МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АЕРОКОСМІЧНИЙ УНІВЕРСИТЕТ
ІМ. М. Є. ЖУКОВСЬКОГО «ХАРКІВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Кваліфікаційна наукова
праця на правах рукопису

ВДОВІЧЕНКО ОЛЕКСАНДР ОЛЕКСАНДРОВИЧ

УДК 004.02:004.38 - 004.052:004.94

ДИСЕРТАЦІЯ

Моделі, метод та засоби реконфігурації програмовних пристроїв безпілотних систем з керованою багаторівневою деградацією

Спеціальність 123 Комп’ютерна інженерія
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. О. Вдовіченко

Науковий керівник (консультант)
Харченко Вячеслав Сергійович, доктор технічних наук, професор

Харків – 2025

АНОТАЦІЯ

Вдовіченко Олександр Олександрович. Моделі, метод та засоби реконфігурації програмовних пристроїв безпілотних систем з керованою багаторівневою деградацією – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 Комп'ютерна інженерія. – Національний аерокосмічний університет імені М.Є. Жуковського «Харківський авіаційний інститут», Харків, 2025.

Дисертаційна робота присвячена розробленню методів і програмних засобів оцінювання та забезпечення надійності програмовних пристроїв безпілотних апаратів в умовах накопичення відмов шляхом впровадження процедур керованої багаторівневої деградації внутрішніх ресурсів.

Об'єктом дослідження є процеси забезпечення надійності систем безпілотних апаратів та їх програмовних пристроїв.

Вперше запропоновано метод керованої багаторівневої деградації програмовних пристроїв, який на відміну від відомих базується, по-перше, на формальному описі умов та аналізі реконфігуропридатності архітектури за допомогою спеціальних метрик об'єму та часу; по-друге, запропонованих процедурах реконфігурації, вибір і реалізація яких здійснюється залежно від типів дефектів з використанням визначеної множини конфігурацій, що забезпечує підвищення надійності та живучості при виникненні та накопиченні відмов апаратних компонентів.

Удосконалено структурні та надійнісні моделі програмовних пристроїв з керованою багаторівневою деградацією шляхом декомпозиції множин внутрішніх компонентів залежно від впливу на працездатність та потенційної можливості програмно-керованої реконфігурації структури без втрати або з частковою втратою працездатності і якості виконання специфікованих функцій, що надає змогу формувати вимоги до засобів керування реконфігурацією та розраховувати показники безвідмовності та живучості при відмовах.

Удосконалено марковські моделі готовності програмовних пристроїв з керованою деградацією, що враховують інтенсивності відмов визначених множин компонентів залежно від можливості їх відновлення завдяки програмно-керованій реконфігурації без втрати або з частковою втратою якості виконання функцій, а також часові і достовірнісні характеристики процедур перебудови, що забезпечує можливість розрахунку функцій готовності при багатоступеневій деградації.

Ключові слова: оброблення та стиснення зображень, марковський процес, моделювання, надійність, живучість, класифікація, часові метрики, оптимальні значення параметрів, Інтернет речей, комп'ютерні системи та програмовні пристрої, безпілотні апарати, контроль і діагностика, прийняття рішень, реконфігуропридатність і реконфігурація, багатостанова система.

Список публікацій здобувача:

Вдовіченко, О., Харченко, В. Програмовні пристрої з керованою багаторівневою деградацією: моделі, методи реконфігурації та аналізу реконфігуропридатності. Електронне моделювання. – 2025 – No. 47(1) – С. 53–76, DOI: 10.15407/emodel.47.01.053.

Вдовіченко, О. Харченко, В. Марковські моделі оцінювання надійності програмовних пристроїв з керованою багаторівневою деградацією: класифікація, розроблення і дослідження. Measuring and computing devices in technological processes. – 2024 – No.4, – С. 433–444. DOI: 10.31891/2219-9365-2024-80-54.

Вдовіченко, О., Перепелицин, А. Аналіз номенклатури мікроконтролерів для побудови елементів домашньої автоматизації. Авіаційно-космічна техніка і технологія. – 2024. – No. 5(199) – С. 118–126. DOI: 10.32620/aktt.2024.5.12.

Вдовіченко, О. Аналіз технологій реконфігурації систем Інтернету речей на рівні програмних модулів та завантажувачів. Авіаційно-космічна техніка і технологія. – 2024. – No. 3(195) – С. 99–108. DOI: 10.32620/aktt.2024.3.09.

Вдовіченко, О., Перепелицин, А. Організація взаємодії пристроїв з доступом в Інтернет на основі мікроконтролерів із обмеженою кількістю ресурсів. Авіаційно-космічна техніка і технологія. – 2023. – No. 6(192) – С. 76–85. DOI: 10.32620/aktt.2023.6.09.

Vdovichenko, O., Perepelitsyn, A. Analysis of Technologies for Reconfiguration of IoT Systems at Level of Software Modules and Bootloaders. Integrated Computer Technologies in Mechanical Engineering - 2023. ICTM 2023. Lecture Notes in Networks and Systems, vol 996. Springer, Cham, Germany, 2023, P 474–486. DOI: 10.1007/978-3-031-60549-9_36.

Perepelitsyn, A., Vdovichenko, O. Technologies and Services of Communication for Embedded Systems over Internet. 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 2023, P. 1–6. DOI: 10.1109/DESSERT61349.2023.10416486.

Perepelitsyn, A., Vdovichenko, O., Mikhalevskyi, V. Service for communication of devices with Internet access: analysis of technologies and method of creation. Radioelectronic and Computer Systems. – 2023. – No. 4(108). – P. 197–208. DOI: 10.32620/reks.2023.4.14.

Вдовіченко О. О. Засоби автоматизації розроблення програмного забезпечення [Текст] / О. О. Вдовіченко // 14 Студентська науково-технічна конференція Перспективні мережні та комп'ютерні технології (ПерСиК 2023). – С. 50. ISBN 978-617-8238-34-6.

Perepelitsyn, A., Duzhyi, V., Vdovichenko, O., Zheltukhin, O. Technologies of Embedded Systems Prototyping using Reconfigurable Nodes: Technical Solutions. 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 2022, P. 1–6. DOI: 10.1109/DESSERT58054.2022.10018581.

Вдовіченко, О., Перепелицин, А., Дужий, В., Желтухін, О. Метод дистанційної діагностики, перепрограмування і реконфігурації вузлів вбудованої системи. Авіаційно-космічна техніка і технологія. – 2022. – No. 6(184). – С. 66–75. DOI: 10.32620/aktt.2022.6.08.

Vdovichenko, O., Perepelitsyn, A. Technologies for building systems of remote lining of communication lines: a practical example of implementation. Radioelectronic and Computer Systems. – 2021. – No. 2(98). – P. 31–38. DOI: 10.32620/reks.2021.2.03.

ABSTRACT

Oleksandr Vdovichenko. Models, method and means of reconfiguration for programmable devices of unmanned systems with controlled multilevel degradation – Manuscript copyright.

Dissertation on competition for scientific degree of Doctor of Philosophy by specialty 123 Computer Engineering. – National Aerospace University “Kharkiv Aviation Institute”, Kharkiv, 2025.

The dissertation is devoted to the development of methods and software tools for estimating and ensuring the reliability of software devices of unmanned vehicles in the conditions of cumulative failures by implementing procedures for controlled multilevel degradation of internal resources.

The object of research is the processes of ensuring the reliability for unmanned vehicles systems and their programmable devices.

Firstly, a method of controlled multi-level degradation of programmable devices is proposed, which, as opposed to the known ones, is based, firstly, on a formal description of the conditions and analysis of the architecture reconfigurability using special capacity and time metrics; secondly, on the proposed reconfiguration procedures, the selection and implementation of which is carried out depending on the types of defects using a certain set of configurations, which ensures increased reliability and survivability in the event of occurrence and accumulation of hardware component failures.

The structural and reliability models of software devices with controlled multilevel degradation are improved by decomposing sets of internal components depending on the impact on the performance and the potential for software-controlled reconfiguration of the structure without loss or with partial loss of performance and quality of the specified functions, which makes it possible to formulate requirements for reconfiguration control tools and calculate the reliability and survivability indicators for certain failures.

The Markov models of availability for programmable devices with controlled degradation are improved, which take into account the failure rates of certain sets of components depending on the possibility of recovery through software-controlled reconfiguration without or with partial loss of function quality, as well as the time and reliability characteristics of reconstruction procedures, which makes it possible to calculate availability functions in case of multi-level degradation.

Keywords: image processing and compression, Markov process, modeling, reliability, survivability, classification, time metrics, optimal parameter values, Internet of Things, computer systems and software devices, unmanned vehicles, monitoring and diagnostics, decision making, reconfigurability and reconfiguration, multistate system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

PLD - Programmable Logic Device

PAL - Programmable Array Logic

UAL - Uncommitted Array Logic

GAL - Generic Array Logic

CPLD - Complex Programmable Logic Device

FPGA - Field-Programmable Gate Array

MCU - Microcontroller Unit

MSS – Multi State System

ПП - Програмовні пристрої

БПС – Безпілотні системи

БРД - Багаторівнева деградація

ВСТУП

Обґрунтування вибору теми дослідження. Незважаючи на різноманіття апаратних рішень для побудови безпілотних апаратів(БПА) та відповідних систем(БПС), їх ресурсні, надійнісні та часові характеристики залишаються обмеженими з точки зору толерування відмов. Для необслуговуваних систем довготривалого використання або пристроїв, які експлуатуються в умовах потужних кіберфізичних впливів, що збільшують інтенсивність збоїв і відмов, необхідні засоби супроводу та підтримки працездатності, що дозволяють відтерміновувати перехід до повної (фатальної) відмови або аварійного стану вже розгорнутих систем.

Однак, можливість такої підтримки може бути обмежена через умови експлуатації, що потребує завчасного розроблення і використання засобів, які б забезпечили її продовження, навіть при частковій працездатності завдяки програмно-апаратній реконфігурації структури. Для цього мають бути оцінені потенційна реконфігуропридатність використованого пристрою, розроблені та впроваджені відповідні засоби реконфігурації.

Використання програмовних пристроїв(ПП) при побудові БПА обумовлено їх гнучкістю і програмовністю структури, що забезпечує конкурентність на ринку електронних компонентів. Застосування ПП в критичних системах з високою ціною відмов також прискорилося завдяки показникам надійності та створенню сертифікованих рішень за вимогами функційної безпечності, що певною мірою розвантажує складні процеси розроблення, верифікації та сертифікації платформних рішень для інформаційно-керуючих систем, зокрема, в атомній енергетиці.

Відповіддю на такі виклики стало формування класу пристроїв і систем з багаторівневою деградацією (БРД), особливість котрих полягає у погіршенні характеристик ПП або невиконання ним частини функцій. Завдяки притаманній ПП гнучкості, БРД може бути контрольованою завдяки процедурам реконфігурації.

Таким чином, актуальною науково-прикладною задачею є розроблення моделей, методів і засобів аналізу та забезпечення надійності програмовних пристроїв безпілотних апаратів шляхом впровадження процедур керованої багаторівневої деградації.

Об'єкт дослідження – процеси забезпечення надійності систем безпілотних апаратів (БПА) та їх програмовних пристроїв (ПП).

Предмет дослідження – моделі, методи та засоби забезпечення надійності ПП БПА в умовах багаторівневої деградації.

Мета і завдання дослідження. Метою дослідження є підвищення надійності ПП БПА шляхом розроблення і впровадження методів і засобів її оцінювання та забезпечення в умовах накопичення відмов і багаторівневої деградації внутрішніх ресурсів.

Для досягнення мети дослідження необхідно вирішити наступні завдання:

- проаналізувати існуючі методи і технології забезпечення надійності програмовних пристроїв безпілотних систем з багаторівневою деградацією при відмовах внутрішніх компонентів. Обґрунтувати задач і методики досліджень;
- розробити моделі надійності програмовних пристроїв з керованою багаторівневою деградацією при відмовах компонентів;
- розробити метод реконфігурації та оцінювання реконфігуропридатності програмовних пристроїв з керованою багаторівневою деградацією;
- розробити марковські моделі готовності програмовних пристроїв з керованою багаторівневою деградацією;
- розробити послідовність оцінювання та засоби забезпечення надійності і живучості програмовних пристроїв при створенні та модернізації безпілотних систем різних типів;
- впровадити розроблені методи і програмно-апаратні засоби забезпечення надійності програмовних пристроїв безпілотних систем.

Методи дослідження. У дисертаційній роботі використовувались методи системного аналізу, оптимізації, математичного моделювання, теорії надійності, теорії графів.

Наукова новизна отриманих результатів:

- **вперше запропоновано** *метод* керованої багаторівневої деградації програмовних пристроїв, який, на відміну від відомих, базується, по-перше, на формальному описі умов та аналізі реконфігуропридатності архітектури за допомогою спеціальних метрик об'єму та часу; по-друге, запропонованих процедурах реконфігурації, вибір і реалізація яких здійснюється залежно від типів дефектів з використанням визначеної множини конфігурацій, що забезпечує підвищення надійності та живучості при виникненні та накопиченні відмов апаратних компонентів;
- **удосконалено** структурні та надійнісні *моделі* програмовних пристроїв з керованою багаторівневою деградацією шляхом декомпозиції множин внутрішніх компонентів залежно від впливу на працездатність та потенційної можливості програмно-керованої реконфігурації структури без втрати або з частковою втратою працездатності і якості виконання специфікованих функцій, що надає змогу формувати вимоги до засобів керування реконфігурацією та розраховувати показники безвідмовності та живучості при певних відмовах;
- **удосконалено** марковські *моделі* готовності програмовних пристроїв з керованою деградацією, що враховують інтенсивності відмов множин компонентів, визначених залежно від можливості відновлення завдяки програмно-керованій реконфігурації без (або) з частковою втратою якості виконання функцій, часові і достовірнісні характеристики процедур перебудови, а також наявність резервування та часткову працездатність засобів реконфігурації, що забезпечує можливість розрахунку функцій готовності при багатоступеневій деградації.

Особистий внесок здобувача полягає у розроблені методів та програмних засобів керування потенційною реконфігуропридатністю програмовних пристроїв

безпілотних систем, та моделі оцінювання їх готовності під час керованої багаторівневої деградації. [1]-[12].

У працях, що опубліковані як індивідуально, так і у співавторстві, автору належать:

- результати аналізу методів взаємодії обчислювальних вузлів вбудованих систем на предмет можливості їх діагностики та дистанційного перепрограмування [1, 2];

- аналіз існуючих типів компонентів побудови IoT систем з оглядом на їх придатність до модифікації [3, 4];

- програмний засіб для впровадження запропонованого методу реконфігурації до програмовних пристроїв з багатоступеневою керованою деградацією [5, 6];

- аналіз можливості використання програмних блоків для модифікації програм мікроконтролера в рамках процедури реконфігурації [7];

- класифікацію вбудованих систем за рівнем організації їхньої реконфігурованої частини [8];

- спрощений процес прийняття рішень щодо вибору елементної бази для побудови компонентів системи домашньої автоматизації [9];

- класифікатор моделей, а також результати розроблення та дослідження для чотирьох ключових мультифрагментних марковських моделей [10];

- структурні схеми надійності, аналітичні моделі та низку процедур реконфігурації та послідовність аналізу реконфігуропридатності програмовних пристроїв для різних сценаріїв багаторівневої деградації [11];

- прототип апаратного засобу з впровадженням запропонованого методу реконфігурації [12];

Апробація матеріалів дисертації. Основні положення та ідеї дисертаційної роботи доповідалися та обговорювалися на: науково-технічних конференціях: "Integrated Computer Technologies in Mechanical Engineering" (м. Харків, 2021, 2022, 2023, 2024); "Dependable System, Services and Technologies Conference

(Athens, Greece 2022, 2023, 2024); “Матеріали НТК “Перспективні Мережеві і Комп'ютерні технології (ПерСиК)(2017, 2023)” (Харків, Україна, 2017-2023 рр.).

Зв’язок з науковими програмами, темами. Дисертаційна робота виконана у Національному аерокосмічному університеті ім. М. Є. Жуковського «Харківський авіаційний інститут» відповідно з державними програмами та планами НДР:

- НДР «Методологічні засади та технології оцінювання та забезпечення безпеки (захисту) критичних інформаційних інфраструктур» (№ ДР 0119U100979, 2019–2021 рр.);
- НДР «Розроблення науково-методичного апарату і засобів для оцінювання та забезпечення надійності та безпечності гарантоздатних ІТ-систем та інфраструктур» (№ ДР 0121U113842, 2021–2023 рр.);
- НДР «Наукові засади і методи забезпечення гарантоздатності флотів БПЛА інтелектуальних систем моніторингу потенційно небезпечних і військових об’єктів» (Д 503-1/2021-Ф, № Д/Р 0121U112172, 2021-2023 рр.);
- НДР «Методи, програмно-апаратні засоби та технології забезпечення гарантоздатності інтелектуальних систем індустріального Інтернету речей» (Д 503-10/2022-П, № Д/Р 0122U001065, 2022-2023);
- НДР «Методи та кейс-технології доказового оцінювання кібербезпеки програмовних систем для забезпечення захисту критичної ІТ-інфраструктури» (Д 503-10/2023-П, № ДР 0123U102106, 2023-2024).

Роль автора у зазначених НДР, в яких автор був безпосереднім виконавцем, полягає у розробці методів та засобів оцінювання та забезпечення реконфігуропридатності в умовах накопичення відмов і багаторівневої деградації внутрішніх ресурсів.

Практичне значення отриманих результатів. Практичні результати полягають у доведенні теоретичних положень дисертаційної роботи до конкретних методів та програмних засобів для підтримки керованої БРД в БПА на основі ПП. Результати дисертаційної роботи впроваджено (додаток А):

- у навчальному процесі Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут» (акт впровадження від 07 березня 2025);
- при виконанні науково-дослідних проєктів, що виконувались у Національному аерокосмічному університеті ім. М. Є. Жуковського «Харківський авіаційний інститут» (акт впровадження від 07 березня 2025).

Структура та обсяг дисертації. Дисертація складається із вступу, чотирьох розділів, висновку, списку виконаних джерел і додатків. Загальний обсяг дисертації складає 171 сторінки, з яких анотація двома мовами на 5 сторінках, зміст на 4 сторінках, перелік умовних позначень на 1 сторінці, основний текст на 120 сторінках, список використаних джерел із 101 найменування на 13 сторінках, додатки на 27 сторінках. Робота містить 11 таблиць та 67 рисунків.

Публікації. За темою дисертаційної роботи було опубліковано 11 наукових публікацій, включно:

- 4 статті у наукових фахових виданнях України (2 статті у виданні з індексацією у Scopus, квартиль Q3);
- 1 стаття опублікована у періодичному виданні Springer (має ISSN та DOI з індексацією у Scopus, квартиль Q4);
- 1 колективна монографія;
- 4 статті у науковому фаховому виданні України (не покриває спеціальність 123);
- 2 публікації у просідінгах міжнародних конференцій з індексацією у Scopus;
- 2 публікації у матеріалах національної конференції.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ЗАБЕЗПЕЧЕННЯ НАДІЙНОСТІ ПРОГРАМОВНИХ ПРИСТРОЇВ БЕЗПІЛОТНИХ СИСТЕМ. ЗАДАЧІ І МЕТОДИКА ДОСЛІДЖЕННЯ

1.1 Аналіз програмовних пристроїв безпілотних систем

Подібно до інших вбудованих систем, БПА потребує обчислювальної платформи у якості системи обробки, яка отримує дані від корисного навантаження та інших модулів[13,14]. Надалі, частково або повною мірою оброблена інформація передається до наземної станції або іншого БПА через пристрої передачі інформації. Більшість обчислювальних платформ комерційних і цивільних БПА являють собою вбудовані системи на базі МК або FPGA-технологій[15]. Однак перед сучасними БПА виникають завдання комплексного оброблення зображень та виявлення об'єктів у реальному часі, що передбачає використання високошвидкісних багатоядерних процесорних системи, графічних процесорів (GPU), ПЛІС або програмованих систем на кристалі у якості віддалених(виділених) засобів[16]. Відповідно, необхідною задачею є вивчення різновидів обчислювальних платформ та їх реалізації у вигляді програмовних пристроїв[17].

1.1.1 Класифікація програмовних пристроїв

Програмовні пристрої(ПП) завдяки своїй доступності та гнучкості підходу до розроблення проектів для вбудованих рішень та систем Інтернету речей різних масштабів[18], набули значної популярності там, де реконфігуропридатність забезпечує можливості еволюціонування в реальному часі та відмовостійкості. Склад та архітектура ПП можуть суттєво відрізнятися, залежно від застосування, однак, незмінною є потенційна здатність до реконфігурації[19, 20]. За методом реалізації компонентів класифікуються такі пристрої, як програмовні логічні

інтегральні схеми, програмовані мікроконтролери, та їх поєднання у вигляді реалізованих на мікросхемах програмовної логіки компонентів апаратного забезпечення [21, 22].

Програмовні логічні інтегральні схеми (Programmable Logic Devices, далі PLD). ПП даного класу використовуються для побудови реконфігурованих цифрових схем [23], які можна запрограмувати для реалізації логічних функцій:

- програмовні матриці логіки (Programmable Array Logic, далі PAL) – тип PLD з фіксованими з'єднаннями його складових з можливістю їх програмування у вигляді логічних блоків. В таких пристроях реконфігурація при відмовах обмежена такими блоками [21];
- некомутовані матриці логіки (Uncommitted Array Logic, далі ULA) – тип PLD з фіксованими набором вже готових логічних блоків з можливістю налаштування з'єднань між ними. Для цих ПП можлива реконфігурація зв'язків для оминання логічних блоків з відмовами [22, 24];
- узагальнені матриці логіки (Generic Array Logic, далі GAL) – тип PLD з фіксованими з'єднаннями його складових з можливістю їх програмування та перезапису у вигляді логічних блоків. Можливості реконфігурації при відмовах аналогічні PAL [23, 25];
- комплексні логічні інтегральні схеми (Complex Programmable Logic Device, далі CPLD) – пристрої, що складаються з декількох PAL, призначені для масштабованих рішень та ускладнених задач. В таких ПП можливості реконфігурації для збереження повної або обмеженої працездатності визначаються на рівнях PAL та їхніх, більш складних конфігурацій;
- програмовні вентильні матриця (Field-Programmable Gate Array, далі FPGA) - більш потужні та гнучкі пристрої, що складаються з конфігурованих логічних блоків та зв’язків між ними, придатних до багаторазового переналаштування. Така технологія забезпечує найбільші можливості реконфігурації при відмовах на рівні логічних елементів і модулів [26, 27].

Програмовні мікроконтролери (Microcontroller Unit, далі MCU). Такі ПП виконані у вигляді мікросхем спеціалізовані комп'ютери, що включають

мікропроцесор, оперативну та постійну пам'ять для збереження виконуваного коду програм і даних, порти вводу-виводу і блоки зі спеціальними функціями. Можуть бути налаштовані для виконання різних програмних завдань, таких як керування іншими пристроями, оброблення інформації чи прийом та передача даних [28]. На даний момент мікроконтролерні пристрої мають високу ступінь залежності від виробника. Тому, відповідно до компаній виробників можна розглядати пристрої, надані нижче.

Microchip Technology та підпорядковані ним **Atmel** виробляють пристрої пам’яті EEPROM, інтегральні схеми Flash-IP, радіочастотні (РЧ) пристрої, симетричні та асиметричні мікросхеми безпеки CryptoAuthentication, сенсорні датчики та мікроконтролери наступних типів [29]:

- енергоекономні AVR 8-розрядні та 32-розрядні;
- універсальні ARM 32-розрядні;
- автомобільні Intel 8051 8-розрядні;
- малопотужні PIC, dsPIC;
- потужні SAM на базі ARM Cortex.

Infineon Technologies та підпорядкована ним **Cypress Semiconductor** випускають флеш-пам'ять NOR, ємнісні сенсорні контролери CapSense, бездротові низькоенергетичні рішення BLE Bluetooth, силові напівпровідники, датчики та мікроконтролери наступних серій[29, 30]:

- безпечні автомобільні Traveo, XMC, FM на базі ARM Cortex;
- потужні Aurix на базі TriCore;
- схеми F-RAM і SRAM;
- пристрої PSoC, PMIC.

STMicroelectronics виробляє інтегральні схеми для спеціальних застосувань, пристрої пам'яті (включаючи EEPROM), мікропроцесори, смарт-карти, та мікроконтролери наступних серій[31]:

- стійкі STM8 для загального користування;
- універсальні STM32 на базі ARM Cortex.

Texas Instruments розробляють таку продукцію, як аналогові мікросхеми, вбудовані процесори, - керуючі пристрої високої потужності C2000 та мікроконтролери таких серій [20]:

- малопотужні MSP430 загального призначення;
- промислові Sitara на базі ARM Cortex A;
- Tiva C на базі ARM Cortex M4 високої потужності.

NXP Semiconductors це виробник гетерогенних процесорів Vybrid, датчиків, аналогових мікросхеми, модулів зв'язку та наступних мікроконтролерів[30]:

- енергоефективні сімейства LPC на базі ARM Cortex;
- Kinetis на базі ARM Cortex-M високої потужності;
- кроссоверні MX базі ARM Cortex-M7;
- JN з вбудованими модулями бездротового зв’язку.

Zilog виробляють спеціалізовані мікропроцесори, спеціалізовані вбудовані системи на кристалі (SoC) та наступні серії мікроконтролерів[33]:

- Оптимізовані та малопотужні Z8 та Z80;
- Zilog ARM високої потужності.

Renesas Electronics пропонує аналогові та змішані інтегральні схеми, пристрої пам'яті та мікроконтролери таких серій, як [30, 34]:

- енергоефективні малопотужні RX, RZ, RE;
- універсальні RL78;
- енергоефективні Synergy(на ARM Cortex-M);
- RH850 високої потужності.

Espressif Systems надають наступні серії мікроконтролерів[35]:

- з вбудованим Wi-Fi модулем ESP8266;
- високопродуктивні ESP32;
- комунікаційні ESP32-S;
- кібербезпечні ESP32-C;
- мікроконтролери для автоматизації ESP32-H.

Silicon Labs виробляють бездротові системи BGM, набори для прототипування SLSTK та мікроконтролери серій [36]:

- енергоефективні EFM32 та EFR32;
- високо потужні C8051.

Analog Devices разом з підпорядкованими до них **Maxim Integrated** пропонують інтегральні схеми (IC) аналогової, змішаної та цифрової обробки сигналів (DSP), спеціалізовані вбудовані системи безпеки та наступні мікроконтролери[37]:

- багатоцільові аналогові ADuC/ADuCM/ADuM;
- енергоефективні MAX21000, MAXREFDES, MAXQ, MAXM;
- високо потужні Blackfin, SHARC, ADSP, SigmaDSP.

Nuvoton Technology виробляє клавіатурні контролери, вбудовані контролери для мобільних платформ і мікросхеми безпеки TPM, а також наступні мікроконтролери [38]:

- низько енергійне сімейства NuMicro;
- малопотужні керуючі N76 та W77;
- енергоефективні серії M051.

Виходячи з розглянутих різновидів програмовних пристроїв, можна зазначити широке різноманіття їх архітектурних рішень. Однак, незалежно від компанії-виробника, серійно їх можна розділити за напрямами розвитку наступних ключових технологій: енергоефективність, потужність, доступність та спеціалізованість. У подальшому буде розглянуто та порівняно найпоширеніших представників кожного з напрямів розвитку, а саме: енергоефективні та доступні AVR серії, потужні та багатофункційні STM32 і спеціалізовані ESP32[9].

1.1.2 Аналіз особливостей побудови і реконфігуропридатності програмовних пристроїв

Для виконання порівняльного аналізу з урахуванням вимог до основних апаратних ресурсів мікроконтролерів важливо відокремити такі параметри, як:

- обсяг пам’яті програм (у вигляді Flash пам'яті), що визначає максимальний обсяг коду програми у вигляді машинних інструкцій та завантажувача[39],

- обсяг адресного простору енергонезалежної пам'яті(EEPROM), який відповідає за збереження налаштувань та стану пристрою,

- обсяг статичної пам’яті змінних (SRAM) що зберігає стек виконання програм. Також для використання важлива кількість виводів, підтримка інтерфейсів, напруга живлення і частота тактування[9,40]. Для аналізу мікроконтролерів буде розглянуто найпоширеніші їх серії, такі, як AVR, STM32 та ESP32.

В результаті аналізу та порівняння для більшості мікроконтролерів AVR для визначення діапазонів всіх параметрів та наведених вище типів пам'яті, які є важливими для проведення реконфігурації, було виявлено, що вони залежать від конкретної моделі в серії, а для різних типів апаратних ресурсів не існує їх прямої залежності від збільшення кількості та об'єму флеш-пам'яті програми. Результати порівняння, включаючи розміри цих типів пам'яті[41], наведено в таблиці 1.1.

Таблиця 1.1. Порівняння трьох серій мікроконтролерів AVR.

№	AVR series	Flash, KB	EEPROM, KB	SRAM, KB
1	ATtiny	0,5 – 16	0 – 0,5	0 – 0,5
2	ATmega	4 – 256	0,5 – 4	0,25 – 16
3	ATxmega	16 – 384	1 – 4	1 – 32

На основі узагальнення результатів порівняння ресурсів, діапазони обсягів пам'яті (за наявності) для мікроконтролерів AVR виглядають наступним чином:

- для флеш-пам'яті становить від 0,5 Кб до 384 Кб, що э обмежуючим фактором при проведенні відновлювальних процедур за допомогою програмних засобів;

- об'єм EEPROM становить від 64 байт до 4 КБ, чого достатньо для зберігання налаштувань, даних калібрування та іншої важливої інформації, яку потрібно зберегти після перезавантаження;

- об'єм SRAM варіюється від 32 байт до 32 КБ, що забезпечує гнучкий діапазон мікроконтролерів за критерієм пам'яті для змінних.

Підтримка USB в сімействі ATmega реалізована в 5 основних моделях: ATmega16U2, ATmega32U2, ATmega32U4 (включає USB, що робить її популярною для USB-пристроїв), ATmega64U2, ATmega128U2. Деякі мініатюрні ATtinys також підтримують USB, наприклад, сімейство ATtiny 25/45/85, що дозволяє створити периферійний компонент комп'ютера з використанням стандартного драйвера або виконувати програмування безпосередньо через USB за допомогою завантажувача[42].

Схожим чином було проаналізовано мікроконтролери STM32, які виробляються компанією STMicroelectronics. Завдяки широкому вибору їх серій та моделей, спостерігається і широкий спектр характеристик мікроконтролера[43] результати порівняння котрих наведено в таблиці 1.2.

Об'єм вбудованої флеш-пам'яті в розглянутих серіях мікроконтролерів STM32 коливається від 8 Кб до 2 Мб, що не є обмеженням, оскільки контролерами передбачено використання зовнішньої флеш-пам'яті, що значно розширює діапазон.

Ще одною характерною рисою описаної серії є відсутність вбудованого EEPROM в апаратній реалізації мікроконтролерів STM32, як це реалізовано в інших мікроконтролерах, таких як AVR. Однак, завдяки значному об’єму програмної пам’яті, її відсутність може бути компенсована програмною реалізацією. Так, для зберігання енергонезалежних даних, які зберігаються при вимкненні живлення, можна використовувати емуляцію EEPROM у вбудованій флеш-пам'яті програми[44].

Обсяг такого EEPROM під час емуляції може варіюватися від декількох байт до декількох кілобайт, залежно від обсягу доступної флеш-пам'яті в мікроконтролері та конфігурації користувача. Наприклад, у мікроконтролері

STM32 з 64 КБ флеш-пам'яті можна виділити 1-4 КБ для емуляції EEPROM, не торкаючись коду та інших даних[45].

Деякі мікроконтролери STM32 серії STM32L (з наднизьким енергоспоживанням) мають вбудовану пам'ять даних EEPROM, яку можна використовувати як звичайну EEPROM. Наприклад, в серіях STM32L0 і STM32L4 вбудована пам'ять даних EEPROM варіюється від 2 до 8 КБ[46].

Таблиця 1.2. Порівняння мікроконтролерів серії STM32

№	AVR series	Flash, KB	SRAM, KB
1	STM32F0	16 – 256	4 – 32
2	STM32F1	16 – 1024	4 – 96
3	STM32F2	128 – 1024	64 – 128
4	STM32F3	16 – 512	16 – 80
5	STM32F4	64 – 2048	64 – 384
6	STM32F7	64 – 2048	256 – 512
7	STM32G0	16 – 512	8 – 128
8	STM32G4	32 – 512	4 – 32
9	STM32H7	128 – 2048	128 – 1433
10	STM32L0	8 – 192	2 – 20
11	STM32L1	32 – 512	4 – 80
12	STM32L4	64 – 1024	40 – 320
13	STM32L5	256 – 512	32 – 256
14	STM32WL	128 – 256	32 – 64
15	STM32WB	256 – 1024	1 – 256

Об'єм SRAM в мікроконтролерах STM32 залежить від серії та моделі, що забезпечує широкий вибір контролерів як для простих застосувань, так і для реалізації складних високопродуктивних алгоритмів обробки з великою кількістю змінних в системному коді. Таким чином, діапазон SRAM в мікроконтролерах STM32 становить від 2 КБ до 1 МБ, що дозволяє вибирати пристрої для

найрізноманітніших завдань, від простих вбудованих систем з мінімальним споживанням до складних додатків з високими вимогами до оперативної пам'яті.

Підтримка USB у розглянутій серії STM32 реалізована не у всіх моделях. Підтримка USB є в: STM32F0 (3 моделі), STM32F1 (6 моделей), STM32F2 (4 моделі), STM32F3 (5 моделей), STM32F4 (15 моделей), STM32F7 (10 моделей), STM32G0 (5 моделей), STM32G4 (7 моделей), STM32L0 (3 моделі), STM32L1 (3 моделі), STM32L4 (7 моделей), STM32L5 (2 моделі), STM32WL (4 моделі), STM32WB (5 моделей)[47].

У випадку мікроконтролерів ESP32 від Espressif Systems ситуація схожа з STM32. У додаток до звичної вбудованої флеш-пам'яті, надається підтримка зовнішньої флеш-пам'яті для забезпечення розширених можливостей. Об'єм флеш-пам'яті в мікроконтролерах ESP32 становить від 256 КБ до 4 МБ (в деяких випадках, з підтримкою зовнішньої флеш-пам'яті SPI, він може досягати 32 МБ в залежності від конкретного пристрою). Таким чином, об'єм вбудованої флеш-пам'яті в мікроконтролерах ESP32 варіюється від 256 КБ до 16 МБ, але його можна значно збільшити за допомогою зовнішньої пам'яті[48].

Відсутність вбудованого EEPROM у мікроконтролерів ESP32, як у STM32, вимагає, використання області у флеш-пам'яті програми для емуляції роботи енергонезалежної пам'яті.

Об'єм оперативної пам'яті в мікроконтролерах ESP32 варіюється в залежності від моделі. Виробник надає достатньо великий обсяг пам'яті для різних застосувань, особливо тих, що пов'язані з IoT, бездротовим зв'язком та вбудованими системами. Об'єм SRAM-пам'яті в мікроконтролерах ESP32 становить від 192 КБ до 520 КБ, в залежності від моделі. Це дозволяє використовувати ці мікроконтролери в широкому спектрі застосувань, зокрема в задачах з високим навантаженням і вимогами до пам'яті, наприклад, для реалізації обміну за стандартами бездротового інтернет-з'єднання[4]. Результати порівняння обсягу програмної та SRAM пам'яті[48] для серії ESP32 наведені в таблиці 3.

Таблиця 1.3. Порівняння мікроконтролерів серії ESP32.

№	AVR series	Flash, KB	SRAM, KB
1	ESP32	256 – 4096	320 – 520
2	ESP32-S2:	1024 – 16384	200 – 320
3	ESP32-S3:	2048 – 16384	384 – 512
4	ESP32-C3:	4096 – 16384	320 – 400
5	ESP32-C6:	4096 – 16384	400 – 512
6	ESP32-H2:	2048 – 16384	192 – 256

З точки зору підтримки інтерфейсів, для набору з 6 мікроконтролерів ESP32 підтримка USB реалізована лише у двох моделях: ESP32-S2 та ESP32-S3 [9, 49].

Отже, за результатами проведеного аналізу розглянутих мікроконтролерів можна зробити висновок, що процес прийняття рішення про використання їх внутрішніх ресурсів має враховувати параметри цих ресурсів, що у свою чергу впливає на порядок забезпечення та підтримки реконфігуропридатності.

1.1.3 Аналіз вимог до надійності програмовних пристроїв безпілотних систем

Щорічний прогрес у розвитку індустрії безпілотних апаратів стимулює підвищення безпекових та надійнісних вимог до її продукції. Наразі, існує чотири основних напрямки, що вирішують задачі забезпечення кращих показників надійності для БПА а саме:

- автономності, що передбачають зменшення участі людини при експлуатації БПА;
- способів та засобів навігації;
- розмірів та продуктивності використовуваних компонентів при побудові;
- джерел живлення та способу зберігання енергії [50].

Для досягнення кращих показників безпеки та надійності передбачається пошук найкращого поєднання між собою рішень кожної з категорій. Так, підвищення автономності потребує покращення можливостей дистанційного керування та мінімізації впливу людського фактору. При реалізації системи керування у вигляді наземної станції, її показник ремонтопридатності є найвищим в усій системі БПА за рахунок її легкодоступності і модульної структури. На випадок втрати сигналу або аварійної ситуації, борти БПА оснащуються системами автоматичної посадка або повернення додому, що запобігає втраті пристрою. Однак, чутливим залишається питання часу відновлення при відмовах і забезпечення стабільності з'єднання БПА з системою керування.

Рівною мірою з автономністю, важливим є питання розвитку навігаційних технологій. Через масове оснащення БПА передовими навігаційними системами, такими як GPS або наземні мережі датчиків, сучасні системи керування стали залежними від їх якості. Наразі; ця частина БПА є найчутливішою до відмов у порівнянні з іншими системами. Тому, на етапі розроблення ці підсистеми оснащуються вагомою кількістю «гарячих» резервних вузлів/підсистем, що впливає на вагові та габаритні характеристики. Тому, питання електронної мініатюризації складових і використання полегшених матеріалів залишається актуальним[51].

Джерело живлення для БПА є критичним компонентом забезпечення працездатності. Наразі, поширеним є варіант використання літій-іонних акумуляторів, через їх високу енергоємність при відносно малій вазі і тривалий термін служби. Через відсутність рухомих частин, система живлення є досить надійною. Однак, через її хімічний аспект, система живлення є чутливою до зовнішніх умов, таких як температура, вологість, наявність випромінювання, що в при тривалих термінах експлуатації можуть призводити до деградації його властивостей [52]. Відповіддю на такий виклик виступають резервні системи живлення, що зменшує вірогідність втрати БПА через втрату живлення. Незважаючи на досягнення за цим напрямком розвитку, актуальними є задачі розвитку хімічної складової джерел живлення, їх захисту від зовнішнього

середовища, та впровадження оптимальних стратегій керування енергопостачанням[51].

1.2 Аналіз методів і засобів забезпечення надійності програмовних пристроїв безпілотних систем

1.2.1 Методи оцінювання надійності програмовних пристроїв безпілотних систем

Аналіз надійності або інженерія надійності - це область знань, яка включає в себе широкий спектр підходів, методів і алгоритмів для оцінки надійності та ризику складних систем. Вибір і застосування методів аналізу надійності в кожному конкретному випадку залежить від структури системи, її типу та сфери застосування.

Одним зі способів аналізу виступає стохастичне моделювання моделювання. Розповсюдженими вважаються наступні способи моделювання:

Binary State System (BSS) – математична модель, яка передбачає використання двох станів функціонування системи та два стани компонентів у математичному представленні.

Multi State System (MSS) – математична модель, яка дозволяє розглядати в математичному представленні більше двох станів функціонування системи та станів компонентів системи. Вона дозволяє детально описувати та аналізувати систему як за відмовами, так і за її деградацією. Однак, методи аналізу надійності MSS вимагають більших обчислювальних ресурсів ніж BSS [53].

Окрім традиційних моделей, що використовуються для аналізу надійності БПА існують практики оцінювання надійності БПС використовують системи машинного навчання для математичного представлення досліджуваної системи [54]. Однак їх використання базується, переважно, на аналізі надійнісних характеристик БПА.

Для аналізу характеристик системи необхідним є визначення наступних критеріїв, за якими призводитиметься оцінювання відмовостійкості та продуктивності ПП в умовах обмеження ресурсів.

Відмовостійкість – здатність системи автоматично зберігати та відновлювати за обмежений час працездатність при специфікованих відмовах її елементів. В системах БПА поточний критерій відноситься не лише до самого ПП, а й до периферії, яка забезпечує безперервний зв'язок з оператором та іншими учасниками взаємодії.

Ремонтопридатність (відновлюваність, обслуговуваність) – здатність системи до відновлення працездатності після відмови. Для систем БПА поточний критерій визначається наявністю систем автоматичного ремонту (самовідновлення) або доступністю до обслуговування через зовнішнє втручання.

Вразливість до зовнішніх впливів (живучість, резильєнтність) – здатність системи протистояти впливу фізичних факторів. Важливою складовою живучості є керованість процесів деградації (зменшенням рівня ефективності) в умовах втрати працездатності частиною елементів системи. Залежно від ступеня впливу зовнішніх чинників у БПА реалізуються процедури керованої деградації, з метою мінімізації його впливу на ефективність.

Вразливість до кіберзагроз (ізольованість, інформаційна безпека) – здатність системи протидіяти впливу інформаційних факторів. Для БПА чутливими виступають комунікаційні канали, через які можуть проводитись спроби перехоплення, перевантаження трафіку або ізоляції від оператора. Наявність сценаріїв кібербезпекових заходів визначає ступінь готовності системи до протидії можливим атакам.

Ефективність – комплексна характеристика, що визначає ступінь відповідності трудовитрат системи до отриманого результату роботи. Окрім ступеня оптимізації внутрішніх ресурсів ПП, до цієї характеристики включається і оцінка економічної доцільності вибору, обслуговування та експлуатації БПА з оглядом на результат.

Для описаних вище характеристик відноситься наступний перелік параметрів для вимірювання:

- **імовірність безвідмовної роботи** за обмежений проміжок часу. Імовірність *P(tx)*, що система буде функціонувати безвідмовно протягом заданого часу *tx* є ключовим показником безвідмовності (надійності), що визначає час напрацювання системи до її відмови;

- **середній час відновлення** програмовного пристрою. Час $t_{ср}$, необхідний для відновлення працездатності системи після відмови;

- **інтенсивності відмов** БПА при відсутності фізичних та інформаційних впливів. Інтенсивність λ_0. визначає умовну густину ймовірностей виникнення відмови згідно їх розподілу у часі;

- **показник економічної ефективності** використання БПА: Характеризується відношенням математичного очікування часу безвідмовної роботи $E|\ t\ |$ з імовірністю *P(tx)*, не нижчою за мінімально необхідне, до сумарної вартості системи[55].

1.2.2 Методи і засоби забезпечення надійності програмовних пристроїв безпілотних систем

В ході аналізу існуючих методів забезпечення надійності ПП БПА було розглянуто ряд досліджень, значна частина котрих присвячується підходам до резервування та керуванню резервом. Так, для систем з ковзним резервуванням розглядається доцільність використання резерву наступних типів:

«Холодний резерв», що зазвичай використовується у енергоефективних та малопотужних системах, коли резервний компонент починає функціонувати лише тоді, коли робочий компонент виходить з ладу.

«Гарячий резерв», що використовується як механізм обходу відмов у випадках, де час відновлення є критичним. При такому підході основний елемент та його резерв перебувають у однаково навантаженому стані.

«Теплий резерв», що застосовується для систем з помірним споживанням енергії та часом відновлення. При нормальному режимі роботи, частково задіяний резерв вимикається тільки у випадку, коли він виходить з ладу[56].

Для ремонтопридатних систем пропонуються методи алгебраїчного аналізу багатостанових k-out-of-n систем, засновані на діаграмах багатозначності прийняття рішень. Їх використання передбачає оцінювання та супровід систем з різними вимогами до кількості працюючих елементів. Відповідно рівням працездатності пропонується стратегія відновлення БПА з передбачуваною деградацією його функцій[57].

В рамках забезпечення керованості пропонуються методи аналізу та підтримки керованості та надійності, які пов'язані з аналітичною методологією масштабування. Аналіз керованості базується на наявному індексі керованості, адаптованому для оцінки масштабу відмов та аналізу стану систем для адаптації її обчислювальної ефективності. Визначення масштабу спирається на сучасну аналітичну методологію без використання баз даних з мультидисциплінарною оптимізацією [58].

Для порядку обслуговування БПА пропонується використання логістичного підходу, заснованого на оцінці надійності та профілактичного і коригувального обслуговування. При цьому система розглядається така, що піддається до повної відмови(«жорстка відмова»), або до часткового зниження продуктивності(«м’яка відмова»). За допомогою профілактичних і коригувальних процедур обслуговування метод дозволяє перевести частину «жорстких» відмов до розряду «м’яких» і усунути певну кількість часткових відмов [59].

1.3 Постановка задачі і обґрунтування методики досліджень

1.3.1 Постановка загальної та часткових завдань дослідження

Загальне завдання: розроблення методів, моделей і програмно-апаратних засобів для відмовостійких програмовних пристроїв безпілотних систем з багаторівневою деградацією при відмовах внутрішніх компонентів.

Часткові завдання:

1) аналіз існуючих методів і технологій забезпечення надійності програмовних пристроїв для безпілотних систем з багаторівневою деградацією при відмовах внутрішніх компонентів. Обґрунтування задач і методики досліджень;

2) розроблення моделей надійності програмовних пристроїв з керованою багаторівневою деградацією при відмовах компонентів безпілотних систем;

3) розроблення методів реконфігурації та оцінювання реконфігуропридатності програмовних пристроїв безпілотних систем з керованою багаторівневою деградацією;

4) розроблення марковських моделей оцінювання готовності програмовних пристроїв для безпілотних систем з керованою багаторівневою деградацією;

5) розроблення послідовності оцінювання та засобів забезпечення надійності і живучості програмовних пристроїв методів і засобів при створенні та модернізації безпілотних систем різних типів (літальних, наземних морських);

6) впровадження розроблених методів і програмно-апаратних засобів забезпечення надійності програмовних пристроїв безпілотних систем.

1.3.2 Обґрунтування методики та математичного апарату дослідження

Під час дослідження буде проведено комплекс робіт для формування заявлених наукових результатів. На підставі вхідних даних буде проаналізовано, розроблено та досліджено компоненти моделі, методи і засоби, представлені схемою на рисунку 1.1.

Розглянуті у першому розділі програмовні пристрої буде представлено у вигляді теоретико-множинних моделей, що представляють ПП сукупністю компонентів різного ступеня реконфігуропридатності. Описуватиметься відповідність множини компонентів до множини дефектів.

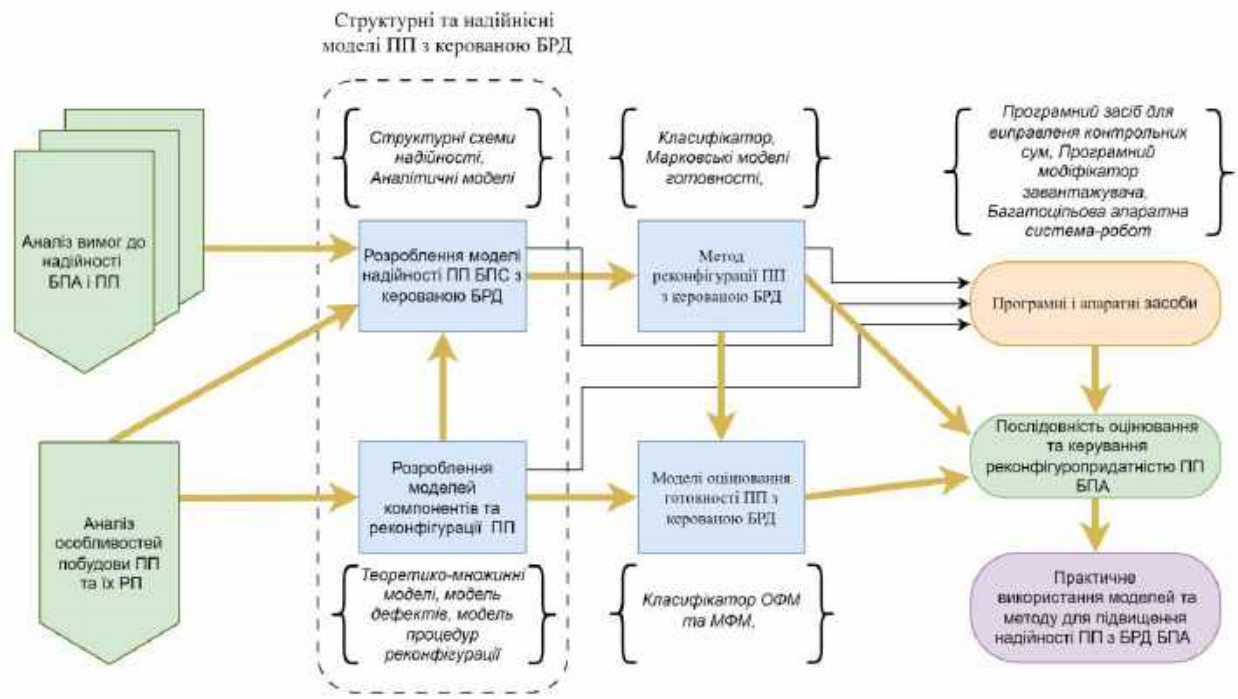

Рисунок 1.1 – Схема взаємозв’язку результатів дослідження

Схожим чином описуватиметься залежність процедур реконфігурації від компонентів, до яких вони можуть застосовуватись. На базі цього сформується модель дефектів, що описує надійнісну поведінку ПП відповідно рівням працездатності. На основі моделі компонентів будуть побудовані удосконалені структурні та надійнісні схеми програмовних пристроїв з урахуванням засобів контролю та реконфігурації. Відповідно до розглянутих структурних схем будуть побудовані аналітичні моделі, на основі котрих прослідковується вплив багаторівневої деградації на ймовірності рівнів працездатності з плином часу відносно моделі надійності базового ПП.

