КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (28), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.28.824
УДК 004.896

Ганенко Людмила Дмитрівна
викладач кафедри інформатики, програмування,
штучного інтелекту та технологічної освіти,
Центральноукраїнський державний університет ім. В. Винниченка,
Кропивницький, Україна
ORCID ID: 0000-0003-2219-8196
hanenkoliudmyla@gmail.com

Жебка Вікторія Вікторівна
доктор технічних наук, доцент,
завідувач кафедри технологій цифрового розвитку,
Державний університет інформаційно-комунікаційних технологій,
Київ, Україна
ORCID ID: 0000-0003-4051-1190
viktoria_zhebka@ukr.net

СТВОРЕННЯ НАВІГАЦІЙНОЇ СИСТЕМИ АВТОНОМНОГО МОБІЛЬНОГО РОБОТА ЗАСОБАМИ ROS 2

Анотація. Робототехнічні системи активно впроваджуються у різні сфери життя людини. Успішна реалізація таких систем значною мірою залежить від вибору надійної програмної платформи. Robot Operating System 2 (ROS 2) є сучасною платформою для розробки, тестування та впровадження робототехнічних систем. В статті проведено порівняльний аналіз Robot Operating System (ROS) та її оновленої версії ROS 2. Проаналізовано архітектурні зміни, включаючи використання DDS (Data Distribution Service) для забезпечення розподіленої комунікації. Окрему увагу приділено аспектам підвищеної продуктивності, модульності та безпеки, які були вдосконалені в ROS 2. З метою практичного використання ROS 2 в роботі створено автономну систему навігації мобільного робота. Використовуючи симуляційне середовище Gazebo, створено модель середовища із статичними перешкодами. Карту змодельованого середовища згенеровано із використанням пакету Cartografer, який дозволяє створювати двовимірні карти на основі сенсорних даних. Для реалізації навігації робота використано пакет Nav2. Даний пакет підтримує інтеграцію з різними типами сенсорів (LiDAR, камери, IMU) та конфігурується завдяки YAML-файлам. Глобальне планування маршруту здійснено за допомогою алгоритму Дейкстри із використанням плагіну Navfn Planner. Під час тестування було використано платформу мобільного робота TurtleBot3 Waffle, яка була створена для проведення експериментів у сфері робототехніки з використанням ROS/ROS 2. Результати дослідження продемонстрували, що ROS 2 є ефективним фреймворком, який інтегрує всі необхідні інструменти для розробки навігаційної системи автономного мобільного робота. ROS 2 забезпечує комплексну взаємодію сенсорів, алгоритмів планування маршруту, локалізації, управління рухом та уникнення перешкод. Дослідження підтвердило доцільність використання симулятора Gazebo для попереднього тестування алгоритмів навігації перед їх впровадженням на реальному обладнанні. Створена навігаційна система може мати широкий спектр застосувань у таких галузях, як промислова автоматизація та сервісна робототехніка.

Ключові слова: автономний мобільний робот; навігаційна система; ROS; ROS 2; Gazebo; Rviz.

ВСТУП

Мобільні роботи активно впроваджуються у різні сфери життя людини: здоров’я, промисловість, логістику, сільське господарство, охорону та побут. Особливе місце в наукових дослідженнях займають автономні мобільні роботи (АМР), які здатні

виконувати завдання навігації у складних середовищах. Забезпечення ефективної, точної та безпечної навігації АМР є одним з ключових завдань робототехніки.

Постановка проблеми. Robot Operating System 2 (ROS 2) є сучасною платформою для розробки, тестування та впровадження робототехнічних систем. Завдяки модульній архітектурі, покращеній продуктивності та можливостям роботи в реальному часі, ROS 2 стала стандартом для створення складних автономних систем. Інструменти ROS 2 дозволяють створювати та тестувати алгоритми навігації, забезпечуючи інтеграцію із сенсорами.

Актуальність створення навігаційних систем автономних мобільних роботів обумовлена потребою в автоматизації процесів у середовищах та зростанням складності задач навігації. У контексті зростаючого попиту на автономні системи, розробка навігаційної системи АМР на основі ROS 2 є важливим кроком до створення ефективних робототехнічних рішень.

Аналіз останніх досліджень і публікацій. Навігаційна система АМР повинна забезпечувати високу точність орієнтації, безпечне переміщення у статичному та динамічному середовищах та ефективну інтеграцію апаратного забезпечення. Розвиток ROS 2 створив нові можливості для адаптивного управління роботами.

У роботі [1] Мілер Я. та співавтори використали ROS 2 для розробки автономної системи розвідки шахт чотириногими роботами. Система містить планування руху, картографування місцевості та локалізацію. Натомість компанія Mission Robotics, яка займається розробкою морських роботів, використала ROS 2 для інтегрування нових датчиків. Компанія Auterion [2] обрала ROS 2 для інтеграції функціональних можливостей вищого рівня в системи дронів. Auterion використовувала симуляцію в Gazebo для проведення тестів програмного забезпечення перед тестуванням апаратного забезпечення. Місія NASA VIPER (Volatiles Investigating Polar Exploration Rover) [3] при розробці місяцеходу VIPER використовувала обчислювальні модулі засновані на ROS 2 і Gazebo. Дані моніторингу надсилались в мережу ROS 2, де група вузлів здійснювала їх обробку. Вузли перетворювали зображення в хмари точок, об'єднували дані для уточнення позиціонування.

