https://doi.org/10.31891/2307-5732-2025-353-60
УДК 004.7:004.75:004.49:004.62

КОЗЕЛЬСЬКИЙ ОЛЕКСАНДР
Хмельницький національний університет
https://orcid.org/0009-0002-7157-6499
e-mail: oleksandr.kozelskiy@khmnu.edu.ua
САВЕНКО БОГДАН
Хмельницький національний університет
https://orcid.org/0000-0001-5647-9979
e-mail: bsavenko@khmnu.edu.ua
САВЕНКО ОЛЕГ
Хмельницький національний університет
https://orcid.org/0000-0002-4104-745X
e-mail: savenko_oleg_st@ukr.net

ДВОРІВНЕВА СТРАТЕГІЯ ПІДВИЩЕННЯ ВІДМОВОСТІЙКОСТІ ОПЕРАЦІЙНИХ СИСТЕМ РЕАЛЬНОГО ЧАСУ З ВИКОРИСТАННЯМ ЙМОВІРНІСНОГО АНАЛІЗУ

У роботі запропоновано дворівневу модель забезпечення відмовостійкості операційних систем реального часу, що поєднує апаратний сторожовий таймер із програмним модулем, який реалізує ймовірнісний моніторинг та проактивне відновлення компонентів. На відміну від класичних підходів, що здебільшого реагують лише після виникнення критичних збоїв, розроблений механізм орієнтований на раннє виявлення ознак деградації роботи системи. Оцінювання ризику збоїв виконується на основі ймовірнісних розрахунків, які враховують час реакції завдань, ступінь заповнення черг повідомлень, стабільність сигналів живості та інші показники. У разі виявлення потенційно небезпечних відхилень система ініціює локальний перезапуск окремих завдань або драйверів ще до того, як стан наблизиться до критичного. Це дозволяє своєчасно стабілізувати роботу, запобігати накопиченню помилок, зменшувати навантаження на мікроконтролер і суттєво скорочувати кількість повних перезавантажень.

Інтеграція розробленої моделі з популярними операційними системами реального часу, зокрема FreeRTOS, спрощує її впровадження у вже наявні вбудовані рішення та забезпечує високу сумісність із апаратними платформами різних виробників. Запропонована стратегія особливо ефективна для систем із обмеженими обчислювальними ресурсами, таких як автономні робототехнічні комплекси, промислові контролери, безпілотні літальні апарати та пристрої Інтернету речей. Експериментальні дослідження показали, що застосування дворівневої моделі дозволяє зменшити середній час простою системи, знизити кількість глобальних перезапусків у кілька разів та підвищити загальну надійність роботи без значного збільшення використання процесорного часу й пам'яті. Отримані результати підтверджують доцільність використання підходу як гнучкого та ефективного засобу підвищення відмовостійкості сучасних кіберфізичних систем.

Ключові слова: операційні системи, ймовірнісний аналіз, сторожовий таймер, відмовостійкість, програмне скидання, кіберфізичні системи, надійність, системи реального часу

KOZELSKYI OLEKSANDR
SAVENKO BOHDAN
SAVENKO OLEG
Khmelnytskyi National University

DUAL-LEVEL STRATEGY FOR ENHANCING RTOS FAULT TOLERANCE USING PROBABILISTIC ANALYSIS

The paper proposes a two-level fault-tolerance model for real-time operating systems that combines a hardware watchdog timer with a software module implementing probabilistic monitoring and proactive component recovery. Unlike classical approaches, which mostly react only after critical failures occur, the developed mechanism focuses on early detection of system performance degradation. Failure risk assessment is performed based on probabilistic calculations that take into account task response times, message queue occupancy, stability of heartbeat signals signals, and other indicators. When potentially dangerous deviations are detected, the system initiates a local restart of individual tasks or drivers before the state approaches a critical level. This enables timely stabilization of operation, prevents error accumulation, reduces the load on the microcontroller, and significantly decreases the number of full system reboots.

Integration of the developed model with popular real-time operating systems, such as FreeRTOS, simplifies its implementation in existing embedded solutions and ensures high compatibility with hardware platforms from various manufacturers. The proposed strategy is particularly effective for systems with limited computing resources, such as autonomous robotic complexes, industrial controllers, unmanned aerial vehicles, and Internet of Things devices. Experimental studies have shown that applying the two-level model can reduce the system's average downtime, decrease the number of global restarts severalfold, and improve overall operational reliability without significantly increasing CPU and memory usage. The obtained results confirm the feasibility of using the proposed approach as a flexible and efficient means of enhancing the fault tolerance of modern cyber-physical systems.

Keywords: operating systems, probabilistic analysis, watchdog timer, fault tolerance, soft reset, cyber-physical systems, reliability, real-time systems

Стаття надійшла до редакції / Received 11.05.2025
Прийнята до друку / Accepted 27.05.2025

Постановка проблеми у загальному вигляді
та її зв'язок із важливими науковими чи практичними завданнями

Сучасні операційні системи реального часу (RTOS) вирізняються значною кількістю паралельно виконуваних завдань та високим рівнем інтеграції апаратних і програмних компонентів. Така архітектурна складність зумовлює серйозні виклики для забезпечення безперервності функціонування при одночасному дотриманні жорстких часових обмежень (дедлайнів). Подібні вимоги є особливо критичними для кіберфізичних систем, де навіть незначні збої або затримки в управлінні фізичними процесами здатні призвести до тяжких наслідків — від повної відмови системи та пошкодження обладнання до порушення функцій, важливих для безпеки [1,2]. З огляду на високий потенціал втрат у таких випадках, підвищення надійності та відмовостійкості залишається одним із пріоритетних напрямів досліджень у сфері вбудованих та реальних обчислювальних систем.

