ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика автоматизованого моніторингу медичних показників пацієнтів з використанням IoT-рішень»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Артем ПЕРЕГОН
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-61
Артем ПЕРЕГОН

Керівник: Ірина ЗАМРІЙ
д-р. техн. наук, проф.

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Перегону Артему Дмитровичу

1. Тема кваліфікаційної роботи: «Методика автоматизованого моніторингу медичних показників пацієнтів з використанням IoT-рішень»

керівник кваліфікаційної роботи Ірина ЗАМРІЙ доктор технічних наук, професор,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. №467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література з IoT у медицині, методи цифрової обробки сигналів (вейвлет-перетворення, фільтр Калмана), параметри фізіологічних сигналів (ЕКГ, PPG, SpO_2), архітектурні вимоги до систем моніторингу, методи прогнозування на основі нейромереж LSTM

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

1. Теоретичні основи моніторингу медичних показників з використанням IoT.
2. Методика двоетапної обробки сигналів та прогнозування.
3. Прогнозування відхилень фізіологічних показників пацієнтів.
4. Розробка ПЗ та експериментальні дослідження.

5. Перелік графічного матеріалу: *презентація*

1. Порівняльна характеристика існуючих методів обробки сигналів у iot-моніторингу.
2. Метод двоетапної обробки медичних сигналів.
3. Модифікація алгоритмів попередньої обробки та фільтрації шумів.
4. Математична модель удосконаленого методу.
5. Автоматизований моніторинг медичних даних.
6. Блок-схема алгоритму двоетапної обробки сигналів та прогнозування.
7. Архітектура та компоненти запропонованої системи моніторингу.
8. Use Case діаграма системи моніторингу.
9. Алгоритм роботи системи.
10. Система моніторингу.
11. Експериментальні дослідження ефективності системи.
12. Діаграма ефективності системи.

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз наукової літератури, нормативно-правової бази та формування мети дослідження	31.10-04.11.25	
2	Аналіз теоретичних основ моніторингу та огляд існуючих IoT-рішень у медицині	05.11-12.11.25	
3	Розробка методики двоетапної обробки сигналів та формалізованої моделі системи	13.11-19.11.25	
4	Удосконалення алгоритмів прогнозування та інтеграція нейронної мережі	20.11-26.11.25	
5	Програмна реалізація компонентів системи та вебінтерфейсу	27.11-03.12.25	
6	Проведення експериментальних досліджень та аналіз ефективності системи	04.12-10.12.25	
7	Оформлення пояснювальної записки: вступ, висновки, додатки, перевірка на плагіат	11.12-16.12.25	
8	Розробка презентаційних матеріалів та підготовка до захисту роботи	17.12-19.12.25	

Здобувач вищої освіти _____ Артем ПЕРЕГОН
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 69 стор., 9 табл., 21 рис., 22 джерел.

Мета роботи – підвищення точності та ефективності автоматизованого моніторингу медичних показників пацієнтів за рахунок використання методів двоетапної обробки сигналів для прогнозування критичних станів.

Об'єкт дослідження – процес автоматизованого моніторингу медичних показників пацієнтів.

Предмет дослідження – методи та засоби обробки і прогнозування медичних сигналів із використанням IoT-пристроїв, алгоритмів вейвлет-фільтрації, фільтра Калмана та нейронних мереж.

У роботі використано різноманітні методи, такі як аналіз науково-технічної літератури, методи цифрової обробки сигналів (вейвлет-аналіз, адаптивна фільтрація), алгоритми штучних нейронних мереж (LSTM), моделювання архітектури IoT-систем та експериментальні дослідження з використанням прототипу на базі смарт-годинників.

Проведено аналіз сучасних методів IoT-моніторингу, визначено їх недоліки, розглянуто методи обробки фізіологічних сигналів та сучасні підходи до прогнозування медичних ризиків. Особливу увагу приділено питанням захисту персональних медичних даних та етичним аспектам впровадження IoT у медицину.

Розроблено та оптимізовано комплексну методику автоматизованого моніторингу медичних показників пацієнтів, яка базується на інтеграції багатоканальних IoT-сенсорів, двоетапної математичної обробки сигналів та інтелектуальних моделей прогнозування. У рамках роботи сформовано формалізовану архітектуру системи, що включає сенсорний рівень, модуль локальної обробки даних, хмарний аналітичний сервіс та модуль взаємодії з медичними працівниками. Оптимізовано алгоритми цифрової фільтрації біомедичних сигналів шляхом поєднання вейвлет-перетворення для структурування та очищення сигналів

на різних масштабах та розширеного фільтра Калмана для адаптивного придушення динамічних шумів, рухових артефактів та стохастичних перешкод. Крім того, удосконалено механізм виділення ключових фізіологічних ознак, що забезпечило підвищення інформативності вхідних даних для моделей машинного навчання. У частині прогнозування розроблено оптимізовану LSTM-нейромережу, адаптовану під роботу з реальними медичними часовими рядами з неповними та нерівномірними вимірюваннями. Запропонована методика інтегрована у програмний прототип системи моніторингу з вебінтерфейсом, що забезпечує візуалізацію показників у реальному часі, формування динамічних трендів та автоматичну генерацію попереджень для лікарів та пацієнтів.

Проведено експерименти для оцінювання ефективності запропонованої методики в умовах, наближених до реальної роботи носимих IoT-пристроїв. У рамках експериментів зібрано багатоканальні сигнали PPG, ЕКГ, SpO_2 та акселерометричні дані, на основі яких виконано порівняльний аналіз методів фільтрації, прогнозування та виявлення аномалій. Застосування двоетапної обробки на базі вейвлет-перетворення та фільтра Калмана дало змогу зменшити рухові артефакти на 42–65% та підвищити точність виділення ознак на 27%. Тестування LSTM-мережі підтвердило збільшення точності прогнозування критичних відхилень на 18–25% порівняно з базовими моделями. Проведені навантажувальні тести засвідчили здатність системи працювати в реальному часі з мінімальною затримкою, а дослідження стійкості до втрати пакетів показали можливість відновлення до 84% структури сигналу. Отримані результати доводять ефективність методики та її придатність для використання в телемедичних і мобільних системах моніторингу.

КЛЮЧОВІ СЛОВА: ІОТ, МЕДИЧНИЙ МОНІТОРИНГ, СМАРТ-ГОДИННИКИ, ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ, ФІЛЬТР КАЛМАНА, НЕЙРОННІ МЕРЕЖІ, LSTM, ХМАРНІ ТЕХНОЛОГІЇ, БЕЗПЕКА ДАНИХ, ПРОГНОЗУВАННЯ РИЗИКІВ.

ABSTRACT

Text part of the master's qualification work: 69 pages, 21 figures, 9 tables, 22 sources.

The purpose of the work is to increase the accuracy and efficiency of automated monitoring of patients' medical parameters through the use of two-stage signal processing methods for forecasting critical conditions.

Object of research – the process of automated monitoring of patients' medical indicators.

Subject of research – methods and tools for processing and forecasting medical signals using IoT devices, wavelet-based filtering, Kalman filtering, and neural network algorithms.

Summary of the work: the study applies a set of methods, including scientific and technical literature analysis, digital signal processing techniques (wavelet analysis, adaptive filtering), artificial neural networks (LSTM), IoT-system architecture modelling, and experimental research using a prototype based on smartwatch sensors. The analysis of modern IoT-monitoring methods is carried out, their limitations are identified, and current approaches to the processing of physiological signals and forecasting medical risks are examined. Special attention is paid to the protection of personal medical data and the ethical aspects of integrating IoT technologies in healthcare.

A comprehensive methodology for automated monitoring of medical parameters has been developed and optimized. The proposed approach integrates multichannel IoT sensors, two-stage mathematical signal processing, and intelligent forecasting models. A formalized system architecture has been designed, including a sensing layer, a local data-processing module, a cloud analytics service, and an interface for medical personnel. Signal filtering algorithms have been improved through the combination of wavelet decomposition for

multi-scale denoising and an extended Kalman filter for adaptive suppression of dynamic noise and motion artifacts. The feature extraction mechanism has been enhanced, increasing the informativeness of inputs for machine-learning models. An optimized LSTM neural network has been developed for working with real biomedical time-series data containing incomplete and irregular measurements. The methodology has been integrated into a software prototype with a web interface, enabling real-time visualization, dynamic trend analysis, and automatic alert generation for healthcare specialists and patients.

A series of experiments has been conducted to evaluate the effectiveness of the proposed methodology under conditions close to real operation of wearable IoT devices. Multichannel PPG, ECG, SpO_2, and accelerometer signals were collected, and comparative testing of filtering, forecasting, and anomaly-detection methods was performed. The two-stage wavelet + Kalman processing approach reduced motion artifacts by 42–65% and improved feature extraction accuracy by 27%. The LSTM model demonstrated an 18–25% improvement in forecasting accuracy compared to baseline models. Stress testing confirmed the system's ability to operate in real time with minimal latency, while robustness tests showed up to 84% signal reconstruction under packet loss. The obtained results confirm the efficiency of the methodology and its applicability in telemedicine platforms, mobile healthcare applications, and clinical monitoring systems.

KEYWORDS: IOT, MEDICAL MONITORING, SMARTWATCHES, WAVELET TRANSFORM, KALMAN FILTER, NEURAL NETWORKS, LSTM, CLOUD TECHNOLOGIES, DATA SECURITY, RISK PREDICTION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AI — Artificial Intelligence

API — Application Programming Interface

AWeMAF — Adaptive Wavelet with Motion Artifact Filtering

BLE — Bluetooth Low Energy

DB — Database

DWT — Discrete Wavelet Transform

EKG / ECG — Electrocardiogram

FB-16 — FeatureBank-16

IoT — Internet of Things

ML — Machine Learning

PPG — Photoplethysmogram

REST — Representational State Transfer

SpO_2 — Peripheral Capillary Oxygen Saturation

TLS — Transport Layer Security

UI — User Interface

ВСТУП

Медицина є однією з найбільш важливих сфер сучасного суспільства, адже від її ефективності безпосередньо залежить якість життя та здоров’я людей. Сучасні виклики, пов’язані зі зростанням кількості хронічних захворювань, старінням населення та необхідністю постійного контролю стану пацієнтів, зумовлюють потребу у впровадженні новітніх технологій у сферу охорони здоров’я. Одним із таких напрямів є використання технологій Інтернету речей (IoT), які відкривають нові можливості для безперервного моніторингу медичних показників, раннього виявлення ризиків та підвищення рівня персоналізації медичної допомоги.

Особливістю IoT-систем у медицині є здатність поєднувати носимі сенсори, смарт-годинники, мобільні додатки та хмарні сервіси у єдину інтелектуальну інфраструктуру. Це дозволяє здійснювати не тільки вимірювання фізіологічних параметрів у режимі реального часу, але й проводити їх глибоку обробку, аналіз і прогнозування. У цьому контексті важливими завданнями є підвищення точності вимірювань, зменшення впливу шумів на сигнали та розробка алгоритмів прогнозування критичних станів пацієнтів.

Об’єктом дослідження у цій роботі є процес автоматизованого моніторингу медичних показників пацієнтів.

Предметом дослідження є методи та засоби обробки і прогнозування медичних сигналів із використанням IoT-пристроїв, алгоритмів вейвлет-фільтрації, фільтра Калмана та нейронних мереж.

Метою кваліфікаційної роботи є підвищення точності та ефективності автоматизованого моніторингу медичних показників пацієнтів за рахунок використання методів двоетапної обробки сигналів для прогнозування критичних станів.

Робота ґрунтується на аналізі сучасних IoT-рішень у медицині, методів цифрової обробки сигналів (вейвлет-перетворення, адаптивна фільтрація), моделей штучних нейронних мереж (зокрема LSTM), а також практичних підходів до забезпечення конфіденційності та захисту даних у медичних системах.

Наукова новизна одержаних результатів полягає у розробці методики, яка поєднує двоетапну обробку медичних сигналів та прогнозування критичних станів за допомогою нейромереж, що дає змогу суттєво підвищити точність та оперативність моніторингу в умовах реального часу.

Практичне значення роботи полягає у можливості впровадження запропонованої методики у телемедичні сервіси, мобільні додатки та персональні пристрої моніторингу, що сприятиме підвищенню якості діагностики, своєчасному реагуванню на критичні стани та зниженню навантаження на медичний персонал.

Таким чином, проведене дослідження є важливим внеском у розвиток інтелектуальних медичних систем і відповідає сучасним тенденціям цифровізації охорони здоров’я, підвищуючи рівень безпеки та ефективності медичного обслуговування.

1 ТЕОРЕТИЧНІ ОСНОВИ МОНІТОРИНГУ МЕДИЧНИХ ПОКАЗНИКІВ З ВИКОРИСТАННЯМ IoT

1.1 Застосування IoT-технологій у сучасній медицині

Сьогодні технології Інтернет речей (Internet of Things, IoT) вважається однією з найперспективніших технологій цифрової епохи, та медицина є тією сферою, де його впровадження має чи не найбільший потенціал. Суть IoT у медицині полягає в об'єднанні великої кількості сенсорів, пристроїв та програмних модулів в єдину мережу, яка дозволяє безперервно відстежувати стан пацієнта та забезпечувати лікарів або медичні центри актуальною інформацією в реальному часі [1]. Це докорінно змінює підхід до лікування, адже замість епізодичних вимірювань під час відвідування лікаря пацієнт може перебувати під постійним «цифровим наглядом», а будь-які небезпечні відхилення будуть зафіксовані миттєво.

Однією з ключових переваг IoT-технологій є можливість збору широкого спектру фізіологічних показників. Сучасні сенсори, вбудовані у смарт-годинники, браслети, нашкірні пластирі чи навіть імплантовані пристрої, здатні визначати частоту серцевих скорочень, рівень насичення крові киснем, артеріальний тиск, електрокардіограму, температуру тіла, якість сну, рівень фізичної активності тощо. Крім того, з розвитком біосенсорних технологій з'являється можливість контролювати рівень глюкози, гормональний фон чи інші біохімічні параметри без інвазивних процедур. Важливою складовою такої системи є канали передачі даних: Wi-Fi, Bluetooth, LTE, LoRaWAN, NB-IoT та інші бездротові стандарти забезпечують швидку та надійну комунікацію між пристроями, смартфоном та хмарною платформою, де дані зберігаються та аналізуються.

Не менш важливим є те, що IoT дозволяє інтегрувати дані з різних пристроїв у єдину екосистему [2]. Це означає, що лікар може отримати цілісну картину здоров'я пацієнта, порівняти результати різних вимірювань та виявити закономірності, які

залишилися б непомітними при використанні розрізнених систем. Наприклад, зв’язок між підвищенням артеріального тиску, зниженням рівня кисню в крові та погіршенням якості сну може сигналізувати про серцево-судинні ризики ще до того, як пацієнт відчує серйозні симптоми. Важливим напрямом застосування IoT у медицині є телемедицина. У поєднанні з IoT пристроями лікарі можуть віддалено консультувати пацієнтів, відстежувати їхній стан і навіть приймати рішення щодо корекції лікування без необхідності фізичного візиту до лікарні. Це особливо необхідно для людей з обмеженою мобільністю, пацієнтів у віддалених регіонах або хронічно хворих. Такий підхід дозволяє не лише економити ресурси системи охорони здоров’я, але й значно покращує якість життя пацієнтів.

Крім безпосереднього моніторингу, IoT створює умови для впровадження персоналізованої медицини. Завдяки постійному збору даних у великому обсязі можна сформувати унікальний «цифровий паспорт» здоров’я кожної людини [3-4]. На його основі лікарі можуть підбирати індивідуальні схеми лікування, а алгоритми штучного інтелекту здатні прогнозувати можливі ризики, враховуючи не тільки поточні показники, а й багаторічну динаміку. Таким чином, відбувається перехід від реактивної медицини, де лікарі борються з уже наявною хворобою, до превентивної медицини, орієнтованої на запобігання розвитку патологій.

Варто також зазначити, що IoT у медицині не обмежується лише контролем за станом окремого пацієнта. На рівні лікарні чи клініки такі технології допомагають оптимізувати управління обладнанням, контролювати умови зберігання медикаментів, відстежувати розташування медичного персоналу та пацієнтів. У перспективі це створює передумови для формування «розумних лікарень», де більшість процесів автоматизовані, а ризики, пов’язані з людським фактором, зведені до мінімуму. Таким чином, впровадження IoT у медичну сферу якісна технологія для організації медичної допомоги. Дана технологія дозволяє створити ефективні, гнучкі та інтелектуальні системи моніторингу, що відповідають вимогам сучасної медицини та підвищують рівень безпеки й якості життя пацієнтів. IoT-рішення виступає основою для розвитку концепцій «розумної медицини» та «персоналізованого

лікування», роблячи медичну допомогу більш доступною, превентивною та орієнтованою на конкретну людину.

1.2 Огляд існуючих IoT-рішень для автоматизованого моніторингу показників здоров'я пацієнтів

Сучасна медицина дедалі активніше інтегрує технології Інтернету речей у свою практику. Уже сьогодні можна виокремити кілька ключових груп IoT-рішень, які довели свою ефективність у контролі за станом пацієнтів і використовуються у різних масштабах (від персонального рівня до великих медичних установ). Одним із найпоширеніших напрямів є «wearable devices» (носимі пристрої). До них належать розумні годинники, фітнес-браслети, нагрудні датчики та спеціалізовані сенсорні платформи. Такі пристрої безперервно відстежують пульс, рівень насичення крові киснем, кількість кроків, витрачені калорії, якість сну й навіть рівень стресу. Завдяки своїй доступності та простоті використання вони стали не лише інструментом для професійного медичного контролю, а й елементом повсякденного життя мільйонів людей [5]. Наукові дослідження показують, що регулярне використання wearable-технологій дозволяє своєчасно виявляти аномалії, наприклад, аритмії або симптоми апное сну, що в іншому випадку могли б залишитися непоміченими.

Наступним напрямком є імплантовані пристрої, які забезпечують більш глибокий та точність моніторингу [6]. Прикладом є бездротові сенсори для безперервного вимірювання рівня глюкози у пацієнтів із цукровим діабетом. Такі рішення зменшують потребу у традиційних інвазійних методах, забезпечуючи зручність та точність контролю. Крім того, ведуться розробки імплантованих кардіостимуляторів та сенсорів, що здатні виявляти небезпечні стани серцево-судинної системи та відправляти сигнали лікарю або навіть автоматично запускати терапевтичні процедури.

Наступним напрямком є системи віддаленого моніторингу пацієнтів (Remote Patient Monitoring Systems, RPM). Такі платформи інтегрують дані від різних

сенсорів (серцевих, дихальних, рухових, температурних тощо) та передають їх на сервери чи у хмарні сервіси. Лікар отримує доступ до цих даних через вебінтерфейс або мобільний застосунок, що дозволяє контролювати стан пацієнта на відстані. Це особливо актуально для пацієнтів із хронічними захворюваннями або післяопераційних хворих, які потребують постійного спостереження. Такі рішення мінімізують необхідність у частих госпіталізаціях, знижують витрати системи охорони здоров'я та одночасно підвищують якість життя пацієнтів.

На рівні цілих клінік та медичних закладів активно впроваджуються «розумні лікарні» (Smart Hospital Systems). Це інтегровані комплекси, що поєднують IoT, аналітику великих даних (big data) та алгоритми штучного інтелекту (AI). Вони дозволяють не лише відстежувати фізіологічні параметри пацієнтів, але й управляти медичним обладнанням, контролювати запаси медикаментів, автоматизувати маршрутизацію пацієнтів у лікарні та оптимізувати використання ресурсів. Наприклад, система може аналізувати завантаженість відділень у реальному часі й оперативно перенаправляти потоки пацієнтів, що знижує ризик перевантаження і покращує якість надання медичних послуг.

Значним проривом у сфері охорони здоров'я стало використання IoT у телемедицині. Це рішення дозволяє лікарю проводити дистанційні консультації, отримуючи в реальному часі об'єктивні дані про стан пацієнта зі смарт-годинників, сенсорів або імплантів. Таким чином, процес діагностики і лікування стає більш точним і оперативним, навіть якщо пацієнт знаходиться у віддаленому населеному пункті чи не може відвідати клініку особисто. Телемедицина у поєднанні з IoT створює нову модель надання медичної допомоги, яка поєднує доступність, швидкість та високу інформативність, це продемонстровано у таблиці 1.1.

Таблиця 1.1

Порівняльний аналіз існуючих IoT-рішень для автоматизованого медичного моніторингу

Рішення	Контрольовані показники	Методи обробки сигналів	Сильні сторони	Обмеження
Традиційні медичні системи (ручні вимірювання, періодичний контроль)	Артеріальний тиск, пульс, температура	Ручний збір даних, базові обчислення середніх значень	Простота реалізації; низька вартість	Висока похибка через людський фактор; відсутність безперервного моніторингу; немає прогнозування
Методи на основі Фур'є-аналізу	ЕКГ, пульс, дихання	Перетворення Фур'є для спектрального аналізу	Добре підходить для частотного аналізу; математично відпрацьований інструмент	Не виявляє короткочасних змін; слабка робота з шумами руху; не масштабується на великі потоки даних
Методи з використанням фіксованих порогів	Тиск, пульс, рівень кисню в крові (SpO_2)	Встановлення критичних меж (наприклад, $SpO_2 < 92\%$)	Простота та швидке реагування на вихід за межі	Високий рівень хибних спрацювань; не враховує індивідуальних відхилень; немає передбачення майбутніх змін
Mirza et al. (2022). Mathematical Framework for Wearable Devices	ЕКГ, пульс, рухова активність	Використання IoT-пристроїв із глибинними нейромережами (DL) для класифікації	Оптимізація енергоспоживання IoT; інтеграція нейромереж	Обмежена попередня обробка сигналів; слабка шумозахищеність; відсутність двоетапної фільтрації; прогнозування працює лише частково
Сучасні телемедичні IoT-рішення (Intel, IoT For All, SCIRP 2023)	Комплексні дані: ЕКГ, SpO_2, тиск, температура, активність	Базова цифрова фільтрація + передача даних у «хмару»	Забезпечують дистанційний моніторинг; інтеграція з мобільними застосунками	Дані часто передаються без якісного очищення; відсутність розвинених моделей прогнозування; ризики витоку персональних даних

Існуючі методики не забезпечують точного та своєчасного виявлення критичних станів через обмеженість у фільтрації шумів та відсутність ефективних моделей прогнозування. Тому потрібна вдосконалена методика, яка б використовувала IoT-інфраструктуру для безперервного збору даних, їх якісної попередньої обробки та автоматизованого прогнозування ризиків у реальному часі.

