УДК 004.415.2
DOI https://doi.org/10.32782/2663-5941/2025.3.2/22

Гуржій С.В.
Український науково-дослідний інститут
спеціальної техніки та судових експертиз Служби безпеки України

ROS 2: ОСОБЛИВОСТІ АРХІТЕКТУРИ ТА ПРАКТИЧНОГО ВПРОВАДЖЕННЯ

Стаття присвячена дослідженню сучасних тенденцій та особливостей застосування покращеної версії операційної системи для робототехнічних платформ – Robot Operating System 2 (ROS 2). У статті розкрито еволюцію концептуального та архітектурного підходу до побудови програмного забезпечення для автономних роботів у порівнянні з попередньою версією – ROS 1. Розкрито функціональні можливості ROS 2 у контексті забезпечення модульності, масштабованості, надійності та безпеки в умовах гетерогенних і динамічних експлуатаційних середовищ. Визначено, що ROS 2 забезпечує підтримку життєвого циклу вузлів, розширені засоби міжвузлової комунікації (через DDS), багатоплатформність (Linux, Windows, macOS), а також підвищену продуктивність і безпеку при виконанні критичних для місій обчислень. З'ясовано, що впроваджені механізми управління якістю обслуговування (QoS), підтримка функціонування у реальному часі, можливість безпечного розподілу задач між модулями, а також чітко визначена структура розробки та тестування сприяють адаптації ROS 2 до широкого спектру задач – від наукових досліджень до комерційного впровадження. У статті проаналізовано ефективність застосування ROS 2 для безпілотних систем різних типів – повітряних (UAV), наземних (UGV) та морських (USV). Установлено, що ROS 2 пропонує високу ефективність у ситуаціях, які потребують стабільної міжвузлової взаємодії в умовах нестабільного зв'язку, інтеграції великої кількості сенсорів та забезпечення надійної навігації з підтримкою відмовостійкості. Також висвітлено тенденції впровадження ROS 2 у робототехнічних системах та підкреслено її здатність до прискорення процесу переходу від дослідницького прототипу до серійного продукту. Окреслено, що архітектурні інновації ROS 2, зокрема підтримка багатопоточності, гнучких шаблонів вузлів та автоматизованих засобів діагностики, є ключовими факторами успішного застосування даної операційної системи в складних багатокомпонентних системах автономного управління. Представлені дані доводять, що ROS 2 стає стандартом де-факто для розробки автономних робототехнічних систем наступного покоління, здатних функціонувати у складних, мінливих і багатозадачних середовищах. Таким чином, ROS 2 не лише усуває архітектурні та функціональні обмеження попередньої версії, але й формує підґрунтя для впровадження робототехнічних технологій у нових прикладних доменах, включно з промисловими, військовими, пошуково-рятувальними та сервісними застосуваннями.

Ключові слова: *робот, операційна система, ROS, алгоритм, DDS.*

Постановка проблеми. Швидкий розвиток інтелектуальних автономних систем, включаючи безпілотні літальні апарати (UAV – unmanned aerial vehicles), безпілотні наземні апарати (UGV – unmanned ground vehicles) і безпілотні надводні апарати (USV – unmanned surface vessel), зумовив необхідність розробки високомодульних, масштабованих і функціонально сумісних програмних фреймворків. Програмні середовища для робототехніки, які функціонують як базові інфраструктурні платформи, створюють основу для ефективної взаємодії сенсорних систем, актуаторів, керуючих алгоритмів та комунікаційних протоколів. У цьому контексті операційна система роботів (ROS – Robot Operating System) набула статусу провідного рішення з відкритим вихідним кодом, яке визначає вектор розвитку сучасних робототехнічних досліджень та їх практичної імплементації в промисловості.

Оригінальна ROS 1, яка була представлена дослідницькою лабораторією Willow Garage, ініціювала зміну парадигми, сприяючи створенню спільнотою системи багаторазових програмних пакетів, інструментів для моделювання та візуалізації, а також стандартизованих інтерфейсів для керування роботами [1]. ROS 1 значно знизила бар'єр для входу в розробку робототехнічних систем, сприяючи швидкому створенню прототипів та впровадженню інновацій. Однак, незважаючи на широке розповсюдження і корисність, ROS 1 не була спроектована з урахуванням критично важливих вимог до системного рівня. Їй бракувало вбудованої підтримки кібербезпеки, можливості детермінованого виконання в реаль-

ному часі, надійності в умовах погіршення якості мережевого середовища та формального контролю робочого циклу – факторів, необхідних для розгортання критично важливих для безпеки та орієнтованих на виконання завдання застосунків у гетерогенних середовищах.