Аналіз наукових досліджень демонструє, що сучасні методи та інструменти для моделювання навігаційних систем, зокрема ROS 2, є надійним базисом для вирішення складних задач автономної навігації і залишається перспективним напрямом для подальших досліджень.

Мета статті. Метою даної статті є аналіз етапів створення навігаційної системи автономного мобільного робота та їх практична реалізація в системі ROS 2 із забезпеченням надійної локалізації АМР, побудовою маршруту руху та уникненням статичних перешкод.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Основою функціональності АМР є навігаційна система, яка забезпечує можливість побудови карти, локалізації, планування маршруту, уникнення статичних та динамічних перешкод у реальному часі (глобальна та локальна навігація) [4].

Архітектуру навігаційної системи автономного мобільного робота представлено на рис. 1.

ISSN 2663 - 4023

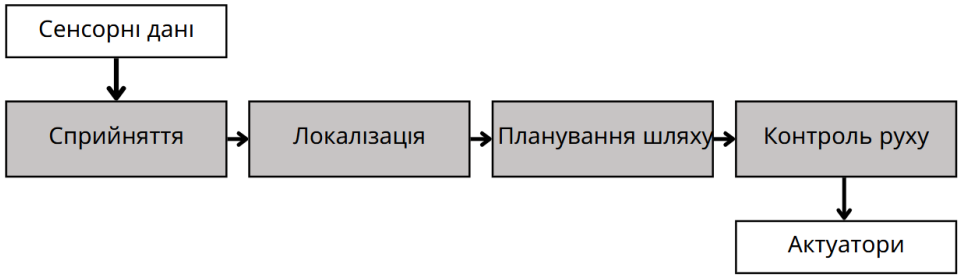

Рис. 1. Архітектура навігаційної системи АМР

Розробка сучасних навігаційних систем неможлива без використання програмних інструментів. Одним із найбільш популярних та гнучких інструментів для створення роботизованих систем є Robot Operating System (ROS)

Порівняльний аналіз функціональних можливостей ROS і ROS 2. ROS — це програмне середовище для розробки, керування та тестування робототехнічних систем. Даний фреймворк забезпечує модульну архітектуру, яка дозволяє адаптувати робототехнічні системи до різних завдань, незалежно від складності проєкту. ROS має можливість тестування в симульованих середовищах, таких як Gazebo, що дозволяє значно зменшити витрати на розробку та пришвидшити тестування алгоритмів.

Розроблено велику кількість пакетів та бібліотек, використовуючи які розробники можуть вирішувати задачі навігації, побудови карт, обробки сенсорних даних та планування руху. ROS має активну спільноту розробників та підтримку великої кількості робототехнічних платформ. Фреймворк надає можливість використовувати різні комунікаційні елементи (повідомлення, теми, сервіси та дії) і налаштовувати їх відповідно до потреб розробника.

Появу ROS пов’язують із створенням у 2006 році фреймворку Stanford Personal Robotics Program Еріком Бергером та Кінаном Віробеком. Реалізація була зумовлена необхідністю вирішення проблеми тривалого повторного впровадження раніше реалізованих програм і алгоритмів в інфраструктуру нових роботів. Базуючись на цій ідеї дослідники Willow Garage у 2008 році розробили ROS. В 2011 році розробники випустили першого робота під назвою Turtlebot, який дозволяв вивчати основи робототехніки за допомогою ROS.

Ключовими елементами ROS є вузли та пакети. Система ROS містить вузли з унікальними назвами. Вузли керують поведінкою робота: планування руху, сенсорне сприйняття або керування приводом. ROS Master — головний вузол і жодна система ROS не працює без його запуску.

Пакети ROS є основним механізмом для розбиття коду на узгоджені одиниці. Кожен пакет чітко визначає свої залежності від інших пакетів. Така модульна структура групує пов’язані ресурси (вузли, конфігураційні файли, набори даних і бібліотеки) та полегшує співпрацю між розробниками.

Основними механізмами зв’язку між вузлами є теми, сервіси та дії.

Теми в ROS є асинхронним однонаправленим каналом зв’язку, через який вузли обмінюються повідомленнями. Кожна тема пов’язана з певним типом повідомлення. Теми дозволяють визначеним вузлам публікувати повідомлення, а іншим вузлам — підписуватися на їх отримання. Зазвичай теми використовуються для передачі даних у реальному часі.

Сервіси забезпечують синхронний двонаправлений зв’язок. Цей режим взаємодії дозволяє клієнтському вузлу робити запит до вузла-сервера, який після обробки запиту

повертає відповідь клієнту. Сервіси призначені для більш тісної взаємодії між вузлами (керування пристроями, доступ до спільних даних).

Дії є асинхронним механізмом зв'язку для виконання складних операцій і моніторингу їх виконання в реальному часі. Клієнтський вузол може надсилати запит на ініціювання дії та отримувати періодичний зворотний зв'язок щодо статусу виконання. Дії також підтримують механізм видалення запиту, тому їх застосовують при обробці складних послідовностей дій і реагування на непередбачені події.

Незважаючи на те, що система ROS успішно реалізовувала складні завдання робототехніки, їй бракувало кількох аспектів, таких як вимоги до виконання в реальному часі, підтримка мультиагентних систем, вбудовані механізми безпеки. Остання версія ROS, яка отримала назву Noetic, була випущена в травні 2020 року.