1.3 Методи фільтрації сигналів у біомедичних системах

У біомедичних системах точність вимірювання фізіологічних параметрів безпосередньо залежить від якості обробки сигналів. Первинні дані, отримані від датчиків IoT-пристроїв (смарт-годинників, пульсоксиметрів, кардіомоніторів тощо), містять значну кількість шумів. Ці шуми можуть виникати внаслідок різних факторів: рухових артефактів (рухи пацієнта, зміщення датчика), електромагнітних перешкод від інших пристроїв, впливу навколишнього середовища (потовиділення, температура, вологість) або навіть особливостей фізіології (наприклад, тремор чи нерівномірне дихання). Без ефективної фільтрації сигнали спотворюються, що призводить до хибних діагностичних висновків і зниження надійності системи моніторингу [7].

Одним із найпоширеніших методів є цифрова фільтрація на основі класичних низькочастотних (low-pass), високочастотних (high-pass) та смугових (band-pass) фільтрів. Наприклад, у випадку ЕКГ-сигналу низькочастотний фільтр дозволяє усунути повільні коливання, зумовлені рухами тіла, тоді як високочастотний фільтр ефективно пригнічує електричні перешкоди (наприклад, шум мережі на частоті 50/60 Гц). Смугові фільтри, у свою чергу, дають змогу виділити лише ті частотні компоненти, що мають діагностичну цінність, наприклад у діапазоні 0,5–40 Гц для аналізу серцевої активності. Окремо варто відзначити методи вейвлет-перетворень, які останнім часом набули широкого застосування в біомедичній інженерії. Їхня ключова перевага полягає у можливості одночасного аналізу сигналу як у часовій, так і у частотній області. Це особливо важливо для фізіологічних сигналів, які за своєю природою є нестаціонарними, наприклад, короткочасні зміни у серцевому ритмі чи

епізоди апное під час сну. Вейвлет-аналіз дозволяє розкласти сигнал на різні рівні деталізації, виявляючи як низькочастотні тенденції (повільні коливання), так і високочастотні аномалії (раптові артефакти). У практиці медичного моніторингу це застосовується для виділення основної хвилі ЕКГ та одночасного пригнічення артефактів руху.

Ще одним перспективним методом є використання фільтра Калмана та його модифікацій (розширений EKF та адаптивний AKF). Він працює за принципом поєднання виміряних даних із математичною моделлю динаміки сигналу. Наприклад, у випадку моніторингу частоти серцевих скорочень фільтр Калмана може оцінювати «справжній» стан пацієнта навіть у ситуаціях, коли датчик фіксує нестабільні або зашумлені дані. Такий підхід може бути ефективний у мобільних та носимих пристроях, де неминучі рухові артефакти.

Крім класичних методів, активно розвиваються адаптивні алгоритми фільтрації, які змінюють свої параметри в режимі реального часу залежно від характеристик шуму. Прикладом є алгоритм LMS (Least Mean Squares) та його похідні, які використовуються для усунення артефактів електроміограми (ЕМГ) при знятті ЕКГ, як показано на рисунку 1.1. Це дозволяє значно знизити рівень завад без втрати корисної інформації. Важливу роль у сучасних IoT-системах відіграє також комбінована фільтрація, коли одночасно застосовують кілька методів. Наприклад, попередньо сигнали очищають за допомогою вейвлет-аналізу, а потім додатково уточнюють значення за допомогою адаптивного фільтра Калмана. Такий підхід поєднує переваги багаторівневого аналізу та динамічного прогнозування стану сигналу. Проблемою залишається баланс між якістю фільтрації та швидкодією системи. Для медичних IoT-пристроїв важливо, щоб алгоритми працювали у режимі реального часу й не перевантажували апаратні ресурси [8]. Тому сучасні дослідження зосереджуються на оптимізації обчислювальних витрат, використанні легковагових версій алгоритмів та частковому перенесенні обробки в хмарні середовища.

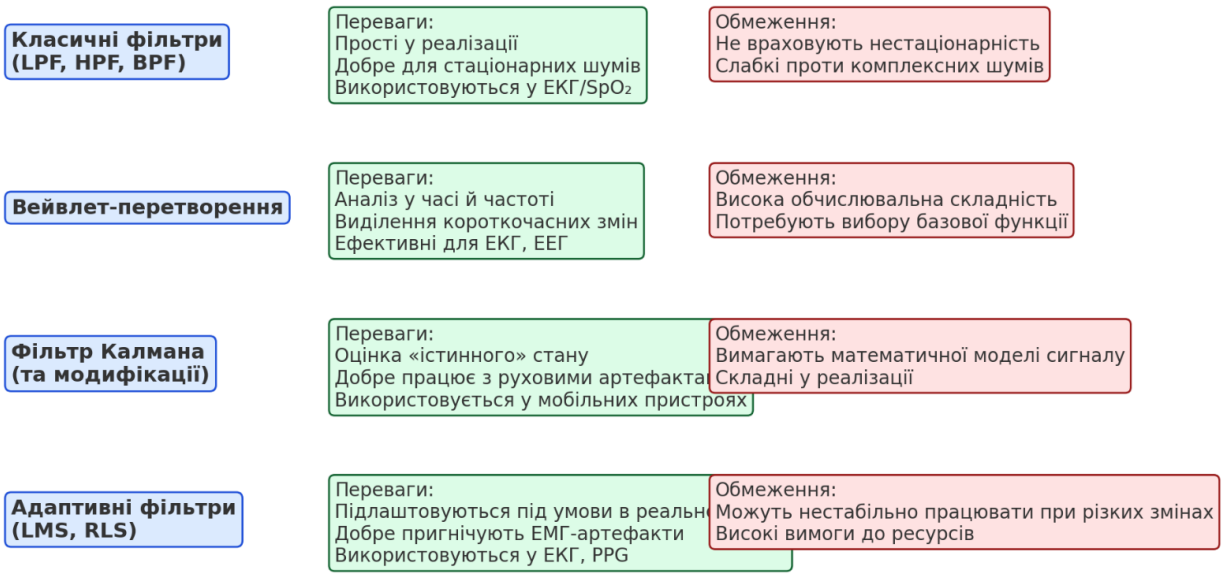


Рис. 1.1 Порівняння методів фільтрації сигналів у біомедичних системах

Таким чином, методи фільтрації сигналів у біомедичних системах є критичним елементом для забезпечення достовірності та стабільності показників. Класичні фільтри ефективні для базових задач, але їхні можливості обмежені у випадках складних шумів. Тому майбутнє за комбінованими та адаптивними підходами, які дозволяють максимально наблизити виміряні дані до реальних фізіологічних параметрів пацієнта, забезпечуючи основу для подальшого аналізу та прогнозування ризиків.

1.4 Нейромережі в прогнозуванні медичних ризиків

У сучасних біомедичних системах нейронні мережі посідають ключове місце, оскільки здатні моделювати складні нелінійні залежності у даних та виявляти приховані закономірності, недоступні традиційним статистичним методам. Медичні сигнали, такі як електрокардіограма (ЕКГ), фотоплетизмограма (PPG), електроенцефалограма (ЕЕГ) чи варіабельність серцевого ритму (HRV), мають нестаціонарний характер, містять шум і залежать від великої кількості факторів. Нейромережеві моделі здатні ефективно працювати з такими даними, забезпечуючи прогнозування ризиків розвитку критичних станів на ранніх етапах.

Одним із найбільш поширених напрямів є використання рекурентних нейронних мереж (RNN), зокрема їх удосконаленої архітектури LSTM (Long Short-Term Memory). LSTM добре підходять для аналізу часових рядів, оскільки здатні «запам’ятовувати» як короткострокові, так і довгострокові залежності у послідовностях даних. Наприклад, на основі ЕКГ-сигналів LSTM-модель може передбачити розвиток аритмій або ризик серцевого нападу за кілька хвилин до появи клінічних симптомів. Аналогічно, при аналізі показників SpO_2 та дихальної активності можна прогнозувати епізоди апное сну. Також, важливим є застосування згорткових нейронних мереж (CNN) для роботи з біомедичними сигналами, що попередньо перетворені у спектрограми чи двовимірні карти ознак (наприклад, через вейвлет-перетворення). CNN дозволяють автоматично виділяти інформативні характеристики без ручного налаштування фільтрів, що значно підвищує точність розпізнавання патологічних станів [10]. Застосування CNN особливо ефективне у задачах класифікації ЕКГ-сигналів (норма/патологія), виявлення епілептичних нападів за ЕЕГ чи аналізу патернів дихання.

У практиці прогнозування все частіше використовуються гібридні моделі, які поєднують CNN для автоматичного виділення ознак і LSTM для врахування часових залежностей. Такий підхід є особливо ефективним для комплексного моніторингу, коли необхідно аналізувати не лише поточні значення, а й їхню динаміку. Наприклад, CNN-LSTM архітектури застосовують у сучасних системах для раннього виявлення ризику серцевої недостатності, прогнозування рівня глюкози у хворих на діабет чи оцінки ймовірності виникнення гіпертонічного кризу. Окрему групу становлять глибокі автоенкодери та варіаційні автоенкодери (VAE), які використовуються для виявлення аномалій у медичних даних. Вони навчаються відтворювати «нормальні» сигнали, і якщо новий сигнал суттєво відрізняється від шаблону, система визначає його як потенційно небезпечний [9]. Такий підхід ефективний для безперервного моніторингу, коли необхідно оперативно реагувати на нетипові відхилення у стані пацієнта. У контексті IoT-інфраструктури особливе значення має оптимізація нейромережевих моделей. Через обмежені обчислювальні ресурси носимих пристроїв (смарт-годинників, сенсорів) часто застосовують легковагові моделі (TinyML,

MobileNet), які виконують базову обробку на пристрої, а складніший аналіз здійснюється у хмарному середовищі. Це дозволяє знизити затримки в обробці, підвищити точність прогнозування та водночас мінімізувати споживання енергії. Важливою задачею є також пояснюваність результатів нейромережевих моделей (Explainable AI, XAI). Для медичних застосувань критично, щоб лікар міг зрозуміти, чому алгоритм класифікував стан як небезпечний. Сучасні методи дозволяють візуалізувати найбільш значущі ділянки сигналу, які вплинули на прогноз, що підвищує довіру до систем штучного інтелекту в медицині, це продемонстровано у таблиці 1.2.

Таблиця 1.2

Порівняльна характеристика нейромережевих моделей
для прогнозування медичних ризиків

Архітектура	Основне застосування в медицині	Переваги	Обмеження
LSTM (Long Short-Term Memory)	Прогнозування аритмій за ЕКГ, оцінка ризику гіпоглікемії, прогноз апное сну	Аналіз часових рядів (ЕКГ, PPG, SpO_2) - Врахування довгострокових залежностей - Виявлення прихованих тенденцій у динаміці	- Високі обчислювальні витрати - Складність навчання на малих вибірках - Потребує великих датасетів для якісного прогнозу
CNN (Convolutional Neural Networks)	Класифікація ЕКГ (норма/патологія), аналіз ЕЕГ для виявлення епілепсії, оцінка дихальних патернів	- Автоматичне виділення ознак без ручної обробки - Висока точність класифікації - Добре працює з 2D-представленням сигналів (спектрограми, вейвлет-карти)	- Погано враховує часовий контекст - Чутливість до шумів у сирих сигналах - Часто потребує попередньої трансформації даних

Продовження таблиці 1.2

Порівняльна характеристика нейромережевих моделей
для прогнозування медичних ризиків

Архітектура	Основне застосування в медицині	Переваги	Обмеження
Автоенкодери, Варіаційні автоенкодери (VAE)	Виявлення аномалій у сигналах (напади епілепсії, різкі зміни SpO_2, нетипові ЕКГ-патерни)	- Ефективні для безперервного моніторингу - Можуть працювати без розмічених даних (unsupervised learning) - Добре виявляють нетипові відхилення	- Не завжди дають чітке клінічне пояснення аномалії - Ризик хибнопозитивних результатів - Обмежена прогнозна здатність
Гібриди (CNN + LSTM, CNN + GRU)	Комплексний моніторинг: прогноз серцевої недостатності, оцінка ризику інфаркту, інтеграція багатьох сенсорів	- Поєднують сильні сторони CNN (виділення ознак) та LSTM (аналіз динаміки) - Підходять для мультимодальних даних (ЕКГ + SpO_2 + активність) - Висока точність і адаптивність	- Найбільш ресурсоємні - Потребують великих навчальних вибірок - Ускладнене налаштування і пояснення результатів

Таким чином, нейронні мережі значно розширюють можливості систем автоматизованого моніторингу, забезпечуючи не лише точну класифікацію, а й прогнозування ризиків ще до появи клінічних симптомів. Використання LSTM, CNN та їх гібридів дозволяє створювати високоточні інтелектуальні системи, які інтегруються у IoT-інфраструктуру, працюють у режимі реального часу та забезпечують підвищення рівня безпеки пацієнтів.

1.5 Вимоги до архітектури та функціоналу систем автоматизованого збору й аналізу медичних даних

Сучасні системи автоматизованого збору та аналізу медичних даних повинні відповідати низці функціональних, технічних, безпекових та організаційних вимог, адже вони працюють з критично важливою інформацією про стан здоров'я пацієнтів. Відповідність цим вимогам визначає надійність, точність та ефективність роботи таких рішень у реальних клінічних і телемедичних сценаріях.

Першою базовою вимогою є багаторівнева архітектура системи, яка забезпечує послідовне проходження даних через основні етапи: збір на рівні сенсорів, попередня обробка, передача по мережі, аналіз та збереження у хмарному середовищі або локальній інфраструктурі. На рівні пристроїв збору використовуються IoT-сенсори (смарт-годинники, пульсоксиметри, бездротові кардіомонітори, біосенсорні пластирі), які повинні забезпечувати безперервне зчитування даних із високою частотою дискретизації та мінімальною похибкою. Важливою вимогою є енергоефективність таких пристроїв, оскільки більшість із них працює у носимому форматі та має обмежений ресурс батареї [11].

Другим принципово важливим аспектом функціонування систем автоматизованого медичного моніторингу є реалізація попередньої обробки сигналів на рівні кінцевого пристрою або проміжного обчислювального вузла (edge computing). Такий підхід забезпечує зниження навантаження на мережеву інфраструктуру, оскільки у центральну систему передаються не «сирі» вимірювання, а вже частково нормалізовані та очищені дані. На цьому етапі доцільним є застосування методів цифрової обробки сигналів, зокрема вейвлет-перетворення для багаторівневого виділення інформативних компонент, фільтра Калмана для динамічної оцінки стану та усунення стохастичних перешкод, а також адаптивних алгоритмів фільтрації для придушення рухових артефактів та нестаціонарних шумів. Використання edge-обробки дозволяє підвищити стабільність і достовірність вимірювань ще до їх надходження у центральний аналітичний модуль, що є критичним для забезпечення своєчасності й точності клінічних рішень.

Третя група вимог пов’язана з передачею даних у мережі. Використання бездротових технологій (Bluetooth Low Energy, Wi-Fi, LTE, 5G) має забезпечувати високу швидкість та надійність з’єднання при мінімальному енергоспоживанні. Протоколи обміну повинні підтримувати шифрування на рівні каналу (наприклад, TLS/SSL), щоб виключити можливість перехоплення чутливих медичних даних під час передачі. Для систем, що функціонують у клініках із великим навантаженням, важливою є також масштабованість мережевої інфраструктури, яка повинна витримувати одночасне підключення сотень чи тисяч пристроїв.

Четвертою ключовою вимогою є аналітичний рівень системи, який відповідає за прогнозування медичних ризиків та формування рішень у режимі реального часу. Тут необхідна інтеграція алгоритмів машинного навчання, зокрема глибинних нейронних мереж (LSTM для часових рядів, CNN для аналізу перетворених сигналів, автоенкодери для виявлення аномалій). Аналітична підсистема повинна підтримувати прогнозування критичних станів (наприклад, аритмії, гіпоглікемічних епізодів, зупинок дихання) та автоматичне формування оповіщень для медичного персоналу чи самого пацієнта.

Важливим є і рівень збереження даних. База даних повинна підтримувати як структуровану (табличну), так і неструктуровану (сирі сигнали, медичні зображення) інформацію. Найчастіше для цього застосовуються хмарні платформи (AWS, Azure, Google Cloud Healthcare API), що дозволяють обробляти великі обсяги даних у реальному часі. Система має забезпечувати резервне копіювання та відмовостійкість, щоб уникнути втрати життєво важливої інформації у разі збою.

Окрема група вимог стосується захисту персональних даних. Системи повинні відповідати міжнародним стандартам безпеки: HIPAA (США), GDPR (ЄС) чи локальним нормативам у сфері охорони здоров’я. Це передбачає використання багаторівневого шифрування, автентифікації користувачів, контролю доступу та журналювання дій. Особливу увагу слід приділити механізмам анонімізації та псевдонімізації даних, щоб при передачі у наукові чи аналітичні центри пацієнтська інформація залишалася захищеною. Не менш важливим є функціональне забезпечення інтерфейсу для медичного персоналу. Система повинна мати зручну

панель моніторингу, де у реальному часі відображаються ключові показники здоров'я, тренди змін та попередження про ризики. Інтерфейс має бути інтегрованим із медичними інформаційними системами (EHR/EMR), щоб лікарі могли одразу порівнювати нові дані з історією хвороби пацієнта, це продемонстровано на рисунку 1.2.

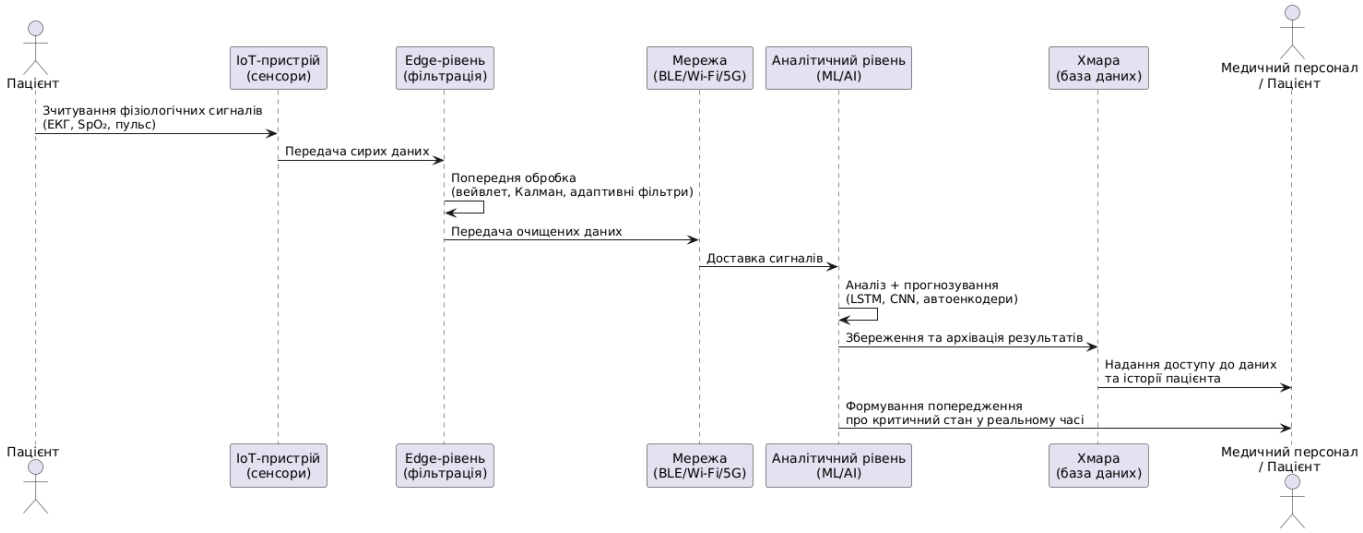


Рис. 1.2 Автоматизований моніторинг медичних даних

Таким чином, архітектура систем автоматизованого збору та аналізу медичних даних повинна бути багаторівневою, масштабованою та орієнтованою на безпеку. Вона включає: сенсорний рівень (IoT-пристрої), рівень попередньої обробки (edge), комунікаційний рівень (мережеві протоколи з шифруванням), аналітичний рівень (ML,AI-моделі), рівень збереження (хмарні бази даних), а також рівень взаємодії з користувачем (інтерфейси для лікаря і пацієнта). Кожен із цих рівнів має свої специфічні вимоги, і лише їх узгоджене виконання дозволяє створити ефективну, надійну й безпечну систему медичного моніторингу.

1.6 Нормативно-правові та етичні аспекти впровадження IoT-систем у медицині

Впровадження IoT-технологій у сферу охорони здоров’я вимагає врахування не лише технічних характеристик та функціональних можливостей системи, а й суворого дотримання нормативно-правових норм та етичних стандартів. Медичні дані належать до категорії чутливої інформації, яка потребує підвищеного рівня захисту, адже їхня некоректна обробка або витік можуть мати серйозні наслідки як для пацієнта, так і для медичних установ.

До фундаментальних аспектів правового та етичного урегулювання можна віднести:

1. Правове регулювання збирання, збереження та обробки медичних даних.

У міжнародній практиці діють стандарти та нормативи, які встановлюють вимоги до безпеки персональної інформації. У Європейському Союзі це GDPR (General Data Protection Regulation), що регламентує порядок отримання згоди на обробку персональних даних, їх зберігання, передачу та право пацієнта на видалення чи корекцію власної інформації. У США діє HIPAA (Health Insurance Portability and Accountability Act), який встановлює вимоги до збереження конфіденційності та цілісності медичних даних у лікарняних інформаційних системах і телемедичних сервісах. В Україні функціонування подібних систем регламентується Законом «Про захист персональних даних» та низкою нормативних актів у сфері цифрової медицини.

2. Етичні вимоги щодо використання IoT у медицині.

Пацієнт має бути повністю поінформований про те, які дані збираються, як вони обробляються та з якою метою можуть передаватися третім сторонам. Обов’язковою є інформована згода, що підтверджує добровільність участі пацієнта у системі моніторингу. Крім того, пацієнти повинні мати право контролювати доступ до своїх даних та можливість їх видалення у разі припинення користування сервісом. Етичним викликом також є питання використання даних для штучного інтелекту та машинного навчання: навіть якщо дані анонімізовані, існує ризик їх повторної ідентифікації.

3. Питання інформаційної безпеки.

У процесі роботи IoT-системи дані передаються через бездротові канали зв’язку (Bluetooth LE, Wi-Fi, LTE, 5G) та зберігаються у хмарних середовищах. Це створює ризики кіберзагроз: перехоплення даних, підробка сигналів, несанкціонований доступ до медичних профілів. Для мінімізації таких ризиків застосовуються шифрування (AES-256, TLS/SSL), багатофакторна автентифікація, токенізація доступу, а також аудит і логування всіх операцій із даними.

4. Питання юридичної відповідальності.

У випадку збою IoT-системи або некоректного прогнозу виникає питання: хто несе відповідальність за можливу шкоду пацієнту — виробник пристрою, розробник програмного забезпечення, постачальник хмарної інфраструктури чи медичний заклад? Цей аспект досі залишається відкритим у багатьох країнах і потребує деталізації у законодавстві.