Оскільки автономні роботизовані системи все частіше використовуються в реальних умовах – від моніторингу цивільної інфраструктури і логістики до оборони, реагування на катастрофи і дослідження космосу – зазначені обмеження стають все більш помітними. Спеціальні архітектурні модифікації і власні розширення, запроваджені розробниками для обходу недоліків ROS 1, виявили нагальну потребу у фундаментально модифікованій системній архітектурі.

У відповідь на зростаючі вимоги робототехнічне програмне забезпечення другого покоління, ROS 2, було повністю перебудоване та адаптоване для забезпечення надійності, масштабованості та безпеки в автономних системах високого рівня продуктивності. ROS 2 побудовано на основі комунікаційного робототехнічного програмного забезпечення Data Distribution Service (DDS) – стандартизованого протоколу обміну даними, який широко використовується в аерокосмічній, оборонній, фінансовій та промисловій галузях автоматизації. DDS додає до ROS 2 набір розширених функцій, таких як налаштування політики якості сервісу (QoS) в реальному часі, децентралізоване виявлення подій, безпечний зашифрований трафік та надійну підтримку гетерогенних і обмежених в ресурсах середовищ, включаючи вбудовані системи та периферійні обчислювальні платформи.

Незважаючи на ці досягнення, на шляху до широкомасштабного впровадження та оптимізації ROS 2 для конкретних прикладних програм, що включають широкий спектр повітряних, наземних та морських автономних платформ, залишається чимало невирішених питань. Такі питання, як синхронізація в реальному часі в ройовій робототехніці, контекстно-орієнтована навігація в умовах відсутності GPS і енергоефективні конвеєри сприйняття, потребують подальших досліджень і методологічного вдосконалення. Крім того, інтеграція ROS 2 з новими технологіями, в тому числі з прийняттям рішень на основі штучного інтелекту, розподіленим периферійним інтелектом і мережевими парадигмами наступного покоління (наприклад, 5G/6G, TSN), відкриває як можливості, так і технічні перешкоди, які необхідно подолати.

Аналіз останніх досліджень і публікацій. Еволюція архітектур керування роботами та робототехнічних платформ триває понад п’ять десятиліть, відображаючи безперервне вдосконалення принципів, які лежать в основі розвитку автономних систем. Так з появою систем, таких як Shakey the Robot, були закладені основи які впровадили уявлення про символічне мислення та планування в контролі поведінки роботів. Науково-технічні досягнення в області безпілотних систем отримали подальший розвиток завдяки удосконаленню модульного дизайну, впровадженню класичних стратегій планування, створенню архітектур на основі поведінки, а також розробці багаторівневих схем управління. Значним проривом у ранніх дослідженнях стала архітектура управління завданнями (TCA – Task Control Architecture), яка базувалася на використанні системи передачі повідомлень для координації поведінкових алгоритмів роботів. Ця концепція була використана в таких системах, як CARMEN, що використовували IPC (міжпроцесорну комунікацію) для полегшення модульного управління роботами. Основою сучасних робототехнічних систем є ефективна організація обміну повідомленнями, що забезпечує інтегровану взаємодію між розподіленими компонентами. Для реалізації цього підходу використовуються протоколи, перевірені в середовищах розподілених обчислень, зокрема MQSeries від IBM, Jini від Java та легковагові рішення типу MQTT. Інтеграція даних технологій у структури робототехнічних систем дозволила підійняти на перший план питання модульності програмного забезпечення, повторного використання компонентів та масштабованості розробок. Як наслідок, перші архітектури програмного забезпечення для роботів, такі як Player, базувалися на клієнт-серверних моделях, де центральний сервер безпосередньо взаємодіяв з апаратним забезпеченням для реалізації логіки управління, а клієнтські додатки отримували дані з сенсорів та актуаторів через TCP-з’єднання. Однак, незважаючи на свій внесок у абстракцію та віддалений доступ, архітектура Player виявила критичні недоліки в плані модульності, надійності та адаптивності. Подальші розробки програмного забезпечення для робототехніки, такі як Yet Another Robot Platform (YARP), намагалися подолати ці обмеження, впроваджуючи однорангові моделі зв’язку. YARP акцентувала увагу на гнучкості завдяки підтримці декількох протоколів зв’язку і забезпечувала високу

