за доступ до zero–copy буферів між паралельними задачами.

5. Спільне проєктування керування і планування. Адаптація параметрів регуляторів з урахуванням нульових копій для уникнення затримок. Комбінування керування і планування з інтеграцією zero–copy для гарантування стабільності системи.

6. Верифікація часових і функціональних гарантій. Перевірка виконання дедлайнів у найгіршому випадку при використанні zero–copy. Аналіз впливу zero–copy на стабільність керування та якість регулювання та перевірка, що використання zero–copy ефективно зменшує затримки і навантаження на шину пам’яті.

7. Експериментальна оцінка. Реалізація підходу на цільовій платформі з апаратними прискорювачами. Порівняння продуктивності та часу доступу при класичному копіюванні і zero–copy. Оцінка виграшу у передбачуваності, затримках та ефективності використання пам’яті.

Схему кроків методу планування процесами реального часу в КФС подамо в контексті інтегрованого підходу до керування процесами в КФС, яку зображено на рис. 3.1. Схема ілюструє інтегрований підхід до керування кіберфізичною системою з апаратно–прискореними пристроями з акцентом на нульове копіювання даних. Центральним елементом є розподілений граф потоків даних, який показує, як сенсорні дані надходять до центрального процесора (ЦП) та направляються до апаратного прискорювача (GPU або FPGA) для швидкої обробки без проміжного копіювання. Стрілки між ЦП та прискорювачем демонструють прямий доступ до спільної пам’яті, що дозволяє уникати багаторазового дублювання даних у проміжних буферах. Блок «Проєктування управління» відповідає за алгоритми керування, налаштування регуляторів та компенсацію

затримок, забезпечуючи, щоб керуючі команди правильно враховували потоки даних у zero–copy режимі.

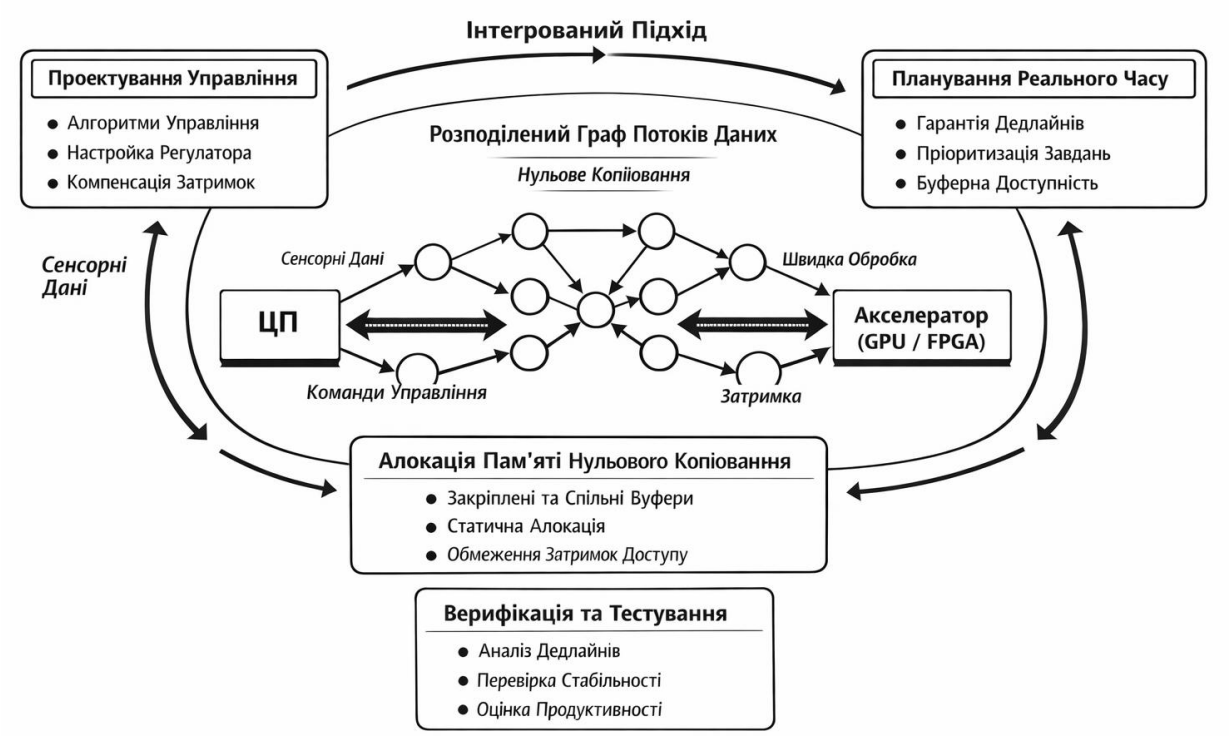


Рисунок 3.1 – Інтегрований підхід до керування процесами в КФС

«Планування реального часу» інтегрує гарантії дедлайнів, пріоритизацію задач та доступність буферів, координуючи порядок обробки задач з наявними ресурсами пам’яті, щоб уникнути блокувань і затримок. Блок «Алокація пам’яті нульового копіювання» реалізує закріплені та спільні буфери, статичну алокацію та обмеження затримок доступу, що гарантує детермінованість і ефективне використання пам’яті прискорювачів. Верифікація та тестування здійснюють аналіз дедлайнів, перевірку стабільності та оцінку продуктивності, підтверджуючи, що потоки даних і керуючі сигнали не створюють надмірного дублювання і забезпечують оптимальне виконання задач. Всі ці елементи та потоки працюють у єдиній інтегрованій системі, де нульове копіювання дозволяє максимально скоротити затримки, підвищити передбачуваність та забезпечити ефективне спільне проєктування керування і планування реального часу.

Основним елементом планування є виконавець (executor), який управляє виконанням зворотних викликів. Стандартний виконавець, відомий як

SingleThreadedExecutor, працює у єдиному потоці та використовує механізм ready set, що зберігає зворотні виклики, готові до виконання. Ready set відображає лише факт готовності, а не кількість подій у черзі. Обробка викликів відбувається у вікнах обробки, між якими існують точки опитування. Це створює потенційні затримки, інверсії пріоритетів та обмежує кількість виконуваних екземплярів одного зворотного виклику. Виконавець застосовує кругову політику обробки з віддачею переваги таймерам над підписками і обмежує виконання одного екземпляра завдання за вікно обробки. Така схема з фіксованими пріоритетами та блокуванням у вікнах обробки ускладнює надання жорстких гарантій реального часу, особливо для складних застосунків. Коли дані надходять на тему, всі підписники активуються, формуючи ланцюги виконання. Це дає можливість аналізувати наскрізні затримки та вплив виконання одного завдання на інші, але стандартний виконавець накладає обмеження через поведінку ready set.

Розглядаючи сумісність ROS2 з класичним плануванням періодичних завдань у реальному часі, виникає питання щодо можливості реалізувати класичні моделі без превентивного втручання. Це можливо за певних умов. Достатньо внести незначні зміни у виконавець подій, включно з плануванням з фіксованим пріоритетом та раннім дедлайном. Виконавець подій ROS2 має властивості, які роблять його більш придатним для цієї мети. Зокрема, він застосовує чергу подій FIFO, відокремлює випуск завдань від їх виконання і може використовувати окремі потоки для таймерів. Потік керування таймером випускає завдання негайно, тоді як стандартний потік виконує їх, що дозволяє уникати блокування випуску таймерних завдань. Така архітектура наближає ROS2 до класичного розділення релізера і планувальника, де релізер вставляє завдання у чергу готовності, а планувальник витягує їх для виконання.

Стандартний виконавець ROS2 суттєво відрізняється від класичного непреемптивного FP–планувальника. У класичних системах завдання звільняються одразу за допомогою таймерного переривання, тоді як у ROS 2 активовані завдання обираються лише на точках опитування. Це може спричиняти пропуск таймерних завдань та збільшення часу відгуку високопріоритетних завдань. Також

стандартний виконавець не надає прямих механізмів пріоритизації, тому порядок виконання визначається лише політикою FIFO та чергою ready set, що не відповідає класичним очікуванням. Для виправлення цих обмежень у виконавця подій застосовуються модифікації, зокрема використання опції Release–Only для відокремлення релізера та планувальника, правильна пріоритизація потоків та перетворення FIFO черги на пріоритетну. Таким чином можна реалізувати негайний випуск таймерних завдань і правильний порядок виконання у системах із фіксованим або динамічним пріоритетом. У результаті модифікацій виконавець подій ROS2 дозволяє суміщати класичні алгоритми із механізмами ROS 2, забезпечуючи контроль часу відгуку та порядок виконання завдань, що критично для копіювання нульового в КФС. Це дозволяє інтегрувати робочі ланцюги з таймерними та підписними завданнями, уникати пропусків релізів та забезпечувати передбачуваний час виконання, зберігаючи гнучкість ROS2 та зменшуючи витрати на проєктування системи. На рис. 3.2 зображено порівняння трьох підходів.

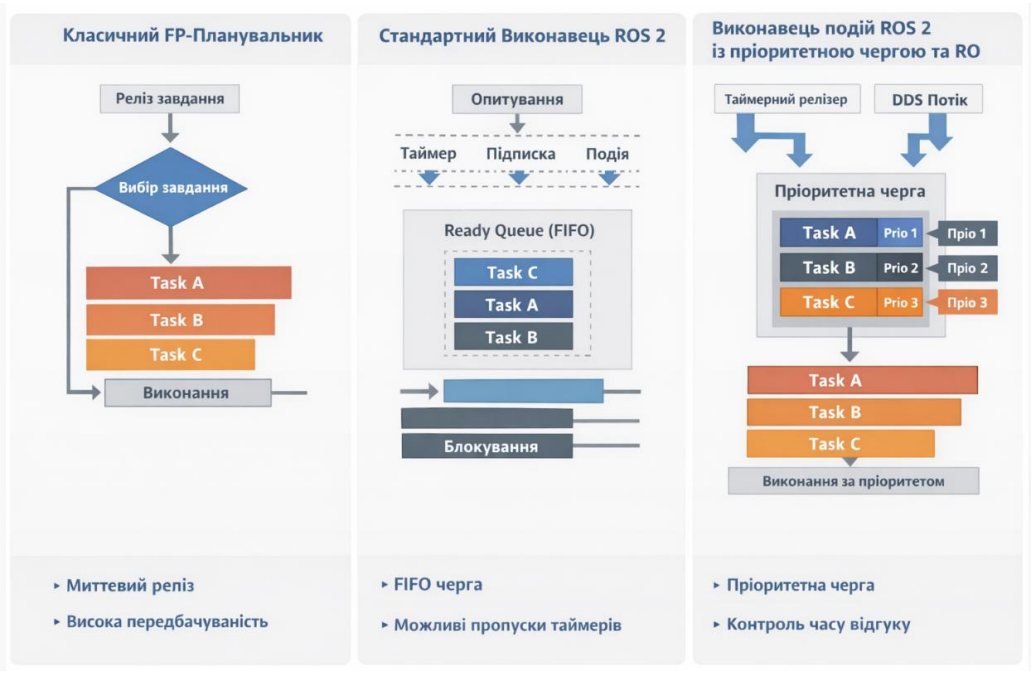


Рисунок 3.2 – Три підходи з використання планувальника завдань

На рис. 3.2 порівнюється три підходи до планування завдань у ROS2 та класичному FP–планувальнику. У класичному непревенптивному FP–

планувальнику завдання звільняються майже одразу за допомогою таймерного механізму, рішення про виконання нового завдання приймається після завершення попереднього, порядок виконання базується на пріоритетах або дедлайнах, що забезпечує високу передбачуваність часу відгуку та гарантію виконання всіх завдань. Стандартний виконавець ROS2 використовує точки опитування і FIFO чергу готових завдань, випуск завдань відбувається лише на точках опитування, а не миттєво після активації, можливі пропуски таймерних завдань і блокування високопріоритетних завдань через активні нижчі пріоритети, рішення про виконання всіх завдань у наборі ready set приймається лише один раз на точку опитування. Виконавець подій ROS2 з пріоритетною чергою та RO (Release–Only) відокремлює реліз таймерних завдань від їх виконання, потоки DDS та таймерів можуть негайно вставляти завдання в чергу подій, черга подій пріоритизована за періодом або часовою міткою, що дозволяє реалізувати FP або EDF, виконання завдань відбувається у порядку пріоритетів, контроль часу відгуку покращений, а пропуски завдань мінімізовані. На рис. 3.2 показані потоки для кожного підходу, черги (ready set або черга подій), порядок випуску та виконання завдань, а також основні відмінності у механізмах планування і контролю пріоритетів.

Удосконалений таким чином планувальник використовує виконавця подій із модифікованою чергою подій, де завдання впорядковуються за пріоритетами. Таймерні завдання випускаються через потік керування таймерами, який має вищий пріоритет, ніж стандартний потік, тому щоразу, коли досягається часова мітка завдання, воно негайно вставляється в чергу подій. Це забезпечує періодичне звільнення робочих місць, а рішення про виконання приймається відразу після завершення завдання, витягуючи завдання з найвищим пріоритетом із черги. Якщо виконавець не працює і з’являється нове завдання, планування відбувається миттєво, починаючи його виконання, що дозволяє планувальнику діяти як типовий непревенптивний планувальник.

Максимальні додаткові витрати релізатора позначаємо $\delta > 0$, що є часом звільнення завдання, який може подовжувати виконання поточного завдання. Враховуючи максимальну кількість релізів під час виконання завдання, отримуємо

межу додаткових витрат для кожного завдання. Завдання звільняються щоразу при досягненні часової позначки, а кількість таких міток для завдання за інтервал. Нехай є мінімальний час, що включає виконання завдання та додаткові витрати на реліз інших завдань. Ці витрати необхідно враховувати при застосуванні класичних результатів непревенптивних планувальників. Верхня межа додатково обмежується, якщо завдання мають обмежені дедлайни і набір завдань уже визнано плануваним, оскільки тоді інші завдання можуть звільнити лише одну роботу під час виконання. Інакше подвійний реліз призведе до пропуску дедлайну та порушення планування.

Планувальник аналітично поводиться як типовий непревенптивний, підтримуючи FP та EDF, з урахуванням додаткових витрат через продовження сесії. Найгірший час відповіді завдання визначається як максимальний інтервал між його випуском і завершенням роботи. Це дозволяє застосовувати класичні підходи з непревенптивного планування періодичних завдань, визначаючи верхню межу найгіршого часу відгуку, якщо набір завдань плануваний.

Важливою метрикою є наскрізна затримка ланцюга причинно–наслідкових завдань, яка показує час проходження даних через ланцюг. Максимальний час реакції та максимальний вік даних еквівалентні, тому використовуємо одне поняття для кінцевої затримки. Хоча аналітичних результатів для стандартного ROS2 обмежено, для періодичних систем дозволяється отримувати детальні оцінки. Планувальник забезпечує можливість застосування таких аналізів для повного циклу оцінки періодичних систем завдань у ROS2.

Для обох моделей планувальників важливо забезпечити пріоритизацію підписних завдань, оскільки без цього важко гарантувати своєчасне оброблення повідомлень і контроль часу відгуку системи. У ROS 2 немає прямого механізму для встановлення пріоритету підписного завдання, що створює обмеження для тонкої настройки планування. Однак існують два перспективні підходи, які дозволяють інтегрувати пріоритети підписних завдань у існуючий механізм виконавця.

Перший підхід ґрунтується на використанні DDS у ROS2 та налаштуванні параметрів якості обслуговування для теми. Кожен підписник у системі підписується на конкретну тему, і ця тема має свої параметри якості, серед яких вже існує дедлайн, що описує очікуваний максимальний інтервал між повідомленнями, що публікуються на тему. Цей механізм можна розширити, додавши параметр якості, який відображає очікуваний мінімальний інтервал між надходженням повідомлень. Це дозволить враховувати мінімальний час прибуття даних при прийнятті рішень про планування та забезпечить більш передбачувану обробку підписних завдань. Для повної інтеграції цього механізму у планувальник потрібні модифікації виконавця, які дозволять використовувати ці параметри якості, а саме враховувати як мінімальний час між надходженнями, так і дедлайни при обчисленні порядку виконання завдань. Таким чином, підписні завдання можна буде ефективно впорядковувати поряд із таймерними завданнями, що значно покращить контроль часу реакції системи.

Другий підхід передбачає створення окремого поля пріоритету безпосередньо для підписок, яке користувач зможе визначати при створенні підписника. Це більш прямий і гнучкий метод, оскільки дозволяє вручну визначати порядок виконання підписних завдань у черзі подій і легко інтегруватися з існуючою логікою виконавця. Використовуючи це поле, можна застосовувати ті ж процедури планування, що використовуються для таймерних завдань у нашому підході, описаному раніше, включаючи підтримку як статичних пріоритетів, так і динамічних, наприклад для алгоритмів FP або EDF. Цей метод дозволяє максимально точно налаштувати пріоритети для будь–яких підписників і гарантує, що критично важливі повідомлення оброблятимуться своєчасно, незалежно від завантаження інших завдань у системі.

У будь–якому з обох випадків корисно передбачити інтерфейс для динамічного встановлення пріоритетів у виконавця, що дозволить змінювати порядок обробки підписних завдань під час роботи системи. Це значно підвищує сумісність із існуючими виконавцями, оскільки не потребує радикальної зміни їхньої архітектури, і водночас дозволяє легко впроваджувати інші політики

планування. Варто відзначити, що пріоритетна черга, яку введено для заміни черги подій, може застосовуватися як для таймерних завдань, так і для підписок. При цьому для обробки підписних завдань не потрібні додаткові зміни у коді користувацьких застосунків, що робить цей підхід зручним і практичним для інтеграції в наявні ROS 2 системи. Таким чином, обидва підходи дозволяють ефективно контролювати пріоритети підписних завдань і забезпечують можливість точного та передбачуваного планування поряд із таймерними завданнями, підвищуючи загальну ефективність роботи системи.

У сучасних кіберфізичних системах із апаратно–прискореними пристроями, такими як GPU, FPGA та AI–акселератори, традиційні підходи до копіювання даних між центральною пам’яттю та пам’яттю прискорювача створюють низку критичних проблем. Зокрема, багаторазове копіювання викликає значні затримки доступу до даних, порушує дедлайни критичних задач реального часу, підвищує навантаження на шину даних і пам’ять, а також ускладнює спільне проєктування керування та планування. Це негативно впливає на передбачуваність та стабільність виконання завдань у системі, що є критичним у реальному часі. Тому забезпечення нульового копіювання (zero–copy), тобто прямого доступу апаратних прискорювачів до даних у пам’яті без проміжного дублювання, стає ключовою вимогою для підвищення ефективності та детермінованості КФС.

Розроблений метод до планування задач у КФС базується на комплексному інтегрованому підході, який поєднує моделювання потоків даних, аналіз архітектури пам’яті прискорювачів, детерміноване виділення пам’яті, пріоритетне планування задач реального часу та спільне проєктування керування і планування. На першому етапі відбувається формалізація задач реального часу та визначення потоків даних, які можуть використовувати zero–copy для прямого доступу апаратних прискорювачів без проміжних буферів. Потім аналізуються апаратні області пам’яті, що підтримують zero–copy, наприклад pinned memory у GPU або shared memory на FPGA, із врахуванням часу доступу та обмежень одночасного використання.

Детермінований алокатор пам’яті забезпечує статичне або напівстатичне виділення memory regions для zero–copy, що дозволяє уникнути динамічного копіювання під час виконання та гарантує верхні межі часу доступу, сумісні з планувальником задач. Детермінований алокатор пам’яті це механізм виділення пам’яті, який гарантує передбачуваний і фіксований час доступу та управління ресурсами. Його основна мета полягає в уникненні непередбачуваних затримок, характерних для динамічного виділення пам’яті, і забезпеченні того, щоб завдання реального часу могли одразу отримати потрібні блоки пам’яті без очікування або фрагментації. Це планове виділення пам’яті, де відомо заздалегідь, які ділянки пам’яті будуть використовуватись і скільки часу це займе, що робить роботу з пам’яттю детермінованою для систем реального часу.

Планування задач реального часу інтегрує інформацію про доступні zero–copy буфери, забезпечуючи, щоб критичні задачі отримували прямий доступ до пам’яті без очікування копіювання, а також мінімізуючи конфлікти за доступ до ресурсів між паралельними задачами. Спільне проєктування керування і планування адаптує параметри регуляторів та алгоритми керування з урахуванням zero–copy, що дозволяє уникати затримок і підтримувати стабільність системи навіть у разі високого навантаження.

Верифікація часових і функціональних гарантій дозволяє перевірити дотримання дедлайнів у найгіршому випадку та оцінити вплив zero–copy на стабільність керування та якість регулювання. Експериментальна оцінка на реальних апаратних прискорювачах демонструє значне скорочення часу доступу до даних, підвищення передбачуваності виконання задач, а також ефективніше використання пам’яті та шини передачі даних порівняно з класичним підходом із копіюванням.

