

**ІНСТИТУТ ПРОБЛЕМ МАТЕМАТИЧНИХ МАШИН І СИСТЕМ
НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ**

Кваліфікаційна наукова праця
на правах рукопису

ОЛЕКСІЄНКО ПЕТРО ДМИТРОВИЧ

УДК 004.94:681.5

ДИСЕРТАЦІЯ

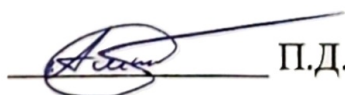
**МОДЕЛІ ТА МЕТОД ПРОГРАМНОГО КЕРУВАННЯ РОЯМИ ДРОНІВ НА
ОСНОВІ ГРУПОВОГО ІНТЕЛЕКТУ**

Спеціальність: 122 «Комп'ютерні науки»

Галузь знань: 12 «Інформаційні технології»

Подається на здобуття ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

 П.Д. Олексієнко

Науковий керівник: Казимир Володимир Вікторович, доктор технічних наук,
професор

АНОТАЦІЯ

Олексієнко П.Д. Моделі та метод програмного керування роями дронів на основі групового інтелекту. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 122 – «Комп’ютерні науки» (12 – «Інформаційні технології»). – Інститут проблем математичних машин і систем НАН України, Київ, 2026.

У дисертаційній роботі виконане конкретне наукове завдання у галузі комп’ютерних наук з розробки методу програмного керування роєм дронів як мультиагентною системою БПЛА з використанням вбудованих моделей алгоритмів поведінки дронів-агентів з підтримкою прийняття групових рішень.

Метою дисертаційної роботи є підвищення ефективності рою дронів при виконанні місії за рахунок використання групового інтелекту та скорочення часу реакції на події при модельно-орієнтованому керуванні дроном-агентом шляхом реалізації MVC-підходу до організації взаємодіючих паралельних процесів.

Об’єктом дослідження є процес керування агентами в ройових системах безпілотних літальних апаратів.

Предметом дослідження є моделі, методи і шаблони формалізованого опису та програмної реалізації алгоритмів керування мультиагентними системами БПЛА.

В ході виконання наукового завдання отримано такі наукові результати:

1. Вперше:

– розроблено формальну модель алгоритму керування агентом у складі рою дронів, яка, на відміну від існуючих, безпосередньо виконує роль трирівневої програми керування з подієво-орієнтовним реагуванням на зміну ситуації, що забезпечує узгоджене прийняття рішень при виконанні місії;

– розроблено мультиагентну модель поведінки рою мультикоптерних дронів при виконанні місії із захисту інфраструктурних об’єктів, яка, на відміну від існуючих, відтворює роль групового інтелекту в процесі відбиття нападу повітряних цілей з урахуванням можливостей підсистем спостереження та розпізнавання, що

дозволяє отримувати попередні оцінки ефективності проектних рішень по системним показникам.

2. Удосконалено формальний апарат керуючих Е-мереж шляхом введення подієвих тригерів переходів, що забезпечує асинхронну активацію елементів моделі на рівнях реагування, планування та взаємодії без використання циклічного сканування станів системи керування, що забезпечує 100-кратне скорочення часу реакції на події.

3. Отримав подальший розвиток метод програмного керування роєм дронів, який, на відміну від існуючих, базується на MVC-підході в запропонованій інтерпретації до процесів керування та взаємодії вбудованих моделей агентів у реальному часі, що забезпечує самоорганізацію поведінки рою дронів з використанням групового інтелекту.

У вступі розглянуто передумови розвитку ройових систем безпілотних літальних апаратів та особливості їх застосування в задачах моніторингу територій, охорони критичної інфраструктури, супроводу об'єктів, пошуку цілей та виконання інших колективних місій. Показано, що зі зростанням кількості агентів у рої суттєво підвищуються вимоги до швидкодії програмного забезпечення, ефективності використання обчислювальних ресурсів та здатності системи своєчасно реагувати на події, що виникають у процесі функціонування. Проаналізовано основні проблеми існуючих підходів до програмного керування ройовими системами, серед яких надлишкові часові затримки, складність масштабування, висока залежність від циклічного опитування станів та обмежені можливості організації великої кількості взаємодіючих паралельних процесів.

У першому розділі виконано аналіз сучасного стану наукових досліджень у сфері побудови систем керування роєм безпілотних літальних апаратів та досліджено основні напрями розвитку ройових технологій. Розглянуто особливості застосування ройових систем у задачах моніторингу територій, охорони об'єктів критичної інфраструктури, пошуку та супроводу цілей, розвідки, патрулювання та координації колективних дій автономних агентів. Показано, що використання ройових принципів дозволяє забезпечити високу відмовостійкість, адаптивність та масштабованість

системи за рахунок розподіленого прийняття рішень і відсутності єдиного центру керування.

Досліджено існуючі архітектури програмного керування агентами рою, включаючи реактивні, поведінкові, агентні, ієрархічні та гібридні підходи. Показано, що більшість існуючих систем використовує механізми циклічного опитування станів або централізованої координації, що призводить до збільшення затримок реагування та обмежує ефективність функціонування рою при зростанні його розмірності.

Окремо розглянуто подієво-орієнтовані підходи до побудови розподілених систем керування та їх застосування у сфері робототехніки і безпілотних літальних апаратів. Проведений аналіз показав, що використання подій як механізму організації взаємодії дозволяє зменшити обсяг зайвих обчислень, підвищити швидкість реагування на зміни середовища та забезпечити більш ефективне використання обчислювальних ресурсів.

У другому розділі досліджено теоретичні засади побудови формалізованих моделей керування для ройових систем безпілотних літальних апаратів та розглянуто можливості застосування мереж Петрі для опису складних паралельних і розподілених процесів. Проведено аналіз класичних мереж Петрі, а також особливості їх використання у задачах моделювання багатокomпонентних систем реального часу. Показано, що традиційні підходи до побудови моделей не завжди забезпечують достатню гнучкість для опису великої кількості взаємодіючих агентів, характерних для ройових систем БПЛА.

Проаналізовано особливості функціонування рою як багатоагентної системи, у якій кожен агент приймає рішення на основі локальної інформації та взаємодіє з іншими учасниками групи через механізми обміну повідомленнями. Розглянуто моделі середовища функціонування рою, обмеження спостереження, особливості передачі інформації між агентами та вплив цих факторів на формування колективної поведінки.

Для дослідження процесів колективної взаємодії розроблено імітаційну модель оборонного рою БПЛА, яка дозволяє відтворювати сценарії виявлення, супроводу та перехоплення повітряних цілей. У моделі формалізовано поведінкові стани агентів,

механізми переходу між режимами функціонування, процеси обміну інформацією та резервування функцій у випадку втрати окремих учасників рою. На основі моделювання проаналізовано вплив різних параметрів системи на ефективність виконання місії та досліджено закономірності формування колективної поведінки агентів.

У третьому розділі представлено удосконалений метод програмного керування роєм безпілотних літальних апаратів, який базується на використанні керуючих Е-мереж як виконуваних моделей поведінки агентів та подієво-орієнтованого підходу до організації взаємодії паралельних процесів. Запропонований метод орієнтований на скорочення часу реакції системи на події, підвищення ефективності використання обчислювальних ресурсів та забезпечення масштабованості при збільшенні кількості агентів рою.

Розроблено формальну модель алгоритму керування агентом, яка поєднує механізми реагування на події, планування дій та координації взаємодії з іншими агентами. Особливу увагу приділено розробленню механізму подієво-орієнтованого виконання моделі. Запропоновано введення подієвих тригерів переходів, які забезпечують асинхронну активацію елементів мережі без необхідності циклічного сканування станів системи. Такий підхід дозволяє виконувати обробку лише тих елементів моделі, для яких виникли відповідні події, що суттєво скорочує обсяг зайвих обчислень та підвищує швидкодію системи керування.

Значну увагу приділено організації паралельного виконання процесів у межах одного агента. Розроблено механізми синхронізації, координації та взаємодії між незалежними потоками обробки інформації, що дозволяє одночасно виконувати завдання спостереження, аналізу ситуації, планування дій та координації з іншими агентами рою. У результаті сформовано цілісну методологічну основу побудови програмних систем керування роєм БПЛА на базі виконуваних моделей, що поєднує формальну строгість опису алгоритмів із можливістю їх практичного використання у системах реального часу.

У четвертому розділі наведено результати практичної реалізації запропонованого методу програмного керування роєм безпілотних літальних апаратів

та виконано його експериментальну перевірку в умовах, наближених до реальних сценаріїв функціонування ройових систем. Розроблено програмний прототип системи керування на основі керуючих Е-мереж, реалізовано механізми подієво-орієнтованої взаємодії паралельних процесів та проведено інтеграцію виконуваних моделей із програмними компонентами агента. Особливу увагу приділено дослідженню часових характеристик функціонування системи та оцінюванню впливу архітектурних рішень на швидкодію програмного забезпечення.

Виконано вимірювання часу реакції системи на події, досліджено особливості використання обчислювальних ресурсів та проаналізовано вплив різних підходів на ефективність функціонування програмного забезпечення. Отримані результати підтвердили суттєве скорочення часу реагування при використанні подієвих тригерів переходів та зниження обсягу зайвих обчислень порівняно з механізмами циклічного опитування станів.

На основі отриманих експериментальних даних виконано модернізацію розробленої імітаційної моделі рою БПЛА та проведено серію порівняльних досліджень результативності виконання місії. У процесі моделювання враховувалися експериментально визначені часові характеристики програмного керування, що дозволило оцінити вплив архітектурних рішень на колективну поведінку рою. Результати експериментів показали, що скорочення часу реакції системи забезпечує більш оперативне виявлення та нейтралізацію цілей, а середня результативність виконання місії для подієво-орієнтованого підходу виявилася вищою порівняно з циклічним керуванням.

Отримані результати підтвердили працездатність запропонованого методу програмного керування, адекватність розроблених моделей та доцільність використання керуючих Е-мереж для побудови масштабованих систем керування роєм БПЛА.

Узагальнюючим результатом дисертаційного дослідження є розробка методу керування роєм дронів на основі вбудованих моделей процесів функціонування дронів-агентів з використанням MVC-підходу для забезпечення підвищення

швидкодії, ефективності використання обчислювальних ресурсів та масштабованості системи керування роєм БПЛА.

Дисертація виконувалася в Інституті проблем математичних машин і систем НАН України.

Результати наукових досліджень були використані під час наукової і науково-технічної діяльності Національного університету «Чернігівська політехніка» в рамках виконання за державним замовленням науково-дослідних робіт: НДР «Мультиагентна система захисту об'єктів інфраструктури на основі рою мультикоптерних дронів» (номер державної реєстрації: 0123U101819 (2023-2025)).

Також результати наукових досліджень дисертаційної роботи впроваджено у Державному науково-дослідному інституті випробувань і сертифікації озброєння та військової техніки для проведення натурних досліджень та відпрацювання програм і методик випробувань БПЛА та у навчальний процес з підготовки аспірантів в Інституті проблем математичних машин і систем НАН України (Додаток А).

Ключові слова: рої дронів, БПЛА, мультиагентна система, військові системи, груповий інтелект, контури управління, адаптивне управління, математична модель, критична інфраструктура, програмне забезпечення, об'єктно-орієнтоване програмування, програмний агент.

ANNOTATION

Oleksiienko P.D. Models and a Method for Software Control of Drone Swarms Based on Collective Intelligence. – Qualification research work presented as a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 122 «Computer Science». – Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine, Kyiv, 2026.

This dissertation addresses a specific research problem in the field of computer science: the development of a method for programmatically controlling a swarm of drones as a multi-agent UAV system, utilizing embedded behavioral models for drone agents that support group decision-making.

The purpose of the dissertation is to improve the efficiency of a drone swarm during mission execution by leveraging swarm intelligence and reducing response times to events in model-based control of a drone agent through the implementation of an MVC approach to organizing interacting parallel processes.

The object of the research is the process of controlling agents in unmanned aerial vehicle (UAV) swarm systems.

The subject of the research is models, methods, and patterns for the formal description and software implementation of control algorithms for multi-agent UAV systems.

During the accomplishment of the scientific task, the following scientific results were obtained:

1. For the first time:

- a formal model of an agent control algorithm within a drone swarm has been developed which, unlike existing approaches, directly performs the role of a three-level control program with event-driven response to changes in the situation, ensuring coordinated decision-making during mission execution;

- a multi-agent behavioral model of a multicopter drone swarm performing infrastructure protection missions has been developed which, unlike existing approaches, reproduces the role of swarm intelligence in the process of countering attacks by aerial

targets while taking into account the capabilities of observation and recognition subsystems, thus enabling preliminary evaluation of design solutions according to system-level performance indicators.

2. The formal apparatus of control E-nets has been improved through the introduction of event-based transition triggers that provide asynchronous activation of model elements at the response, planning, and interaction levels without the use of cyclic scanning of control system states, resulting in a 100-fold reduction in event response time.

3. The method of software control of a drone swarm has been further developed. Unlike existing approaches, it is based on the MVC approach as interpreted in this work for the real-time control and interaction of embedded agent models, which enables the self-organization of drone swarm behavior through the use of collective intelligence.

The introduction examines the prerequisites for the development of unmanned aerial vehicle swarm systems and the features of their application in territory monitoring, critical infrastructure protection, object escort, target search, and other collective missions. It is shown that as the number of agents in a swarm increases, the requirements for software performance, efficient utilization of computational resources, and the ability of the system to respond to events in a timely manner become significantly higher. The main problems of existing approaches to software control of swarm systems are analyzed, including excessive time delays, difficulties in scaling, strong dependence on cyclic state polling, and limited capabilities for organizing a large number of interacting parallel processes.

The first chapter presents an analysis of the current state of scientific research in the field of unmanned aerial vehicle swarm control systems and investigates the main directions of swarm technology development. The features of applying swarm systems to territory monitoring, critical infrastructure protection, target search and tracking, reconnaissance, patrolling, and coordination of collective actions of autonomous agents are considered. It is shown that the use of swarm principles makes it possible to ensure high fault tolerance, adaptability, and scalability of the system through distributed decision-making and the absence of a centralized control center.

An analysis of approaches to representing a UAV swarm as a multi-agent system, in which global behavior emerges as a result of local interactions among individual agents, is

conducted. The principles of self-organization, collective decision-making, and the use of swarm intelligence for solving complex tasks in a dynamic environment are examined. Particular attention is paid to mechanisms of local coordination, information exchange among agents, and the formation of coordinated swarm behavior without the use of centralized control.

Existing software control architectures for swarm agents, including reactive, behavioral, agent-based, hierarchical, and hybrid approaches, are investigated. It is shown that most existing systems employ cyclic state polling mechanisms or centralized coordination, which leads to increased response delays and limits the effectiveness of swarm operation as its size increases.

Event-driven approaches to the construction of distributed control systems and their application in robotics and unmanned aerial vehicles are considered separately. The analysis demonstrates that the use of events as a mechanism for organizing interactions makes it possible to reduce unnecessary computations, increase the speed of response to environmental changes, and ensure more efficient utilization of computational resources.

Based on the conducted analysis, it is established that existing approaches do not sufficiently address the need to combine the formal description of control algorithms with the possibility of their direct execution within real-time software systems. The problem of excessive time costs associated with cyclic state polling, as well as the limited scalability of existing solutions when increasing the number of agents and the complexity of interaction scenarios, is identified.

The second chapter investigates the theoretical foundations of constructing formalized control models for unmanned aerial vehicle swarm systems and considers the possibilities of applying Petri nets to describe complex parallel and distributed processes. An analysis of classical Petri nets, their temporal, hierarchical, and control extensions, as well as the features of applying these formalisms to the modeling of multi-component real-time systems, is carried out. It is shown that traditional approaches to model construction do not always provide sufficient flexibility for describing a large number of interacting agents characteristic of UAV swarm systems.

The features of swarm operation as a multi-agent system, in which each agent makes decisions based on local information and interacts with other members of the group through message exchange mechanisms, are analyzed. Models of the swarm operating environment, observation limitations, characteristics of information transmission between agents, and the influence of these factors on the formation of collective behavior are considered. Particular attention is paid to local safety and coordination rules that ensure collision avoidance, maintenance of the required distance between agents, alignment of movement directions, and joint mission execution.

To investigate collective interaction processes, a simulation model of a defensive UAV swarm was developed, making it possible to reproduce scenarios of aerial target detection, tracking, and interception. The model formalizes agent behavioral states, mechanisms for switching between operating modes, information exchange processes, and function redundancy in the event of the loss of individual swarm members. Based on simulation results, the influence of various system parameters on mission effectiveness was analyzed, and the patterns of collective agent behavior formation were investigated.

The third chapter presents an improved method for software control of an unmanned aerial vehicle swarm based on the use of control E-nets as executable models of agent behavior and an event-driven approach to organizing the interaction of parallel processes. The proposed method is aimed at reducing system response time to events, improving the efficiency of computational resource utilization, and ensuring scalability as the number of swarm agents increases.

A formal model of an agent control algorithm is developed that combines event response mechanisms, action planning, and coordination of interactions with other agents. Unlike traditional approaches, in which the control logic is implemented as separate program code, the proposed solution uses control E-nets as directly executable models that determine state transition sequences and event-processing mechanisms within the system.

Particular attention is devoted to the development of an event-driven execution mechanism for the model. The introduction of event-based transition triggers is proposed, providing asynchronous activation of network elements without the need for cyclic scanning of system states. This approach makes it possible to process only those model elements for

which corresponding events have occurred, thereby significantly reducing unnecessary computations and improving the performance of the control system.

Considerable attention is paid to organizing parallel process execution within a single agent. Mechanisms for synchronization, coordination, and interaction among independent information-processing flows are developed, enabling the simultaneous execution of observation, situation analysis, action planning, and coordination tasks with other swarm agents. As a result, a comprehensive methodological foundation for the development of UAV swarm control software systems based on executable models is established, combining the formal rigor of algorithm description with the possibility of practical implementation in real-time systems.

The fourth chapter presents the results of the practical implementation of the proposed software control method for unmanned aerial vehicle swarms and its experimental validation under conditions close to real swarm operation scenarios. A software prototype of the control system based on control E-nets was developed, mechanisms of event-driven interaction among parallel processes were implemented, and executable models were integrated with the software components of an agent. Particular attention is paid to investigating the temporal characteristics of system operation and evaluating the influence of architectural solutions on software performance.

A series of experiments aimed at comparing the traditional cyclic event-processing approach with the proposed event-driven control mechanism was conducted. Measurements of system response time to events were performed, the characteristics of computational resource utilization were investigated, and the influence of different approaches on software performance was analyzed. The obtained results confirmed a significant reduction in response time when using event-based transition triggers, as well as a reduction in unnecessary computations compared with cyclic state polling mechanisms.

Special attention is devoted to evaluating the scalability of the proposed approach. The influence of increasing the number of agents and the intensity of information exchange on the characteristics of the control system was investigated. It was established that the event-driven approach maintains acceptable temporal characteristics as scenario complexity

and the number of simultaneously executed processes increase, which is important for the practical application of large-scale swarm systems.

Based on the obtained experimental data, the developed UAV swarm simulation model was enhanced, and a series of comparative studies of mission effectiveness was conducted. During simulation, experimentally determined temporal characteristics of software control were taken into account, making it possible to evaluate the influence of architectural solutions on collective swarm behavior. The experimental results showed that reducing system response time ensures more rapid target detection and neutralization, while the average mission effectiveness achieved with the event-driven approach was higher than that obtained with cyclic control.

The obtained results confirmed the feasibility of the proposed software control method, the adequacy of the developed models, and the suitability of using control E-nets for building scalable UAV swarm control systems. The conducted studies demonstrated that the application of the event-driven approach makes it possible to simultaneously improve system performance, increase the efficiency of computational resource utilization, and achieve higher effectiveness in collective mission execution compared with traditional software control approaches.

The overall result of the dissertation research is the development of a method for controlling a swarm of drones based on embedded models of drone-agent operations, using the MVC approach to ensure improved performance, efficient use of computational resources, and scalability of the UAV swarm control system.

The dissertation was carried out at the Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine.

The results of the scientific research were used in the scientific and scientific-technical activities of Chernihiv Polytechnic National University within the framework of state-funded research projects, namely the research project “Multi-Agent System for the Protection of Critical Infrastructure Facilities Based on a Swarm of Multicopter Drones” (state registration number: 0123U101819 (2023–2025)).

The results of the scientific research presented in the dissertation have also been implemented at the State Research Institute for Testing and Certification of Weapons and

Military Equipment for conducting field studies and developing UAV testing programs and methodologies, as well as in the educational process for training graduate students at the Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine (Appendix A).

Keywords: drone swarms, UAVs, multi-agent system, military systems, group intelligence, control circuits, adaptive control, mathematical model, critical infrastructure, software, object-oriented programming, software agent.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Kazymyr V., Oleksiienko P., Chornonoh O., Rohovenko A. Modeling of Defensive Drone Swarms with NetLogo. *Mathematical Modeling and Simulation of Systems (MODS 2023). Lecture Notes in Networks and Systems*. Cham: Springer, 2024. Vol. 1091. P. 377–387. DOI: https://doi.org/10.1007/978-3-031-67348-1_28. Базис: Scopus, WoS.

2. Олексієнко П. Д., Казимир В. В. Вбудовані моделі алгоритму керування роєм БПЛА. *Математичні машини і системи*. 2025. № 3–4. С. 90–100. DOI: <https://doi.org/10.34121/1028-9763-2025-3-4-90-100>. Базис: CrossRef, Google Scholar.

3. Олексієнко П. Д., Казимир В. В. Подієво-орієнтований підхід в реалізації вбудованих систем керування роєм БПЛА. *Технічні науки та технології*. 2025. № 3 (41). С. 225–236. DOI: [https://doi.org/10.25140/2411-5363-2025-3\(41\)-225-236](https://doi.org/10.25140/2411-5363-2025-3(41)-225-236). Базис: CrossRef, Google Scholar.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

4. Suslo M., Kazymyr V., Oleksiienko P., Kagitin D. Semi-Physical Modeling of Drone Swarm Control Processes. *15th International Conference on Advanced Computer Information Technologies (ACIT): conference proceedings, Šibenik, Croatia, 17–19 September 2025*. Šibenik, 2025. P. 102–108. DOI: <https://doi.org/10.1109/ACIT65614.2025.11185750>.

5. Kahitin D., Kazymyr V., Suslo M., Yatchenko Y., Oleksiienko P. Drone Positioning in a Specified Location Using Visual Data Under GPS-Denied Conditions. *IEEE 13th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS): conference proceedings, Gliwice, Poland, 4–6 September 2025*. Gliwice, 2025. P. 318–323. DOI: <https://doi.org/10.1109/IDAACS68557.2025.11322385>.

Наукові праці, які додатково відображають наукові результати дисертації:

6. Сушло М. Ю., Олексієнко П. Д., Сігута А. В., Казимир В. В. Напів натурне моделювання процесів керування роєм дронів. *Новітні технології сучасного суспільства (HTCC-2024)*: тези доп. V Міжнар. наук.-практ. конф., м. Чернігів, 12 груд. 2024 р. Чернігів, 2024. С. 183–184. URL: https://inel.stu.cn.ua/ntss/NTSS_2024_zbirnyk.pdf.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	21
ВСТУП.....	22
РОЗДІЛ 1 СТАН ПРОБЛЕМИ, ПРИКЛАДНІ СЦЕНАРІЇ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ	28
1.1 Ройові системи БПЛА та їх колективна поведінка.....	29
1.1.1 Рій БПЛА як мультиагентна система	30
1.1.2 Аналіз типових місій ройових систем БПЛА	33
1.1.3 Основні обмеження ройових систем БПЛА	35
1.2 Прикладні сценарії з підвищеними вимогами до реактивності.....	37
1.2.1 Сценарії уникнення зіткнень	38
1.2.2 Аварійні та нештатні ситуації.....	39
1.2.3 Сценарії перехоплення та оборони об'єктів	41
1.3 Аналіз методів програмного керування дроном-агентом	42
1.3.1 Огляд формалізмів опису керуючих алгоритмів агентів	43
1.3.2 Реагування на події у модельно-орієнтованому керуванні.....	46
1.4 Постановка задач дослідження та критерії ефективності.....	49
1.4.1 Формулювання наукового завдання.....	49
1.4.2 Критерії часової ефективності.....	50
1.4.3 Критерії ресурсної та функціональної ефективності	52
1.5 Висновки до розділу 1.....	54
РОЗДІЛ 2 МОДЕЛІ КЕРУВАННЯ РОЄМ БПЛА ТА ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ ЙОГО ПОВЕДІНКИ.....	56
2.1 Мультиагентне подання рою	56
2.1.1 Агент як елемент ройової системи	57
2.1.2 Зовнішнє оточення агента-дрона.....	59
2.1.3 Обмеження спостереження та комунікації в рої	60
2.1.4 Локальні правила взаємодії агентів.....	62

2.1.5 Роль групового інтелекту у формуванні колективної поведінки	64
2.2 Формальна модель керування на основі керуючих Е-мереж.....	66
2.2.1 Загальні положення керуючих Е-мереж	66
2.2.2 Позиції моделі та їх інтерпретація	70
2.2.3 Переходи та умови їх спрацювання	74
2.2.4 Передавання даних і маркування	77
2.2.5 Часові затримки та обмеження	81
2.3 Імітаційна модель оборонного рою.....	84
2.3.1 Архітектура імітаційної моделі	85
2.3.2 Поведінкові стани агентів.....	88
2.3.3 Механізми взаємодії та обміну даними	90
2.3.4 Візуалізація параметрів моделі в середовищі NetLogo	93
2.4 Метрики ефективності виконання місії	96
2.4.1 Часові метрики ефективності	97
2.4.2 Метрики результативності місії	99
2.4.3 Метрики покриття та координації.....	100
2.4.4 Метрики стійкості до відмов	102
2.5 Висновки до розділу 2.....	104
РОЗДІЛ 3 МЕТОД ПРОГРАМНОГО КЕРУВАННЯ РОЄМ ДРОНІВ НА	
ОСНОВІ ВБУДОВАНИХ МУЛЬТИАГЕНТНИХ МОДЕЛЕЙ	106
3.1 Метод керування роєм БПЛА на основі вбудованих моделей	107
3.1.1 Впровадження архітектури програмної реалізації агента з вбудованою моделлю	108
3.1.2 Опис дій та умов у програмному коді.....	113
3.1.3 Інтеграція з підсистемами БПЛА	117
3.2 Модифікована архітектура програми керування у стилі MVC	119
3.2.1 Модель як виконувана Е-мережа.....	120
3.2.2 Контролер як механізм виконання	122
3.2.3 Подання як засіб спостереження	123
3.3 Метод подієвої інтеграції з зовнішнім середовищем через слухачів подій.....	125

3.3.1 Типи зовнішніх подій.....	126
3.3.2 Удосконалення механізму обробки подій.....	129
3.3.3 Впровадження відмови від циклічного опитування.....	130
3.4 Реалізація вбудованої моделі керування у виконавчому середовищі	133
3.4.1 Паралельне виконання гілок і черги	134
3.4.2 Синхронізація та об'єднання гілок.....	136
3.5 Висновки до розділу 3.....	138

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНИХ МЕТОДІВ КЕРУВАННЯ РОЄМ ДРОНІВ НА ОСНОВІ ГРУПОВОГО ІНТЕЛЕКТУ 139

4.1 Методика експериментів для перевірки ефективності запропонованих методів керування роєм дронів.....	139
4.1.1 Мета, об'єкт і предмет експериментальних досліджень.....	140
4.1.2 Апаратна та програмна платформа експериментів	141
4.1.3 Умови повторюваності експериментів.....	148
4.2 Показники оцінювання та спосіб обробки результатів.....	149
4.2.1 Часові показники.....	150
4.2.2 Показники ресурсного навантаження	151
4.2.3 Показники ефективності виконання.....	152
4.3 Результати порівняння реакції на події проти циклічного опитування	154
4.3.1 Порівняння часу реакції.....	155
4.3.2 Порівняння ресурсного навантаження.....	157
4.4 Оцінювання результативності місії на основі вдосконаленої імітаційної моделі рою БПЛА.....	161
4.5 Узагальнення методів програмного керування роєм та практичні рекомендації	166
4.5.1 Сценарії з критичністю подієвого підходу	166
4.5.2 Практичні рекомендації щодо проектування систем керування роєм	167
4.6 Висновки до розділу 4.....	172
ВИСНОВКИ.....	173

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	178
ДОДАТКИ	190
Додаток А Акти впровадження результатів дисертаційного дослідження	190
Додаток Б Фрагмент логіки виконання очікування сигналу	193
Додаток В Ключові фрагменти програмного коду системи Е-мережі	194
Додаток Г Структура системи керування дроном у складі мультиагентної системи.....	205
Додаток Д Алгоритм подієво-орієнтованої взаємодії агента рою при розпізнаванні цілі.....	206
Додаток Е Список публікацій здобувача за темою дисертації	207

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CEN – Control E-Network.

CPS – Cyber-Physical System.

Е-мережа – розширена мережа Петрі.

FSM – Finite State Machine.

GNSS – Global Navigation Satellite System.

IMU – Inertial Measurement Unit.

MVC – Model–View–Controller.

UAV – Unmanned Aerial Vehicle.

БПЛА – безпілотний літальний апарат.

ВСТУП

Актуальність теми

Розвиток безпілотних авіаційних комплексів призвів до того, що почалось стрімке формування нового класу розподілених технічних систем – ройових систем БПЛА, у яких колективна поведінка формується на основі локальних правил взаємодії агентів. У даній роботі ройові системи розглядаються як мультиагентні, у яких кожен дрон виконує роль автономного агента, а узгоджена поведінка рою формується як прояв групового інтелекту. Застосування таких системи можливе для моніторингу територій, охорони об'єктів, пошуково-рятувальних операцій, перехоплення та супроводу цілей, а також виконання спеціалізованих оборонних задач.

Характерними особливостями використання ройових систем є мінливість середовища, обмежена спостережуваність, нестабільність каналів зв'язку та наявність жорстких часових обмежень для реакції на події. Саме тому ефективність функціонування рою значною мірою визначається не лише локальними правилами взаємодії, а й способом організації алгоритмів керування окремим агентом.

Різні науковці та наукові групи розглядають ройові системи БПЛА як перспективний напрям робототехніки та кіберфізичних систем, що доволі різносторонньо представлено в роботах Чанга [1], Шахзада [2], Круна [3]. Інтерпретація рою як мультиагентної системи з локальними правилами взаємодії представлена в роботах Барка [4], Росі [5], Рейнольдса [6].

У той же час значна частина наукових робіт присвячена безпосередньо алгоритмам прийняття рішень у формуванні, консенсусу та координації руху (роботи Олфаті-Сабера [7], Ліціо [8], Чена [9]), у плануванні траєкторій і униканні зіткнень (роботи Рахмана [10], Алі [11, 14], Аршида [12], Резаї [13]), у розподілі ролей і формуванні коаліцій (робота Ву [15]).

Але більшість робіт концентруються на алгоритмічних аспектах або задачах оптимізації, тоді як проблема формалізації та виконуваної інтеграції моделей керування в реальні вбудовані системи залишається менш систематизованою, а існуючі реалізації керування агентами рою в основному базується на процедурних

або ієрархічних алгоритмах, що реалізуються у вигляді програмного коду без формалізованого опису станів і переходів. Це ускладнює аналіз паралельності процесів, синхронізацію дій агентів, перевірку коректності поведінки в аварійних ситуаціях та оцінювання часових характеристик системи. У результаті поведінка рою часто залежить від деталей реалізації, що знижує передбачуваність та стійкість системи до збурень і відмов, а це обумовлює потребу у розробленні формалізованого представлення та виконання алгоритмів керування агентами рою БПЛА, який забезпечує подієву реактивність, підтримку паралельних процесів і контроль часової поведінки за умови збереження можливості прямого вбудовування в програмне забезпечення бортових систем.

Зв'язок роботи з науковими програмами, планами, темами, грантами

Дисертаційна робота є складовою частиною досліджень, що проводяться в Національному університеті «Чернігівська політехніка» в рамках виконання за державним замовленням науково-дослідних робіт: НДР «Мультиагентна система захисту об'єктів інфраструктури на основі рою мультикоптерних дронів» (номер державної реєстрації: 0123U101819 (2023-2025)). Також результати наукових досліджень дисертаційної роботи впроваджено у Державному науково-дослідному інституті випробувань і сертифікації озброєння та військової техніки для проведення натурних досліджень та відпрацювання програм і методик випробувань БПЛА та у навчальний процес з підготовки аспірантів в Інституті проблем математичних машин і систем НАН України.

Мета і завдання дослідження

Підвищення ефективності рою дронів при виконанні місії за рахунок використання групового інтелекту та скорочення часу реакції на події при модельно-орієнтованому керуванні дроном-агентом шляхом реалізації MVC-підходу до організації взаємодіючих паралельних процесів.

Для досягнення поставленої мети необхідно розв'язати **такі основні завдання**:

1. Проаналізувати сучасні підходи до моделювання та керування ройовими системами БПЛА.

2. Розробити агентну імітаційну модель рою для дослідження поведінки системи в оборонних сценаріях.

3. Сформувати формальну модель багаторівневого керування агентом із урахуванням подієвої природи зміни станів.

4. Удосконалити формальний апарат керуючих Е-мереж для підтримки асинхронної активації переходів.

5. Розробити метод вбудовування формальної моделі у програмне забезпечення керування БПЛА.

6. Провести експериментальне порівняння подієвого механізму керування та циклічного опитування станів.

7. Оцінити часову, ресурсну та функціональну ефективність запропонованого підходу.

Об'єкт дослідження – процес керування агентами в ройових системах безпілотних літальних апаратів.

Предмет дослідження – моделі, методи і шаблони формалізованого опису та програмної реалізації алгоритмів керування мультиагентними системами БПЛА.

Методи дослідження. Теоретичною основою дослідження стали положення теорії розподілених систем керування, теорії мультиагентних систем, теорії мереж Петрі та їх розширень, методи імітаційного моделювання та аналізу часових характеристик.

Наукова новизна отриманих результатів. У процесі вирішення поставлених задач автором розроблено метод виконуваного формалізованого керування агентом рою БПЛА, що поєднує формальний опис станів і переходів із механізмом подієвої активації в програмному середовищі.

Наукові результати базуються на таких основних положеннях:

Вперше:

– розроблено формальну модель алгоритму керування агентом у складі рою дронів, яка, на відміну від існуючих, безпосередньо виконує роль трирівневої програми керування з подієво-орієнтовним реагуванням на зміну ситуації, що забезпечує узгоджене прийняття рішень при виконанні місії;

– розроблено мультиагентну модель поведінки рою мультикоптерних дронів при виконанні місії із захисту інфраструктурних об’єктів, яка, на відміну від існуючих, відтворює роль групового інтелекту в процесі відбиття нападу повітряних цілей з урахуванням можливостей підсистем спостереження та розпізнавання, що дозволяє отримувати попередні оцінки ефективності проектних рішень по системним показникам.

Удосконалено:

– формальний апарат керуючих Е-мереж шляхом введення подієвих тригерів переходів, що забезпечує асинхронну активацію елементів моделі на рівнях реагування, планування та взаємодії без використання циклічного сканування станів системи керування, що забезпечує 100-кратне скорочення часу реакції на події.

Отримав подальший розвиток

– метод програмного керування роєм дронів, який, на відміну від існуючих, базується на MVC-підході в запропонованій інтерпретації до процесів керування та взаємодії вбудованих моделей агентів у реальному часі, що забезпечує самоорганізацію поведінки рою дронів з використанням групового інтелекту.

Практичне значення отриманих результатів полягає у створенні програмного пакету на мові Python, що реалізує модельно-орієнтований підхід до побудови програмних систем керування шляхом вбудовування розроблених CEN-моделей алгоритмів керування безпосередньо у контур керування об’єктом (дроном) та виконання даних алгоритмів у режимі інтерпретації у реальному часі.

Результати дисертаційної роботи впроваджено у реально діючі макети дронів, що функціонують у складі мультиагентної захисної системи, в ході виконання науково-технічної тематики та підготовки аспірантів та у навчальному процесі з підготовки аспірантів в Інституті проблем математичних машин і систем НАН України, що підтверджується відповідним актом.

Особистий внесок здобувача. До дисертації увійшли наукові результати, отримані здобувачем особисто. З наукових праць, опублікованих у співавторстві, в дисертації використано лише ті ідеї та положення, які є результатом особистої роботи здобувача.

Найважливіші ідеї, висновки, рекомендації, отримані в дисертації, оприлюднені на міжнародних науково-практичних конференціях: «International Conference on Advanced Computer Information Technologies», «International Conference on Intelligent Data Acquisition and Advanced Computing Systems», «Новітні технології сучасного суспільства».

У спільних публікаціях автору належать такі результати:

- у роботі [16] розроблено концептуальну та формалізовану моделі функціонування оборонного рою БПЛА, побудовано структурну схему взаємодії агентів рою, проведено серію обчислювальних експериментів та виконано аналіз отриманих результатів. Робота закладає теоретичні основи моделювання поведінки рою безпілотних літальних апаратів та формує базові положення запропонованого в дисертації підходу до організації колективної взаємодії агентів;

- у роботі [17] розроблено архітектурний підхід до побудови програмного забезпечення керування роєм БПЛА, запропоновано механізм взаємодії компонентів на основі подій та слухачів подій (listeners), створено програмний прототип системи та проведено його експериментальну апробацію. Отримані результати стали основою для розроблення програмної архітектури системи керування роєм, представленої в дисертаційній роботі;

- у роботі [18] проведено порівняльний аналіз циклічного та подієво-орієнтованого підходів до побудови систем керування, виконано експериментальні дослідження продуктивності програмних реалізацій та обґрунтовано доцільність використання подієво-орієнтованої архітектури для задач керування ройовими системами. Результати роботи використані при формуванні методу програмного керування агентами рою в межах дисертаційного дослідження;

- у роботі [19] розроблено модель місії БПЛА на основі керуючих E-мереж, формалізовано процеси прийняття рішень агентами рою, взаємодії між ними та обміну інформацією під час виконання завдань перехоплення цілей. Отримані результати стали основою для побудови формальної моделі керування агентами рою та дослідження процесів їхньої координації;

– у роботі [20] досліджено метод позиціонування безпілотного літального апарата за даними комп'ютерного зору в умовах відсутності супутникової навігації, проведено оцінювання точності визначення координат та проаналізовано вплив локального позиціонування на стабільність виконання групових завдань. Отримані результати доповнюють запропонований у дисертації підхід до побудови ройових систем, у яких колективна поведінка формується на основі локальних дій окремих агентів та їхньої взаємодії.

Апробація результатів дисертації

Основні теоретичні та практичні результати досліджень були представлені та обговорені на таких наукових конференціях:

1. 18th International Conference on Mathematical Modeling and Simulation of Systems – MODS 2023 (13–15 November 2023, Chernihiv, Ukraine).
2. V Міжнародна науково-практична конференція «Новітні технології сучасного суспільства» – НТСС-2024 (12 грудня 2024 року, м. Чернігів, Україна).
3. 13th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications – IDAACS 2025 (4–6 September 2025, Gliwice, Poland).
4. 15th International Conference on Advanced Computer Information Technologies – ACIT 2025 (17–19 September 2025, Šibenik, Croatia).

Публікації. Оовні положення та результати дисертаційного дослідження викладено в 6 наукових публікаціях, серед них 2 публікації у наукових фахових виданнях України, 3 у виданнях, проіндексованих у базі даних Scopus, 1 теза доповіді.

Структура та обсяг роботи

Дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг дисертації становить 208 сторінок. Основний текст викладено на 160 сторінках. Робота містить 24 рисунка, список використаних джерел із 107 найменувань та 6 додатків.

РОЗДІЛ 1

СТАН ПРОБЛЕМИ, ПРИКЛАДНІ СЦЕНАРІЇ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Розвиток безпілотних літальних апаратів зумовив перехід від поодиноких автономних платформ до колективних систем, у яких декілька агентів функціонують як єдиний організований рій, тому рійові системи БПЛА розглядаються як перспективний напрям робототехніки та кіберфізичних систем завдяки їх масштабованості, адаптивності та підвищеній стійкості до відмов [1, 2, 3]. У сучасних дослідженнях рій інтерпретується як мультиагентна система з локальними правилами взаємодії, здатна формувати складну колективну поведінку без централізованого керування [4, 5, 6].

Значна частина наукових робіт присвячена алгоритмам формування, консенсусу та координації руху [7, 8, 9], плануванню траєкторій [10, 11, 12], уникненню зіткнень [13, 14], а також задачам розподілу ролей і коаліцій [15]. Окремий напрям становлять дослідження архітектур комунікації та мережевої взаємодії в роях [21, 22]. Разом із тим, більшість оглядових праць концентруються на алгоритмічних аспектах або задачах оптимізації, тоді як проблема формалізації та виконуваної інтеграції моделей керування в реальні вбудовані системи залишається менш систематизованою.

Введення терміну груповий інтелект (swarm intelligence) передбачає використання його як властивості мультиагентної системи, що полягає у формуванні узгодженої глобальної поведінки рою на основі локальних правил взаємодії автономних агентів без централізованого керування [1, 4, 6, 9, 14].

Особливої актуальності набуває питання реактивності рійових систем у сценаріях з жорсткими часовими обмеженнями (уникнення зіткнень, перехоплення цілей, обробка аварійних станів або швидка реконфігурація рою після втрати агента), коли затримка реакції безпосередньо впливає на результативність місії та безпечність експлуатації [23, 24, 25]. Традиційні підходи до реалізації керуючої логіки часто

базуються на циклічному опитуванні станів і перевірці умов у фіксованих часових інтервалах, що може призводити до нестабільних затримок і надлишкового навантаження на обчислювальні ресурси [26, 27, 28, 29].

На противагу цьому активно розвивається подієво-орієнтоване керування, в якому активація керуючих дій відбувається лише за настання відповідних подій або виконання тригерних умов [30, 31, 35, 36], що дозволило у мультиагентних системах продемонструвати покращену масштабованість та ефективність використання ресурсів [37], однак їх інтеграція у складні вбудовані архітектури БПЛА потребує формальних моделей, що здатні одночасно описувати стан, події, умови спрацювання та часові обмеження.

Проблема ускладняється тим, що в реальних системах керування необхідно поєднати три рівні керуючої логіки: рівень реагування, рівень планування та рівень взаємодії, які в сукупності формують трирівневу програму керування агентом, але більшість сучасних наукових робіт зосереджені або на теоретичних моделях колективної поведінки [38, 39], або на програмних симуляторах рою [40, 41], тоді як механізм перетворення формалізованої моделі на виконувану частину програмного забезпечення агента залишається недостатньо формалізованим.

1.1 Ройові системи БПЛА та їх колективна поведінка

Ройові системи безпілотних літальних апаратів розглядаються як один із ключових напрямів розвитку автономної робототехніки та розподілених кіберфізичних систем, коли на відміну від традиційних централізованих груп БПЛА, функціонування рою представляється як мультиагентна система з децентралізованим прийняттям рішень, у якій глобальна поведінка формується внаслідок локальних взаємодій між агентами [1, 3].

Тоді використання ідей колективної поведінки, що мають глибоке теоретичне підґрунтя в дослідженнях зграйної динаміки та моделей типу flocking, де прості локальні правила призводять до складної групової організації [6], дозволяє їх розвинути у напрямі теорії консенсусу та координації руху і показує, що узгоджена

поведінка агентів може бути досягнута навіть за умов обмеженої топології зв'язку та наявності затримок [7, 42, 43, 44]. Сучасні огляди наукових джерел підтверджують, що ройові алгоритми активно застосовуються для задач формування, покриття території, супроводу цілей, розподілу ролей та колективного планування траєкторій [8, 10, 45, 46, 47].

На даний момент переважна більшість досліджень у галузі керування ройовими БПЛА зосереджена на алгоритмах руху та оптимізації, тоді як архітектурні та системні аспекти побудови керуючого програмного забезпечення агентів часто розглядаються фрагментарно, а у практичних застосуваннях (в задачах оборони об'єктів, реагування на загрози, роботи в умовах втрати зв'язку або GPS), коли від системи вимагається не лише правильність колективної поведінки, але й гарантована реактивність, передбачуваність часу відгуку та стабільність при зростанні кількості агентів [23, 48] результати досліджень не висвітлюють сторону реалізації систем.

Додатковою складністю є те, що ройові системи функціонують у мінливих середовищах: змінна топологія мережі, обмежена пропускна здатність каналів зв'язку, енергетичні ресурси агентів, вплив зовнішніх факторів [22, 49]. За таких умов просте масштабування класичних алгоритмів координації не гарантує збереження властивостей системи, зокрема стабільності та обмеженого часу реакції.

1.1.1 Рій БПЛА як мультиагентна система

При розробці систем керування ройовими БПЛА рій безпілотних літальних апаратів доцільно розглядати як мультиагентну систему, у якій кожен агент є автономною обчислювально-керуючою одиницею, що здатна сприймати стан середовища, приймати рішення на основі локальної інформації та взаємодіяти з іншими агентами через канали зв'язку, що дозволяє перейти до децентралізованих архітектур, в яких глобальна поведінка формується як результат узгоджених локальних дій [4, 39].

У межах теорії мультиагентних систем рій інтерпретується як множина агентів [7, 42], які функціонують у спільному середовищі E та пов'язані відношенням взаємодії $R \subseteq A \times A$:

$$A = \{a_1, a_2, \dots, a_N\}. \quad (1.1)$$

У більшості випадків топологія взаємодії є динамічною та залежить від просторової близькості або параметрів зв'язку, а важливою особливістю є відсутність глобального центру прийняття рішень, що зумовлює необхідність формалізації локальних правил та механізмів координації. Тому для подальшого аналізу та формалізації процесів керування необхідно зафіксувати подання рою як мультиагентної системи з децентралізованою координацією, локальними комунікаціями та замкнутим контуром керування кожного агента, що представлено на рис. 1.1.

