Міністерство освіти і науки України
Національний аерокосмічний університет
«Харківський авіаційний інститут»

Кваліфікаційна наукова
праця на правах рукопису

ЛЮБІМОВ ОЛЕКСАНДР ВІКТОРОВИЧ

УДК 004.415.2:629.783

ДИСЕРТАЦІЯ
АДАПТИВНІ МЕТОДИ, АЛГОРИТМИ ТА ПРОГРАМНІ ЗАСОБИ БАЛАНСУВАННЯ РЕАКТИВНОСТІ ТА ЕНЕРГОСПОЖИВАННЯ БОРТОВИХ ОБЧИСЛЮВАЧІВ НАНОСУПУТНИКІВ CUBESAT

Спеціальність 121 Інженерія програмного забезпечення
Галузь знань 12 Інформаційні технології

Подається на здобуття наукового ступеня
доктора філософії

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

_____ О.В. Любімов

Науковий керівник
Туркін Ігор Борисович, завідувач
кафедри інженерії програмного
забезпечення, доктор технічних наук,
професор

Харків – 2026

АНОТАЦІЯ

Любімов Олександр Вікторович. Адаптивні методи, алгоритми та програмні засоби балансування реактивності та енергоспоживання бортових обчислювачів наносупутників класу CubeSat – кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 «Інженерія програмного забезпечення». Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, 2026.

Дисертаційну роботу присвячено розробленню адаптивних методів, алгоритмів та програмних засобів балансування реактивності й енергоспоживання бортових обчислювачів наносупутників класу CubeSat та безпілотних літальних апаратів.

Об’єктом дослідження є процеси балансування та керування реактивністю бортового програмного забезпечення і енергоспоживанням бортових обчислювачів наносупутників класу CubeSat та безпілотних літальних апаратів.

Предметом дослідження є математичні моделі, методи, алгоритми та програмні засоби адаптивного регулювання режимів роботи бортових обчислювачів з урахуванням обмежень реального часу, обчислювальної продуктивності та енергетичного бюджету.

У дисертаційній роботі виконано систематизований аналіз сучасного стану технологій створення наносупутників класу CubeSat, умов їх експлуатації, стандартів побудови, а також застосування комерційно доступних апаратних і програмних компонентів. Показано, що стрімке зростання кількості місій CubeSat супроводжується значною часткою невдалих запусків і втрати апаратів, істотна частина яких зумовлена неузгодженістю алгоритмів керування енергоспоживанням і реактивністю бортового програмного забезпечення.

На основі систематизованого огляду наукових джерел встановлено, що енергоефективність, обчислювальна продуктивність і реактивність бортових обчислювачів є взаємопов’язаними характеристиками, які визначають надійність та успішність виконання місії. Водночас у наявних підходах ці характеристики

здебільшого розглядаються ізольовано. Обґрунтовано доцільність застосування формалізованих комплексних моделей для їх узгодженого аналізу та оптимізації.

У межах дослідження проведено порівняльний аналіз архітектурних підходів і принципів побудови бортових обчислювальних систем наносупутників класу CubeSat та безпілотних літальних апаратів. Виявлено суттєву концептуальну й функціональну сумісність цих класів систем за умов експлуатації, вимог реального часу, обмежень енергетичного бюджету та використання вбудованих обчислювальних платформ. Це дало змогу узагальнити запропоновані методи, математичні моделі й алгоритми адаптивного балансування реактивності та енергоспоживання, а також обґрунтувати можливість їх застосування як у малих космічних системах, так і в бортових обчислювальних системах безпілотних літальних апаратів.

У роботі вперше розроблено комплексну математичну модель, що інтегрує показники енергоспоживання, обчислювальної продуктивності та реактивності бортового програмного забезпечення. Запропонована модель дає змогу описувати режими функціонування бортового обчислювача, враховувати нестаціонарність обчислювального навантаження та обмеження систем реального часу. Для ідентифікації параметрів моделі запропоновано метод на основі рекурсивного методу найменших квадратів, який забезпечує оперативне оцінювання параметрів у процесі експлуатації за умов обмежених обчислювальних ресурсів, характерних для авіоніки.

Розроблено метод адаптивного регулювання режимів роботи бортових обчислювачів, що ґрунтується на концепції послаблених систем жорсткого реального часу та забезпечує динамічне узгодження вимог до реактивності з енергетичними обмеженнями. Метод реалізовано у вигляді алгоритмів і програмних компонентів бортового програмного забезпечення з використанням компонентно-орієнтованого фреймворку NASA F′ Prime та операційної системи реального часу FreeRTOS. Реалізацію алгоритмів і їх експериментальну валідацію виконано на авторському бортовому обчислювачі «Боривітер 0.1».

Проведені експериментальні дослідження підтвердили адекватність запропонованих моделей та ефективність розробленого адаптивного алгоритму. Отримані результати демонструють зменшення енергоспоживання бортового обчислювача за збереження заданого рівня реактивності програмного забезпечення, що сприяє підвищенню тривалості автономної роботи та надійності функціонування системи.

Практичне значення одержаних результатів полягає в можливості використання запропонованих методів, алгоритмів і програмних засобів під час проєктування та розроблення бортового програмного забезпечення наносупутників класу CubeSat, безпілотних літальних апаратів, а також інших вбудованих систем з обмеженим енергетичним бюджетом і вимогами реального часу.

Ключові слова: наносупутники, безпілотні системи, безпілотні літальні апарати, часові метрики, програмний засіб, енергетична ефективність, реактивність, продуктивність, ідентифікація параметрів моделі, математична модель, системи реального часу, бортові обчислювальні системи, алгоритм синтезу моделі, чисельне моделювання, модель.

Список публікацій та особистий внесок здобувача:

1. Liubimov O., Turkin I., Cheranovskiy V., Volobuieva L. UAV Mission Computer Operation Mode Optimization Focusing on Computational Energy Efficiency and System Responsiveness. Computation. 2024, vol. 12, iss. 12, art. 235. DOI: 10.3390/computation12120235 (Scopus, Q2). *(Особистий внесок: Розробка концепту, методології дослідження, програмного забезпечення, формальний аналіз та валідація, написання, рев'ю та редагування)*

2. Liubimov O., Turkin I., Pavlikov V., Volobuyeva L. Agile Software Development Lifecycle and Containerization Technology for CubeSat Command and Data Handling Module Implementation. Computation, 2023, vol. 11, iss. 9, art. 182. DOI: 10.3390/computation11090182 (Scopus, Q2). *(Особистий внесок: Розробка концепту, методології дослідження, програмного забезпечення, формальний аналіз та валідація, написання, рев'ю та редагування)*

3. Turkin I., Volobuieva L., Chukhray A., Liubimov O. Comparison of equivalent circuit and machine learning methods for CubeSat battery discharge modeling // Radioelectronic and Computer Systems. – 2025. – № 3. – P. 231–244. – DOI: 10.32620/reks.2025.3.16 (Scopus, Q2) *(Особистий внесок: розробка методології дослідження, рев'ю та редагування)*

4. Liubimov O., Turkin I. Cubesats and their on-board computers: systematic literature review. Space Science and Technology. 2025. Vol 6, Iss. 157, pp. 014-037, DOI: 10.15407/knit2025.06.014 (Scopus, Q4). *(Особистий внесок: Розробка концепту, методології дослідження, аналіз вхідних даних, написання, рев'ю та редагування)*

5. Liubimov O., Liubimov M. Use Of Open-Source COTS/MOTS Hardware And Software Platforms For The Build Up Of The Cubesat Nanosatellites. Journal of Rocket-Space Technology, 2023, vol. 31, iss. 4, pp. 138–147. DOI: 10.15421/452318. (Кат. Б) *(Особистий внесок: постановка завдання, аналіз вхідних даних, розробка методології дослідження).*

6. Turkin I., Liubimov O., Zakharenko V. Optimization of energy consumption of the CubeSat on-board computer under real-time limitations. Aerospace Technic and Technology, 2024, vol. 0, iss. 3, pp. 126-137. DOI: 10.32620/aktt.2024.3.10. (Кат. Б) *(Особистий внесок: Формування основної концепції статті та методології наукових питань, формування проблеми оптимізації енергоспоживання, планування та проведення наукового експерименту)*

7. Turkin I., Liubimov O., Volobuieva L., Valkovyi V. Hardware and software of CubeSat nanosatellites' on-board computers: a systematized literature review // Aerospace Technic and Technology. – 2025. – № 3.- P.137-154. - DOI: https://doi.org/10.32620/aktt.2025.3.11 (Кат. Б) *(Особистий внесок: Розробка концепту, методології дослідження)*

8. Liubimov O., Turkin I. Experimental study of FreeRTOS operating system reactivity in power saving modes of the onboard computer microcontroller// Aerospace Technic and Technology. – 2024, № 4, – P. 71–86. – DOI: 10.32620/aktt.2024.4.09 (Кат. Б) *(Особистий внесок: розробка апаратного та програмного забезпечення,*

виконав дослідження, виконання пошуку ресурсів, зроблено візуалізацію та проведена верифікація тексту)

9. Liubimov O., Turkin I., Leshchenko O., Valkovyi V. Experimental evaluation of the efficiency of containerization technology in the onboard software of a nanosatellite CubeSat // Aerospace Technic and Technology. – 2024, № 4, – P. 87–103. – DOI: 10.32620/aktt.2024.4.10 (Кат. Б) *(Особистий внесок: розробка апаратного та програмного забезпечення, виконав дослідження, виконав пошук ресурсів, зробив візуалізацію та провів верифікацію тексту)*

10. Turkin I., Liubimov O., Volobuieva L. Reactivity model of nanosatellite and UAVs embedded computing systems // Aerospace Technic and Technology. – 2025. – № 6. – P. 128–142. – DOI: 10.32620/aktt.2025.6.12 (Кат. Б) *(Особистий внесок: розробка математичної моделі реактивності, виконання експериментів)*

11. Liubimov O., Turkin I. Multicriteria evaluation of industry-known interfaces of onboard network of CubeSat nanosatellites // Aerospace Technic and Technology. – 2024. – № 3. – P. 138–153. – DOI: 10.32620/aktt.2024.3.12. (Кат. Б) *(Особистий внесок: концептуалізація, методологія, розробка моделі порівняння, аналіз результатів, візуалізація, написання)*

12. Turkin I., Liubimov O., Shevchenko I. Combined model of the hardware-software development process for a student CubeSat nanosatellite data processing module // Open Information and Computer Integrated Technologies. – 2024. – № 101. – P. 148–165. – DOI: 10.32620/oikit.2024.101.10. (Кат. В) *(Особистий внесок: концептуалізація, методологія)*

13. Liubimov O., Turkin I., Volobuieva L., Iovenko I., Liubimov M. Developing hardware and software for UAV onboard computers: a combined approach with SWOT analysis // Integrated Computer Technologies in Mechanical Engineering – 2024 : proceedings of the Int. Conf. ICTM 2024 / eds. O. Lytvynov, V. Pavlikov, D. Krytskyi. – Cham : Springer, 2025. – (Lecture Notes in Networks and Systems ; Vol. 1473). – DOI: 10.1007/978-3-031-94845-9_18. (Scopus) *(Особистий внесок: концептуалізація, методологія)*

14. Liubimov O., Turkin I. Data model and methods for ensuring the reliability and relevance of data for the CubeSat projects // Proc. 12th Int. Conf. Dependable Systems, Services and Technologies (DESSERT). – Athens, Greece, 2022. – Pp. 1–7. – DOI: 10.1109/DESSERT58054.2022.10018658. (Scopus) *(Особистий внесок: концептуалізація, методологія)*

15. Liubimov O., Turkin I., Volobuieva L. The WASM3 interpreter as a hard real-time software platform for the on-board computer of a student nanosatellite // Proc. 13th Int. Conf. Dependable Systems, Services and Technologies (DESSERT). – Athens, Greece, 2023. – P. 1–8. – DOI: 10.1109/DESSERT61349.2023.10416436. (Scopus) *(Особистий внесок: концептуалізація, методологія, вимірювання та обробка результатів експериментів)*

16. Liubimov O. Use of micro-services architecture and containerization for the fast development and testing of the CubeSat nanosatellites software // Journal of Rocket-Space Technology. – 2023. – Vol. 31, № 4. – P. 128–137. – DOI: 10.15421/452317 (Кат. Б)

17. Любімов О. Модель модульного та компонентного підходу до проєктування ПЗ наносупутників на базі відкритих рішень NASA F'Prime // Матеріали Міжнародної молодіжної науково-практичної конференції «Людина і Космос – 2024». – Дніпро, 2024. (Кат. Б, тези)

18. Любімов О. Реалізація енергоефективної бортової мережі між типовими модулями малої авіоніки мікросупутників КубСат (CubeSat) // Матеріали Міжнародної молодіжної науково-практичної конференції «Людина і Космос – 2024». – Дніпро, 2024. – DOI: 10.62717/2221-4550-2024-1-067. (Кат. Б, тези)

Усі співавтори погодилися із задекларованим особистим внеском здобувача у спільних публікаціях.

ABSTRACT

Oleksandr Liubimov. Adaptive methods, algorithms, and software tools for balancing reactivity and power consumption of onboard computers of CubeSat nanosatellites – manuscript.

Dissertation submitted for the degree of Doctor of Philosophy (PhD) in specialty 121 “Software Engineering” at the National Aerospace University “Kharkiv Aviation Institute”, Kharkiv, 2026.

The dissertation is devoted to the development of adaptive methods, algorithms, and software tools for balancing reactivity and power consumption of onboard computers of CubeSat nanosatellites and unmanned aerial vehicles (UAVs).

The object of the research is the processes of balancing and controlling the reactivity of onboard software and the power consumption of onboard computers of CubeSat nanosatellites and UAVs.

The subject of the research comprises mathematical models, methods, algorithms, and software tools for adaptive regulation of onboard computer operating modes, considering real-time constraints, computational performance, and energy budget limitations.

The dissertation presents a systematic analysis of the current state of CubeSat nanosatellite technologies, their operating conditions, architectural standards, and the use of commercial off-the-shelf (COTS) hardware and software components. It is shown that the rapid growth of CubeSat missions is accompanied by a high rate of mission failures and satellite losses, a significant proportion of which is caused by inconsistencies between power management algorithms and the reactivity of onboard software.

Based on a comprehensive literature review, it is established that energy efficiency, computational performance, and reactivity of onboard computers are interrelated characteristics that determine mission reliability and success; however, existing approaches typically consider these characteristics in isolation. The necessity of using formalized integrated models for coordinated analysis and optimization of these characteristics is substantiated.

Within the framework of the research, a comparative analysis of architectural approaches and principles for designing onboard computing systems of CubeSat nanosatellites and UAVs is performed. Significant conceptual and functional compatibility between these classes of systems under operating conditions, real-time requirements, energy budget constraints, and the use of embedded computing platforms is identified. This makes it possible to generalize the proposed methods, mathematical models, and algorithms for adaptive balancing of reactivity and power consumption and to substantiate their applicability both to small-class space systems and to onboard computing systems of UAVs.

For the first time, an integrated mathematical model is developed that combines indicators of power consumption, computational performance, and reactivity of onboard software. The proposed model enables the description of onboard computer operating modes, accounts for non-stationary workloads, and considers real-time system constraints. To identify the model parameters, a method based on the recursive least squares algorithm is proposed, which provides online parameter estimation during operation under limited computational resources characteristic of avionics computing systems.

An adaptive method for regulating the operating modes of onboard computers is developed, based on the concept of weakly hard real-time systems, which enables dynamic coordination of reactivity requirements with energy constraints. The method is implemented in the form of algorithms and software components of onboard software using the NASA F′ Prime component-based framework and the FreeRTOS real-time operating system. The implementation of the algorithms and their experimental validation are carried out on the author’s onboard computer “Boryviter 0.1”.

Experimental studies confirm the adequacy of the proposed models and the effectiveness of the developed adaptive algorithm. The obtained results demonstrate a reduction in the power consumption of the onboard computer while maintaining a specified level of software reactivity, which contributes to increased autonomous operation time and improved system reliability.

The practical significance of the research lies in the applicability of the proposed methods, algorithms, and software tools to the design and development of onboard software for CubeSat nanosatellites, UAVs, and other embedded systems with limited energy budgets and real-time requirements.

Keywords: nanosatellites, unmanned systems, unmanned aerial vehicles, temporal metrics, software tool, energy efficiency, reactivity, productivity, model parameter identification, mathematical model, real-time systems, onboard computing systems, model synthesis algorithm, numeric modeling, model.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 15

ВСТУП 18

РОЗДІЛ 1 ОГЛЯД ТА АНАЛІЗ ПРОБЛЕМ ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИЧНОЇ ЕФЕКТИВНОСТІ БОРТОВИХ ОБЧИСЛЮВАЧІВ НАНОСУПУТНИКІВ CUBESAT 26

1.1 Аналіз сучасних тенденцій розвитку супутникових технологій 26

1.1.1 Типова побудова наносупутників CubeSat 29

1.1.2 Стандартизація наносупутників CubeSat 31

1.1.3 Умови експлуатації БО наносупутників CubeSat 37

1.1.4 Використання комерційно доступних апаратних компонентів 39

1.1.5 Використання відкритого програмного забезпечення 40

1.1.6 Проблеми та виклики наносупутників CubeSat 42

1.1.7 Сучасні тенденції розвитку та проблеми наносупутників CubeSat . 45

1.2 Систематизований огляд наукових публікацій, присвячених апаратно-програмному забезпеченню бортових обчислювачів безпілотної аерокосмічної техніки 46

1.2.1 Технологія систематизованого літературного огляду 47

1.2.2 Аналіз публікацій, присвячених апаратному забезпеченню бортових обчислювачів наносупутників 53

1.2.3 Аналіз публікацій, присвячених програмному забезпеченню бортових обчислювачів наносупутників 54

1.2.4 Аналіз публікацій, присвячених програмному забезпеченню бортових обчислювачів БПЛА 58

1.3 Аналіз методів забезпечення енергетичної ефективності обчислень з урахуванням обмежень реального часу та вимог до реактивності 62

1.3.1 Аналіз наукових публікацій за напрямком 63

1.3.2 Таксономія дослідницьких напрямків та методів 65

1.3.3 Методи планування та диспетчеризації завдань в системах реального часу 68
1.3.4 Аналіз здійсненності системи завдань реального часу 71
Висновки до розділу 1 71
РОЗДІЛ 2 КОМПЛЕКСНА МАТЕМАТИЧНА МОДЕЛЬ СПОЖИВАННЯ ЕНЕРГІЇ, ОБЧИСЛЮВАЛЬНОЇ ПРОДУКТИВНОСТІ ТА РЕАКТИВНОСТІ БОРТОВИХ ОБЧИСЛЮВАЧІВ НАНОСУПУТНИКІВ CUBESAT 75
2.1 Вимоги до комплексної математичної моделі енергетичних витрат, обчислювальної продуктивності та реактивності бортових обчислювачів 77
2.1.1 Комплексна математична модель в нотації теорії множин 77
2.1.2 Режими роботи бортового обчислювача 78
2.2 Модель реактивності програмного забезпечення бортових обчислювачів наносупутників 79
2.2.1 Загальні положення 79
2.2.2 Вихідні припущення та обмеження 82
2.2.3 Функційні залежності моделі реактивності 85
2.2.4 Результати чисельного експерименту для перевірки моделі реактивності 87
2.3 Модель продуктивності програмного забезпечення бортових обчислювачів наносупутників 92
2.3.1 Загальні положення 92
2.3.2 Вихідні припущення та обмеження 93
2.3.3 Функційні залежності моделі продуктивності 97
2.4 Модель енергоспоживання бортового обчислювача 99
2.4.1 Вихідні припущення та обмеження 99
2.4.2 Функційні залежності математичної моделі енергоспоживання бортового обчислювача 100
Висновки до розділу 2 104

РОЗДІЛ 3 МЕТОДИ ІДЕНТИФІКАЦІЇ КОМПЛЕКСНОЇ МАТЕМАТИЧНОЇ МОДЕЛІ ТА АДАПТИВНОГО РЕГУЛЮВАННЯ РЕЖИМІВ РОБОТИ БОРТОВИХ ОБЧИСЛЮВАЧІВ НАНОСУПУТНИКІВ CUBESAT 107

3.1 Метод ідентифікації параметрів комплексної математичної моделі..... 107

3.1.1 Обґрунтування припущень та обмежень щодо архітектури та алгоритмів ПЗ 108

3.1.2 Формалізація задачі ідентифікації параметрів комплексної математичної моделі 113

3.1.3 Рекурсивний метод найменших квадратів..... 116

3.1.4 Методика та техніки вимірювання показників реактивності та обчислювальної продуктивності програмного забезпечення 117

3.1.5 Методика та техніки вимірювання енергетичних витрат апаратної складової БО 119

3.1.6 Результати експерименту 124

3.1.7 Перевірка адекватності математичних моделей 126

3.2 Метод адаптивного регулювання режимів роботи БО 129

3.2.1 Адаптивна політика енергозбереження в мікроконтролерних системах послабленого жорсткого часу..... 129

3.2.2 Апріорні оцінки інтенсивності та кластеризації подій..... 135

3.2.3 Реєстрація та дискретизація подій..... 138

3.2.4 Оцінювання інтенсивності та прогнозування періоду 139

3.2.5 Функція прийняття рішення, щодо переходу в енергозберігаючий режим..... 141

3.2.6 Реалізації методу адаптивного регулювання режимів роботи БО на основі цифрового автомату з пам'яттю 143

Висновки до розділу 3 148

РОЗДІЛ 4 АЛГОРИТМИ ТА ЇХ ПРОГРАМНА РЕАЛІЗАЦІЯ У БОРТОВОМУ ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ НАНОСУПУТНИКІВ CUBESAT 150

4.1 Вимоги до бортового програмного забезпечення БО 151

4.1.1 Вимоги до бортового програмного забезпечення БО наносупутників CubeSat 151
4.1.2 Вимоги до бортового програмного забезпечення БО БПЛА 154
4.2 Архітектура бортового програмного забезпечення 155
4.2.1 Обґрунтування використання F′ Prime як основи побудови БПЗ .. 155
4.2.2 Обґрунтування використання ОСРЧ FreeRTOS 157
4.2.3 Шар абстракції від апаратних реалізацій (HAL) 159
4.2.4 Інтеграція HAL, FreeRTOS і F′ для БО “Боривітер” 161
4.2.5 Архітектура БПЗ 164
4.3 Розроблення алгоритмів адаптивного регулювання енергетичними режимами МК БО 169
4.3.1 Постановка задачі для синтезу адаптивних алгоритмів керування енергетичними режимами МК БО 169
4.3.2 Розроблення компонента F′ для реалізації методу адаптивного регулювання режимами енергозбереження МК БО 171
4.4 Експериментальне дослідження ефективності розробленого адаптивного алгоритму регулювання режимів МК БО 182
4.4.1 Планування експерименту, підготовка БПЗ та обладнання 182
4.4.2 Результати експерименту 186
Висновку до розділу 4 190
ВИСНОВКИ 192
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 195
ДОДАТОК А. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ 209
ДОДАТОК Б. ОБЛІКОВА КАРТКА НДДКР 214
ДОДАТОК В. ТЕСТОВІ ТОПОЛОГІЇ БПЗ 216

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АЗ	-	Апаратне Забезпечення
БО	-	Бортовий Обчислювач
БПЗ	-	Бортове Програмне Забезпечення
БПЛА	-	Безпілотний Літальний Апарат
БС	-	Батарея Сонячна
БХ	-	Батарея Хімічна
ДЗЗ	-	Дистанційне Зондування Землі
КА	-	Космічний Апарат
КО	-	Космічна Орбіта Землі
КОМ	-	Комунікаційна підсистема наносупутника
МК	-	Мікроконтролер
МКС	-	Міжнародна Космічна Станція
НДР	-	Науково Дослідницька Робота
ННО	-	Низька Навколоземна Орбіта
ОС	-	Операційна Система
ОСРЧ	-	Операційна Система Реального Часу
ПЗ	-	Програмне Забезпечення
ПК	-	Персональний Комп’ютер
РМНК	-	Рекурсивний Метод Найменших Квадратів
РЧ	-	Реальний Час
СЕЗ	-	Система Електрозабезпечення наносупутника
СПЗ	-	Системне Програмне Забезпечення
ТПЗ	-	Тестове Програмне Забезпечення
ADCS	-	Attitude Determination and Control System
BVLOS	-	Beyond Visual Line of Sight
Cal Poly	-	California Polytechnic State University
CAN	-	Controller Area Network
CCSDS	-	Consultative Committee for Space Data Systems

CDHM	-	Command and Data Handling Module
CDS	-	CubeSat Design Specification
COTS	-	Commercial Off-The-Shelf
CPU	-	Central Processing Unit
CSLI	-	CubeSat Launch Initiative
CSP	-	CubeSat Space Protocol
DFS	-	Dynamic Frequency Scheduling
ECSS	-	European Cooperation for Space Standardization
EDF	-	Earliest Deadline First
EPS	-	Electrical Power System
ESA	-	European Space Agency
F`	-	F`Prime Software Framework
FCC	-	Federal Communications Commission
FDIR	-	Fault Detection Isolation and Recovery
FFT	-	Fast Fourier Transform
FOMS	-	Full Operation Modes Set
GEVS	-	General Environmental Verification Standard
GPIO	-	General Purpose Input Output
HALE	-	High Altitude Long Endurance
HW	-	Hardware
I2C	-	Inter-Integrated Circuit
ICD	-	Interface Control Document
IDD	-	Interface Definition Document
IEEE	-	Institute of Electrical and Electronics Engineers
ISR	-	Interrupt Service Routine
ISS	-	International Space Station
J-SSOD	-	Japanese Small Satellite Orbital Deployer
LEO	-	Low Earth Orbit
LLF	-	Least Laxity First
LPM	-	Low Power Modes

MALE	-	Medium Altitude Long Endurance
MBSE	-	Model-Based Systems Engineering
MOTS	-	Modifiable Off-The-Shelf
NASA	-	National Aeronautics and Space Administration
NRCSD	-	NanoRacks CubeSat Deployer
OBC	-	On-Board Computer
OSS	-	Open-Source Software
P-POD	-	Poly-Picosatellite Orbital Deployer
QoS	-	Quality of Service
QuadPack	-	QuadPack satellite deployer
RISC	-	Reduced Instruction Set Computer
RMS	-	Rate Monotonic Scheduling
ROS	-	Robotic Operation System
RTD	-	Race To Dark
RTOS	-	Real Time Operating System
SEL	-	Single Event Latchup
SEU	-	Single Event Upset
SPI	-	Serial Peripheral Interface
SSE	-	Sum of Squares Error
SSR	-	Sum of Squares Regression
SST	-	Sum of Squares Total
UART	-	Universal Asynchronous Receiver/Transmitter
UAV	-	Unmanned Aerial Vehicle
UML	-	Unified Modeling Language
WASM	-	Web Assembly
WCET	-	Worst Case Execution Time
WFI	-	Wait For Interrupt

ВСТУП

Обґрунтування вибору теми дослідження. Не дивлячись на багаторічну історію розвитку бортових обчислювальних систем космічних апаратів, програмне забезпечення бортових обчислювачів і сьогодні демонструє системні проблеми, що безпосередньо впливають на надійність і ефективність космічних місій. Історія космонавтики містить численні приклади, коли саме програмне забезпечення ставало критичним чинником часткової або повної втрати апаратів. В місіях мікро- та наносупутників, ці ризики суттєво зростають через жорсткі обмеження на енергоспоживання, обчислювальні ресурси та автономність функціонування. Навіть за формальної коректності алгоритмів і дотримання вимог до тривалості та періодичності їх виконання, програмне забезпечення нерідко неспроможне адаптувати власну продуктивність та реактивність до фактичного енергетичного стану системи.

Показовими є приклади CubeSat-місій SwissCube та експериментальних апаратів серії PhoneSat (NASA), у яких фіксувалися збої бортового програмного забезпечення, пов’язані з перевантаженням обчислювальних ресурсів і порушенням енергетичного балансу, що призводило до нестабільної роботи або втрати зв’язку з апаратом. Ці випадки продемонстрували, що проблема полягає не лише в обмеженості енергії як такої, а й у відсутності формалізованого механізму балансування між енергоефективністю та реактивністю бортових обчислювальних засобів, особливо критичних для задач керування орієнтацією, стабілізації та аварійного реагування.

У національному контексті України подібні проблеми також мали практичні наслідки. За відкритими даними, втрати супутників «Січ-2-30» та PolyITAN-HP-30, які мали важливе значення для розвитку вітчизняної космічної індустрії та науково-освітніх програм, пов’язуються, зокрема, з труднощами забезпечення стабільного енергетичного балансу бортових систем. Ці приклади підкреслюють актуальність проблеми не лише в глобальному, але й у національному вимірі, де втрата навіть одного супутника в умовах обмежених ресурсів країни є критичною.

Вагомий внесок у розв’язання проблем функціонування обчислювальних систем у режимах обмежень реального часу зробили окремі науковці та наукові школи, зокрема C. L. Liu, J. W. Layland, G. Buttazzo, L. Palopoli, H. Aydin, M. Pedram та інші, які заклали теоретичні основи планування задач, енергетично-усвідомленого керування ресурсами та оптимізації обчислювальних навантажень. Значний практичний внесок здійснили також провідні космічні організації - NASA, ESA, DLR, JAXA, а також міжнародні консорціуми CCSDS і ECSS, у межах яких були сформовані сучасні стандарти архітектури, надійності та взаємодії бортових обчислювальних систем. На основі цих напрацювань створено відкриті програмні платформи, що широко застосовуються в космічних місіях малого класу, зокрема програмні фреймворки - NASA Core Flight System (cFS), F′ (F′ Prime), розроблені Космічним центром Годдарта NASA та Лабораторією реактивного руху NASA (JPL), а також операційні системи реального часу RTEMS і FreeRTOS. Вони забезпечують модульність, повторне використання програмного коду та детерміноване виконання задач, однак орієнтовані переважно на часову коректність і функціональну надійність. Питання енергоспоживання в цих платформах розглядається як зовнішнє обмеження і не інтегрується безпосередньо в моделі планування та керування виконанням програмних компонентів, що змушує розробників застосовувати фрагментарні й емпіричні підходи.

В результаті залишаються невирішеними питання розробки моделей та методів, які забезпечують формалізований та адаптивний баланс між енергоефективністю, обчислювальною продуктивністю та реактивністю бортових обчислювачів наносупутників у режимах реального часу, а, також, створення програмних засобів, здатних підтримувати такий баланс у змінних умовах експлуатації. Саме ця прогалина визначає актуальність даного дослідження, є його основною мотивацією, та зумовлює необхідність подальшого розвитку теоретичних і практичних підходів до проєктування бортового програмного забезпечення космічних систем малого класу.

Таким чином, задача розробки методів і програмних засобів забезпечення збалансованої енергоефективності та реактивності бортових обчислювачів наносупутників CubeSat, є актуальним науково-прикладним завданням, важливим як для розвитку міжнародних місій малого класу, так і для формування національного космічного потенціалу України.

Вирішення цього завдання долає **протиріччя** – між вимогами до високої реактивності та низького споживання енергії бортовими обчислювачами за відсутності відповідних методів, алгоритмів та програмних засобів.

Науково-прикладне завдання – полягає у створенні комплексу формалізованих математичних моделей, методів та програмних засобів, призначених для узгодженого врахування енергоспоживання, обчислювальної продуктивності й реактивності бортового програмного забезпечення наносупутників та реалізації їх адаптивного балансування в режимі реального часу.

Об'єкт дослідження – реактивність бортового програмного забезпечення та енергоспоживання бортових обчислювачів наносупутників CubeSat.

Предмет дослідження – моделі та методи балансування реактивності бортового програмного забезпечення та енергоспоживання бортових обчислювачів наносупутників CubeSat.

Мета і завдання дослідження. Метою є зменшення енергетичних витрат бортових обчислювачів наносупутників CubeSat за обмежень реального часу та гарантування реактивності бортового програмного забезпечення.

Для досягнення мети дослідження необхідно вирішити наступні завдання:

1) Виконати аналіз умов використання бортових обчислювачів наносупутників CubeSat;

2) Розглянути вимоги та проєктні обмеження до бортового програмного забезпечення, класифікувати існуючі рішення бортового програмного забезпечення наносупутників CubeSat;

3) Розробити комплексну математичну модель для розрахунку енергетичних витрат бортового обчислювача, обчислювальної продуктивності та реактивності бортового програмного забезпечення;

4) Розробити метод ідентифікації параметрів математичної моделі для розрахунку енергетичних витрат бортового обчислювача, обчислювальної продуктивності та реактивності бортового програмного забезпечення, що забезпечує онлайн-ідентифікацію та може бути впроваджений в умовах обмежених обчислювальних ресурсів;

5) Виконати ідентифікацію параметрів математичної моделі для розрахунку енергетичних витрат бортового обчислювача, обчислювальної продуктивності та реактивності бортового програмного забезпечення;

6) Розробити метод адаптивного регулювання режимів роботи бортових обчислювачів наносупутників CubeSat;

7) Розробити алгоритми та бортове програмне забезпечення для адаптивного регулювання режимів роботи бортових обчислювачів наносупутників CubeSat;

8) Підтвердити ефективність розроблених методів, алгоритмів та програмного забезпечення засобами фізичного експерименту.

Методи дослідження.

1) Для аналізу умов використання бортових обчислювачів наносупутників CubeSat, огляду вимог та проєктних обмежень, класифікації існуючих рішень бортового програмного забезпечення використано техніку систематизованого літературного огляду.

2) Для розробки комплексної математичної моделі енергетичних витрат бортового обчислювача, обчислювальної продуктивності та реактивності бортового програмного забезпечення застосовано теоретико-множинне представлення та теорії черг та масового обслуговування, для поєднання в одне ціле моделі енергоспоживання бортового обчислювача, моделей продуктивності та реактивності програмного забезпечення.

3) Для розробки метода ідентифікації параметрів комплексної математичної моделі енергетичних витрат бортового обчислювача, обчислювальної продуктивності та реактивності бортового програмного забезпечення використано рекурсивний метод найменших квадратів.

4) Для аналізу результатів імітаційного моделювання та фізичного експерименту використано методи математичної статистики.

5) Для розробки методу адаптивного регулювання режимів роботи бортових обчислювачів наносупутників CubeSat використано концептуальне рішення послаблених систем жорсткого реального часу (англ. Weakly Hard Real-Time Systems).

6) При розробці прототипу програмного забезпечення для перевірки ефективності запропонованого методу використано методи об'єктно-орієнтованого та компонентно-орієнтованого аналізу та проєктування в парадигмі фреймворку F′Prime.

Наукова новизна отриманих результатів:

Вперше:

– **розроблено** *комплексну математичну модель* для розрахунку енергетичних витрат бортового обчислювача, обчислювальної продуктивності та реактивності бортового програмного забезпечення, яка на відміну від існуючих дозволяє врахувати всі взаємозв’язки між характеристиками системи та забезпечує всебічний аналіз бортових обчислювачів;

– **розроблено** *метод* ідентифікації параметрів комплексної математичної моделі для розрахунку енергетичних витрат, обчислювальної продуктивності та реактивності бортового програмного забезпечення, який на відміну від існуючих рішень використовує рекурсивний метод найменших квадратів, що забезпечує онлайн-ідентифікацію параметрів моделі з урахуванням нестаціонарності навантаження та обмежених обчислювальних ресурсів;

– **удосконалено** *метод* адаптивного регулювання режимів роботи бортових обчислювачів наносупутників CubeSat, який, на відміну від існуючих, використовує концептуальне рішення послаблених систем жорсткого реального

часу, що дозволяє забезпечити гнучкість і баланс між реактивністю бортового програмного забезпечення та енергетичною ефективністю обчислювача.

Дістало подальшого розвитку методичне, алгоритмічне та програмне забезпечення системних програмних засобів бортових обчислювачів наносупутників CubeSat в напряму зменшення енергетичних витрат бортового обчислювача в умовах обмежень систем реального часу та забезпечення реактивності бортових систем.

Публікації. За темою дисертаційної роботи опубліковано 20 наукових праць: 2 статті у міжнародних наукових виданнях, 11 статей у наукових фахових виданнях України, з них дві у виданнях, внесеному до міжнародних наукометричних баз даних, 7 – у матеріалах наукових конференцій.

Усі основні наукові положення, результати, висновки й рекомендації дисертаційної роботи отримано автором самостійно.

Апробація матеріалів дисертації. Основні положення та ідеї дисертаційної роботи доповідалися та обговорювалися на: науково-технічних конференціях: "Integrated Computer Technologies in Mechanical Engineering" (м. Харків, 2022, 2023, 2024); "Dependable System, Services and Technologies Conference (Athens, Greece 2022, 2023); «Людина та Космос» та «Космічні горизонти» (Дніпро, Україна, 2023-2024 рр.), круглих столах МОН «Молоді науковці» (Київ, Україна, 2024-2025), КриКТехС (Харків, Україна, 2025), міжнародному симпозіумі електричних літальних апаратів та БПЛА (Львів, Україна, 2025). А також на хакатонах – «Апаратні рішення для відновлення України» (Львів, Україна, 2024), «Sigma Unicorn Startup» (Київ, Україна, 2024), «Sikorskiy Challenge» (Київ, Україна, 2025), «NASA SpaceApp Challenge» (Київ, Україна, 2025), «Science2Business» (Київ, Україна, 2024).

Зв'язок роботи з науковими програмами, планами, темами. Робота автора над дисертацією проводилася на кафедрі інженерії програмного забезпечення ХАІ відповідно до планів наукових досліджень Національного аерокосмічного університету «Харківський авіаційний інститут» в рамках держбюджетних науково-дослідницьких робіт:

– НДР «Технології розробки програмного забезпечення для платформ екосистем Інтернету речей і хмарних сервісів, терміни виконання» (ДР № 0121U109086 2022-2023 рр.);

– НДР «Технологія розроблення програмного забезпечення для інноваційних систем спеціального та подвійного призначення» (ДР № 0124U001426, 2024-2026 рр.);

– гранту національного фонду досліджень України «Експериментальне відпрацювання бортового обчислювача безпілотного літального апарата подвійного призначення» (ДР № 0124U004847, 2024-2025 рр.).

Роль автора у зазначених НДР, в яких автор був безпосереднім виконавцем, полягає у:

– розробці методів та програмних засобів керування енергетичними режимами мікроконтролера бортового обчислювача, розробці архітектурного концепту та дизайну бортового програмного забезпечення;

– розробці та портуванні відкритих програмних рішень як універсального бортового програмного забезпечення;

– розробці програм і методики випробування бортового програмного забезпечення в складі бортового обчислювача, шляхом симуляційних, лабораторних, стендових та льотних випробувань.

Практичне значення отриманих результатів полягає в тому, що основні наукові положення дисертації реалізовано в вигляді інженерних алгоритмів і відповідних програмних засобів. Розроблений метод адаптивного регулювання режимів роботи бортового програмного забезпечення дозволяє підвищити енергетичну ефективність бортових обчислювачів наносупутників CubeSat. Це забезпечує подовження тривалості автономної роботи апарата, підвищує продуктивність обчислень і знижує ризик відмов у критичних умовах. Результати можуть бути впроваджені у системах керування обчислювальними ресурсами автоматичних мобільних платформ для розширення їх функційності та зниження витрат на експлуатацію. Усі розроблені моделі й методи доведено

до рівня інженерних методик, програмних засобів та комп'ютерної реалізації, ефективність яких обґрунтовано теоретично й підтверджена практично.

Результати роботи було впроваджено (додаток А, додаток Б):

- у навчальному процесі Національного аерокосмічного університету «Харківський авіаційний інститут» (акт впровадження від 17 жовтня 2025);
- при виконанні науково-дослідних проєктів, що виконувались у Національному аерокосмічному університеті «Харківський авіаційний інститут» (облікова картка проєкту 0124U004847);
- у аванпроєкті та стендових випробуваннях в складі КА що розроблявся НТУ КПІ СКБ «Шторм» (лист підтримки від 29 серпня 2025);
- у науково прикладній роботі з провідним космічним підприємством АТ «ХАРТРОН» (акт впровадження від 1 жовтня 2025);
- у прикладній роботі з підприємством ТОВ «АБАКУС-ТЕХ» (акт впровадження від 6 жовтня 2025).

Структура та обсяг дисертації. Загальний обсяг дисертації становить: 228 сторінок, у тому числі анотація на 9 сторінках, зміст на 4 сторінках, перелік умовних скорочень на 3 сторінках, основний текст на 176 сторінках, список використаних джерел із 170 найменувань на 14 сторінках, додатки на 20 сторінках. Робота містить 10 таблиць, з яких 1 таблиця на 1 окремій сторінці і 61 рисунок, з яких 10 рисунків на 10 окремих сторінках.

РОЗДІЛ 1 ОГЛЯД ТА АНАЛІЗ ПРОБЛЕМ ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИЧНОЇ ЕФЕКТИВНОСТІ БОРТОВИХ ОБЧИСЛЮВАЧІВ НАНОСУПУТНИКІВ CUBESAT

1.1 Аналіз сучасних тенденцій розвитку супутникових технологій

Ландшафт космічних досліджень та супутникових технологій зазнав глибоких трансформацій протягом останніх двох десятиліть, фундаментально змінивши питання про те, хто саме може отримати доступ до космосу і наскільки швидко та дешево можуть розгортатися нові та масштабні місії.

Штучні супутники стали невід'ємними елементами сучасної космічної діяльності, забезпечуючи широкий спектр застосувань – від дистанційного зондування Землі (ДЗЗ) та телекомунікацій до наукових досліджень і відпрацювання нових технологій. Якщо перші космічні апарати були великими, високоспеціалізованими й надзвичайно дорогими, то сучасні місії охоплюють набагато ширший спектр розмірів і функційних можливостей. З метою уніфікації підходів до проєктування та спрощення планування місій супутники зазвичай класифікують переважно за їхньою масою.

На сьогодні штучні супутники Землі (або інших планет Сонячної системи) класифіковані на традиційні та мініатюризовані класи.

У межах традиційного класу супутників зазвичай розрізняють такі групи:

1. Великі супутники (> 1 000 кг): геостаціонарні телекомунікаційні супутники або провідні наукові обсерваторії. Вони характеризуються значною вантажністю, високим рівнем резервування та тривалим терміном експлуатації. Для їхнього виведення потрібні спеціальні ракети-носії, а самі місії є найскладнішими й найдорожчими;
2. Середні супутники (500 – 1 000 кг);
3. Малі супутники (100 – 500 кг) – як і середні, використовуються в різноманітних наукових і земле-орієнтованих місіях. Попри відносно високі грошові витрати, вони дедалі частіше будуються з використанням модульних підсистем і можуть скористатися спільними можливостями запуску.

Стрімко зростає група малих космічних апаратів, що використовує переваги мініатюризованої електроніки та стандартизованих платформ. До неї входять:

– мінісупутники (10 – 100 кг);

– мікросупутники (10 – 100 кг);

– наносупутники (1 – 10 кг);

– пікосупутники (100 г – 1 кг);

– фемтосупутники (<100 г).

Мініатюризовані супутники значно дешевші у розробці та запуску, що забезпечує швидші цикли розробки та відкриває доступ до космосу університетам, новим компаніям і космічним програмам країн, які лише розвивають свої можливості у цій сфері. Серед цих категорій найбільшу увагу привертають наносупутники, що поєднують оптимальний баланс між розміром, вартістю та функціональною спроможністю. У класі наносупутників ключову роль відіграє платформа CubeSat.

На тлі поступової еволюції супутникових технологій відбулась без перебільшення революція – в 1999 році професори Роберт Твіггс (Стендфордський університет) і Джорді Пуч-Суарі (Каліфорнійський політехнічний державний університет) здійснили важливу та проривну зміну в парадигмі космічних досліджень, розробивши концепт над-дешевого, компактного, 10-сантиметрового наносупутника CubeSat (Рисунок 1.1). Сьогодні CubeSat – це стандартизований підхід до наносупутників, який зруйнував традиційні бар'єри доступу до космосу завдяки безпрецедентному зниженню вартості, часу розробки та операційної складності. З часом концепція CubeSat також еволюціонувала, адже втрутилась в вагові ніші мікросупутників та пікосупутників. Надалі в тексті дисертації «CubeSat» та «наносупутник» є словами-синонімами. Станом на квітень 2025 року, більш ніж 2700 наносупутників успішно виведені на орбіту (Рисунок 1.2).

Головними причинами успіху наносупутників типу CubeSat є:

– стандартизація;

– суттєва різниця в кошторисі та бюджеті проєктів, як з розробки та виготовлення так і виводу на орбіту, у порівнянні з класичними супутниками;

– використання «комерційної з полиці» (англ. Commercial Off-The-Shelf, COTS) електроніки та програмного забезпечення;

– застосування відкритого програмного забезпечення (англ. Open-Source Software, OSS);

– зростання попиту на супутникові дані (зв’язок, ДЗЗ, наука);

– концепція «нового космосу» – NewSpace [1, 2] і участь приватного ектору;

– поява дешевих та частих запусків (Falcon 9 Rideshare, RocketLab Electron);

– стрімке зростання концепції «сузір'їв» наносупутників. Прикладом є успіх компанії Planet Labs PBCз їх сузір'ям PlanetScope [3], що налічує понад 200 наносупутників формату 3U (серія Dove), які забезпечують щоденну повну зйомку земної поверхні, що є недосяжним для традиційних важких супутників ДЗЗ.

Рисунок 1.1 – Наносупутник CubeSat розміром 1U – V-R3X (Стенфордський університет) [4]

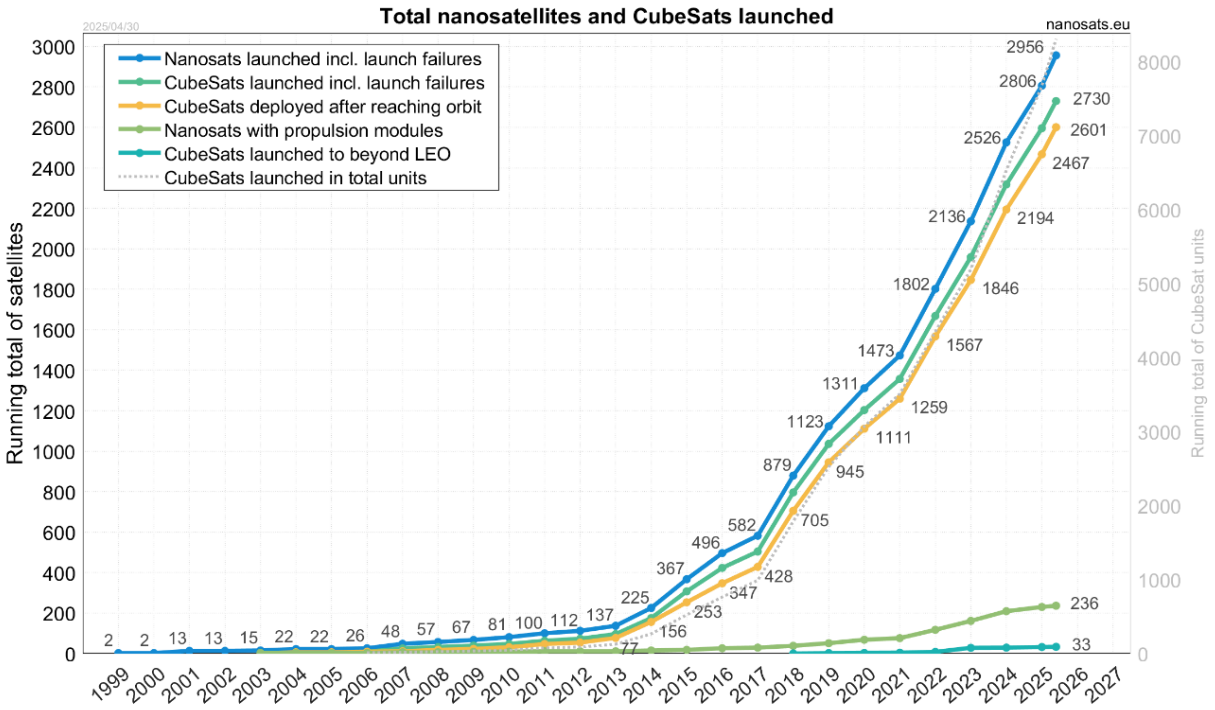

Рисунок 1.2 – Загальна кількість наносупутників класу CubeSat відповідно до бази даних E. Kulu [5, 6], станом на 2025 рік

Використання COTS компонентів, стрімкий розвиток обчислювальних засобів авіоніки та джерел енергії з високою щільністю дозволили космічним апаратам класу CubeSat, станом на 2025 рік, зайняти домінуючу частку за кількістю запусків та темпами розгортання в інженерних, дослідницьких, комерційних та військових галузях [7].

1.1.1 Типова побудова наносупутників CubeSat

Конструкція наносупутника класу CubeSat передбачає наявність кількох основних підсистем авіоніки, які забезпечують його роботу, а також можливість інтеграції корисного навантаження для виконання конкретних наукових, технологічних або комерційних завдань. Основними складовими CubeSat, що відображені на рисунку 1.3, є наступні елементи:

– **структурна підсистема наносупутника** виконує роль несучого каркаса, що забезпечує механічну міцність і захист внутрішніх підсистем під час запуску та експлуатації. Найчастіше каркас виготовляється з алюмінієвих або титанових сплавів, рідше – з вуглепластикових композитів;

– **система електрозабезпечення** (СЕЗ / англ. Electric Power System, EPS) – це мережа постачальників енергії та споживачів, з'єднаних між собою лініями передачі та розподілу. Первинним джерелом енергії є панелі сонячної батареї (БС), встановлені на зовнішніх поверхнях апарата. Вторинне джерело енергії – це хімічна акумуляторна батарея (БХ) з власним блоком керування (англ. Battery Management System, BMS);

– **бортовий обчислювач** (БО / англ. On-Board Computer, OBC) виконує функції керування всіма підсистемами, обробки даних від сенсорів та забезпечення взаємодії з наземною станцією. Його ключові елементи: мікроконтролер (МК) або процесор з низьким енергоспоживанням (ARM, RISC-V, LEON), оперативна та постійна пам’ять (Flash, SDRAM, FRAM, MRAM), системне ПЗ для управління місією тощо;

– **комунікаційна підсистема** (англ. COMmunication Module, COMM) забезпечує обмін даними між наносупутником і наземними станціями або іншими КА;

– **система орієнтації та стабілізації** (англ. Attitude Determination and Control System, ADCS) відповідає за визначення орієнтації супутника та її корекцію;

– **корисне навантаження** (англ. Payload) - основний функційний модуль, що визначає цілі місії. Приклади: наукові прилади – спектрометри, фотометри, магнітометри, камери для завдань ДЗЗ; технологічні експерименти – нові сенсори, системи зв’язку чи матеріали, системи зв’язку для інтернету речей (англ. Internet of Things, IoT) з низької навколоземної орбіти (ННО).

– **теплова підсистема** забезпечує підтримання робочих температур усіх компонентів. У CubeSat зазвичай відсутня активна терморегуляція, і застосовуються пасивні методи: теплові екрани, багатошарові ізоляційні покриття (англ. Multi-Layer Isolation, MLI), теплопровідні елементи; активні методи (рідко) – електронагрівачі або пельтьє-елементи.

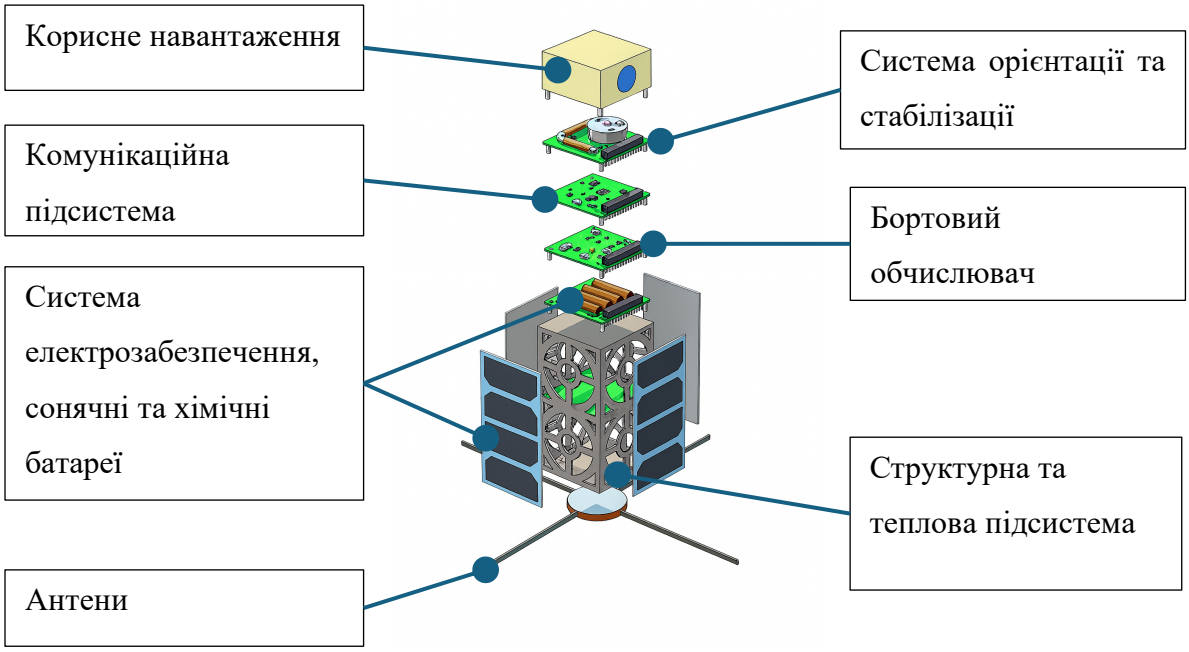

Рисунок 1.3 – Типова побудова наносупутника CubeSat розміром 3U та більше

Усі підсистеми авіоніки CubeSat працюють у тісній взаємодії. Енергетична підсистема живить усі модулі, БО координує їхню роботу, система орієнтації орієнтує супутник для коректної роботи корисного навантаження та антен, а система телекомунікацій забезпечує зв'язок із Землею. Злагоджена, ефективна, надійна та тривала робота цих підсистем гарантує успішність місії.

БО є основним керуючим елементом наносупутника. Для найменших за типорозміром наносупутників (1U, 1,5U) – БО виконує також функції інших підсистем (COM, ADCS, Payload), шляхом виконання ПЗ яке комбінує функції БО з функціями інших підсистем. В такій конфігурації, БО зазвичай інтегрує додаткові апаратні складові, а такий наносупутник можна назвати одно-модульним наносупутником.

1.1.2 Стандартизація наносупутників CubeSat

Стандартизований форм-фактор наносуптуників CubeSat, побудований навколо уніфікованих «блоків» (Unit, U) розміром $10 \times 10 \times 10$ см^3, історично задав спільну геометрію, механічні й масові обмеження для конструкторів і постачальників та розробників компонентів. У результаті розроблення й

розгортання місій перестали бути унікальними «кустарними» проєктами, а перетворилися на процес інтеграції сумісних модулів та програмно-апаратних блоків [8]. Галузева практика підтверджує - саме стандарти та «каталоги» сумісних підсистем скорочують час розробки та підготовки до запуску («time-to-orbit») і суттєво знижують ризики [9]. Ідею COTS підходів (у т.ч. в польотному ПЗ) активно просувають і постачальники, наприклад Bright Ascension, як технологічний наслідок стандартизації CubeSat-ринку [10].

Глобальний ринок CubeSat, оцінений агенцією Straits Research у 423,83 мільйона доларів США у 2024 році [11], яскраво відображає переростання технології від нішевої освітньої платформи до основного інструменту космічної індустрії. Позитивна динаміка зберігається і надалі: за прогнозами Mordor Intelligence, у 2025 році та в подальшому період до 2029 року очікується середньорічне зростання (англ. Combined Annual Growth Rate, CAGR) на рівні понад 15 - 20 % [12], що зумовлено масовим розгортанням комерційних сузір'їв для ДЗЗ та інтернету речей (IoT), або військових проєктів на кшталт Golden Dome [13].

Одним з ключових факторів успіху наносупутників є стандартизація технології побудови. Специфікація на побудову наносупутників (англ. CubeSat Design Specification, CDS) і механізми відокремлення наносупутників від Каліфорнійського Політехнічного Державного Університету (California Polytechnic State University, Cal Poly) визначає габарити, допуски, рейки/напрямні, обмеження щодо центру мас, заборонені зони тощо. Стандарт еволюціонує з потребами галузі, в 2022 році запропоновано проєкт CDS версії 14.1 з низкою оновлень [14].

З погляду класифікації розмірів, сьогодні типовими є 1U, 1.5U, 2U, 3U, 6U, 12U (рисунок 1.4), а також розширені конфігурації до 16U (і навіть 27U у спеціалізованих механізмах відокремлення). Європейська Космічна Агенція (англ. European Space Agency, ESA) визначає типовий діапазон мас ≈ 2–36 кг для нано- та мікросупутників CubeSat-класу, з прикладом 3U (10 см × 10 см × 34 см,

масою до 6 кг). Це емпірично підтверджує, що місії в діапазоні 1U–16U здатні виконувати задачі, котрі раніше вимагали значно більших платформ.

Стандарти інтерфейсу з ракетоносієм зазначені у специфікаціях механізмів відокремлення. Історично базовим є Полі-пікосупутниковий орбітальний механізм відокремлення (англ. Poly-Picosatellite Orbital Deployer, P-POD) від Cal Poly – канонічний «контейнер» на три U (або еквівалент), який задав вимоги до напрямних, пружин/замків, допусків на деформації тощо. Саме P-POD став «механічною конституцією» для безпечного відділення з платформ-носіїв [15].

Для виведення з Міжнародної Космічної Станції (МКС) широковживаним є механізм відокремлення CubeSat компанії NanoRacks (англ. NanoRacks CubeSat Deployer, NRCSD), де Документ визначення інтерфейсу (англ. Interface Definition Document, IDD) задає мінімальні вимоги сумісності місії із системою механізму відокремлення, безпекові аспекти МКС, набір даних для верифікації тощо. Наявність IDD стандартизує й операційні інтерфейси: вимоги з безпеки, процедури інтеграції, цикли звітності.

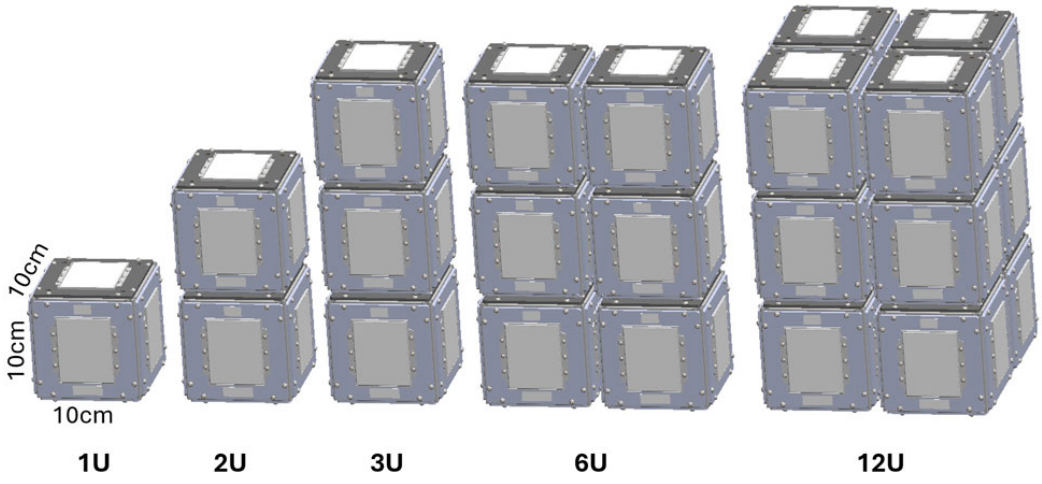

Рисунок 1.4 – Типові розміри наносупутників CubeSat

Усе це створює ланцюжок механічної сумісності: від кубічних рейок і габаритів до «мови» взаємодії із засобами виводу на орбіту – механізмами відокремлення. Підрядники механізмів відокремлення (NRCSD, P-POD, чотиримісного пристрою механізму відокремлення супутників (англ. QuadPack satellite deployer, QuadPack), японський модульний механізм відокремлення

малих супутників (англ. Japanese Small Satellite Orbital Deployer, J-SSOD) тощо) підтримують ті самі геометричні та механічні специфікації, що уможливлює сумісність апаратури між провайдерами пусків. Оглядові матеріали національного управління з аеронавтики та дослідження космічного простору США (англ. National Aeronautics and Space Administration, NASA), а саме стандарти перевірки та валідації космічної техніки NASA (англ. General Environmental Verification Standard, GEVS), та ESA додатково структурують розміри, масові межі й практики кваліфікації.

Рисунок 1.5 – Український наносупутник PolyITAN-HP-30 (справа) та його механізм відділення на чотири супутника (зліва) [16]

Стандартом для внутрішньої інтеграції CubeSat став PC/104-сумісний стек плат (варіанти CubeSat Kit Bus/Pumpkin) [17]. Таке уніфіковане «шасі» для плат (БО, системи визначення та керування орієнтацією, живлення, радіо) надало можливість створення модулів за принципом «підключай та використовуй» (англ. plug-and-play): достатньо витримати вимоги електромеханічної стиковки, висоту та розводку кабелів й мінімальні електричні вимоги. Цей ринок компонентів – прямий наслідок стандартизації: виробники проєктують «сумісні

з PC/104 або CubeSat Kit» блоки, а команди просто компонують стек. Такі компоненти сформували ринкову пропозицію COTS.

На рівні внутрішньої мережевої взаємодії платформ і підсистем свою нішу займає космічний протокол наносупутників (англ. CubeSat Space Protocol, CSP) – легка стекова реалізація для обміну даними та командами між вузлами авіоніки. CSP не замінює стандартів зв'язку «космос-Земля», але дисциплінує внутрішні взаємодії в межах борту, що також знижує бар'єр входу для нових команд та інтеграторів апаратного забезпечення наносупутників. Для більш складних та великих наносупутників існує більш складна реалізація протоколу взаємодії засобів авіоніки як на борту, так і у каналі «космос-Земля» – SpaceWire, який просто інтегрується в протокол консультативного комітету з систем космічних даних (англ. Consultative Committee for Space Data Systems, CCSDS) та підпадає під стандартизацію європейської кооперації з питань космічної стандартизації (англ. European Cooperation for Space Standardization, ECSS), а саме стандарту ECSS-E-ST-50-12C.

Забезпечення якості наносупутників досягнуто завдяки наявності фахових мета-стандартів процесу побудови наносупутників, які регламентують не стільки геометричні, електричні чи програмні інтерфейси апаратів, скільки процес їхнього створення, верифікації та управління життєвим циклом. На відміну від CDS чи протоколів CCSDS, які визначають фізичні параметри й технічну сумісність, мета-стандарти описують методологію роботи: від збору та трасування вимог, через поетапну валідацію та випробування, до ведення документації та управління змінами.

У Європі та ESA це насамперед стандарти організації ECSS:

1) ECSS-E-ST-10C Rev. 1 – загальні вимоги до системної інженерії (трасування вимог, верифікація/валідація, технічні базові лінії), із таблицями «предтейлорингу» під тип продукту;

2) ECSS-Q-ST-20C Rev. 2 – вимоги до програм якості (QA) для космічних продуктів - за потреби передбачає модифікацію під малі місії;

3) ECSS-E-ST-40C – загальні вимоги до інженерії ПЗ - у парі з ECSS-Q-ST-80C для забезпечення якості ПЗ. Навіть коли команди використовують COTS SDK ПЗ (як-от у деяких місіях на базі Bright Ascension), цілісність процесів визначається саме цими нормами;

4) Блок стандартів ECSS-Q-ST-70-xx (матеріали/процеси, кліматичні випробування, адитивні технології) – база для конструктивної сумісності та повторюваності виготовлення.

У США NASA системно використовує GEVS, а саме GSFC-STD-7000B – довідник рівнів/методик середовищних випробувань (вібрація, акустика, термовакуум тощо) [18]. Для CubeSat цей документ є «каталогом» та настановою з обрання опцій випробувань, а також спосіб уніфікувати вимоги замовника з можливостями лабораторій.

На міждержавному рівні стандарт «Космічні системи – Кваліфікаційні та приймальні випробування малих космічних апаратів і їхніх модулів» ISO 19683:2017 (Edition 1) – "Space systems – Design qualification and acceptance tests of small spacecraft and units" [19] визначає методи та вимоги для випробувань з метою кваліфікації конструкції та/або приймальних тестів малих космічних апаратів та блоків (qualification/acceptance), із фокусом на усунення «дитячої смертності» виробів після виведення на орбіту.

Фаховою практикою для переважної більшості наносупутників є використання комунікаційних стандартів від CCSDS, що є спільною «мовою» польоту та комунікацією орбіта-Земля. Для уніфікації, зменшення вартості та ризиків при розробці апаратури наземного сегмента, команди розробників наносупутників широко використовують наступні стандарти CCSDS:

1) Протокол космічних пакетів CCSDS (Space Packet Protocol) що розроблений для забезпечення сумісності систем зв'язку та обробки даних у космічних місіях. Фактично надає аплікаційно-мережеву уніфікацію пакування даних, актуальна редакція 133.0-B-2 (оновлена у 2020 р.) [20], або аналог від міжнародної організації стандартів (ISO) – ISO 22646 [21]. Використовується для:

– передачі даних космічних застосунків. Визначає єдиний спосіб передачі будь-яких прикладних даних (таких як телеметрія, команди, файли) між космічним апаратом та наземною системою, а також між двома космічними апаратами;

– формату даних. Стандартизує структуру, в яку загортаються ці дані, що полегшує їхню ідентифікацію та маршрутизацію.

2) Протоколи телеметрії та команд CCSDS (TM/TC – Telemetry/ Telecommand) Space Data Link Protocols – канальні стандарти для телеметрії та синхронізації даних (TM, 132.0-B-3/-B-5) і телекоманд (TC, 232.0-B-4), із прив’язкою до підшару синхронізації та кодування.

Перевага стандартизації комунікаційних протоколів для наносупутників потрійна:

– взаємна сумісність із наявною інфраструктурою та ПЗ наземних станцій;

– повторне використання перевірених комунікаційних бібліотек (open-source/комерційних), що скорочує час інтеграції й тестів. Чисельні огляди та публікації ESA [22, 23] підкреслювали економію на тестовому/наземному обладнанні завдяки CCSDS-сумісності;

– впровадження концепції «наземна станція як сервіс» (англ. Ground Station as a Service, GSaaS), яскравим прикладом якої є ініціатива AWS Ground Station [24], дозволяє операторам наносупутників відмовитися від капітальних витрат на розбудову власної антенної мережі, переходячи на модель оплати за хвилину сеансу зв'язку (pay-per-minute).

1.1.3 Умови експлуатації БО наносупутників CubeSat

Наступні фактори космічного простору визначають вимоги до БО наносупутника.

1. Космічна радіація (протони, електрони, іонізовані частинки, гамма-випромінювання) може спричиняти короткочасні збої (Single Event Upset (SEU), Single Event Latchup (SEL)), накопичення радіаційної дози веде до повної відмови електроніки.

2. Вплив температури. На освітленій та тіньовій ділянках орбіти відбуваються суттєві температурні перепади від мінус 20 °C до +60 °C, що потребує використання компонентів промислового або автомобільного класу та ретельного теплового моделювання.

3. Режими живлення та енергобаланс. БО повинен працювати в умовах обмеженого енергетичного бюджету, який динамічно змінюється залежно від освітленості орбіти. Постійне живлення забезпечується через сумісну роботу сонячної та хімічної батарей. БО повинен застосовувати техніки енергозбереження для мінімізації споживаної електричної енергії в умовах потреб місії.

4. Обмеження маси, габаритів, інтерфейсів та форм-фактору. Стандарт CubeSat (CDS 1.3/1.4) [14] визначає фізичні обмеження наносупутника. БО найчастіше реалізується на платі стандарту PC/104 або його похідних, що обмежує доступний простір для екранування та роз'ємів. Необхідність взаємодії з різноманітними підсистемами супутника (COM, ADCS, EPS, Payload) вимагає від БО підтримки множинних та надійних шин зв'язку: I^2C (для повільних сенсорів), SPI/UART (для іншої авіоніки) та CAN (для високої надійності та розподіленої архітектури).