Схема інтегрованого методу планування у КФС з zero–copy ілюструє центральну роль розподіленого графа потоків даних, де сенсорні дані надходять до центрального процесора та безпосередньо передаються апаратним прискорювачам для обробки без проміжного копіювання. Прямий доступ до спільної пам’яті дозволяє уникати дублювання даних у проміжних буферах, зменшуючи затримки

та навантаження на систему. Блок «Проєктування управління» відповідає за алгоритми керування, компенсацію затримок та налаштування регуляторів із урахуванням zero–copy, забезпечуючи коректне формування керуючих команд. Такий підхід дозволяє інтегрувати керування і планування реального часу, гарантувати виконання критичних задач у межах дедлайнів та максимально ефективно використовувати апаратні ресурси прискорювачів.

Удосконалений метод забезпечує комплексне вирішення проблеми повторного копіювання в КФС, підвищує детермінованість і стабільність системи, мінімізує затримки передачі даних і забезпечує ефективну інтеграцію керування та планування з використанням нульового копіювання. Це створює основу для реалізації високопродуктивних, передбачуваних і стабільних кіберфізичних систем з апаратними прискорювачами.

3.2 Планування у ROS2 з урахуванням графів

Розглянемо обмеження сучасних методів планування в системі ROS 2, а також основну увагу приділимо задачам, які виходять за межі простих ланцюгових структур, і дослідимо можливості аналізу довільних графів. Дослідження здебільшого фокусувалися на ланцюговому плануванні із тимчасовим аналізом часу відгуку, проте пропонується новий підхід, який базується на застосуванні виконавця подій для реалізації планувальників із фіксованим пріоритетом завдань у довільних графах ROS2. Це дозволяє подолати розрив між усталеною теорією систем реального часу та практичним аналізом планування в ROS2, виходячи за межі ad–hoc підходів і класичних ланцюгових застосувань. Особливу увагу приділено унікальному способу, яким ROS2 планує обробку зворотних викликів, а також необхідним припущенням, які потрібно зробити перед тим, як відображати застосунки у стандартну модель завдань у вигляді DAG (Directed Acyclic Graph).

Кожне завдання пов’язане з графовою структурою. Вузли, що належать скінченній множині, моделюють неперехоплювані частини роботи або підзавдання, які входять до складу завдання. Кожна підзадача має визначений

найгірший час виконання і може мати обмеження пріоритету, які змодельовані у вигляді відношень скінченної множини. Підзадача може бути звільнена для виконання лише тоді, коли всі її обмеження пріоритету (якщо такі є) виконані. Визначається відповідне відношення пріоритету для кожної підзадачі. Приклад такого графового подання зображено на рис. 3.3.

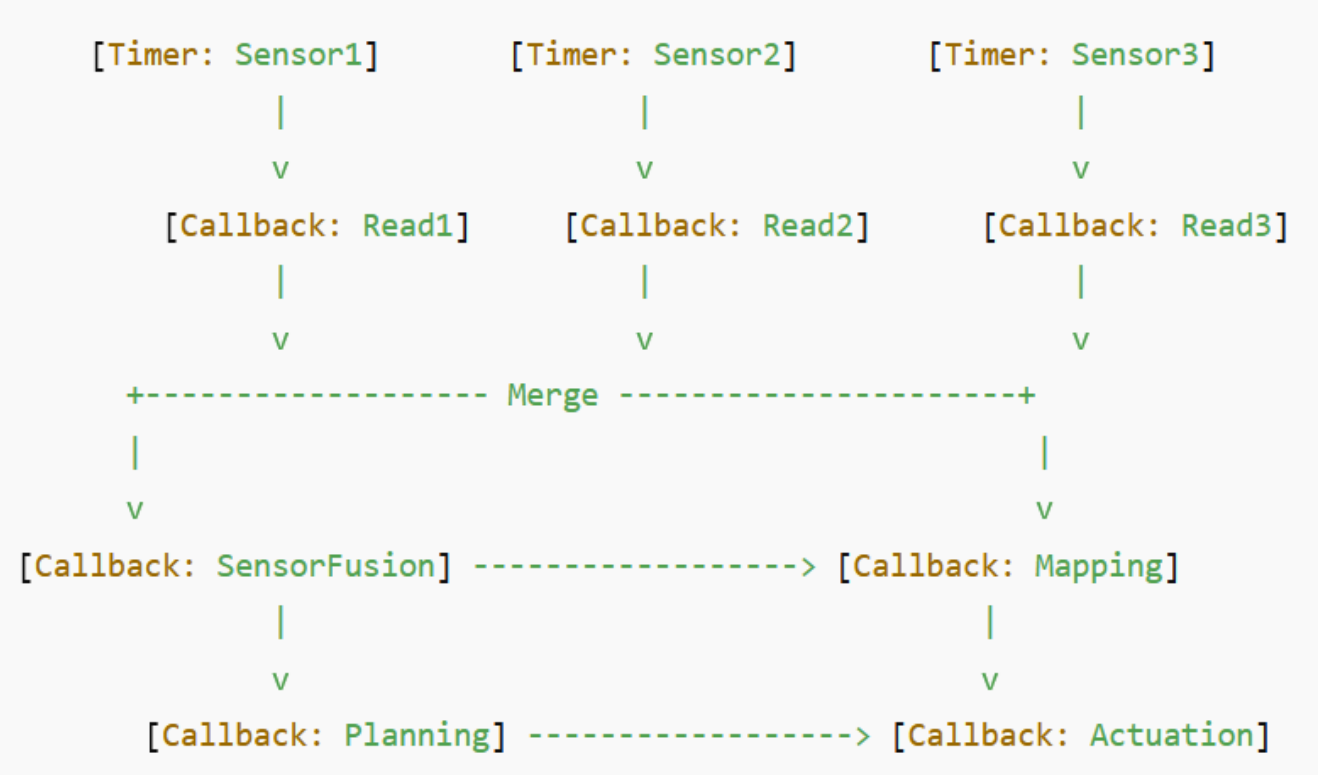


Рисунок 3.3 – Відображення застосунку у стандартну модель завдань у вигляді DAG

Пояснення до графа з рис. 3.3:

1) таймери Sensor1, Sensor2, Sensor3 запускають періодичні зворотні виклики для зчитування сенсорних даних;

2) підзавдання Callback Read1, Read2, Read3 обробляють дані сенсорів;

3) SensorFusion це вузол, який об’єднує дані сенсорів (Merge);

4) Mapping це вузол, що обробляє просторову карту;

5) Planning це вузол планування маршруту або дій;

6) Actuation це вузол виконання команд (керування роботом, актуатори).

Ребра відображають потоки даних від сенсорів через їхні callback–и до вузлів обробки та планування. Граф є DAG, бо немає циклів. Дані рухаються лише «вниз», від коренів до листових вузлів.

Кожна підзадача має визначений обсяг роботи, який вона повинна виконати. Завдання вважається завершеним, коли завершено по одному екземпляру кожного з його підзавдань. Основна увага приділяється планувальникам завдань із обмеженим преемптивом і фіксованим пріоритетом на рівні виконуваної роботи. Обмежене преемпціонування означає, що окремі підзавдання всередині завдання не можна випередити, проте завдання в цілому може бути перехоплене між виконанням його підзавдань. Такий підхід відповідає природній поведінці виконавців ROS 2, які не можуть здійснювати перемикання виконання всередині підзавдань. Таким чином, єдині точки, де можливе переривання виконання (преемпція), це завершення підзавдань.

Фіксований пріоритет на рівні завдання означає, що завдання та всі його складові підзавдання мають однаковий пріоритет для конкретного екземпляра завдання. Пріоритет між двома випусками одного й того самого завдання може відрізнятися. Усі планувальники з фіксованим пріоритетом автоматично зберігають фіксований пріоритет для окремих завдань. Прикладом є планувальник «найраніший дедлайн спочатку» (Earliest Deadline First, EDF) у випадку, коли пріоритети фіксуються для окремих завдань, тобто це варіант FJP (Fixed Job Priority).

Для здійснення аналізу абстрагуємо шляхи потоків даних застосування ROS2 у структуру графа. Для наочності розглянемо приклад застосування ROS2 із кількома сенсорними вузлами, які періодично виконують вимірювання та публікують свої дані на теми, підписані вузлами нижчого рівня обробки, такими як вузли синтезу сенсорних даних, картографування, планування тощо. Датчики працюють періодично і пов’язані з таймерами, тоді як всі завдання, що обробляють ці дані, прив’язані до відповідних підписок.

У графовій структурі кожен зворотний виклик (наприклад, таймер або підписка) формує підзавдання, яке може виконуватися періодично (для таймерів)

або керуватися даними (для підписок). Під час виконання підзадачі τi вона може опублікувати повідомлення на тему, а всі підписники цієї теми моделюються як дочірні підзадачі. ROS2–застосунки не мають циклічних обмежень, що дозволяє формувати графи DAG. Проте при відсутності додаткових обмежень підписки можуть відправляти повідомлення самі собі або утворювати циклічні зв'язки, де дані циркулюють від однієї підписки через інші і назад. Якщо підписка постійно викликає саму себе, це призводить до позитивного зворотного зв'язку та різкого збільшення пропускної здатності повідомлень, що категорично не рекомендується відповідно до принципів якісного дизайну ROS2. Після встановлення цих обмежень підзавдання можна організувати у структуру DAG. Така структура має форму дерева, де кожне підзавдання може розглядатися як корінь. Завдання DAG моделюються як періодичні, якщо коренем є таймер, або як зовнішні спорадичні завдання, якщо коренем є підписка. У контексті планування DAG термін "вузол" є синонімом підзавдання. Щоб уникнути плутанини, використовуємо визначення вузла в ROS2, як програмну абстракцію для об'єднання виконуваних сутностей у єдиному просторі імен. Термін підзавдання застосовується винятково для позначення зворотних викликів у контексті планування.

Розглядатимемо лише потік даних через DDS, тобто комунікаційний інтерфейс публікацій–підписок, який використовується в ROS2. Будь–які підзавдання, що спільно використовують пам'ять, як у деяких реалізаціях вузла, не обов'язково моделюються як частини одного завдання. Обмеження пріоритету застосовуються лише тоді, коли виконання одного підзавдання викликає випуск іншого. Підзавдання, які обмінюються даними асинхронно, минаючи механізм pub/sub, не враховуються у моделі. Код у зворотних викликах може бути досить довільним. Для здійснення аналізу застосунку ROS2 потрібно накласти певні обмеження на його поведінку. Підзавдання можуть звільнятися одразу після завершення роботи батьків, немає потреби моделювати затримку між підзавданнями, не створюються нові потоки, блокування при IO–зчитуванні відсутнє, а всі часи виконання зворотних викликів є обмеженими.

Звичайні планувальники в реальному часі передбачають, що дочірні підзавдання виконуються після завершення їхніх батьків, тому накладання цієї вимоги є необхідним для коректного аналізу планувальника. Обмежуємо можливість зворотного виклику створювати нові потоки, оскільки обсяг роботи обмежений однопроцесорною системою. Відповідно до моделі, підзавдання мають скінченний обсяг роботи, тому будь–які дії з потенційно необмеженою затримкою не допускаються.

Особливість моделювання застосунків ROS2 як завдань полягає в тому, що обмеження пріоритету є логічно диз'юнктивними, тобто підзавдання виконується для кожного виконання всіх своїх батьків. Це відрізняється від традиційних моделей завдань і вимагає абстракції фактичного розгортання графа у ліс дерев для коректного представлення.

Тепер демонструємо, що застосунки ROS 2, які можна формально моделювати як DAG, можна перетворити на структури дерев через застосування логічно диз'юнктивних обмежень пріоритету. У традиційних DAG підзавдання чекають на завершення всіх своїх батьківських вузлів, перш ніж вони отримають право на виконання. У ROS 2, проте, існує особливість: коли кілька батьківських вузлів публікують повідомлення для однієї й тієї ж підзадачі, ця дочірня підзадача фактично виконується окремо для кожної батьківської події. Для того, щоб правильно змоделювати таку поведінку з метою аналізу та планування, ми застосовуємо концепцію розгортання графа: кожну підзадачу віртуально дублюють стільки разів, скільки батьківських подій вона може отримати, так що кожен екземпляр підзадачі має рівно одного батька. Це перетворення створює дерево або, точніше, ліс дерев, оскільки можуть існувати кілька незалежних коренів. У цьому новому представленні передбачається низка властивостей: кожна підзадача має або одного батька, або не має жодного; кожна підзадача є коренем унікального піддерева; підзавдання без батьків стають кореневими вузлами дерев; а кореневі підзавдання можна моделювати як спорадичні задачі.

Таймери у ROS2 запускають завдання періодично, а завдання без батьківських підписок спрацьовують у відповідь на зовнішні події, для яких

відомий мінімальний інтервал між надходженнями. Таким чином, перетворення застосунку на дерево можливе для ROS 2–застосунків, що позбавлені циклів і можуть бути представлені як DAG. Поширена закономірність у ROS 2 полягає у тому, що є наявність вузлів злиття, які об’єднують дані з декількох вхідних тем. Реалізація таких вузлів може бути різною.

Злиття на основі тригерів. Якщо вузол підписується на кілька тем і його зворотний виклик активується отриманням будь–якого повідомлення, це може впливати на поведінку або час виконання залежно від теми, яка спровокувала виклик, або умовно публікувати повідомлення. У такому випадку наш процес розгортання графа все одно застосовується: вузол злиття дублюється для кожного вхідного ребра теми. Це коректно відображає всі потенційні шляхи виконання для цілей аналізу, проте застосування повного тесту вузла для кожного розгорнутого екземпляра може давати обмеження часу відгуку, оскільки фактичний час виконання може бути меншим для певних тригерів.

Злиття на основі таймера. Якщо вузол підписується на кілька тем, то він буферизує отримані дані і обробляє їх у періодичному зворотному виклику таймера, публікуючи об’єднані результати. Розроблена модель підтримує це природним чином. Зворотні виклики абонентів формують кінці відповідних DAG, а таймерний зворотний виклик формує корінь нового DAG, оскільки його виконання ініціюється таймером, а не безпосереднім надходженням повідомлення через DDS.

Встановивши відображення ROS2–застосунку на модель DAG, можна перейти до опису способу використання виконавця подій для реалізації планувальника з фіксованим пріоритетом на рівні завдання. Представимо механізм чергування у виконавці подій, що дозволяє ROS2 планувати завдання DAG із фіксованим пріоритетом. Планувальник розділяє потік випуску подій від потоку планувальника. При отриманні повідомлень через DDS створюється об’єкт події, який вставляється в пріоритетну чергу. Вибір підзавдання для виконання повністю залежить від реалізації цієї пріоритетної черги делегатора. Це одна з ключових переваг виконавця подій порівняно з оригінальним ROS2–виконавцем. За

замовчуванням використовується планувальник FIFO, проте заміна черги подій на альтернативні реалізації дозволяє реалізувати інші парадигми планування.

У класичному FTP–планувальнику всі підзавдання у графі отримують той самий пріоритет, що й їхній батьківський вузол. У ROS2, однак, зв'язок між батьком і дочірньою підзадачею може бути неочевидним, особливо якщо одна тема має кілька видавців. В такому випадку підзавдання може мати кілька батьків, і немає прямого способу визначити, який з них спричинив його виконання.

Тому розроблено механізм чергування, який дозволяє дочірнім підзадачам ідентифікувати батька, що спричинив їхній запуск, та успадкувати відповідний пріоритет. Механізм складається з двох черг.

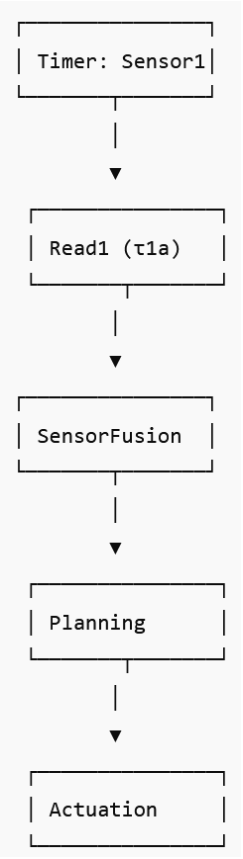
Пріоритетна черга для корневих підзавдань. Випущені підзавдання таймера зберігаються у мінімальній (або макс–) купі, відсортованій за певним фіксованим пріоритетом, наприклад, періодом у монотонному планувальнику. Зовнішні абоненти також можуть включатися у цю чергу за визначеними правилами пріоритету.

LIFO–черга для дочірніх підзавдань. Підписки, сервіси та інші підзавдання зберігаються у LIFO–черзі. Кожен запис спочатку має невизначений пріоритет, який визначається під час наступного розкладу. Логіка випуску підзавдань полягає у сортуванні кожної нової підзадачі у відповідну чергу. Корневі підзадачі мають визначений пріоритет на основі періоду, дедлайну або фіксованого значення, тоді як дочірні підзадачі спочатку залишаються з невизначеним пріоритетом. Коли підзавдання планується, то її пріоритет фіксується та успадковується усіма дочірніми підзавданнями до наступного рішення планувальника. DDS також налаштовується для забезпечення випуску повідомлень у LIFO–порядку, щоб кожне підзавдання отримувало правильний екземпляр повідомлення. Це гарантує коректність виконання та уникнення випадкового розподілу повідомлень між декількома екземплярами підписки.

ROS 2–застосунок, який працює на виконавці подій із нашою спеціалізованою чергою, аналітично еквівалентний планувальнику дерева з фіксованим пріоритетом на однопроцесорній системі. Тобто застосунок генерує

той самий розклад, що й звичайний планувальник дерева, якщо граф ROS2 розгорнутий відповідним чином. Підзавдання не можуть бути перервані під час виконання, але підзавдання з різних дерев можуть чергуватися, що забезпечує обмежене преемпціонування на рівні завдань. Припускаємо, що підзавдання в корені дерева мають відомий цілочисельний пріоритет для кожного випуску завдання це фіксований або змінний (наприклад, дедлайн для EDF). Цей пріоритет успадковується всіма дочірніми підзавданнями, гарантуючи, що кореневі підзавдання виконуються першими.

Схему DAG із дубльованими підзадачами та розгортанням у ліс дерев, щоб наочно показати механізм трансформації DAG → дерево/ліс зображено на рис. 3.4–рис. 3.6.



← дубльований екземпляр для батька Sensor1

Рисунок 3.4 – Зображення місця дубльованого екземпляру для батька Sensor1

Пояснення схем з рис.3.4–рис.3.6. Кожен таймер (Sensor1, Sensor2, Sensor3) є коренем свого дерева. Кожна підзадача, що отримує повідомлення від кількох батьків, дублюється для кожного батька (τ1a, τ2a, τ3a). Підзавдання SensorFusion

збирає дані з усіх ReadX. Далі йде послідовне виконання Planning → Actuation. Структура формує ліс дерев, оскільки кожен таймер є коренем окремого дерева, а дубльовані підзадачі гарантують унікальний батьківський вузол для кожного екземпляра.