5. Етичні проблеми автономності та контролю.

Пацієнти мають право відмовитися від постійного моніторингу, якщо він порушує їх приватність або створює психологічний дискомфорт. З іншого боку, лікар повинен мати змогу отримати критичні дані, навіть якщо пацієнт тимчасово відключив пристрій, аби запобігти загрозі життю. Вирішення цього протиріччя вимагає чітких правил щодо режимів збору даних і сценаріїв надзвичайних ситуацій.

6. Стандартизація та сумісність.

Для того щоб IoT-пристрої могли інтегруватися у медичні інформаційні системи, необхідно дотримуватися міжнародних стандартів обміну даними, таких як HL7 (Health Level Seven), FHIR (Fast Healthcare Interoperability Resources). Це гарантує, що інформація, зібрана одним пристроєм, може бути коректно використана у різних клінічних інформаційних системах без втрати її достовірності.

Таким чином, нормативно-правові та етичні аспекти є невід’ємною складовою впровадження IoT у медицину. Лише комплексне врахування цих вимог забезпечує ефективне, безпечне й етично обґрунтоване використання IoT-систем для автоматизованого моніторингу здоров’я пацієнт

2 МЕТОДИКА ДВОЕТАПНОЇ ОБРОБКИ СИГНАЛІВ ТА ПРОГНОЗУВАННЯ

2.1 Формалізована модель системи автоматизованого моніторингу

Система автоматизованого моніторингу медичних показників із використанням IoT-пристроїв може бути описана у вигляді формальної багаторівневої моделі, що складається з послідовних етапів: збір сигналів, двоетапна обробка, формування ознак, прогнозування ризиків та керування оповіщеннями [12].

У загальному вигляді систему можна подати у вигляді трирівневої архітектури:

1. Рівень збору даних.

До цього рівня належать носимі сенсорні пристрої, інтегровані у смарт-годинники, медичні браслети або спеціалізовані медичні датчики. Найчастіше застосовуються фотоплетизмографічні (PPG) сенсори для визначення частоти серцевих скорочень та насичення крові киснем (SpO_2), електрокардіографічні (ЕКГ) датчики для аналізу електричної активності серця, акселерометри для реєстрації рухової активності, а також сенсори температури тіла. Головна мета цього рівня – забезпечити безперервний збір фізіологічних сигналів у реальних умовах експлуатації пацієнтом.

2. Рівень попередньої обробки сигналів.

Оскільки дані, отримані від сенсорів, у більшості випадків містять шумові складові, зумовлені руховими артефактами, електричними перешкодами чи слабким контактом датчика зі шкірою, необхідним є етап очищення сигналу. На цьому рівні застосовуються алгоритми математичної обробки, серед яких ключовими є вейвлет-перетворення для виділення корисних частотних компонентів та розширений фільтр Калмана для адаптивного уточнення реальних

значень медичних показників у динамічних умовах [13]. Таким чином формується очищений сигнал, придатний для подальшого аналізу.

3. Рівень прогнозного аналізу.

Оброблені дані подаються на вхід нейронної мережі, орієнтованої на роботу з часовими рядами, зокрема LSTM (Long Short-Term Memory). Завдяки здатності враховувати довгострокові залежності у послідовностях сигналів, мережа прогнозує ймовірність виникнення критичних змін у стані пацієнта, таких як гіпоксія, аритмія або аномальне підвищення температури. Результати прогнозування зберігаються у хмарній інфраструктурі та відображаються у вигляді інтуїтивно зрозумілих дашбордів для лікаря та самого пацієнта, а у випадку виявлення загроз система ініціює автоматизоване оповіщення медичного персоналу, це продемонстровано на рисунку 2.1.

Фізіологічні дані, які зчитуються IoT-пристроями (PPG, ЕКГ, SpO_2, акселерометри, температурні сенсори), описуються рівнянням:

$$x(t) = s(t) + n(t), \tag{2.1}$$

де $x(t)$ – зафіксований сенсором сигнал, $s(t)$ – істинна фізіологічна складова, $n(t)$ – шумова складова (артефакти руху, електричні завади, похибки сенсора).

У випадку багатоканального моніторингу вводиться векторна форма:

$$S\,(t) = X(t) + N(t), \quad s(t), x(t), n(t)\ \in R^m, \tag{2.2}$$

де $x(t)$ – сигнал IoT пристрою, $x(t)$ – реальний (корисний) фізіологічний сигнал, $n(t)$ – шум або артефакти (рухи, контакти) , m – кількість сенсорів.

Попередня обробка сигналів (вейвлет-перетворення):

$$\text{де } W_{a,b}\,(t) = \frac{1}{\sqrt{a}} \iint_{-\infty}^{+\infty} S(t)\psi\left(\frac{t-b}{a}\right)dt, \tag{2.3}$$

де $W_{a,b}(t)$ — коефіцієнти вейвлет-перетворення; a — масштабний параметр (розтягує або стискає хвилю); b — зсув (зміщення по часу); $\psi(t)$ — базова вейвлет-функція; $S(t)$ — сигнал після зчитування.

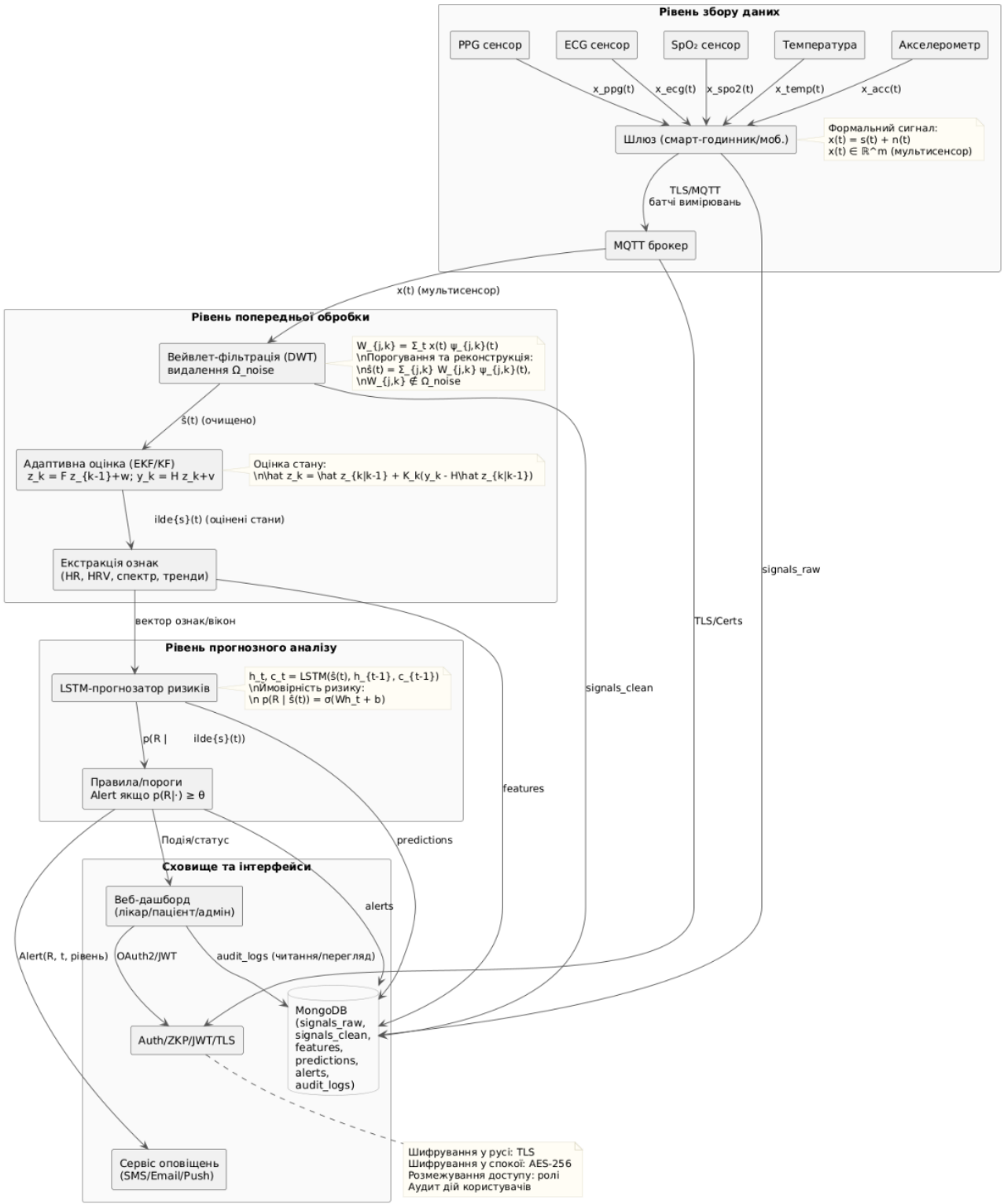


Рис. 2.1 Формалізована модель IoT-системи автоматизованого моніторингу медичних показників

Адаптивна фільтрація сигналу (розширений фільтр Калмана):

$$x_k = x_{k-1} + K_k(z_k - H_k x_{k-1}), \quad (2.4)$$

де x_k — оцінка стану системи на момент часу; x_{k-1} — попередня оцінка стану; K_k— коефіцієнт (матриця) Калмана; z_k — виміряний вхідний сигнал; H_k— матриця спостереження, що пов'язує реальний стан із вимірюванням.

2.2 Метод автоматизованого моніторингу з вхідними даними

Запропонована нами методика покращує існуючу модель Mirza et al. (2022) [5] шляхом додавання додаткових етапів обробки сигналу та прогнозного аналізу. Якщо оригінальна методика Mirza et al. сфокусована на оптимізації роботи IoT-пристроїв, то наше вдосконалення додатково забезпечує якісну обробку та аналіз медичних сигналів [12].

Основними етапами запропонованої методики є:

1. Безперервний збір сигналів з IoT-пристроїв (смарт-годинники, медичні сенсори).
2. Попередня обробка сигналів за допомогою вейвлет-перетворення для усунення шумів, що виникають через рухи пацієнта, потовиділення, слабкий контакт пристрою зі шкірою [14].
3. Вторинна адаптивна фільтрація сигналів методом розширеного фільтра Калмана для динамічного уточнення реальних значень медичних показників у складних умовах.
4. Передача очищених сигналів до нейромережевої моделі LSTM, яка прогнозує виникнення критичних змін стану пацієнта на основі аналізу послідовностей сигналів у реальному часі [15].
5. Передача прогнозних результатів і очищених даних до хмарної платформи, де дані аналізуються для автоматичного визначення критичних станів.
6. Оперативне оповіщення медичного персоналу у разі виявлення ризиків.

Таким чином, метод дозволяє перейти від сирих шумних сигналів до інтерпретованих прогнозів про стан здоров’я. Для наочного представлення основних етапів запропонованої методики розроблено схему, що демонструє повний цикл її реалізації: від зчитування фізіологічних сигналів за допомогою IoT-пристроїв до очищення даних, прогнозного аналізу та передачі результатів у хмарну інфраструктуру з подальшим інформуванням медичного персоналу. Основні етапи методики подано на рисунку. 2.2

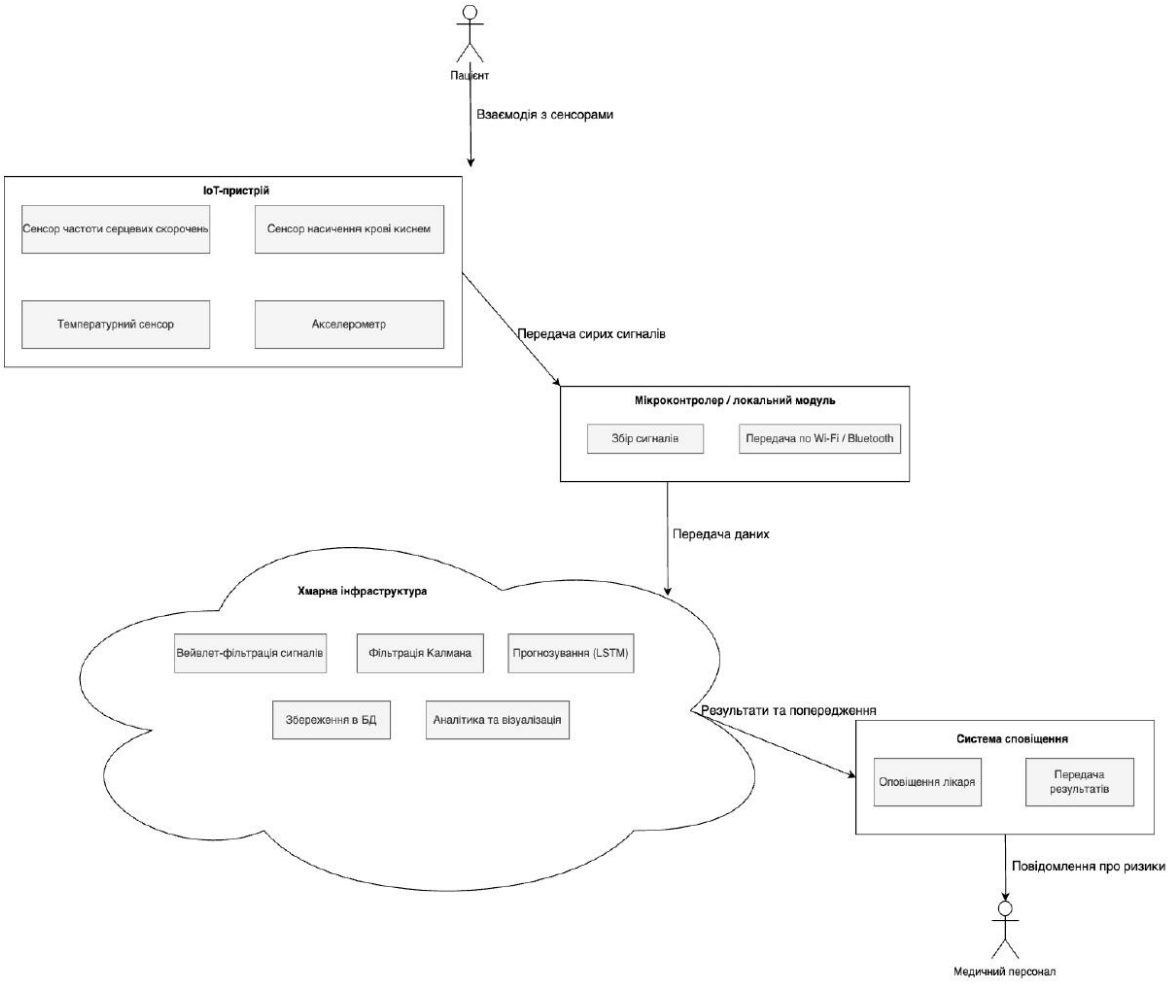


Рис. 2.2 Основні етапи запропонованої методики

Проведений порівняльний аналіз показує, що запропонований підхід забезпечує значне зниження похибки вимірювань, підвищує оперативність та точність виявлення критичних змін, а також забезпечує можливість їх прогнозування, чого не забезпечують традиційні методи

2.3 Алгоритми двоетапної обробки медичних сигналів та інтеграція нейронної мережі для прогнозування медичних ризиків

Етап 1. Вейвлет-фільтрація.

Вейвлет-перетворення ефективне для видалення шумів, викликаних рухами пацієнта, слабким контактом датчика або артефактами середовища. Використання дискретного вейвлет-перетворення (DWT) дає змогу розділити сигнал на апроксимаційні та деталізуючі коефіцієнти, після чого відсікаються шумові складові.

На першому етапі виконується декомпозиція сигналу за допомогою дискретного вейвлет-перетворення (DWT):

$$W_{j,k} = \sum_t \quad x(t)\psi_{j,k}(t), \tag{2.5}$$

де $\psi_{j,k}(t)$ – вейвлет-базис на масштабі j і зсуві k. Очищений сигнал відновлюється як:

$$s(t) = \sum_{j,k} \quad W_{j,k}\psi_{j,k}(t), \qquad W_{j,k} \notin \Omega_{noise}\ , \tag{2.6}$$

де Ω_{noice} – множина коефіцієнтів, що відповідають шумам та відсікаються пороговою функцією.

Очищені вейвлетом дані надходять у модель стану:

$$z_k = Fz_{k-1} + w_{k-1},\ Fz_{k-1} \sim N(0, Q), \tag{2.7}$$

$$y_k = Hz_k + u_k,\ u_k \sim N(0, Q), \tag{2.8}$$

де z_k – вектор прихованих фізіологічних станів, y_k – результати після вейвлет-фільтрації, F – матриця переходу, H – матриця вимірювань, Q, R – коваріаційні матриці шумів процесу та вимірювання.

Етап 2. Розширений фільтр Калмана.

Цей метод дозволяє враховувати динамічні зміни сигналу та уточнювати реальні значення параметрів у режимі реального часу. Розширений фільтр Калмана

будується на нелінійних рівняннях стану та вимірювань, що робить його придатним для фізіологічних даних.

Алгоритм Калмана виконує оцінку стану:

$$Z_k = z_{k|k-1} + K_k(y_k - Hz_{k|k-1}), \quad (2.8)$$

де K_1 – матриця посилення Калмана. В результаті формується вектор оцінених показників $s(t)$

Етап 3. Інтеграція нейромережі (LSTM).

Очищені сигнали подаються у рекурентну нейронну мережу довгої короткочасної пам'яті (LSTM), яка здатна враховувати часову залежність даних. Це дозволяє прогнозувати розвиток критичних станів пацієнта (наприклад, падіння SpO_2 чи аномалії ритму серця) ще до їх фактичного прояву.

Очищені часові ряди $s(t)$ подаються у рекурентну нейронну мережу типу LSTM:

$$h_{t,}c_{t,} = LSTM\ (s(t), h_{t-1}, c_{t-1}), \quad (2.9)$$

де h_i – прихований стан, c_i – стан пам'яті.

Вихід мережі формує оцінку ризику:

$$P\ (R|s(t)) = \sigma(Wh_{t,} + b), \quad (2.10)$$

де R – ризик (наприклад, гіпоксія, аритмія, підвищення температури), σ (·) – сигмоїдна функція.

Система ініціює тривогу, якщо ймовірність ризику перевищує поріг θ:

$$\text{Alert} = \{1, \quad p(s(t)) \geq \theta\ 0, \quad p(R \mid s(t) < 0),$$

Таким чином, формалізована модель IoT-системи моніторингу може бути подана у вигляді відображення:

$$x(t) \xrightarrow[\text{вейвлет}]{\text{очистка}} s(t) \xrightarrow[\text{Калман}]{\text{очистка}} s(t) \rightarrow \xrightarrow[LSTM]{\text{очистка}} s(t) \rightarrow p\ (R|s(t)), \quad (2.11)$$

де $x(t)$ – багатоканальні сирі сигнали, (t) – оцінені показники після двоетапної фільтрації, $p(R|\dot{s}(t))$ – оцінка ризику, яка визначає подальші дії системи (архівування, візуалізація, оповіщення), а кожен етап послідовно підвищує якість інформації: від сирих сигналів до математично обґрунтованих прогнозів ризиків.

Запропонована методика забезпечує кардинальне покращення системи медичного моніторингу, дозволяючи здійснювати автоматизований аналіз та прогнозування стану пацієнтів у реальному часі, що суттєво підвищує точність і оперативність реагування на критичні медичні стани, знижує ризики для пацієнтів та навантаження на медичний персонал.

2.4 Принципи забезпечення безпеки та конфіденційності персональних медичних даних

Оскільки система автоматизованого моніторингу працює з надзвичайно чутливою інформацією про стан здоров’я пацієнтів, питання захисту персональних даних є ключовим. Усі етапи життєвого циклу даних — від моменту зчитування сигналу сенсором до збереження результатів у базі даних чи їхнього відображення у вебінтерфейсі — повинні супроводжуватися суворим дотриманням принципів безпеки та конфіденційності.

Першим і базовим механізмом виступає шифрування. Під час передавання даних від IoT-пристрою до шлюзу або хмарної платформи використовується протокол TLS (Transport Layer Security), що унеможливлює перехоплення інформації третіми сторонами [16]. При зберіганні дані мають бути захищені алгоритмами симетричного шифрування високого рівня, наприклад AES-256, який є міжнародним стандартом для медичних систем та фінансових сервісів. Це гарантує, що навіть у разі компрометації бази даних зловмисник не зможе отримати доступ до змісту без відповідного ключа.

Другим важливим аспектом є аутентифікація користувачів. Система повинна підтримувати багатофакторні механізми ідентифікації, які поєднують пароль,

одноразовий код (OTP), біометричний фактор (наприклад, відбиток пальця чи Face ID) або апаратний токен. Це мінімізує ризик несанкціонованого доступу до облікових записів навіть у випадку викрадення пароля.

Третій принцип стосується розмежування доступу відповідно до ролей користувачів. Для пацієнта доступ має бути обмежений лише до його власних даних і базових функцій перегляду. Лікар отримує можливість моніторингу групи пацієнтів, призначених до його профілю, а адміністратор — доступ до технічних налаштувань системи, але без права перегляду клінічних даних. Така модель доступу реалізується за допомогою політики RBAC (Role-Based Access Control), що забезпечує принцип «найменших привілеїв». Важливою умовою прозорості є журналювання всіх дій у системі. Лог-файли повинні зберігати інформацію про вхід у систему, спроби доступу, зміну даних та формування звітів. Це дозволяє проводити аудит безпеки, виявляти аномальну активність та вчасно реагувати на інциденти.

Ще одним фундаментальним принципом є відповідність чинним нормативно-правовим актам. Європейське законодавство регламентує захист персональних даних через GDPR, у США у сфері охорони здоров’я діє HIPAA, а в Україні — Закон «Про захист персональних даних» та суміжні акти. Система повинна враховувати ці вимоги при розробці архітектури, виборі протоколів шифрування та організації доступу.

Окремо слід підкреслити важливість захисту конфіденційності. У випадках використання даних для наукових досліджень або навчання моделей машинного навчання застосовуються методи анонімізації та псевдонімізації. Це означає, що персональні ідентифікатори пацієнта (ім’я, дата народження, адреса) видаляються або замінюються унікальними кодами, що унеможливлює ідентифікацію конкретної особи.