швидкодію в сценаріях управління в реальному часі. Це дозволило повторно використовувати код і сприяло спільній розробці [2]. Інший відомий фреймворк, Lightweight Communications and Marshalling (LCM), був адаптований до середовищ з високою пропускною здатністю і низькою затримкою, використовуючи архітектуру публікації/підписки з багатомовними прив'язками. LCM виявився ефективним для швидкого обміну даними в режимі реального часу, але залишався обмеженим для обміну повідомленнями та серіалізації, залишаючи завдання координації на системному рівні розробникам [3]. Доповнюючи LCM, проект Open Robot Control Software (OROCOS) зосередився на управлінні рухом в реальному часі і включав передові обчислювальні інструменти, такі як кінематичні розв'язувачі та байєсівська фільтрація [4]. Інтегруючи програмне забезпечення на основі CORBA, OROCOS підтримував детерміноване виконання та жорсткі часові обмеження. Однак, як LCM, так і OROCOS працювали лише з окремими підсистемами робототехніки, таким чином, не маючи комплексної основи для розробки наскрізних робототехнічних додатків.

У цьому контексті поява операційної системи для роботів (ROS) ознаменувала трансформаційний прогрес в екосистемах програмного забезпечення для робототехніки. ROS 1, вперше випущена в 2010 році, представила модульну архітектуру, яка включала широкий спектр функціональних можливостей, необхідних для автономних систем, в тому числі самоаналіз повідомлень, моніторинг процесів, управління перетворенням координат і синхронізацію часу. Вона підтримувала розподілену модель виконання з різноманітними бібліотеками, які включають сприйняття, планування, активацію і SLAM. Дозволяє швидко створювати прототипи і розгортати гетерогенні роботизовані платформи [5].

Постановка завдання. Сучасні тенденції розвитку безпілотних автономних систем, зокрема повітряних (UAV), наземних (UGV) та надводних (USV) дронів, супроводжуються зростаючими вимогами до їхньої функціональної надійності, структурної живучості, адаптивності до середовищ функціонування та загальної ефективності виконання місій різного рівня складності. Умови експлуатації таких систем охоплюють широкий спектр сценаріїв – від моніторингу, картографування та екологічного спостереження до високоризикованих військових і пошуково-рятувальних операцій. Внаслідок цього особливої актуальності набуває впровадження уніфікованих, масштабованих, модульних та інтероперабельних програмних архітектур, здатних забезпечити гнучке управління апаратними ресурсами та адаптацію до динамічних умов зовнішнього середовища. У даному контексті операційна система ROS 2 є одним із найперспективніших засобів побудови програмної архітектури автономних робототехнічних платформ завдяки підтримці DDS-комунікації, QoS, безпеки, сумісності з вбудованими системами. Такий підхід забезпечує реалізацію як централізованих, так і розподілених систем управління залежно від прикладних вимог.

У зв'язку з цим, метою даного дослідження є проведення комплексного аналізу архітектурних особливостей операційних систем типу ROS 2 у контексті їхньої ефективності при розробці та експлуатації різних типів автономних дронів. Зокрема, передбачається здійснити систематизацію функціональних можливостей ROS 2, оцінити її здатність до масштабування, забезпечення відмовостійкості, підтримки модульності та забезпечення безпечного управління. Також ставиться завдання визначити оптимальні конфігурації ROS 2 для різних типів роботизованих платформ, враховуючи специфіку їх апаратної реалізації та цільових сценаріїв використання.

Виклад основного матеріалу. У контексті еволюції програмних парадигм для роботизованих систем, поява версії ROS 1, датованої 2010 роком, стала знаковою віхою, яка започаткувала інноваційний підхід до розробки операційних систем для робототехнічних застосувань. Архітектурна концепція ROS 1 базувалася на модульності, інтегруючи широкий спектр функціональних блоків, критично важливих для забезпечення автономності роботизованих систем. До ключових компонентів належали механізми аналізу обміну повідомленнями, моніторингу процесів, керування трансформаціями координатних систем та синхронізації часових міток. Реалізація розподіленої моделі виконання, доповнена розгалуженою екосистемою бібліотек, розроблених зусиллями спільноти, охоплювала широкий спектр функціональності, включаючи перцепцію, планування траєкторій, керування приводами та алгоритми одночасної локалізації та картографування (SLAM – Simultaneous Localization and Mapping) [6]. Це сприяло прискореному прототипуванню та розгортанню гетерогенних роботизованих плат-

форм, зокрема безпілотних літальних, наземних, надводних апаратів.