Одним з основних механізмів мінімізації ризику збоїв і підвищення надійності є використання сторожового таймера, реалізованого на апаратному або програмному рівні. Традиційна реалізація передбачає перезавантаження мікроконтролера після виявлення зупинки ядра чи критичного збою. Проте подібна реактивна стратегія усуває наслідки лише постфактум, спричиняючи значні часові затримки та додаткові обчислювальні витрати. У міру зростання складності систем і вимог до їхньої надійності дедалі більшого значення набувають гібридні підходи, що поєднують апаратний і програмний моніторинг. Зокрема, архітектура з подвійними сторожовими механізмами забезпечує можливість проактивного локального перезапуску окремих програмних компонентів на основі ймовірнісної моделі оцінки ризику, що дає змогу своєчасно виявляти ознаки відмови, уникати повного перезавантаження та істотно скорочувати час простою системи.

Вбудовані системи функціонують у межах суворих обмежень — зокрема, обмеженої обчислювальної потужності, жорстких часових рамок та дефіциту пам'яті [3]. За таких умов ресурсоємні методи аналізу, включаючи алгоритми машинного навчання або комплексні формальні моделі, часто є непридатними до застосування. Натомість спрощені ймовірнісні підходи забезпечують можливість оцінювання ризику відмов у режимі реального часу, використовуючи обмежений набір ключових системних параметрів. Це відкриває перспективу створення програмних прогнозних контролерів, здатних виявляти потенційно небезпечні тенденції ще до переходу системи в аварійний стан. У результаті стає можливим виконання цільового локального перезапуску окремих завдань, драйверів або модулів з метою запобігання повній відмові.

Аналіз досліджень та публікацій

У сфері забезпечення відмовостійкості вбудованих систем реального часу застосовується низка підходів, що передбачають різні моделі виявлення та усунення збоїв. Традиційно виділяють апаратні, програмні (soft) реалізації та комбіновані рішення, які поєднують обидва підходи: апаратний сторожовий таймер, модель із резервуванням, формальні моделі та верифікація, моделі на основі машинного навчання.

Апаратний сторожовий таймер, як правило, реалізується у вигляді вбудованого периферійного модуля мікроконтролера або зовнішнього наглядового пристрою з власним тактовим генератором. У межах такої архітектури застосунок зобов'язаний періодично оновлювати значення лічильника таймера або передавати сигнал, що підтверджує працездатність системи. Якщо оновлення у заданий інтервал не надходить, сторожовий таймер трактує це як відмову та ініціює повне перезавантаження системи. Завдяки незалежності від ядра операційної системи реального часу цей механізм здатний відновлювати роботу навіть у випадках повного зависання або потрапляння у тупиковий стан. Разом з тим, його дія має виключно реактивний характер, адже перезавантаження виконується лише після фактичного збою, що призводить до втрати поточного контексту виконання та простою. У критично важливих сферах, таких як автомобільна електроніка, промислова автоматизація або медичне обладнання, навіть короткочасне переривання тривалістю кілька секунд є неприпустимим.

Модель, заснована на принципі надмірності, передбачає дублювання критично важливих компонентів з метою підвищення загальної надійності. Найпоширенішими реалізаціями є N-модульна надмірність із логікою більшості та конфігурації з «гарячим» або «холодним» резервом, у яких резервний модуль миттєво або майже миттєво перебирає на себе функції основного у випадку його відмови [18,19]. Подібні рішення є необхідними у випадках, коли навіть мінімальна пауза в роботі неприпустима, наприклад, в авіаційних системах [15] чи на безперервних промислових конвеєрах. Однак впровадження надмірності вимагає значного додаткового апаратного забезпечення, складних механізмів синхронізації станів та суттєвих фінансових витрат [11,16], що робить цей підхід малопридатним для багатьох вбудованих платформ із обмеженими ресурсами [5].

Формальні методи та техніки верифікації, зокрема автомати з таймерами, мережі Петрі та методи перевірки моделей, широко використовуються у критично важливих галузях (наприклад, у рамках стандартів DO-178C, ISO 26262) для доведення відсутності певних класів помилок і тупикових станів ще на етапі проєктування [4]. Застосування таких методів забезпечує суворий аналіз динаміки системи та формальне підтвердження відповідності специфікації. Водночас їх інтеграція у процес онлайн-моніторингу збоїв обмежується [9] значними обчислювальними витратами [8], високою складністю реалізації та проблемою експоненційного зростання простору станів у реальних системах [6,7].

Методи машинного навчання (ML) набули широкого поширення у масштабних хмарних інфраструктурах та високопродуктивних обчислювальних кластерах (HPC), де вони активно застосовуються для прогнозування навантаження, виявлення аномалій і реалізації прогнозної міграції завдань [13]. Попри здатність таких моделей здійснювати аналіз у режимі реального часу, їх використання супроводжується значними вимогами до системних ресурсів, зокрема до обсягу оперативної пам'яті, продуктивності процесора та наявності високоякісних навчальних наборів даних. Навіть відносно легкі ML-алгоритми можуть виявитися надмірно ресурсомісткими для мікроконтролерів із обмеженими обчислювальними можливостями (наприклад, STM32, AVR, ESP32) через високий рівень споживання обчислювальних ресурсів і енергії, що особливо проявляється при реалізації стійкості до збоїв типу Візантійської відмови [14]. Додатковою проблемою є забезпечення детермінованої поведінки моделей ML у рідкісних або граничних сценаріях, що ускладнює їхнє застосування в умовах суворих вимог до верифікації, характерних для критично важливих галузей, таких як авіоніка та медичні технології.

Узагальнюючи, можна зазначити, що апаратні сторожові таймери забезпечують реактивне відновлення роботи після глобальних зависань, а програмні монітори дають змогу швидше реагувати на локальні збої. Водночас кожен із підходів має суттєві обмеження: системи надмірності вимагають значних апаратних ресурсів, тоді як формальні методи та ML-моделі часто є надто ресурсомісткими для застосування у простих вбудованих рішеннях.