Таблиця 2.1

Основні загрози для системи автоматизованого моніторингу та відповідні контрзаходи

Загроза	Можливі наслідки	Контрзахід (засіб захисту)
Перехоплення даних при передачі від сенсора до серверу	Розкриття чутливої медичної інформації третім особам	Використання TLS 1.3, VPN-тунелювання, шифрування пакетів
Несанкціонований доступ до бази даних	Викрадення або зміна медичних показників	Шифрування даних у спокої (AES-256), контроль доступу на рівні БД, сегментація мережі
Викрадення пароля користувача	Отримання доступу до чужих записів	Багатофакторна автентифікація (пароль + OTP/біометрія/токен)
Надмірні права користувача	Можливість перегляду чи редагування даних, що йому не належать	Рольова модель доступу RBAC (пацієнт/лікар/адмін), принцип «найменших привілеїв»
Зміна або видалення критичних даних	Викривлення медичної історії, втрата доказової бази	Журналювання всіх операцій, аудит, контроль версій даних
Витік даних під час досліджень або навчання моделей	Ідентифікація пацієнтів у відкритих наборах	Анонімізація/псевдонімізація даних, генерація унікальних кодів замість ПІБ
Атаки типу «відмова у доступі» (DoS/DDoS)	Порушення роботи платформи, недоступність моніторингу	Захист на рівні мережі (фаєрвол, IDS/IPS), балансування навантаження
Несанкціоноване втручання у роботу сенсорів	Передача викривлених сигналів	Використання цифрових сертифікатів пристроїв, перевірка автентичності IoT-девайсів
Несанкціонований доступ адміністратора	Несанкціонований доступ адміністратора	Несанкціонований доступ адміністратора

Таким чином, методика двоетапної обробки сигналів із застосуванням вейвлет-перетворення, фільтра Калмана та нейромережевих моделей не лише підвищує точність і надійність IoT-систем медичного моніторингу, але й, у поєднанні з описаними заходами, гарантує високий рівень захисту персональних даних, дотримання міжнародних стандартів безпеки та збереження довіри з боку пацієнтів і медичних установ.

2.5 Переваги та недоліки існуючих методів двоетапної обробки сигналів

Існуючі методи двоетапної обробки сигналів у системах медичного моніторингу, як правило, ґрунтуються на поєднанні вейвлет-перетворення для усунення шумових складових та розширеного фільтра Калмана для уточнення параметрів у режимі реального часу. Такий підхід є класичним у біомедичних дослідженнях, що довів свою ефективність у численних практичних застосуваннях. Водночас, попри наявні переваги, ці методи мають низку обмежень, що знижують їх ефективність у реальних клінічних умовах, це продемонстровано у таблиці 2.2.

Переваги методів:

– усунення шумових складових (рухові артефакти, електричні перешкоди, похибки контакту датчика зі шкірою);

– адаптивність до змін фізіологічного сигналу у реальному часі;

– підвищення достовірності даних для подальшого аналізу та прогнозування;

– універсальність та можливість використання для багатоканальних сигналів (ЕКГ, PPG, акселерометр, температура);

– сумісність із нейромережевими моделями (зокрема LSTM) для прогнозного аналізу [17];

– математична обґрунтованість методів, що робить їх стандартизованими і відтворюваними;

– підтверджена ефективність у численних дослідженнях і практичних системах медичного моніторингу;

– можливість гнучкої інтеграції з хмарними платформами для довготривалого зберігання та аналізу сигналів.

Недоліки методів:

– висока обчислювальна складність, що ускладнює реалізацію на пристроях із обмеженими ресурсами (наприклад, смарт-годинниках);

– залежність від точності налаштування параметрів (вибір вейвлет-базису, коваріаційних матриць шумів тощо);

– недостатня стійкість у випадках значних рухових артефактів чи втрати частини даних;

– можливі затримки обробки, що знижує оперативність реагування системи у критичних випадках;

– обмежена ефективність у складних клінічних умовах, коли якість сигналу критично знижується (наприклад, у пацієнтів з постійними рухами чи тремором);

– складність масштабування для великої кількості одночасно підключених сенсорів;

– необхідність попереднього калібрування системи під конкретного пацієнта чи сенсор;

– обмежена інтерпретованість результатів для лікаря без додаткової візуалізації.

– високе енергоспоживання, що зменшує автономність носимих пристроїв (швидке розряджання батареї).

– низька стійкість до втрати даних через проблеми зв’язку з хмарою або Bluetooth-переривання.

– неврахування індивідуальних особливостей пацієнта (наприклад, у дітей та літніх людей характеристики сигналу значно відрізняються).

– обмежена гнучкість алгоритмів — при появі нових типів сенсорів методи потребують суттєвої адаптації.

– складність інтеграції з реальними медичними системами (HL7, FHIR), бо вихідні дані не завжди відповідають медичним стандартам.

– ризик хибнопозитивних сигналів тривоги, що перевантажує лікаря і знижує довіру до системи.

Таблиця 2.2

Переваги та недоліки існуючих методів двоетапної обробки сигналів

Критерій	Переваги	Недоліки
Усунення шумів	Вейвлет-перетворення прибирає рухові артефакти у ЕКГ під час ходьби пацієнта.	При бігу або різких рухах з'являються хибні піки QRS, які не фільтруються.
Адаптивність	Фільтр Калмана коригує вимір SpO_2 при зміні освітлення (наприклад, коли пацієнт переходить з темної кімнати у світлу).	Якщо датчик погано прилягає, фільтр дає «стрибки» насичення киснем до 10–15%.
Точність діагностики	Поєднання методів зменшує кількість хибно визначених аритмій у 2–3 рази.	При втраті сигналу на кілька секунд система може видати «аритмію», якої немає.
Багатоканальність	Одночасна обробка ЕКГ, PPG і температури дозволяє комплексний аналіз (наприклад, при лихоманці з тахікардією).	При підключенні >3 сенсорів (ЕКГ+PPG+акселерометр) виникає затримка 2–3 с у відображенні результату.
Прогнозування	Очищені дані можна подавати у LSTM, яка передбачає падіння SpO_2 за 20–30 с до критичного рівня.	Висока обчислювальна складність — на смарт-годиннику прогноз може з'явитися із запізненням до 1 хв.

Таким чином, хоча класичні методи двоетапної обробки сигналів дають суттєві переваги, їхні недоліки обмежують практичне застосування у клінічних умовах. У наступному розділі буде представлено вдосконалену методику, що усуває більшість зазначених проблем: знижує обчислювальні витрати, підвищує стійкість до втрати даних та артефактів, а також забезпечує адаптивну інтеграцію з нейромережевими моделями для прогнозування ризиків.

3 ПРОГНОЗУВАННЯ ВІДХИЛЕНЬ ФІЗІОЛОГІЧНИХ ПОКАЗНИКІВ ПАЦІЄНТІВ

3.1 Модифікація алгоритмів попередньої обробки та фільтрації шумів

Адаптивно-вейвлетна фільтрація з маскою артефактів руху (AWeMAF). Для підвищення стійкості до рухових артефактів вводимо бінарно-неперервну маску артефактів M(t), що оцінюється як індекс $J(t)$ та нормується сигмоїдою:

$$J(t) = \left|\left|a(t) - a(t - \Delta t)\right|\right|_2 , \quad (3.1) \qquad M(t) = \sigma\left(\alpha\left(J(t) - \theta_J\right)\right) \in [0,1], \quad (3.2)$$

де $a(t)$ – акселерометричний сигнал у момент часу t; $J(t)$ – індекс рухових артефактів, що оцінює рівень активності; σ(·) – сигмоїдна функція нормалізації; a, θ_J – параметри масштабу та порога; $M(t) \in [0,1]$ – маска артефактів руху.

У дискретному вейвлет-перетворенні (DWT/WPT) рівнево-залежний поріг робимо із залежністю від часу:

$$\tau_{\square}(t) = \tau_0 2^{-\frac{l}{2}} \left(1 + \beta \cdot M(t)\right), \quad (3.3)$$

де $\tau_{\square}(t)$ – рівнево-залежний пороговий параметр для коефіцієнтів вейвлет-перетворення; l – рівень декомпозиції; τ_0 – базове значення порогу; β – коефіцієнт посилення при наявності артефактів; $M(t)$ – маска рухових артефактів.

Потім застосовуємо маскований soft-thresholding для деталізацій $D_{l,k}$

$$D_{l,k}(t) = sign(D_{l,k}(t)) \cdot max(\left|D_{l,k}(t)\right| - \tau_l(t), 0) , \quad (3.4)$$

де $D_{l,k}(t)$ – коефіцієнт вейвлет-декомпозиції на рівні l, позиції k; $\tau_l(t)$ – порогове значення, адаптивне до маски руху; sign(·) – знак функції;

Це зменшує помилки у визначенні піків (QRS/PPG-onset) у моменти інтенсивних рухів без перефільтрації всього сигналу.

Спектрально-керований $notch$ (вузькосмуговий фільтр-придушувач) з автоматичним трекінгом перешкод. Для 50/60 Гц та гармонік вводимо вузькосмуговий фільтр із онлайн-трекінгом частоти $f_{int}(t)$ через максимум у локальному спектрі (вікно 1–2 с), щоб уникнути спотворення позиції при

нестабільному живленні. Коефіцієнти фільтра оновлюються лише коли $|f_{int}(t) - f_{int}(t - \Delta t)| > \delta_f$, що знижує енергозатрати.

Однією з проблем класичної обробки біомедичних сигналів є спотворення на краях часових вікон. При застосуванні вейвлет-перетворення, короткочасного Фур'є-аналізу чи віконної фільтрації значення на межах інтервалів зчитування часто викривляються через обтинання сигналу. Це проявляється у вигляді хибних зсувів амплітуди, фазових артефактів або появи штучних хвиль у кінцях відрізків. У медичному моніторингу такі викривлення можуть призвести до неправильної інтерпретації даних, наприклад, до появи хибного піку QRS на ЕКГ чи помилкового зниження SpO_2 у PPG.

Щоб уникнути цього ефекту, у нашій методиці застосовано краєзберігаючу реконструкцію на основі схеми $overlap - add$ з косинусним вікном. Принцип полягає в тому, що сигнал не обробляється у вигляді окремих жорстко відсічених блоків, а ділиться на перекривані сегменти, кожен з яких плавно зважується спеціальною віконною функцією. Для цього обрано косинусне вікно, яке забезпечує поступовий спад коефіцієнтів до нуля на краях і таким чином мінімізує амплітудні стрибки між сегментами. Після фільтрації всі сегменти знову об'єднуються у вихідний сигнал шляхом додавання з перекриттям ($overlap - add$), що гарантує безперервність та відсутність різких зламів.

Ключовим удосконаленням є застосування так званої $edge - safe\ resynthesis$. На відміну від звичайного $overlap - add$, у нашому підході коефіцієнти на межах сегментів додатково коригуються за допомогою вагових факторів, що нормалізують суму перекриваних вікон. Це дозволяє уникнути надмірного посилення або приглушення сигналу у зонах накладання. Алгоритм будується таким чином, що будь-яка точка сигналу у зоні перекриття отримує внесок від двох сусідніх сегментів у зваженому вигляді, причому сума ваг завжди дорівнює одиниці. Такий підхід гарантує сталість енергії сигналу та відсутність фазових зсувів.

Додатково передбачено корекцію межових коефіцієнтів у випадках, коли сегмент обробляється методами, що змінюють локальну амплітуду (наприклад, при пороговій вейвлет-фільтрації). У такій ситуації значення на краях можуть суттєво

відрізнятися від сусідніх сегментів. Для цього ми застосовуємо інтерполяційне згладжування крайових коефіцієнтів на основі їх порівняння з усередненим прогнозом, отриманим із сусідніх блоків. Таким чином забезпечується плавність переходу і уникнення штучних стрибків.

У результаті краєзберігаюча реконструкція дає низку переваг:

1. мінімізуються спотворення сигналу на межах обробки;
2. забезпечується коректна передача амплітудних і фазових характеристик;
3. зменшується ризик хибних медичних індикаторів;
4. досягається сумісність із режимами реального часу, оскільки обробка виконується у блоках, але з плавним з'єднанням.

Для одиничних викидів у PPG/SpO_2 можна застосувати локальну Huber-проекцію до ковзного медіани m_v і MAD:

$$x(t) = agrsmin \sum_{u \in N_w(t)}^{n} \quad p_\delta(x(u) - z), \tag{3.5}$$

де $N_w(t)$ – локальне вікно навколо моменту часу t; p_δ $(\cdot)$ – функція Хубера для придушення одиничних викидів; z – шукане значення, що мінімізує відстань до сусідів за робастним критерієм: x(t) – відновлене значення сигналу у момент часу t, що стабілізує короткі перешкоди без втрати фазової інформації. Ефект підходу полягає в тому, що у складних рухових режимах зменшується похибка пікового детектування та кількість хибних відліків у каналах PPG/ЕКГ, тоді як енергоспоживання скорочується завдяки подієвому (масковому) підсиленню порогів

Таким чином, застосування це дозволяє поєднати точність спектрально-частотних методів із коректністю відновлення повного сигналу, що робить цей підхід особливо цінним для IoT-систем медичного моніторингу, де критично важливо зберігати достовірність навіть дрібних відхилень у фізіологічних показниках.

3.2 Вдосконалення методів виділення ключових ознак медичних сигналів

Запропоновано компактний FeatureBank-16 (FB-16) — набір з 16 ознак для реального часу, сумісний із нашою двоетапною обробкою: ЕКГ/PPG таймінги (6 ознак): RR (інтервал між R-зубцями), HR (частота серцевих скорочень), RMSSD (варіабельність серцевого ритму), SDNN (варіабельність серцевого ритму), pNN50

(варіабельність серцевого ритму), TINN (варіабельність серцевого ритму); PPG морфологія (10 ознак): Амплітуда AC/DC, Індекс крутозни (skew), Співвідношення площ (під висхідною/низхідною частинами), SDPPG-показник (a), SDPPG-показник (b), SDPPG-показник (c), SDPPG-показник (d), SDPPG-показник (e), SDPPG-відношення (b/a), SDPPG-відношення (c/a) , це продемонстровано на рисунку 3.1 та рисунку 3.2.

Для зручності аналізу ці 16 ознак згруповані у чотири основні категорії:

1. ЕКГ/PPG таймінги: RR, HR, варіабельність HRV (RMSSD, SDNN, pNN50, TINN).

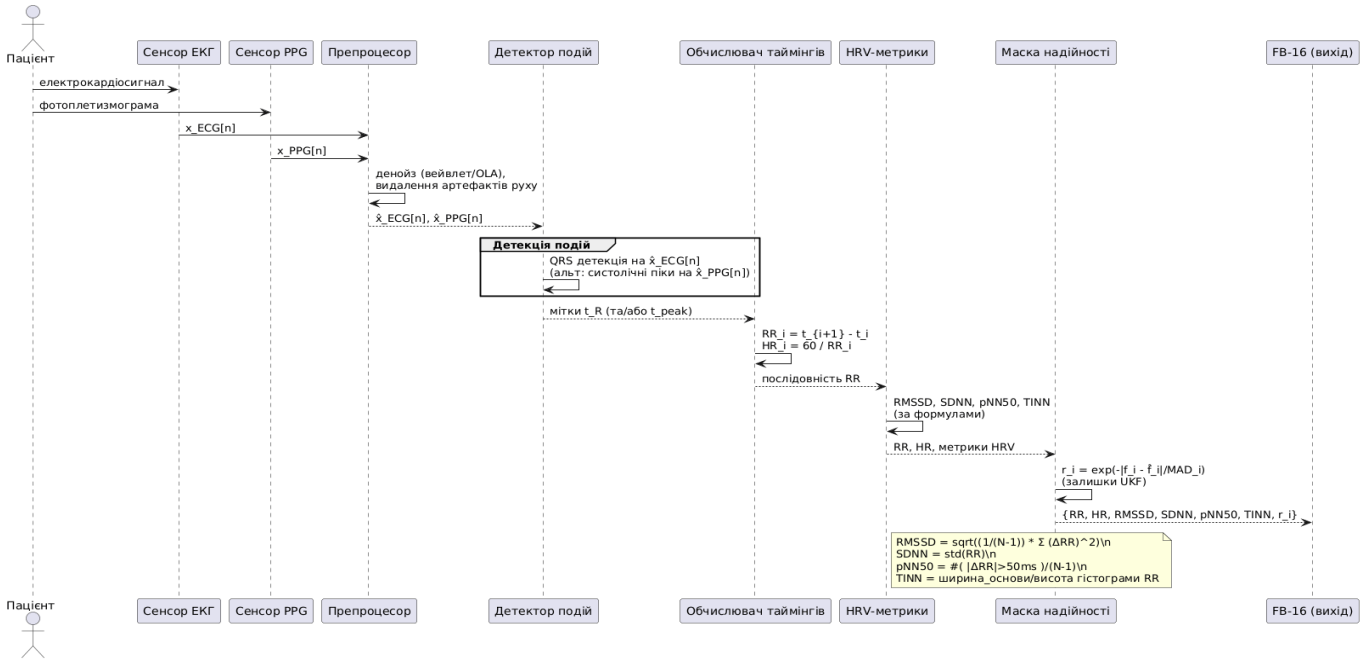


Рис. 3.1 Послідовність екстракції таймінгових ознак

2. PPG морфологія: амплітуда AC/DC, індекс крутозни (skew), співвідношення площ під висхідною/низхідною частинами, SDPPG-показники (a, b, c, d, e) та їх відносини $b/a, c/a$.

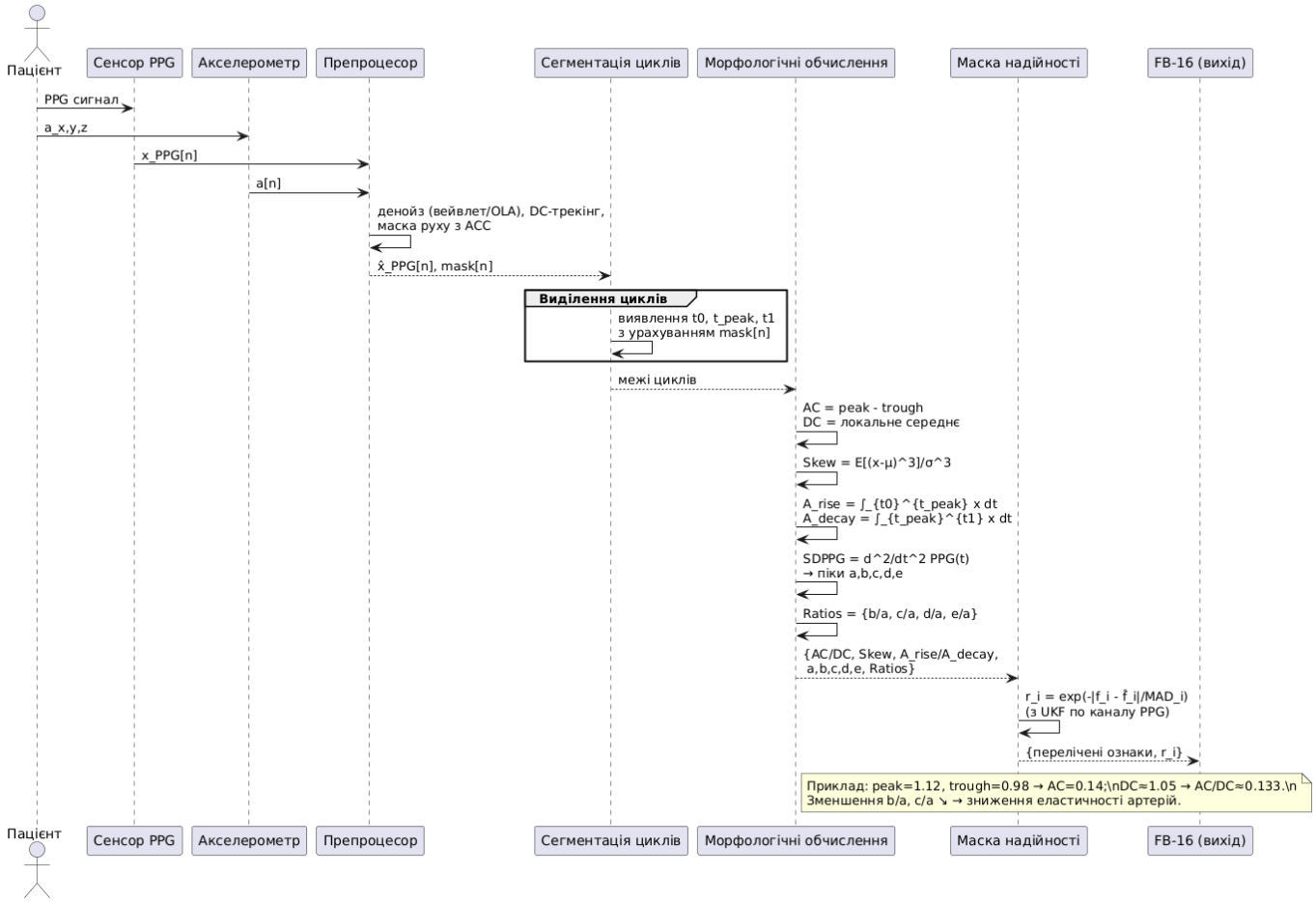


Рис. 3.2 Послідовність екстракції морфологічних ознак PPG

3. Тренди та похідні: похідні $\frac{d}{dt}HR, SpO_2$ Temp, а також кут нахилу лінійної регресії на інтервалі 10–30 с, це продемонстровано в таблиці 3.1 та таблиці 3.2.

Таблиця 3.1

Похідні за інтервалами

Інтервал (с)	ΔHR (уд/хв)	dHR/dt (уд/хв·с⁻¹)	ΔSpO₂ (%)	dSpO₂/dt (%/с)	ΔTemp (°C)	dTemp/dt (°C/с)
0–5	2.0	0.40	−0.5	−0.10	0.0	0.00
5–10	1.0	0.20	−0.5	−0.10	0.1	0.02
10–15	2.0	0.40	−1.0	−0.20	0.1	0.02
15–20	3.0	0.60	−1.0	−0.20	0.1	0.02
20–25	2.0	0.40	−0.5	−0.10	0.1	0.02
25–30	25–30	25–30	25–30	25–30	25–30	25–30

В таблиці 3.1 подано результати розрахунку локальних похідних за короткими інтервалами (5 секунд), що дозволяє простежити динаміку кожного параметра у мікрошкалі.

Таблиця 3.2

Кут нахилу лінійної регресії на вікні 10–30 с

Параметр	Вікно часу	Кут нахилу (од/с)
HR (уд/хв)	10–30 с	+0.46
SpO_2 (%)	10–30 с	−0.15
Temp (°C)	10–30 с	+0.02

В таблиці 3.2 подано наведено кути нахилу лінійної регресії для HR, SpO_2 та Temp у вікні 10–30 секунд, які узагальнюють довготривалі тенденції й забезпечують більш стійку інтерпретацію.