У сукупності, розроблені структурні схеми надійності, аналітичні та теоретико-множинні моделі відповідають першому науковому результатові дослідження про удосконалення **структурних та надійнісних моделей** програмовних пристроїв з керованою багаторівневою деградацією.

Відповідно другому результату дослідження, на основі моделей компонентів буде представлено метод аналізу реконфігуропридатності який базується на показниках об’єму та часу. Метрики об’єму представлятимуться у

вигляді потенційної та реалізованої реконфігуропридатності ПП. Метрики часу будуть представлені інтервальними та моментними показниками часу.

Для другої частини другого результату буде представлено сукупності процедур реконфігурації на основі множин дефектів та метрик реконфігуропридатності. Їх забезпечення призводитиметься за допомогою засобів контролю та реконфігурації, що мають доступ до фабричного програмного забезпечення програмовних пристроїв БПА.

При удосконаленні марковських моделей готовності програмовних пристроїв з керованою БРД, що є третім науковим результатом, буде розроблено класифікатор ознак моделей готовності, форма яких залежить від способу реалізації БРД. На основі класифікатора будуватиметься та досліджуватиметься множина типових моделей готовності на основі ОФМ та МФМ.

Згідно розробленим структурним та надійнісним моделям надійності ПП з керованою БРД, методу реконфігурації та множині моделей готовності будуватиметься комплекс апаратно програмних рішень для проведення реконфігурації, а саме:

- програмний засіб для забезпечення реконфігуропридатності;
- програмний засіб для виправлення контрольних сум;
- апаратний засіб(платформа) для рою роботів з підтримкою функціональної реконфігуропридатності.

Окрім цього визначатиметься послідовність оцінювання та керування реконфігуропридатністю та вибір набору процедур. Послідовність вибору процедур реконфігурації для забезпечення надійності складатиметься з наступних операцій:

- аналіз архітектури МК та можливості реалізації процедур реконфігурації;
- оцінювання надійності ПП з БРД для кожної з процедур;
- визначення доцільності використання кожної з процедур з використанням аналітичних моделей;

- вибір моделі оцінювання готовності ПП з БРД для кожної з визначених процедур реконфігурації з урахуванням інших ознак за класифікатором моделей;
- оцінювання готовності ПП з БРД за обраною моделлю;
- визначення раціонального варіанту побудови ПП з БРД за результатами оцінювання.

1.4 Висновки до першого розділу

На підставі проведеного аналізу актуальності, стану розроблення методів оцінювання і забезпечення надійності і живучості програмовних пристроїв сформулюємо наступні висновки, які стосуються ключових аспектів досліджень.

1. Аналіз програмовних пристроїв.

В ході аналізу програмовних пристроїв їх було класифіковано за архітектурними та модельними особливостями. Були розглянуті наступні особливості:

- об'єми програмної пам'яті;
- об’єми статичної пам’яті змінних;
- об’єми енергозалежної пам’яті;
- наявність універсальних інтерфейсів взаємодії.

В ході аналізу вимог до надійності програмовних пристроїв безпілотних систем було розглянуто виклики за наступними напрямами:

- автономності, що передбачають зменшення участі людини при експлуатації;
- способи та засоби навігації;
- якісні характеристики компонентів побудови;
- джерела живлення та способи зберігання енергії.

2. Аналіз методів і засобів забезпечення надійності програмовних пристроїв БПА.

В ході аналізу методів забезпечення надійності ПП було розглянуто наступні методи оцінювання надійності:

- Binary State System (BSS);
- Multi State System (MSS);
- аналіз за показниками надійності, який включає оцінювання за параметрами імовірності безвідмовної роботи, середнього часу відновлення, інтенсивності відмов, економічної ефективності.

В ході аналізу методів і засобів забезпечення надійності ПП було розглянуто:

- типи резервування;
- методи оцінки та підтримки ремонтопридатності БПА;
- методи оцінки та забезпечення керованості БПА;
- методи обслуговування БПА.

3. На основі проведеного аналізу була сформульована загальна задача дисертаційного дослідження, яка була декомпозована на перелік часткових задач, виконання яких доцільно здійснювати у три етапи:

- розроблення моделей надійності та методу реконфігурації для програмовних пристроїв безпілотних систем;
- розроблення марковських моделей оцінювання готовності програмовних пристроїв;
- розроблення програмно-апаратних засобів забезпечення реконфігуропридатності.

Результати виконання етапів у подальшому можуть бути використані при модернізації БПА з метою підвищення їх надійності за рахунок впровадження багатоступеневої керованої деградації шляхом часткового або повного відновлення через реконфігурацію.

Таким чином, в даному розділі обґрунтовано мету, задачі і методику проведення досліджень, результати яких будуть представлено в наступних розділах.

РОЗДІЛ 2
РОЗРОБЛЕННЯ МОДЕЛЕЙ НАДІЙНОСТІ ТА МЕТОДУ РЕКОНФІГУРАЦІЇ ПРОГРАМОВНИХ ПРИСТРОЇВ БЕЗПІЛОТНИХ СИСТЕМ З КЕРОВАНОЮ БАГАТОРІВНЕВОЮ ДЕГРАДАЦІЄЮ

2.1 Розроблення моделей реконфігуропридатних програмовних пристроїв з керованою багаторівневою деградацією

2.1.1 Розроблення моделей дефектів програмовного пристрою

Виходячи з опису поширених платформ розгортання програмовних систем[11, 60], можна зазначити, що структурно вони складаються з множини функціональних компонентів, яку можна представити як

$$MComp = \{Comp_i \mid i \in \mathbb{N}, i \le n\}, \tag{2.1}$$

де $Comp_i$ це i-тий програмно-апаратний компонент програмовної системи, а n – загальна кількість компонентів. У свою чергу кожному компоненту відповідає множина дефектів

$$Comp_i \sim Def_i = \{def_{ij} \mid j \in \mathbb{N}, j \le m_i\}, \tag{2.2}$$

де Def_i це множина одиничних дефектів def_j компонента $Comp_i$, а m_i – це кількість допустимих дефектів в ньому. Тоді загальна множина дефектів програмовної системи буде описуватись як,

$$MDef = \{\cup Def_i \mid i \in \mathbb{N}, i \le n\}, \forall_{j,k} : j \ne k : Def_j \cap Def_k = \varnothing, \tag{2.3}$$

де Def_j та Def_k це пара різних дефектів, що не має спільних ділянок чи компонентів виникнення. Залежно від наслідків прояву дефекту, загальну множину дефектів $MDef$ можна представити у вигляді підмножин, описаних нижче.

1. Сукупність несуттєвих дефектів, які не впливають на якість чи швидкість функціонування і яку можна описати як

$$MDef_A = \{def_{Ap} \mid p \in \mathbb{N}, i \le m_A\}, \tag{2.4}$$

де m_A це загальна кількість споріднених дефектів def_A. До них можна віднести незначні пошкодження корпусів або з'єднань компонентів, які мають характер збою.

2. Сукупність суттєвих дефектів, які знижують якість функціонування за рахунок зменшення продуктивності, точності або функціональних спроможностей, і яку можна описати як

$$MDef_B = \{def_{Bq} \mid q \in \mathbb{N}, i \le m_B\}, \quad (2.5)$$

де m_B це загальна кількість споріднених дефектів def_B. При виникненні такого роду дефектів система ще зберігає здатність виконувати свої основні функції, але її стан стає частково працездатним. Така підмножина не має спільних областей виникнення з описаними до цього підмножинами, тобто $MDef_B \cap MDef_A = \varnothing$. За характером відновлюваності, множину $MDef_B$ можна розділити на підмножини дефектів

$$MDef_{B1} = \{def_{B1x} \mid x \in \mathbb{N}, x \le m_{B1}\}, \quad (2.6)$$

де m_{B1} це загальна кількість споріднених дефектів def_{B1}. Такі дефекти є придатними до обходу наслідків їх прояву, повертаючи системі повну працездатність. Також варто виділити підмножину дефектів

$$MDef_{B2} = \{def_{B2y} \mid y \in \mathbb{N}, y \le m_{B2}\}, \quad (2.7)$$

обхід наслідків котрих не можливий без часткової втрати працездатності. При цьому, варто зазначити, що виникаючі дефекти можуть відноситись лише до однієї з описаних підмножин, тобто $MDef_{B1} \cap MDef_{B2} = \varnothing$. У свою чергу множина дефектів $MDef_{B2}$ може бути декомпозована на підмножини залежно від рівня деградації, обумовлених дефектами цієї множини. Відповідно до витрачуваних показників можна розрізняти дефекти, впливаючі на функціональні спроможності

$$MDef_{B2} = \{\cup Def_{B2s} \mid s \in \mathbb{N}, s \le d_f\}, \forall_{x,y} : z \ne s : Def_{B2s} \cap Def_{B2z} = \varnothing, \quad (2.8)$$

та дефекти, що впливають на зміну показників якості

$$MDef_{B2} = \{\cup Def_{B2t} \mid t \in \mathbb{N}, t \le d_q\}, \forall_{t,w} : t \ne w : Def_{B2t} \cap Def_{B2w} = \varnothing, \quad (2.9)$$

де d_f та d_q це кількості рівнів деградації, а Def_{B2z} та Def_{B2s} і Def_{B2t} та Def_{B2w} це пари різних дефектів, що не мають спільних ділянок чи компонентів виникнення.. В цьому випадку передбачається перерозподіл ресурсів системи з метою мінімізації

втрат, що виникають при прояві дефекту. Для цього обирається найменш впливовий працездатний елемент, або їх група, за рахунок котрих компенсуються втрати системи.

3. Сукупність суттєвих дефектів, які виникають у критичних областях системи і призводять до повної відмови, можуть бути описані як

$$MDef_C = \{def_{Cr} \mid r \in \mathbb{N}, i \le m_C\}, \quad (2.10)$$

де m_C це загальна кількість споріднених дефектів def_C. Множина дефектів, що не має спільних елементів з іншими описаними підмножинами, тобто $MDef_C \cap MDef_B = \varnothing$, та $MDef_C \cap MDef_A = \varnothing$. За характером множину $MDef_C$ можна розділити на підмножину дефектів

$$MDef_{C1} = \{def_{C1v} \mid v \in \mathbb{N}, i \le m_{C1}\}, \quad (2.11)$$

де m_{C1} це загальна кількість споріднених дефектів def_{C1}. Це множина дефектів, наслідки виникнення котрих є потенційно придатними до обходу. Відповідно слід виділити

$$MDef_{C2} = \{def_{C2u} \mid u \in \mathbb{N}, i \le m_{C2}\}, \quad (2.12)$$

де m_{C2} це загальна кількість споріднених дефектів def_{C2}. Це множина дефектів, наслідки виникнення котрих є потенційно придатними до пом’якшення з частковою втратою працездатності. Також варто виділити

$$MDef_{C3} = \{def_{C3g} \mid g \in \mathbb{N}, i \le m_{C3}\}, \quad (2.13)$$

як множину дефектів у критичних для системи ділянках, де відновлення за допомогою реконфігурації не виявляється можливим.

Отже, відповідно до виразів (2.4 – 2.13) множина дефектів (2.3) може бути представлена виразом

$$MDef = MDef_A \cup MDef_{B1} \cup MDef_{B2} \cup MDef_{C1} \cup MDef_{C2} \cup MDef_{C3}. \quad (2.14)$$

Тоді опишемо ці рівні відношенням за рівнем(тяжкістю) потенційної деградації, яке позначаємо символом «⇒» :

$$Def_A \Rightarrow Def_{B1} \Rightarrow Def_{B2} \Rightarrow Def_{C1} \Rightarrow Def_{C2} \Rightarrow Def_{C3}. \quad (2.15)$$

Робимо консервативне припущення для випадку, коли компонента $Comp_i$ може мати кілька різних дефектів відповідно до описаної декомпозиції множин (2.14) та

їх відношення (2.15), що для неї існує тільки один тип дефектів, який є найгіршим за наслідками. Відповідно до визначених типів дефектів множина компонентів описується наступним виразом

$$MComp = MComp_A \cup MComp_{B1} \cup MComp_{B2} \cup MComp_{C1} \cup MComp_{C2} \cup MComp_{C3}, \quad (2.16)$$

де *MComp* це множина компонентів з однаковим типом дефектів. Підмножини виразу (2.16) не перетинаються на підставі представлених вище міркувань. Зазначимо, що дефекти множин (2.14) є причинами відмов, які обумовлюють різні рівні деградації. Такі рівні характеризуються ступенем зменшення чисельних показників якості або функціональних спроможностей і визначаються кількісно від *0* до *d*, де 0 це відсутність деградації, а *d* це неприпустимий рівень з точки зору життєздатності або безпечності системи.

Тоді, згідно з ланцюгом (2.15) опишемо рівні переходу означених чисельних показників у вигляді Таблиці 2.1.

Таблиця 2.1 – Порядок зміни рівня деградації залежно від типу дефекта

№	Типи дефектів	Рівні деградації	Послідовності змін рівнів деградації
1	Def_A	0	$0 \to 0$
2	Def_{B1}	0	$0 \to x \to 0,\ x \in \mathbb{N},\ x < d$
3	Def_{B2}	$1, \dots, (d-1)$	$0 \to x \to s,\ x \in \mathbb{N},\ x < d,\ s \in \mathbb{N},\ s \le x$
4	Def_{C1}	0	$0 \to x \to 0,\ x \in \mathbb{N},\ x \ge d$
5	Def_{C2}	$0, \dots, d$	$0 \to x \to s,\ x \in \mathbb{N},\ x \ge d,\ s \in \mathbb{N},\ s < d$
6	Def_{C3}	d	$0 \to x \to s,\ x \in \mathbb{N},\ x \ge d,\ s \in \mathbb{N},\ s \le x$

Отже, однорівнева деградація (d = 1) характерна для систем можуть знаходитися лишу двох станах, а саме працездатному та відмовному. Такими системами опікується класична теорія надійності. Якщо рівнів декілька (d > 1), маємо системи з багаторівневою деградацією, які є предметом вивчення теорії живучості.

2.1.2 Розроблення комплексу процедур реконфігурації програмового пристрою

Закладена у програмових пристроях можливість реконфігурації дозволяє її використання у рамках самовідновлення [61]. Залежно від типу використовуваного пристрою, місця та характеру виникнення дефекту, можливості проведення реконфігурації можуть різнитися. В окремих випадках, для одного дефекту може бути застосовним декілька процедур реконфігурації, які матимуть різні результати впливу на наслідки виникнення дефекту. Таким чином, для одного дефекту може бути описаною множина

$$Def_i \sim Rec_i = \{rec_{ij} \mid j \in \mathbb{N}, j \le m_i\}, \tag{2.17}$$

де Def_i це i-тий дефект програмового пристрою, Rec_i це множина одиничних процедур реконфігурації rec_j , а m_i – це кількість допустимих процедур для цього дефекту. Тоді загальна множина процедур реконфігурації може описуватись як

$$MRec = \{\cup Rec_i \mid i \in \mathbb{N}, i \le n\}, \tag{2.18}$$

де n – загальна кількість елементів, до яких реконфігурація може бути задіяна. Множина процедур реконфігурації може бути декомпозована за характером відновлення на підмножини, представлені нижче.

1. Множина процедур відновлення без втрати функціональних можливостей, яку можна описати як

$$MRec_D = \{rec_{Dp} \mid p \in \mathbb{N}, p \le m_D\}, \tag{2.19}$$

де m_D – загальна кількість споріднених процедур rec_D. До них можна віднести відновлення за рахунок незадіяного ресурсу пристрою, завдяки якому система відновлює повну працездатність.

2. Множина процедур відновлення з частковою втратою якісних або функціональних спроможностей, яку можна описати як

$$MRec_E = \{rec_{Ep} \mid q \in \mathbb{N}, q \le m_E\}, \tag{2.20}$$

де m_E – загальна кількість процедур rec_E. В цьому випадку відновлення відбувається за рахунок перерозподілу завдань між працездатними компонентами, що супроводжується певною деградацією самої системи. Відповідно до типів

деградації показників якісних та функціональних спроможностей, ці процедури можна розділити на такі підмножини, як

$$MRec_{E1} = \{rec_{E1x} \mid x \in \mathbb{N}, x \le m_{E1}\}, \quad (2.21)$$

де m_{E1} – кількість процедур rec_{E1}. Відновлення відбувається зі збереженням пріоритетного показника, що визначається згідно виконуваних системою завдань, за рахунок невиконання таких, що не впливають на життєздатність. Також варто виділити множину процедур

$$MRec_{E2} = \{rec_{E2y} \mid y \in \mathbb{N}, y \le m_{E2}\}, \quad (2.22)$$

де m_{E2} – кількість процедур rec_{E2}. Тут відновлення відбувається зі збереженням другорядного показника, що визначається згідно виконуваних системою завдань, за рахунок зниження пріоритетного. Ще одна множина процедур описується виразом:

$$MRec_{E3} = \{rec_{E3z} \mid z \in \mathbb{N}, z \le m_{E3}\}, \quad (2.23)$$

де m_{E3} – кількість процедур rec_{E3}. В цьому випадку деградація впливає на всі показники, що призводить до їх зміни у допустимих межах, які у свою чергу, визначаються згідно виконуваних системою завдань.

3. Множина процедур реконфігурації без можливості відновлення до мінімально необхідних показників системи

$$MRec_{F} = \{rec_{F3r} \mid r \in \mathbb{N}, r \le m_{F}\}, \quad (2.24)$$

де m_F – загальна кількість процедур rec_F. Множина процедур реконфігурації, які запобігають подальшій деградації системи. Відновлення працездатності при таких процедурах або неможливе, або небезпечне.

Отже, виходячи з опису можливих дефектів та потенційних шляхів реконфігурації ПП, можна зробити висновок, що процедури реконфігурації мають різні ступені впливу, які залежать від характеру та місця виникнення дефекту. Повне толерування або пом’якшення впливу дефекту може частково або повністю відновити працездатність ПП.

2.1.3 Дослідження моделей надійності за допомогою діаграм деградації

Діаграми деградації. Теоретико-множинний опис може бути доповнений діаграмами деградації [62] у графічній формі. Ці діаграми описують процес деградації за двома параметрами: показниками якості (кількості функцій, що виконуються) П та часу (моментів часу зміни рівня деградації) t_1 , … , t_d. Цим моментам часу відповідають значення П(t): $П_0$, … , $П_{d-1,}$ 3d - кількість рівнів деградації. Зазначимо, що $П_{d-1}$ є мінімальним припустимим значенням показника. У свою чергу, під час змін рівня деградації простежуються такі моменти часу, як t_{MD} (момент прояву дефекту, що спричинив порушення працездатності), t_{SR} (момент початку процесу відновлення), t_{ER} (момент завершення процесу відновлення), та t_{ER} (момент застосування внесених змін). Для зручності, t_{MD} та t_{ER} позначаються множиною дефектів (2.4-2.12) та процедур реконфігурації(2.18-2.23) відповідно, які спричинили зміну параметрів.

Таким чином, діаграми деградації для системи при прояві множин дефектів $MDef_A$, $MDef_B$, $MDef_C$ матимуть вигляд, представлений на рисунках 2.1-2.3.

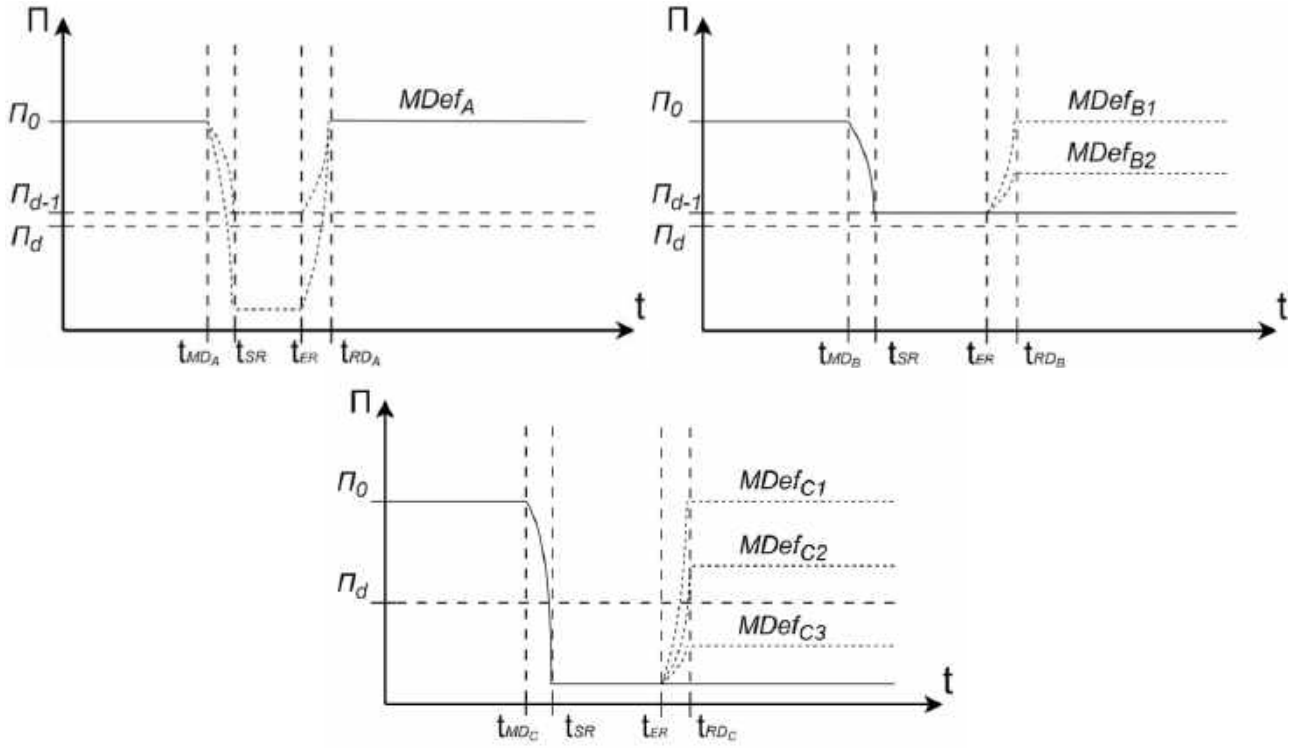

Рисунок 2.1-3 – Діаграми деградації прояву розглянутих множин дефектів

Діаграма демонструє зміну показнику якості за причини прояву розглянутих груп дефектів (2.13). Слід зазначити, що суцільною лінією описується природна поведінка параметру, без впливу засобів реконфігурації. Рівень тяжкості наслідків прояву дефекту впливає на потенційні можливості до відновлення (пунктирна лінія), і обумовлюється типом самого дефекту.

У свою чергу, діаграми деградації для системи при застосуванні множин процедур реконфігурації $MRec_D$ $MRec_E$ $MRec_F$ матимуть вигляд, представлений на рисунках 2.4-2.6.

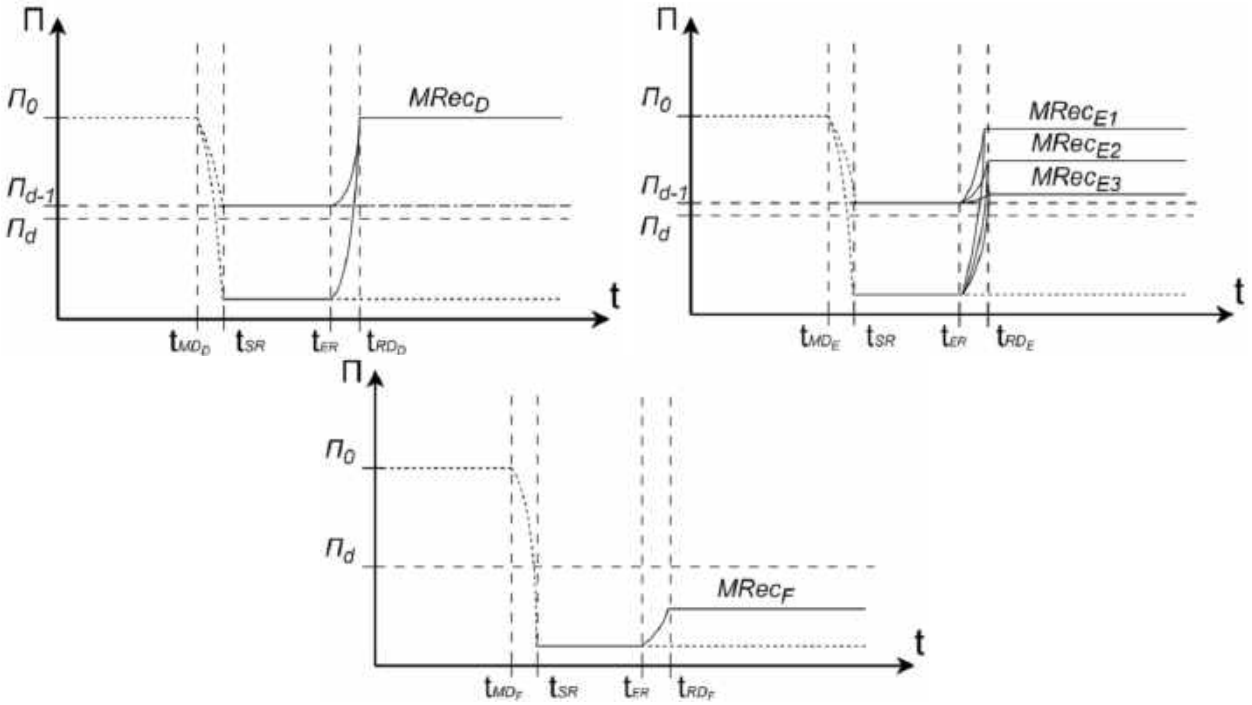

Рисунок 2.4-2.6 – Діаграми деградації впливу розглянутих множин процедур реконфігурації

Діаграма демонструє зміну показнику якості після застосування процедур реконфігурації (2.17). Слід зазначити, що суцільною лінією описується природна поведінка параметру з урахуванням впливу засобів реконфігурації. Рівень тяжкості наслідків прояву дефекту впливає на початкові умови відновлення (пунктирна лінія), що також обумовлюється типом дефекту.

При послідовному прояві різних типів дефектів, ресурс реконфігурації поступово вичерпується, через що рівень якості при працездатному стані починає падати, що призводить до поступової деградації. Залежно від інтенсивності

виникнення тяжких дефектів та кількості процедур відновлення, процес деградації може різнитися. Приклад помірної деградації представлено на рисунку 7.

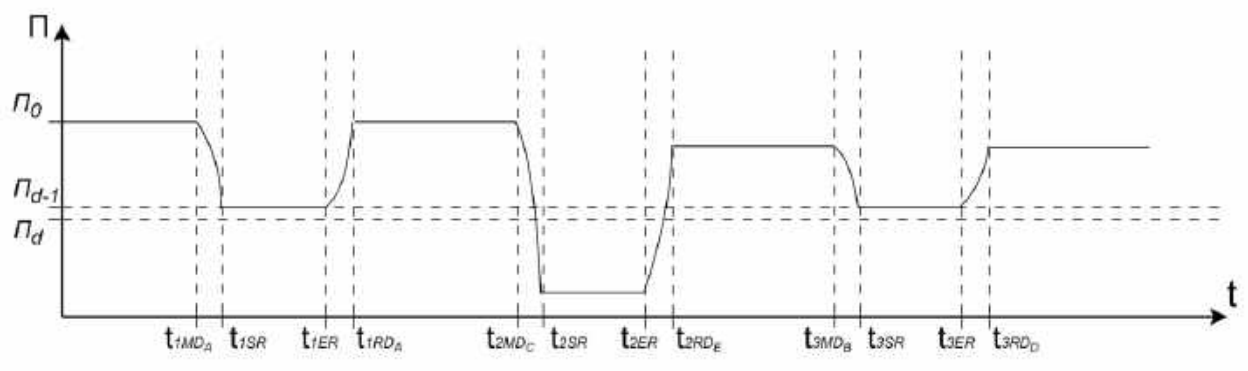

Рисунок 2.7 – Приклад керованої деградації з помірною інтенсивністю виникнення відмов

У даному випадку система має у своєму розпорядженні достатній запас процедур відновлення, що дозволяє зберігати високий рівень якості на протяжному проміжку часу. На це також впливає низька інтенсивність прояву дефектів високої тяжкості, що впливає на інтенсивність вичерпування відновлювальних ресурсів. Приклад деградації високої інтенсивності представлено на Рисунку 8.

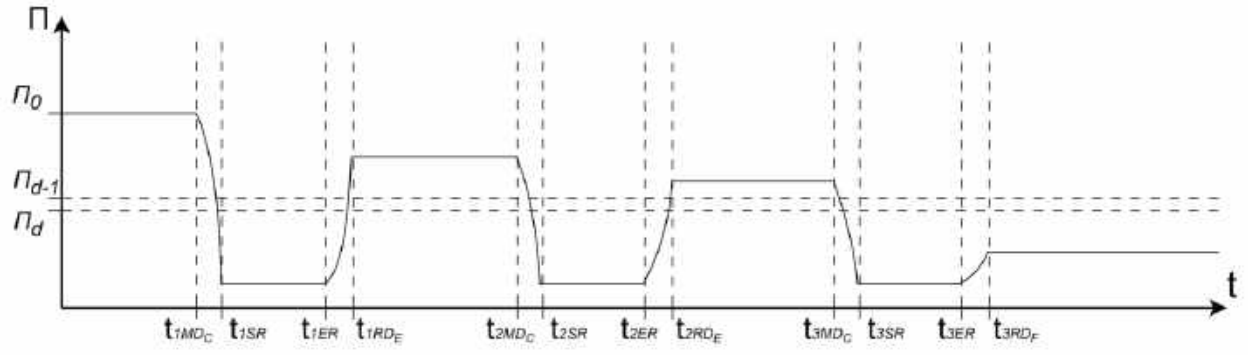

Рисунок 2.8 – Приклад керованої деградації з високою інтенсивністю виникнення відмов

У цьому випадку розглядається зворотна ситуація. Система має у своєму розпорядженні обмежений запас процедур відновлення, що призводить до часткової втрати якості при кожній відмові. Більша інтенсивність прояву дефектів

високої тяжкості спричиняє стрімке вичерпування ресурсу відновлювання, що швидше приводить систему до стану відмови без можливості відновлення.

У спрощеному вигляді, коли детально не розглядається час на реконфігурацію та перехідні процеси, діаграма деградації матиме вигляд, представлений на Рисунку 9.

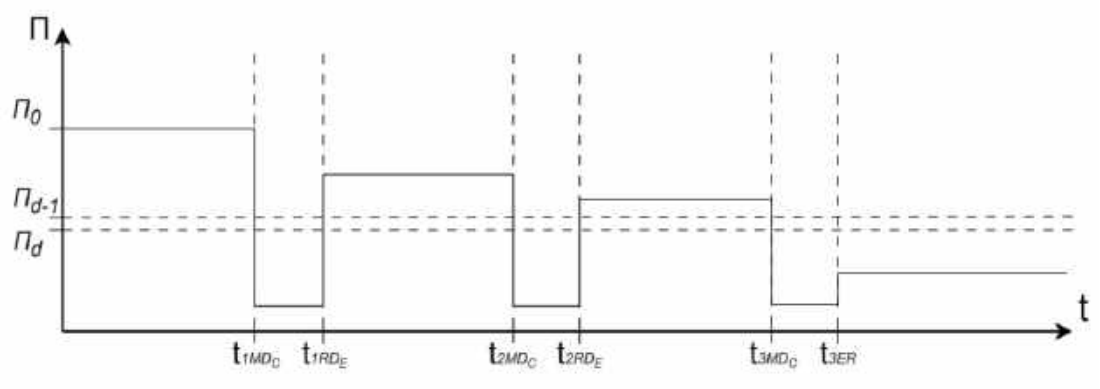

Рисунок 2.9 – Спрощений приклад керованої деградації

Спрощена форма дозволяє наочно прослідкувати динаміку зміни рівня якості системи з протягом часу. Так, залежно від кількості перехідних рівнів якості, є можливість оцінити плавність протікання процесу деградації. Залежно від кількості та розміру прогалин між моментами відмови та відновлення є можливість оцінити час, коли система знаходилась у стані простою.

Отже, керована багаторівнева деградація передбачає процес переведення програмовної системи зі стану S(t) в стан S (t+1) за допомогою:

а) виявлення відмови компонентів програмовної системи, яка знаходиться в момент часу t в стані S(t) (збою або сталої відмови), $Compi \in MComp$;

б) її ідентифікації за типом дефектів відповідно до виразу (2.15);

в) вибору і реалізації відповідної процедури реконфігурації $Rec_j \in MRec$ (2.16);

г) фіксації і запам'ятовування стану, в який перейшла система після відмови і реконфігурації S(t+1).

2.1.4 Розроблення структурних схем надійності та живучості програмовних пристроїв

Виходячи з опису множини компонентів ПП залежно від домінуючого в них типу дефектів, є можливість прослідкувати залежність надійності розгорнутої системи від впровадження засобів реконфігурації [63]. Для цього важливо представити систему у вигляді структурних схем надійності (ССН). Для цього, розташуємо залежні одне від одного компоненти системи відповідно до груп домінуючих у них типів дефектів. Таку структурну схему надійності представлено на Рисунку 2.1.

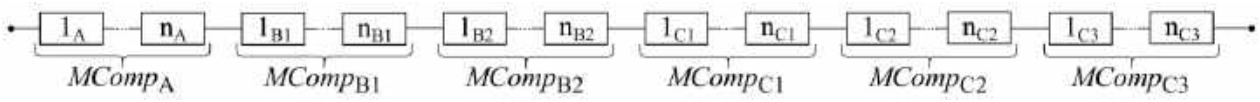

Рисунок 2.10 – Структурна схема надійності реконфігуропридатної системи

Ймовірність безвідмовної роботи системи(ПП), якщо не враховувати аспект багаторівневої керованої деградації, дорівнює

$$P_{ПП}(t) = \prod_{i=1}^{n_A} p_{Ai}(t) * \prod_{i=1}^{n_{B1}} p_{B1i}(t) * \prod_{i=1}^{n_{B2}} p_{B2i}(t) * \prod_{i=1}^{n_{C1}} p_{C1i}(t) \prod_{i=1}^{n_{C2}} p_{C2i}(t) * \prod_{i=1}^{n_{C3}} p_{C3i}(t). \quad (2.25)$$

Зважаючи на те, що відновлення у обчислювальних системах може відбуватися завдяки внутрішнім засобам відновлення[64], для ряду компонентів (2.4) реконфігурація може не задіюватись. Таким чином, множина $MComp_A$ не буде брати участь у відновленні через реконфігурацію, що можна позначити у структурній схемі надійності, наведеній на Рисунку 2.11.

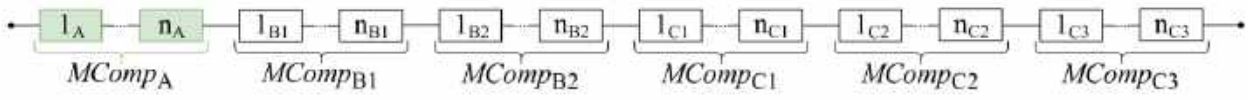

Рисунок 2.11 – Структурна схема надійності реконфігуропридатної системи з самовідновлюванням компонентів множини $MComp_A$

Ймовірність збереження працездатності системи при можливості самостійного відновленні частини компонентів дорівнює

$$P_{ПП/A}(t) = \prod_{i=1}^{n_{B1}} p_{B1i}(t) * \prod_{i=1}^{n_{B2}} p_{B2i}(t) * \prod_{i=1}^{n_{C1}} p_{C1i}(t) * \prod_{i=1}^{n_{C2}} p_{C2i}(t) * \prod_{i=1}^{n_{C3}} p_{C3i}(t) \quad . \qquad (2.26)$$

Для решти множин необхідним є введення контролюючого засобу, який буде забезпечувати виконання відновлювальних процедур. Згідно з характером відновлення, що залежить від множин використовуваних процедур реконфігурації (2.19 – 2.24), для кожної групи компонентів буде вводитись відповідний елемент реконфігурації. Таким чином, множини $MComp_{B1}$ та $MComp_{C1}$ будуть обслуговуватись реконфігуруючим елементом Rec_D, що забезпечує виконання повного відновлення задіяних компонентів. Зв'язок реконфігуруючого елемента з групою обслуговуваних ним компонентів можна позначити у структурній схемі надійності, наведеній на Рисунку 2.12.

Рисунок 2.12 – Структурна схема надійності системи з реконфігурацією за процедурою D з використанням засобів Rec_D при відмовах компонентів множин $MComp_{B1}$ та $MComp_{C1}$

Ймовірність збереження працездатності системи(ПП) при повному відновленні, якщо враховувати аспект багаторівневої керованої деградації, дорівнює

$$P_{ПП/A,B1}(t) = \prod_{i=1}^{n_{B2}} p_{B2i}(t) * \prod_{i=1}^{n_{C1}} p_{C1i}(t) * \prod_{i=1}^{n_{C2}} p_{C2i}(t) * \prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecD}(t), \qquad (2.27)$$

$$P_{ПП/A,C1}(t) = \prod_{i=1}^{n_{B1}} p_{B1i}(t) * \prod_{i=1}^{n_{B2}} p_{B2i}(t) * \prod_{i=1}^{n_{C1}} p_{C1i}(t) * \prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecD}(t), \qquad (2.28)$$

$$P_{ПП/A,B1,C1}(t) = \prod_{i=1}^{n_{B2}} p_{B2i}(t) * \prod_{i=1}^{n_{C2}} p_{C2i}(t) * \prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecD}(t) \, . \qquad (2.29)$$

Оскільки необхідно, щоб

$$P_{ПП/A,B1}(t) \geq P_{ПП/A}(t), \quad (2.30)$$

$$P_{ПП/A,B1}(t) \geq P_{ПП/A}(t), \quad (2.31)$$

$$P_{ПП/A,B1,C1}(t) \geq P_{ПП/A}(t), \quad (2.32)$$

реконфігурація B1 та C1 буде доцільною, якщо

$$p_{RecD}(t) > \frac{P_{ПП/A}(t)}{\prod_{i=1}^{n_{B2}} p_{B2i}(t) * \prod_{i=1}^{n_{C2}} p_{C2i}(t) * \prod_{i=1}^{n_{C3}} p_{C3i}(t)}, \quad (2.33)$$

$$p_{RecD}(t) > \prod_{i=1}^{n_{B1}} p_{B1i}(t) * \prod_{i=1}^{n_{C1}} p_{C1i}(t).$$

За аналогією до попереднього випадку, для множин $MComp_{B2}$ та $MComp_{C2}$ буде вводитись елемент реконфігурації Rec_E, що забезпечує виконання часткового відновлення відповідних до нього компонентів. Таким чином, структурній схема надійності матиме вигляд, представлений на Рисунку 2.13.

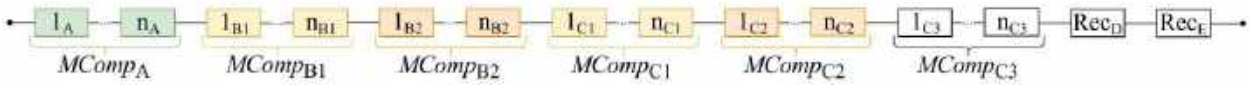

Рисунок 2.13 – Введення елементу реконфігурації Rec_E для множин $MComp_{B2}$ та $MComp_{C1}$

Ймовірність збереження працездатності ПП при частковому відновленні дорівнює

$$P_{ПП/A,B1,B2,C1}(t) = \prod_{i=1}^{n_{C2}} p_{C2i}(t) * \prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecD}(t) * p_{RecE}(t) \quad (2.34)$$

$$P_{ПП/A,B1,C1,C2}(t) = \prod_{i=1}^{N_{B2}} p_{B2i}(t) * \prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecD}(t) * p_{RecE}(t) \quad (2.35)$$

$$P_{ПП/A,B1,B2,C1,C2}(t) = \prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecD}(t) * p_{RecE}(t) \quad (2.36)$$

Оскільки необхідно, щоб

$$P_{ПП/A,B1,B2,C1}(t) \geq P_{ПП/A,B1,C1}(t), \quad (2.37)$$

$$P_{ПП/A,B1,C1,C2}(t) \geq P_{ПП/A,B1,C1}(t), \quad (2.38)$$

$$P_{ПП/A,B1,B2,C1,C2}(t) \geq P_{ПП/A,B1,C1}(t), \quad (2.39)$$

реконфігурація B2 та C2 буде доцільною, якщо

$$p_{RecE}(t) > \frac{P_{ПП/A,B1,C1}(t)}{\prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecD}(t)} \qquad (2.40)$$

$$p_{RecE}(t) > \prod_{i=1}^{n_{B2}} p_{B2i}(t) * \prod_{i=1}^{n_{C2}} p_{C2i}(t)$$

У випадку множини $MComp_{C3}$ відновлення системи при їх відмові не вважається можливим. Однак, через те, що дефекти можуть впливати не тільки на працездатність самої системи, я й на цілісність складових, що знаходяться у оточенні місця виникнення дефекту, є сенс забезпечити ізоляцію дефектного фрагмента пристрою[65]. Для цього, вводиться елемент реконфігурації Rec_F. Слід зазначити, що на відміну від попередніх елементів реконфігурації, елемент Rec_F обслуговує не лише множину $MComp_{C3}$. До його зони впливу також належать і інші елементи реконфігурації. Виходячи з цього, фінальна структурна схема надійності матиме вигляд, представлений на Рисунку 14.

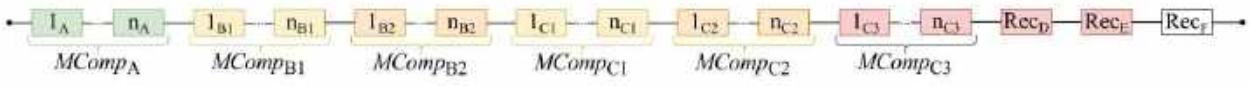

Рисунок 2.14 – Введення елементу реконфігурації Rec_F для множини $MComp_{C3}$ та $Rec_{D,E}$

Ймовірність збереження безпечного стану системи(ПП) при відмові критичних елементів дорівнює

$$P_{ПП/A,B,C1,C2,C3}(t) = p_{RecD}(t) * p_{RecE}(t) * p_{RecF}(t) \qquad (2.41)$$

$$P_{ПП/A,B,C1,C2,RecD}(t) = \prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecE}(t) * p_{RecF}(t) \qquad (2.42)$$

$$P_{ПП/A,B,C1,C2,RecE}(t) = \prod_{i=1}^{n_{C3}} p_{C3i}(t) * p_{RecD}(t) * p_{RecF}(t) \qquad (2.43)$$

$$P_{ПП/A,B,C,RecE,RecD}(t) = p_{RecF}(t) \qquad (2.44)$$

Оскільки необхідно, щоб

$$P_{ПП/A,B,C1,C2,C3}(t) \geq P_{ПП/A,B,C1,C2}(t), \quad (2.45)$$

$$P_{ПП/A,B,C1,C2,RecD}(t) \geq P_{ПП/A,B,C1,C2}(t), \quad (2.46)$$

$$P_{ПП/A,B,C1,C2,RecE}(t) \geq P_{ПП/A,B,C1,C2}(t), \quad (2.47)$$

$$P_{ПП/A,B,C,RecE,RecD}(t) \geq P_{ПП/A,B,C1,C2}(t) \quad (2.48)$$

консервація буде доцільною, якщо

$$p_{RecF}(t) > P_{ПП/A,B,C1,C2}(t)$$
$$p_{RecE}(t) > \prod_{i=1}^{n_{C3}} p_{C3i}(t) * \quad p_{RecD}(t) * p_{RecE}(t) \quad (2.49)$$

Отже, виходячи зі структурної схеми надійності можна зробити висновок, що реконфігурація потребує модифікації програмовного пристрою за рахунок впровадження засобів реконфігурації згідно схеми надійності. Через це до засобів реконфігурації висуваються високі вимоги щодо власної надійності, оскільки вони відносяться до класу компонентів, відмова котрих є критичною для системи.

2.2 Розроблення та дослідження аналітичних моделей надійності програмовних пристроїв

2.2.1 Розроблення аналітичних моделей надійності

Для демонстрації фізичних залежностей для розглянутих параметрів ПП з БРД розглядатиметься аналітична модель, заснована на його ймовірності безвідмовної роботи. Відповідно до внеску ЗР до моделі, буде розглянуто 3 варіанти розвитку ситуації, а саме

- Pc(cleared), група ймовірностей знаходження ПП у станах деградації відповідна до кількості рівнів деградації, без впливу показників надійності ЗР.

- Pp(partial), група ймовірностей знаходження ПП у станах деградації відповідна до кількості рівнів деградації з частковим урахуванням роботи ЗР. В такому випадку ЗР не враховується при на початковому стані ССН. ЗР підключається і працює після настання першої відмови ПП.

- Pa(affected), група ймовірностей знаходження ПП у станах деградації відповідна до кількості рівнів деградації з урахуванням роботи ЗР. В такому випадку ЗР враховується ССН з самого початку роботи ПП.