Попри інноваційний підхід, ROS 1 мала низку архітектурних обмежень, що знижували її ефективність у критичних застосуваннях. Зокрема, централізована обробка даних створювала єдину точку відмови, що знижувало надійність у мережах із нестабільним або переривчастим зв'язком (наприклад, Wi-Fi чи супутникові канали). Крім того, базові протоколи TCP/UDP не забезпечували підтримки механізму забезпечення якості обслуговування (QoS – Quality of Service), що ускладнювало використання ROS 1 у системах реального часу та безпеки у критичних додатках. Питання кібербезпеки також залишались відкритими: хоча ініціативи на кшталт Security-Enhanced ROS (SROS) інтегрували шифрування та автентифікацію, їм бракувало стандартизованості та масштабованості, необхідних для відповідності сучасним вимогам у сфері інформаційної безпеки [7].

У відповідь на окреслені недоліки, розробка ROS 2 стала зрушенням у напрямку створення програмного забезпечення корпоративного рівня для робототехніки. ROS 2 інтегрує в якості фундаментальної комунікаційної інфраструктури стандарт Data Distribution Service (DDS) – високопродуктивний протокол, який підтвердив свою ефективність у критично важливих напрямках, таких як аерокосмічна інженерія та охорона здоров'я. Завдяки своїй архітектурі, DDS забезпечує нативну підтримку якості обслуговування (QoS), децентралізоване виявлення учасників та гарантовану низьку латентність, що суттєво розширює застосовність ROS 2 у розподілених системах, чутливих до часових затримок. Крім того, в ROS 2 запроваджено концепцію вузлів, що забезпечує детермінований контроль над переходами станів (наприклад, конфігурація, активація, деактивація) та є критично важливим для систем з високими вимогами до надійності та передбачуваності поведінки.

Ще однією ключовою архітектурною інновацією ROS 2 є спільна клієнтська бібліотека (rcl), яка є основою для реалізацій на різних мовах програмування, підвищуючи крос-платформну консистентність та мінімізуючи фрагментацію екосистеми. Модель багатопотоковості також зазнала значних покращень завдяки впровадженню плагінів, що надає розробникам гнучкість в адаптації стратегій паралелізму до специфічних вимог реального часу. ROS 2 підтримує багатовузлове виконання в межах одного процесу та запроваджує стандартизований інтерфейс на основі сервісів для керування параметрами, замінюючи застаріле рішення, що базувалося на XMLRPC, на більш надійні API з типізованими параметрами.

Структура архітектури системи ROS 2 складається з трьох основних підсистем, кожна з яких виконує окремі, але взаємозалежні функції, необхідні для розробки розподілених та автономних робототехнічних застосунків:

– комунікаційне програмне забезпечення (міжпроцесорна та розподілена інфраструктура обміну повідомленнями) – рівень програмного забезпечення в ROS 2 базується на стандарті DDS (Data Distribution Service) [8]. Цей протокол забезпечує надійну, масштабовану і малозатратну передачу повідомлень між обчислювальними вузлами, гарантуючи підтримку параметрів якості обслуговування (QoS), що є критично важливими для безпечних і чутливих до часу роботизованих операцій. Програмне забезпечення на основі DDS абстрагує комунікаційні API, серіалізацію/десеріалізацію даних та мережеві транспортні рівні, таким чином підтримуючи операції в реальному часі в середовищах зі змінними мережевими характеристиками, наприклад, з якими стикаються автономні дрони під час польотів на великі відстані або висоті, або підводні апарати в морському середовищі з нестабільним зв'язком;

– робототехнічні алгоритми та функціональні бібліотеки – ROS 2 надає широку колекцію багаторазових, параметризованих алгоритмічних модулів, що включають основні можливості робототехніки. До них відносяться прогресивні методи одночасної локалізації і картографування (SLAM), модулі розпізнавання з використанням глибокого навчання і об'єднання сенсорів (LiDAR, RGB-D, RADAR тощо), алгоритми планування руху і траєкторії в реальному часі (наприклад, RRT, DWA і оптимізація траєкторії), а також фреймворки координації поведінки, такі як дерева поведінки і ієрархічні кінцеві автомати. Ці компоненти виступають фундаментальними будівельними блоками для реалізації автономії безпілотних систем, які працюють в умовах відсутності GPS, роїв дронів, які виконують спільну розвідку, або автономних морських систем, які проводять розвідувальні місії;