У 2018 році з'явилась ROS 2 – оновлена версія ROS. Однією з найбільш важливих змін у ROS 2 є перехід від мережевого протоколу TCP/UDP до стандарту проміжного програмного забезпечення для зв'язку між вузлами (DDS).

DDS — це стандарт, опублікований Object Management Group (OMG), який визначає проміжне програмне забезпечення для надійного розподілу даних у реальному часі відповідно до парадигми публікації/підписки. Використання DDS надало можливість ROS 2 відповідати вимогам безпеки розподілених систем [5].

На відміну від централізованого підходу ROS, ROS 2 усуває потребу в головному вузлі, тим самим вирішуючи проблему єдиної точки відмови та проблему масштабованості. Вузли ROS 2 взаємодіють безпосередньо один з одним за допомогою DDS. Така децентралізована структура може успішно використовуватись для розробки мультиагентних систем і забезпечувати більш ефективний і надійний зв'язок у великомасштабних розподілених роботизованих системах [6].

Порівняння основних характеристик ROS і ROS 2 подано в табл. 1.

Таблиця 1

Порівняння ROS і ROS 2

	ROS	**ROS 2**
Операційна система	Linux	Linux, Windows, MacOS, RTOS
Архітектура	централізована, вузли взаємодіють через ROS Master	децентралізована, кожен вузол може взаємодіяти напряму через DDS, що зменшує затримки та підвищує надійність системи
Комунікація	використовує протоколи ROSTCP/ROSUDP для передачі даних між вузлами; Nodelet API забезпечує запуск кількох алгоритмів в одному процесі без витрат на копіювання	використовує стандарт DDS для комунікації між вузлами; Intra-process API дозволяє вузлам напряму взаємодіяти між собою без використання зовнішнього мережевого інтерфейсу

ROS 2 дозволяє візуалізувати дані за допомогою текстових інструментів (CLI), графічних інтерфейсів (rqt) та 3D-візуалізаторів (RViz).

Таким чином, можна виокремити такі переваги ROS 2 [7], порівняно з ROS:

– система побудована на основі DDS, що дозволяє реалізовувати надійну комунікацію без використання єдиного головного вузла та дає можливість налаштування засобів шифрування, аутентифікації та контролю доступу, тим самим робить ROS 2 придатним для застосувань із високими вимогами безпеки;

– завдяки новій архітектурі ROS 2 підтримує широкомасштабні розподілені системи та забезпечує стабільну роботу у великих робототехнічних проєктах;
– розроблено пакет Nav2, який забезпечує більшу гнучкість завдяки використанню дерев поведінки (BT);
– має підвищену продуктивність у реальному часі.

Навігація AMP в ROS 2. Навігація в робототехніці — це здатність робота локалізувати себе в середовищі, планувати шлях до цільової точки та рухатись відповідно до розробленого маршруту.

Навігаційний стек (Navigation Stack) є одним із інструментів ROS, який контролює швидкість робота, виконуючи замкнутий цикл для переміщення робота до цілі у середовищі. Він отримує інформацію одометрії (положення, швидкість та орієнтацію) і координати цільової точки, а потім генерує контрольні виходи для безпечної навігації.

В ROS 2 автономна навігація реалізується через інтеграцію програмних і апаратних компонентів для самостійного переміщення у середовищі, уникнення перешкод та досягнення заданих цілей роботом. Основним інструментом є пакет Nav2. Nav2 — це розширена версія навігаційного стеку ROS. Nav2 генерує план маршруту, якщо надано поточну та цільову позиції і карту.

Nav2 використовує дерева поведінки (BT), які замінили FMS навігаційного стеку ROS. Головними перевагами використання BT є здатність системи швидко адаптуватися до змін у середовищі та можливість розбивати складні системи на окремі незалежні частини, які можна легко розробляти, тестувати й повторно використовувати [8].

BT Navigator містить дерево поведінки реалізації навігації AMP. Він приймає та обробляє запити. Кожне дерево поведінки моделюється як файл XML.

Nav2 дозволяє реалізувати навігацію AMP за допомогою оркестровки багатьох незалежних модульних серверів. Кожен із серверів повертає індикатори стану назад до BT Navigator. Сервер містить модулі алгоритму, що динамічно завантажуються під час виконання [9**Error! Reference source not found.**].

Вхідними даними для Nav2 є перетворення TF, що відповідають REP-105, карта, файл BT XML і дані датчиків. Виходами є команди швидкості для двигунів робота.

На рис. 2 зображено архітектуру Nav2.

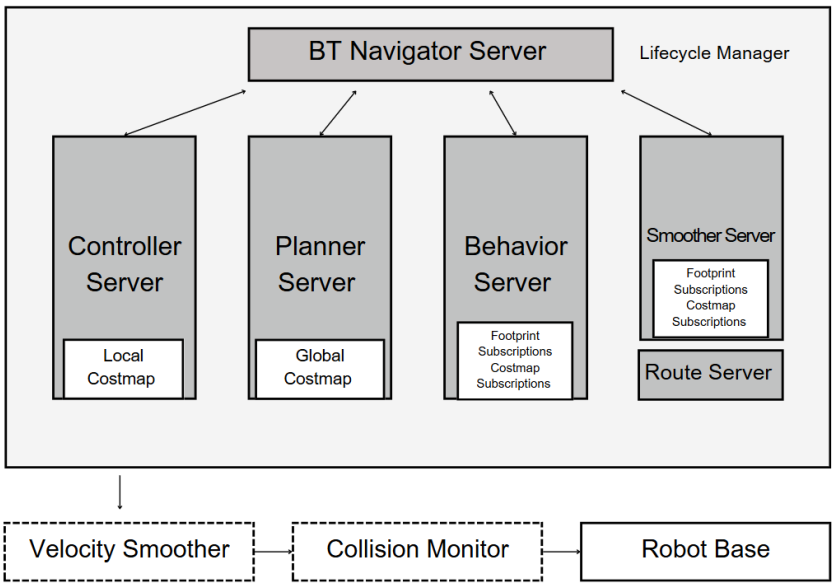

Рис. 2. Структура Nav 2

BT Navigator Server — сервер, який використовує Behavior Trees (BT) для координації та управління різними модулями навігаційної системи.