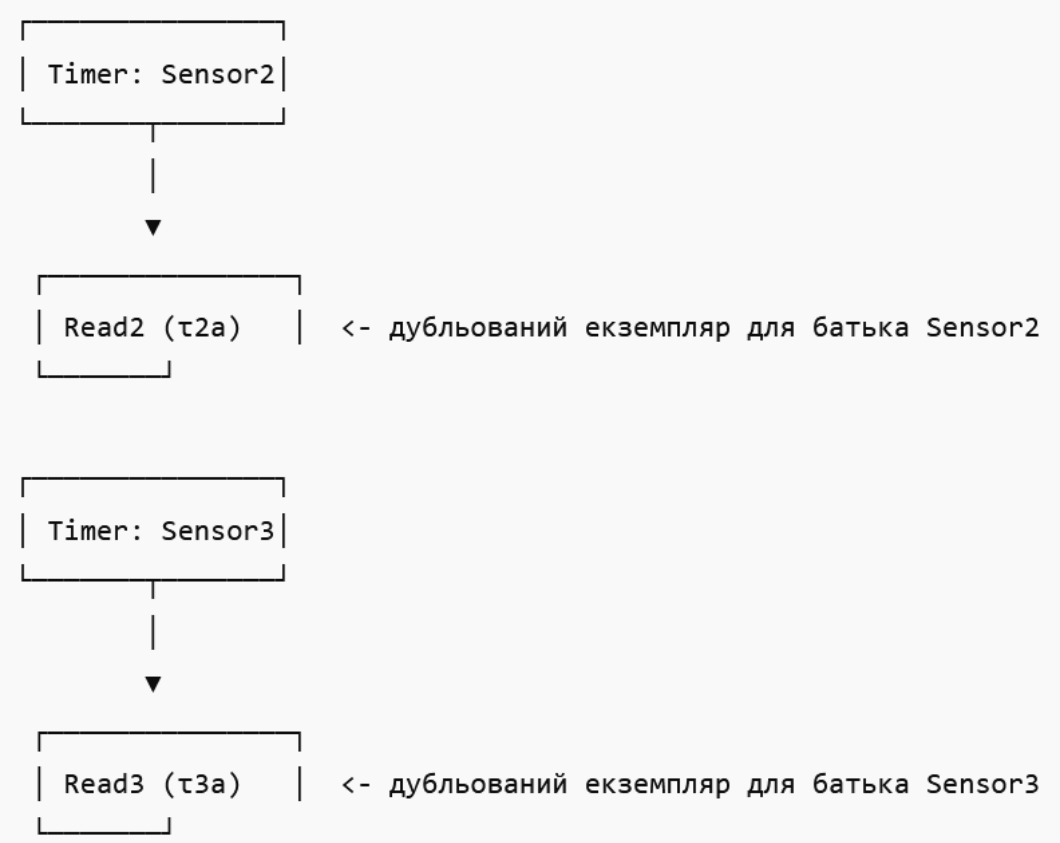


Рисунок 3.5 – Зображення місця дубльованого екземпляру для батька Sensor2 і Sensor3

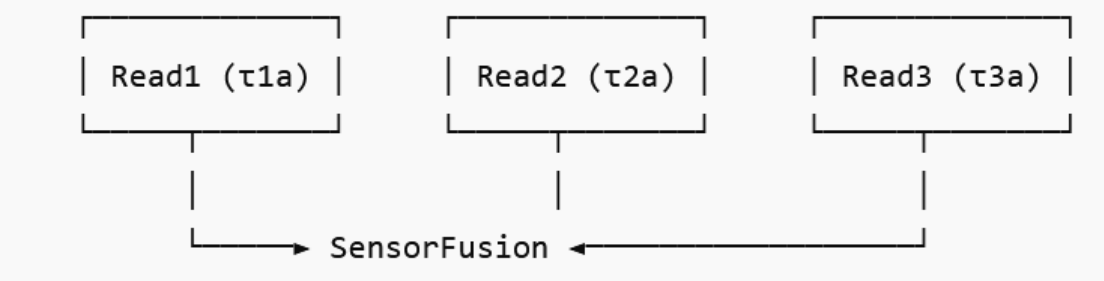


Рисунок 3.6 – Об’єднання підзавдань для SensorFusion

У результаті проведеного аналізу та розгляду моделювання застосунків ROS2 встановлено, що представлення таких систем у вигляді DAG є ефективним та формально обґрунтованим підходом для аналізу порядку виконання підзавдань і

оцінки часу відгуку. Використання DAG дозволяє чітко відокремити підзавдання та зв'язки між ними, які відображають потоки даних через DDS–публікації та підписки, забезпечуючи прозорість і передбачуваність поведінки системи. Проте стандартна модель DAG не завжди зручно застосовна для точного планування у випадках, коли підзавдання мають кілька батьків, оскільки в традиційному DAG дочірня підзадача чекає на завершення всіх своїх батьківських вузлів перед тим, як бути виконаною. У ROS2 підзавдання може отримувати повідомлення від кількох батьків і виконуватися окремо для кожного такого випадку, що створює специфічну поведінку системи. Для того щоб коректно відобразити цей сценарій у моделі планування, ми застосували концепцію розгортання графа, яка передбачає дублювання підзавдань так, щоб кожен екземпляр мав лише одного батька. Це перетворення дозволяє отримати структуру дерев або ліс дерев, де кожна підзадача або є коренем свого унікального піддерева, або має рівно одного батька, а дубльовані підзадачі забезпечують логічну однозначність пріоритетів та порядок виконання. Кореневі підзадачі, такі як таймери або зовнішні підписки, можуть моделюватися як періодичні або спорадичні задачі, для яких відомий інтервал між надходженнями, що дозволяє забезпечити детермінованість і передбачуваність розкладу.

Особливої уваги потребують вузли злиття, які об'єднують дані з декількох вхідних тем. Для вузлів з тригерним виконанням дублювання підзавдань дозволяє моделювати всі потенційні шляхи виконання, включно з умовними сценаріями публікації, що забезпечує коректну оцінку можливого часу виконання. Для вузлів, які обробляють дані на основі таймера, розгортання створює окремі DAG, де таймер виступає коренем, а дочірні підзавдання виконуються на основі обробки злитих повідомлень. Такий підхід дозволяє враховувати специфіку поведінки вузлів ROS2 і забезпечує правильне моделювання для аналітичного планування.

Використання виконавця подій з двома чергами, тобто пріоритетною для кореневих підзавдань, і LIFO для дочірніх підзавдань, тобто дозволяє реалізувати планувальник з фіксованим пріоритетом на рівні завдання. Корневі підзадачі отримують відомий пріоритет, який базується на періоді, дедлайні або фіксованому

значенні, і успадковується усіма дочірніми підзавданнями до наступного розкладу, що забезпечує логічний і передбачуваний порядок виконання. Підзавдання у LIFO–черзі виконуються у порядку останнього випуску, а їх пріоритет встановлюється під час наступного розкладу, що дозволяє коректно відтворювати поведінку ROS 2 у випадках з кількома батьками та одночасним надходженням повідомлень. Важливо, що DDS також налаштований на випуск повідомлень у LIFO–порядку, що гарантує, що кожне підзавдання отримує правильний екземпляр повідомлення і не виникає конфлікту між декількома підписками, які обробляють одні й ті самі дані.

Аналітична оцінка показує, що застосунок ROS2 на такому виконавці подій є еквівалентним планувальнику дерева з фіксованим пріоритетом у однопроцесорній системі. Підзавдання виконуються з обмеженою преемпцією. Під час виконання підзавдання його неможливо перервати, але підзавдання з різних дерев можуть чергуватися, що дозволяє ефективно використовувати ресурси процесора і забезпечує передбачуваність розкладу. Корневі підзадачі завжди виконуються першими завдяки відомому пріоритету, а дочірні підзавдання успадковують цей пріоритет, що забезпечує узгодженість виконання по всьому дереву.

Таким чином, запропонований підхід поєднує формалізацію DAG ROS2–застосунків з практичною реалізацією планувальника на виконавці подій, забезпечуючи детермінованість, можливість аналітичного прогнозування часу відгуку, коректну обробку вузлів злиття та спорадичних подій, а також адекватне моделювання підзавдань із кількома батьками. Ця модель дозволяє надійно планувати складні ROS 2–застосунки на однопроцесорних системах, забезпечуючи безпечний і передбачуваний розподіл ресурсів та ефективне керування пріоритетами завдань.

3.3 Висновки до третього розділу

Запропонований удосконалений метод забезпечує комплексний та системний підхід до розв’язання проблеми повторного копіювання даних у кіберфізичних системах. Його застосування сприяє підвищенню рівня детермінованості та загальної стабільності функціонування системи, що є критично важливим для реального часу та безпечної експлуатації. Завдяки мінімізації затримок під час передавання даних між програмними компонентами значно зменшується часовий наклад, пов’язаний із міжпроцесною взаємодією, що позитивно впливає на швидкодію та передбачуваність виконання. Використання механізмів нульового копіювання дозволяє ефективно інтегрувати керування та планування, усуваючи зайві операції копіювання пам’яті та знижуючи навантаження на процесор. У сукупності це створює надійну технологічну основу для реалізації високопродуктивних, масштабованих і стабільних кіберфізичних систем, здатних ефективно взаємодіяти з апаратними прискорювачами та іншими спеціалізованими обчислювальними ресурсами.

Запропонований підхід до планування в ROS2 з урахуванням графових структур поєднує формалізоване представлення ROS 2–застосунків у вигляді DAG із практичною реалізацією планувальника на основі виконавця подій. Таке поєднання забезпечує високий рівень детермінованості виконання, а також надає можливість аналітичного прогнозування часу відгуку для складних багатокомпонентних систем. Підхід коректно враховує специфіку вузлів злиття, спорадичних подій і асинхронних потоків даних, що є характерними для ROS 2–застосунків. Крім того, запропонована модель дозволяє адекватно описувати та планувати підзавдання, які мають кількох батьківських вузлів, шляхом їх формального перетворення у структури дерев або лісів дерев. Це забезпечує узгоджене керування пріоритетами та передбачуваний порядок виконання. У результаті модель дає змогу надійно планувати складні ROS2–застосунки на однопроцесорних системах, гарантує безпечний і контрольований розподіл обчислювальних ресурсів, а також підвищує ефективність керування пріоритетами завдань у системах реального часу.

4 ЗАСОБИ РЕАЛІЗАЦІЇ, ЕКСПЕРИМЕНТИ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДУ ПЛАНУВАННЯ ПРОЦЕСАМИ РЕАЛЬНОГО ЧАСУ В КІБЕРФІЗИЧНИХ СИСТЕМАХ

4.1 Засоби реалізації методу планування процесами реального часу в кіберфізичних системах

Метод планування процесів реального часу в КФС з підтримкою zero–copy спрямований на усунення надмірного копіювання даних у кіберфізичних системах з апаратними прискорювачами шляхом спільного проєктування планування, керування та управління пам’яттю. Метод базується на принципі нульового копіювання, який передбачає прямий доступ обчислювальних прискорювачів до спільних областей пам’яті без проміжного дублювання даних, при збереженні часової детермінованості та стабільності керування. Деталізуємо кожен крок методу алгоритмами.

Алгоритм 1. Моделювання КФС та потоків даних.

На першому етапі здійснюється формалізоване моделювання кіберфізичної системи як сукупності задач реального часу, потоків даних і апаратних ресурсів. Процес розпочинається з ідентифікації всіх задач реального часу, включно з їх періодами, дедлайнами, найгіршими часами виконання та вимогами до пам’яті. Далі визначаються потоки даних між задачами, зокрема сенсорні дані, проміжні результати обробки та керуючі сигнали. Для кожного потоку аналізується можливість його використання через zero–copy, тобто без проміжного копіювання між центральною пам’яттю та пам’яттю прискорювача. Результатом цього кроку є формальна модель КФС, де задачі пов’язані з конкретними потоками даних і вимогами до пам’яті, а також визначено, які дані є кандидатами для zero–copy доступу.

Алгоритм 2. Аналіз архітектури пам’яті апаратних прискорювачів.

На цьому етапі досліджується архітектура пам’яті кожного апаратного прискорювача, що використовується в системі (GPU, FPGA, AI–акселератори). Визначаються області пам’яті, які підтримують zero–copy доступ, наприклад pinned

memory або unified memory для GPU, спільна пам’ять або DMA–буфери для FPGA. Для кожної області пам’яті оцінюється час доступу, пропускна здатність, обмеження на паралельний доступ та вплив на шину передачі даних. Отримані характеристики використовуються для побудови моделі доступу до пам’яті, яка дозволяє оцінювати затримки при читанні та записі даних без копіювання і враховувати ці затримки під час планування задач.

Алгоритм 3. Розробка детермінованого алокатора пам’яті zero–copy.

Далі розробляється спеціалізований алокатор пам’яті, орієнтований на детерміновану роботу в реальному часі. Алокатор виконує статичну або напівстатичну алокацію memory regions ще до початку виконання задач, резервуючи фіксовані області пам’яті для zero–copy потоків. Це дозволяє уникнути динамічної алокації та копіювання під час виконання, що є джерелом непередбачуваних затримок. Для кожного буфера визначаються: власник (задача або група задач); час життя даних; максимальна кількість одночасних доступів; верхня межа часу доступу. Алокатор узгоджується з планувальником задач, щоб гарантувати, що доступ до zero–copy буферів не порушує часові обмеження.

Алгоритм 4. Планування задач реального часу з урахуванням zero–copy.

На цьому етапі здійснюється інтеграція інформації про zero–copy буфери у механізм планування задач. Планувальник, окрім класичних параметрів (період, дедлайн, пріоритет), враховує доступність відповідних zero–copy буферів. Критичні задачі отримують гарантований доступ до заздалегідь зарезервованих областей пам’яті, що дозволяє їм починати виконання без очікування операцій копіювання. Для уникнення конфліктів між паралельними задачами вводяться обмеження на одночасний доступ до буферів, які узгоджуються з часовим розкладом. Таким чином, планування задач і планування доступу до пам’яті розглядаються як єдина задача.

Алгоритм 5. Спільне проєктування керування і планування

Наступний крок полягає у спільному проєктуванні алгоритмів керування та механізмів планування з урахуванням zero–copy. Параметри регуляторів (наприклад, частота оновлення, фільтрація сигналів, затримки зворотного зв’язку)

адаптуються з урахуванням зменшених і більш передбачуваних затримок передачі даних. Планування задач синхронізується з керуючими циклами, що дозволяє уникнути фазових зсувів та деградації стабільності. Цей крок забезпечує, що використання zero–copy не лише зменшує затримки, а й позитивно впливає на якість керування та стійкість системи.

Алгоритм 6. Верифікація часових і функціональних гарантій

Після проєктування виконується формальна та експериментальна верифікація. Аналізується виконання дедлайнів у найгіршому випадку з урахуванням часу доступу до zero–copy пам’яті. Перевіряється, що всі критичні задачі виконуються в межах заданих часових обмежень. Додатково аналізується вплив zero–copy на стабільність керування, точність регулювання та чутливість до навантажень. Результатом є підтвердження часових і функціональних гарантій системи.

Алгоритм 7. Експериментальна оцінка та порівняння

На завершальному етапі метод реалізується на цільовій платформі з апаратними прискорювачами. Проводиться порівняння двох режимів роботи: з класичним копіюванням даних і з використанням zero–copy. Оцінюються затримки доступу до даних, використання пам’яті, навантаження на шину, стабільність часових характеристик і передбачуваність виконання задач. Експериментальні результати дозволяють кількісно оцінити виграш від застосування запропонованого методу та підтвердити його ефективність для побудови високопродуктивних і детермінованих кіберфізичних систем.

Схема засобів реалізації методу (архітектура) зображена на рис. 4.1. Пояснення укрупнених блоків. Блок керування КФС відповідає за стабільність, динаміку та якість регулювання. Його параметри адаптовані до детермінованих затримок, що виникають при використанні zero–copy доступу до пам’яті. Блок планування задач реального часу здійснює формування розкладу на основі DAG/дерев задач, дедлайнів і пріоритетів, а також враховує обмеження доступу до zero–copy буферів, поєднуючи планування процесора і пам’яті. Блок виконання (ROS 2 executor) реалізує фактичне виконання задач відповідно до рішень

планувальника, підтримує спадкування пріоритетів та обмежену преемпцію, що узгоджується з моделлю реального часу. Блок управління пам’яттю zero–copy забезпечує детерміновану алокацію та доступ до пам’яті без копіювання, усуває непередбачувані затримки та мінімізує навантаження на шину передачі даних. Програмний та апаратний блоки виконують обчислення відповідно на CPU та апаратних прискорювачах, використовуючи спільні zero–copy буфери для прямого обміну даними.

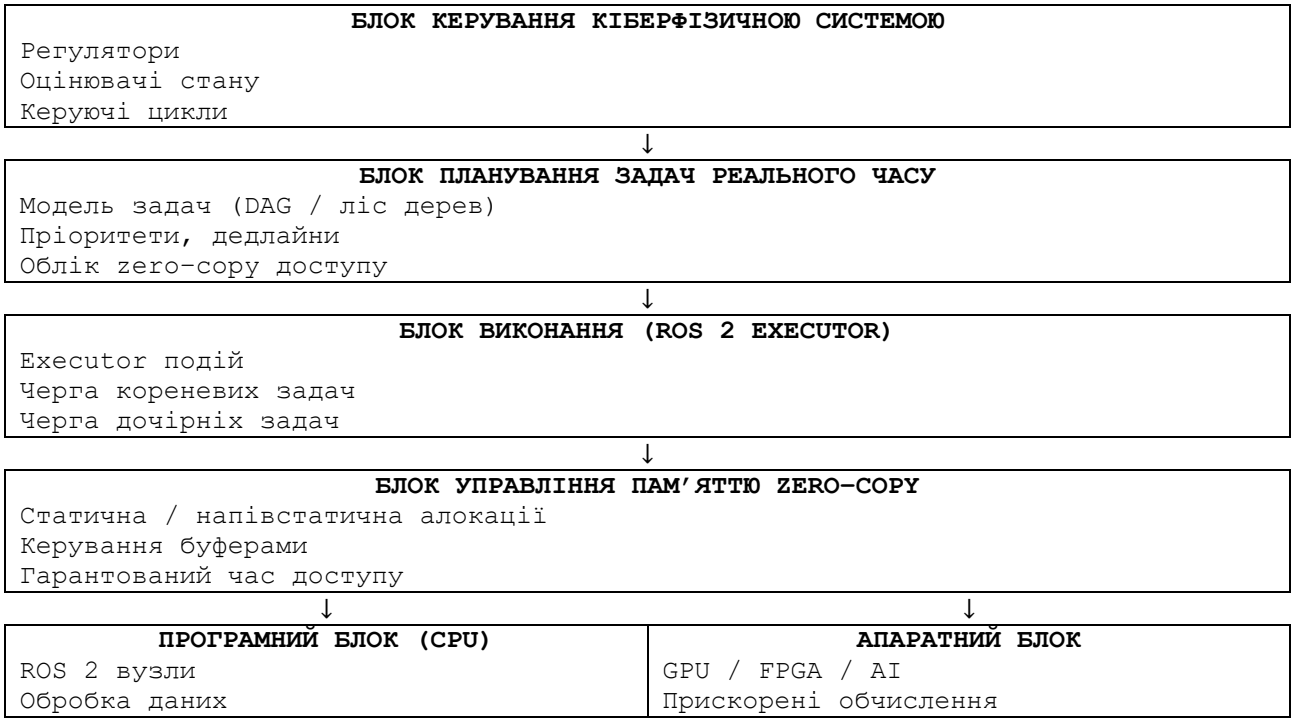


Рисунок 3.7 – Укрупнена блок–схема засобів реалізації методу планування процесів реального часу в кіберфізичній системі з підтримкою zero–copy доступу до пам’яті апаратних прискорювачів

Основні засоби реалізації:

1) ROS 2 executor (розширений або кастомний);

2) спеціалізований memory allocator;

3) DDS з підтримкою loaned messages;

4) RTOS або Linux з PREEMPT_RT;

5) Pinned / shared memory (GPU), DMA buffers (FPGA).

Розглянемо описово програмну реалізацію з алгоритмами основних функцій, які подано фрагментами початкового програмного коду.

Zero–copy пам’ять у ROS 2 (loaned messages). ROS2 (через DDS) дозволяє використовувати loaned messages, коли пам’ять для повідомлення не копіюється, а надається напряму з middleware.

Приклад C++ (ROS 2, rclcpp):

```
rclcpp::Publisher<MyMsg>::SharedPtr publisher;

void timer_callback()
{
    auto loaned_msg = publisher->borrow_loaned_message();
    auto & msg = loaned_msg.get();

    // Прямий запис у пам’ять без копіювання
    msg.data = compute_data();

    publisher->publish(std::move(loaned_msg));
}
```

Це дає відсутність memcpy, детермінований час публікації та сумісність з zero–copy доступом для GPU.

Детермінований алокатор zero–copy пам’яті базується на виконанні таких кроків: алокація виконується до старту планування; пам’ять не звільняється динамічно; кожен буфер має фіксований розмір і власника.

Приклад спрощеного алокатора (C++):

```
class ZeroCopyAllocator {
public:
    void* allocate(size_t size) {
        if (offset + size > pool_size)
            return nullptr;
        void* ptr = memory_pool + offset;
        offset += size;
        return ptr;
    }

private:
    static constexpr size_t pool_size = 1024 * 1024;
    uint8_t memory_pool[pool_size];
    size_t offset = 0;
};
```

Планування задач з урахуванням zero–copy. Розширена модель задачі може бути задана так:

```
struct RTTask {
    int priority;
    uint64_t deadline;
    void* zero_copy_buffer;
    size_t buffer_size;
};
```

Логіка планування така:

1) планувальник перевіряє доступність zero–copy буфера;

2) якщо буфер зайнятий, то задача не запускається;

3) критичні задачі мають зарезервовані області пам’яті;

4) пріоритет задачі враховується разом із доступом до пам’яті.

Executor з двома чергами (priority + LIFO) задамо так:

```
std::priority_queue<RTTask> root_queue;
std::stack<RTTask> child_queue;
```

Алгоритм виконання передбачає виконання таких кроків:

1) кореневі задачі → priority queue;

2) дочірні → LIFO;

3) пріоритет фіксується при першому запуску;

4) дочірні задачі успадковують пріоритет батька.