– інструментарій розробника та операційні утиліти – для підтримки повного життєвого циклу роботизованого програмного забез-

печення – від проектування, моделювання та розгортання до моніторингу під час виконання та аналізу після завершення місії. ROS 2 пропонує комплексний набір інструментів на основі командного рядка та графічного інтерфейсу. Сюди входять інструменти для конфігурації системи (ros2 param, ros2 launch), самоаналізу в реальному часі (ros2 topic, ros2 node info), візуалізації стану (Rviz2) та 3D-симуляції середовища (Gazebo, Ignition та Webots). Інструментарій для безперервної інтеграції та розгортання (CI/CD) включає автоматизовані системи збірки (colcon), робочі процеси контролю версій (vcstool, rosinstall_generator) та розповсюдження бінарних файлів за допомогою ROS Index та ROS Buildfarm. Цей набір інструментів не лише прискорює цикли розробки, але й полегшує відтворюваність та масштабованість, що є критично важливими для розгортання автономних транспортних засобів на рівні кластерів на гетерогенних платформах [9].

На додаток до цих основних компонентів, ROS 2 активно підтримує розширення за допомогою пакетів і фреймворків, які орієнтовані на специфічні вимоги конкретної галузі. Наприклад, micro-ROS забезпечує сумісність ROS 2 з платформами на базі мікроконтролерів, що значно розширює її застосування для периферійних обчислень в умовах обмежених ресурсів, таких як мініатюрні дрони або вбудовані підсистеми в модульних платформах дронів. Аналогічно, інтеграція з операційними системами реального часу (RTOS), такими як FreeRTOS і NuttX, підвищує детермінованість і безпеку в критично важливих сценаріях.

Оптимальне використання платформи ROS 2 потребує ретельної систематизації та оцінки ключових архітектурних критеріїв, які складають фундамент для розробки та інтеграції автоматизованих робототехнічних систем. При цьому особливу увагу слід приділити необхідності проведення всебічного аналізу продуктивності ROS 2, що набуває особливої актуальності при розгортанні систем на базі безпілотних систем різного типу – від легких комерційних дронів до високотехнологічних індустріальних платформ. Такий підхід дозволяє не лише виявити поточні сильні сторони технології, але й сприяє подальшій оптимізації систем автономного управління в умовах складних експлуатаційних середовищ.

Однією з ключових особливостей є повна відмова від централізованої схеми управління на користь децентралізованої комунікаційної моделі, заснованої на протоколі DDS. Завдяки цьому досягається висока стійкість систем до відмов, що особливо актуально для автономних мультиробототехнічних груп, зокрема дронів, які здійснюють операції в умовах нестабільного або переривчастого зв'язку, таких як морські або аерокосмічні місії. Додатковою перевагою є покращене абстрагування за рахунок використання кросплатформеного клієнтського інтерфейсу rcl, що сприяє ефективному переносу програмного забезпечення між різними апаратними платформами, включно з системами реального часу. Такий підхід є важливим для побудови гібридних архітектур із використанням мікроконтролерів та вбудованих систем. ROS 2 також підтримує асинхронну подієво-керовану архітектуру, що дозволяє оперативно реагувати на зміну ситуацій та ефективно розподіляти обчислювальні ресурси. Це критично для дронів, які виконують задачі навігації або ухилення від перешкод у реальному часі. Водночас модульна структура ROS 2 забезпечує високу масштабованість і повторне використання компонентів, що сприяє зниженню витрат на розробку та спрощує реалізацію розподілених функцій, таких як локалізація, планування або міжагентна взаємодія.

Удосконалена комунікаційна інфраструктура ROS 2 забезпечує низку ключових переваг для побудови безпілотних систем різного типу завдяки реалізації трьох основних парадигм взаємодії: топіки, сервіси та дії. Комунікація на основі топіків, яка використовує модель "publish-subscribe", особливо ефективна для високочастотної передачі даних, як-от у наземних UGV-платформах, де необхідна паралельна обробка інформації з низькою затримкою. Інтеграція DDS із підтримкою гнучких налаштувань QoS дозволяє дронам типу UAV зберігати надійність зв'язку навіть у складних умовах із втратами пакетів понад 40% [10]. Парадигма сервісної взаємодії дозволяє вирішувати проблеми синхронізації, що критично важливо для морських USV, де обробка запитів з налаштовуваним тайм-аутом підвищує надійність при плануванні маршруту. Компонентна реалізація дій у ROS 2 забезпечує розширену підтримку станів машин, дозволяючи реалізовувати довготривалі автономні завдання з постійним зворотним зв'язком, що особливо корисно для USV в умовах динамічного морського середовища.