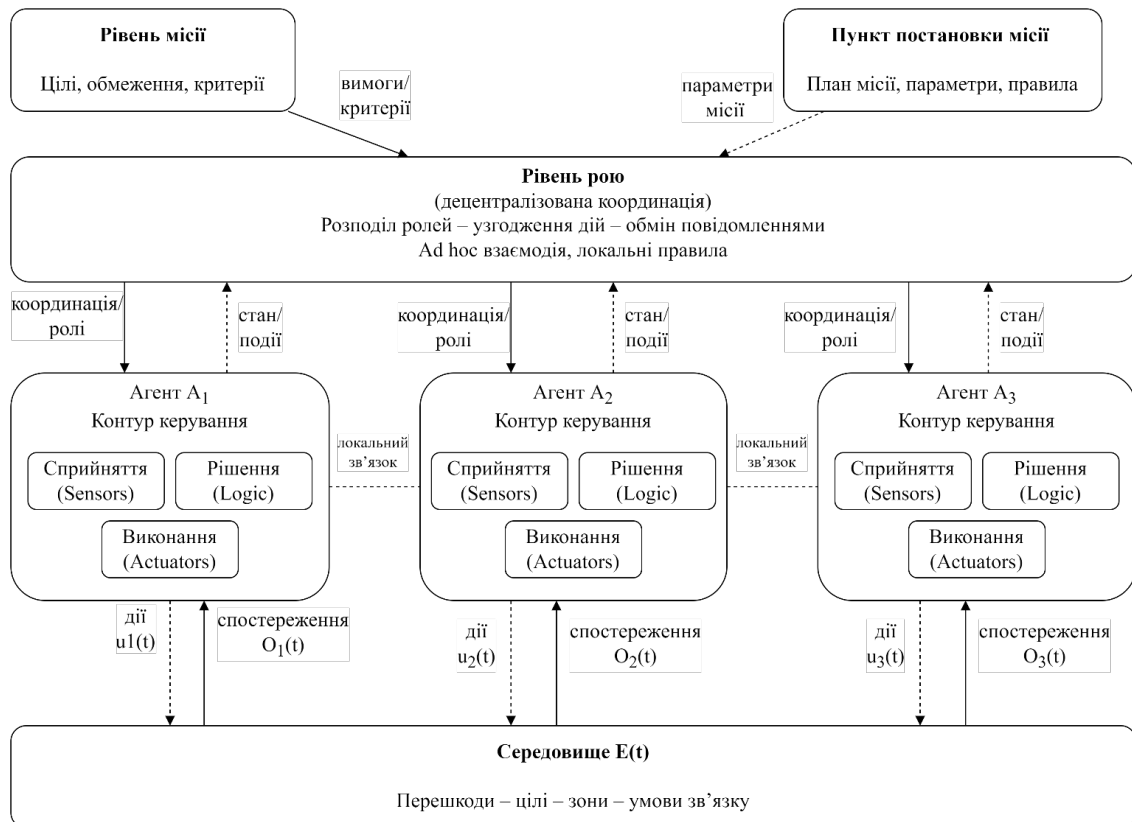


Рисунок 1.1 – Подання рою БПЛА як мультиагентної системи

На даний момент класичні моделі зграйної (ройової) поведінки показують, що навіть прості локальні правила, такі як вирівнювання швидкості, підтримання дистанції, рух до центру мас сусідів, здатні формувати узгоджену групову динаміку [6], що розглядається в області дослідження консенсусу і підтверджується, що за певних умов зв'язності графа взаємодії система досягає асимптотичної узгодженості або стабільної формації [7, 8].

Мультиагентна парадигма у сучасних ройових системах БПЛА використовується для розв'язання широкого спектра задач: колективне покриття території, розподіл цілей, супровід об'єктів, формування геометричних конфігурацій, колективне планування траєкторій [10, 15, 34, 45], а ефективність функціонування рою визначається не лише алгоритмічною коректністю локальних правил, але й архітектурною організацією програмного забезпечення кожного агента.

Особливу увагу в мультиагентному підході приділяють питанням обмеженості інформації, тому що кожен БПЛА має доступ лише до частини глобального стану системи (власних сенсорних даних та інформації від найближчих сусідів), що можна формалізувати у вигляді локальної моделі знань агента:

$$S_i(t) = f(Z_i(t), M_i(t)), \quad (1.2)$$

де $Z_i(t)$ – дані сенсорів, $M_i(t)$ – отримані повідомлення, а $f(\cdot)$ – функція оцінювання стану, що узгоджується з дослідженнями комунікаційних обмежень у роях БПЛА [21, 22].

У практичних реалізаціях мультиагентний підхід стикається з низкою системних обмежень: затримки передавання даних, втрати пакетів, обмеження пропускну здатності, енергетичні ресурси та обчислювальна потужність агентів [48, 49], що призводить до відхилення динаміки рою від теоретично передбаченої і вимагає впровадження формальних моделей керування, здатних враховувати часові обмеження та подієву природу взаємодії.

1.1.2 Аналіз типових місій ройових систем БПЛА

Типові місії ройових систем БПЛА визначають вимоги до колективної поведінки, структури взаємодії агентів та допустимих затримок у прийнятті рішень [50]. На відміну від одиночного БПЛА, де цільова функція найчастіше формулюється як оптимізація траєкторії або підтримка стабільності польоту, ройова система працює як розподілений виконавчий механізм, у якому загальний результат є функцією узгоджених локальних дій багатьох агентів [1, 3, 48].

Узагальнюючи оглядові дослідження з ройової робототехніки та рою БПЛА, типові місії у військових та цивільних системах можна групувати за характером цілі [24, 45, 51, 52]:

- 1) інформаційно-спостережні;
- 2) просторово-покривні;
- 3) координаційно-супровідні;
- 4) протидієві/перехоплювальні;
- 5) місії забезпечення стійкості та відновлення функцій рою.

Першу велику групу становлять місії спостереження та моніторингу, де їх основний зміст – збір сенсорної інформації з великої території, спостереження за ситуацією в режимі наближеному до реального часу, формування карти або актуального уявлення про стан середовища. У таких місіях важливими є розподіл секторів спостереження між агентами, уникнення дублювання збору даних та забезпечення стійкості до втрат окремих БПЛА. Практичне значення таких сценаріїв особливо підкреслюється у задачах реагування на надзвичайні ситуації та цивільної безпеки, де БПЛА виступають мобільними сенсорними платформами [22, 51, 52, 53]. Для спостережних місій ключовими стають метрики повноти покриття інформацією, часу оновлення даних та здатності рою підтримувати заданий темп спостереження при змінній топології зв'язку.

Друга група – це місії покриття простору та патрулювання, що близькі до систем спостереження, але акцент більше робиться не стільки на даних, скільки на геометричному забезпеченні присутності рою в заданій області (рівномірне

розміщення агентів, підтримка дистанцій, контроль меж, періодичне «прочісування» ділянок). У цих задачах зростає роль локальних правил руху і формування, а також вимоги до уникнення зіткнень при збільшенні щільності агентів [13, 53]. Важливо, що покриття та патрулювання часто поєднуються з обмеженнями енергетики: рій має зберігати працездатність протягом часу місії, планувати повернення на заряджання, мінімізувати непотрібні маневри [54, 55, 56].

Третя група – супровід та відстеження рухомих об'єктів (tracking) [57, 58, 59, 60], де ціль є динамічною, тому рій повинен адаптивно перебудовувати конфігурацію: частина агентів може виконувати роль спостереження, інші – роль блокування або перехоплення, ще інші – підтримувати комунікаційну зв'язність між підгрупами. Тут виникає потреба в механізмах розподілу ролей, узгодженого прийняття рішень та коаліційної взаємодії, що прямо пов'язано з підходами до задач алокації в роях [15, 34]. Для місій супроводу зростає чутливість до затримок у передаванні даних: якщо повідомлення про маневр цілі приходить із запізненням, узгоджена реакція рою стає неефективною, а локальні рішення агентів можуть призводити до конфліктів траєкторій.

Четверта група – місії перехоплення, протидії та оборони об'єктів, де їх характерною ознакою є наявність активного противника або загрози, а також необхідність швидкого реагування. У таких сценаріях час реакції рою часто виступає визначальним фактором, тому що затримка у виявленні або узгодженні може призвести до невиконання місії. У роботах [23, 24], що досліджують протидію БПЛА та роботу кооперативних оборонних команд, підкреслюється необхідність високої реактивності та стійкості до відмов, також у таких місіях значно підвищуються вимоги до надійності та кібербезпеки, оскільки атаки на канали зв'язку або підміна даних можуть змінити колективну поведінку рою [48].

П'ята група місій має системний характер і стосується підтримання працездатності рою в умовах деградації: відмови агентів, втрати зв'язку, відхилення сенсорних показників, зміни середовища. Для ройових систем здатність до відновлення функцій може бути не менш важливою, ніж виконання базового сценарію. Це пов'язано з властивістю рою як системи із надлишковістю: функції

можуть бути перерозподілені між агентами, а місія може продовжуватися навіть при втраті частини елементів [1, 3]. Однак така адаптивність вимагає явних механізмів виявлення відмов, подієвого реагування та коректного узгодження між агентами.

Важливо підкреслити, що в реальних застосуваннях місії рою рідко існують у «чистому» вигляді, а найчастіше вони комбінуються: покриття поєднується зі спостереженням, супровід – з перехопленням, і будь-яка місія передбачає необхідність уникнення зіткнень та відновлення зв'язності. Це означає, що керування агентом повинно підтримувати перемикання між режимами, пріоритетність реакцій та обробку подій різної критичності, тому проявляється потреба у моделях, які дозволяють не лише описати логіку поведінки, але й зробити її керованою з точки зору часу реакції та обчислювальних витрат – особливо у вбудованому виконанні [31, 37].

1.1.3 Основні обмеження ройових систем БПЛА

Практичне впровадження ройових систем БПЛА пов'язане з низкою фундаментальних обмежень, які впливають як на архітектуру керування, так і на допустимі алгоритмічні рішення [38]. Особливо гостре це питання постає при віднесенні БПЛА до військових систем, тому що реальні рої функціонують у фізичному середовищі, де кожен агент має обмежені ресурси, канали зв'язку піддаються завадам, а часові затримки можуть критично впливати на результат місії [3], що кардинально їх відрізняє від абстрактних моделей мультиагентних систем з сталим або прогнозованим оточенням.

Одним із ключових обмежень є динамічна топологія мережі взаємодії, яка виникає у зв'язку з постійною зміною взаємного розташування БПЛА у польоті, що призводить до постійної перебудови графа зв'язності:

$$G(t) = (A, R(t)), \quad (1.3)$$

де множина ребер $R(t)$ залежить від відстані між агентами, потужності передавача та умов середовища. Теорія консенсусу показує, що збереження зв'язності графа є необхідною умовою досягнення узгодженої поведінки [7, 42], але в реальних умовах зв'язність може бути порушена через втрату сигналу, затінення перешкодами або обмежену пропускну здатність каналів [21, 22], що вимагає від алгоритмів керування стійкості до тимчасової фрагментації рою.

Другим суттєвим обмеженням є затримки передавання даних та їхня нестабільність, коли інформація про стан сусідніх агентів або про появу цілі надходить із певною затримкою τ , яка може бути випадковою величиною, а за наявності затримок колективна динаміка може втрачати стабільність або переходити в коливальний режим [7]. У задачах перехоплення чи уникнення зіткнень навіть невелика затримка може призвести до конфліктних траєкторій [13], тому часові характеристики системи розглядаються як невід'ємний елемент моделі керування.

Наступним обмеженням є енергетичний ресурс агентів (БПЛА мають обмежену ємність акумуляторів), а отже тривалість місії визначається не лише алгоритмічною ефективністю, але й енергоспоживанням обчислювальної платформи, сенсорів та комунікаційних модулів [54, 55]. У ройових місіях часто передбачаються складні маневри або постійний обмін повідомленнями, що збільшує навантаження на процесор і радіомодуль, тому виникає необхідність мінімізувати надлишкові обчислення та уникати постійного циклічного опитування станів, якщо це не є обґрунтовано необхідним [28, 29].

Обмеження обчислювальної потужності та пам'яті вбудованих систем становлять окрему групу, тому що на відміну від симуляційних середовищ, реальні БПЛА оснащені мікроконтролерами або малопотужними процесорами, які повинні одночасно виконувати задачі стабілізації польоту, обробки сенсорних даних, навігації та координації з іншими агентами, що робить пріоритетними підходи до подієвого керування та мінімізації кількості «холостих» перевірок умов [30, 31, 35].

Втрата БПЛА через технічну несправність, розрядження акумулятора або вплив зовнішніх чинників не повинна призводити до руйнування всієї місії, що робить ризик відмов окремих агентів суттєвим фактором для спостереження, тому ройова

система повинна мати функціональну надлишковість та здатність до реконфігурації [1, 24].

Сучасні ройові системи також стикаються з загрозами безпеки та кіберфізичними ризиками, коли порушення зв'язку, підміна даних та атаки на канали координації можуть змінити поведінку рою або спричинити його дезорганізацію [25, 26, 48].

1.2 Прикладні сценарії з підвищеними вимогами до реактивності

Хоча і показано, що ройові системи БПЛА функціонують в умовах обмежених ресурсів, змінної топології зв'язку та можливих відмов, але не всі прикладні задачі однаково чутливі до цих обмежень [27]. Існує клас сценаріїв, у яких визначальним параметром стає не лише коректність алгоритму чи повнота покриття зони, а саме час реакції системи на подію. У таких випадках навіть незначна затримка у прийнятті рішення або в обміні інформацією може суттєво знизити ефективність місії або зробити її невиконуваною [13, 24].

Сценарії з підвищеними вимогами до реактивності характерні для задач уникнення зіткнень, аварійного маневрування, перехоплення рухомих цілей, захисту об'єктів та динамічної перебудови рою при втраті агентів. У таких ситуаціях система повинна переходити в новий стан практично миттєво після появи тригерної події – наприклад, входження об'єкта в зону виявлення або отримання сигналу про відмову сусіднього агента. На відміну від планових режимів патрулювання чи моніторингу, де допустимі інтервали оновлення можуть вимірюватися секундами, у реактивних сценаріях мова йде про десятки або сотні мілісекунд [31, 37].

Проблема полягає у тому, що традиційні механізми керування, засновані на циклічному опитуванні станів або періодичному перегляді умов, створюють додаткове обчислювальне навантаження і формують затримку, залежну від інтервалу циклу. Натомість подієво-орієнтовані підходи передбачають ініціацію переходів безпосередньо в момент виникнення події, що потенційно дозволяє мінімізувати час реакції та стабілізувати його розкид [31, 35].

1.2.1 Сценарії уникнення зіткнень

Уникнення зіткнень є одним із базових і водночас найкритичніших сценаріїв функціонування ройових систем БПЛА. На відміну від одиночного апарата, де алгоритм запобігання зіткненням розглядається як допоміжний модуль автопілота, у рої ця задача набуває колективного характеру: зіткнення можливі не лише з перешкодами середовища, а й між самими агентами [1, 13].

У загальному вигляді сценарій уникнення зіткнень виникає тоді, коли відстань між агентами $d_{ij}(t)$ або між агентом і перешкодою зменшується до критичного порогу d_{min} . Формально умову небезпеки можна записати як $d_{ij}(t) \leq d_{min}$, тобто агенти не лише перебувають близько, а й рухаються назустріч один одному. У такій ситуації система повинна ініціювати реакцію раніше, ніж відбудеться фізичний контакт. Ключовим параметром стає час доступної реакції

$$T_{react} = (d_{ij} - d_{safe})/v_{rel}, \quad (1.4)$$

де v_{rel} – відносна швидкість зближення. Якщо сумарна затримка спрацювання керуючого механізму перевищує T_{react} , уникнення стає неможливим.

У літературі виділяють декілька підходів до реалізації уникнення зіткнень у роях: потенціальні поля, локальні правила вирівнювання та відштовхування, геометричні методи перетину траєкторій, а також підходи на основі розв’язання задач оптимізації в реальному часі [7, 13, 42]. Проте більшість із них передбачає періодичний перегляд стану сусідів або прогноз траєкторій, що створює обчислювальне навантаження та залежність часу реакції від частоти циклу.

Для ройових систем характерним є також каскадний ефект конфлікту: зміна траєкторії одного агента може спричинити нові небезпечні зближення з іншими. Реакція має бути не лише швидкою, а й узгодженою, що вимагає використання механізму поширення події «загроза зіткнення» між агентами або принаймні локального врахування змін конфігурації рою [3].

Окремий клас становлять сценарії раптової появи перешкоди, коли сенсор фіксує об'єкт на малих відстанях. Тут критичною стає затримка між моментом детекції та ініціацією маневру. Якщо система використовує циклічне опитування датчиків із періодом T_{poll} , то фактична затримка може досягати $T_{poll} + T_{proc}$ (T_{proc} – час обробки). У подієвій архітектурі спрацювання може відбуватися без очікування завершення циклу, що потенційно зменшує затримку до величини, близької до часу обробки переривання або обробника події [31, 35].

Сценарії уникнення зіткнень висувають до системи керування такі вимоги:

- мінімальний і передбачуваний час реакції;
- стабільність затримки при зростанні кількості агентів;
- локальна автономність прийняття рішення;
- узгодженість маневрів у межах групи;
- стійкість до короткочасної втрати зв'язку.

Ці вимоги безпосередньо підводять до необхідності формалізованої моделі, у якій умови небезпеки, події спрацювання та переходи між режимами задаються явно, а механізм їх виконання не залежить від частоти циклічного опитування.

1.2.2 Аварійні та нештатні ситуації

Аварійні та нештатні ситуації у ройових системах БПЛА відрізняються від сценаріїв уникнення зіткнень тим, що вони можуть виникати не лише внаслідок взаємодії агентів або перешкод, а й через внутрішні відмови, деградацію сенсорів, втрату зв'язку чи некоректні дані. У таких випадках реакція повинна бути не просто швидкою, а ще й правильною з точки зору збереження цілісності рою та виконання місії [1, 24].

До типових аварійних ситуацій належать:

- відмова або деградація сенсора (GPS, IMU, далекомір);
- різке зниження напруги живлення;
- втрата каналу зв'язку з групою;
- вихід агента за межі допустимої зони;

- некоректні або суперечливі телеметричні дані;
- виявлення зовнішнього втручання у канал керування [48].

Формально нештатну ситуацію можна описати як подію E_{fault} , що порушує інваріант нормального режиму функціонування:

$$\Phi_{\text{norm}}(x, t) = 1 \Rightarrow \Phi_{\text{norm}}(x, t^+) = 0, \quad (1.5)$$

де x – вектор стану агента або рою. У момент порушення інваріанта повинна ініціюватися процедура переходу до аварійного режиму M_{safe} .

Критичним параметром є час виявлення відмови T_{detect} та час переходу до безпечного стану T_{safe} . Сумарний час реагування, що повинен бути меншим за час, після якого наслідки відмови стають незворотними (наприклад, критичне відхилення траєкторії або втрата стійкості):

$$T_{\text{total}} = T_{\text{detect}} + T_{\text{decision}} + T_{\text{act}}. \quad (1.6)$$

Якщо система базується на періодичному контролі параметрів із періодом T_{poll} , то затримка виявлення може досягати T_{poll} . У щільному рої або при великій кількості контрольованих параметрів це створює додатковий ризик накопичення помилок.

Особливістю ройових систем є те, що аварія одного агента може мати колективні наслідки [60]. Наприклад, втрата вузла, який забезпечував зв'язність між двома підгрупами, може призвести до розпаду графа взаємодії $G(t)$. Аналогічно, відмова агента, що виконував ключову роль (лідер, ретранслятор, носій сенсора з унікальними характеристиками), вимагає перерозподілу функцій у групі [3].

У задачах оборонного або перехоплювального характеру нештатні ситуації можуть бути спричинені активною протидією – спробами подавлення зв'язку, введенням хибних координат або атаками на канал телеметрії [48].

З точки зору архітектури керування, аварійні сценарії потребують:

- явного опису умов спрацювання аварійного режиму;
- гарантованого пріоритету обробки критичних подій;

- можливості переривання менш важливих процесів;
- визначених безпечних станів (зависання, повернення на базу, аварійна посадка, розосередження);
- механізму повідомлення рою про зміну статусу агента.

Особливої уваги заслуговує питання каскадної стабільності. Якщо механізм реагування реалізовано неузгоджено, одночасний перехід декількох агентів у безпечний режим може спричинити нові конфлікти траєкторій або втрату покриття.

У цьому контексті подієво-орієнтована архітектура має принципову перевагу: критичні сигнали можуть оброблятися через механізми переривань або слухачів подій, що мінімізує час виявлення та забезпечує пріоритетність реакції [31, 35].

1.2.3 Сценарії перехоплення та оборони об'єктів

Сценарії перехоплення та оборони об'єктів належать до найбільш складних і чутливих до часу реакції задач у ройових системах БПЛА [24]. Якщо у випадку патрулювання або моніторингу допустима певна інерційність у перебудові конфігурації рою, то в задачах перехоплення динамічної цілі або захисту об'єкта від вторгнення затримка прийняття рішення безпосередньо впливає на ймовірність успіху місії [13, 24].

Загальна постановка задачі перехоплення може бути подана як мінімізація часу зближення рою з рухомою ціллю при дотриманні обмежень на безпеку, зв'язність і енергетичний ресурс агентів. У реальних умовах ці задачі ускладнюються неповною інформацією про траєкторію цілі, затримками передавання координат та необхідністю розподілу ролей між агентами. Частина рою може виконувати функцію спостереження, інші агенти – формувати бар'єр або здійснювати маневр перехоплення. У роботах, присвячених кооперативному перехопленню, підкреслюється роль децентралізованих механізмів узгодження та швидкого прийняття локальних рішень [3, 24].

Для оборонних сценаріїв характерним є поняття реактивного радіуса загрози. Якщо ціль виявляється на відстані R_{det} від об'єкта захисту і рухається зі швидкістю v_T , то максимальний доступний час реагування $T_{avail} = R_{det} - R_{safe} v_T$.

Сумарний час системної реакції T_{total} повинен бути меншим за T_{avail} . До T_{total} входять: час виявлення, час передачі інформації, час прийняття рішення, час ініціації маневру. Якщо керування реалізоване через циклічне опитування, то затримка включає інтервал циклу, що збільшує невизначеність у моменті старту маневру. У подієвій архітектурі перехід у режим перехоплення може ініціюватися одразу після фіксації факту входження цілі в зону загрози [31].

Особливістю перехоплювальних сценаріїв є також необхідність кооперативного розподілу простору. Для підвищення ймовірності успіху рій може формувати охоплюючу конфігурацію або бар'єрну лінію, що мінімізує доступні для цілі траєкторії втечі [7].

В оборонних задачах важливою є стійкість до активної протидії. Порушення зв'язку або спотворення координат може впливати на узгодженість маневрів. Тому механізм керування повинен передбачати перевірку достовірності даних та можливість переходу до локального автономного режиму [48].

1.3 Аналіз методів програмного керування дроном-агентом

Розгляд прикладних сценаріїв із підвищеними вимогами до реактивності показав, що для рйових систем БПЛА ключовим параметром стає не лише коректність алгоритму координації, а й архітектура виконання керуючої логіки. У задачах уникнення зіткнень, реагування на аварійні події та перехоплення динамічних цілей саме організація обробки подій визначає, чи вкладеться система у допустимий час реакції [29, 31].

У вбудованих системах керування БПЛА традиційно застосовуються два базові підходи до організації логіки: циклічне опитування станів (polling-based control) та подієво-орієнтоване керування (event-driven control). Перший підхід передбачає періодичну перевірку умов у головному циклі програми або в межах планувальника

задач. Другий – ініціює виконання логіки лише в момент виникнення події або переривання [32, 33]. Обидва підходи мають свої переваги та обмеження, однак їх ефективність істотно відрізняється в умовах багатоподійності та високої щільності агентів [31].

У ройових системах проблема ускладнюється масштабом: кожен агент повинен одночасно обробляти сенсорні сигнали, повідомлення від сусідів, навігаційні задачі та внутрішні стани. При зростанні кількості агентів та подій частота перевірок у циклічній архітектурі збільшується, що може призводити до перевантаження процесора та варіативності затримки реагування. Натомість подієві механізми дозволяють зменшити кількість «холостих» операцій та стабілізувати час спрацювання переходів [32, 33, 34, 35].

1.3.1 Огляд формалізмів опису керуючих алгоритмів агентів

Виконаний аналіз методів програмного керування дроном-агентом дозволяє сформулювати узагальнену структурну модель автономного БПЛА, у якій виділяються функціональні рівні прийняття рішень та інформаційні зв'язки між ними. Така модель відображає ієрархію керування та взаємодію сенсорних, реактивних, планувальних і кооперативних компонентів (рис. 1.2). У теорії керування та розподілених систем запропоновано низку формалізмів для опису поведінки агентів: скінченні автомати, мережі Петрі, їх розширення, агентні моделі, гібридні системи тощо. Кожен із цих підходів має власні переваги – від наочності до можливості формальної верифікації або аналізу паралельності. Водночас не всі вони однаково придатні для безпосереднього «вбудовування» у програмний код агента та роботи в умовах обмежених ресурсів.

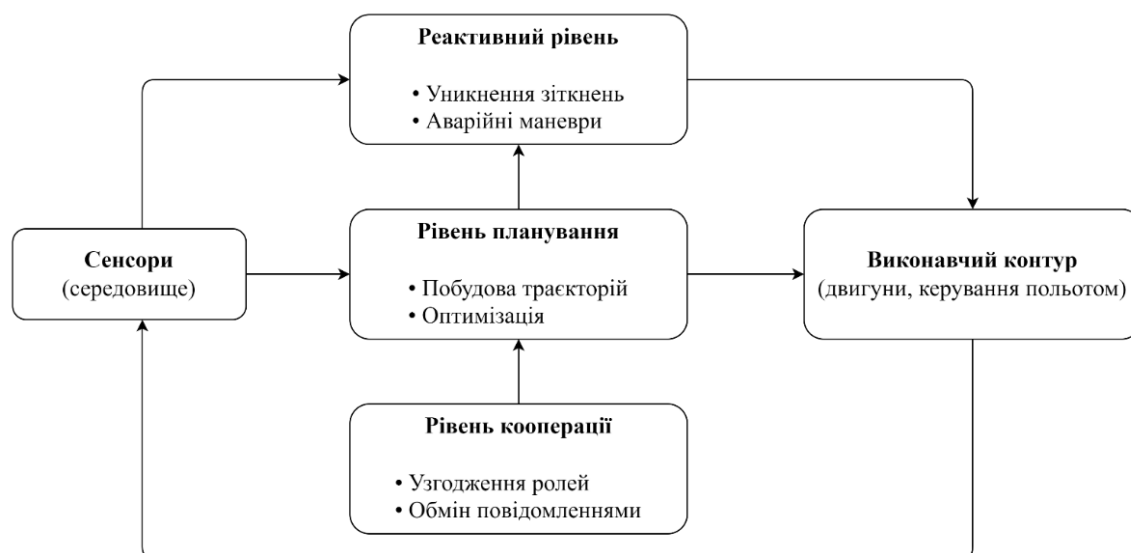


Рисунок 1.2 – Інформаційна модель агента

Скінченні автомати (finite state machines, FSM) є одним із найпоширеніших формалізмів для опису дискретної поведінки керуючих систем. Їх застосування у робототехніці та вбудованих системах зумовлене простотою інтерпретації, наочністю переходів між режимами та можливістю прямої реалізації у програмному коді. У контексті ройових систем БПЛА автомати дозволяють формально задати множину режимів агента та правила переходу між ними залежно від подій або умов [1, 61].

Формально детермінований скінченний автомат можна подати як п'ятірку:

$$A = (S, E, \delta, s_0, F), \quad (1.7)$$

де S – множина станів;

E – множина подій або вхідних символів;

δ – функція переходу ($\delta: S \times E \rightarrow S$);

s_0 – початковий стан ($s_0 \in S$);

$F \subseteq S$ – множина допустимих або кінцевих станів.

У задачах керування БПЛА станами можуть бути, наприклад, «патрулювання», «спостереження», «перехоплення», «ухилення», «аварійний режим». Події E відповідають сигналам сенсорів, повідомленням від сусідів або внутрішнім умовам

(зниження заряду, втрата зв'язку). Функція переходу визначає, у який режим переходить агент при настанні певної події.

Перевагою такого підходу є чіткість структури, коли усі допустимі переходи явно визначені, що спрощує аналіз повноти логіки та дозволяє перевірити відсутність «завислих» станів [62, 63]. Для простих сценаріїв, зокрема перемикання між режимами польоту, FSM забезпечують достатній рівень формалізації і легко інтегруються у подієву архітектуру.

В той же час мережі Петрі є одним із найбільш придатних формалізмів для опису паралельних, конкурентних та асинхронних процесів [64]. На відміну від скінченних автоматів, вони природно відображають одночасність подій, синхронізацію гілок виконання та розподіл ресурсів. Саме ці властивості роблять їх перспективними для опису поведінки агентів у ройових системах БПЛА, де одночасно відбуваються процеси навігації, комунікації, контролю дистанції та обробки аварійних сигналів.

Формально класична мережа Петрі визначається як четвірка

$$N = (P, T, F, M_0), \quad (1.8)$$

де P – множина позицій (places);

T – множина переходів (transitions);

$F \subseteq (P \times T) \cup (T \times P)$ – множина дуг;

M_0 – початкове маркування (розподіл маркерів у позиціях).

Маркер (token) у позиції відображає наявність певного стану або ресурсу. Перехід може спрацювати лише за умови наявності необхідних маркерів у всіх вхідних позиціях. Після спрацювання маркери переміщуються у вихідні позиції відповідно до структури дуг, а динаміка мережі описується зміною маркування $M(t)$.

У загальному вигляді агентна модель визначає:

- множину агентів $A = \{a_1, a_2, \dots, a_n\}$;
- середовище E ;
- локальні правила поведінки R_i для кожного агента;

– механізм взаємодії між агентами та із середовищем.

Стан системи в момент часу t описується вектором

$$X(t) = \{x_1(t), x_2(t), \dots, x_n(t)\}, \quad (1.9)$$

де $x_i(t)$ – стан окремого агента.

Динаміка системи для 1.10 визначається композицією локальних правил:

$$x_i(t + \Delta t) = f_i(x_i(t), N_i(t), E(t)), \quad (1.10)$$

де $N_i(t)$ – множина сусідів агента.

Такий підхід дозволяє моделювати емерджентну поведінку – це складні колективні властивості, що виникають як результат взаємодії простих агентів без явного програмування глобальної стратегії [65]. Для ройових систем це особливо важливо при дослідженні покриття території, формування бар'єру, самоорганізації або кооперативного перехоплення.

Імітаційні середовища (наприклад, NetLogo, Gazebo, MATLAB/Simulink тощо) забезпечують можливість проведення чисельних експериментів, аналізу метрик ефективності та оцінки впливу параметрів на результат місії [66].

1.3.2 Реагування на події у модельно-орієнтованому керуванні

Циклічне опитування станів є одним із найпоширеніших підходів до організації логіки у вбудованих системах. У його основі лежить періодичне виконання головного циклу, в межах якого здійснюється послідовна перевірка умов, зчитування сенсорів, аналіз станів та ініціація відповідних дій. Така архітектура є простою в реалізації, прозорою для відлагодження та широко застосовується в автопілотах і мікроконтролерних системах реального часу [29, 67, 68, 69].

У формальному вигляді циклічне опитування можна представити як періодичну дискретизацію безперервного процесу з кроком T_{poll} . Якщо подія виникає

в момент часу t_e , то її обробка відбудеться в найближчий момент $t_k = kT_{\text{poll}}$, де $t_k \geq t_e$. Максимальна затримка становить T_{poll} .

Середня затримка дорівнює приблизно $T_{\text{poll}}/2$. До цієї величини додається час обробки T_{proc} , що формує сумарний час реакції:

$$T_{\text{react}} = T_{\text{poll}} + T_{\text{proc}}. \quad (1.11)$$

У задачах ройових систем БПЛА, особливо в сценаріях уникнення зіткнень або перехоплення, така затримка може бути критичною, оскільки допустимий час реакції обмежується геометрією зближення та швидкістю руху агентів [13].

Однією з головних проблем циклічного підходу є залежність часу реакції від вибраного періоду опитування. Зменшення T_{poll} покращує реактивність, але водночас збільшує навантаження на процесор. Якщо в циклі перевіряється N умов або обробляється M сенсорних потоків, то обчислювальна складність одного циклу можна оцінити як $O(N + M)$. При зменшенні періоду частота виконання циклу зростає, що призводить до збільшення енергоспоживання та ризику перевантаження системи.

У ройових системах додатковою проблемою є масштабування. Якщо агент повинен перевіряти стани сусідів або обробляти повідомлення від них, кількість перевірок зростає зі збільшенням числа агентів $|A|$. У найпростішій реалізації складність може наближатися до $O(|A|)$ для кожного циклу, що робить архітектуру менш ефективною при великій чисельності рою [7].

До того ж у реальних вбудованих системах час виконання циклу може змінюватися через різну тривалість обробки умов або обмін повідомленнями, що призводить до нестабільності T_{react} [31, 61, 70, 71, 72].

Циклічне опитування також не забезпечує природної пріоритетності подій. У випадку виникнення критичного сигналу він буде оброблений лише після завершення поточного циклу або досягнення відповідного фрагмента коду. Якщо ж у цей момент

виконується менш важлива задача, це може спричинити затримку обробки аварійної ситуації.

Подієво-орієнтоване керування базується на принципі ініціації дій не через періодичну перевірку умов, а безпосередньо в момент виникнення значущої події [31, 35, 37, 62, 63, 72, 73, 75, 76, 77, 78, 79]. У такій архітектурі логіка керування активується лише тоді, коли змінюється стан системи або надходить сигнал, що потребує реакції. Це дозволяє відмовитися від постійного циклічного опитування і перейти до механізму «реакції на тригер» [31].

У загальному вигляді подія E може бути визначена як умова:

$$E: \Phi(x, t) = 1, \quad (1.12)$$

де x – вектор стану агента або рою. На відміну від циклічного підходу, у подієвій архітектурі перевірка умови не здійснюється в межах фіксованого циклу, а реалізується через механізм переривань, слухачів подій або тригерних функцій, які ініціюють перехід одразу після фіксації зміни стану.

Час реакції в такій системі можна оцінити як

$$T_{\text{react}} = T_{\text{signal}} + T_{\text{handler}}, \quad (1.13)$$

де T_{signal} – затримка фіксації події (наприклад, апаратного переривання), а T_{handler} – час виконання обробника. У цьому випадку відсутній додатковий інтервал очікування, характерний для циклічного підходу. За умови правильної реалізації це забезпечує більш передбачувану та меншу затримку реагування [31, 35].

У ройових системах БПЛА подіями можуть виступати:

- входження об'єкта в зону виявлення;
- перевищення порогового значення відстані до сусіда;
- отримання повідомлення від іншого агента;
- втрата зв'язку;
- зміна ролі або стану агента;

– аварійний сигнал від сенсора.

Ключовою перевагою подієвого підходу є зменшення кількості «холостих» операцій. Якщо події виникають із середньою інтенсивністю λ , то середня кількість активацій логіки пропорційна λ , а не частоті циклу.

Разом із тим подієво-орієнтоване керування потребує чіткої формалізації станів і переходів. Якщо обробники подій реалізовані хаотично або без єдиної моделі, система може стати важко передбачуваною. Тому подієвий підхід найбільш ефективний у поєднанні з формальною моделлю керування, де кожна подія відповідає визначеному переходу між станами.

1.4 Постановка задач дослідження та критерії ефективності

1.4.1 Формулювання наукового завдання

Аналіз ройових систем БПЛА показав, що ефективність їх функціонування визначається не лише локальними правилами взаємодії, але й способом організації керуючих алгоритмів агента. В умовах мінливого середовища, обмеженої спостережуваності та нестабільної комунікації агент повинен одночасно забезпечувати реактивність, узгодженість дій і передбачувану часову поведінку.

Існуючі підходи до реалізації алгоритмів керування в ройових системах переважно базуються на процедурному програмуванні або неформальних логічних схемах.

З іншого боку, для ройових систем характерні такі властивості:

- подієва природа зміни станів;
- одночасне виконання кількох процесів;
- необхідність синхронізації дій;
- наявність критичних режимів із жорсткими часовими обмеженнями.

Ці особливості вимагають використання формалізованої моделі, яка дозволяє:

- однозначно описати стани агента та умови переходів між ними;
- представити паралельність і синхронізацію процесів;
- відокремити логіку керування від конкретної реалізації;

– забезпечити можливість аналізу часових та структурних характеристик.

Наукове завдання полягає у розробці методу програмного керування роєм дронів як мультиагентною системою БПЛА з використанням вбудованих моделей алгоритмів поведінки дронів-агентів з підтримкою прийняття групових рішень.

Виконання даного завдання потребує розв’язання цієї задачі необхідно:

- сформулювати формальну модель станів і переходів агента;
- визначити механізм інтеграції цієї моделі з програмною архітектурою;
- забезпечити можливість аналізу реактивності та стійкості моделі;
- експериментально перевірити ефективність запропонованого підходу в умовах імітаційних сценаріїв.

1.4.2 Критерії часової ефективності

У задачах ройових систем БПЛА з підвищеними вимогами до реактивності часові характеристики керуючого механізму виступають не другорядним параметром, а визначальною умовою досягнення цілей місії. Тому критерії часової ефективності повинні бути сформульовані таким чином, щоб дозволити кількісно оцінити як швидкість реагування, так і стабільність цього реагування в умовах змінного навантаження.

1. Час реакції на подію – базовим показником є фактичний час реакції на подію t_k :

$$T_k^{react} = t_{act} - t_{event}, \quad (1.14)$$

де t_{event} – момент виникнення події;

t_{act} – момент ініціації відповідної дії (маневру, переходу режиму тощо).

Для кожного типу подій має бути визначене допустиме граничне значення:

$$T_k^{react} \leq T_k^{max}. \quad (1.15)$$

У задачах уникнення зіткнень T_k^{max} визначається відносною швидкістю зближення та дистанцією безпеки; у задачах перехоплення – швидкістю цілі та радіусом захисту.

2. Середній та максимальний час реакції – оскільки події можуть виникати з різною інтенсивністю, доцільно аналізувати:

– середній час реакції:

$$\bar{T}_{react} = \frac{1}{N} \sum_{k=1}^N T_k^{react}, \quad (1.16)$$

– максимальний (гірший випадок):

$$T_{react}^{max} = \max_k T_k^{react}. \quad (1.17)$$

Для систем із жорсткими часовими вимогами саме T_{react}^{max} має вирішальне значення.

3. Стабільність (розкид) затримки – це передбачуваність реакції є не менш важливою, ніж її середнє значення. Тому вводиться дисперсія або стандартне відхилення часу реакції:

$$\sigma_T = \sqrt{\frac{1}{N} \sum_{k=1}^N (T_k^{react} - \bar{T}_{react})^2}. \quad (1.18)$$

Низьке значення σ_T свідчить про стабільність архітектури. Для циклічних систем розкид зазвичай залежить від періоду опитування та варіативності часу виконання циклу; для подієвих – від механізму обробки переривань.

4. Латентність виявлення події – час реакції можна декомпонувати:

$$T_{react} = T_{detect} + T_{decision} + T_{act}, \quad (1.19)$$

де T_{detect} – час виявлення події;

T_{decision} – час обробки та прийняття рішення;

T_{act} – час ініціації фізичної дії.

Оцінка кожної складової дозволяє локалізувати «вузькі місця» у механізмі керування.

5. Залежність часу реакції від навантаження, що для ройових систем важливо оцінити масштабованість за кількістю агентів $|A|$ та інтенсивністю подій λ :

$$T_{\text{react}} = f(|A|, \lambda). \quad (1.20)$$

6. Частка «холостих» операцій, що дозволяє оцінити ефективність архітектури за рахунок визначення частки обчислень, що виконуються без фактичної необхідності (характерно для циклічного опитування):

$$\eta_{\text{idle}} = \frac{T_{\text{idle}}}{T_{\text{total}}}. \quad (1.21)$$

де T_{idle} – час виконання перевірок без реальних подій.

1.4.3 Критерії ресурсної та функціональної ефективності

Окрім часових характеристик, метод керування ройовою системою БПЛА повинен оцінюватися за показниками ресурсної та функціональної ефективності:

1. Ресурсна ефективність, яка визначає здатністю керуючого механізму працювати в межах обмежень вбудованої платформи без деградації стабільності.

Середнє та пікове завантаження процесора оцінюється наступним чином:

$$C_{\text{CPU}}^{\text{avg}} = \frac{T_{\text{exec}}}{T_{\text{obs}}}, C_{\text{CPU}}^{\text{max}} = \max C_{\text{CPU}}(t), \quad (1.22)$$

де T_{exec} – сумарний час виконання логіки за інтервал спостереження T_{obs} .

Метод керування повинен забезпечувати не перевищення допустимого рівня для гарантування роботи автопілота та сенсорної обробки.

Використання пам'яті передбачає оцінку:

$$C_{\text{MEM}} = C_{\text{code}} + C_{\text{data}}, \quad (1.23)$$

де C_{code} – обсяг програмного коду;

C_{data} – пам'ять для зберігання станів, черг подій, маркерів моделі.

Критерієм є відповідність обмеженням апаратної платформи щодо використання наявної пам'яті:

Оскільки обчислювальна активність впливає на енергоспоживання, вводиться оцінка питомих витрат енергії:

$$E_{\text{ctrl}} = \int_0^T P_{\text{CPU}}(t) dt. \quad (1.24)$$

Зменшення «холостих» операцій у подієвій архітектурі повинно призводити до зниження E_{ctrl} , що особливо важливо для тривалих місій [54].

Масштабованість оцінює залежність ресурсних витрат від кількості агентів $|A|$ та інтенсивності подій λ :

$$C_{\text{CPU}} = f(|A|, \lambda). \quad (1.25)$$

2. Функціональна ефективність, яка характеризує здатність системи досягати цілей місії в умовах обмежень.

Для оцінки результативності місії для різних сценаріїв вводяться відповідні метрики:

- частка успішних перехоплень;
- повнота покриття території;
- середній час нейтралізації загрози.

Для графа взаємодії $G(t)$ може оцінюватися коефіцієнт зв'язності або розмір найбільшої зв'язної компоненти. Підтримання зв'язності є критичною умовою для кооперативних сценаріїв [7].

Для оцінки адаптивності управління оцінюється швидкість перебудови конфігурації після зміни умов:

$$T_{\text{reconf}} = t_{\text{stable}} - t_{\text{change}}. \quad (1.26)$$

Менше значення T_{reconf} свідчить про вищу адаптивність.

1.5 Висновки до розділу 1

У першому розділі виконано системний аналіз стану проблеми керування ройовими системами БПЛА та визначено наукові передумови розроблення формалізованого та виконуваного методів керування агентами рою БПЛА на основі керуючих Е-мереж з подієвою інтеграцією.

Показано, що рій БПЛА доцільно розглядати як мультиагентну систему з локальними правилами взаємодії, у якій глобальна поведінка формується внаслідок взаємодії автономних агентів [7]. Проаналізовано типові місії ройових систем – спостереження, покриття території, супровід, перехоплення та оборона об'єктів – і встановлено, що для низки сценаріїв критичним фактором є мінімізація та стабільність часу реакції на події.

Визначено основні обмеження ройових систем БПЛА: ресурсні (обчислювальні, енергетичні, комунікаційні), часові та структурні. Показано, що в умовах обмежених ресурсів вбудованих платформ традиційні підходи до керування, засновані на циклічному опитуванні станів, призводять до появи варіативних затримок реагування та надлишкового обчислювального навантаження [31].

Проведено аналіз архітектурних підходів до вбудованого керування та формалізмів опису поведінки агентів. Встановлено, що:

– станкові автомати забезпечують простоту опису, але обмежені у представленні паралельності;

– мережі Петрі надають формальні засоби моделювання конкурентних процесів, однак потребують адаптації для прямого використання у програмному виконанні;

– агентні та імітаційні моделі ефективні для дослідження колективної поведінки, але не забезпечують безпосередньої інтеграції у вбудоване середовище.

На основі проведеного аналізу сформульовано наукову задачу, що полягає у вдосконаленні методів керування агентами рою БПЛА на основі модифікованого формалізму мереж Петрі з подієвою інтеграцією, який забезпечує мінімальний і стабільний час реакції за обмежених ресурсів вбудованої системи.

Визначено систему критеріїв оцінювання ефективності методу, що включає:

– часові показники (середній та максимальний час реакції, стабільність затримки);

– ресурсні показники (навантаження процесора, використання пам'яті, масштабованість);

– функціональні показники (результативність місії, стійкість до відмов, адаптивність).

У розділі також обґрунтовано необхідність переходу від традиційних циклічних механізмів керування до подієво-орієнтованої виконуваної моделі, побудованої на основі розширених мереж Петрі.

Матеріали розділу опубліковано у роботах автора: [17].

РОЗДІЛ 2

МОДЕЛІ КЕРУВАННЯ РОЄМ БПЛА ТА ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ ЙОГО ПОВЕДІНКИ

На основі обґрунтування необхідності формалізованого подієво-орієнтованого підходу до керування агентами ройових систем БПЛА з урахуванням часових і ресурсних обмежень вбудованого середовища можна здійснити перехід до формальної виконуваної моделі, що потребує попереднього дослідження поведінки рою на модельному рівні.

Використання імітаційного моделювання дозволяє [34]:

- дослідити закономірності формування колективної поведінки;
- оцінити вплив локальних правил взаємодії на глобальний результат;
- перевірити сценарії з підвищеними вимогами до реактивності;
- визначити ключові показники ефективності місії.

У роботах з ройової робототехніки показано, що локальні правила можуть породжувати узгоджену глобальну поведінку за умови збереження зв'язності та обмеженої взаємодії між агентами [7]. Разом із тим, ефективність таких систем істотно залежить від структури ролей, алгоритмів взаємодії та механізмів перерозподілу задач у разі відмов.

2.1 Мультиагентне подання рою

Якість роботи ройових систем БПЛА визначається не стільки складністю централізованого алгоритму, скільки правильністю формалізації локальної поведінки окремого агента. У цьому контексті мультиагентність виступає не лише способом опису структури рою, а й основою реалізації групового інтелекту, оскільки саме локальні рішення окремих агентів формують колективний результат місії [7].

Такий підхід має низку переваг:

- підвищення масштабованості системи;

- зменшення залежності від централізованого керування;
- стійкість до відмов окремих елементів;
- можливість самоорганізації.

Разом із тим, мультиагентне подання потребує чіткого визначення:

- структури станів агента;
- доступних сенсорних даних;
- правил переходу між режимами;
- протоколів взаємодії;
- обмежень зв'язку та спостереження.

Особливої уваги потребує формалізація локальних правил, оскільки саме вони визначають, чи виникне узгоджена колективна поведінка, чи система перейде у нестабільний режим. Відомо, що навіть прості правила вирівнювання, притягання та уникнення можуть забезпечувати стабільну формаційну поведінку за певних умов [7], однак у прикладних задачах оборони та перехоплення необхідні більш складні механізми ролей, пріоритетів та реакції на події.

2.1.1 Агент як елемент ройової системи

У межах даного дослідження агент розглядається як структурований елемент ройової моделі, поведінка якого формується поєднанням кінематичних характеристик, логічних режимів функціонування та механізмів взаємодії. На відміну від абстрактного опису мультиагентної системи, що був наведений у попередньому розділі, тут агент визначається як об'єкт імітаційної моделі з чітко формалізованими складовими.

Кожен агент характеризується трьома взаємопов'язаними підсистемами:

1. Фізична (кінематична) підсистема – визначає просторове положення, швидкість та напрям руху агента в середовищі моделювання.
2. Логічна підсистема – визначає поточний режим функціонування агента та правила переходу між режимами.

3. Комунікаційна підсистема – забезпечує обмін повідомленнями з агентами в межах радіуса взаємодії та підтримує узгодження ролей.

Агент описується як об'єкт, що має внутрішній стан:

$$S_i = \langle S_i^{phys}, S_i^{logic}, S_i^{comm} \rangle, \quad (2.1)$$

де S_i^{phys} – кінематичні параметри;

S_i^{logic} – режим поведінки;

S_i^{comm} – інформація про сусідів та ролі.