Controller Server контролює локальний рух робота, забезпечуючи дотримання заданого маршруту, використовує Local Costmap для розрахунку локального плану з урахуванням наявних перешкод, виконує команди точного переміщення робота в межах обраної траєкторії.

Planner Server відповідає за глобальне планування маршруту між початковою і кінцевою точками, використовує Global Costmap для обчислення оптимального шляху з урахуванням статичної карти середовища, забезпечує вибір найкращого шляху.

Behavior Server реалізує шаблони поведінки робота під час виконання конкретних задач, використовує Costmap Subscriptions і Footprint Subscriptions для отримання даних про поточний стан середовища, реалізує складну поведінку, таку як уникнення зіткнень, повторний запуск маршруту або очікування.

Smoother Server забезпечує згладжування траєкторії для плавного руху робота.

Route Server управляє передачею маршрутів між різними модулями, підтримує обмін інформацією між Planner, Smoother та іншими модулями.

Lifecycle Manager контролює життєвий цикл усіх серверів та модулів у системі Nav2.

Velocity Smoother плавно змінює швидкість робота для уникнення різких прискорень чи уповільнень.

Collision Monitor постійно перевіряє ризик зіткнення робота з перешкодами, у разі небезпеки ініціює зупинку або коригування траєкторії.

Robot Base безпосередньо управляє апаратною платформою робота, виконує низькорівневі команди, такі як обертання моторів або отримання даних сенсорів.

Nav2 містить методи глобального планування на основі пошуку, а саме, це Navigation Function, 2D-A*, Theta*, Hybrid-A* і State Lattice. Усі методи розглядають діапазон значень ризику на карті витрат [10**Error! Reference source not found.**].

Після того, як глобальний планувальник шляху визначає маршрут в середовищі, локальний планувальник генерує траєкторії без зіткнень з перешкодами. Nav2 містить такі реалізацій локальних планувальників, як Dynamic Window Approach (DWA), Regulated Pure Pursuit (RPP), Model Predictive Path Integral (MPPI) та Rotation Shim. Крім локальних планувальників, які є частиною проєкту Nav2, розроблено планувальники, які доступні у ширшій екосистемі ROS 2, такі як Timed Elastic-Band (TEB) і Graceful Controller [**Error! Reference source not found.**0].

Архітектура Nav2 складається з кількох вузлів життєвого циклу, які незалежно налаштовуються за допомогою файлу YAML [11].

Одним із ключових елементів автономної навігації мобільних роботів у ROS 2 є карта витрат (Costmap). Витрати кожної комірки є цілим числом у діапазоні [0, 255]. Алгоритми планування віддають перевагу шляхам з нижчими витратами та уникають шляхів із вищими витратами.

В ROS 2 карта витрат оновлюється та підтримується за допомогою Layered Costmap. Layered Costmap містить упорядкований список шарів, кожен із яких має свою функцію в представленні та обробці даних про середовище [**Error! Reference source not found.**].

Static Layer забезпечує інформацію про статичні перешкоди, отримані заздалегідь із карти середовища.

Obstacle Layer оновлює карту на основі даних сенсорів. Використовує алгоритм трасування променів Bresenham для визначення перешкод та вільного простору. Дані про перешкоди зберігаються у двовимірній сітці.

Voxel Layer представляє тривимірні перешкоди у вигляді вокселів, які проєктуються на двовимірну карту. Шар моделює середовище з використанням воксельної сітки, підтримує як постійні (Persistent), так і непостійні (Non-Persistent) вокселі.

Non-Persistent Voxel Layer обчислює значення аналогічно воксельному шару, але не зберігає дані між оновленнями.

Spatio-temporal Voxel Layer (STVL) використовує розріджену воксельну сітку (OpenVDB) для моделювання тривимірного середовища в динамічних умовах. Застосовує метод прискорення затухання для видалення неактуальних перешкод.

Range Layer опрацьовує дані від сенсорів, використовує ймовірнісну модель для визначення зайнятості клітин, обробляє як фіксовані, так і змінні значення від сенсорів.

Inflation Layer створює область безпеки навколо перешкод за допомогою функції затухання. Шар використовується для розрахунку конфігураційного простору робота та забезпечує оптимізацію перевірки на зіткнення.

Keepout Layer застосовує маски для позначення зон, які робот повинен уникати, дозволяє визначати області з різним рівнем витрат для навігації.

Speed Layer визначає максимальну швидкість робота в певних зонах, використовується для уповільнення робота при вході або виході з критичних зон.

Binary Layer використовується для вбудовування тригерів у просторові маски та для активації чи деактивації функцій на основі просторових обмежень.