Перспективним напрямом є використання гібридної стратегії, що поєднує переваги цих підходів. У такій архітектурі апаратний сторожовий таймер виконує функцію останнього рівня захисту від катастрофічних зависань, тоді як легкий програмний модуль проактивно виявляє ознаки потенційних відмов і ініціює локальний перезапуск лише уражених компонентів. Реалізація цього прогнозного шару може базуватися як на простих евристичних механізмах, побудованих на порогових значеннях, так і на аналітичних моделях оцінки ризику, сформованих на основі ймовірнісного аналізу [10]. З урахуванням обмежених ресурсів вбудованих середовищ та вимог до низької затримки виконання, оптимальним компромісом виступають спрощені ймовірнісні моделі з фіксованою кількістю дискретних станів. Такі підходи успішно зарекомендували себе, зокрема, у системах автоматизованих транспортних засобів (AGV) та у сфері прогнозного технічного обслуговування промислового обладнання [12], де аналіз історичних переходів між станами дає змогу завчасно оцінювати ризики, зменшувати потребу у повному перезапуску та забезпечувати безперервність функціонування системи.

Формулювання цілей статті

Метою роботи є розроблення та експериментальна перевірка моделі проактивного перезапуску компонентів для вбудованих систем реального часу, заснованої на ймовірнісному моніторингу ризику збоїв і інтегрованої з традиційними апаратними сторожовими таймерами. Такий підхід спрямований на зменшення частоти повних перезапусків системи та мінімізацію витрат часу й ресурсів, необхідних для її відновлення. Крім того, інтеграція розробленої моделі з апаратним рівнем керування підвищує відмовостійкість кіберфізичних систем з обмеженими ресурсами, що працюють в умовах інтенсивної зміни станів. Завдяки високій обчислювальній ефективності запропоноване рішення має значну практичну цінність для розробників вбудованих систем, забезпечуючи високу доступність критично важливих застосунків і зменшуючи наслідки можливих відмов.

Виклад основного матеріалу

Ефективне управління обчислювальними ресурсами та підтримання стабільної роботи операційних систем реального часу (RTOS) становлять фундаментальні умови їхньої надійності, особливо у контексті застосування в кіберфізичних системах (CPS) та середовищах Інтернету речей (IoT). Розширення сфер використання RTOS супроводжується різким зростанням потреби у високошвидкісній обробці значних потоків даних у режимі реального часу, що, своєю чергою, висуває підвищені вимоги до ефективних механізмів забезпечення відмовостійкості. Стандартна архітектура подібних систем зазвичай включає планувальник, який визначає порядок виконання завдань із застосуванням алгоритмів пріоритетного витісняючого планування або Earliest Deadline First (EDF); комплекс незалежних завдань, що реалізують функції збору даних із сенсорних пристроїв, обміну інформацією та управління виконавчими модулями; внутрішні сервіси синхронізації та обміну даними, які додатково можуть містити засоби логування та конфігурації; драйвери апаратних модулів для взаємодії з периферією; а також апаратний сторожовий таймер, що забезпечує контроль працездатності та примусовий перезапуск у разі критичного збою.

У типовій реалізації механізм сторожового таймера функціонує на основі періодичного оновлення його значення (feed_watchdog), що блокує його спрацювання за нормальних умов роботи системи. Якщо оновлення не відбувається у визначений часовий інтервал — наприклад, через зависання критично важливих компонентів — сторожовий таймер ініціює повне перезавантаження мікроконтролера. Попри здатність відновлювати працездатність, така схема має ключовий недолік: її реактивний характер означає, що перезапуск відбувається лише після фактичного виникнення збою, спричиняючи втрату поточного контексту виконання та збільшення часу простою. Це особливо неприпустимо у системах реального часу, де навіть короткочасне порушення роботи, наприклад у

виробничих лініях або медичних пристроях, може мати критичні наслідки. Традиційний підхід полягає у вставленні викликів feed_watchdog у ключові ділянки коду, щоб зменшити ймовірність зайвих перезапусків, однак він не надає можливості виконувати проактивний аналіз стану системи й локалізувати несправності без глобального скидання. У випадках, коли відбувається блокування планувальника чи шин даних, система змушена виконувати повний перезапуск, що неминуче призводить до втрати даних і зупинки роботи.

На рисунку 1 (ліва частина) подано схему типової архітектури ОСРЧ із вбудованим апаратним сторожовим таймером.

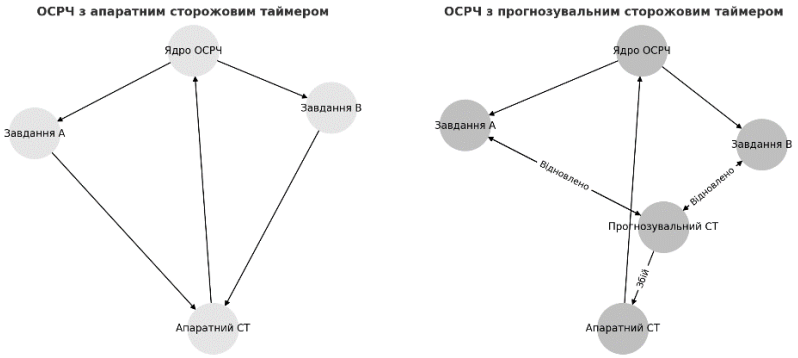

Рис.1. Порівняння структури ОСРЧ з апаратним сторожовим таймером і з інтегрованим прогнозувальним сторожовим таймером

Рисунок 1 ілюструє порівняння цієї базової структури з модернізованою версією, в якій класична конфігурація доповнена спеціалізованим програмним модулем для прогнозування збоїв та локального відновлення. Така надбудова забезпечує випереджальне реагування на відхилення у роботі системи, дозволяючи усунути їх ще до перетворення на критичні проблеми. Загальний вид архітектури, що охоплює всі взаємодіючі елементи, наведено на рисунку 2.