5. Вібраційно-акустичні навантаження під час старту ракети-носія та виведення на орбіту є важливим фактором, який суттєво впливає на надійність БО. Для запобігання відмови використовують компоненти з підвищеною механічною міцністю, роз’єми фіксуються, друкована плата лакується, застосовуються силові точки кріплення.

6. Стійкість до електромагнітних збурень та завад. В умовах щільної компоновки наносупутника виникає високий рівень внутрішніх електромагнітних завад. Чутливі сенсори (зокрема ADCS) погано переносять шум, що створюється цифровими шинами, DC-DC конверторами тощо.

1.1.4 Використання комерційно доступних апаратних компонентів

За останні 30 років використання технології CubeSat та культури прийняття можливих ризиків втрати місій під час виводу на орбіту таких наносупутників, з’явилась та поширилась нова культура – культура інтеграції COTS компонентів у авіоніку наносупутників. Ця культура стала глобальною зміною інженерної парадигми побудови КА та дозволила використовувати COTS компоненти, суттєво зменшуючи як складність та тривалість розробки, так і складнощі з власного виготовлення та вирішення проблем ланцюгів поставок компонентів [25, 26].

Місія SBUDNIC [27] є прикладом COTS підходу, успішно тестуючи Arduino Nano та інші легкодоступні технології в космічному середовищі, використовуючи прості апаратні та програмні рішення з відкритим кодом. Така доступність дозволяє університетським командам, малим компаніям та країнам, що розвиваються, змістовно брати участь у космічній діяльності. Така демократизація космічних технологій була б недосяжною в рамках традиційних парадигм розробки супутників.

Якщо спочатку інженерний підхід використання COTS компонентів переважно використовувався у академічних проєктах, то станом на 2025 рік, такий підхід отримав широке розповсюдження у NASA та ESA, про що свідчать технологічні звіти [28, 29] агенцій.

Станом на 2025 рік, існує декілька десятків постачальників COTS компонентів для побудови наносупутників [30, 31]. Ринок COTS-компонентів для авіоніки та енергетичних систем досяг рівня, коли закупівля здійснюється через спеціалізовані глобальні маркетплейси та агрегатори, такі як Satsearch, CubeSatShop або CubeSat Market [32, 33, 34, 35, 36]. Це свідчить про перехід від штучного виробництва компонентів до індустріальної серійності, що радикально знижує бар'єр входу в космічну галузь.

Додатковою мотивацією у використанні COTS компонентів є суттєві обмеження на придбання та використання спеціалізованих, радіаційно-стійких

компонентів наносупутників, які доволі часто підпадають під експортний контроль та виходять на межі кошторису наукових та академічних проєктів.

Окрім COTS компонентів, також стрімко розвивається підхід використання доступних з полиці компонентів, що можуть бути модифіковані (Modifiable Off-the-shelf Components, MOTS) та дозволяють вносити невеликі фінальні зміни у побудову таких компонентів під час їх інтеграції до складу наносупутника [37].

1.1.5 Використання відкритого програмного забезпечення

За останні десять років використання програмного забезпечення та фреймворків з відкритим кодом у галузі наносупутників зросло стрімкими темпами [38], що зумовлено потребою в економічно ефективних, гнучких та орієнтованих на співпрацю рішеннях.

Сучасна наукова література фіксує тенденцію переходу від закритих, документно-орієнтованих підходів до відкритих, модельно-орієнтованих та модульних методів інженерного проєктування (англ. Model-Based Systems Engineering, MBSE) [39, 40], що забезпечує скорочення циклів розробки, підвищення прозорості та поліпшення надійності місій [41]. Низка платформ з відкритим програмним кодом, що пропонують багаторазові, адаптивні та орієнтовані на спільноту рішення для бортової обробки даних, телеметрії та проєктування систем, набули ключового значення у процесі розроблення CubeSat:

- FloripaSat-2 [42] – відкрита універсальна платформа для CubeSat з модульними системами бортової обробки даних, телеметрії та енергоживлення;
- Nanospace [39] та Nanostar [43] – веб-орієнтовані відкриті фреймворки, що підтримують паралельне інженерне проєктування та методологію системної MBSE-інженерії;
- KubOS – інтегрована платформа з відкритим кодом на базі Linux, яка використовує сервіс-орієнтовану архітектуру та протокол GraphQL для

уніфікації взаємодії між польотним ПЗ та апаратними підсистемами наносупутників [44];

- OpenSatKit та NOS3 – інструменти відкритого коду, підтримані NASA, для моделювання та аналізу місій у межах сузір'їв супутників CubeSat [46];

- PyCubed [46] та VERTECS CCB [47] – відкриті платформи для авіоніки та інтерфейсу корисного навантаження, які забезпечують швидке прототипування та розширені можливості обробки даних;

- NASA F'Prime [48, 49] – компонентно-орієнтований фреймворк, розроблений Лабораторією реактивного руху (JPL) NASA для швидкої розробки та розгортання вбудованого програмного забезпечення (embedded software). Він спеціально призначений для використання у критичних до відмов космічних місіях і включає інструменти для моделювання, тестування та автоматичної генерації коду. F'Prime орієнтований на високу надійність та модульність, що дозволяє інженерам легко інтегрувати та повторно використовувати програмні компоненти. Первинно був розроблений для наносупутників та малих роботів, що керуються вбудованими системами, що базуються на МК. Є адаптація для Linux, ОСРЧ VxWorks, RTEMS, та FreeRTOS [50, 51, 52, 53];

- NASA Core Flight System (cFS) – це відкрита, переносима та багаторазова архітектура польотного програмного забезпечення, розроблена Центром космічних польотів імені Годдарда NASA. Вона складається з трьох основних шарів: Core Flight Executive (cFE), рівень абстракції від операційної системи (OSAL) та Набори Бібліотек (CFS Applications). cFS забезпечує стандартизоване середовище для розробки додатків, що підвищує їхню надійність і дозволяє повторно використовувати програмні компоненти у різних місіях без значних змін. Є достатньо складною архітектурою, що первинно була розроблена для більших супутників та вимагає суттєвих апаратних обчислювальних ресурсів. Адапатована для Linux-подібних операційних систем [54, 55, 56, 57];

- Savior Cordet, більше відомий як CORDET (Component Oriented Development Techniques), що поєднується з архітектурною ініціативою ЄКА –

SAVIOR (Space Avionics Open Interface aRchitecture). CORDET це компонентно-орієнтована архітектура та платформа для розроблення бортового програмного забезпечення, створена P&P Software GmbH у співпраці з ESA. Він розроблений для реалізації сервісно-орієнтованих додатків (Service-Oriented Applications) і суворо дотримується європейського стандарту Packet Utilization Standard (PUS). CORDET забезпечує високу модульність та легкість повторного використання коду, що критично важливо для європейських космічних місій, включаючи навантаження для супутників CHEOPS та SMILE, де його реалізація була підтверджена. [58, 59, 60].

В роботах [26, 61, 62] було виконане порівняння наявних фреймворків для побудови БПЗ наносупутників та були сформульовані наступні висновки:

- F′ та cFS найбільш близькі до сучасних вимог та умов використання БО наносупутника, типових МК класу ARM [63], та його ПЗ, оскільки спеціально проєктувалися як компонентні космічні фреймворки з підтримкою БО, FDIR та повторного використання на місіях класу CubeSat;
- ROS/ROS2 доцільно розглядати як прикладне ПЗ середнього рівня та загального призначення з потужними комунікаційними можливостями, але без вбудованих космічних сервісів і з менш вираженою підтримкою жорсткого реального часу та вимог надійності, характерних для космічних стандартів;
- SAVOIR/CORDET радше задає еталонну архітектуру й процес, а не конкретну реалізацію, і може бути використаний як «методологічна рамка» для обґрунтування архітектури твого бортового ПЗ та його відповідності європейським підходам (OSRA, ECSS).

1.1.6 Проблеми та виклики наносупутників CubeSat

Як тільки наносупутники стали рушіями інженерного прогресу як в освіті, розробці, проєктуванні, так і в виготовлені космічних апаратів (КА), відбулась суттєва кількість виводів таких супутників на НОЗ. З ростом побудови таких КА на комерційно-доступних компонентах – з’явились проблеми та відмови.

За офіційним звітом NASA [64] за 2000 - 2016 роки, 41.3 % від усіх виведених до ННО наносупутників зазнали повної або часткової втрати місії. Найбільшою проблемою є так звана проблема “смерті по прибуттю” (англ. Death on Arrival, DoA), яка також має другу назву “дитяча смертність” (англ. Infant mortality). Рівень "DOA" для університетських супутників історично сягав ~30 – 40 %. В проміжку часу від 2003 до 2018 років, як показує рисунок 1.6, цей показних знизився з 67 % до 23,3 %, але продовжує утримуватись в інтервалі 15 - 25 % [65]

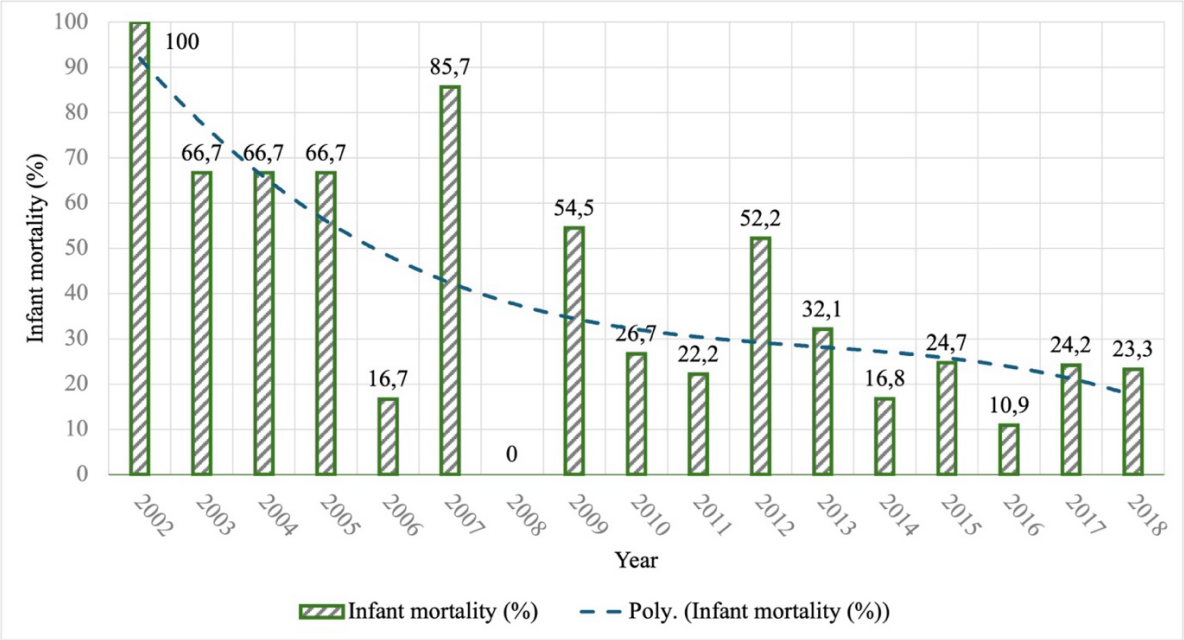

Рисунок 1.6 – Відсоток відмов класу «дитяча смертність» для наносупутників виведених на НОЗ у 2002-2018 рр. (звіт NASA [64]).

Базуючись на статистиці відмов наносупутників, можна зазначити наступне: приблизно половина відмов виникла з невідомих причин (тобто немає можливості отримати достовірні дані), де інша половина вказує на прямі відмови системи енергозабезпечення. Другою за кількістю відмов, є відмова бортового обчислювача (БО) [66] та підсистеми комунікації, що на 30-й день місії складають 20 % та 16 % відповідно, де підсистема енергозабезпечення складає 44 % відмов (рисунок 1.7).

Аналіз експлуатації наносупутників у 2010-2020 роках [67], демонструє зниження надійності наносупутників, що пов’язується авторами з більшою кількістю застосування COTS компонентів, виводом на орбіту наносупутників більшою кількістю країн та установ з меншим досвідом та складністю обладнання наносупутників що постійно зростає.

Автори [68] також зазначають проблеми в якості за «самовпевненості» інженерних команд, особливо в університетському середовищі, як одну з найвпливовіших нетехнічних (процесних) проблем.

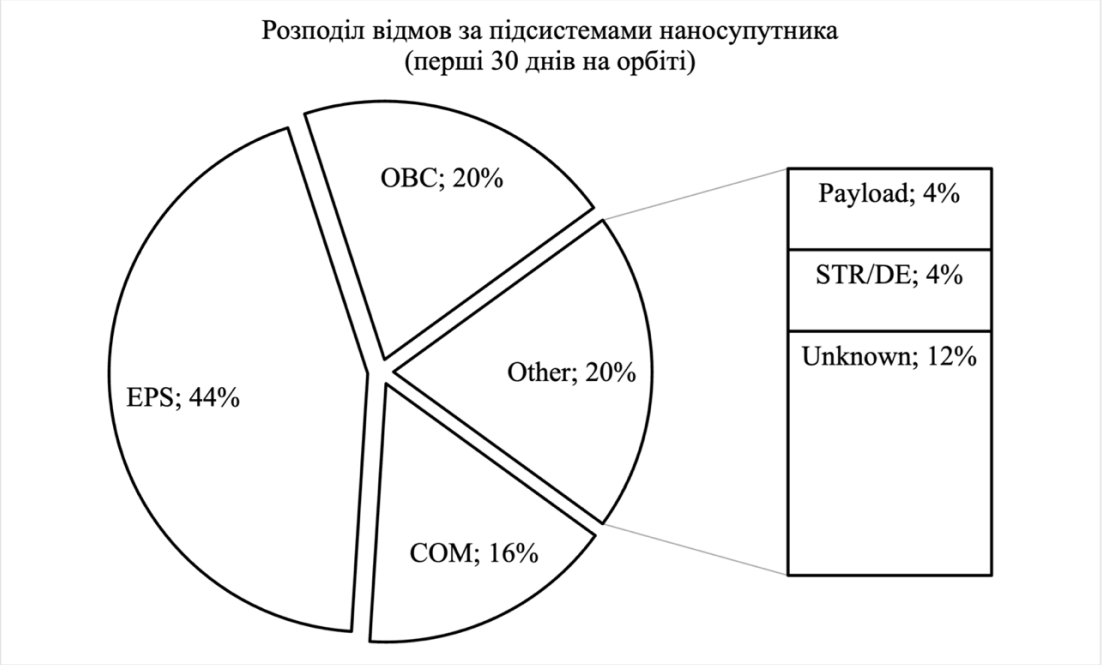

Рисунок 1.7 – Розподіл відмов за підсистемами для наносупутників що були запущені в 2000-2016 р. [68]

Нажаль Україна має свій власний досвід втрати мікросупутника Січ-2-30 [69] та наносупутника PolyITAN-HP-30 [16] з причини неузгодженості алгоритмів керування бортовими підсистемами в умовах обмеженого енергобюджету. Сукупна фінансова втрата для країни склала 245 млн. грн. на виробництво та розробку та 2 млн. доларів США на запуск ракетою-носієм Falcon 9 / Space X.

Відсутність радіаційної стійкості є найбільшою проблемою COTS компонентів [70] але у порівнянні з кошторисом та тривалістю адаптації або

радіаційно-стійкої авіоніки, або власної розробки, все одно дає суттєві переваги для розробників та науковців.

Огляд [71] показує статистику відмов для різних класів (1U, 3U тощо). Це дозволяє стверджувати, що зі зростанням складності CubeSat (перехід від 1U до 3U/6U) старих методів розробки стає недостатньо, і виникає потреба в нових відкритих стандартах.

1.1.7 Сучасні тенденції розвитку та проблеми наносупутників CubeSat

Таким чином, відбулась фундаментальна зміна парадигми космічних досліджень, що характеризується переходом від одиничних, високовартісних державних місій до масового запуску наносупутників, зокрема формату CubeSat. Стандартизація платформ, мініатюризація електроніки та поява доступних послуг запуску забезпечили експоненційне зростання кількості апаратів на орбіті. Наносупутники еволюціонували з інструменту навчання в ефективні засоби для вирішення комерційних, наукових та оборонних завдань, зокрема в складі супутникових сузір'їв.

Уніфікація габаритів та інтерфейсів (відповідно до специфікацій CDS, фахових стандартів ECSS та стеків CCSDS) забезпечила критичні переваги перед традиційними космічними програмами, та забезпечила:

- високу швидкість розробки: завдяки зняттю «вузьких місць» інтеграції та наявності готових платформ, цикл «від задуму до запуску» скоротився з 5-7 років до 12-24 місяців;
- низьку вартість: зниження «ринкової невизначеності» та конкуренція постачальників компонентів COTS призвели до тисячократного зменшення вартості створення супутника (від сотень мільйонів до сотень тисяч доларів) та радикального зниження ціни запуску завдяки програмам групового виводу на ННО (rideshare);
- масштабованість знань: формування спільного фахового лексикону та протоколів взаємодії та відкритих рішень, дозволило масштабувати знання та

спростити міжнародну кооперацію між університетами, стартапами та космічними агентствами.

Уніфікація габаритів та інтерфейсів сформувала глобальну індустрію та ринок уніфікованих COTS підсистем авіоніки, які інтегрують сучасні обчислювальні потужності в малий форм-фактор.

У відповідь на зростання складності систем спостерігається перехід від «документо-орієнтованої» розробки та пропрієтарного коду до використання відкритих програмних платформ (cFS, F’, KubOS) та методів модель-орієнтованої системної інженерії (MBSE). Застосування перевірених архітектурних рішень та фреймворків дозволяє уніфікувати взаємодію між підсистемами, підвищити якість коду та нівелювати ризики, пов'язані з «людським фактором» та брак досвіду розробників. Програмна платформа F’ виглядає найбільш перспективною та обрана для подальшої роботи.

1.2 Систематизований огляд наукових публікацій, присвячених апаратно-програмному забезпеченню бортових обчислювачів безпілотної аерокосмічної техніки

Кількість наукових публікацій, присвячених тільки апаратно-програмному забезпеченню бортових комп'ютерів наносупутників CubeSat стабільно зростала щороку за останні 10 років (рисунок 1.8). Було розглянуто публікації, які оголошують метою - надання огляду літератури в тематиці апаратно-програмного забезпечення бортових комп'ютерів наносупутників CubeSat, використовуючи в назві, або анотації слова та словосполучення: “Literature review”, “Survey”, “State-of-the-art review”, “Meta-analysis”, “Systematic review”.

Проаналізовані огляди літератури [26, 72, 73, 74, 75] мають достатньо недоліків. Головним з них є те, що тільки публікації [26, 75] – єдині, що претендують на структурований або частково систематизований літературний огляд. Решта - це технічні дослідження або вузькотематичні огляди без формалізації методології огляду. Не менш важливим є фрагментарність розгляду - жодна з робіт не пропонує комплексного огляду всіх критичних аспектів

архітектури, ПЗ, енергоспоживання та безпеки бортового обчислювача CubeSat як цілісної системи. Таким чином, треба виконати інтегрований та систематизований огляд літератури за темою АЗ та ПЗ БО, що враховує комплексність проблематики.

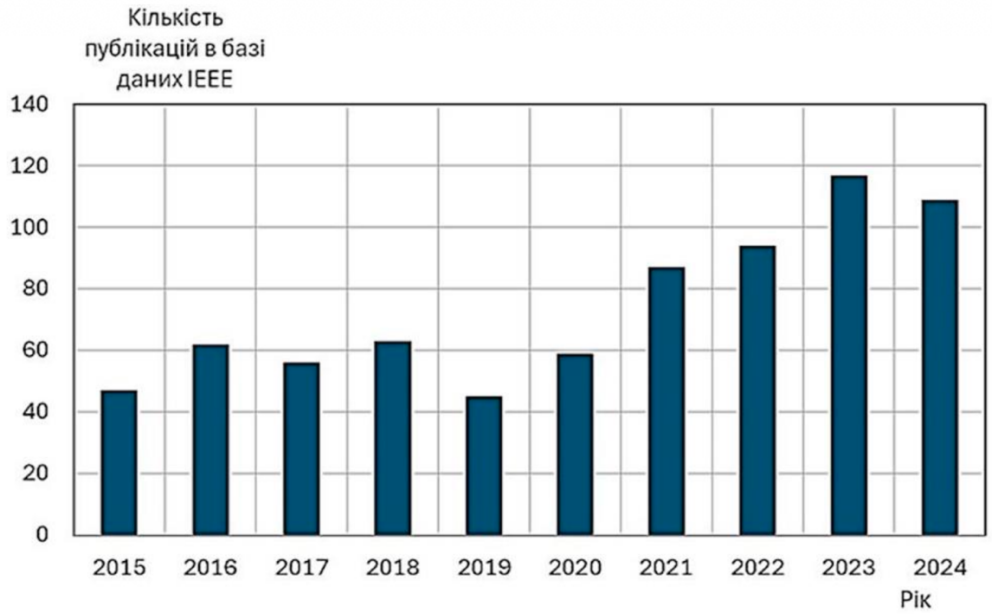

Рисунок 1.8 – Кількість публікацій IEEE Xplore, присвячених апаратно-програмному забезпеченню бортових обчислювачів наносупутників CubeSat

1.2.1 Технологія систематизованого літературного огляду

Для систематизованого огляду літератури використано технологію PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) [76] з метою актуалізації та отримання системного розуміння щодо напрямів досліджень з АЗ та ПЗ БО наносупутників [77]. Було проаналізовано дослідження за останні десять років для пошуку перспективних напрямів подібних досліджень в найближчому майбутньому. Алгоритм систематизованого огляду літератури розбитий на наступні етапи (фази).

ЕТАП 1. Формулювання теми та питань досліджень

Тема досліджень: Апаратно-програмні рішення для бортових комп’ютерів наносупутників CubeSat: сучасні архітектури, принципи розробки та виклики.

Питання досліджень розділено на дві частини.

1. Загальні дослідницькі питання орієнтовані на отримання бібліометричної інформації про напрями та інтенсивність досліджень в області апаратно-програмного забезпечення БО наносупутників CubeSat та включають наступне.

а) Як змінювалась щорічна кількість наукових публікацій за різними напрямами досліджень?

б) Які кластери наукових публікацій можна визначити шляхом автоматичної кластеризації з застосуванням інструменту програм аналізу та візуалізації графів Gephi [78] та VOSviewer [79], що широко використовується в академічній спільноті для бібліометричного аналізу та візуалізації?

2. Спеціальні дослідницькі питання орієнтовані на отримання змістовної деталізованої інформації про напрями досліджень в області апаратно-програмного забезпечення БО наносупутників CubeSat та включають наступне.

1. Наскільки широко використовується комерційна, не радіаційно-стійка електроніка в сучасних місіях CubeSat, і які типові приклади її застосування?

2. Які архітектурні підходи використовуються в БО CubeSat (FPGA, GPU, RISC-V), і чим зумовлений вибір тієї чи іншої архітектури?

3. Які особливості апаратно-програмних рішень у місіях з підвищеними вимогами до надійності?

4. Які методи забезпечення якості програмного забезпечення та оптимізації енергоспоживання апаратної платформи використовуються у CubeSat?

5. Які операційні системи та відкриті фреймворки використовуються у CubeSat, і чим зумовлена популярність їх застосування?

6. Як впливають сучасні гнучкі технології розробки програмного забезпечення та системна інженерія на основі моделей (MBSE - Model-based systems engineering) на ефективність проєктів CubeSat?

7. Які технології проєктування, тестування та забезпечення якості (MBSE, HIL/MIL/SIL) використовуються на різних етапах створення CubeSat?

8. Які загрози і підходи до кібербезпеки розглядаються у сучасних проєктах CubeSat, зокрема в контексті комерційних місій?

ЕТАП 2. Формування переліку наукових публікацій для подальшого експертного та автоматичного аналізу з використанням методології PRISMA

Методологія PRISMA розрізняє повно доступні статті та статті, які містять лише ключові слова та анотації. Повний текст більшості статей є недоступним, тому в переліку статей для аналізу є всі ідентифіковані статті, незалежно від їх відкритості. Будуть розглядатись статті тільки англійською мовою, так як галузь наносупутників в Україні не отримувала достатньо уваги та фінансування в останні роки.

Пошуковий запит повинен знайти всі статті, в анотації яких містяться одночасно три складові.

1. Щонайменше один з можливих варіантів написання слова «наносупутник»: satellite, nanosatellit*, cubesat.

2. Щонайменше один з можливих варіантів написання слова «бортовий обчислювач»: Computer, OBC (On-Board Computer), C&DH (Command and Data Handling), CDHM (Command and Data Handling Module).

3. Щонайменше один з можливих варіантів написання слів, що визначають етапи розробки, вимоги до програмного забезпечення та апаратної складової: Software, Architecture, Design, Implementation, Development, Real-Time, Radiation-Hardened, Firmware, Embedded, Fault-Tolerant.

При формуванні списку синонімічних в контексті даного дослідження слів зроблено попередній експертний аналіз публікацій та використано IEEE Thesaurus [80] і настанови курсу CubeSat 101 від NASA [81].

Таким чином, пошуковий запит є таким:

(satellite* **OR** nanosatellite* **OR** cubesat*)
AND
(OBC **OR** "On-Board Unit" **OR** "C&DH" **OR** "Command and Data Handling" **OR** Computer)
AND
(Hardware **OR** Software **OR** Architecture **OR** Design **OR** Implementation **OR** Development **OR** "Real-Time" **OR** "Radiation-Hardened" **OR** Firmware **OR** Embedded **OR** "Fault-Tolerant")

Для пошуку публікацій обрано дві бази даних: IEEE Xplore [82] та Elsevier Scopus / Science Direct [83]. Діапазон пошуку обмежено останніми 10 роками.

Для експертного відбору та фільтрації зайвих статей, що не відповідають темі досліджень, були використані такі тригери фільтрації (стоп-слова другого рівня): ADCS (Attitude Determination and Control System), EPS (Electric Power System), PPT (Power Point Tracking), 5G, 6G, SDR (Software Defined Radio), radio, propulsion, radar, battery, thruster, robot, navigation, IoT (Internet of Things), thermal. Тригери фільтрації – це слова, присутність яких в тексті є ознакою невідповідності публікації пошуковій темі.

В результаті ідентифікації наукових публікацій для систематизованого літературного огляду з використанням технології PRISMA для подальшого аналізу відібрано 203 статті з IEEE Xplore та Science Direct (рисунок 1.9).

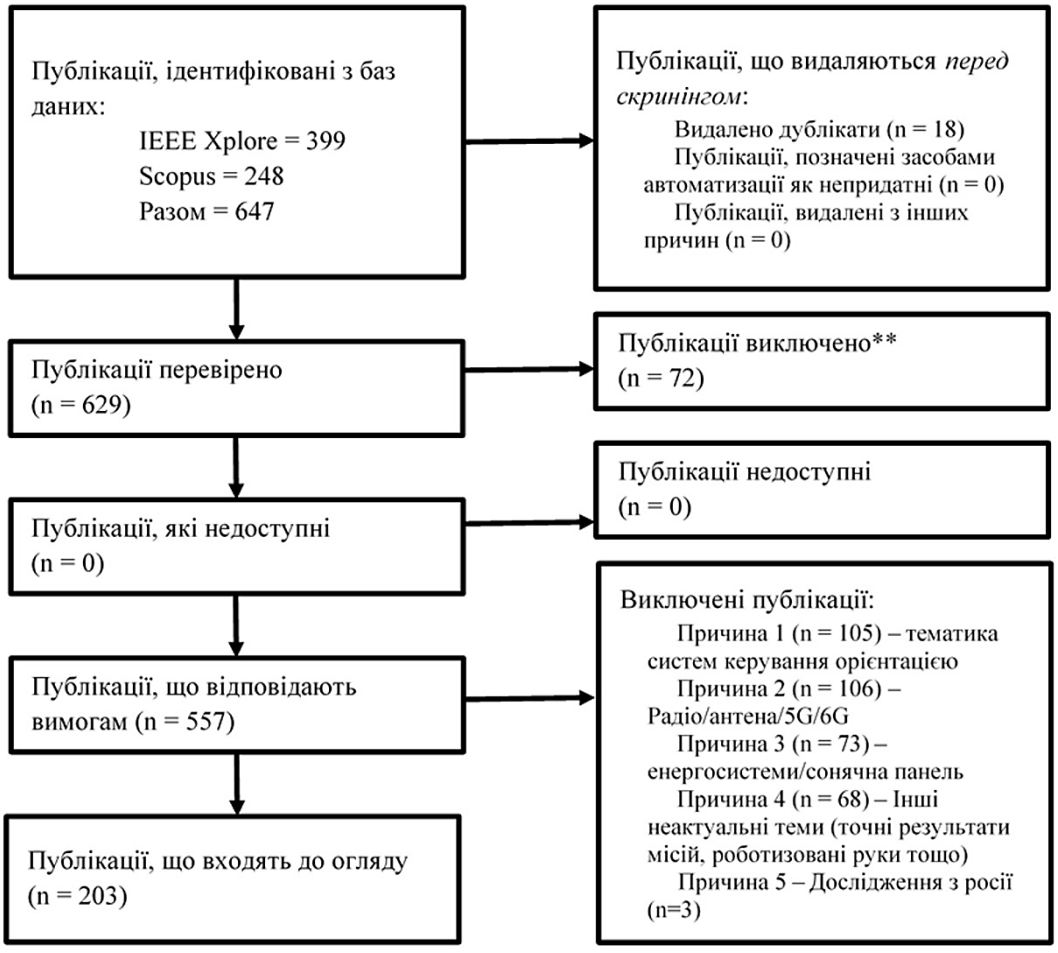

Рисунок 1.9 – Ідентифікація наукових публікацій для систематизованого літературного огляду з використанням технології PRISMA

ЕТАП 3. Автоматична кластеризація публікацій через аналіз ключових слів

Перед кластеризацією виконано нормалізацію ключових слів шляхом уніфікації споріднених термінів через мапінг синонімів з метою скорочення словника та покращення якості тематичного аналізу. VOSviewer дозволяє задати нормалізацію термінів (тезаурус) в спеціальному файлі в текстовому форматі, де вказано, які терміни в контексті дослідження повинні вважатись синонімами. Тезаурус було використано для зменшення кількості вершин графу ключових слів та спрощення подальшого аналізу. Так, наприклад, ключове слово «Power Systems» є узагальненням для словосполучень «Solar panels», «Photovoltaic cells», «Photovoltaic systems», «Batteries», «Power supplies», «Voltage control», «Maximum power point trackers».

В мережевій діаграмі ключових слів (рисунок 1.10) використана наступна нотація: вузли графа – ключові слова, розмір вузла є пропорційним кількості вживання цього ключового слова в відібраних для аналізу публікаціях; ребра - відображають спільне використання пари ключових слів в одній публікації, товщина ребра при цьому – це частотність спільного використання; кольори використано для розмітки автоматично виділених кластерів.

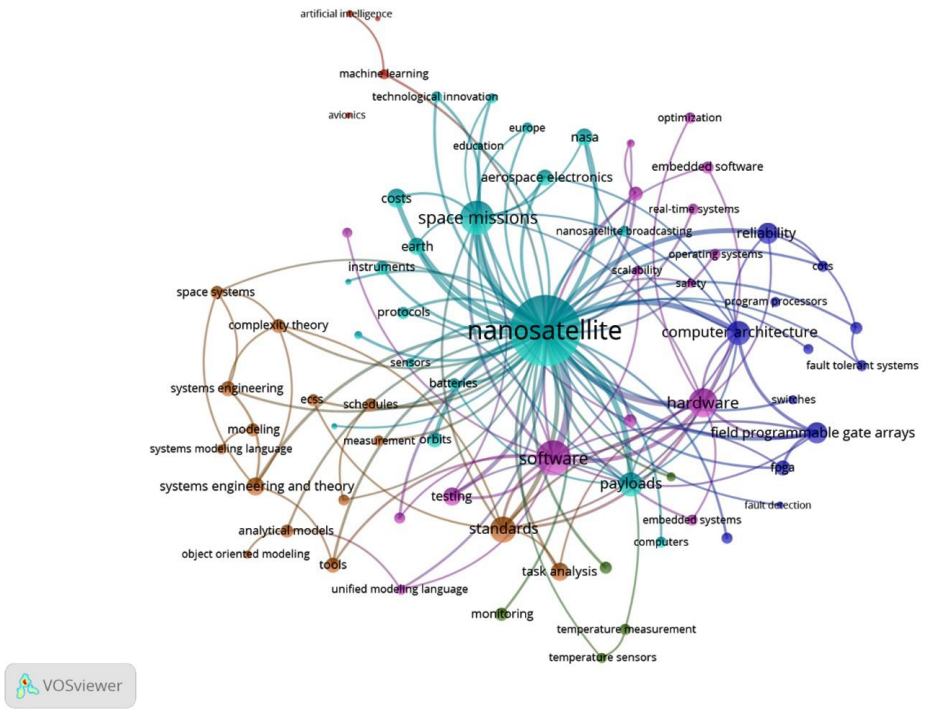

Рисунок 1.10 – Кластеризація ключових слів з використанням ПЗ VOSviewer

Таким чином, програма VOSviewer автоматично виділила наступні шість кластерів публікацій, які визначено через застосування спільних ключових слів (таблиця 1.1).

Таблиця 1.1 Результат кластеризації публікацій через аналіз ключових слів

№	Назва кластера	Кількість публікацій	Ключові слова, що віднесено до кластеру
1	Наносупутники та місії (Nanosatellites & Missions)	50	aerospace electronics, atmospheric modeling, batteries, computational modeling, computers, costs, earth, education, Europe, extraterrestrial measurements, instruments, moon, nanosatellite, NASA, orbits, payloads, protocols, sensors, space missions
2	Апаратні та програмні засоби (Hardware and Software)	31	cyber-physical systems, embedded software, embedded systems, ESA, flight software, hardware, operating systems, optimization, real-time systems, requirements engineering, safety, scalability, software, software architecture, testing, and unified modeling language
3	Системна інженерія та стандарти (System Engineering & Standards)	58	analytical models, complexity theory, ECSS, MBSE, measurement, modeling, object-oriented modeling, project management, schedules, space systems, standards, system engineering (and theory), system modeling language, task analysis
4	Архітектура та надійність комп'ютера (Computer Architecture & Reliability)	44	computer architecture, cots, fault detection, fault tolerance, fault-tolerant systems, field programmable gate arrays (FPGA), microcontrollers, program processors, random access memory (RAM), reliability
5	Моніторинг і телеметрія (Monitoring & Telemetry)	6	monitoring, telemetry, temperature measurements, temperature sensors, visualization
6	Штучний інтелект та машинне навчання (AI/ML)	15	artificial intelligence, avionics, earth observation, machine learning

В якості опорних класів за результатами кластеризації та експертного оцінювання та узгодження, було визначено наступні класи:

1. інженерія бортового програмного забезпечення;

2. апаратні компоненти та системи;

3. проєктування місій та управління;

4. системна архітектура БО;

5. навчання, освітні програми, дисципліни;

6. корисне навантаження.

Докладно результати систематизованого огляду літератури викладено в статті [77, 84].

1.2.2 Аналіз публікацій, присвячених апаратному забезпеченню бортових обчислювачів наносупутників

Подальша класифікація публікацій щодо апаратних компонентів та систем наносупутників, дозволяє ідентифікувати зображені на рисунку 1.11 - три під-класи досліджень.

Рисунок 1.11 – Ідентифіковані підкласи напрямків досліджень за кластером 2

Використання комерційної, не радіаційно стійкої електроніки в бортових комп’ютерах CubeSat, поширюється ще більше, ніж раніше, включаючи місії великих міжнародних космічних гравців, таких як NASA. Типовою практикою є використання звичайних комерційних процесорів, таких як STM32 від ST Microelectronics, ATSAMxx/PIC32CAxx від Atmel/Microchip або MSP430 від Texas Instruments [31].

Використання подвійної архітектури, що складаються з FPGA і центрального процесора, є стабільно зростаючою тенденцією. У той самий час спостерігається зростання використання подвійних рішень, об'єднаних в одному

фізичному чіпі, таких як MPSoC, Xilinx Zynq і Microchip PolarFire. Ключовою мотивацією для використання архітектури на основі FPGA є їх гнучкість, надійність і продуктивність. Використання графічних процесорів для більш складних обчислювальних завдань набирає популярності, особливо для великих супутників або супутників, де OBC, ADCS і корисне навантаження поєднуються для економії маси та об’єму. Спочатку використання графічних процесорів було популярним вибором для завдань AI/ML при обробці зображень на модулі корисного навантаження, але коли воно було прийнято там, воно поширилося і на рішення OBC. Відкрита архітектура RISC-V стає все більш популярною, але сьогодні вона в основному знаходиться на рівні ядер IPU, завантажених в FPGA (Microchip PolarFire або просто Xilix FPGA загального призначення). Це пов'язано з відсутністю належним чином комерціалізованих фізичних процесорів. Ця тенденція потребує подальшої уваги та досліджень, оскільки RISC-V є потужним та енергоефективним рішенням, значення якого продовжуватиме зростати.

1.2.3 Аналіз публікацій, присвячених програмному забезпеченню бортових обчислювачів наносупутників

Подальша класифікація публікацій щодо програмного забезпечення БО наносупутників, дозволяє ідентифікувати зображені на рисунку 1.12 - три під-класи досліджень.

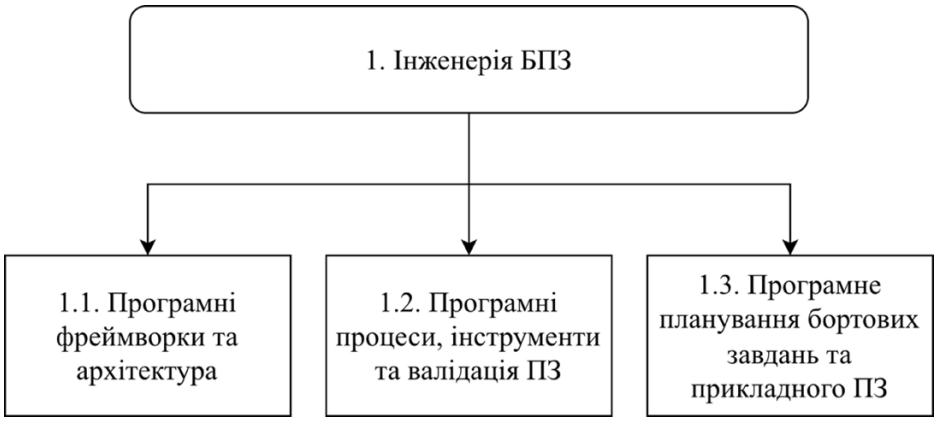

Рисунок 1.12 – Ідентифіковані підкласи досліджень кластеру 1

Використання операційних систем реального часу (FreeRTOS, RTEMS, uCOS (Micrium)) є де-факто галузевим стандартом [26]. ОС дозволяють безпечно реалізовувати більш складні програмні системи і комплекси та розшаровувати ПЗ, як показано на рисунку 1.13, на класичні для програмної інженерії шари ОС. Також помітна тенденція збільшення частки наносупутників, що виконують місії під операційною системою Linux (з розширенням у реальному часі або без нього). Це пов'язано з великою кількістю доступного для використання програмного забезпечення, зростаючою потужністю апаратного забезпечення та зростаючим ступенем інтеграції. Незважаючи на доступність відомих програмних рішень, фреймворків з відкритим вихідним кодом, таких як рішення від NASA cFS і F'Prime і ROS/ROS2, їх використання та розгляд в публікаціях дуже обмежені, що, швидше за все, пов'язано з короткими часовими рамками розробки програмного забезпечення в академічних установах.

Рисунок 1.13 – Типове розгалуження на шари ПЗ наносупутника під керуванням ОСРЧ

Використання програмного забезпечення з відкритим вихідним кодом є галузевим стандартом. У досить великій кількості статей аналізується використання сучасних підходів до розробки програмного забезпечення, таких як Scrum/Agile, та рух до модульності та повторного використання програмного забезпечення. Модульність і повторне використання опиняються в центрі уваги майбутньої розробки програмного забезпечення для польотів і є ключовими методами зниження складності та частоти невдач місій. Багато проєктів і місій

все ще базуються на власних програмних рішеннях, де програмне забезпечення/прошивка розробляється на низькорівневих мовах програмування, таких як C і C++. У той же час, все більше і більше реалізацій використовують мови програмування вищого рівня, такі як Python і Rust.

Віртуалізація і контейнеризація активно розглядаються і досліджуються. У найближчому майбутньому потужні БО, швидше за все, замінять концепцію запуску повного монолітного бінарного програмного образу. Віртуалізація та контейнеризація безпечніші, зрозуміліші для сучасних програмістів та DevOps-інженерів, і загалом надійніші з точки зору стабільності та кібербезпеки. Попередня тенденція використання підходу MBSE та SysML для проектування та розробки бортового програмного забезпечення здається менш актуальною сьогодні темою для досліджень. Це може бути пов'язано із загальною зрілістю технології CubeSat і кращою уніфікацією різних функцій авіоніки CubeSat. Іншими словами, розподіл на те, що робить кожен блок авіоніки CubeSat, став чіткішим і добре задокументованим. Наприкінці 2023 року і надалі інтерес до використання підходу MBSE почав знову зростати. Стійкою тенденцією є використання тестування HIL/MIL/SIL, що призвело до створення відносно складних модульних тестових платформ, призначених для допомоги розробникам CubeSat виконувати процес V&V під час розробки.

Таким чином, кількість щорічних публікацій продовжує лінійно зростати. Це є ознакою добре сформованої, зрілої технології, галузевий інтерес до якої - збільшується. Більшість публікацій певною мірою відноситься до інженерії програмного забезпечення.

Використання COTS компонентів для побудови наносупутників поширюється ще більше, включаючи місії великих міжнародних космічних гравців, таких як NASA, ESA та JAXA.

Виникла окрема гілка «високонадійних» місій, що вимагає використання радіаційно-стійких процесорів, FPGA і загальної електроніки. Такі рішення, зазвичай, призначені для подорожей на більш високі орбіти, ніж ННО. Існує клас «середньо» надійних БО, які дотримуються ключових принципів функційної

безпеки. Багато досліджень проводиться в області використання багатоядерних систем або багатоканальних резервованих БО, як заходи підвищення надійності COTS рішень. Майже всі «звичайні» БО побудовані на COTS рішеннях, що завдяки польотній історії та відпрацьованому ПЗ, є найбільш привабливим та економічно-вигідним підходом до побудови наносупутників.

Енергоспоживання, енергетична ефективність та методи оптимізації БО та ПЗ залишаються відкритими темами досліджень. Різні методи та прийоми використовуються для оптимізації як апаратного забезпечення, та, в основному, бортового програмного забезпечення, яке є ключем до обчислювальної та енергетичної ефективності.

Попри велику кількість досліджень з енергоефективності, не було знайдено жодної статті, що розглядає комплексний підхід до оцінки, планування та моделювання енерговитрат БО. Більшість статей стосуються суто підсистеми електрозабезпечення (EPS), або енергоефективності супутника в цілому. Наявність сучасних програмних фреймворків, як то NASA cFS або F’Prime, та відсутність досліджень щодо побудови енергоефективних рішень на їх основі, є важливим галузевим дослідницьким напрямком.

Підклас 1.3 кластеру 1, адресує дослідження з планування як системних (рівень ОСРЧ), так і прикладних завдань (обробка телеметрії та телекоманд, КН і т.п.) ПЗ БО. Однією з мотивацій розгляду такого планування є дослідження вимог до продуктивності бортового ПЗ в умовах обмеженого енергоспоживання.

В якості подальшого напряму дисертації обирається завдання зменшення енергетичних витрат бортових обчислювачів наносупутників в умовах обмежень систем реального часу та забезпечення реактивності бортових систем, оскільки ця задача є найбільш важливою для вирішення, недостатньо адресована у сучасних дослідженнях, та має переваги системного рішення для будь-яких бортових обчислювальних засобів авіоніки наносупутників.

1.2.4 Аналіз публікацій, присвячених програмному забезпеченню бортових обчислювачів БПЛА

Для аналізу дослідницьких проблем та напрямків дослідження було проведено схожу роботу з систематизованого огляду літератури, але публікація ще знаходиться на стадії передпублікації – preprint.

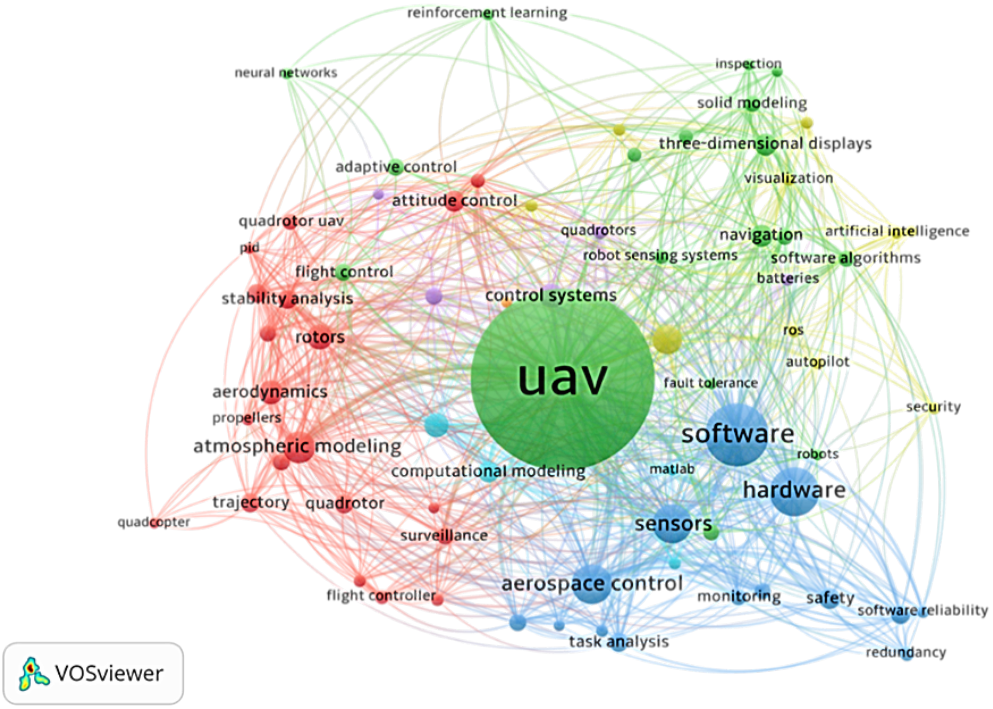

Рисунок 1.14 - Кластеризація ключових слів за темою БПЛА (UAV) з використанням програми VOSviewer

Базуючись на 484 відібраних статтях та їх аналізу за методологією PRISMA та подальшою кластерізацією було ідентифіковано наступні кластери: Кластер 1 - динаміка польоту та системи керування (88 наукових статей і матеріалів конференцій), кластер 2 - захищене автопілотування та керування польотом із використанням методів ШІ/МН та операційних систем реального часу (RTOS) (44 публікації), кластер 3 - надійні апаратно-програмні системи керування польотом (101 публікація), кластер 4 - моделювання систем керування польотом та їх верифікація і валідація (V&V) (36 публікації), кластер 5 - енергоефективність, рекуперація енергії та гібридні силові установки (5

публікацій), кластер 6 - застосування технологій блокчейн у безпілотних літальних апаратах (2 публікації).

Для подальшого розгляду було обрано кластер 3 – надійні апаратно-програмні системи керування польотом. Кластер розглядає дослідницькі напрямки, що зазначені в таблиці 1.2.

Ключовими для подальшого аналізу обрані під-кластери 3.1 та 3.3.

Висновки під-кластеру 3.1 - Оптимізація польотного програмного забезпечення БПЛА та обчислювальне управління енергоспоживанням. Цей підкластер присвячено питанням енергоефективності та результативності авіоніки безпілотних літальних апаратів. Аналіз літератури свідчить, що дана тематика є актуальною для наукової спільноти та промисловості, однак становить відносно невелику частку від загального масиву робіт, що було проаналізовано та відібрано.

Сучасні дослідження акцентують увагу на оптимізації програмного забезпечення та обчислювальному управлінні енергоспоживанням як ключових чинниках підвищення ефективності польоту БПЛА за показниками дальності, висоти та тривалості. Зокрема, аналіз програмного стека ArduCopter демонструє значний потенціал підвищення експлуатаційної ефективності шляхом оптимізації програмних компонентів [85], тоді як адаптивні методи конфігурації, зокрема інструмент «Conf-Adaption», забезпечують подовження тривалості польоту та адаптивність місії на основі аналізу ресурсних залежностей у реальному часі [86].

Проблеми забезпечення вимог реального часу та балансування обчислювального навантаження розглядаються через застосування статичних і динамічних стратегій, реалізованих у розподілених архітектурах на базі шини CAN, що дозволяє усувати вузькі місця використання ресурсів і забезпечувати передбачувану роботу системи [87]. Авторська робота що є частиною цього дослідження також потрапила до аналізу [143].

Таблиця 1.2 Ідентифіковані під-кластери кластера 3 що обрані для подальшого аналізу

Під-кластер	Ключові дослідницькі напрямки	Відібрана кількість публікацій
3.1. Оптимізація польотного програмного забезпечення БПЛА та обчислювальне управління енергоспоживанням	Оптимізація бортового програмного забезпечення БПЛА та управління обчислювальними енергетичними ресурсами з метою підвищення ефективності польоту (тривалості, висоти та дальності)	6
3.2. Кібербезпека та захищений зв'язок у мережах БПЛА	Кібербезпека та захищений зв'язок у мережах безпілотних літальних апаратів, цифрова криміналістика під час розслідування аварій і інцидентів з БПЛА	28
3.3. Програмні фреймворки, контури керування та системи БПЛА на базі ОСРЧ ROS та ін.	Архітектури програмного забезпечення, стеки керування та ROS-орієнтовані системи БПЛА	15
3.4. Валідація вбудованого програмного забезпечення та реалізація стека автопілота	Верифікація та валідація вбудованого програмного забезпечення автономних функцій і програмного стека автопілота	10
3.5. Апаратні платформи БПЛА для сенсорних та обчислювальних застосувань	Передові апаратні платформи БПЛА для сенсорики та обчислень	13
3.6. Відмовостійкі архітектурні рішення систем керування польотом	Методи виявлення та ізоляції відмов, резервовані архітектури систем керування польотом, відмовостійкість у спеціалізованих застосуваннях, прогностичні підходи та методи для систем з підвищеними вимогами до безпеки, оцінювання надійності та аналіз експлуатаційних характеристик, реконструкція даних і надійність сенсорних підсистем, інтегровані модулі виявлення та діагностики відмов.	29

На рівні мініатюризованих систем показовим є приклад нано-БПЛА «Fünfliber-Drone», у якому використано наднизькопотужну систему-на-кристалі PULP SoC, що забезпечує виконання складних нейромережевих алгоритмів автономної навігації в умовах обмеженого енергетичного бюджету [88]. Окремо

підкреслюється критична роль енергоефективності для мереж БПЛА типу FANET/MANET та концепції Internet of Flying Things, де оптимізація обчислювальних і мережних ресурсів є необхідною умовою масштабування таких систем [89, 90].

Висновки під-кластеру 3.3 - Програмні фреймворки, контури керування та системи БПЛА на базі ОСРЧ ROS та ін. Дослідження у сфері безпілотних літальних апаратів зосереджені на розробленні надійних програмних фреймворків, розвинених стеків керування та їх інтеграції з такими платформами, як Robot Operating System (ROS). Ефективні реалізації, зокрема безопераційні (bare-metal) системи керування польотом на базі DSP, забезпечують детерміновану поведінку та оптимальне використання ресурсів у режимах реального часу [91]. Розподілені автономні системи керування, що поєднують ROS, контролери Pixhawk і засоби просторового позиціонування, демонструють низькі затримки обміну та ефективну координацію БПЛА [92].

Окрему увагу приділено адаптивним фреймворкам моніторингу, таким як AMon і GRuM, які використовують предметно-орієнтовані мови та модельно-орієнтовану інженерію для автоматизованої генерації й адаптації механізмів моніторингу під час виконання місій [93, 94]. Для багатоапаратних сценаріїв розроблено платформи на кшталт EAMOS, що спрощують опис місій, синхронізацію та інтеграцію гетерогенних БПЛА [95].

Порівняльні дослідження архітектур інструментальних стеків виявляють спільні підходи між БПЛА та наносупутниками, зокрема переваги використання проміжного програмного забезпечення та операційних систем реального часу, таких як FreeRTOS і Linux RT, для керування сенсорним корисним навантаженням [96]. Інтеграція когнітивних агентів із ROS підвищує рівень абстракції, що позитивно впливає на ефективність програмування та експлуатаційні характеристики інтелектуальних БПЛА [97].

Професійні польотні контролери дедалі частіше поєднують складні апаратні платформи з новими реалізаціями ОСРЧ, використовуючи динамічне пріоритетне планування для підвищення точності, надійності та реактивності

системи [98]. Модельно-орієнтовані підходи до розроблення програмного забезпечення забезпечують автоматизацію процесів верифікації та зменшення сертифікаційного навантаження для критичних застосувань авіоніки [99].

Розвиток програмних платформ для керування групами БПЛА в реальному часі передбачає інтеграцію бездротового зв’язку через Wi-Fi та стільникові мережі, що забезпечує підтримку польотів поза межами прямої візуальної видимості та підвищену гнучкість місій [100]. Для систем із підвищеними вимогами до гарантій коректності застосовуються інструменти модельно-орієнтованої генерації коду для мікроядерних середовищ seL4, які забезпечують строгий поділ компонентів і високий рівень надійності [101]. Архітектури на основі програмної шини сприяють модульності, синхронізації та масштабованості систем керування польотом [102].

Додатково розглядаються онтологічно-орієнтовані підходи до формування вимог, що враховують географічні та середовищні чинники та підвищують повноту специфікацій систем керування польотом [103], а також фреймворк iTaSC, який забезпечує реактивне уникнення перешкод і ефективну навігацію в умовах відсутності GPS [104].

Узагальнюючи, розглянуті дослідження демонструють значний прогрес у розвитку програмних фреймворків, бібліотек керування та ОСРЧ-орієнтованих архітектур, які підвищують надійність, гнучкість і масштабованість систем керування безпілотних літальних апаратів у широкому спектрі застосувань. В той самий час підтверджують, що оптимізовані конфігурації програмного забезпечення, адаптивне управління ресурсами та збалансоване обчислювальне навантаження є ключовими чинниками підвищення ефективності та експлуатаційної стійкості систем керування польотом БПЛА.

1.3 Аналіз методів забезпечення енергетичної ефективності обчислень з урахуванням обмежень реального часу та вимог до реактивності

Бортовий обчислювач наносупутника, як активний обчислювальний засіб, складається з структурних блоків зображених на рисунку 1.15.

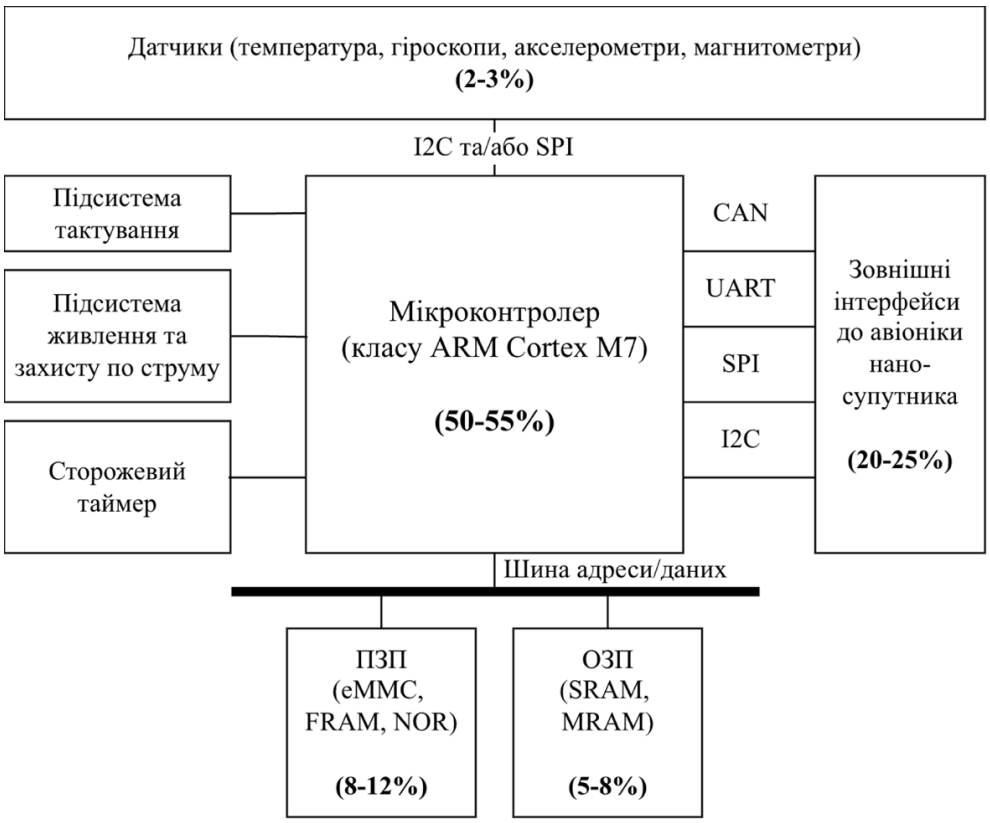

Рисунок 1.15 – Структурна схема типового бортового обчислювача та типовий розподіл споживаної електричної енергії його складовими – експертна оцінка (у дужках)

Переважна кількість енергії що споживається БО, споживається його МК, що, зазвичай, і є основним елементом оптимізації. Другим елементом за споживанням є блок послідовних інтерфейсів, його складові детально розглянуті у оглядово-аналітичній статті з бортових інтерфейсів [105, 106].

1.3.1 Аналіз наукових публікацій за напрямком

Швидкий розвиток обчислювальних можливостей у бортових системах наносупутників супроводжується зростаючими вимогами до енергетичної ефективності, особливо в умовах, коли ресурси живлення суворо обмежені. В одному з найбільш вимогливих контекстів, системах реального часу, необхідно забезпечувати не лише дотримання дедлайнів та заданої реактивності, але і мінімізацію споживаної енергії. Якщо класичні підходи до проєктування систем реального часу зосереджувалися здебільшого на часових гарантіях, то останніми роками спостерігається чіткий інтерес до інтегрованих моделей, у яких часова

коректність, енергоспоживання та надійність розглядаються як взаємопов’язані компоненти єдиного оптимізаційного завдання [107].

У обчислювальних платформах загального призначення енергетична ефективність досягається через такі технології, як динамічне масштабування напруги і частоти (англ. Dynamic Voltage and Frequency Scheduling, DVFS), менеджмент динамічної потужності (англ. Dynamic Power Management, DPM), енерго-орієнтоване планування задач і керування тепловим режимом. Застосування цих підходів у реальному часі ускладнюється необхідністю гарантування реактивності як здатності системи відповідати на зовнішні події у заданому часовому інтервалі. Така необхідність, безумовно, накладає додаткові обмеження на можливі стратегії енергозбереження [107].

Для вбудованих і критичних систем керування ситуація ускладнюється тим, що окрім енергетичного бюджету та часових гарантій, системи повинні дотримуватися вимог безпеки, відмовостійкості та відновлюваності - FDIR. Тут часто використовують моделі (m,k)-firm для задач помірної критичності або реплікацію і потрійне модульне резервування (англ. Triple Modular Redundancy, TMR) для задач високої критичності. Кожен із цих механізмів несе власну енергетичну ціну: резервування збільшує навантаження на обчислювальні ресурси, а жорсткі межі реактивності обмежують ступінь, до якого можна застосовувати DVFS або інші стратегії управління енергією [108, 109]. У багатоядерних чи гетерогенних архітектурах додатковою є проблема теплових обмежень і розподілу навантаження, що вимагає синтезу підходів з енергоменеджменту, планування реального часу та гарантування надійності [110].

Особливо гостро ці виклики проявляються у наносупутниках. Енергетичний бюджет всього наносупутника вимірюється одиницями або десятками ватів (в залежності від розміру), тоді як кількість задач, що виконуються на борту, постійно зростає — від навігації та телеметрії до високопродуктивної обробки зображень, алгоритмів орієнтації, автономних рішень. Енергетичний бюджет БО, в свою чергу, є в межах від 1 до 5 Вт.

Живлення всього наносупутника залежить від його орієнтації, ефективності сонячних панелей, ступеню зарядженості акумулятора та умов орбіти. Урахування таких змінних та обмежень призводить до необхідності враховувати енергетичну динаміку як інтегральну частину завдання планування місії. У цій ситуації класичні підходи до реального часу вимагають адаптації до середовища з високою невизначеністю та суворими фізичними обмеженнями [111, 112].

Останні дослідження [113, 114] свідчать, що успішна експлуатація таких платформ потребує нової генерації енерго-обізнаних механізмів планування, які враховують прогноз надходження енергії, динаміку заряду батарей, теплові характеристики орбіти, а також змінні пріоритети задач залежно від етапу місії. Для сузір'їв наносупутників актуальним стає питання координованого планування задач, де енергетичні можливості кожного вузла впливають на досягнення спільної мети місії. Поєднання реального часу, енергетичної оптимізації та координації між вузлами формує унікальне, динамічне дослідницьке поле [109, 115].

1.3.2 Таксономія дослідницьких напрямків та методів

Виконавши аналіз публікацій, для класифікації та систематизації дослідницьких напрямків з побудови енергоефективних систем реального часу, запропонована наступна ієрархічна таксономія (таблиця 1.3):

1. Моделі завдань. Цей рівень класифікує типи обчислювальних задач, які виконуються у системах реального часу - періодичні, реактивні, подієво-орієнтовані, а також, складні ациклічні графові (англ. Directed Acyclic Graph, DAG) моделі. Від вибору моделі завдання залежить характер часової поведінки системи, можливість застосування DVFS/DPM, а також рівень паралелізму. Це фундаментальний рівень, оскільки він визначає природу обчислень, що потребують енергетичної оптимізації.

2. Моделі навантаження та середовища. Цей рівень описує зовнішні та внутрішні умови, у яких працює система: варіативність WCET, теплові процеси,

енергетичні профілі батарей, режими збирання енергії (англ. energy harvesting). У контексті космічних апаратів цей рівень критичний, бо саме середовище визначає реальні обмеження - доступну потужність, змінні орбітальні умови та теплові цикли.

3. Метрики та критерії. Цей рівень структурує основні метрики, що використовуються для оптимізаційних цілей: дедлайни, затримки виконання (latency), джиттер, енергоспоживання, потужність, надійність, а також метрики рівня місії (QoS, здатність до downlink). Саме метрики задають, що оптимізує планувальник: час, енергію, надійність або їх комбінацію.

4. Методи управління енергією. Рівень містить механізми, які безпосередньо впливають на енергоспоживання: DVFS, DPM, energy harvesting, MPPT-aware підходи, теплові обмеження. Цей рівень описує доступні інструменти системи для балансування між продуктивністю та енергією.

5. Планування та оптимізація. Цей рівень охоплює методи побудови розкладу: класичні алгоритми планування завдань реального часу (EDF, RMS), offline-та online-оптимізацію, планування змішаної критичності (англ. mixed-criticality scheduling), відмово-стійке планування (англ. fault-tolerant scheduling). Він відображає, як саме система приймає рішення, застосовуючи метрики рівня 3 та інструменти рівня 4 до моделей задач рівня 1.

6. Архітектури обчислювальних систем. Описує фізичні обчислювальні платформи: малопотужні МК, багатоядерні SoC, гетерогенні CPU-FPGA-GPU рішення, радіаційно-захищені БО. Архітектура визначає апаратні можливості для енергоменеджменту, надійності та рівня паралелізму, тому вона є природним системним шаром між алгоритмічними методами та реалізацією.

7. Системна інтеграція у наносупутниках. Описує, як методи та архітектури попередніх рівнів застосовуються у реальних космічних системах: побудова БО, ADCS, планування комунікацій, обробка сенсорних потоків. Цей рівень інтегрує технічні рішення у контексті вимог КА - надійність, теплові, радіаційні та енергетичні обмеження.

8. Рівень місії та сузір’їв наносупутників. Найвищий рівень абстракції, що розглядає планування задач не лише всередині одного КА, а й у масштабі всієї місії чи групи наносупутників. Містить задачі QoS місії, колективне планування (cooperative scheduling), розподіл енергетичних ресурсів у сузір’ях, continuous-time планування. Цей шар визначає, як індивідуальні рішення планувальника впливають на результат місії.

Таблиця 1.3 Таксономія складових методів забезпечення енергоефективності у обмеженнях систем реального часу

Рівень 1	Рівень 2	Рівень 3	Роботи
1. Моделі завдань	Періодичні, спорадичні, аперіодичні завдання	Класичні RTS моделі	[107]
	Завдання що активуються подіями	Реактивні системи	[116]
	DAG-моделі	Паралельні графи завдань	[117]
2. Моделі навантаження та середовища	WCET	Стохастичні та інтервальні оцінки	[118]
	Енергетичні моделі	Динаміка батареї, energy harvesting	[119]
	Теплові моделі	Thermal-aware scheduling	[120]
3. Метрики та критерії	Метрики RT	Дедлайни, latency	[84]
	Енергетичні	Потужність, споживання, MPPT	[113]
	Надійність	FT, TMR, (m,k)-firm	[121]
	Місійні метрики	QoS, продуктивність місії	[109]
4. Методи управління енергією	DVFS/DPM	Зниження напруги та частоти, керування режимами енергозбереження	[84]
	Energy harvesting	Планування при нестабільних джерелах енергії	[119]
	MPPT-aware	Інтегроване управління сонячною потужністю	[113]
	Thermal-aware	Контроль температури	[120]
5. Планування та оптимізація	Offline оптимізація	MILP, branch-cut-and-price	[122]
	Online адаптивні методи	Slack reclamation, reactive scheduling	[116]
	Mixed-Criticality	Graceful degradation, energy-aware MCS	[123]
	Fault-tolerant scheduling	TMR, реплікація	[124]

Таблиця 1.3 (Продовження)

Рівень 1	Рівень 2	Рівень 3	Роботи
6. Архітектури систем	MCU та low-power SoC	Cortex-M, DSP	[125]
	Гетерогенні системи	CPU+FPGA, CPU+GPU	[117]
	Радіаційно-захищені БО	LEON, RAD750	[126]
7. Системна інтеграція у космічних апаратах	CubeSat БО	Архітектура, енергобюджети	[127]
	Контроль орієнтації (ADCS)	Task criticality, FT	[121]
	Канали зв’язку	Downlink scheduling	[109, 128]
8. Рівень місії та сузір’їв	Планування місії	Оптимізація QoS	[109]
	Наносупутникові сузір’я	Cooperative scheduling	[109]
	Безперервне планування	Continuous-time optimization	[129]

Запропонована таксономія складається з восьми рівнів, що охоплюють усю вертикаль - від моделей задач і навантаження до архітектури, інтеграції у КА та планування на рівні місії. Така структура дозволяє системно проаналізувати енергоефективні методи у реальному часі, відокремити фундаментальні алгоритмічні аспекти (рівні 1 - 5) від апаратних та прикладних (рівні 6 - 8), та показати природний перехід від теоретичних моделей до застосування у платформах наносупутників та сузірь наносупутників. Побудована таксономія дозволяє побачити, що дослідження не є ізольованими: кожен рівень утворює функційну залежність від нижчих рівнів і, у свою чергу, накладає вимоги на вищі. Це, відповідно, вимагає розгляду комплексних моделей енергоефективності, реактивності і продуктивності, як одного цілого.

1.3.3 Методи планування та диспетчеризації завдань в системах реального часу

Передбачуваність визначається обсягом попередньої інформації про особливості виконання завдань реального часу (РЧ). Вихідними даними для алгоритмів є дані про завдання та їх обмеження РЧ. Існує безліч варіантів розподілу функцій між стадіями планування. Від цього розподілу залежать передбачуваність і гнучкість планування, а також, складність алгоритмів, обсяг проміжних даних, ефективність планування за різними критеріями. З

урахуванням основних особливостей такого розподілу можна виділити три основні концепції планування [130, 131]:

- табличне планування;
- планування з фіксованими пріоритетами;
- планування з динамічними пріоритетами.

Сучасний стан досліджень цих методів, зокрема їх застосування в новітніх вбудованих системах, розглянуто в працях останніх років [131, 135].