Режимна структура поведінки

Для задач оборони та перехоплення в моделі використовується скінченний набір режимів:

- чергування;
- спостереження;
- супровід цілі;
- перехоплення;
- резервування;
- повернення/вихід із задачі.

Перехід між режимами ініціюється локальними умовами: появою цілі в зоні спостереження, отриманням повідомлення від сусіда, втратою зв'язку, зниженням ресурсу або завершенням задачі.

Важливо, що зміна режиму одного агента може впливати на розподіл ролей у групі, однак не потребує централізованого диспетчера. Координація реалізується через локальні повідомлення та правила пріоритету.

У межах оборонного сценарію агент виконує одну з ролей, яка може змінюватися в процесі моделювання:

- активний перехоплювач,
- агент прикриття,
- спостерігач,
- резервний агент.

Зміна ролей дозволяє забезпечити стійкість рою до втрати окремих елементів та адаптацію до зміни конфігурації загроз.

2.1.2 Зовнішнє оточення агента-дрона

Ефективність ройової системи БПЛА визначається не лише внутрішньою логікою агентів, а й властивостями середовища, у якому вони функціонують. У більшості досліджень із ройової робототехніки середовище розглядається як динамічний простір із частковою спостережуваністю, обмеженою дальністю сенсорів і неповною інформацією про глобальний стан системи [7, 80]. Саме ці обмеження формують вимоги до структури локальних правил і механізмів взаємодії.

У межах імітаційної моделі середовище подається як двовимірний або тривимірний простір $\Omega \subset R_n$, у якому розміщуються:

- агенти рою;
- об'єкти, що захищаються;
- потенційні загрози;
- зони обмеження руху.

Кожний агент має обмежену область спостереження радіуса R_s , у межах якої він здатний виявляти об'єкти або інші агенти. Формально область видимості агента a_i визначається як:

$$\Omega_i^{obs}(t) = \{x \in \Omega: |x - X_i(t)| \leq R_s\}, \quad (2.2)$$

де $X_i(t)$ – позиція агента a_i у момент часу t ;

$|x - X_i(t)|$ – відстань між точкою x і агентом.

Таке обмеження відповідає фізичним характеристикам сенсорів БПЛА та враховує неповноту інформації про глобальний стан системи.

Окрім сенсорних обмежень, рій функціонує в умовах обмеженої комунікації. Агент може обмінюватися повідомленнями лише з тими сусідами, що знаходяться в

межах радіуса зв'язку R_c . Це формує граф взаємодії $G(t)$, структура якого визначає можливість узгоджених дій [7].

У разі зниження зв'язності можуть виникати:

- затримки передачі інформації;
- розбіжності у розподілі ролей;
- локальні конфлікти цілей.

Математична модель середовища повинна враховувати можливість часткової втрати зв'язку та асинхронність обміну даними, тому що для оборонних сценаріїв характерною є наявність рухомих загроз із непередбачуваною траєкторією [68], коли середовище має мінливий характер:

$$X_{\text{target}}(t + 1) = f_{\text{env}}(X_{\text{target}}(t)). \quad (2.3)$$

Агент не має прямого доступу до функції f_{env} , а отримує лише локальні вимірювання. Це відповідає концепції частково спостережуваного середовища, що є типовим для мультиагентних систем [7].

Модель середовища в даній роботі враховує:

- просторові обмеження;
- локальність сенсорного сприйняття;
- обмежену комунікацію;
- динамічність загроз;
- можливість часткової втрати зв'язності.

2.1.3 Обмеження спостереження та комунікації в рої

Комунікаційна підсистема є одним із визначальних факторів ефективності ройової системи БПЛА. На відміну від централізованих мереж керування, у рої відсутній єдиний вузол, що володіє повною інформацією про стан усіх агентів. Обмін даними відбувається децентралізовано, в межах локальної взаємодії, що формує часовозмінну структуру зв'язності між елементами системи [7, 69].

У типовій ройовій конфігурації кожний агент може підтримувати зв'язок лише з тими сусідами, що знаходяться в межах певного радіуса комунікації. Це означає, що структура взаємодії описується динамічним графом, топологія якого змінюється разом із переміщенням агентів. Збереження зв'язності цього графа є необхідною умовою для підтримання узгодженості рою, однак у реальних умовах така зв'язність не гарантується постійно [7, 69].

Обмеження смуги пропускання каналу зв'язку суттєво впливають на обсяг і частоту передавання даних, тому що у безпілотних системах зазвичай використовуються канали з обмеженою пропускну здатністю, а це унеможливорює постійний обмін повною інформацією про стан кожного агента, і саме тому подієве або агреговане передавання даних зменшує навантаження на канал, але водночас підвищує вимоги до локальної автономності агента [54].

Додатковою складністю комунікації є затримка передавання повідомлень, тому що час доставки пакета між агентами не є нульовим і може змінюватися залежно від умов середовища. Це призводить до того, що рішення можуть прийматися на основі застарілої інформації і призводити до неузгодженості маневрів або дублювання дій, що особливо критично у сценаріях перехоплення чи уникнення зіткнень та безпосередньо впливає на стійкість децентралізованих алгоритмів [31].

З урахуванням того, що комунікаційний обмін у рої є асинхронним, агенти не мають глобального тактового сигналу і не отримують інформацію одночасно, а це означає, що різні елементи системи можуть формувати рішення на основі різних «версій» поточного стану середовища, тому така асинхронність потребує алгоритмів, які залишаються стійкими навіть за часткової неузгодженості локальних даних [7].

У реальних умовах можливі також тимчасові розриви зв'язку, зумовлені екрануванням сигналу або збільшенням відстані між агентами, тому у таких випадках рій може фрагментуватися на кілька підгруп. Якщо алгоритм керування передбачає жорстку централізацію або глобальну синхронізацію, це може призвести до втрати функціональності [7].

Окремим аспектом є вплив щільності розміщення агентів на комунікаційне навантаження, тому що у щільних конфігураціях кількість потенційних зв'язків між

агентами зростає, а це призводить до перевантаження каналу та збільшення обчислювальних витрат на обробку повідомлень [54].

Для автономних БПЛА надмірна інтенсивність обміну даними може скорочувати тривалість виконання місії, тому що передавання повідомлень пов'язане з енергетичними витратами. Саме тому у практичних реалізаціях застосовуються подієві або адаптивні механізми комунікації, що зменшують кількість переданих пакетів без втрати суттєвої інформації [54].

2.1.4 Локальні правила взаємодії агентів

У ройових системах БПЛА правила безпеки є базовим рівнем локальної поведінки, що забезпечує фізичну цілісність агентів та стабільність колективної конфігурації. На відміну від стратегічних правил взаємодії або розподілу ролей, правила безпеки мають пріоритетний характер і спрацьовують незалежно від поточного режиму виконання місії. У багатьох роботах із колективного керування підкреслюється, що саме механізми уникнення зіткнень і підтримання дистанції формують фундамент стійкості рою [7].

Безпека руху в рої досягається через локальні взаємодії між агентами. Кожний агент приймає рішення на основі інформації про сусідів у межах області спостереження, що відповідає принципу децентралізованої самоорганізації. При цьому правила безпеки повинні бути реалізовані таким чином, щоб не порушувати глобальну структуру рою та не викликати лавиноподібних коливань траєкторій.

Уникнення зіткнень є первинною вимогою до будь-якої системи автономних БПЛА. У контексті ройової моделі ця задача розв'язується через локальне відштовхування між агентами при зменшенні міжагентної дистанції нижче критичного порогу. Якщо X_i та X_j – положення двох агентів, то умова потенційної небезпеки зіткнення визначається як:

$$\|X_i - X_j\| \leq d_{\text{crit}}, \quad (2.4)$$

де d_{crit} – критична дистанція.

У відповідь на таку подію агент формує керуючий вплив, спрямований уздовж вектора, що з'єднує агентів, із протилежним напрямом, що узгоджується з класичними моделями колективного руху, де вводиться складова відштовхування, що запобігає перетину траєкторій [7].

У випадку мінливого середовища задача ускладнюється необхідністю врахування швидкості зближення. Якщо відносна швидкість агентів спрямована назустріч, то навіть при відстані, більшій за d_{crit} , може виникнути ризик зіткнення. Тому в моделі доцільно враховувати прогнозовану дистанцію через короткий часовий інтервал:

$$\|X_i(t + \Delta t) - X_j(t + \Delta t)\|. \quad (2.5)$$

Важливо, що правило безпеки має вищий пріоритет порівняно з місіонними діями (перехоплення, супровід). Це означає, що при виявленні ризику зіткнення агент тимчасово припиняє виконання стратегічної задачі та переходить у режим маневрування.

На відміну від уникнення прямого зіткнення, підтримання мінімальної дистанції забезпечує збереження стабільної формації рою та запобігає надмірному зближенню агентів. У межах моделі вводиться параметр d_{min} , що визначає бажану мінімальну відстань між сусідніми агентами:

$$\|X_i - X_j\| \geq d_{\text{min}}. \quad (2.6)$$

Якщо відстань між агентами стає меншою за d_{min} , формується відштовхувальна складова керування, що поступово збільшує інтервал між ними. На відміну від критичного порогу d_{crit} , де реакція має різкий характер, підтримання мінімальної дистанції реалізується плавним регулюванням траєкторії.

Таке правило виконує дві функції: зменшує ризик зіткнення в умовах затримок комунікації або шуму вимірювань і забезпечує рівномірний розподіл агентів у просторі, що підвищує ефективність покриття території.

У колективних алгоритмах керування рухом ця складова часто поєднується з компонентами вирівнювання швидкості та притягання до групи, формуючи стійку конфігурацію рою [7].

2.1.5 Роль групового інтелекту у формуванні колективної поведінки

Однією з фундаментальних властивостей ройових систем є виникнення впорядкованої колективної поведінки на основі простих локальних правил. У межах мультиагентного підходу глобальна структура рою не задається централізовано, а формується як результат взаємодії автономних агентів із обмеженою інформацією про середовище та сусідів [7].

Локальні правила, розглянуті в попередніх підрозділах (безпека, вирівнювання напрямків, узгодження швидкостей), виконують роль елементарних механізмів регулювання, які при багаторазовому застосуванні формують узгоджену макродинаміку. У теорії колективного руху показано, що поєднання компонент відштовхування, притягання та вирівнювання створює умови для асимптотичної стабілізації формації за відсутності централізованого керування [7].

Важливо підкреслити, що локальні правила не потребують знання глобальної мети всіма агентами. Кожний агент діє відповідно до свого поточного стану та інформації від сусідів. Проте завдяки структурі взаємодії формується узгоджений рух групи, що може інтерпретуватися як колективне прийняття рішення.

Роль локальних правил особливо проявляється в умовах часткової спостережуваності та обмеженої комунікації. Якщо кожний агент реагує лише на події в межах власної області сприйняття, то стабільність рою визначається коректністю цих реакцій. Неправильно налаштовані локальні параметри можуть призводити до осциляцій, фрагментації групи або втрати зв'язності.

У задачах оборони та перехоплення локальні правила визначають механізм розподілу ролей без централізованого призначення. Наприклад, агент, який першим виявив загрозу, може ініціювати перехід до режиму перехоплення, тоді як сусіди переходять у режим прикриття або резерву.

Додавання нових агентів не потребує зміни глобального алгоритму – достатньо, щоб вони дотримувалися тих самих правил взаємодії. Це відповідає принципу інваріантності структури алгоритму відносно кількості елементів системи [7].

При описанні емерджентної поведінки рою можна використати перехід від мікрорівня (стан окремого агента) до макрорівня (формація, покриття, синхронний рух), тоді стабільність макроструктури буде залежати від балансу між складовими відштовхування, вирівнювання та притягання. Тоді зміна вагових коефіцієнтів локальних правил може призводити до різних режимів – від компактної формації до розподіленого покриття території.

Хоча локальні правила і не гарантують оптимальності глобального рішення (не обов'язково мінімізують час виконання місії або витрати ресурсу), але вони забезпечують адаптивність і стійкість системи, тому в прикладних задачах необхідно доповнювати базові координаційні механізми спеціалізованими правилами для конкретних сценаріїв.

Особливу роль локальні правила відіграють у забезпеченні відмовостійкості, що передбачає автоматичну перебудову конфігурації відповідно до тих самих правил взаємодії, якщо один агент виходить із ладу.

З теоретичної точки зору локальні правила можна розглядати як розподілений регулятор, що діє на основі інформації про найближче оточення, коли за певних умов і параметрів такий регулятор забезпечує збіжність траєкторій агентів до узгодженого режиму руху [7].

У межах даної роботи локальні правила розглядаються не лише як інструмент формування колективної поведінки, а й як основа для подальшої формалізації в мережевій моделі керування, що визначають набір подій, умов та переходів, які будуть інтегровані у виконувану модель агента, а емерджентна узгодженість

локальних дій формує груповий інтелект рою, який у подальшому використовується як базова властивість при побудові оборонних сценаріїв.

2.2 Формальна модель керування на основі керуючих Е-мереж

Формалізація алгоритмів керування агентом рою БПЛА потребує такого математичного апарату, який дозволяє одночасно подати стани системи, умови переходу між ними, асинхронні зовнішні події, часові затримки та механізми синхронізації паралельних процесів. Класичні підходи, засновані на кінцевих автоматах або процедурних схемах, є зручними для опису лінійної логіки, однак виявляються обмеженими при моделюванні складної реактивної поведінки мультиагентних систем [73, 74].

Мережі Петрі та їх похідні давно зарекомендували себе як ефективний інструмент дослідження дискретних подієвих систем, зокрема в задачах паралелізму, взаємного блокування, синхронізації та розподіленого виконання [72, 73]. Для задач керування БПЛА особливу цінність мають ті розширення мережевого підходу, які дозволяють пов'язати структуру формальної моделі з реальними змінними стану агента, зовнішніми подіями та програмними механізмами виконання [17, 18, 75].

У межах даного дослідження як базовий формалізм обрано керуючі Е-мережі. Їх застосування дає змогу перейти від неструктурованого опису алгоритмів до виконуваної моделі, де логіка керування визначається сукупністю позицій, переходів, умов активації, часових параметрів і подій середовища [17, 75].

2.2.1 Загальні положення керуючих Е-мереж

Керуючі Е-мережі є розширенням мережевого підходу до моделювання дискретних систем керування, орієнтованим на безпосереднє подання логіки роботи керуючого алгоритму у вигляді сукупності станів, переходів, умов та подій. Їх застосування у вбудованих системах є доцільним у тих випадках, коли необхідно поєднати формальне представлення поведінки з можливістю практичної реалізації у

програмному контурі агента [17, 18, 75]. На відміну від класичних мереж Петрі, де основний акцент робиться на структурі потоків маркерів, керуючі Е-мережі додатково орієнтовані на опис керуючих залежностей, обробку зовнішніх сигналів і включення змінних, пов'язаних зі станом фізичної системи [73, 74].

У загальному вигляді керуючу Е-мережу доцільно задати як впорядкований кортеж

$$CEN = \langle P, T, F, M_0, \tau, G, V, L \rangle, \quad (2.7)$$

де $P = \{p_1, p_2, \dots, p_n\}$ – множина позицій мережі;

$T = \{t_1, t_2, \dots, t_m\}$ – множина переходів;

$F \subseteq (P \times T) \cup (T \times P)$ – множина дуг, що визначає структуру зв'язків між позиціями та переходами;

M_0 – початкове маркування мережі;

τ – функція часових затримок переходів;

G – множина логічних умов спрацювання переходів;

V – множина глобальних змінних, пов'язаних зі станом керованого об'єкта та середовища;

L – множина слухачів подій, через які мережа взаємодіє із зовнішнім світом [17, 75].

Така форма подання узгоджується із сучасним розумінням подієвого керування, у якому прийняття рішення про активацію дії визначається не циклічним переглядом усіх можливих станів, а фактом настання релевантної події та виконанням відповідних умов [31, 35, 77]. Для мультиагентних систем цей підхід є особливо важливим, оскільки дозволяє зменшити обчислювальне навантаження і підвищити масштабованість реактивної логіки [36, 37].

Позиції $p_i \in P$ використовуються для подання окремих станів, етапів або режимів роботи алгоритму керування. У найпростішому випадку наявність маркера в позиції означає активність відповідного стану. Тоді поточне маркування мережі може бути визначене як функція $M: P \rightarrow \{0,1\}$ ($M(p_i) = 1$ – означає, що позиція p_i

маркована, а $M(p_i) = 0$ – що маркер у ній відсутній). Для задач керування БПЛА це дозволяє інтерпретувати маркування як структурне подання поточного логічного стану агента: наприклад, режим спостереження, супроводу, маневру, посадки або очікування події.

Оскільки вбудована модель повинна враховувати не лише логічний стан, а й дані середовища, стан системи доцільно подавати розширеною парою

$$S(t) = \langle M(t), V(t) \rangle, \quad (2.8)$$

де $M(t)$ – маркування у момент часу t , а $V(t)$ – вектор глобальних змінних, що містить значення параметрів сенсорів, службові прапори, значення координат, швидкостей, залишку енергії, ознак загрози або повідомлень від інших агентів [75].

Переходи $t_j \in T$ описують правила зміни станів. Для кожного переходу можна визначити множину вхідних позицій $\bullet t_j$ та множину вихідних позицій $t_j \bullet$. Перехід вважається дозволеним до спрацювання, якщо в усіх його вхідних позиціях присутні необхідні маркери та істинною є логічна умова активації. Формально умову дозволеності переходу t_j можна записати так:

$$(\forall p_i \in \bullet t_j: M(p_i) = 1) \wedge (g_j(V(t)) = 1), \quad (2.9)$$

де $M(p_i, t)$ – маркування позиції;

$g_j(V(t))$ – булева функція, яка задає умову переходу та залежить від поточних значень глобальних змінних [17, 75].

Після спрацювання переходу виконується зміна маркування, яка в загальному випадку описується співвідношенням

$$M'(p) = \begin{cases} 0, & p \in \bullet t_j \setminus t_j \bullet, \\ 1, & p \in t_j \bullet, \\ M(p), & \text{в інших випадках.} \end{cases} \quad (2.10)$$

Водночас із перерозподілом маркерів може змінюватися і вектор глобальних змінних. Якщо функцію оновлення, пов'язану з переходом t_j , позначити через f_j , то

$$V(t^+) = f_j(V(t^-)), \quad (2.11)$$

де t^- і t^+ відповідають станам системи безпосередньо до та після спрацювання переходу.

Це означає, що перехід у керуючій Е-мережі є не лише елементом структурної динаміки, але й носієм керуючої дії, яка може змінювати внутрішні змінні моделі або формувати команди для виконавчих підсистем.

Важливою складовою керуючих Е-мереж є урахування часових характеристик. Якщо для переходу t_j задано часову затримку $\tau(t_j)$, то момент його фактичного завершення можна подати як

$$t_{\text{fire}} = t_{\text{enable}} + \tau(t_j), \quad (2.12)$$

де t_{enable} – момент, коли виконалися умови дозволеності переходу. Уведення часової функції дозволяє моделювати як миттєві реакції, так і затримки, пов'язані з очікуванням сенсорних даних, виконанням маневру, завершенням обчислення або синхронізацією між паралельними гілками [31, 35].

Окреме значення у запропонованому підході мають слухачі подій $l_k \in L$, які виконують роль інтерфейсу між формальною моделлю та зовнішнім середовищем. Узагальнено їх дію можна подати відображенням $l_k: E \rightarrow V$ (E – множина зовнішніх подій). У прикладному сенсі така подія може відповідати зміні висоти, надходженню повідомлення від іншого агента, виявленню перешкоди, досягненню цілі або перевищенню допустимого відхилення параметра польоту. Слухачі можуть змінювати значення змінних V , ініціювати появу маркерів у певних позиціях або безпосередньо активувати переходи, якщо це передбачено механізмом виконання моделі [17, 75].

Керуюча Е-мережа поєднує в собі три принципово важливі властивості. По-перше, вона забезпечує формальне подання структури алгоритму у вигляді мережі станів і переходів. По-друге, вона дозволяє враховувати змінні фізичного середовища та часові характеристики функціонування. По-третє, вона створює основу для виконуваної моделі, придатної до інтеграції у вбудовану архітектуру агента БПЛА [17, 18, 75]. Саме ці властивості роблять керуючі Е-мережі доцільним формалізмом для побудови методу вбудованих моделей керування в ройових системах.

2.2.2 Позиції моделі та їх інтерпретація

Однією з ключових переваг використання керуючих Е-мереж у задачах керування роєм БПЛА є можливість формального подання поведінки агента через систему позицій, кожна з яких відповідає певному логічному стану, етапу виконання алгоритму або режиму обробки подій [73, 74, 75].

У класичних мережах Петрі позиція розглядається як абстрактне місце зберігання маркера, що відображає наявність певного стану або ресурсу. Для задач керування БПЛА таке трактування є недостатнім, оскільки агент функціонує в умовах багаторівневої логіки, взаємодії із сенсорами, отримання зовнішніх повідомлень і виконання дій, що мають часові та контекстні обмеження, тому у межах керуючої Е-мережі позицію доцільно трактувати як формальний носій стану, з яким пов'язано не лише сам факт активності, а й семантику поточного етапу керування [17, 18, 75].

Нехай множина позицій мережі має вигляд

$$P = \{p_1, p_2, \dots, p_n\}. \quad (2.13)$$

Тоді кожна позиція $p_i \in P$ може бути інтерпретована як елемент множини станів агента $p_i \mapsto \sigma_i$, де σ_i – змістовий стан або підстан системи керування. На відміну від звичайної програмної змінної типу state, така інтерпретація допускає як послідовну, так і паралельну активність кількох позицій, що є важливим для опису

одночасного виконання спостереження, обробки комунікаційних повідомлень, контролю безпеки та формування керуючих впливів [72, 73].

Для агента рою БПЛА доцільно виділити кілька базових класів позицій, які відповідають різним функціональним аспектам керування:

1. Позиції режимів функціонування агента, що відображають загальні режими функціонування агента. До них можуть належати, наприклад, стани ініціалізації, очікування місії, патрулювання, супроводу цілі, маневру ухилення, повернення у зону утримання, посадки або завершення виконання задачі.

Формально множину таких позицій можна подати як підмножину $P^{\text{mode}} \subset P$, а стан режиму агента в момент часу t – як:

$$M^{\text{mode}}(t) = \{p_i \in P^{\text{mode}} \mid M(p_i, t) = 1\}. \quad (2.14)$$

Якщо система побудована за принципом взаємовиключних режимів, то в найпростішому випадку виконується умова $|M^{\text{mode}}(t)| = 1$, тобто в кожний момент часу активним є лише один базовий режим. Однак для багаторівневого керування така умова може бути послаблена, якщо частина режимів реалізується паралельно.

2. Позиції обробки подій, що пов'язані із реакцією на зовнішні події. На відміну від стабільних режимів функціонування, ці позиції відображають короточасні стани, які активуються за фактом надходження сигналу від середовища або іншого агента. Такими подіями можуть бути виявлення перешкоди, сигнал про втрату зв'язку, команда зміни ролі, попередження про небезпечне зближення, перевищення допустимого крену чи зниження залишку енергії [13, 21].

Множину таких позицій позначимо через $P^{\text{event}} \subset P$. Тоді активація позиції $p_j \in P^{\text{event}}$ може бути пов'язана з виникненням події $e_k \in E$ відповідно до правила:

$$e_k \Rightarrow M(p_j, t^+) = 1. \quad (2.15)$$

Тут t^+ означає момент часу безпосередньо після фіксації події. Така форма подання узгоджується з принципами подієво-орієнтованого керування, де активізація обробки визначається не циклічним опитуванням, а самим фактом настання події [31, 35, 77].

3. Позиції паралельних процесів, що передбачають необхідність одночасного протікання кількох процесів: навігації, контролю безпеки, аналізу середовища, підтримки комунікації та виконання локальної місії. Тому частина позицій мережі не повинна інтерпретуватись як альтернативні стани, а навпаки – як незалежні або частково незалежні компоненти поточної поведінки агента [72, 75].

Нехай $P^{par} \subset P$ – множина позицій, які можуть бути активними паралельно. Тоді для деякого моменту часу t допустимим є стан:

$$|M^{par}(t)| \geq 1, M^{par}(t) = \{p_i \in P^{par} \mid M(p_i, t) = 1\}. \quad (2.16)$$

Це означає, що агент може одночасно перебувати в декількох підстанах, наприклад, у підстані виконання місії, обробки телеметрії та контролю критичних подій. Саме така особливість робить мережевий формалізм більш придатним для ройових систем, ніж лінійні скінченні автомати.

4. Позиції синхронізації, що використовуються для узгодження паралельних гілок виконання. Такі позиції не завжди мають безпосередню фізичну інтерпретацію на рівні поведінки БПЛА, але є необхідними для формального опису моментів, коли кілька незалежних процесів мають бути зведені в єдину точку прийняття рішення. Прикладом може бути одночасне завершення аналізу сенсорної інформації, перевірки безпечності маневру та отримання підтвердження від сусіднього агента перед переходом до спільної дії [72, 73, 75].

Нехай $p_s \in P$ – позиція синхронізації. Тоді її активація може вимагати одночасної наявності маркерів у кількох попередніх позиціях:

$$\forall p_i \in P^{red}(p_s) M(p_i, t) = 1, \quad (2.17)$$

де $P^{\text{red}}(p_s)$ – множина попередніх позицій, що повинні бути активними перед синхронізованим переходом. Такі конструкції особливо важливі в сценаріях кооперативної поведінки рою, коли окремі агенти або модулі одного агента мають узгоджено завершити свої дії [8, 15].

5. Позиції очікування та часових обмежень, що на практиці використовуються для моделювання сценаріїв на кшталт очікування підтвердження команди, утримання режиму зависання протягом заданого інтервалу, контролю часу відгуку або переходу до аварійного сценарію в разі відсутності потрібної події [31, 35, 71].

Такі позиції можна визначити як підмножину $P^{\text{time}} \subset P$. Тоді для кожної позиції $p_i \in P^{\text{time}}$ доцільно ввести функцію часу перебування

$$\theta_i(t) = t - t_i^{\text{enter}}, \quad (2.18)$$

де t_i^{enter} – момент входу маркера в позицію p_i . Тоді перехід із такої позиції може бути дозволений лише за виконання часової умови $\theta_i(t) \geq \Theta_i$, де Θ_i – наперед заданий часовий поріг.

У задачах керування БПЛА позиція не повинна розглядатися лише як бінарний індикатор активності. У розширеному варіанті з кожною позицією можуть бути пов'язані дані, які конкретизують її зміст. Наприклад, позиція може містити не лише факт активності стану «маневр ухилення», але й параметри цього маневру: напрям, допустимий радіус, пріоритет або час актуальності команди. Тоді позицію p_i доцільно описувати парою $p_i = \langle \sigma_i, d_i \rangle$, де σ_i – логічний зміст позиції, а d_i – асоційований набір даних. Сукупність усіх таких даних можна подати як:

$$D(t) = \{d_i(t) \mid M(p_i, t) = 1\}. \quad (2.19)$$

З огляду на 2.19 повний стан агента набуває вигляду:

$$S(t) = \langle M(t), V(t), D(t) \rangle. \quad (2.20)$$

Це подання є важливим з практичного погляду, оскільки дозволяє безпосередньо відображати у формальній моделі параметри, що надходять із сенсорів, систем технічного зору, каналів зв'язку або підсистем локального планування [21, 22].

З урахуванням специфіки досліджуваної задачі доцільно розглядати позиції мережі як елементи багаторівневої логіки агента. На реактивному рівні позиції відповідають швидким станам, пов'язаним із негайною обробкою подій – виявленням перешкоди, реакцією на зміну висоти, відпрацюванням команди ухилення. На рівні планування вони можуть відображати етапи локального виконання місії – патрулювання сектора, вихід у задану точку, утримання позиції. На кооперативному рівні позиції пов'язані зі взаємодією з іншими агентами – передачею повідомлень, погодженням ролей, підтвердженням готовності до спільної дії [8, 15, 75].

2.2.3 Переходи та умови їх спрацювання

У межах керуючих Е-мереж переходи є ключовими елементами, що визначають динаміку зміни станів системи керування. Якщо позиції відображають стани або підстани агента, то переходи формалізують правила переходу між ними, включаючи логічні умови, часові обмеження та реакцію на зовнішні події. Така інтерпретація дозволяє описати алгоритм керування як сукупність умовно-активованих дій, що виконуються у відповідь на зміну стану середовища або внутрішніх змінних системи [73, 74, 75].

Нехай $T = \{t_1, t_2, \dots, t_m\}$ – множина переходів керуючої Е-мережі. Для кожного переходу $t_j \in T$ визначаються множина вхідних позицій $(\bullet t_j)$, множина вихідних позицій $(t_j \bullet)$, логічна умова активації $(g_j(V(t)))$, функція дії $(f_j(V))$, часова затримка $(\tau(t_j))$.

Згідно (2.10) перехід t_j вважається дозволеним (enabled), якщо одночасно виконуються структурні та логічні умови.

Перехід може бути активований лише тоді, коли:

1. виконано всі передумови за структурою мережі;
2. виконано логічні умови, що відображають стан системи.

Це принципово відрізняє керуючі Е-мережі від процедурних алгоритмів, де перевірка умов виконується послідовно і не пов'язана зі структурою моделі.

Після активації перехід виконує дві основні операції зміна маркування (у відповідності до формули (2.11)) і оновлення стану системи (у відповідності до формули (2.12)).

Для задач керування БПЛА доцільно виділити кілька типів активації переходів.

- детерміновані переходи ($g_j(V) = 1$), що завжди активуються при наявності маркерів;
- умовні (guard) переходи ($g_j(V) = \{1, 0\}$), що залежать від параметрів середовища;
- подієві переходи ($g_j(V) = \phi(e_k)$), де e_k – зовнішня подія;
- паралельні переходи, що активуються незалежно ($\exists t_i, t_j: t_i \parallel t_j$);
- синхронізаційні переходи, що вимагають активації кількох гілок ($\forall p_i \in$
 $\bullet t_j: M(p_i) = 1$).

У контексті агента рою БПЛА переходи відповідають:

- початку або завершенню маневру;
- зміні режиму роботи;
- реакції на подію;
- формуванню керуючого впливу.

Кожен перехід є не просто елементом моделі, а формалізованою дією системи. Так для реалізації алгоритму посадки БПЛА у вигляді керуючої Е-мережі необхідно передбачити наступні кроки:

- агент виконує посадку (Landing);
- перевіряється умова: $H = 0$;
- при виконанні умови – перехід у Stop;
- інакше – повторення циклу;

– якщо умова не виконується: $H \neq 0 \Rightarrow M(p_{loop}) = 1$, тоді система повертається до перевірки;

– перехід завершення передбачає виконання умови $H = 0 \Rightarrow M(p_{stop}) = 1$ (рис. 2.1).

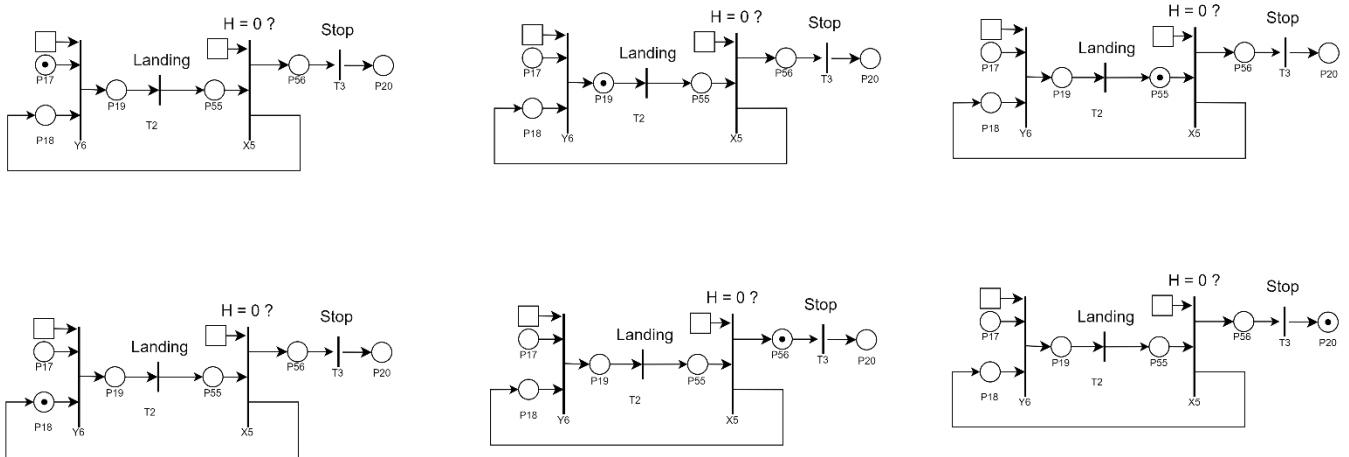


Рисунок 2.1 – Стани керуючої Е-мережі для реалізації алгоритму посадки БПЛА

Один і той самий алгоритм можна реалізувати різними структурами переходів, що означає:

$$\exists CEN_1, CEN_2, \dots, CEN_k: Behavior(CEN_i) = Behavior_{landing}, \quad (2.21)$$

але:

$$Structure(CEN_i) \neq Structure(CEN_j). \quad (2.22)$$

Різні схеми на рисунку відрізняються:

- місцем перевірки умови;
- організацією циклу;
- способом завершення;
- типом переходів.

Це дозволяє:

- оптимізувати виконання;
- змінювати реактивність;
- адаптувати модель під ресурси.

Переходи у керуючих Е-мережах є центральним елементом, що визначає динаміку системи керування. Вони поєднують у собі логіку, часові характеристики та реакцію на події, що дозволяє формально описати поведінку агента рою БПЛА як виконувану модель. Використання різних структур переходів для реалізації однакових алгоритмів забезпечує гнучкість і адаптивність підходу, що є критично важливим для систем із обмеженими ресурсами та високими вимогами до реактивності [17, 75].

2.2.4 Передавання даних і маркування

У задачах керування агентами рою БПЛА однією з ключових вимог до формальної моделі є можливість не лише опису структури переходів між станами, але й передачі даних, що характеризують стан середовища, параметри руху, результати обробки сенсорної інформації та повідомлення від інших агентів. У класичних мережах Петрі маркер (token) має абстрактний характер і не несе змістовного навантаження, тоді як у керуючих Е-мережах маркування розширюється до носія інформації, що безпосередньо впливає на прийняття рішень [72, 73, 75].

У межах керуючої Е-мережі маркер доцільно розглядати як структуру, що містить не лише ознаку активності позиції, а й пов'язані з нею дані. Формально маркер можна подати як кортеж:

$$\tau_i = \langle p_i, d_i, t_i \rangle, \quad (2.23)$$

де $p_i \in P$ – позиція, у якій знаходиться маркер;

d_i – набір даних, асоційованих із маркером;

t_i – часові характеристики (момент створення або останнього оновлення).

Множину всіх маркерів у системі позначимо як:

$$T(t) = \{\tau_i(t)\}. \quad (2.24)$$

У запропонованій реалізації маркер не розглядається як множина незалежних екземплярів, а зберігає роль індикатора активності позиції, доповненого контекстними даними. На відміну від класичних розширених мереж Петрі, де допускається наявність кількох маркерів у позиції, у даній моделі в кожній позиції може перебувати не більше одного маркера.

Розширення функціональності досягається не за рахунок множинності маркерів, а шляхом асоціювання з маркером структури даних, яка передається разом із ним під час переходів. У цьому випадку маркер виступає носієм стану та контексту виконання. Формально розширене маркування можна подати як відображення:

Розширене маркування можна подати як відображення:

$$M: P \rightarrow D, \quad (2.25)$$

де D – множина можливих структурованих даних, що асоціюються з маркером.

Згідно (2.25) кожна позиція характеризується:

- наявністю або відсутністю маркера;
- набором даних, пов'язаних із маркером.

У такій інтерпретації:

- одночасна обробка подій реалізується через паралельні гілки мережі, а не через множинність маркерів;
- накопичення інформації здійснюється шляхом модифікації структури даних, що супроводжує маркер;
- буферизація повідомлень реалізується через відповідні типи переходів (наприклад, чергові переходи), а не через накопичення маркерів у позиції.

Під час спрацювання переходу t_j виконується не лише переміщення маркерів, а й трансформація даних. Формально це можна записати як:

$$d_{\text{out}} = \psi_j(d_{\text{in}}, V(t)), \quad (2.26)$$

де d_{in} – дані вхідного маркеру;

d_{out} – дані, що формуються у вихідних позиціях;

ψ_j – функція обробки даних переходу;

$V(t)$ – глобальні змінні.

Тоді операція спрацювання переходу набуває вигляду:

$$\tau_{out} = \langle p_{out}, d_{out}, t_{fire} \rangle. \quad (2.27)$$

Представлення (2.27) означає, що перехід є не лише структурною операцією, а й виконує функцію обчислення, що відповідає концепції вбудованих моделей керування [17, 18, 75].

У задачах БПЛА часто виникають ситуації, коли:

- один перехід має кілька входів (злиття потоків);
- один перехід має кілька виходів (розгалуження).

Якщо t_j має кілька вхідних позицій, то дані об'єднуються перед обробкою:

$$d_{in} = \cup_{p_i \in \bullet t_j} d_i. \quad (2.28)$$

Якщо перехід має кілька виходів, тоді кожна гілка може отримати модифіковані дані:

$$d_{out}^k = \psi_{jk}(d_{in}), \quad k = 1, 2, \dots, r. \quad (2.29)$$

Окрім локальних даних маркерів, важливу роль відіграють глобальні змінні $V(t)$, що відображають стан системи:

$$V(t) = \{v_1(t), v_2(t), \dots, v_k(t)\}. \quad (2.30)$$

До них можуть належати:

- координати БПЛА;

- швидкість;
- висота;
- рівень заряду;
- сигнали від сенсорів;
- повідомлення від інших агентів.

Для інтеграції даних із маркерів у загальний стан системи проводиться оновлення глобальних змінних:

$$V(t + 1) = f_j(V(t), d_{in}), \quad (2.31)$$

В той же час слухачі подій забезпечують надходження нових даних у модель:

$$l_k: E \rightarrow D, \quad (2.32)$$

де E – множина подій;

D – множина даних.

Після обробки події $d_{new} = l_k(e)$ дані можуть:

- записуватись у глобальні змінні;
- ініціювати появу нового маркера:

$$\tau_{new} = \langle p_i, d_{new}, t \rangle. \quad (2.33)$$

Це дозволяє реалізувати реактивну модель керування, де зміни середовища безпосередньо впливають на поведінку системи [31, 35, 36, 37].

У контексті агента рою БПЛА передавання даних у мережі може відповідати:

- передачі координат і траєкторій;
- результатам обробки зображень;
- сигналам уникнення зіткнень;
- кооперативним повідомленням;
- параметрам місії.

Так для задачі посадки:

$$d = \{H, v_z, target\}, \quad (2.34)$$

де H – висота;

v_z – вертикальна швидкість;

$target$ – координата посадки.

Тоді умова $H = 0$ визначає перехід до стану завершення.

2.2.5 Часові затримки та обмеження

У задачах керування роєм БПЛА часові характеристики відіграють критичну роль, оскільки ефективність алгоритму визначається не лише коректністю прийнятого рішення, а й швидкістю реакції на зміну умов середовища. Наявність затримок у сенсорних підсистемах, каналах зв'язку, обчислювальних модулях і виконавчих механізмах призводить до необхідності явного врахування часу в моделі керування [31, 35, 71].

Керуючі Е-мережі дозволяють інтегрувати часові характеристики безпосередньо у формальну структуру моделі, що забезпечує можливість аналізу динаміки системи, оцінки затримок і гарантування передбачуваності поведінки. На відміну від класичних процедурних реалізацій, де час враховується неявно через затримки або цикли, у мережевому підході він є частиною формалізму.

Для кожного переходу $t_j \in T$ вводиться функція часової затримки:

$$\tau: T \rightarrow R^+, \quad (2.35)$$

де $\tau(t_j)$ – час, що необхідний для виконання переходу.

У відповідності до (2.35) момент спрацювання переходу визначається як:

$$t_{\text{fire}} = t_{\text{enable}} + \tau(t_j), \quad (2.36)$$

де t_{enable} – момент, коли виконано умови дозволеності переходу;

t_{fire} – момент фактичного завершення переходу.

Це дозволяє розрізняти:

- момент логічної готовності до виконання;
- момент завершення дії.

Для аналізу динаміки системи важливо враховувати час перебування маркера у позиції. Нехай $p_i \in P$ – позиція, тоді визначимо:

$$\theta_i(t) = t - t_i^{enter}, \quad (2.37)$$

де t_i^{enter} – момент входу маркера в позицію p_i .

Ця величина дозволяє:

- контролювати тривалість перебування у стані;
- реалізовувати таймаути;
- визначати затримки виконання.

Умови спрацювання переходів можуть залежати не лише від логічних значень змінних, а й від часу. Тоді функція активації переходів набуває вигляду:

$$g_j(V(t), \theta_i(t)) = 1. \quad (2.38)$$

Наприклад, перехід може бути дозволений лише за умови:

$$\theta_i(t) \geq \Theta_i, \quad (2.39)$$

де Θ_i – мінімальний час перебування в позиції.

Це дозволяє моделювати:

- затримку перед виконанням дії;
- стабілізаційні інтервали;
- контроль часу очікування.

У системах БПЛА особливо важливими є таймаути, що забезпечують перехід до альтернативного сценарію у випадку відсутності очікуваної події. Формально це можна записати як:

$$\theta_i(t) > \Theta_{\text{timeout}} \Rightarrow t_{\text{timeout}} \text{ активується.} \quad (2.40)$$

Це означає, якщо подія не відбулася у визначений час, то система переходить до аварійного або резервного режиму.

Такий механізм є критично важливим для забезпечення надійності та безпеки [13, 48].

Тоді час виконання послідовності переходів можна оцінити як:

$$T_{\text{total}} = \sum_{j=1}^k \tau(t_j), \quad (2.41)$$

де k – кількість переходів у сценарії.

У випадку паралельного виконання замість формули (2.41) розраховується найдовший шлях виконання:

$$T_{\text{total}} = \max_{j \in \text{Path}} \left(\sum_{j=1}^k \tau(t_j) \right). \quad (2.42)$$

Тоді час реакції визначається як інтервал між виникненням події та виконанням відповідної дії:

$$T_{\text{react}} = t_{\text{action}} - t_{\text{event}}. \quad (2.43)$$

У керуючих Е-мережах цей час включає:

- затримку активації переходу;
- час виконання переходу;
- затримки передачі даних.

Мінімізація T_{react} є одним із ключових критеріїв ефективності системи [31, 36, 37].

У контексті керування роєм БПЛА часові характеристики відповідають:

- часу обробки сенсорних даних;
- затримкам у каналах зв'язку;
- часу виконання маневру;
- часу узгодження дій між агентами;
- часу реакції на загрозу.

Наприклад, у задачі посадки, де $\tau(t_{\text{landing}}) = t_{\text{descend}}$, де t_{descend} – час зниження до поверхні, а для гарантії стабільної посадки необхідне виконання умови завершення:

$$H = 0 \wedge \theta_{\text{landing}} \geq \Theta_{\text{min}}. \quad (2.44)$$

Зміна часових параметрів переходів може суттєво впливати на:

- стабільність системи;
- частоту реакцій;
- навантаження на процесор;
- узгодженість дій агентів.

Так занадто малі значення τ призводять до перевантаження системи, занадто великі значення τ призводять до запізнення реакції.

Це підтверджується дослідженнями подієвого керування, де баланс між частотою активації та ефективністю є критичним фактором [31, 35].

2.3 Імітаційна модель оборонного рою

Після формалізації агента, середовища та локальних правил взаємодії логічним наступним кроком є перевірка того, як ці правила проявляються на рівні колективної поведінки в прикладному сценарії. Для ройових систем характерно, що глобальні властивості – покриття території, стійкість формації, здатність до перехоплення

загроз – виникають як емерджентний результат локальних взаємодій, і саме тому їх неможливо коректно оцінити лише з опису правил «на папері». Імітаційне моделювання дозволяє відтворити динаміку середовища, врахувати обмеження спостереження та комунікації, а також отримати кількісні показники ефективності для різних параметрів рою [41].

Імітаційна модель оборонного рою в NetLogo побудована як засіб відтворення групового інтелекту рою мультикоптерних дронів у задачі захисту критичних інфраструктурних об'єктів, що дозволяє отримати попередні оцінки ефективності проектних рішень за системними показниками.

Вибір імітаційного підходу особливо обґрунтований для задач оборони та перехоплення, де на поведінку рою впливають стохастичні фактори: час виявлення загрози, випадкові траєкторії цілей, можливі відмови агентів і зміна зв'язності комунікаційного графа. У таких умовах важливо дослідити не одиничний «ідеальний» прогін, а статистику за серією експериментів з визначенням чутливості результатів до параметрів і меж працездатності алгоритму [70].

Інструментом середовищного моделювання в дисертації обрано платформу агентного моделювання NetLogo, яка дозволяє швидко задавати агентів, правила їх поведінки та взаємодію із середовищем, а також виконувати серії експериментів із варіацією параметрів, та широко застосовується для досліджень у сфері складних систем і мультиагентної динаміки [41].

2.3.1 Архітектура імітаційної моделі

Імітаційна модель оборонного рою реалізована як мультиагентна система, у якій кожен елемент середовища представлений окремим типом агентів із власними станами та правилами поведінки. Архітектура моделі побудована відповідно до принципів агентного моделювання: глобальна поведінка системи виникає як результат локальних дій та взаємодій автономних об'єктів [41].

У моделі визначено структуру середовища, набір типів агентів, їх атрибути та механізми переходів між режимами. Важливо, що архітектура не передбачає

централізованого керуючого елемента. Координація досягається через локальні правила та обмін повідомленнями, що відповідає підходам до децентралізованого керування роєм [7].

Модель включає три основні категорії агентів: охоронці (елементи рою), цілі (об'єкти, що захищаються) та порушники (динамічні загрози). Така структура дозволяє формалізувати сценарій оборони як взаємодію двох активних підсистем – рою та загроз – у межах спільного середовища (рис. 2.2).