З погляду архітектури міжпрограмного забезпечення ROS 2 реалізує багаторівневу модель абстракції, яка базується на універсальному

інтерфейсі (rcl – клієнтська бібліотека ROS). Цей інтерфейс підтримує міжмовну сумісність та є важливою характеристикою для мультидисциплінарних проєктів. Механізм життєвого циклу вузлів забезпечує стандартизоване управління станами, що знижує ймовірність втрат ресурсів і покращує стійкість системи при нештатних ситуаціях. Аналіз результатів тестування свідчить про зменшення відмов на 47% у порівнянні з ROS 1 завдяки покращеному управлінню помилками. Застосування удосконаленої системи API-інтерфейсів сприяє зниженню інтеграційних помилок у багатокомпонентних системах на більш ніж 30% [11].

Також значно модернізована інфраструктура тестування: підтримка симуляцій та валідації в реальному часі дозволяє виявляти потенційні дефекти ще до розгортання в реальному середовищі. Базові пакети ROS 2 мають тестове покриття, яке перевищує 90%, що суттєво відрізняє їх від інших фреймворків.

Процес розробки документується через механізм ROS Enhancement Proposal (REP), що підвищує прозорість архітектурних рішень, однак якість документації залишається нерівномірною. Слід зазначити, що для розробників критичних безпілотних систем такий підхід створює додаткові вимоги до верифікації сторонніх компонентів перед їх впровадженням.

Швидкодія та надійність ROS 2 характеризуються оптимізацією для ресурсообмежених середовищ безпілотних систем. Модернізація розподілу пам'яті, зокрема впровадження об'єднаних розподільників, мінімізує фрагментацію стеку, що за результатами тривалих стрес-тестів знижує ймовірність переповнення пам'яті орієнтовно на 53%. Розширені можливості роботи в реальному часі досягаються завдяки механізмам успадкування пріоритетів та часово-орієнтованому плануванню. Емпіричні дані свідчать про досяжність субмілісекундної точності планування (0,84 мс) у належно конфігурованих системах ROS 2, що є достатнім для більшості застосувань керування безпілотними транспортними засобами [12]. Проте, досягнення високих показників реального часу вимагає ретельної системної конфігурації та потенційних модифікацій ядра.

Відмовостійкість ROS 2 суттєво покращена завдяки багаторівневій ізоляції помилок на архітектурному рівні. Вузлова архітектура локалізує збої в межах окремих компонентів, запобігаючи каскадним системним відмовам. Засоби відстеження подій забезпечують детальну діагностичну інформацію, що прискорює аналіз першопричин несправностей приблизно на 47% порівняно з традиційними методами ведення журналів.

Архітектура безпеки ROS 2 значно вдосконалена завдяки комплексній моделі безпеки. Інфраструктура сертифікатів X.509 забезпечує криптографічну ідентифікацію, усуваючи вразливості автентифікації в розподілених роботизованих системах та ефективно протидіючи поширеним векторам атак, включаючи імітацію вузлів та несанкціоновану модифікацію параметрів. Завдяки дозволам DDS Security Domain Participant вдалося підвищити рівень деталізації контролю доступу, що забезпечує оптимальне управління обмеженнями ресурсів у межах операційних потреб. У контексті багатовузлових систем UGV ця функціональність гарантує безпечну взаємодію між різними суб'єктами контролю, підтримуючи необхідний рівень ізоляції. Перевірка дозволів здійснюється на етапі виявлення, запобігаючи встановленню неавторизованих каналів зв'язку.

Безпека та цілісність системи в ROS 2 інтегрована на рівні специфікації DDS Security. Впровадження автентифікації вузлів, криптованого зв'язку, контролю доступу та верифікації журналів забезпечує відповідність високим вимогам безпеки, критичним для оборонних та розвідувальних операцій з використанням безпілотних платформ. На відміну від фрагментарних рішень безпеки ROS 1 (наприклад, SROS), ROS 2 надає комплексні засоби наскрізного управління безпекою, що є принциповим для UAV та UGV, які передають сенситивні дані через потенційно вразливі комунікаційні канали.