Табличне планування. При складанні табличного розкладу вказується розподіл виконання періодичних запитів за часом. Табличне планування забезпечує складання розкладу для завдань РЧ. Аналіз здійсненності визначається вже самим фактом наявності розкладу, згідно з яким всі обмеження РЧ дотримуються. Періодичні запити виконуються згідно із заздалегідь складеним розкладом. Для виконання аперіодичних запитів використовується час, який не був використаний у розкладі, тобто вільний час. У загальному випадку проблема складання розкладу у вигляді таблиці є NP-складною [134], тому алгоритм планування може мати значну часову складність.

Перевага табличного планування – це висока передбачуваність. Також перевагою табличного планування є можливість планування за наявності складних обмежень для завдань РЧ, а також врахування в розкладі складних взаємодій між завданнями, оскільки розклад складається до початку функціонування системи РЧ [133, 134].

Одним з недоліків табличного планування є проблема довжини розкладу (таблиці), оскільки при наявності великої кількості завдань РЧ з різними періодами довжина таблиці часто виявляється неприйнятною для практичних реалізацій. Інший недолік – це складність виконання аперіодичних запитів. Дана концепція планування не володіє необхідною гнучкістю для планування завдань РЧ в системах на базі МК з обмеженими ресурсами.

Метод планування завдань реального часу з фіксованими пріоритетами. Алгоритм планування забезпечує установку фіксованого пріоритету для кожного завдання РЧ, а також такі параметри, як початкове

зміщення і період. Диспетчеризація здійснюється на основі формування запитів періодичних завдань згідно з встановленими початковими зміщеннями і періодами, а також, на основі обслуговування черги сформованих запитів, враховуючи фіксовані пріоритети періодичних завдань. У разі планування з фіксованими пріоритетами аналіз здійсненності може бути досить складним, адже на відміну від табличного планування в даному випадку відсутній розклад. В ході аналізу здійсненності з'ясовується гарантованість дотримання обмежень РЧ на основі встановлених параметрів завдань РЧ.

Відсутність розкладу для розглянутої схеми планування зменшує передбачуваність, але збільшує гнучкість у порівнянні з табличним плануванням, оскільки при цьому підсистема планування може більш ефективно реагувати на появу інших завдань. Це пояснюється тим, що для даного методу планування початок і закінчення запиту завдання РЧ спочатку не фіксуються, що забезпечує можливість більш простих способів виділення вільного часу для вирішення інших завдань. Відомими методами планування з фіксованими пріоритетами є Rate Monotonic Scheduling (RMS) [135, 136] і Deadline Monotonic Scheduling (DMS). Інформацію про теоретичні основи, сфери практичного застосування методів планування з фіксованими пріоритетами можна знайти в [137, 138].

Метод планування завдань реального часу з динамічними пріоритетами. Планування здійснюється на основі формування запитів періодичних завдань відповідно до встановлених початкових зміщень і періодів, а також на основі обслуговування черги сформованих запитів, з урахуванням динамічної зміни пріоритетів.

Так само, як і для статичного підходу, у разі планування з динамічними пріоритетами велике значення приділяється аналізу здійсненності, в ході якого з'ясовується гарантованість дотримання обмежень РЧ. Планування з динамічним призначенням пріоритетів має істотну гнучкість, забезпечуючи вищу ефективність використання процесорного часу (до 100%) порівняно з методами фіксованих пріоритетів [136].

Відомими алгоритмами цього класу є планувальники за – найближчим граничним терміном (англ. Earliest Deadline First, EDF) і найменшим запасом часу (англ. Least Laxity First, LLF). Інформацію про теоретичні основи та сфери практичного застосування методів планування з динамічними пріоритетами (зокрема в IoT та мультимедійних системах) наведено в [138, 139].

1.3.4 Аналіз здійсненності системи завдань реального часу

Аналіз здійсненності - це пошук відповіді на питання про те, чи гарантується дотримання обмежень РЧ. Щоб визначити, чи буде система завдань $\tau = \{\tau_1, \tau_1, \dots, \tau_n\}$ виконуватися без зриву директивних термінів, введемо поняття $u(\tau)$ - завантаженість системи, або ступінь утилізації ресурсів системи τ, при часі обчислень і-тої задачі – C_i, періоду виконання – T_i та граничному терміні її закінчення – D_i. Тоді завантаженість системи на одному обчислювальному ядрі визначається як:

$$u(\tau) = \sum_{i=1}^{n} \frac{C_i}{T_i}. \quad (1.1)$$

Якщо $u(\tau) \to 1$, то система на основі одного процесора є максимально завантаженою. Динамічний алгоритм EDF [140] виконує систему без зриву директивних термінів, в якій $T_i = D_i$, якщо $u(\tau) \leq 1$. Для порівняння, статичний алгоритм планування RMS допускає систему без зриву директивних термінів, в якій $T_i = D_i$, коли $u(\tau) \leq 0.69$ [141]. При великому завантаженні системи статичні методи планування є добре детермінованими, але менш гнучкими в порівнянні з динамічними методами. Що стосується МПС і МК, то умова дотримання директивних термінів для завдань РЧ буде також залежати від кількості обчислювальних ядер процесора та $u(\tau) \to m$.

Висновки до розділу 1

У першому розділі виконано огляд технології наносупутників класу CubeSat та оглянуто специфіку їх побудови, стандартизації, проблем та умов

експлуатації. Наносупутники класу CubeSat займають велику частку сучасних космічних супутникових місій та є важливим напрямком дослідних та конструкторських і виробничих робіт як для комерційних структур, так і науково-дослідних закладів та академії. Використання COTS компонентів забезпечило стрімке зростання обсягів наносупутників на орбіту, кількість яких досягла в 2025 році 2700 одиниць.

Проблема наносупутникової галузі в Україні є надважливою в умовах сучасних геополітичних викликів. Автори [142] зазначають, що складовою гарантування суверенітету та сталого розвитку країни є завдання рівня національної безпеки: до 2030 р. мати національні супутники для оборонних цілей, власну систему ситуаційної обізнаності й контролю космічного простору, а також забезпечити інтеграцію космічних систем в оборонні технології.

В результаті систематизованого огляду літератури встановлено, що основними вимогами до БО аерокосмічної техніки, є забезпечення високої енергетичної ефективності БО та гарантованих показників продуктивності та реактивності бортового ПЗ. Встановлено що вимоги до БО БПЛА та наносупутників мають суттєву схожість вимог, що надають дослідженню міжгалузеву значимість та застосування.

Таким чином, вирішено перше та друге завдання дисертаційного дослідження.

Наукові та практичні результати, отримані в першому розділі, представлено в наукових публікаціях [25, 37, 77, 105, 106].

Подальші дослідницькі завдання, їх взаємозв’язок, отримані результати та підрозділи роботи відображені на рисунку 1.17 та рисунку 1.16.

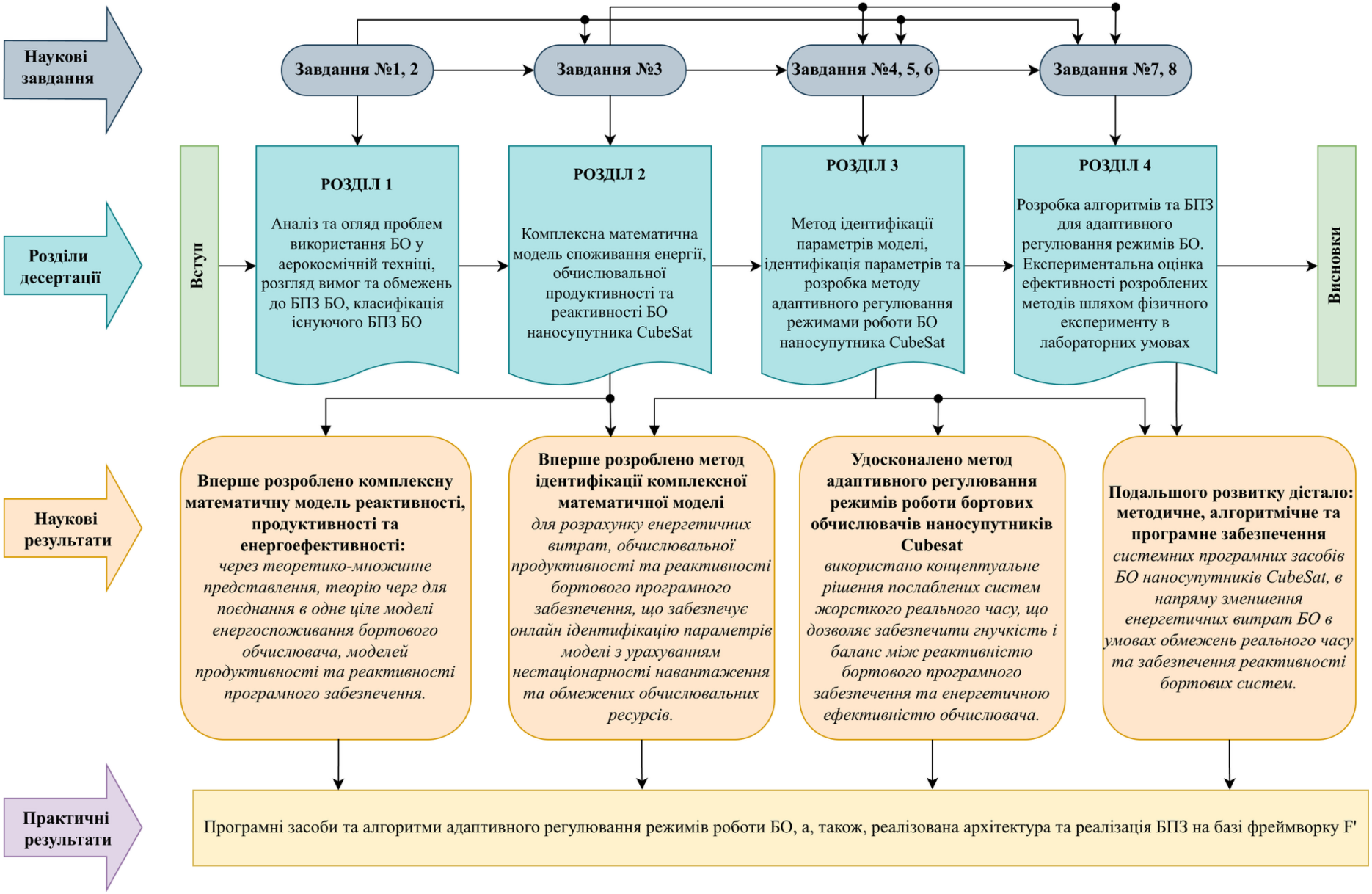

Рисунок. 1.16 – Взаємозв'язок вирішуваних завдань, структури роботи та отриманих результатів

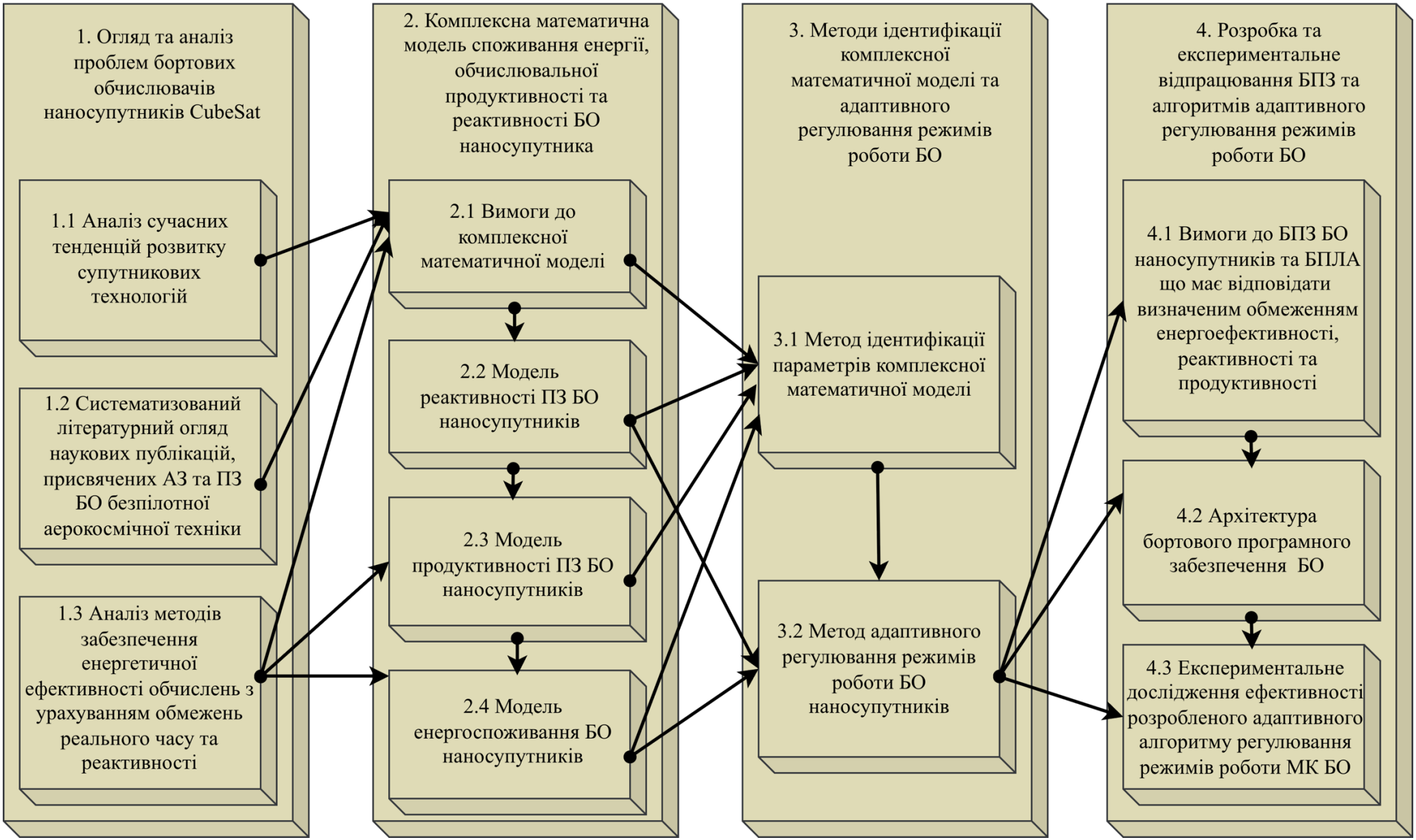

Рисунок 1.17 – Взаємозв’язок складових дисертаційного дослідження

РОЗДІЛ 2 КОМПЛЕКСНА МАТЕМАТИЧНА МОДЕЛЬ СПОЖИВАННЯ ЕНЕРГІЇ, ОБЧИСЛЮВАЛЬНОЇ ПРОДУКТИВНОСТІ ТА РЕАКТИВНОСТІ БОРТОВИХ ОБЧИСЛЮВАЧІВ НАНОСУПУТНИКІВ CUBESAT

Існує головне технічне протиріччя, що відображене на рисунку 2.1, це зменшення енергетичних витрат, що погіршує обчислювальну продуктивність та реактивність системи. Зокрема, споживання енергії зростає квадратично зі збільшенням напруги живлення, що робить цей параметр найважливішим для оцінки енергетичних витрат, але відхилення напруги від рекомендованого виробником значення погіршує показники надійності. Підвищення тактової частоти МК прискорює обчислення (збільшує обчислювальну продуктивність) та покращує реактивність, але, водночас, збільшує споживання енергії. Збільшення частоти системного таймера (частоти системного тику) – це одночасно і збільшення енергетичних витрат на службові операції операційної системи, і покращення реактивності. Використання режимів енергозбереження суттєво знижує споживання енергії, але помітно погіршує продуктивність та збільшує час реакції (погіршує реактивність) через затримки при відновленні активної роботи МК.

Розроблення комплексної математичної моделі споживання енергії, обчислювальної продуктивності та реактивності бортового обчислювача наносупутника має вирішальне значення для ефективності всієї місії.

По-перше, така модель дозволяє точно оцінити баланс між обмеженими енергетичними ресурсами та обчислювальними потребами, що є критично важливим в умовах жорстких енергетичних обмежень наносупутників. По-друге, вона створює теоретичну основу для оптимізації розподілу обчислювальних навантажень, забезпечуючи стабільну роботу системи за різних режимів. По-третє, модель дає змогу аналізувати реактивність системи, гарантуючи здатність бортового комп’ютера своєчасно реагувати на події, важливі для місії. Модель також підтримує процеси планування роботи

супутника, надаючи прогностичні дані про використання ресурсів, що покращує планування розкладу комунікаційних сеансів та роботи корисного навантаження.

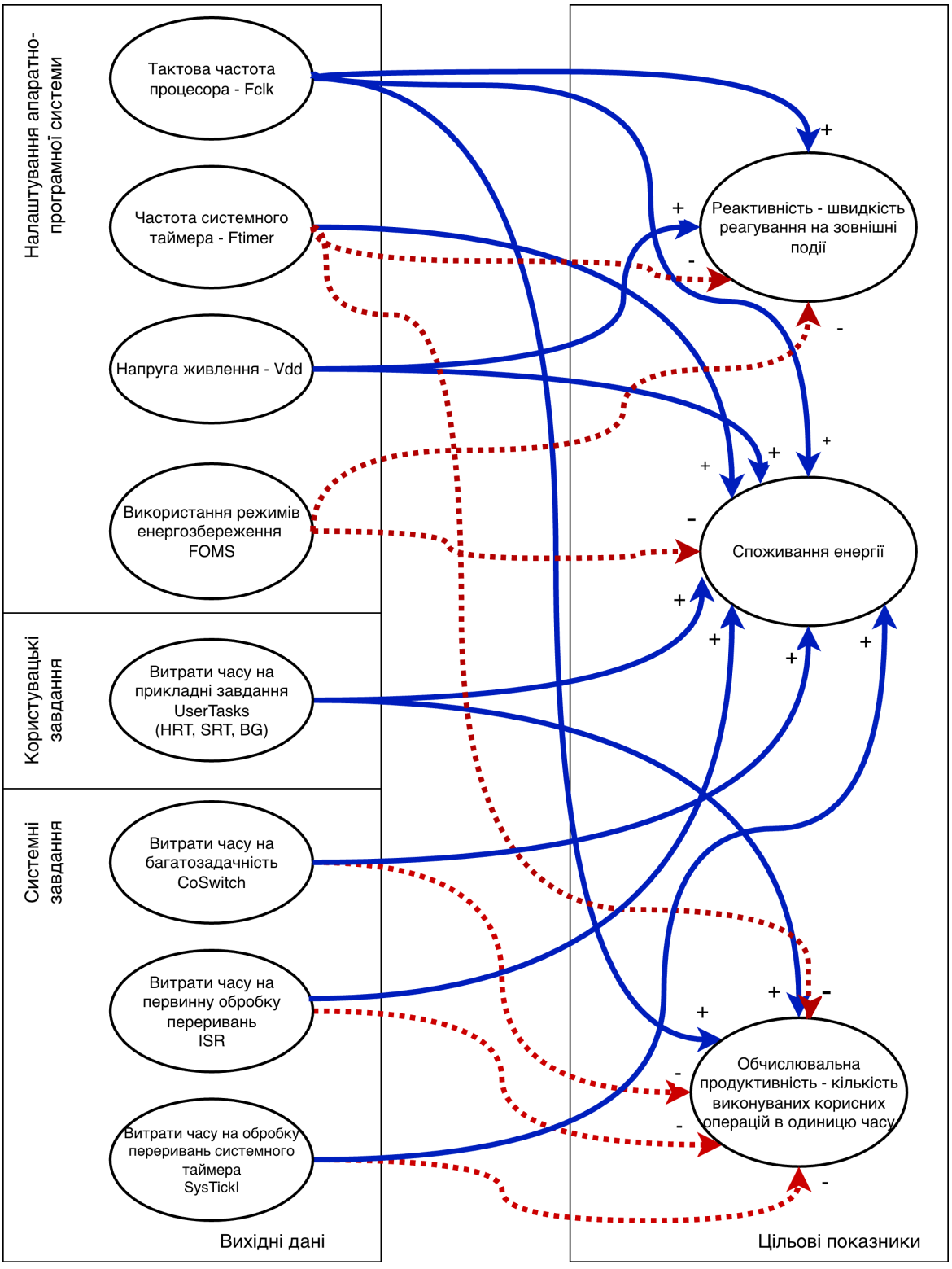

Рисунок 2.1 – Причинно-наслідкова діаграма залежності цільових показників від вихідних даних

На етапі проєктування модель допомагає обмежити простір параметрів майбутньої платформи, дозволяючи інженерам заздалегідь оцінити енергоспоживання, продуктивність та інші характеристики. Під час роботи на орбіті її використання відкриває можливості для адаптивного керування режимами в реальному часі завдяки порівнянню фактичних показників із закладеними у модель. У перспективі така модель сприяє підвищенню відмовостійкості та масштабованості системи, що важливо для розвитку наступних поколінь наносупутників. Таким чином, модель є універсальним інструментом, що поєднує аналіз зовнішніх впливів, апаратних конфігурацій, програмних налаштувань і динамічних політик управління роботою МК.

2.1 Вимоги до комплексної математичної моделі енергетичних витрат, обчислювальної продуктивності та реактивності бортових обчислювачів

Модель призначена для розрахунку залежностей цільових показників ефективності бортового обчислювача від налаштувань апаратно-програмної частини та характеристики завдань, що вирішує обчислювач.

2.1.1 Комплексна математична модель в нотації теорії множин

Узагальнено в нотації теорії множин комплексна математична модель споживання енергії ($Energy$), обчислювальної продуктивності ($Produtivity$) та реактивності ($Reactivity$) бортового обчислювача – це система відображень:

$$\begin{cases} Reactivity: \{F_{clk}, \quad F_{timer} \quad\} \times Modes \times Tasks \to R; \\ Energy: \{F_{clk}, \quad F_{timer}, \quad V_{dd}\} \times Modes \times Tasks \to E; \\ Produtivity: \{F_{clk}, \quad F_{timer} \quad\} \times Modes \times Tasks \to P, \end{cases} \quad (2.1)$$

де $\{F_{clk}, \ F_{timer}, \ V_{dd}\}$ – множина системних параметрів, що характеризує налаштування апаратної частини: F_{clk} – тактова частота, V_{dd} - напруга живлення, F_{timer} – частота переривань від системного таймеру;

$Modes$ – множина можливих політик та відповідних програмних реалізацій керування режимами енергозбереження;

$Tasks = \{E^H, E^S, E^{bw}\}$ – множина завдань, яка містить опис вирішуваних системою завдань трьох видів: E^H – обробка подій жорсткого реального часу, E^S – обробка подій м'якого реального часу, E^{bw} – фонові завдання;

E – середня споживана електрична потужність;

P – обчислювальна продуктивність – кількість виконуваних корисних операцій в одиницю часу.

R – множина показників реактивності програмного забезпечення бортових обчислювачів наносупутників;

2.1.2 Режими роботи бортового обчислювача

МК бортового обчислювача може знаходитись в одному з доступних режимів:

$$Modes = \{Active, Wait, Sleep, Backup\}. \quad (2.2)$$

В активному режимі (Active) МК працює на повній тактовій частоті, виконує всі обчислення, працює з периферією (UART, SPI, ADC, тощо). Працюють периферійні модулі. Споживання енергії – номінальне, найвище серед усіх режимів.

В режимі сну (Sleep) ядро МК зупинено, але периферійні модулі продовжують роботу (наприклад, таймери, UART). Процесор очікує надходження подій від периферії (переривання від таймера, завершення передачі даних, прихід сигналу по комунікаційному інтерфейсу тощо). Споживання енергії – нижче, ніж у Active, оскільки ядро МК не виконує інструкцій.

В режимі сну та очікування (Wait) зупиняється тактування більшості блоків МК, ядро і більшість периферійних модулів вимикаються. Деякі низькоспоживні модулі (такі як годинник реального часу – RTC, або

сторожовий таймер – watchdog) можуть залишатися активними. Вихід із режиму – через зовнішнє чи внутрішнє переривання. Споживання енергії – суттєво нижче, ніж у Sleep, оскільки більшість периферійних модулів зупинена.

В режимі глибокого сну (Backup) майже вся система вимкнена, живляться тільки мінімально необхідні блоки, зазвичай це – RTC (Real-Time Clock), регістри процесора та логіка пробудження. Режим застосовується для дуже тривалого очікування подій, при цьому зберігаються лише критично важливі дані з можливістю пробудження по зовнішньому сигналу чи таймеру RTC (наприклад, датчики з автономним живленням). Споживання енергії – найнижче з можливих.

Режими МК та переходи між ними представлено на рисунку 2.2 у вигляді скінченного автомата. Початковий стан автомата – Активний режим. Переходи «1», «3», «5» характеризують час, необхідний для переходу в відповідні режими енергозбереження, а переходи «2», «4», «6» – це час, необхідний для відновлення активного режиму після режиму енергозбереження, коли прийшло зовнішнє переривання.

2.2 Модель реактивності програмного забезпечення бортових обчислювачів наносупутників

2.2.1 Загальні положення

Реактивність бортового обчислювача – це складова загальної ефективності апаратно-програмного забезпечення бортового обчислювача, що характеризує середню або гарантовану частку подій, на які система встигає відреагувати за одиницю часу, у відношенні до сумарної інтенсивності надходження подій. Надалі будемо розрізняти три типи процесів, які визначають реактивність, як складову загальної ефективності апаратно-програмного забезпечення бортового обчислювача (рисунок 2.3):

– фонові процеси;

– процеси м’якого реального часу;

– процеси жорсткого реального часу.

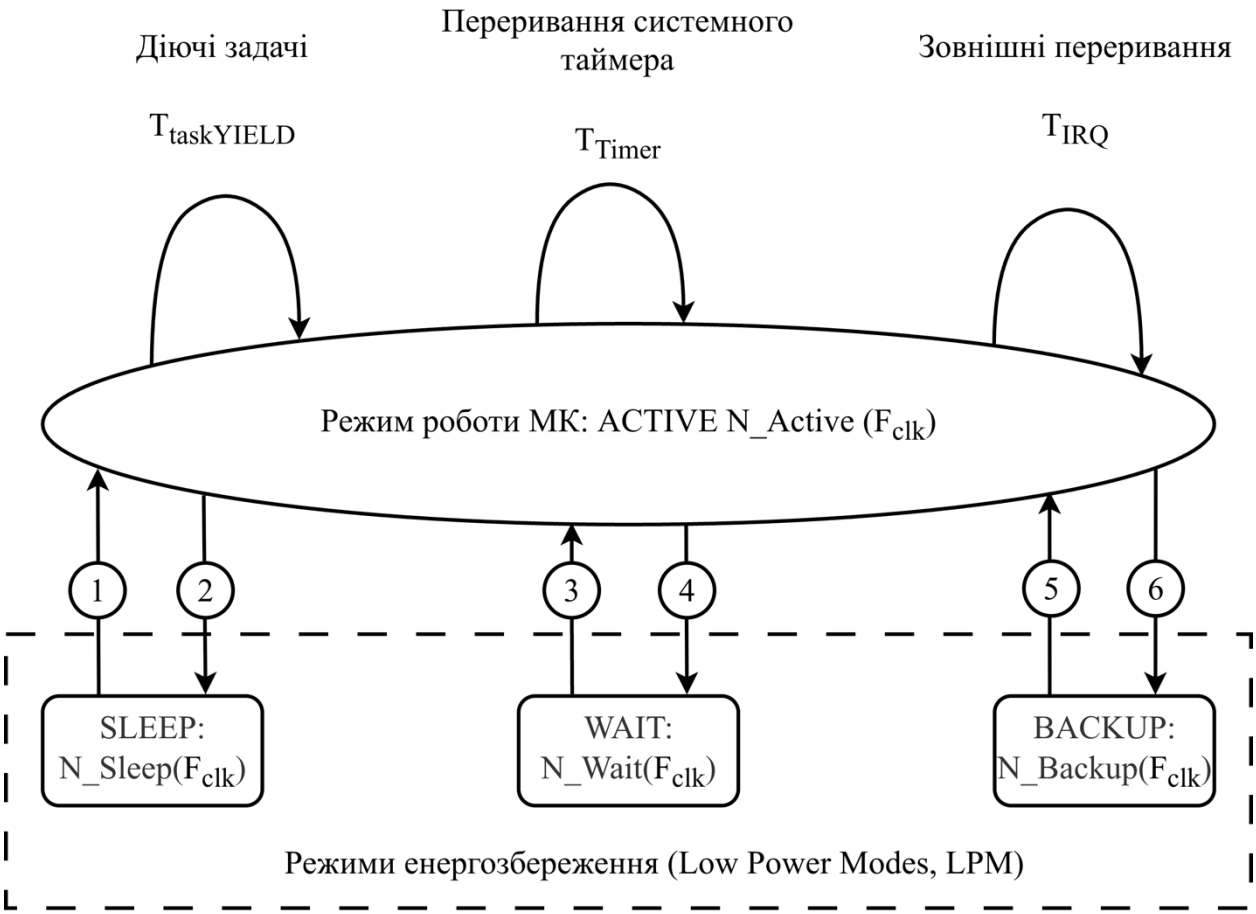

Рисунок 2.2 – Діаграма станів та переходів між режимами МК БО

Процеси м’якого реального часу (soft real-time) та жорсткого реального часу (hard real-time) ініціюються зовнішніми подіями. Події потрапляють в систему для подальшої обробки через механізм переривань. Переривання – це системний сигнал, що може використовуватись як засіб комунікацій між апаратними та програмними складовими.

Надалі в роботі розглянуто варіант, в якому обробка переривання (події) здійснюється в 2 етапи: первинним та вторинним обробниками.

Первинна обробка переривання здійснюється операційною системою в процедурі обробки переривань (англ. Interrupt Service Routine, ISR) та полягає в ініціалізації об’єкта ядра, який дозволяє роботу вторинного обробника переривання. При цьому операційна система бере на себе відповідальність за

такі деталі, як переключення контексту з поточної функції до вторинного обробника переривання з урахуванням пріоритету вторинного обробника.

Вторинний обробник переривання – це користувацька функція, що починає свою роботу після отримання сигналу (перехід об’єкта ядра в сигнальний стан) про виникнення конкретного переривання у системі. Час реакції системи на переривання надалі визначається як інтервал часу між створенням апаратною частиною сигналу переривання та завершенням обробки цього сигналу відповідним вторинним обробником.

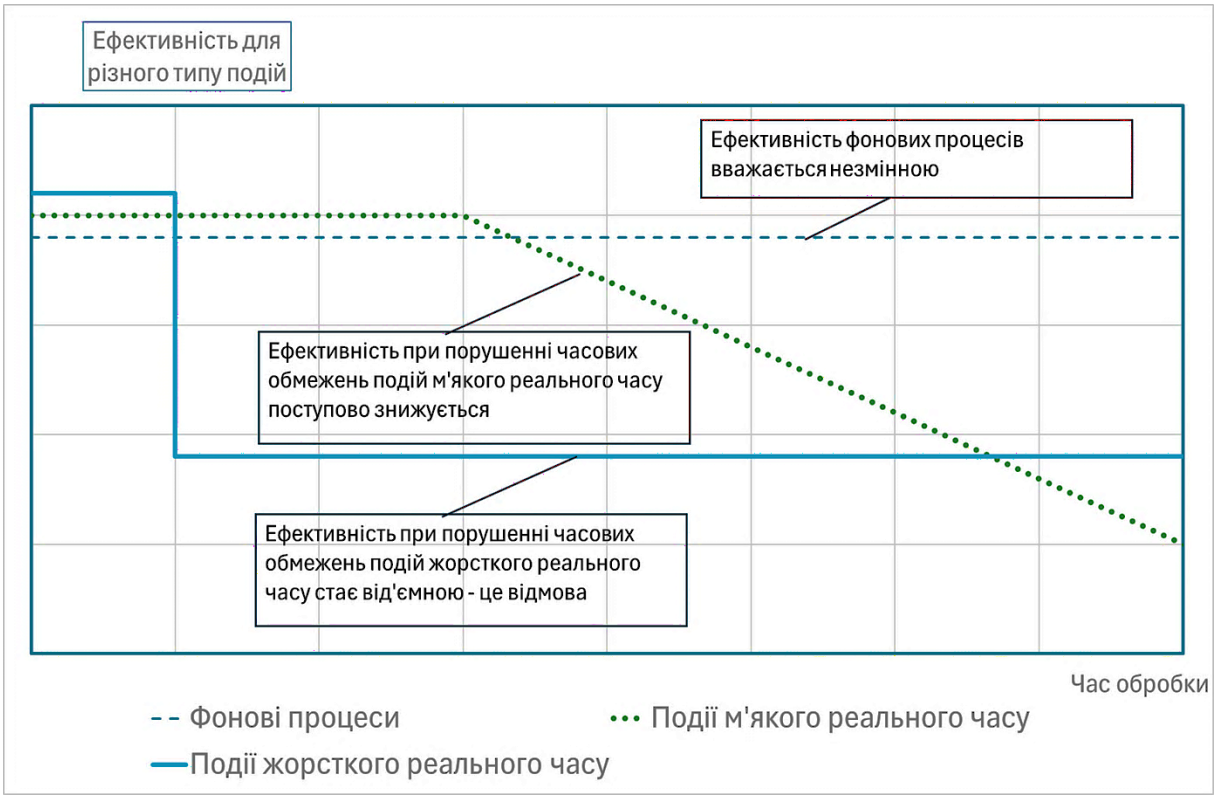

Рисунок 2.3 – Змістовна інтерпретація реактивності бортового обчислювача як складової загальної ефективності апаратно-програмного забезпечення бортового обчислювача

При обробці подій жорсткого реального часу, реактивність – це максимальна кількість зовнішніх подій, що надходять у вигляді переривань на одиницю часу, на які система здатна гарантовано відреагувати так, що час

реакції на кожну подію не перевищує визначений дедлайн (часові обмеження), враховуючи фіксовані та змінні часові витрати на обробку події (переривання).

Якщо мова йде про обробку подій м'якого реального часу, то реактивність бортового обчислювача – це характеристика здатності системи обробляти зовнішні події, визначена через середню кількість подій, які система гарантовано може обробити за одиницю часу, враховуючи затримки первинної та вторинної обробки, накладні витрати ОС і обмеження планувальника.

Зазвичай бортовий обчислювач забезпечує обробку обох класів зовнішніх подій. Окрім завдань обробки зовнішніх подій, бортовий обчислювач витрачає процесорний час на фонові процеси, які не мають обмежень на час їх виконання.

Надалі дотримуємось наступного визначення: реактивність бортового обчислювача – це здатність системи гарантувати обробку всіх подій жорсткого реального часу та забезпечувати мінімальну затримку обробки подій м'якого реального часу з врахуванням стохастичних характеристик потоку подій, часу обслуговування та накладних витрат на обробку. Гарантована обробка жорсткого реального часу в цій постановці завдання є обмеженням.

2.2.2 Вихідні припущення та обмеження

1. Реактивність системи залежить від тактової частоти процесора – F_{clk} та характеристик потоку зовнішніх подій (events) – $E = \{E_i\}$.
2. Потік подій розбито на 2 класи: жорсткого (hard) – E^H та м'якого (soft) реального часу – E^S:

$$E = E^H \cup E^S \,,\; E^H \cap E^S = \emptyset, \tag{2.3}$$

3. Кожен тип події представлено кортежом, який залежить від типу події:

$$\forall E_i \in E^H,\ E_i = \langle f_i^{max}, \quad T_i, \quad D_i \rangle, \tag{2.4}$$

$$\forall E_i \in E^S,\ E_i = \langle \lambda_i, \quad T_i, \quad D_i \rangle, \tag{2.5}$$

де f_i^{max} – максимальна інтенсивність виникнення події жорсткого реального часу (подій/с);

λ_i – середня частота виникнення події м'якого реального часу (подій/с);

$T_i = C_i / F_{clk}$ – максимальний час вторинної обробки (Worst Case Execution Time, WCET), розрахований через кількість тактів роботи процесорного ядра МК – C_i, та тактову частоту F_{clk};

D_i – обмеження на час обробки події (дедлайн) системою.

4. Інтервали між подіями м'якого реального часу статистично моделюються як експоненційні (пуассонівський потік для кожного класу подій).

5. Обробка переривання (події) здійснюється в 2 етапи: первинним (primary) та вторинним (secondary) обробниками.

6. Первинний обробник має найбільший пріоритет, його робота не переривається наступною подією.

7. Час роботи первинного обробника є фіксованим для всіх типів подій та залежить від тактової частоти процесорного ядра МК – F_{clk}:

$$T_{IRQ_p} = C_{IRQ_p} / F_{clk}, \tag{2.6}$$

де C_{IRQ_p} – кількість тактів процесорного ядра МК, необхідних первинному обробнику.

8. Первинний обробник ініціює роботу вторинного обробника через механізми операційної системи (об'єкти ядра). Вторинні обробники виконуються в одиночному сервері (single server), відповідно до механізму планування – EDF (Earliest Deadline First), допускається витиснення (preemption) вторинного обробника.

9. Час роботи вторинного обробника залежить від кількості тактів процесорного ядра МК, необхідних вторинному обробнику – C_{IRQ_s}, та тактової частоти процесорного ядра МК – F_{clk}:

$$T_{IRQ_s} = C_{IRQ_s}/F_{clk}, \tag{2.7}$$

10. Витрати часу вторинного обробника є сумою $T_{Core} + T_{IRQ_s}$

11. Буфер подій вважається необмеженим.

12. Фонова завантаженість процесора (background work) – це обчислювальна робота, яку МК виконує за умови відсутності необроблених подій. Вважаємо, що максимальний об'єм цієї роботи є відомим, МК її повинен виконати без обмежень реального часу, але при цьому знижується середній час, що залишається для обробки подій м'якого реального часу на наступну величину:

$$\Delta T_{bw} = C_{bw}/F_{clk}, \tag{2.8}$$

де C_{bw} – кількість тактів процесорного ядра, необхідних МК для відпрацювання фонової роботи.

13. Накладні витрати часу ядра ОС – T_{Core} на виконання таких операцій, як переключення контексту, роботу з об'єктами ядра є параметрами моделі, та залежать від кількість тактів процесорного ядра, необхідних для реалізації таких функцій – C_{Core} та від тактової частоти процесорного ядра МК – F_{clk}:

$$T_{Core} = C_{Core}/F_{clk}. \tag{2.9}$$

14. До обмежень моделі також відноситься те, що вона не розглядає затримки, які навмисно формуються програмним забезпечення, щоб оптимізувати споживану енергію [143]. До таких затримок відносяться:

– затримки прокрастинації (відкладання на потім – procrastination delays): алгоритми навмисно відкладають виконання деяких завдань, щоб збільшити тривалість періоду простою та ефективніше використовувати режими низького енергоспоживання. Ці затримки ретельно розраховуються, щоб не порушувати часові обмеження завдань;

– затримки планування: час, необхідний для прийняття рішень про зміну режиму енергоспоживання та перепланування завдань. Затримки планування, пов'язані з обчисленням оптимальних моментів для переходу в режим сну та пробудження: певні алгоритми виконують складні обчислення для визначення цих моментів, що може призвести до додаткових затримок;

– затримки передчасного завершення: ці затримки створюють додатковий простір для енергозбереження, але вимагають динамічного перепланування.

2.2.3 Функційні залежності моделі реактивності

Коефіцієнт завантаження процесора (коефіцієнт утилізації) обробниками переривань залежить від сумарної інтенсивності вхідного потоку подій двох класів: жорсткого та м’якого реального часу:

$$U = \begin{matrix} \sum_{\forall E_i \in E^H} f_{E_i}^{max} \cdot \left(T_{IRQ_p} + T_{E_i} + T_{Core} \right) + \\ \sum_{\forall E_j \in E^S} \lambda_{E_j} \cdot \left(T_{IRQ_p} + T_{E_j} + T_{Core} \right). \end{matrix} \quad (2.10)$$

Для гарантованої обробки потоку подій жорсткого часу E^H необхідно, щоб була виконана робота всіх первинних обробників переривань з урахуванням накладних витрат часу на роботу ОС – T_{Core}. Наступне обмеження сформульовано в припущенні, що вторинна обробка подій м’якого реального часу може бути відкладена на певний час:

$$U - \sum_{\forall E_j \in E^S} \lambda_{E_j} \cdot T_{E_j} \leq 1. \quad (2.11)$$

Час реакції на подію типу i буде визначатись двома складовими: часом її обробки та часом перебування події в черзі:

$$T_{resp,\,i} = W_i + \left(T_{IRQ_p} + T_{IRQ_s} + T_i + T_{Core}\right), \tag{2.12}$$

де W_i – час очікування у черзі для події типу i (секунди), що залежить від режиму планування та стану системи в момент приходу події.

Для визначення середнього часу очікування у черзі використано модель Поллачека–Хінчина (Pollaczek–Khinchine) [144], яка є фундаментальною в теорії черг:

$$E[W] = {\Lambda \cdot E[S^2]}\Big/{2 \cdot (1 - \rho)}, \tag{2.13}$$

де $E[W]$ – середній час очікування у черзі (секунди), усереднений за всіма класами;

$\Lambda = \sum_{\forall E_i \in E^H} f_i^{max} + \sum_{\forall E_j \in E^S} \lambda_j$ – сумарна інтенсивність подій жорсткого та м’якого реального часу;

$E[S^2] = Var[S] + (E[S])^2$ – математичне очікування квадрату часу очікування у черзі ($с^2$);

$\rho = \Lambda \cdot E[S]$ – завантаження процесора.

Відповідно, ймовірність того, що подія типу i буде повністю оброблена до встановленого дедлайну (ймовірність своєчасної обробки події), така:

$$P_{Success,i} = Pr\left(T_{resp,\,i} \le D_i\right), \tag{2.14}$$

де D_i – обмеження на час обробки події (дедлайн).

Відповідно до встановлених припущень та обмежень, ймовірність неприйнятної затримки (рисунок 2.3) для події типу i:

$$P_{loss,i} = 1 - P_{Success,i}. \quad (2.15)$$

Розроблена модель реактивності формально описує здатність обчислювача та його МК, обробляти вхідний потік подій у межах заданих часових обмежень. Модель дає змогу оцінювати середні затримки та час очікування при довільному розподілі часу обслуговування, знаючи лише його середнє та дисперсію. Модель допускає два режими аналізу: детерміністичний (аналіз за найгіршим випадком, WCET, перевірки за EDF) і стохастичний (модель черги).

Для практичного використання розробленої моделі необхідно експериментальне визначення наступних параметрів:

– кількість тактів роботи процесорного ядра МК – C_i, потрібних для вторинної обробки і-тої події в найгіршому випадку;

– кількість тактів роботи процесорного ядра МК – C_{IRQ}, потрібних первинному обробнику переривань;

– накладні витрати (C_{Core}) на реалізацію таких операцій операційної системи як переключення контексту, роботу з об’єктами ядра.

2.2.4 Результати чисельного експерименту для перевірки моделі реактивності

Для перевірки моделі реактивності виконано чисельний експеримент, в ході якого імітувалась ситуація, коли в МК поступає пуассонівський потік подій (переривань) м’якого реального часу з інтенсивністю λ_i з детерміністичними часами обслуговування для первинної та вторинної обробок.

Правила моделювання в розробленому авторському програмному забезпеченні для чисельного експерименту наступні:

– первинна обробка має пріоритет над вторинною.

– якщо при надходженні події (переривання) обчислювач зайнятий первинною обробкою – подія ставиться в первинну чергу;

– після завершення первинної обробки для тієї ж події створюється "повідомлення" у вторинну чергу;

– вторинна обробка може початися тільки коли первинна черга порожня;

– вторинна обробка переривається і зберігає залишок часу, якщо в системі з’являється нова подія, яка вимагає первинного обслуговування;

– обслуговування не переривається під час первинної стадії.

Для моделювання використано подієвий (event-driven) підхід:

– події (зовнішні переривання) генеруються з інтервалами відповідно до експоненційного закону розподілення випадкової величини та отримують власний ідентифікатор завдання – job_id;

– для кожного завдання для наступного аналізу реєструються наступні відмітки часу: arrival – час появи переривання, primary_done – виконано первинну обробку, secondary_done – виконано вторинну обробку.

Чисельні експерименти, результати яких використовувались для перевірки розробленої моделі реактивності, проводились при наступних фіксованих вхідних параметрах:

– імітація обробки 10 000 подій (переривань);

– середня інтенсивність потоку випадкових подій незмінна – одиниця на одиничному часовому інтервалі. В чисельному експерименті не використовувалась конкретна фізична розмірність: години, хвилини секунди. Всі виміри часу є безрозмірними;

– тривалість первинної обробки фіксована – 0.1;

– тривалість вторинної обробки є зміною величиною в діапазоні [0, 0.9].

Після обробки вказаної кількості подій програмне забезпечення для чисельного експерименту розраховує наступні показники:

– середній час очікування у первинній черзі;

– середній час очікування у вторинній черзі;

– середній час від переривання до завершення вторинної обробки;

– максимальний час очікування у первинній черзі;

– максимальний час очікування у вторинній черзі;

– максимальний час від переривання до завершення вторинної обробки.

– середня довжина первинної черги;

– середня довжина вторинної черги;

– ймовірність порушення встановлених часових обмежень на час реакції.

Середній час очікування в чергах (рисунок 2.4) досягає ста одиниць часу, якщо система розрахована так, що її завантаженість на первинну та вторинну обробку складає 100 %. Якщо необхідно планувати так, щоби час виконання хоча-б половини завдань, що вирішуються, не перевищував одиницю, потрібно не завантажувати процесорне ядро МК більше, ніж на половину (в прикладі це час, потрібний для первинної обробки – 0.1 та для вторинної обробки – 0.4).

Максимальний час від переривання до завершення вторинної обробки (рисунок 2.5) досягає 400 одиниць часу при стовідсотковій завантаженості процесора. Щоб максимальний час від переривання до завершення вторинної обробки не перевищував одиницю, необхідно обмежити завантаженість процесора величиною 0.2 (в прикладі це ситуація, коли час потрібний для первинної обробки складає 0.1 та для вторинної обробки – 0.1).

Рисунок 2.6 демонструє довжину черги, яка визначає витрати оперативної пам’яті для підтримки запланованих черг. Відповідно до результатів моделювання, при повному завантаженні процесора необхідно розраховувати на середню довжину вторинної черги до 100 елементів.

На рисунку 2.7 представлено результати розрахунку залежності ймовірності порушення часових обмежень від величини цих часових обмежень. Для розглянутих варіантів (для демонстрації обрано варіанти, коли вторинна обробка вимагає 0.4, 0.5, 0.6 одиниць часу) ймовірність порушення часових обмежень не буде перевищувати 0.1, якщо обмеження складає 4 одиниці часу.

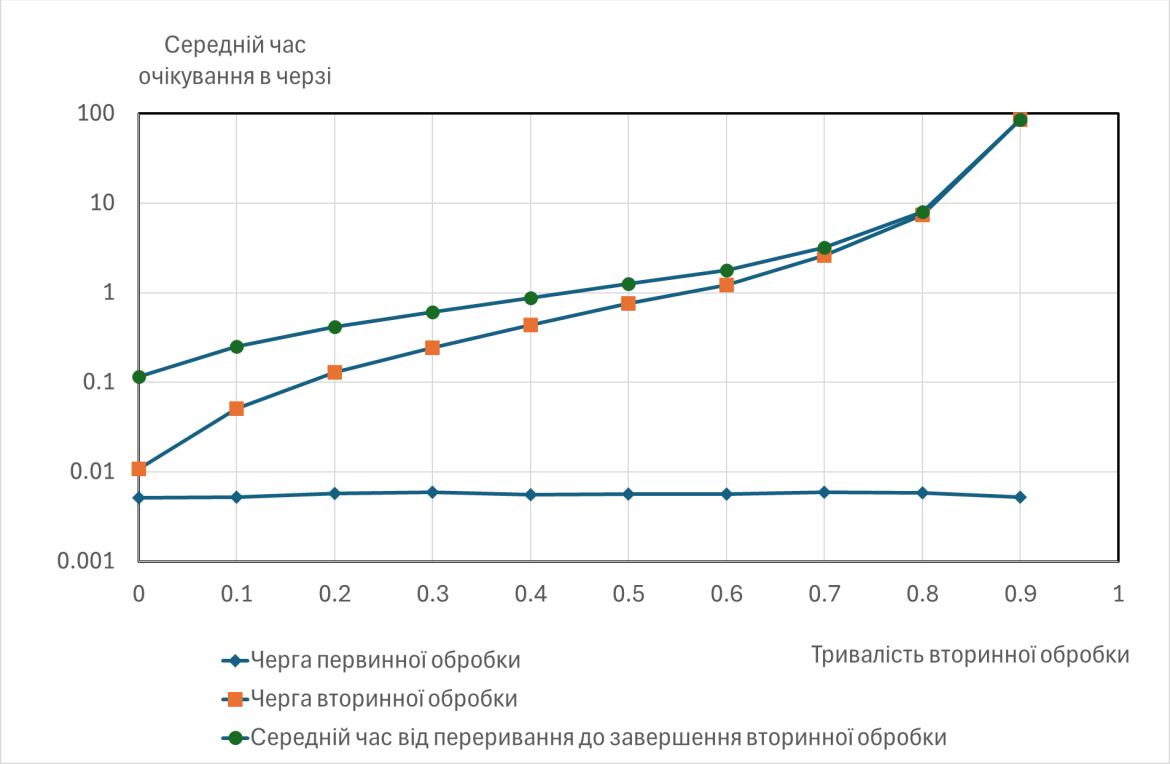

Рисунок 2.4 – Результати розрахунку середнього часу очікування в чергах та часу середнього часу від переривання до завершення обробки переривання

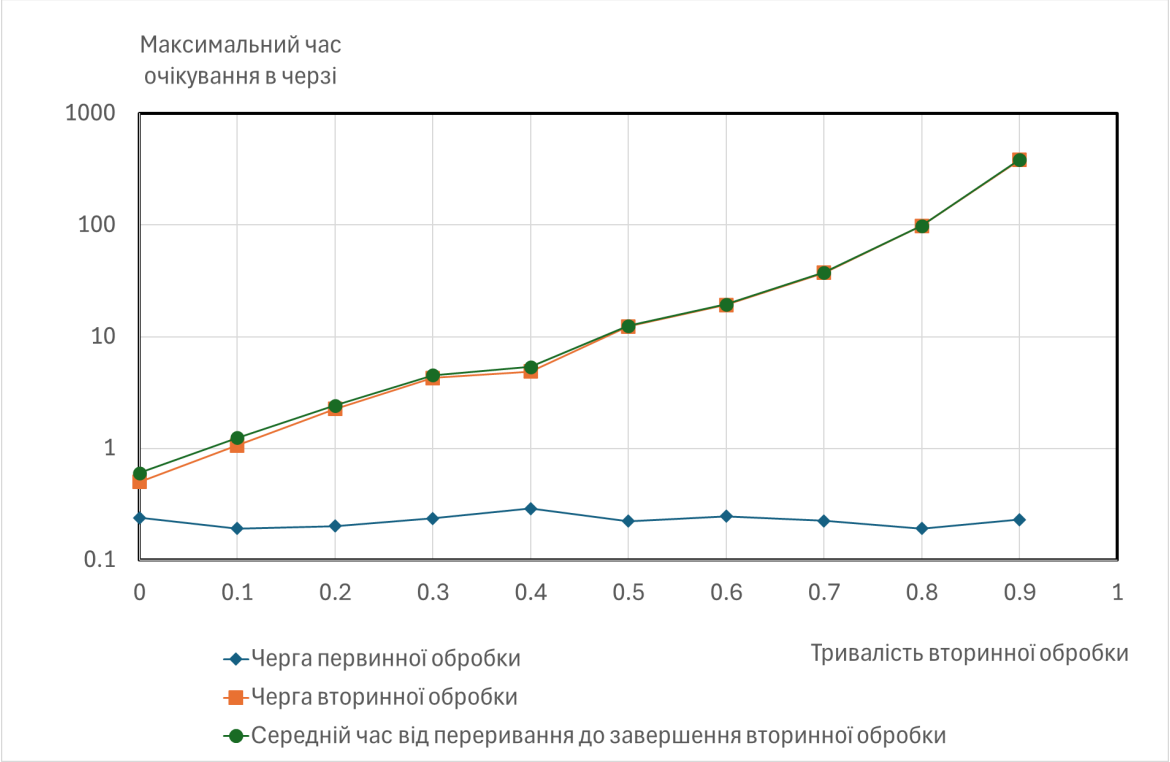

Рисунок 2.5 – Результати розрахунку максимального часу очікування в чергах та часу середнього часу від переривання до завершення обробки переривання

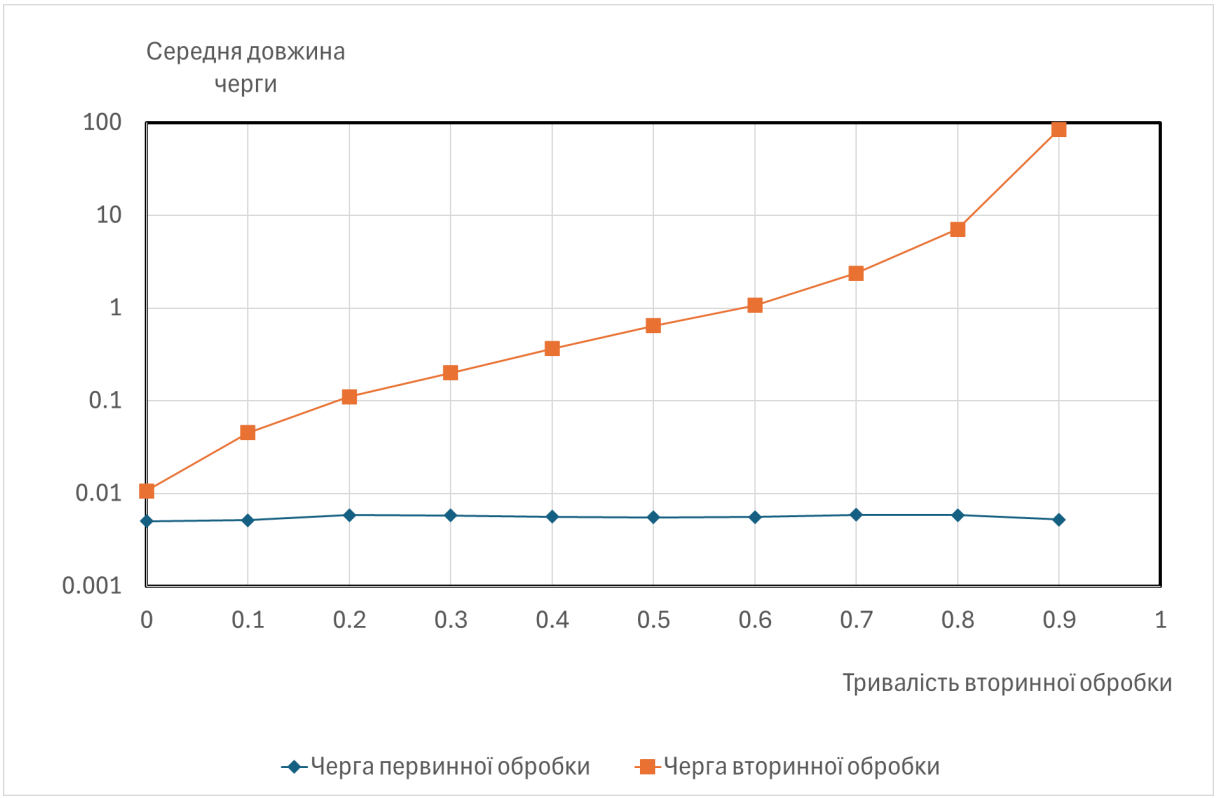

Рисунок 2.6 – Результати розрахунку середньої довжини черги

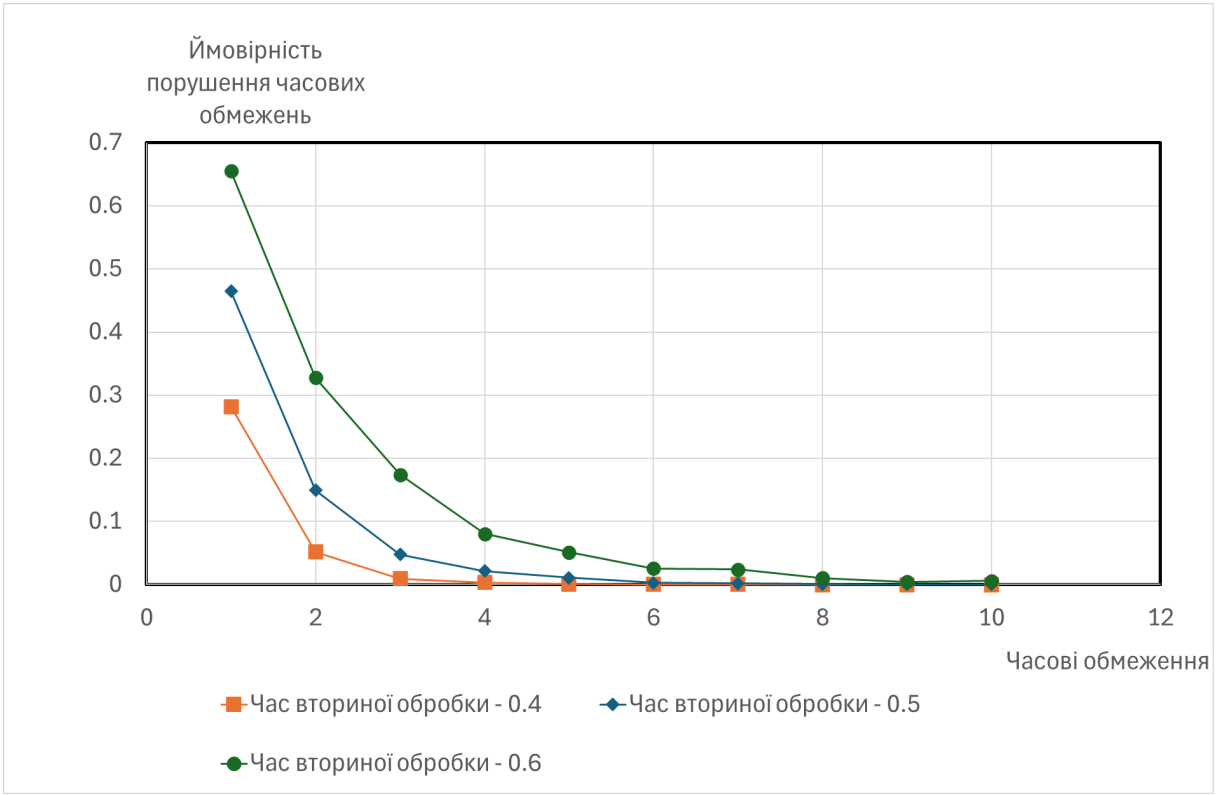

Рисунок 2.7 – Результати розрахунку залежності ймовірності порушення часових обмежень від величини цих часових обмежень

Таким чином, результати чисельного експерименту не суперечать гіпотезі про коректність запропонованої теоретичної моделі реактивності.

2.3 Модель продуктивності програмного забезпечення бортових обчислювачів наносупутників

2.3.1 Загальні положення

Обчислювальна продуктивність (P) бортового обчислювача наносупутника – це кількісна міра його здатності виконувати корисну роботу. До корисної роботи відносяться вторинні обробники переривань для обробки зовнішніх подій та фонові процеси. Обчислювальна продуктивність БО наносупутника має бути змодельована як комплексна функція, яка поєднує можливості апаратного забезпечення з динамічними потребами місії та стратегіями енергозбереження.

МК бортового обчислювача може знаходитись в чотирьох основних режимах $Modes = \{Active, Sleep, Wait, Backup\}$ (див. рисунок 2.2). Окрім основних режимів $Modes$ до можливих станів БО на деталізованій діаграмі станів (state diagram) МК (рисунок 2.8, таблиця 2.1) додано перехідні стани, які характеризують процеси, що відбуваються під час переключення в режим низького енергоспоживання та під час відновлення активної роботи. В режимах низького енергоспоживання $Sleep, Wait, Backup$ та перехідних режимах мікроконтролер бортового обчислювача не працює та, відповідно, не виконує жодної роботи.

Окрім деталізації перехідних режимів на рисунку 2.8 враховано, що в активному режимі МК виконує :

– прикладні завдання (корисну роботу), такі як обробка даних, керування апаратурою, навігація тощо;

– системні завдання для підтримки кооперативної багатозадачності шляхом диспетчеризації потоків після виклику системної функції taskYIELD;

– системні завдання з обробки переривань від системного таймера (System Tick Interrupt) та перепланування й диспетчеризації завдань після такого переривання;

– системні завдання з первинної обробки переривань від зовнішніх джерел (External Interrupt), що реалізуються операційною системою в процедурі обробника переривання (ISR) та полягають в ініціалізації об’єктів ядра, які дозволяють роботу вторинного обробника переривання.

2.3.2 Вихідні припущення та обмеження

1. Як розвиток моделі (див. рисунок 2.2) в перелік режимів включено режим ініціалізації – Init, в якому МК опиняється не тільки після ввімкнення живлення, але й після виходу зі стану $Backup$ (через рестарт).

2. Розширення множини режимів енергозбереження дозволяє врахувати не тільки режими енергозбереження, але, також, і перехідні стани під час переключення в режим низького енергоспоживання та під час відновлення активної роботи:

$$\begin{gathered} Sleep = \{SleepOn, SleepDo, SleepOff\}; \\ Wait = \{WaitOn, WaitDo, WaitOff\}; \\ Backup = \{BackupOn, BackupDo, BackupOff\}, \end{gathered} \tag{2.16}$$

де стани $SleepOn,\ WaitOn, BackupOn$ – описують переходи МК в відповідні режими енергозбереження;

$SleepDo,\ WaitDo, BackupDo$ – власне режими енергозбереження;

$SleepOff,\ WaitOff, BackupOff$ – режими відновлення активної роботи після режимів енергозбереження.

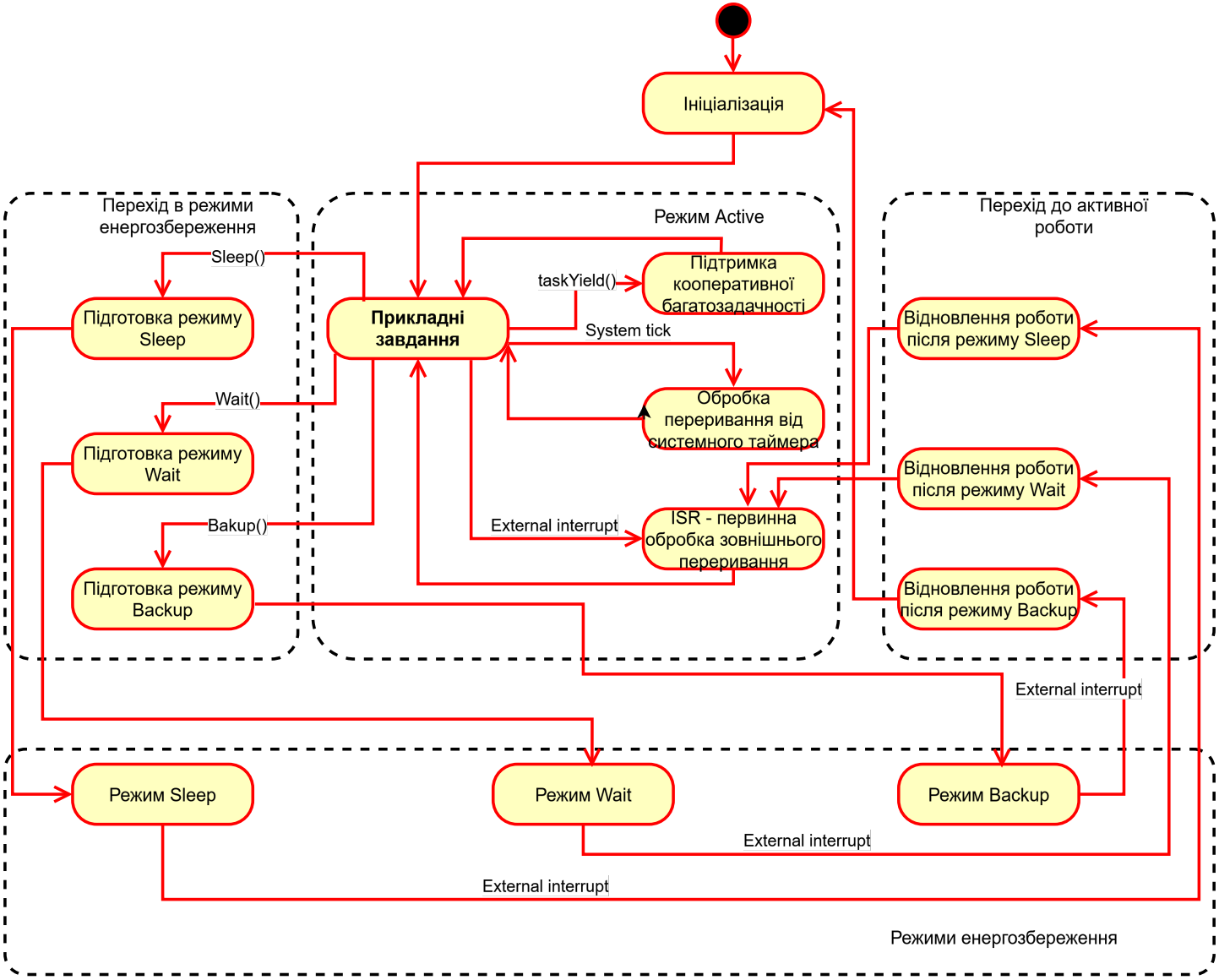

Рисунок 2.8 – Деталізована діаграма станів (state diagram) МК

Таблиця 2.1 Матриця суміжності графу станів та переходів

№	Стан	№															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	$Init$	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2	$SleepOn$	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	$SleepDo$	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4	$SleepOff$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
5	$WaitOn$	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	$WaitDo$	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
7	$WaitOff$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	$BackupOn$	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9	$BackupDo$	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
10	$BackupOff$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	BG	0	1	0	0	1	0	0	1	0	0	0	1	1	1	1	1
12	HRT	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
13	SRT	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
14	CoSwitch	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
15	$SysTickI$	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
16	ISR	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

3. В активному режимі $Active$ програмне забезпечення може виконувати одне з наступних завдань множини $Active$:

$$Active = \{UserTasks, CoSwitch, SysTickI, ISR\}, \quad (2.17)$$

де $UserTasks$ – множина прикладних завдань;

$SysTickI$ (System Tick Interrupt) – завдання з обробки переривань з фіксованим інтервалом від таймера;

ISR (Interrupt Service Routine) – завдання з первинної обробки переривань від зовнішніх джерел;

$CoSwitch$ (Cooperative context switch) – завдання переключення контексту для підтримки кооперативної багатозадачності.

4. Припускаємо, що витрати часу процесорного ядра на переключення контексту для підтримки кооперативної багатозадачності за фіксований інтервал часу будуть пропорційні здобутку кількості фонових завдань (Background Tasks) та кількості періодів системного таймеру на цьому інтервалі часу;

5. В свою чергу, розділимо множину прикладних завдань $UserTasks$ активного режиму $Active$ на три підмножини:

$$UserTasks = \{BG, HRT, SRT\}, \quad (2.18)$$

де BG – множина фонових завдань (Background Tasks), для опису яких достатньо єдиного кількісного показника – фонової завантаженості (коефіцієнта утилізації) процесора;

HRT – множина вторинних обробників переривань подій жорсткого реального часу (Hard Real-Time Event Handlers);

SRT – множина вторинних обробників подій м’якого реального часу (Soft Real-Time Event Handlers).

Таким чином, загальне припущення полягає в тому, що деталізована діаграма станів МК надає можливість врахувати всі режими роботи апаратно-програмного забезпечення МК – FOMS (Full Operating Modes Set):

$$\text{FOMS} = \{Init\} \cup Active \cup Sleep \cup Wait \cup Backup\ . \qquad (2.19)$$

Таблиця 2.1 узагальнює можливі режими мікроконтролера та переходи між ними у вигляді матриці суміжності.