Реалізація автономної навігації мобільного робота в ROS 2. Процес створення навігаційної системи включає наступні етапи [**Error! Reference source not found.**2]:

- створення карти навколишнього середовища;
- встановлення цільової позиції;
- планування маршруту (вибір оптимального шляху до цілі);
- управління локальним переміщенням;
- уникнення роботом небезпечних ділянок маршруту.

ROS 2 інтегрує компоненти, які забезпечують автономну навігацію робота в середовищі:

1. Сенсори (лідари, камери, ультразвукові сенсори, імпульсні датчики, IMU).
2. Локалізація (SLAM, AMCL, одометрія, методи фільтрації).
3. Планування маршруту (глобальне та локальне).
4. Управління рухом (контролери руху, механізми управління траєкторією).
5. Уникнення перешкод (алгоритми уникнення перешкод, сенсори для виявлення перешкод).
6. Візуалізація (RViz, RQt).
7. Запуск і налаштування (файли конфігурації для параметрів навігації, launch-файли).
8. Моніторинг і зворотний зв’язок (теми, сервіси, дії).

Для реалізації навігації AMP засобами Gazebo [**Error! Reference source not found.**3] нами було створено модель середовища із статичними перешкодами і збережено у my_new_gaz.world. Модель середовища зображено на рис. 3.

Рис.3. Модель середовища в Gazebo

Для тестування навігації в Gazebo та Rviz використано платформу мобільного робота TurtleBot3 Waffle. TurtleBot3 Waffle — це невеликий мобільний робот на базі ROS із набором вбудованих датчиків для локалізації та навігації. Модель даного робота представлено за допомогою уніфікованого формату Unified Robot Description Format (URDF). URDF — це специфікація XML для опису робота.

Для того, щоб мобільний робот міг рухатись в симуляційному середовищі необхідно створити карту середовища. Змодельоване середовище було нанесено на карту за допомогою пакету Cartographer. Cartographer — це пакет для SLAM (одночасної локалізації та картографування) у реальному часі. Алгоритм використовується для побудови двовимірної карти сітки зайнятості середовища. Із використанням сенсора Lidar створено точну 2D-карту внутрішнього середовища. Під час картографування переміщення робота в середовищі здійснювалось за допомогою пакету teleop_key, який дозволяє управляти роботом вручну за допомогою клавіатури. В результаті було отримано файл, який містить параметри карти у форматі .yaml і зображення карти у форматі .pgm. Візуалізація процесу створення карти в Rviz продемонстровано на рис. 4.

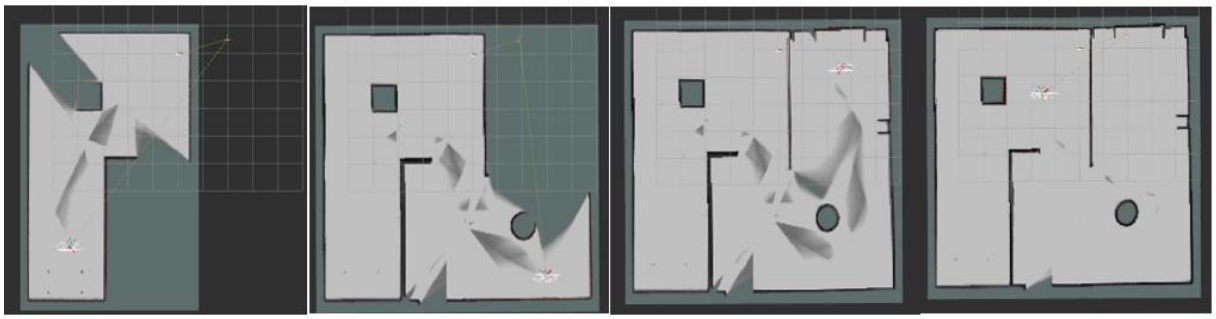

Рис. 4. Створення карти середовища в Rviz

Планування шляху робота включає глобальні та локальні планувальники. Глобальний планувальник використовується для створення найкоротшого шляху до мети, яка представлена червоною лінією (рис. 5). Глобальний планувальник знаходить

оптимальний шлях, маючи попередні знання про середовище та статичні перешкоди, локальний планувальник перераховує шлях, щоб уникнути динамічних перешкод. Після створення глобального плану локальний планувальник перетворює цей шлях у команди швидкості для двигунів робота. Основна мета підходу до локального планування — коригування плану в режимі онлайн.

Для реалізації автономної навігаційної системи використано пакет Nav2. Зокрема, застосовано плагін Navfn Planner, який для планування маршруту використовує алгоритм Дейкстри.

Цільову позицію робота задано за допомогою вузла nav.py, який було створено на мові Python. Далі AMP розраховує оптимальний маршрут до цільової позиції. На рис. 5 наведено серію зображень карт із згенерованим планом маршруту від початкового положення до цільового.