Для визначення початкових величин ймовірностей деградації використовуватиметься Закон Пуассона для розподілу одиничних випадкових величин у часі. Початкові ймовірності Pc_0 та Pp_0 визначатимуться формулами (2.50) та (2.51), де використовується тільки інтенсивність деградації.

$$Pc_0(t) = e^{(-\lambda d_0 t)}, \quad (2.50)$$

$$Pp_0(t) = Pc_0(t), \quad (2.51)$$

$$Pa_0(t) = Pc_0(t) * e^{(-\lambda r_0 t)} = e^{(-\lambda d_0 t)} * e^{(-\lambda r_0 t)} = e^{(-(\lambda r_0 + \lambda d_0)\, t)}, \quad (2.52)$$

У випадку Pa_0, де передбачено вплив ЗР на працездатність самого ПП, для визначення використовуватиметься комплексна інтенсивність виникнення події, а саме поєднання інтенсивностей деградації ПП та ЗР(див. формулу 2.52). Графічно, момент переходу між рівнями деградації в аналітичній моделі представлено на рисунку 2.15

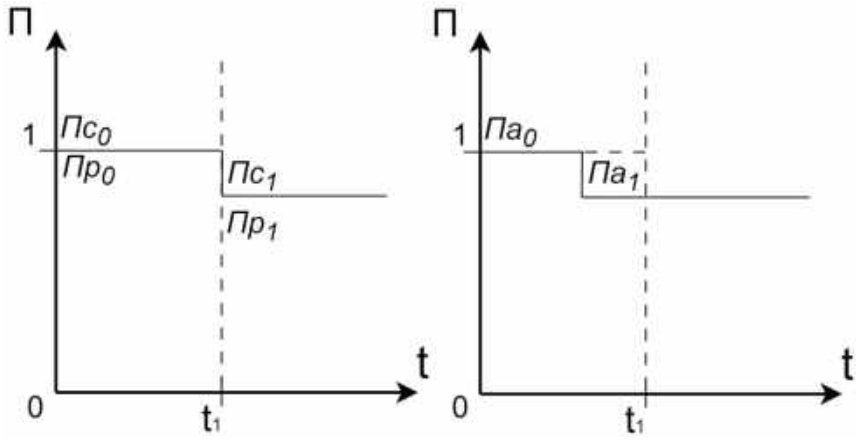

Рисунок 2.15 – Різниця впливу ЗР між Пр та Па деградаціями

При подальшому розвитку деградації, її ймовірності представлятиме добуток ймовірності поточного стану, та ймовірності виходу зі станів, які приводять до виникнення поточного.

$$Pc_1(t) = e^{(-\lambda d_1 t)} * (1 - Pc_0(t)), \quad (2.53)$$

$$Pp_1(t) = e^{(-(\lambda r_1 + \lambda d_1)\, t)} * (1 - Pp_0(t)) , \quad (2.54)$$

$$Pa_1(t) = e^{(-(\lambda r_1 + \lambda d_1)\, t)} * (1 - Pa_0(t)). \quad (2.55)$$

У випадку Pp_1, де передбачено прояв впливу ЗР на працездатність ПП, для визначення використовуватиметься комплексна інтенсивність, що складатиметься з інтенсивностей деградації ПП та ЗР(див. формулу 2.54). Момент переходу між першим($П_1$) та другим($П_2$) рівнями деградації згідно формул 2.56-2.58 в аналітичній моделі представлено на рисунку 2.16

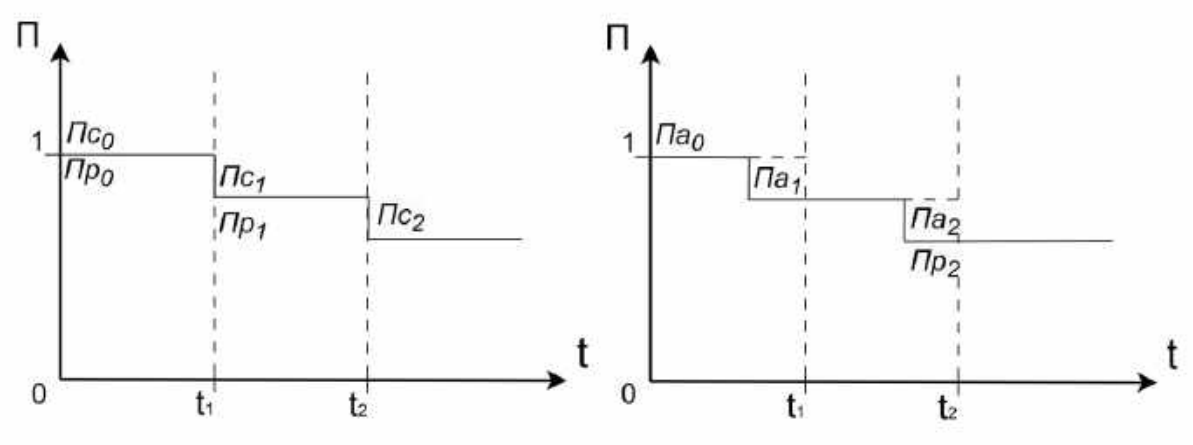

Рисунок 2.16 – Різниця розвитку деградацій Pc та Pp

$$Pc_2(t) = e^{(-\lambda d_2 t)} * (1 - Pc_0(t)) * (1 - Pc_1(t)), \quad (2.56)$$

$$Pp_2(t) = e^{(-(\lambda r_2 + \lambda d_2)\, t)} * (1 - Pp_0(t)) * (1 - Pp_1(t)), \quad (2.57)$$

$$Pa_2(t) = e^{(-(\lambda r_2 + \lambda d_2)\, t)} * (1 - Pa_0(t)) * (1 - Pa_1(t)), \quad (2.58)$$

Подальший розвиток деградації призводиться аналогічним чином, що можна описати формулами 2.59-2.61 для переходу між другим(П2) та третім(П3) рівнями.

$$Pc_3(t) = e^{(-\lambda d_3 t)} * (1 - Pc_0(t)) * (1 - Pc_1(t)) * (1 - Pc_2(t)), \quad (2.59)$$

$$Pp_3(t) = e^{(-(\lambda r_3 + \lambda d_3)\, t)} * (1 - Pp_0(t)) * (1 - Pp_1(t)) * (1 - Pp_2(t)), \quad (2.60)$$

$$Pa_3(t) = e^{(-(\lambda r_3 + \lambda d_3)\, t)} * (1 - Pa_0(t)) * (1 - Pa_1(t)) * (1 - Pa_2(t)), \quad (2.61)$$

Відповідно, перехід між третім(П3) та четвертим(П4) рівнями описуватиметься формулами 2.62 - 2.64.

$$Pc_4(t) = e^{(-\lambda d_4 t)} * (1 - Pc_0(t)) * (1 - Pc_1(t)) * (1 - Pc_2(t)) * (1 - Pc_3(t)), \quad (2.62)$$

$$Pp_4(t) = e^{(-(\lambda r_4 + \lambda d_4) t)} * (1 - Pp_0(t)) * (1 - Pp_1(t)) * (1 - Pp_2(t)) * (1 - Pp_3(t)), \quad (2.63)$$

$$Pa_4(t) = e^{(-(\lambda r_4 + \lambda d_4) t)} * (1 - Pa_0(t)) * (1 - Pa_1(t)) * (1 - Pa_2(t)) * (1 - Pa_3(t)). \quad (2.64)$$

Таким чином, розроблені моделі описують зміну ймовірності безвідмовної роботи при переході між рівнями деградації. Характерною рисою переходу між рівнів є збільшення кількості задіяних ймовірностей, що відображають шлях до розглядуваного рівня. Залежно від застосованих характеристик ПП та ЗР, характер розподілу вірогідностей змінюється відносно рівня деградації, що потребує додаткового дослідження.

2.2.2 Дослідження аналітичних моделей надійності

Для описаних моделей 2.50 - 2.64 з метою дослідження характеру змін у розподілі ймовірностей безвідмовної роботи, було проведено симуляцію з використанням початкових величин, представлених таблицею 2.2.

Таблиця 2.2 – Значення інтенсивностей відмов ЗР та ПП

№(i)	Інтенсивності деградації ПП(λd_i)	Інтенсивності деградації ЗР(λr_i)
0	0.00012	0.00012
1	0.00012	0.00012
2	0.00009	0.00006
3	0.00006	0.00003
4	0.00003	0.00001

При застосуванні описаних формулами 2.50, 2.53, 2.56, 2.59, 2.62 рівнянь на інтервалі часу у 60 000 годин (6.8 років) з кроком у 1000 годин, результат розподілу ймовірностей виглядатиме наступним чином, зображеним на рисунку 2.17.

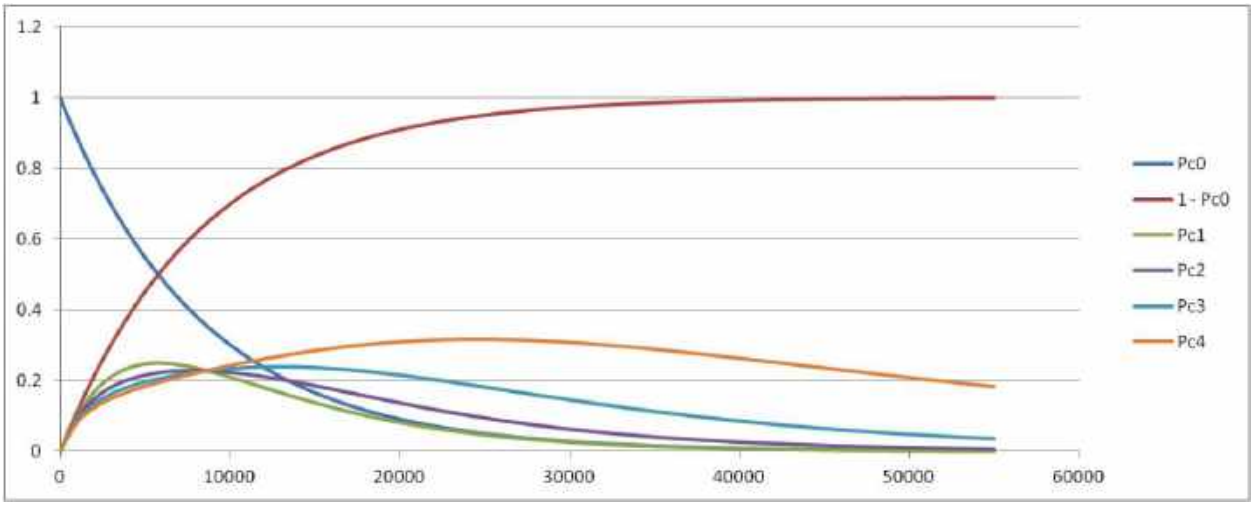

Рисунок 2.17 – Розподіл величин при зміні рівнів деградації ПП без впливу ЗР

Розподіл демонструє зниження ймовірності безвідмовної роботи при деградації на перший рівень деградації. Однак, вже після другого рівня, ймовірність підвищується і перевищує початковий показник. Відповідно, на третьому та четвертому рівнях тенденція до підвищення ймовірностей зберігається, що подовжує тривалість їх спаду.

При застосуванні описаних формулами 2.51, 2.54, 2.57, 2.60, 2.63 рівнянь на інтервалі часу у 60 000 годин (6.8 років) з кроком у 1000 годин, результат розподілу ймовірностей виглядатиме наступним чином, зображеним на рисунку 2.18.

Розподіл демонструє суттєве зниження ймовірності безвідмовної роботи при деградації на першому та другому рівні деградації. Однак, після третього рівня, ймовірність підвищується і перевищує початковий показник. Відповідно, після на четвертому рівнях тенденція до підвищення ймовірності зберігається, що подовжує тривалість її спаду.

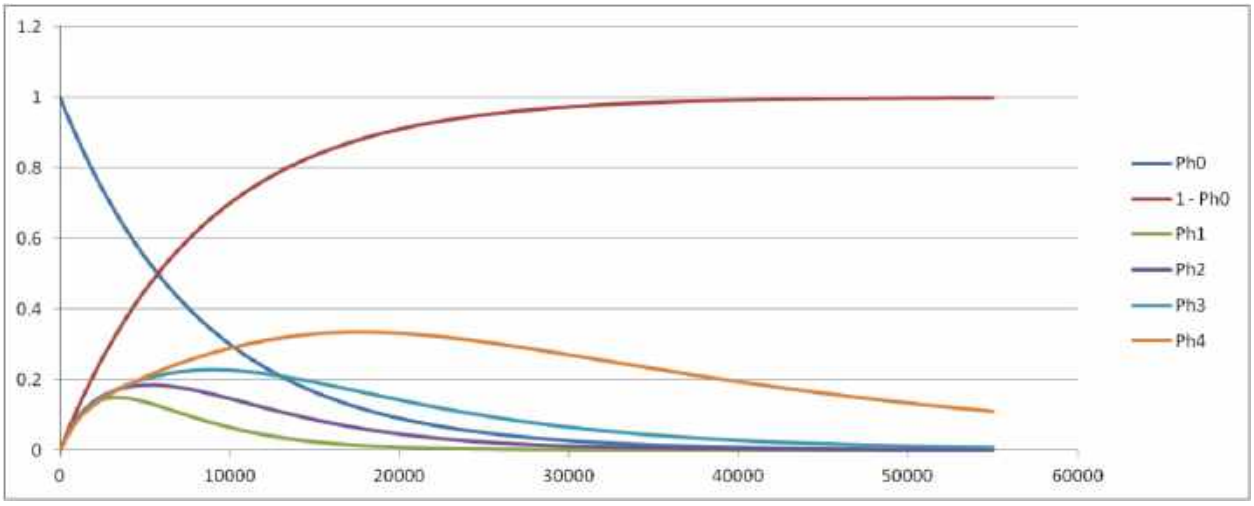

Рисунок 2.18 – Розподіл величин при зміні рівнів деградації ПП з частковим впливом ЗР

При застосуванні описаних формулами 2.52, 2.55, 2.58, 2.61, 2.64 рівнянь на інтервалі часу у 60 000 годин (6.8 років) з кроком у 1000 годин, результат розподілу ймовірностей виглядатиме наступним чином, зображеним на рисунку 2.19.

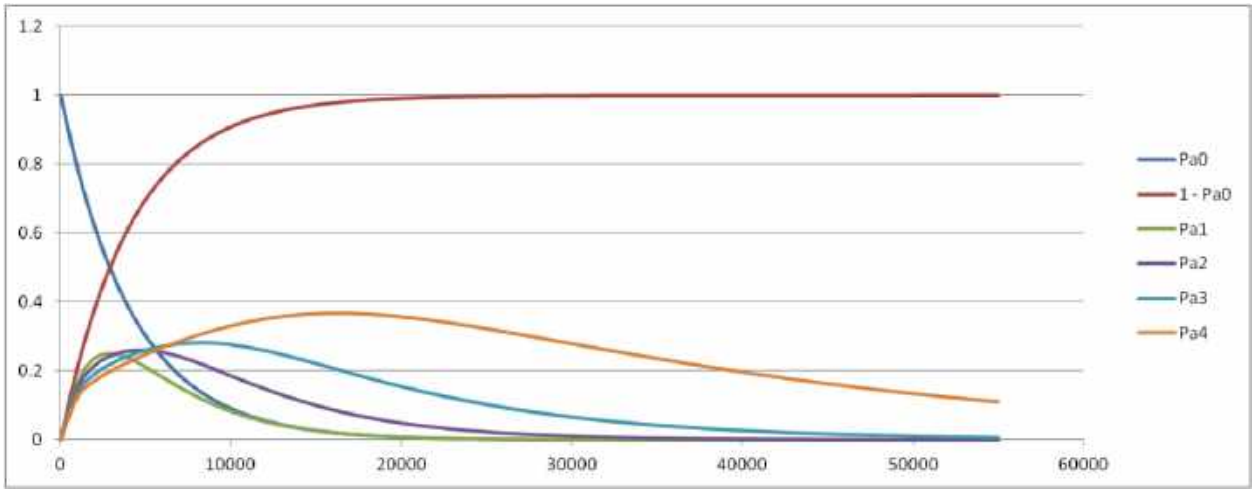

Рисунок 2.19 – Розподіл величин при зміні рівнів деградації ПП з впливом ЗР

Розподіл демонструє несуттєве зниження ймовірності безвідмовної роботи при деградації на перший рівень деградації, що усувається у перші 10 000 годин роботи. Завдяки тимчасовому зростанню ймовірностей на другому та наступних рівнях , що подовжує тривалість їх спаду та подовжує тривалість функціонування ПП.

Для варіантів розподілу Pc та Pa їх різниця у ймовірностях виглядатиме наступним чином, представленим на рисунку 2.20.

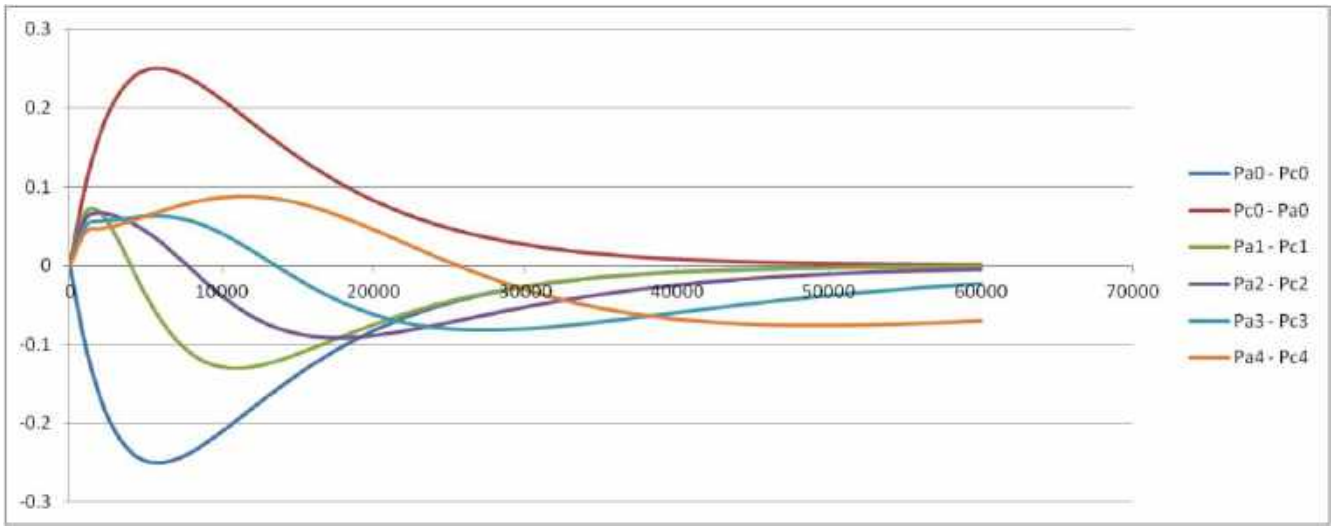

Рисунок 2.20 – Різниця розподілу величин при зміні рівнів деградації ПП з впливом ЗР та без нього

Згідно наведеного графіку, прослідковуються наступні відмінності. Початковий показник безвідмовної роботи моделі Pa значно поступається аналогічному показнику моделі Pc. Це обумовлено впливом ЗР на початкову ймовірність безвідмовної роботи у Pa. Однак, при деградації, вже з перших рівнів Pa набуває суттєвих, однак не тривалих темпів зросту у порівнянні з аналогічними у моделі Pc. На довгому проміжку часу, модель Pa втрачає інтенсивність росту функції, переходячи у помірний спад, який залежить від рівня деградації. Чим більший рівень деградації – тим плавніший спад. Після 30 000годин (3.4 роки) моделі Pc починає переважувати модель Pa за всіма показниками.

2.3 Розроблення методу аналізу реконфігуропридатності та реконфігурації програмовних пристроїв при відмовах

2.3.1 Визначення умов реконфігуропридатності

Важливим кроком впровадження реконфігурації у якості інструмента відновлення є визначення властивостей пристрою, до якого це впровадження є

можливим [66]. Розглянувши множину компонентів на підставі розроблених моделей, можна зазначити, що процес відновлення не завжди супроводжується виконанням процедури реконфігурації. Так, для випадку компонентів множини $MComp_A$ відновлення виконується завдяки вбудованим у компонент засобам. Зазначимо також, що реконфігурація не завжди застосовується для відновлення. У випадку компонентів множин $MComp_{C3}$ відновлення не відбувається. Однак, відповідна до неї множина Rec_F реконфігурація забезпечує ізоляцію місця виникнення небезпечного дефекту та переведення ПП у захищений (безпечний) стан.

Для визначення реконфігуропридатності ПП, слід розглядати тільки ті випадки, де реконфігурація використовується для відновлення працездатності (повної або часткової). Система є потенційно реконфігуропридатною, якщо

$$\exists!\ Comp_i : Comp_i \in \{ MComp_{B1} \cup MComp_{B2} \cup MComp_{C1} \cup MComp_{C2}\}, \quad (2.50)$$

або

$$MComp_{B1} \cup MComp_{B2} \cup MComp_{C1} \cup MComp_{C2} \neq \varnothing. \quad (2.51)$$

У випадку впровадження інструментів до програмовної системи, однієї наявності реконфігурованих компонентів буде не достатньо. Для проведення відновлення необхідні засоби, які будуть виконувати власне реконфігурацію. Можна виділити множину процедур реконфігурації (2.19), яка забезпечує виокремлення і відключення дефектів або пом'якшення їх наслідків. Виходячи з цього, система є реконфігуропридатною, якщо

$$\exists!\ Rec : Rec_i \in \{ MComp_{B1} \cup MComp_{B2} \cup MComp_{C1} \cup MComp_{C2}\}, \quad (2.52)$$

що обумовлюється наявністю хоча б одного засобу реконфігурації, та хоча б одного компонента, реконфігурація якого підтримується наявним засобом. Множини процедур реконфігурації мають бути відповідними до множин компонентів, які вони реконфігурують:

$$(MComp_{B1} \cup MComp_{C1} \neq \varnothing)\ \&\ (\exists Rec_D), \quad (2.53)$$

$$(MComp_{B2} \cup MComp_{C2} \neq \varnothing)\ \&\ (\exists Rec_E). \quad (2.54)$$

Отже, при розгляді ознак реконфігуропридатності слід розрізняти потенційну реконфігуропридатність та реалізовану. За наявності засобів

реконфігурації виникає можливість часткової реалізації реконфігуропридатності. Чим більша частка процедур реконфігурації для наявних компонентів з множини (2.16), тим вищим є показник реконфігуропридатності пристрою.

2.3.2 Розроблення та аналіз метрик реконфігуропридатності

На підставі розроблених моделей (2.16) та формалізації понять реконфігуропридатності пропонуються її кількісні показники – метрики реконфігуропридатності (*MtrRCA*). За об'ємом реконфігуропридатності розрізняють метрики, наведені нижче.

1. Потенційна реконфігуропридатність (*MtrRCAp*) визначає частку придатних до реконфігурації компонентів. Згідно розподілу компонентів відносно домінуючого типу дефектів розрізняють:

- потенційну реконфігуропридатність за дефектами типу B1, що представляється у вигляді виразу

$$MtrRCAp_{B1} = \frac{|MComp_{pB1}|}{|MComp|}, \qquad (2.55)$$

де $MComp_{pB1}$ множина потенційно придатних до реконфігурації компонентів з дефектами типу B1;

- потенційну реконфігуропридатність за дефектами типу B2, що представляється у вигляді виразу

$$MtrRCAp_{B2} = \frac{|MComp_{pB2}|}{|MComp|}, \qquad (2.56)$$

де $MComp_{pB2}$ множина потенційно придатних до реконфігурації компонентів з дефектами типу B2;

- потенційну реконфігуропридатність за дефектами типу C1, що представляється у вигляді виразу

$$MtrRCAp_{C1} = \frac{|MComp_{pC1}|}{|MComp|}, \qquad (2.57)$$

де $MComp_{pC1}$ множина потенційно придатних до реконфігурації компонентів з дефектами типу С1.

- потенційну реконфігуропридатність за дефектами типу С2, що представляється у вигляді виразу

$$MtrRCAp_{C2} = \frac{|MComp_{pC2}|}{|MComp|}, \quad (2.58)$$

де $MComp_{pC2}$ множина потенційно придатних до реконфігурації компонентів з дефектами типу С2.

Таким чином, загальна множина *MtrRCAp* потенційної реконфігуропридатності визначатиметься виразом

$$MtrRCAp = \frac{|MComp_{pB1} \cup MComp_{pB2} \cup MComp_{pC1} \cup MComp_{pC21}|}{|MComp|}. \quad (2.59)$$

2. Реалізована реконфігуропридатність (*MtrRCAg*) визначає дійсну долю придатних до реконфігурації компонентів, залежну від доступності процедур реконфігурації (2.18). Згідно розподілу компонентів відносно домінуючого типу дефектів розрізняють:

- реалізовану реконфігуропридатність за дефектами типу В1, що представляється у вигляді виразу

$$MtrRCAg_{B1} = \frac{|MComp_{gB1}|}{|MComp|}, \quad (2.60)$$

де $MComp_{B1g} \subset MComp_{B1}$, таке що його елементи можуть бути реконфігуровані при відмовах за наявності процедури Rec_D;

- реалізовану реконфігуропридатність за дефектами типу В2, що представляється у вигляді виразу

$$MtrRCAg_{B2} = \frac{|MComp_{gB2}|}{|MComp|}, \quad (2.61)$$

де $MComp_{B2g} \subset MComp_{B2}$, таке що його елементи можуть бути реконфігуровані при відмовах за наявності процедури Rec_E;

- реалізовану реконфігуропридатність за дефектами типу C1, що представляється у вигляді виразу

$$MtrRCAg_{C1} = \frac{|MComp_{gC1p}|}{|MComp|}, \quad (2.62)$$

де $MComp_{C1g} \subset MComp_{C1}$, таке що його елементи можуть бути реконфігуровані при відмовах за наявності процедур Rec_D.

- реалізовану реконфігуропридатність за дефектами типу C2, що представляється у вигляді виразу

$$MtrRCAg_{C1} = \frac{|MComp_{gC2p}|}{|MComp|}, \quad (2.63)$$

де $MComp_{C2g} \subset MComp_{C2}$, таке що його елементи можуть бути реконфігуровані при відмовах за наявності процедур Rec_E.

Таким чином, загальна множина *MtrRCAg* реалізованої реконфігуропридатності визначатиметься виразом

$$MtrRCAg = \frac{|MComp_{gB1} \cup MComp_{gB2} \cup MComp_{gC1} \cup MComp_{gC2}|}{|MComp|}. \quad (2.64)$$

Окрім метрик об’єму РП важливими є часові метрики ПП [67]. Слід розрізняти інтервальні показники та моменти виникнення події.

1. Інтервальні показники часу показують часові проміжки між виникненням подій під час роботи системи. Серед них можна виділити:

- часовий інтервал стабільного стану (Δt_S), що відображає час неперервного знаходження системи у працездатному стані, або стані відмови;
- перехідний часовий інтервал (Δt_J), що означує час переходу системи між сталими станами.

2. Моменти виникнення подій демонструють точки у часі, в яких система починає зазнавати зміни у власному стані. Зазвичай характеризують припинення перебування системи у сталому стані. Серед них можна виділити:

- парні моменти(t_{SR}, t_{ER}), які передбачають наявність зв'язку між групою моментів. Так, момент початку переходу між станами супроводжується моментом закінчення цього переходу;
- одиничні моменти(t_{MD}, t_{RD}), що передбачають мало пов'язані події, виникнення котрих не носить систематичного характеру.

2.4 Послідовність оцінювання реконфігуропридатності, надійності та живучості та реконфігурації програмовних пристроїв

2.4.1 Послідовність оцінювання реконфігуропридатності

При впровадженні реконфігурації у якості інструменту відновлення, оцінювання РП та живучості ПП виконується у такій послідовності.

1. Проведення аналізу програмовної системи[68]. Означення типів компонентів та їх дефектів $Comp_i \sim Def_i$ згідно виразу (2.2).
2. Формування опису програмовної системи відповідно до виразів (2.14) і (2.16).
3. Проведення оцінювання потенційної реконфігуропридатності з використанням метрик за виразами (2.55-2.59).
4. Аналіз можливостей застосування процедур реконфігурації за виразами (2.19 – 2.24).
5. Проведення оцінювання реалізованої реконфігуропридатності з використанням метрик за виразами (2.60-2.64).
6. Розроблення засобів реконфігурації (2.19 – 2.24)
7. Оцінювання часових метрик реконфігуропридатності (t_{SR}, t_{ER}, Δt_S)
8. Аналіз отриманих результатів та їх відповідності вимогам до РП [69].
9. Оцінювання показників надійності та живучості реконфігуропридатних ПП та їх відповідності вимогам [70].
10. Коригування рішень, якщо значення показників не відповідатиме вимогам до ПП. Варіантами такого коригування є наступні:

- перегляд декомпозиції множини компонентів ПП з точки зору перерозподілу за рівнями працездатності та можливостей відновлення;

- розгляд можливостей введення додаткових процедур реконфігурації та їх програмної підтримки задля підвищення надійності та живучості;

- аналіз можливостей пом’якшення вимог до показників ПП.

2.4.2 Послідовність реконфігурації програмовних пристроїв при відмовах

Під час контролю багаторівневої керованої деградації виконується повторювана послідовність дій для кожного з рівнів працездатності. Вона включає наступні етапи:

1) Перевірка пристрою на наявність пошкоджень, що забезпечується безперервною та безпомилковою роботою засобів контролю;

2) Ідентифікація дефекту за класифікацією наведеною класифікацією, відповідно до класу B1, B2, C1, C2, або C3;

3) Пошук конфігурації пристрою у множині процедур реконфігурації за класами D, E1, E2, E3 або F;

4) Застосування доступної конфігурації програмовного пристрою та відновлення.

2.5 Висновки до другого розділу

Основними результатами досліджень, які виконано у розділу є настпуні.

1. Розроблено моделі реконфігуропридатних програмовних пристроїв, що проводилося за наступними етапами:

- розроблення моделей дефектів, програмовного пристрою;

- розроблення комплексу процедур реконфігурації відповідно до моделей дефектів;

- дослідження моделей надійності за допомогою діаграм деградації;

- розроблення структурних схем надійності та живучості програмовних пристроїв з урахуванням засобів реконфігурації.

2. Розроблено та досліджено аналітичні моделі надійності ПП призводилось наступним чином:

- призводилась параметризація системи відповідно до розроблених структурних схем надійності;

- вираховувались ймовірності знаходження на рівнях втрати якісних характеристик під час деградації;

- проводився порівняльний аналіз часових характеристик переходів системи між рівнями залежно від способу виконання засобів реконфігурації.

3. Розроблено метод аналізу реконфігуропридатності та реконфігурації програмовних пристроїв при відмовах.

В ході визначення класів компонентів ПП було визначено наступні метрики реконфігуропридатності:

- метрики об’єму, що визначають потенційну та реалізовану реконфігуропридатність;

- метрики часу, що визначають інтервальні та моментні показники.

4. Визначено послідовності оцінювання реконфігуропридатності, надійності та живучості безпілотних систем.

В ході їх формування визначено ключові етапи проведення оцінювання ПП на наявність потенційної та реалізованої реконфігуропридатності за рахунок визначених метрик. Для проведення реконфігурації було запропоновано послідовність дій залежно від виникненні відмов з ріщним ступенем впливу на працездатність.

Таким чином, в розділі отримано перший і другий нові наукові результати:

- вперше запропоновано метод керованої багаторівневої деградації програмовних пристроїв, який, на відміну від відомих, базується, по-перше, на формальному описі умов та аналізі реконфігуропридатності архітектури за допомогою спеціальних метрик об’єму та часу; по-друге, запропонованих процедурах реконфігурації, вибір і реалізація яких здійснюється залежно від типів дефектів з використанням визначеної множини конфігурацій, що забезпечує

підвищення надійності та живучості при виникненні та накопиченні відмов апаратних компонентів;

- удосконалено структурні та надійнісні моделі програмовних пристроїв з керованою багаторівневою деградацією шляхом декомпозиції множин внутрішніх компонентів залежно від впливу на працездатність та потенційної можливості програмно-керованої реконфігурації структури без втрати або з частковою втратою працездатності і якості виконання специфікованих функцій, що надає змогу формувати вимоги до засобів керування реконфігурацією та розраховувати показники безвідмовності та живучості при певних відмовах.

РОЗДІЛ 3
РОЗРОБЛЕННЯ ТА ДОСЛІДЖЕННЯ МОДЕЛЕЙ ТА МЕТОДУ ОЦІНЮВАННЯ ГОТОВНОСТІ ПРОГРАМОВНИХ ПРИСТРОЇВ БЕСПІЛОТНИХ СИСТЕМ З КЕРОВАНОЮ БАГАТОРІВНЕВОЮ ДЕГРАДАЦІЄЮ

3.1 Розроблення класифікатора моделей готовності програмовних пристроїв з багаторівневою деградацією

3.1.1 Обґрунтування ознак класифікації

Для програмовних пристроїв з багаторівневою деградацією існує ряд ознак, які впливають на їх поведінку під час роботи. При визначенні суттєвих ознак для контролю деградаційних процесів, їх кількість може відрізнятися в залежності від типу та умов експлуатації використовуваного ПП [71]. Однак, узагальнено можна виділити сім ознак для класифікації марковських моделей. Для зручності розгляду і подальшого аналізу, доцільно розподілити ці ознаки на декілька груп, що відповідатимуть зростанню складності використання ресурсу ПП:

1) наявність змінних параметрів ПП при деградації, що визначається зміною показників ПП після відновлення працездатності. Динаміка змін може слугувати індикатором адаптивності до умов роботи ПП, що дозволяє оцінити, як система реагує на деградацію. Це допомагає прогнозувати подальший розвиток деградаційних процесів та ефективно планувати заходи для відновлення працездатності;

2) наявність потенційної реконфігуропридатності, що визначається можливістю керування ресурсами системи з метою їх перерозподілу задля часткового або повного відновлення працездатності. Оскільки деградація може призводити до часткового або повного виходу з ладу компонентів, здатність пристрою реконфігуруватися, тобто перепризначити ресурси для відновлення працездатності, що дає змогу мінімізувати негативні наслідки деградації, підвищуючи надійність і стійкість системи до відмов;

3) характер реконфігуропридатності, який визначається відповідністю реконфігуровних компонентів до класів B, C1, C2 згідно наданої класифікації. Чітка ідентифікація частин системи, здатних до відновлення через перерозподіл функціональності, дозволяє супроводжуючим системам контролю та забезпечення реконфігурації оптимізувати частоту перевірок та кількість допустимих процедур реконфігурації;

4) врахування помилок засобів контролю, яке визначається способом організації контролю прояву дефектів, та реалізацією засобів контролю та діагностики. При моделюванні суттєвим є врахування помилок не лише досліджуваного ПП, але і засобів контролю. Через пряму залежність якості контролю з якістю оцінки стану ПП, мінімізація впливу засобів контролю і діагностики дозволяє зменшити кількість хибних та злоякісних спрацювань. Для випадків, коли повністю виключити вплив засобів контролю неможливо, його врахування при оцінці стану системи надасть кращої керованості самому процесові деградації;

5) відповідність змінних параметрів, що розрізняє характер деградації відповідно до змін параметрів. З кожним додатковим змінним параметром, марковська модель набуває змін за ще одним напрямком (виміром). Деградація ПП проходить у відповідності до змінних параметрів та їх сполук. "Вимірність" моделі залежить від кількості задіяних у деградації параметрів, а "глибина" кожного виміру – від відповідного ресурсу ПП та супроводжуючого обладнання;

6) комбінація залежних одне від одного змінних параметрів, яка визначається можливою одночасною змінністю параметрів і обмежуються їх кількістю. В окремих випадках параметри можуть бути взаємозалежними, що визначає їх належність до груп параметрів, де зміни відбувається комплексно. Означення таких груп дозволяє оптимізувати процедури контролю та реконфігурації при керуванні деградацією, що впливає на кількість "вимірів" моделі при побудові;

7) кратність резервування засобів реконфігурації, яка визначається запасом ресурсів, відмова яких не призводить до часткової або повної

працездатності. Резервування ресурсів є розповсюдженим та універсальним методом забезпечення стійкості системи до відмов. Кратність резервування вказує на кількість ресурсів, які можуть бути витрачені без порушення працездатності, що дозволяє зберегти функціональність системи у випадку часткових або, в окремих випадках, повних відмов.

Таким чином, для ефективного моделювання контролю деградації програмовних пристроїв з багаторівневою деградацією необхідно враховувати низку ознак, які визначають поведінку системи при відмовах. Представлений перелік ознак не є вичерпним, але достатнім для охоплення розповсюджених випадків відмови ПП. Для подальшого використання, приведений перелік буде використовуватися у складі класифікатора, що описує ключові сутності кожної з ознак, та зв'язки між ними.

3.1.2 Розроблення класифікатора та його використання для формування множини моделей

Важливим етапом є визначення системи взаємодії складових класифікації ознак. Для переліку з сімох ознак класифікації марковських моделей у відповідності до заданого порядку виділяють сутності, наведені нижче.

За наявності змінних параметрів розрізняють:

- однофрагментні моделі (ОФМ), в яких відсутні змінні параметри. В такому випадку деградація є одноріневою і множина станів не є повторюваною;
- мультифрагментні моделі (МФМ), в яких є змінні параметри. В такому випадку деградація є багаторівневою і множина станів є повторюваною. Такі множини повторюваних станів називаються фрагментами[72, 73].

При відсутності реалізації реконфігуропридатності, відновлення після відмови з частковою або повною втратою працездатності є неможливим, що не передбачає застосування керування БРД. Однак, при її наявності у системи є можливість до відновлення через реконфігурацію при наступних дефектах:

- дефекти групи B1(Def_{B1}). Ця група визначає сукупність дефектів, які призводять до стану часткової відмови некритичних ресурсів ПП. При цьому зберігається можливість повного відновлення працездатності;
- дефекти групи B2(Def_{B2}). Сукупність дефектів, які призводять до стану часткової відмови некритичних ресурсів ПП. За таких дефектів повне відновлення стає неможливим, однак зберігається можливість пом'якшення наслідків відмови до припустимого рівня;
- дефекти групи C1(Def_{C1}). Ця група визначає сукупність дефектів, які призводять до повної відмови некритичних ділянок системи. При цьому зберігається можливість повного відновлення працездатності;
- дефекти групи C2(Def_{C2}). Ця група визначає сукупність дефектів, які призводять до повної відмови некритичних ділянок системи. При цьому зберігається можливість пом'якшення наслідків відмови з частковим відновленням працездатності.

У випадку, коли засоби контролю та реконфігурації не враховуються, їх виконання вважається ідеальним або близьким до ідеального. Однак, наближеним до реальності вважається варіант, коли обслуговуючі реконфігурацію засоби є співставними за надійнісними характеристиками ПП, який вони супроводжують. При цьому описуються стани системи, які виникають у результаті хибних спрацювань системи контролю, діагностування та реконфігурації. Внаслідок цього передбачається виникнення наступних станів:

- стан прихованої відмови. Описує випадок, коли система контролю не відстежила виникнення дефекту, та не надала команди на проведення реконфігурації;
- стан хибної відмови. Описує випадок, коли система контролю надала команду на проведення реконфігурації без виникнення дефекту;
- стан некоректної ідентифікації відмови. Описує випадок, коли система контролю внаслідок виникнення дефекту надала команду до проведення реконфігурації, не дотичної до місця і/або виду дефекту.

Для сукупності ознак та зв’язків між ними, відповідно до ОФМ, частина класифікатора матиме вигляд, представлений схемою на рисунку 3.1

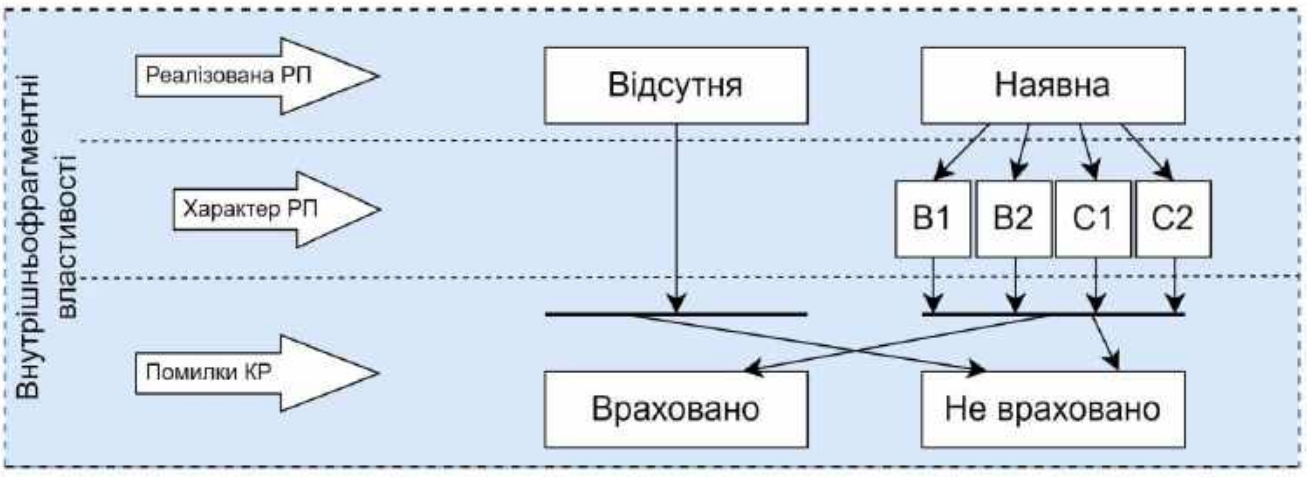

Рисунок 3.1 – Представлення ознак і сутностей ОФМ у класифікаторі

Для ознак МФМ притаманним є наявність параметрів, що змінюються протягом деградації. З кожним додатковим змінним параметром, марковська модель стає мультифрагментною за ще одним напрямком (виміром). Для програмовних пристроїв з керованою БРД визначимо наступні змінні параметри:

- кількість рівнів деградації ресурсів власне програмовного пристрою;
- кількість рівнів деградації засобів реконфігурації (ЗР);
- кількість рівнів деградації ресурсу реконфігурації (РР);
- кількість рівнів деградації засобів контролю (ЗК).

Якщо враховувати дефекти та вразливості програмних засобів (кодів електронних проєктів ПП), то змінними також можуть бути інтенсивності прояву відповідних дефектів та атак на вразливості при їх детектуванні та усуненні [74].

Додатково, групою змінних слід вважати комбінації залежних одне від одного змінних параметрів, що визначаються їх можливою одночасною змінністю і обмежуються кількістю змінних параметрів.

При резервуванні засобів реконфігурації ще одним параметром виступає кратність резервування, яка визначається запасом ресурсів, відмова яких не призводить до часткової або повної працездатності.

Виходячи з описаних груп ознак, побудовано класифікатор, представлений на рисунку 3.2.

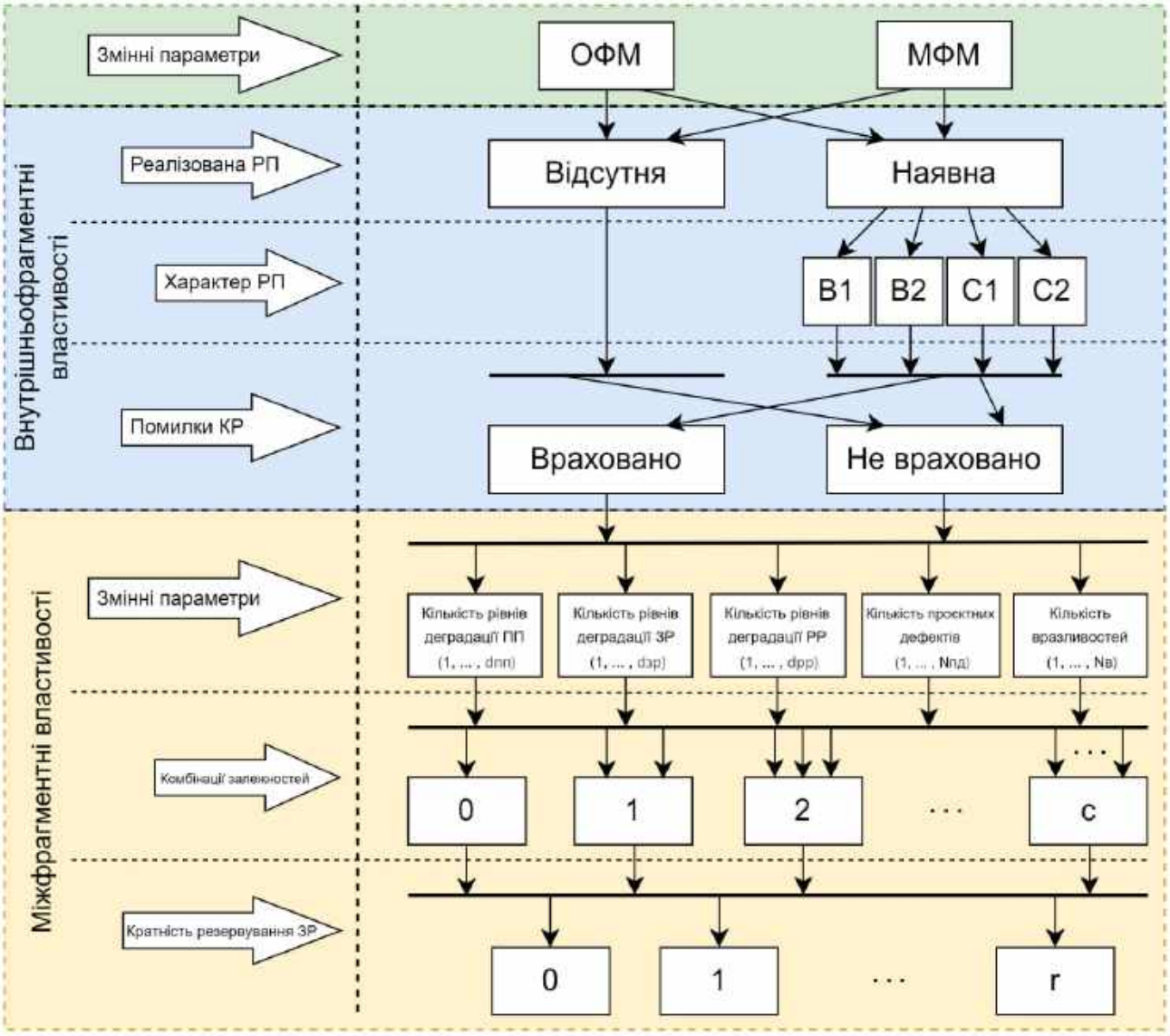

Рисунок 3.2 – Класифікатор моделей

Таке представлення надає можливість покроково аналізувати варіанти марковської моделі, враховуючи ознаки як одного фрагмента, що визначає кількість початкових станів системи та можливості використання внутрішнього ресурсу, так і ознаки множин фрагментів, де береться до уваги їх підпорядкованість одне одному з урахуванням змінних параметрів та характеру взаємодії.

3.2 Розроблення моделей готовності програмовних пристроїв

3.2.1 Множина однофрагментних моделей

В рамках розглянутої класифікації, важливим етапом буде дослідження впливу обраних факторів на структурні та поведінкові зміни марковських моделей. Пряма залежність складності моделі від задіяних ознак прослідковується як на рівні одного фрагменту(ОФМ), так і на рівні міжфрагментних зв'язків(МФМ). Для зручності, розглядатимемо базові моделі за зростанням їх складності, керуючись їх розділенням на множини ОФМ та МФМ. Так, першим буде розглянуто найпростіший варіант виконання ОФМ з наступною мінімальною кількістю станів:

Su (Up-state) – працездатний стан, що передбачає відповідність характеристик системи до виконання поставлених перед нею завдань. При цьому стані прояву дефектів системи не відбувається.

Sf (Failure-state) – стан відмови, що передбачає прояв дефектів, який у свою чергу викликає порушення нормального режиму роботи пристрою. Дефект, що призводить до виникнення цього стану, за класифікацією відноситься до типу C3(Def_{C3}), що вказує на відсутність можливості відновлення після його прояву.

Переходи між станами ОФМ характеризуються інтенсивностями виникнення відповідних до них подій. Для простішого випадку з двох станів, розглядається наступна інтенсивність:

λf – інтенсивність виникнення відмов програмовного пристрою.

Представлена ОФМ немає можливостей до самовідновлення або відновлення за рахунок зовнішніх засобів підвищення надійності. На практиці, подібні системи не є розповсюдженими, оскільки їх виконання виключає використання будь-яких елементів керування, у тому числі і керування живленням. Приближеним до реальності варто вважати варіант, де існує можливість самовідновлення системи. Модель такого вигляду, окрім описаних вище станів, буде включати наступні:

Se (Error-state) – стан збою, що усувається забезпеченням самого пристрою автоматично, або через мінімальне зовнішнє втручання, таке як перезавантаження джерела живлення.

Переходи між станами ОФМ характеризуються наступними інтенсивностями:

λe – інтенсивність виникнення збоїв ПП,

μe – інтенсивність відновлення після збоїв ПП.

В описаному випадку, модель вже має певну стійкість до подій втрати працездатності, які були спричинені деякими факторами зовнішнього середовища та несуттєвими змінами показників ПП.

Описані ОФМ характеризуються відсутністю реконфігуропридатності її складових компонентів. Відповідно до цього їх можна представити як множину базових ОФМ за ознакою нереконфігуропридатності. Згідно класифікатора, множина формується наступним чином, представленим на рисунку 3.3.

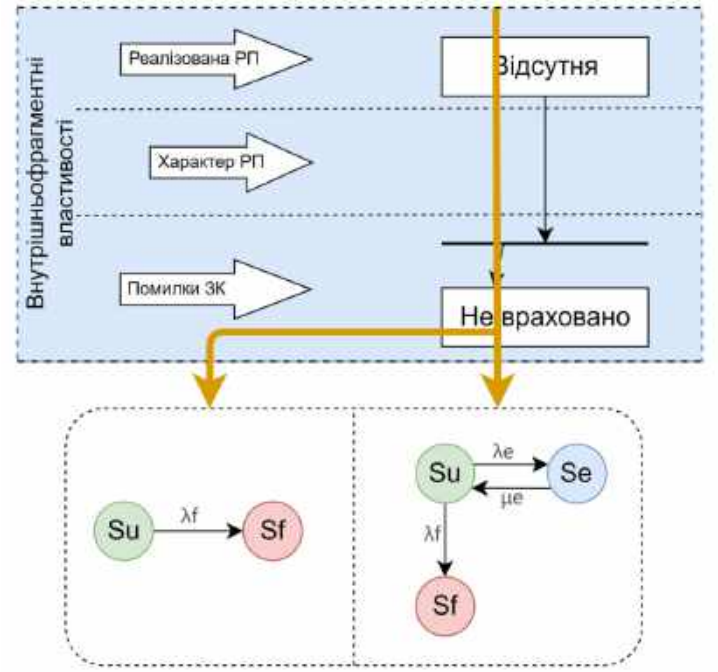

Рисунок 3.3 – Формування множин ОФМ без потенційної реконфігуропридатності

При наявних ознаках реконфігуропридатності, у додаток до базового набору станів моделі додаються стани, які характеризують тип спостережної деградації, а саме:

St (Tally-state) – стан ресурсної деградації, при якому ПП не зазнає вагомих втрат при відновленні,

Sd (Degradation-state) – стан функційної деградації, при якому ПП втрачає вагому частину своїх можливостей та/або якісних характеристик.

Переходи до станів деградації характеризується відповідними інтенсивностями:

λt – інтенсивність ресурсної деградації ПП,

λd – інтенсивність функційної деградації ПП.