У вдосконаленій архітектурі зберігаються всі ключові компоненти класичної RTOS, включаючи планувальник, завдання, системні сервіси та драйвери пристроїв, а також апаратний сторожовий таймер, який використовується як резервний механізм захисту у випадках фатального зависання. Нововведенням є спеціалізований модуль завдання прогнозування стану, що працює з високим пріоритетом і виконує комплекс функцій моніторингу та превентивного відновлення. Його робота базується на постійному аналізі активності системи, що здійснюється шляхом збору даних від завдань через черги повідомлень або глобальні змінні з метою оцінки стабільності роботи. На основі отриманих параметрів, зокрема завантаженості черг, часу виконання завдань та частоти сигналів активності, модуль формує ймовірнісну оцінку ризику переходу системи у критичний стан. Якщо цей ризик перевищує заданий поріг, ініціюється локальне відновлення — перезапуск проблемного завдання або повторна ініціалізація драйвера.

У разі успішного локального відновлення завдання прогнозування стану продовжує оновлювати апаратний сторожовий таймер у встановлені інтервали, підтверджуючи стабільну роботу системи. Якщо ж відновлення не дає результату після кількох спроб, модуль навмисно припиняє подачу керуючого сигналу на апаратний таймер, що призводить до глобального перезапуску мікроконтролера. Такий механізм дозволяє поєднати превентивний аналіз, вибіркове усунення збоїв і класичний аварійний перезапуск, завдяки чому суттєво підвищується стійкість ОСРЧ до відмов, скорочується час простою та зростає адаптивність системи до змінних умов експлуатації.

Запропонована архітектура поєднує апаратний сторожовий таймер як кінцевий рубіж захисту від критичних збоїв та інтелектуальний програмний модуль (прогнозувальний сторожовий таймер), призначений для прогнозування ймовірності відмов і виконання локального відновлення компонентів до настання критичного стану. Її функціональна логіка реалізує дворівневий механізм реагування. На першому рівні програмний модуль превентивного контролю аналізує робочі параметри системи та, у разі виявлення відхилень від норми, виконує вибіркове інтелектуальне скидання окремих завдань або драйверів [17,20]. На другому рівні, якщо програмне відновлення не призводить до стабілізації або система повністю перестає реагувати, активується апаратний сторожовий таймер, який здійснює повний перезапуск.

Така дворівнева схема істотно зменшує частоту глобальних перезавантажень, скорочує час простою та забезпечує стабільну роботу вбудованих і кіберфізичних систем, що функціонують за умов обмежених ресурсів і жорстких часових обмежень. Оцінка стану виконується дискретно у моменти часу $t = k \cdot \Delta t$, де Δt — період виконання завдання прогнозування стану в операційній системі реального часу. На кожному такті система може перебувати в одному з фіксованих станів, наприклад “Normal”, “Critical”, “Fail”. Перехід між цими станами визначається аналізом поточних експлуатаційних показників, таких як тривалість виконання завдань, рівень завантаженості черг чи частота отримання сигналів активності. Запроваджений ймовірнісний підхід дозволяє передбачати потенційні збої та ініціювати превентивні дії до моменту переходу системи у критичний режим.

Отже є скінчена множина станів:

$$S = \{N, C, F\}, \quad (1)$$

де N, C, F – стани “Normal”, “Critical”, “Fail”.

У кожен такт $t^k = k\Delta t$ система (або завдання прогнозування стану, яке відображає її стан) опиняється в одному зі станів $x(k) \in S$. Для спрощення припустимо, що N (Normal) – система в нормальному діапазоні (черги не переповнені, сигнали про активність усіх завдань приходять вчасно), C (Critical) – ризик швидкого збою високий (суттєві затримки, часті пропуски дедлайнів, завдання припиняє надсилати сигнали про активність). F (Fail) – фактичне зависання / збій (у моделі – це поглинаючий стан або індикатор, що відбувся глобальне скидання).

Перехід між станами описується матрицею ймовірностей переходу $P \in R^{3\times3}$:

$$P = \begin{bmatrix} P_{NN} & P_{NC} & P_{NF} \\ P_{CN} & P_{CC} & P_{CF} \\ 0 & 0 & 1 \end{bmatrix}, \quad (2)$$

де $p_{ij} = P(x+1) = j | x(k) = i)$,а $x(k)$ – стан системи в момент часу k. Стан F є поглинаючим: після його досягнення виконується жорсткий перезапуск системи. На кожному кроці k обчислюється ймовірність переходу до стану збою:

$$P_F(k) = P\big(x(k+1)\big) = F\big|x(k)\big), \quad (3)$$

тобто:

$$P_F(k) = P_{CF} * I_{x(k)=C} + P_{NF} * I_{x(k)=N} = N. \quad (4)$$

Встановлюється поріг спрацювання $\theta \in [0,1]$. Якщо:

$$P_F(k) \geq \theta, \quad (5)$$

то виконується програмний (soft) перезапуск задачі, яку вважаємо причиною критичного стану:

$$Restart(T_i) := ReInit(T_i) = D(T_i) \circ C(T_i), \quad (6)$$

де T_i — ідентифікатор задачі, яку слід перезапустити, $D(T_i)$— операція з видалення задачі з планувальника RTOS, $C(T_i)$ — повторне створення задачі з її початковою конфігурацією, $\circ$ — композиція (послідовне виконання дій).

Нехай Δt — крок оновлення стану, а T_{HW} — таймаут апаратного сторожового таймера. Якщо протягом m кроків система перебуває у стані C, і локальне відновлення не дало ефекту, прогнозувальний сторожовий таймер навмисно припиняє подачу сигналу оновлення апаратному сторожовому таймері у момент часу $k+m$:

$$feedWDT(k+m) = 0, \quad (7)$$

що запускає повне скидання. Щоб забезпечити адаптивність, ймовірності переходів p_{ij} можуть оновлюватися експоненційно-зваженим методом:

$$P_{ij}^{(k)} = \alpha \cdot I_{x(k-1)=i \wedge x(k)=j} + (1-a) \cdot P_{ij}^{(k-1)}, \quad (8)$$

де $\alpha \in (0,1)$ — коефіцієнт згладжування (наприклад, $\alpha = 0{,}1$), $I_{x(k-1)=i \wedge x(k)=j}$ —це індикаторна функція, яка набуває значення 1 лише тоді, коли в попередній такт $k-1$ система була в стані i, а в поточному такті k перейшла в стан j. Інакше вона дорівнює 0.