2.3.3 Функційні залежності моделі продуктивності

Головне завдання цього підрозділу – визначити, яку частину процесорного часу в активному режимі МК витратить на прикладні завдання (корисну роботу), такі як обробка даних, керування апаратурою, навігація тощо. Така частина часу процесорного ядра змістовно є коефіцієнтом корисної дії обробки даних МК в режимі $Active$ та визначається як співвідношення витрат часу на вирішення $UserTasks$ та загального часу перебування в режимі $Active$:

$$\eta_{Data} = \frac{\tau_{UserTasks}}{\tau_{Active}} = \frac{\tau_{UserTasks}}{\tau_{UserTasks}+\tau_{SystemTasks}} = 1 - \frac{\tau_{SystemTasks}}{\tau_{Active}}\ , \qquad (2.20)$$

Загальний час роботи МК в активному режимі τ_{Active} на одиничному інтервалі часу складається з суми часів перебування МК в доступних станах:

$$\begin{gathered} \tau_{Active} = \tau_{UserTasks} + \tau_{SystemTasks}; \\ \tau_{UserTasks} = \tau_{BG} + \tau_{HRT} + \tau_{SRT}; \\ \tau_{SystemTasks} = \tau_{\text{CoSwitch}} + \tau_{SysTickI} + \tau_{ISR}, \end{gathered} \qquad (2.21)$$

де τ_{BG}=η_{BG} – відносна завантаженість процесорного ядра МК фоновими завданнями, визначається під час проєктуванні системи;

τ_{HRT} – максимальні витрати часу на роботу вторинних обробників переривань подій жорсткого реального часу (див. 2.10):

$$\tau_{HRT} = \sum_{\forall E_i \in E^H} f_{E_i}^{max} \cdot T_{E_i}; \quad (2.22)$$

τ_{SRT} – середні витрати часу на роботу вторинних обробників переривань подій м'якого реального часу (див. 2.10):

$$\tau_{SRT} = \sum_{\forall E_j \in E^S} \lambda_{E_j} \cdot T_{E_j}; \quad (2.23)$$

витрати часу на переключення контексту для підтримки кооперативної багатозадачності пропорційні здобутку частоти системного таймера – F_{timer} та кількості завдань в системі – $\|BG\|$:

$$\tau_{\mathrm{CoSwitch}} = T_{\mathrm{taskYIELD}}(F_{clk}) \cdot F_{timer} \cdot \|BG\|, \quad (2.24)$$

де $T_{\mathrm{taskYIELD}}$ – час виконання системної функції taskYIELD();

витрати часу на завдання з обробки переривань з фіксованим інтервалом від таймера дорівнюють:

$$\tau_{SysTickIrq} = T_{Timer}(F_{clk}) \cdot F_{timer}, \quad (2.25)$$

де T_{Timer} – час роботи обробника переривання від таймера;

витрати часу на завдання з первинної обробки переривань від зовнішніх джерел:

$$\tau_{ISR} = \left(\sum_{\forall E_i \in E^H} f_{E_i}^{max} + \sum_{\forall E_j \in E^S} \lambda_{E_j}\right) T_{IRQ_p}(F_{clk}), \quad (2.26)$$

де $T_{IRQ_p}(F_{clk})$ – залежність часу роботи первинного обробника переривання від тактової частоти.

Для $\tau_{Active} = 1$, формулу 2.20 можна представити зручніше:

$$\eta_{Data} = 1 - F_{timer} \cdot (T_{\text{taskYIELD}}(F_{clk}) \cdot \|BG\| + T_{Timer}) - \left(\sum_{\forall E_i \in E^H} f_{E_i}^{max} + \sum_{\forall E_j \in E^S} \lambda_{E_j} \right) T_{IRQ_p(F_{clk})}. \quad (2.27)$$

Змістовно формула 2.27 визначає відносний процесорний час, доступний для вирішення прикладних завдань.

2.4 Модель енергоспоживання бортового обчислювача

Для синтезу моделі енергоспоживання розглянемо авторський БО «Боривітер 0.1» (далі «Боривітер»), що побудований на базі типового для сучасних наносупутників, мікроконтролері ARM® Cortex-M7™, та за типовою архітектурою бортового обчислювача.

2.4.1 Вихідні припущення та обмеження

Енергоспоживання бортового обчислювача наносупутника визначається через споживання енергії його складових:

– основного мікроконтролера ARM Cortex-M7;

– допоміжного мікроконтролера ATSAMD21E17A;

– контролерів периферійних інтерфейсів, контролерів вбудованих сенсорів та різних типів пам’яті (FRAM, MRAM, PSRAM) тощо.

Енергоспоживання залежить від режимних параметрів – тактової частоти (F_{clk}), напруги живлення (V_{dd}) та режиму роботи (FOMS).

На описовому рівні відомо, що:

– зі зростанням тактової частоти споживана потужність збільшується лінійно;

– зі зростанням напруги – квадратично;

– при зменшенні тактової частоти або переході в енергозберігаючі режими енергоспоживання знижується.

Частково ці питання розглядались раніше в наукових публікаціях, що надають загальне бачення проблематики:

– окремий аналіз енергоспоживання різних типів пам'яті (FRAM, MRAM, RRAM тощо) для MCU виконано в роботі [145];

– аналіз впливу режимів та технологій енергозбереження на споживання енергії мікроконтролерами STM32F407 та ARM Cortex-M0+ Core SoC [146, 147].

2.4.2 Функційні залежності математичної моделі енергоспоживання бортового обчислювача

Для аналізу енергоспоживання бортового обчислювача необхідно розширити визначення множини режимів з формули (2.2). На відміну від (2.2) врахуємо існування перехідних режимів (2.16). Тоді, відповідно до деталізованої діаграми станів мікроконтролера (див. рисунок 2.8), всі режими роботи апаратно-програмного забезпечення мікроконтролера – це FOMS (Full Operating Modes Set) (2.19).

Якщо визначити послідовність режимів, як множину:

$$T = \{\langle t_k,\ foms_k \rangle\}_{k=0}^{n}, \tag{2.28}$$

де $t_k \in \Re$ – моменти часу, коли в мікроконтролері відбувається зміна режиму на $foms_k \in FOMS$, впорядковані так, що $t_0 < t_1 < \cdots < t_n$,

то функція стану процесора – це відображення:

$$\phi\colon T \rightarrow FOMS. \tag{2.29}$$

Загальне споживання енергії на досліджуваному інтервалі часу $[t_0, t_n]$ – це інтеграл від миттєвої потужності, що може бути представлений наближено сумою:

$$E = \int_{t_0}^{t_n} e(t)dt \approx \sum_{k=0}^{n-1} e_{foms_k} \cdot (t_{k+1} - t_k), \quad (2.30)$$

де $e(t)$ – залежність споживаної електричної потужності від часу – t,

e_{foms_k} – споживана потужність на режимі $foms_k$.

Також необхідно врахувати, що протяжність перехідних режимів (див. формулу 2.16) визначається схемними рішеннями мікроконтролера та залежить тільки від тактової частоти F_{clk}. Формула (2.31) визначає множину перехідних (англ. transition) режимів - $FOMS_{tr}$, а (2.32) визначає залежність протяжності перехідних режимів від тактової частоти F_{clk}:

$$FOMS_{tr} = \begin{Bmatrix} SleepOn,\ SleepOff, \\ WaitOn,\ WaitOff, \\ BackupOn,\ BackupOff \end{Bmatrix}, \quad (2.31)$$

$$\forall foms_k \in FOMS_{tr} \left(\tau_{foms_k}(F_{clk}) = c_{foms_k} + d_{foms_k} \cdot \frac{1}{F_{clk}}\right), \quad (2.32)$$

де c_{foms_k}, d_{foms_k} – коефіцієнти оберненої залежності часу $\tau_{foms_k}(F_{clk})$ від тактової частоти F_{clk}.

Як припущення, будемо вважати, що споживана електрична потужність в усіх стаціонарних режимах $foms_k \notin FOMS_{tr}$ лінійно зростає з ростом тактової частоти F_{clk} та квадратично залежить від напруги живлення V_{dd} [148, 149] (рисунок 2.9). З урахуванням припущення, що напруга живлення – незмінна та дорівнює номінальній величині залежність спрощується:

$$e_{foms_k}(F_{clk}) = a_{foms_k} + b_{foms_k} \cdot F_{clk}, \quad (2.33)$$

де перший доданок описує статичну складову споживання енергії, а другий – динамічну, тобто ту, що характеризує корисну роботу процесора;

a_{fomsk}, b_{fomsk} – коефіцієнти лінійної моделі.

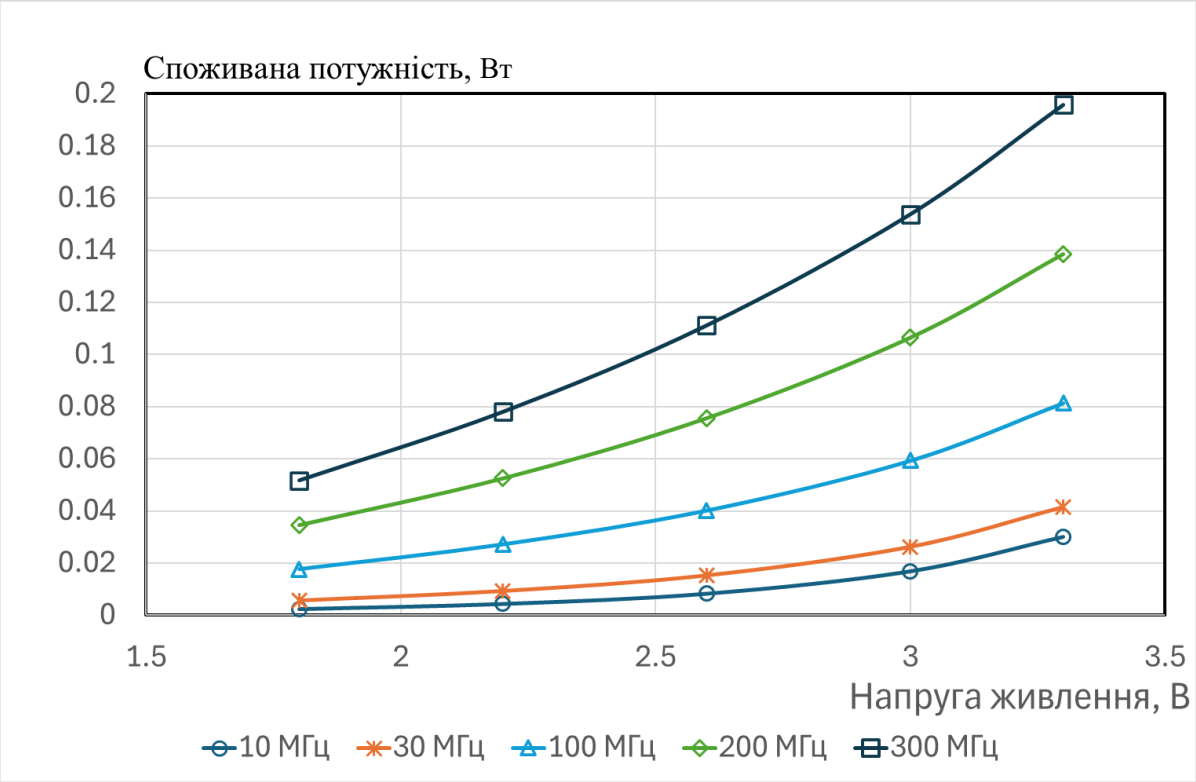

Рисунок 2.9 – Залежність споживаної електричної потужності від напруги живлення та тактової частоти мікроконтролера (приклад)

Розглянемо ситуацію, коли існує деяка задача з відомою трудомісткістю (кількістю машинних інструкцій мікроконтролера), тоді час вирішення цієї задачі є обернено пропорційним тактовій частоті (рисунок 2.10), а витрати енергії на вирішення цієї задачі є мінімальними при налаштуванні мінімальної напруги живлення та використанні максимально допустимої тактової частоти (рисунок 2.11).

Висновками з наведеного модельного розрахунку є наступне:

– зниження тактової частоти мікроконтролера є енергетично неефективним, оскільки на загальне енергоспоживання негативно впливає статична складова споживання енергії;

– зменшення напруги живлення від рекомендованого виробником є небажаним, оскільки негативно впливає на надійність, а надійність бортового обчислювача наносупутника є ключовою системною вимогою. Надійність бортового обчислювача не є предметом вивчення в роботі;

– для досягнення компромісу між мінімумом споживання енергії та максимуму обчислювальної продуктивності й реактивності бортового обчислювача наносупутника необхідно будувати відповідні методи, алгоритми та програмні рішення керування енергозберігаючими режимами $Sleep, Wait, Backup$.

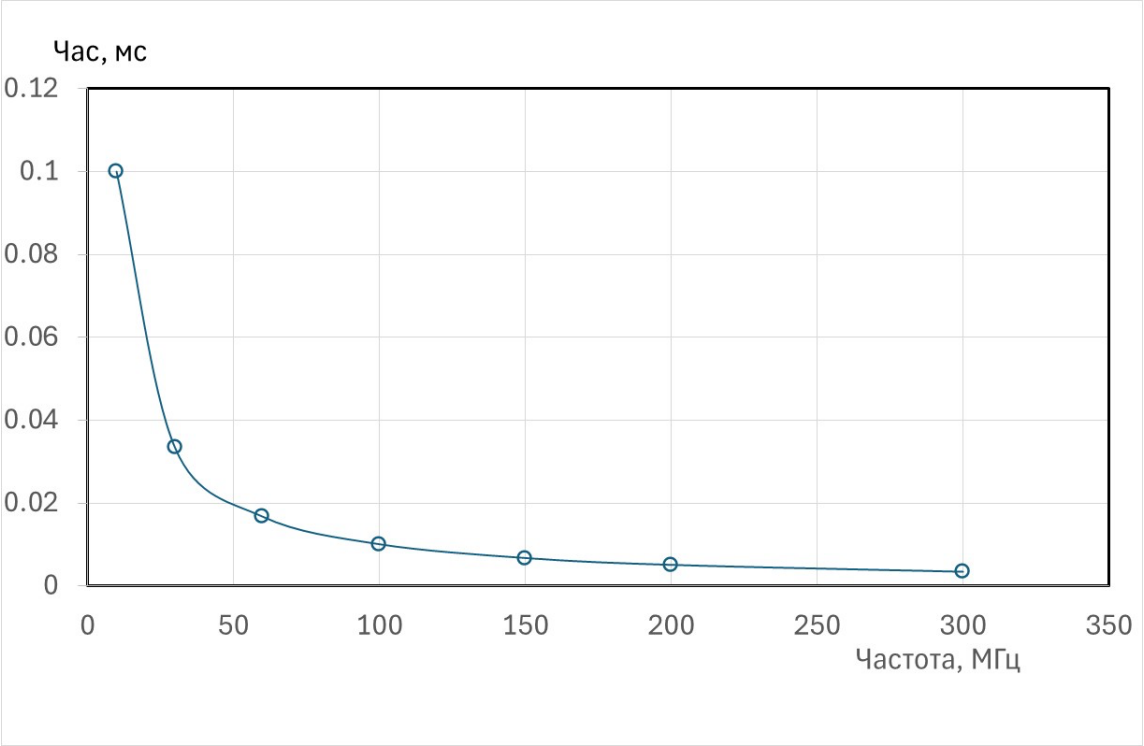

Рисунок 2.10 – Залежність витраченого часу від тактової частоти мікроконтролера

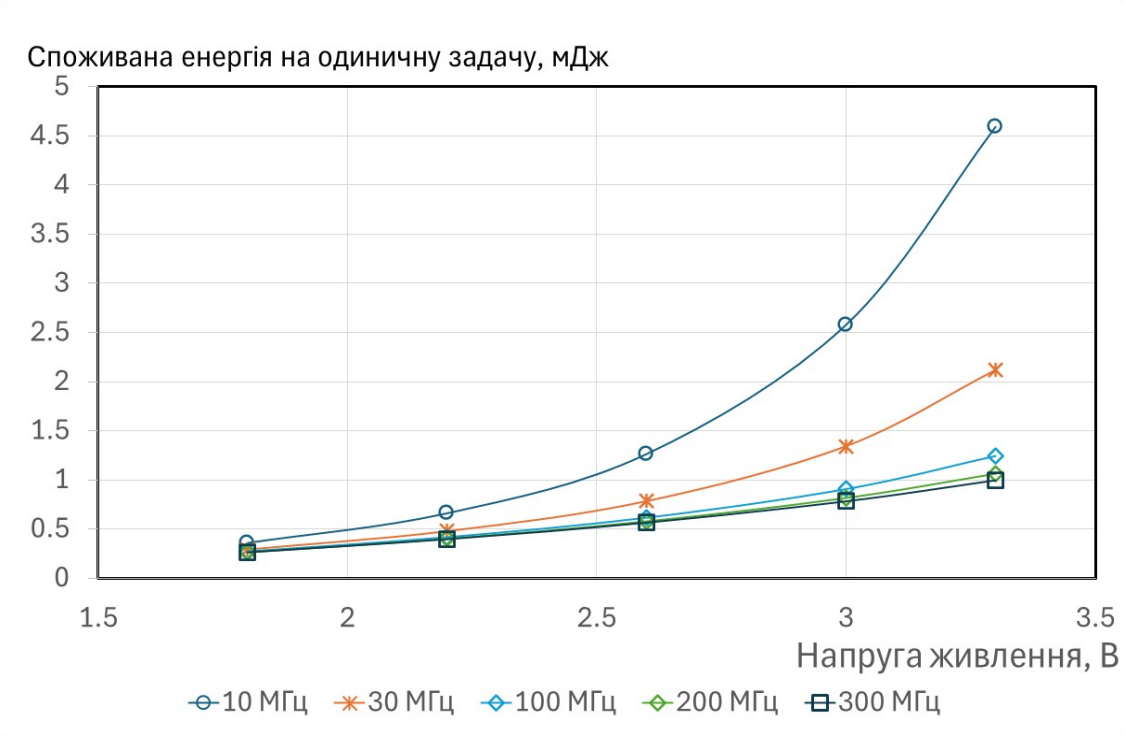

Рисунок 2.11 – Залежність споживаної енергії від напруги живлення та тактової частоти мікроконтролера

Необхідне також наступне уточнення. Під час перехідних режимів $FOMS_{tr}$ відбувається динамічна зміна споживаної електричної потужності, тому надалі в роботі використовується спрощений розрахунок споживаної потужності для $\forall foms_k \in FOMS_{tr}$:

$$e_{foms_k}(F_{clk}) = \frac{prev\left(e_{foms_k}(F_{clk})\right)+next\left(e_{foms_k}(F_{clk})\right)}{2}, \qquad (2.34)$$

де $prev\left(e_{foms_k}(F_{clk})\right), next\left(e_{foms_k}(F_{clk})\right)$ – споживана електрична потужність на стаціонарних режимах, що були до ($prev(\quad)$) та після ($next(\quad)$) перехідного.

Висновки до розділу 2

У другому розділі розроблено комплексну математичну модель, що формалізує взаємозв’язок між енергоспоживанням, обчислювальною

продуктивністю та реактивністю бортового обчислювача наносупутника. Модель враховує апаратно-програмні параметри мікроконтролера, режими його роботи та характеристики потоку подій і дозволяє кількісно оцінювати компроміси між зазначеними характеристиками в умовах обмеженого енергетичного балансу. Визначено основне технічне протиріччя системи, що полягає у необхідності одночасного зниження енергетичних витрат і підтримання заданого рівня продуктивності та реактивності, що зумовлює потребу в адаптивних методах керування.

Модель у нотації теорії множин формалізує систему відображень між множинами режимних параметрів мікроконтролера, характеристиками потоків подій і цільовими показниками ефективності. Введення узагальненої множини режимів роботи FOMS дозволило описати функціонування бортового обчислювача у вигляді скінченного автомата зі станами та переходами, що супроводжуються часовими та енергетичними витратами. Такий підхід забезпечує кількісну оцінку впливу режимів енергозбереження та переходів між ними на продуктивність і реактивність системи.

Формалізована модель реактивності описує здатність бортового обчислювача своєчасно обробляти зовнішні події жорсткого та м’якого реального часу з урахуванням інтенсивності потоку подій, доступного процесорного часу та накладних витрат операційної системи. Реактивність визначено як імовірність своєчасної обробки подій, що дозволяє виконувати детерміністичний і стохастичний аналіз ефективності бортового програмного забезпечення.

У моделі обчислювальної продуктивності встановлено залежність корисної обчислювальної роботи від режиму функціонування мікроконтролера, частки часу виконання прикладних задач і накладних витрат операційної системи. Математична модель енергоспоживання описує залежність споживаної потужності від напруги живлення, тактової частоти та режиму роботи мікроконтролера. Показано, що зниження тактової частоти без відповідного зменшення напруги є енергетично неефективним, а досягнення

компромісу між енергоспоживанням, продуктивністю та реактивністю можливе лише за рахунок адаптивного керування режимами енергозбереження.

Таким чином, розроблена комплексна математична модель забезпечує єдину формалізовану основу для аналізу та оптимізації енергоспоживання, продуктивності й реактивності бортових обчислювачів наносупутників, а, також, створює підґрунтя для розроблення адаптивних методів керування режимами роботи.

Отже, у другому розділі вирішено третє завдання дисертаційного дослідження, а отримані результати відображено в наукових публікаціях [150, 151].

РОЗДІЛ 3 МЕТОДИ ІДЕНТИФІКАЦІЇ КОМПЛЕКСНОЇ МАТЕМАТИЧНОЇ МОДЕЛІ ТА АДАПТИВНОГО РЕГУЛЮВАННЯ РЕЖИМІВ РОБОТИ БОРТОВИХ ОБЧИСЛЮВАЧІВ НАНОСУПУТНИКІВ CUBESAT

3.1 Метод ідентифікації параметрів комплексної математичної моделі

Комплексна математична модель енергетичних витрат, обчислювальної продуктивності та реактивності бортового програмного забезпечення містить набір параметрів, значення яких не можуть бути задані апріорно з достатньою точністю. Ці параметри залежать від конкретної апаратної реалізації бортового обчислювача, умов його експлуатації та поточного обчислювального навантаження.

З огляду на це, у роботі розв’язується задача ідентифікації параметрів математичної моделі, яка використовується для передпроєктного оцінювання (етапу аванпроєкту) бортового обчислювача та його програмного забезпечення, а, також, для уточнення моделі в процесі функціонування системи. Ідентифіковані параметри гарантують коректну роботу алгоритмів адаптивного регулювання режимів роботи бортового обчислювача з урахуванням обмежень реального часу.

Метод ідентифікації параметрів реалізовано через наступну послідовність етапів:

– обґрунтування припущень та обмежень щодо вимог до архітектури та алгоритмів ПЗ;

– формалізація задачі ідентифікації параметрів комплексної математичної моделі;

– планування експерименту, визначення методів обробки результатів;

– проведення експерименту, обробка результатів та перевірка адекватності моделей.

3.1.1 Обґрунтування припущень та обмежень щодо архітектури та алгоритмів ПЗ

Метод ідентифікації параметрів математичної моделі передбачає попереднє експериментальне обґрунтування архітектурних та алгоритмічних припущень, що визначають область застосовності моделі та допустимі режими функціонування бортового програмного забезпечення. Отримані результати використовуються для звуження простору розглядуваних рішень і формування коректної постановки задачі ідентифікації параметрів.

На цьому етапі розглянуто наступні два рішення.

1. Експериментальна перевірка доцільності використання контейнеризації на основі інтерпретатора WebAssembly WASM3 у складі бортового програмного забезпечення. Контейнеризація, згідно визначення від компанії IBM, полягає в упакуванні певного функціонального додатку ПЗ в окремий файл – контейнер, який стабільно розгортається та працює на будь-якій інфраструктурі [152].

Основна перевага контейнеризації полягає в тому, що такий підхід дозволяє «написати код один раз (на одній платформі) і запустити будь-де». Контейнеризація також пропонує інші переваги, такі як ізоляція та стійкість до помилок [153], легке керування, спрощена безпека, тощо. Оскільки значення вбудованих систем та відповідних бізнес-застосунків зростає, а потужність апаратури збільшується, то розвиток мікросервісної архітектури та її упакування в контейнери на вбудованих системах просувається вперед.

Щоб оцінити додаткові витрати процесорних ресурсів при переході на контейнерну архітектуру зроблено експериментальне порівняння обчислювальної продуктивності на двох програмних архітектурах: нативний застосунок «bare metal», розроблений за допомогою мови С без операційної системи, та застосунок, розроблений на мовах С та С++ кросс-скомпільований для інтерпретатору WASM3, який визначає портативний формат двійкового коду. Експеримент виконано для порівняння продуктивності МК:

– чотирьох алгоритмів з різною обчислювальною складністю (таблиця 3.1), серед яких тільки алгоритм швидкого перетворення Фур'є (англ. Fast Fourier Transfer, FFT), має велику кількість операцій з даними у форматі float;

– шести різних розмірів вхідного масиву даних, які визначаються як ряд $N = 2n$, оскільки алгоритм FFT має саме таке обмеження: N = 64, 128, 256, 512, 1024, 2048.

Таблиця 3.1 Алгоритми, які використані в експерименті

№	Клас часової складності	Алгоритм	Часова складність в О-нотації
1	Лінійний час	CRC – обчислення контрольної суми	O(N)
2	Лінійно-логарифмічний час	FFT – швидке перетворення Фур'є	O(N log N)
3	Квадратичний час	BS(P) Бульбашкове сортування з масивом, що попередньо відсортован	$O(N^2)$
4		BS(R) Бульбашкове сортування з масивом, що відсортован у зворотному порядку	

Результати експерименту, що відображені на рисунку 3.1, показали, що використання WASM3 призводить до суттєвого зростання часових витрат в 25-50 разів. Такий результат унеможливлює дотримання вимог реального часу для бортових застосувань. У зв’язку з цим відповідні архітектурні рішення були виключені з подальшого розгляду, а математична модель та процедура ідентифікації параметрів формувались для нативного виконання програмного коду без додаткових інтерпретаційних шарів. Докладно планування, проведення експерименту та аналіз його результатів викладено в статті [154].

2. Для визначення механізмів енергозбереження, які можуть бути враховані в математичній моделі та процедурі ідентифікації параметрів, виконано експериментальну перевірку можливості оптимізації енергоспоживання бортового обчислювача шляхом динамічного масштабування частоти (DFS) та застосування алгоритму швидкого виконання

та засинання (англ. race-to-dark, RTD). RTD - це поширений метод енергозбереження, коли МК працює на максимальній частоті, щоб якомога швидше виконати обчислення та повернутися до сну (режиму енергозбереження) або перейти до мінімально можливої тактової частоти.

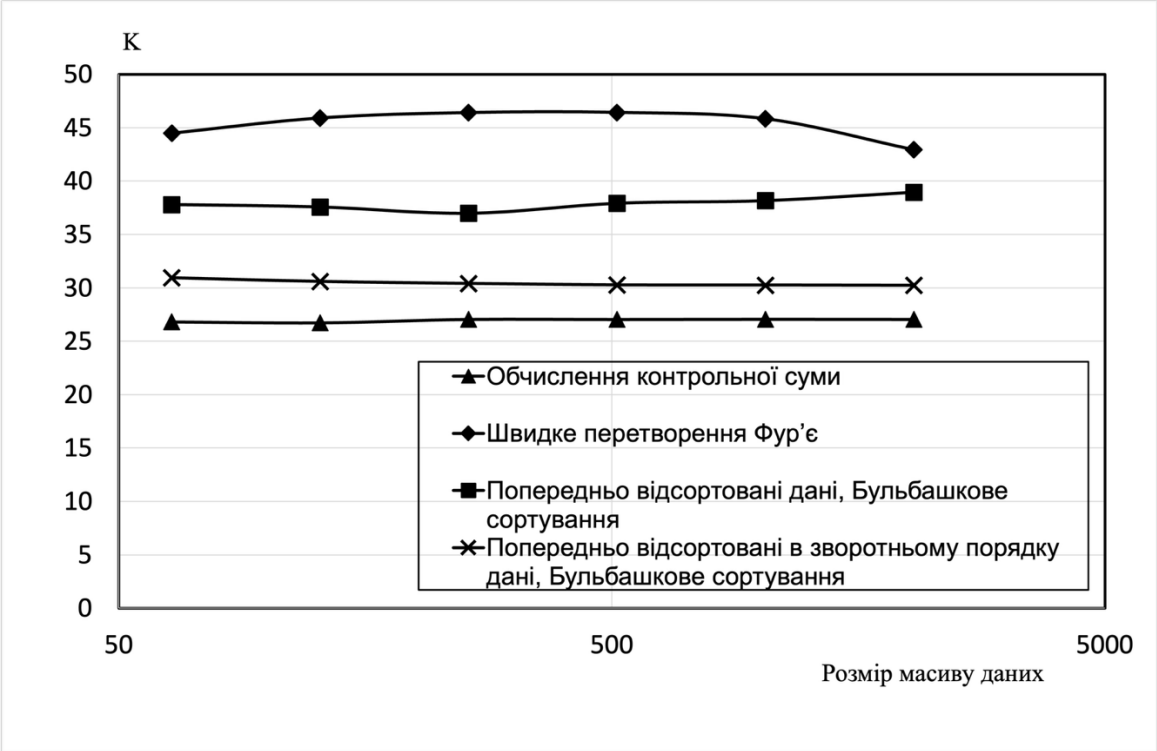

Рисунок 3.1 – Відносні накладні витрати часу при використанні контейнеризації

Експеримент проводився на платформі, яка виконувала лише одне циклічно повторюване завдання, в якості якого використано завдання сортування цілочисельного масиву за допомогою бульбашкового алгоритму. Після кожного циклу сортування інструкція МК – очікування на переривання (англ. Wait for Interrupt, WFI), переводить процесор у режим сну, де функції процесора зупинені, а системні годинники включені. Споживання енергії в цьому режимі цілком залежить від роботи периферійного обладнання. Далі переривання за таймером викликає пробудження процесора та йде наступне виконання сортування.

Для вимірювання часу виконання алгоритму при різних комбінаціях частоти та напруги використана системна функція FreeRTOS. Для вимірювання споживаної бортовим комп’ютером електроенергії, використовувалося високоточне стендове джерело живлення постійного струму Keithley 2281S, що гарантує точність вимірювань (похибка часового інтервалу – не гірше 15 мс, похибка потужності споживаної електроенергії - не гірше 0.0001 Вт).

Постановка завдання. Якщо відомий коефіцієнт завантаження процесора МК при певному наборі завдань на максимальній тактовій частоті процесора, то для економії енергії потрібно обрати стратегію енергозбереження: або динамічне регулювання частоти, або використанням алгоритму RTD, або комбінацію цих режимів енергозбереження.

Критерієм оптимізації є мінімальне енергоспоживання бортового комп'ютера при різних режимах енергозбереження:

$$P(\eta) = min\big(P_{DFS}(\eta), P_{RTD}(\eta), P_{DFS+RTD}(\eta)\big), \qquad (3.1)$$

де η – коефіцієнт утилізації процесора;

$P_{DFS}(\eta), P_{RTD}(\eta), P_{DFS+RTD}(\eta)$ – витрати енергії на роботу бортового комп'ютера при застосуванні таких підходів до енергозбереження:

– динамічне масштабування частоти (DFS) [155];

– алгоритм RTD виконує завдання максимально швидко, щоб процесор можна було перевести в сплячий режим, який мінімізує струм витоку [156];

– комбінований алгоритм (DFS + RTD), коли програмне забезпечення встановлює мінімальну тактову частоту після роботи на максимальній частоті перед переходом у сплячий режим.

В результаті експерименту отримано оптимістичну оцінку можливості енергозбереження за рахунок використання програмного керування режимами роботи комп’ютера. Оцінка, що зображена на рисунку 3.2, є оптимістичною, оскільки не враховує витрати на роботу планувальника та диспетчера операційної системи. Водночас, отримані кількісні результати є основою для

оптимізації конструкції апаратних компонентів бортового комп'ютера. Отримані кількісні результати свідчать про те, що використання DFS є більш вигідним, ніж RTD, але необхідно враховувати, що використання DFS зменшить реактивність системи. Тому необхідно буде шукати певний компроміс між енергозбереженням і реактивністю системи.

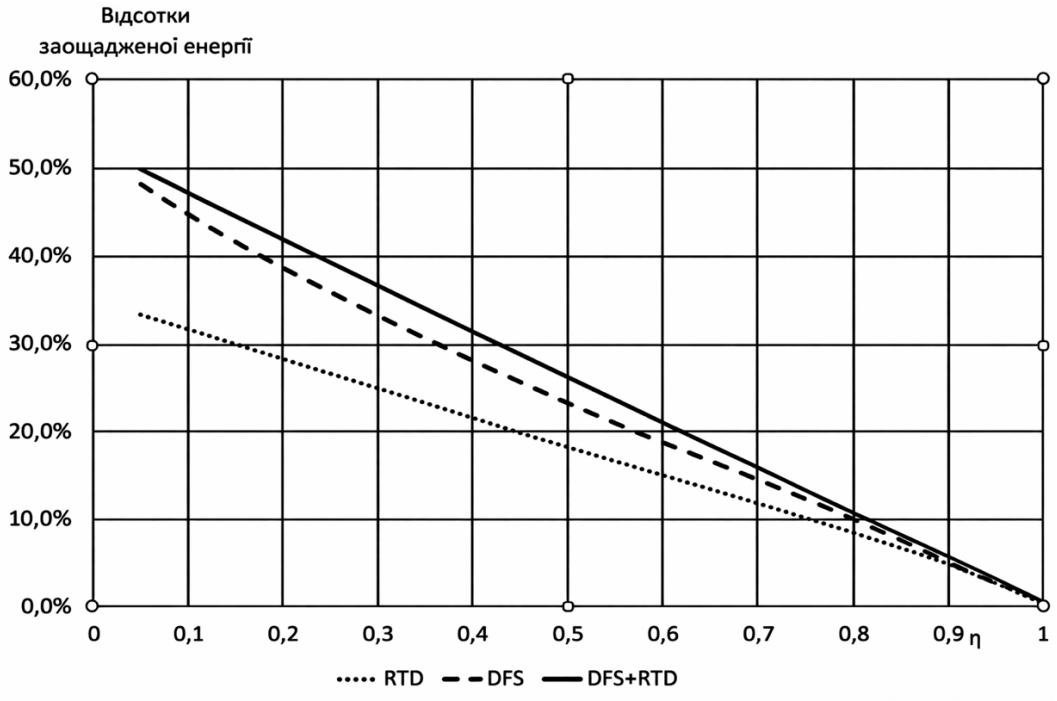

Рисунок 3.2 – Залежність збереженої бортовим комп'ютером енергії від η при використанні різних режимів енергозбереження

Використання DFS та політики RTD не забезпечує суттєве зменшення енергетичних витрат без порушення вимог до реактивності бортового програмного забезпечення. У зв'язку з цим, ці механізми не були включені до математичної моделі, а ідентифікація параметрів виконувалась для режимів роботи з фіксованою тактовою частотою МК. Докладно планування, проведення експерименту та аналіз його результатів викладено в статті [157].

Отримані негативні результати за двома перевіреними рішеннями мають принципове значення, оскільки дозволили виключити архітектурні та алгоритмічні рішення, несумісні з вимогами реактивності, та сформувати коректну постановку задачі ідентифікації параметрів математичної моделі.

3.1.2 Формалізація задачі ідентифікації параметрів комплексної математичної моделі

Задача ідентифікації параметрів комплексної математичної моделі формулюється за таких припущень:

– бортове програмне забезпечення виконується нативно без додаткових інтерпретаційних або контейнерних шарів;

– режими роботи з динамічним масштабуванням частоти та алгоритмами типу RTD - не використовуються;

– напруга живлення встановлена незмінною, та відповідає рекомендаціям виробника МК у типовому середовищі застосування БО;

– зміни параметрів зумовлені нестаціонарністю навантаження та умов експлуатації.

В загальному вигляді вектор вимірюваних характеристик $\boldsymbol{Y}(k)$ залежить від вектору вхідних змінних $\boldsymbol{X}(k)$ та шуму $\boldsymbol{\upsilon}(k)$:

$$\boldsymbol{Y}(k) = \boldsymbol{F}(\boldsymbol{X}(k),\ \boldsymbol{\Theta}) + \boldsymbol{\upsilon}(k), \tag{3.2}$$

де $\boldsymbol{\Theta}$ – вектор невідомих параметрів, що підлягають ідентифікації,

k - номер вимірювального кроку, що відповідає дискретному моменту часу та визначається періодом збору експериментальних даних та виконанням одного кроку ідентифікації параметрів;

вектор $\boldsymbol{\upsilon}(k)$ описує адитивні збурення та похибки вимірювань, а, також, невраховані динамічні ефекти, зумовлені особливостями функціонування бортового обчислювача та програмного забезпечення.

Відповідно до комплексної математичної моделі (див. розділ 2) вектор вхідних змінних $\boldsymbol{X}(k)$ містить тільки один елемент – тактову частоту:

$$\boldsymbol{X}(k) = [F_{clk}(k)]. \tag{3.3}$$

З метою зручності подальшого аналізу та ідентифікації параметрів цільовий вектор вихідних величин $\mathbf{Y}(k)$ подано у вигляді конкатенації двох

підвекторів, що відповідають показникам реактивності та обчислювальної продуктивності програмного забезпечення – $\boldsymbol{Y_{SW}}(k)$, а, також, енергетичних витрат апаратної складової БО – $\boldsymbol{Y_{HW}}(k)$:

$$\boldsymbol{Y}(k) = \begin{bmatrix} \boldsymbol{Y_{SW}}(k) \\ \boldsymbol{Y_{HW}}(k) \end{bmatrix}. \tag{3.4}$$

Вектор $\boldsymbol{Y_{SW}}(k)$ - це характеристики витрат часу на роботу системного програмного забезпечення (позначення змінних відповідає підрозділам 2.2.2, 2.3.3):

$$\boldsymbol{Y_{SW}}(k) = \left[T_{IRQ_p}(k),\ T_{Core}(k),\ T_{\text{taskYIELD}}(k),\ T_{Timer}(k) \right]^T. \tag{3.5}$$

Інші складові моделей реактивності та обчислювальної продуктивності програмного забезпечення описують характеристики прикладних завдань, що вирішуються, та визначаються через частоту та інтенсивність появи подій м’якого та жорсткого реального часу та фонове навантаження.

Для побудови моделі енергоспоживання бортового обчислювача необхідно визначити енергоспоживання на кожному режимі та протяжності перехідних режимів. Якщо загальну модель (3.2) представити як лінійну за параметрами, що відповідає комплексній математичній моделі (розділ 2), то формула (3.2) трансформується в лінійну модель з розширеним вектором параметрів:

$$\boldsymbol{Y}(k) = \boldsymbol{\Phi}(\boldsymbol{k})\boldsymbol{\Theta} + \boldsymbol{v}(k), \tag{3.6}$$

де $\boldsymbol{\Phi}(\boldsymbol{k})$ – блочно-діагональна матриця;

$\boldsymbol{\Theta}$ – вектор невідомих параметрів, що підлягають ідентифікації.

Для моделі енергоспоживання бортового обчислювача – $\boldsymbol{Y_{HW}}(k)$ формула (3.6) деталізується в наступному вигляді (позначення змінних відповідає підрозділу 2.4.2):

$$\boldsymbol{Y_{HW}}(k) = \begin{bmatrix} e_{Active}(k) \\ e_{Sleep}(k) \\ \tau_{SleepOn}(k) \\ \tau_{SleepOff}(k) \\ e_{Wait}(k) \\ \tau_{WaitOn}(k) \\ \tau_{WaitOff}(k) \\ e_{Backup}(k) \\ \tau_{BackupOn}(k) \\ \tau_{BackupOff(k)} \end{bmatrix} = \Phi(k)\Theta + \boldsymbol{v}(k), \quad (3.7)$$

де $\Phi(k)$ – блочно діагональна матриця базисних функцій:

$$\Phi(k) = \begin{bmatrix} 1 & F_{clk}(k) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & F_{clk}(k) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{1}{F_{clk}(k)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{F_{clk}(k)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & F_{clk}(k) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{F_{clk}(k)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{F_{clk}(k)} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & F_{clk}(k) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{F_{clk}(k)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{F_{clk}(k)} \end{bmatrix}, \quad (3.8)$$

Θ – вектор параметрів, що підлягають ідентифікації:

$$\Theta^T = [a_{Active}\ ,\ b_{Active}\ ,\ a_{Sleep},\ b_{Sleep},\ c_{SleepOn},\ d_{SleepOn},\ c_{SleepOff},\ d_{SleepOff},\ a_{Wait},\ b_{Wait}, c_{WaitOn},\ d_{WaitOn},\ c_{WaitOff},\ d_{WaitOff},\ a_{Backup,}\ b_{Backup},\ c_{BackupOn},\ d_{BackupOn},\ c_{BackupOff},\ d_{BackupOff}]. \quad (3.9)$$

Для показників реактивності та обчислювальної продуктивності програмного забезпечення – $\boldsymbol{Y_{SW}}(k)$ формула (3.6) деталізується у вигляді:

$$\boldsymbol{Y_{SW}}(k) = \begin{bmatrix} T_{IRQ_p}(k) \\ T_{Core}(k) \\ T_{\text{taskYIELD}}(k) \\ T_{Timer}(k) \end{bmatrix} = \begin{bmatrix} C_{IRQ_p} \\ C_{Core} \\ C_{\text{taskYIELD}} \\ C_{Timer} \end{bmatrix} \left[\frac{1}{F_{clk}(k)}\right] + \boldsymbol{v}(k). \quad (3.10)$$

3.1.3 Рекурсивний метод найменших квадратів

Для ідентифікації параметрів математичної моделі використано рекурсивний метод найменших квадратів (РМНК), який забезпечує онлайн-оцінювання параметрів із обмеженою обчислювальною складністю та дозволяє враховувати нестаціонарність характеристик бортового обчислювача шляхом введення коефіцієнта забування.

Перевагами РМНК є швидка збіжність, низькі вимоги до пам’яті, добра інтерпретованість параметрів. Основними обмеженнями РМНК в загальному випадку є потреба коректної ініціалізації та лінійність за параметрами, що не є недоліком в контексті розроблених математичних моделей та лабораторних умов проведення вимірів.

Задача ідентифікації параметрів моделі полягає у знаходженні такої оцінки вектору параметрів $\widehat{\boldsymbol{\theta}}(k)$, яка мінімізує квадратичну похибку між вимірюваними $\boldsymbol{Y}(i)$ та модельними значеннями $\widehat{\boldsymbol{Y}}(i)$ вихідних величин на множині доступних експериментальних даних:

$$\boldsymbol{J}(k) = \sum_{i=1}^{k} \lambda^{k-i} \left\| \boldsymbol{Y}(i) - \widehat{\boldsymbol{Y}}(i) \right\|^2, \quad (3.11)$$

де $\lambda \in (0, 1]$ - коефіцієнт забування.

Вхідними даними для роботи РНМК є:

$\widehat{\boldsymbol{\theta}}(k-1)$ – остання оцінка вектора параметрів моделі;

$\boldsymbol{P}(k-1)$ – остання матриця коваріації оцінок;

$\boldsymbol{Y}(k)$ – виміряне значення параметрів;

$\boldsymbol{\Phi}(\boldsymbol{k})$ – відома блочно-діагональна матриця регресорів.

Власне РНМК реалізується через наступну послідовність кроків.

Крок 1. Розрахунок апріорної векторної оцінки помилки прогнозу:

$$\boldsymbol{E}(k) = \boldsymbol{Y}(k) - \boldsymbol{\Phi}(\boldsymbol{k})\widehat{\boldsymbol{\theta}}(k-1). \quad (3.12)$$

Крок 2. Розрахунок коефіцієнта підсилення:

$$\boldsymbol{K}(k) = \frac{\boldsymbol{P}(k-1)\boldsymbol{\Phi}(\boldsymbol{k})}{\lambda + \boldsymbol{\Phi}(\boldsymbol{k})\boldsymbol{P}(k-1)\boldsymbol{\Phi}(\boldsymbol{k})}. \quad (3.13)$$

Крок 3. Оновлення матриці параметрів:

$$\boldsymbol{\theta}(k) = \widehat{\boldsymbol{\theta}}(k-1) + \boldsymbol{K}(k)\boldsymbol{E}^{\boldsymbol{T}}(k). \quad (3.14)$$

Крок 4. Оновлення коваріаційної матриці:

$$\boldsymbol{P}(k) = \frac{1}{\lambda}\big(\boldsymbol{P}(k) - \boldsymbol{K}(k)\boldsymbol{\Phi}(\boldsymbol{k})\boldsymbol{P}(k-1)\big). \quad (3.15)$$

3.1.4 Методика та техніки вимірювання показників реактивності та обчислювальної продуктивності програмного забезпечення

Оцінка показників реактивності та обчислювальної продуктивності $\boldsymbol{Y_{SW}}(k)$ виконана, як послідовність наступних кроків.

Крок 1. Попередньо (див. підрозділ 3.1.1) було визначено тестові алгоритми, час виконання яких знаходиться в інтервалі 100-200 мкс для тактових частот ядра МК $F_{clk} \in \{30\,MHz, 100\,MHz,\ 300\,MHz\}$. Для кожної тактової частоти підібрано відповідне власне тестове навантаження, а саме, алгоритм з відомим часом виконання в інтервалі 100 – 200 мкс.

Крок 2. Частоту переривань від системного таймера встановлено рівною 1000 Гц, відповідно 10 – 20 % ітерацій тестового алгоритму буде виконуватись за час, який збільшено на тривалість обробки переривання від системного таймера – $T_{Timer}(k)$.

Крок 3. Частоту переривань від зовнішнього джерела встановлено рівною 1500 Гц, відповідно 7 – 15 % ітерацій тестового алгоритму буде виконуватись за час, який збільшено на тривалість обробки переривання від зовнішнього джерела – $T_{IRQ_p}(k)$. Якщо частоти переривань (зовнішніх і від системного таймера) відрізняються у 1,5 – 2 рази, то їхній вплив на загальному графіку дуже легко розрізнити: достатньо підрахувати кількість вимірювань, що потрапили на відповідні «полиці» (рисунок 3.3). Результати чисельного моделювання фактичного часу виконання кожної ітерації розподіляються на чотири «полиці» залежно від того, чи була ітерація перервана для обробки переривань. Ці «полиці» відповідають таким ситуаціям:

– переривання відсутні;

– відбулося переривання від системного таймера;

– відбулося зовнішнє переривання;

– ітерацію було перервано двічі - системним таймером і зовнішнім перериванням.

Крок 5. Для визначення часу виконання операції захоплення м'ютекса $T_{Core}(k)$ для кожної частоти МК - F_{clk} було виконано наступний алгоритм: програмне забезпечення, яке працює під ОС FreeRTOS налаштовується так, щоб існувала лише одна задача, яка блокує та звільняє м'ютекс, після чого записує час, необхідний для виконання цих двох операцій, у масив.

Крок 6. Для визначення часу виконання операції з кооперативної диспетчеризації $T_{\mathrm{taskYIELD}}(k)$ для кожної частоти МК F_{clk} було виконано наступний алгоритм:

– створюється та запускається на одночасне виконання 10 копій однієї задачі;

– кожна задача в циклі блокує м'ютекс, записує час і номер задачі у спільний масив logEntries, звільняє м'ютекс, а потім викликає системну функцію кооперативної диспетчеризації taskYIELD();

– задачі працюють паралельно, створюючи умови для оцінки накладних витрат на перемикання між задачами. Апаратний таймер вимірює час

виконання операцій із роздільною здатністю 66,66 нс. Масив logEntries зберігає час виконання операцій і номер задачі, що дає змогу проаналізувати результати після завершення роботи програми.

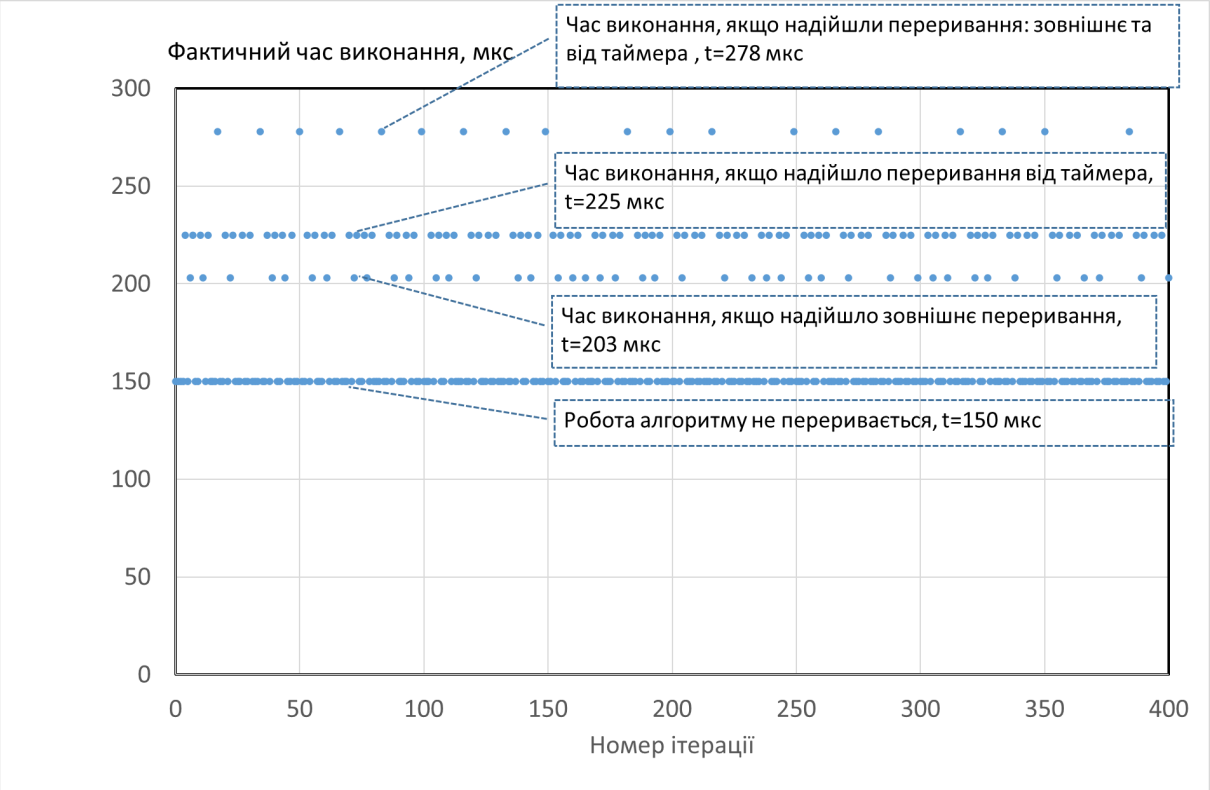

Рисунок 3.3 – Результати чисельного моделювання техніки вимірювання витрат часу на обробку переривань: зовнішнього та від системного таймера

3.1.5 Методика та техніки вимірювання енергетичних витрат апаратної складової БО

Дослідний стенд для експерименту (рисунки 3.4, 3.5) побудовано з наступного обладнання.

1. Генератор прямокутних імпульсів (GW Instek GFG-8219A) генерує зовнішні сигнали переривання із заданим періодом.

2. Високоточний блок живлення та вимірювач електричної потужності (напруги та струму) (Keithley 2281S-20-6) використовується для вимірювання споживаної електричної енергії. Він гарантує точність вимірювань із

похибкою не гірше 15 мс для часового інтервалу та не більше 0.0001 Вт для електричної потужності.

3. Багатоканальний осцилограф з пам’яттю (LeCroy WavePro 7200A) використовується для вимірювання часового інтервалу між двома подіями:

– зовнішнім сигналом, що призводить до апаратного переривання, згенерованим генератором прямокутних імпульсів;

– виконанням першої команди обробника такого переривання – T_{ISR}, якою є зміна стану попередньо визначеного порту (Порт 1) на протилежний. Оскільки команда зміни порту є атомарною (тобто виконується за один обчислювальний цикл в RISC-архітектурі), час зміни стану порту можна вважати незначним.

4. Модифікований БО «Боривітер» з дротовими з’єднаннями між осцилографом, генератором сигналів та двома портами МК, що налаштовані як порти вводу/виводу загального призначення (англ. General Purpose Input Output, GPIO) – Порт 2 та Порт 3, що зображено на рисунку 3.7.

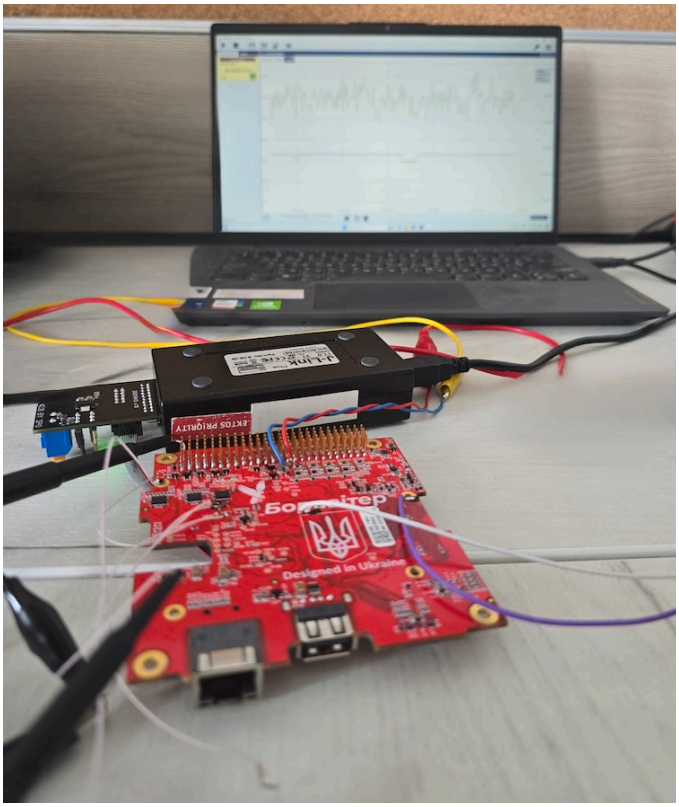

Рисунок 3.4 – Обладнання експериментального стенду

Рисунок 3.5 – Вимірювальна апаратура стенду

Для оцінки витрат часу на вхід в режим енергозбереження та вихід з нього було розроблено більш складну схему підключення (рисунки 3.6, 3.7). Ключова ідея полягає у використанні шунтуючого резистора, вже впаяного на друковану плату БО, разом із операційним підсилювачем для забезпечення належного масштабу та лінійності сигналу струму, який фіксується осцилографом.

З цим доповненням схема підключення для другого випробувального стенда включає:

- високоточний блок живлення та вимірювач потужності (Keithley 2281S-20-6): Використовується для вимірювання споживаної електричної потужності;

- багатоканальний осцилограф (Siglent SDS1204X-E): Використовується для вимірювання події-тригера (trigger event) між командою процесора «перейти в режим зниженого енергоспоживання» та реакцією мікроконтролера у вигляді зміни споживання струму;

- апаратно модифікований бортовий обчислювач «Боривітер», де осцилограф підключений до виходу операційного підсилювача монітора струму для забезпечення належної лінійності сигналу та низького рівня шуму.

Конструктивні рішення, використані для отримання вимірювань, відповідають схемам підключення, представленим нижче.

Для вимірювання енергоспоживання в стаціонарних режимах на різних частотах – e_{foms_k}, перехідних режимах – $e_{FOMS_{tr}}$, протяжності перехідних режимів – $\tau_{FOMS_{tr}}$, було розроблено тестове ПЗ що переводить МК в зазначений режим енергоспоживання за одиночним сигналом генератора (Порт 3, рисунок 3.7).

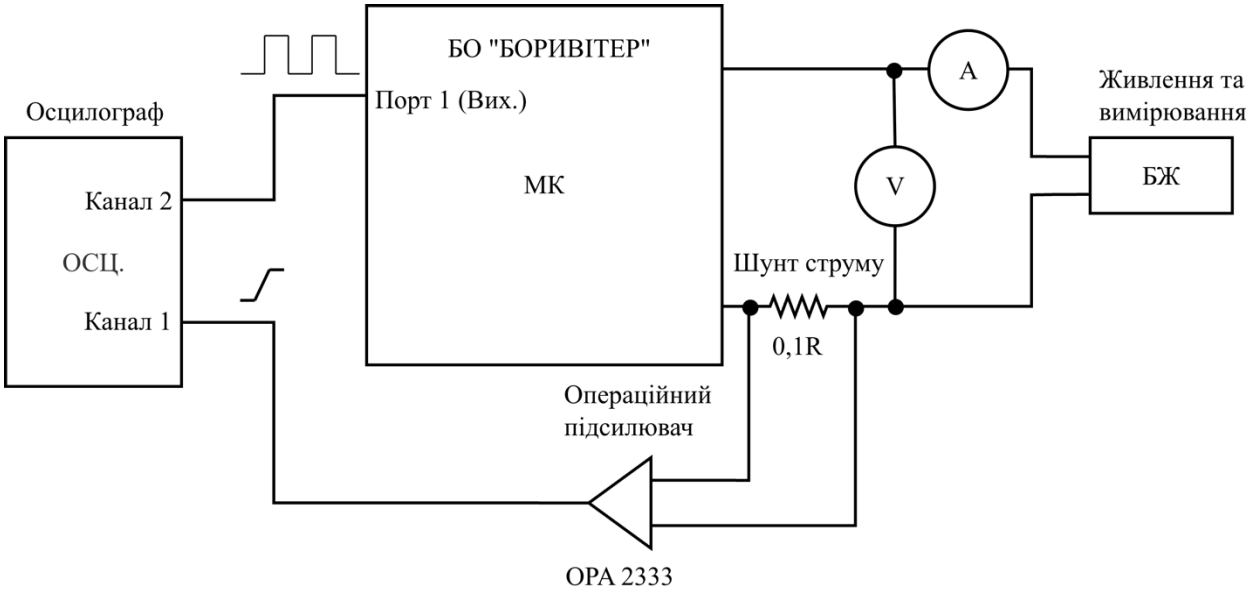

Рисунок 3.6 – Схема підключення стенду для вимірювання енергоспоживання в режимах ($e_{Active}(k)$, $e_{Sleep}(k)$, $e_{Wait}(k)$, $e_{Backup}(k)$) та витрат часу на переходи в режими енергозбереження ($\tau_{SleepOn}(k)$, $\tau_{WaitOn}(k)$, $\tau_{BackupOn}(k)$.

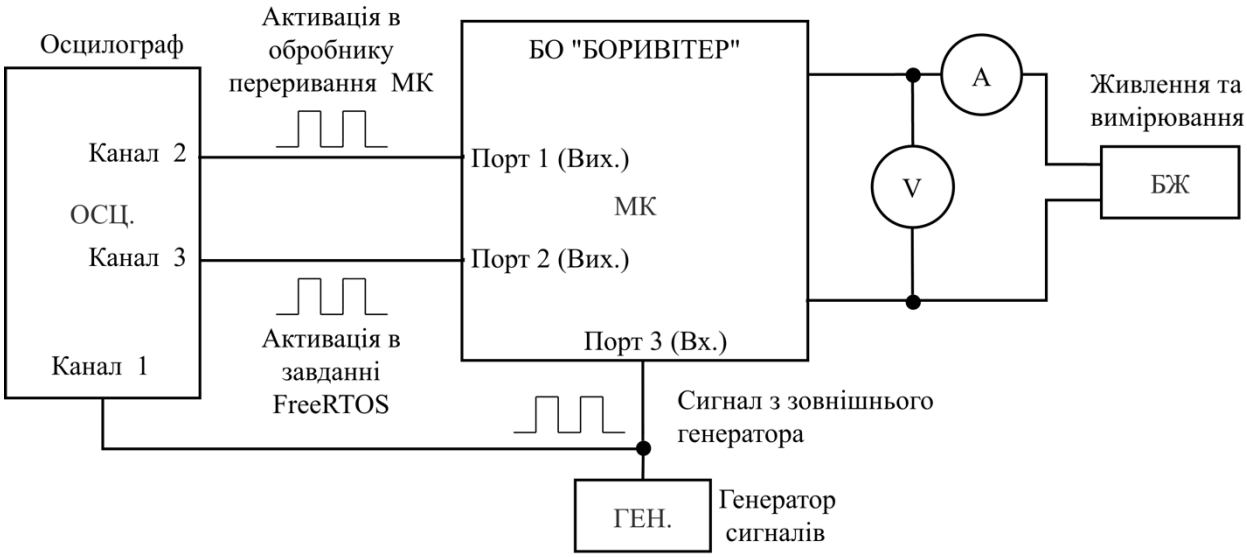

Рисунок 3.7 – Схема підключення стенду для вимірювання витрат часу на переходи в активний режим $\tau_{SleepOff}(k)$, $\tau_{WaitOff}(k)$, $\tau_{BackupOff}(k)$.

Існує суттєва різниця між використанням режимів енергозбереження.

Режим Backup є найглибшим режимом енергозбереження та може бути залишений лише через скидання процесора. У зв’язку з цим, тестове програмне забезпечення було модифіковане так, щоб перемикання визначеного GPIO-виводу виконувалося першою операцією після старту програми. Час виходу з режимів низького енергоспоживання (LPM) істотно залежить від конкретного режиму та способу вимкнення периферійних модулів мікроконтролера. Якщо режими Sleep і Wait дозволяють повернення до попереднього обчислювального контексту й, тому, є відносно простими для вимірювання, то режим Backup потребує запуску через вектор скидання, що ускладнює програмну реалізацію та, зазвичай, вимагає використання зовнішньої енергонезалежної пам’яті для збереження контексту.

Для вимірювання часу виходу з режимів LPM було використано два тестові сценарії із зовнішнім перериванням від генератора сигналів, яке синхронізує осцилограф для фіксації моменту перемикання GPIO-виводу після готовності мікроконтролера до виконання інструкцій.

Сценарій 1 (Sleep, Wait). Джерело пробудження налаштовується на зовнішнє переривання з GPIO, підключеного до генератора сигналів із

частотою 10 Гц. В обробнику переривань першою операцією виконується перемикання GPIO. Час пробудження визначається як різниця між фронтом наростання вхідного сигналу та фронтом на вихідному GPIO.

Сценарій 2 (Backup). Аналогічно використовується зовнішнє переривання з частотою 10 Гц, однак через проходження мікроконтролера через вектор скидання перемикання GPIO здійснюється першою операцією у функції main() після конфігурації портів. Оскільки програмне забезпечення не містить значних глобальних змінних, час ініціалізації секції .bss не враховується. Час пробудження визначається аналогічно до першого сценарію.

3.1.6 Результати експерименту

Результати вимірювання показників реактивності та обчислювальної продуктивності, що відображені на рисунку 3.8, якісно відповідають попереднім очікуванням, що відображені на рисунку 3.3. Завдяки особливостям обчислювальної архітектури ARM Cortex-M, яка використовує два незалежні лічильники інструкцій - основний лічильник команд (Program Counter, PC) та емуляційний регістр повернення (Link Register, LR), система характеризується високими показниками швидкодії та реактивності. Операційна система FreeRTOS забезпечує ефективну підтримку архітектури ARM Cortex-M, не розмежовуючи механізми обробки переривань і виконання звичайних функцій.

Експериментальні дослідження підтвердили, що за умов кооперативної диспетчеризації час виконання системної функції taskYield() не залежить від кількості задач у системі. Це пояснюється використанням у FreeRTOS механізму циклічної (round-robin) диспетчеризації задач, розміщених у черзі виконання.

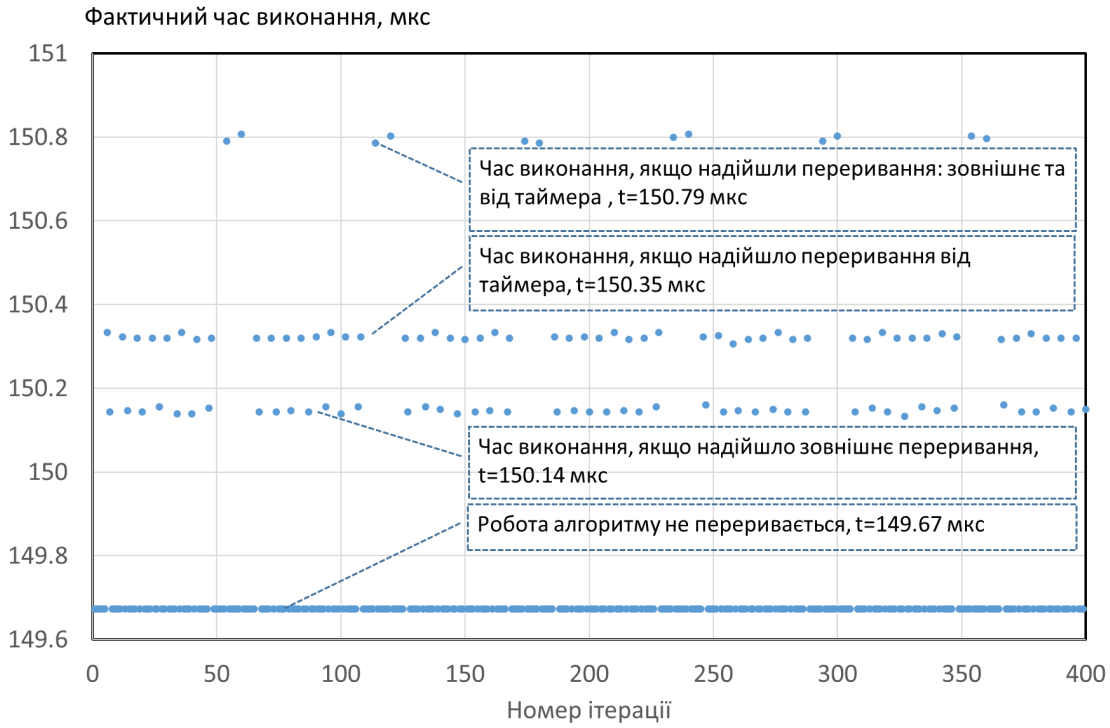

Рисунок 3.8 – Результати вимірювання показників реактивності та обчислювальної продуктивності на стенді

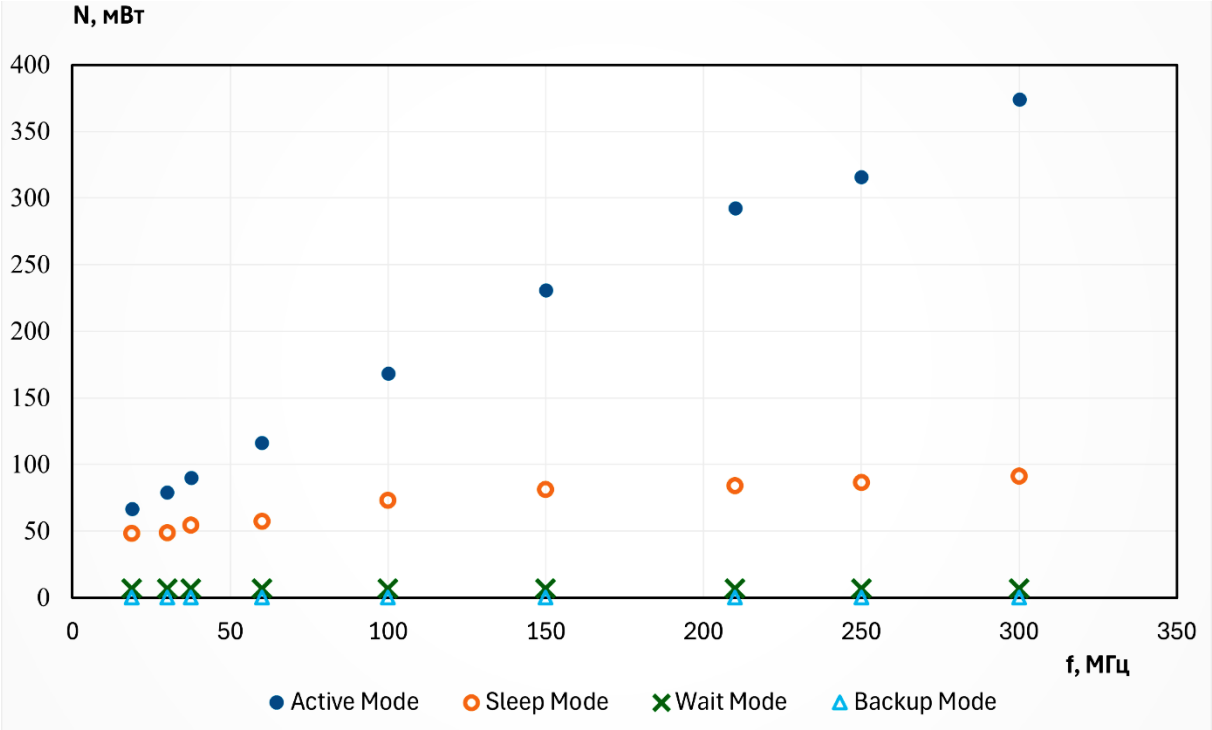

Рисунок 3.9 – Залежність споживання електричної енергії від тактової частоти МК у різних режимах роботи МК при постійному значенні напруги живлення 3.3 В.