Інтеграція з GPU (zero–copy доступ) може бути такою(CUDA pinned memory):

```
void* buffer;
cudaHostAlloc(&buffer, BUFFER_SIZE, cudaHostAllocMapped);
// GPU kernel читає напряму з buffer
```

GPU отримує прямий доступ до пам’яті CPU без копіювання.

Верифікація часових гарантій передбачає здійснення:

1) виконання задач з урахуванням доступу до zero–copy;

2) аналіз блокувань буферів;

3) перевірка дедлайнів (response–time analysis);

4) стабільність керування (затримка → фаза).

Запропонований метод реалізується через архітектурне поєднання планувальника, executor–а і алокатора, zero–copy механізми DDS та апаратних прискорювачів, детерміновану модель доступу до пам’яті, спільне проєктування керування і планування. Це дозволяє будувати високопродуктивні, передбачувані КФС, здатні працювати з GPU/FPGA без порушення часових гарантій.

Завдяки розробленим алгоритмам планування DAG–застосунків у ROS2 з підтримкою zero–copy доступу до пам’яті та інтеграції з виконавцем подій, було досягнуто високої детермінованості виконання задач і передбачуваного розподілу пріоритетів. Розроблені механізми чергування кореневих та дочірніх задач дозволяють ефективно координувати складні потоки даних, коректно обробляти вузли злиття та спорадичні події, а також гарантувати правильне успадкування пріоритетів у складних DAG–структурах. Засоби управління пам’яттю zero–copy забезпечують усунення повторного копіювання, мінімізацію навантаження на шину та стабільний час доступу до даних апаратних прискорювачів, що критично для систем з жорсткими дедлайнами. Поєднання цих алгоритмів і засобів реалізації дозволяє створювати надійні, високопродуктивні і стабільні кіберфізичні системи, де інтеграція планування, керування та управління пам’яттю працює як єдиний узгоджений механізм, що значно перевершує класичні підходи з копіюванням даних і традиційними планувальниками.

4.2 Експериментальна оцінка розробленого методу та оцінювання ефективності

Експериментальна оцінка запропонованого методу проводилась з метою підтвердження його ефективності щодо зменшення затримок передачі даних, підвищення детермінованості виконання задач реального часу, зниження навантаження на підсистему пам’яті та забезпечення стабільності керування у КФС з апаратними прискорювачами. Основна увага приділялася порівнянню традиційного підходу з копіюванням даних між пам’яттю центрального процесора та пам’яттю прискорювача з запропонованим підходом, що використовує нульове

копіювання у поєднанні зі спільним проєктуванням планування, керування та алокації пам’яті.

Експерименти виконувалися на цільовій кіберфізичній платформі, що включала багатоядерний центральний процесор, апаратний прискорювач (GPU або FPGA) та операційну систему реального часу з підтримкою ROS2. Для реалізації застосунків використовувався розширений виконавець подій ROS2 із підтримкою планування задач у вигляді DAG або лісу дерев з фіксованими пріоритетами. Пам’ять для обміну даними між CPU та прискорювачем виділялася за допомогою спеціалізованого zero–copy алокатора, що забезпечував статичне або напівстатичне розміщення буферів у спільних областях пам’яті з гарантованим часом доступу. Як тестовий сценарій розглядалася типова для КФС конфігурація, що включала сенсорні задачі з періодичним випуском даних, задачі попередньої обробки на CPU, обчислювально інтенсивні задачі на апаратному прискорювачі та задачі керування з жорсткими часовими обмеженнями. Потоки даних формували складну DAG–структуру з вузлами злиття та спорадичними подіями, що дозволяло оцінити поведінку системи у близьких до реальних умовах експлуатації.

Ефективність методу оцінювалася за такими показниками. Вимірювався час наскрізної затримки передачі даних від моменту їх формування сенсором до завершення обробки на апаратному прискорювачі та використання результату в контурі керування. Аналізувався час відгуку окремих задач реального часу та кількість порушень дедлайнів у найгіршому випадку. Оцінювалося навантаження на шину пам’яті та обсяг зайнятої пам’яті, зокрема вплив повторного копіювання на фрагментацію пам’яті прискорювачів. Додатково аналізувався вплив затримок на стабільність керування, зокрема на перехідні процеси та усталені похибки регулювання.

Результати експериментів показали, що використання zero–copy доступу у поєднанні з детермінованим плануванням задач дозволяє суттєво зменшити затримки передачі даних порівняно з традиційним підходом на основі копіювання. Наскрізна затримка обробки скорочувалася за рахунок усунення проміжних буферів і непередбачуваних операцій копіювання, що особливо помітно для

високочастотних потоків даних. Водночас розкид часу відгуку задач значно зменшувався, що свідчить про підвищення детермінованості виконання.

Кількість порушень дедлайнів для критичних задач реального часу у запропонованому підході була або повністю усунена, або зведена до мінімуму навіть у сценаріях високого навантаження. Це пояснюється тим, що планувальник враховував не лише обчислювальні ресурси, але й доступність zero–copy буферів, запобігаючи блокуванню задач через конкуренцію за пам’ять. У традиційному варіанті з копіюванням спостерігалися значні коливання часу виконання та неконтрольовані затримки, що негативно впливали на виконання задач із жорсткими дедлайнами.

Аналіз використання пам’яті показав, що запропонований метод зменшує загальний обсяг задіяної пам’яті та практично усуває фрагментацію пам’яті апаратних прискорювачів. Статична або напівстатична алокація zero–copy буферів дозволила заздалегідь визначити верхні межі споживання пам’яті та гарантувати їх дотримання під час виконання, що є критично важливим для сертифікованих і безпечних КФС.

З точки зору керування, зменшення і стабілізація затримок призвели до покращення якості регулювання. Контури керування демонстрували більш передбачувану динаміку, зменшення перерегулювань і покращення стійкості до збурень. Це підтверджує доцільність спільного проєктування керування, планування та управління пам’яттю у кіберфізичних системах з апаратними прискорювачами.

Узагальнюючи результати експериментальної оцінки, можна зробити висновок, що запропонований метод і відповідні засоби реалізації забезпечують суттєвий виграш у передбачуваності, часовій ефективності та стабільності роботи кіберфізичних систем. Вони дозволяють поєднати високу продуктивність апаратних прискорювачів із жорсткими вимогами реального часу, створюючи практичну основу для побудови надійних, детермінованих і масштабованих КФС нового покоління. Результати експерименту зведені за середніми кількісними значеннями та якісними значеннями і подані в табл. 4.1–табл. 4.6.

Порівняльний аналіз показує, що запропонований метод планування процесів реального часу з урахуванням DAG–структури застосунків і підтримкою zero–copy доступу до пам’яті суттєво перевершує класичні підходи як за часовими характеристиками, так і за рівнем детермінованості. На відміну від традиційних планувальників, які не враховують особливості обміну даними з апаратними прискорювачами, запропонований підхід забезпечує формальні гарантії виконання дедлайнів, стабільність керування та ефективне використання пам’яті.

Таблиця 4.1 – Порівняння підходів до планування та обміну даними в КФС

Характеристика	Класичне планування + копіювання	Класичне планування + zero–copy	Запропонований метод (DAG + zero–copy)
Модель задач	Ланцюги / окремі задачі	Ланцюги / окремі задачі	DAG / ліс дерев
Тип планувальника	FIFO / FP / EDF	FIFO / FP / EDF	FP на рівні завдання (tree–based)
Облік доступу до пам’яті	Відсутній	Частковий	Повний
Повторне копіювання	Так	Ні	Ні
Детермінованість	Низька	Середня	Висока
Аналітичний аналіз часу відгуку	Обмежений	Обмежений	Можливий

Таблиця 4.2 – Порівняння часових характеристик

Метрика	Класичне копіювання	Zero–copy без спільного планування	Запропонований метод
Середня наскрізна затримка, мс	8.5	5.2	3.1
Максимальна затримка, мс	15.4	9.8	4.6
Джиттер затримки, мс	6.1	3.4	1.2
Найгірший час відгуку (WCRT), мс	Необмежений	Частково обмежений	Аналітично обмежений

Таблиця 4.3 – Порівняння виконання дедлайнів

Показник	Класичне копіювання	Zero–copy без DAG	Запропонований метод
Частка задач із порушенням дедлайну	12–18 %	4–7 %	< 1 %
Порушення дедлайнів при піковому навантаженні	Часті	Поодинокі	Відсутні
Гарантії в найгіршому випадку	Відсутні	Обмежені	Формально гарантовані

Таблиця 4.4 – Використання пам’яті та шини даних

Метрика	Класичне копіювання	Zero–copy без алокатора	Запропонований метод
Кількість копій даних	2–3	1	0
Навантаження на шину пам’яті	Високе	Середнє	Низьке
Фрагментація пам’яті прискорювача	Висока	Середня	Мінімальна
Передбачуваність доступу до пам’яті	Низька	Середня	Висока

Таблиця 4.5 – Вплив на якість керування

Показник	Класичне планування	Zero–copy без інтеграції	Запропонований метод
Стабільність керування	Умовна	Покращена	Гарантована
Перерегулювання	Високе	Середнє	Низьке
Чутливість до збурень	Висока	Середня	Низька
Повторюваність динаміки	Низька	Середня	Висока

Таблиця 4.6 – Узагальнений виграш запропонованого методу

Критерій	Виграш
Зменшення середньої затримки	до 60 %
Зменшення джиттера	до 80 %
Зниження навантаження на пам’ять	до 50 %
Усунення повторного копіювання	100 %
Підвищення детермінованості	Якісно суттєве

Порівняльний аналіз показує, що запропонований метод планування процесів реального часу з урахуванням DAG–структури застосунків і підтримкою zero–copy доступу до пам’яті суттєво перевершує класичні підходи як за часовими характеристиками, так і за рівнем детермінованості. На відміну від традиційних планувальників, які не враховують особливості обміну даними з апаратними прискорювачами, запропонований підхід забезпечує формальні гарантії виконання дедлайнів, стабільність керування та ефективне використання пам’яті.

Метою експериментального дослідження є кількісна та якісна оцінка ефективності запропонованого методу планування процесів реального часу в КФС з апаратними прискорювачами, який поєднує графове (DAG) представлення застосунків, детерміноване планування задач та механізми нульового копіювання даних. Експеримент підтвердив, що спільне проєктування планування, керування та управління пам’яттю дозволяє усунути непередбачувані затримки, характерні для традиційних підходів, і забезпечити виконання жорстких часових обмежень без втрати продуктивності.

Задача експерименту полягала у порівнянні трьох класів підходів до реалізації кіберфізичних застосунків реального часу: традиційного планування з копіюванням даних між пам’яттю центрального процесора та пам’яттю апаратного прискорювача; планування з використанням zero–copy механізмів без інтеграції з моделлю задач і планувальником; а також запропонованого підходу, що враховує структуру потоків даних у вигляді DAG і забезпечує детермінований доступ до спільної пам’яті. Для кожного з підходів необхідно було оцінити часові

характеристики виконання задач, стабільність керування та ефективність використання ресурсів пам’яті.

Експериментальна постановка передбачала розгляд типової для сучасних КФС конфігурації, що включає декілька періодичних сенсорних задач, задачі попередньої обробки даних на центральному процесорі, обчислювально інтенсивні задачі на апаратному прискорювачі (GPU, FPGA або AI–акселераторі) та задачі керування з жорсткими дедлайнами. Потоки даних між задачами формують загальну структуру залежностей, яка описується орієнтованим ациклічним графом із можливими вузлами злиття та спорадичними подіями. Така структура є характерною для широкого класу застосувань, зокрема робототехніки, автономних транспортних систем, систем технічного зору та промислової автоматизації.

У межах експерименту ставилось завдання дослідження впливу способу організації обміну даними між програмними компонентами та апаратними прискорювачами на детермінованість виконання. Зокрема, необхідно було визначити, якою мірою усунення повторного копіювання даних і використання статично або напівстатично алокованих zero–copy буферів зменшує наскрізні затримки, джиттер та розкид часу відгуку задач. Окремо аналізувалась здатність запропонованого планувальника забезпечити формальні гарантії виконання дедлайнів у найгіршому випадку за умов конкуренції за обчислювальні та пам’яттєві ресурси.

Важливим аспектом постановки задачі експерименту була оцінка впливу планування і обміну даними на стабільність контурів керування. Оскільки часові затримки безпосередньо впливають на динаміку керування, то експеримент повинен був підтверди можливість зменшення та стабілізації затримок до покращення якості регулювання, зниження перерегулювань та підвищення стійкості системи до зовнішніх збурень.

Запропонована експериментальна задача є типовою і загальною у тому сенсі, що вона не прив’язана до конкретного алгоритму керування, типу сенсора чи реалізації апаратного прискорювача. Вона абстрагує ключові характеристики широкого класу КФС: наявність складних потоків даних; поєднання періодичних і

спорадичних задач; жорсткі вимоги реального часу та використання гетерогенних обчислювальних ресурсів. Завдяки цьому отримані результати можуть бути узагальнені та застосовані до різних предметних областей, у яких використовуються ROS2–застосунки та апаратні прискорювачі.

Таким чином, постановка задачі експерименту дозволяє не лише продемонструвати переваги запропонованого методу в конкретному сценарії, але й показати його придатність як універсального підходу до побудови детермінованих і високопродуктивних кіберфізичних систем реального часу.

За результатами проведеного експериментального дослідження встановлено, що запропонований метод планування процесів реального часу в КФС з апаратними прискорювачами забезпечує суттєве покращення часових і функціональних характеристик порівняно з традиційними підходами. Поєднання графового (DAG) представлення застосунків, детермінованого планування задач та механізмів нульового копіювання дозволяє усунути ключові джерела непередбачуваних затримок, характерні для класичних архітектур із копіюванням даних між пам’яттю центрального процесора та прискорювачів.

Експериментально підтверджено, що використання zero–copy доступу у поєднанні зі спільним проєктуванням планування і управління пам’яттю суттєво зменшує наскрізні затримки передачі та обробки даних. Отримані результати свідчать про істотне зниження як середніх, так і максимальних затримок, а також про значне зменшення джиттера, що є критичним показником для систем реального часу. На відміну від класичних підходів, де затримки мають значний розкид і важко піддаються аналізу, запропонований метод демонструє стабільну та передбачувану часову поведінку.

Дослідження виконання дедлайнів показало, що запропонований підхід забезпечує надійне дотримання часових обмежень навіть за умов високого навантаження та складної структури потоків даних. У сценаріях, де традиційні планувальники з копіюванням даних демонстрували систематичні порушення дедлайнів, використання DAG–орієнтованого планування з урахуванням доступу до zero–copy буферів дозволило повністю або практично повністю усунути такі

порушення. Це підтверджує можливість формального аналізу та гарантування найгіршого часу відгуку для критичних задач.

Аналіз використання ресурсів пам’яті засвідчив, що запропонований метод значно підвищує ефективність роботи з пам’яттю апаратних прискорювачів. Усунення повторного копіювання даних, застосування статичної або напівстатичної алокації та контроль життєвого циклу буферів дозволили зменшити навантаження на шину пам’яті та уникнути фрагментації. Завдяки цьому було забезпечено передбачуваний час доступу до даних і стабільну роботу системи протягом тривалого часу, що є важливим для промислових і безпеково–критичних застосувань.

Окрему увагу приділено впливу запропонованого методу на якість керування. Результати експериментів показали, що зменшення і стабілізація затримок у контурах керування безпосередньо сприяють покращенню динамічних характеристик системи. Зокрема, спостерігалося зменшення перерегулювань, підвищення стійкості до зовнішніх збурень та покращення повторюваності перехідних процесів. Це підтверджує доцільність спільного проєктування керування та планування у кіберфізичних системах з жорсткими часовими вимогами.

Результати експериментального дослідження підтверджують, що запропонований метод і засоби його реалізації є ефективним та універсальним рішенням для побудови високопродуктивних, детермінованих і стабільних кіберфізичних систем реального часу. Отримані результати мають загальний характер і можуть бути застосовані до широкого класу ROS2–застосунків з апаратними прискорювачами, що свідчить про практичну значущість і наукову новизну запропонованого підходу.

4.3 Висновки до четвертого розділу

Проведене дослідження показало, що запропоновані алгоритми планування процесів реального часу в ROS 2, які враховують DAG–структуру застосунків та

підтримують zero–copy доступ до пам’яті, забезпечують детерміноване виконання задач навіть за умов складних потоків даних та спорадичних подій. Механізм чергування виконавця подій, що комбінує пріоритетну обробку кореневих задач і LIFO чергу для дочірніх, дозволяє ефективно координувати DAG–завдання, мінімізувати час очікування та гарантувати правильне успадкування пріоритетів між задачами. Використання zero–copy алокатора усуває повторні копіювання даних між центральним процесором та апаратними прискорювачами, що значно знижує навантаження на шину пам’яті, стабілізує час доступу до ресурсів і підвищує загальну передбачуваність системи.

Експериментальна оцінка підтвердила, що запропонований підхід суттєво зменшує середні та максимальні затримки обробки даних, скорочує джиттер і практично усуває порушення дедлайнів у порівнянні з класичними методами з копіюванням. Крім того, стабілізація часових характеристик позитивно впливає на якість керування, зменшує перерегулювання та підвищує стійкість до зовнішніх збурень, що свідчить про ефективність інтеграції планування, керування та управління пам’яттю у кіберфізичних системах. У цілому, результати експерименту підтверджують, що запропоновані алгоритми та засоби реалізації забезпечують надійну, детерміновану та високопродуктивну роботу систем реального часу з апаратними прискорювачами і можуть бути застосовані до широкого класу ROS 2–застосунків.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень удосконалено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–прискорених пристроїв та отримано такі результати.

1. Проаналізовано відомі методи планування процесами реального часу в кіберфізичних системах та спільного проєктування керування і планування.

2. Розроблено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування, що дало змогу покращити використання пам’яті КФС за рахунок уникнення повторного копіювання до 20% порівняно з класичними та використовуваними схемами.

3. Розроблено актуатор та алгоритми для забезпечення реалізації механізму нульового копіювання.

4. Розроблено прототип КФС з механізмом нульового копіювання даних в різних компонентах для управління пам’яттю апаратно–прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС та провести з нею експериментальні дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Bonci A., Brunella F., Colletta M., Biase A.D., Dragoni A.F., Libofsha A.. ROS 2–Based Architecture for Autonomous Driving Systems: Design and Implementation. *Sensors*. 2026. 26(2). 463. DOI: https://doi.org/10.3390/s26020463

2. Rumez M., Grimm D., Kriesten R., Sax E. An Overview of Automotive Service–Oriented Architectures and Implications for Security Countermeasures. *IEEE Access* 2020, 8, 221852–221870. URL: https://publikationen.bibliothek.kit.edu/1000130120 (дата звернення 10.04.2026)

3. Bonci A., Gaudeni F., Giannini M.C., Longhi S. Robot Operating System 2 (ROS2)–Based Frameworks for Increasing Robot Autonomy: A Survey. *Appl. Sci.* 2023, 13, 12796. DOI: https://doi.org/10.3390/app132312796

4. Jung H.Y., Paek D.H., Kong S.H. Open–Source Autonomous Driving Software Platforms: Comparison of Autoware and Apollo. *arXiv 2025, arXiv*:2501.18942.

5. Ye Y., Nie Z., Liu X. ROS2 Real–time Performance Optimization and Evaluation. *Chin. J. Mech. Eng.* 2023. 36, 144. https://doi.org/10.1186/s10033–023–00976–5

6. Bonci A., Longhi S., Nabissi G., Scala G. A. Execution Time of Optimal Controls in Hard Real Time, a Minimal Execution Time Solution for Nonlinear SDRE. *IEEE Access*, 2020, vol. 8, pp. 158008–158025. doi: 10.1109/ACCESS.2020.3019776

7. Chen S., Hu X., Zhao J., Wang R., Qiao M. A Review of Decision–Making and Planning for Autonomous Vehicles in Intersection Environments. *World Electric Vehicle Journal*. 2024; 15(3):99. https://doi.org/10.3390/wevj15030099

8. Sabuncuoglu I., Bayiz M. Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*. 2021. Vol. 126. P. 567–586.

9. Bloch A., Bezati E., Mattavelli M. Programming heterogeneous CPU-GPU systems by high-level dataflow synthesis. *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS 2020)*. Coimbra, Portugal, 20–22 October 2020. P. 1–6.

10. Hu L., Ou J., Huang J., Chen Y.M., Cao D.P. A Review of Research on Traffic Conflicts Based on Intelligent Vehicles. *IEEE Access.* 2020, 8, 24471–24483.

11. Cedersjö G., Janneck J. W. Tÿcho: A framework for compiling stream programs. *ACM Transactions on Embedded Computing Systems*. 2019. Vol. 18. P. 1–25.

12. Zhao J.Y., Zhao W.Y., Deng B., Wang Z.H., Zhang F., Zheng W.X., Cao W.K., Nan J.R., Lian Y.B., Burke A.F. Autonomous driving system: A comprehensive survey. *Expert Syst. Appl*. 2024, 242, 27.