Таким чином, визначення N, C, F залежить від прикладного сценарію й встановлених порогових значень. Такі пороги визначають емпірично чи на основі специфіки застосунку.

Ймовірності переходів між станами у прогнозувальному сторожовому таймері є динамічними і можуть коригуватися в реальному часі залежно від змін у ключових системних метриках. Основними факторами впливу є заповненість черги, затримка сигналів активності системи, використання пам’яті та інші параметри продуктивності системи.

Запропонована модель реалізує двоетапний підхід до відновлення системи:

1) Якщо аналіз показує високу ймовірність ($\geq\theta\geq\theta$) переходу до стану Fail (F), завдання прогнозування стану ініціює локальне відновлення проблемного компонента, виконавши інтелектуальне скидання.

2) Якщо після інтелектуального скидання система не стабілізувалася (стан Critical (C) зберігається), програмний шар навмисно не оновлює апаратний сторожовий таймер (feed_watchdog() не викликається). За час T_{HW} (таймаут апаратного сторожового таймера) відбудеться повне скидання мікроконтролера, що поверне систему до стану Normal (N).

Таким чином, якщо відхилення не надто глибоке, інтелектуальне скидання дає змогу уникнути глобального перезапуску і зменшити час простою. Апаратний сторожовий таймер зберігається як останній рубіж, якщо система вийшла з-під контролю програмного шару. Формально якщо на кроці k запущено процедуру інтелектуального скидання, то на кроці $k+1$ штучно встановлюємо $x(k+1) = N$ (якщо перезапуск відбувся успішно).

Оскільки ми маємо $\Delta t \ll T_{HW}$, тоді програмний шар має кілька спроб (наприклад, 10–20 циклів із кроком Δt) виконати програмне скидання, перш ніж спливе таймаут апартного сторожового таймера. Якщо завдання прогнозування стану фіксує, що стан знову став Normal (або Warning без прогресу до Critical), він викликає feed_watchdog(), продовжуючи нормальну роботу. Якщо локальний збій не усувається, завдання прогнозування стану свідомо не передає керуючий сигнал апаратному сторожовому таймеру, що призводить до його спрацювання та ініціює повний перезапуск системи (Рисунок 3).

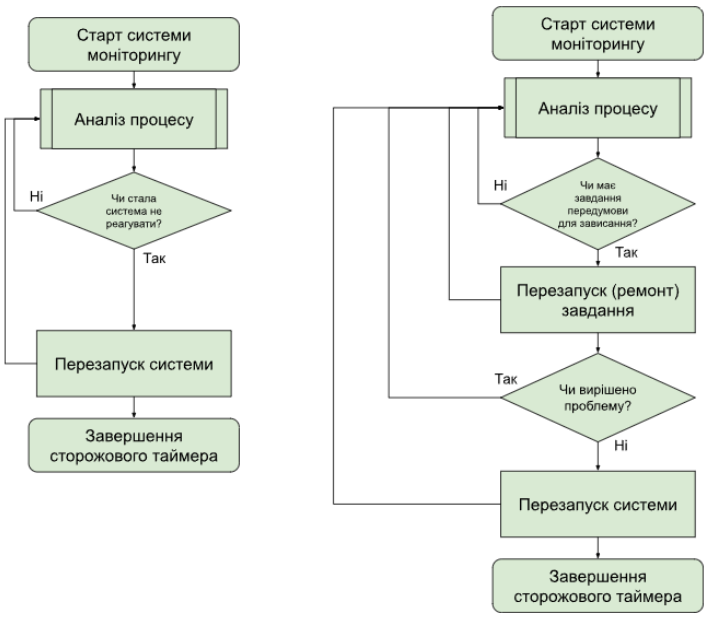

Рис. 3. Порівняльна схема роботи стандартного й модифікованого сторожового таймера

Математична інтерпретація запропонованого підходу описується як послідовність формалізованих кроків. У дискретний момент часу стан системи позначається як $x(k)$ і належить скінченій множині можливих станів S. Перехід між станами визначається ймовірностями, заданими матрицею переходів P. Ймовірність переходу зі стану "Critical" до стану "Fail" (P_{iF}) розглядається як показник ризику критичного збою. Якщо цей показник перевищує встановлений поріг θ, ініціюється превентивне локальне відновлення (інтелектуальне скидання). У разі потреби прогнозування може бути розширене на rrr тактів уперед шляхом піднесення матриці переходів до степеня P^r. Якщо локальне відновлення не дає результату, система утримується у стані "Critical" до завершення тайм-ауту апаратного сторожового таймера, після чого відбувається глобальний перезапуск.

Запропонований метод базується на дискретному відстеженні змін стану з можливістю динамічного оновлення матриці P у реальному часі, що забезпечує виявлення потенційних збоїв при мінімальних обчислювальних витратах. Її концепція поєднує ймовірнісну оцінку ризику з дворівневою стратегією відновлення: на першому рівні виконується локальне інтелектуальне скидання, яке активується при зростанні ймовірності відмови та дозволяє швидко відновити роботу окремих завдань або драйверів; на другому рівні здійснюється глобальний перезапуск через апаратний сторожовий таймер, що використовується лише у випадках, коли програмні засоби виявилися неефективними.

Завдяки такій організації модель здатна усувати більшість нефатальних збоїв без необхідності повного скидання, зберігаючи при цьому можливість гарантованого відновлення працездатності у разі серйозних відмов.