На рисунку 3.9 експериментально підтверджено, що енергоспоживання МК в активному режимі за фіксованої напруги живлення залежить лінійно від частоти. В режимі Sleep помітно виражена нелінійність, а в режимах Wait та Backup можно вважати, що енергоспоживання від тактової частоти не залежить.

З використанням рекурсивного методу найменших квадратів (РМНК) отримано наступне чисельне значення вектору параметрів моделі енергоспоживання з формули 3.9:

$$\Theta^T = [51.78\,,\ 1.098\,,\ 49.25\,,\ 0.1584,\ 20.52,\ 602.46,\ 0.173,\ 145.3,\ 6.96,\ 0,\ 155.07,\ 0,\ 295.4,\ 48146,\ 0.46,\ 0,\ 1500,\ 0,\ 5000,\ 0]. \quad (3.16)$$

Розмірності змінних в формулі (3.16) такі:

$$R^T(\Theta) = [\text{мВт}\,,\ \frac{\text{мВт}}{\text{МГц}},\ \text{мВт}\,,\ \frac{\text{мВт}}{\text{МГц}},\ \text{мс},\ \text{мс}\cdot\text{МГц},\ \text{мс},\ \text{мс}\cdot\text{МГц},\ \text{мВт}\,,\ \frac{\text{мВт}}{\text{МГц}},\text{мс},\ \text{мс}\cdot\text{МГц},\ \text{мс},\ \text{мс}\cdot\text{МГц},\ \text{мВт}\,,\ \frac{\text{мВт}}{\text{МГц}},\ \text{мс},\ \text{мс}\cdot\text{МГц},\ \text{мс},\ \text{мс}\cdot\text{МГц}]. \quad (3.17)$$

Експериментально визначені параметри моделей реактивності та обчислювальної продуктивності програмного забезпечення формули 3.10 дорівнюють (розмірність коефіцієнтів – мкс*МГц = 1):

$$\begin{bmatrix} C_{IRQ_p} \\ C_{Core} \\ C_{\text{taskYIELD}} \\ C_{Timer} \end{bmatrix} = \begin{bmatrix} 192.8 \\ 232.1 \\ 217.9 \\ 121.0 \end{bmatrix}. \quad (3.18)$$

3.1.7 Перевірка адекватності математичних моделей

Перевірку адекватності виконано в припущенні, що дотримано наступні обмеження.

1. Функційна форма моделі є коректною та лінійною відносно своїх експериментально визначених параметрів (оцінюваних коефіцієнтів), що забезпечує існування та єдиність оцінок методом найменших квадратів.

2. Результати вимірювань спостереження є репрезентативними: незалежно відібрані, випадкові, без систематичного упередження.

3. Похибки моделі ε_i є випадковими величинами, що утворюють незалежну та ідентично розподілену вибірку з нормального розподілу з нульовим математичним сподіванням та сталою дисперсією σ^2:

$$\varepsilon_i \sim N(0,\ \sigma^2). \tag{3.19}$$

Подальша перевірка адекватності заснована на розрахунку трьох параметрів, що характеризують розкид залежної змінної та відхилення модельних розрахунків від емпіричних спостережень.

1. Загальна варіація залежної змінної (SST - Sum of Squares Total):

$$SST = \sum_{i=1}^{n}(y_i - \bar{y})^2, \tag{3.20}$$

де y_i - фактичні значення відгуку;

$\bar{y}$ - середнє фактичних значень.

2. Регресійна сума квадратів (SSR - Sum of Squares Regression) характеризує, яку частину варіації експериментальних даних пояснює модель:

$$SSR = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2, \tag{3.21}$$

де $\hat{y}_i$ - розрахункові модельні значення.

3. Залишкова сума квадратів (SSE – Sum of Squares Error) показує невраховану моделлю варіацію:

$$SSR = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3.22}$$

Адекватність розроблених моделей перевірено через послідовність наступних двох перевірок.

1. Визначення коефіцієнта детермінації R^2, який показує, яку частку варіації залежної змінної пояснює побудована модель. Чим ближче R^2 до 1, тим краще модель описує дані:

$$R^2 = \frac{SSR}{SST}. \tag{3.23}$$

2. Перевірка статистичної гіпотези, що модель значуща з визначеним рівнем значущості α. Емпіричне значення критерію Фішера розраховується так:

$$F_{emp} = \frac{SSR/k}{SSE/(n-k-1)} = \frac{R^2}{1-R^2} \cdot \frac{n-k-1}{k}, \tag{3.24}$$

де k - кількість незалежних змінних;

n - кількість спостережень.

Якщо розраховане емпіричне значення критерію Фішера більше, ніж табличне критичне значення $F_{emp} > F_{krit}$, то модель адекватна. Для розрахунку табличного критичного значення критерію Фішера використано функцію MS Excel: F.INV.RT(α; df1; df2):

$$F_{krit} = \text{F. INV. RT}(\alpha,\ n-k-1, \text{k}), \tag{3.25}$$

де α=0.05 - рівень значущості;

n - кількість спостережень у вибірці;

k - кількість пояснювальних змінних (факторів).

В таблиці 3.2 наведено результати перевірки адекватності тільки тих моделей, де існує статистично підтверджена залежність показників реактивності та обчислювальної продуктивності програмного забезпечення, а,

також, енергетичних витрат апаратної складової. Табличне критичне значення F_{krit} в усіх випадках дорівнює 161.45.

Таблиця 3.2 Результати перевірки адекватності моделей

Модель	Коефіцієнт детермінації R^2	Емпіричне значення F_{emp}
T_{IRQ_p}	0.99999	99999
T_{Core}	0.99999	99999
$T_{taskYIELD}$	0.99999	99999
T_{Timer}	0.99972	3570.429
e_{Active}	0.9943	174.4386

Загальний висновок за результатами перевірки адекватності такий:

- результати параметричної ідентифікації коефіцієнтів моделей реактивності та обчислювальної продуктивності програмного забезпечення повністю відповідають поставленій меті, є повторюваними та можуть бути використані не тільки при проєктуванні, але й при адаптивному регулюванні режимів роботи бортових обчислювачів наносупутників CubeSat;

- параметри моделі енергоспоживання дають загальне розуміння процесів, що відбуваються, але точність моделі достатня тільки для передпроєктного оцінювання. Виключенням з цього загального положення є залежність енергоспоживання в активному режимі від частоти – e_{Active}.

3.2 Метод адаптивного регулювання режимів роботи БО

3.2.1 Адаптивна політика енергозбереження в мікроконтролерних системах послабленого жорсткого часу

У межах цього дослідження розглядається клас мікроконтролерних систем, що є складовою переважної частини архітектур сучасних БО, які функціонують у режимі послабленого жорсткого реального часу (Weakly Hard

Real-Time) [158, 159]. На відміну від класичних систем жорсткого реального часу, такі системи допускають контрольовані локальні пропуски або затримки в обробці подій, за умови, що вони не виходять за межі визначених характеристик якості обслуговування (англ. Quality of Service, QoS).

Концепцію систем послабленого жорсткого реального часу (Weakly Hard Real-Time Systems) було започатковано ще в 2001 році [160]: одноразове, або навіть багаторазове порушення часових обмежень однією задачею не завжди відмова, оскільки існує певна надлишковість системи. При такому формулюванні можна аналізувати обчислювальну систему, в якій одночасно необхідно вирішувати завдання реального часу двох видів: жорсткого та м'якого. Останнє повністю відповідає моделі реактивності, що представлена в підрозділі 2.2.

В концепції систем послабленого жорсткого реального часу кожна задача описується як кортеж:

$$\tau_i = \langle C_i,\ D_i,\ T_i,\ m_i,\ K_i \rangle, \tag{3.26}$$

де C_i - максимальний час, потрібний задачі τ_i;

T_i - мінімальний інтервал прибуття задачі τ_i. Якщо це періодична задача, то T_i - це її період;

D_i - часові обмеження (дедлайн) на виконання роботи задачею ($D_i \leq T_i$);

$m_i,\ K_i$ - послаблені обмеження задачі реального часу ($m_i < K_i$): на послідовності з K_i робіт цієї задачі дозволено порушити дідлайни не більше m_i робіт. Якщо це задача жорсткого реального часу (hard real-time task), то $m_i = 0,\ K_i = 1$.

Зазвичай в системах послабленого жорсткого часу використовуються планувальники – RMS, EDF та LLF (див. підрозділ 1.4.4). Додатково існують алгоритми з резервуванням ресурсів – сервер постійної пропускної здатності (англ. Constant Bandwidth Server, CBS) та сервер загальної пропускної

здатності (англ. Total Bandwidth Server, TBS), а, також гібридні підходи, як то – багаторівневі черги (англ. Multi-Level Feedback Queues, MLFQ)

Одне з можливих та відомих на сьогодні рішень алгоритму роботи планувальника в системі послабленого жорсткого реального часу є таким, що зображене на рисунку 3.10, та складається з наступних кроків:

1. Якщо всі заплановані дедлайни всіх робіт для всіх завдань можуть бути дотримані за допомогою EDF, використати EDF. Якщо ні – то на п.2;

2. Відсортувати роботи всіх задач за критерієм кількості дедлайнів робіт, що ще дозволено цим задачам порушити на інтервалі часу планування;

а) до класу «0» - попадуть всі задачі, які не дозволяють жодного пропуску,

б) до класу «1» - ті, що можуть один раз порушити дедлайн. Як приклад: якщо для задачі $m_i = 2,\ K_i = 4$, тобто дозволено порушити дедлайни 2-х робіт з 4-х послідовних робіт, то задача може попасти в класи $\{0, 1, 2\}$ в залежності від кількості вже пропущених робіт,

в) В клас «2» - всі інші задачі, як то фонові або без обмежень часу,

3. Використати планування за принципом EDF спочатку для класу 0, потім для класу 1 тощо для впорядкування послідовності вирішуваних задач.

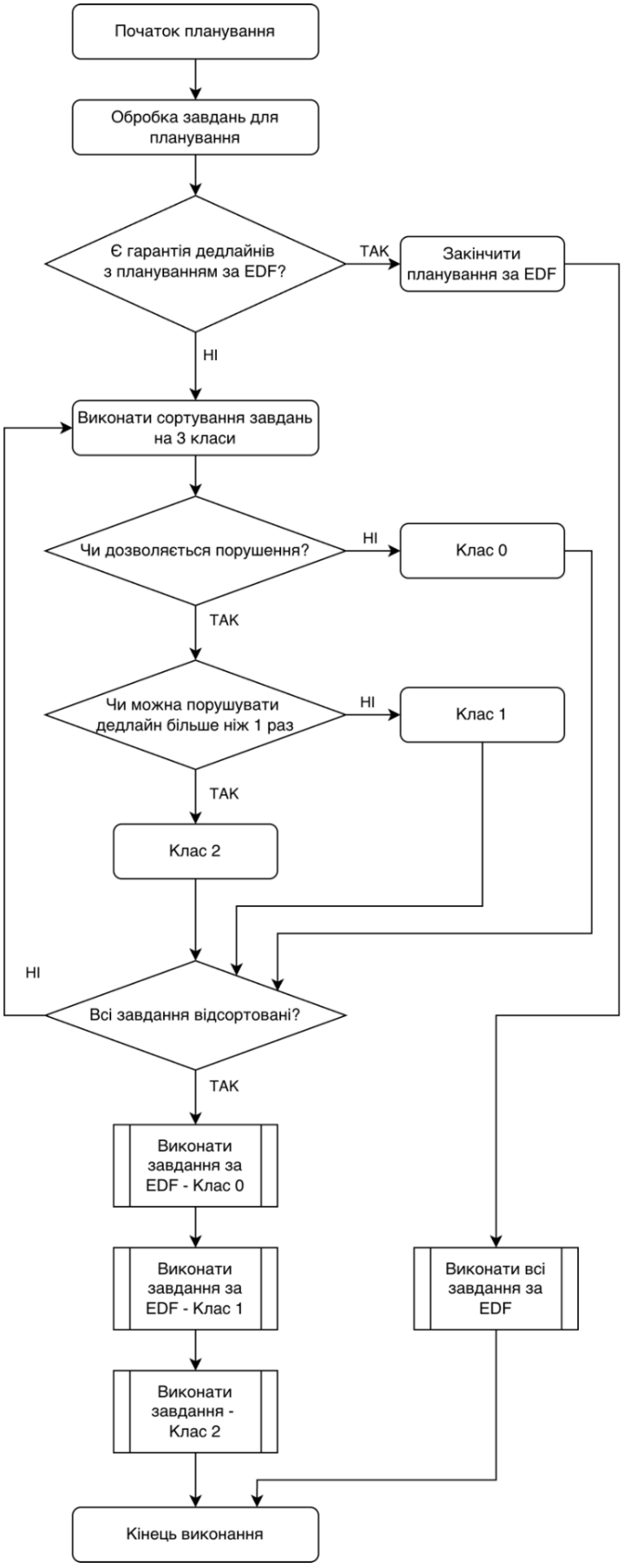

Рисунок 3.10 – Блок-схема алгоритму роботи планувальника в системі послабленого жорсткого реального часу

В дисертації вирішується завдання з розробки методу адаптивного регулювання режимів роботи бортових обчислювачів наносупутників, з метою зменшення енергоспоживання БО за умов дотримання обмежень реального часу та забезпечення гарантованої реактивності бортових систем. В такому формулюванні необхідно вирішити науково-прикладне завдання, як саме пристосувати відому концепцію систем послабленого жорсткого реального часу до побудови адаптивної політики енергозбереження.

Наступні припущення та обмеження, яких не існує в загальній теорії систем послабленого жорсткого реального часу, є важливими в контексті дисертації.

У системі виконується двоетапна модель обробки зовнішніх подій. На першому етапі виконується первинна обробка переривання, що має найвищий пріоритет і не може бути перервана іншими зовнішніми подіями. Первинний обробник виконує мінімальний набір дій: фіксує факт надходження події та активує відповідний об’єкт ядра операційної системи. На другому етапі працюють вторинні обробники переривань, кожен із яких відповідає за повний цикл обробки конкретного типу події. Вторинні обробники можуть накопичуватися в черзі, яка в цьому дослідженні вважається необмеженою. Фонові завдання виконуються лише за відсутності подій у чергах первинних і вторинних обробників.

Система розглядається як односерверна: події надходять послідовно, пробудження з режиму енергозбереження відбувається з надходженням наступного зовнішнього переривання. Після зовнішнього переривання витрачається відомий детермінований час на перехід в активний режим роботи мікроконтролера, потім виконується первинний обробник переривань (ISR), далі - вторинний. Під час роботи первинного ISR нові події лише накопичуються в черзі. Пріоритети впорядковані: первинний обробник, вторинні обробники, фонові процеси.

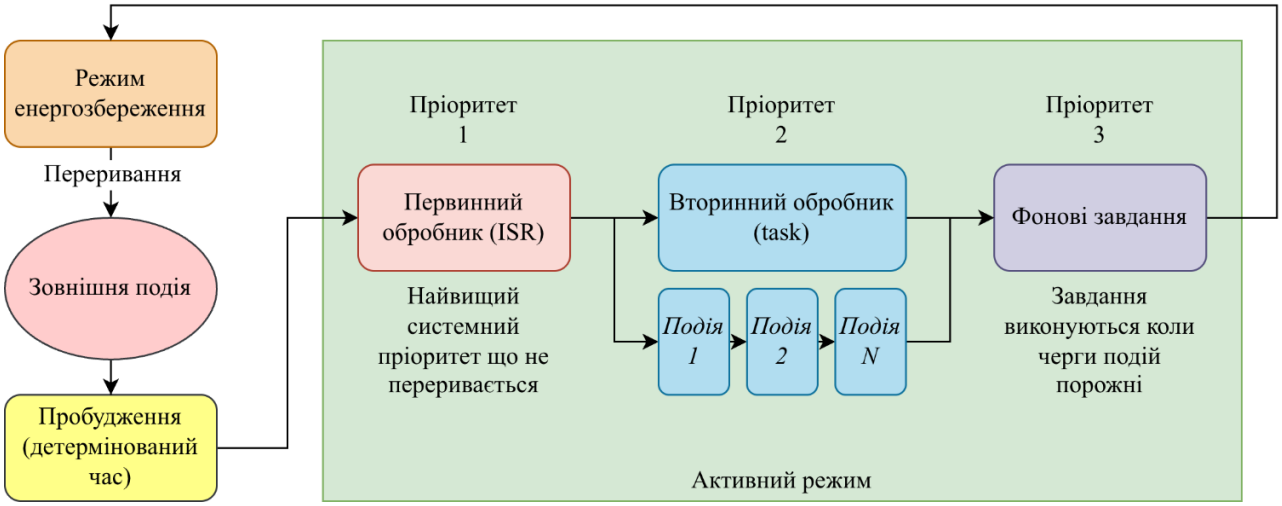

Рисунок 3.11 – Архітектура односерверної системи

Події кожного типу характеризуються так, як прийнято в концепції систем послабленого жорсткого реального часу (3.3).

Усі параметри енергоспоживання та часові витрати, пов’язані з переходами між режимами Active, Sleep, Wait і Backup, вважаються детермінованими та відомими (див. п.п. 2.2, 2.4). Пробудження можливе лише внаслідок таймерного переривання або надходження зовнішньої події. Період системного таймера, що задає системний квант планування - фіксований.

Тоді, перший етап вирішення задачі полягає у розробці адаптивної політики енергозбереження, яка, виходячи з поведінки системи в реальному часі:

- оцінює поточні характеристики подій;
- оптимізує обрання режимів роботи для досягнення мінімального енергоспоживання та одночасно гарантує дотримання вимог до системи послабленого жорсткого реального часу.

Тобто, політика визначає доцільність переходу мікроконтролера в енергозберігаючі режими Sleep, Wait або Backup, не порушуючи вимог щодо реактивності та часової коректності обробки подій. З огляду на те, що переходи між режимами потребують додаткового часу, а пробудження знижує реактивність системи, політика повинна знаходити баланс між економією енергії та вимогами реактивності і обмеженнями систем реального часу.

3.2.2 Апріорні оцінки інтенсивності та кластеризації подій

Першою дією у оцінці інтенсивності подій, потрібно виконати апріорну оцінку, тобто відому для поточного типу давача таких подій.

Початкове рішення щодо можливості переходу в режим енергозбереження ґрунтується на граничних, найконсервативніших припущеннях щодо частоти надходження подій. Це дозволяє сформувати базову безпечну політику, що гарантує дотримання часових обмежень для найгірших можливих умов. Надходження подій можна описати як квазі-періодичний процес із джитером (шумом) або ситуацію, де події мають певну статистичну регулярність (наприклад, пакети даних сенсорів, пінг від вузла мережі, натискання кнопки користувачем).

Припустимо, що система має відому множину джерел подій $E = \{e_1, e_2, \dots, e_n\}$ (англ. Events, E). Для кожного з джерел, знаючи природу події, можна визначити інтенсивність подій та інтервал між ними. На практиці в мікроконтролерних системах ці джерела подій – це переривання від внутрішніх або зовнішніх периферійних модулів, що потрапляють та агрегуються первинним обробником переривання.

Для кожного з джерел подій e_i визначимо λ_i^{max} – максимальну кількість подій за одиницю часу. Зазвичай, такі обмеження можна визначити під час аналізу фізичної природи джерела подій, що відбувається в процесі проєктування системи керування та алгоритмів роботи БО наносупутника.

Мінімальний інтервал між подіями – τ_{min}, це час за який подія не може надійти повторно.

$$\tau_i^{min} = \frac{1}{\lambda_i^{max}}\ . \tag{3.27}$$

Відповідно мінімальний інтервал для всієї системи можна формалізувати як:

$$\tau_{min}^{sys} = \min_{i \in E}\left(\tau_i^{min}\right). \tag{3.28}$$

Апріорні оцінки мають бути визначені шляхом аналізу конфігурації МК, завдань, що виконуються МК БО, та зовнішніми подіями, що залежать від виконуючих механізмів або сенсорів. Цей аналіз проводиться на етапі проєктування БО та розробки ПЗ.

Для кожної події в системі потрібен «спостерігач» що зможе реєструвати такі події та обчислювати їх інтенсивність. Для з'ясування необхідного часу реакції на подію, треба виконати кластеризацію подій.

Оскільки можлива кількість спостерігачів для сучасних бортових обчислювачів достатньо велика, то такий підхід достатньо ресурсоємний. Пропонується кластеризувати події на критичні або швидкі та користувацькі. Критичні – мережеві пакети, послідовні інтерфейси, що працюють в режимі «ведений», а користувацькі – спрацювання повільного сенсору температури, або напруги батареї.

Таким чином, формулюємо наступну класифікацію кластерів подій за критичністю:

$$C_{crit} \cup C_{norm} = E, \tag{3.29}$$

$$C_{crit} \cap C_{norm} = \emptyset, \tag{3.30}$$

де C_{crit} – кластер джерел подій, що вимагають швидкої реакції (події жорсткого реального часу – E^H (2.2), C_{norm} – кластер джерел подій, що допускають затримку реакції системи (події м'якого реального часу – E^S(2.2)). Тоді, за апріорною оцінкою для кожного з джерел подій e_i, можна сформулювати опис події за кортежом апріорних параметрів P_i:

$$P_i = \langle \lambda_i^{min}, \lambda_i^{max}, \tau_i^{min} \rangle, \tag{3.31}$$

де $\lambda_i^{min}, \lambda_i^{max}$ – апріорні межі інтенсивності, τ_i^{min} – мінімальний фізичний інтервал між подіями.

Планується, що метод керування енергоспоживанням буде використовувати всі доступні режими енергоспоживання – $m \in \{ACTIVE, SLEEP, WAIT, BACKUP\}$. Таким чином можемо сформувати вектор характеристик ефективності:

$$H(m_j) = \langle P^j_{cons}, \theta^j_{wakeup}, \theta^j_{breakeven} \rangle, \quad (3.32)$$

де, P^j_{cons} – потужність споживання в даному режимі МК за умови $P^{sleep} > P^{wait} > P^{backup}$, θ^j_{wakeup}- час виходу з режиму (латентність), $\theta^j_{breakeven}$ - точка енергетичної беззбитковості (мінімальний час перебування в режимі, необхідний для компенсації енергетичних витрат на вхід/вихід).

Представлення множини режимів як упорядкованого набору станів дозволяє перейти від простої логіки "активний/сон" до задачі дискретної оптимізації. Введення параметру $\theta^j_{breakeven}$ є критично важливим для багаторівневих систем, оскільки глибокі режими (BACKUP), попри низьке статичне споживання, мають високу "вартість" перемикання. Модель враховує цей фактор, щоб запобігти ситуаціям, коли часте перемикання в глибокий сон призводить до більших витрат енергії, ніж перебування в легкому сні.

Таким чином виконана формалізація статичної конфігурації об'єкта керування, визначаючи скінченну множину стохастичних джерел подій, що впливають на обчислювальну систему. Кластеризація джерел на критичні та нормальні встановлює фундаментальні обмеження, щодо допустимої латентності реакції, що є необхідною умовою для диференціації стратегій енергозбереження. Введення вектору апріорних параметрів дозволяє виконати ініціалізацію системи з безпечними граничними умовами, забезпечуючи її стійкість на етапі "холодного старту" до моменту накопичення достатньої кількості статистичних даних, щодо реальної інтенсивності подій. А введення параметру $\theta^j_{breakeven}$ – дозволяє спроєктувати багаторівневу систему керування.

3.2.3 Реєстрація та дискретизація подій

Вхідний потік подій від джерела e_i у фізичному часі є неперервним. Для обробки цього потоку в цифровій системі необхідно виконати перехід від неперервного представлення до дискретного часового ряду.

Розглянемо моделювання вхідної події як послідовність дельта-функцій Дірака:

$$x_i(t) = \sum_{j=1}^{\infty} \delta(t_i - t_{i,j}), \tag{3.33}$$

де $t_{i,j}$ - момент настання j-ї події джерела e_i.

Вводиться дискретизація часу з фіксованим кроком квантування Δt (тривалість слота спостереження). Кількість подій $n_i[k]$ на k-му інтервалі дискретизації визначається інтегруванням функції потоку:

$$n_i[k] = \int_{(k-1)\Delta t}^{k\Delta t} x_i(t)dt, \tag{3.34}$$

де $k \in \mathbb{Z}^+$.

Таким чином ми представляємо асинхронні апаратні переривання через послідовність дельта-функцій, що дозволяє змоделювати вхідний потік подій у неперервному часі, зберігаючи інформацію про миттєві моменти їх настання. Перехід до дискретного часу через інтегрування на фіксованих інтервалах квантування Δt – дозволяє адаптувати модель до цифрової системи керування з обмеженими ресурсами. Такий підхід дозволяє ефективно поєднати високочастотну природу надходження переривань з низькочастотним циклом прийняття рішень менеджером енергоспоживання, зменшуючи обчислювальне навантаження на МК.

Кількість подій $n_i[k]$ може мати значну дисперсію через, як приклад, виникнення стохастичного шуму (джитера) або в випадку пакетної передачі даних. Для підвищення стійкості моделі використано ковзні вікна спостереження. Якщо довжина вікна спостереження – W, то акумульована активність подій $N_i[k]$ від джерела e_i на горизонті спостереження буде визначатись як згортка:

$$N_i[k] = \sum_{m=0}^{W-1} n_i[k-m] \cdot h[m], \tag{3.35}$$

де $h[m]$ – вагова функція вікна.

Інтегрування кількості подій на історичному горизонті W дозволяє виділити стійку тенденцію зміни інтенсивності навантаження, формуючи згладжену метрику для прогнозування. Це забезпечує гістерезис системи керування, запобігаючи явищу частого перемикання режимів при нерівномірному розподілі подій у часі.

3.2.4 Оцінювання інтенсивності та прогнозування періоду

На основі відфільтрованих даних на кожному кроці k виконується перерахунок параметрів потоку. Задача полягає у визначенні поточної апостеріорної інтенсивності та прогнозуванні часу до наступної події.

Середню інтенсивність подій $\hat{\lambda}_i[k]$ (подій/с) на поточному кроці визначаємо наступним шляхом:

$$\hat{\lambda}_i[k] = \frac{N_i[k]}{W \cdot \Delta t}. \tag{3.36}$$

Тоді оцінку математичного сподівання часу до наступної події $\hat{\tau}_i[k]$ можна розрахувати як величину обернену до інтенсивності подій (3.36) та викласти наступним чином:

$$\hat{\tau}_i[k] = \begin{cases} \tau_i^{max},\ \hat{\lambda}_i[k] \geq (\tau_i^{max})^{-1} \\ \frac{1}{\hat{\lambda}_i[k]},\ 0 < \hat{\lambda}_i[k] < (\tau_i^{max})^{-1}, \\ \infty,\ \hat{\lambda}_i[k] = 0 \end{cases} \quad (3.37)$$

де, перший випадок – це режим насичення подіями – потік подій максимального рівня для системи;

другий – активний режим, де потік подій в діапазоні доступним для обробки;

третій – режим простою, подій немає.

Таким чином, був формалізований механізм перетворення накопичених статистичних даних у динамічну оцінку параметрів потоку, що дозволяє системі адаптуватися до змінних умов функціонування в реальному часі. Визначення математичного сподівання часу до наступної події $\hat{\lambda}_I$ базується на оберненій залежності від інтенсивності, проте, модифікується введенням логіки насичення для врахування фізичних обмежень МК БО. Такий підхід поєднує імовірнісне прогнозування з детермінованими апаратними обмеженнями, гарантуючи, що прогноз ніколи не перевищить фізично можливу пропускну здатність каналу обробки подій.

Для подальшого розрахунку глобального безпечного часу сну треба виконати кон'юнкцію векторів станів окремих джерел подій у глобальну метрику системи. Для цього введемо індикатор критичності активності системи $\Phi_{crit}[k]$, як булеву функцію, що визначає навантаження системи з джерел подій жорсткого реального часу - кластера C_{crit}:

$$\Phi_{crit}[k] = \bigvee_{e_i \in C_{crit}} \left(\hat{\lambda}_i[k] > \lambda_{threshold}\right). \quad (3.38)$$

За допомогою цієї функції виконуємо розрахунок глобального безпечного горизонту сну – $\tau_{sys}[k]$ як математичний мінімум серед прогнозованих періодів усіх системних джерел подій:

$$\tau_{sys}[k] = \min_{e_i \in E}(\hat{\tau}_i[k]). \quad (3.39)$$

Описаний процес агрегації виконує синтез локальних метрик окремих джерел подій у єдиний вектор глобального стану системи, необхідний для прийняття централізованого рішення про перехід у режим зниженого енергоспоживання. Формування логічного індикатора критичної активності $\Phi_{crit}[k]$ дозволяє реалізувати пріоритетне обслуговування подій жорсткого реального часу - високошвидкісних інтерфейсів, забороняючи переходи у стани з високою латентністю виходу. Визначення глобального безпечного горизонту $\tau_{sys}[k]$ через оператор мінімізації реалізує принцип "найгіршого випадку" (англ. worst-case scenario), гарантуючи цілісність часових обмежень для всієї сукупності активних задач.

3.2.5 Функція прийняття рішення, щодо переходу в енергозберігаючий режим

Для здійснення переходів між режимами $M[k]$, необхідно реалізувати каскадну (поступову) функцію перевірки можливості переходу для кожного з режимів, починаючи з найлегшого для переходу та найшвидшого для виходу (забезпечення реактивності).

Для кожного з режимів енергозбереження $m_j \in \{SLEEP, WAIT, BACKUP\}$, визначимо булеву функцію доступності переходу – $Fit(m_j, k)$:

$$Fit(m_j, k) =$$

$$\begin{cases} 1, \text{ якщо} - \left(\neg\Phi_{crit}[k] \vee m_j = m_{sleep}\right) \wedge \left(\tau_{sys}[k] > \theta^{j}_{breakeven} + \Delta_{safe}\right) \\ 0, \text{ в іншому випадку} \end{cases} \quad (3.40)$$

де перша частина функції – це оцінка безпеки переходу в енергозберігаючий режим, а друга частина – це оцінка ефективності, чи доцільний перехід з точки зору енергії що заощаджується.

Для реалізації каскадного закону керування специфікуємо цифровий автомат з наступними станами:

$$M[k] = \begin{cases} m_{backup,} \text{ якщо } Fit(m_{backup}, k) = 1 \\ m_{wait}, \text{ якщо } Fit(m_{wait}, k) = 1 \\ m_{sleep}, \text{ в іншому випадку} \end{cases}. \quad (3.41)$$

Загальний розрахунок можливої тривалості сну за показниками інтенсивності вхідних подій можна розрахувати наступним чином:

$$T_{sleep}[k] = \max\left(0, \min\left(T_{OS}, \alpha \cdot \tau_{sys}[k], \tau_{sys}^{max} - \theta_{wakeup}(M[k])\right)\right), \quad (3.42)$$

де T_{OS} – час сну що розрахований планувальником ОС;

α – коефіцієнт довіри (скалярна величина $0 < \alpha < 1$, наприклад 0,8);

$\tau_{sys}[k]$ – статистичний горизонт до наступної події (ковзне вікно);

τ_{sys}^{max} – апаратна межа найшвидшого критичного джерела подій;

$M[k]$ – поточний обраний автоматом режим енергозбереження;

$\theta_{wakeup}(M[k])$ – час затримки пробудження для обраного режиму.

Фінальна логіка вибору режиму реалізує адаптивну стратегію, що балансує між трьома рівнями енергозбереження:

- BACKUP активується лише за умов повної відсутності активності критичних інтерфейсів та наявності тривалого прогнозованого вікна простою ($\tau_{sys} \gg \theta_{breakeven}^{backup}$);

- WAIT виступає як проміжний буферний режим для середніх інтервалів очікування;

- SLEEP гарантує мінімальну економію навіть при високому навантаженні.

Така каскадна модель дозволяє максимізувати час перебування в найглибшому можливому сні без порушення часових обмежень та енергетичної ефективності.

3.2.6 Реалізації методу адаптивного регулювання режимів роботи БО на основі цифрового автомату з пам'яттю

Для забезпечення надійності системи та запобігання часовим гонкам (race conditions) при переході між режимами, вводиться цифровий автомат з пам'яттю - супервізор. Він діє як фільтр низьких частот для керуючих рішень, дозволяючи перехід у енергозберігаючі стани лише за умови накопичення статистичної впевненості. Для подальшої роботи, припустимо, що статистична впевненість це 100 вимірів (100 системних обчислювальних циклів).

Автомат можна описати наступним кортежем:

$$\mathcal{A} =< Q,\ \Sigma,\ \delta,\ q_0,\ C >. \tag{3.43}$$

Множина станів Q відображає поточний рівень дозволу (Permission Level) на енергозбереження:

$$Q = \{L_{block}, L_{sleep}, L_{wait}, L_{backup}\}, \tag{3.44}$$

де L_{block} – початковий стан (блокування, тільки ACTIVE);

- L_{sleep} – дозволено перехід ACTIVE → SLEEP;

- L_{wait} – дозволено перехід ACTIVE → WAIT (а також SLEEP);

- L_{backup} – дозволено перехід ACTIVE → BACKUP (а також WAIT, SLEEP).

Початковий стан автомату: $q_0 = L_{block}$. Автомат використовує лічильник стабільності - акумулятор:

$$C[k] \in \{0, \dots, N_{stable}\}, \tag{3.45}$$

де $N_{stable} = 100$.

Вхідними даними для автомата Σ на такті k є результат розрахунку попередньої моделі $M[k]$ та $T_{sleep}[k]$, а саме перевірка придатності режимів:

$$V[k] = \langle v_{sleep}, v_{wait}, v_{backup} \rangle, \tag{3.46}$$

де $v_j = Fit(m_j, k)$ - булевий прапорець, чи задовольняє поточний прогноз часу вимогам режиму j.

Логіку станів та переходів між ними представлено на рисунку 3.12. Переходи між рівнями «дозволу» відбуваються суворо послідовно та базуються на накопиченні лічильника – $C[k]$. Логіка оновлення стану $q[k+1]$ та лічильника $C[k+1]$ на кожному такті наступна:

- стан L_{block} та накопичення статистики для переходу до SLEEP (рисунок 3.13);

- стан L_{sleep} та накопичення статистики для переходів до WAIT та BACKUP (рисунок 3.14).

Розглянемо інтеграцію супервізора та механізма розрахунку часу перебування в енергозберігаючому режимі. Супервізор $\mathcal{A}$ (3.44) фактично накладає маску (маскує) на рішення метода розрахунку тривалості сну (3.43). Припустимо, що Fit-функції це частина алгоритму $M_{algo}[k]$ – тобто алгоритму викладеного в 3.42.

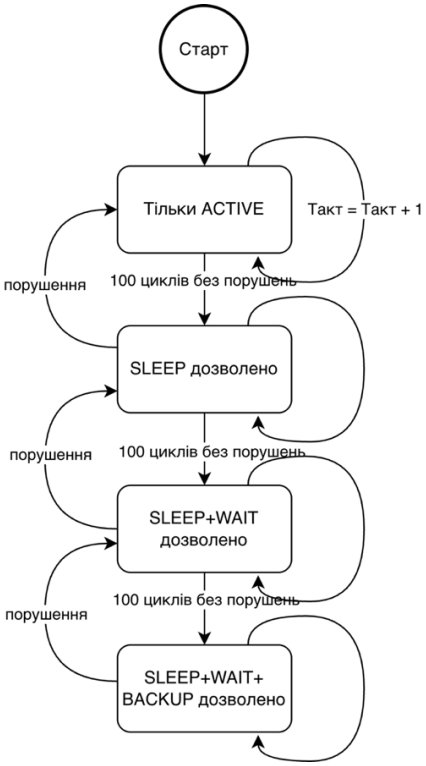

Рисунок 3.12 – Логічна схема переходів між режимами енергозбереження як машина станів

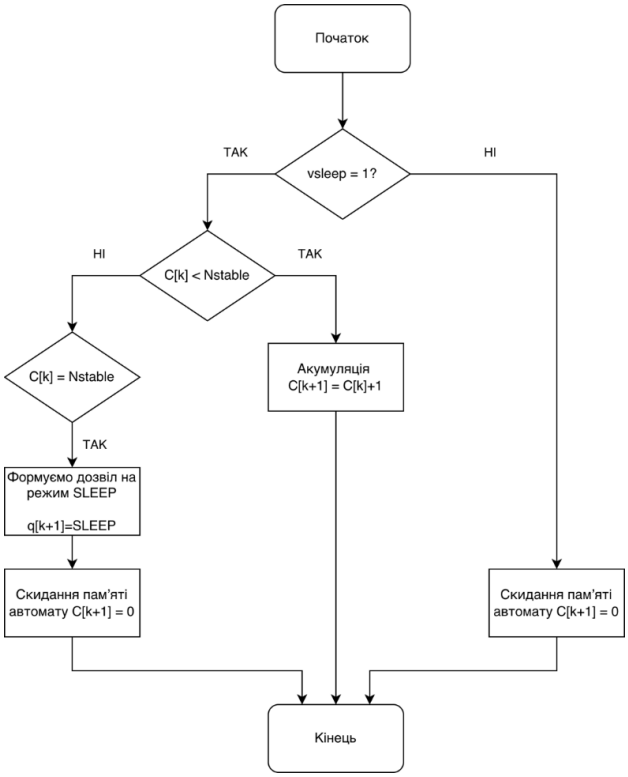

Рисунок 3.13 – Алгоритм накопичення пам’яті та формування дозволу переходу до режиму SLEEP

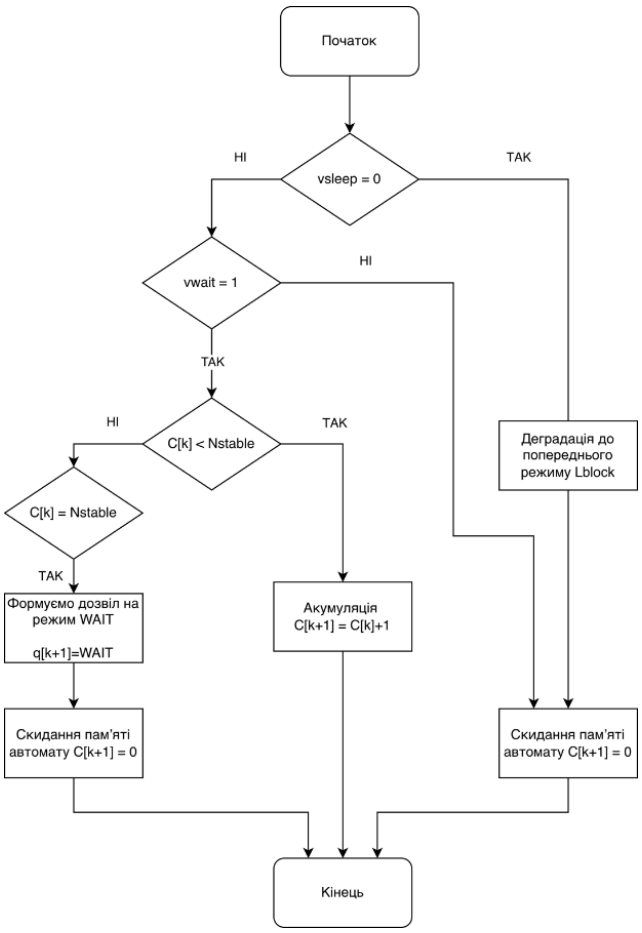

Рисунок 3.14 – Алгоритм накопичення пам’яті та формування дозволу переходу до режимів WAIT та BACKUP

Введемо множину режимів, що дозволені поточним станом автомата $\mathcal{A}$ (3.44) – $Permitted(q[k])$, що може бути описана наступним чином:

Таблиця 3.3 Множина режимів автомату переходів

Стан маски	Дозволені автоматом $\mathcal{A}$ режими роботи
$Permitted(L_{block}) = \{0\}$	Дозволено тільки режим ACTIVE
$Permitted(L_{sleep}) = \{m_{sleep}\}$	Дозволені режими – ACTIVE, SLEEP
$Permitted(L_{wait}) = \{m_{sleep}, m_{wait}\}$	Дозволені режими – ACTIVE, SLEEP, WAIT
$Permitted(L_{backup}) = \{m_{sleep}, m_{wait}, m_{backup}\}$	Дозволені всі режими

Таким чином, фінальне рішення системи $M_{final}[k]$ можна формалізувати наступним чином:

$$M_{final}[k] = \begin{cases} M_{algo}[k], \text{ якщо } M_{algo}[k] \in Permitted(q[k]) \\ max\big(Permitted(q[k])\big), \text{ якщо } M_{algo}[k] \notin Permitted(q[k]) \text{ і } Permitted \neq 0 \cdot \\ ACTIVE \end{cases} \tag{3.47}$$

Введення надбудови у вигляді цифрового автомата з пам'яттю (State Machine Supervisor) дозволяє реалізувати стратегію "консервативного поглиблення сну". Вимоги до накопичення статистики (3.45) та послідовного проходження рівнів (3.46, 3.48, рисунок 3.12) формують часовий фільтр, що повністю усуває проблему нестабільності при короткочасних змінах режиму роботи ("flapping").

Впорядкована послідовність станів буде визначатись наступним ланцюгом:

$$L_{block} \xrightarrow{N_{stable}} L_{sleep} \xrightarrow{N_{stable}} L_{wait} \xrightarrow{N_{stable}} L_{backup}. \tag{3.48}$$

Такий підхід дозволяє реалізувати топологію переходів, де метод явно розділяє поняття "дозволеного рівня" (Permission Level) та "фізичного переходу". Це дозволяє реалізувати можливість фізичних переходів - тільки зі стану ACTIVE, що вимагається необхідністю виконувати фонові задачі для поточного обчислювального циклу системи керування. Автомат керує саме множиною доступних цільових станів для цього переходу. Наприклад, перебування автомата в стані L_{wait} не змушує систему переходити в WAIT, але дозволяє планувальнику обрати цей режим при наступному виході в Idle ОСРЧ, якщо розрахунковий час сну це дозволяє.

Реалізована політика суворого скидання, в свою чергу, гарантує безпеку системи: будь-яке порушення часових обмежень (наприклад, раптове

переривання під час накопичення статистики) миттєво анулює "довіру" до поточного прогнозу, повертаючи систему на попередній, перевірений рівень енергозбереження. Це забезпечує адаптацію до найгірших умов (Worst-Case) навіть за наявності попередньої позитивної статистики.

Висновки до розділу 3

У третьому розділі дисертації розроблено та обґрунтовано методи ідентифікації параметрів комплексної математичної моделі бортового обчислювача, а, також, метод адаптивного регулювання режимів його роботи з урахуванням обмежень реального часу та вимог до енергоефективності. Основний акцент зроблено на забезпеченні реактивності бортового програмного забезпечення наносупутників класу CubeSat за мінімально можливих енергетичних витрат БО.

У межах ідентифікації параметрів моделі проведено критичний аналіз архітектурних та алгоритмічних рішень. Експериментально доведено недоцільність використання контейнеризації на основі WebAssembly у бортових застосуваннях через кратне (у 25 – 50 разів) зростання часових витрат, несумісне з вимогами реального часу. Аналогічно встановлено, що застосування динамічного масштабування частоти та політики race-to-dark не забезпечує прийнятного компромісу між енергозбереженням і реактивністю, що обґрунтувало їх виключення з подальшого моделювання. Отримані негативні результати мають принципове значення, оскільки дозволили звузити область застосовності моделей і сформувати коректну постановку задачі ідентифікації.

Для оцінювання параметрів комплексної математичної моделі обрано рекурсивний метод найменших квадратів, який забезпечує онлайн-ідентифікацію з урахуванням нестаціонарності навантаження та обмежених обчислювальних ресурсів. Розроблено детальні методики вимірювання показників реактивності, обчислювальної продуктивності та енергоспоживання, що ґрунтуються на керованих експериментах із

використанням високоточних вимірювальних засобів. Перевірка адекватності моделей за допомогою коефіцієнта детермінації та критерію Фішера підтвердила статистичну значущість моделей реактивності та продуктивності, а, також, обмежену, але достатню для передпроєктного етапу, точність моделі енергоспоживання.

У другій частині розділу запропоновано метод адаптивного регулювання режимів роботи бортового обчислювача в системах послабленого жорсткого реального часу. Метод базується на оцінюванні інтенсивності подій, їх кластеризації за критичністю, прогнозуванні безпечного горизонту сну та каскадному виборі енергозберігаючих режимів. Введення цифрового автомата з пам’яттю забезпечує стійкість керування, запобігає нестабільним перемиканням і гарантує дотримання часових обмежень навіть за різкої зміни умов роботи. У сукупності отримані результати формують науково обґрунтовану основу для побудови енергоефективних і реактивних бортових обчислювачів наносупутників.

Таким чином, в розділі вирішені четверте, п’яте та шосте завдання дисертації. Наукові результати, отримані в третьому розділі, представлено в наукових публікаціях [143, 157, 161, 162, 163].

РОЗДІЛ 4 АЛГОРИТМИ ТА ЇХ ПРОГРАМНА РЕАЛІЗАЦІЯ У БОРТОВОМУ ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ НАНОСУПУТНИКІВ CUBESAT

Четвертий розділ дисертаційної роботи присвячено розробці та верифікації алгоритмів керування споживанням енергії та реактивністю бортового обчислювача наносупутника. Основна увага зосереджується на практичній реалізації запропонованих підходів у складі бортового програмного забезпечення, що функціонує в умовах жорстких ресурсних обмежень і вимог реального часу, характерних для космічних апаратів малого класу.

Частиною реалізації алгоритмів та підходів є архітектурний дизайн самого бортового програмного забезпечення (БПЗ), як невід'ємної частини комплексного підходу до вирішення польотних місій з урахуванням обмежень та вимог до такого ПЗ.

Бортове ПЗ виконується мікроконтролером або мікропроцесором БО наносупутника. БО є ключовим елементом платформи, відповідальним за виконання задач керування, збору та обробки телеметричної інформації, діагностики та керуванням відмов наносупутника, а, також, обслуговування корисного навантаження. Водночас, його робота обмежена суворими енергетичними бюджетами, обмеженою обчислювальною потужністю та необхідністю забезпечення детермінованої реактивності системи у відповідь на зовнішні та внутрішні події. За таких умов, ефективне керування режимами роботи МК, плануванням задач та використанням програмних ресурсів набуває критичного значення для забезпечення стабільної та надійної роботи наносупутника протягом усієї місії.

У даному розділі розглядаються алгоритми, спрямовані на динамічну адаптацію режимів роботи бортового обчислювача з метою зменшення енергоспоживання та підтримання необхідного рівня реактивності програмної системи. Окрему увагу приділено питанням верифікації цих алгоритмів,

зокрема перевірці їх коректності, часових характеристик та впливу на загальну поведінку бортового програмного забезпечення в умовах реального часу під час фізичного лабораторного та квазіексплуатаційного експериментів.

Для побудови алгоритмів, що можуть забезпечити ефективне застосування у БО наносупутника, треба сформулювати загальні вимоги до середовища виконання таких алгоритмів, а саме структури БПЗ, його побудови та складових частин.

4.1 Вимоги до бортового програмного забезпечення БО

4.1.1 Вимоги до бортового програмного забезпечення БО наносупутників CubeSat

До БПЗ висуваються такі узагальнені вимоги та проєктні обмеження, узгоджені з підходами ECSS-E-ST-40/ECSS-Q-ST-80 до космічного ПЗ та сучасною практикою розроблення польотного ПЗ для наносупутників.

Вимоги реального часу та планування задач. БПЗ повинно забезпечувати детерміноване виконання критичних циклів керування (ADCS, захист живлення, обробка телеметрії, реакція на збої) із гарантованими часовими обмеженнями, які узгоджуються з вимогами до затримки команд та телеметрії (не більше 100 мс для критичних алгоритмів керування). Планувальник задач ОС має підтримувати пріоритетність задач та примусове (preemptive) витіснення для забезпечення своєчасного виконання високо пріоритетних функцій, або зовнішніх переривань.

Відмовостійкість, FDIR та самовідновлення. З огляду на радіаційне середовище та вимоги до надійності, БПЗ повинно реалізовувати механізми детектування, ізоляції та відновлення збоїв (англ. Fault Detection, Isolation and Recovery, FDIR), включно з багаторівневим моніторингом стану підсистем, виявленням відхилень телеметрії, переходом у безпечний режим та ініціюванням коригувальних дій (перезапуск задач, перезавантаження підсистем, конфігураційне перезавантаження БО).

БПЗ має інтегрувати апаратний сторожовий таймер, підтримувати наявні механізми контролю цілісності пам’яті (англ. Error Detection And Correction, EDAC) та логіки, а також алгоритми початкового (під час включення) та періодичного самотестування, що мінімізують ймовірність невиявлених відмов до рівня, сумісного з цільовим напрацюванням на відмову (англ. Mean Time Before Failure, MTBF) та вимог до успішності місії.

Управління ресурсами та енергетично-обмежене виконання. Через обмежений енергобаланс та теплові умови експлуатації наносупутника, БПЗ має підтримувати багато режимів енергоспоживання (активний, зниженої частоти, режим сну, режим глибокого сну), конфігурацію живлення периферії та періодичність опитування датчиків згідно з поточним профілем місії та доступною потужністю. БПЗ має надавати гнучкі механізми планування, як системних, так і прикладних бортових завдань.

Проєктні обмеження включають жорсткі ліміти на використання оперативної пам’яті, обсяги енергонезалежної пам’яті під журнали та дані місії, а, також, на частоту доступу до зовнішніх периферійних пристроїв (пам'ять, датчики) для зниження споживаної потужності й тепловиділення.

Надійний обмін даними та інтеграція з інтерфейсами. В умовах множинних інтерфейсів (I²C, SPI, CAN, UART, D-Shot, PWM, CRSF, S-BUS) та високого рівня електромагнітних завад БПЗ повинно забезпечувати захищений від помилок обмін даними: застосовувати контроль цілісності даних, механізми повторної передачі, тайм-аути та неблокуючий доступ до шин, а, також, чітке розділення пріоритетів трафіку (критичні команди, телеметрія, дані корисного навантаження).

Архітектура БПЗ має підтримувати модульні драйвери інтерфейсів та узгоджений формат телеметрії/телекоманд, що спрощує інтеграцію з іншими підсистемами та наземним сегментом.

Вимоги до конфігурованості, оновлюваності та безпеки. БПЗ повинно підтримувати безпечне віддалене оновлення «через повітря» (англ. Firmware Over the Air, FOTA) із використанням криптографічного захисту

завантажуваних образів ПЗ (цифровий підпис, хеш-функції) та механізмом двох банків пам’яті, який гарантує можливість повернення («відкату») до останньої працездатної версії у разі збою встановлення оновлення.

Конфігураційні параметри місії (розклад виконання бортових завдань, режими корисного навантаження, керування енергоспоживанням, пороги FDIR) мають зберігатися в енергонезалежній пам’яті з контролем цілісності та можливістю контрольованої зміни з боку наземної станції.

Кожна підсистема та програмний модуль БПЗ повинні забезпечувати формування телеметричної інформації та приймання телекоманд через уніфікований інтерфейс обміну з бортовою системою керування даними. Телеметрія повинна включати щонайменше інформацію про стан, режими роботи, помилки та діагностичні параметри модуля, а телекерування - команди керування режимами, конфігураційними параметрами та ініціації сервісних процедур. Зазначені механізми повинні забезпечувати підтримку функцій моніторингу, FDIR, дистанційної реконфігурації та безпечної експлуатації наносупутника протягом усього терміну місії.

Вимоги до структури, модульності та верифікації. З урахуванням стандартів ECSS архітектура БПЗ має бути структурована на ієрархію модулів (служби ядра, драйвери, утиліти, управління місією, FDIR, обробка даних, комунікація), з чітко визначеними інтерфейсами та трасуванням вимог «від місії до коду». БПЗ поділяється на два шари: системне програмне забезпечення (СПЗ), що не залежить від конкретної місії та виконує системні завдання БПЗ, та польотне (або місійне) програмне забезпечення, що включає програмні модулі, що специфічні для конкретної місії та/або конфігурації наносупутника.

Проєктні обмеження передбачають застосування формальних процедур верифікації, включно з HIL-тестами, що відповідають вимогам стандартів як ECSS, так і RTCA DO-178C (EUROCAE ED-12C), а також проведення випробувань БПЗ у сценаріях, що відтворюють запуск, перехід у безпечний режим (Safe Mode), тривалу автономну роботу та типові відмови апаратури.

Зазначені вимоги та обмеження забезпечують узгодженість БПЗ із реальними умовами експлуатації БО наносупутників, вимогами до його надійності та енергетичними/конструктивними лімітами, а, також, створюють основу для подальшої формалізації специфікації БПЗ відповідно до стандартів ECSS, RTCA/EUROCAE та сучасних підходів до розроблення БПЗ для бортових обчислювальних платформ.

4.1.2 Вимоги до бортового програмного забезпечення БО БПЛА

На відміну від маневрених дронів тактичного рівня, БПЛА класів HALE/MALE виконують довготривалі місії у верхніх шарах атмосфери, що зближує профіль їх роботи з орбітальними апаратами. Архітектура авіоніки таких систем зазвичай передбачає розділення контурів керування: низькорівнева стабілізація та керування двигунами/поверхнями виконується окремою відмовостійкою підсистемою (Flight Controller + Electric Speed Controller), тоді як БО виконує роль комп'ютера місії (Mission Computer). Така архітектура формує специфічні вимоги до БПЗ, що корелюють з вимогами до супутникових систем.

Додатково, з урахуванням специфіки місій HALE/MALE, до бортових обчислювачів БПЛА висуваються підвищені вимоги щодо тривалої безперервної роботи (endurance) та деградаційної відмовостійкості: БПЗ має підтримувати контрольовану деградацію функцій із збереженням безпечної траєкторії польоту та можливості завершення місії або повернення апарата. З огляду на типовий для БПЛА класів HALE/MALE сценарій експлуатації поза межами прямої візуальної видимості (англ. Beyond Visual Line of Sight, BVLOS), БПЗ повинно бути орієнтоване на високий рівень автономності, а взаємодія з оператором обмежується переважно етапами запуску, набору висоти та посадки, а, також, винятковими ситуаціями. У крейсерському режимі та під час виконання місії БПЗ має забезпечувати автономне виконання польотного плану з мінімальною залежністю від зовнішніх команд, за умови

збереження пріоритету каналів командно-керувального зв’язку (англ. Command&Control, C2) для команд безпеки та аварійного втручання.

Таким чином, схожість архітектури (розшарування авіоніки на відокремлені підсистеми: стабілізації, корисного навантаження, КОМ) та умов експлуатації (дефіцит енергії, автономність) робить можливим пряме перенесення архітектурних рішень БПЗ з наносупутника на БПЛА класу HALE/MALE.

Питання кіберстійкості бортового програмного забезпечення та каналів зв’язку, не розглядаються детально та винесені за межі цього дисертаційного дослідження, оскільки вони потребують окремого комплексного аналізу, специфічних моделей загроз та методів оцінювання, що не входять до поставлених наукових завдань.

4.2 Архітектура бортового програмного забезпечення

Розроблена архітектура базується на ОСРЧ FreeRTOS з фреймворком F′Prime, що дозволяє створювати структуровані компоненти системи, що повторно використовуються та легко масштабуються. ОСРЧ FreeRTOS [164] гарантує детерміноване виконання задач і ефективно підтримує ресурсно-обмежені вбудовані системи. Компонентно-орієнтований підхід на базі фреймворку NASA F′ надає засоби для модульної розробки, моделювання, тестування та верифікації компонентів у форматі фреймворку, сумісному з вимогами космічних програм NASA JPL. Такий підхід дозволяє гарантувати модульність, спостережуваність та відмовостійкість.

4.2.1 Обґрунтування використання F′ Prime як основи побудови БПЗ

Фреймворк F′ (F′ Prime), розроблений у Лабораторії реактивного руху NASA (Jet Propulsion Laboratory, JPL), є відкритою платформою для розробки вбудованого програмного забезпечення, призначеною для систем, що працюють у режимі реального часу – зокрема, в космічних місіях, робототехнічних комплексах та літальних апаратах. Архітектура платформи

побудована на компонентно-орієнтованому підході, який забезпечує чітке розділення функціональності між модулями, що, у свою чергу, підвищує надійність, спрощує тестування та повторне використання коду в інших проєктах. Фреймворк F′ довів свій польотний досвід на таких місіях як Mars Helicopter – Ingenuity [165], NEA Scout [166] та Lunar Flashlight [167], де модульність та спроможність виконання завдань жорсткого реального часу були вирішальними чинниками обрання. Фреймворк активно підтримується та розвивається, останньою версією є 4.1.4 (грудень 2025).

Ключові елементи архітектури F′ включають (рисунок 4.1):

– компоненти – базові функціональні блоки, які інкапсулюють певну поведінку або алгоритм і взаємодіють між собою через визначені інтерфейси;

– порти – механізми зв’язку, що дозволяють компонентам обмінюватися подіями або даними через жорстко типізовані інтерфейси;

– топології – що визначають, як компоненти підключені один до одного. Вони описують логіку взаємодії компонентів, включаючи машину станів, що координує їх роботу та забезпечує реалізацію місійних і системних функцій. Топологія, зазвичай, відображається у вигляді діаграми або моделі, що спрощує аналіз структури ПЗ.

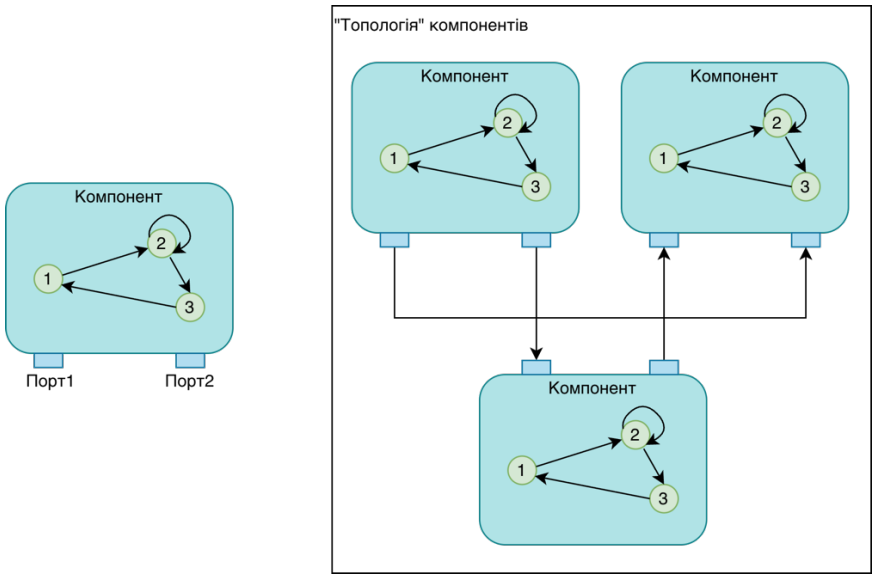

Рисунок 4.1 – Архітектурні рішення фреймворку F′ – порти, компоненти та топології

Наявність таких універсальних компонентів дозволяє проектувати структуроване, детерміноване та легко масштабоване бортове програмне забезпечення, що відповідає вимогам систем критичного призначення бортових обчислювачів наносупутників CubeSat.

Для використання фреймворку F′ на цільовій операційній системі FreeRTOS, реалізовано прошарок абстракції від операційної системи (англ. Operation System Abstraction Layer, OSAL). Цей прошарок дозволяє реалізацію ПЗ що є інтероперабельним та легко переноситься, та може використовуватись на різних ОСРЧ.

Програмування для складових OSAL виконується на мові програмування – C або C++, а програмування на рівні компонентів та топологій F′ - суто на мові – C++.

Окрім архітектурних переваг, F′ надає набір готових до інтеграції компонентів, що істотно зменшує обсяг розробки базової інфраструктури БПЗ. Фреймворк містить стандартні компоненти для:

– прийому та диспетчеризації команд (CmdDispatcher);

– виконання командних послідовностей (CmdSequencer);

– збору та маршрутизації телеметрії (TlmChan);

– зберігання та оновлення параметрів (PrmDb);

– подієвого й текстового логування (ActiveLogger, ActiveTextLogger);

– підвищення надійності - механізми контролю працездатності компонентів (Health) тощо.

Наявність цих компонентів «з коробки» дозволяє зосередитися на реалізації місійно-специфічної логіки, підвищуючи перевикористання коду, узгодженість архітектури та загальну надійність польотного програмного забезпечення для наносупутників.

4.2.2 Обґрунтування використання ОСРЧ FreeRTOS

Для забезпечення детермінованого виконання компонентів фреймворку F′ та ефективного керування обчислювальними ресурсами обрано операційну

систему реального часу – FreeRTOS (Free Real-Time Operating System). FreeRTOS забезпечує розподіл процесорного часу між завданнями, синхронізацію процесів і управління пам’яттю, що створює стабільне середовище для роботи бортового програмного забезпечення.

При виборі ОСРЧ для бортового обчислювача наносупутника враховувалися такі ключові чинники:

– обмеженість доступних ресурсів (обсяг пам’яті, енергоспоживання, продуктивність процесора);

– детермінованість виконання завдань і передбачувана затримка реакції;

– простота коду та можливість модифікації під специфіку місії;

– наявність перевіреного досвіду застосування в авіаційних і космічних проєктах;

– широке галузеве використання та підтримка;

– час на адаптацію реалізацій компонентів, що вже були розроблені під інші ОСРЧ або фреймворки за допомогою «обгорток» ОСРЧ;

– зрозуміле та просте ліцензування, що було б доступним як для академічних, так і комерційних проєктів.

FreeRTOS – це портативна, масштабована та мінімалістична операційна система реального часу з відкритим вихідним кодом, спеціально розроблена для вбудованих систем. Її основна мета – забезпечити ефективне та детерміноване управління виконанням завдань у пристроях з обмеженими ресурсами, таких як мікроконтролери.

Код FreeRTOS є відкритим і поширюється за ліцензією Масачусетского Інституту Технології (MIT - Massachusetts Institute of Technology), що є найвільнішою ліценцію на повторне використання ПЗ, яка дозволяє адаптувати код FreeRTOS під специфічні умови місії та інтегрувати з архітектурними надбудовами, такими як F′. В результаті FreeRTOS забезпечує реалізацію системного рівня БПЗ, тоді як F′ формує логіку компонентної взаємодії та керування процесами та завданнями місії, створюючи

багаторівневу архітектуру для надійного функціонування бортового обчислювача.

Архітектура FreeRTOS показана на рисунку 4.2.

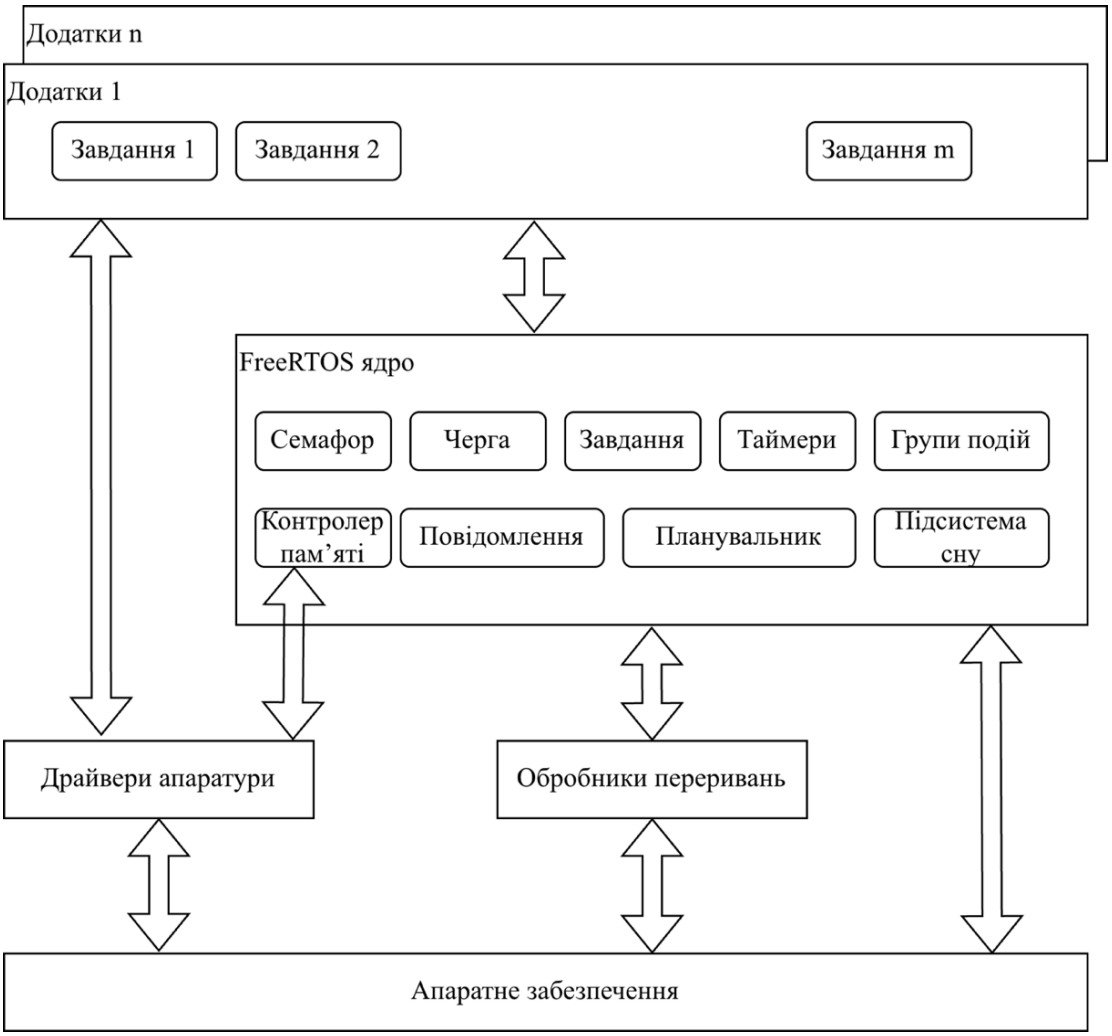

Рисунок 4.2 – Архітектура та складові ОСРЧ FreeRTOS

4.2.3 Шар абстракції від апаратних реалізацій (HAL)

Для ефективного використання апаратних ресурсів БО «Боривітер» та використаного для його побудови МК – Microchip PIC32CZ2051CA70144 на базі ядра ARM Cortex-M7 (або його аналог Atmel ATSAMV71Q21), використано функціональність програмного фреймворку абстракції від апаратних реалізацій HAL (Hardware Abstraction Layer) - MPLAB Harmony v3 [168], що надається виробником МК – Microchip (рисунок 4.3).

Шар HAL є критично важливою складовою, що забезпечує стандартизований інтерфейс для взаємодії програмного забезпечення прикладного рівня з периферійними пристроями МК. Цей рівень абстракції усуває пряму залежність коду від апаратних особливостей конкретної архітектури, або МК, що значно спрощує портування та підвищує модульність розроблених програмних рішень. Таким чином, HAL від MPLAB сприяє реалізації надійної вбудованої системи, що масштабується та легко підтримується.