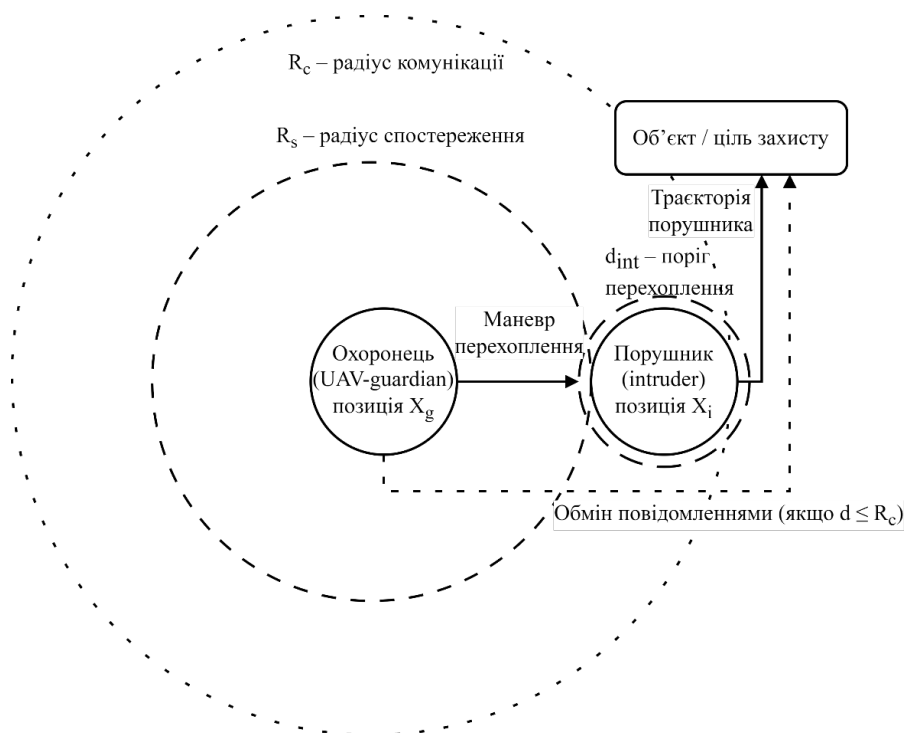


Рисунок 2.2 – Взаємодія охоронця і порушника (радіуси спостереження/зв'язку та поріг перехоплення)

Типізація агентів у моделі оборонного рою не є формальною класифікацією, а визначає функціональну структуру всієї системи. Саме через різні ролі агентів задається сценарій оборони, формуються локальні правила та оцінюються метрики ефективності. В імітаційній моделі використовується поділ на активних мобільних агентів рою, об'єкти, що захищаються, та динамічні загрози, що відповідає загальноприйнятій структурі оборонних мультиагентних систем, де середовище містить кооперативні і конфліктні підсистеми [7, 41].

Важливо підкреслити, що кожен тип агента має власну модель стану, набір параметрів та алгоритм оновлення, що реалізується у циклі симуляції. Тип агента визначає не лише його поведінку, а й спосіб участі у колективній динаміці – чи є він ініціатором взаємодії, об'єктом захисту або джерелом загрози.

Охоронці є основними активними агентами рою та моделюють БПЛА, що здійснюють оборонну місію, де кожен охоронець розглядається як автономний агент із локальною інформацією та обмеженим радіусом спостереження. Агент характеризується вектором стану, який включає просторові координати, швидкість, напрям руху, поточний режим функціонування, а також містить параметри сенсорики та комунікації.

Функціонально охоронці реалізують три групи поведінкових механізмів:

- патрулювання, що передбачає підтримання покриття зони оборони;
- виявлення, що передбачає фіксацію порушника в межах радіуса сенсорного сприйняття.

- перехоплення, що передбачає зближення з ціллю та нейтралізацію загрози.

Кожен агент діє на основі локальних правил безпеки та координації, а рішення про перехоплення приймається агентом на основі аналізу локальних подій, що відповідає принципам розподіленого керування в роях [7].

Стан охоронця змінюється дискретно відповідно до внутрішнього автомата станів («патрулювання», «супровід», «перехоплення», «резерв») і особливістю моделі є можливість резервування, що передбачає передачу функції агента іншому найближчому агенту, якщо агент, що здійснює перехоплення, втрачає зв'язок або виходить із ладу. Це дає змогу підвищити стійкість рою до часткових відмов – одну з ключових властивостей мультиагентних систем [70].

Цілі моделюють об'єкти, що захищаються, та визначають просторову структуру задачі оборони і можуть бути стаціонарними або умовно рухомими (в рамках моделі виступають як точки або області, досягнення яких порушником вважається невиконанням місії).

Порушники в запропонованому підході є динамічними агентами з власною траєкторією руху, що не координується з роєм, і їх поведінка може задаватися різними алгоритмами:

- прямолінійне наближення до цілі;
- стохастичний рух із випадковими змінами напрямку;
- адаптивний рух із урахуванням положення охоронців.

Взаємодія між охоронцями та порушниками формує ключову динаміку моделі. Якщо відстань між охоронцем і порушником стає меншою за встановлений поріг перехоплення, фіксується подія нейтралізації. Якщо ж порушник досягає області цілі, місія вважається невдалою.

З методологічної точки зору введення окремого типу порушників дозволяє розглядати рій як систему, що функціонує в умовах конфліктної взаємодії, і це наближує модель до прикладних задач оборони об'єктів і відповідає підходам, використаним у дослідженнях ройових оборонних сценаріїв [70].

2.3.2 Поведінкові стани агентів

У рамках імітаційної моделі оборонного рою поведінка охоронців структурована як набір дискретних режимів, між якими агент перемикається в залежності від подій навколишнього середовища та свого власного стану, що дозволяє формалізувати поведінку агента у вигляді керованої динамічної системи зі змінною структурою, що відповідає класичним методам моделювання автономних агентів [7, 41].

Режими охорони виступають основними станами функціонування агента в умовах відсутності безпосередньої загрози або під час очікування появи порушника. Саме ці режими визначають просторову структуру рою, рівномірність покриття території та початкову готовність до реагування. У моделі виділяються два ключових режими: чергування та активне спостереження.

Режим чергування є основним станом охоронця в умовах, коли загрози не виявлені, і головна функція полягає в підтриманні рівномірного покриття зони

оборони та збереженні зв'язності рою, коли агент здійснює рух за патрульною траєкторією:

- кругове патрулювання навколо об'єкта;
- розподілене покриття секторів;
- стохастичне переміщення в межах заданої області;
- підтримання формації з сусідніми агентами.

Агент постійно контролює простір у межах радіуса спостереження R_s , однак не здійснює агресивного маневрування. Важливо, що режим чергування спрямований не лише на економію енергії, а й на зниження хаотичності руху, що підвищує передбачуваність колективної структури рою.

З погляду колективної динаміки, саме режим чергування визначає первинну геометрію рою. Якщо алгоритм розподілу секторів реалізовано правильно, середня відстань між агентами стабілізується, а зона покриття зменшує «сліпі» області. У дослідженнях, присвячених колективному управлінню, доведено, що підтримка розподіленого патрулювання підвищує готовність системи до перехоплення загроз [7].

У симуляції NetLogo перехід із режиму чергування відбувається за умовою

$$\|x_i - x_{\text{intr}}\| \leq R_s, \quad (2.45)$$

де x_{intr} — позиція порушника, що переводить стан агента в активне спостереження.

Режим активного спостереження активується після виявлення потенційної загрози або отримання повідомлення від сусіднього агента. На відміну від чергування, цей режим характеризується підвищеною частотою оновлення траєкторії, корекцією швидкості та пріоритетною орієнтацією на об'єкт загрози.

У цьому стані агент:

- фіксує координати порушника;
- прогнозує його траєкторію;
- передає інформацію сусідам у межах радіуса R_c ;

– оцінює доцільність переходу до режиму перехоплення.

Швидкість агента може моделюватися як

$$v_i = v_{\text{patrol}} + k_{\text{obs}}(x_{\text{intr}} - x_i), \quad (2.46)$$

де k_{obs} – коефіцієнт посилення реакції на загрозу.

Активне спостереження не означає негайне перехоплення. У цьому режимі відбувається уточнення інформації про загрозу, що дозволяє уникнути хибних спрацювань та нераціональних маневрів. Якщо кілька агентів одночасно виявляють порушника, формується локальна координація, яка визначає, хто з них здійснюватиме перехоплення.

З точки зору моделювання складних систем, режим активного спостереження є перехідним станом між стабільною структурою рою та конфліктною фазою взаємодії. Саме в цьому режимі найбільш критичними є часові характеристики реагування та стійкість комунікації, що безпосередньо пов'язано з метою подальших досліджень у дисертації [70].

У симуляції перехід до режиму перехоплення відбувається за умовою:

$$\|x_i - x_{\text{intr}}\| \leq R_{\text{attack}}, \quad (2.47)$$

де R_{attack} – поріг ініціації активного маневру перехоплення.

2.3.3 Механізми взаємодії та обміну даними

Колективна поведінка оборонного рою формується не лише під впливом локальних правил руху, а також завдяки механізмам взаємодії між агентами. У рамках імітаційної моделі додатково враховується обмін інформацією, що забезпечує узгодженість рішень під час виявлення та нейтралізації загроз.

У мультиагентних системах комунікація виконує дві основні функції:

– зменшує невизначеність локального сприйняття;

– запобігає конфліктам дій між агентами [7].

У моделі оборонного рою реалізовано децентралізований механізм обміну повідомленнями в межах радіуса комунікації R_c . Це дозволяє зберегти масштабованість системи та уникнути перевантаження каналу зв'язку, що особливо важливо для ройових систем БПЛА [70].

Комунікація в моделі має подієвий характер: агент не передає інформацію постійно, а ініціює повідомлення лише при настанні значущих подій, що знижує навантаження на мережу та відповідає принципам реактивного керування.

Інформаційна взаємодія включає:

- передачу координат виявленого порушника;
- узгодження ролей між агентами;
- повідомлення про завершення перехоплення;
- сигнал про втрату контакту або відмову агента.

Формально повідомлення можна представити як кортеж:

$$m = \{id_i, type, x_{intr}, t\}, \quad (2.48)$$

де id_i – ідентифікатор агента-джерела;

$type$ – тип події;

x_{intr} – координати загрози;

t – час фіксації події.

Передача повідомлення відбувається всім агентам j , для яких виконується умова:

$$\|x_i - x_j\| \leq R_c. \quad (2.49)$$

Так формується локальний інформаційний граф, що може змінюватися у часі залежно від просторової конфігурації рою.

Повідомлення про виявлення є ініціатором реактивної фази роботи рою. Коли агент і фіксує порушника в межах сенсорного радіуса R_S , він формує подію типу «detect» та передає її сусідам.

Передаване повідомлення містить:

- координати порушника;
- оцінку швидкості;
- власну позицію;
- рівень впевненості (за наявності шумів вимірювання).

Отримавши повідомлення, сусідній агент виконує перевірку релевантності:

$$\|x_j - x_{\text{intr}}\| \leq R_{\text{interest}}, \quad (2.50)$$

де R_{interest} – радіус зацікавленості.

Якщо умова виконується, агент переходить у режим активного спостереження або супроводу.

Такий механізм дозволяє скоротити час реагування рою в порівнянні з повністю ізольованим сприйняттям, оскільки інформація поширюється локально через мережу агентів. У роботах з колективного керування показано, що навіть обмежена комунікація значно підвищує ефективність кооперативних дій [7].

У фазі перехоплення виникає потреба в координації ролей, щоб уникнути дублювання дій або конфліктів траєкторій. У моделі використовується децентралізований механізм вибору «головного перехоплювача».

Кандидатами стають агенти, для яких:

$$\|x_i - x_{\text{intr}}\| \leq R_{\text{attack}}. \quad (2.51)$$

Серед них обирається агент з мінімальною прогнозованою відстанню до точки перехоплення або з максимальною швидкістю зближення.

Після прийняття рішення формується повідомлення типу «role_assign», яке передається сусідам. Інші агенти переходять у режим супроводу або резерву.

Передача ролі також може відбуватися у випадку:

- втрати зв'язку з перехоплювачем;
- його відмови;
- зміни траєкторії порушника.

У цьому випадку найближчий агент приймає роль автоматично, що забезпечує стійкість системи до відмов. Такий механізм відповідає принципам самоорганізації та адаптивної взаємодії в ройових системах [70].

2.3.4 Візуалізація параметрів моделі в середовищі NetLogo

Середовище агентного моделювання NetLogo надає можливість інтерактивно змінювати параметри експериментальної моделі та спостерігати за динамікою поведінки агентів у реальному часі [16, 41, 81, 82]. Панель керування моделі дозволяє встановлювати значення ключових параметрів середовища та ройової системи, зокрема кількість дронів, радіус спостереження, кількість потенційних цілей, обмеження швидкості руху, а також геометричні параметри зони виявлення. Зміна цих параметрів безпосередньо впливає на характер взаємодії агентів і результати виконання місії [70].

Приклад інтерфейсу керування параметрами моделі наведено на рис. 2.3. Графічне середовище NetLogo дозволяє відображати процес виконання моделі в режимі реального часу [16, 41, 81]. На екрані моделювання використовуються умовні позначення для представлення різних типів агентів і елементів середовища. Червона область навколо об'єкта визначає зону виявлення, у межах якої дрон здатний ідентифікувати потенційну загрозу.

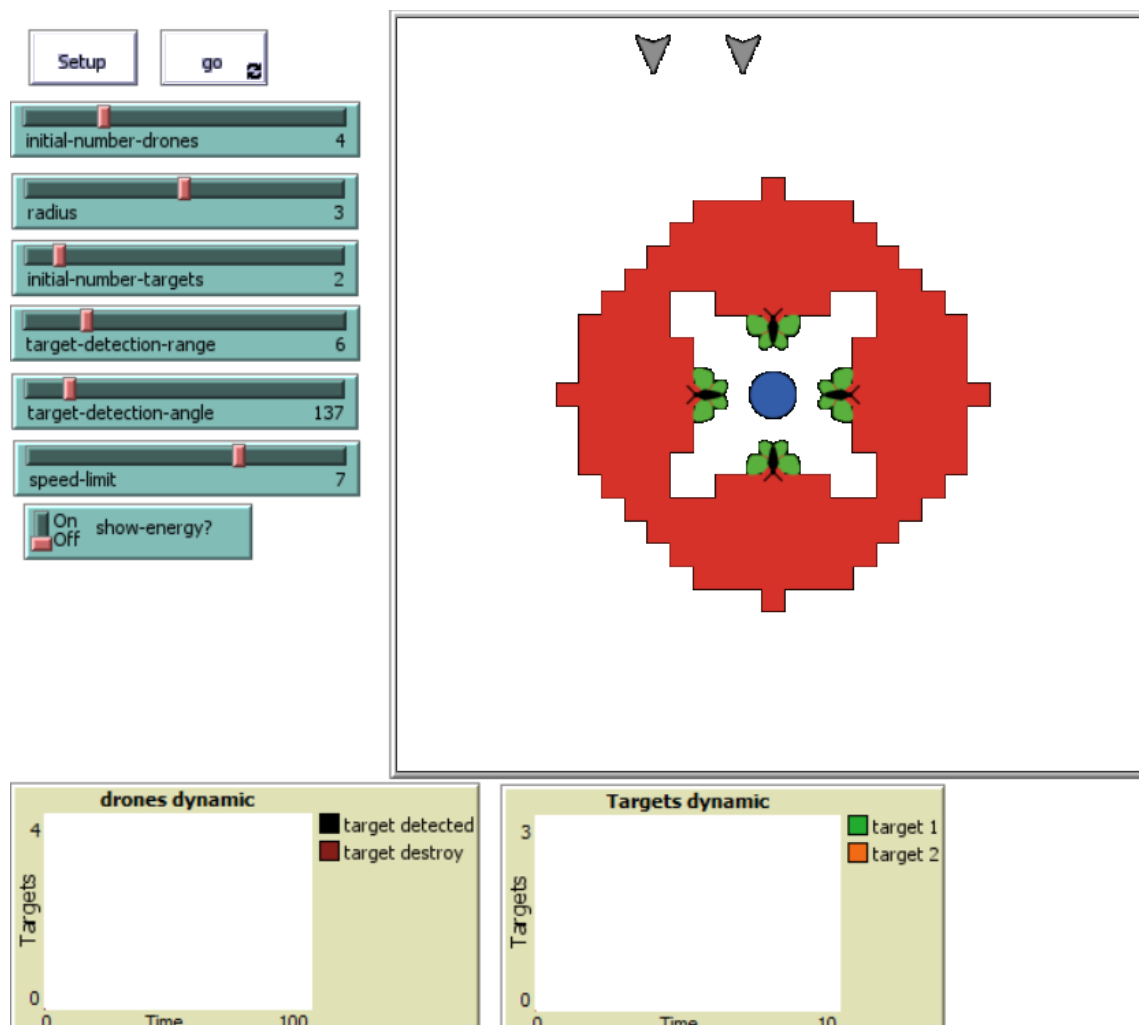


Рисунок 2.3 – Панель керування параметрами імітаційної моделі в середовищі NetLogo

Центральний об'єкт, що підлягає захисту, позначений темно-синім кольором. Така візуалізація дозволяє інтуїтивно спостерігати за розвитком сценарію оборони об'єкта під час реалізації моделі. У разі виникнення потенційної загрози та входження цілі в межі зони спостереження рою активується механізм перехоплення. Агент, що знаходиться на найменшій відстані до виявленої цілі, автоматично переходить у режим перехоплення та змінює свій стан. У моделі це відображається шляхом візуального виділення відповідного агента. Одночасно з моментом входження цілі до зони виявлення фіксується подія детектування, яка відображається на графічних діаграмах моделі.

Після знищення виявленої цілі відповідний агент може бути перенаправлений на інший об'єкт загрози, що розташований поблизу. У цей час інші агенти рою

продовжують виконувати функції спостереження та контролю території. Якщо інша ціль потрапляє до зони спостереження іншого агента, він автоматично переходить у режим перехоплення.

Після нейтралізації всіх виявлених цілей зона активного спостереження рою зменшується, оскільки частина агентів повертається до режиму чергування у своїх відповідальних секторах, що зменшує активну область контролю навколо захищеного об'єкта. У той же час агенти, які тимчасово не виконують активних дій, продовжують здійснювати моніторинг для своєчасного виявлення нових потенційних загроз.

На рис. 2.4 відображено зміну кількості цілей, що перебувають у зоні спостереження рою, у функції часу, що характеризує ефективність системи моніторингу території.

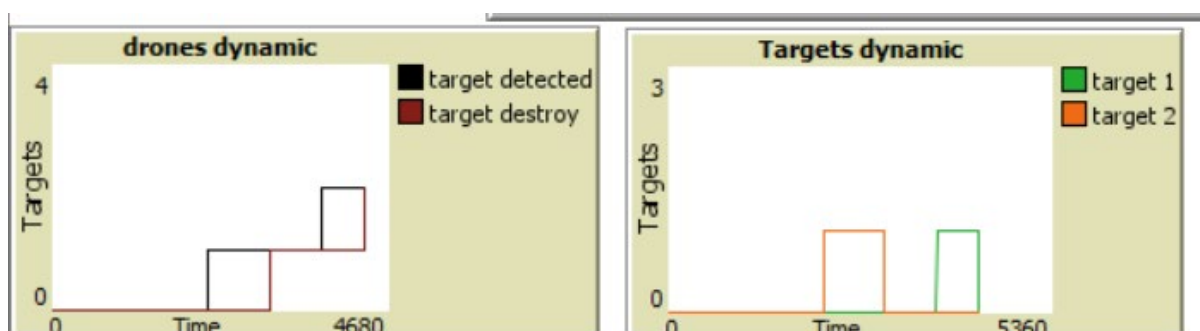


Рисунок 2.4 – Приклад результату виконання моделі

Аналіз результатів імітаційного моделювання показує, що час перехоплення цілей може істотно відрізнятися залежно від їхнього просторового розташування відносно агентів рою. Наприклад, у деяких сценаріях перша ціль знищується значно пізніше після моменту її виявлення порівняно з наступними цілями.

Використання середовища NetLogo для дослідження ройових алгоритмів дозволяє швидко змінювати параметри моделі, досліджувати різні сценарії поведінки агентів та оцінювати ефективність алгоритмів колективної взаємодії, що широко застосовується для аналізу систем колективної поведінки автономних агентів і дозволяє досліджувати складні динамічні процеси у ройових системах [70].

2.4 Метрики ефективності виконання місії

Розробка імітаційної моделі оборонного рою та формалізація поведінкових режимів агентів є лише початковим етапом дослідження. Для підтвердження адекватності запропонованих правил взаємодії та механізмів резервування необхідно перейти до кількісної оцінки їхньої ефективності. Саме на цьому етапі модель, яка спочатку є якісним інструментом аналізу, перетворюється на засіб експериментального підтвердження робочих гіпотез.

У дослідженнях мультиагентних систем підкреслюється, що оцінювання повинно включати не лише кінцевий результат (наприклад, факт перехоплення), а й динамічні характеристики процесу: час реагування, стабільність поведінки, чутливість до параметрів середовища та стійкість до відмов [7, 41].

У межах імітаційної моделі оборонного рою необхідно оцінити такі аспекти:

- швидкість виявлення загрози;
- час досягнення стану перехоплення;
- ймовірність успішної нейтралізації;
- рівень покриття зони, що захищається;
- вплив кількості агентів на ефективність місії;
- стійкість до часткових відмов.

Ці показники слід розглядати в рамках серії експериментів, що передбачають варіювання ключових параметрів: радіуса виявлення, швидкості порушника, кількості охоронців, радіуса комунікації та інтенсивності появи загроз. Це відповідає принципам сценарного аналізу, що дозволяє дослідити межі працездатності ройової системи [70].

Особливу увагу слід приділити часовим характеристикам. У контексті подальших розділів дисертації ці метрики стають містком для аналізу реактивності керуючих механізмів та обґрунтування переваг подієво-орієнтованої реалізації.

У сукупності ці метрики формують набір системних показників, за якими можна отримати попередні оцінки ефективності проектних рішень щодо організації групового інтелекту рою.

2.4.1 Часові метрики ефективності

Часові характеристики є основними показниками ефективності оборонного рою, оскільки швидкість реагування визначає можливість системи запобігти проникненню загрози до захищованого об'єкта. У мультиагентних системах час реагування формується як сукупність локальних затримок, що виникають під час сприйняття, прийняття рішень та комунікації [7].

В рамках імітаційної моделі вводяться формалізовані часові метрики, що дозволяють кількісно оцінити реактивність системи та її залежність від параметрів середовища. Основним показником у цьому підрозділі є час виявлення загрози.

Час виявлення T_{detect} визначає інтервал між появою порушника у середовищі та фіксацією факту його виявлення хоча б одним агентом рою.

Формально:

$$T_{\text{detect}} = t_{\text{detect}} - t_{\text{start}}, \quad (2.52)$$

де t_{start} – момент появи порушника;

t_{detect} – момент першої події «detect».

Ця метрика відображає здатність рою підтримувати ефективне покриття території та залежить від:

- кількості агентів;
- радіуса спостереження R_s ;
- конфігурації патрулювання;
- швидкості порушника.

Час виявлення безпосередньо впливає на подальший сценарій перехоплення. Чим меншим є T_{detect} , тим більша ймовірність успішної нейтралізації загрози до досягнення об'єкта [70].

Метрика «від першого контакту» визначає час між моментом входження порушника в радіус спостереження конкретного агента та моментом фактичної фіксації події в системі.

Формально:

$$T_{\text{contact}} = t_{\text{detect}} - t_{\text{entry}}, \quad (2.53)$$

t_{entry} – момент, коли виконується умова $\|x_i - x_{\text{intr}}\| \leq R_s$.

Цей показник відображає внутрішню затримку обробки події агентом. У рамках імітаційної моделі ця затримка визначається частотою оновлення стану та алгоритмом перевірки умов.

Якщо використовується циклічне опитування, то час першого контакту менше або дорівнює часу оновлення: $T_{\text{contact}} \leq T_{\text{cycle}}$.

У подієвій моделі ця затримка наближається до найменшого кроку симуляції, а метрика «від першого контакту» дає змогу оцінити ефективність механізму реагування без урахування геометрії рою.

Метрика «від входу в зону» визначає інтервал між моментом, коли порушник перетинає межу охоронюваної території, та моментом його виявлення.

Формально:

$$T_{\text{zone}} = t_{\text{detect}} - t_{\text{zone_entry}}, \quad (2.54)$$

де $t_{\text{zone_entry}}$ – момент входження порушника в зону оборони.

Цей показник є системним, оскільки враховує просторову структуру рою та конфігурацію патрулювання. Якщо зона покриття є нерівномірною або існують «сліпі» області, T_{zone} зростає.

Метрика дозволяє оцінити:

- якість розподілу агентів;
- вплив кількості охоронців на швидкість виявлення;
- чутливість до радіуса спостереження.

2.4.2 Метрики результативності місії

На відміну від часових характеристик, які описують динаміку реагування, метрики результативності оцінюють кінцевий ефект роботи рою з точки зору виконання місії. У контексті оборонного сценарію основним критерієм є здатність системи запобігти проникненню загрози до об'єкта, що захищається.

У дослідженнях мультиагентних систем підкреслюється, що оцінювання ефективності повинно включати як динамічні, так і цільові показники [7].

Успішність нейтралізації визначає, чи перехоплення виконано до моменту досягнення порушником зони, що захищається.

Для k -го прогону моделі вводиться бінарний показник:

$$S^{(k)} = \begin{cases} 1, & \text{якщо порушник нейтралізований} \\ 0, & \text{якщо відбувся прорив} \end{cases}. \quad (2.55)$$

Інтегральна успішність визначається як середнє значення:

$$P_{success} = \frac{1}{N} \sum_{k=1}^N S^{(k)}, \quad (2.56)$$

де N – кількість експериментальних запусків.

Цей показник відображає ймовірність виконання місії та є базовим критерієм для порівняння різних конфігурацій рою.

Частка успішних дій може розглядатися у двох інтерпретаціях:

- ймовірність успішної нейтралізації одного порушника;
- відношення кількості нейтралізованих загроз до загальної кількості атак у багатокроковому сценарії.

У другому випадку:

$$P_{eff} = \frac{N_{neutralized}}{N_{total}}, \quad (2.57)$$

де $N_{neutralized}$ – кількість успішно перехоплених порушників;

N_{total} – загальна кількість атак.

Ця метрика дозволяє оцінити не лише реакцію на одиничну загрозу, а й працездатність системи при послідовних або паралельних атаках. Зростання частки успішних дій при збільшенні кількості агентів або радіуса спостереження підтверджує ефективність масштабування системи [70].

Кількість пропусків N_{miss} – це число сценаріїв, у яких порушник досяг об'єкта, що захищається (цей показник є критичним для оборонних застосувань, оскільки навіть невелика кількість проривів може бути неприйнятною з практичної точки зору).

$$N_{miss} = N_{total} - N_{neutralized}. \quad (2.58)$$

Аналіз пропусків дозволяє:

- визначити межі працездатності рою;
- оцінити чутливість до швидкості порушника;
- виявити критичні конфігурації, при яких алгоритм стає нестійким.

Особливо показовим є аналіз залежності N_{miss} від співвідношення швидкостей $\left(\frac{v_{intr}}{v_{uav}}\right)$, оскільки при перевищенні певного порогу система втрачає можливість гарантованого перехоплення.

2.4.3 Метрики покриття та координації

Окрім безпосередньої нейтралізації загроз, ефективність оборонного рою залежить від якості контролю території. Якщо зона охорони містить «сліпі» ділянки, навіть швидке перехоплення не забезпечить своєчасного виявлення загрози. Тому в імітаційній моделі вводяться метрики покриття зони, які описують просторову структуру розміщення агентів та їх здатність контролювати визначену область.

У теорії розподіленого керування задача покриття розглядається як оптимізація розміщення агентів у просторі з урахуванням їх радіусів спостереження [7]. У прикладних дослідженнях ройових систем показано, що рівномірність покриття безпосередньо впливає на час виявлення загроз і загальну результативність місії [70].

У моделі зона охорони задається як область $\Omega \subset R^2$ (у двовимірній постановці NetLogo). Кожен агент контролює кругову область радіуса R_s . Сукупність покриття формується як об'єднання цих областей.

Відсоток покриття визначає частку площі зони Ω , яка перебуває в межах хоча б одного радіуса спостереження агента.

Формально:

$$P_{cov} = \frac{|\Omega_{covered}|}{|\Omega|} \cdot 100\%, \quad (2.59)$$

де $\Omega_{covered}$ – це об'єднання всіх кругових областей $B(x_i, R_s)$ навколо агента i .

У дискретній імітаційній моделі (на сітці NetLogo) покриття обчислюється як частка клітин, що потрапляють у межі хоча б одного сенсорного радіуса.

Значення P_{cov} залежить від:

- кількості агентів n ;
- радіуса спостереження R_s ;
- способу розподілу (патрулювання чи формаційна структура).

При малих n зона покриття може мати значні «сліпі» області, що призводить до зростання часу виявлення T_{zone} . Зі збільшенням кількості агентів відсоток покриття зростає, однак після досягнення повного перекриття додаткові агенти вже не дають суттєвого ефекту, формуючи надлишкове перекриття областей.

Повне покриття не забезпечує оптимальної організації рою. Якщо агенти зосереджуються в одній частині області, а інші ділянки контролюються лише одним агентом, система стає вразливою до локальних відмов. Тому вводиться метрика рівномірності покриття.

2.4.4 Метрики стійкості до відмов

Стійкість до відмов є однією з ключових характеристик ройових систем БПЛА, особливо в умовах оборони, де система діє у потенційно агресивному середовищі та в умовах невизначеності. На відміну від централізованих архітектур, де відмова керуючого вузла може призвести до повної втрати функціональності, ройові системи базуються на принципах функціональної надлишковості та децентралізації [7].

Стійкість аналізується в двох аспектах: структурному (втрата агентів) та параметричному (зміна характеристик системи). Для кожного з цих аспектів розробляються відповідні експериментальні сценарії та метрики.

Відмова агента в імітаційній моделі означає його повне або часткове виключення з процесу взаємодії. Це може проявлятися у вигляді зникнення з простору, втрати здатності до комунікації або припинення виконання поведінкових режимів.

Формально відмова агента i може бути описана як:

$$x_i(t) = \text{const}, \quad v_i(t) = 0, \quad \text{comm}_i = 0, \quad (2.60)$$

тобто агент припиняє рух та обмін повідомленнями.

Стійкість системи оцінюється за зміною інтегральних показників – часу перехоплення $\bar{T}_{int}$, частки успішних нейтралізацій $P_{success}$ та покриття зони P_{cov} .

Випадкові відмови моделюють ситуацію, коли окремі агенти виходять із ладу незалежно один від одного. У кожному прогоні моделі з імовірністю p_f випадковий агент стає неактивним.

Такий сценарій відображає реалістичні умови експлуатації БПЛА: втрату живлення, збої сенсорів або локальні пошкодження.

Основна метрика в цьому випадку – залежність ефективності від частки відмов: $P_{success}(p_f)$, $\bar{T}_{int}(p_f)$. Якщо при малих значеннях p_f показники змінюються незначно, це свідчить про наявність функціональної надлишковості. Збільшення часу

перехоплення при збереженні високої частки успішності вказує на адаптацію рою за рахунок перерозподілу ролей [70].

Особливо важливим є визначення критичного порогу p_{crit} , при якому система переходить у нестійкий режим, тобто різко зростає кількість пропусків.

Масові відмови моделюють втрату значної частини агентів одночасно, наприклад, унаслідок зовнішнього впливу або помилки координації.

У цьому сценарії з моделі одночасно вилучається частка агентів α , де:

$$\alpha = \frac{n_{failed}}{n_{total}}. \quad (2.61)$$

Аналізується вплив різних значень α на:

- повноту покриття;
- час виявлення;
- час перехоплення;
- частку успішних дій.

Масові відмови дозволяють оцінити межу функціональної деградації. Якщо при $\alpha = 0.3$ система зберігає прийнятний рівень результативності, можна говорити про високий рівень структурної стійкості. Якщо ж невелике збільшення α призводить до різкого зростання N_{miss} , система має низький запас міцності.

Радіус виявлення R_s безпосередньо впливає на час виявлення загрози та покриття зони. Збільшення R_s призводить до зростання площі контролю: $|\Omega_{covered}| \propto \pi R_s^2$.

Однак надмірне збільшення R_s може призвести до перекриття зон спостереження та зростання надлишковості. У серії експериментів зазвичай аналізується залежність $P_{success}(R_s)$ і $T_{detect}(R_s)$ та як правило, існує область оптимальних значень, після якої приріст ефективності стає незначним.

Кількість агентів n є одним із найважливіших параметрів рою. Теоретично збільшення n повинно зменшувати час виявлення та підвищувати частку успішних дій. Проте практично спостерігається ефект насичення.

Залежність результативності від n має нелінійний характер, але після досягнення оптимального значення n_{opt} подальше збільшення не дає суттєвого виграшу. Аналіз чутливості дозволяє визначити мінімально необхідну кількість агентів для забезпечення заданого рівня $P_{success}$. Це має практичне значення для планування складу рою.

2.5 Висновки до розділу 2

У другому розділі сформовано цілісну модель поведінки рою БПЛА на основі агентного підходу та імітаційного моделювання, що дозволило перейти від загальної постановки задачі до структурованого опису взаємодії автономних агентів у межах оборонної місії. Побудована концепція рою як децентралізованої мультиагентної системи дала змогу обґрунтувати доцільність локального прийняття рішень, що забезпечує масштабованість, відсутність критичних центрів керування та підвищену живучість системи.

Запропонована система метрик надає всебічну оцінку ефективності роботи рою. Часові показники відображають швидкість реагування, результативні – рівень виконання завдання, а просторові – ступінь покриття контрольованої території. Введення показників розкиду та довірчих інтервалів дозволяє аналізувати не лише середні значення, але й стабільність функціонування системи в серії експериментів.

Дослідження впливу випадкових та масових втрат агентів, а також варіацій радіуса виявлення і чисельності рою, дало змогу виявити критичні межі деградації та запас функціональної надлишковості. Отримані залежності демонструють характер змін результативності при порушенні структури системи та підтверджують доцільність децентралізованої організації.

Розроблена система сценаріїв імітаційних експериментів охоплює широкий спектр умов функціонування – від нормального режиму до комбінованих збурень з підвищеним навантаженням і частковими відмовами, що забезпечує репрезентативність результатів і дозволяє виявити закономірності, які не проявляються в окремих ізольованих тестах.

Аналіз трирівневої структури управління агентом виявив доцільність розмежування рівнів реагування, планування та взаємодії, що створює основу для подальшої формалізації у вигляді виконуваної програми управління, де реактивний контур забезпечує безпеку та мінімальний час відгуку, тоді як кооперативний рівень гарантує узгодженість траєкторій та оптимізацію виконання місії за часовими та енергетичними критеріями.

Матеріали розділу опубліковано у роботах автора: [16, 17, 18].

РОЗДІЛ 3

МЕТОД ПРОГРАМНОГО КЕРУВАННЯ РОЄМ ДРОНІВ НА ОСНОВІ ВБУДОВАНИХ МУЛЬТИАГЕНТНИХ МОДЕЛЕЙ

Подальший розвиток ройових систем БПЛА пов'язаний не лише з удосконаленням алгоритмів колективної поведінки, маршрутизації, розподілу ролей і координації, але й з побудовою таких засобів керування окремим агентом, які забезпечують реактивність, масштабованість і передбачуваність функціонування в умовах мінливого середовища [1, 3, 16, 21, 24]. У сучасних роботах з ройової робототехніки значна увага приділяється задачам узгодженого руху, уникнення зіткнень, коаліційної взаємодії та подієвого керування, однак архітектурне питання безпосереднього вбудовування формальної моделі в програмний контур агента висвітлено значно менше [8, 13, 14, 15, 36, 37].

Більшість практичних реалізацій систем управління безпілотними літальними апаратами (БПЛА) базуються на процедурному програмному коді, в якому логіка реагування на події, синхронізація процесів, перевірка умов та часові обмеження визначаються неявно. Це ускладнює формальний аналіз, повторне використання моделей, верифікацію поведінки та перенесення рішень між імітаційним середовищем і вбудованою платформою. Особливо це стає помітним у сценаріях, де агент повинен одночасно реагувати на сигнали сенсорів, повідомлення від інших дронів, зміни параметрів місії та виникнення аварійних ситуацій [31, 35, 71].

Необхідність удосконалення формальних засобів опису управління в мультиагентних та безпекокритичних системах підтверджується дослідженнями, в яких мережі Петрі та їх розширення використовуються для моделювання взаємодії агентів, синхронізації процесів і аналізу коректності поведінки [72, 73, 74]. Водночас для систем БПЛА цього недостатньо, оскільки формальна модель повинна не лише описувати алгоритм, а й бути придатною до прямого виконання в межах програмної архітектури агента, взаємодіяти з подіями зовнішнього середовища та підтримувати розподілену реактивну логіку [17, 18, 75].

Існує науково-прикладна потреба в методі вбудованих моделей управління, який об'єднує формальну строгість мережевого подання, підтримку подієвої активації, можливість паралельного виконання та безпосередню інтеграцію з програмним середовищем управління агентом рою БПЛА. Одним із найбільш відповідних формалізмів для цього є керуючі Е-мережі, які розвивають концепції мереж Петрі в напрямку виконуваних моделей управління [17, 18, 75, 76].

3.1 Метод керування роєм БПЛА на основі вбудованих моделей

Сам по собі формальний опис керуючої Е-мережі не гарантує ефективного функціонування системи в умовах реального середовища, де ключове значення мають питання інтеграції моделі з програмною архітектурою агента, обмеженими обчислювальними ресурсами та асинхронною природою подій.

У сучасних дослідженнях ройових БПЛА значна увага приділяється алгоритмам координації, планування траєкторій та колективної поведінки, проте питання безпосереднього вбудовування формальних моделей керування у контур роботи агента часто залишається недостатньо розкритим [1, 21, 16]. Зокрема, більшість підходів передбачає використання процедурних або модульних реалізацій, у яких логіка керування розподіляється між різними компонентами системи, що ускладнює її формальний аналіз і повторне використання [40, 83, 84].

Водночас, результати сучасних досліджень у сфері подієво-орієнтованого управління та вбудованих систем свідчать про доцільність застосування моделей, які безпосередньо інтегруються в цикл виконання програми та визначають поведінку системи на рівні її внутрішньої логіки [31, 35, 36, 37], що дозволяє забезпечити узгодженість між формальною моделлю та її реалізацією, що є критично важливим для систем реального часу та безпекокритичних застосувань.

У контексті ройових БПЛА це вказує на необхідність переходу від моделі як інструменту аналізу до моделі як активного елемента системи управління, що безпосередньо впливає на процес прийняття рішень, обробку подій та взаємодію з навколишнім середовищем. Впровадження керуючих Е-мереж у такому форматі

створює умови для розробки вбудованих моделей управління, де формальний опис алгоритму перетворюється на виконувану структуру, інтегровану з сенсорними підсистемами, каналами зв'язку та виконавчими механізмами агента [17, 75].

В той же час використання імітаційних середовищ та напівнатурного моделювання дозволяє дослідити поведінку таких моделей у наближених до реальних умовах, враховуючи обмеження апаратної платформи, затримки передачі даних і взаємодію між агентами [34, 78, 85, 86].

3.1.1 Впровадження архітектури програмної реалізації агента з вбудованою моделлю

Реалізація вбудованої моделі керування агентом рою БПЛА потребує формування такої архітектури програмної системи, у якій формальна модель не лише описує поведінку, а безпосередньо визначає логіку виконання алгоритмів у реальному часі [7, 87, 88, 89]. На відміну від традиційних підходів, де модель використовується як допоміжний інструмент аналізу, у запропонованому підході вона є центральним компонентом системи керування [17, 18, 75].

Архітектура, представлена на рис. 3.1, реалізує принцип вбудованої моделі (embedded model), відповідно до якого всі основні процеси керування організовані навколо керуючої Е-мережі. У межах цієї архітектури можна виділити чотири основні компоненти: модель, контролер, знімок стану (view) та середовище виконання [90, 91].

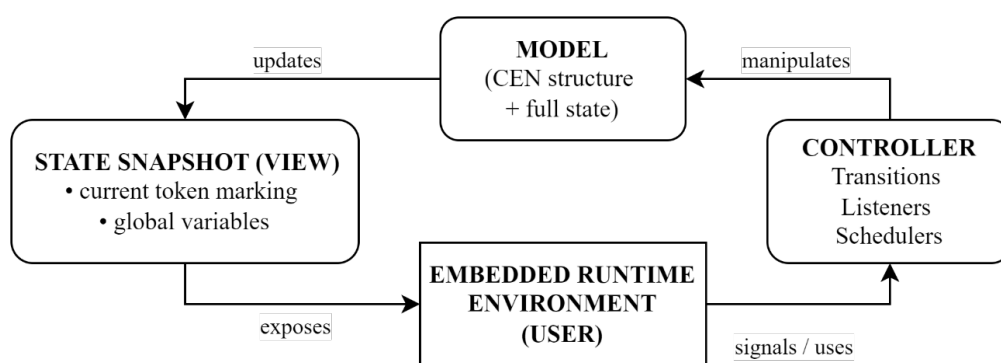


Рисунок 3.1 – MVC архітектура виконання вбудованої моделі

Компонент *MODEL* містить повне формальне представлення системи керування у вигляді керуючої Е-мережі:

$$MODEL = \langle CEN, S(t) \rangle, \quad (3.1)$$

де *CEN* – структура мережі (позиції, переходи, зв'язки);

S(t) – поточний стан (маркування + глобальні змінні + дані).

Модель виконує дві функції:

- зберігає логіку алгоритму;
- акумулює поточний стан системи.

Вона є єдиним джерелом істини щодо стану агента.

Компонент *CONTROLLER* відповідає за виконання моделі. Він включає:

- *Transitions* – механізм активації переходів;
- *Listeners* – обробники подій;
- *Schedulers* – планувальники виконання.

Формально контролер реалізує оператор переходу:

$$S(t + 1) = \Phi(S(t), E(t)), \quad (3.2)$$

де *E(t)* – множина подій.

Контролер:

- аналізує доступні переходи;
- перевіряє умови;
- виконує зміни стану;
- генерує керуючі дії.

Компонент *STATE SNAPSHOT* формує поточне представлення стану системи:

$$VIEW(t) = \langle M(t), V(t) \rangle, \quad (3.3)$$

де *M(t)* – маркування;

$V(t)$ – глобальні змінні.

Цей компонент:

- не змінює стан;
- забезпечує доступ до нього;
- використовується для моніторингу, логування або передачі даних.

Компонент EMBEDDED RUNTIME ENVIRONMENT відповідає за взаємодію із зовнішнім середовищем:

$$E(t) = \{e_1, e_2, \dots, e_n\}, \quad (3.4)$$

де e_i – події від сенсорів, каналів зв'язку, інших агентів, користувача або системи керування.

Середовище:

- генерує події;
- приймає сигнали керування;
- забезпечує інтеграцію з апаратною частиною.

Архітектура працює за циклічним принципом:

1. Середовище генерує події $E(t)$;
2. Контролер обробляє події;
3. Активуються переходи;
4. Оновлюється модель $S(t)$;
5. Формується новий стан;
6. Генеруються керуючі сигнали.

Формально цикл можна записати як:

$$S(t + 1) = \Phi(S(t), E(t)). \quad (3.5)$$

Логіка системи централізована в моделі, що дає:

- формальну перевірюваність;
- повторне використання;

- переносимість між середовищами;
- узгодженість реалізації і моделі.

У традиційній архітектурі:

- логіка розподілена по модулях;
- стан не централізований;
- складно відслідкувати поведінку.

У запропонованій архітектурі:

- модель є ядром системи;
- контролер – це виконавець моделі;
- всі рішення формалізовані.

У контексті агента БПЛА:

- MODEL – це логіка польоту та поведінки;
- CONTROLLER – це автопілот і логіка реакції;
- RUNTIME – це сенсори, зв'язок, середовище;
- VIEW – це телеметрія.

Особливості використання вбудованих моделей керування у задачах ройових БПЛА значною мірою визначається способом представлення стану системи та організацією передавання даних між її компонентами [20, 22, 23, 92]. На відміну від традиційних підходів, де стан агента задається набором змінних і структур даних у програмному коді, у запропонованому підході стан системи формується безпосередньо в межах керуючої Е-мережі та визначається її маркуванням і пов'язаними з ним даними [17, 18, 75], що дозволяє перейти від процедурного опису до формального подання стану, у якому кожен активний елемент моделі має чітку інтерпретацію та може бути використаний для прийняття рішень. Це особливо важливо для систем із високим рівнем паралелізму та асинхронності, характерних для ройових БПЛА [15, 21, 93, 94, 95, 96].

У відповідності до (3.14) стан агента вбудованої системи керування визначається не окремою змінною, а сукупністю структурних і інформаційних компонентів, що відображають як логіку виконання алгоритму, так і поточні параметри середовища.

На відміну від класичних мереж Петрі, у яких маркер є абстрактним, у керуючих Е-мережах у відповідності з (3.17) маркер виступає носієм даних, що дозволяє:

- зберігати інформацію безпосередньо в структурі моделі;
- передавати дані разом із переходами;
- реалізовувати контекстно-залежну поведінку.

Особливістю запропонованого підходу є те, що стан системи є розподіленим по позиціях мережі. Це означає, що:

$$M(t) = \{p_i \mid M(p_i, t) = 1\}, \quad (3.6)$$

а відповідні дані:

$$D(t) = \bigcup_{p_i \in M(t)} d_i. \quad (3.7)$$

Такий підхід дозволяє:

- реалізувати паралельну обробку;
- уникнути централізованих структур стану;
- забезпечити масштабованість системи.

У задачах керування роєм БПЛА дані в станах можуть включати:

$$d_i = pos, vel, target, state, msg, \quad (3.8)$$

де *pos* – координати;

vel – швидкість;

target – ціль;

state – режим;

msg – повідомлення.

Це дозволяє реалізувати:

- кооперативну взаємодію;

- обмін інформацією;
- узгоджене прийняття рішень.

Наведене формальне представлення стану створює основу для його візуалізації у вигляді маркованої мережі, яка відображає як структуру алгоритму, так і поточний стан системи, що дозволяє безпосередньо інтерпретувати активні позиції як виконувані підзадачі, а розподіл маркерів – як поточну конфігурацію поведінки агента.

3.1.2 Опис дій та умов у програмному коді

Впровадження вбудованої моделі управління на основі керуючих Е-мереж передбачає відображення формального опису переходів, умов їх активації та обробки подій у програмному коді і на відміну від традиційних методів, де логіка управління реалізується у вигляді послідовних інструкцій, у запропонованій архітектурі програмна реалізація безпосередньо відповідає структурі моделі та відтворює її поведінку в реальному часі [17, 75].

У процесі формалізації запропонованого методу програмного керування роєм БПЛА, заснованого на керуючих Е-мережах, виникає потреба у деталізації основних структурних патернів, які описують динаміку руху маркерів у мережі, де важливо відобразити умови активації переходів, а також механізми синхронізації та передачі керуючих впливів між позиціями.

На рис. 3.2 представлено фрагмент керуючої Е-мережі, що ілюструє процес активації переходів за наявності маркерів у вхідних позиціях та виконанні логічних умов. Круглі вершини на малюнку відповідають позиціям мережі, які відображають стани системи або місця перебування маркера. Переміщення маркера між позиціями відбувається через переходи, які в цій моделі представлені у вигляді лінійних елементів (вертикальних рисок), що ілюструють факт передачі керування або даних. Квадратні елементи на схемі не є переходами в класичному розумінні мереж Петрі, а виконують функцію допоміжних керуючих вузлів і використовуються для відображення логічних умов, зовнішніх подій та механізмів ініціації переходів.