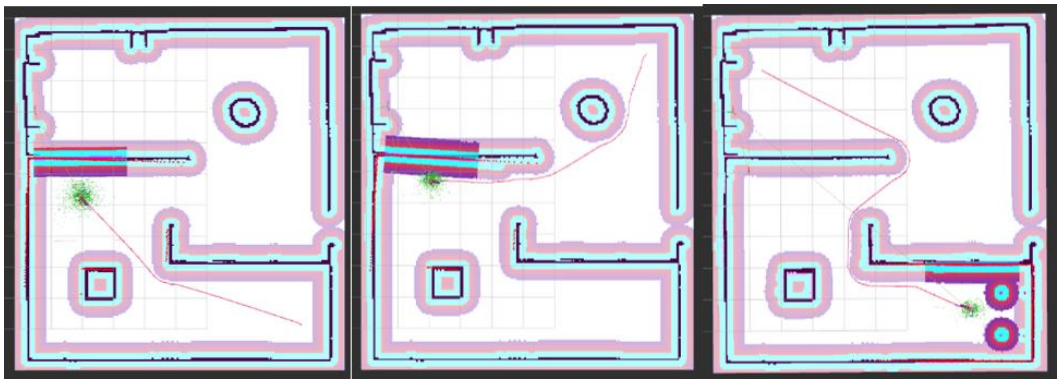

Рис. 5. Карти із запланованими маршрутами AMP

Отримавши команду переходити до цілі, робот рухається до цільової позиції, дотримуючись запланованого шляху та уникаючи зіткнень з перешкодами (рис. 6).

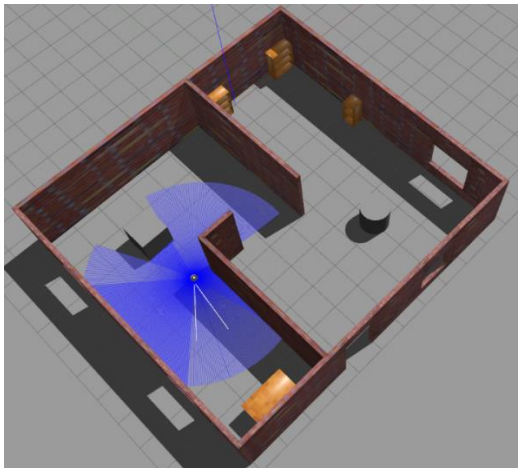

Рис. 6. Рух мобільного робота до цільової позиції

При виявленні перешкоди на шляху система контролю зупиняє навігацію та запускає модулі безпеки для запобігання зіткнень. Залежно від перешкоди робот може зупинитися, розвернутись або здійснити крок назад. Якщо перешкоди унеможливлюють створення шляху або координати цільової позиції знаходяться у недоступних місцях, процес навігації припиняється.

На рис. 7 наведено блок-схему реалізації навігації автономного мобільного робота.

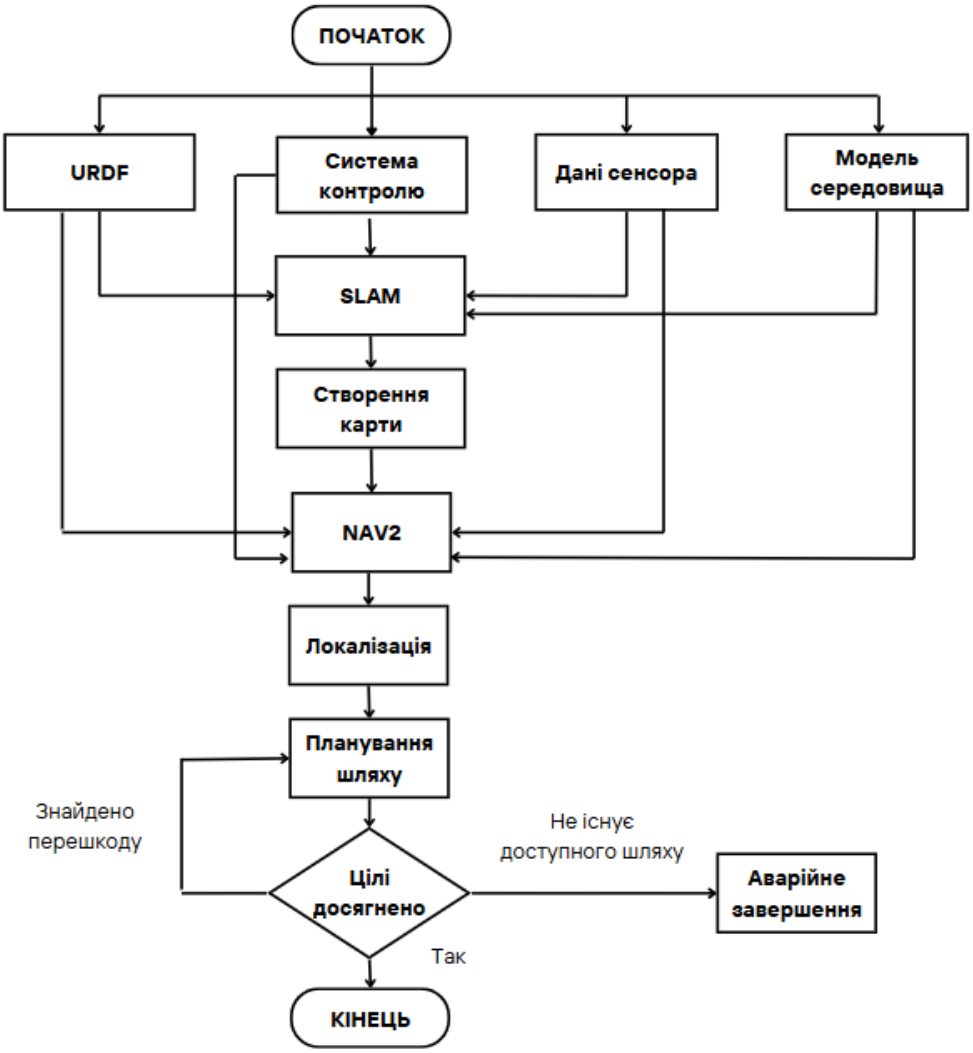

Рис. 7. Блок-схема навігації АМР

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

ROS 2 є ефективним фреймворком, який містить інструменти для розробки навігаційних систем автономного робота. Навігаційна система робота на основі ROS 2 забезпечує комплексну взаємодію сенсорів, алгоритмів планування маршруту, локалізації, управління рухом та уникнення перешкод. Всі ці компоненти працюють разом для забезпечення автономного переміщення робота в різних середовищах.