Так, для ОФМ з можливістю реконфігуропридатності, графи станів виглядатимуть наступним чином, представленим на Рисунку 3.4:

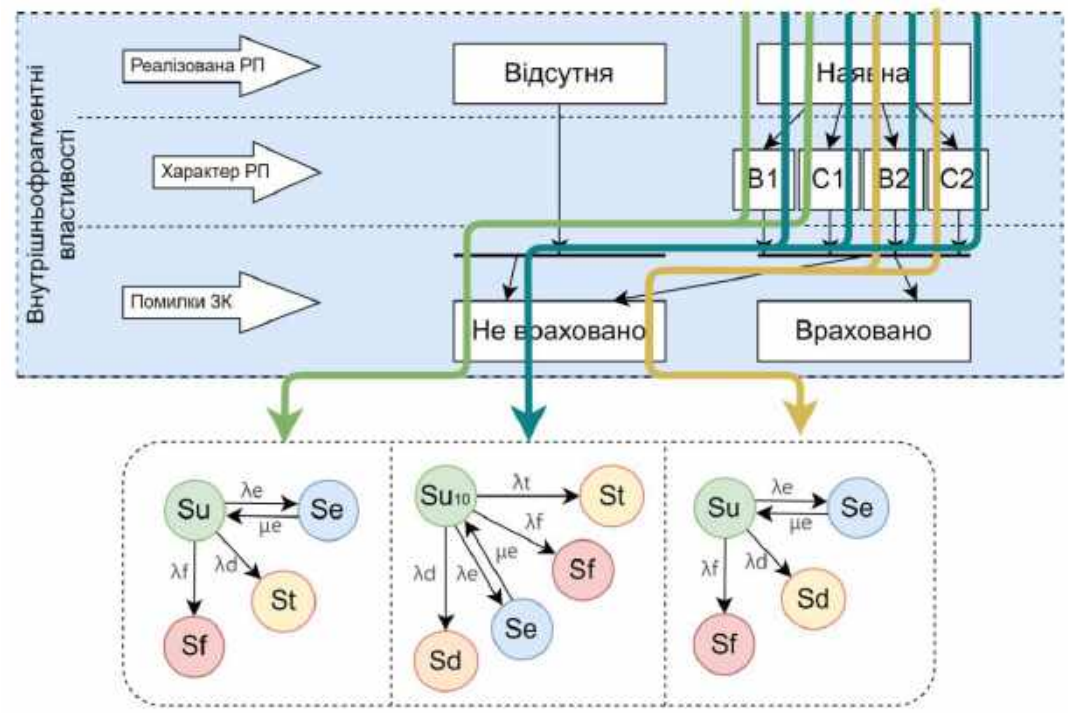

Рисунок 3.4 – Формування множин ОФМ з реалізованою реконфігуропридатністю

Слід зазначити, що варіант представлення фрагмента, який містить у своєму наборі станів одночасно кілька станів деградації також є можливим.

Для моделей, де засоби контролю розглядаються як вразлива, не абсолютно надійна частина системи, важливим є враховувати можливість її некоректної роботи. В цьому випадку модель містить

Sh (Hidden-state) – стан прихованої відмови, коли визначення характеру деградації/відмови засобами контролю не виконується.

У випадку прихованої відмови, перехід до станів деградації може здійснюватися через стан Sh, перехід до якого характеризується інтенсивністю

λh – інтенсивність виникнення прихованих відмов ПП, що можна описати рівнянням (1)

$$\lambda hd = (1- Dd) * \lambda d \quad (3.1)$$

$$\lambda ht = (1- Dt) * \lambda t, \quad (3.2)$$

$$\lambda hf = (1 - Df) * \lambda f, \quad (3.3)$$

де kh, kf, kt – це зважуючі коефіцієнти відповідного типу, D – це достовірність(ймовірність достовірного) контролю, а λ0 – це базова інтенсивність подій ПП. Аналогічно, для інших інтенсивностей будуть відповідними вирази (2 - 3).

Так, згідно сполук внутрішніх властивостей фрагменту для реконфігуропридатних ОФМ можна надати їх множину за ознакою врахування помилок засобів контролю, зображену на рисунку 3.5.

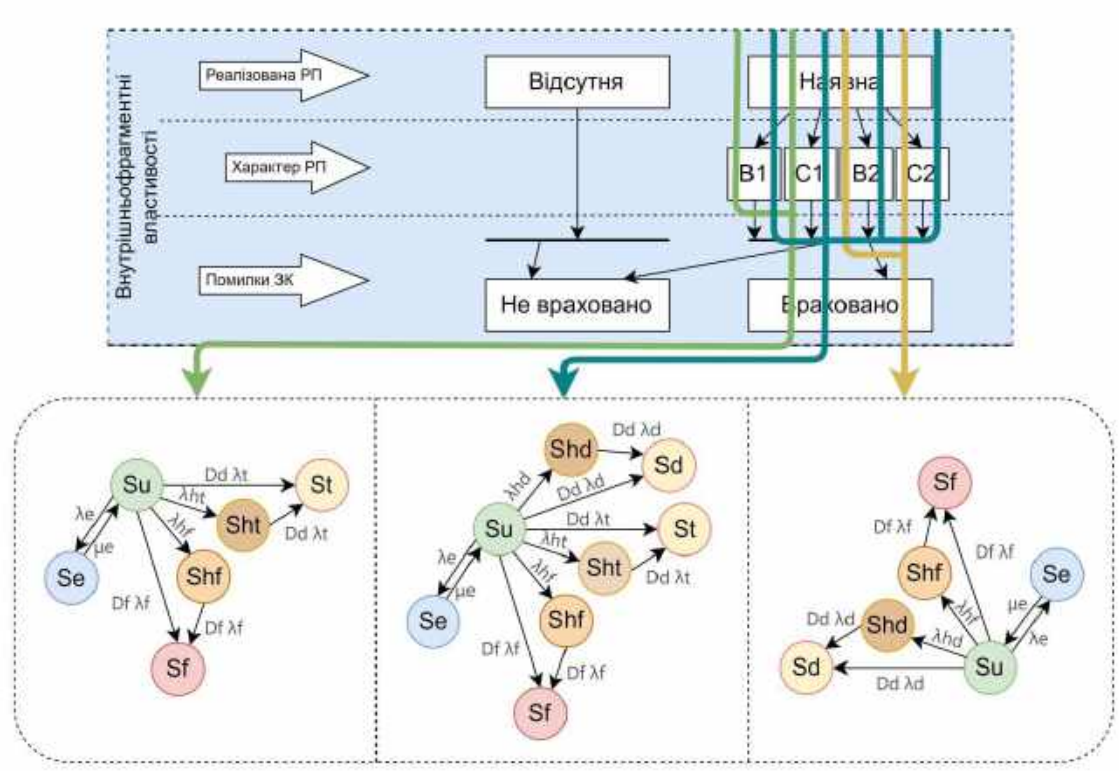

Рисунок 3.5 – Формування множин ОФМ з врахування помилок засобів контролю

Як результат, множини базових марковських ОФМ програмовних пристроїв без урахування комбінованих варіантів побудови, можна представити схемою, представленою на рисунку 3.6.

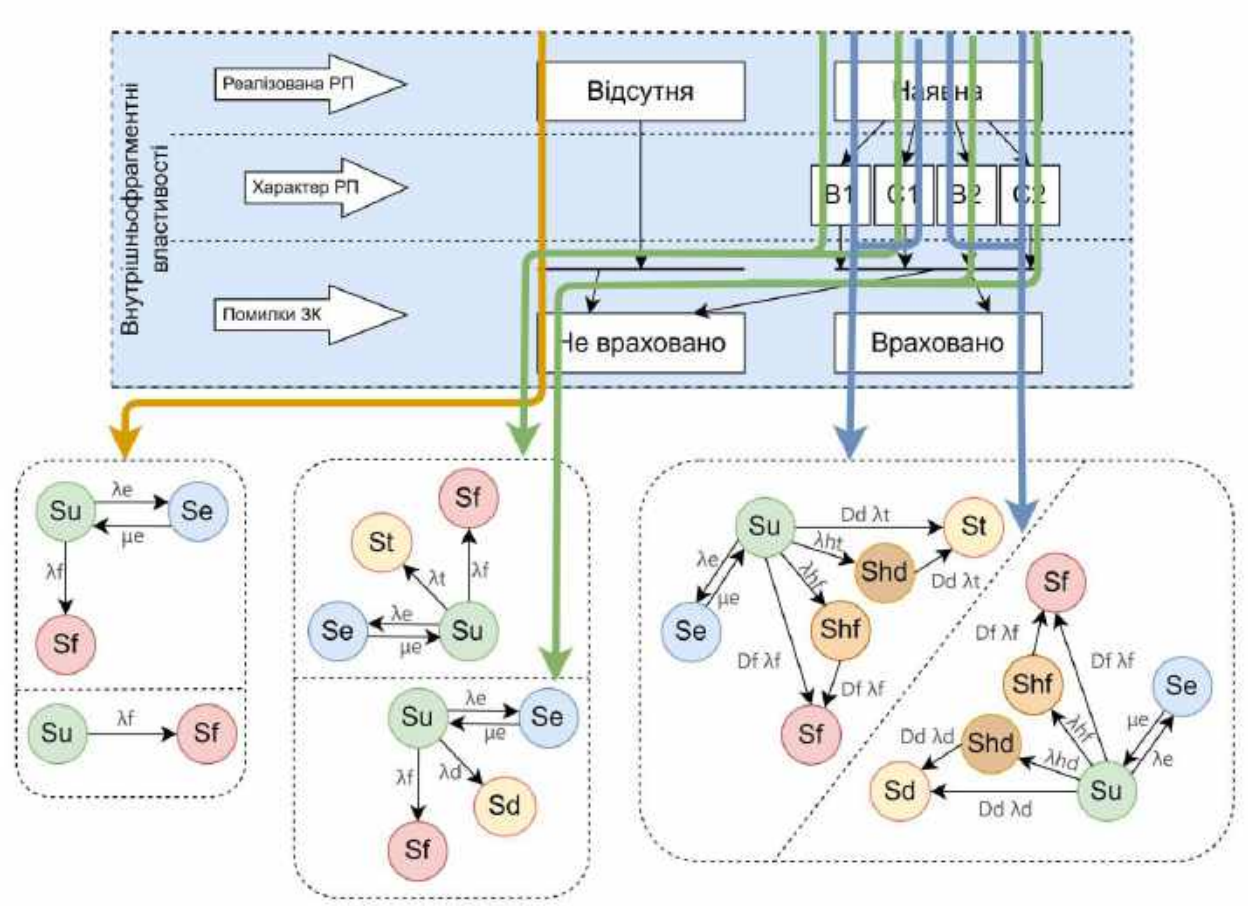

Рисунок 3.6 – Формування множин типових ОФМ

3.2.2 Множина мультифрагментних моделей

Варіанти побудови МФМ, залежно від врахування певних ознак, набувають додаткових перехідних станів в межах використовуваних ОФМ, та вимірів – ланцюгових повторень фрагментів з ОФМ зі зміною параметру, що й викликав деградацію. Так, для МФМ зі зміною одного параметра, таке повторення здійснюється в одному вимірі, а сама МФМ має вигляд лінійного графу фрагментів, представленого на рисунку 3.7.

Слід зазначити, що для лінійних графів, ознака взаємозалежності параметрів не є застосовною, що обумовлено відсутністю "парного" до змінюваного параметру.

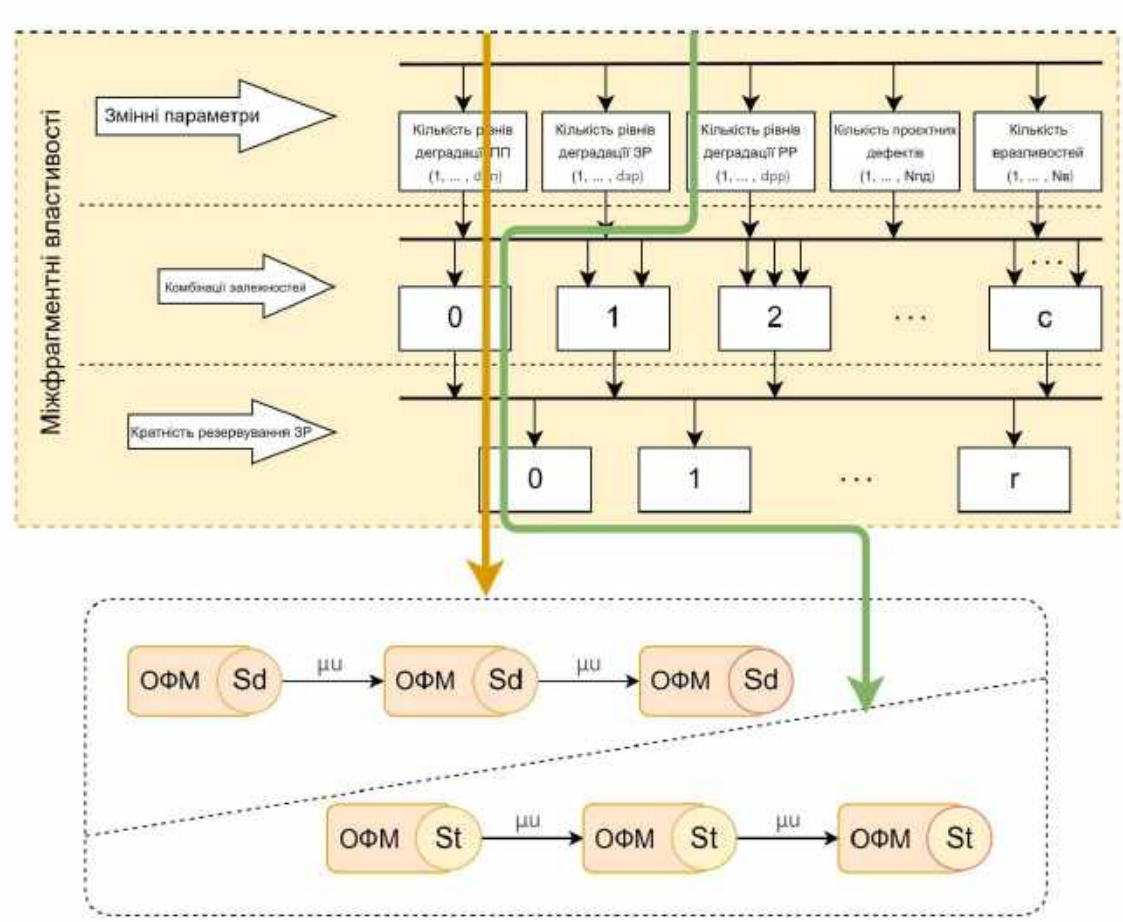

Рисунок 3.7 – Формування множин одновимірних МФМ(лінійні)

Додатковими внутрішньофрагментними станами для переходів між фрагментами є стани деградації (St,Sd,Sr…) з доступним працездатним станом (Su), що відповідає змінам у ПП. Переходи між фрагментами здійснюються через подію відновлення (реконфігурації), яка характеризується інтенсивністю відновлення після реконфігурації (μu).

При розгляді двовимірних МФМ з’являється можливість простежити вплив ознак взаємозалежності параметрів та резервування засобів реконфігурації, що передбачають наявність мінімум двох змінних. Відповідно до описаних зв’язків між фрагментами моделей, варіанти побудови двомірних МФМ з варіаціями застосування ознаки взаємозалежності параметрів можуть бути представлені рисунком 3.8.

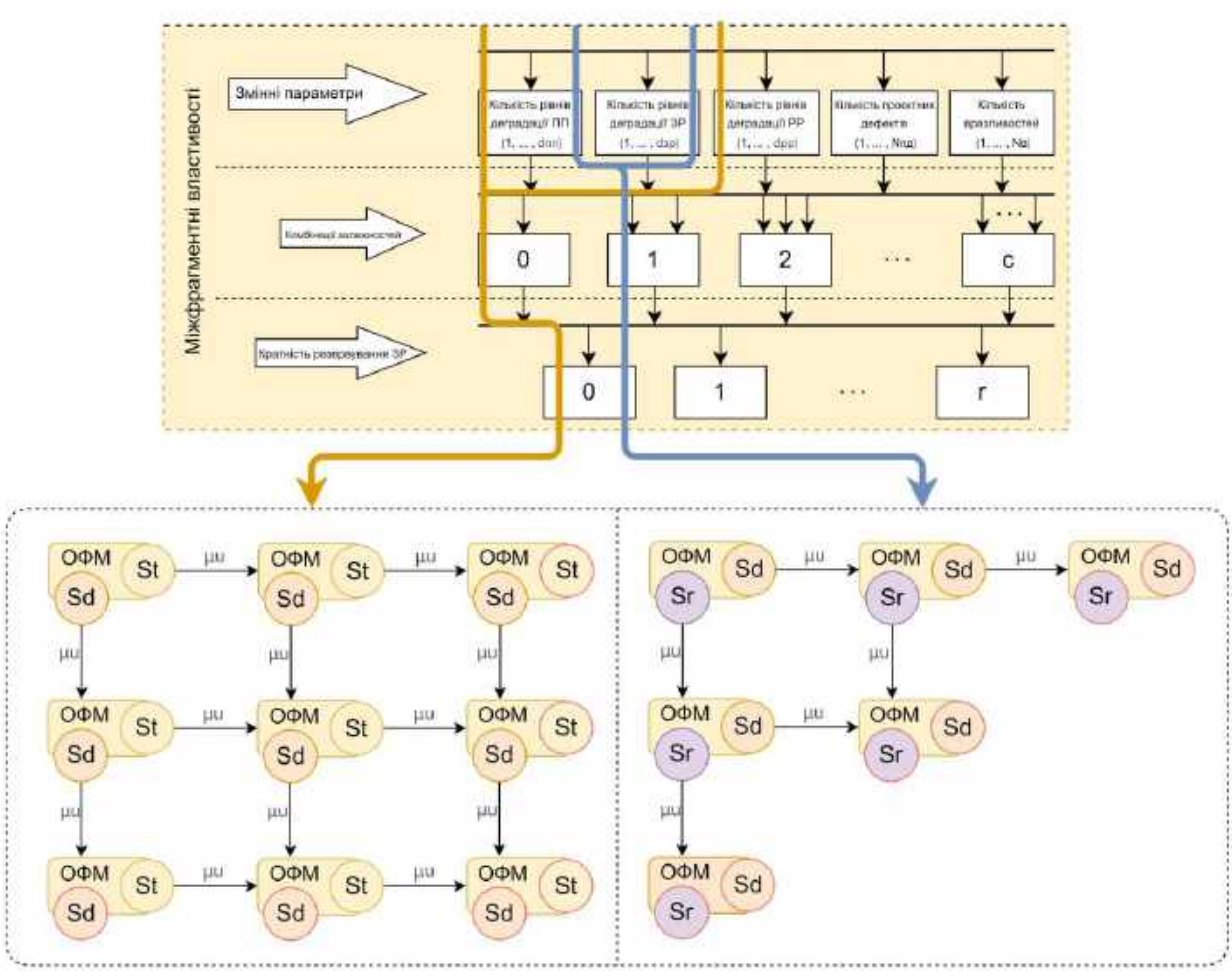

Рисунок 3.8 – Формування множин двовимірних МФМ (прямокутникова/трикутникова)

При відсутності взаємозалежності, деградація проходить незалежно від поточного стану кожного з показників. Таким чином, граф фрагментів набуває прямокутникового вигляду. Відповідно, при наявності взаємозалежності, деградація відбувається з оглядом на стан спорідненого параметру, що призводить до зміни кількості рівнів деградації з її протіканням. Граф фрагментів такого сценарію набуває трикутникового вигляду.

Для деяких параметрів, які впливають на деградацію, може бути врахована можливість резервування. Так, резервуючи зовнішні частини системи (засоби контролю, керування реконфігурацією, резервні ресурси), які не входять до складу самого ПП), можна уповільнити деградацію відповідного засобу. Прояв впливу резервування засобів реконфігурації у двовимірних МФМ виражається у дублюванні певних рівнів деградації.

Залежно від типу резервування засобу реконфігурації, можливі варіанти дублювання без підтримки циклічного перемикання засобів реконфігурації, коли моделлю не передбачається проведення реконфігурації дублікату ЗР, так і варіанти, де дублікат також може реконфігуруватись.

У першому випадку дублювання засобу реконфігурації, реакція на часткову або повну його відмову відбувається за обраним методом резервування. При виході з ладу одного засобу, у систему вводиться резерв з впровадженим контролем його деградації. Відповідно такого варіанту мультифрагментної моделі буде дублюватись тільки перший рівень деградації програмовного пристрою, лишаючі параметри програмовного пристрою незмінними до початку деградації вже задіяного резерву засобів реконфігурації.

У випадку, коли для засобу забезпечення реконфігурації передбачено контроль його деградації в обох дублікатах, передбачається циклічна оцінка ушкоджень кожного з резервів, та введення найбільш працездатного варіанту у роботу. Таким чином, кожний рівень деградації самого програмовного пристрою буде дубльованим відповідно кількості задіяних резервних копій засобів реконфігурації. Передбачається, що використання оцінювання та порівняння рівнів деградації резервів ЗР між собою може бути корисним при прийнятті рішення при їх задіюванні.

Описаний перелік варіантів застосування резервування засобів реконфігурації не є вичерпним, оскільки не враховує впливу підходів до резервування. Також варто зазначити, що сама задача керування резервом потребує оцінки з точки зору впливу на надійнісні характеристики ПП при її реалізації. Однак, покриття класичних варіантів резервування буде достатнім для формування множини базових двовимірних МФМ, варіанти побудови котрих представлено рисунком 3.9.

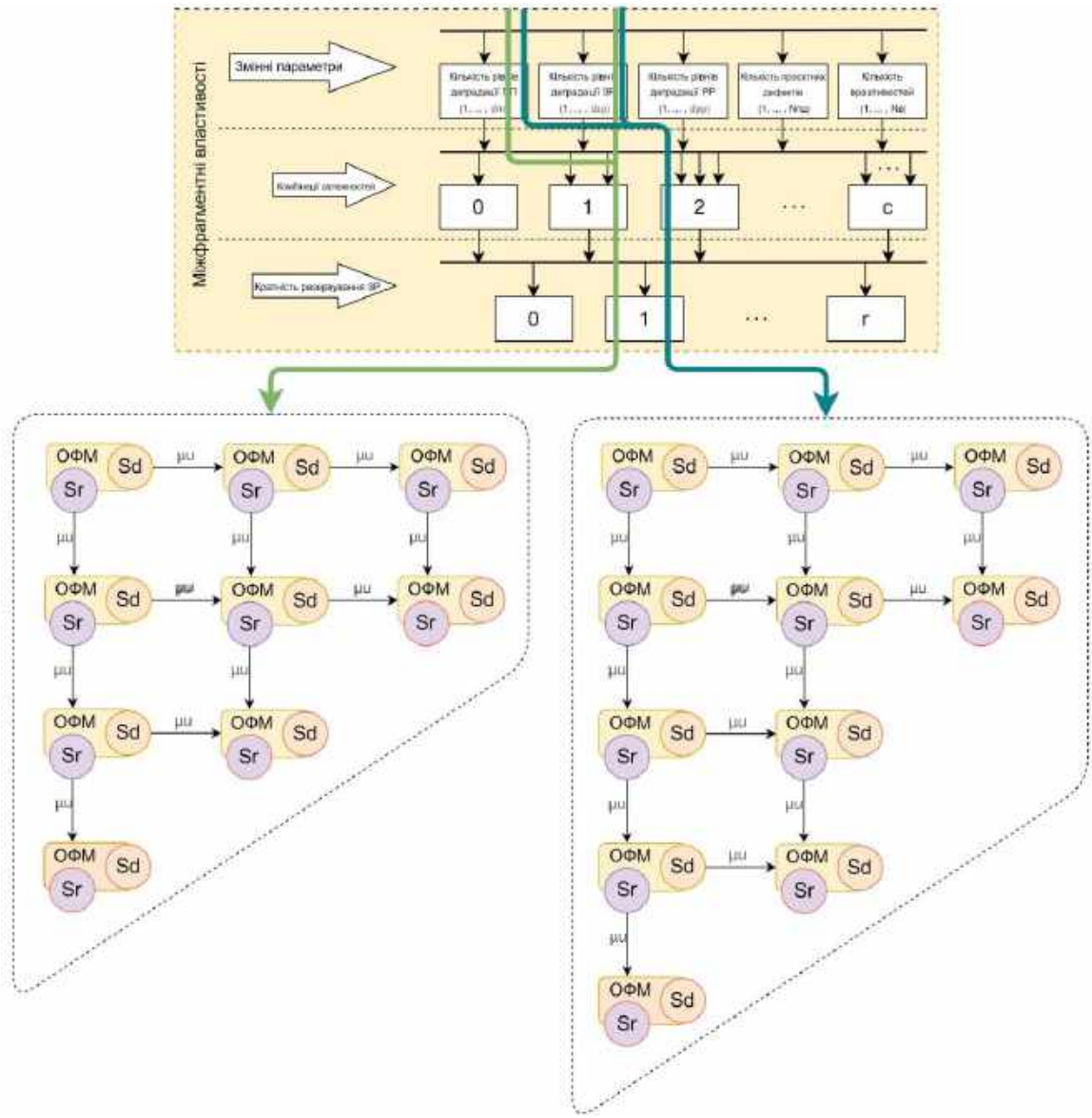

Рисунок 3.9 – Формування множин двовимірних МФМ (багатокутна/мішана)

При розгляді багатовимірних МФМ слід зазначити, що вплив розглянутих у двовимірних МФМ ознак зберігається, що передбачає велику варіативність їх побудови. У якості прикладу побудови багатовимірної МФМ, розглядається поєднання розглянутих раніше ознак у вигляді тривимірної "призматичної" МФМ, наведеної на рисунку 3.10.

Варто зазначити, що в зображеному прикладі взаємозалежними є параметри деградації програмовного пристрою та деградації засобів реконфігурації, що не впливає на незалежний параметр деградації ресурсу реконфігурації.

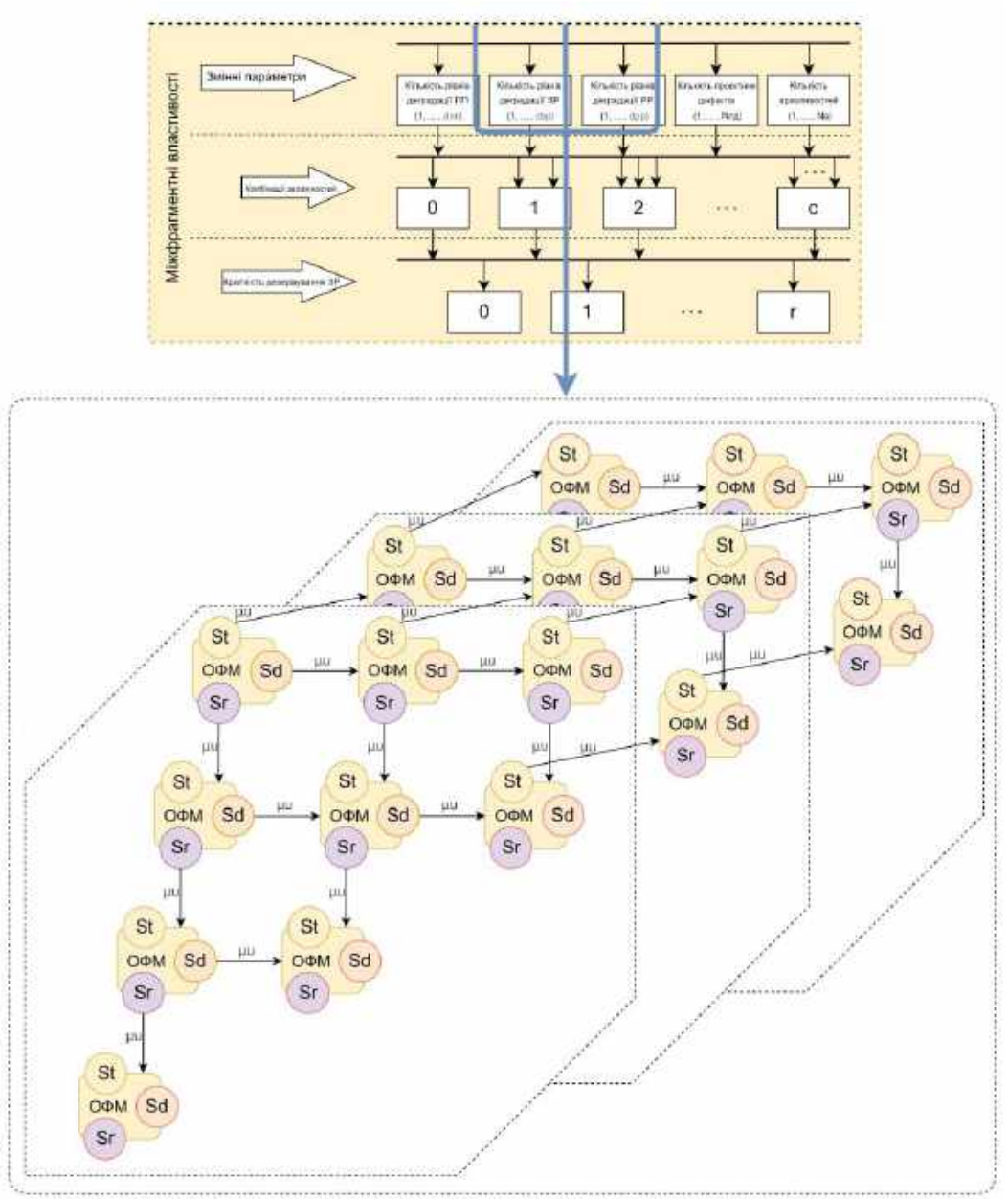

Рисунок 3.10 – Формування тривимірної МФМ (призматична)

Як результат, множини базових марковських МФМ програмовних пристроїв без урахування комбінованих варіантів побудови, можна представити схемою на рисунку 3.11.

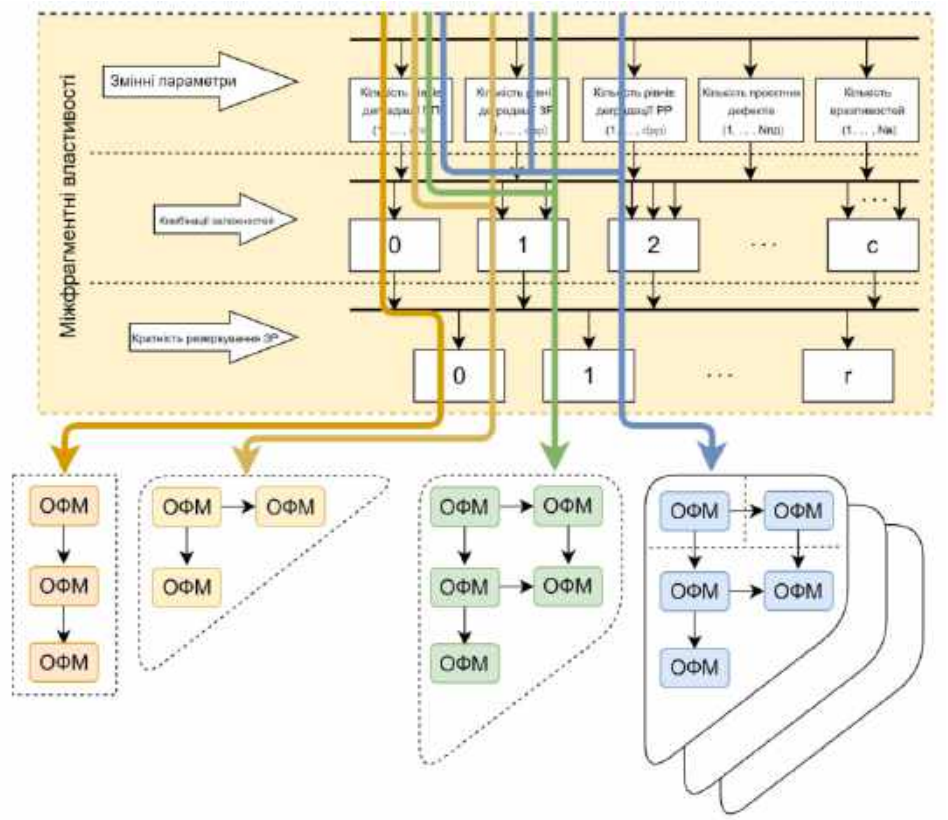

Рисунок 3.11 – Формування множини типових МФМ

Таким чином, для ефективного моделювання контролю деградації програмовних пристроїв з багаторівневою деградацією необхідно враховувати низку ознак, які визначають поведінку системи при відмовах. Представлений перелік ознак не є вичерпним, але достатнім для охоплення розповсюджених випадків відмови ПП. Для подальшого використання, приведений перелік буде використовуватися у складі класифікатора, що описує ключові сутності кожної з ознак, та зв’язки між ними.

3.3 Дослідження однофрагментних моделей готовності програмовних пристроїв

3.3.1 Обґрунтування припущень і параметрів при розробленні однофрагментних моделей готовності програмовних пристроїв

Оскільки визначення моделі передбачає спрощене представлення системи або об'єкта реального світу, доречним буде означити ряд припущень щодо її складових.

Для засобів контролю(ЗК) застосовуватимуться наступні припущення:

- система контролю є абсолютно надійною;
- реакція на зміни у пристрою, що супроводжується, є миттєвою;
- на час після виникнення несправності та час проведення реконфігурації ПП перебуває в неробочому стані (idle), але спостереження продовжуються.

Причиною накладання таких обмежень є те, що для ПП система діагностики переважно має форму окремого пристрою. У такому вигляді вона може бути застосована до різних моделей, тому її вплив при обчисленні надійнісних характеристик моделей буде пропорційно однаковим. Таким чином, впливом ЗК на розглянуті моделі під час симуляції, можна знехтувати для кращої наочності отриманих результатів.

Відповідно, до самої моделі застосовуватимуться такі припущення, як:

- виключення виникнення ланцюгових реакцій при відмовах.
- множинні відмови, що виникають, розподілені в часі подібно до одиночних(ординарних) відмов.

І ці обмеження викликані характеристиками простого потоку відмов - ординарністю і відсутністю післядії, що додатково забезпечується алгоритмом переривання роботи при виникненні відмови.

3.3.2 Розроблення моделі ОФМ1 з деградацією ресурсів програмовних пристроїв

Для моделювання пропонується розглянути систему, побудовану на основі 8-контактного мікроконтролера ATtiny13A [75]. При нормальному режимі роботи мікроконтролера відмова самого чіпу, або хоча б одного контакту порушує його нормальний режим роботи, що призводить до відмови ПП. Така поведінка може бути представлена у вигляді ОФМ на рисунку 3.12.

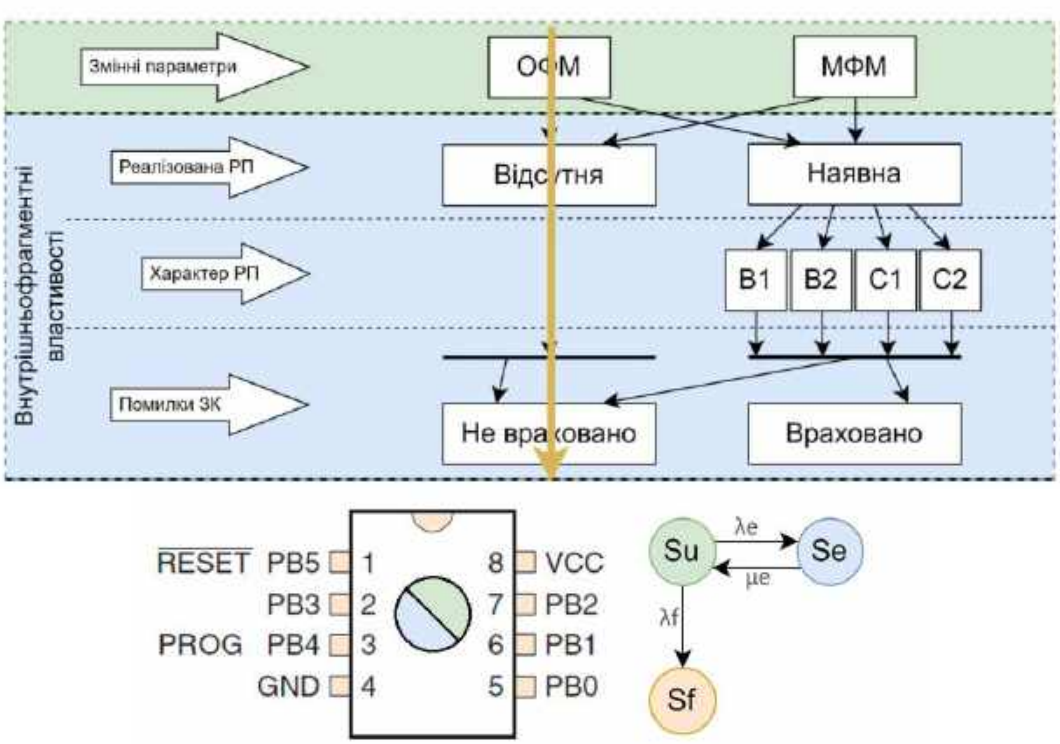

Рисунок 3.12 – ОФМ для мікроконтролера ATtiny13A без реалізації РП

Визначимо, що відмови критичних елементів чипу, та однотипних контактів порту PB можуть бути спричинені дефектами типів B2 та C2. Це пов’язано з тим, що порт B оснащується двонаправленими контактами з підтягувальними резисторами. Завдяки такій структурі контакти здатні виконувати спільні маніпуляції і діяти як резерв один для одного [74], що є ознаками потенційної РП. Завдяки програмній модифікації завантажувача мікроконтролера з’являється інструмент для віддаленого коригування коду, що надає змогу перерозподіляти виконувані завдання між працездатними контактами, тобто реалізовувати потенційну РП. Згідно цьому, можливі варіанти ОФМ, що ілюструються відповідними траєкторіями на рисунку 3.13.

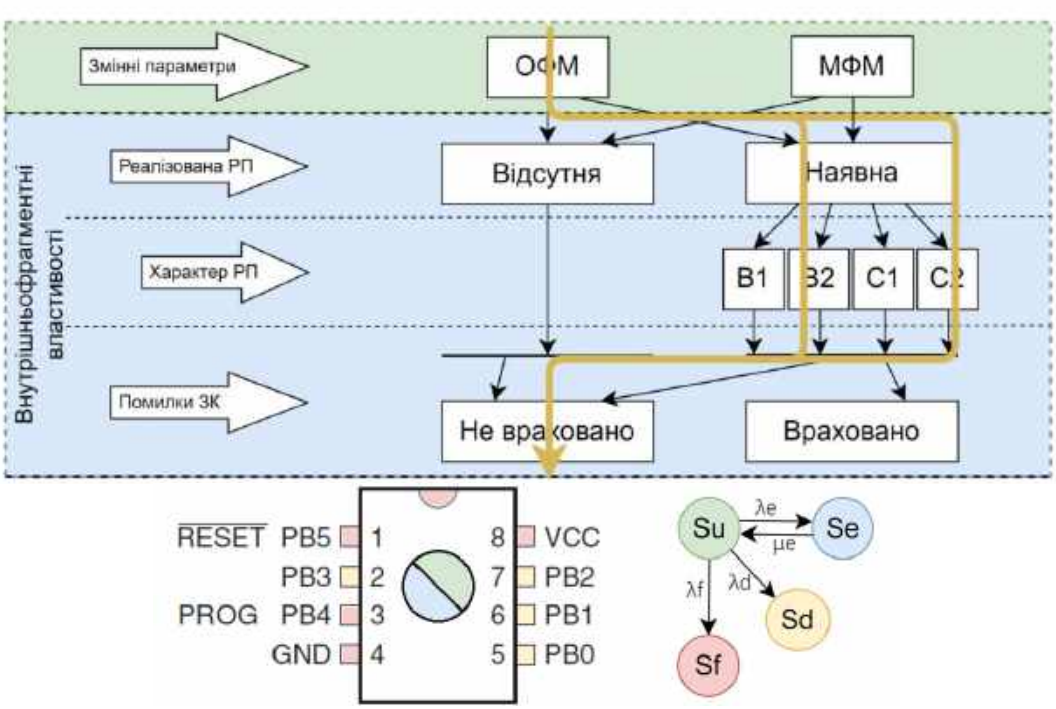

Рисунок 3.13 – ОФМ для мікроконтролера ATtiny13A з реалізацією РП

Управління навантаженням на обрані контакти обмежено різницею між кількістю наявних контактів, та кількістю мінімально необхідних для обміну даними. Згідно з існуючими форматами обміну сигналами, для виконання обміну потрібно мінімум два контакти, що при наявних чотирьох надає можливість двократного відновлення при їхніх одиничних відмовах.

3.3.3 Дослідження моделі ОФМ1 з деградацією ресурсів програмовних пристроїв

Для розгляду представленої моделі буде використовуватись засіб для симуляції на основі середовища Matlab. Процес побудови ОФМ для ПП складається з кількох кроків. Спочатку ініціалізуються інтенсивності переходів між станами пристрою, такі як інтенсивність обриву контакту (λ_{pin}), інтенсивність деградації (λd_{x0}), інтенсивність відмов (λf) та інтенсивність збою (λe)[74]. Крім того, задається часовий інтервал моделювання, а функція odeset використовується для налаштування чисельного розв'язувача із заданими рівнями допусків. Детальна параметризація представлена таблицею 3.1.

Таблиця 3.1 – Інтенсивності переходів ОФМ

№	**Параметр**	**Значення** $*$ $\mathbf{10^{-5}}$, 1/год
1	Інтенсивність відмови діоду (λ_{diod})	0.02
2	Інтенсивність відмови резистора (λ_{res})	0.01
3	Інтенсивність відмови мікроконтролеру (λ_{tiny})	0.002
4	Інтенсивність відмови контакту (λ_{pin})	$\lambda_{res} + \lambda_{diod} = 0.03$
3	Інтенсивність першої деградації (λd_{x0})	$4 * \lambda_{pin} = 0.12$
4	Інтенсивність відмови критичного елемента (λf)	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
5	Інтенсивність збоїв (λe)	$8 * \lambda_{pin} + \lambda_{tiny} = 0.242$
6	Інтенсивність відновлення після збоїв (μe)	1/60 = 1666

Далі функція fM0() визначає матрицю переходів станів V та матрицю подій E, які описують переходи між різними станами пристрою (такими як робочий, помилка, відмова тощо). Функція для візуалізації спрямованого графа марковської моделі представляє стани та переходи ОФМ, зображені на рисунку 3.14.

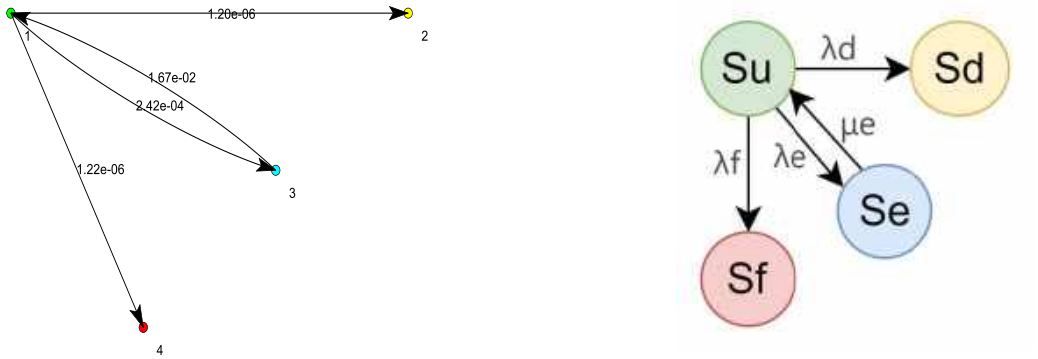

Рисунок 3.14 – Графи станів та переходів ОФМ у Matlab та draw.io

Система диференціальних рівнянь Колмогорова-Чепмена, відповідних до наведеного графу та таблиці параметрів, представляється наступним чином

$$\begin{cases} \frac{dP_u}{dt} = -\left(\lambda_u + \lambda_d + \lambda_f\right)P_u + \mu_e P_e; \\ \frac{dP_d}{dt} = \lambda_d P_u; \ \frac{dP_f}{dt} = \lambda_f P_u; \\ \frac{dP_e}{dt} = -\mu_e P_e + \lambda_e P_u; \\ P_u + P_d + P_e + P_f = 1. \end{cases} \qquad P_u(0) = 1, P_d(0) = P_e(0) = P_f(0) = 0 \qquad (3.4)$$

Наступним кроком є побудова матриці A, яка представляє швидкості переходу між різними станами системи. Матриця A обчислюється за допомогою функції, де створюється початковий вектор стану P0, де ПП знаходиться в повністю робочому початковому стані (P0 = [1 0 0 0]). Після визначення початкових умов системи та матриці переходів, призводиться обчислення вірогідності перебування ПП у кожному зі станів за допомогою ODE-розв'язувача. Функція готовності, у даному випадку, відповідає ймовірності перебування системи в робочому стані Su. Результати моделювання візуалізуються шляхом побудови графіків розподілу ймовірностей станів системи в часі. На рисунках 3.15 – 3.17 зображено графік з кількома кривими, кожна з яких представляє ймовірність перебування системи в одному зі станів (працездатний, збій, відмова або деградований)

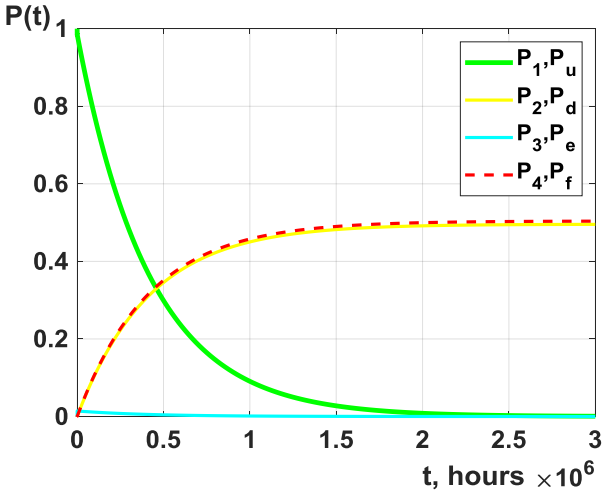

Рисунок 3.15 – Графік розподілу ймовірностей перебування у станах

На рисунку 3.15 представлена функція готовності, яка зменшується від 1 до 0 протягом інтервалу 3-106 годин (342,5 років). Цей час зменшується в умовах

агресивного середовища. Ймовірності перебування в станах Sd і Sf приблизно рівні, тоді як ймовірність перебування в стані Se є найнижчою.

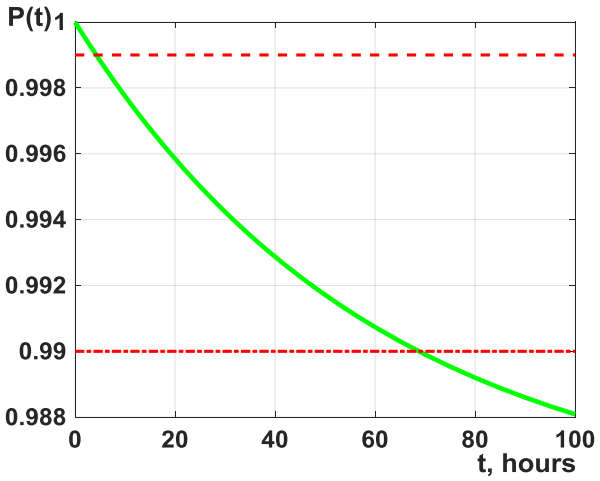

Рисунок 3.16 – Графік перетину функцією готовності показника у 0.99

На рисунку 3.16 показано, що функція готовності падає до 0,999 протягом перших 6 годин роботи і досягає 0,99 протягом перших 69 годин.

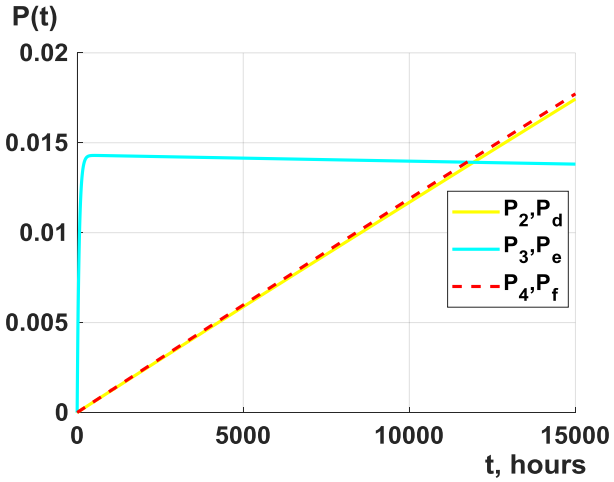

Рисунок 3.17 – Точка перетину функцій відмови/деградації та збою

Результати на рисунку 3.17 ілюструють, що після 12 000 годин роботи (1,4 роки) ймовірність перебування в станах Sd і Sf починає перевищувати ймовірність перебування системи в стані Se.

3.4 Дослідження мультифрагментних моделей готовності програмовних пристроїв

3.4.1 Обґрунтування припущень і параметрів при розробленні мультифрагментних моделей готовності програмовних пристроїв з багаторівневою деградацією

Схожим до ОФМ чином, для МФМ передбачається визначення ряду припущень щодо ПП, ЗК та ЗР.

Для засобів контролю(ЗК) застосовуватимуться наступні припущення:

- система контролю є абсолютно надійною, окрім випадків, де враховуються її помилки у роботі;

- суттєвими вважаються випадки, коли ЗК не фіксує виникнення відмов, що призводить до виникнення станів прихованої відмови;

- прихована відмова може бути виявленою при подальших процедурах діагностики.

Надані припущення спираються на той факт, що до розгляду будуть надані МФМ з урахуванням помилок ЗК. При цьому стани хибної на некоректно ідентифікованої відмов розглядатись не будуть, оскільки через накладені обмеження вони проявлятимуть себе як збої, що усуваються без втручання засобів реконфігурації. Через те, що ЗК працюють окремо від супроводжуваного ПП, його контроль буде призводитись безперервно, за рахунок чого подальші процедури діагностики, що виникатимуть при прояві не купованого дефекту, будуть виявляти приховані відмови.