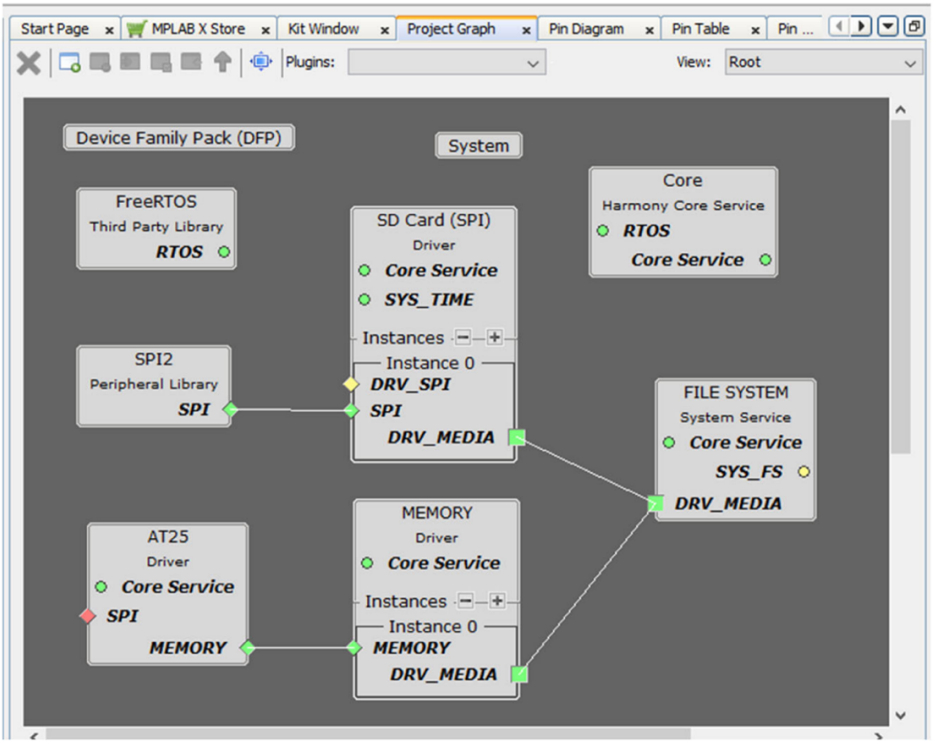

Рисунок 4.3 – Фреймворк HAL від MPLAB Harmony v3 – приклад конфігурації для взаємодії з FreeRTOS

Фреймворк Harmony v3 для серії PIC32CZ інтегрує стандартні заголовки CMSIS (Cortex Microcontroller Software Interface Standard), що забезпечує сумісність з екосистемою ARM. Така підтримка додає переносимість на рівні драйверів на інші МК з архітектурою ARM® Cortex™ серії M.

Для повної системної інтеграції з апаратурою МК та інших інтегральних схем БО було використано два шари фреймворку HAL Harmony v3: PLIBs

(згенерований код, що записує в регістри периферії МК) та Drivers (реалізує блокуючі драйвери периферійних автоматів, наприклад I2C, UART, тощо).

4.2.4 Інтеграція HAL, FreeRTOS і F´ для БО “Боривітер”

Програмно-апаратна архітектура обчислювального середовища для реалізації бортового програмного забезпечення БО “Боривітер” реалізована у вигляді чотирьох-шарової структури, яка поєднує фреймворк апаратної абстракції - HAL, ОСРЧ FreeRTOS та компонент-орієнтований фреймворк F´ на програмному рівні, та апаратне забезпечення БО «Боривітер», що базується на МК PIC32 та множині апаратних периферійних пристроїв БО (рисунок 4.4).

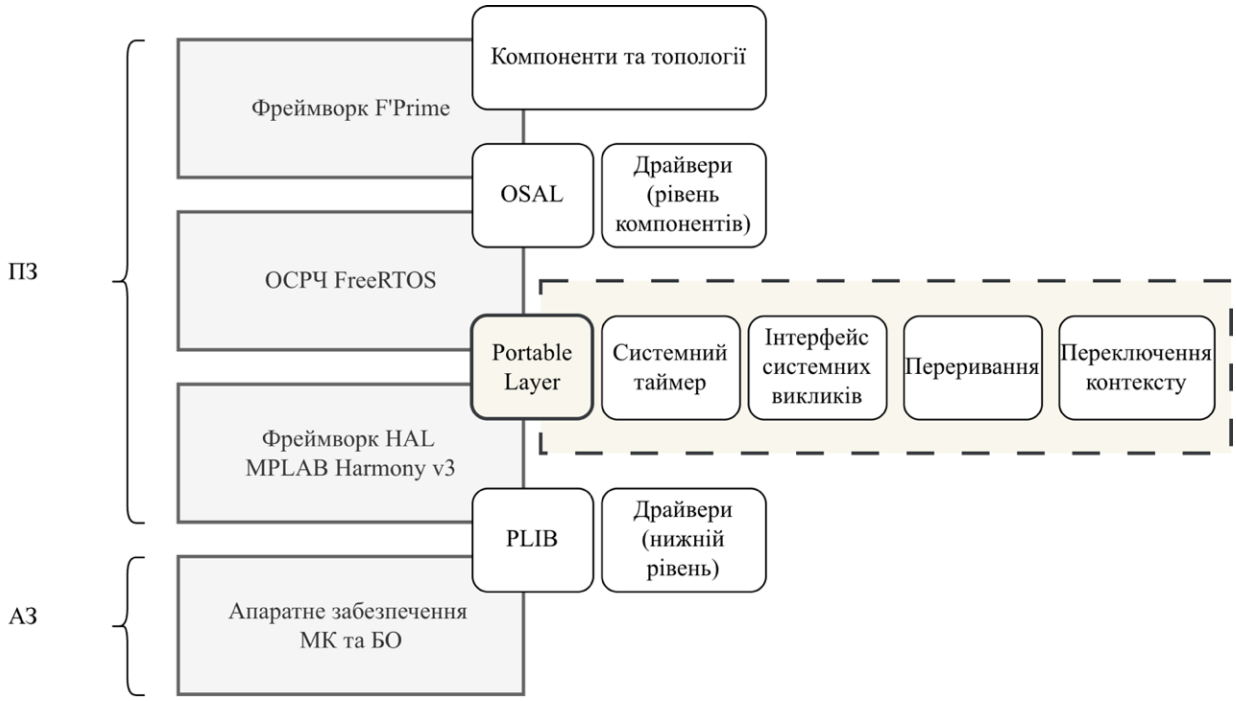

Рисунок 4.4 – Комбінована апаратно-програмна архітектура для побудови ПЗ

Кожен із цих рівнів виконує власну функцію у забезпеченні стабільної, детермінованої та масштабованої роботи системи, а їх інтеграція формує цілісне середовище для розроблення, тестування та виконання БПЗ.

Для реалізації взаємодії різних шарів побудови БПЗ, кожен з компонентів пропонує «обгортки» (wrappers), що дозволяють виконати абстрагування від специфічних інтерфейсів складових БПЗ. Такі «обгортки»

дозволяють уникнути щільної зв’язки різних шарів ПЗ та забезпечити їх універсальність. Універсальність шарів БПЗ призводить до менших змін впродовж життєвого циклу ПЗ та дозволяє накопичувати польотну історію (англ. flight heritage), що є галузевою метрикою надійності компонентів наносупутників та інших КА.

Фреймворк HAL MPLAB Harmony v3 забезпечує стандартизований інтерфейс між апаратним забезпеченням і системним програмним рівнем, спрощуючи взаємодію з мікроконтролером та периферійними пристроями. Реалізує підтримку шару портування ОСРЧ FreeRTOS, а саме: системного таймеру, інтерфейсу системних викликів, обробників переривань та засобів переключення обчислювального контексту.

Основні завдання HAL, що реалізуються його складовими PLIB та драйверами:

– інкапсуляція доступу до периферійних модулів (UART, SPI, I2C, CAN, ADC, PWM, GPIO тощо);

– реалізація драйверів вводу-виводу для внутрішніх і зовнішніх інтерфейсів;

– адаптація під конкретні схемотехнічні рішення БО “Боривітер”, включно з контролем живлення, сенсорними модулями та засобами зв’язку;

– надання системних примітивів для вищого рівня ОСРЧ, таких як системний таймер, обробники переривань та інтерфейси системних викликів і переключення контексту;

– забезпечення портованості коду: вищі рівні ПЗ (FreeRTOS та F´Prime) залишаються незалежними від конкретної апаратної реалізації.

Таким чином, HAL виступає фундаментом архітектури, який уніфікує роботу з апаратними ресурсами, знижує ризики помилок у роботі з периферією та спрощує оновлення апаратної платформи без зміни верхніх рівнів програмного забезпечення.

Над апаратним шаром функціонує операційна система реального часу FreeRTOS, яка забезпечує планування, синхронізацію та керування ресурсами

в межах бортового обчислювача. FreeRTOS виконує роль ядра виконання задач, що координує роботу всіх компонентів і гарантує детерміноване виконання критичних процесів у межах заданих часових обмежень. FreeRTOS формує стабільне середовище виконання для компонентів F´Prime, забезпечуючи мінімальні затримки реакції та прогнозовану поведінку системи навіть в умовах обмежених ресурсів. Основні функції системного рівня:

– диспетчеризація завдань за пріоритетами, що забезпечує передбачувану реакцію системи;

– синхронізація потоків через черги, семафори, події та м’ютекси;

– керування пам’яттю й стеком кожного завдання;

– обробка таймерів і переривань, включно з відкладеними подіями;

– надання «обгорток» для активних компонентів F´ у вигляді системних завдань (tasks);

– підтримка класичних алгоритмів планування (RMS, EDF).

Найвищий рівень архітектури становить фреймворк F´, який реалізує логіку місійних і системних функцій у вигляді компонентно-орієнтованої структури, а також надає доступ до виконання цих компонентів засобами ОСРЧ FreeRTOS через шар абстракції – OSAL.

Кожен компонент інкапсулює окрему функціональність – від обробки телеметрії до керування виконавчими механізмами – і взаємодіє з іншими через чітко визначені порти. Взаємодія між компонентами реалізується через механізм портів і топологій, що вбудовано у F´. У процесі виконання кожен компонент або група компонентів працює у контексті задач FreeRTOS, використовуючи її механізми синхронізації та обміну даними. Модель топології системи визначає логічні зв’язки між компонентами, а машина станів координує їхню роботу відповідно до сценарію місії. Таке розділення дозволяє швидко модифікувати або замінювати окремі частини ПЗ без втручання у решту системи, що критично важливо для авіоніки наносупутників.

4.2.5 Архітектура БПЗ

Для побудови БПЗ запропоновано розбиття всього ПЗ на два основних рівні: польотне програмне забезпечення (ППЗ), що може підмінятись тестовим програмним забезпеченням (ТПЗ), та системне програмне забезпечення (СПЗ).

СПЗ відповідає за виконання системних функцій, як-то планування завдань на рівні ОСРЧ та фреймворку, реалізація підсистем обміну та архівацією бортових даних, реалізація підсистеми заощадження бортової енергії, тощо. СПЗ включає компоненти F′, шар абстракції від ОС – OSAL, та складові операційної системи та модулів підтримки апаратного забезпечення. Компоненти F′ включають наступні складові: стандартні компоненти фреймворку F′, драйвери верхнього та нижнього рівнів та утиліти.

ППЗ відповідає за виконання польотного завдання та взаємодію з зовнішньою до БО авіонікою борту (підсистема комунікації, керування рухом, навігації і т.п.). Усе ППЗ побудовано як множина топологій компонентів F′.

Діаграму розгортання БПЗ на апаратній платформі БО «Боривітер» наведено на рисунку 4.5.

Компоненти-фреймворки HAL та F′, а, також, компонент абстрагування від ОС – OSAL, разом з ОСРЧ FreeRTOS додаються компонувальником (англ. linker) програмного забезпечення у єдиний бінарний файл прошивки разом з БПЗ. Подальше у структурних схемах, прийнято рішення відображати HAL, FreeRTOS, F′ OSAL та інші компоненти F′ (драйвери та утиліти), як частину БПЗ, а, точніше, його складової – СПЗ. Загальна структура верхнього рівня БПЗ зображена на рисунку 4.6, та деталізована на рисунку 4.8.

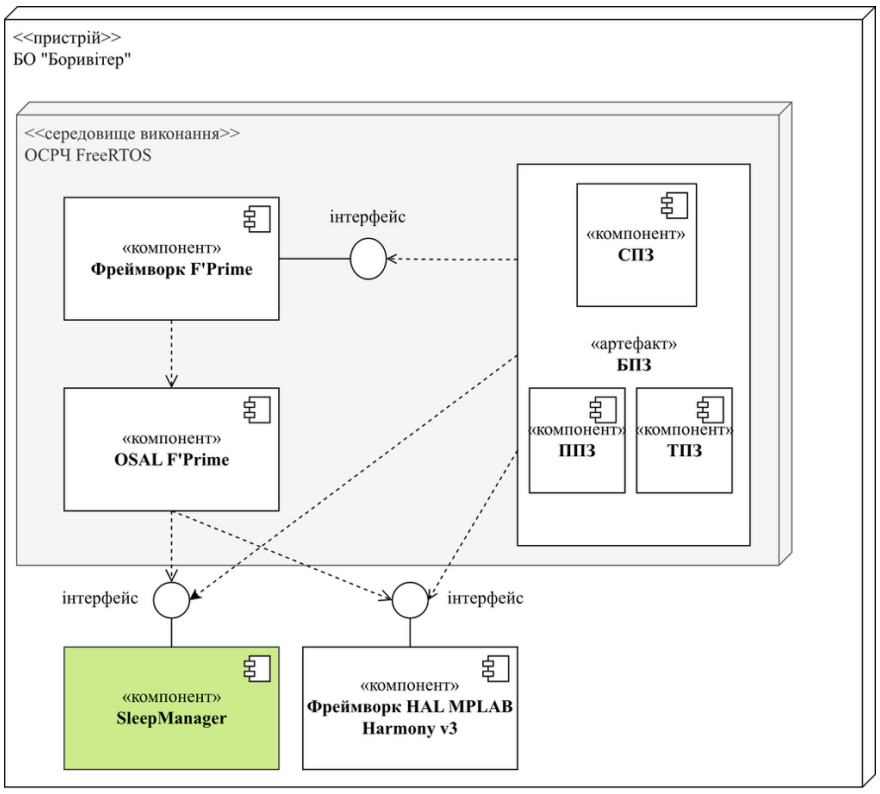

Рисунок 4.5 – UML Діаграма розгортання БПЗ

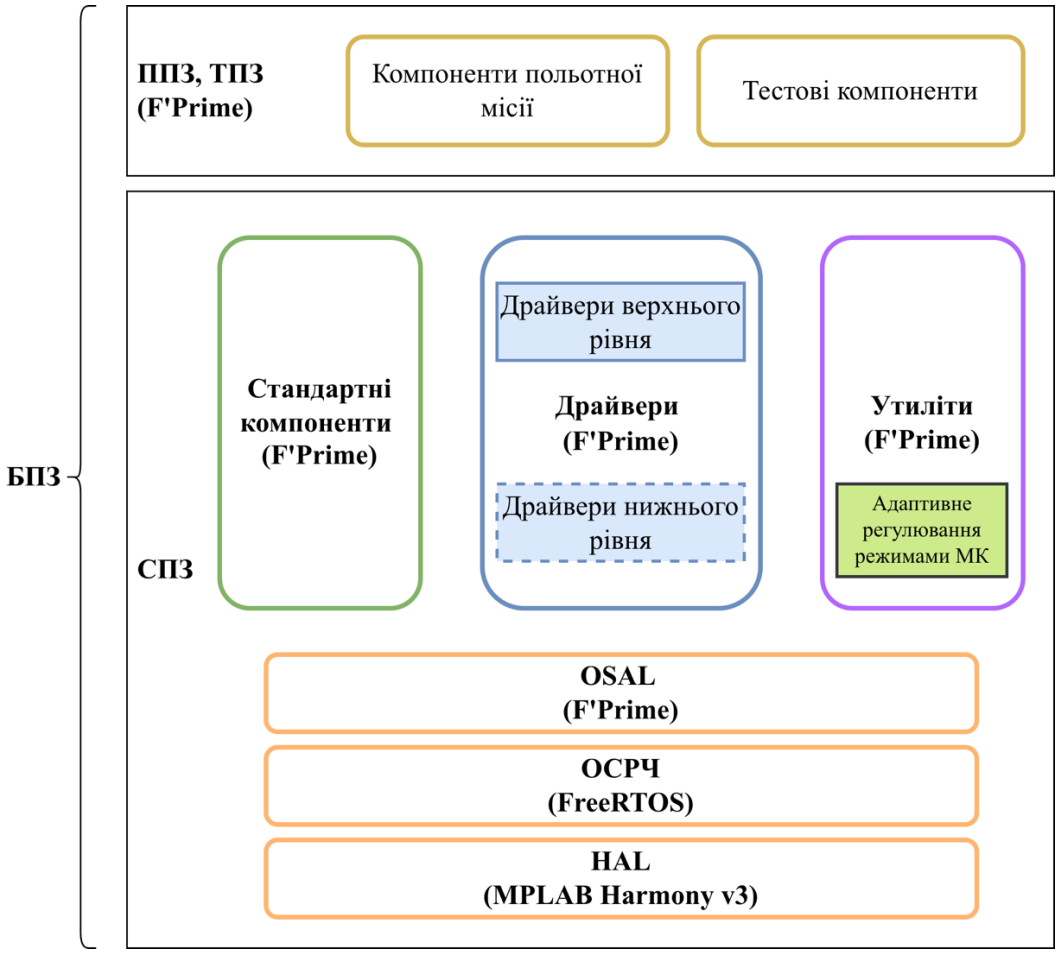

Рисунок 4.6 – Структурна схема БПЗ (верхній рівень)

Розроблене СПЗ складається з двох шарів.

1) **Шар абстракції від операційної системи OSAL** відповідає за абстрагування взаємодії з операційною системою, як то роботи з потоками, таймерами, синхронізацією, файловою системою, та іншими базовими функціями ОС. Цей шар дозволяє розробляти вихідний код незалежним від конкретної операційної системи, що полегшує міграцію на нові платформи, не змінюючи основну логіку компонентів, топологій та функціональних застосунків, що на них будуються. В шарі OSAL реалізовано наступні класи для взаємодії з ОСРЧ FreeRTOS:

– Task (завдання, процеси або потоки) забезпечує контроль та керування завданнями, що виконуються;

– Task Registry (реєстр завдань) призначений для відстеження всіх екземплярів завдань у системі.

– Mutex (мьютекс) надає функціональність мьютекса – типового об'єкта синхронізації, який використовується в багатопоточних або багатозадачних системах для забезпечення взаємовиключного та безпечного доступу до спільного системного ресурсу (ділянка пам’яті або інтерфейс);

– Message queues (черги повідомлень) – механізм міжпроцесної комунікації, який дозволяє процесам або потокам обмінюватися повідомленнями у впорядкованому вигляді, що забезпечує асинхронну передачу даних між різними частинами програми, не вимагаючи їх прямого взаємодії або синхронізації в реальному часі;

– Interval Timer (інтервальний таймер) використовується для вимірювання часу виконання програмного забезпечення, або для позначення часу;

– Watchdog Timer (сторожовий таймер) планує зворотний виклик на вказаний час у майбутньому. Використання сторожового таймеру дозволяє запобігти так званому «зависанню» системи шляхом примусового скидання мікроконтролера у випадку протермінованого часу зворотнього виклику;

– Interrupt Lock (Блокування переривання) запобігає перериванню коду, що виконується;

– File (файл) відповідає за системну абстракцію «файл» що підтримує стандартні функції відкриття, закриття файлів, читання- запису, тощо;

– File System (файлова система) відповідає за системну абстракцію «файлова система» що підтримує функції створення, видалення та перейменування тек та файлів;

– Log (Журнал) – відповідає за реєстрацію системних текстових повідомлень що формуються за заздалегідь визначеними подіями.

2) **Програмні компоненти F′** – стандартні компоненти, драйвери та утиліти. Надають доступ до апаратного забезпечення БО “Боривітер”, вирішують завдання взаємодії з OSAL, надають базові функції F′, реалізовують драйверну модель для апаратного забезпечення БО та реалізують утилітарні системні завдання.

Драйвери, що представлені шарами нижнього та верхнього рівня, абстрагують низькорівневу роботу з периферійними пристроями та фізичними інтерфейсами, дозволяючи іншим частинам системи працювати з апаратними ресурсами через стандартизовані інтерфейси, без прямої залежності від конкретного апаратного забезпечення. Разом із тим, у деяких випадках доцільним є застосування більш складної трьохшарової моделі “Manager – Worker – Driver”, запропонованої розробниками NASA/JPL (рисунок 4.7), яка дозволяє ефективніше керувати доступом до ресурсів та забезпечувати більшу гнучкість у складних сценаріях. Структурну схему компонентів БПЗ та їх розподіл на окремі шари та складові наведено на рисунку 4.8.

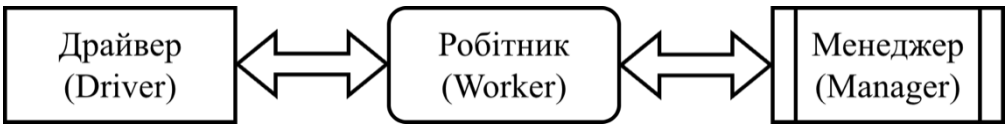

Рисунок 4.7 – Патерн побудови драйверної моделі для застосування в F′

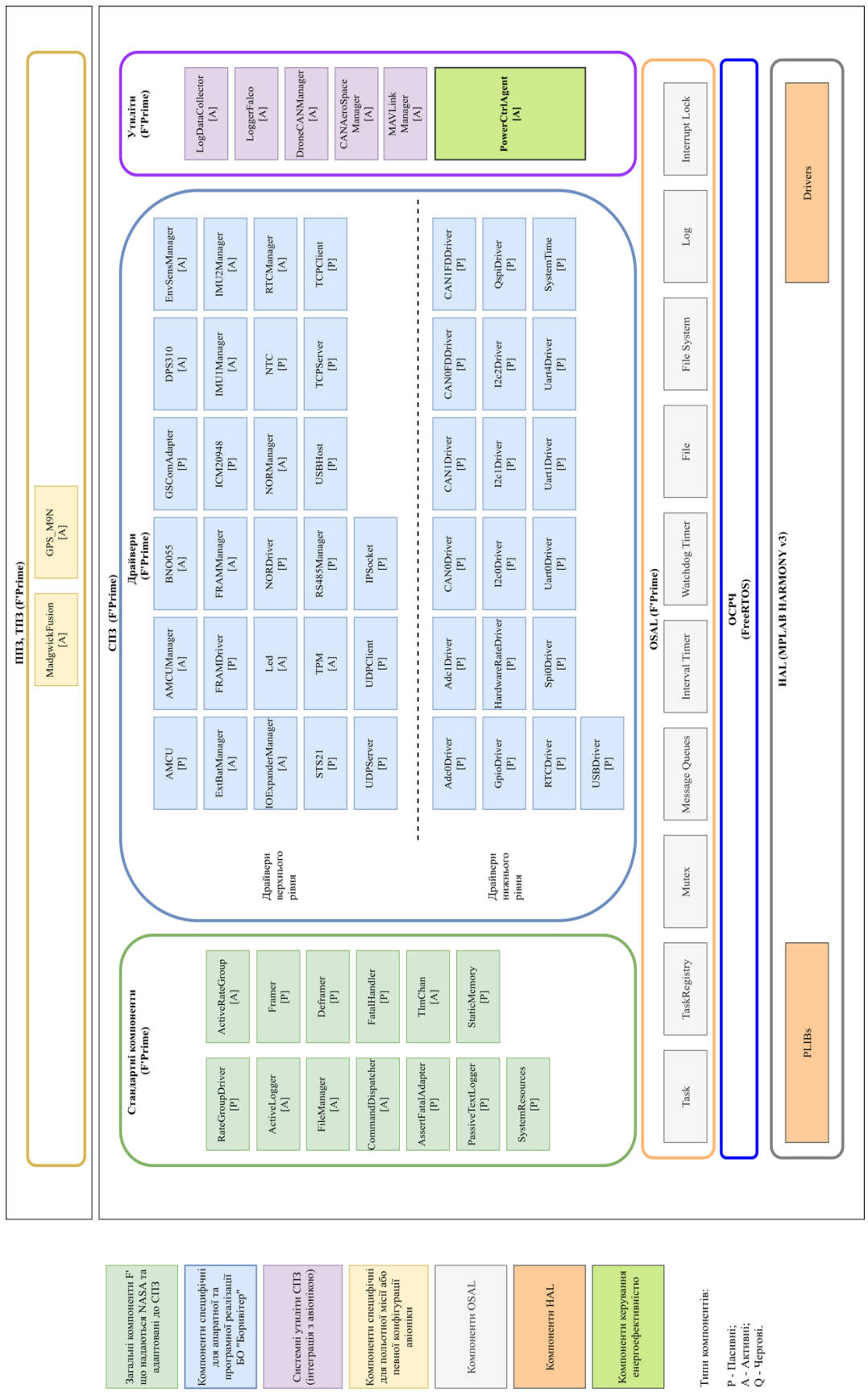

Рисунок 4.8 – Деталізована структурна схема БПЗ адаптованого до БО «Боривітер 0.1»

4.3 Розроблення алгоритмів адаптивного регулювання енергетичними режимами МК БО

4.3.1 Постановка задачі для синтезу адаптивних алгоритмів керування енергетичними режимами МК БО

Розроблення адаптивних алгоритмів керування енергетичними режимами МК є складовою концепції інтелектуального контролю вбудованих систем, у якій енергоспоживання розглядається не як побічний ефект обчислень, а як керований процес, підпорядкований алгоритмам оптимізації.

У сучасних МК, зокрема в PIC32CZ2051CA70144, що є основою побудови БО «Боривітер», реалізовано широкий спектр енергетичних режимів роботи - ACTIVE, SLEEP, WAIT, BACKUP. Проте ці режими є лише механізмами апаратного рівня. Вони не визначають, у яких умовах і наскільки часто система має переходити між станами, аби зберігати баланс між енергоспоживанням і часовими характеристиками - реактивністю.

Саме тому виникає потреба у створенні алгоритмів, що у тісній інтеграції з операційним середовищем FreeRTOS, забезпечують інтелектуальне керування режимами роботи МК та його шин і периферійних підсистем. Реалізація адаптивних алгоритмів керування режимами роботи МК БО «Боривітер» виконана через розроблення компонента F′ – PowerCtrlAgent та системних розширень ОСРЧ FreeRTOS. Ці алгоритми не лише реагують на поточне навантаження, а й формують політику поведінки системи на основі накопиченої історії та прогнозують час простою. Таким чином, енергоспоживання перетворюється з пасивного параметра на динамічну керовану змінну, що може оптимізуватися в режимі реального часу.

Основна ідея полягає в тому, щоб сформувати алгоритмічну архітектуру, здатну:

– відповідати як загальним вимогам до БПЗ, так і вимогам компонентної побудови ПЗ в середовищі F′;

– інтегруватись до наявних механізмів планування завдань ОСРЧ FreeRTOS;

– спостерігати поточний стан системи (навантаження, периферію, таймери);

– оцінювати часові та енергетичні обмеження з урахуванням завдань, що впливають на якість обслуговування системи (QoS - Quality of Service,), а саме оцінювати потенційний час «простою» МК;

– ухвалювати рішення про перехід у певний режим енергоспоживання;

– виконувати цей перехід без втрати обчислювального контексту чи порушення дедлайнів;

– аналізувати наслідки рішень через систему моніторингу;

- адаптувати політику для подальших ітерацій.

Під час розроблення модульної структури алгоритмів було дотримано таких принципів:

- ієрархічність і модульність: різні аспекти керування (оцінка, прийняття рішень, виконання, адаптація) виділені в окремі взаємопов’язані алгоритмічні блоки;

- детермінізм: усі переходи описано у вигляді формальних скінченних автоматів, що дозволяє проводити часовий аналіз і верифікацію;

- прозорість інтеграції: модуль працює в рамках стандартних механізмів FreeRTOS (tickless idle, idle hook), не змінюючи модель планування задач;

- адаптивність: політика може змінюватися під час виконання відповідно до реальних умов роботи системи.

Таким чином, модуль керування адаптивною політикою енергоефективності є не просто елементом оптимізації споживання енергії, а регулятором, що об’єднує у собі принципи теорії керування, планування в реальному часі та аналізу поведінкових даних. Його розроблення стало відображенням загального підходу: поєднати інженерну формальність,

апаратні можливості та алгоритмічну гнучкість у єдину систему, що адаптується та саморегулюється.

Як результат, алгоритмічне керування живленням є частиною загальної архітектури планування завдань, інтегрованої з ядром FreeRTOS та програмною структурою тестового та системного ПЗ – F′. З огляду на це, розроблено програмний компонент PowerCtrlAgent, призначений для:

- централізованого керування політикою переходів між енергетичними режимами;
- збору та агрегації статистики енергоспоживання;
- надання телеметричної інформації та подій верхньорівневим компонентам системи та до наземної станції в межах загального потоку телеметричної інформації;
- забезпечення чіткого розділення відповідальностей між логікою прийняття рішень та інтерфейсами керування і спостереження.

4.3.2 Розроблення компонента F′ для реалізації методу адаптивного регулювання режимами енергозбереження МК БО

Компонент PowerControlAgent реалізовано як активний компонент фреймворку F′, що має власний потік виконання. PowerControlAgent відповідає за:

- прийом команд керування політикою енергозбереження (дозвіл/заборона переходів у режими сну, блокування глибоких режимів) (CMD, GroundSystem);
- передачу відповідних параметрів політики до нижньорівневого сервісу (декодування команд в політики);
- періодичне зчитування агрегованої статистики та подій з нижнього рівня;
- публікацію телеметрії (загальна зекономлена енергія, статистика по режимах, лічильники перебування) (TLM, GroundSystem);
- генерацію подій для системи логування (LogDataCollector).

Принциповим рішенням є те, що компонент не приймає рішень щодо вибору енергетичного режиму та не взаємодіє безпосередньо з апаратурою або ОСРЧ, а працює як з'єднувальний елемент між політикою керування та механізмами спостереження. Таке архітектурне рішення, відображене на рисунку 4.9, відповідає вимогам з дизайну, розроблення та тестування компонентів F′, що, у свою чергу, гарантує їх переносимість між різними платформами та різними версіями F′.

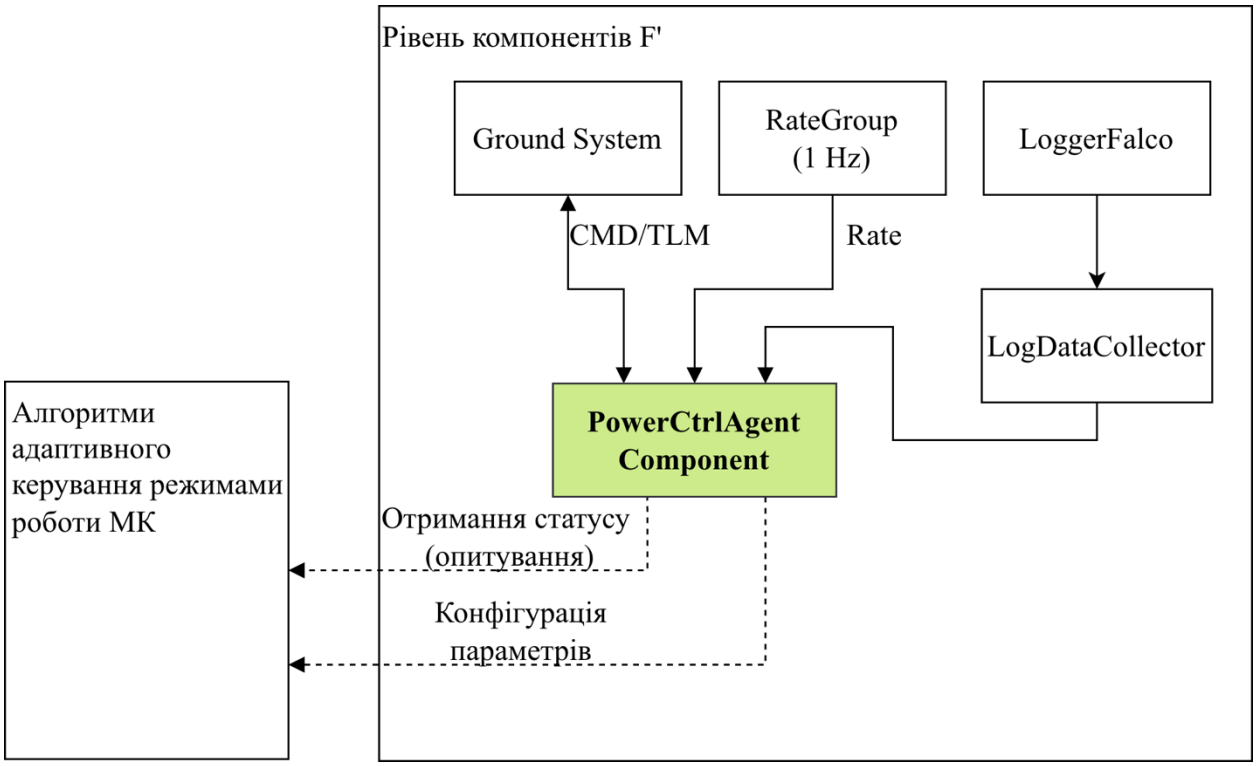

Рисунок 4.9 – Структурна схема взаємодії компонента F′ - PowerControlAgent, та алгоритмів модулю адаптивного керування

Складові діаграми класів компонента PowerCtrlAgent відображено на рисунку 4.10, та представлено наступними складовими:

- прапори політик сну - PolicyFlags;
- інтерфейси до телекоманд, телеметрії та логування через базовий клас PowerCtrlAgentComponentBase;
- поточний стан бортової «енергетики» БО – StatsSnapshot;
- інтерфейс до низькорівневих алгоритмів керування режимами роботи МК надається класом sleepManager.

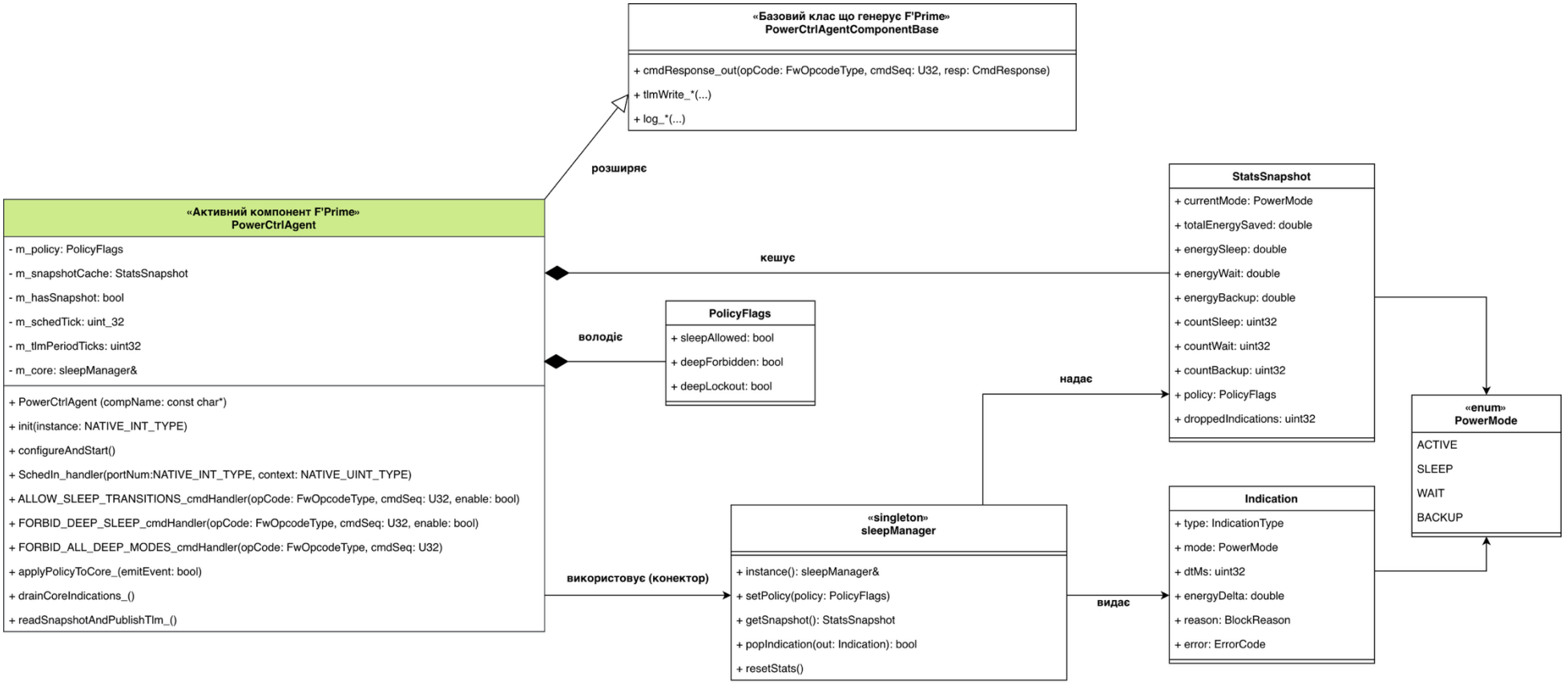

Рисунок 4.10 – Діаграма класів компонента F´ - PowerCtrlAgent

Клас **sleepManager**, що є інтерфейсом взаємодії класа-компонента F′ - PowerCtrlAgent з інструментами керування режимами роботи МК БО рівня ОСРЧ, розділено на три рівні:

- **агрегатор подій** (Event Aggregator) інкапсулює логіку збору статистичних даних системних подій в реальному часі. Реалізує «легкі» алгоритми обробки подій в реальному часі в первинному обробнику переривань (ISR) завдяки використанню атомарних лічильників та усереднення «ковзним вікном». Кількість інстанцій класу дорівнює потужності множині подій - E;

- **супервізор стабільності** (Stability Supervisor) реалізує скінчений автомат з пам’яттю. Відповідає за надійність та стабільність роботи системи, дозволяючи перехід до глибших режимів енергозбереження лише після позитивної історії стабільної поведінки системи;

- **менеджер сну** (Sleep Manager) виконує роль інтегруючого компонента. Виконує агрегацію метрик від усіх системних спостерігачів, опитує супервізор стабільності щодо дозволених режимів та виконує фінальний розрахунок часу сну. Такий розрахунок є результатом багатопараметричної оптимізації. Менеджер сну є точкою взаємодії побудованих алгоритмів та системного планувальника ОСРЧ, а, також, інтерфейсом взаємодії з компонентом рівня F′ - PowerCtrlAgent;

Оскільки в реальних МК-системах існує та використовується ціла низка апаратних переривань, використано патерн проєктування – «Спостерігач» (Observer) для усунення жорсткого зв’язку між обробниками таких переривань та логікою подальшого статистичного аналізу в контексті рівня – Event Aggregator. В патерн «Спостерігач» входять наступні складові:

- суб’єкт або видавець (subject);
- обробники апаратних переривань – ISR;
- спостерігач або підписник (observer);
- клас-агрегатор (Event Aggregator).

Для розділення оброблення критичних та некритичних подій в межах патерну проєктування Спостерігач, агрегатор відповідає за маршрутизацію подій. Таким чином, будучи єдиною точкою входу для ISR, агрегатор реалізує таблицю відповідності, що дозволяє класифікувати джерело подій і оновлювати відповідну статистику. Відповідна архітектура відображена на рисунку 4.11.

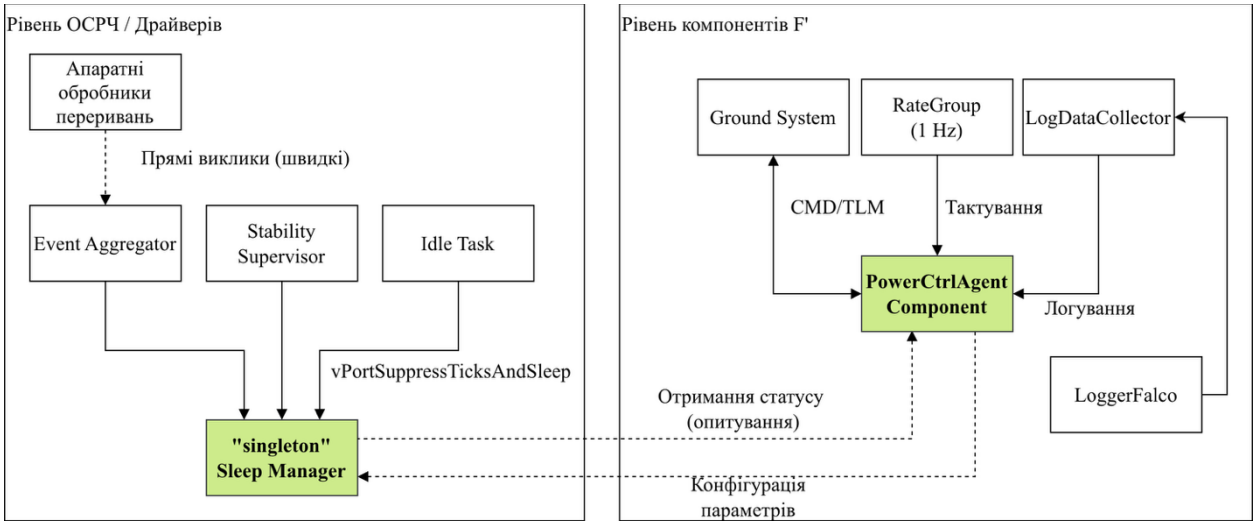

Рисунок 4.11 – Деталізована структурна схема взаємодії компонента F′ та алгоритмів модулю адаптивного керування режимами енергозбереження МК

Діаграму класів підсистеми адаптивного керування енергозбереженням наведено на рисунку 4.12, а діаграма станів автомату $\mathcal{A}$ – на рисунку 4.13.

Діаграма станів демонструє гістерезисну природу керування. Перехід праворуч (у глибші режими енергозберігання) є повільним та ітеративним, тоді як перехід ліворуч (у безпечні режими) є миттєвим, що відповідає вимогам систем реального часу.

Практична реалізація адаптивних алгоритмів керування режимами роботи МК реалізується у callback-функції ОСРЧ FreeRTOS – vApplicationIdleHook(), що викликається системним планувальником у випадку, коли більше немає завдань готових до виконання.

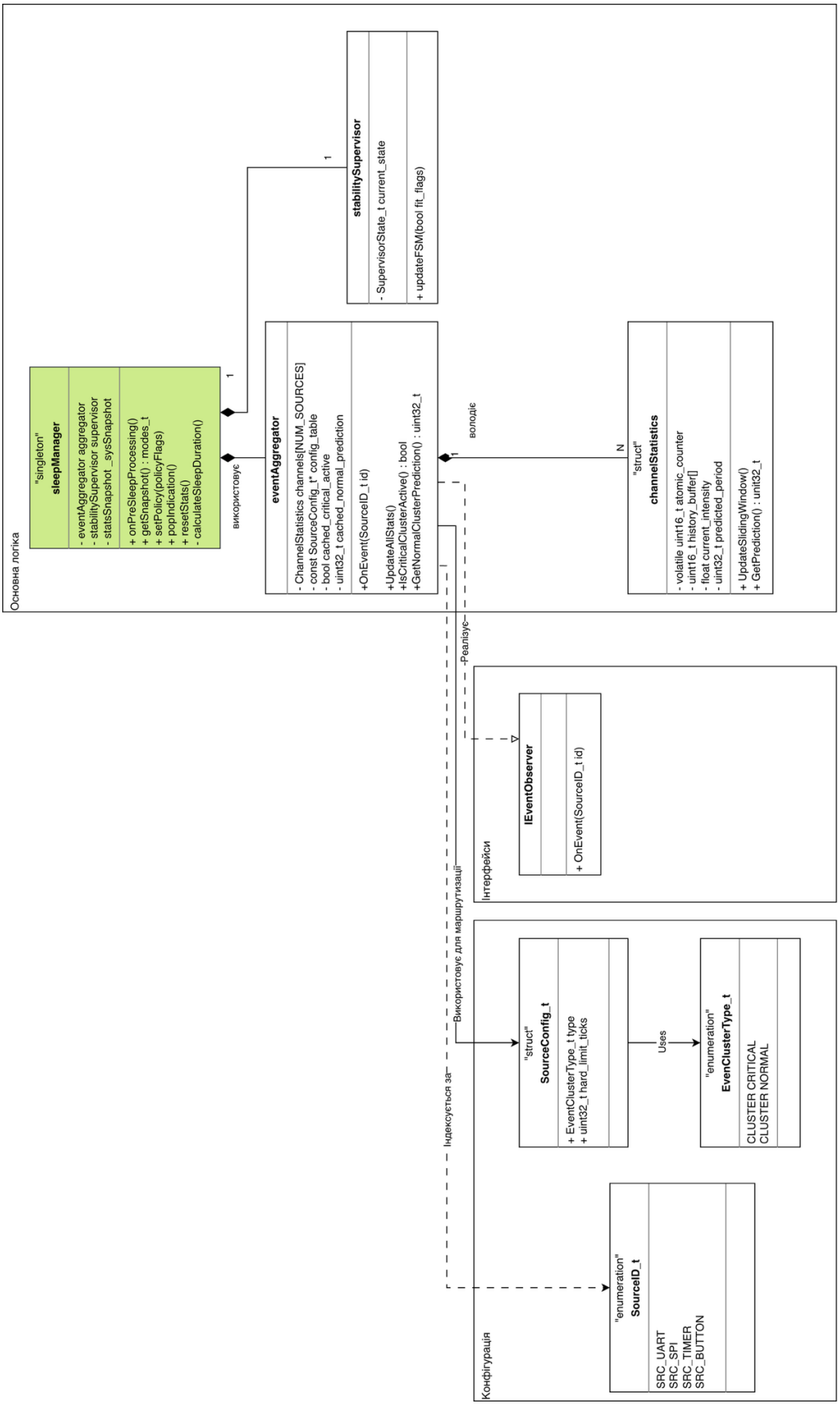

Рисунок 4.12 – Діаграма класів підсистеми адаптивного керування енергозбереженням (UML)

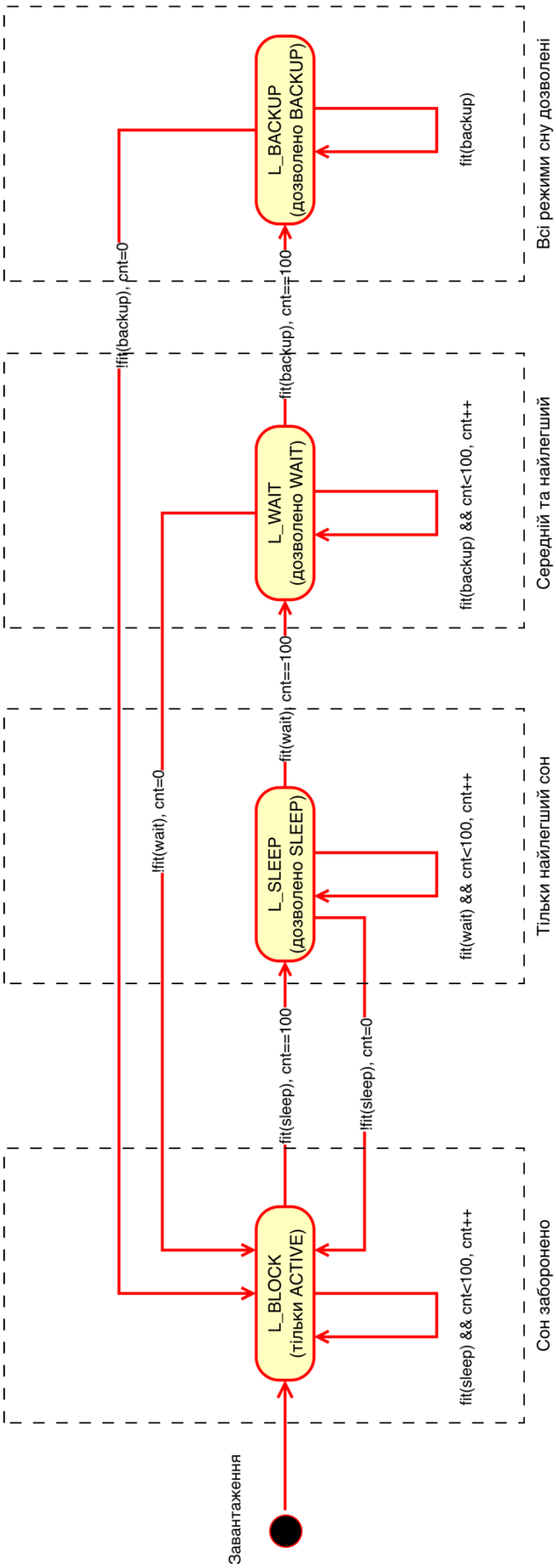

Рисунок 4.13 – Діаграма станів автомату $\mathcal{A}$ (Режими керування сном)

Для МК класу ARM Cortex-M у функції vApplicationIdleHook() потрібно перевести МК у один з режимів енергозбереження та викликати системну функцію wfi() – Wait For Interrupt, попередньо виконавши конфігурацію переривання «пробудження» або від зовнішнього, або внутрішнього джерела переривань - таймера, годинника реального часу, тощо.

Для активації виклику функції wfi() у ОСРЧ FreeRTOS потрібно виконати наступні кроки:

- у файлі конфігурації FreeRTOSConfig.h, встановити системний параметр #define configUSE_IDLE_HOOK в одиничне значення;

- реалізувати тіло функції у користувацькому коді. В нашому випадку це частина системного ПЗ загальної збірки БПЗ БО.

У даній функції забороняється використовувати блокуючі виклики функцій або системних примітивів синхронізації ОСРЧ FreeRTOS. Будь-яке порушення цих обмежень викликає довічне блокування функції vApplicationIdleHook() та/або непередбачувану поведінку планувальника ОСРЧ.

Діаграму послідовності роботи адаптивного модуля керування режимами роботи МК наведено на рисунку 4.14. Для читання цієї діаграми слід зазначити наступні ключові моменти:

- швидкі дії, що виконуються в обробнику переривання (ISR): Мінімальний час виконання, тільки інкремент лічильника через OnEvent;

- повільна періодична дія (Update Task): Періодичний перерахунок статистики та маршрутизація по кластерах подій - Критичні/Нормальні;

- повільна дія – прийняття рішення (PreSleep): Менеджер режимів читає готові прапорці, виконує машину станів не виконуючи важких розрахунків.

Діаграма активності (Activity Diagram) на рисунку 4.15 показує ієрархію прийняття рішень: спочатку безпека (Critical check), потім стабільність (Supervisor check), і тільки потім ефективність (Breakeven check).

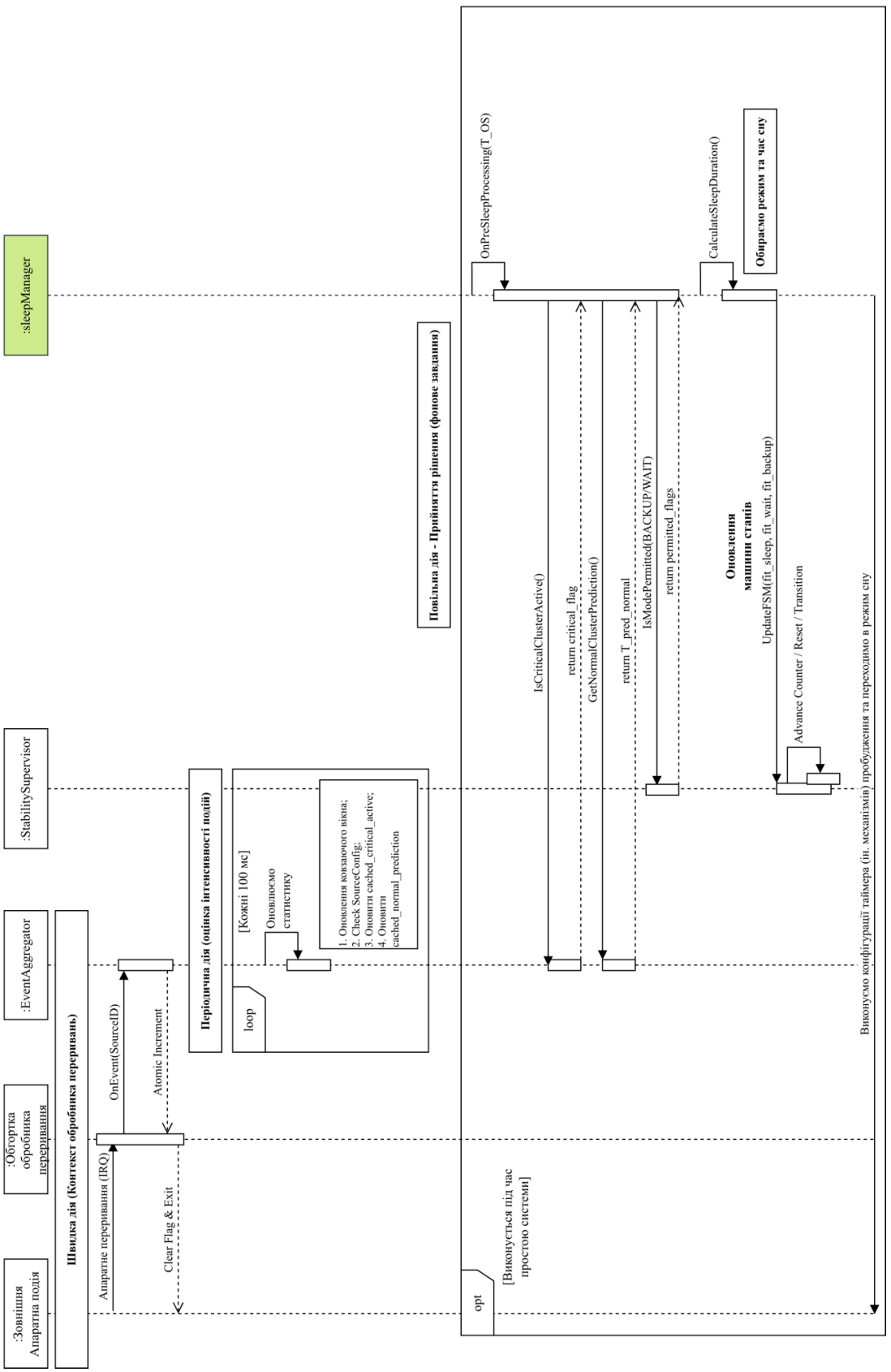

Рисунок 4.14 – Діаграма послідовності роботи адаптивного модуля регулювання режимами роботи МК

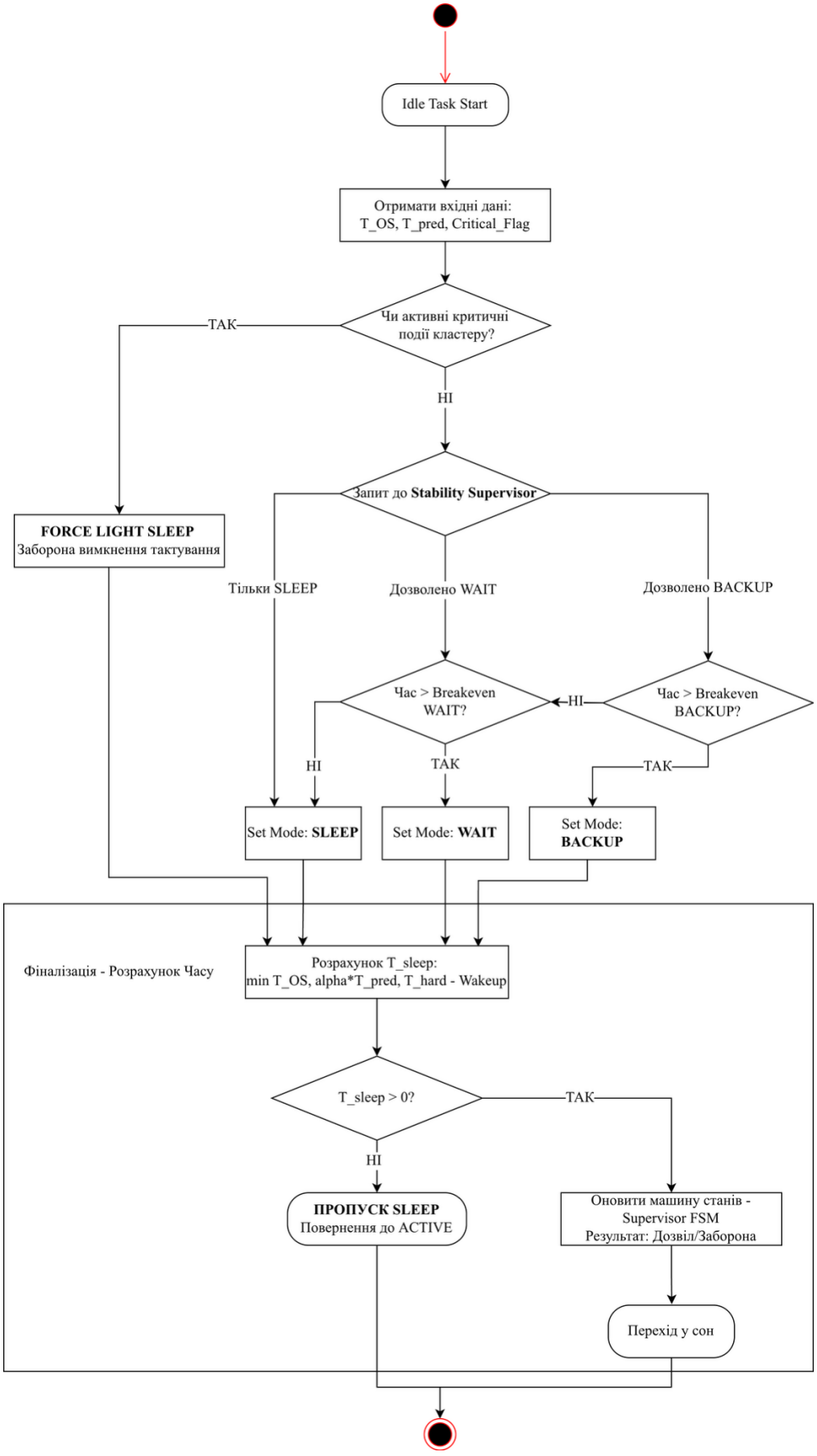

Рисунок 4.15 – Діаграма активності модуля регулювання режимів роботи МК БО

Логіку діаграми активності, що відображена на рисунку 4.15, можна пояснити наступним чином:

- в стані «заборони» за активності критичних джерел дозволено використання тільки режиму SLEEP;

- в стані «фільтрування», коли супервізор не дозволяє глибокий режим сну, виконати деградацію автомату до дозволеного стану;

- в стані «оптимізації» виконати перевірку енергетичної доцільності переходу.

Таким чином, алгоритм методу адаптивного регулювання режимів енергозбереження та супервізора представлено в вигляді діаграми активності, що дозволило досягти наступних результатів:

- математична складова (UpdateAllStats) є частиною прикладної логіки (цикл loop), коли всі обробники переривань (ISR) виконуються з мінімальним та детермінованим часом, що є ключовою вимогою в системах реального часу (див. рисунок 4.14);

- трьохрівнева ієрархія обмежень (див. рисунок 4.15) з розділенням пріоритетності обмежень на 3 рівні (безпека даних, стабільність системи, енергетична ефективність) гарантує політику економії енергії.

Завдання керування енергозабезпеченням та його будова до бортового обчислювача наносупутника вирішується на рівні операційної системи. Для застосунків, що розробляються на рівні F′ – підсистема керування енергозабезпеченням є «прозорим» системним додатком.

Архітектура - дворівнева:

- рівень виконання побудовано на SleepManager, EventAggregator та StabilitySupervisor. Ці системні функції розробляються за крос-платформним підходом на мові C++. Прикладна частина ПЗ на основі F′ – нічого не знає про ці функції та класи;

- рівень керування, діагностики та статистичних даних є частиною компонентів F′.

4.4 Експериментальне дослідження ефективності розробленого адаптивного алгоритму регулювання режимів МК БО

Метою експериментального дослідження є перевірка ефективності запропонованого алгоритму енергозбереження для реального БО, шляхом кількісної оцінки зменшення енергоспоживання при збереженні функціональної коректності системи.

4.4.1 Планування експерименту, підготовка БПЗ та обладнання

Для верифікації запропонованого адаптивного алгоритму керування режимами енергозбереження розроблено експериментальний лабораторний стенд (див. підрозділ 3.1.5, рисунки 3.4, 3.5).

Для забезпечення відтворюваності експерименту:

- використовувалася фіксована конфігурація апаратного та програмного забезпечення;
- частоти тактування обирались з множини в 30, 100, 150 МГц;
- параметри режимів живлення залишалися незмінними протягом усіх вимірювань;
- всі вимірювання виконувалися за однакових сценаріїв навантаження;
- під час вимірів, струми витоку МК та спорадичне живлення елементів БО через ніжки МК – не враховувались.

ТПЗ спроєктоване таким чином, щоб емулювати типове змішане навантаження на процесор, як-то обчислювальні задачі, обробка даних та реагування на асинхронні системні або зовнішні події. Набір завдань було сформовано відповідно до типових обчислювальних бортових завдань – перетворення Фур’є та розрахунок контрольної суми. Це дозволяє оцінити часові характеристики планувальника задач при конкурентному доступі до обчислювального ядра та шини даних.

ТПЗ відповідає наступним вимогам:

- ТПЗ має реалізовувати компонент F′ (tstFourierComp) – перетворення Фур'є, що моделює інтенсивне обчислювальне навантаження на МК БО, а саме на його арифметико-логічний пристрій (англ. Arithmetic Logical Unit, ALU) та пристрій роботи з числами з рухомою комою (англ. Floating Point Unit, FPU). Параметри компонента: період виконання – 1 с., кількість елементів масиву для перетворення Фур'є – 1024, тип даних – числа з рухомою комою типу float (IEEE 754);

- ТПЗ має реалізовувати компонент F′ (tstCRC32Comp) – розрахунок контрольної суми, що моделює більш легке, але часте навантаження на МК БО, характерне для задач контролю цілісності даних. Параметри компонента: період виконання – 1 с., кількість елементів масиву для розрахунку контрольної суми – 16384, тип даних – цілі 4-х байтові числа;

- ТПЗ має реалізовувати компонент F′ (tstGPIOAndLEDComp) – активація світлодіода за прибуттям апаратного переривання з ніжки МК (критична подія жорсткого реального часу). Компонент сконфігуровано таким чином, що він може виводити МК з режимів сну SLEEP та WAIT. Обробник переривання керує зовнішнім світлодіодом, який змінює частоту миготіння (низькі частоти) та яскравість (високі частоти), для візуалізації впливу частоти переривань;

- ТПЗ має надавати кількість тактів виконання кожного компонента з використанням атомарних лічильників;

- для розрахунку використаної та збереженої енергії, ТПЗ використовує експериментально визначені параметри електричного споживання для різних режимів роботи МК та різних частот його тактування.

При розробці ТПЗ використано загальний підхід з проєктування в парадигмі F′, що має назву підтопологія (Додаток В. Таблиця В.1). Технічно підтопологія описується окремим файлом, а в головній топології виконується її інстанціювання та підключення. Взаємозв'язок розроблених підтопологій наведено на рисунку В.1 (Додаток В).

У складі експериментальної конфігурації використано стандартні компоненти F′:

– компонент телеметрії (TlmChan) – збирає та серіалізує дані для відправки до наземної станції (лабораторного ПК);

– компонент команд та взаємодії з наземною станцією (CmdDisp) – приймає команди від наземної станції (лабораторного ПК) на запуск та зупинку ТПЗ;

– компонент системного тактування завдань (RateGroupDriver) - забезпечує періодичний виклик тестових та системних компонентів;

– інші системні компоненти та драйвери, що забезпечуються коректне функціонування БПЗ.

Наявність системних компонентів дозволяє наблизити експериментальні умови до реальних сценаріїв експлуатації БО у складі наносупутника під керуванням розробленого БПЗ.

Для збору експериментальних даних використовувалася вбудована телеметрія F′ - TlmChan. Для зменшення впливу системних компонентів F’ для телеметричної інформації було використано спрощений контекст обміну даними, що дозволяв виводити текстові дані у консоль лабораторного ПК під керуванням MS Windows, а не у симулятор наземної станції.

Під час експерименту реєструвалися такі параметри (відгуки експерименту):

– кількість циклів виконаних прикладних компонентів;

– загальна кількість обчислювальних циклів системи;

– кількість переходів у кожен режим енергоспоживання;

– сумарний час перебування системи в кожному режимі енергоспоживання;

– сумарна енергія заощаджена в кожному режимі енергоспоживання.

Для підтвердження ефективності в порівняльному експерименті було використано один тестовий БО з двома різними конфігураціями БПЗ.

1) БПЗ зі стандартними налаштуваннями ОСРЧ FreeRTOS щодо використання часу «простою» - idle: під час «простою» ОСРЧ виконує

безкінечний цикл очікування нового обчислювального такту від планувальника. Функція vApplicationIdleHook() – не реалізована.

2) БПЗ має модифікований системний рівень ОСРЧ FreeRTOS, що використовує адаптивне керування режимами роботи МК у відповідності до запропонованих алгоритмів та методів. Функція vApplicationIdleHook() перевантажена викликом розроблених алгоритмів, розрахунки часу перебування та лічильників переходів до певних режимів реалізовані у системній функціїї – vPortSuppressTicksAndSleep(). Режим роботи планувальника ОСРЧ – Tickless Mode.

Наступні чотири тести було заплановано до виконання.

Тест 1. Перевірка навантаження МК на різних тактових частотах при сталому навантаженні. Перехід до режимів зниженого енергоспоживання МК – заборонений. Для навантаження МК було обрано конфігурацію БПЗ з наступної кількості розрахункових завдань: tstFourierComp – 120 екземплярів, tstCRC32Comp – 191 екземпляр, що надало наступне системне навантаження (рисунок 4.16). Зовнішні переривання не застосовувались – вихід генератора сигналів – відключений.

Тест 2. Перевірка впливу частоти зовнішніх переривань на коефіцієнт утилізації МК. Перехід до режимів зниженого енергоспоживання МК – заборонений. Для перевірки реакції БПЗ БО на зовнішні переривання та їх вплив на коефіцієнт утилізації МК було обрано конфігурацію БПЗ з наступної кількості розрахункових завдань: tstFourierComp – 75 екземплярів, tstCRC32Comp – 35 екземплярів, тактова частота МК – 30 МГц, зовнішні переривання у діапазоні 0 – 135 кГц. Зовнішні переривання від генератора прямокутних сигналів. Для експерименту був розроблений обробник переривання, що має оптимізований розмір за кількістю інструкцій та відправляє системну подію, що призводить для додаткового навантаження на планувальник ОСРЧ.

Тест 3. Оцінка залежності коефіцієнта утилізації МК від частоти зовнішніх переривань на різних тактових частотах МК. Перехід до режимів зниженого енергоспоживання МК – заборонений.

Тест 4. Оцінка залежності коефіцієнта утилізації МК від обчислювальної завантаженості системи при фіксованій тактовій частоті 30 МГц. Зовнішні переривання відсутні.

Критеріями оцінки ефективності запропонованих алгоритмів є:

– зменшення часу перебування системи в режимі ACTIVE;

– позитивне значення заощадженої енергії E_{saved};

– відсутність втрати функціональності прикладних і системних компонентів.

4.4.2 Результати експерименту

У тестах 1 та 2 знайдено залежності коефіцієнта утилізації МК від тактової частоти при фіксованому навантаженні (рисунок 4.16), та залежність коефіцієнта утилізації МК від частоти зовнішніх переривань при фіксованій тактовій частоті та обчислювальному навантаженні (рисунок 4.17).

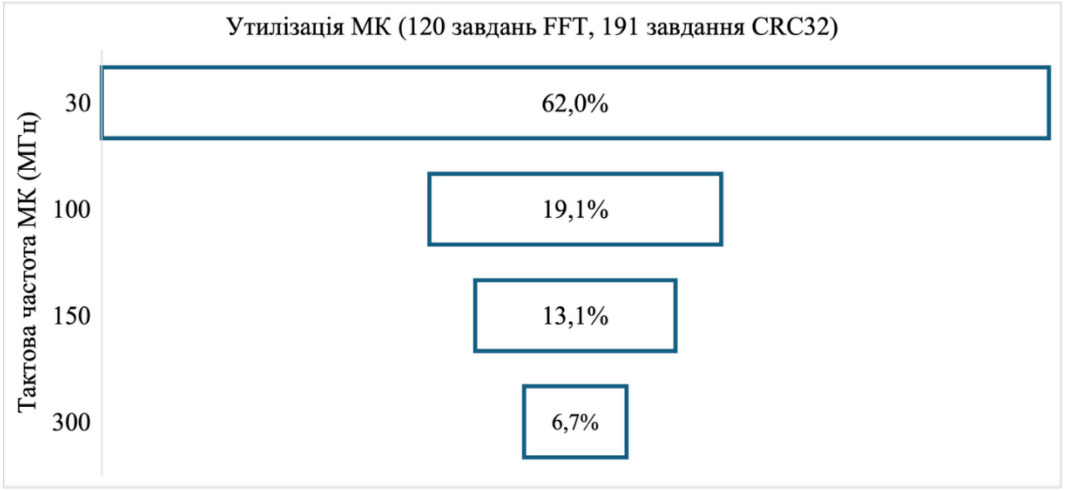

Рисунок 4.16 – Залежність коефіцієнта утилізації МК від тактової частоти.