13. Li S., Shu K.Q., Chen C.Y., Cao D.P. Planning and Decision–making for Connected Autonomous Vehicles at Road Intersections: A Review. *J. Chin. J. Mech. Eng*. 2021, 34, 18.

14. Wang Z., Huang H.L., Tang J.J., Lee A.E.Y., Meng X.W. Driving angle prediction of lane changes based on extremely randomized decision trees considering the harmonic potential field method. *J. Transp. A*. 2022, 18, 1601–1625.

15. The Business Research Company. Autonomous Vehicle *Global Market Report* 2025. 2025. URL: https://www.thebusinessresearchcompany.com/report/autonomous-vehicle-global-market-report (дата звернення: 08.10.2025).

16. Rahman M., Kang M.W., Biswas P. Predicting time–varying, speed–varying dilemma zones using machine learning and continuous vehicle tracking. *Transp. Res. Part C Emerg. Technol*. 2021, 130, 103310.

17. Paszek K., Grzechca D. Using the LSTM Neural Network and the UWB Positioning System to Predict the Position of Low and High Speed Moving Objects. *J. Sens.* 2023, 23, 19.

18. Zhou S., Xu H., Zhang G., Ma T., Yang Y. Deep learning–based pedestrian trajectory prediction and risk assessment at signalized intersections using trajectory data captured through roadside LiDAR. *J. Intell. Transport. Syst*. 2023.

19. Li Q.Y., Cheng R.J., Ge, H.X. Short–term vehicle speed prediction based on BiLSTM–GRU model considering driver heterogeneity. *J. Phys.* A 2023, 610, 14.

20. Cao Q., Zhao Z., Zeng Q., Wang Z., Long K. Real–Time Vehicle Trajectory Prediction for Traffic Conflict Detection at Unsignalized Intersections. *J. Adv. Transp*. 2021, 2021, 8453726.

21. Lian J., Yu F., Li L., Zhou Y. Early intention prediction of pedestrians using contextual attention–based LSTM. *J. Multimed. Tools Appl*. 2023, 82, 14713–14729.

22. Alghodhaifi H., Lakshmanan S. Holistic Spatio–Temporal Graph Attention for Trajectory Prediction in Vehicle–Pedestrian Interactions. *J. Sens*. 2023, 23, 7361.

23. Ji Z., Shen G., Wang J., Collotta M., Liu Z., Kong X. Multi–Vehicle Trajectory Tracking towards Digital Twin Intersections for Internet of Vehicles. *J. Electron*. 2023, 12, 275.

24. Yao R., Zeng W., Chen Y., He Z. A deep learning framework for modelling left–turning vehicle behaviour considering diagonal–crossing motorcycle conflicts at mixed–flow intersections. *Transp. Res. Part C Emerg. Technol*. 2021, 132, 103415.

25. Liang E., Stamp M. Predicting pedestrian crosswalk behavior using Convolutional Neural Networks. *J. Traffic Inj. Prev*. 2023, 24. Pp. 338–343.

26. Sun J., Qi X., Xu Y., Tian Y. Vehicle Turning Behavior Modeling at Conflicting Areas of Mixed–Flow Intersections Based on Deep Learning. *J. IEEE Trans. Intell. Transp. Syst.* 2020, 21, 3674–3685.

27. Komol M.M.R., Pinnow J., Elhenawy M., Yasmin S., Masoud M., Glaser S., Rakotonirainy A. A Review on Drivers' Red Light Running Behavior Predictions and Technology Based Countermeasures. *IEEE Access.* 2022, 10, 25309–25326.

28. Jeong Y. Stochastic Model–Predictive Control with Uncertainty Estimation for Autonomous Driving at Uncontrolled Intersections. *J. Appl. Sci.* 2021, 11, 9397.

29. Chen Q., Zhu S., Wu J., Chang H., Wang H. An Acceleration Denoising Method Based on an Adaptive Kalman Filter for Trajectory in Merging Zones. *J. Adv. Transp*. 2023, 2023, 2661136.

30. Qian L.P., Feng A., Yu N., Xu W., Wu Y. Vehicular Networking–Enabled Vehicle State Prediction via Two–Level Quantized Adaptive Kalman Filtering. *IEEE Internet Things J.* 2020, 7, 7181–7193.

31. Yurtsever E., Lambert J., Carballo A., Takeda K. A survey of autonomous driving: common practices and emerging technologies. *IEEE Access*. 2020. Vol. 8. P. 58443–58469.

32. Mao C., Bao L., Yang S., Xu W., Wang Q. Analysis and Prediction of Pedestrians' Violation Behavior at the Intersection Based on a Markov Chain. *Sustainability.* 2021, 13, 5690.

33. Nasernejad P., Sayed T., Alsaleh R. Modeling pedestrian behavior in pedestrian–vehicle near misses: A continuous Gaussian Process Inverse Reinforcement Learning (GP–IRL) approach. *Accid. Anal. Prev*. 2021, 161, 106355.

34. Zhang M., Fu R. Morris D.D., Wang C. A Framework for Turning Behavior Classification at Intersections Using 3D LIDAR. *IEEE Trans. Veh. Technol*. 2019, 68, 7431–7442.

35. Sun L., Zhan W., Wang D., Tomizuka M. Interactive Prediction for Multiple, Heterogeneous Traffic Participants with Multi–Agent Hybrid Dynamic Bayesian Network. *In Proceedings of the IEEE Intelligent Transportation Systems Conference (IEEE–ITSC)*, Auckland, New Zealand, 27–30 October 2019; pp. 1025–1031.

36. Xu Q., Wu H., Wang J., Xiong H., Liu J., Li K. Roadside pedestrian motion prediction using Bayesian methods and particle filter. *J. IET Intell. Transp. Syst*. 2021, 15, 1167–1182.

37. Xu Y., Wu R., Zhou Z., Qiao Q. Modeling the Behavior of Pedestrians in a Group at Signalized Intersections Using Dynamic Bayesian Method. *In Proceedings of the 19th COTA International Conference of Transportation Professionals (CICTP)–Transportation in China 2025*, Nanjing, China, 6–8 July 2019; pp. 112–123.

38. Yang L., Luo X., Zuo Z., Zhou S., Huang T., Luo S. A novel approach for fine–grained traffic risk characterization and evaluation of urban road intersections. *Accid. Anal. Prev.* 2023, 181, 106934.

39. Khoshkangini R., Mashhadi P., Tegnered D., Lundström J., Rögnvaldsson T. Predicting Vehicle Behavior Using Multi–task Ensemble Learning. *Expert Syst. Appl.* 2023, 212, 16.

40. Sethuraman R., Sellappan S., Shunmugiah J., Subbiah N., Govindarajan V., Neelagandan S. An optimized AdaBoost Multi–class support vector machine for driver behavior monitoring in the advanced driver assistance systems. *Expert Syst. Appl.* 2023, 212, 12.

41. Xu D., Liu M.F., Yao X.P., Lyu N.C. Integrating Surrounding Vehicle Information for Vehicle Trajectory Representation and Abnormal Lane–Change Behavior Detection. *Sensors.* 2023, 23, 9800.

42. Liu P.F., Fan W. Extreme Gradient Boosting (XGBoost) Model For Vehicle Trajectory Prediction in Connected and Autonomous Vehicle Environment. *Promet–Traffic Transp*. 2021, 33, 767–774.

43. Horng G.–J., Huang Y.–C., Yin Z.–X. Using Bidirectional Long–Term Memory Neural Network for Trajectory Prediction of Large Inner Wheel Routes. *Sustainability*. 2022, 14, 5935.

44. Zhou X., Ren H.Y., Zhang T.T., Mou X.A., He Y., Chan C.Y. Prediction of Pedestrian Crossing Behavior Based on Surveillance Video. *Sensors.* 2022, 22, 1467.

45. Wang C., Sun Q.Y., Li Z., Zhang H.J., Ruan, K.L. Cognitive Competence Improvement for Autonomous Vehicles: A Lane Change Identification Model for Distant Preceding Vehicles. *IEEE Access*. 2019, 7, 83229–83242.

46. Hu X.H., Chen S.Z., Zhao J.H., Cao Y.S., Wang R., Zhang T.T., Long B., Xu Y.M., Chen X.H., Zheng M.T.Y. Bayesian Global Optimization Gated Recurrent Unit Model for Human–Driven Vehicle Lane–Change Trajectory Prediction Considering Hyperparameter Optimization. *Transp. Res. Rec*. 2023, 2023, 03611981231182390.

47. Rahman M.S., Abdel–Aty M., Lee J., Rahman M.H. Safety benefits of arterials' crash risk under connected and automated vehicles. *Transp. Res. Part C Emerg. Technol.* 2019, 100, 354–371.

48. Xu Y., Ma Z., Sun, J. Simulation of turning vehicles' behaviors at mixed–flow intersections based on potential field theory. *Transp. B Transp. Dyn.* 2019, 7, 498–518.

49. Noh S. Decision–Making Framework for Autonomous Driving at Road Intersections: Safeguarding Against Collision, Overly Conservative Behavior, and Violation Vehicles. *IEEE Trans. Ind. Electron.* 2019. 66, 3275–3286.

50. Sun C., Leng J., Lu B. Interactive Left–Turning of Autonomous Vehicles at Uncontrolled Intersections. *IEEE Trans. Autom. Sci. Eng*. 2022.

51. Xin X., Jia N., Ling S., He Z. Prediction of pedestrians' wait–or–go decision using trajectory data based on gradient boosting decision tree. *Transp. B Transp. Dyn*. 2022, 10. Pp. 693–717.

52. Zhang T., Fu M., Song W. Risk–Aware Decision–Making and Planning Using Prediction–Guided Strategy Tree for the Uncontrolled Intersections. *IEEE Trans. Intell. Transp. Syst.* 2023, 24, 10791–10803.

53. Tian D., Wang Y., Yu T. Fuzzy Risk Assessment Based on Interval Numbers and Assessment Distributions. *Int. J. Fuzzy Syst*. 2020, 22, 1142–1157.

54. Chu H.Q., Guo L.L., Yan Y.J., Gao B.Z., Chen H. Self–Learning Optimal Cruise Control Based on Individual Car–Following Style. *IEEE Trans. Intell. Transp. Syst.* 2021, 22, 6622–6633.

55. Xu X., Zuo L., Li X., Qian L.L., Ren J.K., Sun Z.P. A Reinforcement Learning Approach to Autonomous Decision Making of Intelligent Vehicles on Highways. *IEEE Trans. Syst. Man, Cybern. Syst.* 2020, 50, 3884–3897.

56. Hu T., Luo B., Yang C., Huang T. MO–MIX: Multi–Objective Multi–Agent Cooperative Decision–Making with Deep Reinforcement Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 2023, 45, 12098–12112.

57. Gutierrez–Moreno R., Barea R., Lopez–Guillen E., Araluce J., Bergasa L.M. Reinforcement Learning–Based Autonomous Driving at Intersections in CARLA Simulator. *Sensors.* 2022, 22, 8373.

58. Shu H., Liu T., Mu X., Cao D. Driving Tasks Transfer Using Deep Reinforcement Learning for Decision–Making of Autonomous Vehicles in Unsignalized Intersection. *IEEE Trans. Veh. Technol.* 2022, 71, 41–52.

59. Li G., Lin S., Li S., Qu X. Learning Automated Driving in Complex Intersection Scenarios Based on Camera Sensors: A Deep Reinforcement Learning Approach. *IEEE Sens. J.* 2022, 22, 4687–4696.

60. Nan J., Deng W., Zheng B. Intention Prediction and Mixed Strategy Nash Equilibrium–Based Decision–Making Framework for Autonomous Driving in Uncontrolled Intersection. *IEEE Trans. Veh. Technol*. 2022, 71, 10316–10326.

61. Kiran B.R., Sobh I., Talpaert V., Mannion P., Al Sallab A.A., Yogamani S., Pérez P. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Trans. Intell. Transp. Syst.* 2022, 23, 4909–4926.

62. Abdoos M. A Cooperative Multiagent System for Traffic Signal Control Using Game Theory and Reinforcement Learning. *IEEE Intell. Transp. Syst. Mag*. 2021, 13, 6–16.

63. Stryszowski M., Longo S., Velenis E., Forostovsky G. A Framework for Self–Enforced Interaction between Connected Vehicles: Intersection Negotiation. *IEEE Trans. Intell. Transp. Syst.* 2021, 22, 6716–6725.

64. Wang H., Meng Q., Chen S., Zhang X. Competitive and cooperative behaviour analysis of connected and autonomous vehicles across unsignalised intersections: A game–theoretic approach. *Transp. Res. Part B Methodol.* 2021. Vol. 149. Pp. 322–346.

65. Cheng Y., Zhao Y., Zhang R., Gao L. Conflict Resolution Model of Automated Vehicles Based on Multi–Vehicle Cooperative Optimization at Intersections. Sustainability. 2022, 14, 3838.

66. Jia S.Z., Zhang Y.X., Li X., Na X.X., Wang Y.H., Gao B.Z., Zhu B., Yu R.J. Interactive Decision–Making With Switchable Game Modes for Automated Vehicles at Intersections. *IEEE Trans. Intell. Transp. Syst*. 2023, 15, 11785–11799.

67. Козельський О.В. Метод тензорної декомпозиції для адаптивного розподілу ресурсів у системах реального часу. *Measuring and computing devices in technological processes*. 2025. №82(2). С. 426–433. DOI: https://doi.org/10.31891/2219–9365–2025–82–61

68. Козельський О., Савенко Б. Дворівнева стратегія підвищення відмовостійкості операційних систем реального часу з використанням ймовірнісного аналізу. *Herald of Khmelnytskyi National University. Technical Sciences*. 2025. №353(3.2). С. 438–446. DOI: https://doi.org/10.31891/2307–5732–2025–353–60

69. Козельський О., Савенко Б. Зовнішня адаптивна система кластеризації для підвищення гнучкості операційних систем реального часу на основі динамічного планування завдань. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки*. Том 5. 2025. С. 124–132 DOI: https://doi.org/10.32782/2663–5941/2025.4.2/38

70. Козельський О., Савенко О. Проактивний механізм інформаційної безпеки в операційних системах реального часу з використанням гібридного сторожового таймера. *Measuring and computing devices in technological processes*. 2025. №83(3). С. 459 – 466. DOI: https://doi.org/10.31891/2219–9365–2025–83–57

71. Козельський О., Савенко О. Виявлення зловмисних атак на сенсори та підробки телеметрії в кіберфізичних системах на основі модифікованого фільтра Калмана. *Measuring and computing devices in technological processes*. 2025. №84(4). С. 228–235. DOI: https://doi.org/10.31891/2219–9365–2025–84–24

72. Shu K.Q., Mehrizi R.V., Li S., Pirani M., Khajepour A. Human Inspired Autonomous Intersection Handling Using Game Theory. *IEEE Trans. Intell. Transp. Syst*. 2023, 24, 11360–11371.

73. Darko J., Folsom L., Pugh N., Park H., Shirzad K., Owens J., Miller A. Adaptive personalized routing for vulnerable road users. *IET Intell. Transp. Syst*. 2022, 16, 1011–1025.

74. Kozelskiy O., Kashtalian A., Stetsyuk V., Martiniuk D., Sachenko A. A model of an intelligent clustering system with an external module for the architecture of RTOS with intensive changes of states regarding their flexibility and balancing. *1st International Workshop on Advanced Applied Information Technologies: (AdvAIT–2024)*, Khmelnytskyi, Ukraine and Zilina, Slovakia, December 5, 2024. Vol. 3899. P. 234–243. URL: https://ceur–ws.org/Vol–3899/paper21.pdf

75. Kozelskiy O., Drozd A., Savenko B., Gaj P. A model for probabilistic monitoring and proactive restart of real–time operating systems under intensive state changes in cyber–physical systems. *2nd International Workshop on Intelligent & CyberPhysical Systems: (ICyberPhyS 2025)*, Khmelnytskyi, Ukraine, July 4, 2025. Vol. 4013. P. 198–210. URL: https://ceur–ws.org/Vol–4013/paper16.pdf

76. Gaj P., Sochor T., Korobchynskyi M., Savenko B., Kozelskyi O. Hybrid Method for Protecting RTOS from Failures and Cyberattacks Using Compact Markov Models. *The International Workshop on Applied Intelligent Security Systems in Law*

Enforcement (AISSLE–2025), Vinnytsia, Ukraine, October, 30 – November, 02, 2025. Vol. 4126. pp. 361–376. URL: https://ceur–ws.org/Vol–4126/paper19.pdf

77. Zhao X., Su J., Cai J., Yang H., Xi T. Vehicle anomalous trajectory detection algorithm based on road network partition. *Appl Intell.* 2022, 52, 8820–8838.

78. Wei W., Gao F., Scherer R., Damasevicius R., Polap D. Design and Implementation of Autonomous Path Planning for Intelligent Vehicle. *J. Internet Technol.* 2021, 22, 957–965.

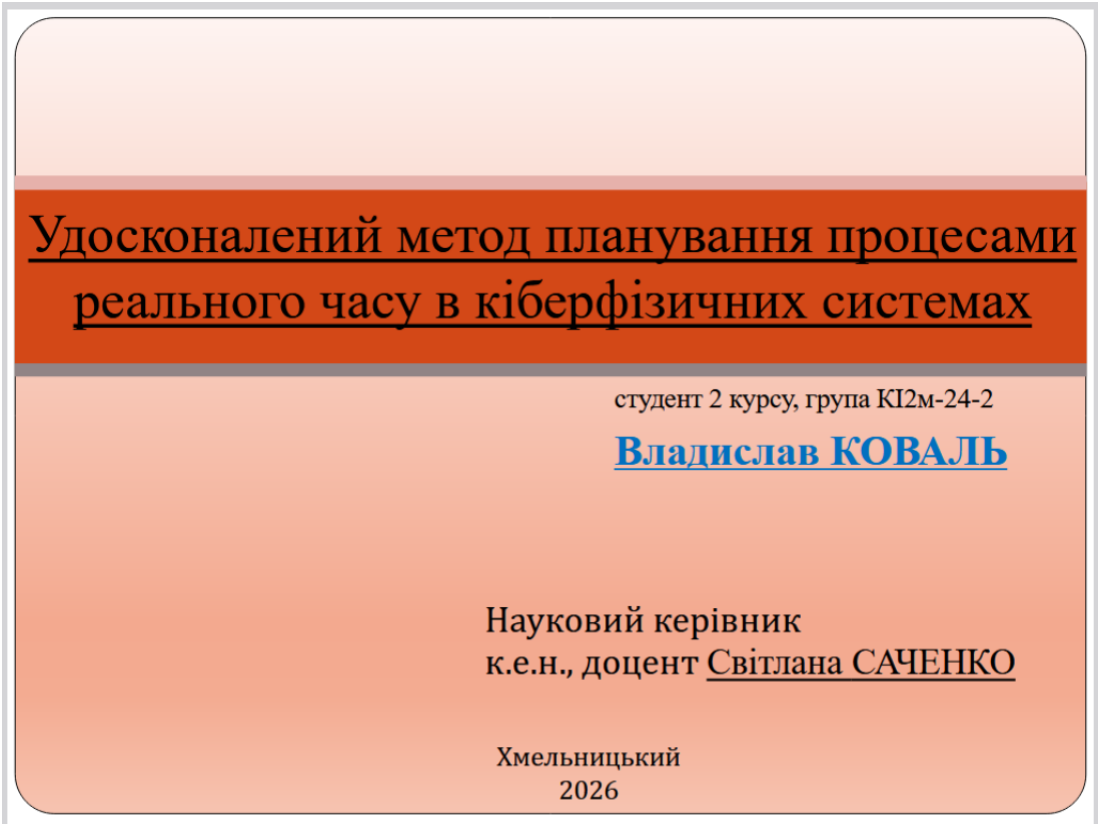
79. Zhang J., Wu J., Shen X., Li Y.S. Autonomous land vehicle path planning algorithm based on improved heuristic function of A–Star. *Int. J. Adv. Rob. Syst*. 2021, 18, 10.