Для засобів реконфігурації(ЗР) застосовуватимуться такі припущення:

- процедури реконфігурації можуть бути застосовані тільки після виявлення та ідентифікації несправності;

- реконфігурація ПП передбачає наявність втрат (ресурсних або функціональних), які є незворотніми.

Ці обмеження мотивовані специфікою роботи засобу, що забезпечує реконфігурацію ПП. Вибір виконуваної процедури залежить від типу дефекта, що

проявляється та ресурсу самого ЗР. Крім того, використаний ресурс відновлення не може бути оновлений або поповнений без зовнішнього втручання.

До самої моделі застосовуватимуться наступні припущення:

- виключення виникнення ланцюгових реакцій при відмовах;
- деградація параметрів моделі не залежить від фактора часу;
- виключення варіантів виникнення кластерних відмов.

Подібно до ОФМ, поточні обмеження обумовлені наближенням моделі до характеристик простого потоку відмов при виконанні переривання роботи в наслідок виникнення відмови.

3.4.2 Розроблення моделі МФМ1 з деградацією ресурсів програмовних пристроїв

Для випадку реалізації засобів реконфігурації (ЗР) у форматі зовнішнього пристрою або системи пристроїв, їх впливом на загальну схему надійності можна знехтувати при побудові МФМ ПП з БРД. Таким чином, простий варіант БРД описуватиметься МФМ1 з лінійною структурою з'єднання фрагментів, представленим на рисунку 6.

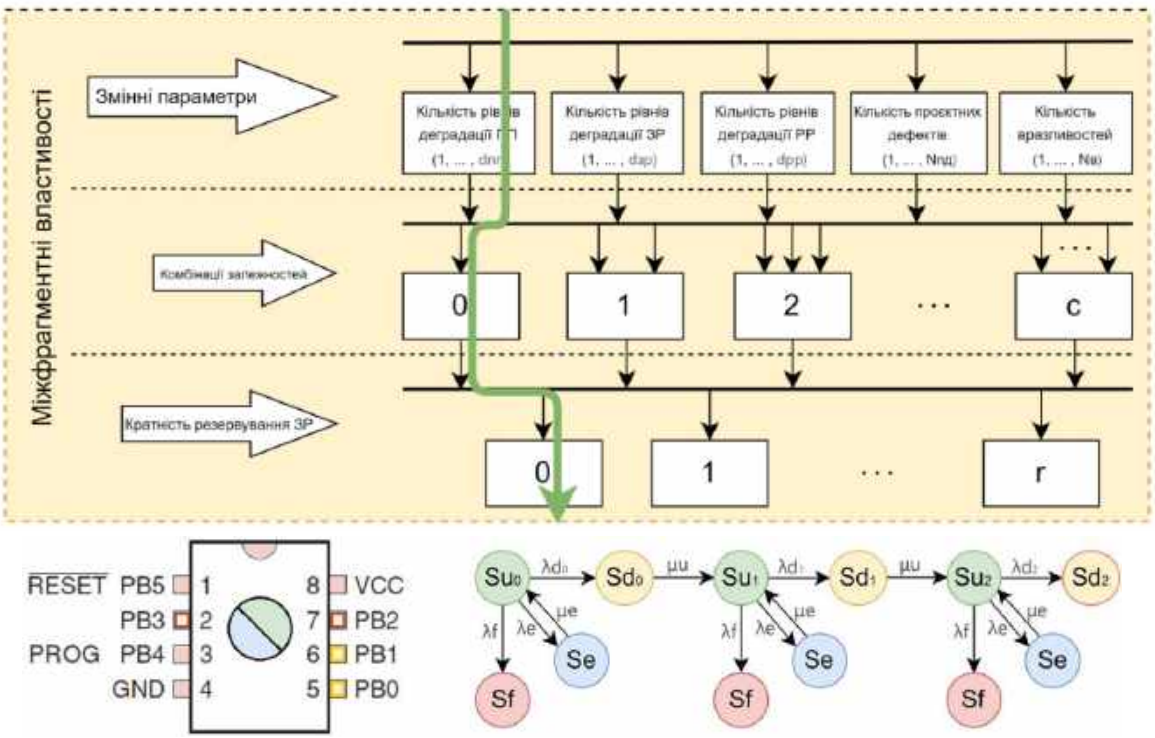

Рисунок 3.17 – Побудова лінійної МФМ1 ПП з БРД

Описана вище поведінка системи відповідає випадку, коли засоби керування реконфігурацією вважаються абсолютно надійними та коректними, а процедури перерозподілу навантаження на контакти є доступними та виконуваними. Цикли перезапису вмісту мікросхеми не обмежені, а самі процедури не пов'язані з витрачанням ресурсів мікросхеми або самої системи керування реконфігурацією.

Відповідно до представленої лінійної МФМ1, існуючі стани в межах фрагменту набувають наступних змін:

- працездатні стани (Su_i) кожного з фрагментів нумеруються відповідно рівню виконаної деградації;
- перехід між фрагментами здійснюється за рахунок відновлення після реконфігурації, яка характеризується інтенсивністю μu.
- стани деградації (Sd_i) та відповідні події деградації з відповідною інтенсивністю (λd_i) мають індекси працездатних станів, для яких вони є похідними;

3.4.3 Дослідження моделі МФМ1 з деградацією ресурсів програмовних пристроїв

Для розгляду МФМ1 за допомогою Matlab-функції fM01() буде створюватись серія взаємопов'язаних компонент на основі вхідних параметрів. Детальна її параметризація представлена таблицею 3.2. Окрім звичної ініціалізації інтенсивностей переходів між станами пристрою, таких як інтенсивність обриву контакту (λ_{pin}), інтенсивність відмов (λf), інтенсивність збою (λe) буде впроваджена модифікована інтенсивність деградації (λd_i). Її значення буде залежати від рівня деградації, який позначається індексом ***i***, що у свою чергу позначатиме кількість пройдених процедур реконфігурації. Кількість рівнів визначає кількість фрагментів по горизонталі, і в нашому випадку сягає розмірності (size(λd,2)).

Таблиця 3.2 – Інтенсивності переходів МФМ1

№	Параметр	Значення $*$ 10^{-5}, 1/год
1	Інтенсивність відмови діоду (λ_{diod})	0.02
2	Інтенсивність відмови резистора (λ_{res})	0.01
3	Інтенсивність відмови мікроконтролеру (λ_{tiny})	0.002
4	Інтенсивність відмови контакту (λ_{pin})	$\lambda_{res} + \lambda_{diod} = 0.03$
5	Інтенсивність першої деградації (λd_0)	$4 * \lambda_{pin} = 0.12$
6	Інтенсивність другої деградації (λd_1)	$3 * \lambda_{pin} = 0.09$
7	Інтенсивність третьої деградації (λd_2)	$2 * \lambda_{pin} = 0.06$
8	Інтенсивність відмови критичного елемента (λf)	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
9	Інтенсивність збоїв (λe)	$8 * \lambda_{pin} + \lambda_{tiny} = 0.242$
10	Інтенсивність відновлення після збоїв (μe)	1/60 = 1666

Функція fM01() будує матрицю переходів між фрагментами шляхом їх ітерації через λd. Інтенсивності деградації у такому застосуванні виступають у якості інтенсивності відмов фрагментів, кожен з яких представляється групою станів та переходів між ними. Функція для візуалізації спрямованого графа марковської моделі представляє стани та переходи МФМ1, зображені на рисунку 3.18.

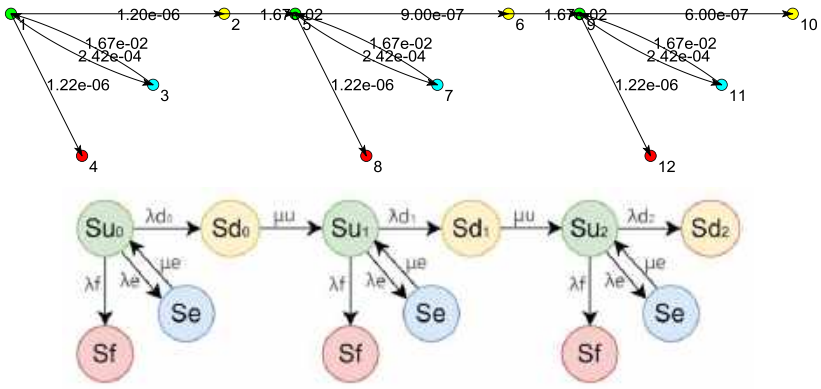

Рисунок 3.18 – Графи станів та переходів лінійної МФМ у Matlab та draw.io

Отримана модель використовується для моделювання поведінки ПП за плином часу при використанні вирішувача ode15s. Результати моделювання показано на рисунках 3.19–3.21.

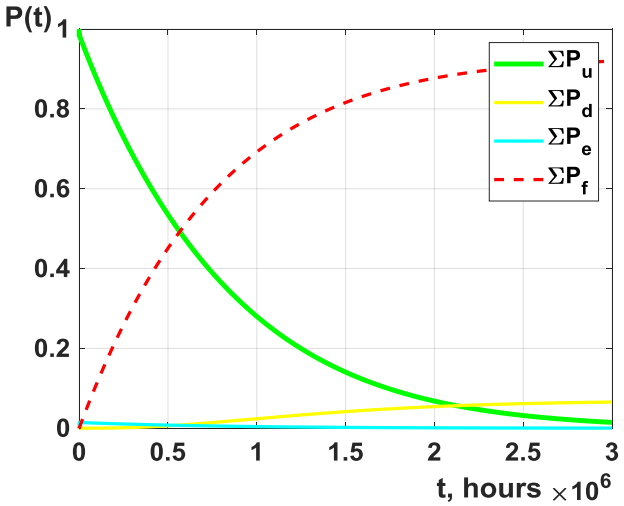

Рисунок 3.19 – Графік розподілу ймовірностей перебування у станах

На діаграмі рисунку 3.19 функція доступності спадає від 1 до 0 на інтервалі 3-106 годин. На відміну від результатів ОФМ, сумарна ймовірність перебування в станах Sf значно перевищує сумарну ймовірність перебування в станах Sd. Найменшою є сумарна ймовірність перебування у станах Se.

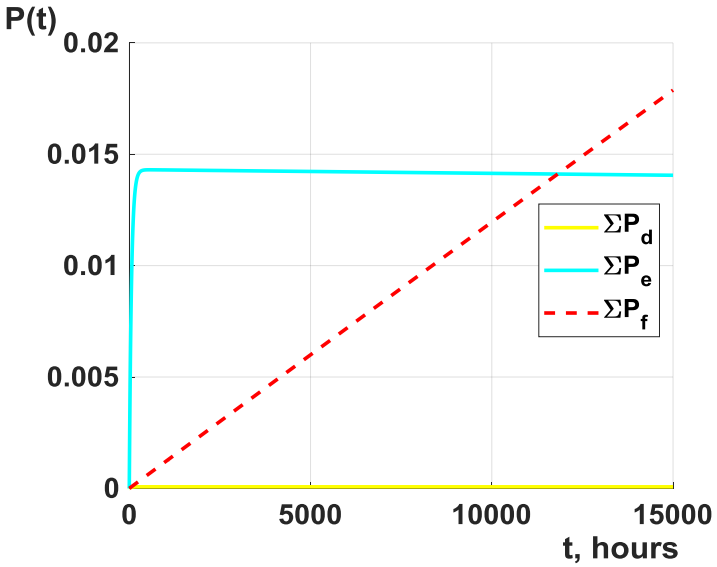

Рисунок 3.20 – Точка перетину функцій відмови\деградації та збою

Як видно з рисунку 3.20, сумарна ймовірність перебування в Sf-станах починає перевищувати ймовірність перебування в Se-станах після 12 000 годин (1,4 року) експлуатації.

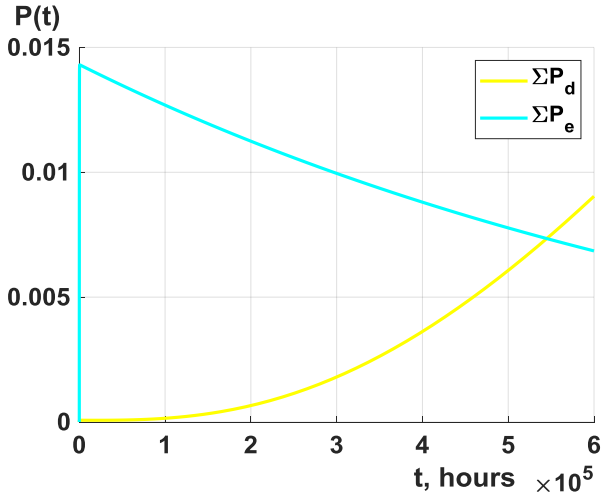

Рисунок 3.21 – Точка перетину функцій деградації та збою

Рисунок 3.21 демонструє сумарну ймовірність перебування в станах Sd, яка починає перевищувати сумарну ймовірність перебування в станах Se після 540 000 годин (61,6 років) експлуатації.

3.4.4 Розроблення моделі МФМ2 з деградацією ресурсів програмовних пристроїв та засобів реконфігурації

У випадку, коли реалізації засобів реконфігурації(ЗР) вимагає врахування їх впливу на модель, кількість змінних параметрів збільшується, що, у свою чергу, додає ще один вимір МФМ. Оскільки якість ЗР впливає на кількість можливих процедур відновлення, то її втрата (деградація внаслідок власних відмов) буде призводити до зменшення кількості можливих рівнів (довжини ланцюга) деградації ПП. Таким чином, варіант БРД за двома параметрами деградації ПП та ЗР описуватиметься МФМ з «трикутниковим» розташуванням фрагментів, як представлено на рисунку 3.22.

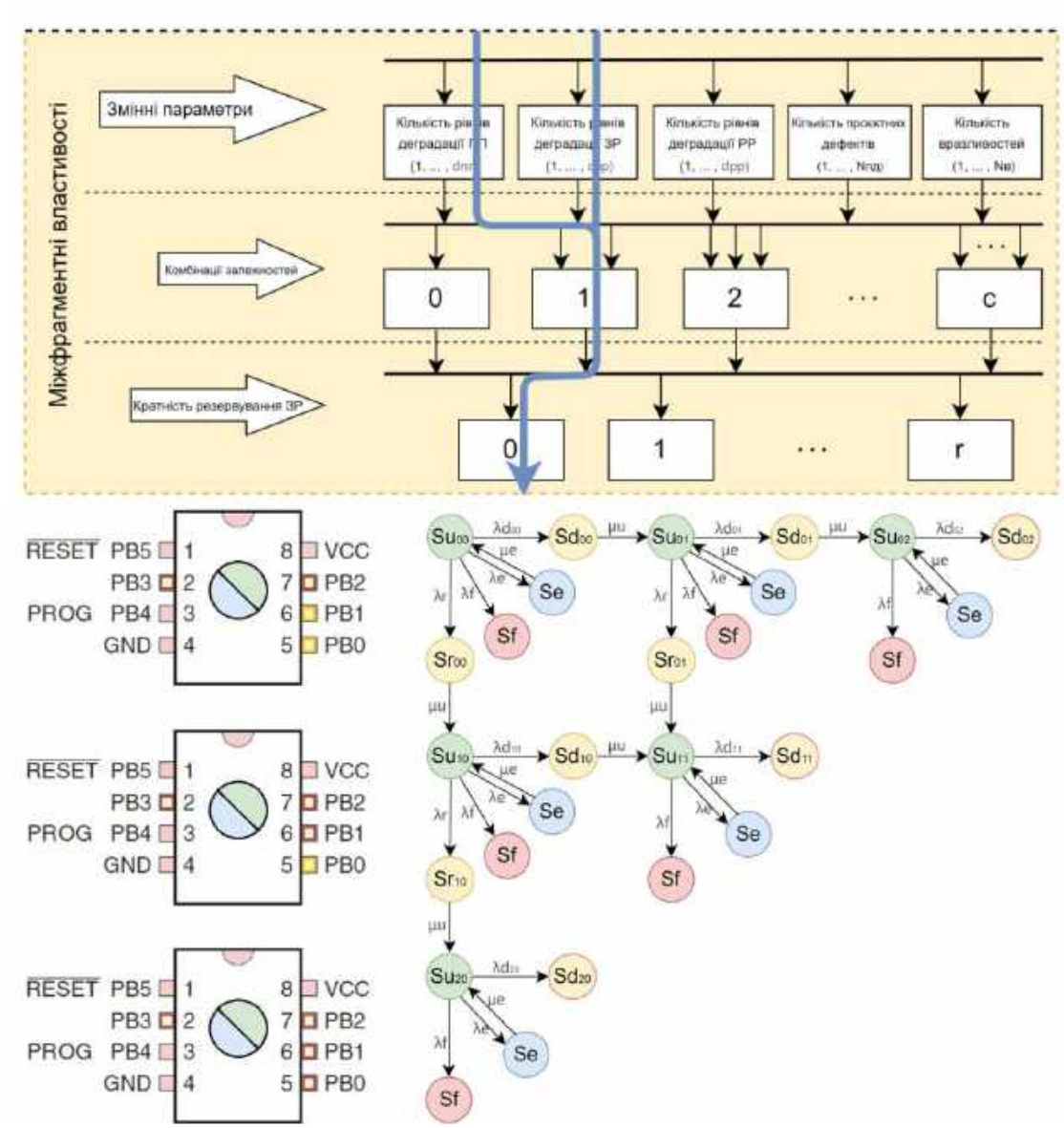

Рисунок 3.22 – Побудова трикутникової МФМ ПП з БРД

У даному випадку ЗР може деградувати в межах своїх можливостей керування і/або повністю відмовити, що призведе до поведінки ПП, яка описуватиметься одним фрагментом, аналогічно ОФМ без реалізації РП.

Відповідно до представленої трикутникової МФМ, існуючі стани в межах фрагменту набувають наступних змін:

- працездатні стани (Su_{ij}) кожного з фрагментів нумеруються додатковим другим індексом відповідно до рівня деградації ЗР;
- стани деградації (Sd_{ij}) та події деградації з відповідною інтенсивністю (λd_{ij}) мають індекси працездатних станів, для яких вони є похідними;

- для переходу між фрагментами, відповідно до деградації ЗР, в межах фрагменту позначається стан (Sr_{ij}) та подія деградації ЗР з відповідною інтенсивністю λr_{ij}. Відновлення після настання такого стану здійснюється також внаслідок події, яка характеризується інтенсивністю відновлення після реконфігурації μu.

3.4.5 Дослідження моделі МФМ2 з деградацією ресурсів програмовних пристроїв та засобів реконфігурації

За аналогією до попереднього випадку, функція fM02() будує матрицю горизонтальних переходів між фрагментами МФМ2 через додатково модифіковану інтенсивність деградації (λd_{ij}). В цьому випадку її значення залежить не лише від рівня деградації ПП з індексом *j*, а й від рівня деградації ЗР, з відповідним індексом ***i***.

Кількість вертикальних фрагментів відповідає першій розмірності вхідної змінної λr, що пов'язано з ресурсом деградації ЗР. При моделюванні МФМ2 для фрагментів розроблено матрицю довільних розмірів N на M. Таким чином досягається «прямокутна» структура моделі, яка у подальшому маскується матрицею нульових переходів для набуття «трикутної» форми. Відповідна зміна була введена для більшої гнучкості моделі при симуляція за рахунок різних розмірів N та M. Детальна параметризація представлена таблицею 3.3.

Слід зазначити, оскільки у моделі присутні «неактивні» фрагменти, для яких ймовірність переходу дорівнює 0, розрахунки призводяться без їх урахування. Відповідно, завдяки нульовій ймовірності перебування у станах ізольованих фрагментів, досягається ідентичність з розробленою МФМ2.

У наведеному прикладі матриця розподілу параметру λd має розмірність 3 на 3, що відповідає 9 загальним фрагментам, 6 з яких можуть бути досяжними через ненульові переходи. Аналогічно, матриця розподілу параметру λr має розмірність 2 на 2, що свідчить про 4 загальних фрагменти, 3 з котрих мають ненульові переходи(зважені за λr).

Таблиця 3.3 – Інтенсивності переходів МФМ2

№	**Параметр**	**Значення** $*$ $\mathbf{10^{-5}}$, 1/год
1	Інтенсивність відмови контакту (λ_{pin})	$\lambda_{res} + \lambda_{diod} = 0.03$
2	Інтенсивність першої деградації (λd_{x0})	$4 * \lambda_{pin} = 0.12$
3	Інтенсивність другої деградації (λd_{x1})	$3 * \lambda_{pin} = 0.09$
4	Інтенсивність третьої деградації (λd_{x2})	$2 * \lambda_{pin} = 0.06$
5	Інтенсивність відмови критичного елемента (λf)	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
6	Інтенсивність першої відмови ЗР (λr_{00})	$2 * \lambda_{pin} + \lambda_{tiny} = 0.062$
7	Інтенсивність другої відмови ЗР (λr_{01})	$1.33 * \lambda_{pin} + \lambda_{tiny} = 0.041$
8	Інтенсивність третьої відмови ЗР (λr_{10})	$0.66 * \lambda_{pin} + \lambda_{tiny} = 0.020$
9	Інтенсивність збоїв (λe)	$8 * \lambda_{pin} + \lambda_{tiny} = 0.242$
10	Інтенсивність відновлення після збоїв (μe)	1/60 = 1666
11	Інтенсивність відновлення після деградації (μe)	1/60 = 1666

Завдяки спільній точці початку розгортання описаних матриць, результат їх накладання утворює структуру, представлену рисунком 3.23.

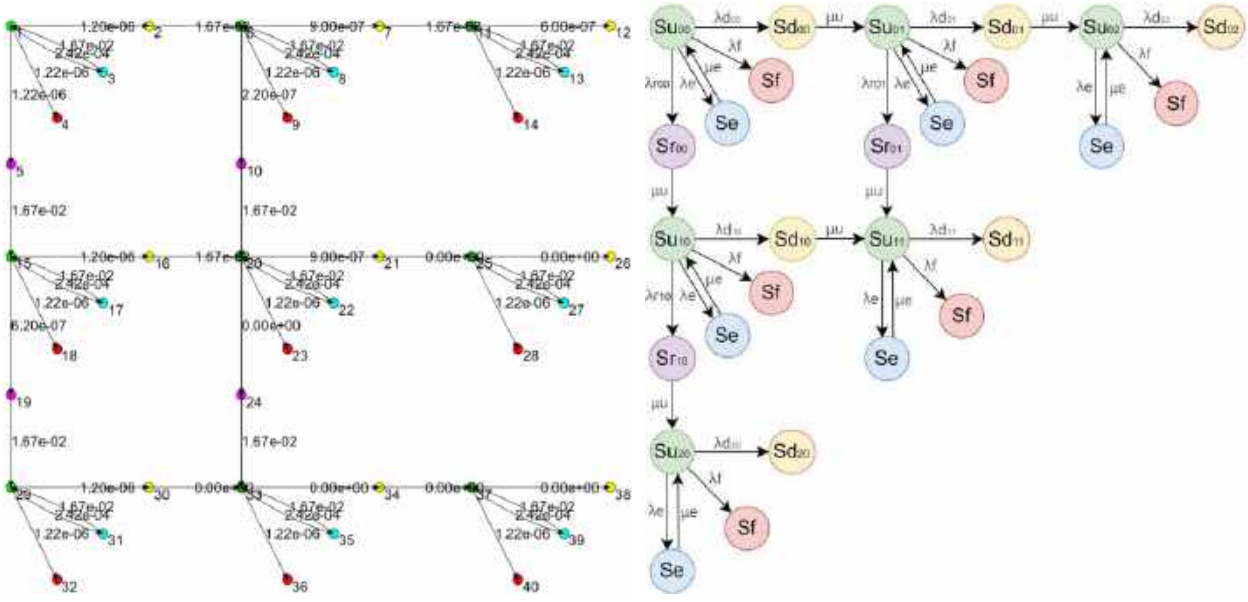

Рисунок – 3.23 МФМ2 з БРД та реконфігурацією в Matlab та draw.io

Результати моделювання представлені на рисунках 3.24 – 3.26. Як і в попередніх моделях, функція доступності зменшується від 1 до 0 на інтервалі 3-

106 годин (342,5 років). Сума ймовірностей перебування в станах Sf значно перевищує суму ймовірностей перебування в станах Sd.

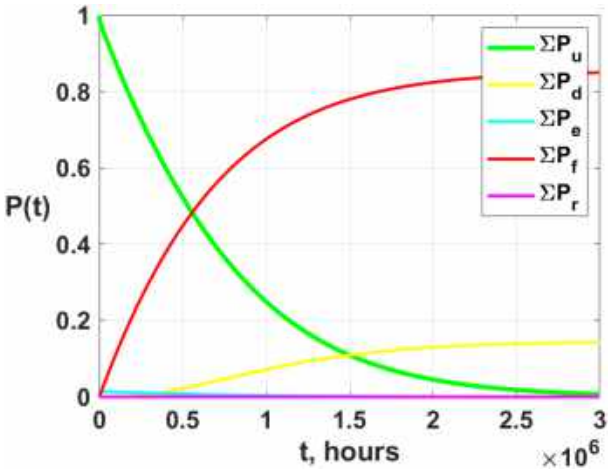

Рисунок 3.24 – Графік розподілу ймовірностей перебування у станах

Сума ймовірностей перебування в станах Se вже не є найменшою і суттєво перевищує суму ймовірностей перебування в станах Sr. Як і в моделі MFM01, сума ймовірностей перебування в Sf-станах після 12 000 годин (1,4 року) роботи починає перевищувати суму ймовірностей перебування в Se-станах.

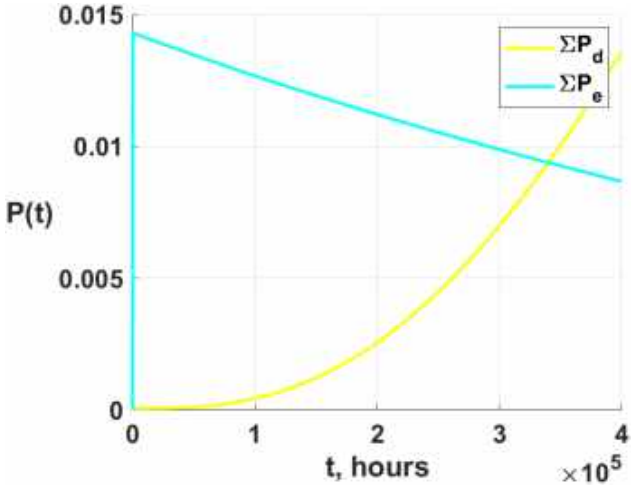

Рисунок 3.25 – Точка перетину функцій деградації та збою

Як показано на рисунку 3.25, сума ймовірностей перебування в Sd-станах після 340 000 годин (38,3 років) експлуатації починає перевищувати суму ймовірностей перебування в Se-станах. Це відмінність від попередньої моделі МФМ1.

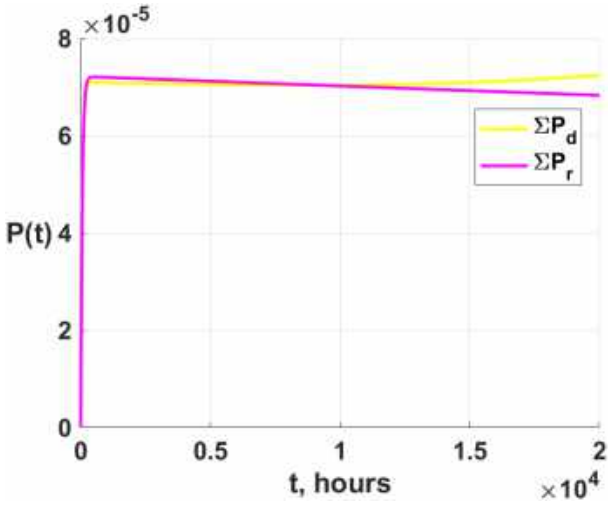

Рисунок 3.26 – Точка перетину функцій деградації ПП та деградації ЗР

3.4.6 Розроблення моделі МФМЗ з деградацією ресурсів програмовних пристроїв та резервованих засобів реконфігурації

Однак, завдяки відомим засобам підвищення надійності на деградацію засобів реконфігурації(ЗР) можливо впливати. В рамках МФМЗ розглядатиметься варіант резервування ЗР, представлений рисунком 3.27.

Хоча резервована ЗР додає додаткових елементів до вже існуючої МФМ, вона не збільшує кількість можливих процедур відновлення, але суттєво знижує їх втрату під час деградація внаслідок власних відмов.

Графічно, така модифікація призводить до збільшення кількості рівнів деградації ЗР за рахунок дублювання одного з ланцюгів деградації ПП. Таким чином, варіант БРД за двома параметрами деградації ПП та резервованої ЗР описуватиметься МФМ з «багатокутниковим» розташуванням фрагментів. У даному випадку перед деградацією ЗР в межах своїх можливостей, першу його відмову компенсує резерв, що не впливає на поведінку ПП.

Відповідно до представленої багатокутникової МФМ, існуючі стани в межах фрагментів не набувають суттєвих змін, оскільки подібна модифікація впливає лише на кількість вже існуючих фрагментів.

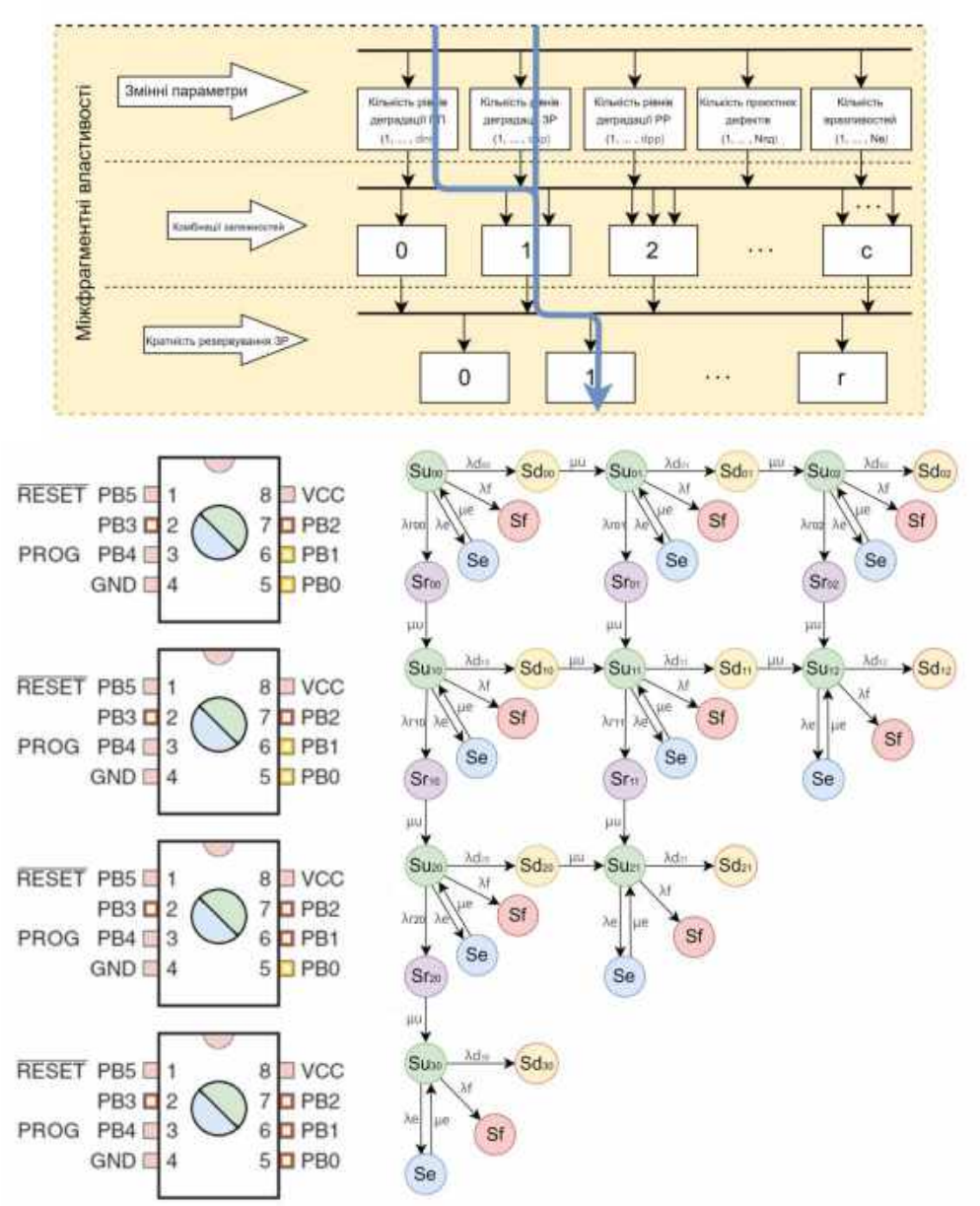

Рисунок 3.27 – Побудова резервованої трикутникової МФМ ПП з БРД

3.4.7 Дослідження моделі МФМ3 з деградацією ресурсів програмовних пристроїв та резервованих засобів реконфігурації

Через схожість МФМ2 та МФМ3 (fM03), для моделювання останньої було налагоджено вже розроблену матрицю параметрів з довільною розмірністю N на M. Ключовою особливістю є те, що тепер усі фрагменти МФМ3 містять по 5 станів, а їх задіювання регулюється матрицею маскування, що окрім міжфрагментних переходів, почала охоплювати і переходи між станами в межах задіяних фрагментів.

За аналогією до МФМ2, кількість горизонтальних фрагментів одного ланцюга МФМ3 визначається змінною кількістю рівнів деградації λd_{ij}, що визначається індексом ***j*** цього параметру. Кількість вертикальних переходів регулюється змінною λr_{ij}, що виражається її індексом ***i***, максимальне значення котрого визначається сукупністю ресурсу ЗР, кратністю та типом виконуваного резервування. При однократному резервуванні ЗР, глибина ітерацій складає 3 . Детальна параметризація МФМ3 надана у таблиці 3.4.

Таблиця 3.4 – Інтенсивності переходів МФМ3

№	**Параметр**	**Значення** $*$ $\mathbf{10^{-5}}$, 1/год
1	Інтенсивність відмови контакту (λ_{pin})	$\lambda_{res} + \lambda_{diod} = 0.03$
2	Інтенсивність першої деградації (λd_{x0})	$4 * \lambda_{pin} = 0.12$
3	Інтенсивність другої деградації (λd_{x1})	$3 * \lambda_{pin} = 0.09$
4	Інтенсивність третьої деградації (λd_{x2})	$2 * \lambda_{pin} = 0.06$
5	Інтенсивність відмови критичного елемента (λf)	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
6	Інтенсивність першої відмови ЗР (λr_{00})	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
7	Інтенсивність другої відмови ЗР (λr_{01})	$2 * \lambda_{pin} + \lambda_{tiny} = 0.062$
8	Інтенсивність третьої відмови ЗР (λr_{02})	$\lambda_{pin} + \lambda_{tiny} = 0.031$
9	Інтенсивність четвертої відмови ЗР (λr_{10})	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
10	Інтенсивність п'ятої відмови ЗР (λr_{11})	$2 * \lambda_{pin} + \lambda_{tiny} = 0.041$
11	Інтенсивність шостої відмови ЗР (λr_{20})	$0.66 * \lambda_{pin} + \lambda_{tiny} = 0.020$
12	Інтенсивність збоїв (λe)	$8 * \lambda_{pin} + \lambda_{tiny} = 0.242$
13	Інтенсивність відновлення після збоїв (μe)	1/60 = 1666
14	Інтенсивність відновлення після деградації (μe)	1/60 = 1666

У наведеному на рисунку 3.27 прикладі матриця розподілу параметру λd відповідає 12 загальним фрагментам, 9 з котрих мають ненульові переходи. Матриця розподілу параметру λr має розмірність 3 на 4, що свідчить про 12 загальних фрагменти, 6 з котрих мають ненульові переходи.

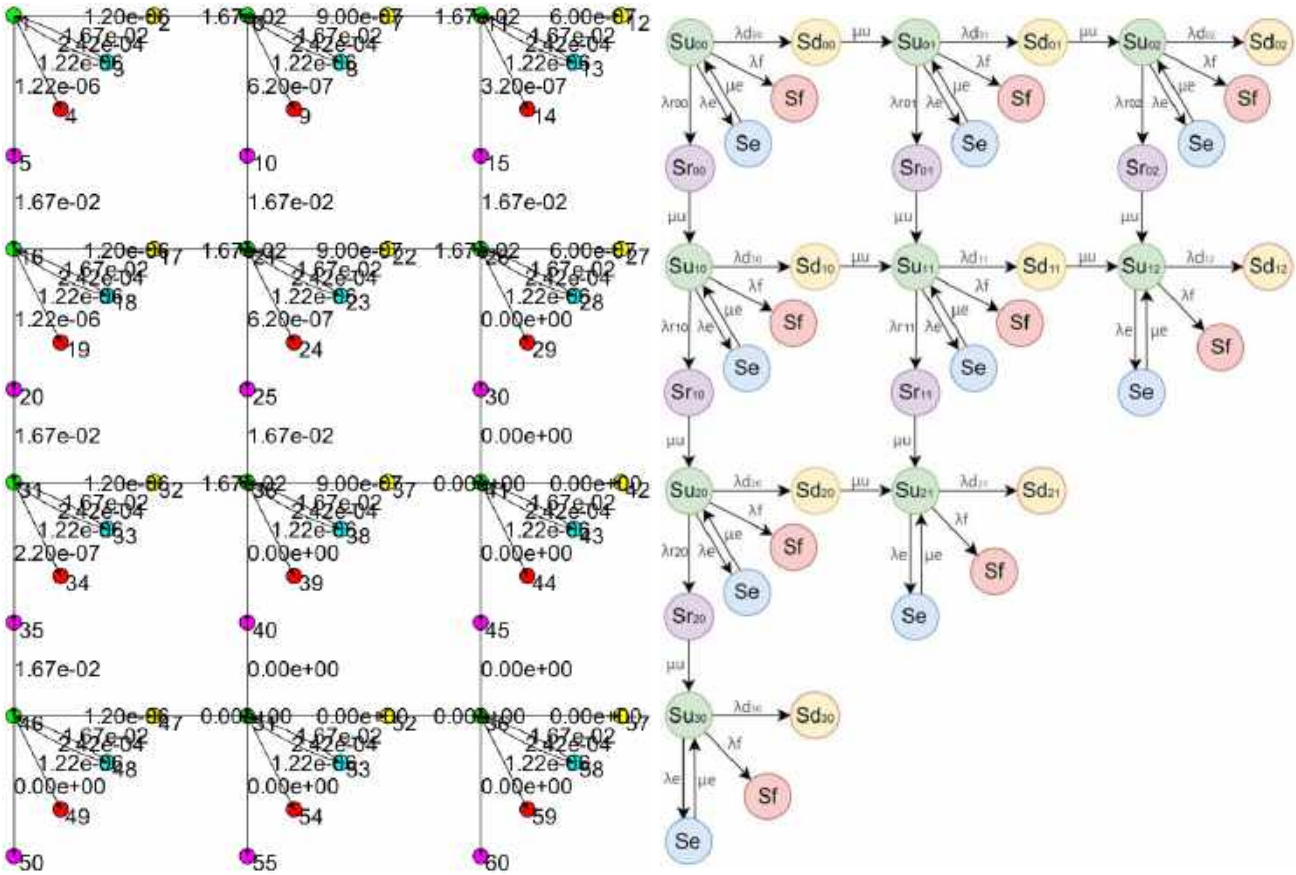

Рисунок 3.27 – МФМ3 з БРД та реконфігурацією в Matlab та draw.io

Результати моделювання, представлені на рисунку 3.28 демонструють зменшення функції від 1 до 0 доступності на інтервалі 3-106 годин (342,5 років).

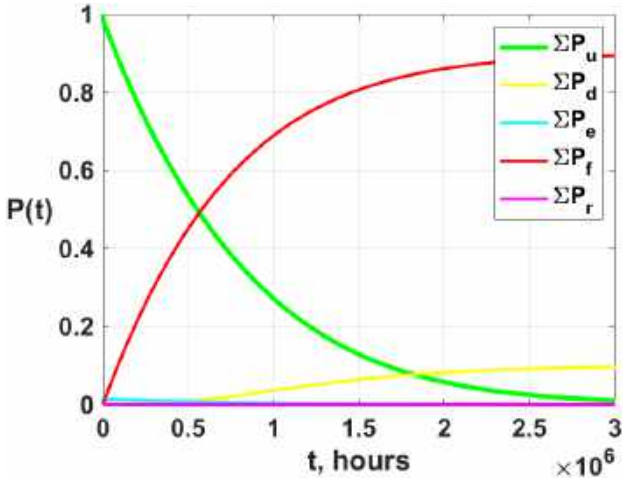

Рисунок 3.28 – Графік розподілу ймовірностей перебування у станах

Сума ймовірностей перебування в станах Se не є найменшою і суттєво перевищує суму ймовірностей перебування в станах Sr. Як і в МФМ2, сума ймовірностей перебування в Sf-станах після 12 000 годин (1,4 року) роботи перевищує суму ймовірностей перебування в Se-станах.

3.5 Особливості розроблення і дослідження мультифрагментної моделі готовності програмовних пристроїв МФМ4 з врахуванням помилок контролю та реконфігурації

Ще одним, вартим розгляду варіантом поведінки МФМ, варто вважати випадок, де вплив засобів контролю(ЗК) не може бути виключеним у повному обсязі.

Для МФМ4, що за способом організації її фрагментів є ідентичною до МФМ2, буде внесено зміни до складу станів власне самого фрагмента. Передбачається, що якість ЗК впливає на можливість реконфігурації ПП після прояву дефекту, що у свою чергу, впливає на швидкість його відновлення при контрольованій деградації. При цьому змінюється розподіл часу виникнення відмов та збоїв через збільшення часу простою.

Варто зазначити, що фіксація деградації ЗР призводиться безпомилково, зважаючи на те, що контроль призводиться лише за ПП. Окрім цього, виникаючі у ПП збої вважаються частиною нормального циклу роботи ПП і не входять до зони спостереження ЗК. Таким чином, варіант БРД з урахуванням помилок ЗК за двома параметрами деградації ПП та ЗР описуватиметься МФМ з «трикутниковим» розташуванням фрагментів, як представлено на рисунку 3.29.

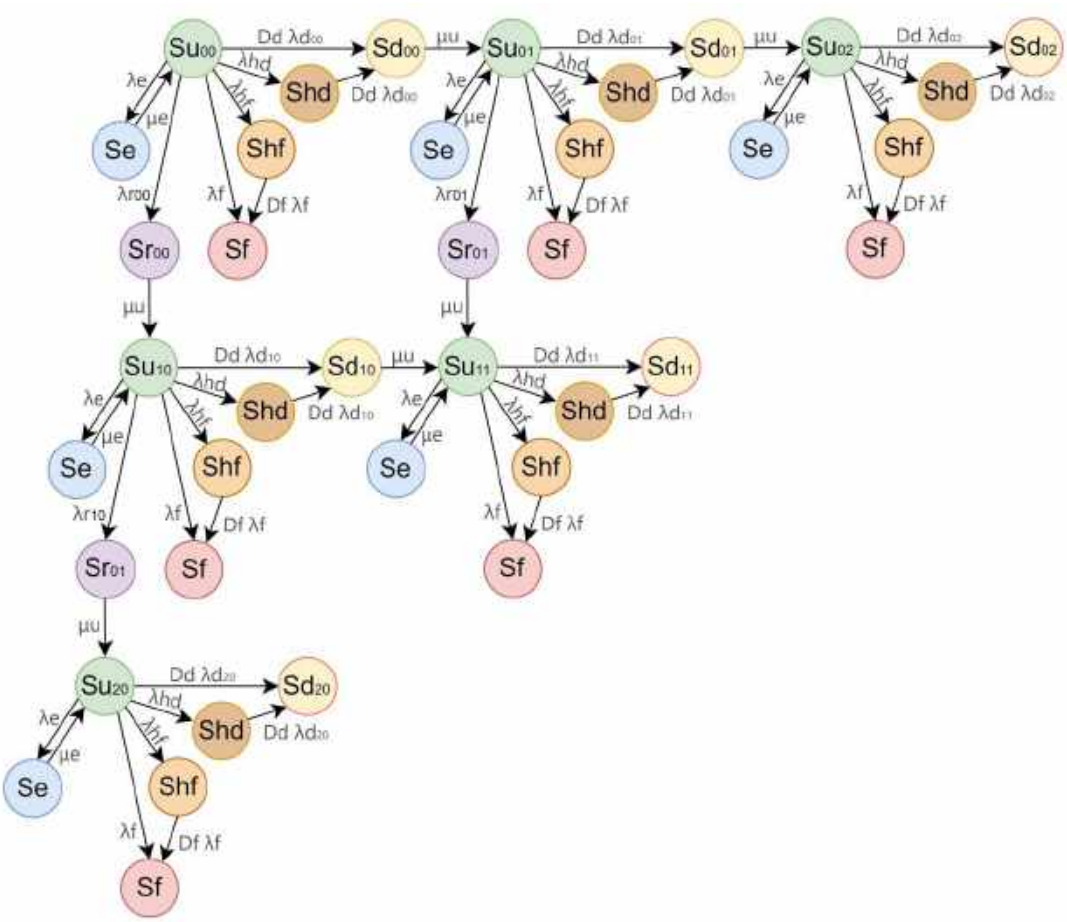

Рисунок 3.29 – МФМ4 з БРД та реконфігурацією в draw.io

Відповідно до представленої трикутникової МФМ, існуючі стани в межах фрагменту набувають наступних змін:

- до кожного стану відмови(Sf) додається стан прихованої відмови (Shf), до якого система може перейти замість стану відмови з інтенсивністю $\lambda hf = \lambda f * (1 - Df)$, де Df – достовірність контролю відмов;
- до кожного стану деградації(Sd_{ij}) додається стан прихованої деградації (Shd_{ij}), до якого система може перейти замість основного стану деградації з інтенсивністю λhd_x що дорівнює $\lambda d_{xi} * (1 - Dd)$, де Dd – достовірність контролю деградації.

Новим параметром, що впливає на поведінку МФМ4 виступає достовірність контролю(D). Залежно від параметру, що спостерігається, достовірність буде змінюватись відповідно складності споріднених контролю процедур.

За аналогією до попереднього випадку, при моделюванні МФМ4 для фрагментів розроблено матрицю довільних розмірів N на M. Для навігації між фрагментами моделі будуть взяті параметри, що використовувались при побудові МФМ2 з врахуванням заміни її базового фрагмента (ОФМ) на його ускладнений варіант зі станами прихованих відмов. Числові значення параметрів, що використовуються при симуляції МФМ4, представлено у таблиці 3.5.

Таблиця 3.5 – Інтенсивності переходів МФМ4

№	**Параметр**	**Значення** $*$ $\mathbf{10^{-5}}$, 1/год
1	Інтенсивність першої деградації (λd_{X0})	$4 * \lambda_{pin} = 0.12$
2	Інтенсивність другої деградації (λd_{X1})	$3 * \lambda_{pin} = 0.09$
3	Інтенсивність третьої деградації (λd_{X2})	$2 * \lambda_{pin} = 0.06$
4	Інтенсивність відмови (λf)	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
5	Інтенсивність першої відмови ЗР (λr_{00})	$2 * \lambda_{pin} + \lambda_{tiny} = 0.062$
6	Інтенсивність другої відмови ЗР (λr_{01})	$1.33 * \lambda_{pin} + \lambda_{tiny} = 0.041$
7	Інтенсивність третьої відмови ЗР (λr_{10})	$0.66 * \lambda_{pin} + \lambda_{tiny} = 0.02$
8	Достовірність ЗК деградації (Dd)	0.6
9	Достовірність ЗК відмови елемента (Df)	0.6
10	Інтенсивність збоїв (λe)	$8 * \lambda_{pin} + \lambda_{tiny} = 0.242$
11	Інтенсивність відновлення після збоїв (μe)	1/60 = 1666
12	Інтенсивність відновлення після деградації (μe)	1/60 = 1666

Таким чином, «прямокутна» структура моделі з параметризацією та маскуванням нульових переходів для набуття «трикутної» форми, представлена рисунком 3.30.