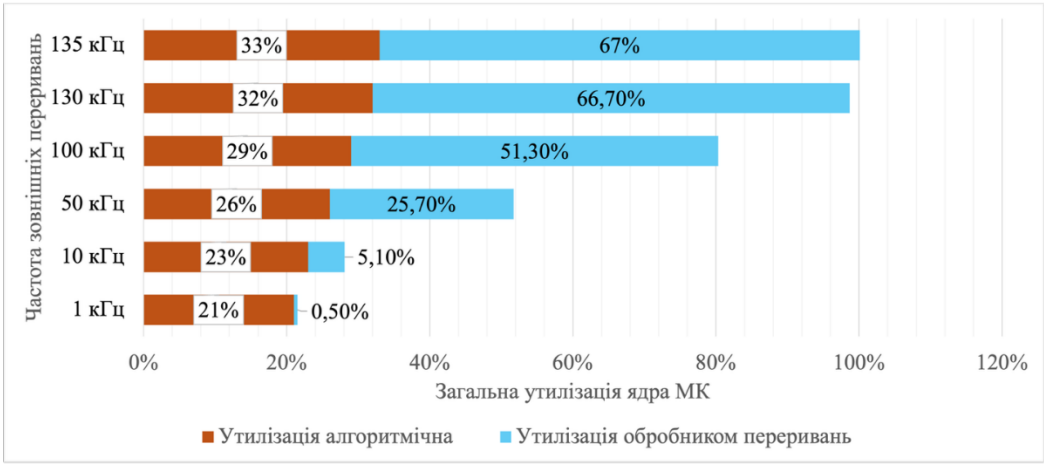

Рисунок 4.17 – Залежність коефіцієнта утилізації МК від частоти зовнішніх переривань при фіксованій тактовій частоті та обчислювальному навантаженні

Результати тесту 3 (Рисунок 4.18) свідчать, що:

– частота зовнішніх переривань в 100 кГц є фактичною стелею утилізації на 30 МГц, наявність таких частих переривань суттєво впливає на коефіцієнт утилізації МК;

– на більших тактових частотах, критичне навантаження на 30 МГц, стає зовсім несуттєвим, вже з 100 МГц коефіцієнт утилізації (21,7%) є незначним, що робить доцільним застосування адаптивних алгоритмів керування режимами роботи МК – час простою МК майже 80%.

В тесті 4 використано п’ять сценаріїв навантаження (Таблиця 4.2).

Таблиця 4.2 Опис сценаріїв навантаження в тесті 4

Сценарій навантаження	Кількість завдань FFT1024	Кількість завдань CRC32-16K	Коефіцієнт утилізації (%)
1	75	35	21
2	130	65	37
3	120	191	62
4	242	145	76
5	218	180	85

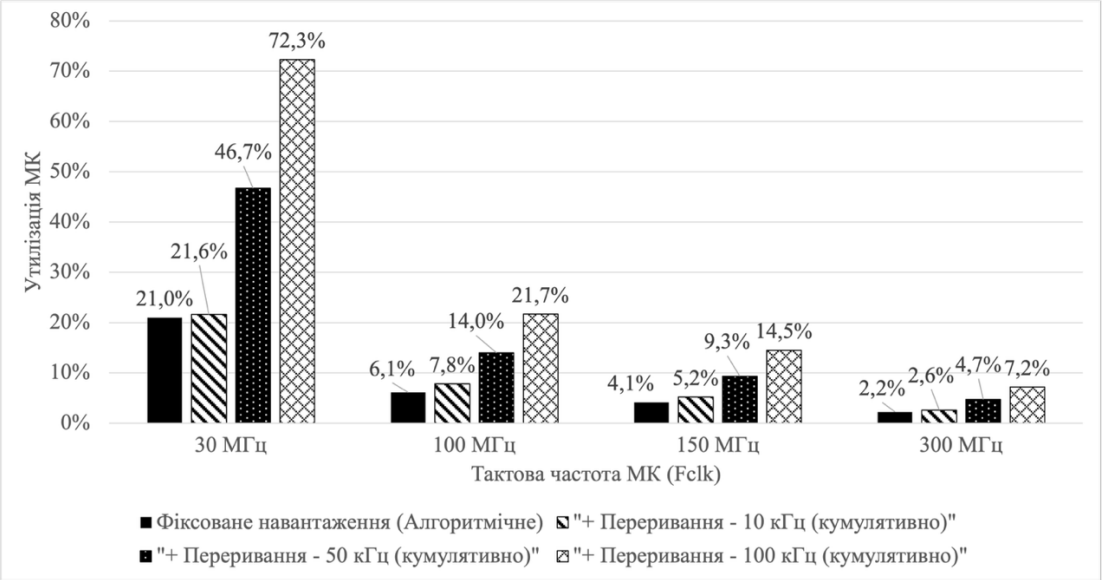

Рисунок 4.18 – Залежність коефіцієнта утилізації ядра МК від тактової частоти та частоти зовнішніх переривань (обчислювальне навантаження – фіксоване)

Отримано наступні результати роботи адаптивних алгоритмів керування режимами роботи МК (Таблиці 4.3-4.4, Рисунок 4.19):

Таблиця 4.3 Розподіл кількості режимів роботи МК під час 2 годинного лабораторного експерименту

Сценарій навантаження / коефіцієнт утилізації	Загальна кількість циклів – 1 сек.	Кількість режимів ACTIVE	Кількість режимів SLEEP	Кількість режимів WAIT	Кількість режимів BACKUP
1 / 21%	7474	101	101	101	7171
2 / 37%	7523	101	101	7321	0
3 / 62%	7389	101	2836	4452	0
4 / 76%	7443	101	7342	0	0
5 / 85%	7601	7601	0	0	0

Експеримент підтверджує ефективність спроєктованих алгоритмів, які в залежності від завантаженості дозволяють різні режими енергозбереження. Адаптивний метод дозволяє зберегти до енергозбереження до 77 % енергії при низькому навантаженні та майже 10% при великому.

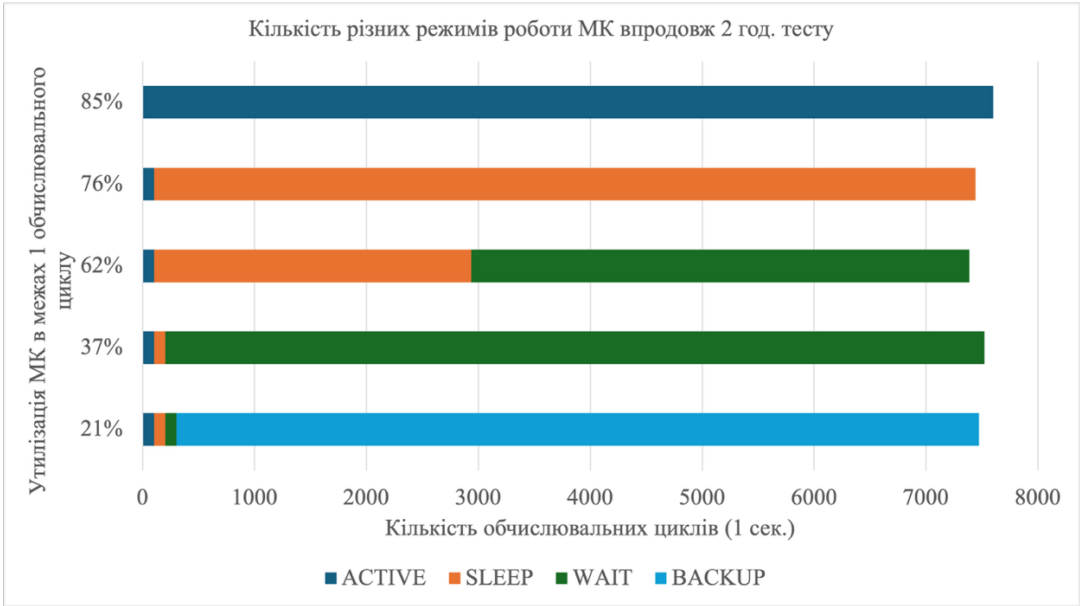

Рисунок 4.19 – Розподіл режимів роботи МК під час експерименту з різним навантаженням.

Таблиця 4.4 Кількість збереженої енергії для сценаріїв навантаження

Сценарій навантаження / коефіцієнт утилізації	Повна споживана енергія – тільки ACTIVE, Дж	Сумарна витрачена енергія (всі режими), Дж	Збережена енергія, Дж	Відносний відсоток заощадженої енергії, %
1 / 21%	591,9	137,1	454,8	77
2 / 37%	595,8	260,7	335,1	56
3 / 62%	585,2	430,3	154,9	26
4 / 76%	589,5	535,9	53,5	9
5 / 85%	601,9	601,9	0	0

Отримані експериментальні дані дозволяють зробити кількісні висновки щодо ефективності запропонованого алгоритму енергозбереження та підтверджують його здатність адаптивно змінювати режими роботи мікроконтролера залежно від поточного навантаження системи.

Висновку до розділу 4

У четвертому розділі дисертаційної роботи вирішено прикладну науково-технічну задачу розроблення, інтеграції та експериментальної верифікації адаптивних алгоритмів керування енергоспоживанням і реактивністю бортового обчислювача наносупутника. Показано, що ефективне керування енергетичними режимами МК неможливе без урахування архітектури бортового програмного забезпечення, механізмів ОСРЧ та вимог реального часу.

Сформульовано узагальнені вимоги до БПЗ БО наносупутників класу CubeSat, узгоджені зі стандартами ECSS та сучасною практикою розроблення космічного ПЗ. Обґрунтовано вибір компонентно-орієнтованої архітектури на основі FreeRTOS і фреймворку F´Prime, а, також, запропоновано чотиришарову модель інтеграції HAL–FreeRTOS–OSAL–F´, яка забезпечує модульність, переносимість, детермінованість виконання та накопичення «flight heritage».

Ключовим результатом розділу є розроблення компонента F´ PowerCtrlAgent та пов’язаного з ним системного модуля адаптивного керування режимами енергозбереження МК. Запропонований підхід ґрунтується на ієрархічній, детермінованій та гістерезисній моделі прийняття рішень, що поєднує вимоги безпеки, стабільності та енергетичної ефективності. Реалізація алгоритмів у стандартних механізмах FreeRTOS (idle hook, tickless mode) дозволила інтегрувати енергокерування без зміни моделі планування задач і без впливу на прикладні компоненти.

Експериментальні дослідження на реальному бортовому обчислювачі підтвердили коректність і ефективність запропонованого методу. Показано, що адаптивне регулювання режимів енергозбереження, дозволяє зменшити час перебування МК у режимі ACTIVE та досягти економії енергії до 77 % за низького навантаження і близько 10 % за високого, без втрати функціональної коректності та реактивності системи. Оскільки БО наносупутників переважно

працюють з середнім, або низьким навантаженням, прогнозований відсоток енергії, що може бути заощаджена – високий.

Отримані результати доводять практичну доцільність застосування розроблених алгоритмів у бортових обчислювачах наносупутників та створюють основу для їх подальшого розвитку й узагальнення.

Таким чином, в розділі вирішені сьоме та восьме завдання дисертації. Наукові результати, отримані в четвертому розділі, представлено в наукових публікаціях [169, 170].

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-прикладну задачу зменшення енергетичних витрат бортових обчислювачів наносупутників CubeSat за обмежень реального часу та забезпечення реактивності бортового програмного забезпечення на основі розроблення адаптивних методів, алгоритмів і програмних засобів балансування енергоспоживання та обчислювальної реактивності.

Наступні висновки сформульовано за результатами вирішення поставлених завдань.

1. В результаті аналізу умов експлуатації бортових обчислювачів наносупутників CubeSat встановлено, що обмежений енергетичний бюджет у поєднанні з вимогами реального часу є визначальним фактором, який суттєво впливає на надійність та успішність місій малих космічних апаратів.

2. Систематизований огляд наукових публікацій, присвячених апаратно-програмним рішенням бортових обчислювачів наносупутників CubeSat та вимог до їх апаратного та програмного забезпечення, показав, що існуючі методи енергозбереження не враховують реактивність програмного забезпечення як формалізований показник, що обмежує можливість їх застосування в критичних до часу бортових системах.

3. Розроблена комплексна математична модель для розрахунку енергетичних витрат бортового обчислювача, обчислювальної продуктивності та реактивності бортового програмного забезпечення на відміну від відомих раніше рішень забезпечує узгоджений опис зазначених характеристик, дозволяє враховувати режими роботи обчислювача та нестаціонарність навантаження в умовах реального часу.

4. Розроблений метод ідентифікації параметрів комплексної математичної моделі на основі рекурсивного методу найменших квадратів забезпечує онлайн-ідентифікацію параметрів за умов обмежених

обчислювальних ресурсів та може бути безпосередньо використаний при розробці програмно-керованого, адаптивного регулювання.

5. Ідентифікація параметрів комплексної математичної моделі виконана з використанням результатів фізичного експерименту. Перевірка адекватності моделі підтвердила її точність, достатню для практичних завдань.

6. Розроблений метод адаптивного регулювання режимів роботи бортових обчислювачів наносупутників CubeSat базується на концепції послаблених систем жорсткого реального часу та забезпечує динамічне узгодження вимог до реактивності програмного забезпечення з енергетичними обмеженнями, що дозволяє зменшити енергоспоживання без порушення допустимих часових характеристик виконання задач.

7. Розроблені алгоритми та програмні засоби бортового програмного забезпечення адаптивного керування режимами роботи бортового обчислювача реалізовано у вигляді компонентів фреймворку NASA F′ Prime для операційної системи реального часу FreeRTOS.

8. Проведені експериментальні дослідження підтвердили адекватність розроблених математичних моделей та ефективність запропонованого адаптивного алгоритму, продемонстрували можливість суттєвого зменшення енергетичних витрат бортового обчислювача за умови збереження заданого рівня реактивності бортового програмного забезпечення.

Практичне значення отриманих результатів полягає у можливості їх використання при проєктуванні та розробленні бортового програмного забезпечення наносупутників CubeSat, безпілотних літальних апаратів та інших вбудованих систем з обмеженим енергетичним бюджетом і вимогами реального часу. Результати дисертаційної роботи доведено до рівня інженерних методик і програмних засобів та впроваджено у навчальному процесі, науково-дослідних і прикладних проєктах.

Таким чином, мету дисертаційної роботи досягнуто, усі поставлені завдання вирішено, а отримані результати є науково обґрунтованими, завершеними та такими, що відповідають вимогам до дисертацій на здобуття

наукового ступеня доктора філософії за спеціальністю 121 «Інженерія програмного забезпечення».

Подальші дослідження доцільно зосередити на:

- узагальненні комплексної математичної моделі на багатоядерні та гетерогенні архітектури бортових обчислювачів (MPSoC, FPGA+CPU, GPU) з урахуванням конкуренції за ресурси, можливості міграції задач, теплових обмежень, енергоспоживання та реактивності;

- розвитку адаптивних методів та алгоритмів із використанням машинного навчання та прогнозування для оцінювання інтенсивності подій, навантаження та деградації енергетичних характеристик;

- узагальненні методів адаптивного балансування для розподілених бортових систем із кількома вузлами та внутрішніми мережами (CAN, SpaceWire, Ethernet);

- валідації методів у льотних (БПЛА) та орбітальних (CubeSat) експериментах для підтвердження ефективності в реальних умовах;

- подальшій інтеграції з космічними та БПЛА фреймворками NASA cFS, ROS 2, архітектурами TASTE (ЄКА), RODOS (Німеччина), KOSMOS (Франція), GFSW (Іспанія), ArduPilot, BetaFlight та PX4, для підвищення переносимості та прикладної застосовності;

- формалізації показника «реактивності» як інженерної метрики та елементу стандартів проєктування, MBSE-процесів, верифікації та валідації бортового ПЗ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Paek S. W., de Weck O. L., Hoffman J. A., Linares R. Definition of New Space—Expert survey results and key technology trends // IEEE Journal on Miniaturization for Air and Space Systems. – 2020. – Vol. 1, № 3. – P. 180–190. – DOI: 10.1109/JMASS.2020.3045851.

2. Weinzierl M. Space, the final economic frontier // Journal of Economic Perspectives. – 2018. – Vol. 32, № 2. – P. 173–192. – DOI: 10.1257/jep.32.2.173.

3. Boshuizen C., Mason J., Klupar P., Spanhake S. Results from the Planet Labs Flock constellation // Proc. 28th Annual AIAA/USU Conf. on Small Satellites. – Logan, UT, USA, 2014. – Paper SSC14-I-1. – URL: https://digitalcommons.usu.edu/smallsat/2014/YearReview/1/ (дата звернення: 20.11.2025).

4. Heidt H., Puig-Suari J., Moore A. S., Nakasuka S., Twiggs R. J. CubeSat: A new generation of microsatellites for education and industry low cost space experimentation // Proc. 14th Annual AIAA/USU Conf. on Small Satellites. – Logan, UT, USA, 2000. – Paper SSC00-V-5. – URL: https://digitalcommons.usu.edu/smallsat/2000/all2000/32/ (дата звернення: 20.11.2025).

5. Kulu E. Nanosats Database [Електронний ресурс]. – URL: https://www.nanosats.eu (дата звернення: 20.11.2025).

6. Kulu E. CubeSats & nanosatellites — 2024 statistics, forecast and reliability // Proc. 75th Int. Astronautical Congress (IAC 2024). – Milan, Italy, 2024. – Paper IAC-24-B4.6A.13. – URL: https://iafastro.directory/iac/paper/id/83232/summary/ (дата звернення: 20.11.2025).

7. BryceTech. Smallsats by the numbers 2025 [Електронний ресурс]. – Alexandria, VA : BryceTech, 2025. – URL: https://brycetech.com (дата звернення: 22.11.2025).

8. Paiva S. B., Morais A. M., Julia R. M. The coming of age of space satellite industry: transitioning from a growth to a maturity life cycle phase // Journal of Aerospace Technology and Management. – 2023. – Vol. 15. – DOI: 10.1590/jatm.v15.1296.

9. Alkan B. et al. Improving efficiency in CubeSat mass production: a modular and standardized approach // Acta Astronautica. – 2025. – Vol. 232. – DOI: 10.1016/j.actaastro.2024.11.045.

10. Walsh A. et al. Flight software development for the EIRSAT-1 mission // Journal of Astronomical Telescopes, Instruments, and Systems. – 2023. – Vol. 9, № 1. – Art. 017002. – DOI: 10.1117/1.JATIS.9.1.017002.

11. CubeSat market size, share & trends analysis report [Електронний ресурс]. – Straits Research, 2024. – URL: https://straitsresearch.com/report/cubesat-market (дата звернення: 22.11.2025).

12. Nanosatellite and microsatellite market size & share analysis – growth trends & forecasts (2024–2029) [Електронний ресурс]. – Mordor Intelligence, 2024. – URL: https://www.mordorintelligence.com/industry-reports/nanosatellite-and-microsatellite-market (дата звернення: 22.11.2025).

13. Turning Star Wars into reality with space-based missile defense and information command-and-control systems / Aerospace America [Електронний ресурс]. – URL: https://aerospaceamerica.aiaa.org/year-in-review/turning-star-wars-into-reality-with-space-based-missile-defense-and-information-command-and-control-systems/ (дата звернення: 22.12.2025).

14. The CubeSat Program. CubeSat Design Specification (CDS). Rev. 14.1 [Електронний ресурс]. – San Luis Obispo : California Polytechnic State University, 2022. – URL: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/62193b7fc9e72e0053f00910/1645820809779/CDS+REV14_1+2022-02-09.pdf (дата звернення: 22.11.2025).

15. State of the art of small spacecraft technology [Електронний ресурс] / NASA Small Spacecraft Systems Virtual Institute. – Washington, D.C. : NASA Ames Research Center, 2023. – URL: https://s3vi.ndc.nasa.gov/soa/2023/03-structures-materials-and-mechanisms (дата звернення: 22.11.2025).

16. QBUA01 (PolyITAN-HP-30) spacecraft – Nanosats Database [Електронний ресурс]. – URL: https://www.nanosats.eu/sat/qbua01 (дата звернення: 20.12.2025).

17. PC/104 Specification. Version 2.6 [Електронний ресурс]. – PC/104 Embedded Consortium, 2008. – URL: https://pc104.org/wp-content/uploads/2015/02/PC104_Spec_v2_6.pdf (дата звернення: 22.11.2025).

18. NASA Goddard Space Flight Center. General Environmental Verification Standard (GEVS) for GSFC flight programs and projects : GSFC-STD-7000B [Електронний ресурс]. – Greenbelt : NASA GSFC, 2021. – URL: https://standards.nasa.gov/standard/GSFC/GSFC-STD-7000 (дата звернення: 22.11.2025).

19. ISO 19683:2017. Space systems – design qualification and acceptance tests of small spacecraft and units [Електронний ресурс]. – Geneva : ISO, 2017. – URL: https://www.iso.org/standard/65767.html (дата звернення: 22.11.2025).

20. CCSDS 133.0-B-2. Space Packet Protocol : recommended standard [Електронний ресурс]. – Issue 2. – Washington, DC : CCSDS, 2020. – URL: https://public.ccsds.org/Pubs/133x0b2e1.pdf (дата звернення: 22.11.2025).

21. ISO 22646:2005. Space data and information transfer systems — space packet protocol [Електронний ресурс]. – Geneva : ISO, 2005. – URL: https://www.iso.org/standard/36381.html (дата звернення: 22.11.2025).

22. Overview of space communications protocols : report concerning space data system standards : CCSDS 130.0-G-3 [Електронний ресурс]. – Issue 3. – Washington, DC : CCSDS, 2014. – July. – URL: https://public.ccsds.org/Pubs/130x0g3.pdf (дата звернення: 22.11.2025).

23. Merri M., Russu A. The CCSDS and ECSS standards for space operations // *Proc. SpaceOps 2010 Conf.* (Huntsville, AL, USA, Apr. 25–30, 2010). – Huntsville : AIAA, 2010. – DOI: 10.2514/6.2010-2128.

24. AWS Ground Station. Fully managed ground station as a service [Електронний ресурс]. – Amazon Web Services, Inc., 2024. – URL: https://aws.amazon.com/ground-station/ (дата звернення: 22.11.2025).

25. Turkin I., Liubimov O., Volobuieva L., Valkovyi V. Hardware and software of CubeSat nanosatellites onboard computers: a systematized literature review // *Aerospace Technic and Technology*. – 2025. – № 3. – P. 116–137. – DOI: 10.32620/aktt.2025.3.11.

26. Cratere L., Gagliardi G. A., Sanca F., Golmar F., Dell'Olio F. On-board computer for CubeSats: state-of-the-art and future trends // *IEEE Access*. – 2024. – Vol. 12. – P. 99537–99569. – DOI: 10.1109/ACCESS.2024.3428388.

27. Bigagli L., Fleeter R. SBUDNIC: lessons learned in implementing a CubeSat mission // *EGU General Assembly 2023 Proceedings*. – 2023. – Art. EGU23-16127. – DOI: 10.5194/egusphere-egu23-16127.

28. Hart M., Kang S. H. Small spacecraft technology: state of the art [Електронний ресурс]. – Moffett Field, CA : NASA Ames Research Center, 2022. – Report No. NASA/TP–20220018183. – URL: https://ntrs.nasa.gov/citations/20220018183 (дата звернення: 20.11.2025).

29. Martinez S. et al. Risk assessment mitigation: reliability impacts of single event latch-up (SEL) in commercial-off-the-shelf (COTS) [Електронний ресурс]. – Greenbelt, MD : NASA Goddard Space Flight Center, 2025. – Document ID: 20250007864. – URL: https://ntrs.nasa.gov/citations/20250007864 (дата звернення: 20.11.2025).

30. Sajjad W., Shafique A., Mahmood R. Designing reliable, low-power, and performance-efficient onboard computer architecture for CubeSats // *IEEE Journal on Miniaturization for Air and Space Systems*. – 2024. – Vol. 5, № 2. – P. 59–72. – DOI: 10.1109/JMASS.2023.3342208.

31. Di Roberto R., Brandolini E., Sparvieri G., Graziani F. Best practices on adopting open-source and commercial low-cost devices in small satellite missions // *Acta Astronautica*. – 2023. – Vol. 211. – P. 37–48. – DOI: 10.1016/j.actaastro.2023.06.001.

32. CubeSat Market : the first marketplace for CubeSats [Електронний ресурс]. – URL: https://www.cubesat.market/ (дата звернення: 20.11.2025).

33. Satsearch : the global marketplace for the space industry [Електронний ресурс]. – URL: https://satsearch.co/ (дата звернення: 20.11.2025).

34. CubeSat components / SatCatalog [Електронний ресурс]. – URL: https://www.satcatalog.com/components/?form-factor=CubeSat (дата звернення: 20.11.2025).

35. CubeSatShop.com : one stop shop for CubeSat missions [Електронний ресурс]. – URL: https://www.cubesatshop.com/ (дата звернення: 20.11.2025).

36. CubeSat industry participants / CubeSat.org [Електронний ресурс]. – URL: https://www.cubesat.org/suppliers (дата звернення: 20.11.2025).

37. Liubimov O., Liubimov M. Use of open-source COTS/MOTS hardware and software platforms for the build up of the CubeSat nanosatellites // *Journal of Rocket-Space Technology*. – 2023. – Vol. 31, № 4. – P. 138–147.

38. McComas D. et al. NASA's core flight system: a reusable, open source flight software framework // *IEEE Software*. – 2016. – Vol. 33, № 6. – P. 6–9. – DOI: 10.1109/MS.2016.138.

39. Chaumette E. et al. MBSE challenges in the concurrent preliminary design of CubeSats: Nanospace study // *Proc. IEEE Aerospace Conf. (AERO 2022)*. – Big Sky, MT, USA, 2022. – P. 1–11. – DOI: 10.1109/AERO53065.2022.9843393.

40. Kaslow D. et al. Developing a CubeSat model-based system engineering (MBSE) reference model – interim status // *Proc. IEEE Aerospace Conf.* – Big Sky, MT, USA, 2015. – P. 1–14. – DOI: 10.1109/AERO.2015.7118965.

41. Sæther E. et al. A comparative survey on flight software frameworks for "New Space" nanosatellite missions // *Proc. 35th Annual Small Satellite Conf.* – Logan, UT, USA, 2021. – Paper SSC21-XII-02. – URL: https://digitalcommons.usu.edu/smallsat/2021/all2021/235/ (дата звернення: 20.11.2025).

42. Marcelino G. M., Mattos B. S. FloripaSat-2: an open-source platform for CubeSats // *IEEE Embedded Systems Letters*. – 2024. – Vol. 16, № 1. – P. 77–80. – DOI: 10.1109/LES.2023.3260372.

43. Vaudolon J. et al. Nanostar project: student challenges & tools – developing a collaborative open-source platform for nanosatellite education and capacity building // *Proc. 73rd Int. Astronautical Congress (IAC)*. – Paris, France, 2022. – Paper IAC-22,E1,2,5,x69488. – URL: https://iafastro.directory/iac/paper/id/69488/summary/ (дата звернення: 20.11.2025).

44. Culpepper M. KubOS: an open source operating system for small satellites // *Proc. 31st Annual AIAA/USU Conf. on Small Satellites* (Logan, UT, USA, Aug. 5–10, 2017). – Logan : Utah State University, 2017. – Paper SSC17-X-02. – URL: https://digitalcommons.usu.edu/smallsat/2017/all2017/129/ (дата звернення: 22.11.2025).

45. Morris J., Grubb M., Lucas J., O'Connor M. NASA Operational Simulator for Small Satellites (NOS3) // *Proc. 32nd Annual AIAA/USU Conf. on Small Satellites*. – Logan, UT, USA, 2018. – Paper SSC18-XII-08. – URL: https://digitalcommons.usu.edu/smallsat/2018/all2018/283/ (дата звернення: 22.11.2025).

46. Holliday M., Iacopino C. PyCubed: an open-source, radiation-tested CubeSat platform programmable in Python // *Proc. 33rd Annual AIAA/USU Conf.*

on Small Satellites (Logan, UT, USA, Aug. 3–8, 2019). – Logan : Utah State University, 2019. – Paper SSC19-XII-02. – URL: https://digitalcommons.usu.edu/smallsat/2019/all2019/168/ (дата звернення: 22.11.2025).

47. Sano T. et al. Development of image processing subsystem for 6U CubeSat VERTECS using commercial off-the-shelf components // *Proc. 33rd Int. Symp. on Space Technology and Science (ISTS)* (Beppu, Oita, Japan, Feb. 2022). – 2022. – Paper 2022-f-24.

48. Bocchino R. L. et al. F Prime: an open-source framework for small-scale flight software systems // *Proc. 32nd Annual AIAA/USU Conf. on Small Satellites*. – Logan, UT, USA, 2018. – Paper SSC18-XII-04. – URL: https://digitalcommons.usu.edu/smallsat/2018/all2018/328/ (дата звернення: 22.11.2025).

49. NASA. F′ (F Prime) flight software framework : GitHub repository [Електронний ресурс]. – 2025. – URL: https://github.com/nasa/fprime (дата звернення: 22.11.2025).

50. Johnson A., Cloutier K., Fogle C., Razzaghi M. Application of F′ software framework to space flight systems // *Proc. IEEE Aerospace Conf. (AERO 2019)*. – 2019. – DOI: 10.1109/AERO.2019.8742188.

51. Fogle C. A component-based software framework for mission-critical embedded systems: F′ Prime // *Proc. IEEE Aerospace Conf. (AERO 2017)*. – 2017. – DOI: 10.1109/AERO.2017.7943960.

52. Fogle C., Cloutier K., Dvorak E., Johnson A. F′: a flight software and ground system framework // *Proc. IEEE Aerospace Conf. (AERO 2018)*. – 2018. – DOI: 10.1109/AERO.2018.8396556.

53. Eshaq M., Zitouni M. S., Atalla S., Al-Mansoori S., Macdonald M. CubeSat flight software: insights and a case study // *Journal of Spacecraft and Rockets*. – 2025. – DOI: 10.2514/1.A35882.

54. Shishko R., Davis C., Wilmot C. The Core Flight System: enabling software reuse across NASA missions // *Proc. IEEE Aerospace Conf. (AERO 2013)*. – 2013. – DOI: 10.1109/AERO.2013.6496924.

55. Shishko R., Wilmot C., Davis C., Rash R. The Core Flight System (CFS) for satellite on-board software // *Proc. AIAA SPACE 2015 Conf.* – 2015. – DOI: 10.2514/6.2015-4424.

56. Shishko R., Davis C., Rash R. Integrating external software into the Core Flight System // *Proc. IEEE Aerospace Conf. (AERO 2016)*. – 2016. – DOI: 10.1109/AERO.2016.7500742.

57. Goseva-Popstojanova K., Hood D., Schumann J., Nkwocha N. The anatomy of software changes and bugs in autonomous operating system // *Proc. IEEE 47th Annual Computers, Software, and Applications Conf. (COMPSAC 2023)*. – 2023. – P. 28–38. – DOI: 10.1109/COMPSAC57700.2023.00004.

58. Miranda D. J. F., Ferreira M., Kucinskis F. N., McComas D. A comparative survey on flight software frameworks for “New Space” nanosatellite missions [Електронний ресурс]. – ResearchGate, 2019. – DOI: 10.13140/RG.2.2.14811.00803 (дата звернення: 22.11.2025).

59. Hach S. Specification of distributed computing for small satellite control : doctoral dissertation. – Technische Universität München, 2024. – DOI: 10.14459/2024.187.

60. Carbo J. E. P., Valdes J. M. A., Beker B. B. On-board computer for CubeSats: state-of-the-art and future trends // *IEEE Access*. – 2024. – DOI: 10.1109/ACCESS.2024.3429399.

61. NASA Small Spacecraft Systems Virtual Institute. State-of-the-art of small spacecraft technology 2024 [Електронний ресурс]. – Washington, DC : NASA, 2024. – URL: https://www.nasa.gov/smallsat-institute/sst-soa/ (дата звернення: 22.11.2025).

62. de Souza K. V. C. K., Bouslimani Y., Ghribi M. Flight software development for a CubeSat application // *IEEE Journal on Miniaturization for Air and Space Systems*. – 2022. – Vol. 3, № 4. – P. 184–196. – DOI: 10.1109/JMASS.2022.3206713.

63. França L. F. L., Benevenuti F., Gobatto L. R., Bastos R. P., Azambuja J. R. F., Kastensmidt F. L. Open source port of NASA's Core Flight System to FreeRTOS in Arm Cortex-M-based microprocessor // *Proc. IEEE 16th Latin America Symp. on Circuits and Systems (LASCAS 2025)*. – Bento Gonçalves, Brazil, 2025. – P. 1–5. – DOI: 10.1109/LASCAS64004.2025.10966295.

64. Swartwout M. CubeSat Database // *Proc. 16th Annual CubeSat Developers Workshop*. – San Luis Obispo, CA, 2019. – URL: https://ntrs.nasa.gov/api/citations/20190002705/downloads/20190002705.pdf (дата звернення: 22.11.2025).

65. Musial A. How not to kill your CubeSat [Електронний ресурс]. – Alicja.Space, 2019. – URL: https://alicja.space/blog/how-not-to-kill-your-cubesat (дата звернення: 22.11.2025).

66. De Oliveira F. F. et al. CubeSats: a survey on subsystems, command and data handling, and failure analysis // *IEEE Access*. – 2023. – Vol. 11. – P. 53369–53394. – DOI: 10.1109/ACCESS.2023.3277547.

67. Grile T. M., Bettinger R. A., Schubert Kabban C. M. Statistical reliability estimation of Earth-orbiting satellites operating from 1991 to 2020 // *Journal of Spacecraft and Rockets*. – 2025. – Vol. 62, № 3. – P. 805–826. – DOI: 10.2514/1.A35971.

68. Langer M., Bouwmeester J. Reliability of CubeSats – statistical data, developers' beliefs and the way forward // *Proc. 30th Annual AIAA/USU Conf. on Small Satellites*. – Logan, UT, USA, 2016. – Paper SSC16-X-4. – URL: https://digitalcommons.usu.edu/smallsat/2016/TS10AdvTech2/4/ (дата звернення: 22.11.2025).

69. Демченко О. Космічний довгобуд: що сталося із супутником «Січ-2-30» та хто за це відповідатиме // *Економічна правда*. – 2022. – 10 лют. – URL: https://www.epravda.com.ua/publications/2022/02/10/682259/ (дата звернення: 22.11.2025).

70. Lopez-Calle I., Franco A. I. Comparison of CubeSat and microsat catastrophic failures in function of radiation and debris impact risk // *Scientific Reports*. – 2023. – Vol. 13. – Art. 385. – DOI: 10.1038/s41598-022-27327-z.

71. Yilmaz B. S. Reliability analysis of CubeSat subsystems // *IEEE Aerospace and Electronic Systems Magazine*. – 2021. – Vol. 36, № 6. – P. 42–50. – DOI: 10.1109/MAES.2021.3061568.

72. Siewert S., Rocha K., Butcher T., Pederson T. Comparison of common instrument stack architectures for small UAS and CubeSats // *Proc. IEEE Aerospace Conf. (AERO 2021)*. – Big Sky, MT, USA, 2021. – P. 1–17. – DOI: 10.1109/AERO50100.2021.9438438.

73. Essasoumati S., Said A., Gharnati F., Raoufi M. Exploring the frontiers: an in-depth study on nanosatellites as the pinnacle of embedded systems in space technology // *Proc. Int. Conf. on Global Aeronautical Engineering and Satellite Technology (GAST 2024)*. – Marrakesh, Morocco, 2024. – P. 1–5. – DOI: 10.1109/GAST60528.2024.10520782.

74. De R., Abegaonkar M. P., Basu A. Enabling science with CubeSats—trends and prospects // *IEEE Journal on Miniaturization for Air and Space Systems*. – 2022. – Vol. 3, № 4. – P. 221–231. – DOI: 10.1109/JMASS.2022.3209897.

75. Bi Z., Yung K. L., Ip A. W. H., Tang Y. M., Zhang C. W. J., Xu L. D. The state of the art of information integration in space applications // *IEEE Access*. – 2022. – Vol. 10. – P. 110110–110135. – DOI: 10.1109/ACCESS.2022.3215154.

76. PRISMA 2020 statement [Електронний ресурс]. – URL: https://www.prisma-statement.org/prisma-2020 (дата звернення: 01.03.2025).

77. Liubimov O., Turkin I. CubeSats and their on-board computers: systematic literature review // *Space Science and Technology*. – 2025. – Vol. 31, № 6. – P. 014–037. – DOI: 10.15407/knit2025.06.014.

78. The Open Graph Viz Platform (Gephi) [Електронний ресурс]. – URL: https://gephi.org/ (дата звернення: 17.03.2025).

79. VOSviewer : visualizing scientific landscapes [Електронний ресурс]. – URL: https://www.vosviewer.com/ (дата звернення: 01.03.2025).

80. IEEE Thesaurus Access Portal [Електронний ресурс]. – URL: https://www.ieee.org/publications/services/thesaurus-access-page.html (дата звернення: 24.04.2025).

81. CubeSat101 : basic concepts and processes for first-time CubeSat developers [Електронний ресурс]. – URL: https://www.nasa.gov/wp-content/uploads/2017/03/nasa_csli_cubesat_101_508.pdf (дата звернення: 24.04.2025).

82. IEEE Xplore [Електронний ресурс]. – URL: https://ieeexplore.ieee.org (дата звернення: 01.03.2025).

83. ScienceDirect [Електронний ресурс]. – URL: https://www.sciencedirect.com (дата звернення: 01.03.2025).

84. Turkin I., Liubimov O., Volobuieva L., Valkovyi V. Hardware and software of CubeSat nanosatellites on-board computers: a systematized literature review // *Aerospace Technic and Technology*. – 2025. – № 3. – P. 116–137. – DOI: 10.32620/aktt.2025.3.11.

85. Jijina S., Amyette A., Shoghi N., Hadidi R., Kim H. Understanding the software and hardware stacks of a general-purpose cognitive drone // *Proc. IEEE*

Int. Symp. on Performance Analysis of Systems and Software (ISPASS 2020). – 2020. – P. 212–214. – DOI: 10.1109/ISPASS48437.2020.00036.

86. Yan S., Li S., Liu X., Zeng C., Liao X., Wang J. Conf-Adaption: adaptive adjustment of software configuration on UAV by resource dependency analysis // *Proc. IEEE 8th Joint Int. Information Technology and Artificial Intelligence Conf. (ITAIC 2019)*. – 2019. – P. 155–161. – DOI: 10.1109/ITAIC.2019.8785785.

87. Rui Y., Xiujuan L., Chuntao L., Xingyu Z. Design of load balancing strategy of UAV flight control software // *Proc. IEEE CSAA Guidance, Navigation and Control Conf. (CGNCC 2018)*. – 2018. – P. 1–6. – DOI: 10.1109/GNCC42960.2018.9018913.

88. Müller H., Palossi D., Mach S., Conti F., Benini L. Fünfiiber-Drone: a modular open-platform 18-grams autonomous nano-drone // *Proc. Design, Automation & Test in Europe Conf. (DATE 2021)*. – 2021. – P. 1610–1615. – DOI: 10.23919/DATE51398.2021.9474262.

89. Zaidi S., Atiquzzaman M., Calafate C. T. Internet of flying things (IoFT): a survey // *Computer Communications*. – 2021. – Vol. 165. – P. 53–74. – DOI: 10.1016/j.comcom.2020.10.023.

90. Hussain S., Li S., Hussain T., Lin X., Ali F., AlZubi A. A. Computing challenges of UAV networks: a comprehensive survey // *Computers, Materials and Continua*. – 2024. – Vol. 81, № 2. – P. 1999–2051. – DOI: 10.32604/cmc.2024.056183.

91. Bilal M., Munawar K., Shaikh M. S., Al-Saggaf U. M. Resource-optimized bare-metal DSP implementation of UAV flight control software // *Proc. 6th Int. Conf. on Intelligent and Advanced Systems (ICIAS 2016)*. – 2016. – P. 1–5. – DOI: 10.1109/ICIAS.2016.7824125.

92. Gargioni G., Peterson M., Persons J. B., Schroeder K., Black J. A full distributed multipurpose autonomous flight system using 3D position tracking and ROS // *Proc. Int. Conf. on Unmanned Aircraft Systems (ICUAS 2019)*. – 2019. – P. 1458–1466. – DOI: 10.1109/ICUAS.2019.8798163.

93. Vierhauser M., Wohlrab R., Stadler M., Cleland-Huang J. AMon: a domain-specific language and framework for adaptive monitoring of cyber–physical systems // *Journal of Systems and Software*. – 2023. – Vol. 195. – Art. 111507. – DOI: 10.1016/j.jss.2022.111507.

94. Vierhauser M., Garmendia A., Stadler M., Wimmer M., Cleland-Huang J. GRuM — a flexible model-driven runtime monitoring framework and its application to automated aerial and ground vehicles // *Journal of Systems and Software*. – 2023. – Vol. 203. – Art. 111733. – DOI: 10.1016/j.jss.2023.111733.

95. Gutmann M., Rinner B. EAMOS: execution of aerial multidrone mission operations and specifications framework // *Proc. Int. Conf. on Unmanned Aircraft Systems (ICUAS 2023)*. – 2023. – P. 761–768. – DOI: 10.1109/ICUAS57906.2023.10156107.

96. Siewert S., Rocha K., Butcher T., Pederson T. Comparison of common instrument stack architectures for small UAS and CubeSats // *Proc. IEEE Aerospace Conf. (AERO 2021)*. – 2021. – P. 1–17. – DOI: 10.1109/AERO50100.2021.9438438.

97. Silva G. R., Becker L. B., Hübner J. F. Embedded architecture composed of cognitive agents and ROS for programming intelligent robots // *IFAC-PapersOnLine*. – 2020. – Vol. 53, № 2. – P. 10000–10005. – DOI: 10.1016/j.ifacol.2020.12.2718.

98. Rico R., Rico-Azagra J., Gil-Martínez M. Hardware and RTOS design of a flight controller for professional applications // *IEEE Access*. – 2022. – Vol. 10. – P. 134870–134883. – DOI: 10.1109/ACCESS.2022.3232749.

99. Prosvirin D. A., Kharchenko V. P. Model-based solution and software engineering environment for UAV critical onboard applications // *Proc. IEEE Int. Conf. Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD 2015)*. – 2015. – P. 312–315. – DOI: 10.1109/APUAVD.2015.7346629.

100. Santos N., Raimundo A., Peres D., Sebastião P., Souto N. Development of a software platform to control squads of unmanned vehicles in real time // *Proc. Int. Conf. on Unmanned Aircraft Systems (ICUAS 2017)*. – 2017. – P. 1–5. – DOI: 10.1109/ICUAS.2017.7991528.

101. Belt J. et al. Model-driven development for the seL4 microkernel using the HAMR framework // *Journal of Systems Architecture*. – 2023. – Vol. 134. – Art. 102789. – DOI: 10.1016/j.sysarc.2022.102789.

102. Fan Y., Li X., Li C., Jia Z. Design of distributed flight control software based on software bus // *Proc. IEEE Chinese Guidance, Navigation and Control Conf. (CGNCC 2016)*. – 2016. – P. 725–729. – DOI: 10.1109/CGNCC.2016.7828875.

103. Hu X., Liu J. Ontology construction and evaluation of UAV FCMS software requirement elicitation considering geographic environment factors // *IEEE Access*. – 2020. – Vol. 8. – P. 106165–106182. – DOI: 10.1109/ACCESS.2020.2998843.

104. Verbeke J., Vantilt J., Vanthienen D., Vochten M., Debruyne S., De Schutter J. A constraint-based flight control system architecture for UAVs using the iTaSC framework // *Proc. Int. Conf. on Unmanned Aircraft Systems (ICUAS 2016)*. – 2016. – P. 310–319. – DOI: 10.1109/ICUAS.2016.7502540.

105. Liubimov O., Turkin I. Multicriteria evaluation of industry-known interfaces of onboard network of CubeSat nanosatellites // *Aerospace Technic and Technology*. – 2024. – № 3. – P. 138–153. – DOI: 10.32620/aktt.2024.3.12.

106. Любімов О. Реалізація енергоефективної бортової мережі між типовими модулями малої авіоніки мікросупутників КубСат (CubeSat) // *Матеріали Міжнародної молодіжної науково-практичної конференції «Людина і Космос – 2024»*. – Дніпро, 2024. – DOI: 10.62717/2221-4550-2024-1-067.

107. Bambagini M., Marinoni M., Aydin H., Buttazzo G. Energy-aware scheduling for real-time systems: a survey // *ACM Transactions on Embedded Computing Systems*. – 2016. – Vol. 15, № 1. – Art. 7. – DOI: 10.1145/2808231.

108. Saifullah S., Fahmida S., Modekurthy V. P., Fisher N., Guo Z. CPU energy-aware parallel real-time scheduling // *Proc. 32nd Euromicro Conf. on Real-Time Systems (ECRTS 2020)*. – 2020. – (LIPIcs ; Vol. 165). – Art. 2. – DOI: 10.4230/LIPIcs.ECRTS.2020.2.

109. Seman L. O., Ribeiro B. F., Rigo C. A., Morsch Filho E., Camponogara E., Leonardi R., Bezerra E. A. An energy-aware task scheduling for quality-of-service assurance in constellations of nanosatellites // *Sensors*. – 2022. – Vol. 22, № 10. – Art. 3715. – DOI: 10.3390/s22103715.

110. Rigo C. A., Morsch Filho E., Seman L. O., Loures L., Leithardt V. R. Q. Instance and data generation for the offline nanosatellite task scheduling problem // *Data*. – 2023. – Vol. 8, № 3. – Art. 62. – DOI: 10.3390/data8030062.

111. Slongo L. K., Martínez S. V., Eiterer B. V. B., Pereira T. G., Bezerra E. A., Paiva K. V. Energy-driven scheduling algorithm for nanosatellite energy harvesting maximization // *Acta Astronautica*. – 2018. – Vol. 147. – P. 141–151. – DOI: 10.1016/j.actaastro.2018.03.052.

112. She R., Wu Y., Cui E. Energy-aware scheduling for reliability-oriented real-time parallel applications allocation on heterogeneous computing systems // *Future Generation Computer Systems*. – 2025. – Vol. 168. – DOI: 10.1016/j.future.2025.107738.

113. Wu G., Wang H., Pedrycz W., Li H., Wang L. Satellite mission planning considering energy constraints and battery lifetime // *IEEE Transactions on Aerospace and Electronic Systems*. – 2023. – Vol. 59, № 4. – P. 3676–3689. – DOI: 10.1109/TAES.2023.3236872.

114. Saeed B. et al. An integrated thermal-electrical model for simulations of battery behavior in CubeSats // *Applied Sciences*. – 2021. – Vol. 11, № 4. – Art. 1554. – DOI: 10.3390/app11041554.

115. Camponogara E., Seman L. O., Rigo C. A., Morsch Filho E., Ribeiro B. F., Bezerra E. A. A continuous-time formulation for optimal task scheduling and quality-of-service assurance in nanosatellites // *Computers & Operations Research*. – 2022. – Vol. 147. – Art. 105945. – DOI: 10.1016/j.cor.2022.105945.

116. Iqbal S. M. Z., Saifullah A. Schedulability analysis for energy harvesting real-time systems with discrete energy levels // *Proc. 32nd Euromicro Conf. on Real-Time Systems (ECRTS 2020)*. – Dagstuhl, 2020. – (LIPIcs ; Vol. 165). – Art. 2. – DOI: 10.4230/LIPIcs.ECRTS.2020.2.

117. Iqbal S. M. Z., Saifullah A. Optimal energy-aware task scheduling for batteryless IoT devices // *IEEE Transactions on Parallel and Distributed Systems*. – 2020. – Vol. 31, № 6. – P. 1296–1312. – DOI: 10.1109/TPDS.2019.2959123.

118. Gupta R., Krishna C. M. Scheduling of real-time tasks with energy harvesting // *Journal of Systems Architecture*. – 2019. – Vol. 97. – P. 43–52. – DOI: 10.1016/j.sysarc.2019.03.004.

119. Farid G., Abdelrahman M. Orbit determination of CubeSats using Global Positioning System // *Acta Astronautica*. – 2017. – Vol. 135. – P. 64–74. – DOI: 10.1016/j.actaastro.2017.03.003.

120. Huang P., Kumar P., Stoimenov K., Thiele L. Throughput maximization for energy harvesting embedded systems with suspend and change tasks // *IEEE Transactions on Sustainable Computing*. – 2018. – Vol. 3, № 1. – P. 1–13. – DOI: 10.1109/TSUSC.2017.2673805.

121. Niu L., Quan G. Energy-aware scheduling for real-time tasks under uncertainty in energy harvesting systems // *Journal of Systems Architecture*. – 2018. – Vol. 88. – P. 1–13. – DOI: 10.1016/j.sysarc.2018.05.002.

122. Seman L. O., Rigo C. A., Camponogara E., Munari P., Bezerra E. A. Improving energy-aware nanosatellite task scheduling by a branch-cut-and-price algorithm // *Computers & Operations Research*. – 2023. – Vol. 158. – Art. 106292. – DOI: 10.1016/j.cor.2023.106292.

123. Zhao Q. et al. Semi-partitioned scheduling of mixed-criticality systems on multiprocessors // *Real-Time Systems*. – 2021. – Vol. 57, № 1. – P. 96–128. – DOI: 10.1007/s11241-020-09366-0.

124. Yoon M.-K., Gerstlauer A. Integrated power management and scheduling for energy harvesting real-time systems // *IEEE Transactions on Computers*. – 2017. – Vol. 66, № 6. – P. 933–946. – DOI: 10.1109/TC.2016.2625090.

125. George A. D., Wilson C. M. Onboard processing with hybrid and reconfigurable computing on small satellites // *Proceedings of the IEEE*. – 2018. – Vol. 106, № 3. – P. 458–470. – DOI: 10.1109/JPROC.2018.2799518.

126. Lovelly T. M., Wise T. W., Holtzman S. H., George A. D. Benchmarking analysis of space-grade central processing units and field-programmable gate arrays // *Journal of Aerospace Information Systems*. – 2018. – Vol. 15, № 8. – P. 518–529. – DOI: 10.2514/1.I010621.

127. Sánchez-Sanjuan S., González-Llorente J. Small satellite avionics: a survey // *IEEE Access*. – 2022. – Vol. 10. – P. 119933–119957. – DOI: 10.1109/ACCESS.2022.3221867.

128. Radhakrishnan R., Edmonson W. W., Afghah F., Rodriguez-Osorio R. M., Pinto F., Burleigh S. C. Survey of inter-satellite communication for small satellite systems: physical layer to network layer view // *IEEE Communications Surveys & Tutorials*. – 2016. – Vol. 18, № 4. – P. 2442–2473. – DOI: 10.1109/COMST.2016.2564990.

129. Seman L. O. et al. Optimal task scheduling for nanosatellite constellations using hybrid algorithms // *Acta Astronautica*. – 2022. – Vol. 192. – P. 252–262. – DOI: 10.1016/j.actaastro.2021.12.011.

130. Пасічник М. Ю., Зайцев В. Є. Методи планування завдань у системах реального часу // *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. – 2024. – № 56. – С. 39–43.

131. Buttazzo G. C. Hard real-time computing systems: predictable scheduling algorithms and applications. – 3rd ed. – New York : Springer, 2011. – 521 p. – DOI: 10.1007/978-1-4614-0676-1.

132. Marwedel P. Embedded system design: embedded systems foundations of cyber-physical systems, and the Internet of Things. – 4th ed. – Cham : Springer, 2021. – 433 p. – DOI: 10.1007/978-3-030-60910-7.

133. Burns A., Davis R. I. A review of research into mixed criticality systems // *ACM Computing Surveys*. – 2024. – Vol. 50, Iss. 6. – Art. 82.

134. Романкевич А. М., Романкевич В. О., Кононова О. В. Методи забезпечення відмовостійкості в розподілених системах реального часу //

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – 2021. – № 73. – С. 12–18. – DOI: 10.20535/1810-0546.2021.102856.

135. Zolfaghari B., Bibak K. Real-time and embedded systems: life cycle, ecosystem and scheduling trends. – Amsterdam : Elsevier, 2024. – 412 p. – ISBN 978-0-443-21494-3.

136. Baruah S., Burns A., Davis R. I. Response time analysis for real-time systems. – Dagstuhl : Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021. – 257 p. – DOI: 10.4230/OASIcs.Vol8.

137. Saikrishna P. S., Meghanathan N. A review of CPU scheduling algorithms in real-time systems // *International Journal of Computer Applications*. – 2020. – Vol. 176, № 35. – P. 26–32. – DOI: 10.5120/ijca2020920863.

138. Sahoo S. R., Gupta A. K., Sahoo K. S. A comprehensive survey on real-time scheduling algorithms for Internet of Things applications // *Journal of King Saud University – Computer and Information Sciences*. – 2023. – Vol. 35, Iss. 8. – Art. 101638. – DOI: 10.1016/j.jksuci.2023.101638.

139. Dai X., Burns A. Schedulability analysis for real-time systems with dynamic priority scheduling // *ACM Computing Surveys*. – 2023. – Vol. 55, Iss. 12. – Art. 245. – DOI: 10.1145/3570321.

140. Bhatia R., Kumar S. Analysis of real-time scheduling algorithms: RM, EDF, and LLF // *Proc. Third Int. Conf. on Computing, Communications, and Cyber-Security*. – Singapore : Springer, 2023. – P. 451–461. – (Lecture Notes in Networks and Systems ; Vol. 421). – DOI: 10.1007/978-981-19-1142-2_36.

141. Ma L., Gu X. A review of fixed priority scheduling for hard real-time systems // *Journal of Physics: Conference Series*. – 2021. – Vol. 1757. – Art. 012084. – DOI: 10.1088/1742-6596/1757/1/012084.

142. Horbulin V. P., Badrak V. V. Historic decision: Ukraine builds military capabilities in the space domain // *Visnyk Natsionalnoi Akademii Nauk Ukrainy*. – 2025. – № 5. – P. 3–10. – DOI: 10.15407/visn2025.05.003.

143. Liubimov O., Turkin I., Cheranovskiy V., Volobuieva L. UAV mission computer operation mode optimization focusing on computational energy efficiency and system responsiveness // *Computation*. – 2024. – Vol. 12, № 12. – Art. 235. – DOI: 10.3390/computation12120235.

144. Ложковський А. Г. Теорія масового обслуговування в телекомунікаціях : навч. посіб. – Одеса : ОНАЗ ім. О. С. Попова, 2010. – 112 с.

145. Qi L., Fan J., Cai H., Fang Z. A survey of emerging memory in a microcontroller unit // Micromachines. – 2024. – Vol. 15, № 4. – Art. 488. – DOI: 10.3390/mi15040488.

146. Wu H., Chen C., Weng K. An energy-efficient strategy for microcontrollers // Applied Sciences. – 2021. – Vol. 11. – Art. 2581. – DOI: 10.3390/app11062581.

147. IBM. A 2.7 pJ/cycle 16 MHz, 0.7 µW deep sleep power ARM Cortex-M0+ core SoC in 28 nm FD-SOI technology [Електронний ресурс]. – URL: https://research.ibm.com/publications/a-27-pjcycle-16-mhz-07-mw-deep-sleep-power-arm-cortex-m0-core-soc-in-28-nm-fd-soi (дата звернення: 20.07.2024).

148. Xia F., Ehsan S., Yang T., Sun Y. Control-theoretic dynamic voltage scaling for embedded controllers [Електронний ресурс] // *arXiv preprint*. – 2008. – URL: https://arxiv.org/abs/0806.0132 (дата звернення: 20.07.2024).

149. Ma W., Cheng Q., Gao Y., Xu L., Yu N. An ultra-low-power embedded processor with variable micro-architecture // Micromachines. – 2021. – Vol. 12, № 3. – Art. 292. – DOI: 10.3390/mi12030292.

150. Turkin I., Liubimov O., Volobuieva L. Reactivity model of nanosatellite and UAVs embedded computing systems // *Aerospace Technic and Technology*. – 2025. – № 6. – P. 128–142. – DOI: 10.32620/aktt.2025.6.12.

151. Liubimov O., Turkin I. Experimental study of FreeRTOS operating system reactivity in power saving modes of the onboard computer microcontroller // *Aerospace Technic and Technology*. – 2024. – № 4. – P. 71–86. – DOI: 10.32620/aktt.2024.4.09.

152. IBM. Containers in the enterprise [Електронний ресурс]. – URL: https://www.ibm.com/downloads/cas/VG8KRPRM (дата звернення: 20.07.2024).

153. Tamanaka G. T. B., Aroca R. V., de Paula Caurin G. A. Fault-tolerant architecture and implementation of a distributed control system using containers // *Proc. Latin American Robotics Symp. (LARS 2022), Brazilian Symp. on Robotics (SBR 2022), Workshop on Robotics in Education (WRE 2022)*. – São Bernardo do Campo, Brazil, 2022. – P. 1–6. – DOI: 10.1109/LARS/SBR/WRE56824.2022.9995745.

154. Liubimov O., Turkin I., Leshchenko O., Valkovyi V. Experimental evaluation of the efficiency of containerization technology in the onboard software of a nanosatellite CubeSat // *Aerospace Technic and Technology*. – 2024. – № 4. – P. 87–103. – DOI: 10.32620/aktt.2024.4.10.

155. Zidar J., Matić T., Aleksi I., Hocenski Ž. Dynamic voltage and frequency scaling as a method for reducing energy consumption in ultra-low-power embedded systems // *Electronics*. – 2024. – Vol. 13, № 5. – Art. 826. – DOI: 10.3390/electronics13050826.

156. Ali H., Tariq U. U., Hardy J., Zhai X., Lu L., Zheng Y., Bensaali F., Amira A., Fatema K., Antonopoulos N. A survey on system-level energy optimisation for MPSoCs in IoT and consumer electronics // *Computer Science Review*. – 2021. – Vol. 41. – Art. 100416. – DOI: 10.1016/j.cosrev.2021.100416.

157. Turkin I., Liubimov O., Zakharenko V. Optimization of energy consumption of the CubeSat on-board computer under real-time limitations // *Aerospace Technic and Technology*. – 2024. – № 3. – P. 126–137. – DOI: 10.32620/aktt.2024.3.10.

158. Salamun K., Pavić I., Džapo H., Čuljak I. Weakly hard real-time model for control systems: a survey // *Sensors*. – 2023. – Vol. 23, № 10. – Art. 4652. – DOI: 10.3390/s23104652.

159. Choi H., Kim H., Zhu Q. Job-class-level fixed priority scheduling of weakly-hard real-time systems // *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2019)*. – 2019. – DOI: 10.1109/RTAS.2019.00028.

160. Bernat G., Burns A., Liamosi A. Weakly hard real-time systems // *IEEE Transactions on Computers*. – 2001. – Vol. 50, № 4. – P. 308–321. – DOI: 10.1109/12.919277.

161. Liubimov O., Turkin I., Pavlikov V., Volobuieva L. Agile software development lifecycle and containerization technology for CubeSat command and data handling module implementation // *Computation*. – 2023. – Vol. 11, № 9. – Art. 182. – DOI: 10.3390/computation11090182.

162. Turkin I., Volobuieva L., Chukhray A., Liubimov O. Comparison of equivalent circuit and machine learning methods for CubeSat battery discharge modeling // *Radioelectronic and Computer Systems*. – 2025. – № 3. – P. 231–244. – DOI: 10.32620/reks.2025.3.16.

163. Liubimov O., Turkin I., Volobuieva L. The WASM3 interpreter as a hard real-time software platform for the on-board computer of a student nanosatellite // *Proc. 13th Int. Conf. on Dependable Systems, Services and Technologies (DESSERT 2023)*. – Athens, Greece, 2023. – P. 1–8. – DOI: 10.1109/DESSERT61349.2023.10416436.

164. FreeRTOS : official website of the real-time operating system [Електронний ресурс]. – URL: https://www.freertos.org/ (дата звернення: 15.12.2025).

165. Mars Helicopter (Ingenuity) [Електронний ресурс] / NASA Science. – URL: https://science.nasa.gov/mission/mars-2020-perseverance/ingenuity-mars-helicopter/ (дата звернення: 20.12.2025).

166. Near-Earth Asteroid Scout (NEA Scout) [Електронний ресурс] / NASA Jet Propulsion Laboratory. – URL: https://science.nasa.gov/mission/nea-scout/ (дата звернення: 20.12.2025).

167. Lunar Flashlight [Електронний ресурс] / NASA Jet Propulsion Laboratory. – URL: https://www.jpl.nasa.gov/missions/lunar-flashlight (дата звернення: 20.12.2025).

168. MPLAB Harmony v3 Embedded Software Development Framework [Електронний ресурс] / Microchip Technology Inc. – URL: https://www.microchip.com/en-us/tools-resources/configure/mplab-harmony (дата звернення: 20.12.2025).

169. Любімов О. Модель модульного та компонентного підходу до проєктування ПЗ наносупутників на базі відкритих рішень NASA F'Prime // *Матеріали Міжнародної молодіжної науково-практичної конференції «Людина і Космос – 2024»*. – Дніпро, 2024.

170. Liubimov O. Use of micro-services architecture and containerization for the fast development and testing of the CubeSat nanosatellites software // *Journal of Rocket-Space Technology*. – 2023. – Vol. 31, № 4. – P. 128–137. – DOI: 10.15421/452317.

ДОДАТОК А. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

«ЗАТВЕРДЖУЮ»
Проректор з НПР
Національного аерокосмічного університету
«Харківський авіаційний інститут»
_____ Андрій ГУМЕННИЙ
«___» _____ 2025 р.

Акт

Науково-технічної комісії щодо впровадження результатів дисертаційної роботи на здобуття наукового ступеню доктора філософії аспіранта Національного аерокосмічного університету «ХАІ»

Любімова Олександра Вікторовича

Науково-технічна комісія у складі: голова комісії – заступник завідувача кафедри інженерії програмного забезпечення (603) Волобуєва Ліна Олексіївна та члени комісії: к.т.н., доцент каф. 603 Соколова Євгенія Віталіївна, к.т.н., доцент каф. 603 Захаренко Володимир Олександрович склали даний акт про те, що результати дисертаційної роботи Любімова Олександра Вікторовича впроваджені у навчальний процес кафедри 603 як складові нової дисципліни «Інженерія бортового програмного забезпечення БПЛА та наносупутників» освітньо-наукової програми «Інженерія програмного забезпечення» підготовки магістрів, а саме:

- математична модель для розрахунку енергетичних витрат, обчислювальної продуктивності та реактивності бортових обчислювачів наносупутників;
- метод адаптивного регулювання режимів роботи бортових обчислювачів наносупутників;
- метод структурно-параметричної ідентифікації комплексної математичної моделі для розрахунку енергетичних витрат, обчислювальної продуктивності та реактивності бортового програмного забезпечення.

Розроблені в дисертації моделі, методи та алгоритми є основою теорії та практики авторського курсу «Інженерія бортового програмного забезпечення БПЛА та наносупутників», методичне забезпечення якого представлено в силабусі дисципліни - https://mentor.khai.edu/course/view.php?id=9429.

Голова комісії Ліна ВОЛОБУЄВА

Члени комісій Євгенія СОКОЛОВА

Володимир ЗАХАРЕНКО

«ЗАТВЕРДЖУЮ»
Голова правління
АТ «ХАРТРОН»
_____ Микола ВАХНО
«1» жовтня 2025 р.

АКТ
використання наукових результатів
дисертаційної роботи Любімова Олександра Вікторовича на тему
«Адаптивні методи, алгоритми та програмні засоби балансування реактивності та енергоспоживання бортових обчислювачів наносупутників CubeSat», представленої на здобуття наукового ступеню доктора філософії за спеціальністю
121 - інженерія програмного забезпечення

Складений 30 вересня 2025 року комісією у складі: голова комісії – голова правління, голова Науково-технічної ради, кандидат технічних наук Микола ВАХНО, члени комісії: заступник голови правління Юрій НЕЙМИРОК, технічний директор АТ «ХАРТРОН», секретар Науково-технічної ради Сергій РУСАКОВ.

У період з 10 серпня по 30 вересня 2025 року комісія розглянула наукові результати дисертації Любімова Олександра Вікторовича, представленої на здобуття наукового ступеня доктора філософії за спеціальністю 121 – Інженерія програмного забезпечення. Дисертаційна робота виконана Любімовим О.В. на кафедрі інженерії програмного забезпечення Національного Аерокосмічного Університету «ХАІ».

Комісія встановила:
1. Наукові та практичні результати дисертації отримано в рамках діючої Угоди № 4 від 28.11.2024 року між Національним аерокосмічним університетом та АТ «ХАРТРОН» про наміри комерційного впровадження результатів проєкту № 168/0143.
2. В проєктній практиці групи компаній АТ «ХАРТРОН» планується використовувати розроблений в дисертації метод структурно-параметричної ідентифікації моделі енергоефективності як інструмент ескізного проєктування перспективних енергоефективних бортових систем керування мікросупутників.

Голова комісії:
Голова правління
АТ «ХАРТРОН Микола ВАХНО

Члени комісії:
Заступник голови
правління Юрій НЕЙМИРОК

Технічний директор
АТ «ХАРТРОН» Сергій РУСАКОВ

«ЗАТВЕРДЖУЮ»
Генеральний директор
ТОВ «АБАКУС-ТЕХ»
Олександр ТАРГОНСЬКИЙ
«6 » жовтня 2025 р.

АКТ
використання наукових результатів
дисертаційної роботи Любімова Олександра Вікторовича на тему
«Адаптивні методи, алгоритми та програмні засоби балансування реактивності та енергоспоживання бортових обчислювачів наносупутників CubeSat», представленої на здобуття наукового ступеню доктора філософії за спеціальністю
121 – інженерія програмного забезпечення

Складений 30 вересня 2025 року комісією у складі: голова комісії – генеральний директор Олександр ТАРГОНСЬКИЙ, члени комісії: керівник з розробки продуктів – Олександр Гальцев, провідний інженер - Костянтин СІРЕНКО.

У період з 4 серпня по 22 вересня 2025 року комісія розглянула наукові результати дисертації Любімова Олександра Вікторовича, представленої на здобуття наукового ступеня доктора філософії за спеціальністю 121 – Інженерія програмного забезпечення. Дисертаційна робота виконана Любімовим О.В. на кафедрі інженерії програмного забезпечення Національного Аерокосмічного Університету «ХАІ».

Комісія встановила:
1. Наукові та практичні результати дисертації отримано в рамках діючої Угоди № 17 від 02.12.2024 року між Національним аерокосмічним університетом та ТОВ «АБАКУС-ТЕХ» про наміри комерційного впровадження результатів проєкту № 168/0143 «Експериментальне відпрацювання бортового обчислювача безпілотного літального апарата подвійного призначення».
2. В проєктному портфелі компанії ТОВ «АБАКУС-ТЕХ» планується використовувати розроблений в дисертації метод адаптивного регулювання режимами роботи бортового обчислювача як підхід забезпечення автономності перспективних енергоефективних бортових систем керування БПЛА.
3. Узгодити авантпроєкт з застосування авторського бортового обчислювача «Боривітер» як системи керування перспективними БПЛА під керуванням ПЗ PX4 Autopilot.

Голова комісії:
Генеральний директор
ТОВ «АБАКУС-ТЕХ» Олександр ТАРГОНСЬКИЙ

Члени комісії:
Керівник розробки продуктів
ТОВ «АБАКУС-ТЕХ» Олександр ГАЛЬЦЕВ

Провідний інженер
ТОВ «АБАКУС-ТЕХ» Костянтин СІРЕНКО

УКРАЇНА
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ОСОБЛИВЕ КОНСТРУКТОРСЬКЕ БЮРО «ШТОРМ»
03056, Київ-56
вул. Янгеля, 7А, корп.11
Тел. (044) 204-90-16
Тел./факс (044) 236-20-91
sipulka@kpi.ua

Вих. № 4251 від 29.08, 2025 р.

ЛИСТ-ПІДТРИМКА

ОКБ «Шторм», який є науково виробничим структурним підрозділом КПІ ім. Ігоря Сікорського займається розробкою та виготовленням різноманітних електронних і механічних систем, в тому числі бортової космічної апаратури та супутників в цілому.