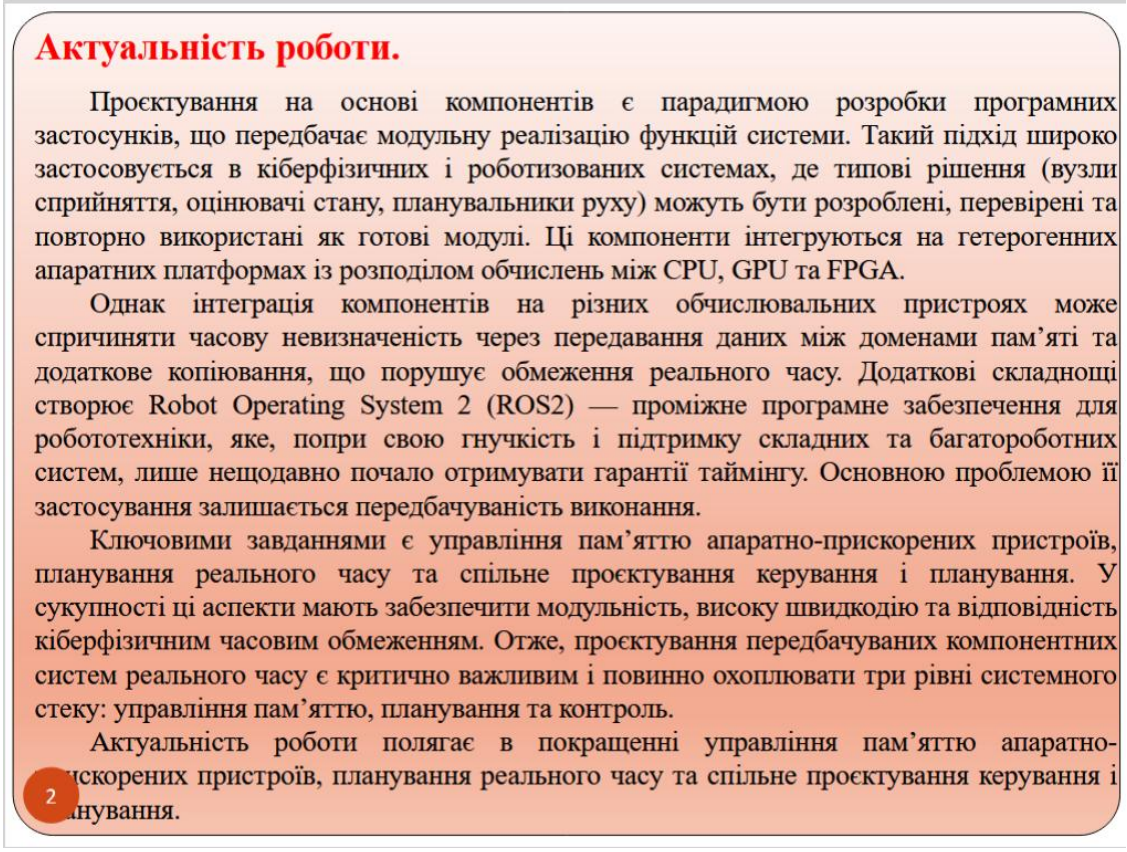
80. Xidias E., Zacharia P., Nearchou A. Intelligent fleet management of autonomous vehicles for city logistics. *Appl. Intell*. 2022, 52, 18030–18048.

81. Коваль В., Каштальян А. Метод планування процесами реального часу в кіберфізичних системах на основі нульового копіювання. *Herald of Khmelnytskyi National University. Technical Sciences*. 2026. № 361(1). С. 568–574. DOI: https://doi.org/10.31891/2307–5732–2026–361–78

ДОДАТОК А
(обов'язковий)

Презентація до роботи





Об'єктом дослідження є планування процесів реального часу в кіберфізичних системах.

Предметом дослідження є методи та засоби планування процесів реального часу в кіберфізичних системах.

Метою кваліфікаційної роботи магістра є покращення планування процесами реального часу в кіберфізичних системах за рахунок розроблення нульового копіювання даних в різних компонентах при управлінні пам'яттю апаратно-прискорених пристроїв, забезпеченні планування реального часу та спільного проєктування керування і планування.

Поставлена мета досягається розв'язанням таких основних завдань:

- проаналізувати відомі методи планування процесами реального часу в кіберфізичних системах та спільного проєктування керування і планування;
- розробити метод планування процесами реального часу в кіберфізичних системах на основі механізму нульового копіювання;
- розробити актуатор та алгоритми для забезпечення реалізації механізму нульового копіювання;
- розробити КФС з механізмом нульового копіювання даних в різних компонентах для управління пам'яттю апаратно-прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС та провести з [illegible]експериментальні дослідження.

3

Для розв'язання поставлених задач використовувалися методи планування в операційних системах реального часу, методи синтезу кіберфізичних систем, теорія множин.

Наукова новизна отриманих результатів:

- удосконалено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам'яттю апаратно-прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування, що дає змогу покращити використання пам'яті КФС.

Практична значимість отриманих результатів полягає у розроблені механізму нульового копіювання даних в різних компонентах при управлінні пам'яттю апаратно-прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС.

4

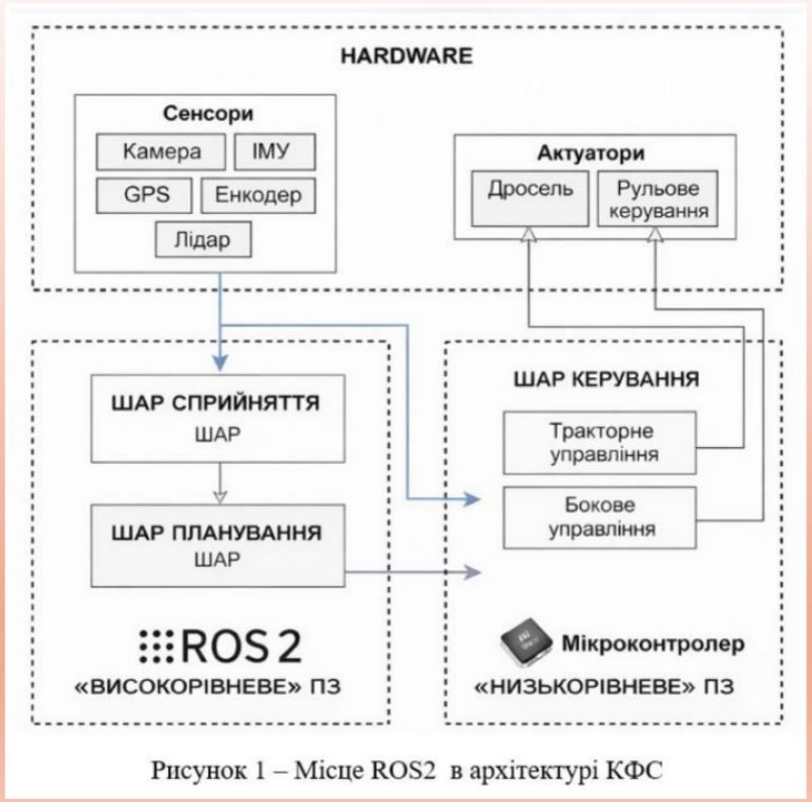


Рисунок 1 – Місце ROS2 в архітектурі КФС

5

Формально КФС подамо як ієрархічну замкнену динамічну систему з керуванням, яка поєднує фізичний об'єкт, обчислювальні підсистеми та канали взаємодії між ними. Узагальнено КФС задамо кортежем так:

$$M_{KFS} = \langle P, S, A, C, N \rangle, \quad (1)$$

де P – фізичний об'єкт (процес); S – множина сенсорів; A – множина актуаторів; C – множина обчислювальних та керуючих підсистем; N – канали інформаційної та керуючої взаємодії.

Фізичний об'єкт P задамо системою диференціальних або різницевих рівнянь стану так:

$$x_1(t) = f\big(x(t), u(t), w(t)\big);$$
$$y_1(t) = h\big(y(t), v(t)\big), \quad (2)$$

де x(t) ϵR^n – вектор стану фізичної системи (положення, швидкість, орієнтація тощо); u(t) ϵR^m – вектор керуючих впливів, що реалізуються актуаторами (дросель, рульове керування); w(t) – зовнішні збурення; y(t) – вихідні величини, доступні для вимірювання; v(t) – вимірювальний шум; $(x_1(t), y_1(t))$ – координати об'єкту P; t – поточний час.

6

Шар сприйняття виконує оцінку стану та середовища. Задамо процес його виконання так:

$$\hat{x}_k = C_1(z_k), \quad (6)$$

де $\hat{x}_k$ – оцінений стан системи та її оточення, отриманий шляхом фільтрації, злиття даних і побудови внутрішніх моделей.

Шар планування формує цілі та траєкторії. Задамо його так:

$$r_k = C_2(\hat{x}_k, Q), \quad (7)$$

де r_k – вектор цільових значень або траєкторій; Q – множина глобальних цілей і обмежень.

Шар керування перетворює цільові параметри у сигнали актуаторів. Задамо його так:

$$u_k = C_3(r_k, \hat{x}_k), \quad (8)$$

де C_3 реалізується на мікроконтролері та працює в реальному часі; r_k – вектор цільових значень або траєкторій; $\hat{x}_k$ – оцінений стан системи та її оточення, отриманий шляхом фільтрації, злиття даних і побудови внутрішніх моделей.

7

Актуаторна підсистема A здійснює відображення керуючих сигналів у фізичні дії так:

$$u(t) = A(u_k), \quad (9)$$

де u_k – сигнали актуаторів.

Цифрові команди перетворюються на електричні, механічні або гідравлічні впливи. У даній КФС це відповідає керуванню дроселем (поздовжня динаміка) та рульовим механізмом (бокова динаміка).

Взаємодія між усіма компонентами відбувається через мережу зв'язків N, яка забезпечує:

1) передачу сенсорних даних $S \to C_1$;

2) передачу рішень $C_2 \to C_3$;

3) передачу керування $C_3 \to A$;

4) зворотній зв'язок $P \to S$.

Таким чином, формальна модель КФС замикається у цикл:

$$x \to y \to z \to \hat{x} \to r \to u \to x, \quad (10)$$

який реалізує безперервну кіберфізичну взаємодію між фізичним процесом і цифровими обчисленнями.

8

ПРОЦЕС

Локальна частина алокатора (локальна пам'ять процесу)

```
remap()      unmap()
share()
copy_from()  copy_to()  copy()
```

Вибір реалізації за типом пристрою та стратегією

↓ **remap**

Спільна частина алокатора (спільна пам'ять System V)

Тип пристрою
Номер пристрою
Стратегія розподілу
Лічильники посилань
Параметри пулу пам'яті

Адресація через зсуви (**offset**)

↓ **offset**

Пул пам'яті домену (Host / GPU / FPGA)

Дані повідомлень
allocate / deallocate (0(1))

copy_from / copy_to / copy

9

Рисунок 2 - Структура алокатора та її зв'язок з інтерфейсом

Основні кроки методу планування процесами реального часу в КФС, які повинні забезпечити уникнення проблеми з повторним копіюванням.

1. Моделювання КФС та потоків даних. Формалізація задач реального часу і їхніх вимог до обчислень та пам'яті. Ідентифікація даних, що можуть бути використані через zero-copy (прямий доступ апаратного прискорювача без проміжних буферів).

2. Аналіз архітектури пам'яті прискорювача. Визначення областей пам'яті, які підтримують zero-copy (наприклад, pinned memory у GPU або shared memory на FPGA). Моделювання часу доступу до цих областей і обмежень на одночасний доступ.

3. Розробка детермінованого алокатора пам'яті. Статична або напівстатична алокація memory regions для zero-copy, щоб уникнути динамічного копіювання під час виконання. Визначення верхніх меж часу доступу та забезпечення сумісності з планувальником задач.

4. Планування задач реального часу з урахуванням zero-copy. Інтеграція інформації про доступні zero-copy буфери у планування та забезпечення того, щоб критичні задачі отримували прямий доступ до пам'яті без очікування копіювання. Мінімізація конфліктів за доступ до zero-copy буферів між паралельними задачами.

5. Спільне проєктування керування і планування. Адаптація параметрів регуляторів з урахуванням нульових копій для уникнення затримок. Комбінування керування і планування з інтеграцією zero-copy для гарантування стабільності системи.

6. Верифікація часових і функціональних гарантій. Перевірка виконання дедлайнів у найгіршому випадку при використанні zero-copy. Аналіз впливу zero-copy на стабільність керування та якість регулювання та перевірка, що використання zero-copy ефективно зменшує затримки і навантаження на шину пам'яті.

7. Експериментальна оцінка. Реалізація підходу на цільовій платформі з апаратними прискорювачами. Порівняння продуктивності та часу доступу при класичному копіюванні і zero-copy. Оцінка виграшу у передбачуваності, затримках та ефективності використання пам'яті.

10

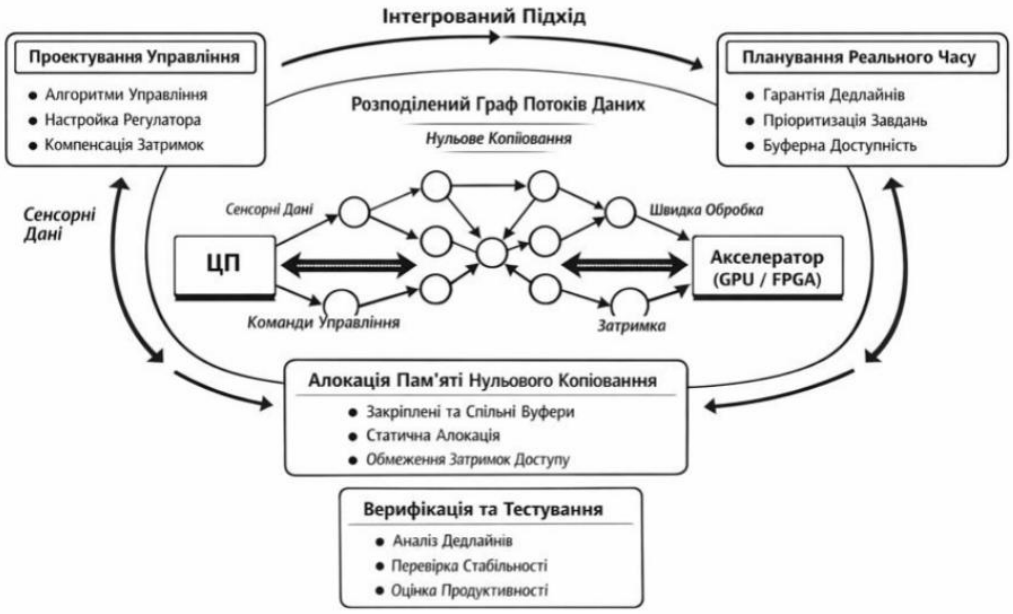


Рисунок 3 – Інтегрований підхід до керування процесами в КФС

11

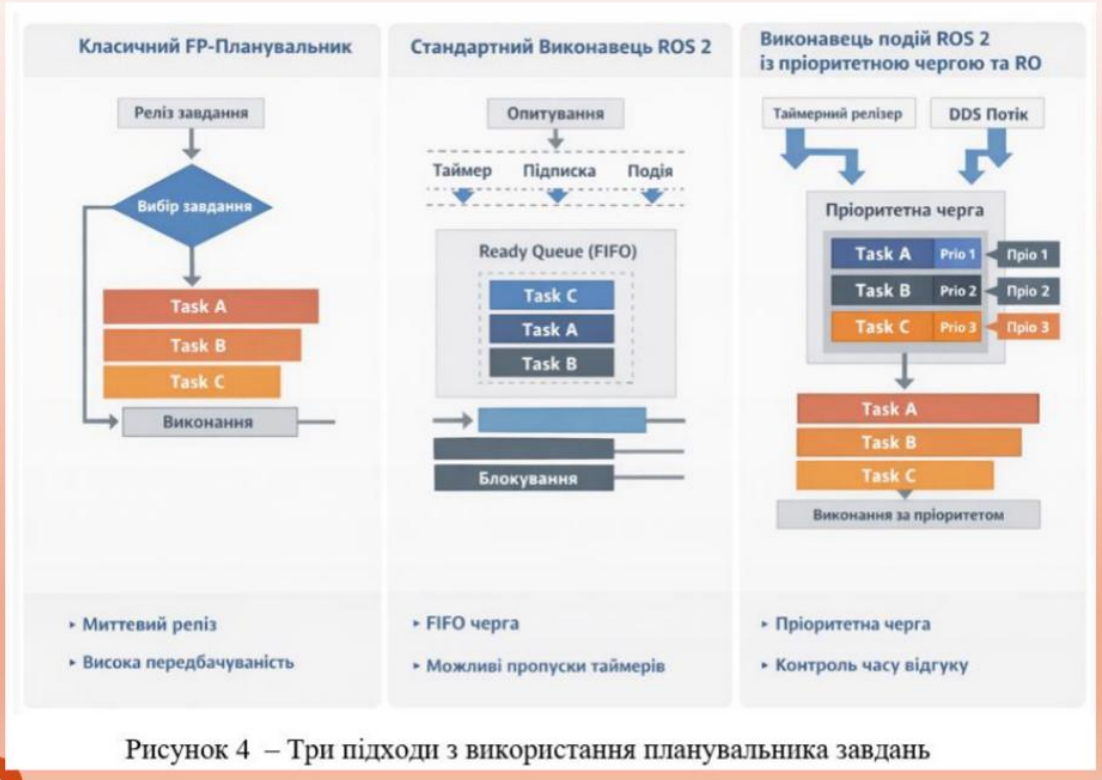


Рисунок 4 – Три підходи з використання планувальника завдань

12

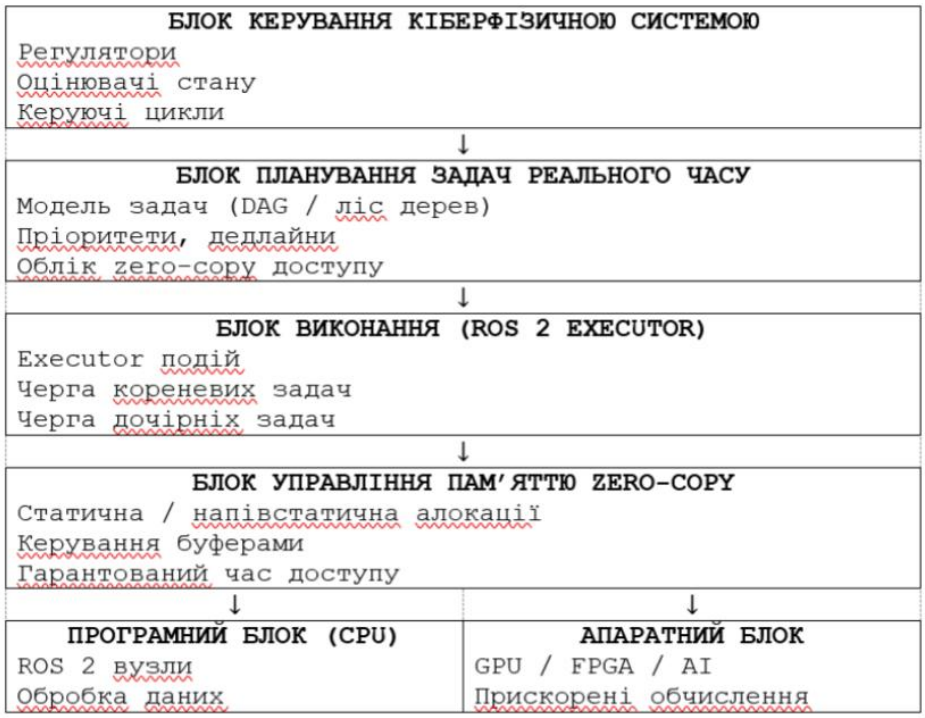


Рисунок 5 – Укрупнена блок-схема засобів реалізації методу планування процесів реального часу в кіберфізичній системі з підтримкою zero-copy доступу до пам'яті апаратних прискорювачів

13

Узагальнений виграш запропонованого методу

Критерій	Виграш
Зменшення середньої затримки	до 60 %
Зменшення джиттера	до 80 %
Зниження навантаження на пам'ять	до 50 %
Усунення повторного копіювання	100 %
Підвищення детермінованості	Якісно суттєве

14

ВИСНОВКИ

- У роботі за результатами виконаних теоретичних та практичних досліджень удосконалено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам'яттю апаратно-прискорених пристроїв та отримано такі результати.
- 1. Проаналізовано відомі методи планування процесами реального часу в кіберфізичних системах та спільного проєктування керування і планування.
- 2. Розроблено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам'яттю апаратно-прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування, що дало змогу покращити використання пам'яті КФС за рахунок уникнення повторного копіювання до 20% порівняно з класичними та використовуваними схемами.
- 3. Розроблено актуатор та алгоритми для забезпечення реалізації механізму нульового копіювання.
- 4. Розроблено прототип КФС з механізмом нульового копіювання даних в різних компонентах для управління пам'яттю апаратно-прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС та провести з нею експериментальні дослідження.