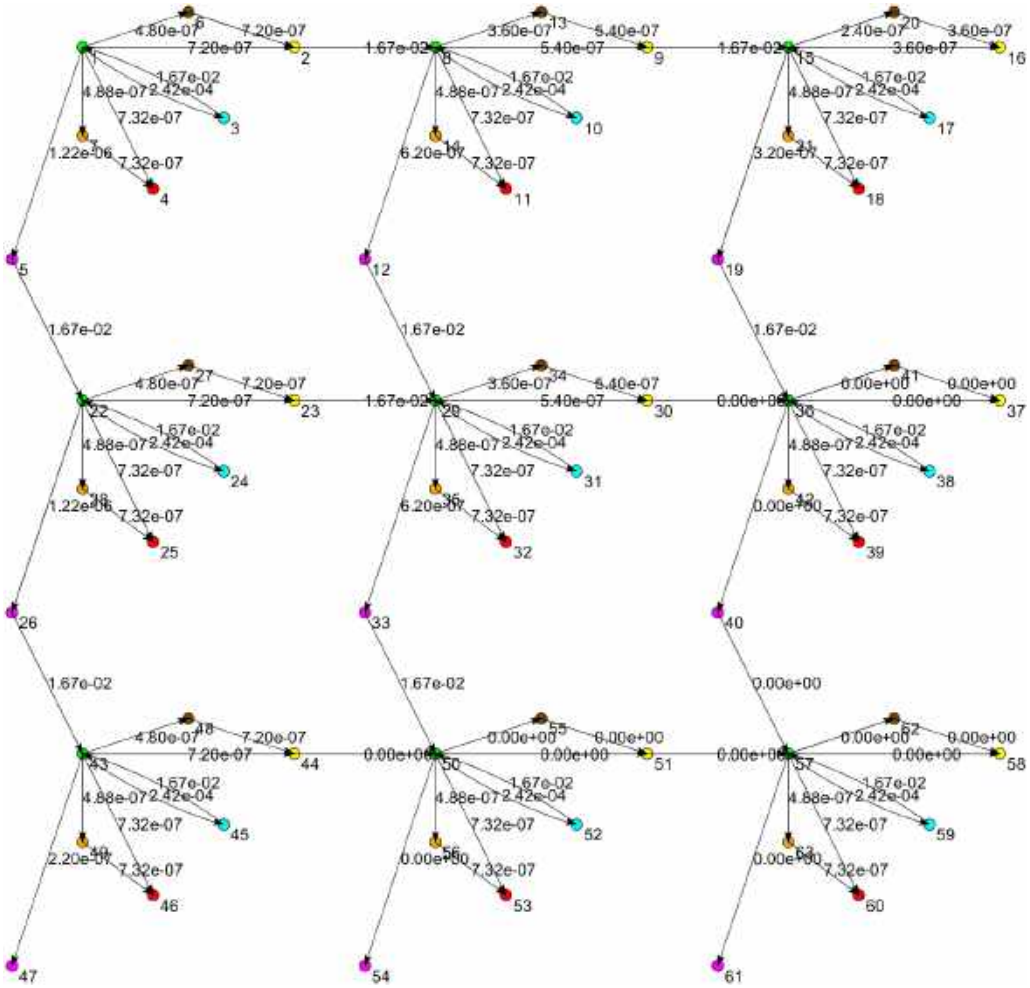

Рисунок 3.30 – МФМ4 з БРД та реконфігурацією в MatLab

Відповідно до результатів симуляції, представленої на рисунку 3.31, функція готовності спадна від 1 до 0 на інтервалі $3*10^6$ годин (342,5 років). Сума Ймовірностей перебування в станах Sf суттєво перевищує суму ймовірностей перебування в станах Sd. Сума ймовірностей перебування в станах Se вже не найменша, а суттєво перевищує суму ймовірностей перебування в решті станів системи. Вірогідність перебування у прихованих станах на інтервалі у перші 273 років функціонування є переважливою за стан деградації та відмови.

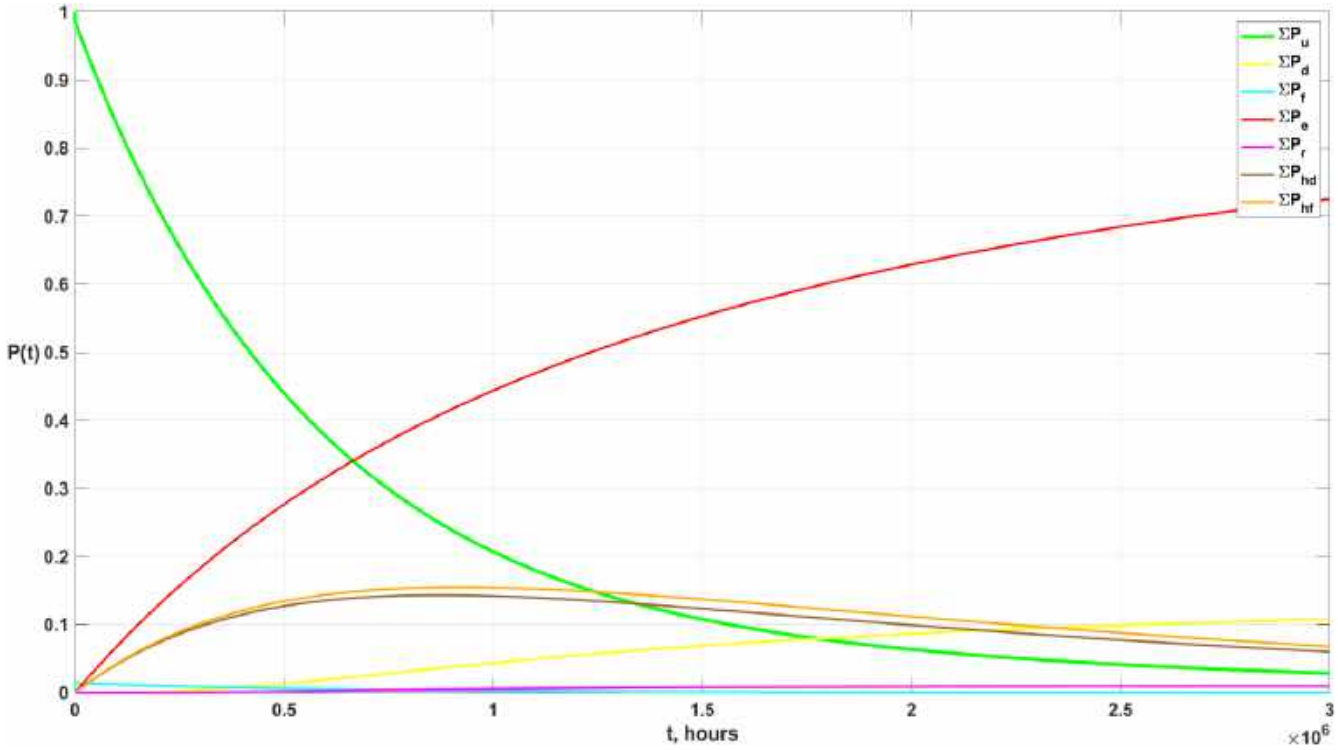

Рисунок 3.31 – Графік розподілу ймовірностей перебування у станах

Таким чином, завдяки високому ступеню виникнення простоїв, спад функції готовності протягом терміну експлуатації має помірніший характер, ніж у випадку, коли похибки при роботі ЗК не враховуються.

3.6 Порівняльний аналіз результатів дослідження моделей МФМ1-МФМ4

При порівнянні працездатності розроблених МФМ використовувався спільний набір числових значень ключових параметрів самого програмовного пристрою(ПП), супроводжуючого засобу реконфігурації(ЗР) та відповідного до них засобу контролю(ЗК). Таким чином при порівнянні буде прослідковуватися вплив підходу до організації керованої багаторівневої деградації(БРД) на схожі між собою моделі. Зведений перелік числових значень розглянутих моделей представлено таблицею 3.6.

Таблиця 3.5 – Зведена таблиця інтенсивності переходів МФМ

№	**Параметр**	**Значення** $*$ $\mathbf{10^{-5}}$, 1/г
1	Інтенсивність відмови діоду (λ_{diod})	0.02
2	Інтенсивність відмови резистора (λ_{res})	0.01
3	Інтенсивність відмови мікроконтролеру (λ_{tiny})	0.002
4	Інтенсивність відмови контакту (λ_{pin})	$\lambda_{res} + \lambda_{diod} = 0.03$
5	Інтенсивність першої деградації (λd_{X0})	$4 * \lambda_{pin} = 0.12$
6	Інтенсивність другої деградації (λd_{X1})	$3 * \lambda_{pin} = 0.09$
7	Інтенсивність третьої деградації (λd_{X2})	$2 * \lambda_{pin} = 0.06$
8	Інтенсивність відмови (λf)	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
6	Інтенсивність першої відмови ЗР (λr_{00})	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
7	Інтенсивність другої відмови ЗР (λr_{01})	$2 * \lambda_{pin} + \lambda_{tiny} = 0.062$
8	Інтенсивність третьої відмови ЗР (λr_{02})	$\lambda_{pin} + \lambda_{tiny} = 0.031$
9	Інтенсивність четвертої відмови ЗР (λr_{10})	$4 * \lambda_{pin} + \lambda_{tiny} = 0.122$
10	Інтенсивність п'ятої відмови ЗР (λr_{11})	$2 * \lambda_{pin} + \lambda_{tiny} = 0.041$
11	Інтенсивність шостої відмови ЗР (λr_{20})	$0.66 * \lambda_{pin} + \lambda_{tiny} = 0.020$
12	Інтенсивність збоїв (λe)	$8 * \lambda_{pin} + \lambda_{tiny} = 0.242$
13	Інтенсивність відновлення після збоїв (μe)	1/60 = 1600
14	Інтенсивність відновлення після деградації (μe)	1/60 = 1600
15	Достовірність ЗК деградації (Dd)	0.6
16	Достовірність ЗК відмови елемента (Df)	0.6

При порівнянні між собою лінійної МФМ1 та трикутникової МФМ2, що демонструє рисунок 3.31, сума ймовірностей перебування в станах деградації Sd після 7000 годин (9,7 місяців) роботи починає перевищувати суму ймовірностей перебування в стана деградації ЗР (Sr).

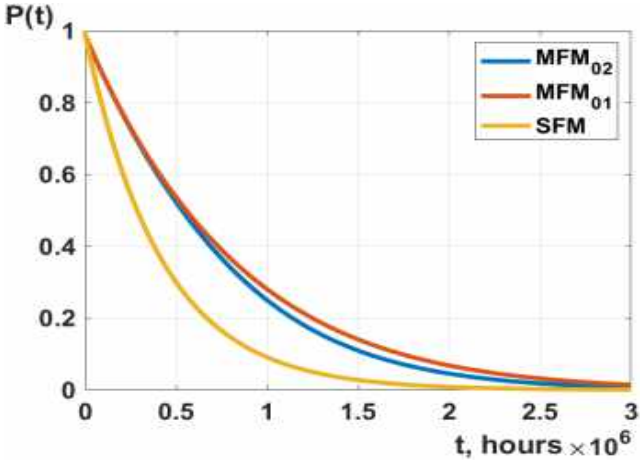

Рисунок 3.31. – Порівняння результатів оцінки надійності ПП з використанням ОФМ, МФМ

При наближеному вивченні динаміки функцій готовності, представлених графіками рисунку 3.32, можна зазначити, що МФМ1 має стабільнішу криву розвитку деградації. У той самий час МФМ2 показує невеликий приріст готовності у перші 150 годин експлуатації.

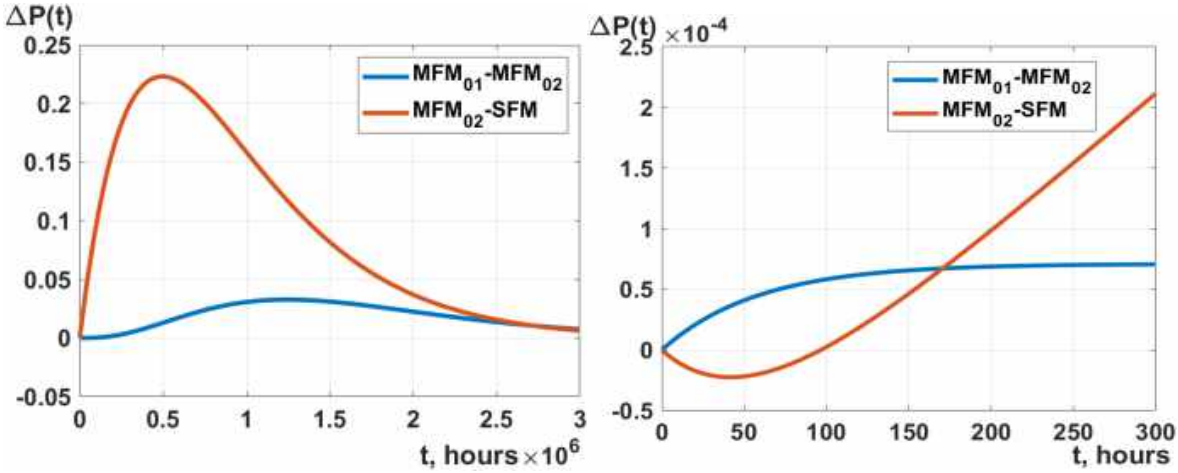

Рисунок 3.32. – Аналіз переваг експлуатації МФМ1, МФМ2 та ОФМ.

У порівнянні з ОФМ, модель МФМ2 демонструє втрату $0{,}25*10^{-4}$ в інтервалі часу [0...100] годин. Однак у наступному спостерігається покращення на 0,03, з піком на $1{,}2*10^{6}$ годин (137 років).

При порівнянні між собою функцій готовності трикутникової МФМ2 та трикутникової з резервуванням МФМ3, що демонструє рисунок 3.33, сума ймовірностей перебування в станах деградації Sd після 14000 годин (1,6 року) роботи починає перевищувати суму ймовірностей перебування в Sr-станах.

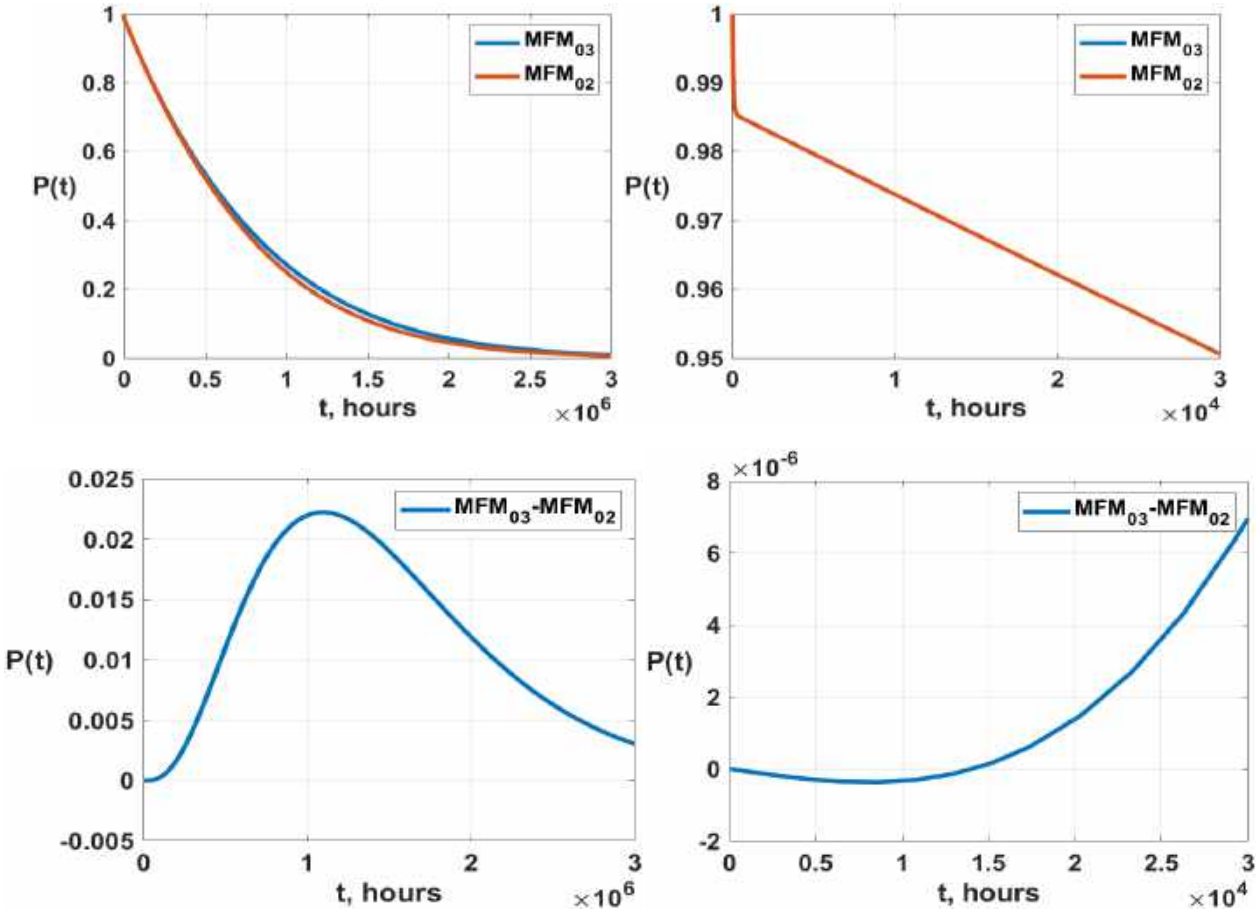

Рисунок 3.33. – Аналіз переваг для тривалого та короткого часу експлуатації МФМ2 та МФМ3.

При порівнянні між собою функцій готовності трикутникової МФМ2 та трикутникової з врахуванням прихованих відмов МФМ4, що демонструє рисунок 3.34, сума ймовірностей перебування в станах деградації Sd після 14000 годин (1,6 року) роботи починає перевищувати суму ймовірностей перебування в Sr-станах.

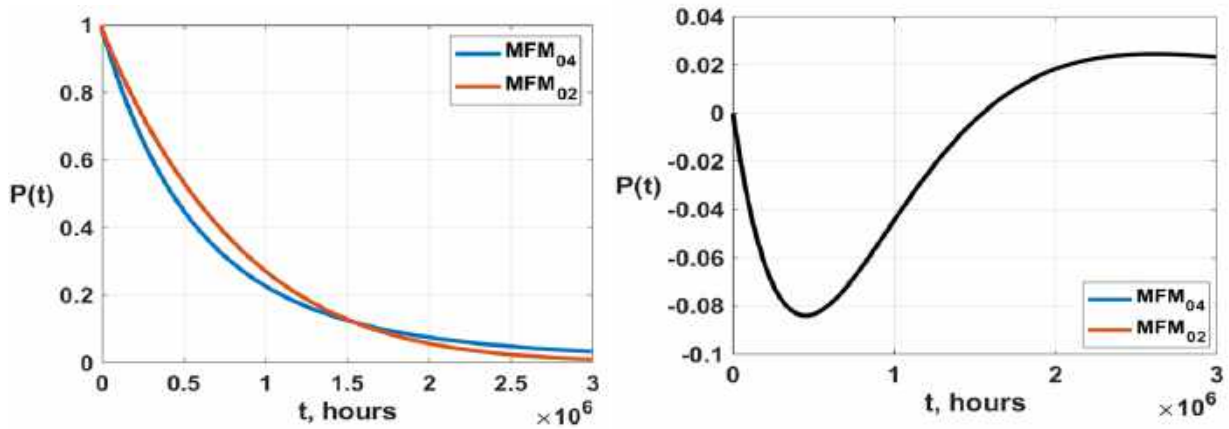

Рисунок 3.34. – Аналіз переваг експлуатації МФМ2 та МФМ4.

При порівнянні результатів симуляції МФМ2 та МФМ4 можна зробити висновок, що на інтервалі до $1{,}54*10^6$ годин модель з прихованими відмовами МФМ4 програє в готовності моделі МФМ2 без прихованих відмов, через високу інтенсивність виникнення станів прихованих відмов/деградацій. Однак, після перетину цієї точки у часі ситуація змінюється на протилежну за рахунок простою МФМ4, що виступило у якості короткотривалих режимів зберігання ресурсу деградації, розтягуючи його використання у часі.

3.7 Висновки до третього розділу

У розділі виконано дослідження, що дають змогу сформулювати наступні висновки.

1. Розроблення класифікатора моделей готовності програмовних пристроїв з багаторівневою деградацією. Під час розроблення класифікатора було обґрунтовано 7 ознак моделей готовності, а саме:

- наявність змінних параметрів ПП;
- наявність потенційної реконфігуропридатності;
- характер реконфігуропридатності;
- врахування помилок засобів контролю;
- відповідність змінних параметрів;
- кратність комбінацій залежних одне від одного змінних параметрів;
- кратність резервування засобів реконфігурації.

Для описаних ознак було виділено ключові сутності та зв'язки між ними, що було у графічному вигляді у якості класифікатора.

2. Розроблення моделей готовності програмовних пристроїв. Під час розроблення моделей готовності було визначено дві основні групи моделей – однофрагментні та мультифрагментні. Для кожної групи було розроблено ряд типових моделей з поступовим додаванням врахованих ознак, таких як: наявність помилок контролю, кількість деградуючих параметрів, наявність резервування засобів реконфігурації, тощо.

3. Дослідження однофрагментних моделей готовності програмовних пристроїв призводилось за рахунок параметризації ОФМ та розрахунку її функції готовності.

4. Дослідження мультифрагментних моделей готовності програмовних пристроїв. Призводилось за рахунок їх параметризації та розрахунку функції готовності при врахуванні різних комбінацій ознак.

В рамках дослідження було розглянуто 4 мультифрагментні моделі, а саме:

- одновимірна «лінійна» МФМ1 з деградацією ПП;
- двовимірна «трикутникова» МФМ2 з деградацією ПП та ЗР;
- двовимірна «багатокутна» МФМ3 р з деградацією резервованої ЗР та ПП;
- двовимірна «трикутникова» МФМ4 з деградацією ПП ,ЗР та помилками ЗК.

5. Порівняльний аналіз результатів дослідження розроблених МФМ показав найкращі результати при обчисленні функції готовності у «лінійної» МФМ1. Такий результат демонструє вплив способу реалізації засобів контролю та реконфігурації на працездатність ПП при керуванні багаторівневою деградацією.

Таким чином, в розділі отримано третій новий науковий результат:

- удосконалено марковські моделі готовності програмовних пристроїв з керованою деградацією, що враховують інтенсивності відмов множин компонентів, визначених залежно від можливості відновлення завдяки програмно-керованій реконфігурації без (або) з частковою втратою якості виконання функцій, часові і достовірнісні характеристики процедур перебудови, а також наявність резервування та часткову працездатність засобів реконфігурації, що забезпечує можливість розрахунку функцій готовності при багатоступеневій деградації.

РОЗДІЛ 4
РОЗРОБЛЕННЯ ЗАСОБІВ ЗАБЕЗПЕЧЕННЯ НАДІЙНОСТІ ПРОГРАМОВНИХ ПРИСТРОЇВ БЕСПІЛОТНИХ СИСТЕМ З КЕРОВАНОЮ БАГАТОРІВНЕВОЮ ДЕГРАДАЦІЄЮ. ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Послідовність використання та особливості програмно-апаратної реалізації запропонованих моделей і методів

На підставі розроблених в другому і третьому розділах моделей і методів сформулюємо наступну послідовність розроблення засобів оцінювання і забезпечення надійності програмовних пристроїв для безпілотних систем:

- визначення вимог до показників надійності ПП, особливостей їх побудови і можливостей реалізації реконфігурації при відмовах;

- аналіз внутрішньої структури ПП, класифікація компонентів за ознаками дефектів та їх впливу на рівень працездатності з використанням структурних і теоертико-множинних моделей, описаних в підрозділі 2.1;

- визначення комплексу процедур реконфігурації та оцінювання показників надійності ПП з керованою БРД при впровадженні засобів контролю та реконфігурації з використанням розроблених аналітичних моделей (підрозділ 2.2);

- аналіз і оцінювання реконфігуропридатності (підрозділ 2.3), визначення складових засобів контролю та реконфігурації згідно методу реконфігурації програмовних пристроїв безпілотних систем з керованою багаторівневою деградацією (підрозділ 2.4);

- оцінювання готовності програмовних пристроїв для безпілотних систем з керованою багаторівневою деградацією з використанням розроблених моделей. Для цього на підставі аналізу ПП та моделей (підрозділи 2.1, 2.2), визначення можливостей і обсягу реалізації запропонованого методу реконфігурації (підрозділи 2.3, 2.4) обирають одну з базових моделей для оцінювання готовності

ПП на підставі класифікатора (підрозділ 3.1) та виконують її розроблення та дослідження (підрозділи 3.2-3.4);

- врахування за необхідності можливих помилок засобів контролю і реконфігурації з використанням моделей і методики, наланих у підрозділ 3.5;

- формування висновку на підставі аналізу результатів оцінювання (підрозділ 3.6) стосовно відповідності ПП та сформованого комплексу рішень вимогам та їх коригування за необхідністю;

- розроблення засобів забезпечення надійності і живучості програмовних пристроїв для вбудовування і модернізації безпілотних систем різних типів (літальних, наземних морських).

Для підтвердження достовірності наукових результатів і можливостей практичної реалізації запропонованих методу та моделей розроблено комплекс програмних і апаратних засобів:

- програмний засіб моніторингу та виправлення контрольних сумм, що надає набір інструментів для часткового контролю цілісності даних при взаємодії програмовних пристроїв.

- програмний модифікатор завантжувача програмовних пристроїв для забезпечення керованості процессу їх реконфігурації;

- багатоцільова апаратна робот-система для багатоверсійної реалізації засобів контролю та рекнфігурації з різними типами деградації внутрішнього ресурсу.

Для зазначеного комплексу програмних і апаратних засобів надамо їх стислий опис в підрозділах 4.2-4.4.

4.2 Програмний засіб для забезпечення реконфігуропридатності

4.2.1 Структура програмного засобу “Optiboot”

Програмний засіб “ Optiboot ” має однорівневу структуру[76] та складається монолітного модуля, та набору його параметрів та підпрограм для модифікацій

різних моделей мікроконтролерів. На рисунку 4.1 зображена діаграма, де представлено залежності компонентів програмного засобу.

Блок *pin_defs.h* є бібліотечним файлом конфігурацій портів ПП. У цьому модулі представлено шаблони налаштувань портів відповідних моделей ПП, що підтримуються програмним засобом.

Блок *Common_Def.h* є файлом конфігурації програмних блоків. У цьому модулі представлено налаштування розмірів, позиціонування у пам'яті та кількості програмних блоків, якими оперує завантажувач при перепрограмуванні.

Блок *boot_opt.h* є файлом налаштувань режимів роботи завантажувача[77]. Модуль містить інструкції щодо проведення збросу, порядку передачі та роботи зі змістом програмних блоків.

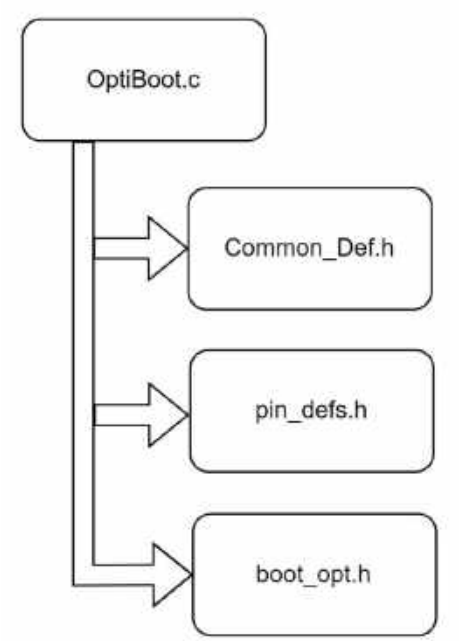

Рисунок 4.1 – Діаграма змістовних блоків " Optiboot "

Модуль *OptiBoot.c* реалізує загальні функції з модифікації ПП (налаштування черги обміном даних, розподіл програмних блоків, тощо) та забезпечує функцію прийому та передачі даних. Зміст поточного модуля можна представити у вигляді діаграми класів на рисунку 4.2, де продемонстровано основні елементи засобу.

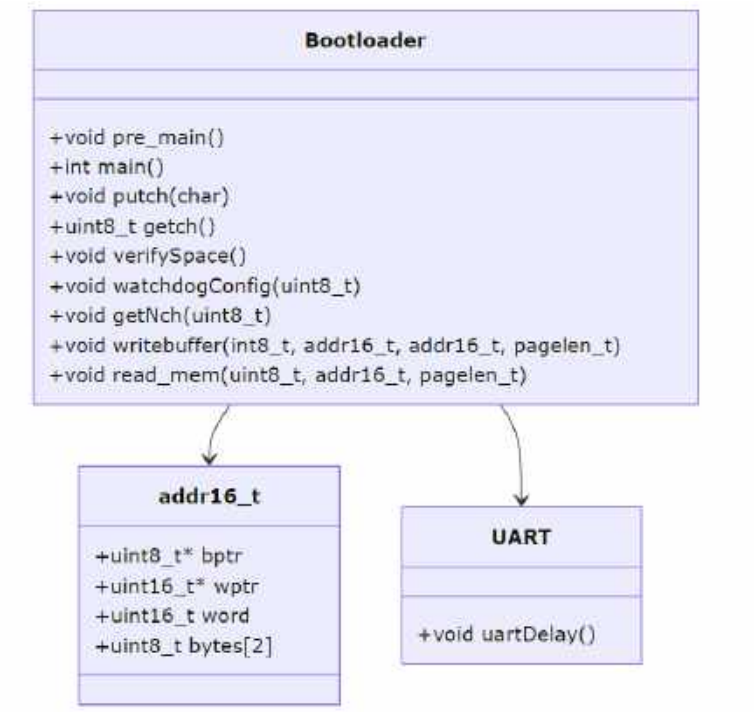

Рисунок 4.2 – Діаграма класів модуля “ OptiBoot.c ”

Модуль не продукує нових даних під час своєї роботи, оскільки він працює з вже наявним змістом ПП, до складу якого він належить. В окремих випадках та модифікаціях залишається можливість змін та збереження версій[78] використованого програмного засобу.

4.2.2 Опис базового функціоналу програмного засобу “Optiboot”

Оскільки програмний засіб “Optiboot” представляється у вигляді комплекту файлів модифікації прошивки ПП, елементом взаємодії з ним виступають текстові файли DSV (TSKV)[79] або h-файли декларації та макро-визначень[80] C/C++. Відповідно до них, користувачеві доступні для налаштування наступні характеристики програмного засобу:

1. *Налаштування розміру та кількості сторінок програмних блоків*. Залежно від потреб систем, до складу яких входить ПП, розмір пакетів та тривалість обміну можуть варіюватися. Відповідна можливість налаштування додає гнучкості при побудові вбудованих систем.

2. *Налаштування розміру завантажувача*. Опція дозволяє визначати об'єм пам'яті, який виділяється для завантажувача, що надає змогу користувачеві підбирати оптимальне співвідношення пам'яті програми до пам'яті для завантажувача відповідно до вирішуваних мікроконтролером завдань.

3. *Налаштування формату обліку версій програмного засобу*. Опція необхідна для контролю версій програмного забезпечення ПП під час реконфігурацій.

4. *Налаштування частот взаємодії пристроїв*. Необхідне для синхронізації та усунення колізій під час обміну даними між пристроями.

5. *Вибір пріоритетного способу взаємодії пристроїв*. Залежно від складу пристроїв у мультиконтролерних системах, налаштування їх інтерфейсів взаємодії можуть різнитися. Для цього передбачається кілька варіантів взаємодії, які можна змінювати під час роботи.

4.2.3 Приклади застосування програмного засобу “Optiboot”

При використанні завантажувача Arduino є можливим розглядати проект, який має «двопровідний» (напівдуплексний) послідовний інтерфейс RS-485, що є доцільним, коли роз'єм ISP є важкодоступним через корпусне виконання плати. Тому виникає потреба використовувати інтерфейс RS-485 для програмування мікросхеми. Для цього підходить апаратний UART мікросхеми для інтерфейсу RS-485 і завантажувач, підлаштований на напівдуплексний режим[81,82].

Прикладом застосування Optiboot є ПП з інтерфейсом зв’язку RS485 з додатковим модулем SN75176AD, де за рахунок використання лінії RX/TX взаємодія з іншими пристроями мультиконтролерної системи об’єднується з каналом програмування плати. Завдяки такій модифікації, інші контакти порту ПП залишаються незадіяними і можуть виступати додатковим ресурсом при реконфігураціях[83,84]. Схематичне представлення внесених змін представлено на рисунку 4.3.

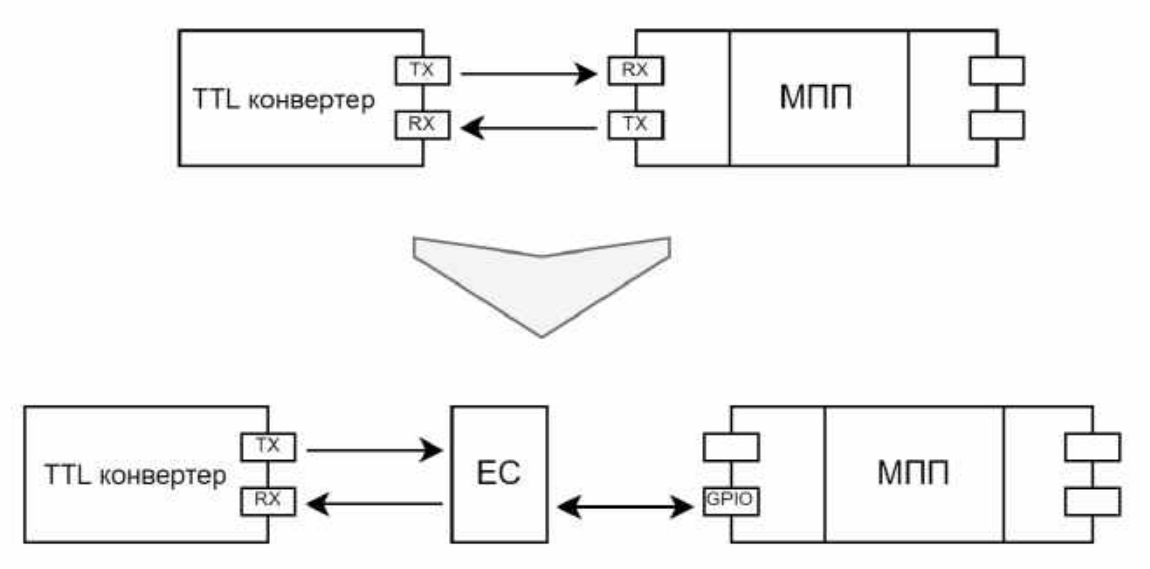

Рисунок 4.3 – Схема модифікації взаємодії ПП при застосуванні засобу “Optiboot” для виділення резервного каналу зв’язку

Після модифікації ПП за допомогою optiboot для перевірки завантаження та усунення перешкод від драйвера RS485 на лінії RX за допомогою стандартних ліній RX/TX, додавши до основної функції наступне:

```
DDRD |= _BV(PORTD5) ;
PORTD |= _BV(PORTD5);
```

Де цифровий вивід 5 (PORTD5) – це лінія керування, яка розподіляється між виводами DE/RE на драйвері. При переведенні драйвера в режим передачі (DE = RE = HIGH) його лінія RX звільняється, що дозволяє завантажувати дані за допомогою спарених стандартних ліній RX/TX переводячи трансивер у стан передачі, а потім повертаючи його назад у стан прийому[85,86].

Схожим прикладом є використання зв'язки RS485 та MAX483 для реалізації TX/RX обміну. В цьому випадку використовується цифровий PIN2 для керування передачею/прийомом даних. При кожному надсиланні даних у RS485 трансиверу повідомляється, коли перейти у режим передачі за допомогою контролю над лінією DE/RE функцією putch(), що відповідає за побайтове надсилання даних послідовних каналом[87]. Таким чином, для перепрограмування достатньо надіслати пакет оновлення через RS485 у режимі прийому. При цьому немає

потреби використовувати кнопок скидання, живлення або іншого прямого з'єднання. Схематичне представлення внесених змін представлено на рисунку 4.4.

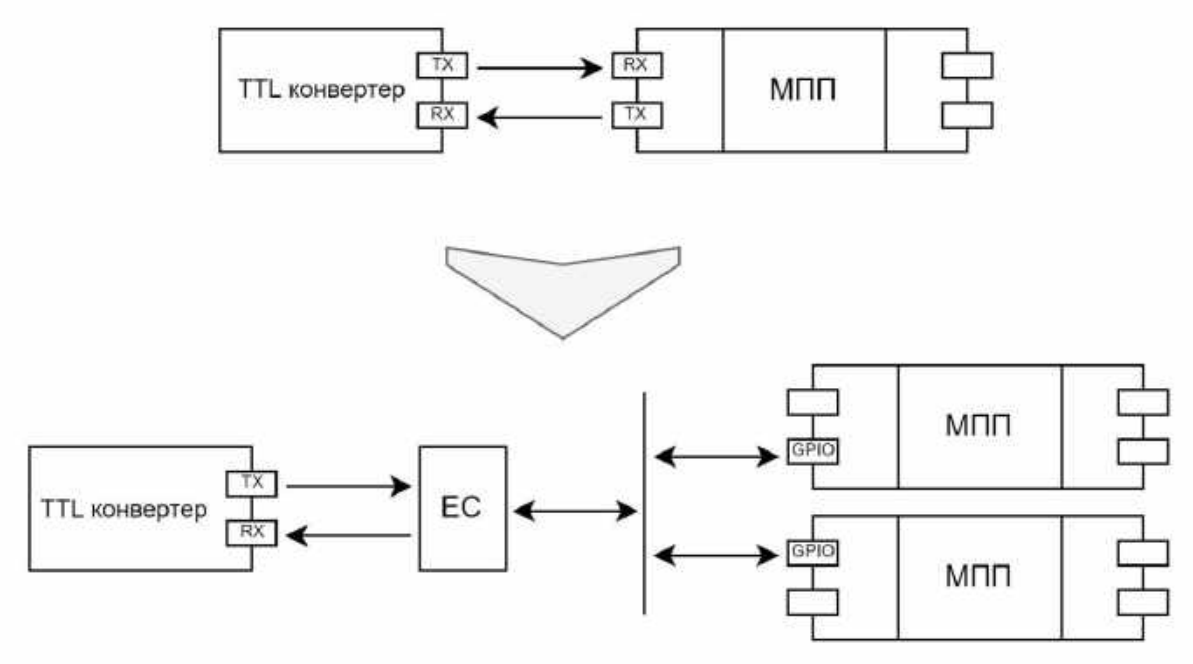

Рисунок 4.4 – Схема модифікації взаємодії ПП при застосуванні засобу “Optiboot” для взаємодії з групою пристроїв

При керуванні оновленням скриптів у мережі пристроїв, є можливість автоматизації процесу їх реконфігурації. Завдяки модифікації завантажувачів керованих пристроїв стає можливим надсилання команд «reset» перед завантаженням програмного блоку вибірково до одного або групи пристроїв[88]. Таким чином стає можливим перепрограмування обраних пристроїв не лише за допомогою конвертера, а й з використанням інших мікроконтролерів, які належать до поточної мережі пристроїв.

4.3 Програмний засіб для виправлення контрольних сум у файлах для програмування AVR-мікроконтролерів “FixAVRHexCheckSum”

4.3.1 Структура програмного засобу “FixAVRHexCheckSum”

Програмний засіб “FixAVRHexCheckSum” має модульну форму, в якій кожен з модулів відповідає за окремі функції застосунка. На рисунку 4.5

зображена діаграма класів, де представлено залежності компонентів програмного засобу.

Модуль *MainForm.cs* реалізує графічний інтерфейс з набором функцій для ручної роботи зі змістом hex-файлів[89]. У цьому модулі представлено базові функції роботи з файлами, такі як копіювання, запис, видалення, та графічне відокремлення змістовних масивів даних за призначенням.

Модуль *AvrHex.cs* реалізує набір функцій для автоматизованого визначення та корекції порушень у змісті hex-файлів. До складу набору відносяться такі функції, як відстежування, перерахунок та виправлення контрольних сум[90].

Підмодуль *AvrHexLine.cs* містить набір функцій для зміни форми представлення hex-файлів. Підмодуль є складовим модуля *AvrHex.cs* і відповідає за представлення hex-файлів у вигляді набору строк регламентованої довжини, що необхідно при розподілі даних строки за функціональним призначенням у зручному для аналізу вигляді[91].

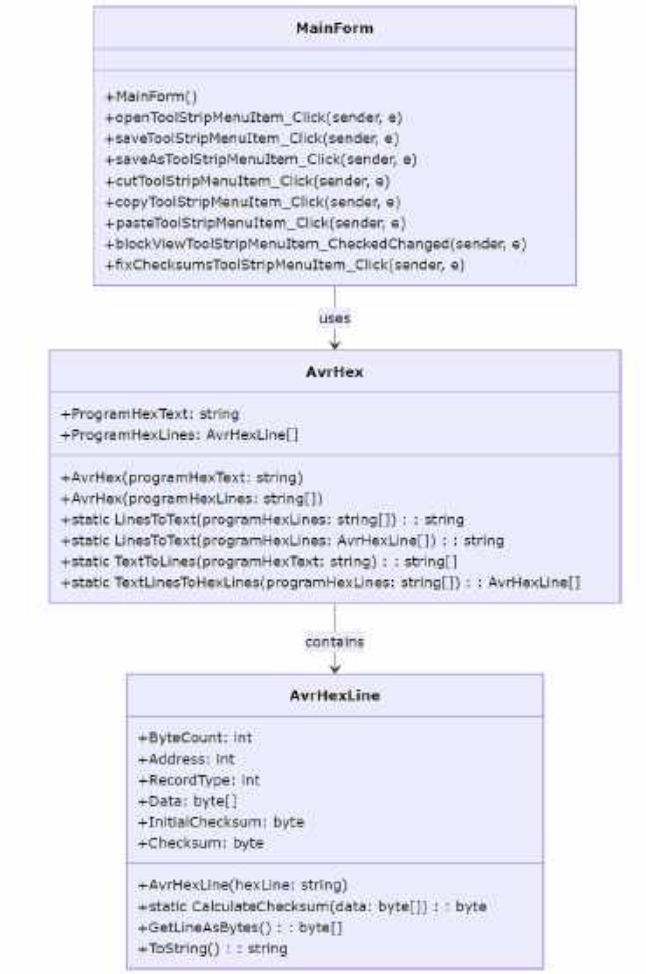

Рисунок 4.5 – Діаграма класів програмного засобу “FixAVRHexCheckSum”

Збережені засобом дані, що формуються в наслідок його роботи, можуть використовуватися під час діагностичних операцій при передачі програмних блоків між ПП[92] та виступати у якості інструменту збереження цілісності даних при обміні.

4.3.2 Опис базового функціоналу “FixAVRHexCheckSum”

Графічний інтерфейс програмного засобу “FixAVRHexCheckSum” являє собою вікно редактора hex-файлів та набір інструментів взаємодії зі змістом. Зовнішній вигляд вікна представлено на рисунку 4.6:

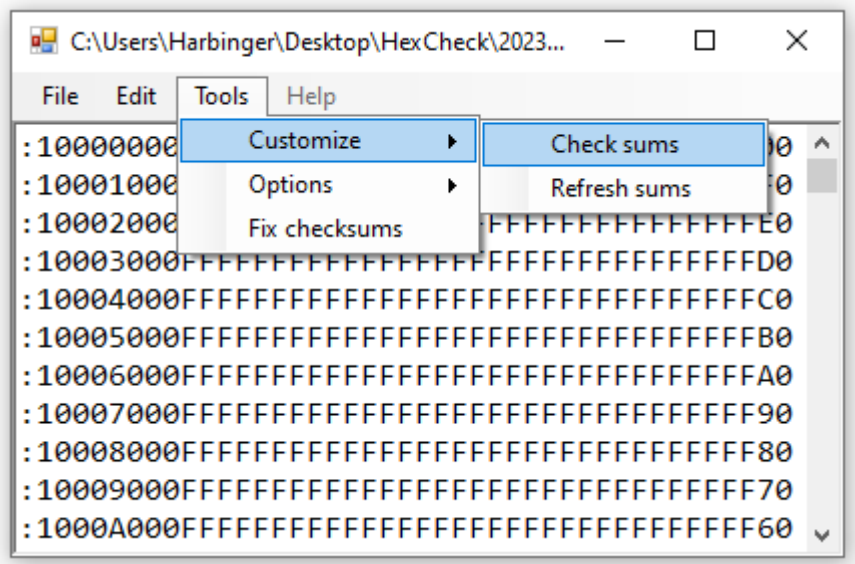

Рисунок 4.6 – Графічний інтерфейс “FixAVRHexCheckSum”

Відповідно, у вікні програмного засобу користувачеві доступні для налаштування наступний набір інструментів:

1. *Набір для роботи з файловою системою*. Налічує стандартний набір функцій для роботи з файлами, таких як відкрити файл(Open), зберегти зміни(Save), зберегти з модифікатором (Save as) та закрити (Close).

2. *Набір для роботи зі змістом файлу*. Набір дозволяє накладати виділення на зміст файлу (Select All), дописувати (Paste), видаляти (Cut), копіювати (Copy), проводити навігацію по змінах у файлі (Undo, Redo).

3. *Набір для автоматизованого оброблення змісту файлу*. Інструмент має розділення за призначенням, а саме графічні та виправляючі. До графічних можна віднести кольорову розмітку змісту файлу за функцій ним призначенням частин строк, маркування строк даних, де було виявлено невідповідність контрольних сум. До виправляючи відносяться функції перевірки, оновлення та виправлення контрольних сум.

4.3.3 Приклади застосування програмного засобу “FixAVRHexCheckSum”

Виходячи з зазначених функцій програмного засобу, його використання передбачає роботу з hex-файлами обмеженого розміру. При використанні його у парі з програмним засобом "Optiboot", який використовує посторінкове надсилання даних, застосунок може використовуватися як частина засобів контролю[93]. Приклад такого застосування зображено на рисунку 4.7.

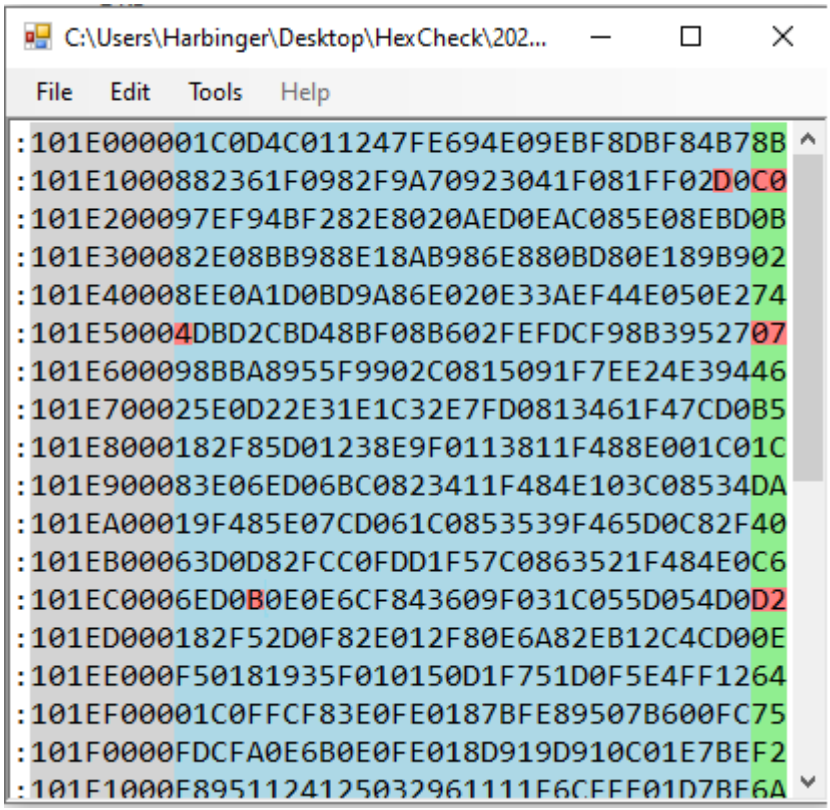

Рисунок 4.7 – Контроль цілісності даних у “FixAVRHexCheckSum”

Завдяки інформації про наявність невідповідності стає можливим детектувати збої у роботі відстежуваного обладнання. У деяких випадках, локалізація і аналіз перетворень даних здатна надати інформацію про місце та характер збою[94], що полегшує процес пошуку несправності.

У випадку, коли прояв носить сталий характер і не має тенденції до поширення свого впливу на систему, програмний засіб може виступати у ролі тимчасового коригувальника надходячих до нього даних[95]. Застосування виправлення даних за допомогою контрольних сум та методів виправлення одиничних спотворень бітів може тимчасово усунути ефекти прояву дефекту. Варіант такого використання наведено на рисунку 4.10.

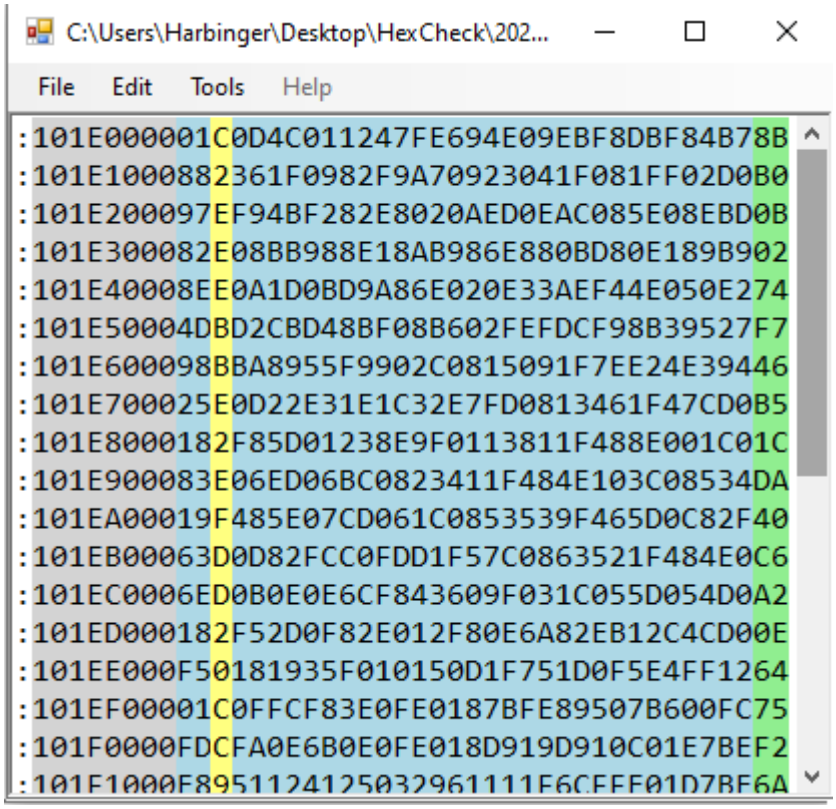

Рисунок 4.8 – Візуалізація коригування даних у “FixAVRHexCheckSum”

В такому випадку, використання програмного засобу дозволяє перевести прояв частини дефектів у формат збою, які можливо усунути програмно, що позитивно впливатиме на показники готовності системи.