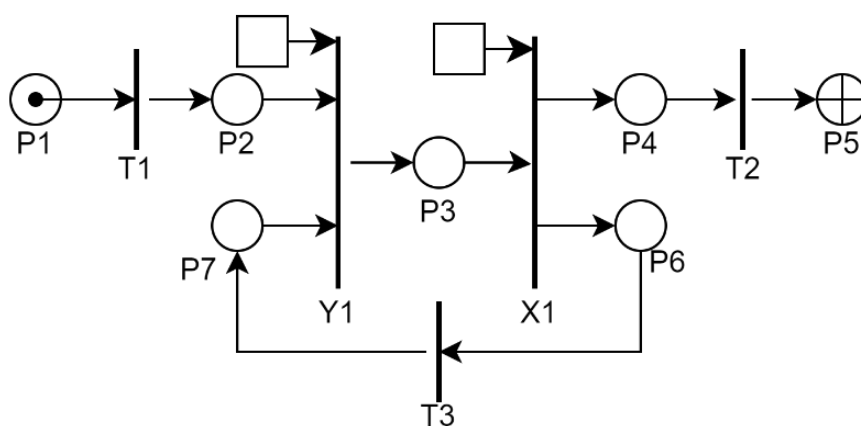


Рисунок 3.2 – Фрагмент Е-мережі з умовною активацією переходів

Наявність маркера у позиції визначається функцією маркування $M(p_i, t)$, а активація переходу додатково залежить від логічної функції $g_j(V(t))$, що враховує значення глобальних змінних системи.

На рис. 3.3 зображено початковий стан маркера, яке ініціює виконання переходу, що призводить до його переміщення у наступні позиції та подальшого запуску ланцюга обчислень.

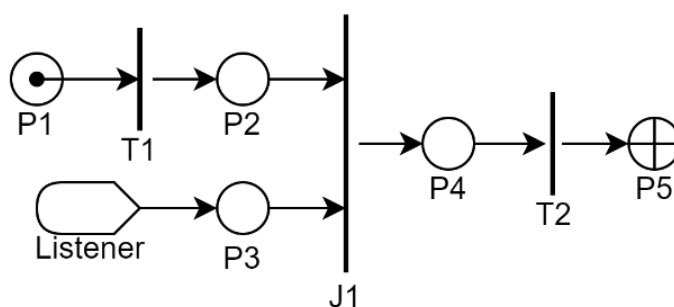


Рисунок 3.3 – Приклад поширення маркера у подієво-орієнтованій Е-мережі

Представлений механізм реалізує подієво-орієнтовану модель виконання, у якій кожна зміна стану (поява маркера) автоматично породжує подію, що може активувати наступні переходи. У результаті формується реактивний ланцюг виконання, який продовжується до досягнення кінцевого стану або виникнення необхідності очікування зовнішнього сигналу.

Формальна модель представлена формулою (2.7), а формальні переходи t_j реалізується як функція або метод.

У програмному коді це удосконалення реалізовано наступним чином:

```
def transition_landing(state):
    if state.H == 0:
        state.mode = "STOP"
    else:
        state.mode = "LANDING"
```

Така функція:

- перевіряє умову;
- змінює стан;
- виконує дію.

Умови переходів відповідають функціям активації переходів $(g_j(V(t)))$:

```
def guard_landing(state):
    return state.H == 0
```

Активація переходу:

```
if guard_landing(state):
    transition_landing(state)
```

Система працює у подієвому режимі, що у коді це реалізується через обробники:

```
def on_sensor_update(data):
    state.H = data.height
def on_message(msg):
    state.target = msg.target
```

Це дозволяє:

- реагувати лише на події;
- уникати постійного опитування;
- зменшити навантаження.

Контролер виконує цикл:

```
while True:
```

```
events = get_events()
update_state(events)
evaluate_transitions()
```

Планувальник:

- визначає порядок виконання;
- контролює частоту оновлення;
- забезпечує реактивність.

Дані передаються через стан:

- state.position;
- state.velocity;
- state.target.

Це забезпечує:

- узгодженість даних;
- доступ усіх компонентів;
- спрощення архітектури.

В той же час у системі можуть одночасно працювати кілька процесів:

```
threads = [
    navigation_loop,
    sensor_loop,
    communication_loop
]
```

Це дозволяє:

- обробляти кілька задач одночасно;
- підвищити швидкість реакції.

Програмна реалізація взаємодії з:

- сенсорами (IMU, GPS, камери);
- автопілотом (Pixhawk);
- обчислювальним модулем (Jetson);
- середовищем моделювання (Gazebo).

3.1.3 Інтеграція з підсистемами БПЛА

Принципи використання вбудованих моделей керування в ройових системах БПЛА значною мірою визначається можливістю їх інтеграції з реальними підсистемами апарата, такими як сенсорні модулі, системи навігації, системи зв'язку та виконавчими механізмами [97, 98, 99]. На відміну від ізольованих моделей, що застосовуються лише на етапі імітації, запропонований підхід передбачає безпосереднє включення керуючої Е-мережі у контур керування агентом [17, 75].

У такій архітектурі модель виступає центральним компонентом, який отримує дані від підсистем, обробляє їх відповідно до логіки мережі та формує керуючі впливи.

Інтеграцію можна формально описати як взаємодію трьох основних множин:

$$S(t + 1) = \Phi(S(t), E_{\text{sens}}(t), E_{\text{comm}}(t)), \quad (3.9)$$

де $E_{\text{sens}}(t)$ – події від сенсорів;

$E_{\text{comm}}(t)$ – події від комунікаційної підсистеми.

Вихідні сигнали:

$$U(t) = \Psi(S(t)), \quad (3.10)$$

де $U(t)$ – керуючі дії.

Сенсорні підсистеми забезпечують надходження первинної інформації про стан середовища та самого БПЛА. До них належать:

- IMU (інерційні датчики);
- GPS / GNSS;
- барометр;
- камери та системи комп'ютерного зору;
- датчики відстані.

Сенсорні дані перетворюються у події:

$$e_i^{sens} = f_{sens}(raw_i). \quad (3.11)$$

А після цього передаються в модель через слухачів:

$$l_k(e_i^{sens}) \rightarrow V(t). \quad (3.12)$$

Це дозволяє інтегрувати реальні вимірювання у процес прийняття рішень [53].

Навігаційна підсистема відповідає за визначення координат, швидкості та орієнтації БПЛА:

$$Nav(t) = \{x, y, z, v_x, v_y, v_z, \psi, \theta, \phi\}. \quad (3.13)$$

Ці дані використовуються:

- в умовах активації переходів;
- при формуванні траєкторій;
- для оцінки стану.

У ройових системах важливу роль відіграє обмін інформацією між агентами:

$$msg_i = \langle id, type, data \rangle. \quad (3.14)$$

Отримані повідомлення:

$$E_{comm}(t) = \{msg_1, msg_2, \dots\}, \quad (3.15)$$

обробляються через:

$$l_k(msg_i) \rightarrow V(t). \quad (3.16)$$

Це забезпечує:

- координацію;
- узгодження дій;
- колективну поведінку [8, 21].

Результатом роботи моделі є керуючі сигнали $U(t) = \{u_1, u_2, \dots, u_k\}$, які передаються до:

- автопілота;
- контролерів двигунів;
- систем стабілізації.

Реалізація вбудованої моделі виконується на обчислювальних модулях:

- NVIDIA Jetson;
- Raspberry Pi;
- бортових комп'ютерах.

Особливості використання цих систем:

- обмежені ресурси;
- необхідність реального часу;
- асинхронна обробка подій.

Для тестування використовується імітаційне середовище (Gazebo), що дозволяє:

- перевіряти алгоритми;
- тестувати сценарії;
- оцінювати ефективність [34, 78].

Узагальнено інтеграцію можна подати як цикл:

$$Sensors \rightarrow Model \rightarrow Control \rightarrow Actuators \rightarrow Environment. \quad (3.17)$$

Цей цикл є безперервним і подієвим.

3.2 Модифікована архітектура програми керування у стилі MVC

Побудова системи управління агентом рою БПЛА вимагає не лише формалізації алгоритмів у вигляді керуючих Е-мереж та їх інтеграції з підсистемами апарата, але й визначення програмної архітектури, яка забезпечує узгодженість між моделлю, обробкою подій та взаємодією із зовнішнім середовищем. У попередніх

підрозділах було продемонстровано, що формальна модель може бути центральним елементом системи управління, визначаючи логіку поведінки агента, але реалізація такої моделі в програмному середовищі вимагає чіткої структуризації компонентів системи, зокрема розподілу відповідальностей між зберіганням стану, виконанням логіки та обробкою зовнішніх подій. Відсутність такого розподілу призводить до ускладнення супроводу системи, зниження її масштабованості та порушення узгодженості між формальною моделлю та програмною реалізацією.

У контексті запропонованого підходу використання MVC дозволяє природним чином інтегрувати керуючі Е-мережі у програмну архітектуру, де формальна модель виступає у ролі компонента Model, механізми активації переходів реалізуються в Controller, а підсистеми сенсорів, комунікації та виконавчих механізмів взаємодіють із системою через рівень View, що забезпечує узгодженість між формальною моделлю та її виконанням, що є критично важливим для систем реального часу та ройових БПЛА [17, 18, 75].

3.2.1 Модель як виконувана Е-мережа

У запропонованому підході до побудови системи керування агентом рою БПЛА ключовою особливістю є використання формальної моделі не лише як засобу опису алгоритму, а як безпосередньо виконуваного компонента програмної системи. Це дозволяє усунути розрив між етапами моделювання та реалізації, який є характерним для більшості традиційних підходів до розробки керуючих систем.

У межах архітектури MVC компонент Model інтерпретується як керуюча Е-мережа, яка визначає структуру станів, переходів і умов їх активації [17, 18, 75].

Модель системи керування представлена формулою (3.1), де на відміну від класичних моделей, де функція Φ задається окремо від структури, у даному підході вона визначається самою мережею:

$$\Phi \equiv Execution(CEN). \quad (3.18)$$

Це означає, що виконання моделі є безпосереднім результатом її структури.

Алгоритм виконання можна подати як:

1. Формування множини активних переходів:

$$T_{\text{active}}(t) = \{t_j \mid t_j \text{ дозволений}\}. \quad (3.19)$$

2. Вибір переходів для виконання:

$$T_{\text{fire}}(t) \subseteq T_{\text{active}}(t). \quad (3.20)$$

3. Оновлення стану:

$$S(t + 1) = \text{Fire}(S(t), T_{\text{fire}}(t)). \quad (3.21)$$

Цей процес може виконуватися циклічно або подієво.

Модель реалізується як структура даних:

class Model:

```
def __init__(self):
    self.places = []
    self.transitions = []
    self.state = []
```

А виконання:

```
def step(model, events):
    enabled = get_enabled_transitions(model, events)
    fire(enabled, model)
```

Керуючі Е-мережі дозволяють реалізувати паралельне виконання:

$$\exists t_i, t_j \in T_{\text{active}}, \quad i \neq j. \quad (3.22)$$

Це означає, що модель може:

- обробляти кілька подій одночасно;
- виконувати незалежні дії;
- масштабуватися.

Модель реагує лише на події $E(t)$, що забезпечує:

- ефективність;
- зменшення навантаження;
- адаптивність [31, 36, 37].

3.2.2 Контролер як механізм виконання

У межах запропонованого підходу до побудови системи керування агентом рою БПЛА принципово важливою є інтерпретація моделі не як допоміжного засобу опису алгоритму, а як безпосередньо виконуваного компонента програмної системи.

Компонент Model можна ототожнити з керуючою Е-мережею, яка визначає множину станів, переходів і умов їх активації. На відміну від класичних підходів, де модель існує окремо від програмної реалізації, у даному випадку вона виступає центральним елементом, що безпосередньо визначає поведінку агента в процесі виконання. Це означає, що логіка системи не дублюється у вигляді процедурного коду, а задається структурою мережі та правилами її функціонування [17, 18, 75].

На відміну від традиційних програмних реалізацій, де стан системи задається множиною змінних, у запропонованому підході стан повністю визначається моделлю, тобто виконується співвідношення:

$$State_{system} \equiv S(t). \quad (3.23)$$

Це означає, що відсутні дублюючі представлення стану, а вся інформація про поточну конфігурацію системи зосереджена у маркуванні мережі та пов'язаних із ним даних. Така централізація дозволяє уникнути розсинхронізації між різними компонентами системи та спрощує її аналіз.

Виконання моделі може здійснюватися як у циклічному режимі, так і в подієво-орієнтованому. У другому випадку зміна стану відбувається лише за наявності релевантних подій, що дозволяє значно зменшити обчислювальне навантаження та підвищити ефективність системи. Подієва природа виконання узгоджується із сучасними підходами до керування в реальному часі та забезпечує адаптивність системи до змін середовища [31, 36, 37].

Особливу роль у виконуваний моделі відіграє можливість паралельної активації переходів, що впливає з мережевої структури. Це дозволяє одночасно обробляти кілька незалежних процесів, таких як навігація, обробка сенсорних даних, комунікація та контроль безпеки. У формальному вигляді це означає, що в кожен момент часу може існувати кілька активних переходів, які впливають на різні частини стану системи.

Під час програмної реалізації модель представлялась у вигляді структур даних з описом позиції, переходів та зв'язків між ними, а також механізмів виконання, які забезпечують перевірку умов і зміну стану. Це забезпечує відповідність між формальною моделлю та програмною реалізацією, що є важливою передумовою для верифікації та подальшого розвитку системи.

Представлення моделі у вигляді виконуваної керуючої Е-мережі дозволяє розглядати її як активний елемент системи керування, який визначає поведінку агента БПЛА у реальному часі [17, 75].

3.2.3 Подання як засіб спостереження

У межах архітектури MVC компонент View виконує функцію відображення стану системи та забезпечує взаємодію з зовнішнім середовищем без втручання у внутрішню логіку моделі. На відміну від моделі, яка визначає поведінку системи, і контролера, який реалізує механізм її виконання, подання виступає інтерфейсним рівнем, що дозволяє здійснювати спостереження, аналіз і передавання інформації про стан агента БПЛА.

У контексті запропонованого підходу подання базується на поточному стані моделі ($S(t)$). Подання реалізує відображення стану системи у форму, придатну для спостереження або передачі іншим компонентам. Формально це можна подати як функцію:

$$VIEW(t) = \Psi(S(t)), \quad (3.24)$$

де Ψ – оператор перетворення внутрішнього стану моделі у представлення, доступне для зовнішнього використання. Важливо, що ця функція не змінює стан системи, а лише відображає його, забезпечуючи розділення між логікою виконання та інтерфейсом взаємодії.

У задачах керування БПЛА подання може мати різні форми, зокрема у вигляді телеметричних даних, графічного інтерфейсу, логів або повідомлень для інших агентів. Наприклад, значення координат, швидкості, орієнтації та режиму роботи можуть бути перетворені у структуру, що передається через канал зв'язку або відображається у середовищі моделювання.

Особливістю використання подання в поєднанні з керуючою Е-мережею є можливість візуалізації стану системи у вигляді маркованого графа, де активні позиції відображають поточні підпроцеси, а розподіл маркерів характеризує конфігурацію поведінки агента.

Подання забезпечує можливість спостереження за динамікою системи у часі. Послідовність станів $S(t)$, що виникають у процесі виконання, може бути використана для формування журналів подій або побудови часових діаграм, які відображають зміну активних позицій і значень змінних. Це створює основу для аналізу ефективності алгоритмів, оцінки затримок і виявлення аномалій у поведінці системи.

У ройових системах БПЛА подання також виконує функцію обміну інформацією між агентами. Частина стану моделі може бути трансформована у повідомлення, що передаються через комунікаційну підсистему. Формально це можна описати як:

$$msg(t) = \Omega(S(t)), \quad (3.25)$$

де Ω – функція формування повідомлень. Отримані повідомлення, у свою чергу, можуть впливати на стан інших агентів, створюючи розподілену систему взаємодії [8, 21].

Важливо підкреслити, що подання не впливає на логіку роботи моделі, а лише забезпечує доступ до її стану, що виступає важливим елементом системи керування, який забезпечує спостереження за станом агента БПЛА, передачу даних і взаємодію із зовнішнім середовищем. Використання подання у поєднанні з виконуваною моделлю та контролером дозволяє сформувати цілісну архітектуру системи, у якій кожен компонент виконує чітко визначену функцію, що сприяє підвищенню надійності, масштабованості та зручності аналізу роботи системи [17, 75].

3.3 Метод подієвої інтеграції з зовнішнім середовищем через слухачів подій

Ключовим аспектом функціонування системи керування агентом рою БПЛА є спосіб інтеграції з зовнішнім середовищем, яке генерує події, що визначають подальшу еволюцію стану моделі. У задачах керування БПЛА такими подіями можуть виступати сенсорні вимірювання, зміни навігаційних параметрів, повідомлення від інших агентів, а також сигнали від системи керування або користувача.

Традиційні підходи до інтеграції із середовищем базуються на циклічному опитуванні джерел даних, що призводить до надлишкового навантаження на обчислювальні ресурси та затримок у реакції системи, а подієво-орієнтований підхід передбачає активацію обчислень лише у відповідь на виникнення релевантних подій, що дозволяє підвищити ефективність та забезпечити своєчасну реакцію системи на зміни умов функціонування [31, 35, 36, 37].

У запропонованій архітектурі подієва інтеграція реалізується через механізм слухачів подій (listeners), які забезпечують перетворення сигналів зовнішнього середовища у внутрішні події моделі. Слухачі виступають проміжною ланкою між

підсистемами БПЛА та виконуваною Е-мережею, здійснюючи обробку вхідних даних, їх інтерпретацію та передачу у вигляді структурованих подій, придатних для використання в умовах переходів.

3.3.1 Типи зовнішніх подій

У рамках подієво-орієнтованого підходу до побудови вбудованих моделей керування ключову роль відіграє формалізація зовнішніх подій, що надходять до системи та визначають її подальшу поведінку. У запропонованій архітектурі такі події виступають основними тригерами зміни стану керуючої Е-мережі, оскільки саме вони ініціюють перевірку умов активації переходів і виконання відповідних дій.

Загалом зовнішню подію ($E(t)$) доцільно розглядати як елемент інформаційного потоку, що надходить із середовища функціонування агента БПЛА та відображає зміну його стану або параметрів оточення. У межах даної роботи прояв групового інтелекту доцільно оцінювати через характеристики узгодженості дій агентів, своєчасності реакції на події та здатності системи до адаптивного перерозподілу ролей у процесі виконання місії.

Узагальнено подію можна представити у вигляді кортежу:

$$e_i = \langle type_i, source_i, data_i, t_i \rangle, \quad (3.26)$$

де $type_i$ – клас події,

$source_i$ – джерело події,

$data_i$ – інформаційне наповнення,

t_i – момент виникнення.

У контексті вбудованих моделей керування на основі керуючих Е-мереж обробка таких подій здійснюється за допомогою слухачів (listeners), які виконують функцію перетворення зовнішніх сигналів у внутрішні зміни стану моделі.

З урахуванням особливостей функціонування агента рою БПЛА та аналізу реалізованих *embedded models* доцільно виділити кілька основних класів зовнішніх подій, які відрізняються за джерелом виникнення та характером впливу на систему.

Перший клас становлять сенсорні події, що формуються на основі даних, отриманих від вимірювальних підсистем БПЛА. До таких подій належать зміни координат, швидкості, орієнтації, відстані до об'єктів або результати обробки зображень. Вони є найбільш частими та визначають поточний стан середовища і самого агента. Формально сенсорну подію можна подати як:

$$e_i^{\text{sens}} = \langle \text{sensor}, \text{data}_{\text{sens}}, t \rangle, \quad (3.27)$$

де $\text{data}_{\text{sens}}$ містить значення відповідних фізичних параметрів. Саме ці події використовуються в умовах переходів і визначають динаміку руху та поведінки агента [53].

Другий клас становлять події, пов'язані зі станом енергетичної підсистеми. До них належать сигнали про зниження рівня заряду акумулятора, перевищення допустимого навантаження або критичні режими роботи. Такі події мають пріоритетний характер, оскільки можуть ініціювати зміну режиму роботи агента або перехід до аварійного сценарію. У формальному вигляді їх можна представити як:

$$e^{\text{energy}} = \langle \text{battery}, \text{level}, t \rangle, \quad (3.28)$$

де level визначає поточний стан енергетичного ресурсу. Обробка таких подій дозволяє реалізувати механізми енергетично-орієнтованого керування.

Третій клас утворюють події комунікаційної підсистеми, що відображають обмін інформацією між агентами або з наземною станцією. До них належать повідомлення про стан інших БПЛА, координати цілей, сигнали синхронізації або підтвердження виконання дій. Такі події забезпечують кооперативну поведінку системи та узгодження дій у межах рою. Формально їх можна подати як:

$$e^{comm} = \langle id, msg, t \rangle, \quad (3.29)$$

де msg містить передану інформацію. Інтеграція таких подій у модель забезпечує реалізацію розподіленого керування [8, 21].

Окрему групу становлять події, пов'язані зі станом каналів зв'язку, зокрема втрата або відновлення з'єднання, зміна якості сигналу або перевантаження мережі. Такі події не несуть безпосередньої інформації про середовище, але впливають на доступність даних і можливість координації між агентами. Їх врахування дозволяє адаптувати поведінку системи до умов функціонування комунікаційної інфраструктури.

Ще один важливий клас складають тригерні події, що формуються на основі внутрішніх умов моделі. Вони виникають у результаті виконання певних логічних або часових умов, наприклад досягнення цільової точки, завершення маневру або перевищення допустимого часу перебування у стані. Формально такі події можуть бути визначені як:

$$e^{trigger} = g(V(t), \theta(t)), \quad (3.30)$$

де g – функція, що визначає умову виникнення події. Цей клас подій забезпечує зв'язок між динамікою моделі та її внутрішніми процесами.

Важливою характеристикою зовнішніх подій є їх асинхронність, що означає відсутність фіксованого порядку та моменту виникнення. Це вимагає використання механізмів обробки, здатних реагувати на події у довільний момент часу, що і забезпечується використанням слухачів подій, що узгоджується з концепцією подієво-орієнтованого керування, у якій обчислення виконуються лише за необхідності, що дозволяє зменшити навантаження на обчислювальні ресурси та підвищити ефективність системи [31, 35, 36, 37].

3.3.2 Удосконалення механізму обробки подій

У подієво-орієнтованій архітектурі керування агентом рою БПЛА ключовим елементом є механізм обробки подій, який забезпечує трансформацію зовнішніх сигналів у зміни стану виконуваної моделі. На відміну від традиційних підходів, де події накопичуються і обробляються у циклі опитування, у запропонованій системі використовується реактивна модель, у якій обробка ініціюється безпосередньо в момент виникнення події.

Основою цього механізму є використання слухачів подій, які виконують функцію зв'язку між середовищем і керуючою Е-мережею. Кожен слухач отримує зовнішню подію e_i , інтерпретує її відповідно до типу та контексту і виконує оновлення внутрішнього стану моделі.

Залежно від способу інтеграції подій у модель у вбудованих системах керування доцільно розрізняти два основні підходи: реактивну обробку подій та відкладену обробку через прапори стану. У першому випадку, який є базовим для запропонованої архітектури, слухач події безпосередньо впливає на стан моделі, ініціюючи зміну маркування або оновлення змінних. У другому випадку подія лише фіксується у вигляді певного прапора, а її обробка виконується пізніше у межах основного циклу системи.

Реактивний підхід передбачає негайне внесення змін до моделі у відповідь на подію, що може бути реалізовано через безпосередню вставку маркера у відповідну позицію мережі. Окрім зміни маркування, слухач події може оновлювати глобальні змінні або дані маркерів, що дозволяє інтегрувати нову інформацію у стан моделі без зміни її структури. Це особливо важливо для сенсорних подій, де значення параметрів безпосередньо впливають на умови активації переходів.

Після обробки події відбувається перевірка умов активації переходів. Якщо для деякого переходу t_j виконуються структурні та логічні умови, він може бути активований, а це передбачає безпосередню ініціацію подією процесу виконання моделі, що відповідає реактивній природі системи.

Підхід із відкладеною обробкою подій базується на використанні прапорів стану, які сигналізують про наявність певної події, але не викликають негайних змін у моделі, тоді слухач виконує операцію, а обробка події здійснюється у межах основного циклу контролера.

Порівняння цих підходів показує, що реактивні слухачі забезпечують значно менший час реакції, оскільки виключають необхідність періодичного опитування стану. Це особливо важливо для систем реального часу, до яких належать ройові БПЛА, де затримка навіть у декілька мілісекунд може впливати на стабільність і безпеку функціонування [31, 71].

Важливим аспектом механізму обробки подій є також узгодження змін стану у випадку одночасного надходження кількох подій. Оскільки події можуть виникати асинхронно, необхідно забезпечити коректність оновлення маркування та змінних. Це досягається шляхом послідовної або атомарної обробки подій, що гарантує цілісність стану, та дозволяє уникнути конфліктів і некоректних переходів.

У контексті реалізації агента БПЛА механізм обробки подій забезпечує безперервний зв'язок між середовищем і моделлю. Сенсорні дані, повідомлення від інших агентів та внутрішні тригери трансформуються у зміни стану мережі, що призводить до активації переходів і формування керуючих дій.

3.3.3 Впровадження відмови від циклічного опитування

У рамках запропонованої архітектури, заснованої на вбудованих моделях керування та керуючих Е-мережах, реалізується принципова відмова від циклічного опитування на користь подієво-орієнтованого виконання [17, 75].

У подієво-орієнтованій моделі виконання визначається наявністю подій, що означає відсутність змін стану за відсутності нових подій:

$$S(t + 1) = \begin{cases} \Phi(S(t), E(t)), & E(t) \neq \emptyset, \\ S(t), & E(t) = \emptyset, \end{cases} \quad (3.31)$$

Концепція ланцюгового виконання полягає в тому, що кожна подія, оброблена слухачем, може ініціювати зміну стану моделі, яка, у свою чергу, активує один або кілька переходів, що генерують нові події або змінюють глобальні змінні. Це формує послідовність взаємопов'язаних змін стану:

$$e_i \rightarrow S(t^+) \rightarrow T_{\text{fire}} \rightarrow S(t^{++}), \quad (3.32)$$

де кожен етап залежить від попереднього і система функціонує як реактивний ланцюг, у якому обчислення поширюються через структуру моделі.

Експериментальні дослідження, представлені у роботі [17], демонструють, що відмова від циклічного опитування дозволяє суттєво зменшити обчислювальне навантаження на систему. Зокрема, при використанні циклічного підходу значна частина обчислень виконується без реальної необхідності, тоді як у подієвій моделі обчислення прямо пропорційні кількості подій:

$$Load_{\text{event}} \sim |E(t)|, \quad Load_{\text{poll}} \sim \text{const.} \quad (3.33)$$

Це означає, що при низькій інтенсивності подій подієва модель забезпечує значно менше навантаження, що є критично важливим для вбудованих систем з обмеженими ресурсами.

Ще одним важливим аспектом є затримка реакції системи. У циклічному підході максимальна затримка визначається періодом опитування, що означає, що навіть при миттєвому виникненні події її обробка може бути відкладена до наступного циклу. У подієво-орієнтованому підході затримка визначається лише часом обробки події, що забезпечує значно швидшу реакцію системи.

Для наочного представлення запропонованого підходу до організації стану вбудованої моделі керування доцільно розглянути структурну реалізацію керуючої Е-мережі, яка відображає як логіку алгоритму, так і поточний стан системи у вигляді маркованого графа (рис. 3.4).

Маркований граф представляє керуючу Е-мережу у вигляді орієнтованого графа, що складається з множини позицій P , переходів T та дуг F , які визначають структуру зв'язків між ними згідно формулі (3.1).

Робота агента рою в межах запропонованої контрольної Е-мережі реалізується як безперервний процес моніторингу середовища, аналізу подій та колективного прийняття рішень. Алгоритм функціонування агента можна подати у вигляді послідовності етапів.

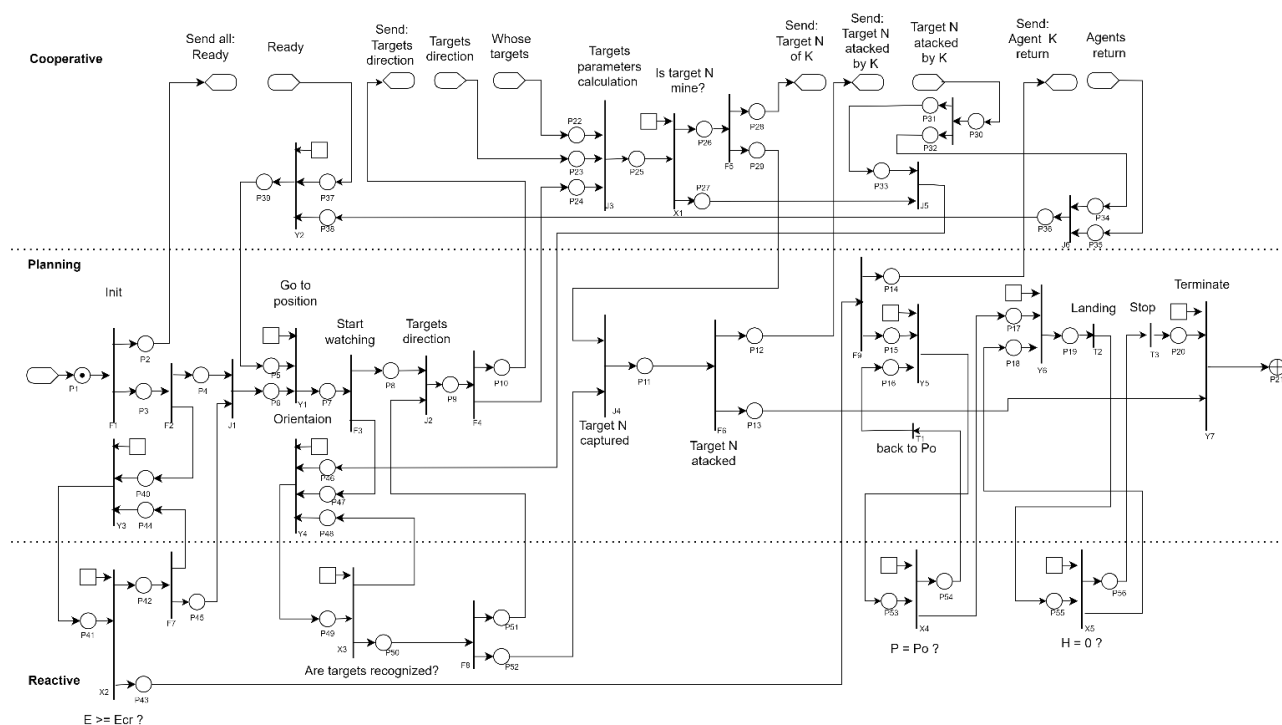


Рисунок 3.4 – Приклад розробленої керуючої Е-мережі програмного керування БПЛА

На початковому етапі агент переходить у режим виконання місії після завершення процедур ініціалізації, орієнтації та виходу у визначену зону спостереження. Після цього активується основний цикл моніторингу середовища, у межах якого агент безперервно виконує аналіз сенсорної інформації, оновлення параметрів навколишнього середовища та перевірку наявності потенційних цілей.

У випадку виявлення та розпізнавання потенційної цілі агент формує локальне повідомлення про параметри цілі та передає інформацію іншим агентам рою.

Передача включає напрямок руху цілі, її координати, а також службові параметри, необхідні для колективного аналізу ситуації. Після передачі інформації агент переходить у стан очікування зовнішніх повідомлень від інших учасників рою, однак при цьому не припиняє виконання локального моніторингу середовища.

Логіку взаємодії агентів після виявлення цілі наведено у Додатку Д.

Представлена модель дозволяє одночасно відобразити:

- множину можливих станів агента;
- умови переходів між ними;
- паралельні процеси;
- поточний стан системи у вигляді розподілу маркерів.

У контексті керування БПЛА така структура мережі дозволяє реалізувати:

- багаторівневу логіку поведінки;
- паралельну обробку задач;
- динамічну реакцію на зміну середовища;
- кооперативну взаємодію в межах рою.

Кожна активна позиція відповідає певному функціональному компоненту системи, а переходи – зміні режимів або реакції на події.

У контексті ройових БПЛА це має принципове значення, оскільки затримки у прийнятті рішень можуть призводити до втрати стабільності, порушення координації або виникнення аварійних ситуацій.

3.4 Реалізація вбудованої моделі керування у виконавчому середовищі

Реалізація вбудованої моделі керування на основі керуючих Е-мереж потребує наявності виконавчого середовища, яке забезпечує інтерпретацію моделі, обробку подій, планування виконання та узгодження паралельних процесів. У попередніх підрозділах показано, що модель визначає логіку системи, а контролер реалізує механізм її виконання. Разом з тим, саме виконавче середовище відповідає за організацію цього процесу в умовах обмежених ресурсів і асинхронного надходження подій.

Виконавче середовище можна розглядати як проміжний рівень між формальною моделлю та апаратною платформою та реалізує механізми паралельного виконання, керування чергами подій і задач, а також синхронізації між окремими гілками виконання.

3.4.1 Паралельне виконання гілок і черги

Однією з ключових властивостей керуючих Е-мереж є можливість паралельного виконання незалежних гілок, що безпосередньо відображає розподілений характер задач керування ройовими БПЛА. На рівні формальної моделі це проявляється у можливості одночасної активації кількох переходів, що не конфліктують між собою.

У загальному випадку множина активних переходів може містити кілька елементів, за умови, що відповідні переходи не мають спільних вхідних позицій і не призводять до конфліктів у зміні стану:

$$T_{\text{active}}(t) = \{t_i, t_j, \dots\}, \quad i \neq j. \quad (3.34)$$

У виконавчому середовищі така можливість реалізується через розподіл виконання між окремими потоками або асинхронними задачами. Кожна гілка моделі може бути представлена як незалежний процес, що обробляє власний піднабір подій і змін стану. При цьому важливо, що паралельність не порушує цілісності моделі, оскільки всі зміни узгоджуються через спільний стан $S(t)$.

Паралельне виконання дозволяє одночасно обробляти різні аспекти функціонування агента, зокрема навігацію, аналіз сенсорних даних, комунікацію та контроль безпеки. На відміну від паралельного виконання у подієво-орієнтованій архітектурі виконавче середовище повинно забезпечувати впорядковану обробку подій і задач, для чого використовується механізм черг.

Нехай $Q(t)$ – черга подій і задач у момент часу t , тоді її можна представити як впорядковану множину:

$$Q(t) = \langle e_1, e_2, \dots, e_n \rangle. \quad (3.35)$$

Кожна подія після надходження додається до черги, а обробка виконується відповідно до обраної політики планування. У найпростішому випадку використовується принцип першого входу – першого виходу, однак у складніших системах можуть застосовуватися пріоритетні черги, де порядок обробки визначається важливістю подій.

Кожен агент використовує однакову модель керування, реалізовану у вигляді керуючої Е-мережі, однак у конкретний момент часу може перебувати на різних етапах її виконання і передбачає не паралельне виконання гілок у межах однієї моделі, а незалежне виконання однакових моделей різними агентами.

Паралельність на рівні окремого агента в рамках керуючої Е-мережі реалізується інакше – через одночасне виконання кількох функціональних процесів, таких як:

- оновлення інформації про стан акумулятора;
- підтримка просторової стабілізації (висоти, положення);
- обробка сенсорних даних (зокрема, розпізнавання об'єктів).

Ці процеси відповідають різним гілкам виконання Е-мережі та можуть виконуватися одночасно в межах одного агента, що забезпечує його реактивну та адаптивну поведінку.

Зображені області спостереження агентів (рис. 3.5) відображають зони, в яких формується подієва інформація типу *sensor events*.

При потраплянні об'єкта або іншого агента в межі цієї зони генерується подія, яка передається до моделі керування. У контексті Е-мережі така подія інтерпретується як активація відповідної позиції або ініціація переходу, що запускає подальше поширення маркерів.

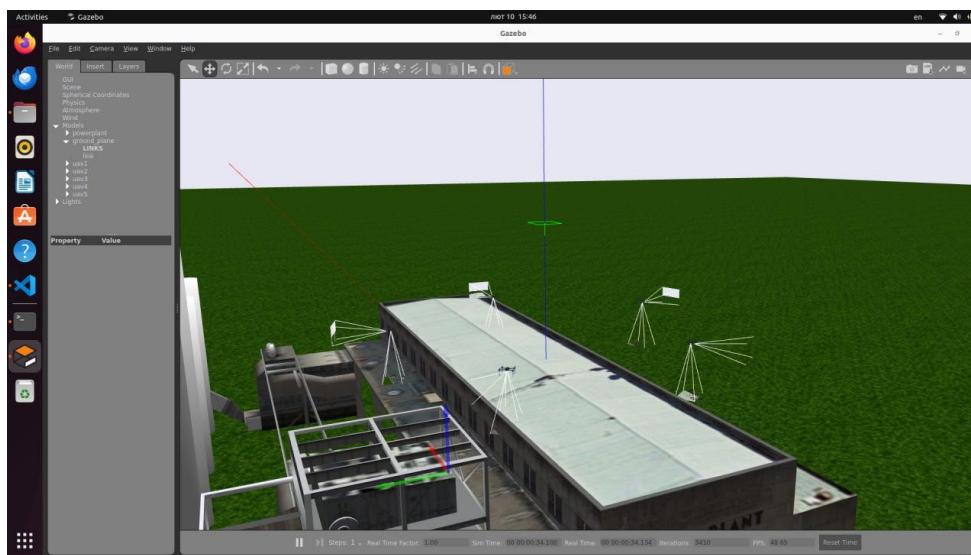


Рисунок 3.5 – Візуалізація сенсорних зон агентів як джерел подій у системі керування

3.4.2 Синхронізація та об'єднання гілок

Наявність паралельних гілок виконання у керуючій Е-мережі зумовлює необхідність їх синхронізації та об'єднання для забезпечення узгодженості стану системи. Перехід t_j може бути активований лише після заповнення маркерами необхідних позицій, тоді у формальному вигляді умова синхронізації може бути подана наступним чином:

$$\forall p_i \in P^{\text{red}}(t_j): M(p_i, t) = 1. \quad (3.36)$$

У виконавчому середовищі це відповідає механізмам синхронізації потоків або задач, коли виконання певної дії відкладається до завершення кількох паралельних процесів.

Крім структурної синхронізації, важливу роль відіграє узгодження доступу до спільних змінних. Оскільки паралельні гілки можуть одночасно змінювати глобальні змінні $V(t)$, необхідно забезпечити атомарність операцій оновлення $V(t + 1) = f(V(t))$, що виконується у контрольованому середовищі без конфліктів.

На рис. 3.6 наведено приклад ситуації, у якій реалізується механізм синхронізації даних між агентами рою в процесі прийняття колективного рішення.



Рисунок 3.6 – Візуалізація умов синхронізації переходу з множинними вхідними позиціями

У даному сценарії один із безпілотних агентів здійснює первинне виявлення та розпізнавання цілі на основі власних сенсорних даних. Однак отриманої інформації недостатньо для точного визначення координат об'єкта та формування оптимальної стратегії реагування, що зумовлює необхідність залучення додаткових даних від інших агентів.

Після розпізнавання цілі агент переходить у стан очікування зовнішньої події, що відповідає надходженню уточненої інформації від іншого учасника рою. У термінах керуючої Е-мережі це відповідає ситуації, коли подальше виконання переходів блокується до моменту активації відповідного подієвого тригера, що сигналізує про наявність необхідних даних. Фрагмент логіки виконання очікування сигналу наведено у Додатку Б, а Ключові фрагменти програмного коду системи Е-мережі у Додатку В.

Після надходження зовнішнього сигналу виконується синхронізація інформації, що дозволяє сформувати узгоджене уявлення про положення цілі. На основі об'єднаних даних здійснюється прийняття керуючого рішення щодо розподілу

ролей між агентами, зокрема визначення агента, який буде здійснювати перехоплення, та агента, що виконуватиме функцію підтримки або страхування.

3.5 Висновки до розділу 3

У третьому розділі дисертаційного дослідження було розроблено та обґрунтовано метод вбудованих моделей керування агентом рою безпілотних літальних апаратів, що базується на керуючих Е-мережах. Також визначено особливості програмної реалізації, інтеграції в середовище та функціонування в умовах реального часу.

Модель MVC отримала подальший розвиток завдяки реалізації подієво-орієнтованого керування в мультиагентних системах, де керуюча Е-мережа відповідає компоненту моделі, об'єкти керування представлені програмними агентами, механізм обробки подій і активації переходів реалізується контролером, а подання забезпечує доступ до стану системи та взаємодію із зовнішнім середовищем.

У процесі дослідження було встановлено, що представлення стану системи у вигляді маркування керуючої Е-мережі з асоційованими даними дозволяє реалізувати розподілену структуру стану, в якій інформація зберігається безпосередньо в елементах моделі.

Особливу увагу приділено удосконаленню механізму подієвої інтеграції, який базується на використанні слухачів подій як інтерфейсу між середовищем і моделлю. Показано, що застосування реактивних слухачів забезпечує безпосереднє перетворення зовнішніх сигналів у зміни стану моделі, що дозволяє мінімізувати затримки та підвищити швидкість реакції системи.

У розділі обґрунтовано доцільність відмови від циклічного опитування як основного механізму взаємодії із середовищем. На основі експериментальних даних встановлено, що подієво-орієнтований підхід дозволяє скоротити затримку реакції системи та забезпечити більш стабільну поведінку у мінливому середовищі.

Матеріали розділу опубліковано у роботах автора: [16, 17, 18, 20].

РОЗДІЛ 4

ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНИХ МЕТОДІВ КЕРУВАННЯ РОЄМ ДРОНІВ НА ОСНОВІ ГРУПОВОГО ІНТЕЛЕКТУ

Розроблений у попередньому розділі метод вбудованих моделей керування на основі керуючих Е-мереж потребує експериментального підтвердження своєї ефективності в умовах, наближених до реального функціонування рою безпілотних літальних апаратів. Незважаючи на формальну коректність моделі та її відповідність сучасним підходам до подієво-орієнтованого керування, ключовим питанням залишається оцінка її практичних характеристик, зокрема швидкодії, ефективності використання обчислювальних ресурсів та здатності до масштабування.

У зв'язку з тим, що запропонований у дисертаційній роботі підхід, заснований на подієво-орієнтованій обробці та використанні керуючих Е-мереж як виконуваної моделі, виникає необхідність у комплексному експериментальному дослідженні, спрямованому на порівняння традиційного підходу циклічного опитування та подієво-орієнтованого підходу, реалізованого на основі керуючих Е-мереж.

4.1 Методика експериментів для перевірки ефективності запропонованих методів керування роєм дронів

Експериментальна перевірка запропонованих методів потребує чіткої формалізації цілей дослідження, визначення об'єкта та предмета експериментів, а також розробки методики їх проведення, що забезпечує відтворюваність та достовірність отриманих результатів [100, 101, 102]. В умовах дослідження систем керування роєм БПЛА особливої важливості набуває забезпечення порівнюваності результатів між різними підходами [103], що вимагає уніфікації сценаріїв, параметрів середовища та умов виконання.

Основною особливістю дослідження є необхідність оцінювання не лише функціональної коректності системи керування, але й її динамічних характеристик,

пов'язаних із часом реакції на події, рівнем використання обчислювальних ресурсів та стабільністю роботи в умовах змінного навантаження [100, 104, 105]. Це обумовлює використання комплексного підходу до експериментування, який поєднує моделювання поведінки системи, інструментальне вимірювання показників та статистичну обробку результатів.

Запропонована методика експериментів базується на використанні розробленого програмного середовища моделювання, яке реалізує виконувачу модель керуючої Е-мережі та дозволяє здійснювати порівняльний аналіз різних підходів до організації керування, що забезпечує можливість проведення серії контрольованих експериментів з фіксованими параметрами та подальшою обробкою отриманих даних [Помилка! Джерело посилання не знайдено.].

4.1.1 Мета, об'єкт і предмет експериментальних досліджень

Метою експериментальних досліджень є кількісна оцінка ефективності запропонованих методів удосконалення керування агентами рою безпілотних літальних апаратів, що базується на використанні вбудованих моделей на основі керуючих Е-мереж, а також підтвердження набутих переваг порівняно з традиційним підходом циклічного опитування.

Досягнення поставленої мети передбачає вирішення комплексу взаємопов'язаних завдань, спрямованих на дослідження динамічних характеристик системи керування. Зокрема, необхідно визначити залежності часу реакції системи на події від параметрів навантаження, оцінити ефективність використання обчислювальних ресурсів, а також проаналізувати вплив обраного підходу на масштабованість системи при зростанні кількості оброблюваних подій [19].

Особливу увагу в експериментальних дослідженнях приділено порівняльному аналізу двох принципово різних механізмів організації керування – циклічного опитування та подієво-орієнтованого підходу. При цьому оцінювання проводиться за сукупністю часових, ресурсних та інтегральних показників ефективності, що дозволяє отримати комплексну характеристику функціонування системи.

Об'єктом експериментальних досліджень є процес функціонування системи керування агентом рою безпілотних літальних апаратів у середовищі з асинхронним надходженням подій, що включає сенсорні сигнали, комунікаційні повідомлення та внутрішні тригери.

У рамках дослідження розглядається програмна реалізація системи керування, що базується на використанні виконуваної керуючої Е-мережі як моделі поведінки агента, а також альтернативна реалізація, що використовує механізм циклічного опитування стану системи, а предметом експериментальних досліджень є характеристики ефективності функціонування системи керування, які визначаються обраним підходом до організації обчислень. До таких характеристик належать:

- час реакції системи на події [61];
- рівень використання обчислювальних ресурсів (процесорного часу, пам'яті);
- інтенсивність виконання операцій та кількість перевірок умов;
- залежність продуктивності від параметрів навантаження;
- масштабованість системи при зростанні кількості подій.

4.1.2 Апаратна та програмна платформа експериментів

Експериментальні дослідження проводилися на апаратно-програмній платформі, що забезпечує умови, наближені до реального функціонування вбудованих систем керування безпілотними літальними апаратами. У конструкції квадрокоптера мультиагентної системи обрано польотний контролер Pixhawk 6C (рис. 4.1), а як базову обчислювальну платформу використано одноплатний комп'ютер сімейства NVIDIA Jetson Nano.

Структура розробленої системи керування дроном у складі мультиагентної системи представлена у Додатку Г, на базі якої розроблено діючі прототипи дронів (рис. 4.2), а результати дослідження увійшли у звіт до «Мультиагентна система захисту об'єктів інфраструктури на основі рою мультикоптерних дронів» (номер державної реєстрації: 0123U101819 (2023-2025)).

Дрони побудовано на базі Jetson Nano, яка є енергоефективною обчислювальною платформою, що поєднує достатній рівень продуктивності з

обмеженими ресурсами, характерними для вбудованих систем. Це дозволяє використовувати його як репрезентативне середовище для оцінювання ефективності алгоритмів керування агентами рою БПЛА.