У підсумку математична схема має такі кроки:

1) Стан системи позначається як $x(k)$ – дискретний момент часу у скінченому просторі S.

2) Перехід між станами визначаються ймовірностями:

3) Ймовірність збою за один крок визначається як p_{iF} (з "Critical" у "Fail") При перевищенні порога θ – запускаємо превентивні дії (інтелектуальне скидання).

4) За необхідності розглядаємо прогноз на r кроків через P^r.

5**)** Якщо локальне відновлення неефективне, система переходить у стан C (Critical) та продовжує працювати до таймауту апаратного сторожового таймера.

Запропонована модель базується на дискретному аналізі змін стану системи з можливістю динамічного оновлення матриці ймовірностей P у реальному часі, що забезпечує ефективне виявлення потенційних збоїв без значних обчислювальних витрат. Основна концепція моделі поєднує імовірнісний підхід до оцінки ризику відмов із двоступеневою стратегією відновлення, яка передбачає:

1) Локальне превентивне відновлення (інтелектуальне скидання) – застосовується у разі виявлення зростання ризику збою, дозволяючи відновити працездатність окремих завдань чи драйверів без глобального перезапуску.

2) Глобальний перезапуск через апаратний сторожовий таймер – виконується лише у разі, якщо програмне відновлення не дало ефекту, що гарантує відновлення працездатності системи у критичних випадках.

Таким чином, модель дозволяє ефективно уникати дрібних збоїв, знижуючи потребу у повному скиданні, водночас забезпечуючи стійкість до критичних збоїв, коли програмні методи не дають результату.

Експерименти

Для перевірки ефективності двоступеневого механізму відмовостійкості, що поєднує апаратний сторожовий таймер і програмний рівень з адаптивним аналізом, проведено експерименти на платформі STM32F407 (Cortex-M4, 168 МГц) з ОС реального часу FreeRTOS. Було реалізовано п'ять завдань: TaskA — основне, з інтенсивною роботою з периферією (SPI/UART); TaskB і TaskC — допоміжні для обробки сигналів і взаємодії з датчиками; Logger — сервіс реєстрації подій; Завдання прогнозування стану — модуль інтелектуального контролю з періодом 100 мс. Апаратний сторожовий таймер мав таймаут 2 с. Завдання прогнозування стану аналізувало сигнали активності й заповнення черг, використовуючи динамічну ймовірнісну модель з історією 50 останніх переходів. При перевищенні порога ймовірності переходу в Fail виконувався селективне інтелектуальне скидання, а у разі неуспіху — свідоме неоновлення апаратного сторожового таймера, що через 2,8 с спричиняло повне скидання.

Сценарії збоїв включали штучне блокування TaskA, перевантаження черг понад 80% та затримки сигналів активності. Вимірювали час виявлення збою— від початку аномалії до входу в Critical і запуску інтелектуального скидання, та час відновлення — від аномалії до повернення в Normal. Порівнювали тривалість інтелектуального скидання і повного перезапуску через апаратний сторожовий таймер. Для кожного сценарію проводили щонайменше 10 повторів з фіксацією результатів у Logger та FreeRTOS+Trace.

Додатково оцінювався коефіцієнт глобальних перезапусків, який показував частку випадків, коли локальне відновлення було безуспішним і відбувався повний скидання. Аналізували, чи ефективно завдання прогнозування стану запобігає фатальним збоям, не пропускає критичні ситуації та не виконує зайві інтелектуальні скидання. Також проводилося профілювання FreeRTOS із вбудованими засобами трасування, щоб перевірити додаткове навантаження на процесор від роботи моделі. Результати підтвердили, що обчислювальні витрати мінімальні і не впливають на критичні завдання.

У тестах перевантаження SPI зависання TaskA виявлялися за 0,11–0,13 с (період перевірки 100 мс), а інтелектуальне скидання займало до 0,35 с. Для порівняння, апаратний сторожовий таймер потребував близько 3 с, що давало 4,5% простою за 3 хвилини (Рисунок 4). При взаємному блокуванні TaskA у 100% випадків відновлення відбувалося локально без глобального скидання, зменшуючи простій до 1,88% (на 42% менше, ніж у першому тесті).

Отримані дані підтверджують, що регулярний м'який перезапуск проблемних компонентів суттєво скорочує середній час простою порівняно з повним перезапуском, зберігаючи при цьому надійний резервний захист на випадок неуспішного відновлення.

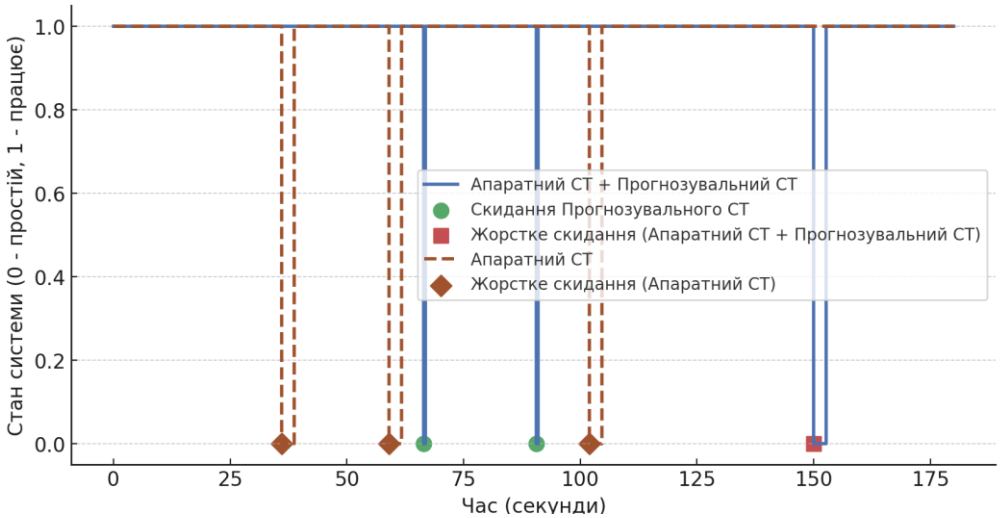

Рис.4. Оцінка ефективності системи з прогнозувальним сторожовим таймером та без нього в умовах випадкових збоїв (5 хвилин)

Отримані результати свідчать про доцільність застосування двоступеневого (програмно-апаратного) механізму для підвищення надійності вбудованих та кіберфізичних систем. Запропонований підхід особливо важливий у сферах, де простої системи можуть мати критичні наслідки – промислова автоматика, робототехніка, транспорт, медичні пристрої – але при цьому є жорсткі обмеження на апаратні ресурси, що унеможливлює використання складних ML-моделей або резервних апаратних схем.