4.4 Роботизований засіб для прокладання ліній мережевих комунікацій.

В рамках апробації розроблених програмних засобів до систем БПА, їх було застосовано до системи дистанційного прокладання кабелю. Ця система виконана у вигляді апаратного комплексу з модульною(сегментною) архітектурою. Основою в цьому рішенні виступає сегмент, а саме самостійна структурна одиниця пристрою, що слугує базовим блоком при побудові подібних БПА[96,97]. Сегментованість пристрою дозволяє розширювати та модифікувати його за потреби. Кожен сегмент оснащається набором програмно-апаратних модулів наступного типу:

1) руховий модуль, що відповідає за керування двигунами та корисним навантаженням;

2) керуючий модуль, що містить у собі керуючі плати та компоненти для обміну даними з оператором[98].

Залежно від задачі прокладання кабелю, розміри такого БПА можуть варіюватися[99,100]. Мінімально необхідною комплектацією вважається пристрій з 2х сегментів, що дозволяє пристрою рухатись в умовах обмеженого простору. Однак для більшої стабільності рекомендується використовувати комплектацію з 3х сегментів, представлену на рисунку 4.12.

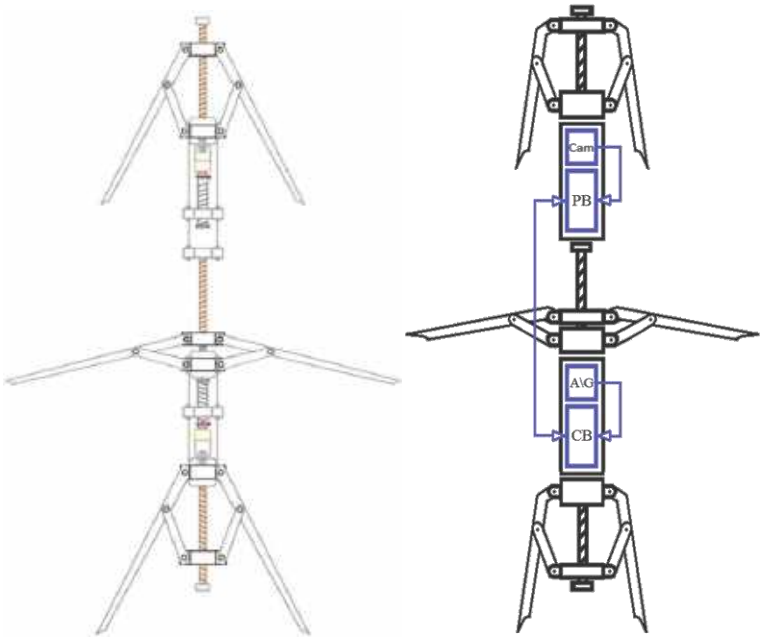

Рисунок 4.9 – Комплектація системи прокладання з 3х сегментів

Відповідно, додавання до системи сегментів буде призводити до дублювання і програмно-апаратних модулів. Завдяки їх однотипності та повторюваності при впровадженні засобів реконфігурації з’являється можливість моніторингу та керування однотипними модулями і їх використання у якості ресурсу реконфігурації для відновлень при збоях.

Крім відомих способів забезпечення надійності, пропонується використовувати можливість реконфігурації технічних систем як засіб часткового відновлення функціональності в разі їхньої відмови. Для цього передбачається виділення сегментів системи та їхня градація за ступенем важливості і групування за схожістю елементної бази[101]. У такому разі виникає можливість використовувати елементи менш важливих сегментів системи як резерв для елементів важливіших сегментів.

4.5 Аналіз результатів впровадження розроблених методів та засобів в програмовні пристрої безпілотних апаратів та роботизованих систем

Отримані методи та засоби були використані при виконанні міжнародного проєкту ERASMUS+ ALIOT, а також проєктів кафедри, виконаних на замовлення МОН України, і навчальному процесі – при підготовці лекційних матеріалів, матеріалів для практикумів, при обґрунтуванні та формуванні вимог до надійності та живучості обладнання безпілотних систем, а також варіантів програмно-апаратних засобів їхнього забезпечення.

Розроблені автором наукові положення впроваджено у:

- навчальному процесі кафедри комп'ютерних систем, мереж і кібербезпеки ХАІ (лекціях та практичних заняттях з навчальних дисциплін «Надійність та функціональна безпека ІКС», «Програмування систем IoT», «Надійність та відмовостійкість КС», «Комплексні системи захисту інформації») для бакалаврів, магістрів і аспірантів, що навчаються за спеціальностями 123 – Комп’ютерна інженерія та 125 – Кібербезпека і захист інформації;

- науково-дослідницьких роботах кафедри, виконаних на замовлення МОН України, зокрема: «Розроблення науково-методичного апарату і засобів для оцінювання та забезпечення надійності та безпечності гарантоздатних ІТ-систем та інфраструктур» (№ ДР 0121U113842, 2021–2023 рр.), «Наукові засади і методи забезпечення гарантоздатності флотів БПЛА інтелектуальних систем моніторингу потенційно небезпечних і військових об’єктів» (Д 503-1/2021-Ф, № Д/Р 0121U112172, 2021-2023 рр.), що виконуються на кафедрі комп’ютерних систем, мереж та кібербезпеки.

Впровадження цих результатів надало змогу:

- підвищити показники надійності і живучості програмовних пристрої та безпілотних систем, систем Інтернету речей та інших програмно-апаратних рішень для перспективних застосувань врамках виконання науково-дослідних проєктів. Кількісні оцінки позитивного ефекту за показниками безвідмовності, готовності та живучості надано в розділах 2-3, де ілюструються рівні їх зростання залежно від значень параметрів ПП, засобів зовнішньої реконфігурації безпілотних систем;

- покращити наочність, фундаментальність та практичну спрямованість навчальних курсів, які викладаються на кафедрі та університеті, а також курсів розроблених в рамках міжнародних проєктів.

4.6 Висновки до четвертого розділу

На підставі викладеного матеріалу, аналізу розроблених засобів та впроваджень, можна констатувати наступне.

1. Визначено загальну послідовність використання запропонованих моделей, методів і засобів для оцінювання показників надійності і живучості, аналізу відповідності ПП вимогам, а також розроблення програмно-апаратних засобів реконфігуровних програмовних пристроїв безпілотних систем.

2. Для програмного засобу забезпечення реконфігуропридатності було описано його структуру та суттєві його компоненти, а саме:

- модуль *OptiBoot.c;*

- блок *boot_opt.h;*

- блок *Common_Def.h;*

- блок *pin_defs.h.*

Функційне призначеня кожного блоку було описано у вигляді переліку базових функцій засобу, а саме налаштувань розмірів і формату програмних блоків та частот, що використовуються при обміні даними.

Для програмного засобу було описано два приклади застосування, демонструючи можливості виділення та керування додатковими ресурсами при реконфігураціях.

2. Для програмного засобу контролю та виправлення контрольних сум було описано його структуру та суттєві його компоненти, а саме:

- модуль *MainForm.cs;*

- модуль *AvrHex.cs;*

- підмодуль *AvrHexLine.cs.*

Функційне призначеня кожного модуля було описано і враховано під час переліку базових функцій засобу, а саме інструментів взаємодії з файловою системою, інструментів роботи зі змістом файлу, інструмент автоматизованого оброблення змісту файлу.

Для програмного засобу було описано два приклади застосування, демонструючи можливості виявлення та часткового усунення наслідків прояву фізичних дефектів програмовних пристроїв.

Для програмного засобу було описано два приклади застосування, демонструючи можливості виділення та керування додатковими ресурсами при реконфігураціях.

3. Для роботизованого засобу прокладання ліній комунікації було описано його структуру і змістовні модулі, а саме:

- руховий модуль;

- керуючий модуль.

Була описана здатність системи до масштабування, переваги котрого були також описані. Функційне призначеня описаних модулів передбачає наявність дублікатів виконавчих пристроїв що було розглянуто з точки зору потенційної реконфігуропридатності для надійнісного та функційного покращення характеристик системи.

4. Аналіз результатів впровадження розроблених методів та засобів в програмовні пристрої БПА та роботизованих систем продемонстрував практичну значущість та підтвердив новизну результатів дослідження, їх залучення до міжнародних та кафедральних проєктів.

ВИСНОВКИ

1. У дисертаційній роботі розв’язано актуальну науково-прикладну задачу і розроблено моделі, методи і програмні засоби оцінювання та забезпечення надійності програмовних пристроїв безпілотних апаратів в умовах накопичення відмов і багаторівневої деградації внутрішніх ресурсів. Одержано низку нових наукових результатів.

2. Вперше запропоновано метод керованої багаторівневої деградації програмовних пристроїв, який, на відміну від відомих, базується, по-перше, на формальному описі умов та аналізі реконфігуропридатності архітектури за допомогою спеціальних метрик об’єму та часу; по-друге, запропонованих процедурах реконфігурації, вибір і реалізація яких здійснюється залежно від типів дефектів з використанням визначеної множини конфігурацій, що забезпечує підвищення надійності та живучості при виникненні та накопиченні відмов апаратних компонентів.

3. Удосконалено структурні та надійнісні моделі програмовних пристроїв з керованою багаторівневою деградацією шляхом декомпозиції множин внутрішніх компонентів залежно від впливу на працездатність та потенційної можливості програмно-керованої реконфігурації структури без втрати або з частковою втратою працездатності і якості виконання специфікованих функцій, що надає змогу формувати вимоги до засобів керування реконфігурацією та розраховувати показники безвідмовності та живучості при певних відмовах.

4. Удосконалено марковські моделі готовності програмовних пристроїв з керованою деградацією, що враховують інтенсивності відмов множин компонентів, визначених залежно від можливості відновлення завдяки програмно-керованій реконфігурації без (або) з частковою втратою якості виконання функцій, часові і достовірнісні характеристики процедур перебудови, а також наявність резервування та часткову працездатність засобів реконфігурації, що забезпечує можливість розрахунку функцій готовності при багатоступеневій деградації.

5. Практичні результати полягають у доведенні теоретичних положень дисертаційної роботи до конкретних методів та програмних засобів для підтримки керованої БРД в БПА на основі ПП. Зокрема розроблено:

- програмний засіб для забезпечення реконфігуропридатності;
- програмний засіб для виправлення контрольних сум;
- апаратний засіб (платформа) для рою роботів з підтримкою функціональної реконфігуропридатності.

6. Запропоновані нові наукові та практичні результати забезпечують виконання вимог та підвищення показників надійності і живучості ПП БПА. Кількісні оцінки відносного і абсолютного їх підвищення визначаються типами ПП, їх показниками безвідмовності, можливостями повного або часткового впровадження запропонованих процедур реконфігурації з урахуванням обмежень обладнання та програмного забезпечення безпілотних систем.

7. Результати дисертаційної роботи впроваджено:

- у навчальному процесі Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут» (акт впровадження від 07 березня 2025 року);

- при виконанні чотирьох науково-дослідних проєктів, що виконувались у Національному аерокосмічному університеті ім. М. Є. Жуковського «Харківський авіаційний інститут» (акт впровадження від 07 березня 2025 року).

7. Подальші дослідження доцільно зосередити на:

- розробленні методів запобігання поширення відмов при виникненні кластерних та ланцюгових відмов;

- вдосконаленні механізму формування процедур реконфігурації шляхом розширення кількості враховуваних ознак програмовних систем;

- поглибленому дослідженні впливу засобів контролю та реконфігурації на працездатність програмовного пристрою БПА.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вдовіченко, О., Перепелицин, А., Дужий, В., Желтухін, О. Метод дистанційної діагностики, перепрограмування і реконфігурації вузлів вбудованої системи. *Авіаційно-космічна техніка і технологія*. – 2022. – No. 6(184). – С. 66–75. DOI: 10.32620/aktt.2022.6.08.

2. Perepelitsyn, A., Duzhyi, V., Vdovichenko, O., Zheltukhin, O. Technologies of Embedded Systems Prototyping using Reconfigurable Nodes: Technical Solutions. *2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece, 2022, P. 1–6. DOI: 10.1109/DESSERT58054.2022.10018581.

3. Вдовіченко, О., Перепелицин, А. Організація взаємодії пристроїв з доступом в інтернет на основі мікроконтролерів із обмеженою кількістю ресурсів. *Авіаційно-космічна техніка і технологія*. – 2023. – No. 6(192) – С. 76–85. DOI: 10.32620/aktt.2023.6.09.

4. Perepelitsyn, A., Vdovichenko, O. Technologies and Services of Communication for Embedded Systems over Internet. *2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece, 2023, P. 1–6. DOI: 10.1109/DESSERT61349.2023.10416486.

5. Perepelitsyn, A., Vdovichenko, O., Mikhalevskyi, V. Service for communication of devices with Internet access: analysis of technologies and method of creation. *Radioelectronic and Computer Systems.* – 2023. – No. 4(108). – P. 197–208. DOI: 10.32620/reks.2023.4.14.

6. Вдовіченко О. О. Засоби автоматизації розроблення програмного забезпечення [Текст] / Вдовіченко О. О.// *14 Студентська науково-технічна конференція Перспективні мережні та комп'ютерні технології (ПерСиК 2023)*. 2023, С. 50. ISBN 978-617-8238-34-6.

7. Vdovichenko, O., Perepelitsyn, A. Analysis of Technologies for Reconfiguration of IoT Systems at Level of Software Modules and Bootloaders. *Integrated Computer Technologies in Mechanical Engineering* – 2023. ICTM 2023.

Lecture Notes in Networks and Systems, Vol 996. Springer, Cham, Germany, 2023, P 474–486. DOI: 10.1007/978-3-031-60549-9_36.

8. Вдовіченко, О. Аналіз технологій реконфігурації систем інтернету речей на рівні програмних модулів та завантажувачів. *Авіаційно-космічна техніка і технологія.* – 2024. – No. 3(195) – С. 99–108. DOI: 10.32620/aktt.2024.3.09.

9. Вдовіченко, О., Перепелицин, А. Аналіз номенклатури мікроконтролерів для побудови елементів домашньої автоматизації. *Авіаційно-космічна техніка і технологія*. – 2024. – No. 5(199) – С. 118–126. DOI: 10.32620/aktt.2024.5.12.

10. Вдовіченко, О., Харченко, В. Марковські моделі оцінювання надійності програмовних пристроїв з керованою багаторівневою деградацією: класифікація, розроблення і дослідження. *Measuring and computing devices in technological processes.* – 2024 – No.4, – С. 433–444. DOI: 10.31891/2219-9365-2024-80-54.

11. Вдовіченко, О., Харченко, В. Програмовні пристрої з керованою багаторівневою деградацією: моделі, методи реконфігурації та аналізу реконфігуропридатності. *Електронне моделювання*. – 2025 – No. 47(1) – С. 53–76, DOI: 10.15407/emodel.47.01.053.

12. Vdovichenko, O., Perepelitsyn, A. Technologies for building systems of remote lining of communication lines: a practical example of implementation. *Radioelectronic and Computer Systems.* – 2021. – No. 2(98). – P. 31–38. DOI: 10.32620/reks.2021.2.03.

13. Meng, L., Hirayama, T., Oyanagi, S. Underwater-Drone With Panoramic Camera for Automatic Fish Recognition Based on Deep Learning, *IEEE,* Vol. 6, P. 17880-17886, 2018, DOI: 10.1109/ACCESS.2018.2820326.

14. Zhang, H., Yang, X., Gao, J. A Low-cost Hardware Platform of UAV Attitude Estimation Based on Mahony Algorithm, *2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP)*, Xi'an, China, 2023, P. 1267-1271, DOI: 10.1109/ICSP58490.2023.10248709.

15. Shi, X., Zou, A. Preemptive FPGA Scheduling Based on Dynamic Partial Reconfiguration, *2024 Conference of Science and Technology for Integrated Circuits (CSTIC)*, Shanghai, China, 2024, P. 1-3, DOI: 10.1109/CSTIC61820.2024.10531952.

16. Sezer, S., Heron, J., Woods, R. Turner and A. Marshall, Fast partial reconfiguration for FCCMs, *Proceedings. IEEE Symposium on FPGAs for Custom Computing Machines (Cat. No.98TB100251)*, Napa Valley, CA, USA, 1998, P. 318-319, DOI: 10.1109/FPGA.1998.707934.

17. Szasz C., Chindris, V. Self-healing and fault-tolerance abilities development in embryonic systems implemented with FPGA-based hardware, *Proceedings of 2009 IEEE International Conference on Intelligent Engineering Systems*, 2009, P. 215-220, DOI: 10.1109/INES.2009.4924765.

18. Vishwakarma, S., Upadhyaya, P., Kumari B., Mishra, A. Smart Energy Efficient Home Automation System Using IoT, *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Ghaziabad, India, 2019, P. 1-4, DOI: 10.1109/IoT-SIU.2019.8777607.

19. Doboli, A., Kane P., Van Ess D. Dynamic reconfiguration in a PSoC device, *2009 International Conference on Field-Programmable Technology*, Sydney, NSW, Australia, 2009, P. 361-363, DOI: 10.1109/FPT.2009.5377613.

20. Liu, L. A dynamically reconfigurable SOC, *2008 IEEE International Conference on Industrial Technology*, Chengdu, China, 2008, P. 1-4, DOI: 10.1109/ICIT.2008.4608650.

21. Nozal, L., Oliveira, R., Lorenzo S., Mohamed, M. A new vision system: programmable logic devices and digital signal processor architecture (PLD+DSP), *Proceedings IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation*, Kobe, Japan, 1991, P. 2014-2018 Vol.3, DOI: 10.1109/IECON.1991.239032.

22. Li C., Wang W., Yang M., Wang J., Ma T., Wang D. Research on the Classification Method of Reliability Critical Parts and Important Parts. *2022 4th International Conference on System Reliability and Safety Engineering (SRSE)*. 2022; P. 1-6, Guangzhou, China, DOI: 10.1109/SRSE56746.2022.10067334.

23. Deng R., Deng C., Wang R., Gong W., Cai Z., Jiang K. Research on Reliability Life Evaluation Method Based on Airborne T/ R Components. *2021 22nd International Conference on Electronic Packaging Technology (ICEPT)*. 2021; P. 1-4, Guangzhou, China, DOI: 10.1109/ICEPT52650.2021.9568016.

24. Marchand, J. An alterable programmable logic array, *IEEE Journal of Solid-State Circuits*, Vol. 20, No. 5, P. 1061-1066, Oct. 1985, DOI: 10.1109/JSSC.1985.1052437.

25. Levashenko, V., Lukyanchuk, I., Zaitseva, E., Kvassay, M., Rabcan J., Rusnak, P. Development of Programmable Logic Array for Multiple-Valued Logic Functions, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 39, No. 12, P. 4854-4866, Dec. 2020, DOI: 10.1109/TCAD.2020.2966676.

26. Tian W., Zha W., Lei H., Yang X. Research and Analysis of Reliability Evaluation Methods for Automotive Electronic Components. *2024 25th International Conference on Electronic Packaging Technology (ICEPT)*. 2024; P. 1 - 4, Tianjin, China, DOI: 10.1109/ICEPT63120.2024.10668438.

27. Федасюк Д. В., Волочій С. Б. Структурно-автоматна модель відмовостійких систем для автоматизації використання методу фаз Ерланга. *Радіоелектронні і комп'ютерні системи*. 2016; No. 3 (77), С. 78 - 92, DOI: 10.32620/reks.2016.3.10.

28. Одарущенко О. М., Одарущенко О.Б., Харченко В. С. Марковські моделі оцінювання функціональної безпеки програмно-технічних комплексів на самодіагностовних програмовних платформах з урахуванням помилок засобів контролю. *Радіоелектронні і комп'ютерні системи*. 2019; С. 15 - 29, DOI: 10.32620/reks.2019.4.02.

29. Quick Guide to Microchip Development Tools [Online]. – *Microchip Technology Inc.*, 2016. – 36 p. – Available at: http://ww1.microchip.com/downloads/ en/ DeviceDoc/50001894E.pdf. – 19.07.2024.

30. Lasiuk; Z., Verma; P., Andrews, J. The Insider's Guide to Arm Cortex-M Development: Leverage embedded software development tools and examples to become an efficient Cortex-M developer , *Packt Publishing*, 2022.

31. Wu, S., Abdulghani, A., Ansari, S., Imran, M., Abbasi, Q. IoT enabled Smart Lighting System using STM32 microcontroller with high performance ARM® Cortex®-M3 core, *2020 International Conference on UK-China Emerging Technologies (UCET)*, Glasgow, UK, 2020, P. 1-4, DOI: 10.1109/UCET51115.2020.9205363.

32. TM4C Microcontrollers Product Selection Guide [Online]. – *Texas Instruments*, 2021. – 12 p. – Available at: https://www.ti.com/lit/sg/spmt285e/spmt285e.pdf?ts=1729420360166. – 23.07.2024.

33. Lison, A. Hardware Standardization and State-Socialist Piracy: The Global Reach of the Zilog Z80, *IEEE Annals of the History of Computing*, Vol. 46, No. 2, P. 38-51, April-June 2024, DOI: 10.1109/MAHC.2024.3384915.

34. Van Singel, S. The Renesas Automotive Story in the History of the Microprocessor, *IEEE Micro*, Vol. 41, No. 6, P. 107-108, 1 Nov.-Dec. 2021, DOI: 10.1109/MM.2021.3113821.

35. Maier, A., Sharp A., Vagapov, Y. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things, *2017 Internet Technologies and Applications (ITA)*, Wrexham, UK, 2017, P. 143-148, DOI: 10.1109/ITECHA.2017.8101926.

36. AN136: Silicon Labs Production Programming Options [Online]. – *Silicon Labs*, 2021. – 12 p. – Available at: https://www.silabs.com/documents/public/application-notes/AN136-production-programming-options.pdf – 25.07.2024.

37. Zaynidinov, H., Ibragimov S., Tojiboyev, G. Comparative Analysis of the Architecture of Dual-Core Blackfin Digital Signal Processors, *2021 International Conference on Information Science and Communications Technologies (ICISCT)*, Tashkent, Uzbekistan, 2021, P. 1-4, DOI: 10.1109/ICISCT52966.2021.9670135.

38. Chen, W., Huang, C., Chen, K., Chang, L,. Tsai, M. Design and Realization of Microcontroller-Based Remote Status Monitoring System for Smart

Factory Task Planning Applications, *2021 60th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Tokyo, Japan, 2021, P. 1500-1505.

39. Li, X. Design and Verification of MCU Chip Bootloader, *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China, 2020, P. 395-402, DOI: 10.1109/ITAIC49862.2020.9339147.

40. Neyestanak, M., Azizi E., Salarpour, E. IoT Performance Improving for Indoor and Outdoor Environments, *2021 5th International Conference on Internet of Things and Applications (IoT)*, Isfahan, Iran, 2021, P. 1-6, DOI: 10.1109/IoT52625.2021.9469604.

41. Elmenreich, W., Rosenblattl, M., Wolf, A. Fixed Point Library Based on ISO/IEC Standard DTR 18037 for Atmel AVR Microcontrollers, *2007 Fifth Workshop on Intelligent Solutions in Embedded Systems*, Leganes, Spain, 2007, P. 101-113, DOI: 10.1109/WISES.2007.4408492.

42. Mischie S., Pazsitka, R. Designing a MSP430 Bootloader, *2019 International Conference on Applied Electronics (AE)*, Pilsen, Czech Republic, 2019, P. 1-4, DOI: 10.23919/AE.2019.8866999.

43. Ali, D., Ridha, E., Mohamed, N. Assessing the Efficacy of TinyML Implementations on STM32 Microcontrollers: A Performance Evaluation Study, *2024 IEEE 7th International Conference on Advanced Technologies, Signal and Image Processing (ATSIP)*, Sousse, Tunisia, 2024, P. 267-271, DOI: 10.1109/ATSIP62566.2024.10638900.

44. Chen He, P., Kuhn, T., Niset, M. Reliability model and implementation for EEPROM emulation using flash memory, *2004 IEEE International Reliability Physics Symposium. Proceedings, Phoenix*, AZ, USA, 2004, P. 657-658, DOI: 10.1109/RELPHY.2004.1315437.

45. Shen, Y. Application of STM32 Microcontroller Control System in Automation of Computer Laboratory Equipment, *2024 International Conference on Intelligent Computing and Robotics (ICICR)*, Dalian, China, 2024, P. 169-175, DOI: 10.1109/ICICR61203.2024.00039.

46. Gu, H., Zhang L., Yin, X. Design of Intelligent Low-power Micro Vibration Gating System, *2023 IEEE 6th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC)*, Chongqing, China, 2023, P. 1023-1027, DOI: 10.1109/ITNEC56291.2023.10082230.

47. Jacko, P., Kováč, D., Bučko, R., Vince, T., Kravets, O. The parallel data processing by nucleo board with STM32 microcontrollers, *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, Kremenchuk, Ukraine, 2017, P. 264-267, DOI: 10.1109/MEES.2017.8248906.

48. Gaiba, F., Bedogni, L., Gori, G., Melis, A., Prandini, M. Wi-Fi Sensing for Human Identification Through ESP32 Devices: An Experimental Study, *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2024, P. 206-209, DOI: 10.1109/CCNC51664.2024.10454655.

49. Mituletu, I., Muresan, V. Wireless Communication System with High Data Flow using an ESP32-Based Interface, *2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES)*, Veliko Tarnovo, Bulgaria, 2024, P. 1-6, DOI: 10.1109/CIEES62939.2024.10811225.

50. Wanner, D., Hashim, A., Srivastava, S., Steinhauer, A. UAV avionics safety, certification, accidents, redundancy, integrity, and reliability: a comprehensive review and future trends. *Drone Systems and Applications*. 2024. No.12: P. 1-23. DOI:10.1139/dsa-2023-0091.

51. Petritoli, E, Leccese, F, Ciani, L. Reliability and Maintenance Analysis of Unmanned Aerial Vehicles. *Sensors*. 2018. No.18(9):3171. DOI:10.3390/s18093171.

52. Perepelitsyn, A., Tetskyi, A. Method of creation of power sources for home appliances under constraints of limited resources. *Radioelectronic and Computer Systems*, 2023. No. 2 (106). P. 81-93. URL: DOI:10.32620/reks.2023.2.07.

53. Zaitseva, E. Reliability Assessment of UAV Fleets. In: Klymash, M., Luntovskyy, A., Beshley, M., Melnyk, I., Schill, A. (eds) Emerging Networking in the Digital Transformation Age. TCSET 2022. *Lecture Notes in Electrical Engineering*, 2023, Vol 965. Springer, Cham. DOI:10.1007/978-3-031-24963-1_19.

54. Landman, E. Degradation Monitoring - from a Vision to Reality, *2019 IEEE International Reliability Physics Symposium (IRPS)*, Monterey, CA, USA, 2019, P. 1-4, DOI: 10.1109/IRPS.2019.8720527.

55. Terenyk, D., Kharchenko, V. Choosing strategies for deployment and ensuring the reliability of a UAV swarm to support communications in destruction conditions. *Innovative technologies and scientific solutions for industries*, 2024, No.3 (29), P. 91–103. DOI:10.30837/2522-9818.2024.3.091.

56. Aghaei, M. Redundancy allocation problem fork-out-of-n systems with a choice of redundancystrategies [Text] / M. Aghaei, A. Z. Hamadani, M. A.Ardakan // *Journal of Industrial Engineering International*. – 2017. – Vol. 13, No. 1. – P. 81-92. DOI:10.1007/s40092-016-0169-3.

57. Pascual-Ortigosa, P. Algebraic analysis ofmultistate k-out-of-n systems [Text] / P. Pascual-Ortigosa, E. Sáenz-De-Cabezón, H. P. Wynn //*Monografías de la Real Academia de Ciencias*. – 2018.– Vol. 43. – P. 131-134.

58. Liscouët, J., Pollet, F., Jézégou, J., Budinger, M., Delbecq, S., Moschetta, J. A methodology to integrate reliability into the conceptual design of safety-critical multirotor unmanned aerial vehicles, *Aerospace Science and Technology*, 2022, Vol. 127, DOI:10.1016/j.ast.2022.107681.

59. Petritoli, E, Leccese, F, Ciani, L. Reliability and Maintenance Analysis of Unmanned Aerial Vehicles. *Sensors (Basel)*. 2018 Vol. 19;18(9):3171. DOI: 10.3390/s18093171.

60. Lee K., Kim, Y. Design and Analysis of Digital PID Controller in MCU and FPGA, *2018 International SoC Design Conference (ISOCC), Daegu, Korea (South)*, 2018, P. 261-262, DOI: 10.1109/ISOCC.2018.8649909.

61. Одарущенко, О. М., Одарущенко, О.Б., Харченко, В. С., Марковські моделі оцінювання функціональної безпеки програмно-технічних комплексів на самодіагностовних програмовних платформах з урахуванням помилок засобів контролю. *Радіоелектронні і комп'ютерні системи*. 2019; С. 15 - 29, DOI: 10.32620/reks.2019.4.02.

62. Ozirkovskyy, L., Volochiy, B., Shkiliuk, O., Zmysnyi, M., Kazan, P. Functional safety analysis of safety-critical system using state transition diagram. *Radioelectronic and computer systems*. 2022; P. 145 - 158, DOI:10.32620/reks.2022.2.12.

63. Федасюк, Д. В., Волочій, С. Б. Структурно-автоматна модель відмовостійких систем для автоматизації використання методу фаз Ерланга. *Радіоелектронні і комп'ютерні системи*. 2016; No. 3 (77), С. 78 - 92, DOI: 10.32620/reks.2016.3.10.

64. Dong, Q. Impact of Self-healing Control on Reliability Evaluation in Distribution System with Microgrid, *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, Espoo, Finland, 2021, P. 1-5, DOI: 10.1109/ISGTEurope52324.2021.9640207.

65. Bhardwaj, N., Liggesmeyer, P. A Conceptual Framework for Safe Reconfiguration in Open System of Systems, *2018 IEEE/ACM 6th International Workshop on Software Engineering for Systems-of-Systems (SESoS)*, Gothenburg, Sweden, 2018, P. 17-20.

66. Tahiri, I, Philippot, A, Carré-Ménétrier, V, Tajer, A. A Fault-Tolerant and a Reconfigurable Control Framework: Application to a Real Manufacturing System. *Processes*. 2022; 10(7):1266. DOI:10.3390/pr10071266.

67. Zaitseva, E., Levashenko, V., Rabcan, J. A new method for analysis of Multi-State systems based on Multi-valued decision diagram under epistemic uncertainty. *Reliability Engineering & System Safety*, 2023, Vol. 229., DOI:10.1016/j.ress.2022.108868.

68. Pan G., Li D., Li Q., Huang C., Mo B. A Reliability Evaluation Method for Multi-performance Degradation Products Based on Accelerated Degradation Testing. *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. 2022; P. 1871 – 1875, Chongqing, China, DOI: 10.1109/ITAIC54216.2022.9836924.

69. Lisnianski, A., Levitin, G., Multi-State System Reliability. *World Scientific Publishing Company*. 2003; Available at: https://www.perlego.com/book/853040/

multistate-system-reliability-assessment-optimization-and-applications-pdf (Accessed: 14 October 2022).

70. Харченко, В. С., Зайцева, Е. О деградирующих системах с деградирующими компонентами. *Радіоелектронні і комп'ютерні системи*. 2010. No. 7. С. 288–292. Режим доступу: http://nbuv.gov.ua/UJRN/recs_2010_7_58.

71. Харченко, В. С. О системах з багатоступеневою деградацією та відновленням. *Інформаційно-керуючі системи на залізничному транспорті*. 1997. No. 1. С. 3–9.

72. Brezhnev, E., Fesenko, H., Kharchenko, V., Levashenko, V., Zaitseva, E. MSS Models of Smart Grids with Multi-level Degradation and Recovery. In: Kharchenko, V., Kondratenko, Y., Kacprzyk, J. (eds) *Green IT Engineering: Concepts, Models, Complex Systems Architectures. Studies in Systems, Decision and Control*. Springer. Cham. 2017. Vol 74. P.339-355. DOI: 10.1007/978-3-319-44162-7_11.

73. Lisnianski, A., Levitin, G. Multi-State System Reliability Assessment, Optimization and Applicationshttps. Series on Quality, *Reliability and Engineering Statistics*. 2003. Vol 6, DOI: 10.1142/5221.

74. Vdovichenko, O., Kharchenko, V., Perepelitsyn, A. Fault-tolerant Microcontroller Chip with Controlled Degradation: Models of Failures, Bootloader-based Reconfiguration and Dependability Assessment. *2024 14th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. Athens. Greece. 2024. P. 1-7.

75. Ahmed, A. I., Said, L. A., Madian, A. H. Over-The-Air Firmware Updating Model suitable for Industrial IoT based on Microchip AVR MCU. *2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)*, Arad, Romania. 2021. P. 492-496, DOI: 10.1109/ICCCA52192.2021.9666279.

76. Simhandl, G., Paulweber, P., Zdun, U. Developer's Cognitive Effort Maintaining Monoliths vs. Microservices - An Eye-Tracking Study, *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, Seoul, Korea, Republic of, 2023, P. 339-348, DOI: 10.1109/APSEC60848.2023.00044.

77. Rath, A., Roy, D., Teja, D., Kumar, G. Embed-ded Hardware Testing Using Bootloader, *International Conference on Smart Electronics and Communication, ICOSEC 2020*, 2020, P. 1–6, DOI:10.1109/ICOSEC49089.2020.9215327.

78. Sebastian, A., Sankar, S. Design of a Dynamic Boot Loader for Loading an Operating System, *Journal of Computer Science*, Vol. 15, No. 1, 2019, P. 190-196, DOI: 10.3844/jcssp. 2019.190.196.

79. Introduction to file extensions [Online]. – *TechTarget*, 2025. – 3 p. – Available at: https://www.techtarget.com/whatis/definition/file-format. – 20.02.2025.

80. Bancila, M. Modern C++ Programming Cookbook: Master modern C++ including the latest features of C++23 with 140+ practical recipes , *Packt Publishing*, 2024.

81. Modifying Optiboot to Upload Sketches over RS485 [Online]. – *Arduino Forum*, 2025. – 14 p. – Available at: https://forum.arduino.cc/t/modifying-optiboot-to-upload-sketches-over-rs485/77706. – 22.02.2025.

82. Li, Y., Wang, T., Zhao, Z., Peng, M., Wang, W. Relay Mode Selection and Power Allocation for Hybrid One-Way/Two-Way Half-Duplex/Full-Duplex Relaying, *IEEE Communications Letters*, Vol. 19, No. 7, P. 1217-1220, July 2015, DOI: 10.1109/LCOMM.2015.2433260.

83. Upload code over one wire: custom programmer and bootloader [Online]. – *Arduino Forum*, 2025. – 9 p. – Available at: https://forum.arduino.cc/t/upload-code-over-one-wire-custom-programmer-and-bootloader/600649. – 22.02.2025.

84. Kulkarni, G., Jadhawar, B. Dual Microcontroller Redundancy System for Critical Applications, *2013 International Conference on Machine Intelligence and Research Advancement*, Katra, India, 2013, P. 353-355, DOI: 10.1109/ICMIRA.2013.74.

85. Upload firmware over half duplex RS-485 [Online]. – *Arduino Forum*, 2025. – 8 p. – Available at: https://forum.arduino.cc/t/upload-firmware-over-half-duplex-rs-485/183031. – 22.02.2025.

86. Bidirectional communication from RS-232 to half-duplex RS-485 using arduino nano [Online]. – *Arduino Forum*, 2025. – 7 p. – Available at:

https://forum.arduino.cc/t/bidirectional-communication-from-rs-232-to-half-duplex-rs-485-using-arduino-nano/1232187. – 22.02.2025.

87. Serial output on TX pin [Online]. – *Arduino Forum*, 2025. – 7 p. – Available at: https://forum.arduino.cc/t/serial-output-on-tx-pin/1220190. – 22.02.2025.

88. Remote sketch upload by rs485. Possible? [Online]. – *Arduino Forum*, 2025. – 20 p. – Available at: https://forum.arduino.cc/t/remote-sketch-upload-by-rs485-possible/318038 – 22.02.2025.

89. List of File Formats with Types and Extensions [Online]. – *Geeksforgeeks Tutorials*, 2025. – 11 p. – Available at: https://www.geeksforgeeks.org/list-of-file-formats/ – 12.02.2025.

90. Saxena, N., McCluskey, E. Analysis of checksums, extended-precision checksums, and cyclic redundancy checks, *IEEE Transactions on Computers*, Vol. 39, No. 7, P. 969-975, July 1990, DOI: 10.1109/12.55701.

91. Cortesi, A., Olliaro, M. M-String Segmentation: A Refined Abstract Domain for String Analysis in C Programs, *2018 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, Guangzhou, China, 2018, P. 1-8, DOI: 10.1109/TASE.2018.00009.

92. Lagin, L., Medley, S., Bergin, W. The TFTR Charge Exchange Diagnostic Software Systems, *IEEE Transactions on Nuclear Science*, Vol. 32, No. 4, P. 1268-1271, Aug. 1985, DOI: 10.1109/TNS.1985.4333589.

93. Huang, W., Xu, H., Wang, J., Miao, C., Ren Y., Wang, L. Redundancy Management for Fault-tolerant Control System of an Unmanned Underwater Vehicle, *2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, Dalian, China, 2020, P. 291-296, DOI: 10.1109/CACRE50138.2020.9230038.

94. Bustamante, L. Al-Asaad, H. Detection of soft errors through checksums in redundant execution systems, *2015 IEEE AUTOTESTCON*, National Harbor, MD, USA, 2015, P. 134-137, DOI: 10.1109/AUTEST.2015.7356479.

95. Saxena, N., McCluskey, E. Control-flow checking using watchdog assists and extended-precision checksums, *The Nineteenth International Symposium on Fault-*

Tolerant Computing. Digest of Papers, Chicago, IL, USA, 1989, P. 428-435, DOI: 10.1109/FTCS.1989.105615.

96. Shi, X., Wu, J., Xiong, J., Cheng, P., Ding, L., Lin, H. Research on Remote Resource Cooperation Strategy Based on Modular UAVs, *2024 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, Toronto, ON, Canada, 2024, P. 1-6, DOI: 10.1109/BMSB62888.2024.10608317.

97. Meshworm [Online]. – *SciTechDaily,* 2024. – 3 p. – Available at: https://scitechdaily.com/meshworm-a-soft-autonomous-robot-that-moves-like-an-earthworm/ – 14.09.2024.

98. Yang, Z., Blanke, M. The robust control mixer module method for control reconfiguration, *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, Chicago, IL, USA, 2000, P. 3407-3411 Vol.5, DOI: 10.1109/ACC.2000.879200.

99. Heilemann, M., Culley, S. Capability audit for modular system development assessing important factors for establishing and maintaining common modular system architectures, *2015 IEEE International Symposium on Systems Engineering (ISSE)*, Rome, Italy, 2015, P. 398-405, DOI: 10.1109/SysEng.2015.7302789.

100. Dias, B., Kulasekera, A., Chathuranga, D., Foresi, G. An evolutionary approach for line of sight relay node placement in a sensor network, *2019 IEEE 23rd International Symposium on Consumer Technologies (ISCT)*, Ancona, Italy, 2019, P. 339-344, DOI: 10.1109/ISCE.2019.8900985.

101. Amaricai, A., Dobre, A., Boncalo, O., Tanase, A., Valuch, C. Models and implementations of hardware interface modules in a multi-processor system-on-chip simulator, *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Timisoara, Romania, 2011, P. 433-437, DOI: 10.1109/SACI.2011.5873042.

ДОДАТОК А
АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

Затверджую
Проректор з науково-педагогічної роботи
Національного аерокосмічного університету
ім. М.Є. Жуковського
«Харківський авіаційний інститут»
к.т.н., доцент
_____ Андрій ГУМЕННИЙ
«07» _____ 2025 року

АКТ ВПРОВАДЖЕННЯ

наукових результатів дисертаційної роботи
Вдовіченка Олександра Олександровича, виконаної на здобуття наукового ступеня доктора філософії, у навчальному процесі кафедри комп'ютерних систем, мереж і кібербезпеки

Комісія у складі: голови комісії - декана факультету радіоелектроніки, комп'ютерних систем та інфокомунікацій к.т.н. Олексія Одокієнка, і членів - професора кафедри комп'ютерних систем, мереж і кібербезпеки, к.т.н. Клайда Фурманова, доцента кафедри комп'ютерних систем, мереж і кібербезпеки, к.т.н. Вячеслава Дужого, доцента кафедри комп'ютерних систем, мереж і кібербезпеки, к.т.н. Дмитра Узуна встановила, що наукові результати, а саме:

– метод реконфігурації програмовних пристроїв безпілотних апаратів з керованою багаторівневою деградацією внутрішніх ресурсів;

– моделі оцінювання готовності програмовних пристроїв з керованою багаторівневою деградацією;

– програмний засіб для виправлення контрольних сум;

– апаратний засіб(платформа) для рою роботів з підтримкою функціоанльної реконфігуропридатності

реалізовані у навчальному процесі кафедри комп'ютерних систем, мереж і кібербезпеки у вигляді:

- лекційного матеріалу і практичних занять з використання інструментальних засобів та методів при аналізі та оцінці програмовних систем, зокрема, системи безпілотних апаратів, оцінювання та вибору процедур реконфігурації для відновлення систем з багаторівневою деградацією відповідно до вимог у навчальних дисциплінах «Надійність та відмовостійкість комп'ютерних систем» (6 годин), «Програмування систем IoT» (4 години), «Надійність та функціональна безпека інформаційно-комунікаційних систем» (4 години), «Системи технічного захисту інформації» (4 години).

Це дозволило покращити фундаментальність викладання матеріалу з надійності сучасних обчислювальних систем, а саме, мобільних систем, інтернету речей, літаючих сенсорних мереж, наочність та практичну спрямованість навчального процесу, якість підготовки фахівців за напрямами комп'ютерних інженерії, кібербезпеки та захисту інформації.

Голова комісії Олексій ОДОКІЄНКО

Члени комісії Клайд ФУРМАНОВ

Вячеслав ДУЖИЙ

Дмитро УЗУН

Затверджую
Проректор з науково-педагогічної роботи
Національного аерокосмічного університету
ім. М.Є. Жуковського
«Харківський авіаційний інститут»
к.т.н., доцент
_____ Андрій ГУМЕННИЙ
«07» березня 2025 року

АКТ ВПРОВАДЖЕННЯ

наукових результатів дисертаційної роботи Вдовіченка Олександра Олександровича,
виконаної на здобуття наукового ступеня доктора філософії,
у науково-дослідних проєктах Національного аерокосмічного університету ім. М. Є. Жуковського «ХАІ»

Комісія у складі: голови – декана факультету радіоелектроніки, комп'ютерних систем та інфокомунікацій к.т.н. Олексія Одокієнка і членів – професора кафедри комп'ютерних систем, мереж і кібербезпеки д.т.н., професора Ольги Морозової, к.т.н., професора кафедри компютерних систем, мереж і кібербезпеки доцента Олександра Орєхова, доцента кафедри компютерних систем, мереж і кібербезпеки к.т.н., Артема Тецького, встановила, що наукові результати, а саме:

– метод реконфігурації програмовних пристроїв безпілотних апаратів з керованою багаторівневою деградацією;

– структурні на надійнісні моделі програмовних пристроїв з керованою багаторівневою деградацією;

– програмний засіб для забезпечення реконфігуропридатності

реалізовані у вигляді наукових положень і розробок, використаних при виконанні науково-дослідних проєктів за замовленням Міністерства освіти та науки України:

– Методологічні засади та технології оцінювання та забезпечення безпеки (захисту) критичних інформаційних інфраструктур (Національний аерокосмічний університет ім. М.Є. Жуковського «Харківський авіаційний інститут», ДР № 0119U100979, 2019–2021);

– Розроблення науково-методичного апарату і засобів для оцінювання та забезпечення надійності та безпечності гарантоздатних ІТ-систем та інфраструктур (Національний аерокосмічний університет ім. М.Є. Жуковського «Харківський авіаційний інститут», ДР № 0121U113842, 2021–2023 рр.;

– Наукові засади і методи забезпечення гарантоздатності флотів БПЛА інтелектуальних систем моніторингу потенційно небезпечних і військових об'єктів (Національний аерокосмічний університет ім. М.Є. Жуковського «Харківський авіаційний інститут», Д 503-1/2021-Ф, Д/Р № 0121U112172, 2021-2023);

– Методи, програмно-апаратні засоби та технології забезпечення гарантоздатності інтелектуальних систем індустріального інтернету речей (Національний аерокосмічний університет ім. М.Є. Жуковського «Харківський авіаційний інститут», Д 503-10/2022-П, Д/Р № 0122U001065, 2022-2023).

Це дозволило підвищити показники кібербезпеки та точності оцінювання кіберризиків для безпілотних мобільних і стаціонарних систем та критичних інфраструктур, які досліджувалися в рамках виконання НДР впродовж 2019-2023 рр.