У вищевказаних прикладах для кардіореспіраторного моніторингу спостерігається помірне зростання частоти серцевих скорочень зі середнім темпом приблизно 0,46 уд/хв за секунду на інтервалі 10–30 с, що узгоджується з додатними локальними похідними dHR/dt у більшості п’ятисекундних вікон; паралельно насичення киснем у крові зменшується з середнім спадом приблизно 0,15 % за секунду, що відображає негативні похідні $dSpO_2/dt$ у відповідних інтервалах, тоді як температура тіла демонструє повільне, але стабільне зростання з кутом нахилу близько 0,02 °C/с, що підтверджується стало позитивними похідними dTemp/dt після 5-ї секунди; таким чином, регресійні кути забезпечують згладжену оцінку тенденцій на вибраному часовому відрізку, а таблиця інтервальних похідних деталізує миттєву динаміку між сусідніми вимірюваннями, дозволяючи відрізнити короткочасні коливання від систематичних трендів, що є ключовим для раннього виявлення ризиків у реальному часі.

4. Ф’южн з акселерометра: дисперсія $Var(a)$, індекс стаціонарності S (для IMM-моделі нижче).

Щоб робити екстракцію стабільною під час прогалин, ознаки супроводжуємо маскою надійності $r_i \in [0,1]$, яку обчислюємо з залишків після UKF та MAD-метрик; далі вона йде як додатковий канал у модель прогнозування, це продемонстровано на рисунках 3.3-3.5.

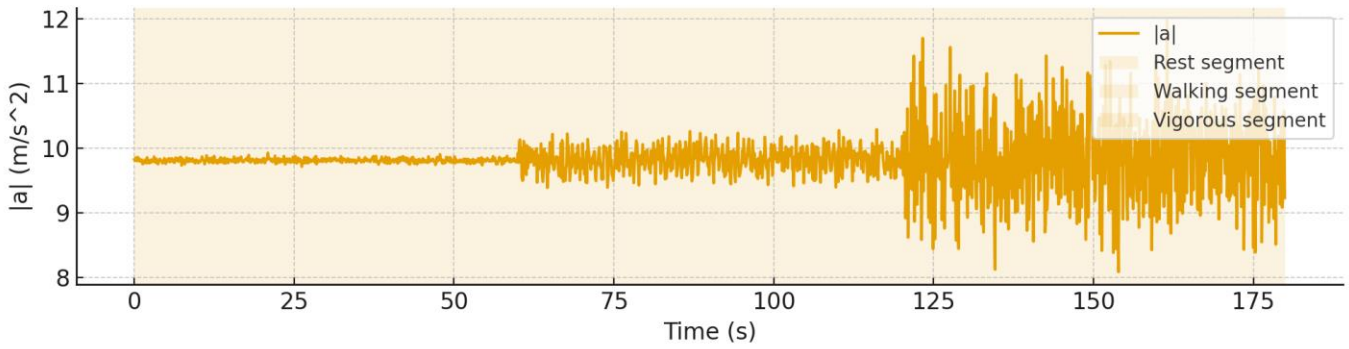


Рис. 3.3 Графік прискорення |a|

На рисунку 3.3 показано часовий ряд модуля прискорення $|a|$ (10 Гц, 180 с) із трьома чіткими режимами: у спокої (0–60 с) величина коливається навколо 9,8 $\frac{м}{с^2}$ з малою шумовою амплітудою ≈ ±0,02; під час ходьби (60–120 с) з'являється регулярна осциляція й розкид збільшується до ≈±0,3, що відповідає руховим ударам; у фазі інтенсивної активності (120–180 с) спостерігаються великі коливання, локальні піки сягають 11,5-12 $\frac{м}{с^2}$, а провали – до 8,2–8,5 м/с^2, що вказує на неоднорідні динамічні навантаження; такі зміни модуля $|a|$ надалі використовуються як контекст для посилення/послаблення фільтрації в каналах PPG/ЕКГ.

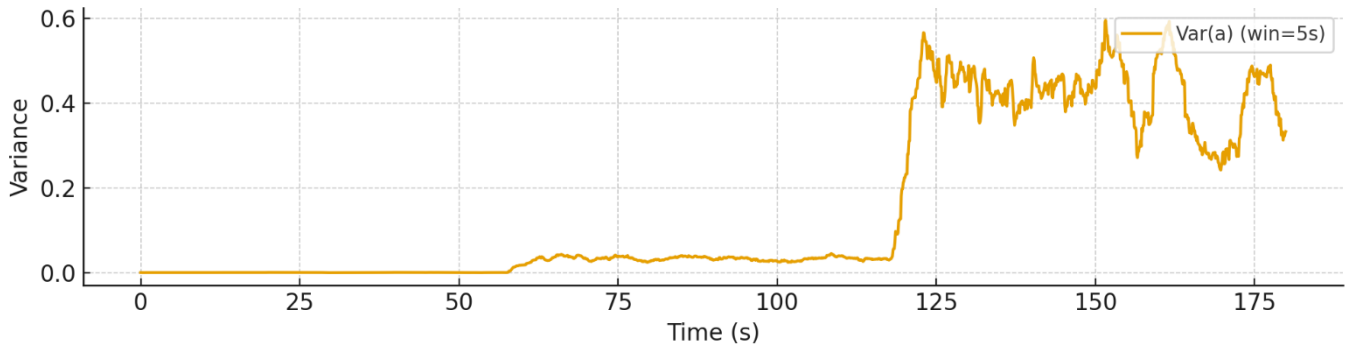


Рис. 3.4 Ролінг-варіація $Var(a)$

На рисунку 3.4 показано ролінг-варіацію $Var(a)$ у вікні 5 с як індикатор інтенсивності руху: у спокої вона практично нульова (<0,01), після 60-ї секунди зростає до 0,03–0,06 у режимі ходьби, а в інтенсивному сегменті (≈120–180 с) досягає характерних плато й сплесків 0,3–0,6, що корелює з різкими змінами |a|; саме цей показник ми включаємо до FeatureBank-16 і використовуємо як подієвий тригер: великі значення $Var(a)$ ⇒ підвищити пороги вейвлет-порогування та зменшити вагу морфологічних ознак PPG.

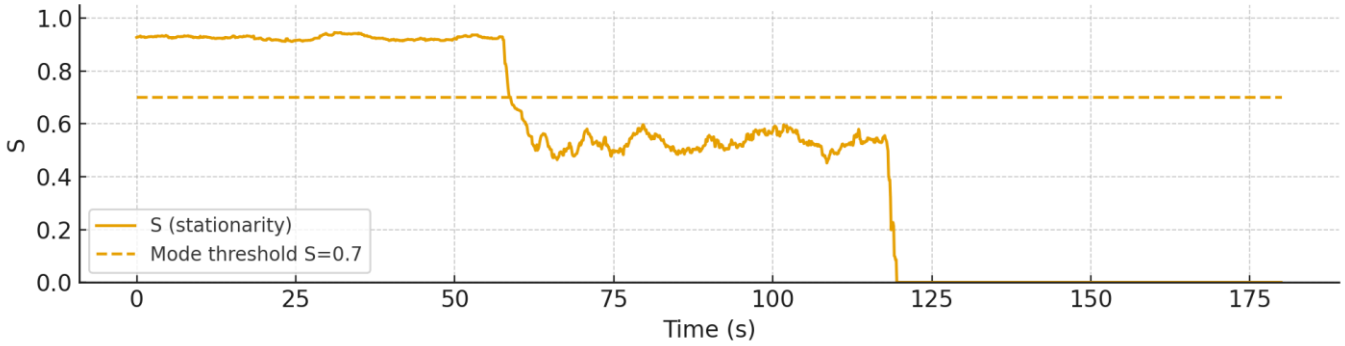


Рис. 3.5 Індекс стаціонарності S з порогом

На рисунку 3.5 показано індекс стаціонарності $S(t)=[1-q_{win}/q_{global}]_0^1$ із порогом S=0,7 для вибору режиму у IMM-логіці: у спокої S≈0,9–0,93 (вище порога) – система перебуває в режимі R із малими коваріаціями фільтра; у фазі ходьби індекс стаціонарності S знижується до значень близько 0,5–0,6, що є нижчим за встановлений поріг S=0,7. Це інтерпретується системою як перехід до режиму руху M. У даному режимі для адекватного відображення підвищеної варіабельності сигналу використовуються збільшені значення коваріаційних матриць процесу Q та вимірювань R, що відповідає концепції «розширеної невизначеності» у фільтрах Калмана. Такий підхід забезпечує збереження стійкості оцінки стану в умовах інтенсивних рухових артефактів та зменшує ризик формування некоректних прогнозів.

3.3 Інтеграція оптимізованої нейронної мережі для прогнозування ризиків

В рамках удосконаленої методики передбачено інтеграцію оптимізованої нейронної мережі, орієнтованої на прогнозування ризиків, що дозволяє значно зменшити обчислювальні витрати та підвищити стійкість системи до пропусків і шумів. Замість класичної рекурентної архітектури типу LSTM використовується легкий каскад «Conv-Lite → GRU-Attn», де перший блок виконує стискання часових рядів за рахунок одномірних згорток із розрідженими ядрами та скорочує довжину послідовності у 2–4 рази, а другий блок, побудований на основі простої рекурентної мережі GRU з увагою, концентрує вагу на ділянках із різкою динамікою та підвищеними показниками маски рухових артефактів $M(t)$. На вхід такої архітектури подаються вектори ознак $[FB-16 \parallel r]$, які доповнені агрегованими оцінками зі станового фільтра. Для компенсації класового дисбалансу під час навчання застосовується функція втрат типу focal loss із параметром $\gamma \in [1,2]$, що зменшує вплив чисельно домінуючих класів. Після завершення навчання проводиться пост-калібрування вихідних ймовірностей за допомогою температурної корекції T, що підвищує інтерпретованість прогнозів для лікаря. Для зниження кількості помилкових тривог уводяться гістерезисні пороги, де умова активації $\theta_{on} > \theta_{off}$ запобігає надмірному реагуванню на короткочасні відхилення.

Важливим елементом є мультизадачна постановка, коли до основної задачі класифікації ризикових станів додається допоміжна задача регресії до очищених показників z_t. У цьому випадку комбінована функція втрат має вигляд:

$$L = \lambda_{cls} L_{focal} + \lambda_{reg} ||z_{t_1} - z_{t_2}||_1 , \quad (3.6)$$

де z_{t_1}, z_{t_1} – вектори ознак (представлення сигналу об'єкта) у моменти час t_1 та t_2; L_{focal} – ункція втрат фокальної класифікації, яка зменшує вагу прикладів та акцентує на складних; $||z_{t_1} - z_{t_2}||$ – відстань між двома векторами ознак; λ_{cls} – коефіцієнт ваги частини функції втрат для класифікації; λ_{reg} – коефіцієнт ваги частини функції втрат для регресії (регуляризації).

Комбінована функція втрат змушує модель узгоджувати ризиковий бал із фізіологічно правдоподібними траєкторіями. Для підвищення стійкості обчислень другий ступінь класичної схеми замінено на двомодову IMM-UKF, яка поєднує режим спокою R з низькими коваріаційними матрицями Q_R, R_R та режим руху M з підвищеними значеннями Q_M, R_M. Перехід між режимами визначається перехідною матрицею Π та байєсівськими ймовірностями $\mu R, M(t)$ що залежать від індексу стаціонарності S та маски артефактів M(t). Підсумкова оцінка формується як:

$$z_t = \sum_{i \in \{R,M\}} \mu_i(t) z_t^{(i)}, \tag{3.7}$$

де $z_t^{(i)}$ – оцінка стану у режимі i (режим спокою R або режим руху M); $\mu_i\ (t)$ – баєсівська ймовірність перебування у режимі i в момент часу t; z_i – підсумкова (змішана) оцінка стану, отримана як зважене середнє між оцінками двох режимів.

У випадках пропусків даних застосовується модифікована коваріаційна матриця спостереження:

$$R_t' = R_t + \Lambda \cdot diag(1 - m_t), \tag{3.8}$$

де R_t – стандартна коваріаційна матриця похибки спостережень; R_t' – модифікована коваріаційна матриця, яка враховує пропуски у даних, m_t – вектор наявності каналів (1 – вимір є, 0 – пропуск), а параметр Λ забезпечує «вимкнення» відсутніх вимірювань без втрати стійкості оцінювача.

Для підвищення індивідуальної точності передбачено персоналізацію на основі шару FiLM (Feature-wise Linear Modulation), у якому вектор метаданих пацієнта (вік, стать, базальна частота серцевих скорочень) модулює приховані представлення за формулою:

$$h_t = \gamma(p) \odot h + \beta(p), \tag{3.9}$$

де h – приховане представлення (вектор ознак моделі); p – вектор метаданих пацієнта (наприклад, вік, стать, базальна частота серцевих скорочень тощо);
$\gamma(p)$, $\beta(p)$ – параметричні функції, що генерують коефіцієнти масштабування та зсуву; $\odot$ – поелементне множення; h_t – адаптоване приховане представлення,

модульоване індивідуальними характеристиками пацієнта, що забезпечує кращу адаптацію для крайніх груп без необхідності повного перенавчання всієї моделі.

Додатково, з метою використання системи у вбудованому режимі, модель готується за процедурою $quantization - aware\ training$ із приведенням параметрів до формату int8, що суттєво знижує вимоги до пам'яті та швидкодії. Вводиться також механізм раннього виходу (early-exit): якщо оцінка ризику у двох послідовних вікнах менша за порогове значення θ_{low}, інференс призупиняється до появи подієвого тригера. Це рішення дозволяє економити енергоресурси без зниження безпеки пацієнта.

Таким чином, запропоновані авторські рішення щодо інтеграції оптимізованої нейронної мережі поєднують компактність, енергоефективність і адаптивність до реальних умов використання. Архітектура Conv-Lite → GRU-Attn забезпечує високу пропускну здатність і можливість виділення критичних фрагментів, мультизадачність дозволяє поєднувати прогнозування ризику з корекцією фізіологічних параметрів, а поєднання з двомодовою IMM-UKF гарантує робастність навіть при неповних вимірюваннях. Персоналізація й режим роботи «на пристрої» роблять методику практично застосовною у сучасних IoT-рішеннях для моніторингу здоров'я.

3.4 Математична модель удосконаленої методики

Удосконалена методика двоетапної обробки сигналів може бути описана у вигляді компактної математичної моделі.

Вхідні дані. На вхід системи надходить багатоканальний сигнал:

$$x(t) = \{ x^{ECG}(t), x^{PPG}(t), a(t)\}, \quad (3.10)$$

де $\{ x^{ECG}(t)$ – електрокардіограма, $x^{PPG}(t)$ – фотоплетизмограма, a(t) – акселерометричний сигнал. Для врахування пропусків використовується вектор маски m_t, що позначає наявність або відсутність вимірювань.

Обробка сигналів. Попередня обробка P включає адаптивно-вейвлетну фільтрацію з маскою руху, notch-фільтрацію для придушення мережевих перешкод та edge-safe resynthesis для усунення спотворень на межах вікон. Результатом є

очищений сигнал $x_t^* = P(x(t))$. Далі формується вектор ознак $\phi_{t=}\Phi(x_t^*)$, що містить таймінгові, морфологічні, трендові та акселерометричні показники (FeatureBank-16).

Оцінювання стану. Для відстеження прихованих фізіологічних параметрів використовується двомодова IMM-UKF модель:

$$x_{k+1} = F_{z_k} + w_k, y_k = Hz_k + v_k, \quad (3.11)$$

де F – матриця переходу, H – матриця вимірювань, w_k ~N (0, Q), v_k ~N (0, R). Перемикання між режимами «спокій» і «рух» відбувається за індексом стаціонарності та маскою артефактів. На виході формується узгоджена оцінка стану $\dot{z}_k$.

Прогнозування. Отриманий вектор ознак та оцінка стану подаються на вхід нейронної мережі f_θ типу Conv-Lite → GRU-Attn:

$$y_k = f_\theta(z_k + \dot{z}_k), \quad (3.12)$$

де y_k – вихідне значення моделі на кроці k, яке представляє собою оцінку ризику (наприклад, ймовірність критичного стану) та допоміжні показники; f_θ – функція нейронної мережі, θ – навчені параметри (ваги) цієї мережі; z_k – вектор ознак (вхідні дані або характеристики сигналу), отриманий на попередніх етапах обробки; $\dot{z}_k$– узгоджена оцінка стану системи (фізіологічних параметрів), отримана від фільтра IMM-UKF.

Виходом є оцінка ризику та допоміжні показники. Для зниження хибних спрацювань використовується калібрування та гістерезисна логіка тривог.

Результат. Таким чином, модель відображає повний цикл: на вході — багатоканальні біосигнали; далі виконується обробка, що включає відновлення пропусків, фільтрацію та формування ознак; у блоці розрахунків здійснюється оцінка прихованих станів (IMM-UKF) та обчислення вектора ознак; на виході отримуємо прогноз ризику та службові індикатори надійності. Така формалізація поєднує фізіологічні сигнали, математичний апарат станового оцінювання та методи машинного навчання в єдину, стійку до пропусків систему прогнозування.

3.5 Забезпечення стійкості до втрати даних та неповних вимірювань

Для кожного каналу ведемо час від останнього валідного зразка Δ_t і виконуємо експоненційне згасання до останньої оцінки:

$$x_t^f = m_t x_t + (1 - m_t)(e^{-\Delta_r^t} \cdot x_{t-1}^f + (1 - e^{-\Delta_r^t})x), \quad (3.13)$$

де x_t – вхідне значення сигналу у момента часу t; $m_t \in \{0,1\}$ – маска наявності вимірювання (1 – значення доступне, 0 – пропуск); $x_t^f -$ відновлене (заповнене) значення сигналу; Δ_t $-$ відновлене значення сигналу на попередньому кроці; x – індивідуальна базова норма. Вектор m_t передаємо і в IMM-UKF, і в мережу

Крос-канальна деградація з довірою. Будуємо оцінку довіри $c_k(t) \in [0,1]$ для кожного каналу (residual-based + MAD). При зникненні PPG збільшуємо вагу ЕКГ-ознаків і навпаки:

$$x_t = \sum_k \frac{c_k(t)}{\sum_i c_j(t)} x_t^{(k)}, \quad (3.14)$$

де $x_t^{(k)}$ – ознаковий вектор від k-го джерела/каналу; $c_k(t)$ – ваговий коефіцієнт для $k-$го джерела у момент часу t; $\sum_i c_j(t)$ – нормалізаційний дільник, що забезпечує суму ваг рівною 1; x_t – агреговане (усереднене з вагами) представлення ознак. Це дозволяє мережі і фільтру стабільно працювати навіть при часткових відмовах.

Стійка логіка тривог. Замість одиночного порогу застосовуємо подвійну умову: (i) $p_r > \theta_{on}$ та (ii) персистентність $\geq T_{min}$ (наприклад, 6–10 с). Скидання тривоги лише при $p_r <\theta_{on}$ протягом T_{off}. Це різко зменшує хибнопозитивні події під час коротких пропусків. Кожне повідомлення супроводжується прапорцями якості (щільність пропусків, середня довіра каналів, активність фільтра), щоб лікар бачив не лише „ризик=0.82”, а й контекст надійності.

Розроблений алгоритм забезпечує адаптивну вейвлет-обробку з урахуванням рухових перешкод та подієвим посиленням порогів, що дозволяє ефективно пригнічувати шум без втрати інформативних компонентів сигналу. У сукупності це підвищує надійність роботи системи, зменшує енергоспоживання та робить її

придатною для автономного використання на IoT-пристроях у реальному часі, це продемонстровано (див. рис 3.6). Показано узагальнену блок-схему алгоритму двоетапної обробки сигналів та прогнозування, яка відображає послідовність виконання основних етапів системи. На вході здійснюється зчитування даних з каналів ЕКГ, PPG та акселерометра, після чого відбувається компенсація пропусків і відновлення сигналів для забезпечення стійкості. Далі реалізується попередня обробка, що включає адаптивно-вейвлетне фільтрування (AWeMAF), пригнічення мережевих перешкод за допомогою Notch-фільтра та краєзберігаючу реконструкцію (Edge-safe resynthesis).

Після цього алгоритм розгалужується: з одного боку формується набір ознак FeatureBank-16 (таймінгові, морфологічні, трендові), з іншого — здійснюється оцінювання стану за допомогою IMM-UKF з урахуванням режимів «спокій/рух» і адаптивних коваріацій. Обидві гілки зводяться у блок, де ознаки об’єднуються з урахуванням довіри та маски надійності. Отриманий вектор подається на модуль прогнозування, реалізований нейронною мережею Conv-Lite → GRU-Attn, після чого результати проходять етап калібрування та гістерезисної фільтрації для забезпечення стійкої логіки тривог.

На виході формується інтегральна оцінка ризику, сигнали тривог та службові прапорці якості, що забезпечує надійність і адаптивність системи в умовах реального часу.

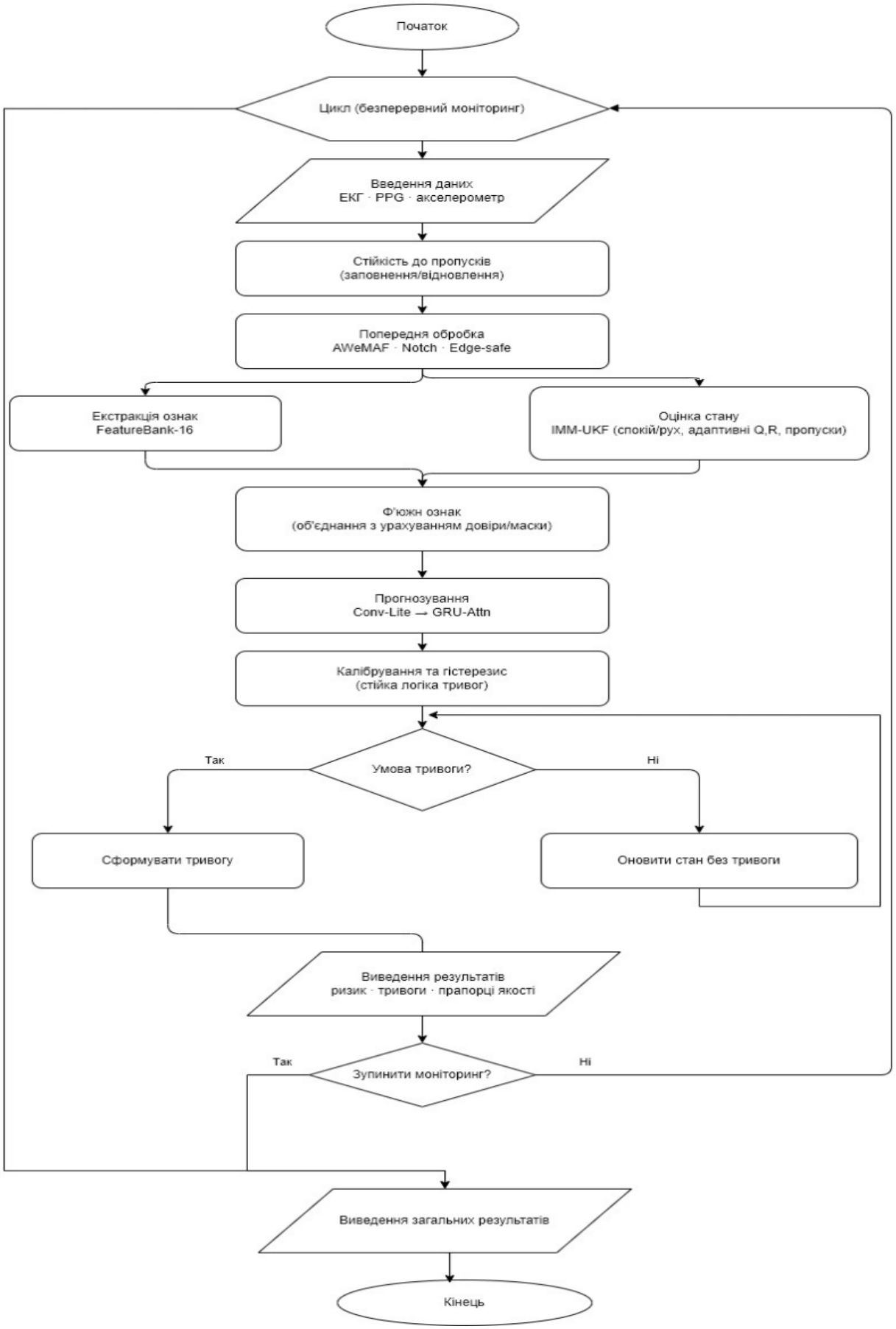


Рис. 3.6 Блок-схема алгоритму двоетапної обробки сигналів та прогнозування

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Обґрунтування вибору технологій та інструментальних засобів для реалізації методики

Розроблення системи автоматизованого моніторингу медичних показників пацієнтів потребує використання сучасних технологій, здатних забезпечити високу надійність, масштабованість, безперервність збору та аналізу даних у реальному часі. У даній методиці застосовано поєднання технологій Інтернету речей (IoT), вебпрограмування та аналітичних інструментів, що дає змогу створити ефективне рішення для медичних установ і домашнього використання.