Таким чином, запропонована модель забезпечує компроміс між адаптивною реакцією на відмови та гарантованою стабільністю системи, мінімізуючи кількість глобальних перезапусків, скорочуючи час простою та не створюючи надмірного навантаження на обчислювальні ресурси.

Висновки з даного дослідження
і перспективи подальших розвідок у даному напрямі

У дослідженні запропоновано дворівневу концепцію підвищення відмовостійкості для вбудованих і кіберфізичних систем, що поєднує апаратний сторожовий таймер як фінальний бар'єр від критичних збоїв та інтелектуальний програмний модуль (прогнозувальний сторожовий таймер), здатний у проактивному режимі оцінювати ймовірність відмови й виконувати локальне відновлення окремих компонентів. Ключовий принцип підходу полягає у випереджальному реагуванні: замість очікування фактичного зависання система проводить динамічний аналіз поточних параметрів, прогнозує ризик переходу в критичний режим і своєчасно ініціює превентивні дії. У випадку, коли локальні заходи не забезпечують відновлення, програмний шар навмисно допускає спрацювання апаратного таймера, що призводить до повного перезапуску при глибокому зависанні.

Запропонована архітектура продемонструвала високу ефективність у контексті систем реального часу, де особливо важливим є швидке та надійне відновлення працездатності за умов жорстких часових обмежень. Використання компактної ймовірнісної моделі дає змогу виконувати оцінку стану без залучення ресурсоємних формальних методів чи алгоритмів машинного навчання, а поєднання програмного превентивного відновлення з апаратним резервним механізмом створює збалансовану комбінацію реактивних та проактивних стратегій. Експериментальні результати підтверджують, що такий підхід є доцільним у широкому спектрі застосувань — від промислової автоматики й робототехніки до автомобільних та медичних систем, — де мінімізація часу простою і безперервність роботи мають критичне значення. Крім того, модель відкриває перспективи для подальшої оптимізації та розробки нових методів підвищення відмовостійкості.

Література

1. Hassan, M., & Patel, H. (2020). A framework for probabilistic timing analysis of real-time systems under soft errors. *IEEE Transactions on Computers*, 69(4), 548–561. DOI: 10.1109/TC.2019.2958071.
2. Sun, X., Qin, X., Yang, L., & Zhang, Z. (2021). Design and implementation of a hybrid software-hardware watchdog timer for safety-critical embedded systems. *IEEE Transactions on Industrial Electronics*, 68(12), 12294–12303. DOI: 10.1109/TIE.2020.3035476.
3. Pradhan, A., & Dutt, N. (2022). Lightweight fault-tolerant mechanisms for resource-constrained real-time embedded systems. *ACM Transactions on Embedded Computing Systems*, 21(5), 1–27. DOI: 10.1145/3514257.
4. Huang, C., Chen, J., & Yang, C. (2020). Formal verification of safety-critical real-time systems using timed automata. *IEEE Access*, 8, 185902–185915. DOI: 10.1109/ACCESS.2020.3029897.
5. Kim, Y., Kim, H., & Lee, S. (2021). Resource-efficient redundancy techniques for fault-tolerant embedded real-time systems. *Microprocessors and Microsystems*, 84, 104291. DOI: 10.1016/j.micpro.2021.104291.
6. Tang, X., Chen, L., & Li, J. (2021). State space reduction in model checking of real-time embedded systems. *Journal of Systems Architecture*, 118, 102168. DOI: 10.1016/j.sysarc.2021.102168.
7. Hasan, R., & Islam, S. (2022). Scalability issues in formal verification of real-time distributed systems. *Future Generation Computer Systems*, 135, 267–281. DOI: 10.1016/j.future.2022.05.011.
8. Abdallah, M., & Rehman, S. (2023). Computational overhead analysis of runtime verification in embedded real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(5), 1584–1597. DOI: 10.1109/TCAD.2022.3189801.
9. Ahmed, M., & Khan, S. (2021). Limitations of formal methods in real-time embedded system fault detection. *ACM Transactions on Embedded Computing Systems*, 20(6), 1–26. DOI: 10.1145/3477138.
10. Magliano, E. (2025). *Real-Time Embedded System Fault Injector Framework for...* Journal of Microelectronic Reliability. DOI: 10.1007/s10836-025-06170-w.
11. He, X., & Wei, L. (2020). Cost analysis of redundant architectures for fault-tolerant real-time embedded systems. *IEEE Transactions on Industrial Electronics*, 67(12), 10650–10659. DOI: 10.1109/TIE.2019.2952821.
12. Mahale, Y. (2025). *AI Applications in Vehicle Maintenance Strategies and Diagnostics*. SN Applied Sciences. DOI: 10.1007/s42452-025-06681-3.
13. Rahman, M. M., Gupta, D., Bhatt, S., Shokouhmand, S., & Faezipour, M. (2024). *A Comprehensive Review of Machine Learning Approaches for Anomaly Detection in Smart Homes: Experimental Analysis and Future Directions*. Future Internet, 16(4), 139. DOI: 10.3390/fi16040139.
14. Xu, G. (2023). *An Optimized Byzantine Fault Tolerance Algorithm for the Medical Field*. Electronics, 12(24), 5045. DOI: 10.3390/electronics12245045.
15. Zhou, Y., & Xu, Z. (2021). Redundancy strategies in avionics systems for fault tolerance. *Aerospace Science and Technology*, 116, 106851. DOI: 10.1016/j.ast.2021.106851.
16. Li, F., & Zhao, H. (2022). Synchronization mechanisms in hot-standby redundancy for industrial control systems. *Journal of Manufacturing Systems*, 62, 516–526. DOI: 10.1016/j.jmsy.2021.12.003.