Голова комісії Олексій ОДОКІЄНКО

Члени комісії Ольга МОРОЗОВА

Олександр ОРЄХОВ

Артем ТЕЦЬКИЙ

ДОДАТОК Б

ЛІСТИНГИ КОДІВ ПРОГРАМНИХ ЗАСОБІВ

Б.1 Лістинг коду програмного засобу “Optiboot”

```
/**********************************************************/
/*           Optiboot bootloader for Arduino              */
/*     На основі OptiBoot з підтримкою програмування      */
/*              Однією лінією з емуляцією UART            */
/**********************************************************/
#define FUNC_READ 1
#define FUNC_WRITE 1
#if !defined(OPTIBOOT_CUSTOMVER)
    #define OPTIBOOT_CUSTOMVER 0
#endif
#include <inttypes.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
typedef union {
    uint8_t  *bptr;
    uint16_t *wptr;
    uint16_t word;
    uint8_t bytes[2];
} addr16_t;
#include "boot_opt.h"
#include "pin_defs.h"
#include "stk500.h"

#ifndef LED_START_FLASHES
    #define LED_START_FLASHES 0
#endif
/* базові швидкості передачі даних UART */
#ifndef BAUD_RATE
    #if F_CPU >= 8000000L
        #define BAUD_RATE   115200L // Підтримується win32
    #elif F_CPU >= 1000000L
        #define BAUD_RATE   9600L
    #elif F_CPU >= 128000L
        #define BAUD_RATE   4800L
    #else
        #define BAUD_RATE 1200L     // Підходить і для 32768Hz
    #endif
#endif
#ifndef UART
    #define UART 0
#endif
#ifndef SOFT_UART
    #ifdef SINGLESPEED
        #define BAUD_SETTING (( (F_CPU + BAUD_RATE * 8L) / ((BAUD_RATE * 16L))) - 1 )
        #define BAUD_ACTUAL (F_CPU/(16 * ((BAUD_SETTING)+1)))
    #else
        #define BAUD_SETTING (( (F_CPU + BAUD_RATE * 4L) / ((BAUD_RATE * 8L))) - 1 )
        #define BAUD_ACTUAL (F_CPU/(8 * ((BAUD_SETTING)+1)))
    #endif
    #if BAUD_ACTUAL <= BAUD_RATE
    #define BAUD_ERROR (( 100*(BAUD_RATE - BAUD_ACTUAL) ) / BAUD_RATE)
    #if BAUD_ERROR >= 5
        #error BAUD_RATE off by greater than -5%
```

```
    #elif BAUD_ERROR >= 2  && !defined(PRODUCTION)
        #warning BAUD_RATE off by greater than -2%
    #endif
    #else
    #define BAUD_ERROR (( 100*(BAUD_ACTUAL - BAUD_RATE) ) / BAUD_RATE)
    #if BAUD_ERROR >= 5
        #error BAUD_RATE off by greater than 5%
    #elif BAUD_ERROR >= 2  && !defined(PRODUCTION)
        #warning BAUD_RATE off by greater than 2%
  #endif
#endif
#if BAUD_SETTING > 250
    #error Unachievable baud rate (too slow) BAUD_RATE
#endif
#if (BAUD_SETTING - 1) < 3
    #if BAUD_ERROR != 0 // permit high bitrates (ie 1Mbps@16MHz) if error is zero
        #error Unachievable baud rate (too fast) BAUD_RATE
    #endif
#endif
/* Watchdog */
#define WATCHDOG_OFF    (0)
#define WATCHDOG_16MS   (_BV(WDE))
#define WATCHDOG_32MS   (_BV(WDP0) | _BV(WDE))
#define WATCHDOG_64MS   (_BV(WDP1) | _BV(WDE))
#define WATCHDOG_125MS  (_BV(WDP1) | _BV(WDP0) | _BV(WDE))
#define WATCHDOG_250MS  (_BV(WDP2) | _BV(WDE))
#define WATCHDOG_500MS  (_BV(WDP2) | _BV(WDP0) | _BV(WDE))
#define WATCHDOG_1S     (_BV(WDP2) | _BV(WDP1) | _BV(WDE))
#define WATCHDOG_2S     (_BV(WDP2) | _BV(WDP1) | _BV(WDP0) | _BV(WDE))
#ifndef __AVR_ATmega8__
#define WATCHDOG_4S     (_BV(WDP3) | _BV(WDE))
#define WATCHDOG_8S     (_BV(WDP3) | _BV(WDP0) | _BV(WDE))
#endif
/* Альтернативний розмір сторінок */
#if SPM_PAGESIZE > 255
    typedef uint16_t pagelen_t ;
    #define GETLENGTH(len) len = getch()<<8; len |= getch()
#else
    typedef uint8_t pagelen_t;
    #define GETLENGTH(len) (void) getch(); len = getch()
#endif
/* Видалення таблиці векторів переривань */
void pre_main(void) __attribute__ ((naked)) __attribute__ ((section (".init8")));
int main(void) __attribute__ ((OS_main)) __attribute__ ((section (".init9")))
__attribute__((used));
void __attribute__((noinline)) __attribute__((leaf)) putch(char);
uint8_t __attribute__((noinline)) __attribute__((leaf)) getch(void) ;
void __attribute__((noinline)) verifySpace();
void __attribute__((noinline)) watchdogConfig(uint8_t x);
static void getNch(uint8_t);
#if LED_START_FLASHES > 0
    static inline void flash_led(uint8_t);
#endif
static inline void watchdogReset();
static inline void writebuffer(int8_t memtype, addr16_t mybuff,
                   addr16_t address, pagelen_t len);
static inline void read_mem(uint8_t memtype,
                        addr16_t, pagelen_t len);
#ifdef SOFT_UART
    void uartDelay() __attribute__ ((naked));
#endif
#if !defined(RAMSTART)  // newer versions of gcc avr-libc define RAMSTART
```

```
    #define RAMSTART 0x100
    #if defined (__AVR_ATmega644P__)
        #undef SIGNATURE_2
        #define SIGNATURE_2 0x0A
    #endif
#endif //?
static addr16_t buff = {(uint8_t *)(RAMSTART)};
/* Підтримка віртуального завантажувального розділу  */
#ifdef VIRTUAL_BOOT_PARTITION
    #define rstVect0_sav (*(uint8_t*)(RAMSTART+SPM_PAGESIZE*2+4))
    #define rstVect1_sav (*(uint8_t*)(RAMSTART+SPM_PAGESIZE*2+5))
    #define saveVect0_sav (*(uint8_t*)(RAMSTART+SPM_PAGESIZE*2+6))
    #define saveVect1_sav (*(uint8_t*)(RAMSTART+SPM_PAGESIZE*2+7))
    // Загальне оброблення векторів для малих об'ємів пам'яті
    #if !defined (save_vect_num)
        #if defined (SPM_RDY_vect_num)
            #define save_vect_num (SPM_RDY_vect_num)
        #elif defined (SPM_READY_vect_num)
            #define save_vect_num (SPM_READY_vect_num)
        #elif defined (EE_RDY_vect_num)
            #define save_vect_num (EE_RDY_vect_num)
        #elif defined (EE_READY_vect_num)
            #define save_vect_num (EE_READY_vect_num)
        #elif defined (WDT_vect_num)
            #define save_vect_num (WDT_vect_num)
        #else
            #error Cant find SPM or WDT interrupt vector for this CPU
        #endif
    #endif
// Альтернативне оброблення векторів для великих об'ємів пам'яті
    #if FLASHEND > 8192
        #define rstVect0 2
        #define rstVect1 3
        #define saveVect0 (save_vect_num*4+2)
        #define saveVect1 (save_vect_num*4+3)
        #define appstart_vec (save_vect_num*2)
    #else
        #define rstVect0 0
        #define rstVect1 1
        #define saveVect0 (save_vect_num*2)
        #define saveVect1 (save_vect_num*2+1)
        #define appstart_vec (save_vect_num)
    #endif
    #else
        #define appstart_vec (0)
    #endif
    /* Передзапуск */
    void pre_main(void) {
  // Таблиця переходів
    asm volatile (
        "   rjmp    1f\n"
        #ifndef APP_NOSPM
        "   rjmp    do_spm\n"
        #else
        "   ret\n"
        #endif
        "1:\n"
    );
    }
    /* Запуск */
    int main(void) {
    uint8_t ch;
```

```
/* Регістрова ініціалізація */
register addr16_t address;
register pagelen_t  length;
asm volatile ("clr __zero_reg__");
    #if defined(__AVR_ATmega8__)    ||  defined (__AVR_ATmega32__) || \
        defined (__AVR_ATmega162__)
        SP=RAMEND;
    #endif
    #if defined(__AVR_ATmega162__)
    ch = MCUCSR;
    #else
    ch = MCUSR;
    #endif
if (ch != 0) {
    if(((ch & _BV(WDRF)) | (UART_PIN & _BV(UART_RX_BIT))) != _BV(UART_RX_BIT)){
        if(UART_PIN & _BV(UART_RX_BIT)){
            #if defined(__AVR_ATmega162__)
        MCUCSR = ~(_BV(WDRF));
                #else
        MCUSR = ~(_BV(WDRF));
                #endif
        }
            __asm__ __volatile__ ("mov r2, %0\n" :: "r" (ch));
        watchdogConfig(WATCHDOG_OFF);
        __asm__ __volatile__ (
                #ifdef VIRTUAL_BOOT_PARTITION
            "ldi r30,%[rstvec]\n"
            "clr r31\n"
            "ijmp\n"::[rstvec] "M"(appstart_vec)
        );
  }
}
    #endif
    #if LED_START_FLASHES > 0
        #if defined(__AVR_ATtiny45__)||defined(__AVR_ATtiny85__)
        TCCR1 = 0x0E;
        #elif defined(__AVR_ATtiny43__)
        #error "LED flash for Tiny43 not yet supported"
        #else
        TCCR1B = _BV(CS12) | _BV(CS10);
        #endif
    #endif
    /* Налаштування передачі даних */
    #ifndef SOFT_UART
    #if defined(__AVR_ATmega8__) || defined (__AVR_ATmega32__)
        #ifndef SINGLESPEED
            UCSRA = _BV(U2X); ////Подвійна швидкість передачі USART
        #endif
        UCSRB = _BV(RXEN) | _BV(TXEN);
        UCSRC = _BV(URSEL) | _BV(UCSZ1) | _BV(UCSZ0);
        UBRRL = (uint8_t)BAUD_SETTING;
    #else
    #ifdef LIN_UART
        LINCR = (1 << LSWRES);
        LINBRRL=(uint8_t)BAUD_SETTING;
        LINBTR = (1 << LDISR) | (8 << LBT0);
        LINCR = _BV(LENA) | _BV(LCMD2) | _BV(LCMD1) | _BV(LCMD0);
        LINDAT=0;
    #else
  #ifndef SINGLESPEED
        UART_SRA = _BV(U2X0); //Подвійна швидкість передачі UART
  #endif
```

```
        UART_SRB = _BV(RXEN0) | _BV(TXEN0);
        UART_SRC = _BV(UCSZ00) | _BV(UCSZ01);
        UART_SRL = (uint8_t)BAUD_SETTING;
        #endif
  #endif
#endif
watchdogConfig(WATCHDOG_4S);
#if (LED_START_FLASHES > 0) || defined(LED_DATA_FLASH) || defined(LED_START_ON)
  LED_DDR |= _BV(LED);
#endif
#ifdef SOFT_UART
  UART_PORT &= ~_BV(UART_TX_BIT);
#endif
// Індикація при використанні завантажувача
#if LED_START_FLASHES > 0
  flash_led(LED_START_FLASHES * 2);
#else
#if defined(LED_START_ON)
  LED_PORT |= _BV(LED);
#endif
#endif
for (;;) {
    ch = getch();
    if(ch == STK_SET_DEVICE) {
      getNch(20);
    }
    else if(ch == STK_SET_DEVICE_EXT) {
      getNch(5);
    } // Прийом адреси
    else if(ch == STK_LOAD_ADDRESS) {
      address.bytes[0] = getch();
      address.bytes[1] = getch();
        #ifdef RAMPZ
      if (address.bytes[1] & 0x80) {
        RAMPZ |= 0x01;
      }
      else {
        RAMPZ &= 0xFE;
      }
        #endif
      address.word *= 2;
      verifySpace();
    }
    else if(ch == STK_UNIVERSAL) {
            #ifdef RAMPZ
        if ( AVR_OP_LOAD_EXT_ADDR == getch() ) {
        getch();
        RAMPZ = (RAMPZ & 0x01) | ((getch() << 1) & 0xff);
        getNch(1);
        putch(0x00);
        }
      else {
        getNch(3);
        putch(0x00);
      }
            #else
        getNch(4);
        putch(0x00);
            #endif
    }
    else if(ch == STK_PROG_PAGE) {
      // ПРОГРАМНА СТОРІНКА - програмування флеш-пам'яті
```

```
      uint8_t desttype;
      uint8_t *bufPtr;
      pagelen_t savelength;
      GETLENGTH(length);
      savelength = length;
      desttype = getch();
      bufPtr = buff.bptr;
      do *bufPtr++ = getch();
      while (--length);
      verifySpace();
            #ifdef VIRTUAL_BOOT_PARTITION
                #if FLASHEND > 8192
                // Загальне оброблення векторів для малих об'ємів пам'яті
                    #if FLASHEND > (128*1024)
                        #error "Can't use VIRTUAL_BOOT_PARTITION with more than 128k of
Flash"
                    #endif
            if (address.word == 0) {
                    // Налаштування порядку запуску Завантажувача
                        rstVect0_sav = buff.bptr[rstVect0];
                        rstVect1_sav = buff.bptr[rstVect1];
                buff.bptr[rstVect0] = ((uint16_t)main) & 0xFF;
                buff.bptr[rstVect1] = ((uint16_t)main) >> 8;
                        #if (save_vect_num>SPM_PAGESIZE/4)
                    } else if (address.word ==
(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/4)))) {
                        saveVect0_sav = buff.bptr[saveVect0-
(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/4)))];
                        saveVect1_sav = buff.bptr[saveVect1-
(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/4)))];
                buff.bptr[saveVect0-(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/4)))] =
rstVect0_sav;
                buff.bptr[saveVect1-(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/4)))] =
rstVect1_sav;
                }
                        #else
                    saveVect0_sav = buff.bptr[saveVect0];
                            saveVect1_sav = buff.bptr[saveVect1];
                    buff.bptr[saveVect0] = rstVect0_sav;
                    buff.bptr[saveVect1] = rstVect1_sav;
                    #endif
                    #else
            if (address.word == rstVect0) {
                        rstVect0_sav = buff.bptr[rstVect0];
                        rstVect1_sav = buff.bptr[rstVect1];
                        addr16_t vect;
                        vect.word = ((uint16_t)main);
                    buff.bptr[0] = vect.bytes[0];
                        buff.bptr[1] = vect.bytes[1] | 0xC0;
                    #if (save_vect_num > SPM_PAGESIZE/2)
                    } else if (address.word ==
(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/2)))) {
                        addr16_t vect;
                        vect.bytes[0] = rstVect0_sav;
                        vect.bytes[1] = rstVect1_sav;
                        saveVect0_sav = buff.bptr[saveVect0-
(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/2)))];
                        saveVect1_sav = buff.bptr[saveVect1-
(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/2)))];
                        vect.word = (vect.word-save_vect_num);
                buff.bptr[saveVect0-(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/2)))] =
vect.bytes[0];
```

```
                buff.bptr[saveVect1-(SPM_PAGESIZE*(save_vect_num/(SPM_PAGESIZE/2)))] =
(vect.bytes[1] & 0x0F)| 0xC0;  //
            }
                    #else
                        saveVect0_sav = buff.bptr[saveVect0];
                        saveVect1_sav = buff.bptr[saveVect1];
                        vect.bytes[0] = rstVect0_sav;
                        vect.bytes[1] = rstVect1_sav;
                        vect.word = (vect.word-save_vect_num);// виконання переривання
                        buff.bptr[saveVect0] = vect.bytes[0];
                    buff.bptr[saveVect1] = (vect.bytes[1] & 0x0F)| 0xC0;
                vect.word = ((uint16_t)main);
                buff.bptr[0] = vect.bytes[0];
      }
            #endif
        #endif
#endif
writebuffer(desttype, buff, address, savelength);
}
/* Зчитування програмного блоку  */
else if(ch == STK_READ_PAGE) {
    uint8_t desttype;
  GETLENGTH(length);
  desttype = getch();
  verifySpace();
  read_mem(desttype, address, length);
  }
else if(ch == STK_READ_SIGN) {
  verifySpace();
  putch(SIGNATURE_0);
  putch(SIGNATURE_1);
  putch(SIGNATURE_2);
  }
else if (ch == STK_LEAVE_PROGMODE) {
  watchdogConfig(WATCHDOG_16MS);
  verifySpace();
    }
putch(STK_OK);
}

    void putch(char ch) {
        #ifndef SOFT_UART
        #ifndef LIN_UART
            while (!(UART_SRA & _BV(UDRE0))) {  /* Spin */ }
        #else
            while (!(LINSIR & _BV(LTXOK)))   {  /* Spin */ }
        #endif
    UART_UDR = ch;
        #else
        UART_DDR |= _BV(UART_TX_BIT);
        __asm__ __volatile__ (
            "   rcall uartDelay\n"
            "   rcall uartDelay\n"
            "   com %[ch]\n"
            "   sec\n"
            "1: brcc 2f\n"
            "   cbi %[uartPort],%[uartBit]\n"
            "   rjmp 3f\n"
            "2: sbi %[uartPort],%[uartBit]\n"
            "   nop\n"
            "3: rcall uartDelay\n"
            "   rcall uartDelay\n"
```

```
        "   lsr %[ch]\n"
        "   dec %[bitcnt]\n"
        "   brne 1b\n":
        :[bitcnt] "d" (10),
    [ch] "r" (ch),
    [uartPort] "I" (_SFR_IO_ADDR(UART_PORT)),
    [uartBit] "I" (UART_TX_BIT):
    "r25"
    );
UART_DDR &= ~_BV(UART_TX_BIT);
    #endif
}
uint8_t getch(void) {
uint8_t ch;
    #ifdef LED_DATA_FLASH
        #if defined(__AVR_ATmega8__)    ||  \
            defined(__AVR_ATmega162__)  || defined(__AVR_ATmega32__)
        LED_PORT ^= _BV(LED);
        #else
        LED_PIN |= _BV(LED);
        #endif
    #endif
    #ifdef SOFT_UART
    watchdogReset();
        __asm__ __volatile__ (
          "1: sbic  %[uartPin],%[uartBit]\n"
          "   rjmp  1b\n"
          "   rcall uartDelay\n"
          "2: rcall uartDelay\n"
          "   rcall uartDelay\n"
          "   clc\n"
          "   sbic  %[uartPin],%[uartBit]\n"
          "   sec\n"
          "   dec   %[bitCnt]\n"
          "   breq  3f\n"
          "   ror   %[ch]\n"
          "   rjmp  2b\n"
          "3:\n":
           [ch] "=r" (ch):
           [bitCnt] "d" (9),
           [uartPin] "I" (_SFR_IO_ADDR(UART_PIN)),
           [uartBit] "I" (UART_RX_BIT)
          :"r25"
            );
    #else
    #ifndef LIN_UART
    while(!(UART_SRA & _BV(RXC0)))  {  /* Spin */ }
        if (!(UART_SRA & _BV(FE0))) {
    #else
                while(!(LINSIR & _BV(LRXOK)))  {  /* Spin */ }
                if (!(LINSIR & _BV(LFERR))) {
    #endif
                        watchdogReset();
                }
                        ch = UART_UDR;
#endif
#ifdef LED_DATA_FLASH
    #if defined(__AVR_ATmega8__)    ||  \
        defined(__AVR_ATmega162__)  || defined(__AVR_ATmega32__)
        LED_PORT ^= _BV(LED);
    #else
        LED_PIN |= _BV(LED);
```

```
        #endif
    #endif
        return ch;
    }

#ifdef SOFT_UART
    // Альтернативне оброблення векторів для великих об'ємів пам'яті
    #define UART_B_VALUE (((F_CPU/BAUD_RATE)-20)/6)
        #if UART_B_VALUE > 255
            #error Baud rate too slow for soft UART
        #endif
        #if UART_B_VALUE < 6
            #error Baud rate too high for soft UART
        #endif
    void uartDelay() {
  __asm__ __volatile__ (
    "ldi r25,%[count]\n"
    "1:dec r25\n"
    "brne 1b\n"
    "ret\n"
    ::[count] "M" (UART_B_VALUE)
  );
}
#endif
void getNch(uint8_t count) {
  do getch(); while (--count);
  verifySpace();
}
void verifySpace() {
  if (getch() != CRC_EOP) {
    watchdogConfig(WATCHDOG_16MS);
    while (1);
  }
  putch(STK_INSYNC);
}
#if LED_START_FLASHES > 0
void flash_led(uint8_t count) {
  do {
    #if defined(__AVR_ATtiny45__)||defined(__AVR_ATtiny85__)
        TCNT1 = -(F_CPU/(8196*16));
        TIFR = _BV(TOV1);
        while(!(TIFR & _BV(TOV1)));
    #elif defined(__AVR_ATtiny43__)
        #error "LED flash for Tiny43 not yet supported"
    #else
        TCNT1 = -(F_CPU/(1024*16));
        TIFR1 = _BV(TOV1);
        while(!(TIFR1 & _BV(TOV1)));
    #endif
    #if defined(__AVR_ATmega8__)    ||  \
        defined(__AVR_ATmega162__)  || defined(__AVR_ATmega32__)
        LED_PORT ^= _BV(LED);
    #else
        LED_PIN |= _BV(LED);
    #endif
  watchdogReset();
    #ifndef SOFT_UART
        #ifndef LIN_UART
        if (UART_SRA & _BV(RXC0))
        #else
            if (LINSIR & _BV(LRXOK))
        #endif
```

```
                    break;
    #endif
    } while (--count);
}
#endif
void watchdogReset() {
  __asm__ __volatile__ (
    "wdr\n"
  );
}
void watchdogConfig(uint8_t x) {
#ifdef WDCE
 #ifdef WDTCSR
  WDTCSR = _BV(WDCE) | _BV(WDE);
 #else
  WDTCR= _BV(WDCE) | _BV(WDE);
 #endif
#else
  CCP=0xD8;
#endif
#ifdef WDTCSR
  WDTCSR = x;
#else
  WDTCR= x;
#endif
}
static inline void writebuffer(int8_t memtype, addr16_t mybuff,
                    addr16_t address, pagelen_t len)
{
    switch (memtype) {
        case 'E': // EEPROM
#if defined(SUPPORT_EEPROM) || defined(BIGBOOT)
        while(len--) {
                eeprom_write_byte((address.bptr++), *(mybuff.bptr++));
        }
#else
        while (1);
#endif
        break;
    default:
        {
        // Копіювання буферу
        uint16_t addrPtr = address.word;
            #ifdef FOURPAGEERASE
        if ((address.bytes[0] & ((SPM_PAGESIZE<<2)-1))==0) {
            #endif
            __boot_page_erase_short(address.word);
            boot_spm_busy_wait();
            #ifdef FOURPAGEERASE
        }
            #endif
        do {
                __boot_page_fill_short((uint16_t)(void*)addrPtr, *(mybuff.wptr++));
                addrPtr += 2;
        } while (len -= 2);
        __boot_page_write_short(address.word);
        boot_spm_busy_wait();
            #if defined(RWWSRE)
        __boot_rww_enable_short();
            #endif
        } break;
  }
```

```
}

static inline void read_mem(uint8_t memtype, addr16_t address, pagelen_t length)
{
    uint8_t ch;
    switch (memtype) {
#if defined(SUPPORT_EEPROM) || defined(BIGBOOT)
    case 'E': // EEPROM
            do {putch(eeprom_read_byte((address.bptr++)));}
            while (--length);
        break;
#endif
    default:
            do {
#ifdef VIRTUAL_BOOT_PARTITION
                if (address.word == rstVect0) ch = rstVect0_sav;
                else if (address.word == rstVect1) ch = rstVect1_sav;
                else if (address.word == saveVect0) ch = saveVect0_sav;
                else if (address.word == saveVect1) ch = saveVect1_sav;
                else ch = pgm_read_byte_near(address.bptr);
                address.bptr++;
#elif defined(RAMPZ)
            __asm__ ("elpm %0,Z+\n" : "=r" (ch), "=z" (address.bptr): "1" (address));
#else
            __asm__ ("lpm %0,Z+\n" : "=r" (ch), "=z" (address.bptr): "1" (address));
#endif
            putch(ch);
            } while (--length); break;
    }
}
#ifndef APP_NOSPM
static void do_spm(uint16_t address, uint8_t command, uint16_t data)  __attribute__
((used));
static void do_spm(uint16_t address, uint8_t command, uint16_t data) {
    asm volatile (
        "    movw  r0, %3\n"
    "    __wr_spmcsr %0, %1\n"
    "    spm\n"
    "    clr  r1\n":
    : "i" (_SFR_MEM_ADDR(__SPM_REG)),
        "r" ((uint8_t)command),
        "z" ((uint16_t)address),
        "r" ((uint16_t)data)
    : "r0"
    );
    boot_spm_busy_wait();
        #if defined(RWWSRE)
    if ((command & (_BV(PGWRT)|_BV(PGERS))) && (data == 0) ) {
      __boot_rww_enable_short();
    }
        #endif
}
#endif
#ifdef BIGBOOT
    #define xstr(s) str(s)
    #define str(s) #s
    #define OPTFLASHSECT __attribute__((section(".fini8")))
    #define OPT2FLASH(o) OPTFLASHSECT const char f##o[] = #o "=" xstr(o)
    #ifdef LED_START_FLASHES
        OPT2FLASH(LED_START_FLASHES);
    #endif
    #ifdef LED_DATA_FLASH
```

```
    OPT2FLASH(LED_DATA_FLASH);
    #endif
    #ifdef LED_START_ON
    OPT2FLASH(LED_START_ON);
    #endif
    #ifdef LED_NAME
        OPTFLASHSECT const char f_LED[] = "LED=" LED_NAME;
    #endif
    #ifdef SUPPORT_EEPROM
        OPT2FLASH(SUPPORT_EEPROM);
    #endif
    #ifdef BAUD_RATE
        OPT2FLASH(BAUD_RATE);
    #endif
    #ifdef SOFT_UART
        OPT2FLASH(SOFT_UART);
    #endif
    #ifdef UART
        OPT2FLASH(UART);
    #endif
    OPTFLASHSECT const char f_date[] = "Built:" __DATE__ ":" __TIME__;
    #ifdef BIGBOOT
        OPT2FLASH(BIGBOOT);
    #endif
    #ifdef VIRTUAL_BOOT_PARTITION
        OPTFLASHSECT const char f_boot[] = "Virtual_Boot_Partition";
    #endif
        OPT2FLASH(F_CPU);
        OPTFLASHSECT const char f_device[] = "Device=" xstr(__AVR_DEVICE_NAME__);
#endif

/*************************************************************/
/*                  Optiboot Налаштування                    */
/*************************************************************/
#ifndef __COMMON_DEFINE_H
#define __COMMON_DEFINE_H
#include "stdint.h"
#define INTVECT_PAGE_ADDRESS 0x000  // Розташування початку адреси таблиці векторів
переривань
#define PAGE_SIZE 256 // Розмір сторінки флеш-пам'яті
#define BOOT_PAGE_ADDRESS 0x7800 //Початок розділу завантажувача
#define TOTAL_NO_OF_PAGES  128 // 16 КБ флеш-пам'яті, розділеної на сторінки розміром 128
байт

/* Кількість сторінок, що використовуються для коду завантажувача */
#define BOOTLOADER_PAGES (TOTAL_NO_OF_PAGES - BOOT_PAGE_ADDRESS/PAGE_SIZE)
/* перевірка меж під час запису/стирання сторінки */
#define LAST_PAGE_NO_TO_BE_ERASED   (TOTAL_NO_OF_PAGES - BOOTLOADER_PAGES)
#define SELFPROGEN SPMEN
#define MAX__APP_ADDR BOOT_PAGE_ADDRESS // Максимальна адреса програми

/*************************************************************************/
/*Адреса в EEPROM, де буде зберігатися ідентифікатор версії програми*/
#define EEMEM_ADDR_AVERSION 0xFF
#define BVERSION    0x10
#define CSTARTUP_ADDRESS 0x800
#define BOOT_FLAG 0x2A
#define SWUART_PORT_REG  PORTB
#define SWUART_PIN_REG PINB
#define SWUART_PIN PB0
#endif
```

```
/**********************************************************/
/*         Optiboot Визначення портів                     */
/**********************************************************/
#if    defined(__AVR_ATmega168__)   \
    || defined(__AVR_ATmega168P__)  \
    || defined(__AVR_ATmega328__)   \
    || defined(__AVR_ATmega328P__)  \
    || defined(__AVR_ATmega8__)
/*-------------------------------------------------------- */

#if !defined(LED)
#define LED B5
#endif
/* Порти для програмного UART */
#ifdef SOFT_UART
#define UART_PORT   PORTB
#define UART_PIN    PINB
#define UART_DDR    DDRB
#define UART_TX_BIT 0
#define UART_RX_BIT 0
#endif
#endif
#ifndef SOFT_UART
#if UART == 0
#if defined(UDR0)
# define UART_SRA UCSR0A
# define UART_SRB UCSR0B
# define UART_SRC UCSR0C
# define UART_SRL UBRR0L
# define UART_UDR UDR0
#elif defined(UDR)
# define UART_SRA UCSRA
# define UART_SRB UCSRB
# define UART_SRC UCSRC
# define UART_SRL UBRRL
# define UART_UDR UDR
#elif defined(LINDAT)
# define LIN_UART 1
# define UART_SRA UCSRA
# define UART_SRB UCSRB
# define UART_SRC UCSRC
# define UART_SRL UBRRL
# define UART_UDR LINDAT
#else
# error UART == 0, but no UART0 on device
#endif
#elif UART == 1
#if !defined(UDR1)
#error UART == 1, but no UART1 on device
#endif
# define UART_SRA UCSR1A
# define UART_SRB UCSR1B
# define UART_SRC UCSR1C
# define UART_SRL UBRR1L
# define UART_UDR UDR1
#elif UART == 2
#if !defined(UDR2)
#error UART == 2, but no UART2 on device
#endif
# define UART_SRA UCSR2A
# define UART_SRB UCSR2B
# define UART_SRC UCSR2C
# define UART_SRL UBRR2L
# define UART_UDR UDR2
#elif UART == 3
#if !defined(UDR3)
#error UART == 3, but no UART3 on device
#endif
# define UART_SRA UCSR3A
# define UART_SRB UCSR3B
# define UART_SRC UCSR3C
# define UART_SRL UBRR3L
# define UART_UDR UDR3
#endif
#endif
#if    defined(__AVR_ATmega8__)     \
    || defined (__AVR_ATmega32__)   \
    || defined (__AVR_ATmega16__)
  #define UCSR0A    UCSRA
  #define UDR0      UDR
  #define UDRE0     UDRE
  #define RXC0      RXC
  #define FE0           FE
  #define TIFR1     TIFR
  #define WDTCSR    WDTCR
#endif
#if    defined (__AVR_ATmega32__)   \
    || defined (__AVR_ATmega16__)
  #define WDCE      WDTOE
#endif
/* Підтримка Sanguino (та інших 40-
контактних DIP-процесорів) */
#if defined(__AVR_ATmega162__)
#if !defined(LED)
#define LED         B0
#endif
/* Порти для програмного UART */
#ifdef SOFT_UART
#define UART_PORT   PORTD
#define UART_PIN    PIND
#define UART_DDR    DDRD
#define UART_TX_BIT 1
#define UART_RX_BIT 0
#endif
#endif
#if defined(__AVR_ATmega32__)
#if !defined(LED)
#define LED         B0
#endif
#if defined(__AVR_ATmega32__)
#define UCSR0A  UCSRA
#define UDR0    UDR
#define UDRE0   UDRE
#define RXC0    RXC
```

```
#define FE0     FE
#define TIFR1   TIFR
#define WDTCSR  WDTCR
#endif
#if defined(__AVR_ATmega32__)
#define WDCE        WDTOE
#endif
/* Порти для програмного UART */
#ifdef SOFT_UART
#define UART_PORT   PORTD
#define UART_PIN    PIND
#define UART_DDR    DDRD
#define UART_TX_BIT 1
#define UART_RX_BIT 0
#endif
#endif
#if defined(__AVR_ATmega169__)
#if !defined(LED)
#define LED         B5
#endif
#define UCSR0A UCSRA
#define UCSR0B UCSRB
#define UCSR0C UCSRC
#define UBRR0L UBRRL
#define UDR0 UDR
#define UDRE0 UDRE
#define RXC0 RXC
#define FE0 FE
#define WDTCSR WDTCR
#define U2X0 U2X
#define RXEN0 RXEN
#define TXEN0 TXEN
#define UCSZ00 UCSZ0
#define UCSZ01 UCSZ1
/* Порти для програмного UART */
#ifdef SOFT_UART
#define UART_PORT   PORTE
#define UART_PIN    PINE
#define UART_DDR    DDRE
#define UART_TX_BIT 1
#define UART_RX_BIT 0
#endif
#endif
/*---------------------------------- */
#if defined(__AVR_ATmega169P__) ||
defined(__AVR_ATmega329__) \
|| defined(__AVR_ATmega329P__) \
|| defined(__AVR_ATmega3290__) ||
defined(__AVR_ATmega3290P__)
/*---------------------------------- */
#if !defined(LED)
#define LED         B5
#endif
#define WDTCSR WDTCR
/* Порти для програмного UART */
#ifdef SOFT_UART
#define UART_PORT   PORTE
#define UART_PIN    PINE
#define UART_DDR    DDRE
#define UART_TX_BIT 1
#define UART_RX_BIT 0
#endif
/*---------------------------------- */
#if defined(__AVR_ATmega169__)
/*---------------------------------- */
#if !defined(LED)
#define LED         B5
#endif
#define UCSR0A UCSRA
#define UCSR0B UCSRB
#define UCSR0C UCSRC
#define UBRR0L UBRRL
#define UDR0 UDR
#define UDRE0 UDRE
#define RXC0 RXC
#define FE0 FE
#define WDTCSR WDTCR
#define U2X0 U2X
#define RXEN0 RXEN
#define TXEN0 TXEN
#define UCSZ00 UCSZ0
#define UCSZ01 UCSZ1
/* Порти для програмного UART */
#ifdef SOFT_UART
#define UART_PORT   PORTE
#define UART_PIN    PINE
#define UART_DDR    DDRE
#define UART_TX_BIT 1
#define UART_RX_BIT 0
#endif
#endif
/*---------------------------------- */
#if defined(__AVR_ATmega169P__) ||
defined(__AVR_ATmega329__) \
|| defined(__AVR_ATmega329P__) \
|| defined(__AVR_ATmega3290__) ||
defined(__AVR_ATmega3290P__)
/*---------------------------------- */
#if !defined(LED)
#define LED         B5
#endif
#define WDTCSR WDTCR
/* Порти для програмного UART */
#ifdef SOFT_UART
#define UART_PORT   PORTE
#define UART_PIN    PINE
#define UART_DDR    DDRE
#define UART_TX_BIT 1
#define UART_RX_BIT 0
#endif
#endif
/*---------------------------------- */
#if defined(__AVR_ATtiny84__)
/*---------------------------------- */
#if !defined(LED)
#define LED   B2
#endif

#ifdef SOFT_UART
#define UART_PORT   PORTA
#define UART_PIN    PINA
#define UART_DDR    DDRA
#define UART_TX_BIT 1
```

```
#define UART_RX_BIT 2
#endif
#endif
/*------------------------------------ */
#if defined(__AVR_ATtiny44__)
/*------------------------------------ */
#if !defined(LED)
#define LED   B2
#endif

#ifdef SOFT_UART
#define UART_PORT   PORTA
#define UART_PIN    PINA
#define UART_DDR    DDRA
#define UART_TX_BIT 1
#define UART_RX_BIT 2
#endif
#endif
/*------------------------------------ */
#if defined(__AVR_ATtiny85__)
/*------------------------------------ */
#if !defined(LED)
#define LED   B2
#endif

#ifdef SOFT_UART
#define UART_PORT   PORTB
#define UART_PIN    PINB
#define UART_DDR    DDRB
#define UART_TX_BIT 0
#define UART_RX_BIT 1
#endif
#endif
/*------------------------------------ */
#if defined(__AVR_ATtiny45__)
/*------------------------------------ */
#if !defined(LED)
#define LED   B2
#endif
#ifdef SOFT_UART
#define UART_PORT   PORTB
#define UART_PIN    PINB
#define UART_DDR    DDRB
#define UART_TX_BIT 0
#define UART_RX_BIT 1
#endif
#endif
/*------------------------------------ */
#if defined(__AVR_ATtiny48__)
/*------------------------------------ */
#if !defined(LED)
#define LED   B5
#endif
#ifdef SOFT_UART
#define UART_PORT   PORTD
#define UART_PIN    PIND
#define UART_DDR    DDRD
#define UART_TX_BIT 6
#define UART_RX_BIT 7
#endif
#endif
/*------------------------------------ */
#if defined(__AVR_ATtiny167__)
/*------------------------------------ */
#if !defined(LED)
#define LED     A3
#endif
#endif
// Мітки портів.
#define A0 0x100
#define A1 0x101
#define A2 0x102
#define A3 0x103
#define A4 0x104
#define A5 0x105
#define A6 0x106
#define A7 0x107
#if !defined(PORTA)
#if LED >= A0 && LED <= A7
#undef LED
#define LED -1
#endif
#endif
#define B0 0x200
#define B1 0x201
#define B2 0x202
#define B3 0x203
#define B4 0x204
#define B5 0x205
#define B6 0x206
#define B7 0x207
#if !defined(PORTB)
#if LED >= B0 && LED <= B7
#undef LED
#define LED -1
#endif
#endif
#define C0 0x300
#define C1 0x301
#define C2 0x302
#define C3 0x303
#define C4 0x304
#define C5 0x305
#define C6 0x306
#define C7 0x307
#if !(defined(PORTC))
#if LED >= C0 && LED <= C7
#undef LED
#define LED -1
#endif
#endif
#define D0 0x400
#define D1 0x401
#define D2 0x402
#define D3 0x403
#define D4 0x404
#define D5 0x405
#define D6 0x406
#define D7 0x407
#if !(defined(PORTD))
#if LED >= D0 && LED <= D7
#undef LED
#define LED -1
```

```
#endif
#endif
#define E0 0x500
#define E1 0x501
#define E2 0x502
#define E3 0x503
#define E4 0x504
#define E5 0x505
#define E6 0x506
#define E7 0x507
#if !(defined(PORTE))
#if LED >= E0 && LED <= E7
#undef LED
#define LED -1
#endif
#endif
#if LED == B0
#define LED_NAME "B0"
#undef LED
#define LED_DDR     DDRB
#define LED_PORT    PORTB
#define LED_PIN     PINB
#define LED     PINB0
#elif LED == B1
#define LED_NAME "B1"
#undef LED
#define LED_DDR     DDRB
#define LED_PORT    PORTB
#define LED_PIN     PINB
#define LED     PINB1
#elif LED == B2
#define LED_NAME "B2"
#undef LED
#define LED_DDR     DDRB
#define LED_PORT    PORTB
#define LED_PIN     PINB
#define LED     PINB2
#elif LED == B3
#define LED_NAME "B3"
#undef LED
#define LED_DDR     DDRB
#define LED_PORT    PORTB
#define LED_PIN     PINB
#define LED     PINB3
#elif LED == B4
#define LED_NAME "B4"
#undef LED
#define LED_DDR     DDRB
#define LED_PORT    PORTB
#define LED_PIN     PINB
#define LED     PINB4
#elif LED == B5
#define LED_NAME "B5"
#undef LED
#define LED_DDR     DDRB
#define LED_PORT    PORTB
#define LED_PIN     PINB
#define LED     PINB5
#elif LED == B6
#define LED_NAME "B6"
#undef LED
#define LED_DDR     DDRB
#define LED_PORT    PORTB
#define LED_PIN     PINB
#define LED     PINB6
#elif LED == B7
#define LED_NAME "B7"
#undef LED
#define LED_DDR     DDRB
#define LED_PORT    PORTB
#define LED_PIN     PINB
#define LED     PINB7
#elif LED == C0
#define LED_NAME "C0"
#undef LED
#define LED_DDR     DDRC
#define LED_PORT    PORTC
#define LED_PIN     PINC
#define LED     PINC0
#elif LED == C1
#define LED_NAME "C1"
#undef LED
#define LED_DDR     DDRC
#define LED_PORT    PORTC
#define LED_PIN     PINC
#define LED     PINC1
#elif LED == C2
#define LED_NAME "C2"
#undef LED
#define LED_DDR     DDRC
#define LED_PORT    PORTC
#define LED_PIN     PINC
#define LED     PINC2
#elif LED == C3
#define LED_NAME "C3"
#undef LED
#define LED_DDR     DDRC
#define LED_PORT    PORTC
#define LED_PIN     PINC
#define LED     PINC3
#elif LED == C4
#define LED_NAME "C4"
#undef LED
#define LED_DDR     DDRC
#define LED_PORT    PORTC
#define LED_PIN     PINC
#define LED     PINC4
#elif LED == C5
#define LED_NAME "C5"
#undef LED
#define LED_DDR     DDRC
#define LED_PORT    PORTC
#define LED_PIN     PINC
#define LED     PINC5
#elif LED == C6
#define LED_NAME "C6"
#undef LED
#define LED_DDR     DDRC
#define LED_PORT    PORTC
#define LED_PIN     PINC
#define LED     PINC6
#elif LED == C7
#define LED_NAME "C7"
#undef LED
```

```
#define LED_DDR     DDRC
#define LED_PORT    PORTC
#define LED_PIN     PINC
#define LED     PINC7
#elif LED == D0
#define LED_NAME "D0"
#undef LED
#define LED_DDR     DDRD
#define LED_PORT    PORTD
#define LED_PIN     PIND
#define LED     PIND0
#elif LED == D1
#define LED_NAME "D1"
#undef LED
#define LED_DDR     DDRD
#define LED_PORT    PORTD
#define LED_PIN     PIND
#define LED     PIND1
#elif LED == D2
#define LED_NAME "D2"
#undef LED
#define LED_DDR     DDRD
#define LED_PORT    PORTD
#define LED_PIN     PIND
#define LED     PIND2
#elif LED == D3
#define LED_NAME "D3"
#undef LED
#define LED_DDR     DDRD
#define LED_PORT    PORTD
#define LED_PIN     PIND
#define LED     PIND3
#elif LED == D4
#define LED_NAME "D4"
#undef LED
#define LED_DDR     DDRD
#define LED_PORT    PORTD
#define LED_PIN     PIND
#define LED     PIND4
#elif LED == D5
#define LED_NAME "D5"
#undef LED
#define LED_DDR     DDRD
#define LED_PORT    PORTD
#define LED_PIN     PIND
#define LED     PIND5
#elif LED == D6
#define LED_NAME "D6"
#undef LED
#define LED_DDR     DDRD
#define LED_PORT    PORTD
#define LED_PIN     PIND
#define LED     PIND6
#elif LED == D7
#define LED_NAME "D7"
#undef LED
#define LED_DDR     DDRD
#define LED_PORT    PORTD
#define LED_PIN     PIND
#define LED     PIND7
#elif LED == E0
#define LED_NAME "E0"
#undef LED
#define LED_DDR     DDRE
#define LED_PORT    PORTE
#define LED_PIN     PINE
#define LED     PINE0
#elif LED == E1
#define LED_NAME "E1"
#undef LED
#define LED_DDR     DDRE
#define LED_PORT    PORTE
#define LED_PIN     PINE
#define LED     PINE1
#elif LED == E2
#define LED_NAME "E2"
#undef LED
#define LED_DDR     DDRE
#define LED_PORT    PORTE
#define LED_PIN     PINE
#define LED     PINE2
#elif LED == E3
#define LED_NAME "E3"
#undef LED
#define LED_DDR     DDRE
#define LED_PORT    PORTE
#define LED_PIN     PINE
#define LED     PINE3
#elif LED == E4
#define LED_NAME "E4"
#undef LED
#define LED_DDR     DDRE
#define LED_PORT    PORTE
#define LED_PIN     PINE
#define LED     PINE4
#elif LED == E5
#define LED_NAME "E5"
#undef LED
#define LED_DDR     DDRE
#define LED_PORT    PORTE
#define LED_PIN     PINE
#define LED     PINE5
#elif LED == E6
#define LED_NAME "E6"
#undef LED
#define LED_DDR     DDRE
#define LED_PORT    PORTE
#define LED_PIN     PINE
#define LED     PINE6
#elif LED == E7
#define LED_NAME "E7"
#undef LED
#define LED_DDR     DDRE
#define LED_PORT    PORTE
#define LED_PIN     PINE
#define LED     PINE7
#elif LED == A0
#define LED_NAME "A0"
#undef LED
#define LED_DDR     DDRA
#define LED_PORT    PORTA
#define LED_PIN     PINA
#define LED     PINA0
#elif LED == A1
```

```
#define LED_NAME "A1"
#undef LED
#define LED_DDR     DDRA
#define LED_PORT    PORTA
#define LED_PIN     PINA
#define LED     PINA1
#elif LED == A2
#define LED_NAME "A2"
#undef LED
#define LED_DDR     DDRA
#define LED_PORT    PORTA
#define LED_PIN     PINA
#define LED     PINA2
#elif LED == A3
#define LED_NAME "A3"
#undef LED
#define LED_DDR     DDRA
#define LED_PORT    PORTA
#define LED_PIN     PINA
#define LED     PINA3
#elif LED == A4
#define LED_NAME "A4"
#undef LED
#define LED_DDR     DDRA
#define LED_PORT    PORTA
#define LED_PIN     PINA
#define LED     PINA4
#elif LED == A5
#define LED_NAME "A5"
#undef LED
#define LED_DDR     DDRA
#define LED_PORT    PORTA
#define LED_PIN     PINA
#define LED     PINA5
#elif LED == A6
#define LED_NAME "A6"
#undef LED
#define LED_DDR     DDRA
#define LED_PORT    PORTA
#define LED_PIN     PINA
#define LED     PINA6
#elif LED == A7
#define LED_NAME "A7"
#undef LED
#define LED_DDR     DDRA
#define LED_PORT    PORTA
#define LED_PIN     PINA
#define LED     PINA7
#else

// Завершення компіляції.
#if LED == -1
#error Unrecognized LED name.  Should be like "B5"
// Завершення компіляції.
#pragma GCC diagnostic warning "-Wfatal-errors"
#error Nonexistent LED PORT.  Check datasheet.
#endif
#pragma GCC diagnostic warning "-Wfatal-errors"
#error Unrecognized LED name.  Should be like "B5"
#endif
```

Б.2 Лістинг коду програмного засобу “FixAVRHexCheckSum”

Модуль MainForm.cs

```
using System;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace FixAvrHexCheckSum
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "*.hex|*.hex|All files|*.*";
            if (ofd.ShowDialog() == DialogResult.OK) {
                Text = ofd.FileName;
                richTextBox.Text = File.ReadAllText(ofd.FileName);
            }
        }

        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (File.Exists(Text))
            {
                File.WriteAllText(Text, richTextBox.Text, Encoding.ASCII);
            }
            else {
                saveAsToolStripMenuItem_Click(null, null);
            }
        }

        private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFileDialog sfd = new SaveFileDialog();
            sfd.Filter = "*.hex|*.hex|All files|*.*";
            if (sfd.ShowDialog() == DialogResult.OK)
            {
                if (sender == null) Text = sfd.FileName;
                File.WriteAllText(sfd.FileName, richTextBox.Text, Encoding.ASCII);
            }
        }



        private void cutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            richTextBox.Cut();
        }

        private void copyToolStripMenuItem_Click(object sender, EventArgs e)
```

```
        {
            richTextBox.Copy();
        }

        private void pasteToolStripMenuItem_Click(object sender, EventArgs e)
        {
            richTextBox.Paste();
        }

        private void blockViewToolStripMenuItem_CheckedChanged(object sender, EventArgs e)
        {
            if (blockViewToolStripMenuItem.Checked)
            {
                for (int c = 0; c < richTextBox.Text.Length; c++)
                {
                    int i = c % 44;
                    if (i > 0 && i < 9) {
                        richTextBox.Select(c, 8);
                        richTextBox.SelectionBackColor = Color.LightGray;
                        c += 7;
                    }
                    if (i > 8 && i < 41) {
                        richTextBox.Select(c, 32);
                        richTextBox.SelectionBackColor = Color.LightBlue;
                        c += 31;
                    }
                    if (i > 40 && i < 43) {
                        richTextBox.Select(c, 2);
                        richTextBox.SelectionBackColor = Color.LightGreen;
                        c += 1;
                    }
                }
                richTextBox.DeselectAll();
            }
            else {
                richTextBox.SelectAll();
                richTextBox.SelectionBackColor = Color.Transparent;
                richTextBox.DeselectAll();
            }
        }

        private void fixChecksumsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            AvrHex avrHex = new AvrHex(richTextBox.Text);
            richTextBox.Text = avrHex.ProgramHexText;
        }
    }
}
```

Модуль AvrHex.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FixAvrHexCheckSum
{
    public class AvrHex
    {
        public class AvrHexLine
        {
            public int ByteCount { get; private set; }
            public int Address { get; private set; }
            public int RecordType { get; private set; }
            public byte[] Data { get; private set; }
            public byte InitialChecksum { get; private set; }
            public byte Checksum { get; private set; }

            public AvrHexLine(string hexLine)
            {
                string line = hexLine;
                if (line.StartsWith(":"))
                {
                    line = line.Length > 1 ? line.Substring(1) : "";
                }

                if (line.Length >= 2)
                {
                    ByteCount = int.Parse(line.Substring(0, 2),
                    System.Globalization.NumberStyles.HexNumber);
                    line = line.Remove(0, 2);
                }

                if (line.Length >= 4)
                {
                    Address = int.Parse(line.Substring(0, 4),
                    System.Globalization.NumberStyles.HexNumber);
                    line = line.Remove(0, 4);
                }

                if (line.Length >= 2)
                {
                    RecordType = int.Parse(line.Substring(0, 2),
                    System.Globalization.NumberStyles.HexNumber);
                    line = line.Remove(0, 2);
                }

                if (line.Length >= 2 * ByteCount)
                {
                    Data = new byte[ByteCount];
                    for (int i = 0; i < ByteCount; i++)
                    {
                        Data[i] = byte.Parse(line.Substring(0, 2),
                        System.Globalization.NumberStyles.HexNumber);
                        line = line.Remove(0, 2);
                    }
                }
```

```
            if (line.Length >= 2)
            {
                InitialChecksum = byte.Parse(line.Substring(0, 2),
                System.Globalization.NumberStyles.HexNumber);
                line = line.Remove(0, 2);
            }
            Checksum = CalculateChecksum(GetLineAsBytes());
        }

        public static byte CalculateChecksum(byte[] data)
        {
            byte result = 0;
            for (int i = 0; i < data.Length; i++)
            {
                result += (byte)(data[i] & (byte)0xff);
            }
            result ^= 0xff;
            result++;

            return result;
        }

        public byte[] GetLineAsBytes()
        {
            List<byte> result = new List<byte>();
            result.Add((byte)ByteCount);
            result.Add((byte)(Address >> 8));
            result.Add((byte)(Address & 0xff));
            result.Add((byte)RecordType);
            result.AddRange(Data);
            result.Add((byte)Checksum);
            return result.ToArray();
        }

        public string ToString()
        {
            string result = ":";
            byte[] bytes = GetLineAsBytes();
            foreach (byte b in bytes)
            {
                result += b.ToString("X2");
            }
            return result;
        }
    }
    public string ProgramHexText {get { return LinesToText(ProgramHexLines);}
    set { ProgramHexLines = TextLinesToHexLines(TextToLines(value));} }

    public AvrHexLine[] ProgramHexLines { get; set; }



    public AvrHex(string programHexText)
    {
        ProgramHexText = programHexText;
    }

    public AvrHex(string[] programHexLines)
    {
        ProgramHexLines = TextLinesToHexLines(programHexLines);
```

```
        }

        static public string LinesToText(string[] programHexLines)
        {
            string result = "";
            foreach (string s in programHexLines)
            {
                result += s + "\r\n";
            }
            return result;
        }

        static public string LinesToText(AvrHexLine[] programHexLines)
        {
            string result = "";
            foreach (AvrHexLine s in programHexLines)
            {
                result += s.ToString() + "\r\n";
            }
            return result;
        }

        static public string[] TextToLines(string programHexText)
        {
            List<string> result = new List<string>();
            string[] lines = programHexText.Split(new char[] {'\r','\n'},
             StringSplitOptions.RemoveEmptyEntries);

            result.AddRange(lines);

            return result.ToArray();
        }

        static public AvrHexLine[] TextLinesToHexLines(string[] programHexLines)
        {
            List<AvrHexLine> result = new List<AvrHexLine>();
            foreach (string s in programHexLines)
            {
                result.Add(new AvrHexLine(s));
            }

            return result.ToArray();
        }
    }
}
```