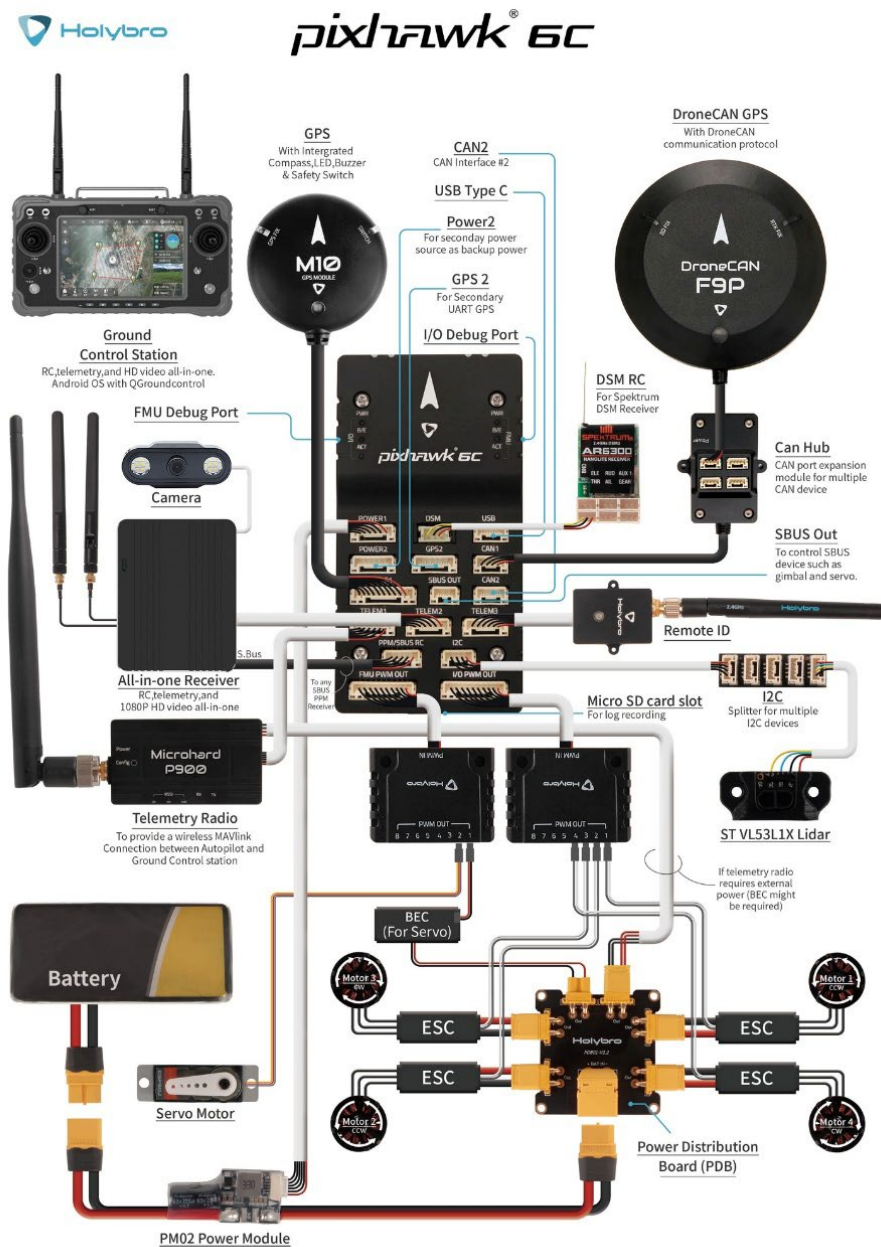


Рисунок 4.1 – Базова схема квадрокоптера на Pixhawk 6C [107]



Рисунок 4.2 – Розроблені в межах дослідження діючі прототипи дронів

Апаратна конфігурація експериментальної платформи включає:

- центральний процесор ARM Cortex-A57 з тактовою частотою до 1.43 ГГц;
- графічний процесор NVIDIA Maxwell з підтримкою паралельних обчислень;
- оперативну пам'ять обсягом 4 ГБ;
- енергоефективний режим роботи, характерний для мобільних і вбудованих систем.

Використання саме такої платформи обумовлено необхідністю оцінки ефекту від впровадження запропонованих методів в умовах обмежених обчислювальних ресурсів, що є типовими для бортових систем БПЛА.

Програмна складова експериментальної системи реалізована у вигляді спеціалізованого програмного середовища, закодованого засобами об'єктно-орієнтованого програмування, яке включає:

- реалізацію виконуваної моделі керуючої Е-мережі;
- модуль обробки подій на основі механізму слухачів;
- реалізацію альтернативного підходу циклічного опитування;
- підсистему збору статистики та профілювання виконання;
- модулі візуалізації та аналізу результатів.

У межах проведеного дослідження розроблено програмну реалізацію моделі керування на основі керуючих Е-мереж, яка забезпечує виконання формалізованих

алгоритмів керування агентом рою БПЛА в умовах подієво-орієнтованого середовища. Реалізація виконана у вигляді модульної бібліотеки мовою об'єктно-орієнтованою мовою програмування Python, що дозволяє використовувати її як основу для проведення експериментів, а також для подальшої інтеграції у вбудовані системи керування.

Загальна структура програмного рішення організована у вигляді набору взаємопов'язаних компонентів, що реалізують основні елементи Е-мережі: позиції, переходи та механізм виконання (рис. 4.3).

Центральним елементом реалізації є клас `E_Network`, який виконує функції керуючого ядра моделі. Він відповідає за збереження структури мережі, керування позиціями та переходами, а також за виконання алгоритму просування маркерів. У складі цього класу реалізовано механізми ініціалізації моделі, запуску процесу виконання (`run()`), а також обробки подій через методи типу `set_token_to_position()`.

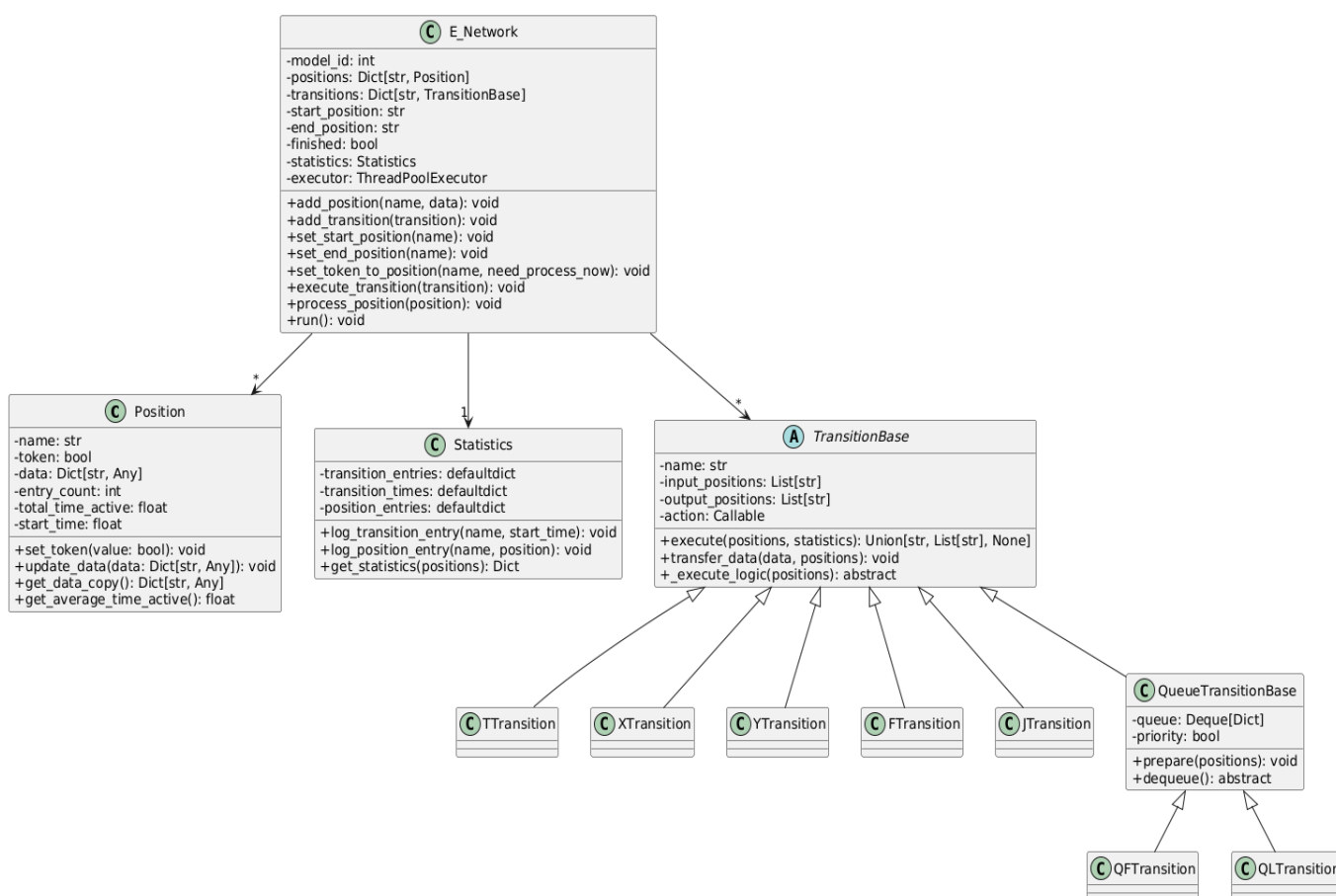


Рисунок 4.3 – Діаграма класів програмної реалізації системи

Передбачено два режими просування маркера:

1. Синхронний ланцюг – якщо перехід має всі необхідні маркери, він спрацьовує одразу, і обробка продовжується по ланцюгу в рамках одного виклику ``process_position()``.

2. Очікування зовнішньої події – якщо перехід потребує маркер, якого ще немає (наприклад, ``JTransition`` чекає всі входи), тоді обробка зупиняється. Коли зовнішній код викличе ``set_token_to_position(..., need_process_now=True)``, ланцюг продовжиться з нового місця.

Метод ``run()`` поєднує обидва режими: спочатку запускає синхронний ланцюг від стартової позиції, а потім чекає (``while not self.finished``) поки зовнішні тригери або паралельні гілки не доведуть модель до кінцевої позиції.

Стан системи представлено через об'єкти класу `Position`, які зберігають інформацію про наявність маркерів, пов'язані дані та часові характеристики перебування маркера у позиції, що дозволяє не лише відслідковувати стан системи, але й здійснювати збір статистичних даних, необхідних для подальшого аналізу ефективності.

Для забезпечення збору та обробки статистичної інформації використовується клас `Statistics`, який реалізує функції фіксації переходів, обчислення часових характеристик та формування узагальнених показників. Це дозволяє інтегрувати процес вимірювання безпосередньо у виконання моделі без додаткових накладних витрат.

Для реалізації класу `E_Network` використано наступні поля:

- `model_id` (тип `str`) – ідентифікатор моделі, використовується у логах для розрізнення кількох паралельних моделей;
- `positions` (тип `Dict[str, Position]`) – реєстр усіх позицій мережі (ключ – ім'я позиції);
- `transitions` (тип `Dict[str, TransitionBase]`) – реєстр усіх переходів мережі (ключ – ім'я переходу);
- `start_position` (тип `str`) – ім'я стартової позиції, з якої починається виконання;
- `end_position` (тип `str`) – ім'я кінцевої позиції; коли маркер потрапляє сюди, модель завершується (`finished = True`);

- `finished` (тип `bool`) – прапорець завершення моделі;
- `statistics` (тип `Statistics`) – об'єкт збору метрик: кількість входів, час виконання переходів, час активності позицій;
- `executor` (тип `ThreadPoolExecutor`) – пул потоків (`max 10`) для паралельного виконання гілок при `FTransition`.

Абстрактний клас `TransitionBase` визначає загальний інтерфейс для всіх типів переходів, включає опис вхідних та вихідних позицій, а також механізм виконання, що реалізується через методи обробки логіки переходу, що забезпечує можливість розширення моделі шляхом додавання нових типів переходів без зміни базової архітектури.

На основі `TransitionBase` реалізовано набір спеціалізованих класів переходів, які відповідають розширеному формалізму керуючих Е-мереж. у програмній реалізації представлені базові переходи (`TTransition`), що виконують загальну логіку обробки даних і переміщення маркера між позиціями. Для реалізації умовного вибору використовуються переходи типу `XTransition` та `YTransition`, які забезпечують гілкування виконання залежно від значень вхідних даних або стану системи.

Переходи синхронізації (`JTransition`) застосовуються для об'єднання кількох гілок виконання та забезпечують активацію лише за наявності необхідних маркерів у всіх вхідних позиціях. Для організації паралельного виконання використовуються переходи розгалуження (`FTransition`), які ініціюють незалежні гілки обробки.

Окремий клас становлять переходи для роботи з чергами (`QueueTransition` та його похідні), що забезпечують буферизацію даних і підтримку асинхронної взаємодії між підпроцесами.

Така ієрархія переходів дозволяє декларативно описувати складні сценарії керування, включаючи паралельне виконання, умовні переходи, синхронізацію та обробку потоків даних (рис. 4.4).

Особливістю реалізації є використання подієво-орієнтованого механізму виконання, який базується на принципі ланцюгового просування маркерів. При надходженні маркера у певну позицію система автоматично визначає всі переходи, для яких ця позиція є вхідною, та ініціює їх перевірку.

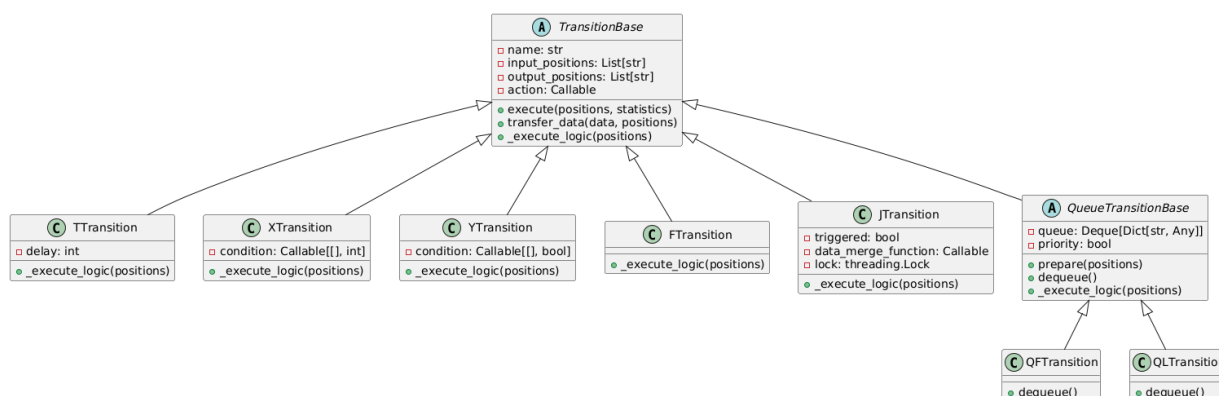


Рисунок 4.4 – Діаграма класів реалізації формалізмів керуючих Е-мереж

У випадку виконання переходу маркер переміщується у наступну позицію, що запускає наступний етап обробки і формується реактивний ланцюг виконання, який продовжується до досягнення кінцевого стану або до необхідності очікування зовнішньої події.

Алгоритм роботи системи поєднує два режими: синхронний та подієвий. У синхронному режимі виконання відбувається безперервно в межах одного виклику, поки існують активні переходи. У подієвому режимі виконання призупиняється до моменту надходження зовнішнього сигналу, після чого обробка відновлюється з відповідної позиції. Така організація забезпечує ефективне використання ресурсів та узгоджується з концепцією подієво-орієнтованого виконання.

Структура проєкту організована у вигляді окремих модулів, що розділяють логіку ядра моделі, реалізацію переходів, приклади використання та експериментальні сценарії. Зокрема, виділено модуль бібліотеки Е-мережі, модуль прикладної моделі, а також підсистему проведення експериментів і збору результатів. Така модульність забезпечує гнучкість використання та можливість повторного застосування компонентів (рис. 4.5).

Операційна система базується на Linux-дистрибутиві, оптимізованому для платформи Jetson Nano, що забезпечує стабільність виконання, підтримку багатозадачності та можливість роботи в режимі реального часу.

```

drone_convertor/
├── e_network/
│   ├── __init__.py          # Бібліотека Е-мережі (pip install)
│   ├── core.py             # Публічний API
│   └── transitions.py       # Position, Statistics, E_Network
├── cen_model_python.py     # Всі типи переходів
├── example/                # Приклад використання бібліотеки
│   ├── drone_convertor-*.whl # Готовий приклад для кінцевого користувача
│   └── main.py              # Зібраний wheel-пакет
├── tests/                  # Приклад моделі
│   └── test_model_patterns.py # Тести верифікації патернів моделей
├── experiments/            # Тести верифікації патернів моделей
│   ├── comparison_example_v3.py # Експерименти: порівняння polling vs reactive та ін.
│   ├── run_comparison_and_visualize.py
│   ├── run_comparative_analysis.py
│   ├── algorithmic_complexity_analysis.py
│   ├── test_reaction_stability.py
│   └── visualize_results.py
└── results/                # Результати експериментів (генеруються автоматично)

```

Рисунок 4.5 – Структура розробленого проєкту

Вибір зазначеної апаратно-програмної платформи дозволяє:

- оцінити ефект від впровадження запропонованих методів в умовах обмежених ресурсів;
- забезпечити відповідність умов експерименту реальним сценаріям використання;
- гарантувати можливість подальшого перенесення розроблених рішень у бортові системи БПЛА.

4.1.3 Умови повторюваності експериментів

Забезпечення повторюваності експериментальних досліджень є необхідною умовою наукової достовірності отриманих результатів. У зв'язку з цим у роботі сформовано набір умов і обмежень, що гарантують можливість відтворення експериментів незалежно від виконавця.

Усі експерименти проводилися у контрольованому програмному середовищі з фіксованими параметрами виконання. Початкові умови моделі, включаючи початкове маркування керуючої Е-мережі та значення глобальних змінних, задавалися однаковими для всіх серій експериментів.

Для забезпечення статистичної достовірності результати кожного експерименту отримувалися на основі багаторазових запусків із однаковими параметрами. Кількість повторень вибиралася таким чином, щоб мінімізувати вплив випадкових факторів та забезпечити стабільність оцінок. При цьому

використовувалося фіксоване значення генератора псевдовипадкових чисел, що дозволяло отримувати ідентичні послідовності подій при повторних запусках.

Параметри експериментів, зокрема інтенсивність надходження подій, тривалість моделювання та конфігурація системи, задавалися явно та залишалися незмінними в межах кожного сценарію. Це забезпечує коректність порівняння між різними підходами.

Для усунення впливу зовнішніх факторів під час проведення експериментів обмежено фонове навантаження на обчислювальну платформу, а всі вимірювання виконувалися у стабільному режимі роботи системи. Додатково враховувалася затримка ініціалізації системи, що дозволяло уникнути спотворення результатів на початкових етапах виконання.

Результати експериментів фіксувалися у вигляді логів, що містять часові мітки, значення параметрів та інформацію про виконання переходів. Це дозволяє не лише відтворити експерименти, але й виконати їх повторний аналіз у разі необхідності.

4.2 Показники оцінювання та спосіб обробки результатів

Оцінювання ефекту від впровадження запропонованих методів програмного керування на основі керуючих Е-мереж потребує використання системи кількісних показників, які дозволяють об'єктивно порівняти його з традиційним підходом циклічного опитування. Враховуючи, що досліджувані системи функціонують у режимі реального часу та взаємодіють із мінливим середовищем, доцільним є використання комплексного підходу до оцінювання, який охоплює часові характеристики, ресурсне навантаження та інтегральні показники ефективності.

Особливістю проведених експериментів є необхідність аналізу не лише середніх значень показників, але й їх розподілу, варіативності та статистичної значущості відмінностей між підходами. Це обумовлює використання методів статистичної обробки результатів, що дозволяють підвищити достовірність висновків та уникнути впливу випадкових факторів.

4.2.1 Часові показники

Часові характеристики є одними з ключових показників ефективності систем керування ройовими БПЛА, оскільки визначають здатність системи своєчасно реагувати на події та забезпечувати стабільну поведінку в умовах мінливого середовища [32, 33]. У рамках даного дослідження особлива увага приділяється оцінці часу реакції системи на події, затримок обробки та загальної динаміки виконання алгоритмів.

Основним часовим показником є час реакції системи на подію, який визначається як інтервал між моментом виникнення події та моментом завершення відповідної дії у моделі:

$$T_{react}^{(i)} = t_{out}^{(i)} - t_{in}^{(i)}, \quad (4.1)$$

де $t_{in}^{(i)}$ – момент надходження i -тої події;

$t_{out}^{(i)}$ – момент завершення її обробки, що супроводжується зміною стану системи або формуванням керуючого сигналу.

Для отримання узагальненої оцінки використовується середній час реакції:

$$\bar{T}_{react} = \frac{1}{N} \sum_{i=1}^N T_{react}^{(i)}, \quad (4.2)$$

де N – загальна кількість оброблених подій. Цей показник дозволяє оцінити типову швидкодію системи.

Окрім середнього значення, важливим є аналіз варіативності часу реакції, що характеризує стабільність роботи системи. Для цього використовується дисперсія:

$$\sigma_{react}^2 = \frac{1}{N} \sum_{i=1}^N \left(T_{react}^{(i)} - \bar{T}_{react} \right)^2. \quad (4.3)$$

Ці показники дозволяють оцінити рівень флуктуацій часу реакції та визначити, наскільки система є передбачуваною.

Додатково оцінюється інтенсивність обробки подій, що визначається як кількість подій, оброблених за одиницю часу:

$$\mu = N / \Delta t, \quad (4.4)$$

де Δt – тривалість експерименту. Цей показник характеризує пропускну здатність системи.

У процесі експериментів часові показники фіксуються з використанням високоточних часових міток, що дозволяє отримати детальну інформацію про динаміку обробки подій. На основі отриманих даних формуються розподіли часу реакції, які аналізуються за допомогою статистичних методів, зокрема з використанням гістограм та діаграм розмаху.

4.2.2 Показники ресурсного навантаження

Поряд із часовими характеристиками важливим аспектом оцінювання ефективності систем керування ройовими БПЛА є рівень використання обчислювальних ресурсів. У вбудованих системах, до яких належать бортові обчислювальні платформи безпілотних апаратів, ресурси є обмеженими, що обумовлює необхідність мінімізації навантаження при забезпеченні заданої функціональності.

У рамках даного дослідження аналіз ресурсного навантаження здійснюється за двома основними напрямками: використання процесорного часу та використання оперативної пам'яті. Такі показники дозволяють оцінити ефективність реалізації алгоритмів керування та визначити їх придатність для використання у системах реального часу.

Одним із ключових показників є завантаження центрального процесора, яке визначається як відношення часу, протягом якого процесор зайнятий виконанням задач, до загального часу спостереження:

$$CPU_{load} = \frac{T_{active}}{T_{total}}, \quad (4.5)$$

де T_{active} – сумарний час виконання обчислень;

T_{total} – загальна тривалість експерименту.

Для отримання узагальненої оцінки використовується середнє значення завантаження процесора:

$$\overline{CPU} = \frac{1}{N} \sum_{i=1}^N CPU_{load}^{(i)}, \quad (4.6)$$

де $CPU_{load}^{(i)}$ – значення завантаження у момент часу i .

Аналіз використання оперативної пам'яті здійснюється через показник обсягу зайнятої пам'яті у процесі виконання. На основі цих даних визначається середнє значення використання пам'яті, а також максимальне значення, що дозволяє оцінити пікові потреби системи у пам'яті.

Важливою характеристикою є також динаміка зміни використання ресурсів у часі. У подієво-орієнтованому підході навантаження має нерівномірний характер і визначається інтенсивністю надходження подій, тоді як у випадку циклічного опитування спостерігається більш рівномірне, але постійно підвищене навантаження.

У процесі експериментів показники ресурсного навантаження фіксувалися з використанням засобів моніторингу операційної системи, що дозволило отримати детальні часові залежності використання CPU та пам'яті. На основі отриманих даних побудовано графіки зміни навантаження, які відображають характер роботи системи при різних підходах до організації керування.

4.2.3 Показники ефективності виконання

Оцінювання ефективності функціонування системи керування ройовими БПЛА не може обмежуватися лише аналізом окремих часових або ресурсних характеристик. Для отримання узагальненої оцінки доцільно використовувати інтегральні показники,

які враховують взаємозв'язок між швидкістю реакції системи, обчислювальними витратами та інтенсивністю обробки подій.

У рамках даного дослідження ефективність виконання розглядається як здатність системи забезпечувати своєчасну обробку подій при мінімальних витратах обчислювальних ресурсів, що дозволяє оцінити не лише абсолютні значення показників, але й їх співвідношення, що є критично важливим для вбудованих систем.

Одним із базових інтегральних показників є пропускна здатність системи, яка визначається як кількість подій, оброблених за одиницю часу:

$$Throughput = \frac{N_{processed}}{\Delta t}, \quad (4.7)$$

де $N_{processed}$ – кількість оброблених подій за інтервал часу Δt , що характеризує здатність системи працювати в умовах інтенсивного потоку подій.

Важливим показником є також коефіцієнт обробки подій, що відображає співвідношення між кількістю оброблених та згенерованих подій:

$$K_{proc} = \frac{N_{processed}}{N_{generated}}. \quad (4.8)$$

Коефіцієнт обробки подій дозволяє оцінити, наскільки система справляється з навантаженням. Значення $K_{proc} \approx 1$ свідчить про повну обробку подій, тоді як його зменшення вказує на перевантаження системи.

Додатково вводиться показник ефективності обчислень, що характеризує кількість корисних операцій відносно загального обсягу виконаних перевірок:

$$E_{comp} = \frac{N_{useful}}{N_{total}}, \quad (4.9)$$

де N_{useful} – кількість операцій, що призвели до реальної зміни стану системи;

N_{total} – загальна кількість виконаних перевірок.

У випадку циклічного опитування значна частина перевірок є надлишковою, що знижує значення цього показника.

У процесі експериментів зазначені показники обчислюються на основі зібраної статистики та використовуються для побудови порівняльних характеристик досліджуваних підходів. Аналіз отриманих значень дозволяє встановити переваги подієво-орієнтованого підходу у контексті підвищення ефективності виконання, зменшення обчислювального навантаження та покращення масштабованості системи.

4.3 Результати порівняння реакції на події проти циклічного опитування

На основі сформованої системи показників оцінювання та визначеної методики експериментів проведено серію досліджень, спрямованих на порівняльний аналіз ефективності двох підходів до організації керування – циклічного опитування та подієво-орієнтованого підходу, реалізованого на основі керуючих Е-мереж.

Основною метою даного етапу дослідження є встановлення кількісних залежностей між параметрами функціонування системи та її ефективністю, а також виявлення характерних відмінностей у поведінці системи при використанні різних механізмів обробки подій. Особливу увагу приділено оцінці часу реакції, рівня обчислювального навантаження та здатності системи адаптуватися до зміни інтенсивності подій.

Порівняння здійснюється в умовах однакових експериментальних сценаріїв, що забезпечує коректність отриманих результатів і дозволяє виключити вплив сторонніх факторів. При цьому аналіз результатів базується не лише на середніх значеннях показників, але й на їх статистичних характеристиках, включаючи розподіли, варіативність та наявність пікових значень.

Отримані результати представлені у вигляді часових залежностей, статистичних розподілів та узагальнених показників, що дозволяє комплексно оцінити динаміку функціонування системи, що забезпечує можливість не лише порівняння ефективності підходів, але й глибшого розуміння причин виявлених відмінностей.

4.3.1 Порівняння часу реакції

Одним із головних критеріїв ефективності системи керування агентом рою БПЛА є час реакції на зовнішню подію, оскільки саме він визначає здатність системи своєчасно змінювати свій стан і формувати керуючі дії в умовах мінливого середовища. Для порівняння циклічного підходу та подієво-орієнтованого підходу проведено серію експериментів, результати яких подано у вигляді графіків часових залежностей, гістограм розподілу та узагальнених статистичних показників (рис. 4.6).

Графік порівняння часу реакції демонструє часову динаміку системи в обох режимах функціонування. За цим графіком видно, що у випадку циклічного опитування значення часу реакції є більшими та мають більш виражену нерівномірність.

Реакція системи при циклічному підході визначається моментом найближчої ітерації циклу опитування, внаслідок чого фактичний час відгуку залежить від положення події відносно часової сітки виконання. Це призводить до появи характерних коливань і локальних піків.

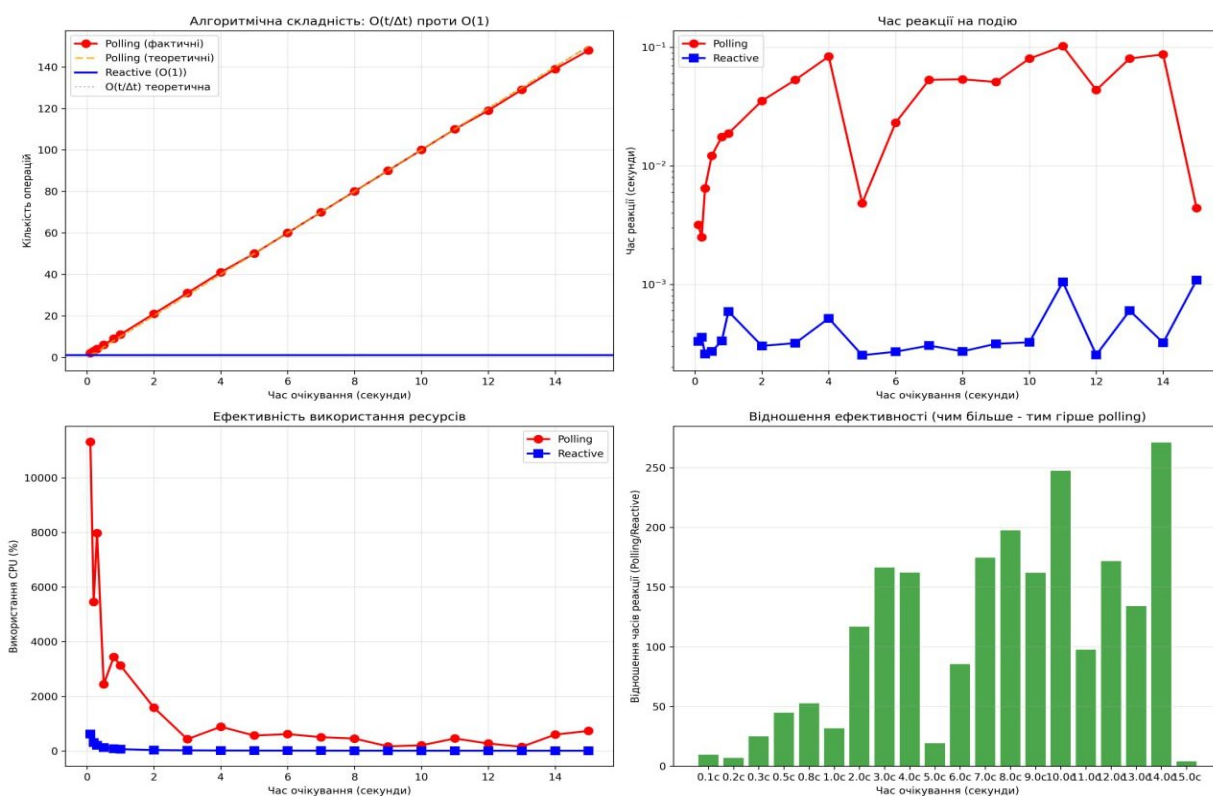


Рисунок 4.6 – Діаграми аналізу алгоритмічної складності методів очікування

Аналіз графіків показав, що у подієво-орієнтованому режимі часовий ряд є значно більш компактним, а значення часу реакції залишаються на нижчому рівні.

Набір графіків, наведений на рис. 4.7, ілюструє узагальнене порівняння продуктивності та ресурсного навантаження для циклічного та подієво-орієнтованого підходів до керування.

Аналіз графіка продуктивності показує, що подієво-орієнтований підхід забезпечує суттєво менший час реакції порівняно з циклічним опитуванням, при цьому рівень використання процесорних ресурсів є значно нижчим, тому що в подієвій моделі обчислення виконуються лише за наявності події, тоді як у циклічному підході система постійно витрачає ресурси на перевірку умов.

Аналіз графіку використання ресурсів показав, що обсяг споживаної пам'яті та кількість потоків для обох підходів є співставними, а це свідчить про відсутність додаткових витрат на рівні базової інфраструктури виконання, і ключова відмінність проявляється тільки у характері використання процесорного часу.

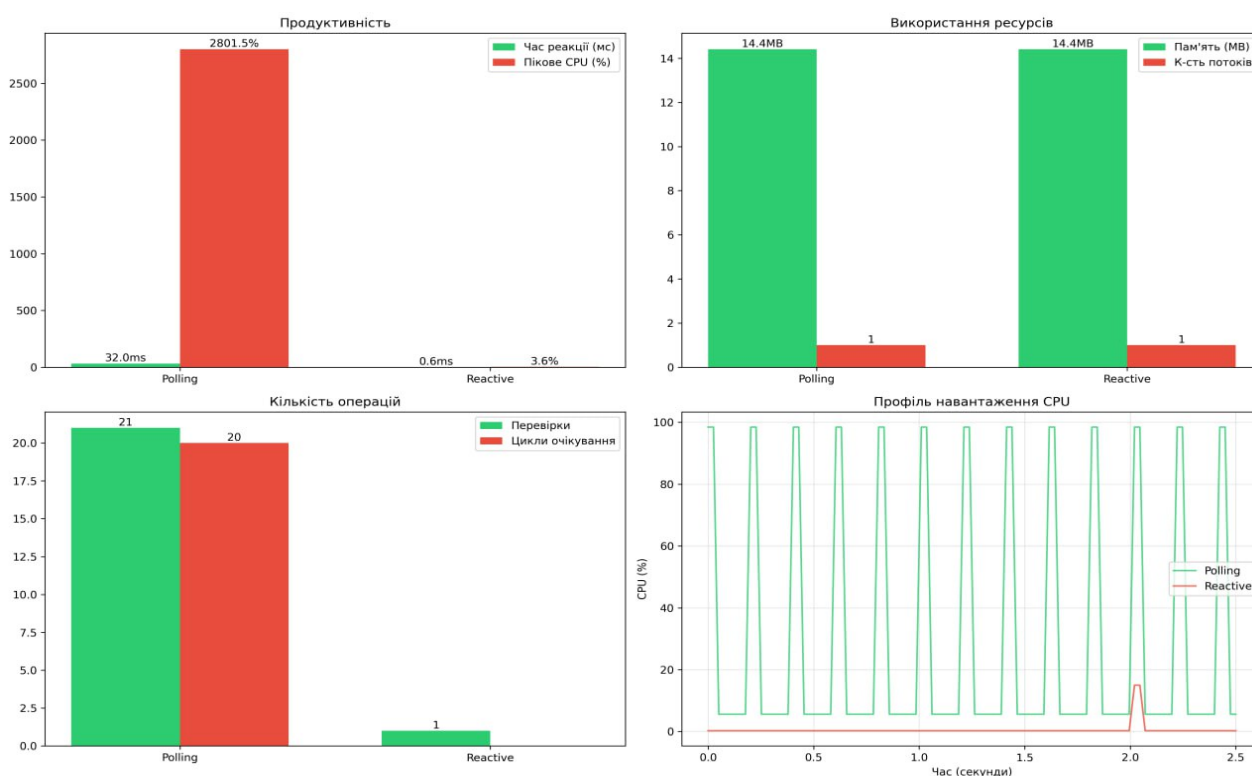


Рисунок 4.7 – Порівняння продуктивності та ресурсного навантаження для циклічного та подієво-орієнтованого підходів

Порівняння кількості операцій показує, що для циклічного підходу характерна велика кількість перевірок і циклів опитування, більшість з яких не призводить до корисних дій, а у подієво-орієнтованому підході обробка ініціюється виключно при виникненні події, що зменшує кількість виконаних операцій і підвищує ефективність обчислювального процесу.

Регулярні піки навантаження у навантаженні CPU у часі відповідають виконанню циклів опитування, тоді як подієво-орієнтований підхід демонструє більш рівномірний та низький рівень завантаження, з короткими імпульсами активності лише в моменти обробки подій, що додатково підтверджує переваги подієво-орієнтованого підходу.

4.3.2 Порівняння ресурсного навантаження

Одним із головних критеріїв ефективності системи керування агентом рою БПЛА є час реакції на зовнішню подію, оскільки саме він визначає здатність системи своєчасно змінювати свій стан і формувати керуючі дії в умовах мінливого середовища. У межах експериментального дослідження проведено комплексну оцінку ефективності двох підходів до обробки подій: циклічного опитування та подієво-орієнтованого механізму. Оцінювання здійснювалось за чотирма ключовими показниками: час реакції, навантаження на процесор, кількість виконаних перевірок та використання пам'яті (рис. 4.8).

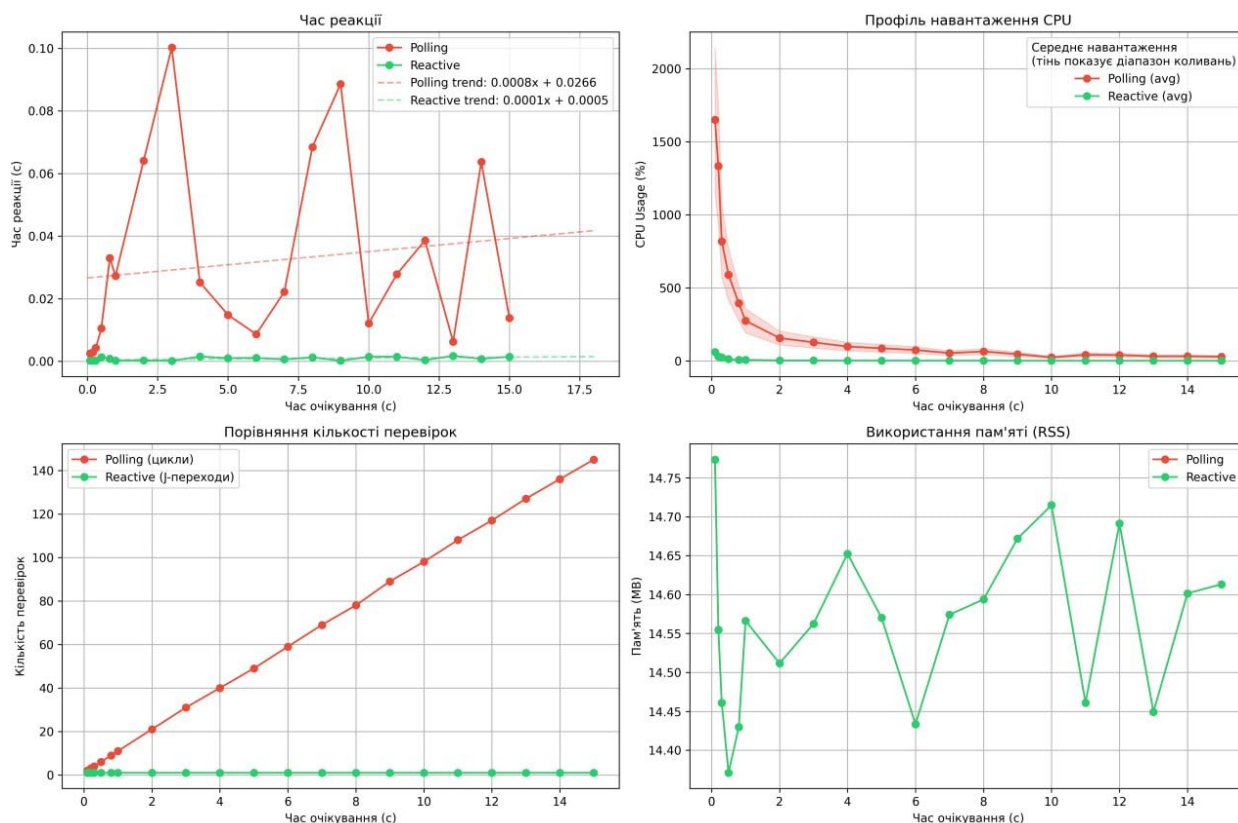


Рисунок 4.8 – Комплексна оцінка ефективності циклічного і подієво-орієнтованого підходів за показниками часу реакції, навантаження CPU, кількості операцій та використання пам'яті

На графіку часу реакції представлено залежність часу реакції системи від тривалості інтервалу очікування, де циклічний підхід характеризується значними коливаннями, піками до ~ 0.1 с, залежністю від інтервалу перевірки, подієво-орієнтований підхід майже сталий (~ 0.001 с), у нього відсутні піки, спостерігається стабільна поведінка. Час реакції при циклічному підході визначається моментом наступного циклу опитування, що призводить до випадкових затримок. Подієво-орієнтований підхід забезпечує практично миттєву реакцію, оскільки обробка виконується безпосередньо в момент виникнення події.

Профіль навантаження CPU для циклічного підходу показав дуже високі пікові навантаження (до 2000%), різкий спад, але нестабільність, а при подієво-орієнтованому підході стабільно низьке навантаження (~ 0 –50%).

Циклічний підхід створює надлишкове навантаження через постійні перевірки навіть при відсутності подій, що показано на графіку кількості перевірок, де

відзначено лінійне зростання ($\approx O(t)$) для циклічного підходу і майже константу для подієво-орієнтованого.

Графік використання пам'яті показав стабільне значення для подієво-орієнтованого підходу ($\sim 14.4\text{--}14.8$ МВ) і невеликі коливання для циклічного.

Отримані результати свідчать про суттєву перевагу подієво-орієнтованого підходу над циклічним опитуванням за всіма ключовими показниками, за винятком використання пам'яті, де відмінності є незначними. Зокрема, забезпечується зменшення часу реакції, зниження навантаження на процесор та усунення надлишкових обчислень, що підтверджує ефективність застосування подієво-орієнтованих механізмів у системах керування роєм БПЛА.

Такий характер залежності підтверджує, що у випадку використання слухачів подій та прямої активації переходів Е-мережі усувається очікування наступного циклу опитування, а обробка починається фактично одразу після виникнення події.

Для поглибленого аналізу ефективності досліджуваних підходів проведено серію експериментів із багаторазовим вимірюванням часу реакції системи. Це дозволило оцінити не лише середні значення, але й стабільність, варіативність та характер розподілу часових показників (рис. 4.9).

Графік часу реакції за тестами показав, що циклічний підхід має значні коливання ($\approx 0.032 - 0.048$ с), нестабільну поведінку, випадкові піки, а подієво-орієнтований підхід представлено практично горизонтальною лінією (~ 0.0004 с) з мінімальними відхиленнями.

Циклічний підхід демонструє залежність часу реакції від моменту виникнення події відносно циклу опитування, що призводить до нестабільності. Подієво-орієнтований підхід забезпечує детерміновану поведінку з мінімальними варіаціями.

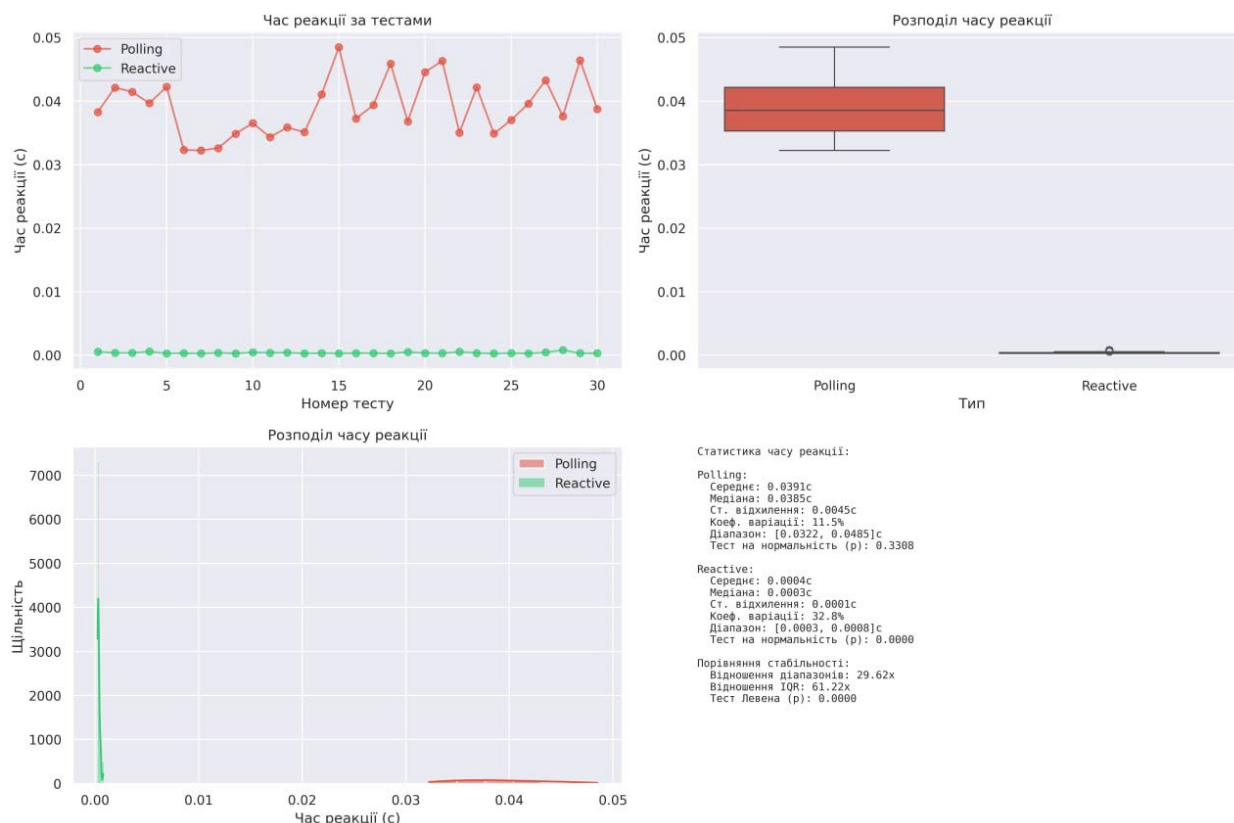


Рисунок 4.9 – Порівняльний статистичний аналіз стабільності часу реакції системи при використанні циклічного та подієво-орієнтованого підходів

Boxplot діаграма показала, що циклічний підхід має широкий міжквартильний розмах, значну варіативність, високі значення, а подієво-орієнтований підхід майже точковий розподіл з мінімальною дисперсією.

Гістограма розподілу показала, що циклічний підхід має широкий розподіл у діапазоні ~ 0.03 – 0.05 с, а подієво-орієнтований підхід вузький пік біля ~ 0.0004 с.

Статистичні показники для циклічного підходу:

- середнє: ~ 0.0391 с;
- стандартне відхилення: ~ 0.005 с;
- коефіцієнт варіації: $\sim 11.5\%$.

Статистичні показники для подієво-орієнтованого підходу:

- середнє: ~ 0.0004 с;
- стандартне відхилення: ~ 0.0001 с;
- коефіцієнт варіації: $\sim 3.2\%$.

Подієво-орієнтований підхід не лише швидший, але й значно стабільніший, що критично для систем реального часу.

Отримані результати підтверджують, що подієво-орієнтований підхід забезпечує не лише зменшення середнього часу реакції, але й суттєве підвищення стабільності та передбачуваності функціонування системи.

4.4 Оцінювання результативності місії на основі вдосконаленої імітаційної моделі рою БПЛА

Проведені експериментальні дослідження дозволили визначити часові характеристики функціонування програмної системи керування роєм БПЛА для циклічного та подієвого підходів та розрахувати математичне очікування часу реакції системи на подію і середньоквадратичне відхилення відповідних значень для кожного з досліджуваних підходів.