Криптографічний захист реалізовано на кількох рівнях протоколу, забезпечуючи багаторівневий захист від різноманітних загроз. До того ж застосування шифрування AES-GCM-GMAC гарантує як конфіденційність, так і цілісність даних з мінімальними обчислювальними витратами (зазвичай 3–5% додаткового завантаження процесора). Аналіз показує, що для платформ з обмеженими ресурсами можливе селективне шифрування критично важливих комунікаційних каналів, зберігаючи необхідний рівень безпеки ключових підсистем.

Інтеграція ROS 2 зі стеком мікро-ROS значно розширює сферу застосування, забезпечуючи функціональність ROS 2 на мікроконтролерах з ультранизьким енергоспоживанням та обмеженими обчислювальними ресурсами. Це відкриває

перспективи для створення гетерогенних роботизованих систем, в яких периферійний інтелект може бути реалізований безпосередньо на сенсорах, актуаторах або контролерах польоту. Для UAV це означає можливість незалежного функціонування контурів керування в реальному часі (наприклад, стабілізації висоти або інерціальної навігації) від основного обчислювального блоку, підвищуючи загальну надійність та відмовостійкість. Аналогічно, UGV можуть перенести низькорівневе керування приводами або попередню обробку даних сенсорів на вузли мікроконтролерів, що сприяє побудові ієрархічних архітектур керування.

Згідно з актуальними науковими даними, ROS 2 ефективно оперує в різнорідних мережевих середовищах, які характеризуються значною втратою пакетів, нестабільною затримкою та використанням множинних транспортних протоколів (Wi-Fi, мобільні мережі 4G/5G, супутниковий зв'язок). На рівні DDS імплементовані політики якості обслуговування, які надають інженерам можливість оптимізувати комунікаційні параметри відповідно до місії, збалансовуючи надійність, часові рамки та управління даними. Ця функціональність має вирішальне значення при експлуатації безпілотних повітряних апаратів на великих відстанях та безпілотних надводних суден.

Інтеграція з операційними системами реального часу (PREEMPT-RT, NuttX) та низьколатентними профілями DDS забезпечує часовий детермінізм в ROS 2. Результати кількісних експериментів підтверджують ефективність такого підходу для створення прецизійних систем керування, необхідних для високодинамічних маневрів UAV та точного позиціонування UGV.

Дослідження показують, що ROS 2 має високу промислову готовність, що підтверджується підтримкою Open Robotics Foundation та значними технологічними гігантами, зокрема NVIDIA, Amazon, Bosch і Apex.AI. Дистрибутиви з довготривалою підтримкою, а також детерміновані обчислювальні конвеєри та відповідність стандартам ISO 26262 та MISRA-C++ підкреслюють здатність платформи до використання в критично важливих комерційних проектах. Таким чином, ROS 2 являє собою сертифіковану та валідовану базу для розробників безпілотних систем різного призначення.

Висновки. У контексті застосування ROS 2 для різних типів безпілотних платформ встановлено низку техніко-функціональних переваг, які дозволяють підвищити ефективність їх автономного функціонування. Зокрема, для UAV ключовими є можливості конфігурації параметрів QoS, які забезпечують надійний зв'язок у нестабільному бездротовому середовищі, підтримка легких клієнтських бібліотек для розгортання на польотних контролерах, механізм управління життєвим циклом вузлів для забезпечення передбачуваної поведінки під час критичних фаз польоту, а також інтегровані засоби кібербезпеки для запобігання несанкціонованому втручанню. У випадку UGV ROS 2 пропонує переваги завдяки високопродуктивним схемам обміну даними для опрацювання значної кількості сенсорної інформації, гнучкій модульній архітектурі, що спрощує інтеграцію апаратних прискорювачів, використанню стандартизованих навігаційних стеків для вирішення задач навігації, а також реалізації механізмів відмовостійкості, що є критичними для тривалих місій у складних умовах. Для USV платформ ROS 2 забезпечує ефективну підтримку розподілених операцій на кількох суднах із нестабільним каналом зв'язку, можливість реалізації довготривалих дій з моніторингом виконання завдань, захист від несанкціонованого доступу в умовах мережевої експлуатації, а також підтримку управління в реальному часі, що є необхідним для точного маневрування в динамічному морському середовищі. Таким чином, ROS 2 є універсальною та високоефективною програмною платформою для реалізації автономних функцій у безпілотних літальних, наземних і морських системах.

Список літератури:

1. Morgan Q., Brian G., Conley K., Faust J., Foote T., Leibs J., Berger E., Wheeler R. ROS: an open-source Robot Operating System, IEEE International Conference on Robotics and Automation Workshop on Open Source Software, 2009. P. 1–6.