ДОДАТОК Б

(обов'язковий)

Наукова праця здобувача

Technical sciences *ISSN 2307-5732*

https://doi.org/10.31891/2307-5732-2026-361-78
УДК 004.75

КОВАЛЬ ВЛАДИСЛАВ
Хмельницький національний університет
https://orcid.org/0009-0008-6581-0238
vladik1145@gmail.com
КАШТАЛЬЯН АНТОНІНА
Хмельницький національний університет
https://orcid.org/0000-0002-4925-9713
e-mail: yantonina@ukr.net

МЕТОД ПЛАНУВАННЯ ПРОЦЕСАМИ РЕАЛЬНОГО ЧАСУ В КІБЕРФІЗИЧНИХ СИСТЕМАХ НА ОСНОВІ НУЛЬОВОГО КОПІЮВАННЯ

У статті запропоновано модель та архітектуру кіберфізичних систем (КФС), що відображають їхні ключові властивості, зокрема ієрархічну організацію, наявність замкнених контурів керування, розподіл функцій між високорівневими та низькорівневими алгоритмами, а також тісну взаємодію між обчислювальними та фізичними компонентами. Розроблена типова архітектура КФС інтегрує сенсори, актуатори, обчислювальні ресурси та програмне забезпечення різних рівнів через чітко визначені інформаційні й керуючі зв'язки та формує базовий клас систем для дослідження задач реального часу.

Основну увагу приділено архітектурі реалізації методу на основі ROS2, яка включає блок керування КФС, планувальник реального часу, виконавець задач ROS 2 та підсистему управління пам'яттю з підтримкою zero-copy доступу. Запропоновані алгоритми планування враховують DAG-структуру застосунків і забезпечують детермінований доступ до спільної пам'яті, що є критичним для систем з апаратними прискорювачами. Поєднання пріоритетної обробки кореневих задач із використанням LIFO-черги для дочірніх задач дозволяє зменшити затримки виконання та коректно реалізувати механізми спадкування пріоритетів у складних графах потоків даних. Використання zero-copy алокатора усуває повторне копіювання даних між центральним процесором та апаратними прискорювачами, стабілізує часові характеристики та знижує навантаження на підсистему пам'яті.

Експериментальна оцінка підтвердила ефективність запропонованого підходу, продемонструвавши суттєве зменшення середніх і максимальних затримок, джиттера та кількості порушень дедлайнів порівняно з класичними методами. Додатково зафіксовано покращення якості керування, зокрема зменшення перерегулювань і підвищення стійкості до зовнішніх збурень. Отримані результати свідчать про доцільність інтегрованого проєктування планування, керування та управління пам'яттю та підтверджують практичну придатність запропонованого підходу для широкого класу КФС з апаратними прискорювачами.

Ключові слова: *кіберфізичні системи, проміжне програмне забезпечення, пам'ять, архітектура, черги завдань*

KOVAL VLADISLAV
KASHTALIAN ANTONINA
Khmelnytskyi National University

SCHEDULING METHOD FOR REAL-TIME PROCESSES IN CYBERPHYSICAL SYSTEMS BASED ON ZERO COPY

The article proposes a model and architecture of cyber-physical systems (CFS) that reflect their key properties, including hierarchical organization, the presence of closed control loops, the distribution of functions between high-level and low-level algorithms, as well as close interaction between computational and physical components. The developed typical KFS architecture integrates sensors, actuators, computing resources and software of different levels through well-defined information and control links and forms a basic class of systems for researching real-time problems.

The main attention is paid to the architecture of the implementation of the method based on ROS2, which includes the KFS control unit, the real-time scheduler, the ROS 2 task executor and the memory management subsystem with support for zero-copy access. The proposed scheduling algorithms take into account the DAG structure of applications and provide deterministic access to shared memory, which is critical for systems with hardware accelerators. The combination of priority processing of root tasks with the use of a LIFO queue for child tasks allows to reduce execution delays and correctly implement priority inheritance mechanisms in complex graphs of data flows. The use of the zero-copy allocator eliminates the repeated copying of data between the central processor and hardware accelerators, stabilizes time characteristics and reduces the load on the memory subsystem.

Experimental evaluation confirmed the effectiveness of the proposed approach, demonstrating a significant reduction of average and maximum delays, jitter, and the number of deadline violations compared to classical methods. In addition, an improvement in the quality of control was recorded, in particular, a reduction in over-regulation and an increase in resistance to external disturbances. The obtained results indicate the expediency of the integrated design of planning, control and memory management and confirm the practical suitability of the proposed approach for a wide class of KFS with hardware accelerators.

Keywords: *cyber-physical systems, middleware, memory, architecture, task queues*

Стаття надійшла до редакції / Received 07.12.2025
Прийнята до друку / Accepted 11.01.2026
Опубліковано / Published 29.01.2026

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Проєктування на основі компонентів є парадигмою розробки програмних застосунків, що передбачає модульну реалізацію функцій системи. Такий підхід широко застосовується в кіберфізичних і роботизованих

системах, де типові рішення (вузли сприйняття, оцінювачі стану, планувальники руху) можуть бути розроблені, перевірені та повторно використані як готові модулі. Ці компоненти інтегруються на гетерогенних апаратних платформах із розподілом обчислень між CPU, GPU та FPGA. Однак інтеграція компонентів на різних обчислювальних пристроях може спричиняти часову невизначеність через передавання даних між доменами пам'яті та додаткове копіювання, що порушує обмеження реального часу. Додаткові складнощі створює Robot Operating System 2 (ROS2) проміжне програмне забезпечення для робототехніки, яке, попри свою гнучкість і підтримку складних та багатороботних систем, лише нещодавно почало отримувати гарантії таймінгу. Основною проблемою її застосування залишається передбачуваність виконання.

Ключовими завданнями є управління пам'яттю апаратно-прискорених пристроїв, планування реального часу та спільне проєктування керування і планування. У сукупності ці аспекти мають забезпечити модульність, високу швидкодію та відповідність кіберфізичним часовим обмеженням. Отже, проєктування передбачуваних компонентних систем реального часу є критично важливим і повинно охоплювати три рівні системного стеку: управління пам'яттю, планування та контроль.

Актуальність роботи полягає в покращенні управління пам'яттю апаратно-прискорених пристроїв, планування реального часу та спільне проєктування керування і планування.

Аналіз досліджень та публікацій

Проєктування на основі компонентів є важливою парадигмою сучасної розробки програмних застосунків, спрямованою на зменшення складності систем шляхом декомпозиції функціональності на логічно завершені модулі. Такий підхід спрощує тестування, супровід і масштабування програмного забезпечення та широко застосовується в кіберфізичних і роботизованих системах, де складна поведінка формується через композицію типових функціональних елементів. Стандартизовані модулі, зокрема компоненти сприйняття, оцінювання стану, локалізації та планування руху, можуть бути незалежно розроблені, перевірені й повторно використані в різних застосуваннях. Ці модулі зазвичай працюють на гетерогенних апаратних платформах із залученням CPU, GPU та FPGA, що забезпечує високу продуктивність і енергоефективність, але ускладнює інтеграцію. Розподіл обчислень між різними пристроями та доменами пам'яті може спричиняти часову невизначеність, що є критичним для систем реального часу.

Додаткову складність вносить використання Robot Operating System 2 (ROS2), яка надає потужні засоби для розробки робототехнічних систем, але історично орієнтована на гнучкість, а не на суворі часові гарантії. Через це основною проблемою застосування ROS2 у кіберфізичних системах залишається недостатня передбачуваність виконання. Для її забезпечення необхідно вирішити завдання ефективного управління пам'яттю для апаратних прискорювачів, розробки планувальників реального часу та узгодженого проєктування керування і планування з урахуванням часових обмежень. Комплексне поєднання цих підходів дозволяє досягти передбачуваної, високопродуктивної та надійної роботи компонентних кіберфізичних систем реального часу. ROS2 є поширеним програмним фреймворком для розробки робототехнічних застосунків реального часу, який підтримує модульну архітектуру, але має обмежену передбачуваність продуктивності через стандартні механізми планування. Це особливо критично для складних застосунків із довільними графами потоків даних. Основними абстракціями ROS2 є вузли, що інкапсулюють функціональність, та теми, які реалізують обмін даними за парадигмою публікації-підписки через проміжне програмне забезпечення, зазвичай на основі DDS. Логіка застосунку формується ланцюгами зворотних викликів, які можуть створювати неявні залежності виконання. Архітектура ROS2 має ієрархічну структуру, де верхній рівень відповідає за мовні прив'язки та рішення щодо планування, середній рівень rcl реалізує базові сервіси, а рівень RMW забезпечує абстракцію над комунікаційною платформою.

У роботі [9] ROS2 було використано для розробки застосунку керування дроном, що слідує за рухомою ціллю. Система включала багатоступеневий сенсорний конвеєр із виявленням і відстеженням об'єкта за допомогою YOLOv4, оцінюванням орієнтації та стану об'єкта з використанням фільтра Калмана, а також оптимізацією траєкторії дрона для уникнення перешкод. Подальші дослідження [10] розширили функцію вартості, врахувавши плавність руху та видимість об'єкта. Попри ефективність, застосунок характеризується значними витратами пам'яті через передачу великих карт між апаратно-прискореними компонентами, що ускладнює системне проєктування. У роботі [11] розглянуто планувальник ROS2 для складних графів із об'єднуючими потоками даних, що вказує на потребу надійного планування в компонентних системах. Zero-copy memory management є сучасним підходом до організації обміну даними в гетерогенних обчисленнях [12, 13], за якого CPU, GPU та спеціалізовані прискорювачі спільно використовують фізичну пам'ять без копіювання даних [14]. На відміну від традиційної передачі через PCI Express [15], такий підхід зменшує затримки, енергоспоживання та надлишкові витрати пам'яті. Zero-copy особливо ефективний для систем реального часу, потокової обробки, машинного навчання та високопродуктивних обчислень [16]. Його реалізація потребує апаратної й програмної підтримки [17], зокрема технологій уніфікованої пам'яті та спільного віртуального адресного простору, що спрощує модель програмування. Водночас ключовими викликами залишаються синхронізація доступу та забезпечення когерентності кешів, які вимагають спеціалізованих механізмів керування [18]. Попри ці труднощі, zero-copy підхід вважається перспективним напрямом розвитку гетерогенних систем завдяки підвищенню продуктивності та енергоефективності [19].

Таким чином, при проєктуванні операційних систем реального часу в кіберфізичних системах на основі компонентного підходу важливими завданнями виступають завдання з планування реального часу,

забезпечення зменшення складності процесу, покращенні управління пам'яттю апаратно-прискорених пристроїв та спільне проєктування керування і планування.

Формулювання цілей статті

Метою роботи є покращення планування процесами реального часу в кіберфізичних системах за рахунок розроблення нульового копіювання даних в різних компонентах при управлінні пам'яттю апаратно-прискорених пристроїв, забезпеченні планування реального часу та спільного проєктування керування і планування.

Виклад основного матеріалу

Для забезпечення ефективного керування пам'яттю в системах реального часу з метою вивільнення пам'яті різних пристроїв КФС розробимо комунікаційне проміжне програмне забезпечення з нульовим копіюванням, яке підтримує обізнаність про домен пам'яті, виконання контекстно необхідних операцій пам'яті між різними доменами пам'яті та забезпечення нульового копіювання між двома компонентами в одному й тому ж домені пам'яті. Спочатку розглянемо архітектуру всієї КФС для типів якої будуть розроблені рішення. Зобразимо на рис. 1 архітектуру типового класу КФС так, щоб відобразити місце основних компонентів систем та місце ROS2 і зв'язки між ними.

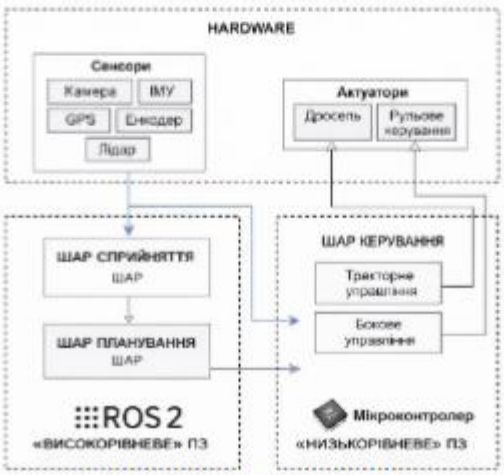


Рис.1. Місце ROS2 в архітектурі КФС

Схема на рис. 1 ілюструє типову архітектуру кіберфізичної системи, у якій фізичні компоненти, обчислювальні ресурси та програмне забезпечення різних рівнів взаємодіють через визначені інформаційні й керуючі зв'язки. Фізичний рівень включає сенсори та актуатори, безпосередньо пов'язані з реальним середовищем. Сенсори, зокрема камера, IMU, GPS, енкодери та лідар, формують багатовимірний цифровий потік даних, що описує стан системи та її оточення. Сенсорні дані передаються до високорівневого програмного забезпечення на основі ROS 2, де на рівні сприйняття виконуються фільтрація, синхронізація та злиття інформації, а також побудова внутрішніх представлень стану і середовища. Далі ці дані надходять до шару планування, який формує траєкторії, цільові стани та послідовності дій з урахуванням обмежень і поточного контексту. Сформовані рішення передаються до низькорівневого програмного забезпечення на мікроконтролері, де реалізується шар керування. Він відповідає за перетворення високорівневих команд у детерміновані керуючі сигнали реального часу для поздовжнього та бокового руху. Зворотний зв'язок від сенсорів використовується для стабілізації та корекції руху. Керуючі сигнали подаються на актуатори, зокрема дросель і рульове керування, які забезпечують зміну швидкості та напряму руху. Таким чином реалізується замкнений цикл «сприйняття – планування – керування – фізична дія – нове сприйняття», що є базовою характеристикою кіберфізичних систем. Актуатори виступають ключовою ланкою, через яку цифрові рішення впливають на фізичний об'єкт, а їхня надійність і точність визначають загальну ефективність і стійкість КФС.

Уся архітектура демонструє чіткий поділ відповідальності між рівнями, але водночас тісну інтеграцію між ними через спрямовані інформаційні та керуючі зв'язки. Високорівневе програмне забезпечення забезпечує інтелектуальну обробку інформації та прийняття рішень, тоді як низькорівневе ПЗ гарантує детерміноване, швидке та надійне виконання цих рішень у фізичному світі. Саме така ієрархічна, але взаємопов'язана структура є типовою для сучасних КФС і дозволяє поєднати складні алгоритми з реальними обмеженнями фізичних процесів.

Формально КФС подамо як ієрархічну замкнену динамічну систему з керуванням, яка поєднує фізичний об'єкт, обчислювальні підсистеми та канали взаємодії між ними. Узагальнено КФС задамо кортежем так:

$$M_{KFS} = \langle P, S, A, C, N \rangle, \quad (1)$$

де P – фізичний об'єкт (процес); S – множина сенсорів; A – множина актуаторів; C – множина обчислювальних та керуючих підсистем; N – канали інформаційної та керуючої взаємодії.

Фізичний об'єкт P задамо системою диференціальних або різницевих рівнянь стану так:

$$x_1(t) = f\big(x(t), u(t), w(t)\big);$$
$$y_1(t) = h\big(y(t), v(t)\big), \quad (2)$$

де x(t) ϵR^n – вектор стану фізичної системи (положення, швидкість, орієнтація тощо); u(t) ϵR^m – вектор керуючих впливів, що реалізуються актуаторами (дросель, рульове керування); w(t) – зовнішні збурення; y(t) – вихідні величини, доступні для вимірювання; v(t) – вимірювальний шум; $(x_1(t), y_1(t))$ – координати об'єкту P; t – поточний час.

Сенсорна підсистема S реалізує відображення фізичного стану у цифрові дані і задамо її так:

$$z_1(t) = S\big(y(t)\big), \quad (3)$$

де y(t) – вихідні величини, доступні для вимірювання; t – поточний час.

У дискретному часі визначення відображення фізичного стану у цифрові дані задамо так:

$$z_k = h_k(y_k) + v_k, \quad (4)$$

де y_k – вихідні величини, доступні для вимірювання у визначений час та через певні інтервали часу; v_k – вимірювальний шум у визначений час та через певні інтервали часу.

Обчислювальна підсистема C має ієрархічну структуру. Подамо її композицією трьох рівнів так:

$$C = C_1 \cdot C_2 \cdot C_3, \quad (5)$$

де C_1 – шар сприйняття; C_2 – шар планування; C_3 – шар керування.

Шар сприйняття виконує оцінку стану та середовища. Задамо процес його виконання так:

$$\hat{x}_k = C_1(z_k), \quad (6)$$

де $\hat{x}_k$ – оцінений стан системи та її оточення, який отримано шляхом фільтрації, злиття даних і побудови внутрішніх моделей.

Шар планування формує цілі та траєкторії. Задамо його так:

$$r_k = C_2(\hat{x}_k, Q), \quad (7)$$

де r_k – вектор цільових значень або траєкторій; Q – множина глобальних цілей і обмежень.

Шар керування перетворює цільові параметри у сигнали актуаторів. Задамо його так:

$$u_k = C_3(r_k, \hat{x}_k), \quad (8)$$

де C_3 реалізується на мікроконтролері та працює в реальному часі; r_k – вектор цільових значень або траєкторій; $\hat{x}_k$ – оцінений стан системи та її оточення, отриманий шляхом фільтрації, злиття даних і побудови внутрішніх моделей.

Актуаторна підсистема A здійснює відображення керуючих сигналів у фізичні дії так:

$$u(t) = A(u_k), \quad (9)$$

де u_k – сигнали актуаторів.

Цифрові команди перетворюються на електричні, механічні або гідравлічні впливи. У даній КФС це відповідає керуванню дроселем (поздовжня динаміка) та рульовим механізмом (бокова динаміка).

Взаємодія між усіма компонентами відбувається через мережу зв'язків N, яка забезпечує: передачу сенсорних даних $S \to C_1$; передачу рішень $C_2 \to C_3$; передачу керування $C_3 \to A$; зворотній зв'язок $P \to S$.

Таким чином, формальна модель КФС замикається у цикл:

$$x \to y \to z \to \hat{x} \to r \to u \to x. \quad (10)$$

Такий цикл реалізує безперервну кіберфізичну взаємодію між фізичним процесом і цифровими обчисленнями. Запропонована модель відображає ключові властивості типових КФС: ієрархічність; наявність замкнених контурів керування; розподіл між високорівневими та низькорівневими алгоритмами; зв'язок між обчислювальними та фізичними компонентами. Розроблено типову архітектуру кіберфізичної системи, у якій інтегровані фізичні компоненти, обчислювальні ресурси та програмне забезпечення різних рівнів із чітко визначеними інформаційними й керуючими зв'язками. Фізичний рівень включає сенсори та актуатори, безпосередньо пов'язані з реальним середовищем. Камера, ІМУ, GPS, енкодери та лідар забезпечують збір багатовимірних даних про стан системи й оточення, які в цифровому вигляді надходять до обчислювальної частини. Запропонована архітектура формує базовий клас КФС і є основою для дослідження підходів до управління пам'яттю апаратно-прискорених пристроїв, планування реального часу та спільного проєктування керування і планування.

Архітектура реалізації методу наведена на рис. 2. Блок керування КФС забезпечує стабільність і якість регулювання з урахуванням детермінованих затримок zero-copy доступу до пам'яті. Планувальник реального часу формує розклад задач на основі графів, дедлайнів і пріоритетів, поєднуючи планування обчислень і пам'яті. ROS 2 executor виконує задачі відповідно до розкладу з підтримкою спадкування пріоритетів і обмеженої преемпції, а блок zero-copy управління пам'яттю гарантує детермінований доступ без копіювання та зменшує навантаження на шину даних. Обчислення виконуються на CPU та апаратних прискорювачах із використанням спільних zero-copy буферів.

Ключові засоби реалізації: ROS 2 executor (розширений або кастомний); спеціалізований memory allocator; DDS з підтримкою loaned messages; RTOS або Linux з PREEMPT_RT; Pinned / shared memory (GPU), DMA buffers (FPGA).

Рисунок 2 – Укрупнена блок-схема засобів реалізації методу планування процесів реального часу в кіберфізичній системі з підтримкою zero-copy доступу до пам'яті апаратних прискорювачів

Експерименти

Експериментальна оцінка запропонованого методу була спрямована на перевірку зменшення затримок передачі даних, підвищення детермінованості виконання задач реального часу, зниження навантаження на підсистему пам'яті та покращення стабільності керування у кіберфізичних системах з апаратними прискорювачами. Метод порівнювався з традиційним підходом, що базується на копіюванні даних між CPU та прискорювачем.

Експерименти проводилися на КФС-платформі з багатоядерним CPU, апаратним прискорювачем та ROS2, із використанням розширеного виконавця задач і zero-copy алокації пам'яті. Тестовий сценарій включав сенсорні, обчислювальні та керуючі задачі, пов'язані складною DAG-структурою з жорсткими часовими обмеженнями, що дозволило оцінити систему в умовах, наближених до реальних.

Результати (табл. 1) показали, що поєднання zero-copy доступу з детермінованим плануванням суттєво зменшує наскрізні затримки, знижує розкид часу відгуку задач і практично усуває порушення дедлайнів навіть за високого навантаження. Також було зафіксовано зменшення використання пам'яті та усунення фрагментації, що є критичним для безпечних КФС.