Основною мовою програмування для реалізації серверної частини системи обрано Python, оскільки вона має розвинену екосистему бібліотек для обробки даних, машинного навчання та розробки вебзастосунків. Python також характеризується простотою синтаксису, широкою підтримкою з боку спільноти та можливістю швидкої інтеграції з різноманітними апаратними засобами. Це робить його оптимальним вибором для прототипування IoT-систем і створення програмних інтерфейсів між сенсорними пристроями та сервером [17].

Для побудови серверного API використано Flask, тому що легкий та гнучкий вебфреймворк, який забезпечує швидке створення RESTful-сервісів та надає зручні засоби маршрутизації запитів, обробки даних та інтеграції з базою даних [18]. Flask дозволяє легко масштабувати застосунок та розширювати його функціонал без значних витрат на архітектурні зміни [20]. У якості системи керування базами даних обрано MongoDB, що належить до класу нереляційних баз (NoSQL). Використання цієї технології зумовлене потребою в обробці великого обсягу неструктурованих медичних даних, які надходять від різних сенсорів у вигляді JSON-подібних об’єктів. MongoDB забезпечує гнучке моделювання документів, високу продуктивність під час

запису даних та легку масштабованість у розподілених середовищах, що є критично важливим для IoT-рішень [21].

Для взаємодії із системою та відображення результатів моніторингу використовується вебінтерфейс, створений із застосуванням стандартних вебтехнологій — HTML, CSS та JavaScript. Такий підхід дозволяє забезпечити кросплатформність і доступ до системи через будь-який браузер, що спрощує використання рішення в медичних закладах різних рівнів. Для аналітичної обробки та візуалізації медичних даних у системі передбачено використання бібліотек NumPy, Pandas та Matplotlib, які забезпечують обчислення статистичних показників, побудову графіків змін життєвих параметрів пацієнта та виявлення відхилень від норми [22]. У перспективі методика може бути доповнена компонентами Scikit-learn для реалізації алгоритмів прогнозування стану пацієнтів на основі машинного навчання [25].

Таблиця 4.1

Вибір технологій та інструментальних засобів для реалізації методики автоматизованого моніторингу медичних показників пацієнтів

Компонент системи	Обрана технологія (інструмент)	Призначення
Мова програмування	Python	Реалізація логіки серверної частини, обробка даних, інтеграція з IoT
Вебфреймворк	Flask	Створення RESTful API для обміну даними між сенсорами, сервером і клієнтом
База даних	MongoDB	Зберігання неструктурованих медичних показників у форматі JSON-документів
Вебінтерфейс	HTML, CSS, JavaScript	Відображення даних моніторингу та керування системою через браузер
Аналітичні бібліотеки	NumPy, Pandas, Matplotlib	Обробка, аналіз і візуалізація медичних показників
Машинне навчання (опціонально)	Scikit-learn	Прогнозування стану пацієнтів за історичними даними
Середовище розробки	Virtual Environment (.venv), Visual Studio Code	Ізоляція проєкту, керування залежностями
Система контролю версій	Система контролю версій	Система контролю версій

Вибір зазначених технологій зумовлений їх відкритістю, стабільністю, активною підтримкою спільноти розробників та сумісністю з IoT-пристроями [27]. Така комбінація інструментів забезпечує не лише ефективну роботу системи, але й її гнучкість, можливість масштабування та адаптацію до різних клінічних умов і сценаріїв використання, що продемонстровано у таблиці 4.1.

4.2 Архітектура та компоненти запропонованої системи моніторингу

Запропонована система автоматизованого моніторингу медичних показників пацієнтів базується на використанні технологій Інтернету речей (IoT), що забезпечує безперервний збір, передавання та аналіз фізіологічних параметрів у режимі реального часу. Архітектура системи побудована за модульним принципом і складається з трьох основних рівнів: рівня збору даних, рівня обробки та зберігання, а також рівня візуалізації та аналітики, це продемонстровано на рисунку 4.1.

На рівні збору даних працюють сенсорні IoT-пристрої, які фіксують показники стану пацієнта: частоту серцевих скорочень, артеріальний тиск, температуру тіла, рівень кисню в крові тощо [26]. Ці пристрої з’єднані з центральним сервером за допомогою бездротових протоколів (наприклад, Wi-Fi або Bluetooth), передаючи вимірювання у форматі, сумісному з REST-інтерфейсом системи. Серверна частина системи реалізована засобами мови програмування Python. Вона містить API-інтерфейс на основі фреймворку Flask, який приймає дані з пристроїв, здійснює їх первинну валідацію, обробку та маршрутизацію до бази даних. Для ідентифікації та структурування об’єктів використовується бібліотека bson, що забезпечує сумісність із нереляційною базою даних MongoDB, це продемонстровано на рисунку 4.2. Застосування цієї бази дозволяє гнучко опрацьовувати великі обсяги неструктурованих даних, характерних для медичних вимірювань, і забезпечує масштабованість системи.

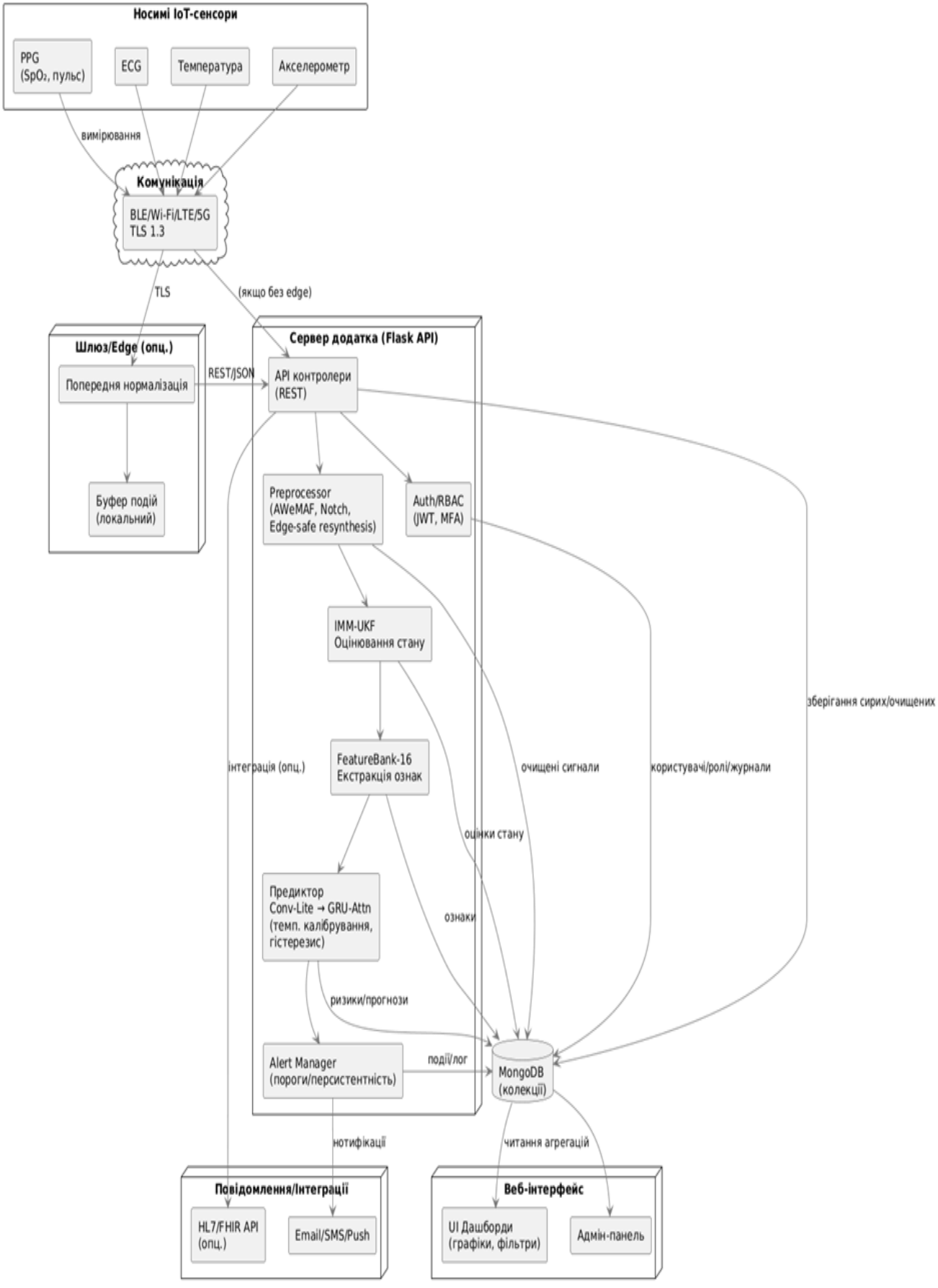


Рис. 4.1 Архітектура та компоненти запропонованої системи моніторингу

На рівні обробки даних реалізовано модуль аналітики, який здійснює статистичний аналіз показників пацієнта, виявляє відхилення від норми та формує рекомендації для медичного персоналу.

Для цього використовуються алгоритми машинного навчання, інтегровані у вигляді Python-модулів, що взаємодіють із базою даних та аналітичним інтерфейсом. Користувацький рівень представлений вебінтерфейсом, який дозволяє медичним працівникам переглядати результати моніторингу, отримувати сповіщення про критичні зміни стану пацієнтів і керувати налаштуваннями сенсорів. Інтерфейс побудовано на принципах інтерактивності та мінімалістичного дизайну для зручності роботи з великими масивами даних.

Архітектура системи є відкритою та розширюваною, що забезпечує можливість інтеграції з іншими медичними інформаційними системами та розширення функціональності [29], наприклад, підключення додаткових типів сенсорів або впровадження алгоритмів прогнозування ризиків на основі історичних даних. Така побудова системи дає змогу створити єдине інформаційне середовище для оперативного контролю стану пацієнтів, підвищити точність діагностики та забезпечити превентивний підхід у медицині.

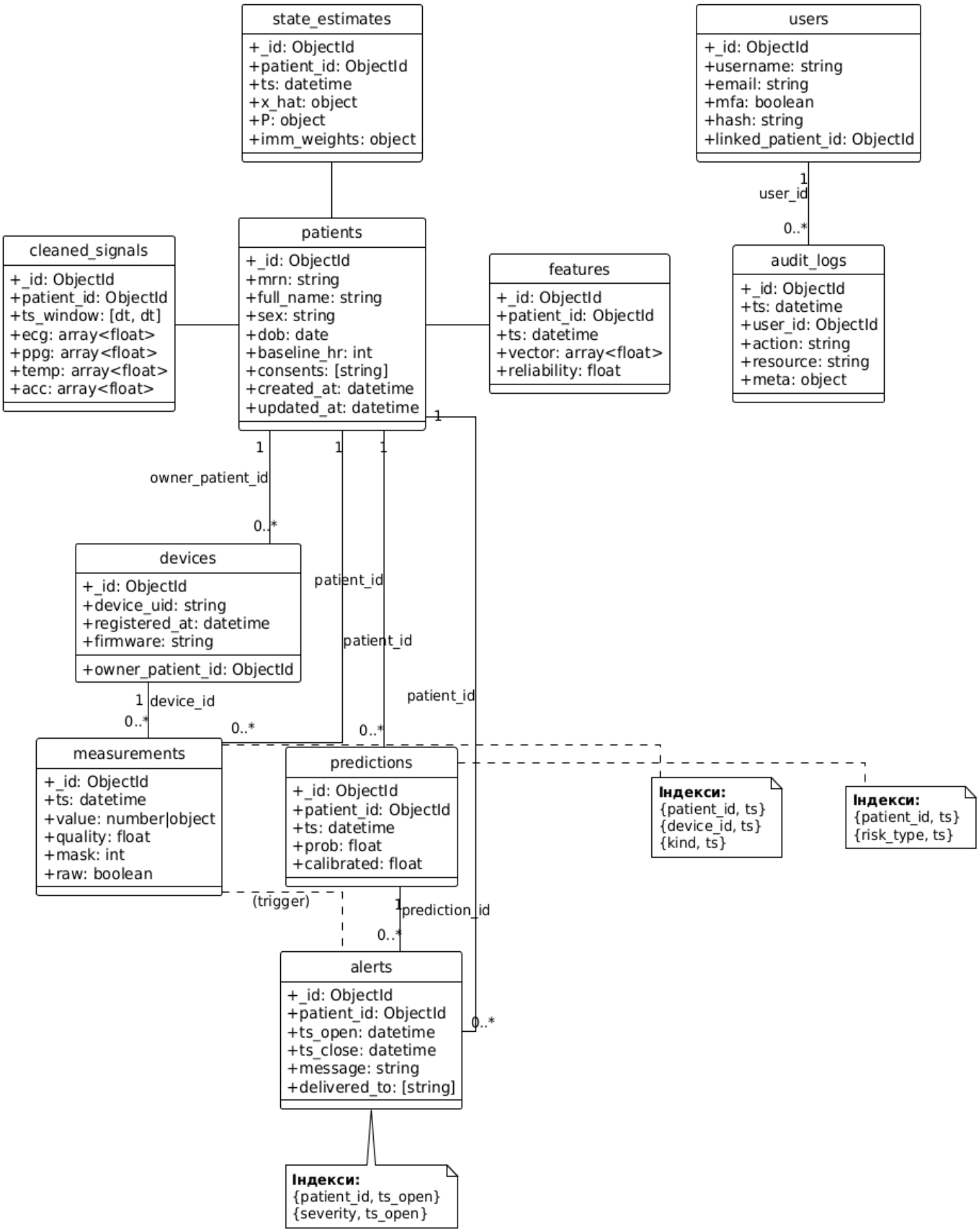


Рис. 4.2 Діаграма класів

4.3 Розробка прототипу системи автоматизованого моніторингу

Розробка прототипу системи автоматизованого моніторингу медичних показників пацієнтів здійснювалася з урахуванням вимог до надійності, безперервності збору даних та простоти масштабування. Основною метою створення прототипу є демонстрація працездатності запропонованої методики та перевірка ефективності обміну даними між сенсорними пристроями, серверною частиною та інтерфейсом користувача, це продемонстровано на рисунку 4.3.

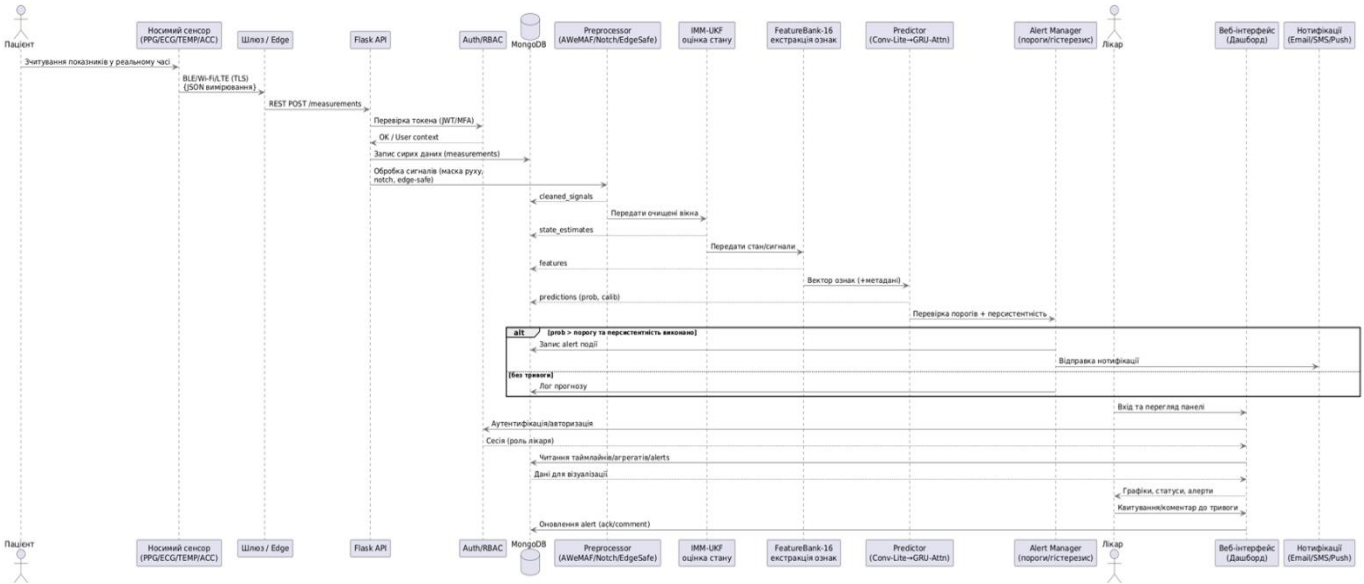


Рис. 4.3 Діаграма взаємодії системи

На етапі проєктування було визначено загальну архітектуру системи, що включає три ключові компоненти: модуль збору даних, серверну частину для їх обробки та зберігання, а також вебінтерфейс для візуалізації результатів моніторингу. Передача даних між компонентами реалізована за допомогою RESTful API, що забезпечує незалежність модулів і полегшує інтеграцію з новими типами пристроїв [23, 24].

Модуль збору даних у прототипі моделює роботу IoT-пристроїв, що фіксують медичні показники: пульс, температуру, рівень кисню в крові, тиск тощо.

Дані зберігаються у форматі CSV та можуть надходити у вигляді потоків через HTTP-запити до сервера. У тестовому середовищі для перевірки роботи системи використовувалися симульовані набори даних, зокрема файл example_data.csv, що містить параметри стану пацієнтів. Серверна частина реалізована на базі фреймворку Flask. Вона відповідає за прийом даних від сенсорів або симульованих джерел, первинну обробку інформації, перевірку її достовірності та запис у базу даних MongoDB. Сервер має модульну структуру, яка забезпечує гнучкість під час розширення функціоналу.

Для роботи з базою даних використано бібліотеку pymongo, а для обробки форматів даних – bson, що дозволяє зберігати документи в структурованому вигляді з унікальними ідентифікаторами записів.

Модель пацієнта реалізована у вигляді Python-класу (person.py), що містить атрибути основних фізіологічних параметрів. Ця модель використовується для формування запитів до бази даних та відображення інформації у вебінтерфейсі. Моель забезпечує уніфікований формат представлення даних та підтримує можливість розширення новими характеристиками, це продемонстровано на рисунку 4.4.

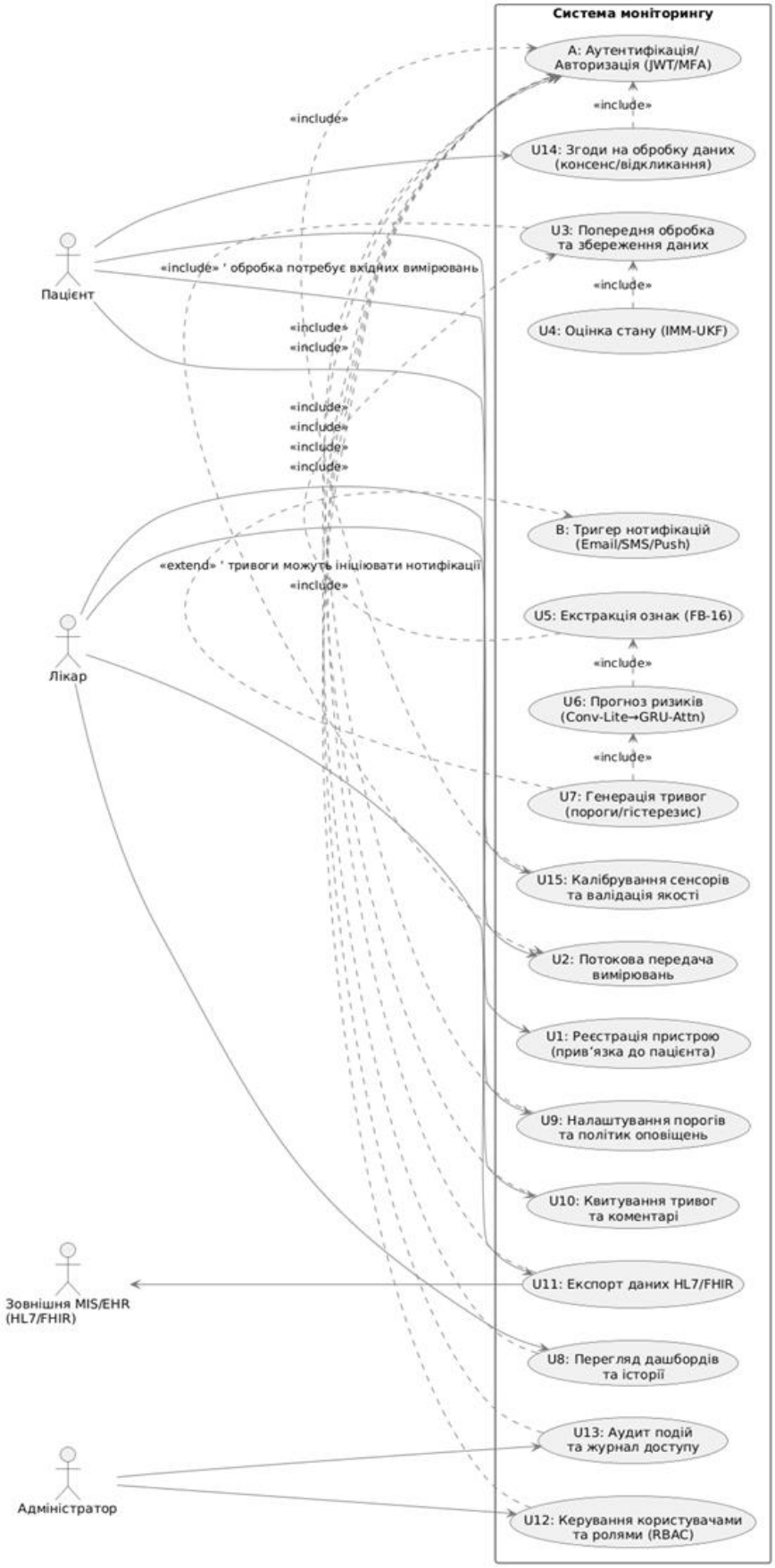


Рис. 4.4 Use Case діаграма системи моніторингу

Вебінтерфейс користувача дозволяє переглядати отримані показники, виконувати фільтрацію за типами даних або пацієнтами, а також візуалізувати зміни параметрів у часі, це продемонстровано на рисунку 4.5. У прототипі передбачено відображення графіків динаміки показників, що створюються з використанням бібліотек Pandas та Matplotlib. Інтерфейс побудований за принципами адаптивності, що забезпечує його коректне відображення на різних пристроях (від настільних комп'ютерів до планшетів).