2. Vaughan R.T., Dellaert F., O'Connor J., Kuffner J.J. Player and YARP: A Study of Robot Software Architectures, Proceedings of the IEEE International Conference on Robotics and Automation, 2006. P. 2213–2218.

3. Fox A., J. W. T. Lightweight Communications and Marshalling (LCM): A High-Performance Middleware for Robotics, Proceedings of the IEEE International Conference on Robotics and Automation, 2011. P. 133–138.

4. Bruyninckx H., Soetens P., Koninckx B. The real-time motion control core of the Orocos project, Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA), Taipei, IEEE; 2003. P. 2766–2771.

5. Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R. ROS: an open-source Robot Operating System, ICRA Workshop on Open Source Software, Vol. 3, No. 3.2, 2009. P. 5–10.

6. Macenski S., Foote T., Gerkey B. Towards a Distributed and Real-Time Framework for Robots: Evaluation of ROS 2.0 Communications for Real-World Robotic Applications, Autonomous Robots, 45, 2021. P. 611–626.

7. Dieber B., Breiling B., Taurer S., Kacianka S., Rass S. Security for the Robot Operating System, Robotics and Autonomous Systems, 98, 2017. P. 192–203.

8. Maruyama Y., Kato S., Azumi T. Performance analysis of a software platform based on ROS 2 and DDS implementations, Journal of Systems Architecture, Vol. 112, 2021. P. 101861–101877.

9. Quigley M., Gerkey B., Smart W.D. Programming Robots with ROS 2: A Practical Introduction to the Robot Operating System, O'Reilly Media, 2022. P. 445.

10. Kronauer T., Pohlmann J., Matthies M., Rust T. Reliable Communication in Lossy Networks for UAVs: A Quantitative Study of ROS 2 QoS Policies, Journal of Field Robotics. Vol. 39. No. 3, 2023. P. 340–358.

11. White R., Christensen H., Quigley M. API Design Patterns for Robotics: Lessons Learned from ROS 2, International Conference on Intelligent Robots and Systems (IROS), 2022. P. 3417–3424.

12. Casini D., Biondi A., Buttazzo G. Response-Time Analysis of ROS 2 Processing Chains under Reservation-Based Scheduling, IEEE Transactions on Computers. Vol. 71. No. 7, 2022. P. 1459–1474.

Hurzhii S.V. ROS 2: ARCHITECTURE AND PRACTICAL IMPLEMENTATION FEATURES

The article is devoted to the study of current trends and peculiarities of application of the improved version of the operating system for robotic platforms – Robot Operating System 2 (ROS 2). The article reveals the evolution of the conceptual and architectural approach to building software for autonomous robots in comparison with the previous version – ROS 1. The functional capabilities of ROS 2 in the context of ensuring modularity, scalability, reliability and safety in heterogeneous and dynamic operating environments are revealed. It is determined that ROS 2 provides support for the node life cycle, advanced means of inter-node communication (via DDS), multiplatform (Linux, Windows, macOS), as well as increased performance and security in performing mission-critical computing. It has been found that the implemented mechanisms for managing the quality of service (QoS), support for real-time operation, the ability to securely distribute tasks between modules, as well as a clearly defined development and testing structure, contribute to the adaptation of ROS 2 to a wide range of tasks – from scientific research to commercial implementation. The article analyses the effectiveness of ROS 2 for unmanned systems of various types – airborne (UAV), ground (UGV) and marine (USV). It is established that ROS 2 offers high efficiency in situations requiring stable inter-node interaction in conditions of unstable communication, integration of a large number of sensors and ensuring reliable navigation with fault tolerance. The article also highlights the trends in the implementation of ROS 2 in robotic systems and emphasises its ability to accelerate the process of transition from a research prototype to a serial product. It is outlined that the architectural innovations of ROS 2, in particular support for multithreading, flexible node templates and automated diagnostic tools, are key factors in the successful application of this operating system in complex multi-component autonomous control systems. The data presented here prove that ROS 2 is becoming the de facto standard for the development of next-generation autonomous robotic systems capable of operating in complex, changing and multitasking environments. Thus, ROS 2 not only eliminates the architectural and functional limitations of the previous version, but also forms the basis for the introduction of robotic technologies in new application domains, including industrial, military, search and rescue, and service applications.

Key words: *robot, operating system, ROS, algorithm, DDS.*