Створена навігаційна система може мати широкий спектр застосувань у таких галузях, як промислова автоматизація та сервісна робототехніка. Можливості автономної навігації також сприяють виконанню завдань у надзвичайних ситуаціях, коли людське втручання є обмеженим або ускладненим.

ROS 2 — проміжне програмне забезпечення, яке містить необхідні інструменти та функції для реалізації автономності роботів. Але, будучи відносно новим, ROS 2 все ще знаходиться на стадії розробки як з точки зору перенесення функціональних можливостей, раніше розроблених для ROS, так і щодо розробки та аналізу нових функцій.

Подальші дослідження можуть охоплювати розширення моделі середовища шляхом додавання динамічних об’єктів, що дозволить моделювати більш реалістичні сценарії функціонування робототехнічних систем та стане передумовою для впровадження алгоритмів планування маршруту, здатних адаптуватися до змін у середовищі в реальному часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Miller, I. D., Cladera, F., Cowley, A., Shivakumar, S. S., Lee, E. S., Jarin-Lipschitz, L., Bhat, A., Rodrigues, N., Zhou, A., Cohen, A., Kulkarni, A., Laney, J., Taylor, C. J., & Kumar, V. (2020). Mine Tunnel Exploration Using Multiple Quadrupedal Robots. *IEEE Robotics and Automation Letters, 5(2),* 2840–2847. https://doi.org/10.1109/lra.2020.2972872
2. Meier, L., Honegger, D., & Pollefeys, M. (2015). PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. *In 2015 IEEE International Conference on Robotics and Automation (ICRA).* IEEE. https://doi.org/10.1109/icra.2015.7140074
3. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics, 7(66).* https://doi.org/10.1126/scirobotics.abm6074
4. Zhang, T., Hu, X., Xiao, J., & Zhang, G. (2022). A survey of visual navigation: From geometry to embodied AI. *Engineering Applications of Artificial Intelligence, 114, 105036*. https://doi.org/10.1016/j.engappai.2022.105036
5. Bonci, A., Gaudeni, F., Giannini, M. C., & Longhi, S. (2023). Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey. *Applied Sciences, 13(23), 12796*. https://doi.org/10.3390/app132312796
6. Portugal, D., Rocha, R. P., & Castilho, J. P. (2024). Inquiring the robot operating system community on the state of adoption of the ROS 2 robotics middleware. *International Journal of Intelligent Robotics and Applications*. https://doi.org/10.1007/s41315-024-00393-4
7. Mazzeo, G., & Staffa, M. (2019). TROS: Protecting Humanoids ROS from Privileged Attackers. *International Journal of Social Robotics, 12(3),* 827–841. https://doi.org/10.1007/s12369-019-00581-4
8. Iovino, M., Scukins, E., Styrud, J., Ögren, P., & Smith, C. (2022). A survey of Behavior Trees in robotics and AI. *Robotics and Autonomous Systems, 104096.* https://doi.org/10.1016/j.robot.2022.104096
9. Macenski, S., Martin, F., White, R., & Clavero, J. G. (2020). The Marathon 2: A Navigation System. *In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. https://doi.org/10.1109/iros45743.2020.9341207
10. Macenski, S., Moore, T., Lu, D. V., Merzlyakov, A., & Ferguson, M. (2023b). From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems, 104493.* https://doi.org/10.1016/j.robot.2023.104493
11. Horelican, T. (2022). Utilizability of Navigation2/ROS2 in Highly Automated and Distributed Multi-Robotic Systems for Industrial Facilities. *IFAC-PapersOnLine, 55(4),* 109–114. https://doi.org/10.1016/j.ifacol.2022.06.018
12. M, A. A. (2020). Features of the Construction and Control of the Navigation System of a Mobile Robot. *International Journal of Emerging Trends in Engineering Research, 8(4),* 1445–1449. https://doi.org/10.30534/ijeter/2020/82842020
13. Martini, M., Eirale, A., Cerrato, S., & Chiaberge, M. (2023). PIC4rl-gym: a ROS2 Modular Framework for Robots Autonomous Navigation with Deep Reinforcement Learning. *In 2023 3rd International Conference on Computer, Control and Robotics (ICCCR).* IEEE. https://doi.org/10.1109/icccr56747.2023.10193996

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (28), 2025
ISSN 2663 - 4023

Liudmyla Hanenko
Lecturer, Department of Informatics, Programming,
Artificial Intelligence and Technological Education,
Volodymyr Vynnychenko Central Ukrainian State University,
Kropyvnytskyi, Ukraine
ORCID ID: 0000-0003-2219-8196
hanenkoliudmyla@gmail.com

Viktoriia Zhebka
Doctor of Technical Sciences, Associate Professor,
State University of Information and Communication Technologies,
Kyiv, Ukraine
ORCID ID: 0000-0003-4051-1190
viktoria_zhebka@ukr.net.com