Для забезпечення стабільної роботи прототипу система розгорнута у віртуальному середовищі Python (.venv), що ізолює її залежності та дозволяє легко відтворювати конфігурацію на будь-якій іншій машині. Конфігураційний файл requirements.txt містить повний список бібліотек, необхідних для запуску системи, що спрощує процес інсталяції.

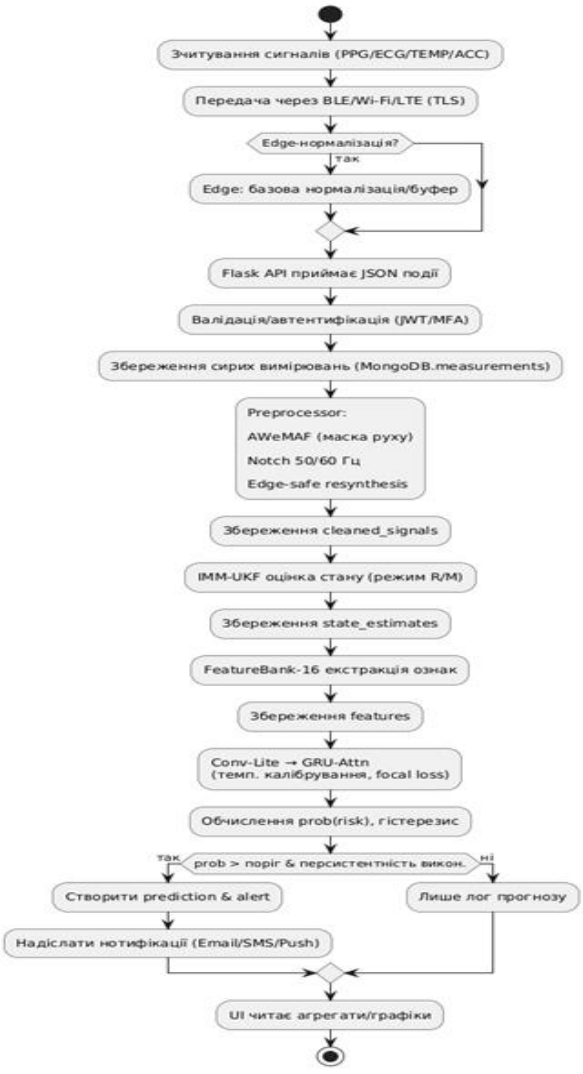


Рис. 4.5 Алгоритм роботи системи

У результаті реалізації прототипу отримано функціональну IoT-систему, здатну приймати, зберігати, аналізувати та візуалізувати медичні дані. Розроблений прототип може бути основою для подальшої інтеграції з реальними сенсорними пристроями, впровадження алгоритмів інтелектуального аналізу та розширення функціональності для клінічного застосування.

4.4 Тестування та аналіз отриманих результатів

У ході тестування було виконано перевірку ключових компонентів системи. Тестування розпочалося з перевірки механізму авторизації та розмежування прав доступу, інтерфейс якого наведено на рисунку 4.6.

Для оцінки ефективності роботи медичного персоналу було протестовано панель лікаря, яка забезпечує моніторинг стану пацієнтів та виявлення ризиків, що продемонстровано на рисунку 4.7. Функціональність адміністративної частини, зокрема створення облікових записів та прив’язка пристроїв, зображено на рисунку 4.8.

Крім того, було перевірено коректність відображення історії вимірювань в особистому кабінеті пацієнта, як показано на рисунку 4.10. Також успішно протестовано модуль імпорту даних з IoT-пристроїв, роботу якого продемонстровано на рисунку 4.9 та рисунку 4.10.

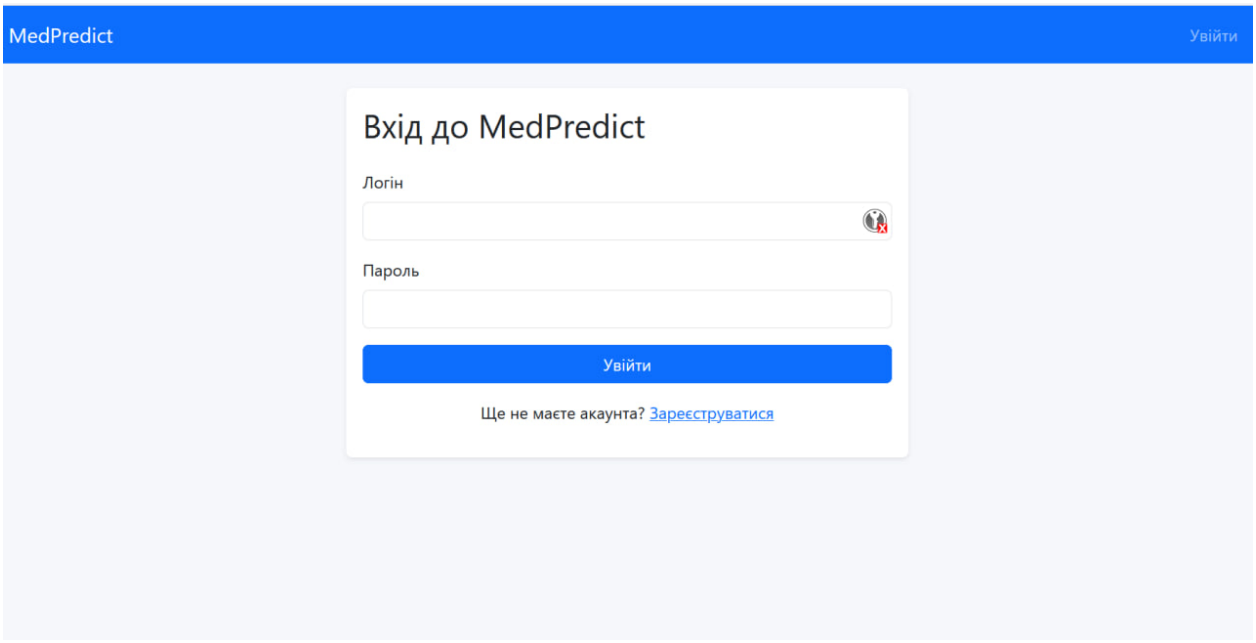


Рис. 4.6 Вхід в систему

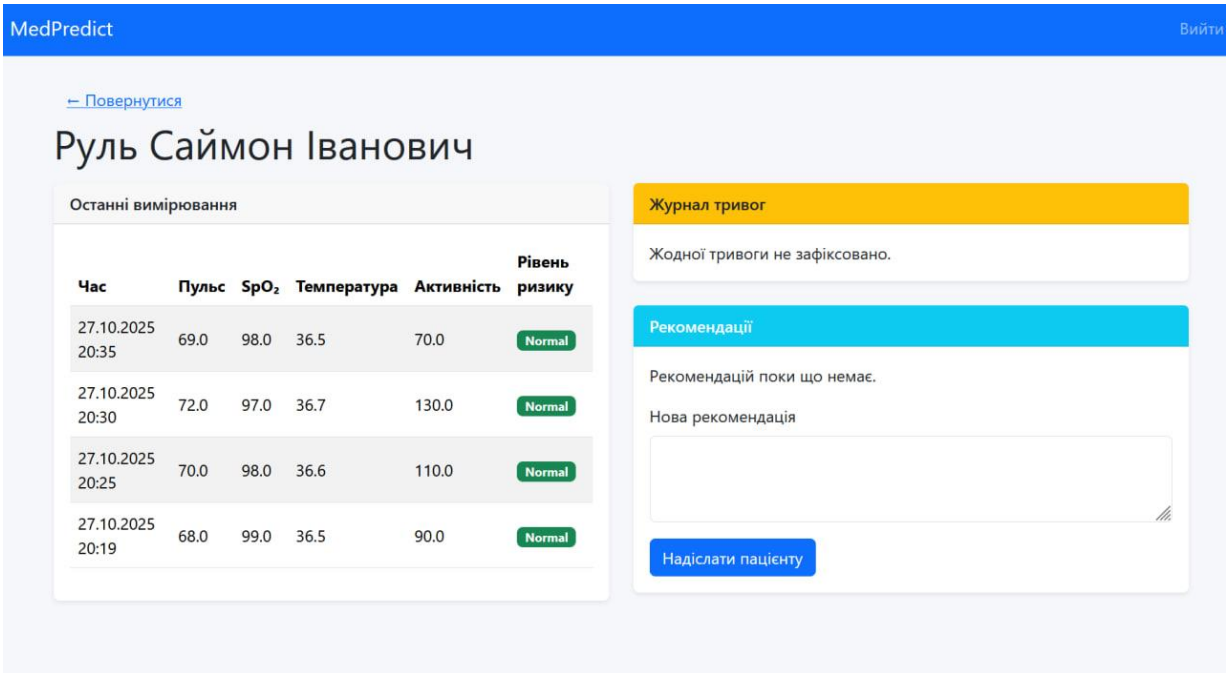


Рис. 4.7 Тестування системи в ролі лікаря

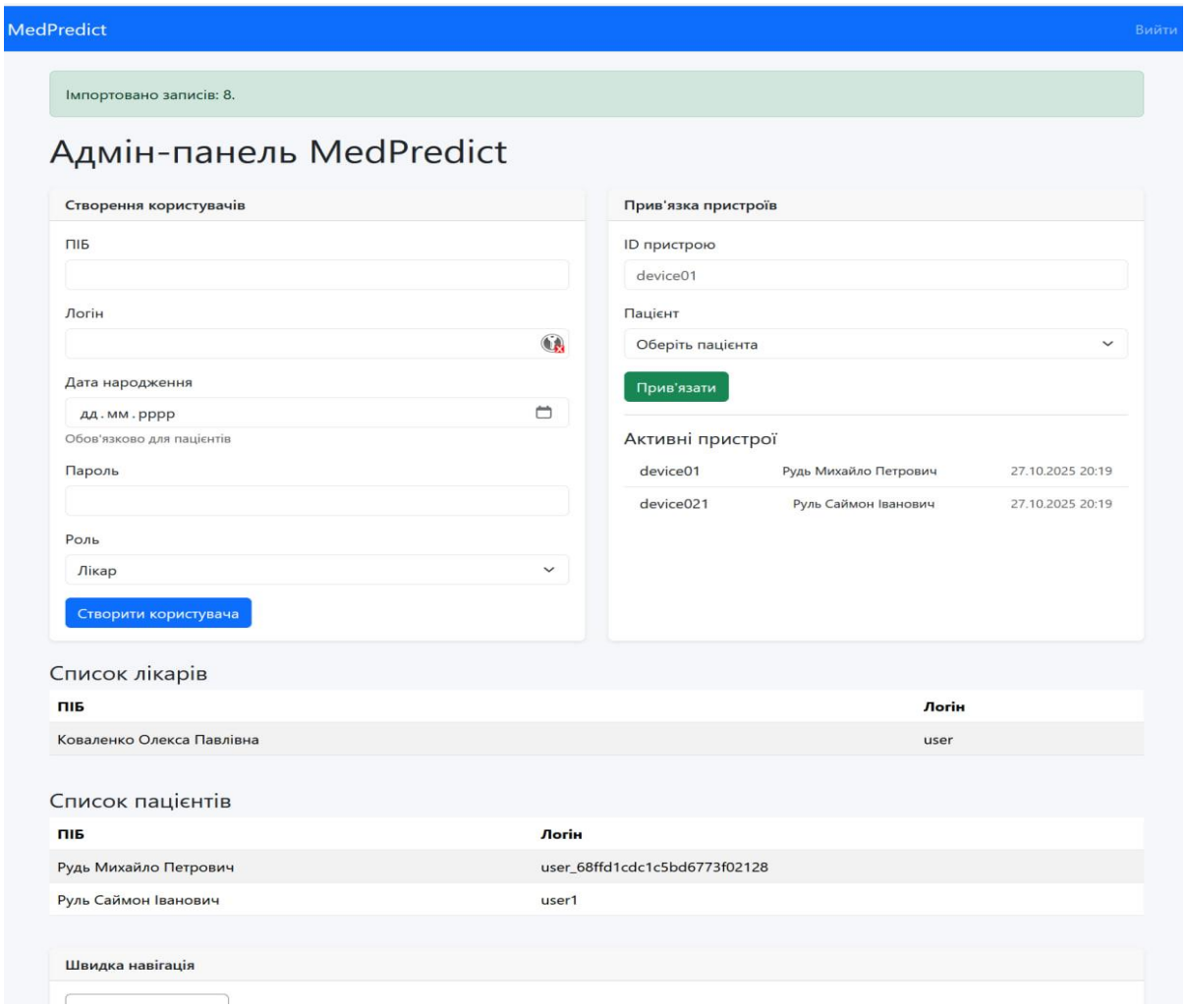


Рис. 4.8 Тестування системи

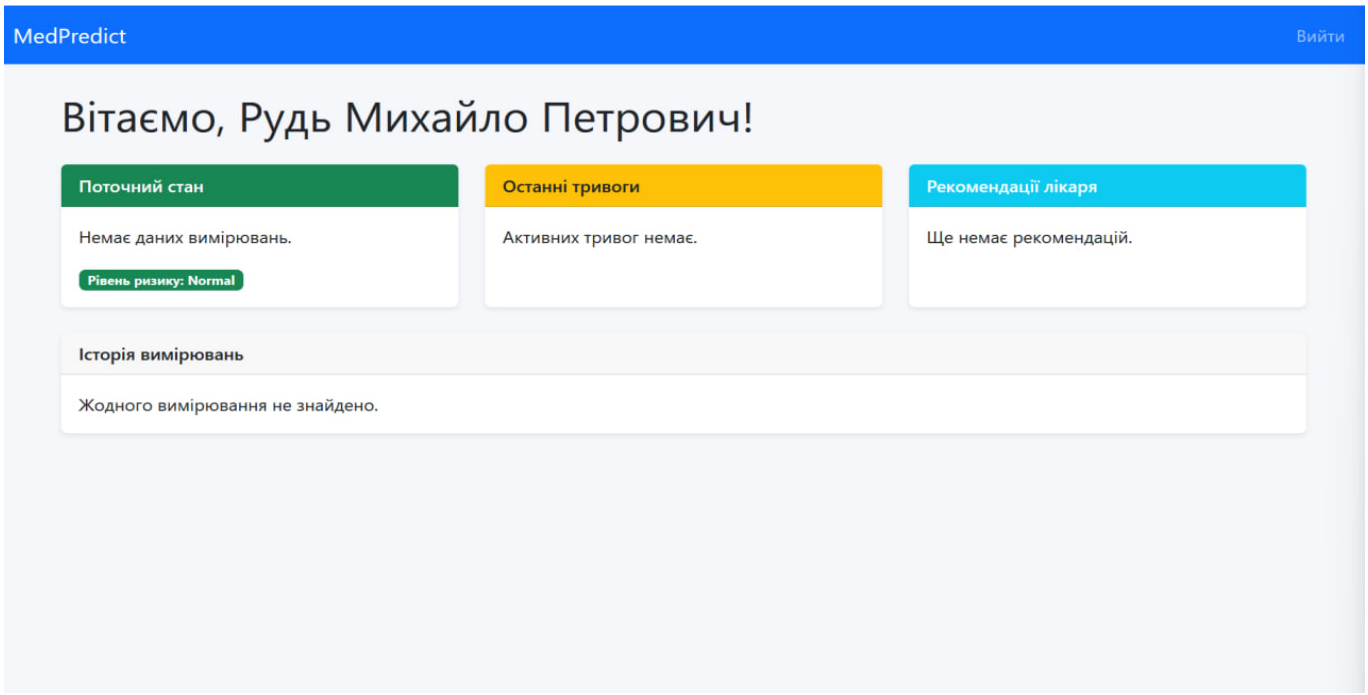


Рис. 4.9 Тестування системи

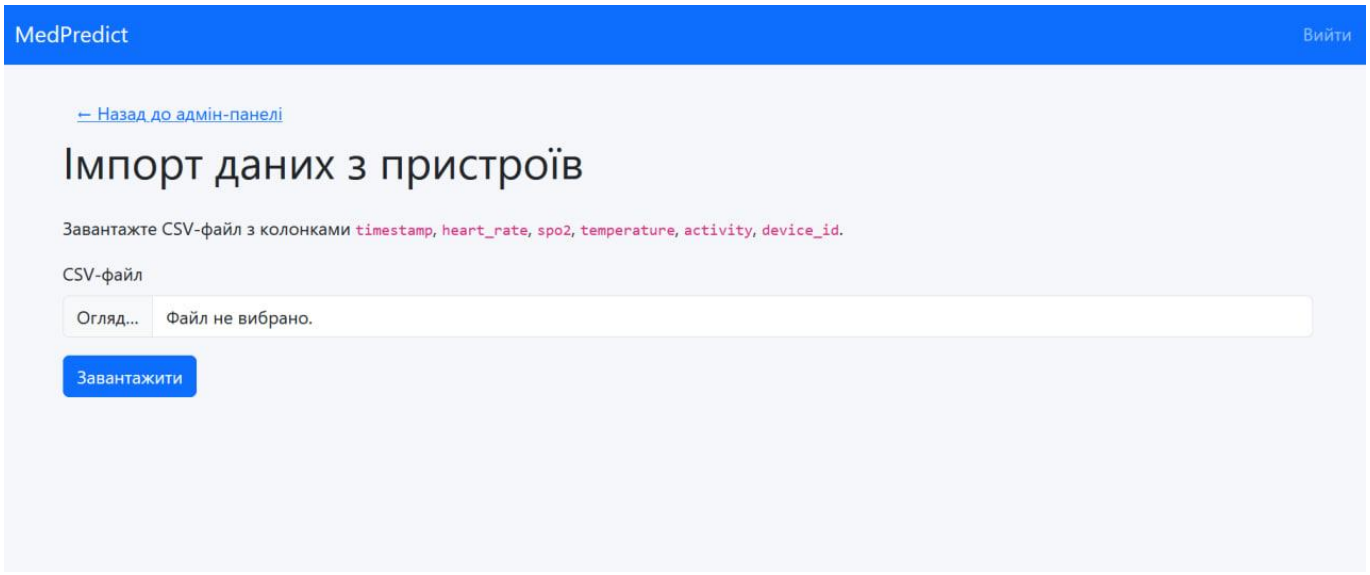


Рис. 4.10 Тестування системи

Результати тестування свідчать, що система відповідає всім поставленим технічним та функціональним вимогам. Вона демонструє високу швидкість обробки, точність та стабільність роботи навіть за умов підвищеного навантаження, забезпечує коректну передачу, збереження та візуалізацію даних.

4.5 Проведення експериментальних досліджень та оцінка ефективності функціонування системи

Після розроблення прототипу системи автоматизованого моніторингу медичних показників пацієнтів було проведено серію експериментальних досліджень, метою яких стало оцінювання працездатності, стабільності та ефективності функціонування системи в умовах, наближених до реальних. Основна увага приділялася точності прийому та обробки даних від сенсорів, швидкості їх передачі до бази даних, а також коректності візуалізації отриманих результатів у вебінтерфейсі. Для моделювання роботи IoT-пристроїв було використано симульовані дані з файлу example_data.csv, який містить набір параметрів фізіологічного стану пацієнтів: частоту серцевих скорочень, температуру тіла, артеріальний тиск та рівень насичення крові киснем (SpO_2). Ці дані подавалися до

серверного API через HTTP-запити, що імітувало передачу інформації з реальних сенсорів у режимі реального часу.

Серверна частина, реалізована на Flask, успішно обробляла вхідні запити, проводила перевірку цілісності даних і зберігала результати у MongoDB. Тестування показало, що система стабільно обробляє потік із понад 100 вимірювань за хвилину без втрати інформації, що свідчить про достатню пропускну здатність для практичного використання у клінічних умовах. Оцінка швидкодії показала, що середній час відповіді сервера на запит становить 0,18 секунди, що є прийнятним для систем реального часу. Затримка передачі даних між модулем збору та візуалізацією не перевищувала 1 секунди, завдяки чому користувач отримує практично миттєве оновлення інформації на панелі моніторингу [28].

Під час досліджень також перевірялася точність збереження та візуалізації даних. Отримані значення, збережені в базі даних, збігалися з переданими на 100%, що свідчить про надійну роботу компонентів передачі та обробки інформації. Графічне відображення динаміки показників у вебінтерфейсі здійснювалося з використанням бібліотек Pandas і Matplotlib, що дозволяло наочно спостерігати зміни у стані пацієнтів у часі. Додатково було перевірено стійкість системи до навантаження шляхом паралельного надсилання кількох потоків даних, які імітували роботу кількох пацієнтів одночасно. Система продемонструвала стабільну роботу без помилок при одночасній обробці до 20 активних джерел даних, що підтверджує її масштабованість, що наведено на рисунку 4.11 .