17. Zeyi Liu, Songqiao Hu, & Xiao He (2023). *Real-time safety assessment of dynamic systems in non-stationary environments: A review of methods and techniques*. arXiv. Retrieved April 25, 2023.

18. Wang, D., & Zhang, T. (2022). N-modular redundancy with majority voting for safety-critical applications: Design and evaluation. *IEEE Transactions on Reliability*, 71(4), 1421–1434. DOI: 10.1109/TR.2021.3118246.

19. Park, J., & Choi, K. (2023). Cold and hot standby redundancy in real-time embedded systems: A performance comparison. *Microelectronics Reliability*, 139, 114869. DOI: 10.1016/j.microrel.2023.114869.

20. Tinglue Wang et al. (2025). *FlexStep: Enabling flexible error detection in multi/many-core real-time systems*. arXiv. Retrieved March 18, 2025.

References

1. Hassan, M., & Patel, H. (2020). A framework for probabilistic timing analysis of real-time systems under soft errors. *IEEE Transactions on Computers*, 69(4), 548–561. DOI: 10.1109/TC.2019.2958071.

2. Sun, X., Qin, X., Yang, L., & Zhang, Z. (2021). Design and implementation of a hybrid software-hardware watchdog timer for safety-critical embedded systems. *IEEE Transactions on Industrial Electronics*, 68(12), 12294–12303. DOI: 10.1109/TIE.2020.3035476.

3. Pradhan, A., & Dutt, N. (2022). Lightweight fault-tolerant mechanisms for resource-constrained real-time embedded systems. *ACM Transactions on Embedded Computing Systems*, 21(5), 1–27. DOI: 10.1145/3514257.

4. Huang, C., Chen, J., & Yang, C. (2020). Formal verification of safety-critical real-time systems using timed automata. *IEEE Access*, 8, 185902–185915. DOI: 10.1109/ACCESS.2020.3029897.

5. Kim, Y., Kim, H., & Lee, S. (2021). Resource-efficient redundancy techniques for fault-tolerant embedded real-time systems. *Microprocessors and Microsystems*, 84, 104291. DOI: 10.1016/j.micpro.2021.104291.

6. Tang, X., Chen, L., & Li, J. (2021). State space reduction in model checking of real-time embedded systems. *Journal of Systems Architecture*, 118, 102168. DOI: 10.1016/j.sysarc.2021.102168.

7. Hasan, R., & Islam, S. (2022). Scalability issues in formal verification of real-time distributed systems. *Future Generation Computer Systems*, 135, 267–281. DOI: 10.1016/j.future.2022.05.011.

8. Abdallah, M., & Rehman, S. (2023). Computational overhead analysis of runtime verification in embedded real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(5), 1584–1597. DOI: 10.1109/TCAD.2022.3189801.

9. Ahmed, M., & Khan, S. (2021). Limitations of formal methods in real-time embedded system fault detection. *ACM Transactions on Embedded Computing Systems*, 20(6), 1–26. DOI: 10.1145/3477138.

10. Magliano, E. (2025). *Real-Time Embedded System Fault Injector Framework for...* Journal of Microelectronic Reliability. DOI: 10.1007/s10836-025-06170-w.

11. He, X., & Wei, L. (2020). Cost analysis of redundant architectures for fault-tolerant real-time embedded systems. *IEEE Transactions on Industrial Electronics*, 67(12), 10650–10659. DOI: 10.1109/TIE.2019.2952821.

12. Mahale, Y. (2025). *AI Applications in Vehicle Maintenance Strategies and Diagnostics*. SN Applied Sciences. DOI: 10.1007/s42452-025-06681-3.

13. Rahman, M. M., Gupta, D., Bhatt, S., Shokouhmand, S., & Faezipour, M. (2024). *A Comprehensive Review of Machine Learning Approaches for Anomaly Detection in Smart Homes: Experimental Analysis and Future Directions*. Future Internet, 16(4), 139. DOI: 10.3390/fi16040139.

14. Xu, G. (2023). *An Optimized Byzantine Fault Tolerance Algorithm for the Medical Field*. Electronics, 12(24), 5045. DOI: 10.3390/electronics12245045.

15. Zhou, Y., & Xu, Z. (2021). Redundancy strategies in avionics systems for fault tolerance. *Aerospace Science and Technology*, 116, 106851. DOI: 10.1016/j.ast.2021.106851.

16. Li, F., & Zhao, H. (2022). Synchronization mechanisms in hot-standby redundancy for industrial control systems. *Journal of Manufacturing Systems*, 62, 516–526. DOI: 10.1016/j.jmsy.2021.12.003.

17. Zeyi Liu, Songqiao Hu, & Xiao He (2023). *Real-time safety assessment of dynamic systems in non-stationary environments: A review of methods and techniques*. arXiv. Retrieved April 25, 2023.

18. Wang, D., & Zhang, T. (2022). N-modular redundancy with majority voting for safety-critical applications: Design and evaluation. *IEEE Transactions on Reliability*, 71(4), 1421–1434. DOI: 10.1109/TR.2021.3118246.

19. Park, J., & Choi, K. (2023). Cold and hot standby redundancy in real-time embedded systems: A performance comparison. *Microelectronics Reliability*, 139, 114869. DOI: 10.1016/j.microrel.2023.114869.

20. Tinglue Wang et al. (2025). *FlexStep: Enabling flexible error detection in multi/many-core real-time systems*. arXiv. Retrieved March 18, 2025.