ОКБ «Шторм» ознайомлений з результатами дисертаційних досліджень Любімова Олександра Вікторовича на тему "АДАПТИВНІ МЕТОДИ, АЛГОРИТМИ ТА ПРОГРАМНІ ЗАСОБИ БАЛАНСУВАННЯ РЕАКТИВНОСТІ ТА ЕНЕРГОСПОЖИВАННЯ БОРТОВИХ ОБЧИСЛЮВАЧІВ НАНОСУПУТНИКІВ CUBESAT". Результати дисертації можуть бути застосовані при розробці та виготовленні космічної бортової апаратури, наприклад, в складі наносупутників формату CubeSat.

На початку червня 2025 року ОКБ «Шторм» отримав запит від Любімова Олександра Вікторовича з проханням проведення аналізу та перевірки одного зразка електронної плати керування та забезпечення функціонування, в тому числі системи навігації та стабілізації наносупутника формату CubeSat. Електронна плата була розроблена в рамках проекту № 168/0143 «Експериментальне відпрацювання бортового обчислювача безпілотного літального апарата подвійного призначення». Також перевірці передбачалась перевірка тестового програмного забезпечення для дистанційного контролю роботи електронної плати в реальному часі.

Протягом липня - серпня 2025 року з використанням технічних засобів ОКБ «Шторм» та інших підрозділів КПІ ім. Ігоря Сікорського були проведені наступні роботи з перевірки функціонування наданої електронної плати:

- стійкість на вплив механічних навантажень в трьох площинах в діапазоні частот 20 – 2000 Гц з щільністю від 0.0044 g^2/Гц до 0.00644 g^2/Гц, GRMS з щільністю 5.13 g^2/Гц (плата не підключена до джерела живлення);
- вплив електромагнітного випромінення (плата не підключена до джерела живлення) в діапазоні частот 80 – 7000 МГц з відповідними рівнями сигналів 130 dBm V/m - 157 dBm V/m;
- роботу в крайніх точках діапазону робочих температур від мінус 60 °С до +60 °С (плата підключена до джерела живлення та контролюється роботоздатність в реальному часі);
- термовакуумні випробування електронної плати (під захисним тепловим екраном) з імітацією темної та сонячної сторін орбіти при зміні температур на

поверхні захисного екрану від мінус 100 °С до +120 °С зі швидкістю 2 °С/хв та тиском на рівні 10^{-5}Па.

Перевірки функціонування проводились паралельно з випробуванням бортової космічної апаратури, виготовленої ОКБ «Шторм», тому фінансові затрати саме на перевірку електронної плати розробки Любімова Олександра Вікторовича склали 0 грн.

Результати перевірки функціонування електронної плати розробки Любімова Олександра Вікторовича – БО «Боривітер»:

- **впливу** механічних навантажень в трьох площинах в діапазоні частот 20 – 2000 Гц з щільністю від 0.0044 g^2/Гц до 0.00644 g^2/Гц, GRMS з щільністю 5.13 g^2/Гц на якість монтажу електронних компонентів, електричних з'єднань та цілісності пасивних та активних компонентів **не виявлено**;
- **впливу** електромагнітного випромінення в діапазоні частот 80 – 7000 МГц з відповідними рівнями сигналів 130 dBm V/m - 157 dBm V/m на функціонування електронної плати **не виявлено**;
- діапазон робочих температур, при яких плата нормально функціонувала в реальному часі склав від **мінус 25°С до +60°С**, що є достатньою для роботи в складі наносупутників формату CubeSat;

  Примітка. Середня температура бортової апаратури в внутрішньому об'ємі наносупутника може коливатись від мінус 10 до + 25 °С, в залежності до використаних матеріалів та конструкції механічних елементів зовнішнього корпусу.
- після проведення семи термовакуумних циклів (імітація семи обертів на робочій орбіті) електронна плата нормально функціонувала, при цьому телеметричний датчик температури, що входив до складу вимірювальної апаратури термовакуумної камери та був закріплений на друкованій платі, показував зміну температур від мінус 5 °С до +25°С. **Впливу** низького вакууму на рівні 10^{-5}Па впродовж 12 годин проведення випробувань на функціонування електронної плати **не виявлено**.

Таким чином було підтверджено відповідними випробуваннями роботоздатність та нормальне функціонування електронної плати системи керування, навігації та стабілізації в рамках проекту № 168/0143 «Експериментальне відпрацювання бортового обчислювача безпілотного літального апарата подвійного призначення» для використання в наносупутниках формату CubeSat.

ОКБ «Шторм» КПІ ім. Ігоря Сікорського зацікавлений в використанні представлених в дисертації Любімова О. В. методів, алгоритмів та відповідного програмного забезпечення в майбутніх проєктах, що наразі будуть виконуватись в рамках космічної програми університету в період 2026 – 2027 років.

Напрямами можливих спільних досліджень є узгодження алгоритмів роботи в складі всієї бортової апаратури, враховуючи відповідне наукове призначення космічного апарату, а також адаптація по механічним (розміри, вага, моменти інерції, тощо) та електричним інтерфейсам (напруга, споживання, інтерфейси взаємодії, тощо) бортового обчислювача.

З повагою.

Головний інженер ОКБ «Шторм» **Сергій ПУХА**

ДОДАТОК Б. ОБЛІКОВА КАРТКА НДДКР

Облікова картка НДДКР

Державний обліковий номер: 0224U032745

Державний реєстраційний номер: 0124U004847

Відкрита

Дата реєстрації: 03-12-2024

1. Етапи виконання

Номер етапу: 1

Назва етапу: Підготовка до експериментального відпрацювання бортового обчислювача (БО) безпілотного літального апарата подвійного призначення (БПЛА ПП)

Початок етапу: 08-2024

Закінчення етапу: 11-2024

Вид звітного документа: Проміжний звіт

2. Виконавець

Назва організації: Національний аерокосмічний університет ім. М. Є. Жуковського "Харківський авіаційний інститут"

Код ЄДРПОУ/ІПН: 02066769

Підпорядкованість: Міністерство освіти і науки України

Адреса: вул. Чкалова, буд. 17, м. Харків, Харківський р-н., Харківська обл., 61070, Україна

Телефон: 380573151131

Телефон: 380573151056

E-mail: khai@khai.edu

WWW: http://www.khai.edu

3. Власник результатів НДДКР (продукції)

Назва організації: Національний аерокосмічний університет ім. М. Є. Жуковського "Харківський авіаційний інститут"

Код ЄДРПОУ/ІПН: 02066769

Адреса: вул. Чкалова, буд. 17, м. Харків, Харківський р-н., Харківська обл., 61070, Україна

Підпорядкованість: Міністерство освіти і науки України

Телефон: 380573151131

Телефон: 380573151056

E-mail: khai@khai.edu

WWW: http://www.khai.edu

4. Джерела та напрями фінансування

Підстава для проведення робіт: 34 - договір (замовлення) з центральним органом виконавчої влади, академією наук (головними розпорядниками бюджетних коштів на проведення НДДКР)

КПКВК: 2201300

Напрям фінансування: 2.2 - прикладні дослідження і розробки

Джерела фінансування

Джерело фінансування: 7713 - кошти держбюджету

Фактичний обсяг фінансування за звітний етап: 774.529 тис. грн.

5. Науково-технічна робота

Назва роботи (укр)

Експериментальне відпрацювання бортового обчислювача безпілотного літального апарата подвійного призначення

Назва роботи (англ)

Experimental development of the on-board computer of a dual-purpose unmanned aerial vehicle

Реферат (укр)

Сучасна залежність українських виробників безпілотних літальних апаратів (БПЛА) від імпортних компонентів, зокрема китайського виробництва, створює критичні ризики. Насамперед, вони включають технічні обмеження, вразливість до засобів радіоелектронної боротьби (РЕБ), а також можливість ембарго постачальників. Більшість БПЛА працюють на базі відкритої платформи ArduPilot, яка хоча і популярна, має обмежену стійкість до сучасних викликів. Проект спрямований розробці національної альтернативи з використанням сучасного програмного забезпечення на основі відкритої платформи F Prime (JPL/NASA) та високонадійних апаратних компонентів: бортового обчислювача (БО) "Боривітер" для БПЛА подвійного призначення з рівнем технологічної готовності TRL 5/6. Ключовими особливостями розробки є підтримка апаратною складовою сучасних стандартів та вимог до надійності, захищеності, швидкодії та енергетичної ефективності, а також гарантування вимог до програмного забезпечення через використання відкритої платформи F Prime (JPL/NASA) у поєднанні з відкритою операційною системою реального часу FreeRTOS. Результати виконання першого етапу: виконана інтеграція платформи F Prime з FreeRTOS для бортового обчислювача "Боривітер", розроблена проєктна документація та документація користувача для системного програмного забезпечення; апаратна складова підготовлена до механічних та кліматичних випробувань; розроблені програми-методики випробувань. Проект є значним кроком до створення вітчизняної уніфікованої платформи для управління БПЛА. Він забезпечує незалежність України від імпорту критичних компонентів та пропонує надійне рішення для військових і цивільних завдань. Подальші перспективи включають масштабування розробки, адаптацію для різних класів БПЛА та інтеграцію нових технологій для підвищення ефективності.

Реферат (англ)

Ukrainian manufacturers of unmanned aerial vehicles (UAVs) ' current dependence on imported components, particularly Chinese production, creates critical risks. First, they include technical limitations, vulnerability to electronic warfare (EW), and the possibility of supplier embargoes. Second, most UAVs work based on the open platform ArduPilot, which, although popular, has limited resistance to modern challenges. The project is aimed at the development of a national alternative using modern software based on the open platform F Prime (JPL/NASA) and highly reliable hardware components: the onboard computer (BO) "Boriviter" for a dual-purpose UAV with the technological readiness level TRL 5/6. Key features of the development are: hardware supports modern standards and requirements for reliability, security, speed, and energy efficiency, as well as guaranteeing software requirements through the open platform F Prime (JPL/NASA) combined with the open real-time operating system FreeRTOS. The results of the first project stage are: the integration of the F Prime platform with FreeRTOS for the onboard computer "Boryviter" has been completed; developed project documentation and user documentation for system software; the hardware component is prepared for mechanical and climatic tests; the test methods programs have been created. The project is a significant step towards creating a domestic unified platform for UAV control. It ensures Ukraine's independence from importing critical components and offers a reliable solution for military and civilian tasks. Prospects include scaling the development, adapting it for different classes of UAVs, and integrating new technologies to improve efficiency.

Індекс УДК: , , , 004.272, 629.7.05, 629.735.055

Коди тематичних рубрик НТІ: 20.57.01, 20.58.02, 28.33.03

6. Науково-технічна продукція (НТП)

НТП 1

Назва продукції (укр): Системне програмне забезпечення бортового обчислювача безпілотного літального апарата ПП

Назва продукції (англ): System software (SS) of the on-board computer of a dual-purpose unmanned aerial vehicle

Очікувані результати: Програмні продукти

Галузь застосування: Виробництво БПЛА, наносупутників та роботизованих платформ

Опис продукції (укр): -СПЗ забезпечує функціонування програмної платформи F Prime (F') від JPL (Jet Propulsion Laboratory/NASA) на БО "Боривітер" під управлінням операційної системи реального часу FreeRTOS.

Соціально-економічна спрямованість НТП: Створення принципово нової продукції (матеріалів, технологій тощо) для забезпечення експортного потенціалу та заміщенню імпорту

Стадія завершеності НТП: Дослідний зразок

Впровадження НТП: Не впроваджено

Строки впровадження: 08.202411.2024

Виробник продукції: ХАІ

Споживачі продукції:

Перспективні ринки:

Права інтелектуальної власності: Подано заявку на видачу охоронного документу

Форми та умови передачі продукції: Продаж ліцензій, Продаж продукції

НТП 2

Назва продукції (укр): Програми-методики випробувань БО БПЛА ПП на механічну та кліматичну стійкості

Назва продукції (англ): Programs-methodology for experimental training of a dual-purpose UAV BO for mechanical and climatic resistance

Очікувані результати: Нормативні документи

Галузь застосування: Виробництво БПЛА, наносупутників та роботизованих платформ

Опис продукції (укр): Програми-методики випробувань БО БПЛА ПП на температурну стійкість, вологостійкість, стійкість до перепадів тиску, ударостійкість, вібростійкість.

Соціально-економічна спрямованість НТП: Створення принципово нової продукції (матеріалів, технологій тощо) для забезпечення експортного потенціалу та заміщенню імпорту

Стадія завершеності НТП: Звіт по НДДКР, Дослідний зразок

Впровадження НТП: Не впроваджено

Строки впровадження: 08.202411.2024

Виробник продукції: ХАІ

Споживачі продукції:

Перспективні ринки:

Права інтелектуальної власності: За договорами

Форми та умови передачі продукції: Продаж продукції

НТП 3

Назва продукції (укр): Технічна документація та документація користувача на СПЗ та БО БПЛА ПП

Назва продукції (англ): Technical documentation and user documentation for the SSP and a dual purpose UAV BO

Очікувані результати: Технології

Галузь застосування: Виробництво БПЛА, наносупутників та роботизованих платформ

Опис продукції (укр): Звіт з портування відкритої програмної платформи F Prime на БО "Боривітер" під управлінням операційної системи реального часу FreeRTOS; Специфікація вимог до системного програмного забезпечення; Документ «Проєктна архітектура системного програмного забезпечення»; Специфікація вимог до тестового програмного забезпечення (ТПЗ). ТПЗ забезпечуватиме випробування БО "Боривітер" на механічну та кліматичну стійкості.

Соціально-економічна спрямованість НТП: Створення принципово нової продукції (матеріалів, технологій тощо) для забезпечення експортного потенціалу та заміщенню імпорту

Стадія завершеності НТП: Експериментальний (макетний зразок)

Впровадження НТП: Не впроваджено

Строки впровадження: 08.202411.2024

Виробник продукції: ХАІ

Споживачі продукції:

Перспективні ринки:

Права інтелектуальної власності: Подано заявку на видачу охоронного документу

Форми та умови передачі продукції: Продаж патента

7. Бібліографічний опис

- Liubimov, O., Turkin, I., Cheranovskiy, V., & Volobujeva, L. (2024). UAV Mission Computer Operation Mode Optimization Focusing on Computational Energy Efficiency and System Responsiveness. Computation, 12(12), 235. https://doi.org/10.3390/computation12120235;

- Liubimov O., Turkin I. Experimental study of FreeRTOS operating system reactivity in power saving modes of the onboard computer microcontroller // Aerospace Technic and Technology. – 2024. – № 4. – С. 71–86. – DOI: https://doi.org/10.32620/aktt.2024.4.09.

- Liubimov O., Volobujeva L., Turkin I. Experimental evaluation of the efficiency of containerization technology in the onboard software of a nanosatellite CubeSat // Aerospace Technic and Technology. – 2024. – № 4. – С. 87–103. – DOI: https://doi.org/10.32620/aktt.2024.4.10.

8. Звітна документація

Кількість сторінок в звіті: 42

Мова звіту: Українська

Умови поширення в Україні: Не заборонено

Умови передачі іншим країнам: Не заборонено

Кількість файлів у звіті: 1

9. Заключні відомості

Перелік осіб-виконавців

Волобуєва Ліна Олексіївна (к. т. н., доц.)

Делейко Борис Сергійович

Любімов Олександр Вікторович

Керівник організації:

Литвинов Олексій Миколайович (д.ю.н., професор)

Керівники роботи:

Туркін Ігор Борисович (д. т. н., професор)

Керівник відділу реєстрації наукової діяльності УкрІНТЕІ

Юрченко Т.А.

ДОДАТОК В. ТЕСТОВІ ТОПОЛОГІЇ БПЗ

Таблиця В.1 Розроблені та використані в ТПЗ підтопології F’

№	Назва підтопології	Призначення	Рисунок
1	RateGroups	Організація періодичного тактування компонентів	В.2
2	FaultProtection	Передача фатальних подій до обробника помилок	В.3
3	Downlink	Формування даних до наземної станції	В.4
4	Uplink	Прийом і розбір даних із наземної станції	В.5
5	LedConnections	Сигналізація режиму роботи світлодіодом	В.6
6	ComModuleConnections	Адаптер для обміну даними та командами з наземною станцією	В.7
7	CAN0Connections	Інтеграція DroneCAN пристрою з інтерфейсом CAN0	В.8
8	CAN1Connections	Тестування CAN1 периферії	В.9
9	ICM20948Connections	Підключення IMU ICM20948 по I^2C_0	В.10
10	BNO055Connections	Підключення IMU BNO055 по UART4	В.11
11	IMUConnections	Взаємодія з IMU сенсорами	В.12
12	AMCUConnections	Керування і моніторинг допоміжного мікроконтролера AMCU	В.13
13	EnvSensConnections	Збір даних із сенсорів температури та тиску	В.14
14	LoggerConnections	Збір даних для логування	В.15
15	NORConnection	Організація роботи із зовнішньою NOR пам'яттю	В.16
16	RTCConnections	Організація роботи з внутрішнім RTC	В.17
17	FRAM	Організація роботи із зовнішньою FRAM пам'яттю	В.18
18	TPMConnections	Організація роботи з TPM модулем	В.19
19	PowerCtrlAgent	Організація роботи зі збору інформації що до спожитої енергії та регулюванням режимів роботи МК БО	В.20

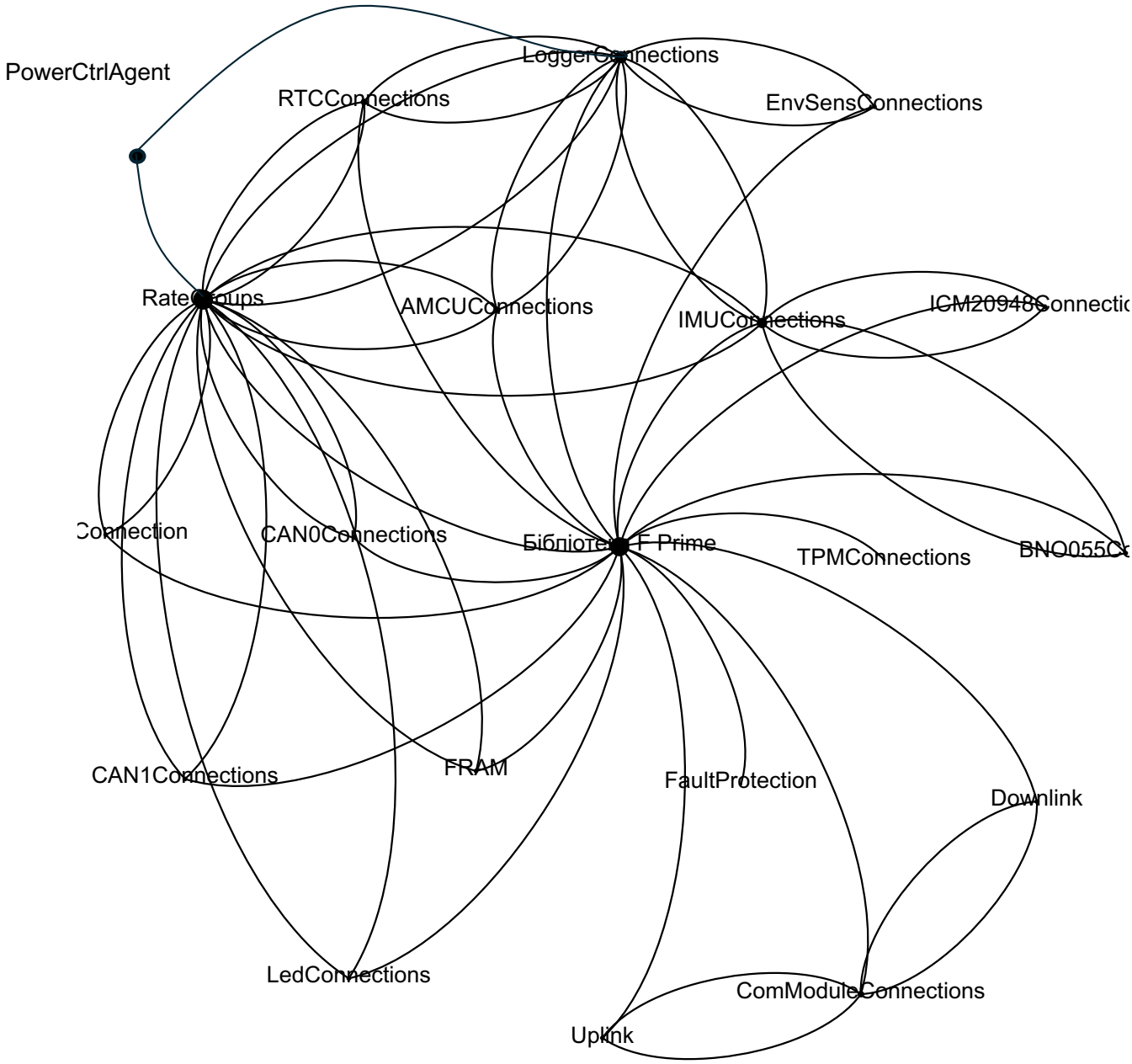

Рисунок В.1 – Підтопології компонентів СПЗ, ТПЗ та зв’язки між ними

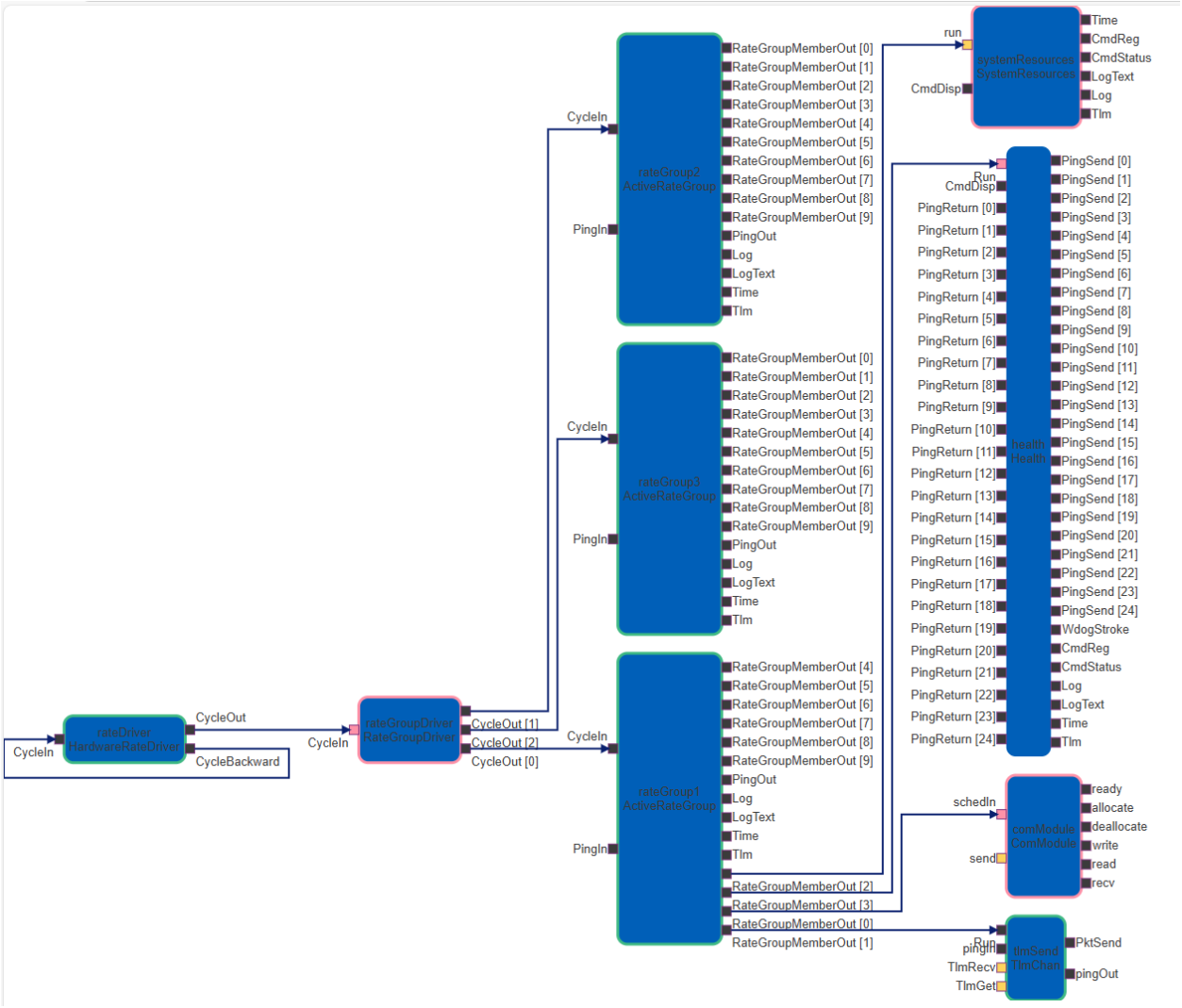

Рисунок В.2 – Підтопологія RateGroups

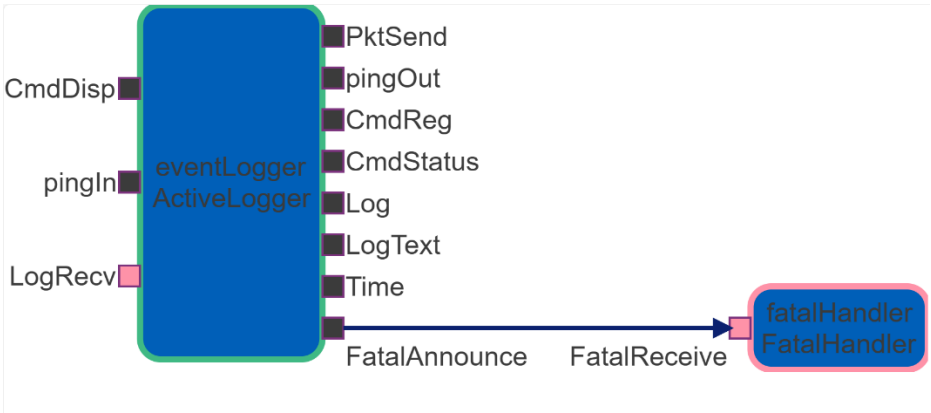

Рисунок В.3 – Підтопологія FaultProtection

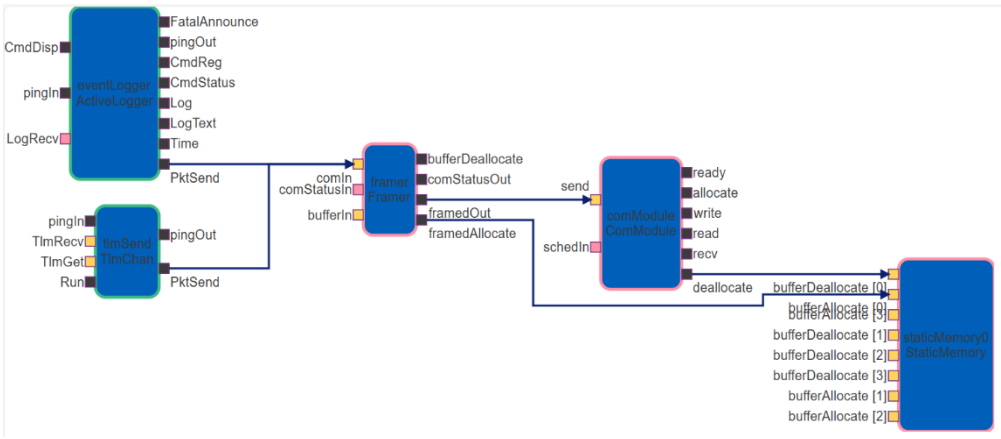

Рисунок В.4 – Підтопологія Downlink

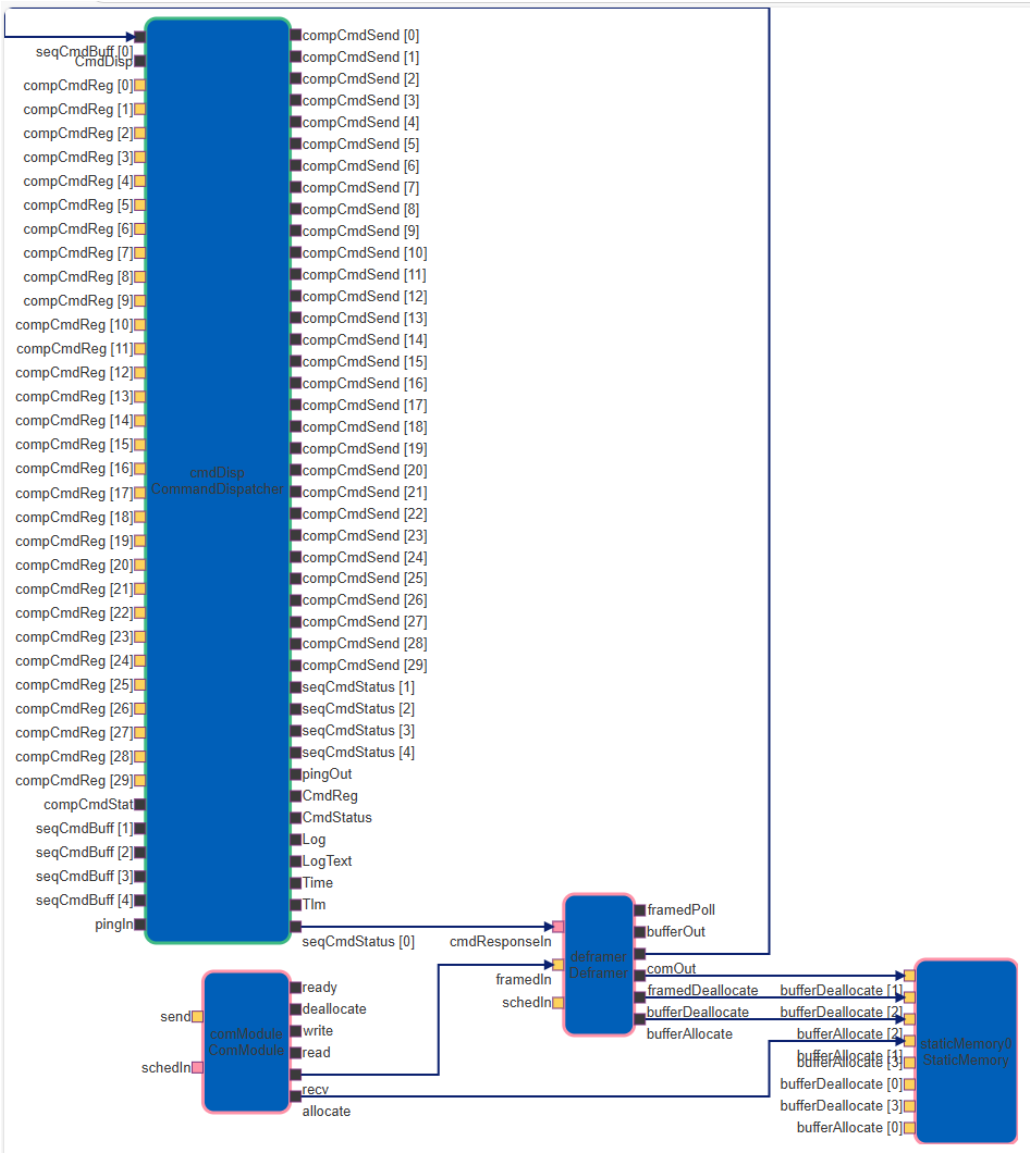

Рисунок В.5 – Підтопологія Uplink

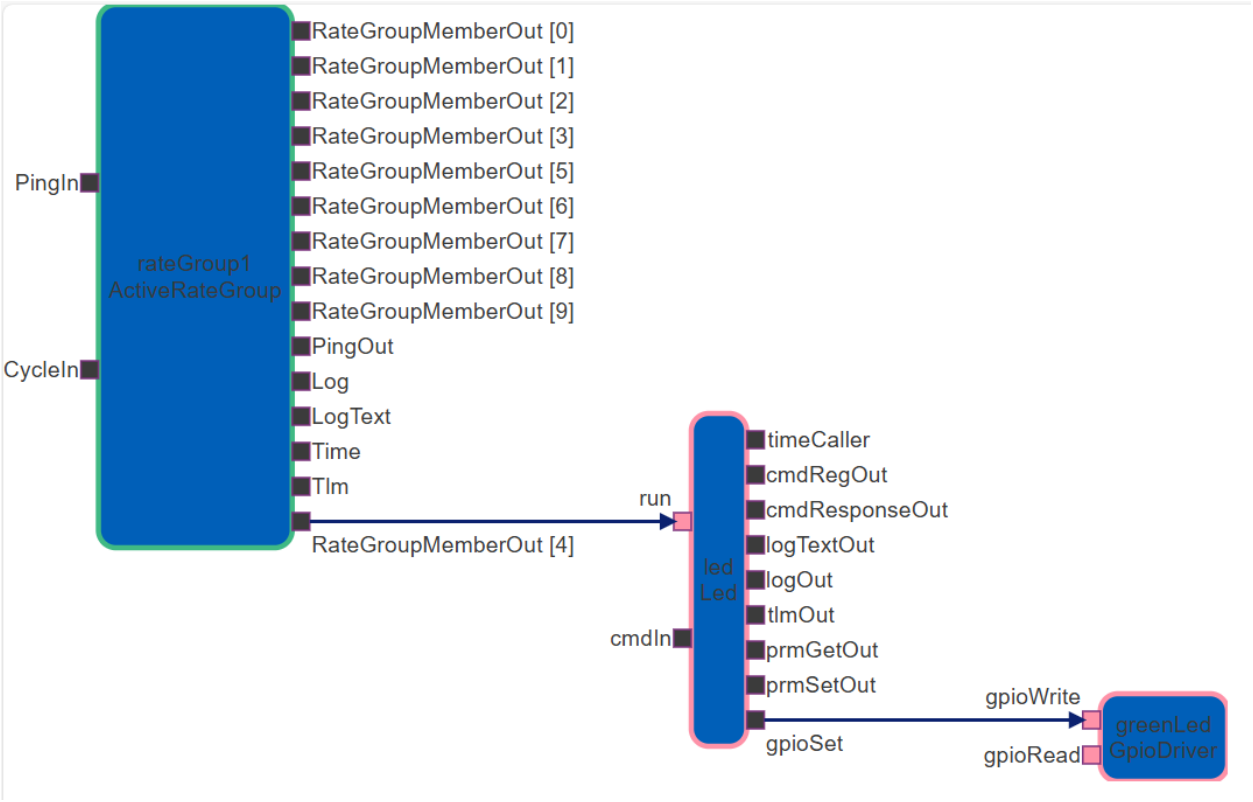

Рисунок В.6 – Підтопологія LedConnections

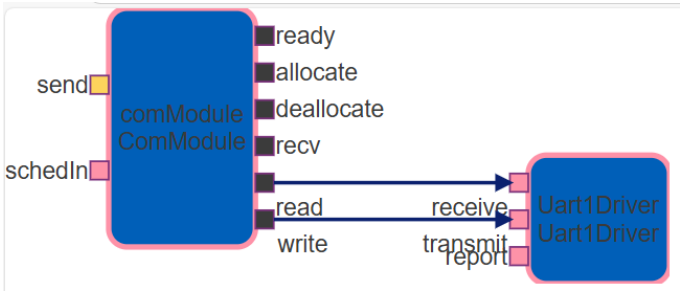

Рисунок В.7 – Підтопологія ComModuleConnections

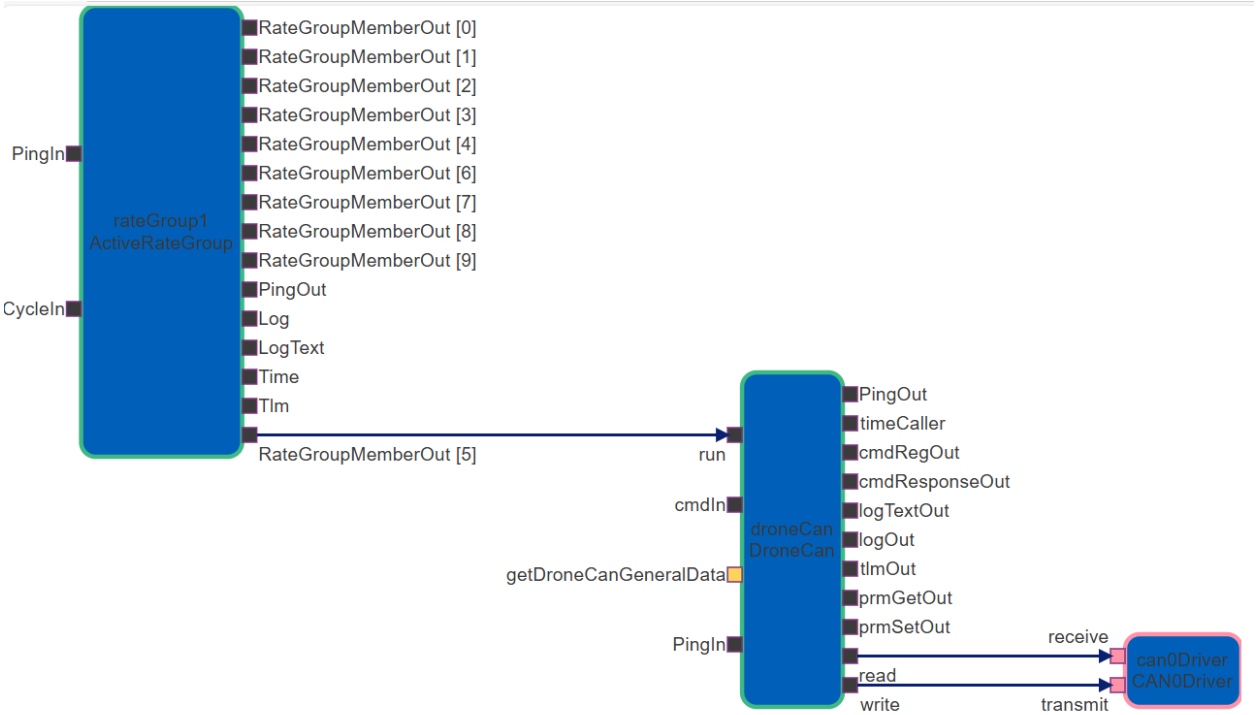

Рисунок В.8 – Підтопологія CAN0Connections

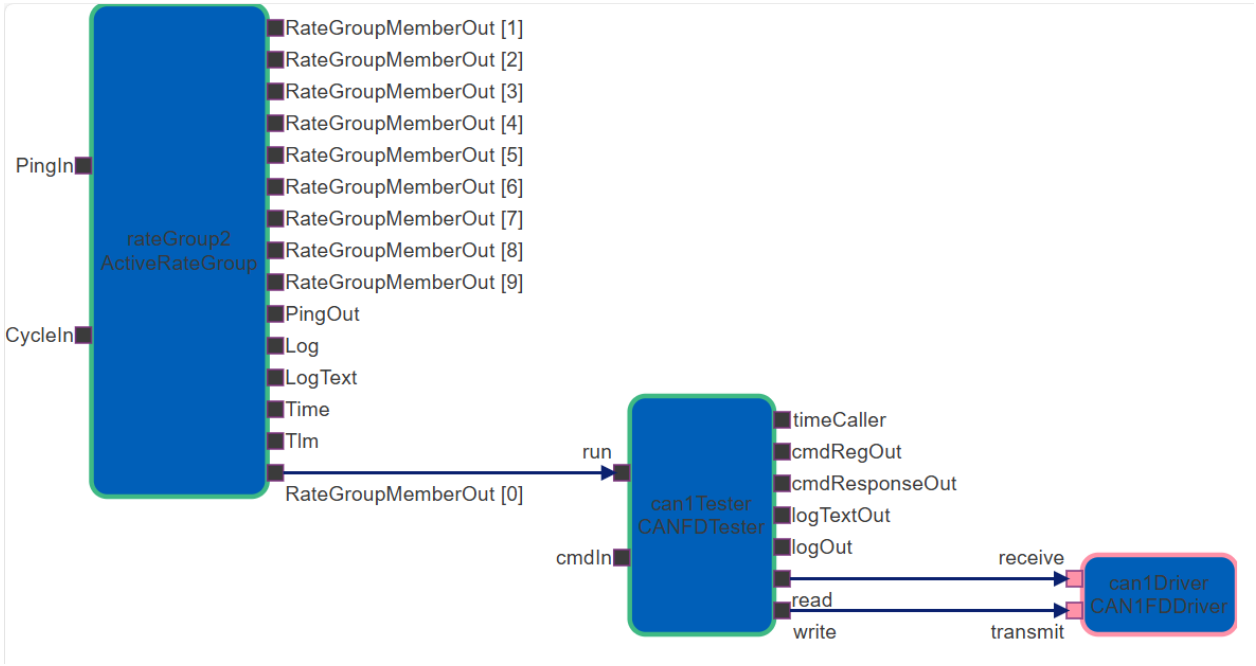

Рисунок В.9 – Підтопологія CAN1Connections

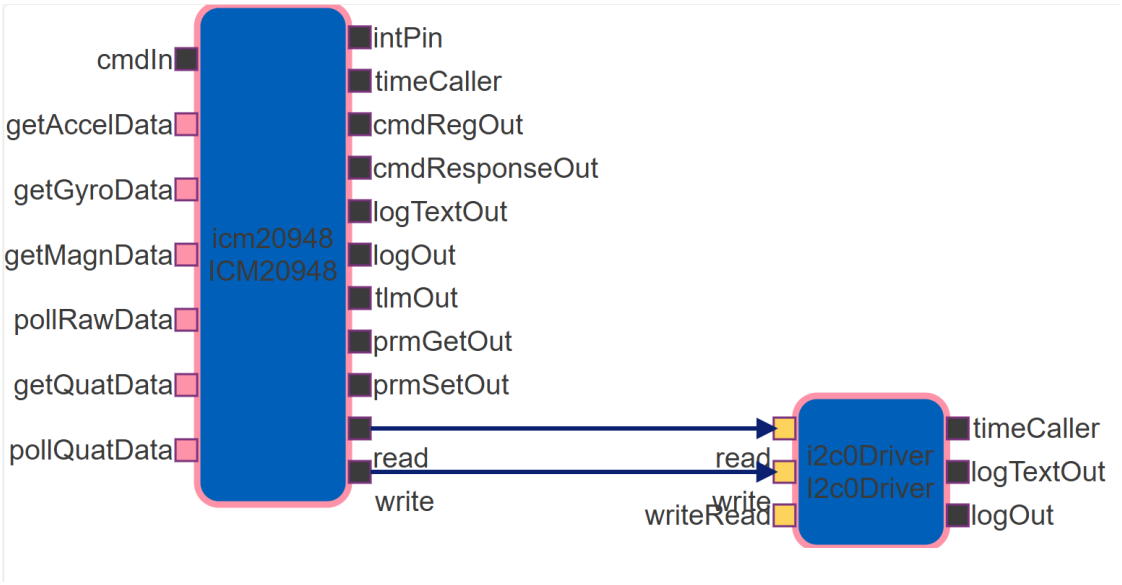

Рисунок В.10 – Підтопологія ICM20948Connections

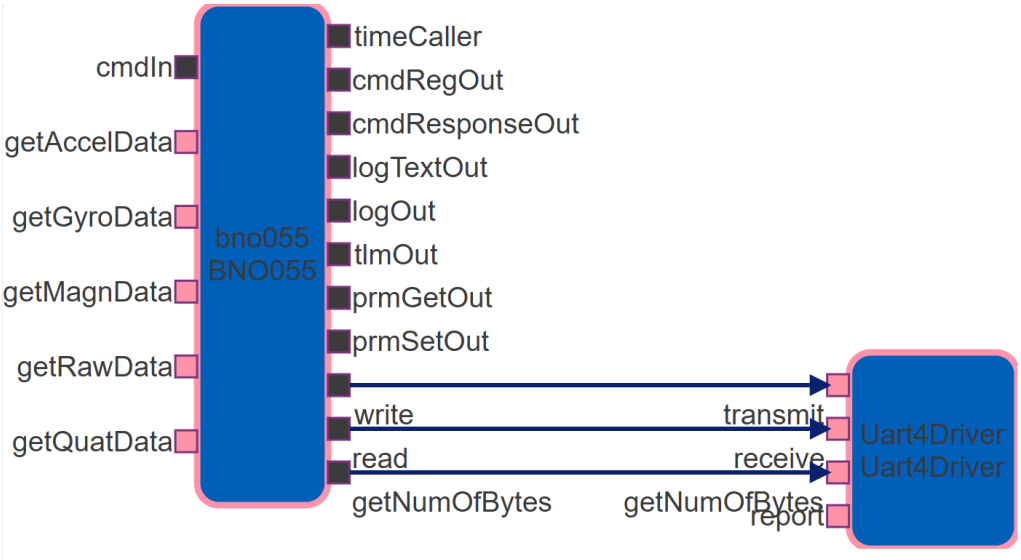

Рисунок В.11 – Підтопологія BNO055Connections

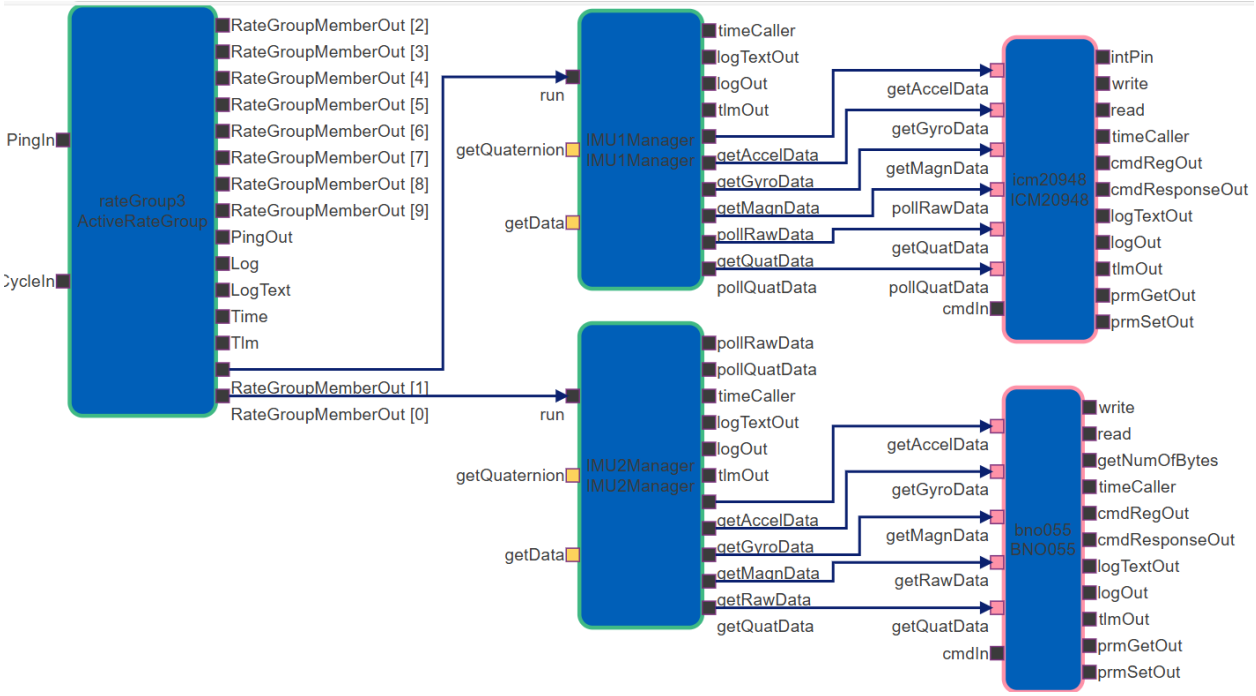

Рисунок В.12 – Підтопологія IMUConnections

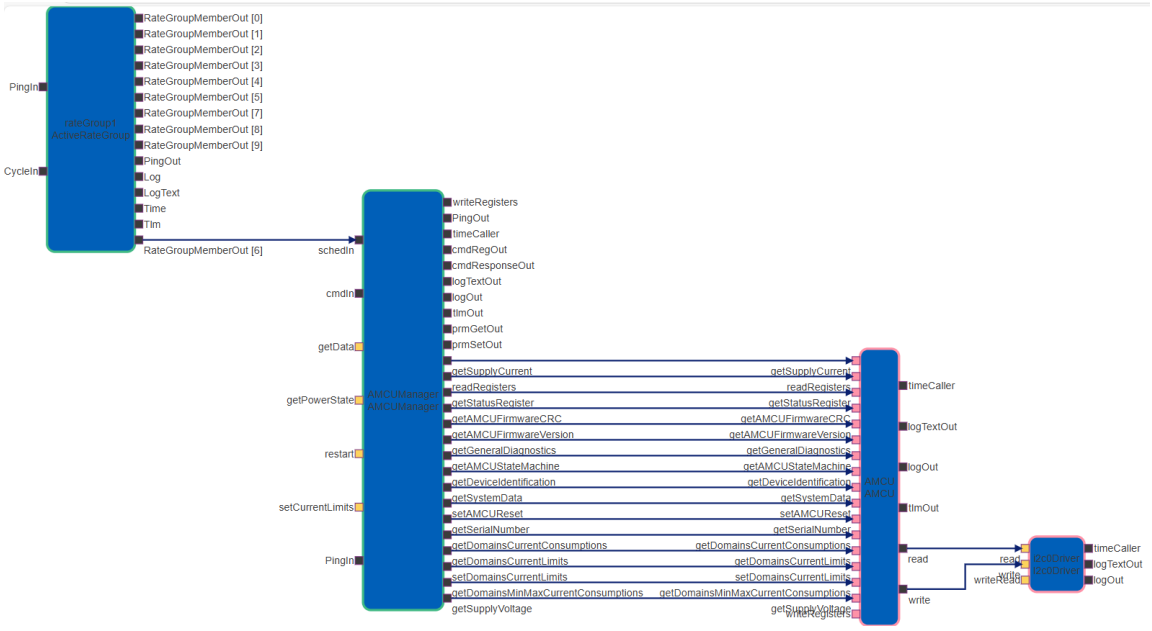

Рисунок В.13 – Підтопологія AMCUConnections

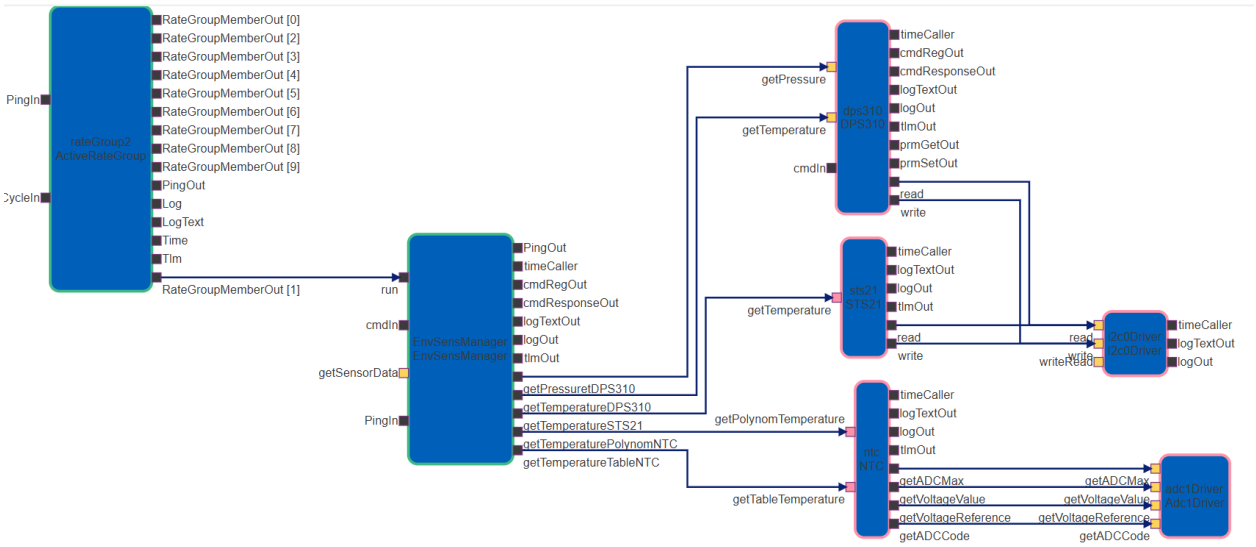

Рисунок В.14 – Підтопологія EnvSensConnections

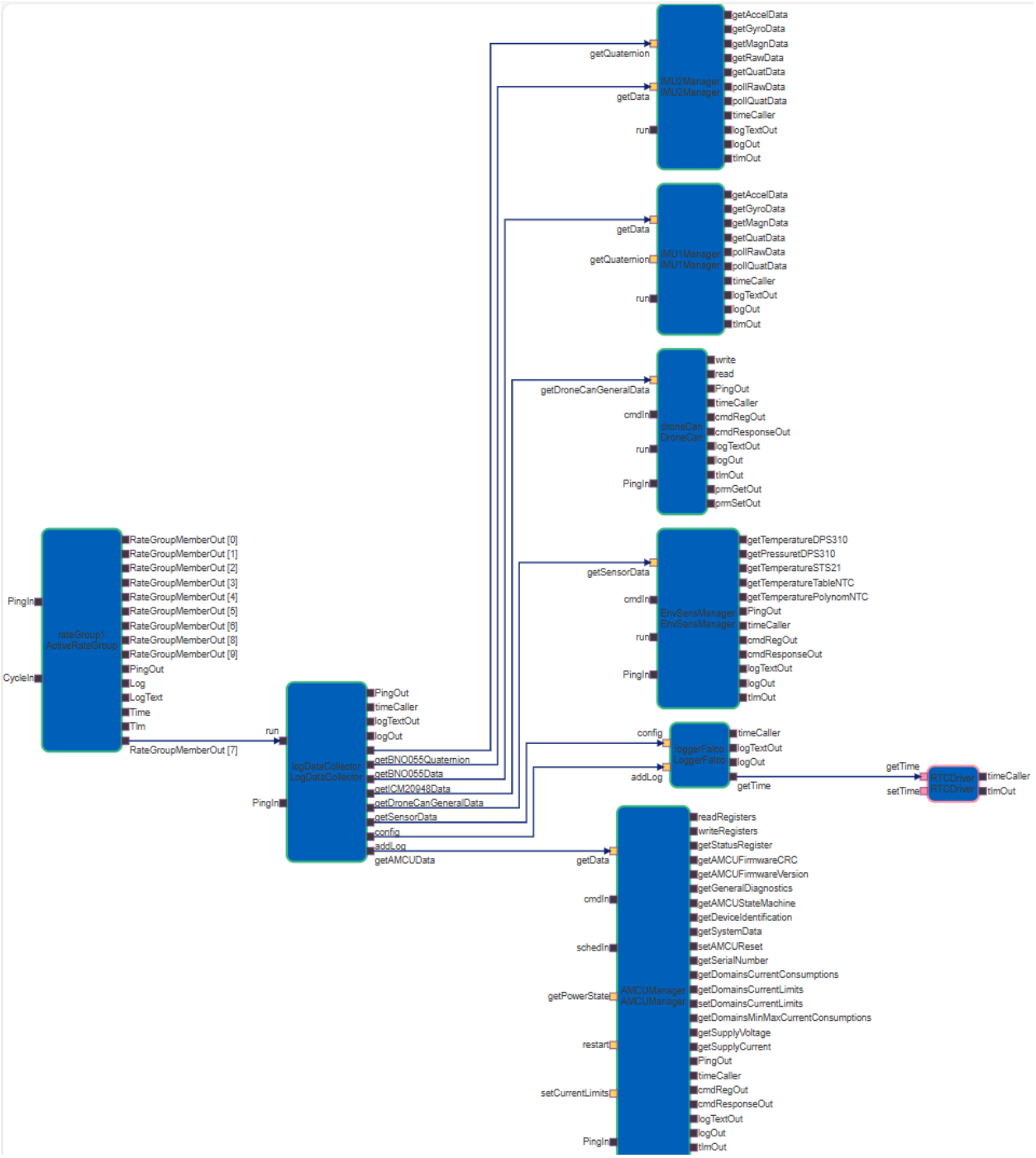

Рисунок В.15 – Підтопологія LoggerConnections

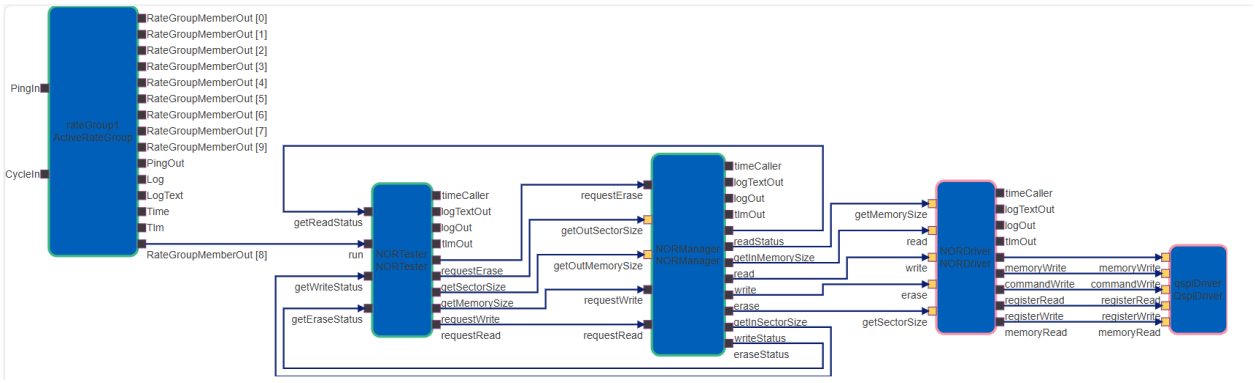

Рисунок В.16 – Підтопологія NORConnection

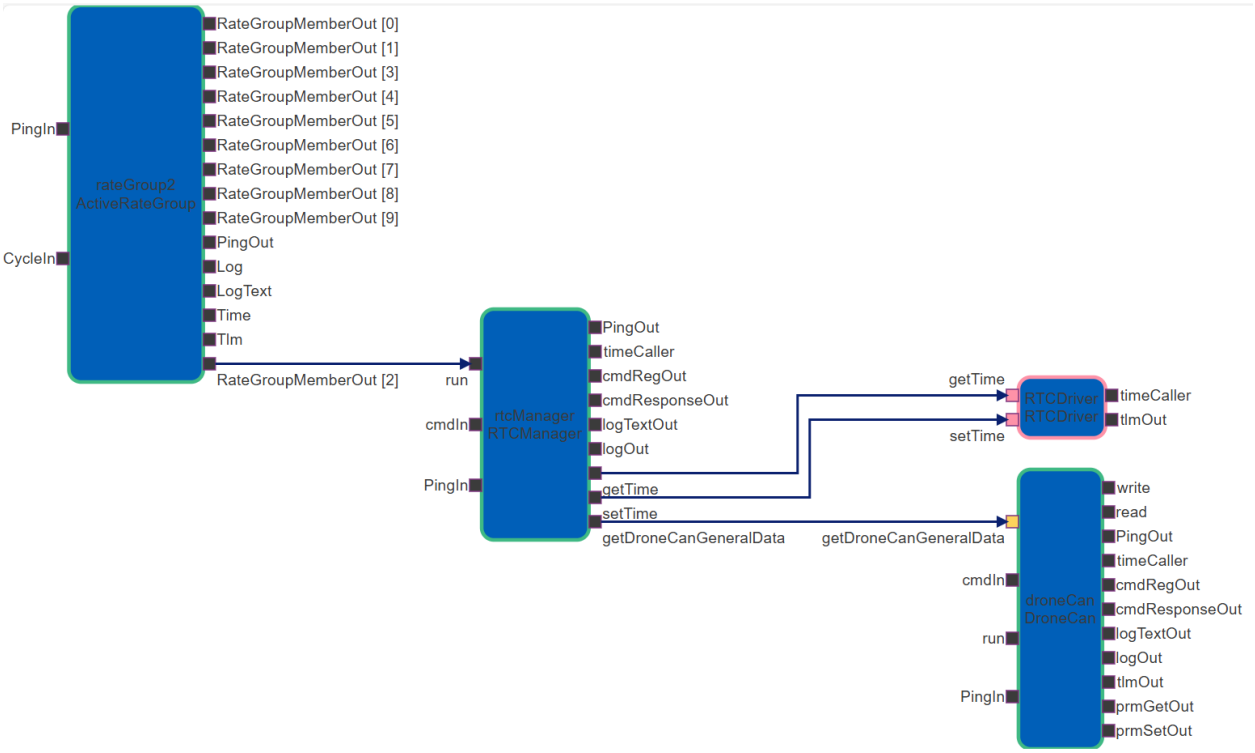

Рисунок В.17 – Підтопологія RTCConnections

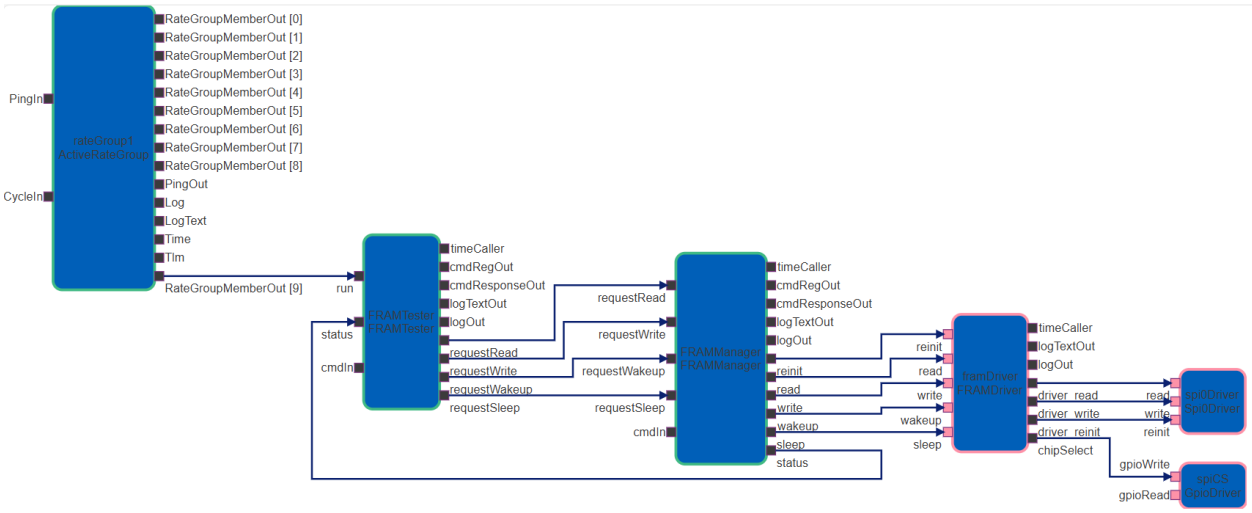

Рисунок В.18 – Підтопологія FRAM

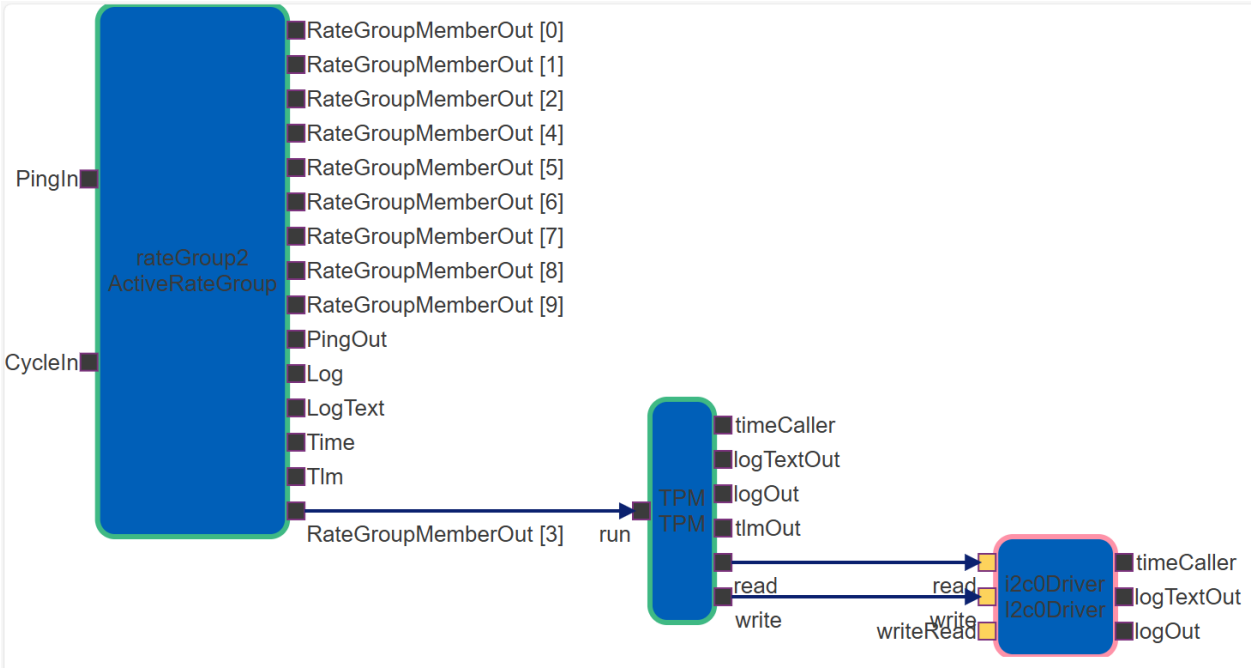

Рисунок В.19 – Підтопологія TPMConnections

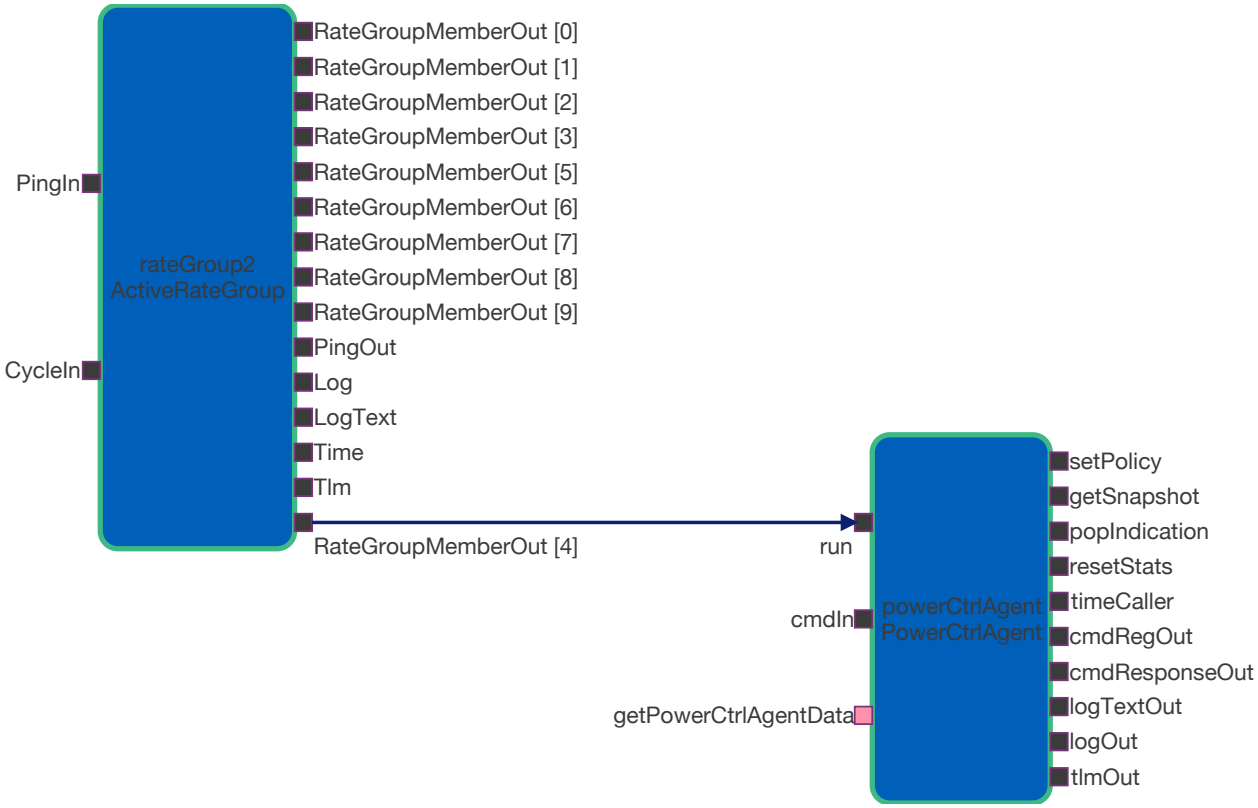

Рисунок В.20 – Підтопологія PowerCtrlAgent