DEVELOPMENT OF THE NAVIGATION SYSTEM OF AN AUTONOMOUS MOBILE ROBOT USING ROS 2

Abstract. Autonomous mobile robots are being actively implemented in various spheres of human life. The successful implementation of robotic systems functionality largely depends on the choice of a reliable software platform. Robot Operating System 2 (ROS 2) is a modern platform for the development, testing, and implementation of robotic systems. The article presents a comparative analysis of the Robot Operating System (ROS) and its updated version ROS 2. Architectural changes, including the use of DDS (Data Distribution Service) to provide distributed communication, are analyzed. Special attention is paid to the aspects of increased performance, modularity and security that have been improved in ROS 2. For the purpose of practical use of ROS 2, an autonomous navigation system for a mobile robot was created. Using the Gazebo simulation environment, a model of the environment with static obstacles was created. The map of the modeled environment was generated using the Cartografer package, which allows creating two-dimensional maps based on sensor data. The Nav2 package was used to implement the robot's navigation. This package supports integration with various types of sensors (LiDAR, cameras, IMU) and is easily configured using YAML files. Global route planning was performed using the Dijkstra algorithm with the Navfn Planner plug-in. During the testing, the TurtleBot3 Waffle mobile robot platform was used, developed for conducting experiments in the field of robotics using ROS/ROS 2. The results of the study demonstrated that ROS 2 is an effective framework that integrates all the necessary tools for developing an autonomous mobile robot navigation system. ROS 2 provides a comprehensive interaction of sensors, route planning, localization, motion control, and obstacle avoidance algorithms. The study confirmed the feasibility of using the Gazebo simulator for preliminary testing of navigation algorithms before their implementation on real equipment. The created navigation system can have a wide range of applications in such industries as industrial automation and service robotics.

Keywords: autonomous mobile robot; navigation system; ROS; ROS 2; Gazebo; Rviz.

REFERENCES (TRANSLATED AND TRANSLITERATED)

14. Miller, I. D., Cladera, F., Cowley, A., Shivakumar, S. S., Lee, E. S., Jarin-Lipschitz, L., Bhat, A., Rodrigues, N., Zhou, A., Cohen, A., Kulkarni, A., Laney, J., Taylor, C. J., & Kumar, V. (2020). Mine Tunnel Exploration Using Multiple Quadrupedal Robots. *IEEE Robotics and Automation Letters, 5(2),* 2840–2847. https://doi.org/10.1109/lra.2020.2972872
15. Meier, L., Honegger, D., & Pollefeys, M. (2015). PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. *In 2015 IEEE International Conference on Robotics and Automation (ICRA).* IEEE. https://doi.org/10.1109/icra.2015.7140074
16. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics, 7(66).* https://doi.org/10.1126/scirobotics.abm6074

17. Zhang, T., Hu, X., Xiao, J., & Zhang, G. (2022). A survey of visual navigation: From geometry to embodied AI. *Engineering Applications of Artificial Intelligence, 114, 105036*. https://doi.org/10.1016/j.engappai.2022.105036
18. Bonci, A., Gaudeni, F., Giannini, M. C., & Longhi, S. (2023). Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey. *Applied Sciences, 13(23), 12796*. https://doi.org/10.3390/app132312796
19. Portugal, D., Rocha, R. P., & Castilho, J. P. (2024). Inquiring the robot operating system community on the state of adoption of the ROS 2 robotics middleware. *International Journal of Intelligent Robotics and Applications*. https://doi.org/10.1007/s41315-024-00393-4
20. Mazzeo, G., & Staffa, M. (2019). TROS: Protecting Humanoids ROS from Privileged Attackers. *International Journal of Social Robotics, 12(3),* 827–841. https://doi.org/10.1007/s12369-019-00581-4
21. Iovino, M., Scukins, E., Styrud, J., Ögren, P., & Smith, C. (2022). A survey of Behavior Trees in robotics and AI. *Robotics and Autonomous Systems, 104096*. https://doi.org/10.1016/j.robot.2022.104096
22. Macenski, S., Martin, F., White, R., & Clavero, J. G. (2020). The Marathon 2: A Navigation System. *In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. https://doi.org/10.1109/iros45743.2020.9341207
23. Macenski, S., Moore, T., Lu, D. V., Merzlyakov, A., & Ferguson, M. (2023b). From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems, 104493*. https://doi.org/10.1016/j.robot.2023.104493
24. Horelican, T. (2022). Utilizability of Navigation2/ROS2 in Highly Automated and Distributed Multi-Robotic Systems for Industrial Facilities. *IFAC-PapersOnLine, 55(4),* 109–114. https://doi.org/10.1016/j.ifacol.2022.06.018
25. M, A. A. (2020). Features of the Construction and Control of the Navigation System of a Mobile Robot. *International Journal of Emerging Trends in Engineering Research, 8(4),* 1445–1449. https://doi.org/10.30534/ijeter/2020/82842020
26. Martini, M., Eirale, A., Cerrato, S., & Chiaberge, M. (2023). PIC4rl-gym: a ROS2 Modular Framework for Robots Autonomous Navigation with Deep Reinforcement Learning. *In 2023 3rd International Conference on Computer, Control and Robotics (ICCCR)*. IEEE. https://doi.org/10.1109/icccr56747.2023.10193996