Стабілізація затримок позитивно вплинула на якість керування, забезпечивши кращу динаміку та стійкість системи. Загалом експеримент підтвердив, що запропонований підхід перевершує класичні методи за передбачуваністю, часовою ефективністю та стабільністю, створюючи основу для побудови детермінованих і масштабованих кіберфізичних систем реального часу.

Порівняльний аналіз показує, що запропонований метод планування процесів реального часу, який поєднує DAG-представлення застосунків і zero-copy доступ до пам'яті, суттєво перевершує класичні підходи за часовими характеристиками та рівнем детермінованості. На відміну від традиційних планувальників, він враховує особливості обміну даними з апаратними прискорювачами та забезпечує стабільність керування, ефективне використання пам'яті й формальні гарантії виконання дедлайнів.

Таблиця 1

Узагальнений виграш запропонованого методу

Критерій	Виграш
Зменшення середньої затримки	до 60 %
Зменшення джиттера	до 80 %
Зниження навантаження на пам'ять	до 50 %
Усунення повторного копіювання	100 %
Підвищення детермінованості	Якісно суттєве

Метою експериментального дослідження була оцінка ефективності запропонованого підходу в КФС з апаратними прискорювачами шляхом порівняння трьох класів рішень: традиційного планування з копіюванням даних, zero-copy без інтеграції з планувальником та запропонованого DAG-орієнтованого методу з детермінованим доступом до пам'яті. Експерименти проводилися на типовій КФС-конфігурації з сенсорними, обчислювальними та керуючими задачами, пов'язаними складною DAG-структурою потоків даних.

Результати показали, що спільне проєктування планування, керування та управління пам'яттю суттєво зменшує наскрізні затримки, джиттер і розкид часу відгуку задач, а також практично усуває порушення дедлайнів навіть за високого навантаження. Додатково було зафіксовано зростання ефективності використання пам'яті за рахунок усунення копіювання та фрагментації.

Стабілізація затримок позитивно вплинула на якість керування, зменшивши перерегулювання та підвищивши стійкість системи до зовнішніх збурень. Отримані результати підтверджують, що запропонований

метод є універсальним і практично придатним підходом до побудови детермінованих, високопродуктивних кіберфізичних систем реального часу з апаратними прискорювачами.

Висновки з даного дослідження
і перспективи подальших розвідок у даному напрямі

Запропонована модель відображає ключові властивості кіберфізичних систем, зокрема ієрархічну структуру, наявність замкнених контурів керування, розподіл між високорівневими та низькорівневими алгоритмами і тісний зв'язок між обчислювальними та фізичними компонентами. Розроблена типова архітектура КФС інтегрує сенсори, актуатори, обчислювальні ресурси та програмне забезпечення різних рівнів із чітко визначеними інформаційними й керуючими зв'язками та формує базовий клас систем для дослідження планування реального часу, zero-copy управління пам'яттю та спільного проєктування керування і планування.

Архітектура реалізації методу (рис. 2) включає блок керування КФС, планувальник реального часу, ROS 2 executor і підсистему zero-copy управління пам'яттю. Запропоновані алгоритми планування в ROS2, що враховують DAG-структуру застосунків і детермінований доступ до пам'яті, забезпечують передбачуване виконання задач за умов складних потоків даних. Комбінування пріоритетної обробки кореневих задач із LIFO-чергою для дочірніх дозволяє зменшити затримки та коректно реалізувати спадкування пріоритетів, а використання zero-copy алокатора усуває повторне копіювання даних і стабілізує часові характеристики.

Експериментальні результати підтвердили суттєве зменшення затримок, джиттера та порушень дедлайнів порівняно з класичними підходами, а також позитивний вплив на якість керування, зокрема зменшення перерегулювань і підвищення стійкості до збурень. Це свідчить про ефективність інтегрованого підходу до планування, керування та управління пам'яттю у КФС з апаратними прискорювачами.

Подальші дослідження доцільно спрямувати на деталізацію та розширення архітектури КФС, зокрема формалізацію моделей часової поведінки для багаторівневих DAG-застосунків, підтримку динамічної реконфігурації графів задач і адаптивного перерозподілу ресурсів, інтеграцію механізмів формальної верифікації дедлайнів і стабільності керування, розширення підходу на багатовузлові та розподілені КФС, дослідження енергоефективності та спільного планування часу, пам'яті й енергоспоживання, апробацію підходу на реальних робототехнічних і автономних системах із різними типами апаратних прискорювачів.

Література

1. Bonci A., Brunella F., Colletta M., Biase A.D., Dragoni A.F., Libofsha A. ROS 2-Based Architecture for Autonomous Driving Systems: Design and Implementation. *Sensors*. 2026. 26(2). 463. https://doi.org/10.3390/s26020463

2. Rumez M., Grimm D., Kriesten R., Sax E. An Overview of Automotive Service-Oriented Architectures and Implications for Security Countermeasures. *IEEE Access* 2020, 8, 221852–221870.

3. Bonci A., Gaudeni F., Giannini M.C., Longhi S. Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey. *Appl. Sci.* 2023, 13, 12796.

4. Jung H.Y., Paek D.H., Kong S.H. Open-Source Autonomous Driving Software Platforms: Comparison of Autoware and Apollo. *arXiv* 2025, arXiv:2501.18942.

5. Ye Y., Nie Z., Liu X. ROS2 Real-time Performance Optimization and Evaluation. *Chin. J. Mech. Eng.* 2023. 36, 144. https://doi.org/10.1186/s10033-023-00976-5

6. Bonci A., Longhi S., Nabissi G., Scala G. A. Execution Time of Optimal Controls in Hard Real Time, a Minimal Execution Time Solution for Nonlinear SDRE. *IEEE Access*, 2020, vol. 8, pp. 158008-158025. doi: 10.1109/ACCESS.2020.3019776

7. Chen S., Hu X., Zhao J., Wang R., Qiao M. A Review of Decision-Making and Planning for Autonomous Vehicles in Intersection Environments. *World Electric Vehicle Journal*. 2024; 15(3):99. https://doi.org/10.3390/wevj15030099

8. Wu Y., Zhang S.J., Hao J.M., Liu H., Wu X.M., Hu J.N., Walsh M.P. Wallington T.J., Zhang K.M., Stevanovic S. On-road vehicle emissions and their control in China: A review and outlook. *J. Sci. Total Environ.* 2017, 574, 332–349.

9. Kerner B.S. Failure of classical traffic flow theories: Stochastic highway capacity and automatic driving. *J. Phys.* A 2016, 450, 700–747.

10. Hu L., Ou J., Huang J., Chen Y.M., Cao D.P. A Review of Research on Traffic Conflicts Based on Intelligent Vehicles. *IEEE Access*. 2020, 8, 24471–24483.

11. Paden B., Čáp M., Yong S.Z., Yershov D., Frazzoli E. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *J. IEEE T. Intell. Veh.* 2016, 1, 33–55.

12. Zhao J.Y., Zhao W.Y., Deng B., Wang Z.H., Zhang F., Zheng W.X., Cao W.K., Nan J.R., Lian Y.B., Burke A.F. Autonomous driving system: A comprehensive survey. *Expert Syst. Appl.* 2024, 242, 27.

13. Bedratyuk L., Savenko O., *MATCH Commun. Math. Comput. Chem.* 2018, 79, 407–414.

14. Wang Z., Huang H.L., Tang J.J., Lee A.E.Y., Meng X.W. Driving angle prediction of lane changes based on extremely randomized decision trees considering the harmonic potential field method. *J. Transp.* A. 2022, 18, 1601–1625.

15. Li M., Chen X.Q., Lin X., Xu D.Y., Wang Y.H. Connected vehicle-based red-light running prediction for adaptive signalized intersections. *J. Intell. Transport. Syst.* 2018, 22, 229–243.

16. Rahman M., Kang M.W., Biswas P. Predicting time-varying, speed-varying dilemma zones using machine learning and continuous vehicle tracking. *Transp. Res. Part C Emerg. Technol.* 2021, 130, 103310.

17. Paszek K., Grzechca D. Using the LSTM Neural Network and the UWB Positioning System to Predict the Position of Low and High Speed Moving Objects. *J. Sens.* 2023, 23, 19.

18. Zhou S., Xu H., Zhang G., Ma T., Yang Y. Deep learning-based pedestrian trajectory prediction and risk assessment at signalized intersections using trajectory data captured through roadside LiDAR. *J. Intell. Transport. Syst.* 2023.

19. Li Q.Y., Cheng R.J., Ge, H.X. Short-term vehicle speed prediction based on BiLSTM-GRU model considering driver heterogeneity. *J. Phys.* A 2023, 610, 14.

References

1. Bonci A., Brunella F., Colletta M., Biase A.D., Dragoni A.F., Libofsha A. ROS 2-Based Architecture for Autonomous Driving Systems: Design and Implementation. *Sensors*. 2026. 26(2). 463. https://doi.org/10.3390/s26020463

2. Rumez M., Grimm D., Kriesten R., Sax E. An Overview of Automotive Service-Oriented Architectures and Implications for Security Countermeasures. *IEEE Access* 2020, 8, 221852–221870.

3. Bonci A., Gaudeni F., Giannini M.C., Longhi S. Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey. *Appl. Sci.* 2023, 13, 12796.

4. Jung H.Y., Paek D.H., Kong S.H. Open-Source Autonomous Driving Software Platforms: Comparison of Autoware and Apollo. *arXiv* 2025, arXiv:2501.18942.

5. Ye Y., Nie Z., Liu X. ROS2 Real-time Performance Optimization and Evaluation. *Chin. J. Mech. Eng.* 2023. 36, 144. https://doi.org/10.1186/s10033-023-00976-5

6. Bonci A., Longhi S., Nabissi G., Scala G. A. Execution Time of Optimal Controls in Hard Real Time, a Minimal Execution Time Solution for Nonlinear SDRE. *IEEE Access*, 2020, vol. 8, pp. 158008-158025. doi: 10.1109/ACCESS.2020.3019776

7. Chen S., Hu X., Zhao J., Wang R., Qiao M. A Review of Decision-Making and Planning for Autonomous Vehicles in Intersection Environments. *World Electric Vehicle Journal*. 2024; 15(3):99. https://doi.org/10.3390/wevj15030099

8. Wu Y., Zhang S.J., Hao J.M., Liu H., Wu X.M., Hu J.N., Walsh M.P. Wallington T.J., Zhang K.M., Stevanovic S. On-road vehicle emissions and their control in China: A review and outlook. *J. Sci. Total Environ.* 2017, 574, 332–349.

9. Kerner B.S. Failure of classical traffic flow theories: Stochastic highway capacity and automatic driving. *J. Phys.* A 2016, 450, 700–747.

10. Hu L., Ou J., Huang J., Chen Y.M., Cao D.P. A Review of Research on Traffic Conflicts Based on Intelligent Vehicles. *IEEE Access*. 2020, 8, 24471–24483.

11. Paden B., Čáp M., Yong S.Z., Yershov D., Frazzoli E. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *J. IEEE T. Intell. Veh.* 2016, 1, 33–55.

12. Zhao J.Y., Zhao W.Y., Deng B., Wang Z.H., Zhang F., Zheng W.X., Cao W.K., Nan J.R., Lian Y.B., Burke A.F. Autonomous driving system: A comprehensive survey. *Expert Syst. Appl.* 2024, 242, 27.

13. Bedratyuk L., Savenko O., *MATCH Commun. Math. Comput. Chem.* 2018, 79, 407–414.

14. Wang Z., Huang H.L., Tang J.J., Lee A.E.Y., Meng X.W. Driving angle prediction of lane changes based on extremely randomized decision trees considering the harmonic potential field method. *J. Transp.* A. 2022, 18, 1601–1625.

15. Li M., Chen X.Q., Lin X., Xu D.Y., Wang Y.H. Connected vehicle-based red-light running prediction for adaptive signalized intersections. *J. Intell. Transport. Syst.* 2018, 22, 229–243.

16. Rahman M., Kang M.W., Biswas P. Predicting time-varying, speed-varying dilemma zones using machine learning and continuous vehicle tracking. *Transp. Res. Part C Emerg. Technol.* 2021, 130, 103310.

17. Paszek K., Grzechca D. Using the LSTM Neural Network and the UWB Positioning System to Predict the Position of Low and High Speed Moving Objects. *J. Sens.* 2023, 23, 19.

18. Zhou S., Xu H., Zhang G., Ma T., Yang Y. Deep learning-based pedestrian trajectory prediction and risk assessment at signalized intersections using trajectory data captured through roadside LiDAR. *J. Intell. Transport. Syst.* 2023.

19. Li Q.Y., Cheng R.J., Ge, H.X. Short-term vehicle speed prediction based on BiLSTM-GRU model considering driver heterogeneity. *J. Phys.* A 2023, 610, 14.

ДОДАТОК В

(обов'язковий)

Програмний код для реалізації КФС

```
// DAGNode.hpp
#pragma once
#include <vector>
#include <memory>
#include <functional>

class DAGNode {
public:
    using Ptr = std::shared_ptr<DAGNode>;

    DAGNode(std::string name, int priority)
        : name(name), priority(priority) {}

    void addChild(Ptr child) { children.push_back(child); }
    void addParent(Ptr parent) { parents.push_back(parent); }
    bool readyToExecute() const;

    void execute(); // Основна логіка підзадачі

    std::string name;
    int priority;
    std::vector<Ptr> parents;
    std::vector<Ptr> children;
    bool completed = false;
};

// Реалізація перевірки готовності
bool DAGNode::readyToExecute() const {
    for (auto& p : parents) {
        if (!p->completed) return false;
    }
    return true;
}

// Виконання підзадачі
void DAGNode::execute() {
    // Тут можна інтегрувати zero-copy доступ до буфера
    // Наприклад, передача даних на GPU
    completed = true;
    // повідомити Executor про готовність дочірніх
}



// Executor.hpp
#pragma once
```

```
#include "DAGNode.hpp"
#include <queue>
#include <stack>
#include <mutex>

class Executor {
public:
    void submitRoot(DAGNode::Ptr root);
    void submitChild(DAGNode::Ptr child);

    void run();

private:
    std::priority_queue<DAGNode::Ptr, std::vector<DAGNode::Ptr>,
        std::function<bool(DAGNode::Ptr,         DAGNode::Ptr)>>
rootQueue
        {[](DAGNode::Ptr a, DAGNode::Ptr b){ return a->priority
< b->priority; }};

    std::stack<DAGNode::Ptr> childStack;
    std::mutex mtx;
};



// Executor.cpp
#include "Executor.hpp"

void Executor::submitRoot(DAGNode::Ptr root) {
    std::lock_guard<std::mutex> lock(mtx);
    rootQueue.push(root);
}

void Executor::submitChild(DAGNode::Ptr child) {
    std::lock_guard<std::mutex> lock(mtx);
    childStack.push(child);
}

void Executor::run() {
    while (!rootQueue.empty() || !childStack.empty()) {
        while (!rootQueue.empty()) {
            auto node = rootQueue.top(); rootQueue.pop();
            if (node->readyToExecute()) node->execute();
            for (auto& c : node->children) submitChild(c);
        }

        while (!childStack.empty()) {
            auto node = childStack.top(); childStack.pop();
            if (node->readyToExecute()) node->execute();
            for (auto& c : node->children) submitChild(c);
        }
    }
}
```

```
// ZeroCopyBuffer.hpp
#pragma once
#include <cstddef>

class ZeroCopyBuffer {
public:
    ZeroCopyBuffer(size_t size);
    ~ZeroCopyBuffer();

    void* getCPUAddress();
    void* getAcceleratorAddress(); // Наприклад GPU pointer
private:
    void* cpuPtr;
    void* accPtr; // GPU або FPGA
    size_t size;
};
```

Програмний комплекс складається з таких основних модулів:

1. Модуль DAG–завдань представляє DAG граф застосунку, кожна підзадача – окремий об’єкт із пріоритетом, списком батьків і дочірніх підзадач.

2. Виконавець подій (Executor) організований як двочергова система: пріоритетна черга для кореневих задач (таймери та спорадичні події); LIFO–черга для дочірніх підзадач. Обробляє DAG та керує звільненням підзадач залежно від готовності батьків.

3. Zero–copy менеджер пам’яті виділяє статичні або напівстатичні буфери в пам’яті CPU та прискорювачів і забезпечує прямий доступ апаратних прискорювачів до даних без проміжних копій.

4. Модуль керування та планування інтегрує інформацію про пріоритети задач, дедлайни, ресурси апаратних прискорювачів, розраховує послідовність виконання та контролює обмежене преемпційне виконання.

5. Модуль DDS–комунікації відповідає за обмін даними між вузлами ROS 2 через pub/sub та забезпечує LIFO–порядок доставки повідомлень для дочірніх підзадач.

Засоби реалізації

1. ROS 2 Foxy / Humble – для модульного побудови застосунків, pub/sub.

2. C++17/20 – основна мова для алгоритмів планування.

3. CUDA / OpenCL / Xilinx Vitis – доступ до апаратного прискорювача через zero–copy.

4. DDS (Fast DDS, Cyclone DDS) – для комунікації між вузлами.

5. Модуль алокації zero–copy – статичне виділення пам’яті для CPU та прискорювачів.

6. Стандартні контейнерні структури STL – для черг, стеків та DAG.

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Владислав КОВАЛЬ

Співавтор:

Назва: Удосконалений метод планування процесами реального часу в кіберфізичних системах

Експерт: Світлана САЧЕНКО

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:5.96%

Коефіцієнт подібності 2:3.36%

Мікропробіли: 149

Заміна букв: 2

Інтервали: 0

Білі знаки: 1

Дата створення звіту: 2026-04-20 19:39:30.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-04-20 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 26.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

ID: 270562 Назва: МКР Удосконалений метод планування процесами реального часу в кіберфізичних системах Додано в БД: 2026-04-20 Автора: Владислав КОВАЛЬ Керівники: Світлана САЧЕНКО Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	143410	983	37707 (26%)	258 (26%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
269847	Назва: Звіт з НДП Удосконалений метод планування процесами реального часу в кіберфізичних системах Додано в БД: 2026-03-17 Автора: Коваля В.С. Керівники: Павлова О.О. Консультанти: Опоненти:	37107 (26.0%)	255 (26.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Здобувач: Владислав КОВАЛЬ

Тема: Удосконалений метод планування процесами реального часу в кіберфізичних системах

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи магістра:

Кількість листів креслень – ; кількість сторінок записки 78

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано метод розпаралелювання динамічних послідовних системних програм із використанням мереж багатогранних процесів

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі подано об'єкт та предмет дослідження, мету, наукову новизну та практичну цінність роботи, а також характеристику структури роботи.

У першому розділі проведено аналіз відомих рішень щодо кіберфізичних систем і планування реального часу.

У другому розділі здійснено дослідження предметної області, запропоновано архітектуру та компоненти кіберфізичних систем, включно з актуаторами, а також досліджено коректність запропонованих рішень щодо дослідження типового класу КФС.

У третьому розділі розроблено удосконалено метод планування процесами реального часу в кіберфізичних системах, особливістю якого є реалізація нульового копіювання даних в різних компонентах при управлінні пам'яттю апаратно-прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування, що дало змогу покращити використання пам'яті КФС.

У четвертому розділі здійснено розроблення алгоритмів реалізації методу планування процесами реального часу в кіберфізичних системах, засоби реалізації та подано результати експериментальних досліджень з розробленою КФС, включно з обгрунтуванням постановки експмериментів.

У висновках підведено підсумки досягнення результатів з розв'язання завдань дослідження.

4. Позитивні сторони роботи: розроблені механізму нульового копіювання даних в різних компонентах при управлінні пам'яттю апаратно-прискорених пристроїв, забезпечення планування реального часу та спільного проєктування керування і планування в КФС.

5. Негативні сторони роботи:

6. Оцінка графічного оформлення та пояснювальної записки роботи: __=____

7. Відгук про роботу в цілому: __В загальному робота виконана на високому рівні.

8. Інші зауваження: ___=

9. Оцінка кваліфікаційної роботи магістра:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи магістра вважаю, що робота заслуговує оцінки «відмінно» 95.00 (А)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) ____ д.т.н., професор, Мартинюк В.В., професор кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

"__1 травня__"_____2026р.

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Владислав КОВАЛЬ

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-24-2

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Удосконалений метод планування процесами реального часу в кіберфізичних системах
Автор Владислав КОВАЛЬ
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: к.е.н., доцент Світлана САЧЕНКО

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 5,96% та системою Anti-Plagiarism складає 0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

30.04.2026

Завідувач кафедри _____ (Підпис) Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми _____ (Підпис) Олег САВЕНКО (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ (Підпис) Світлана САЧЕНКО (Ім'я, ПРІЗВИЩЕ)