Отримані параметри дозволяють перейти від локального оцінювання окремої реакції системи до аналізу загальної результативності виконання місії, для чого вдосконалено розроблену імітаційну модель рою БПЛА, де враховується експериментально визначені характеристики часу реакції та дозволяється виконувати серії порівняльних експериментів для різних способів організації програмного керування.

Основною метою дослідження є перевірка можливості використання розробленої моделі для оцінювання результативності місії, визначеної у підрозділі 2.4.2 як співвідношення кількості нейтралізованих цілей до загальної кількості виявлених цілей. Саме цей показник (рівняння 2.48) використовується як інтегральний критерій ефективності функціонування рою БПЛА.

На відміну від початкової версії моделі у новій версії основну увагу приділено перевірці працездатності алгоритму взаємодії агентів рою на шляху до досягнення поставленої мети. Дослідження часових характеристик вдосконаленої імітаційної моделі рою БПЛА в системі NetLogo отримано статистичні параметри для двох досліджуваних підходів до організації програмного керування – циклічного та подієвого (рис. 4.10).

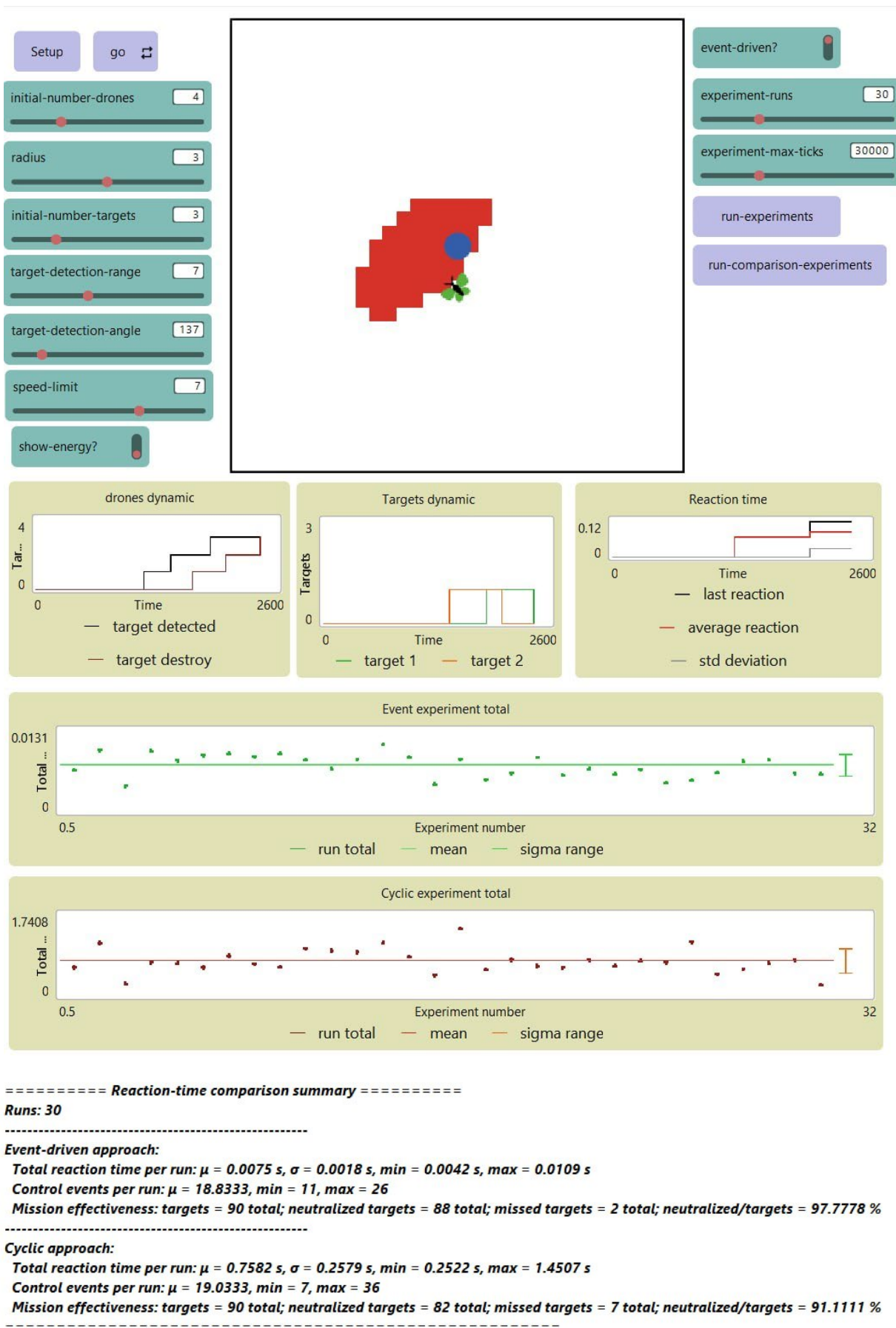


Рисунок 4.10 – Результати експерименту для вдосконаленої імітаційної моделі рою
БПЛА

На відміну від попередньої реалізації, де процеси взаємодії агентів розглядалися спрощено, а вдосконалена імітаційна модель рою БПЛА враховує часові характеристики виконання окремих етапів обробки подій. Це дозволяє моделювати не лише факт виконання певної дії, а й реальний часовий вплив програмної архітектури на загальну швидкість системи, чим модель наближається до реальних умов функціонування рою БПЛА та забезпечує можливість аналізу впливу архітектурних рішень на досягнення цілей місії.

Особливістю запропонованого підходу є те, що отриманий час реакції під час експериментів на базовій моделі характеризує не повний цикл виявлення та нейтралізації цілі, а затримку між виникненням певної події та початком виконання відповідної логіки обробки. У реальній системі під час виконання місії виникає значна кількість додаткових подій різного типу (наприклад, після виявлення цілі агент може отримати повідомлення про результати розпізнавання, інформацію від інших агентів рою, дані про зміну ролей, призначення на виконання атаки або повідомлення про вже виконані дії іншими учасниками групи тощо), що створює додаткові часові витрати.

При моделюванні роботи системи між агентами рою формувалася потік подій, що виникають під час обміну інформацією, прийняття рішень та координації спільних дій, а кожна подія супроводжувалась певною затримкою реакції, параметри якої визначались на основі експериментально отриманих значень математичного очікування та середньоквадратичного відхилення.

Для кожного запуску моделі фіксуються:

- загальна кількість подій;
- сумарний час їх обробки;
- кількість виявлених цілей;
- кількість нейтралізованих цілей;
- результативність місії.

На відміну від аналізу окремої реакції, модель дозволяє оцінити сумарний ефект від великої кількості подій, які виникають протягом одного циклу виконання місії.

Для реалізації можливості порівняльного аналізу до інтерфейсу моделі додано набір нових параметрів керування експериментами:

1) реалізовано перемикач режиму функціонування, який дозволяє обирати між циклічним та подієвим підходами до програмного керування, що забезпечує проведення досліджень в однакових умовах при використанні різних принципів організації програмної взаємодії компонентів системи;

2) додано параметр кількості запусків експериментальної серії дозволило виконувати багаторазове повторення моделювання та накопичувати статистичні дані для подальшого аналізу, оскільки окремий запуск може залежати від випадкових факторів, а використання серій експериментів забезпечує отримання більш об'єктивних результатів та дозволяє оцінити стійкість досліджуваних підходів.

На рис. 4.11 наведено приклади результатів виконання місії для двох досліджуваних підходів до організації програмного керування роєм БПЛА, де чорна лінія відображає накопичену кількість виявлених цілей, а червона – кількість нейтралізованих цілей у процесі виконання місії. По осі абсцис відкладено модельний час, а по осі ординат – кількість цілей.

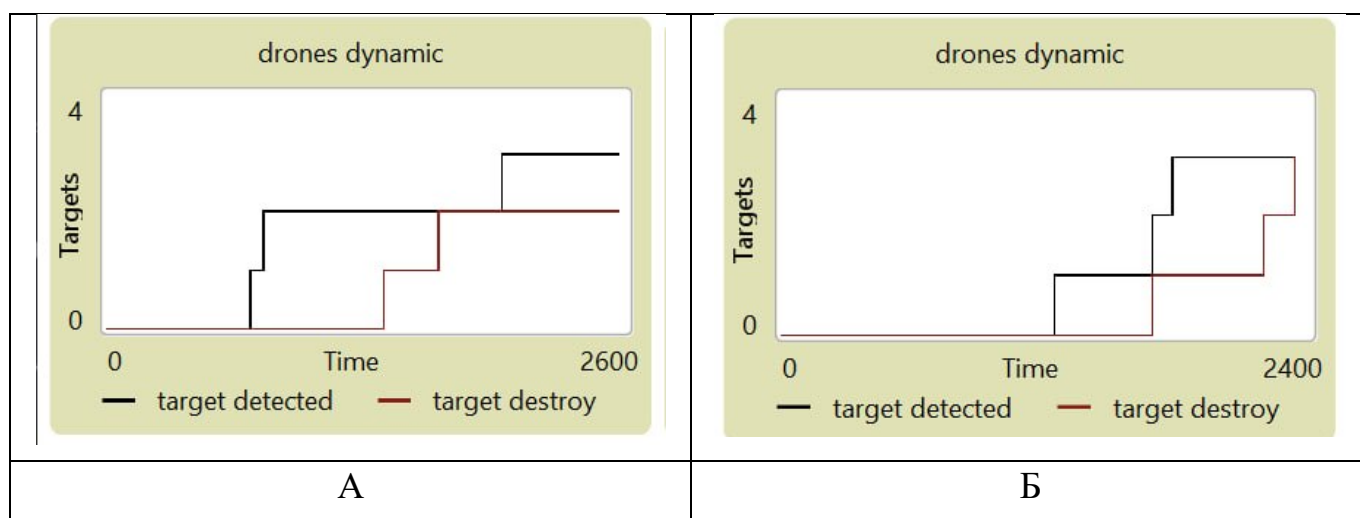


Рисунок 4.11 – Приклади результатів виконання місії для циклічного (а) та подієвого (б) підходів

Аналіз результатів для циклічного підходу (рис. 4.11, а) показує наявність помітних часових інтервалів між моментами виявлення цілей та їх подальшою нейтралізацією, що обумовлено необхідністю періодичного опитування станів системи керування та накопиченням часу очікування між послідовними циклами обробки подій.

Для подієво-орієнтованого підходу (рис. 4.11, б) спостерігається більш оперативна реакція системи на появу нових подій, а відстань між моментом виявлення та моментом нейтралізації цілі є меншою, що свідчить про скорочення затримок при передачі інформації та прийнятті рішень.

Слід зазначити, що обидва підходи забезпечують високий відсоток успішності виконання місії, але характер зміни графіків демонструє різну швидкість реагування на загрози та різну інтенсивність виконання операцій перехоплення (рис. 4.12).

```

-----
Event-driven approach:
  Total reaction time per run:  $\mu = 0.0081$  s,  $\sigma = 0.0019$  s, min = 0.0048 s, max = 0.0125 s
  Control events per run:  $\mu = 20.1667$ , min = 12, max = 29
  Mission effectiveness: targets = 90 total; neutralized targets = 89 total; missed targets = 1 total; neutralized/targets = 98.8889 %
-----
Cyclic approach:
  Total reaction time per run:  $\mu = 0.7603$  s,  $\sigma = 0.1927$  s, min = 0.4619 s, max = 1.1755 s
  Control events per run:  $\mu = 19.2333$ , min = 11, max = 31
  Mission effectiveness: targets = 90 total; neutralized targets = 82 total; missed targets = 8 total; neutralized/targets = 91.1111 %
=====

-----
Event-driven approach:
  Total reaction time per run:  $\mu = 0.0065$  s,  $\sigma = 0.002$  s, min = 0.0023 s, max = 0.0106 s
  Control events per run:  $\mu = 16.5333$ , min = 8, max = 26
  Mission effectiveness: targets = 90 total; neutralized targets = 87 total; missed targets = 3 total; neutralized/targets = 96.6667 %
-----
Cyclic approach:
  Total reaction time per run:  $\mu = 0.7458$  s,  $\sigma = 0.1535$  s, min = 0.3656 s, max = 1.0433 s
  Control events per run:  $\mu = 18.7$ , min = 9, max = 27
  Mission effectiveness: targets = 90 total; neutralized targets = 78 total; missed targets = 12 total; neutralized/targets = 86.6667 %
=====

-----
Event-driven approach:
  Total reaction time per run:  $\mu = 0.0073$  s,  $\sigma = 0.0018$  s, min = 0.0034 s, max = 0.011 s
  Control events per run:  $\mu = 18.2333$ , min = 8, max = 26
  Mission effectiveness: targets = 90 total; neutralized targets = 90 total; missed targets = 0 total; neutralized/targets = 100 %
-----
Cyclic approach:
  Total reaction time per run:  $\mu = 0.6658$  s,  $\sigma = 0.2051$  s, min = 0.3628 s, max = 1.0977 s
  Control events per run:  $\mu = 17.0333$ , min = 9, max = 33
  Mission effectiveness: targets = 90 total; neutralized targets = 85 total; missed targets = 5 total; neutralized/targets = 94.4444 %
=====

```

Рисунок 4.12 – Приклади результатів множинного запуску імітаційної моделі

Встановлено, що використання подієво-орієнтованого механізму керування позитивно впливає на інтегральний показник результативності місії. Незважаючи на однакові початкові умови моделювання, подієвий підхід забезпечив вищу частку нейтралізованих цілей у всіх серіях експериментів. Це підтверджує, що зменшення часу реакції системи трансформується у покращення функціональної ефективності рою БПЛА та дозволяє більш повно реалізувати потенціал групового інтелекту під час виконання місії.

Отримані результати моделювання в системі NetLogo підтверджують доцільність використання подієво-орієнтованого підходу для організації програмного керування роєм БПЛА, тому що навіть при однакових початкових умовах моделювання подієво-орієнтований підхід забезпечує більш швидке реагування на появу загроз та зменшує часові витрати на виконання місії в цілому.

4.5 Узагальнення методів програмного керування роєм та практичні рекомендації

Проведені експериментальний та аналітичний аналіз дозволяють сформулювати узагальнені положення щодо доцільності використання різних підходів до організації керування в системах ройових безпілотних літальних апаратів. Отримані результати свідчать, що ефективність системи керування визначається не лише вибором алгоритмів прийняття рішень, але й способом організації обчислювального процесу, зокрема механізмом обробки подій.

Порівняння циклічного та подієво-орієнтованого підходів показало, що останній забезпечує суттєві переваги у швидкості реакції, ефективності використання ресурсів та масштабованості системи. Водночас результати дослідження свідчать, що вибір підходу має здійснюватися з урахуванням специфіки задачі, характеру середовища та вимог до системи керування.

4.5.1 Сценарії з критичністю подієвого підходу

Подієво-орієнтований підхід є найбільш ефективним у системах, де поведінка визначається асинхронними змінами середовища та потребує негайної реакції. До таких сценаріїв належать задачі, у яких події виникають нерегулярно, але мають критичне значення для функціонування системи.

У першу чергу це системи уникнення зіткнень, де затримка у реакції навіть на незначний інтервал часу може призвести до аварійної ситуації. У таких задачах важливо мінімізувати час від моменту виявлення перешкоди до формування керуючого сигналу, що забезпечується саме подієво-орієнтованим підходом.

Іншим прикладом є задачі координації руху рою у змінному середовищі, де агенти повинні реагувати на дії інших учасників, зміну траєкторій або появу нових обмежень. У цьому випадку подієво-орієнтований підхід дозволяє обробляти лише актуальні зміни стану системи, що забезпечує як швидкодію, так і зменшення обчислювального навантаження.

До сценаріїв, де подієвий підхід є критичним, також належать системи обробки сенсорної інформації, де дані надходять нерівномірно та мають різну пріоритетність, тому що використання циклічного опитування призводить до або надмірних витрат ресурсів, або до збільшення затримок, а подієвий підхід забезпечує адаптивну реакцію на вхідні сигнали.

Незважаючи на виявлені обмеження, циклічний підхід до організації керування не втрачає актуальності та може бути ефективно застосований у ряді задач, де його недоліки не мають критичного впливу на функціонування системи:

- у системах із стабільним та передбачуваним середовищем, де зміни стану відбуваються регулярно та не потребують негайної реакції, коли періодичне виконання перевірок забезпечує достатню точність і не призводить до суттєвих втрат ефективності;

- для задач з невеликою кількістю контрольованих параметрів, де обчислювальні витрати залишаються низькими навіть при постійному виконанні перевірок, коли циклічне опитування виправдане за рахунок простоти реалізації та передбачуваності поведінки;

- для систем з жорстко регламентованими часовими інтервалами виконання, де необхідно забезпечити регулярне оновлення стану незалежно від наявності подій, тоді циклічний підхід дозволяє гарантувати детермінований режим роботи, що може бути критично важливим для певних типів контролю.

4.5.2 Практичні рекомендації щодо проєктування систем керування роєм

Проведене теоретичне, модельне та експериментальне дослідження дозволяє сформулювати практичні рекомендації щодо проєктування систем керування роєм безпілотних літальних апаратів, у яких поєднуються вимоги до швидкодії,

масштабованості, енергоефективності та передбачуваності поведінки. На відміну від локальних висновків, отриманих у попередніх підрозділах, даний підрозділ узагальнює результати всієї роботи і подає їх у прикладній формі, придатній для використання під час розробки реальних або напівнатурних систем керування.

Передусім результати дослідження підтверджують, що при проєктуванні сучасних систем керування роєм БПЛА доцільно відмовлятися від уявлення про керуючу логіку як про набір ізольованих процедур або послідовних програмних модулів, що викликаються у фіксованому циклі. Практика проєктування має базуватися на тому, що логіка поведінки агента повинна бути структурованою, формально описаною та безпосередньо пов'язаною з механізмом її виконання.

З цієї точки зору доцільно використовувати виконувані моделі керування на основі керуючих Е-мереж як центральний компонент системи. Практична перевага такого підходу полягає в тому, що одна і та сама структура одночасно виконує роль формального опису алгоритму, внутрішньої логіки виконання та основи для експериментального аналізу. Це означає, що під час розробки систем керування роєм БПЛА рекомендується будувати архітектуру не навколо окремих процедурних обробників, а навколо єдиної мережевої моделі, у якій стани, переходи, умови активації та часові обмеження задані явно. Така організація спрощує перевірку коректності, дає можливість контролювати поведінку системи на рівні структури та забезпечує більшу узгодженість між етапами проєктування, реалізації та тестування.

Отримані результати також показують, що під час проєктування систем керування роєм особливу увагу необхідно приділяти способу інтеграції із зовнішнім середовищем. У традиційних системах ця інтеграція часто реалізується через циклічне опитування сенсорів, каналів зв'язку та службових змінних, однак проведений аналіз показав, що саме цей підхід є одним із головних джерел надлишкового навантаження. Тому при розробці практичних систем рекомендується застосовувати подієво-орієнтовану схему інтеграції, у якій всі зовнішні сигнали перетворюються у події, що безпосередньо впливають на стан моделі. Іншими словами, система повинна не «шукати» події шляхом постійних перевірок, а «отримувати» їх через слухачі подій і реактивно перебудовувати власний стан.

Це має пряме прикладне значення для організації сенсорного контуру БПЛА. Практично це означає, що вимірювання IMU, камери, дальномірів, барометра, GPS або комунікаційних модулів не повинні безпосередньо викликати розгалужену процедурну логіку. Натомість вони повинні проходити через шар інтерпретації, який переводить сирі дані у формалізовані події певного типу: навігаційні, сенсорні, енергетичні, комунікаційні або тригерні. Така нормалізація подій дозволяє зменшити зв'язаність між апаратною частиною та керуючою логікою і створює більш стабільну основу для подальшого масштабування системи.

Важливим практичним висновком є те, що для ройових систем не слід зводити представлення стану агента до однієї глобальної змінної режиму. Результати розділу 3 та експериментальна перевірка у розділі 4 показали, що значно ефективнішим є розподілене подання стану через маркування мережі, глобальні змінні та асоційовані дані. Це дозволяє одночасно представляти кілька активних підпроцесів, що є природним для агента рою, який паралельно виконує навігацію, аналізує середовище, обмінюється повідомленнями та контролює безпечність руху. У практичному проектуванні це означає, що слід уникати архітектур, де вся поведінка агента згортається до одного лінійного автомату або одного циклу обробки, і натомість проектувати модель як систему частково незалежних, але синхронізованих гілок виконання.

Окремої уваги потребує організація паралелізму. Для задач керування роєм БПЛА проведене дослідження підтвердило, що паралельне виконання є не додатковою оптимізацією, а базовою вимогою архітектури. Тому при практичній реалізації рекомендується від самого початку передбачати наявність незалежних або частково незалежних контурів виконання, які можуть обробляти різні типи подій. Разом із цим паралелізм не повинен бути стихійним або неформалізованим. Він має бути відображений у самій структурі моделі через механізми `fork/join`, умовні переходи, злиття потоків та черги. Це забезпечує керованість паралельного виконання і дозволяє запобігати конфліктам між потоками даних або переходами.

Практична цінність запропонованого підходу особливо проявляється у використанні черг задач і подій. У реальних системах БПЛА події надходять нерівномірно, а їх важливість може суттєво відрізнятися. З цієї причини

рекомендовано використовувати не лише стандартну буферизацію, а моделі черг із контрольованою політикою обробки, наприклад FIFO, LIFO або пріоритетні черги. Це особливо важливо для сценаріїв, де команди безпеки або аварійні сигнали повинні оброблятися раніше за другорядні повідомлення. Отже, у практичних системах обробка подій не повинна бути «пласкою»; вона має відображати функціональну вагу подій і бути інтегрованою у загальну архітектуру мережевої моделі.

Суттєвим висновком роботи є те, що під час проєктування системи керування необхідно враховувати не лише функціональну правильність алгоритму, але й його профіль виконання. Проведене дослідження показало, що навіть формально коректна модель може бути непридатною для практичного використання, якщо вона породжує надмірне навантаження на процесор, нестабільні затримки або пікове споживання пам'яті. У зв'язку з цим рекомендується інтегрувати механізми статистичного моніторингу безпосередньо в програмну реалізацію моделі. Практично це означає, що кожен перехід, кожна позиція та кожен цикл обробки повинні залишати слід у системі профілювання: кількість активацій, час виконання, сумарний час активності, середні та пікові значення, що дозволяє не лише виявляти вузькі місця, але й робити архітектурні висновки ще до етапу натурних випробувань.

З точки зору вибору апаратної платформи результати дослідження дають підстави рекомендувати енергоефективні edge-платформи класу NVIDIA Jetson Nano або споріднені рішення як придатне середовище для первинної реалізації та перевірки подієво-орієнтованих систем керування. Це обумовлено тим, що саме на таких платформах особливо виразно проявляються переваги запропонованого методу: зменшення середнього завантаження процесора, відсутність постійного фону обчислень, краща масштабованість і більш стабільний профіль виконання. Водночас робота показує, що при перенесенні системи на ще більш обмежені платформи потрібно окремо контролювати довжину ланцюгів переходів, розміри черг і вартість обробки сенсорних подій, аби не втратити переваги подієвого підходу через невдалу реалізацію.

Практичне значення має й висновок щодо вибору між чисто подієво-орієнтованим і змішаним підходом. Дослідження показало, що у більшості критичних для БПЛА сценаріїв доцільно віддавати перевагу саме подієвій моделі. Проте це не

означає повної відмови від будь-яких циклічних механізмів. У реальних системах допустимим є використання періодичних перевірок для допоміжних або діагностичних функцій, зокрема для фонові перевірки працездатності компонентів, журналювання або періодичного оновлення некритичних параметрів.

Під час проєктування ройових систем також доцільно орієнтуватися не на універсальний «середній» алгоритм, а на клас сценаріїв, для яких система призначена. Якщо мова йде про уникнення зіткнень, аварійне реагування, кооперативне маневрування, захист об'єкта або обробку нестандартних ситуацій, подієво-орієнтована модель має бути основою архітектури. Якщо ж система виконує регулярний моніторинг, патрулювання за стабільними маршрутами або роботу в середовищі зі слабкою динамікою, допускається включення елементів циклічного керування. Отже, практичне проєктування повинно бути сценарно-орієнтованим, а не універсально однаковим для всіх типів місій.

Важливою рекомендацією, яка впливає з усієї роботи, є необхідність збереження єдності між формальною моделлю, її програмною реалізацією та експериментальною перевіркою. У багатьох системах модель створюється окремо від коду, а експерименти взагалі проводяться на абстрактному макеті, що ускладнює доведення адекватності результатів. Запропонований підхід показує, що значно ефективніше будувати всю систему навколо однієї виконуваної моделі, яка одночасно є об'єктом формального аналізу, ядром програмної реалізації та джерелом даних для експериментальної оцінки. Саме ця цілісність є одним із головних практичних результатів роботи.

Можна сформулювати таку узагальнену рекомендацію: системи керування роєм БПЛА доцільно проєктувати як подієво-орієнтовані, мережево-структуровані та виконувані моделі, у яких зовнішні сигнали перетворюються у внутрішні події через слухачі, стан агента подається розподілено через маркування та дані, паралельні процеси організуються явно через структуру мережі, а профілювання і контроль працездатності вбудовуються в систему з самого початку.

4.6 Висновки до розділу 4

У четвертому розділі проведено комплексну експериментальну перевірку запропонованого методу програмного керування роєм безпілотних літальних апаратів, що базується на використанні керуючих Е-мереж та подієво-орієнтованого підходу до організації взаємодії паралельних процесів. Дослідження виконувались як на рівні окремих механізмів обробки подій, так і на рівні виконання місії роєм БПЛА в цілому.

Експериментально підтверджено, що використання подієвих тригерів переходів забезпечує суттєве скорочення часу реакції системи порівняно з циклічним опитуванням станів, а аналіз ресурсного навантаження показав, що подієво-орієнтований механізм керування забезпечує більш раціональне використання обчислювальних ресурсів. На відміну від циклічного опитування запропонований підхід активує обробку лише в моменти виникнення подій, що дозволяє зменшити середнє навантаження на обчислювальну платформу.

Проведене дослідження масштабованості показало, що зі збільшенням кількості агентів та складності сценарію подієво-орієнтований підхід зберігає стабільні характеристики швидкодії, тоді як для циклічного опитування спостерігається зростання часових та обчислювальних витрат. Отримані результати підтверджують доцільність використання реактивних механізмів у системах керування великими роями БПЛА.

На основі експериментально визначених часових характеристик було виконано модернізацію імітаційної моделі рою БПЛА та проведено серію порівняльних експериментів для циклічного і подієвого підходів до керування. Це дозволило перейти від аналізу окремих подій до оцінювання результативності виконання місії в цілому.

Отримані результати підтверджують адекватність запропонованої моделі програмного керування роєм БПЛА на основі керуючих Е-мереж та доцільність використання подієво-орієнтованого підходу в системах колективної взаємодії автономних агентів.

Матеріали розділу опубліковано у роботах автора: [19].

ВИСНОВКИ

У дисертаційній роботі досягнуто поставленої мети, що полягала у підвищенні ефективності контролю за роєм дронів при виконанні місії за рахунок використання групового інтелекту та скорочення часу реакції на події при модельно-орієнтованому керуванні дроном-агентом з використанням MVC-підходу до організації взаємодіючих паралельних процесів. Запропонований метод дозволяє представити процеси керування у вигляді виконуваної моделі, що дозволяє безпосередньо інтегрувати її у програмне забезпечення керування агентами. На відміну від традиційних підходів, які базуються на циклічному опитуванні станів і призводять до надлишкового обчислювального навантаження та варіативної затримки реакції, розроблені підходи реалізують механізм активації обчислень виключно у момент виникнення подій, що суттєво підвищує ефективність функціонування системи.

У межах досягнення поставленої мети забезпечено подієво-орієнтовану обробку змін станів, яка дозволяє системі реагувати на зовнішні та внутрішні впливи без необхідності постійного контролю умов, що є критично важливим для мінливих середовищ функціонування рою БПЛА. Водночас реалізовано підтримку паралельності процесів на рівні окремого агента, що дає змогу одночасно виконувати декілька функціональних підпроцесів, таких як обробка сенсорних даних, оцінка стану, підтримка стабілізації та прийняття керуючих рішень. Така організація обчислень забезпечує більш ефективне використання ресурсів і дозволяє уникнути блокування виконання при обробці незалежних задач.

Особливу увагу приділено забезпеченню передбачуваної часової поведінки системи, що досягається за рахунок відмови від циклічних механізмів і переходу до подієвої моделі виконання. Це дозволяє мінімізувати затримки реакції та зменшити їх варіативність, що є критичним для систем реального часу, до яких належать рої безпілотних літальних апаратів. У результаті запропонований метод формує основу для побудови масштабованих, ефективних та реактивних систем керування, здатних функціонувати в умовах невизначеності та змінного середовища.

У результаті виконаного дослідження отримано сукупність наукових результатів, що мають ознаки новизни та спрямовані на розвиток методів формалізованого керування мультиагентними системами:

1. Вперше:

– розроблено формальну модель алгоритму керування агентом у складі рою дронів, яка, на відміну від існуючих, безпосередньо виконує роль трирівневої програми керування з подієво-орієнтовним реагуванням на зміну ситуації, що забезпечує узгоджене прийняття рішень при виконанні місії. На відміну від існуючих формальних моделей, які використовуються переважно для аналізу або імітаційного моделювання, запропонована модель має виконуваний характер і може бути безпосередньо інтегрована у програмне забезпечення керування;

– розроблено мультиагентну модель поведінки рою мультикоптерних дронів при виконанні місії із захисту критичних інфраструктурних об'єктів, яка, на відміну від існуючих, відтворює роль групового інтелекту в процесі відбиття нападу повітряних цілей з урахуванням можливостей підсистем спостереження та розпізнавання, що дозволяє отримувати попередні оцінки ефективності проектних рішень по системним показникам;

2. Удосконалено формальний апарат керуючих Е-мереж шляхом введення подієвих тригерів переходів, що забезпечує асинхронну активацію елементів моделі на рівнях реагування, планування та взаємодії без використання циклічного сканування станів системи керування. Результати серії імітаційних експериментів показали, що застосування подієво-орієнтованого підходу забезпечує не лише скорочення часу реакції приблизно у 100 разів, але й підвищення результативності виконання місії рою БПЛА в середньому з 90,83% до 98,33% за рахунок зменшення кількості пропущених цілей та більш оперативної координації дій агентів.

3. Отримав подальший розвиток метод програмного керування роєм дронів, який, на відміну від існуючих, базується на MVC-підході в запропонованій інтерпретації до процесів керування та взаємодії вбудованих моделей агентів у реальному часі, що забезпечує самоорганізацію поведінки рою дронів з використанням групового інтелекту. У межах запропонованої інтерпретації модель

представлена керуючими Е-мережами, що формалізують логіку переходів та обробку станів, контролер реалізує механізм виконання переходів і керує динамікою обчислювального процесу, тоді як подання забезпечує засоби спостереження, візуалізації та профілювання поведінки системи в реальному часі.

У сукупності отримані результати формують новий підхід до побудови систем керування роєм безпілотних літальних апаратів, який базується на інтеграції формальних моделей, подієво-орієнтованого виконання та сучасних архітектурних принципів програмних систем.

У процесі виконання дисертаційного дослідження повністю розв'язано поставлені завдання, що забезпечило досягнення визначеної мети. Зокрема, проведено ґрунтовний аналіз сучасних підходів до моделювання та керування ройовими системами безпілотних літальних апаратів, у результаті якого виявлено їх ключові обмеження, пов'язані з недостатньою реактивністю, обмеженою масштабованістю та неефективним використанням обчислювальних ресурсів.

Завершальним етапом дослідження стало проведення експериментального порівняння подієво-орієнтованого механізму керування з традиційним циклічним опитуванням станів, у ході якого виконано оцінку часових характеристик, ресурсного навантаження та функціональної ефективності системи. На основі експериментально визначених параметрів часу реакції було виконано модернізацію імітаційної моделі рою БПЛА та проведено серію порівняльних досліджень для циклічного і подієво-орієнтованого підходів. Отримані результати показали, що зменшення часу реакції позитивно впливає на результативність виконання місії, забезпечуючи більш оперативне виявлення та нейтралізацію повітряних цілей, що підтвердило можливість використання розробленої моделі для оцінювання ефективності програмного керування роєм БПЛА та продемонструвало переваги запропонованого підходу в умовах мінливого середовища.

Розроблена мультиагентна модель поведінки рою БПЛА забезпечує можливість отримання попередніх оцінок результативності місії, що дозволяє враховувати вплив часових характеристик програмного забезпечення на процеси виявлення та нейтралізації цілей, а серія імітаційних експериментів підтвердила, що зменшення

часу реакції системи підвищує оперативність виконання місії та покращує узгодженість колективної поведінки агентів.

Отримані в дисертаційній роботі результати мають важливе теоретичне значення, оскільки розвивають підходи до формалізації керування складними розподіленими системами. Запропонована інтеграція керуючих Е-мереж із подієво-орієнтованими механізмами виконання дозволяє формалізувати паралельні та асинхронні процеси у мультиагентних системах, що забезпечує можливість опису динаміки подібних систем у вигляді виконуваних моделей, поєднує формальну строгість із практичною реалізованістю та створює підґрунтя для подальшого розвитку методів моделювання і керування в задачах колективної поведінки автономних агентів.

Практичне значення отриманих результатів полягає у можливості їх безпосереднього застосування при розробці систем керування роєм безпілотних літальних апаратів, робототехнічними комплексами та іншими розподіленими інформаційно-керуючими системами, що функціонують у реальному часі. Запропонований метод забезпечує можливість прямого вбудовування формальної моделі у програмне забезпечення керування, що спрощує процес її інтеграції та впровадження, а реалізація подієво-орієнтованого підходу дозволяє зменшити обчислювальне навантаження, підвищити швидкодію системи та забезпечити її масштабованість при збільшенні кількості агентів.

Експериментальні дослідження, проведені в роботі, підтвердили ефективність запропонованого підходу. Зокрема, встановлено, що подієво-орієнтований механізм керування забезпечує суттєве зменшення часу реакції системи на події, зниження навантаження на обчислювальні ресурси та підвищення стабільності й передбачуваності її поведінки. Отримані результати експериментів демонструють, що подієво-орієнтований підхід не лише зменшує час реакції системи, але й забезпечує більш узгоджену реалізацію механізмів групового інтелекту, оскільки дозволяє агентам реагувати на події незалежно та асинхронно.

У роботі показано, що використання керуючих Е-мереж сприяє формалізації алгоритмів управління агентами рою БПЛА, механізмів обробки подій та узгодженої

взаємодії, що створює умови для реалізації колективної поведінки рою як прояву групового інтелекту без централізованого управління.

Результати дисертаційного дослідження впроваджено у наукову та науково-технічну діяльність Національного університету «Чернігівська політехніка» в межах виконання державних науково-дослідних робіт [106], що підтверджує їх практичну значущість та прикладну цінність. Також результати дисертаційного дослідження впроваджено у Державному науково-дослідному інституті випробувань і сертифікації озброєння та військової техніки для проведення натурних досліджень та відпрацювання програм і методик випробувань БПЛА та у навчальний процес з підготовки аспірантів в Інституті проблем математичних машин і систем НАН України (Додаток А).

Подальші дослідження доцільно спрямувати на розширення запропонованої моделі для гетерогенних роїв безпілотних апаратів [32], інтеграцію з методами машинного навчання для підвищення адаптивності систем, оптимізацію алгоритмів у реальних апаратних середовищах [33], а також на підвищення відмовостійкості та безпеки систем керування в умовах невизначеності та зовнішніх впливів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chung S. J., Paranjape A. A., Dames P., Shen S., Kumar V. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*. 2018. Vol. 34, No 4. P. 837–855. DOI: <https://doi.org/10.1109/TRO.2018.2857475>.
2. Shahzad M. M., Saeed Z., Akhtar A., Munawar H., Yousaf M. H., Baloach N. K., Hussain F. A review of swarm robotics in a nutshell. *Drones*. 2023. Vol. 7, No 4. Article 269. DOI: <https://doi.org/10.3390/drones7040269>.
3. Coppola M., McGuire K. N., De Wagter C., de Croon G. C. H. E. A survey on swarming with micro air vehicles: fundamental challenges and constraints. *Frontiers in Robotics and AI*. 2020. Vol. 7. Article 18. DOI: <https://doi.org/10.3389/frobt.2020.00018>.
4. Barca J. C., Sekercioglu Y. A. Swarm robotics reviewed. *Robotica*. 2013. Vol. 31, No 3. P. 345–359. DOI: <https://doi.org/10.1017/S026357471200032X>.
5. Rossi F., Bandyopadhyay S., Wolf M., Pavone M. Review of multi-agent algorithms for collective behavior: a structural taxonomy. *IFAC-PapersOnLine*. 2018. Vol. 51, No 12. P. 112–117. DOI: <https://doi.org/10.1016/j.ifacol.2018.07.097>.
6. Reynolds C. W. Flocks, herds and schools: a distributed behavioral model. *ACM SIGGRAPH Computer Graphics*. 1987. Vol. 21, No 4. P. 25–34. DOI: <https://doi.org/10.1145/37402.37406>.
7. Olfati-Saber R., Murray R. M. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*. 2004. Vol. 49, No 9. P. 1520–1533. DOI: <https://doi.org/10.1109/TAC.2004.834113>.
8. Lizzio F. F., Capello E., Guglieri G. A review of consensus-based multi-agent UAV implementations. *Journal of Intelligent and Robotic Systems*. 2022. Vol. 106. Article 43. DOI: <https://doi.org/10.1007/s10846-022-01743-9>.
9. Chen Q., Wang Y., Jin Y., Wang T., Nie X., Yan T. A survey of an intelligent multi-agent formation control. *Applied Sciences*. 2023. Vol. 13, No 10. Article 5934. DOI: <https://doi.org/10.3390/app13105934>.

10. Rahman M., Sarkar N. I., Lutui R. A survey on multi-UAV path planning: classification, algorithms, open research problems, and future directions. *Drones*. 2025. Vol. 9, No 4. Article 263. DOI: <https://doi.org/10.3390/drones9040263>.
11. Ali Z. A., Zhangang H., Hang W. B. Cooperative path planning of multiple UAVs by using max-min ant colony optimization along with Cauchy mutant operator. *Fluctuation and Noise Letters*. 2021. Vol. 20, No 1. Article 2150002. DOI: <https://doi.org/10.1142/S0219477521500024>.
12. Arshid K., Krayani A., Marcenaro L., Gomez D. M., Regazzoni C. Toward autonomous UAV swarm navigation: a review of trajectory design paradigms. *Sensors*. 2025. Vol. 25, No 18. Article 5877. DOI: <https://doi.org/10.3390/s25185877>.
13. Rezaee M. R., Hamid N. A. W. A., Hussin M., Zukarnain Z. A. Comprehensive review of drones collision avoidance schemes: challenges and open issues. *IEEE Transactions on Intelligent Transportation Systems*. 2024. Vol. 25, No 7. P. 6397–6426. DOI: <https://doi.org/10.1109/TITS.2024.3375893>.
14. Ali Z. A., Israr A., Alkhamash E. H., Hadjouni M. A leader-follower formation control of multi-UAVs via an adaptive hybrid controller. *Complexity*. 2021. Vol. 2021. P. 1–16. Article 9231636. DOI: <https://doi.org/10.1155/2021/9231636>.
15. Wu H., Peng Q., Shi M., Xing L. A survey of UAV swarm task allocation based on the perspective of coalition formation. *International Journal of Swarm Intelligence Research*. 2022. Vol. 13, No 1. P. 1–22. DOI: <https://doi.org/10.4018/IJSIR.311499>.
16. Kazymyr V., Oleksiienko P., Chornonoh O., Rohovenko A. Modeling of Defensive Drone Swarms with NetLogo. Mathematical Modeling and Simulation of Systems. MODS 2023. Lecture Notes in Networks and Systems. Cham: Springer, 2024. Vol. 1091. P. 377–387. DOI: https://doi.org/10.1007/978-3-031-67348-1_28.
17. Олексієнко П. Д., Казимир В. В. Вбудовані моделі алгоритму керування роєм БПЛА. *Математичні машини і системи*. 2025. № 3–4. С. 90–100. DOI: <https://doi.org/10.34121/1028-9763-2025-3-4-90-100>.
18. Олексієнко П. Д., Казимир В. В. Подієво-орієнтований підхід в реалізації вбудованих систем керування роєм БПЛА. *Технічні науки та технології*. 2025. № 3(41). С. 225–236. DOI: [https://doi.org/10.25140/2411-5363-2025-3\(41\)-225-236](https://doi.org/10.25140/2411-5363-2025-3(41)-225-236).

19. Suslo M., Kazymyr V., Oleksiienko P., Kagitin D. Semi-Physical Modeling of Drone Swarm Control Processes. 2025 15th International Conference on Advanced Computer Information Technologies (ACIT): conference proceedings, Šibenik, Croatia, 17–19 September 2025. Šibenik, 2025. P. 102–108. DOI: <https://doi.org/10.1109/ACIT65614.2025.11185750>.
20. Kahitin D., Kazymyr V., Suslo M., Yatchenko Y., Oleksiienko P. Drone Positioning in a Specified Location Using Visual Data Under GPS-Denied Conditions. 2025 IEEE 13th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS): conference proceedings, Gliwice, Poland, 4–6 September 2025. Gliwice, 2025. P. 318–323. DOI: <https://doi.org/10.1109/IDAACS68557.2025.11322385>.
21. Chen X., Tang J., Lao S. Review of unmanned aerial vehicle swarm communication architectures and routing protocols. *Applied Sciences*. 2020. Vol. 10, No 10. Article 3661. DOI: <https://doi.org/10.3390/app10103661>.
22. Yanmaz E., Yahyanejad S., Rinner B., Hellwagner H., Bettstetter C. Drone networks: communications, coordination, and sensing. *Ad Hoc Networks*. 2018. Vol. 68. P. 1–15.
23. Castrillo V. U., Manco A., Pascarella D., Gigante G. A review of counter-UAS technologies for cooperative defensive teams of drones. *Drones*. 2022. Vol. 6, No 3. Article 65.
24. Abdelkader M., Güler S., Jaleel H., Shamma J. S. Aerial swarms: recent applications and challenges. *Current Robotics Reports*. 2021. Vol. 2. P. 309–320.
25. Alladi T., Chamola V., Sahu N., Guizani M. Applications of blockchain in unmanned aerial vehicles: a review. *Vehicular Communications*. 2020. Vol. 23. Article 100249. DOI: <https://doi.org/10.1016/j.vehcom.2020.100249>.
26. Aydin Y., Karabulut Kurt G., Ozdemir E., Yanikomeroglu H. Authentication and handover challenges and methods for drone swarms. *IEEE Journal of Radio Frequency Identification*. 2022. Vol. 6. P. 220–228. DOI: <https://doi.org/10.1109/JRFID.2022.3158392>.

27. Abro G. E. M. et al. A comprehensive review of next-gen UAV swarm robotics. *Intelligent Automation and Soft Computing*. 2025. Vol. 40. P. 99–123. DOI: <https://doi.org/10.32604/iasc.2025.060364>.
28. Yang J., Minturn D. B., Hady F. When poll is better than interrupt. *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST'12)*. San Jose, CA, USA, 2012. P. 25–31.
29. Harris B., Altiparmak N. When poll is more energy efficient than interrupt. *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '22)*. New York: ACM, 2022. P. 59–64. DOI: <https://doi.org/10.1145/3538643.3539747>.
30. Abeywardena D., Kodagoda S., Dissanayake G., Munasinghe R. Improved state estimation in quadrotor MAVs: a novel drift-free velocity estimator. *IEEE Robotics & Automation Magazine*. 2013. Vol. 20, No 4. P. 32–39. DOI: <https://doi.org/10.1109/MRA.2012.2225472>.
31. Heemels W. P. M. H., Johansson K. H., Tabuada P. An introduction to event-triggered and self-triggered control. *Proceedings of the 51st IEEE Conference on Decision and Control (CDC 2012)*. Maui, HI, USA, 2012. P. 3270–3285. DOI: <https://doi.org/10.1109/CDC.2012.6425820>.
32. Hamara A., Kilpatrick B., Baratta A., Kofink B., Freeman A. C. Low-Latency Scalable Streaming for Event-Based Vision. *arXiv preprint*. 2024. DOI: <https://doi.org/10.48550/arXiv.2412.07889>.
33. Bauersfeld L., Scaramuzza D. Low-Latency Event-Based Velocimetry for Quadrotor Control in a Narrow Pipe. *arXiv preprint*. 2025. arXiv:2507.15444.
34. Phadke A., Medrano F. A., Sekharan C. N., Chu T. Designing UAV swarm experiments: a simulator selection and experiment design process. *Sensors*. 2023. Vol. 23, No 17. Article 7359. DOI: <https://doi.org/10.3390/s23177359>.
35. Zhang X.-M., Han Q.-L., Ge X.-H. et al. An overview of recent advances in event-triggered control. *Science China Information Sciences*. 2025. Vol. 68, No 6. Article 161201. DOI: <https://doi.org/10.1007/s11432-024-4437-9>.

36. Ji L. et al. Consensus formation of multi-agent systems with obstacle avoidance via event-triggered impulsive control. *Journal of Intelligent & Robotic Systems*. 2023. Vol. 109. Article 61. DOI: <https://doi.org/10.1007/s10846-023-01987-z>.
37. Dou Y., Xing G., Ma A., Zhao G. A review of event-triggered consensus control in multi-agent systems. *Journal of Control and Decision*. 2025. Vol. 12, No 1. P. 1–23. DOI: <https://doi.org/10.1080/23307706.2024.2388551>.
38. Hassanalian M., Abdelkefi A. Classifications, applications, and design challenges of drones: a review. *Progress in Aerospace Sciences*. 2017. Vol. 91. P. 99–131. DOI: <https://doi.org/10.1016/j.paerosci.2017.04.003>.
39. Leonard N. E., Levin S. A. Collective intelligence as a public good. *Collective Intelligence*. 2022. Vol. 1, No 1. DOI: <https://doi.org/10.1177/26339137221083293>.
40. Mairaj A., Baba A. I., Javaid A. Y. Application specific drone simulators: recent advances and challenges. *Simulation Modelling Practice and Theory*. 2019. Vol. 94, No 4. P. 100–117.
41. Wilensky U., Rand W. An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo. *Cambridge: The MIT Press*, 2016. 482 p.
42. Jadbabaie A., Lin J., Morse A. S. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*. 2003. Vol. 48, No 6. P. 988–1001. DOI: <https://doi.org/10.1109/TAC.2003.812781>.
43. Kolaric P., Chen C., Dalal A., Lewis F. L. Consensus controller for multi-UAV navigation. *Control Theory and Technology*. 2018. Vol. 16, P. 110–121. DOI: <https://doi.org/10.1007/s11768-018-8013-5>.
44. Wang C., Li S., Xia W., Sun J., Xie G. Formation control for multiple agents with local measurements: continuous-time and sampled-data-based cases. 2019 *IEEE 58th Conference on Decision and Control (CDC)*. Nice, France, 2019. P. 1862–1867. DOI: <https://doi.org/10.1109/CDC40024.2019.9028870>.
45. Dias P. G. F., Silva M. C., Rocha Filho G. P., Vargas P. A., Cota L. P., Pessin G. Swarm robotics: a survey from a multi-tasking perspective. *Sensors*. 2021. Vol. 21, No 6. Article 2062. DOI: <https://doi.org/10.3390/s21062062>.

46. Zhao Z., Chen S., Ru L., Hu G., Wang W. A Quality Evaluation Method for Drone Swarm Command and Control Networks Based on Complex Network. *Drones*. 2025. Vol. 9, No 12. Article 839. DOI: <https://doi.org/10.3390/drones9120839>.
47. Xie Y., Han L., Dong X., Li Q., Ren Z. Bio-Inspired Adaptive Formation Tracking Control for Swarm Systems with Application to UAV Swarm Systems. *Neurocomputing*. 2021. Vol. 453. P. 272–285. DOI: <https://doi.org/10.1016/j.neucom.2021.05.015>.
48. Wang X., Zhao Z., Yi L., Ning Z., Guo L., Yu F. R., Guo S. A survey on security of UAV swarm networks: attacks and countermeasures. *ACM Computing Surveys*. 2024. Vol. 57, No 3. P. 1–37. DOI: <https://doi.org/10.1145/3703625>.
49. Bu Y., Yan Y., Yang Y. Advancement challenges in UAV swarm formation control: a comprehensive review. *Drones*. 2024. Vol. 8, No 7. Article 320. DOI: <https://doi.org/10.3390/drones8070320>.
50. Checker L. et al. Systematic review of multi-objective UAV swarm mission planning. *Drones*. 2025. Vol. 9, No 7. Article 509. DOI: <https://doi.org/10.3390/drones9070509>.
51. Schranz M., Umlauft M., Sende M., Elmenreich W. Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*. 2020. Vol. 7. Article 36. DOI: <https://doi.org/10.3389/frobt.2020.00036>.
52. Luthra M. Aerial robot swarms: a review. *IAES International Journal of Robotics and Automation*. 2023. Vol. 12, No 2. P. 137–145. DOI: <https://doi.org/10.11591/ijra.v12i2>.
53. Hildmann H., Kovacs E. Review: using unmanned aerial vehicles (UAVs) as mobile sensing platforms (MSPs) for disaster response, civil security and public safety. *Drones*. 2019. Vol. 3, No 3. Article 59. DOI: <https://doi.org/10.3390/drones3030059>.
54. Sánchez P., Casado R., Bermúdez A. Real-time collision-free navigation of multiple UAVs based on bounding boxes. *Electronics*. 2020. Vol. 9, No 10. Article 1632. DOI: <https://doi.org/10.3390/electronics9101632>.

55. Tahir A., Biling J., Haghbayan M.-H., Toivonen H. T., Plosila J. Swarms of unmanned aerial vehicles – a survey. *Journal of Industrial Information Integration*. 2019. Vol. 16. Article 100106. DOI: <https://doi.org/10.1016/j.jii.2019.100106>.
56. Phalapanyakoon K., Siripongwutikorn P. Route planning of unmanned aerial vehicles under recharging and mission time constraints. *International Journal of Mathematical Engineering and Management Sciences*. 2021. Vol. 6, No 5. P. 1439–1459. DOI: <https://doi.org/10.33889/IJMEMS.2021.6.5.087>.
57. Udugama B. Review on efficient strategies for coordinated motion and tracking in swarm robotics. *arXiv:2302.06360*. 2023. URL: <https://arxiv.org/abs/2302.06360>.
58. Li L. et al. Cooperative search for dynamic targets by multiple UAVs under communication data losses. *ISA Transactions*. 2021. Vol. 114, No 4. P. 230–241. DOI: <https://doi.org/10.1016/j.isatra.2020.12.055>.
59. Yang L. et al. Multi-UAV collaborative target search method in dynamic uncertain scenario. *Sensors*. 2024. Vol. 24, No 23. Article 7639. DOI: <https://doi.org/10.3390/s24237639>.
60. Zhang X., Zhao W., Liu C., Li J. Distributed Multi-Target Search and Surveillance Mission Planning for Unmanned Aerial Vehicles in Uncertain Environments. *Drones*. 2023. Vol. 7, No 6. Article 355. DOI: <https://doi.org/10.3390/drones7060355>.
61. Du Z. et al. Quantized Consensus Control for Multi-UAVs Based on Prescribed Performance under Communication Constraints. *International Journal of Control, Automation and Systems*. 2022. Vol. 20. P. 321–333. DOI: <https://doi.org/10.1007/s12555-021-0010-7>.
62. Yang L. et al. Event-triggered Control of Quadrotor UAVs Based on a Novel Sliding-Mode Surface. *International Journal of Control, Automation and Systems*. 2025. Vol. 23. P. 2569–2583. DOI: <https://doi.org/10.1007/s12555-025-0303-3>.
63. Vitale A., Renner A., Nauer C., Scaramuzza D., Sandamirskaya Y. Event-driven Vision and Control for UAVs on a Neuromorphic Chip. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2021. P. 103–109. DOI: <https://doi.org/10.1109/ICRA48506.2021.9560881>.

64. Gallego G., Delbruck T., Orchard G. et al. Event-Based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2022. Vol. 44, No 1. P. 154–180. DOI: <https://doi.org/10.1109/TPAMI.2020.3008413>.
65. Bertolaso A., Raeissi M. M., Farinelli A., Muradore R. Using Petri Net Plans for Modeling UAV-UGV Cooperative Landing. *Proceedings of the 22nd European Conference on Artificial Intelligence*. 2016. P. 1720–1721. DOI: <https://doi.org/10.3233/978-1-61499-672-9-1720>.
66. Alfeo A. L., Cimino M. G., De Francesco N., Lega M., Vaglini G. Design and simulation of the emergent behavior of small drones swarming for distributed target localization. *Journal of Computer Science*. 2018. Vol. 29. P. 19–33.
67. Phadke A., Medrano F. A., Sekharan C. N., Chu T. An Analysis of Trends in UAV Swarm Implementations in Current Research: Simulation versus Hardware. *Drone Systems and Applications*. 2024. Vol. 12. P. 1–10. DOI: <https://doi.org/10.1139/dsa-2023-0099>.
68. Zhao L., Chen B., Hu F. Research on Swarm Control Based on Complementary Collaboration of Unmanned Aerial Vehicle Swarms Under Complex Conditions. *Drones*. 2025. Vol. 9, No 2. Article 119. DOI: <https://doi.org/10.3390/drones9020119>.
69. Alam M. M., Moh S. Joint Topology Control and Routing in a UAV Swarm for Crowd Surveillance. *Journal of Network and Computer Applications*. 2022. Vol. 204. Article 103427. DOI: <https://doi.org/10.1016/j.jnca.2022.103427>.
70. Lim R. Y. H., Lim J. M.-Y., Lan B. L., Ho P. W. C., Ho N. S., Ooi T. W. M. UAV Swarm Communication Reliability Based on a Comprehensive SINR Model. *Vehicular Communications*. 2024. Vol. 47. Article 100781. DOI: <https://doi.org/10.1016/j.vehcom.2024.100781>.
71. Alqudsi Y., Makaraci M. UAV Swarms: Research, Challenges, and Future Directions. *Journal of Engineering and Applied Science*. 2025. Vol. 72, No 12. P. 1–20. DOI: <https://doi.org/10.1186/s44147-025-00582-3>.
72. Han X., Yu P., Lv M., Shi Y., Wang N. Event-driven Prescribed-time Tracking Control for Multiple UAVs with Flight State Constraints. *Machines*. 2025. Vol. 13, No 3. Article 192. DOI: <https://doi.org/10.3390/machines13030192>.

73. Gonçalves E. M. N., Machado R. A., Rodrigues B. C., Adamatti D. CPN4M: Testing Multi-Agent Systems under Organizational Model Moise+ Using Colored Petri Nets. *Applied Sciences*. 2022. Vol. 12, No 12. Article 5857. DOI: <https://doi.org/10.3390/app12125857>.
74. Boucherit A., Castro L. M., Khababa A., Hasan O. Petri Net and Rewriting Logic Based Formal Analysis of Multi-Agent Based Safety-Critical Systems. *Multiagent and Grid Systems*. 2020. Vol. 16, No 1. P. 47–66. DOI: <https://doi.org/10.3233/MGS-200320>.
75. Celaya J. R., Desrochers A. A., Graves R. J. Modeling and Analysis of Multi-Agent Systems Using Petri Nets. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. Montreal, Canada, 2007. P. 1439–1444. DOI: <https://doi.org/10.1109/ICSMC.2007.4413960>.
76. Kazymyr V., Pasko O., Usik A., Sydorenko D. New Paradigm of Model-Oriented Control in IoT. *Information and Software Technologies: Proceedings of ICIST 2019. Communications in Computer and Information Science*. Cham: Springer, 2019. Vol. 1078. P. 605–614. DOI: https://doi.org/10.1007/978-3-030-30275-7_46.
77. Qin H. et al. Petri-Net Based Multi-Objective Optimization in Multi-UAV Aided Wireless Powered Communication Networks. *Remote Sensing*. 2021. Vol. 13, No 13. Article 2611. DOI: <https://doi.org/10.3390/rs13132611>.
78. Ge X., Han Q.-L., Dong H. Dynamic Event-Triggered Control and Estimation: A Survey. *International Journal of Automation and Computing*. 2021. Vol. 18. P. 1294–1316. DOI: <https://doi.org/10.1007/s11633-021-1306-z>.
79. Grife A., Eickhoff J., Trimpe S. Event-Triggered and Distributed Model Predictive Control for Guaranteed Collision Avoidance in UAV Swarms. *IFAC-PapersOnLine*. 2022. Vol. 55, No 13. P. 79–84. DOI: <https://doi.org/10.1016/j.ifacol.2022.07.239>.
80. Сапатий П. С. Управління розподіленими системами за допомогою патернів просторового захоплення. *Математичні машини і системи*. 2023. № 4. С. 11–25.

81. Calik S. K., Kugu E., Birtane S., Sahingoz O. K. A Multi-Agent Solution for UAV Path Planning Problem with NetLogo. *International Journal of Applied Engineering Research*. 2016. Vol. 11, No 15. P. 8397–8401.
82. Zhu Z., Yi W., Ruifeng F., Liya L., Meng Q. Modeling on Anti-UAV System-of-Systems Combat OODA Loop Based on NetLogo. *Journal of System Simulation*. 2021. Vol. 33, No 8. P. 1791–1800.
83. Ling H., Luo H., Chen H., Bai L., Zhu T., Wang Y. Modelling and Simulation of Distributed UAV Swarm Cooperative Planning and Perception. *International Journal of Aerospace Engineering*. 2021. Vol. 2021. P. 1–11. DOI: <https://doi.org/10.1155/2021/9977262>.
84. Soria E., Schiano F., Floreano D. SwarmLab: A Matlab Drone Swarm Simulator. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020. P. 8005–8011.
85. Zhang J., Yan J. A Novel Control Approach for Flight-Stability of Fixed-Wing UAV Formation with Wind Field. *IEEE Systems Journal*. 2021. Vol. 15, No 2. P. 2098–2108. DOI: <https://doi.org/10.1109/JSYST.2020.3002809>.
86. Bonnefond A. Flocking Models Based on Local Communications: *From Theory to Simulations : doctoral thesis* / Institut National des Sciences Appliquées de Lyon. Lyon, 2023. URL: <https://theses.insa-lyon.fr/publication/2023ISAL0040/these.pdf>
87. Lizzio F. F., Capello E., Guglieri G. A Review of Consensus-Based Multi-Agent UAV Applications. *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*. 2021. P. 1548–1557. DOI: <https://doi.org/10.1109/ICUAS51884.2021.9476858>.
88. Liu H., Liu H. H. T., Chi C., Zhai Y., Zhan X. Navigation Information Augmented Artificial Potential Field Algorithm for Collision Avoidance in UAV Formation Flight. *Aerospace Systems*. 2020. Vol. 3. P. 229–241.
89. Zhao H., Wu S. A Method to Estimate Relative Position and Attitude of Cooperative UAVs Based on Monocular Vision. *Proceedings of the IEEE CSAA Guidance, Navigation and Control Conference (GNCC)*. Xiamen, China, 2018. P. 1–6. DOI: <https://doi.org/10.1109/GNCC42960.2018.9018876>.

90. Yadawad R., Kulkarni U. P. MVC Architecture and C2C File-Transfer Protocol for Smart Home IoT Appliance Control. *SN Computer Science*. 2023. Vol. 4. Article 369. DOI: <https://doi.org/10.1007/s41870-023-01349-w>.
91. Khan S., Sharma N. Design and Development of an IoT-Based Web Application for an Intelligent Remote SCADA System. *International Journal of Advanced Computer Science and Applications*. 2018. Vol. 9, No 3. P. 41–49.
92. Xu P., Liu J., Sun X., Chen H., Chen Y. Distributed Consensus Control Research of Unmanned Aerial Vehicle (UAV) Swarms Based on Lennard-Jones Potential. *Proceedings of the 3rd International Conference on Machine Learning, Cloud Computing and Intelligent Mining (MLCCIM 2024)*. *Lecture Notes in Electrical Engineering*. Singapore: Springer, 2025. Vol. 1327. P. 154–165. DOI: https://doi.org/10.1007/978-981-96-1694-7_14.
93. Cumino P. et al. Cooperative UAV Scheme for Enhancing Video Transmission and Global Network Energy Efficiency. *Sensors*. 2018. Vol. 18. Article 4155.
94. Cimino M. G., Lega M., Monaco M., Vaglini G. Adaptive Exploration of a UAVs Swarm for Distributed Targets Detection and Tracking. *Proceedings of the International Conference on Pattern Recognition Applications and Methods (ICPRAM)*. Prague, Czech Republic, 2019. P. 837–844.
95. Zhao Z. et al. Software-Defined Unmanned Aerial Vehicles Networking for Video Dissemination Services. *Ad Hoc Networks*. 2019. Vol. 83. P. 68–77.
96. Martti L., Bill H. Mini-Drones Swarms and Their Potential in Conflict Situations. *Proceedings of the 15th International Conference on Cyber Warfare and Security*. Norfolk, Virginia, USA, 2020. P. 326–334.
97. Saxena V. K. Swarm Drones: Attacker's Delight, Defender's Nightmare. *Status Report*. New Delhi: Vivekananda International Foundation, 2020. 31 p.
98. Alsammak I. L. H., Mahmoud M. A., Gunasekaran S. S., Ahmed A. N., AlKilabi M. Nature-Inspired Drone Swarming for Wildfires Suppression Considering Distributed Fire Spots and Energy Consumption. *IEEE Access*. 2023. Vol. 11. P. 50962–50983.

99. High Level Architecture (HLA), Release 3.0. AIOTI WG03 – IoT Standardisation. 2017. 40 p. URL: <https://aioti.eu/wp-content/uploads/2017/06/AIOTI-HLA-R3-June-2017.pdf>.
100. Kazymyr V. V., Posadska A. S., Sysa D. M. Cloud Simulation Environment Based on HLA. *Радіоелектронні і комп'ютерні системи*. 2016. № 5(79). С. 20–25.
101. Dashkevych A., Vorontsova D., Rosokha S. An Approach for the Determination of Drone Positions Set with Maximal Terrain Visibility. *ICT in Education, Research and Industrial Applications (ICTERI 2021)*. CEUR Workshop Proceedings. 2021. Vol. 3013. P. 64–73.
102. Macal C. M., North M. J. Tutorial on Agent Based Modelling and Simulation. *Journal of Simulation*. 2010. Vol. 4. P. 151–162. DOI: <https://doi.org/10.1057/jos.2010.3>.
103. Shmelova T., Sterenharz A., Burlaka O. Optimization of Flows and Flexible Redistribution of Autonomous UAV Routes in Multilevel Airspace. *ICT in Education, Research and Industrial Applications Workshops (ICTERI 2019)*. CEUR Workshop Proceedings. 2019. Vol. 2393. P. 704–715.
104. Sheng Z., Cui L., Nasir A. A., Wang R., Fang Y. URLLC Oriented Secure Communication for UAV Relay Assisted Networks. *Physical Communication*. 2023. Vol. 59. Article 102063. DOI: <https://doi.org/10.1016/j.phycom.2023.102063>.
105. Estivill-Castro V., Hexel R., Lusty S. Continuous Integration for Testing Full Robotic Behaviours in a GUI-Stripped Simulation. *CEUR Workshop Proceedings*. 2018. Vol. 2245. Paper 3. P. 1–12.
106. Сушло М. Ю., Олексієнко П. Д., Сігута А. В., Казимир В. В. Напівнатурне моделювання процесів керування роєм дронів. *Новітні технології сучасного суспільства: матеріали V Міжнар. наук.-практ. конф.*, м. Чернігів, 12 груд. 2024 р. Чернігів, 2024. С. 183–184.
107. Документація до Holybro Pixhawk 6C. URL: https://docs.px4.io/main/uk/flight_controller/pixhawk6c (дата звернення: 09.01.2026).

ДОДАТКИ

Додаток А

Акти впровадження результатів дисертаційного дослідження

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ		MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»	тел. +38(0462) 665-103; факс +38(0462) 665-105 E-mail: csu@stu.cn.ua www.stu.cn.ua Код ЄДРПОУ 05460798	CHERNIHIV POLYTECHNIC NATIONAL UNIVERSITY
вул. Шевченка, 95, Чернігів, 14035, Україна		95, Shevchenko str., Chernihiv, 14035, Ukraine

01.06.2025 № 227/227

ДОВІДКА ПРО ВПРОВАДЖЕННЯ

результатів дисертаційної роботи Олексієнко Петра Дмитровича на тему «Моделі та метод програмного керування роєм дронів на основі групового інтелекту», представленої на здобуття вченого ступеня доктора філософії зі спеціальності 122 - Комп'ютерні науки.

Результати дисертаційного дослідження Олексієнко П. Д., а саме:

- мультиагентна модель поведінки рою мультикоптерних дронів при виконанні місії із захисту інфраструктурних об'єктів, яка відтворює роль групового інтелекту в процесі відбиття нападу повітряних цілей з урахуванням можливостей підсистем аудіо- та відео- спостереження та розпізнавання;
- формальна CEN-модель алгоритму керування агентом у складі рою дронів, яка вбудовується безпосередньо у контур управління дроном та виконує роль тривірневої програми керування з подієво-орієнтовним реагуванням на зміну ситуації;
- метод керування роєм дронів, що базується на MVC-підході проектування програмних застосунків в інтерпретації до процесів керування і забезпечує самоорганізацію поведінки рою дронів в реальному часі за рахунок використання групового інтелекту;
- програмна реалізація розроблених алгоритмів в діючих макетах дронів та їх верифікація в ході модельних та натурних експериментів – були впроваджені в науково-дослідній роботі (прикладне дослідження) "Мультиагентна система захисту об'єктів критичної інфраструктури на основі рою мультикоптерних дронів" (номер державної реєстрації 0123U101819), що виконувалась в Національному університеті «Чернігівська політехніка» в період з 2023 по 2025 роки, що дозволило отримувати попередню оцінку ефективності проектних рішень та забезпечило прийняття групових рішень із зменшенням часу реакції до 100 разів у порівнянні із стандартним циклічним підходом.

Проректор з наукової роботи



[Handwritten signature]

А.Л. Приступа



МІНІСТЕРСТВО ОБОРОНИ
УКРАЇНИ
ДЕРЖАВНИЙ
НАУКОВО-ДОСЛІДНИЙ
ІНСТИТУТ ВИПРОБУВАНЬ
І СЕРТИФІКАЦІЇ ОЗБРОЄННЯ
ТА ВІЙСЬКОВОЇ ТЕХНІКИ
до » травня 2026 р.
№ 625/ 3163
18000, м. Черкаси

ДОВІДКА

про використання результатів дисертаційної роботи Олексієнка Петра Дмитровича на тему
«Моделі та метод програмного керування роєм дронів на основі групового інтелекту»,
представленої на здобуття ступеня доктора філософії
зі спеціальності 122 – Комп'ютерні науки

Окремі результати, отримані в ході проведеного Олексієнком П.Д. дисертаційного дослідження, були використані під час проведення натурних досліджень та відпрацювання програм і методик випробувань безпілотних авіаційних комплексів у Державному науково-дослідному інституті випробувань і сертифікації озброєння та військової техніки. Зокрема, практичне застосування знайшли такі результати:

- 1) метод керування роєм дронів, що базується на MVC-підході проектування програмних застосунків в інтерпретації до процесів керування і забезпечує самоорганізацію поведінки рою дронів в реальному часі за рахунок використання групового інтелекту;
- 2) формальну CEN-модель алгоритму керування агентом у складі рою дронів, яка вбудовується безпосередньо у контур управління дроном та виконує роль тривірневої програми керування з подієво-орієнтовним реагуванням на зміну ситуації;
- 3) мультиагентну модель поведінки рою мультикоптерних дронів при виконанні місії із захисту інфраструктурних об'єктів, яка відтворює роль групового інтелекту в процесі відбиття нападу повітряних цілей з урахуванням можливостей підсистем аудіо- та відеоспостереження та розпізнавання.

Використання зазначених матеріалів дисертації дозволило удосконалити програми та методики випробувань безпілотних авіаційних комплексів групового застосування, сформувати об'єктивні критерії оцінювання їхньої здатності до самоорганізації та досягти скорочення часу реакції під час натурних досліджень. Також вказана реалізація дозволила отримувати попередні теоретичні оцінки ефективності проектних рішень розробників перед проведенням льотних експериментів, що загалом підвищило якість планування натурних експериментів.

Головний науковий співробітник науково-дослідного управління випробувань безпілотних та автоматизованих систем Державного науково-дослідного інституту випробувань і сертифікації озброєння та військової техніки
доктор технічних наук, професор

Володимир РУДНИЦЬКИЙ

Підпис Рудницького В.М. згідно:
Учений секретар Державного науково-дослідного інституту випробувань і сертифікації озброєння та військової техніки, кандидат технічних наук, старший науковий співробітник

Олександр БУРСАЛА

ВІДКРИТА ІНФОРМАЦІЯ



**НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОБЛЕМ МАТЕМАТИЧНИХ МАШИН І СИСТЕМ
ІПММС**

Проспект Академіка Глушкова, 42, м. Київ, 03187, Україна, тел. 0445262497,
Факс: 0445266457, ел. пошта: ipmms@immsp.kiev.ua, код ЄДРПОУ 05417503

05.06.2026 148/ 10-33

На № _____ від _____

ДОВІДКА ПРО ВПРОВАДЖЕННЯ

результатів дисертаційної роботи Олексієнка Петра Дмитровича на тему
«Моделі та метод програмного керування роями дронів на основі
групового інтелекту», представленої на здобуття наукового ступеня
доктора філософії за спеціальністю 122 - Комп'ютерні науки.

Основні положення та результати дисертаційного дослідження
Олексієнка Петра Дмитровича на тему «Моделі та метод програмного
керування роями дронів на основі групового інтелекту» використовуються
в Інституті проблем математичних систем і машин Національної академії
наук України при викладанні дисциплін «Методологія, організація та
технологія наукових досліджень», «Проектування систем штучного
інтелекту» та «Ймовірнісне моделювання об'єктів і процесів» в процесі
навчання аспірантів за освітньо-науковою програмою 122 – комп'ютерні
науки.

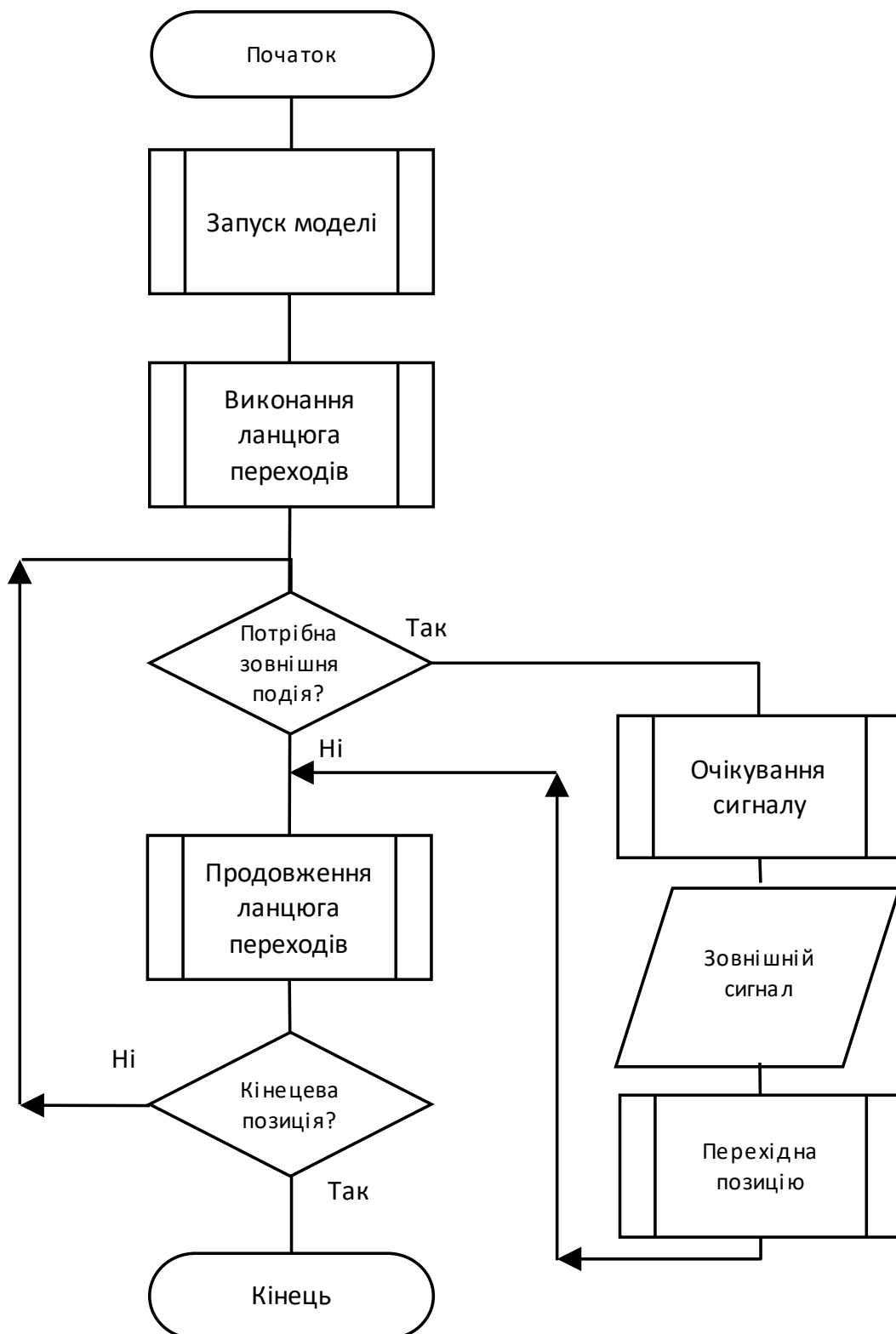
В.о. директора
д.ф.-м.н., професор



Віталій КЛИМЕНКО

Додаток Б

Фрагмент логіки виконання очікування сигналу



Додаток В

Ключові фрагменти програмного коду системи Е-мережі

1. Клас `Position` – представлення позиції Е-мережі

```

```python
from typing import Dict, Any
import time

class Position:
 def __init__(self, name: str, data: Dict[str, Any] = None):
 self.name = name
 self.token = False # маркер активності позиції
 self.data: Dict[str, Any] = {} if data is None else data.copy()
 self.entry_count = 0 # лічильник кількості активацій
 self.total_time_active = 0 # сумарний час у активному стані (секунди)
 self.start_time = None # мітка часу останньої активації

 def set_token(self, value: bool):
 """Встановлює або знімає токен, автоматично оновлюючи лічильники часу."""
 if value:
 if self.token is False: # нова активація – фіксуємо початок
 self.start_time = time.time()
 self.entry_count += 1
 else:
 if self.token and self.start_time:
 self.total_time_active += time.time() - self.start_time
 self.start_time = None
 self.token = value

 def update_data(self, data: Dict[str, Any]):
 self.data.update(data)

```

```

def get_data_copy(self) -> Dict[str, Any]:
 return self.data.copy()
def get_average_time_active(self) -> float:
 """Повертає середній час перебування позиції в активному стані."""
 return self.total_time_active / self.entry_count if self.entry_count > 0 else 0
'''

```

## 2. Клас `Statistics` – збір метрик виконання

```

'''python
from collections import defaultdict
import statistics
class Statistics:
 def __init__(self):
 self.transition_entries = defaultdict(int) # лічильники спрацювань переходів
 self.transition_times = defaultdict(list) # часи виконання кожного переходу
 self.position_entries = defaultdict(int) # лічильники входів у позиції
 def log_transition_entry(self, transition_name: str, start_time: float):
 """Фіксує одне спрацювання переходу та час його виконання."""
 self.transition_entries[transition_name] += 1
 self.transition_times[transition_name].append(time.time() - start_time)
 def get_statistics(self, positions: Dict[str, "Position"]) -> dict:
 """Формує зведений звіт по всіх переходах і позиціях моделі."""
 stats = {"transitions": {}, "positions": {}}
 for name, times in self.transition_times.items():
 stats["transitions"][name] = {
 "entries": self.transition_entries[name],
 "avg_time": statistics.mean(times) if times else 0,
 "min_time": min(times) if times else 0,
```

```

 "max_time": max(times) if times else 0,
 "variance": statistics.variance(times) if len(times) > 1 else 0,
 }
 for name, pos in positions.items():
 stats["positions"][name] = {
 "entries": pos.entry_count,
 "avg_time": pos.get_average_time_active(),
 "total_time": pos.total_time_active,
 }
 return stats
'''
'''

3. Клас `E_Network` – виконавче середовище E-мережі
'''python
from concurrent.futures import ThreadPoolExecutor
class E_Network:
 def __init__(self, model_id):
 self.model_id = model_id
 self.positions: Dict[str, Position] = {}
 self.transitions: Dict[str, TransitionBase] = {}
 self.start_position: str = ""
 self.end_position: str = ""
 self.finished = False
 self.statistics = Statistics()
 self.executor = ThreadPoolExecutor(max_workers=10) # пул потоків для
паралельних гілок
— Конфігурація моделі
 def add_position(self, name: str, data: Dict[str, Any] = None):
 self.positions[name] = Position(name, data)

```

```

def add_transition(self, transition: "TransitionBase"):
 self.transitions[transition.name] = transition
def set_start_position(self, name: str):
 if name in self.positions:
 self.start_position = name
def set_end_position(self, name: str):
 if name in self.positions:
 self.end_position = name
— Реактивний зовнішній тригер
def set_token_to_position(self, name: str, need_process_now: bool = False):
 """
 Встановлює токен у довільну позицію ззовні (зовнішня подія).
 При need_process_now=True негайно запускає обробку – O(1) реакція.
 """
 if name in self.positions:
 self.positions[name].set_token(True)
 if need_process_now:
 self.process_position(name)
— Виконання переходу
def _execute_transition_iterative(self, transition: "TransitionBase"):
 """
 Виконує один перехід і повертає наступні активні позиції.
 Для F-переходу (Fork) запускає кожну гілку у окремому потоці пулу.
 """
 next_positions = transition.execute(self.positions, self.statistics)
 if isinstance(next_positions, list):
 if isinstance(transition, FTransition):
 # Fork: паралельний запуск усіх вихідних гілок
 futures = []
 for pos in next_positions:

```

```

 self.positions[pos].set_token(True)
 futures.append(self.executor.submit(self.process_position, pos))
 for future in futures:
 future.result() # бар'єр – чекаємо завершення всіх гілок
 return None
else:
 for pos in next_positions:
 self.positions[pos].set_token(True)
 return next_positions
elif isinstance(next_positions, str):
 self.positions[next_positions].set_token(True)
 if next_positions == self.end_position:
 self.finished = True # досягнуто кінцевої позиції
 return next_positions
return None

— Ітеративний обхід мережі
def process_position(self, position_name: str):
 """
 Ітеративний (не рекурсивний) обхід Е-мережі починаючи з position_name.
 Використовує явну чергу замість стеку викликів, що уникає переповнення
 стеку на моделях великої глибини.
 """
 positions_to_process = [position_name]
 while positions_to_process and not self.finished:
 current = positions_to_process.pop(0)
 for transition in self.transitions.values():
 if current in transition.input_positions:
 next_pos = self._execute_transition_iterative(transition)
 if next_pos:
 if isinstance(next_pos, list):

```

```

 positions_to_process.extend(next_pos)
 else:
 positions_to_process.append(next_pos)
— Запуск моделі —
def run(self):
 """Запускає виконання моделі і блокується до досягнення кінцевої позиції."""
 self.positions[self.start_position].set_token(True)
 self.process_position(self.start_position)
 while not self.finished:
 time.sleep(0.5) # очікування зовнішніх подій
 ...

4. Абстрактний базовий клас `TransitionBase`
```python
from abc import ABC, abstractmethod
from typing import Callable, List, Union

class TransitionBase(ABC):
    def __init__(self, name: str,
                  input_positions: Union[str, List[str]],
                  output_positions: Union[str, List[str]],
                  action: Callable = None):
        self.name = name
        self.action = action
        # нормалізація до списку незалежно від вхідного типу
        self.input_positions = (
            [input_positions] if isinstance(input_positions, str) else input_positions
        )
        self.output_positions = (
            [output_positions] if isinstance(output_positions, str) else output_positions

```

)

```
def execute(self, positions, statistics) -> Union[str, List[str], None]:
    """Запускає перехід і фіксує метрику часу у об'єкті статистики."""
    start_time = time.time()
    result = self._execute_logic(positions)
    if result is not None:
        statistics.log_transition_entry(self.name, start_time)
    return result
```

@abstractmethod

```
def _execute_logic(self, positions) -> Union[str, List[str], None]:
    """Конкретна логіка переходу, що реалізується у підкласах."""
    pass

def transfer_data(self, data: Dict[str, Any], positions):
    """Копіює словник даних до всіх вихідних позицій."""
    for output_name in self.output_positions:
        positions[output_name].update_data(data.copy())
```

...

5. Лінійний перехід `TTransition` та умовний перехід `XTransition`

```python

```
class TTransition(TransitionBase):
 """T-перехід: лінійне проходження з необов'язковою дією та затримкою."""
 def __init__(self, name: str, input_position: str, output_position: str,
 action: Callable = None, delay: int = 0):
 super().__init__(name, [input_position], [output_position], action)
 self.delay = delay
 def _execute_logic(self, positions):
 if self.delay > 0:
 time.sleep(self.delay) # моделювання часу виконання операції
```

```

input_data = positions[self.input_positions[0]].get_data_copy()
if self.action:
 self.action(input_data)
self.transfer_data(input_data, positions)
positions[self.input_positions[0]].set_token(False)
return self.output_positions[0]

```

```
class XTransition(TransitionBase):
```

```
 """X-перехід: умовне розгалуження – активується лише одна вихідна позиція."""
```

```
 def __init__(self, name: str, input_position: str, output_positions: List[str],
```

```
 action: Callable = None, condition: Callable[[], int] = None):
```

```
 super().__init__(name, [input_position], output_positions, action)
```

```
 self.condition = condition if condition else lambda: 0 # за замовч. – перший вихід
```

```
 def _execute_logic(self, positions):
```

```
 input_data = positions[self.input_positions[0]].get_data_copy()
```

```
 if self.action:
```

```
 self.action(input_data)
```

```
 chosen_index = self.condition() # умова повертає індекс вихідної позиції
```

```
 if 0 <= chosen_index < len(self.output_positions):
```

```
 selected = self.output_positions[chosen_index]
```

```
 positions[selected].update_data(input_data)
```

```
 positions[self.input_positions[0]].set_token(False)
```

```
 return selected
```

```
 return None # умова не виконана – перехід не відбувається
```

```
...
```

```

```

```
6. Паралельні переходи `FTransition` та `JTransition` (Fork / Join)
```

```
```python
```

```
import threading
```

```
class FTransition(TransitionBase):
```

```
    """F-перехід (Fork): один вхід активує всі виходи паралельно."""
```

```

def __init__(self, name: str, input_position: str, output_positions: List[str]):
    super().__init__(name, [input_position], output_positions)
def _execute_logic(self, positions):
    input_data = positions[self.input_positions[0]].get_data_copy()
    self.transfer_data(input_data, positions) # дані копіюються до кожної гілки
    positions[self.input_positions[0]].set_token(False)
    return self.output_positions # список → сигнал для паралельного запуску
class JTransition(TransitionBase):
    """J-перехід (Join): усі входи мають містити токен – тільки тоді перехід
спрацьовує."""
    def __init__(self, name: str, input_positions: List[str], output_position: str,
        action: Callable = None,
        data_merge_function: Callable[[Dict[str, Dict[str, Any]]], Dict[str, Any]] =
None):
        super().__init__(name, input_positions, [output_position], action)
        self.data_merge_function = data_merge_function
        self.lock = threading.Lock() # захист від гонки між паралельними потоками
    def _execute_logic(self, positions):
        all_tokens = [positions[pos].token for pos in self.input_positions]
        if all(all_tokens): # синхронізаційний бар'єр
            with self.lock: # атомарна секція – лише один потік входить
                data_dict = {
                    pos: positions[pos].get_data_copy()
                    for pos in self.input_positions
                }
                # злиття даних із усіх гілок або вибір даних першої гілки
                merged_data = (
                    self.data_merge_function(data_dict)
                    if self.data_merge_function
                    else data_dict[self.input_positions[0]]

```



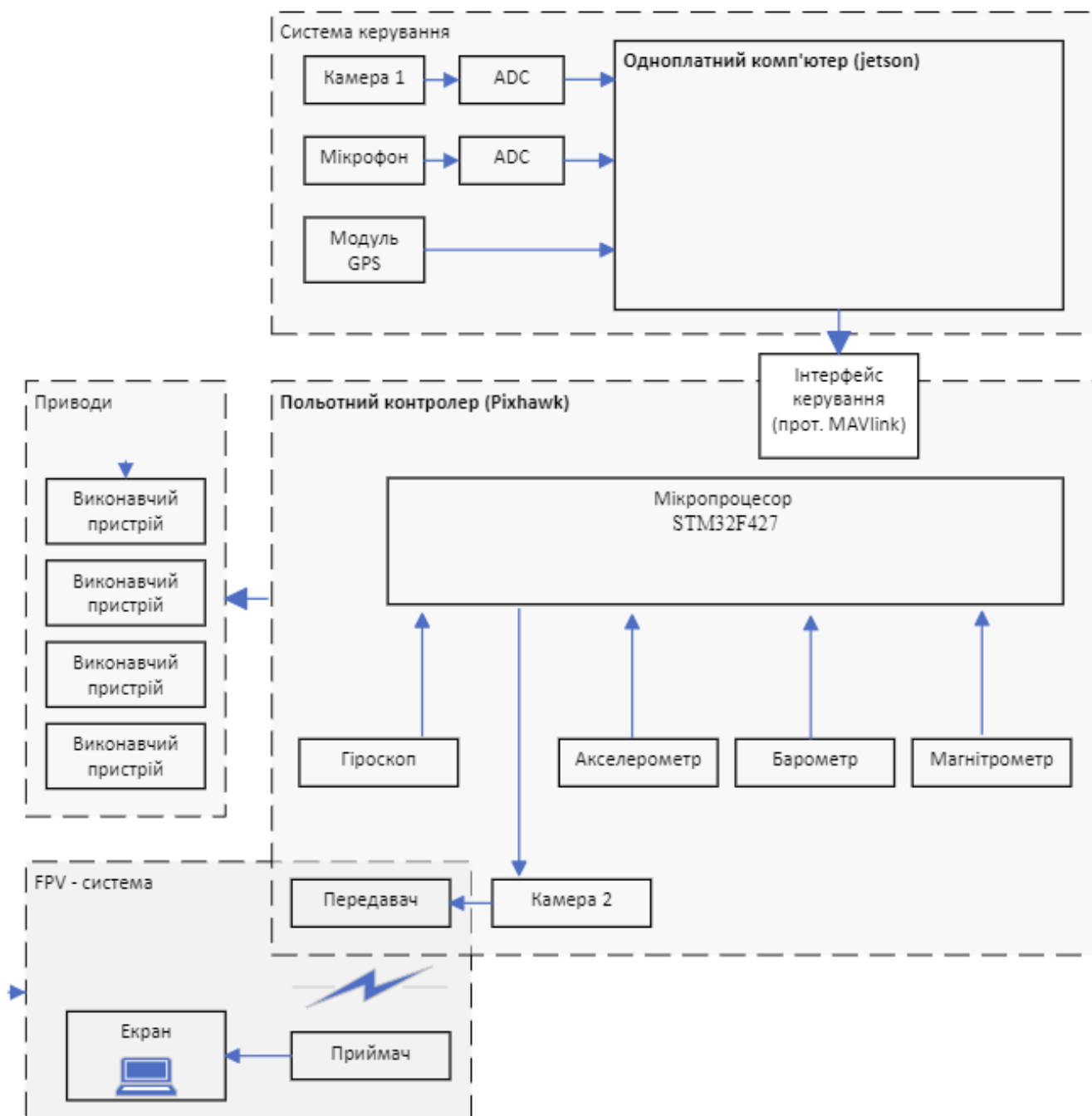
```

    if self.action:
        self.action(selected_data)
        positions[self.output_positions[0]].update_data(selected_data)
        positions[self.output_positions[0]].set_token(True)
    @abstractmethod
    def dequeue(self) -> Dict[str, Any]:
        pass
class QFTransition(QueueTransitionBase):
    """QF-перехід: черга FIFO – перший прийшов, перший вийшов."""
    def dequeue(self) -> Dict[str, Any]:
        if self.priority:                # пріоритетний режим: вибираємо з найвищим
priority
            selected = max(self.queue, key=lambda d: d.get("priority", 0))
            self.queue.remove(selected)
            return selected
        return self.queue.popleft()      # стандартний режим: з початку черги
class QLTransition(QueueTransitionBase):
    """QL-перехід: стек LIFO – останній прийшов, перший вийшов."""
    def dequeue(self) -> Dict[str, Any]:
        if self.priority:
            selected = max(self.queue, key=lambda d: d.get("priority", 0))
            self.queue.remove(selected)
            return selected
        return self.queue.pop()          # стандартний режим: з кінця черги
...

```

Додаток Г

Структура системи керування дроном у складі мультиагентної системи



Додаток Д

Алгоритм подієво-орієнтованої взаємодії агента рою при розпізнаванні цілі

Вхідні дані:

S_i – поточний стан агента a_i ;

Z_i – локальні сенсорні дані агента;

M_i – повідомлення від інших агентів;

T – множина виявлених цілей;

R_i – роль агента у рої.

Вихідні дані:

D_i – дія агента: моніторинг, супровід, перехоплення або страхування.

1. Ініціалізувати агента a_i .
2. Перейти у режим моніторингу середовища.
3. Поки місія не завершена виконувати:
 - 3.1. Отримати локальні сенсорні дані Z_i .
 - 3.2. Оновити поточний стан агента S_i .
 - 3.3. Виконати аналіз середовища на наявність цілі.
 - 3.4. Якщо ціль не розпізнано, то:
 - продовжити моніторинг;
 - перейти до кроку 3.1.
 - 3.5. Якщо ціль розпізнано, то:
 - сформувавши повідомлення про ціль:
 - координати цілі;
 - напрямок руху;
 - час виявлення;
 - ідентифікатор агента-джерела;
 - передати повідомлення іншим агентам рою.
 - 3.6. Продовжувати локальний моніторинг середовища.
 - 3.7. Очікувати надходження уточнювальних даних від інших агентів.
 - 3.8. Якщо уточнювальні дані не отримано, то:
 - оновлювати локальну оцінку параметрів цілі;
 - продовжити моніторинг;
 - перейти до кроку 3.6.
 - 3.9. Якщо уточнювальні дані отримано, то:
 - об'єднати локальні та зовнішні дані про ціль;
 - розрахувати параметри цілі;
 - визначити, якому агенту доцільно виконувати перехоплення.
 - 3.10. Якщо ціль призначена агенту a_i , то:
 - встановити роль $R_i = \text{"перехоплювач"};$
 - сформувавши траєкторію перехоплення;
 - перейти до виконання маневру перехоплення.
 - 3.11. Інакше:
 - встановити роль $R_i = \text{"страхування"} \text{ або } \text{"спостереження"};$
 - продовжити моніторинг цілі та середовища.
4. Завершити виконання алгоритму після завершення місії.

Додаток Е

Список публікацій здобувача за темою дисертації

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Kazymyr V., Oleksiienko P., Chornonoh O., Rohovenko A. Modeling of Defensive Drone Swarms with NetLogo. *Mathematical Modeling and Simulation of Systems (MODS 2023). Lecture Notes in Networks and Systems*. Cham: Springer, 2024. Vol. 1091. P. 377–387. DOI: https://doi.org/10.1007/978-3-031-67348-1_28. Базис: Scopus, WoS.
2. Олексієнко П. Д., Казимир В. В. Вбудовані моделі алгоритму керування роєм БПЛА. *Математичні машини і системи*. 2025. № 3–4. С. 90–100. DOI: <https://doi.org/10.34121/1028-9763-2025-3-4-90-100>. Базис: CrossRef, Google Scholar.
3. Олексієнко П. Д., Казимир В. В. Подієво-орієнтований підхід в реалізації вбудованих систем керування роєм БПЛА. *Технічні науки та технології*. 2025. № 3 (41). С. 225–236. DOI: [https://doi.org/10.25140/2411-5363-2025-3\(41\)-225-236](https://doi.org/10.25140/2411-5363-2025-3(41)-225-236). Базис: CrossRef, Google Scholar.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

4. Suslo M., Kazymyr V., Oleksiienko P., Kagitin D. Semi-Physical Modeling of Drone Swarm Control Processes. *15th International Conference on Advanced Computer Information Technologies (ACIT): conference proceedings, Šibenik, Croatia, 17–19 September 2025*. Šibenik, 2025. P. 102–108. DOI: <https://doi.org/10.1109/ACIT65614.2025.11185750>.
5. Kahitin D., Kazymyr V., Suslo M., Yatchenko Y., Oleksiienko P. Drone Positioning in a Specified Location Using Visual Data Under GPS-Denied Conditions. *IEEE 13th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS): conference proceedings, Gliwice, Poland, 4–6 September 2025*. Gliwice, 2025. P. 318–323. DOI: <https://doi.org/10.1109/IDAACS68557.2025.11322385>.

Наукові праці, які додатково відображають наукові результати дисертації:

6. Суло М. Ю., Олексієнко П. Д., Сігута А. В., Казимир В. В. Напів натурне моделювання процесів керування роєм дронів. *Новітні технології сучасного суспільства (HTCC-2024)*: тези доп. V Міжнар. наук.-практ. конф., м. Чернігів, 12 груд. 2024 р. Чернігів, 2024. С. 183–184. URL: https://inel.stu.cn.ua/ntss/NTSS_2024_zbirnyk.pdf.