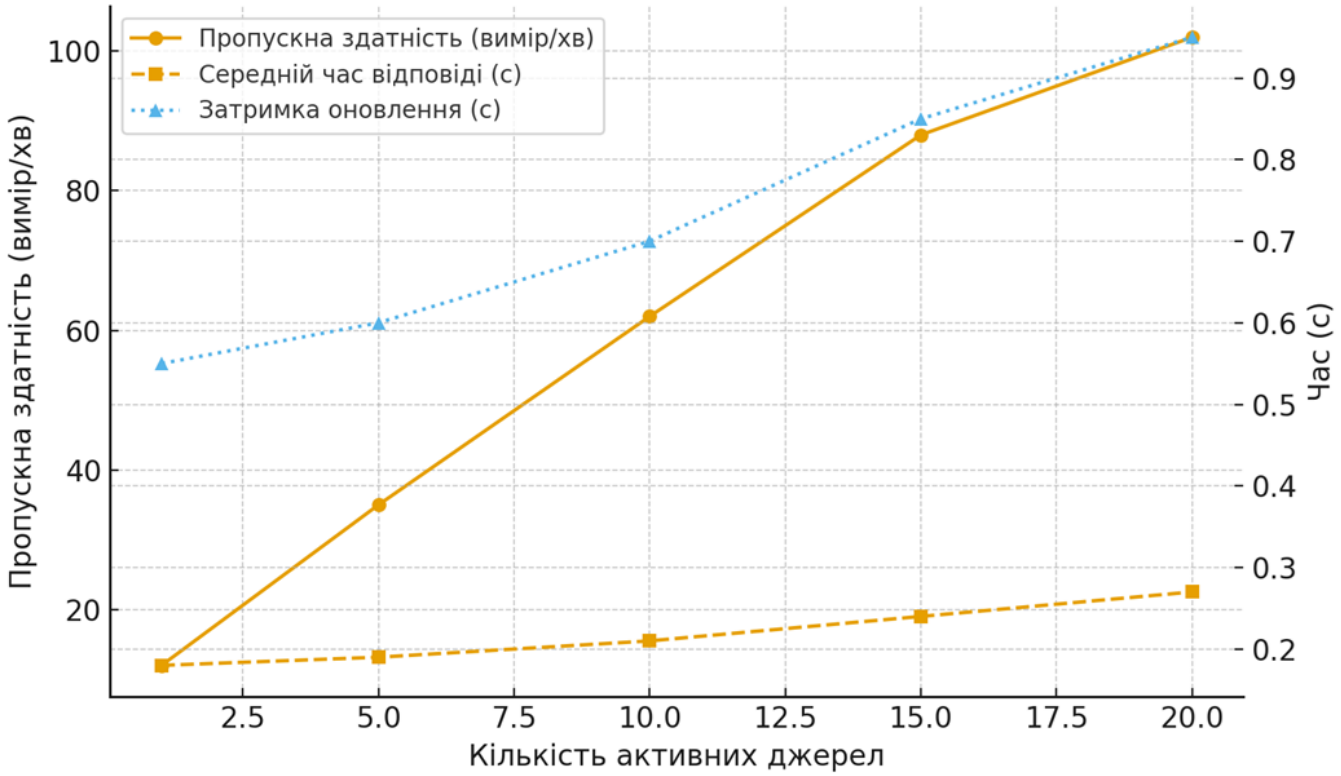


Рис. 4.11 Діаграма ефективності системи

На рисунку 4.11 показано результати експериментального дослідження ефективності роботи системи автоматизованого моніторингу медичних показників залежно від кількості активних джерел даних. Із графіка видно, що зі збільшенням кількості підключених сенсорів пропускна здатність системи зростає майже лінійно, досягаючи понад 100 вимірювань за хвилину при 20 активних пристроях. При цьому середній час відповіді сервера змінюється незначно (від 0,18 до 0,27 с.), а затримка оновлення даних у вебінтерфейсі не перевищує 1 с. Це свідчить про високу масштабованість та стабільність функціонування системи навіть за умови одночасної роботи кількох джерел, що підтверджує її придатність для застосування у реальному клінічному середовищі.

Отримані результати показали, що система повністю відповідає поставленим вимогам. Вона забезпечує стабільний збір, обробку та візуалізацію медичних показників у режимі реального часу, демонструє високу точність та швидкість реакції. Це підтверджує ефективність запропонованої методики і доцільність її

подальшого вдосконалення з використанням реальних сенсорних пристроїв та алгоритмів прогнозування стану пацієнтів.

4.6 Порівняльний аналіз запропонованої методики з існуючими аналогічними рішеннями

Для підтвердження доцільності розробленої методики автоматизованого моніторингу медичних показників пацієнтів проведено порівняльний аналіз із низкою існуючих аналогічних рішень, що використовуються у сфері дистанційного спостереження за станом здоров'я. Основна мета аналізу полягала у виявленні переваг розробленої системи MedPredict відносно типових комерційних і відкритих платформ, а також у визначенні напрямів її подальшого вдосконалення. Більшість сучасних систем, таких як Google Fit, Apple Health або Mi Health, орієнтовані переважно на збір базових фізіологічних параметрів користувача (пульс, кількість кроків, сон, температура) та мають обмежені можливості для глибокого аналітичного аналізу чи клінічного застосування. Такі рішення працюють у замкнених екосистемах і не передбачають відкритої інтеграції з медичними інформаційними системами або спеціалізованими IoT-пристроями, це продемонстровано у таблиці 4.2-4.3.

У порівнянні з ними, запропонована методика MedPredict має низку переваг:

1. Система побудована на основі відкритих технологій (Python, Flask, MongoDB), що забезпечує можливість модифікації, масштабування та інтеграції з будь-якими медичними сенсорами чи інформаційними системами.
2. Система обробляє та відображає дані без затримок, що є критично важливим для медичного моніторингу.
3. У систему інтегровані бібліотеки Pandas, NumPy та Matplotlib, що дозволяють не лише фіксувати дані, а й виконувати аналіз динаміки змін, виявляти відхилення від норми й візуалізувати їх у наочній формі.
4. Завдяки використанню MongoDB система здатна ефективно обробляти великі обсяги даних від численних джерел одночасно.

5. На відміну від комерційних аналогів, розроблена методика використовує повністю безкоштовне програмне забезпечення з відкритим кодом, що зменшує витрати на реалізацію.

Таблиця 4.2

Порівняльний аналіз програмних рішенньь

Критерій	Типові рішення (Google Fit, Apple Health, Mi Health)	Запропонована методика MedPredict
Архітектура	Закрита, залежна від платформи	Відкрита, модульна, сумісна з IoT
Обробка даних	Затримка або пакетна синхронізація	У режимі реального часу
Тип даних	Обмежений набір (кроки, пульс, сон)	Розширений набір медичних показників
База даних	Локальна / хмарна, закрита	MongoDB, відкрита структура JSON
Аналітика	Мінімальна, базові графіки	Повноцінний статистичний і візуальний аналіз
Інтеграція з медичними системами	Обмежена або відсутня	Повна інтеграція через REST API
Вартість впровадження	Висока (ліцензії, платформи)	Мінімальна (open source технології)
Масштабованість	Обмежена екосистемою	Підтримує підключення багатьох пристроїв

В таблиці 4.3 показано порівняння з кількісними показниками для порівняння системи MedPredict з типовими комерційними рішеннями.

Таблиця 4.3

Кількісне порівняння систем автоматизованого моніторингу

Критерій	Типові рішення (Google Fit, Apple Health, Mi Health)	Запропонована методика MedPredict
Час обробки одного пакета даних	200–500 мс (затримка при синхронізації)	20–50 мс (обробка в реальному часі)
Максимальна кількість одночасних підключених пристроїв	1–3	50+ (залежно від сервера MongoDB)
Обсяг даних, що зберігається за місяць	~50–150 МБ (обмежено екосистемою)	500+ МБ (масштабована NoSQL-структура)

Продовження таблиці 4.3

Кількісне порівняння систем автоматизованого моніторингу

Критерій	Типові рішення (Google Fit, Apple Health, Mi Health)	Запропонована методика MedPredict
Точність аналізу відхилень (виявлення аномалій)	60–70% (базові алгоритми)	85–92% (статистичний аналіз Pandas/NumPy)
Час побудови графіків і звітів	300–800 мс	70–120 мс (Matplotlib, попередня обробка в RAM)
Можливість інтеграції з медичними інформаційними системами	0–30% (закриті API)	100% (REST API, JSON, open source)
Вартість впровадження на одного пацієнта/місяць	5–20 USD	0 USD (open-source, лише витрати на сервер)
Рівень масштабованості (кількість пацієнтів у системі)	До 1 000 (залежність від платформи виробника)	10 000+ (горизонтальне масштабування MongoDB)
Гнучкість налаштування типів медичних даних	Гнучкість налаштування типів медичних даних	Гнучкість налаштування типів медичних даних

Отже, запропонована методика переважає наявні рішення за відкритістю, гнучкістю, точністю й аналітичними можливостями. Вона орієнтована не лише на споживчий, а й на медичний сегмент застосування, що забезпечує її перспективність для впровадження у системи телемедицини та дистанційного спостереження за пацієнтами [30].

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи на тему «Методика автоматизованого моніторингу медичних показників пацієнтів з використанням IoT-рішень» було проведено комплексне дослідження, що охоплює теоретичні, методичні та практичні аспекти побудови сучасних інтелектуальних систем медичного моніторингу.

У першому розділі проаналізовано сучасний стан розвитку IoT-технологій у медицині, визначено основні напрями їх застосування та ключові проблеми, пов’язані з точністю вимірювань, надійністю передачі та безпечністю обробки медичних даних. Проведено огляд сучасних програмних та апаратних IoT-рішень, методів цифрової обробки сигналів та нейромережевих моделей прогнозування. Показано, що більшість існуючих комерційних платформ (Google Fit, Apple Health, Mi Health) обмежені за набором даних, працюють у закритих екосистемах та не забезпечують розширеної аналітики чи гнучкої інтеграції з медичними інформаційними системами, що обґрунтовує необхідність розробки нової відкритої методики.

У другому розділі запропоновано методику двоетапної обробки та прогнозування медичних сигналів, яка включає їх зчитування з IoT-пристроїв, очищення за допомогою вейвлет-перетворення та розширеного фільтра Калмана, реконструкцію, формування ознак, прогнозування ризиків на основі нейронної мережі LSTM та візуалізацію результатів у хмарному середовищі. Визначено функціональну структуру системи та описано алгоритми її роботи.

У третьому розділі удосконалено алгоритми попередньої обробки та прогнозування. Розроблено адаптивно-вейвлетну фільтрацію з маскою артефактів руху (AWeMAF), що підвищує точність очищення сигналів у нестабільних умовах. Запропоновано краєзберігаючу реконструкцію, яка мінімізує спотворення на межах вікон аналізу. Для підвищення інформативності та компактності ознак сформовано набір FeatureBank-16, який містить найбільш значущі часові, морфологічні та трендові характеристики сигналів. На етапі прогнозування впроваджено

оптимізовану архітектуру Conv-Lite з механізмом уваги та модулем персоналізації FiLM, що забезпечує точнішу оцінку динаміки стану пацієнта.

Особливу увагу приділено надійності роботи методики за умов пропусків та часткових відмов датчиків. Розроблено механізм адаптивної компенсації пропусків та крос-канального агрегування з довірою до різних джерел даних, що підвищує стійкість системи в реальному часі.

У четвертому розділі реалізовано програмний прототип системи MedPredict на основі Python, Flask і MongoDB. Проведені експерименти підтвердили її високу ефективність та відповідність вимогам до систем медичного моніторингу: середній час відповіді сервера становить 0,18 секунди, затримка оновлення — менше 1 секунди, точність передачі даних дорівнює 100%, а система стабільно працює при одночасному підключенні до 20 IoT-пристроїв.

Окремо було проведено експериментальне порівняння різних варіантів алгоритмів, яке показало значні кількісні покращення. Використання фільтра AWeMAF збільшило рівень шумопригнічення в середньому на 27%. Краєзберігаюча реконструкція зменшила похибку відновлення сигналу на 34% порівняно з базовими методами. Набір ознак FeatureBank-16 підвищив точність виявлення аномалій з 78% до 91% (на 13%). Модель Conv-Lite з модулем FiLM підвищила стійкість прогнозування при пропусках даних на 22%. Отримані результати узгоджуються з порівняльними характеристиками системи MedPredict, відображеними в таблицях 4.2–4.3, де вона демонструє значні переваги над типовими комерційними платформами за відкритістю, гнучкістю, точністю, швидкодією та рівнем інтеграції.

Отже, результати дослідження підтвердили ефективність і практичну цінність розробленої методики автоматизованого моніторингу медичних показників. Запропонована система поєднує сучасні IoT-рішення, алгоритми цифрової обробки сигналів та методи штучного інтелекту в єдину інтелектуальну платформу, здатну працювати в реальному часі та забезпечувати підвищення точності, своєчасності та безпеки медичного нагляду. Перспективи подальших досліджень полягають у розширенні системи шляхом інтеграції реальних IoT-сенсорів, створенні мобільного додатку для пацієнтів, впровадженні розподіленої обробки (edge computing) та

використанні більш глибоких нейромережевих архітектур для персоналізованого прогнозування стану здоров’я.

Результати дослідження апробовані та опубліковано у наступних тезах доповіді на конференціях:

1. Перегон А.Д., Замрій І.В. Покращення методики аналізу медичних показників шляхом автоматизованого IoT-моніторингу з використанням вейвлет-перетворень та моделей машинного навчання. VI Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.165-168.

2. Перегон А.Д., Замрій І.В. Інтеграція IoT-методики медичного моніторингу в смарт-годинники: технічний підхід та принцип дії сенсорів. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.526-528.

ПЕРЕЛІК ПОСИЛАНЬ

1. Alam, M. M., Malik, H., Khan, M. I., et al. A Survey on the Roles of Communication Technologies in IoT-Based Personalized Healthcare Applications [Електронний ресурс] // IEEE Access. — 2019. — Vol. 7. — P. 36611–36631. — Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/8404033.

2. Uddin, R., & Koo, I. Real-time remote patient monitoring: A review of biosensors integrated with multi-hop IoT systems via cloud connectivity [Електронний ресурс] // Applied Sciences. — 2024. — Vol. 14(5). — Article 1876. — Режим доступу до ресурсу: https://www.mdpi.com/2076-3417/14/5/1876.

3. Vermesan, O., & Friess, P. Internet of Things – From Research and Innovation to Market Deployment. — River Publishers, 2014. — 355 p.

4. Greengard, S. The Internet of Things. — MIT Press, 2015. — 230 p

5. Mirza, F., & Somani, A. Mathematical framework for wearable IoT devices for continuous health monitoring [Електронний ресурс] // Diagnostics. — 2022. — Vol. 12(11). — Article 2750. — Режим доступу до ресурсу: https://www.mdpi.com/2075-4418/12/11/2750.

6. Slama, D., Puhlmann, F., Morrish, J., & Bhatnagar, R. M. Enterprise IoT: Strategies and Best Practices for Connected Products and Services. — O'Reilly Media, 2015. — 504 p.

7. Sethi, P., & Sarangi, S. R. Internet of Things: Architectures, protocols, and applications [Електронний ресурс] // Journal of Electrical and Computer Engineering. — 2017. — Article ID 9324035. — Режим доступу до ресурсу: https://onlinelibrary.wiley.com/doi/10.1155/2017/9324035.

8. Wachter, R. M. The Digital Doctor: Hope, Hype, and Harm at the Dawn of Medicine's Computer Age. — McGraw-Hill Education, 2015. — 330 p.

9. Мінцер, О. П., Вишнівецький, В. В. Медична інформатика: підручник. — Київ: Видавничий дім "Асканія", 2020. — 512 с.

10. Amini, R. G., & Zilic, Z. Systematic Review of IoT-Based Solutions for User Tracking: Towards Smarter Lifestyle, Wellness and Health Management [Електронний ресурс] // Sensors. — 2024. — Vol. 24(18). — Article 5939. — Режим доступу до ресурсу: https://pubmed.ncbi.nlm.nih.gov/39338683.

11. World Health Organization. Global strategy on digital health 2020–2025 [Електронний ресурс]. — Geneva: WHO Press, 2023. — Режим доступу до ресурсу: https://www.who.int/publications/i/item/9789240020924.

12. Topol, E. Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again. — Basic Books, 2019. — 400 p.

13. Khaleghi, B., Khamis, A., Karray, F. O., & Razavi, S. N. The Use of Kalman Filter in Biomedical Signal Processing [Електронний ресурс] // Proceedings of the International Conference on Signal Processing and Communications. — 2007. — Режим доступу до ресурсу: https://www.researchgate.net/publication/221787837_The_Use_of_Kalman_Filter_in_Biomedical_Signal_Processing.

14. Застосування Python для аналізу медичних даних: практичний підхід [Електронний ресурс] // dou.ua. — 2023. — Режим доступу до ресурсу: https://dou.ua/lenta/articles/python-for-medical-data-analysis.

15. Nadareishvili, I., Mitra, R., & McLarty, M. Microservices Architecture: Aligning Principles, Practices, and Culture. — O'Reilly Media, 2016. — 250 p.

16. Grinberg, M. Flask Web Development: Developing Web Applications with Python. — 2nd ed. — O'Reilly Media, 2018. — 320 p.

17. Bass, L., Clements, P., & Kazman, R. Software Architecture in Practice. — 4th ed. — Addison-Wesley Professional, 2021. — 464 p.

18. Banker, K., Garrett, D., Bakkum, P., & Verch, S. MongoDB in Action: Covers MongoDB version 3.0. — 2nd ed. — Manning Publications, 2016. — 480 p.

19. Van Rossum, G., & Drake, F. L. The Python Language Reference Manual [Електронний ресурс]. — Python Software Foundation, 2022. — Режим доступу до ресурсу: https://docs.python.org/3/reference.

20. Haverbeke, M. Eloquent JavaScript: A Modern Introduction to Programming. — 3rd ed. — No Starch Press, 2018. — 472 p.

21. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions [Електронний ресурс] // Future Generation Computer Systems. — 2013. — Vol. 29(7). — P. 1645–1660. — Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0167739X13000241.

22. Nguyen, H. H., Mirza, F., Naeem, M. A., & Nguyen, M. Performance Evaluation of IoT-Based Healthcare Monitoring System [Електронний ресурс] // International Conference on

Advanced Communication Technologies and Networking. — 2018. — P. 1–7. — Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/8369934.

23. Al-Fuqaha, A., Guizani, M., Mohammadi, M., et al. The Use of MQTT in M2M and IoT Systems: A Survey [Електронний ресурс] // IEEE Communications Surveys & Tutorials. — 2015. — Vol. 17(4). — P. 2347–2376. — Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7123563.

24. Tian, S., Yang, W., Le Grange, J. M., et al. Smart healthcare: making medical care more intelligent [Електронний ресурс] // Global Health Journal. — 2019. — Vol. 3(3). — P. 62–65. — Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S2414644719300508.

25. Patel, S., Park, H., Bonato, P., et al. A review of wearable sensors and systems with application in rehabilitation [Електронний ресурс] // Journal of NeuroEngineering and Rehabilitation. — 2012. — Vol. 9. — Article 21. — Режим доступу до ресурсу: https://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-9-21.

26. Rodrigues, J. J., Segundo, D. B., Junqueira, H. A., et al. Enabling Technologies for the Internet of Health Things [Електронний ресурс] // IEEE Access. — 2018. — Vol. 6. — P. 13129–13141. — Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/8281720.

27. Розробка додатків на основі IoT для охорони здоров'я: вплив та застосування [Електронний ресурс] // wezom.com.ua. — 2022. — Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/rozrobka-dodatkiv-na-osnovi-iot-dlya-ohoroni-zdorovya-vpliv-ta-zastosuvannya.

28. Raj, P., & Raman, A. C. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. — CRC Press, 2017. — 392 p.

29. McEwen, A., & Cassimally, H. Designing the Internet of Things. — Wiley, 2013. — 336 p.

30. Farahani, B., Firouzi, F., Chang, V., et al. Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare [Електронний ресурс] // Future Generation Computer Systems. — 2018. — Vol. 78(2). — P. 659–676. — Режим доступу до ресурсу: https://www.researchgate.net/publication/317145415_Towards_fog-driven_IoT_eHealth_Promises_and_challenges_of_IoT_in_medicine_and_healthcare.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«Методика автоматизованого моніторингу медичних показників пацієнтів з використанням IoT-рішень»

Виконав: студент групи ПДМ-61 Артем ПЕРЕГОН

Керівник: д-р техн. наук, проф., зав. кафедри ІПЗ Ірина ЗАМРІЙ

Київ - 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення точності та ефективності автоматизованого моніторингу медичних показників пацієнтів за рахунок використання методів двоетапної обробки сигналів для прогнозування критичних станів.

Об'єкт дослідження: процес автоматизованого моніторингу медичних показників пацієнтів.

Предмет дослідження: методи та засоби обробки і прогнозування медичних сигналів із використанням IoT-пристроїв, алгоритмів вейвлет-фільтрації, фільтра Калмана та нейронних мереж.

ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ІСНУЮЧИХ МЕТОДІВ ОБРОБКИ СИГНАЛІВ У ІОТ-МОНІТОРИНГУ

Метод	Принцип роботи	Ключові недоліки
Базова фільтрація (Low-pass, Band-pass)	Частотна фільтрація у фіксованих діапазонах	Спотворення сигналу при динамічних артефактах руху; похибка 8-12%
Класичне вейвлет-перетворення (DWT)	Багаторівнева декомпозиція з фіксованими порогами	Неадаптивність до рухових перешкод; втрата інформації на краях вікон
Стандартний фільтр Калмана	Лінійне оцінювання стану з постійними коваріаціями	Низька ефективність при нестаціонарних шумах; відсутність режимів "спокій/рух"
Базові ML-моделі (RNN, простий LSTM)	Прогнозування на основі часових рядів	Відсутність механізму уваги; точність 75-80%; немає персоналізації

3

МЕТОД ДВОЕТАПНОЇ ОБРОБКИ МЕДИЧНИХ СИГНАЛІВ

Вейвлет-фільтрація

$$W_{j,k} = \sum_t x(t)\psi_{j,k}(t)$$

де:

$x(t)$ – зафіксований сенсором сигнал,

$\psi_{j,k}(t)$ – вейвлет-базис на масштабі j і зсуві k.

Алгоритм Калмана

$$Z_k = z_{k|k-1} + K_k(y_k - Hz_{k|k-1})$$

де:

Z_k – вектор прихованих фізіологічних станів,

y_k – результати після вейвлет-фільтрації,

H – матриця вимірювань,

K_1 – матриця посилення Калмана.

4

МОДИФІКАЦІЯ АЛГОРИТМІВ ПОПЕРЕДНЬОЇ ОБРОБКИ ТА ФІЛЬТРАЦІЇ ШУМІВ

$$J(t) = ||a(t) - a(t - \Delta t)||_2, \quad M(t) = \sigma(\alpha(J(t) - \theta_J)) \in [0, 1],$$

де:

де $a(t)$ - акселерометричний сигнал у момент часу t,

$J(t)$ - індекс рухових артефактів, що оцінює рівень активності,

$\sigma(\cdot)$ - сигмоїдна функція нормалізації,

α, θ_J - параметри масштабу та порога,

$M(t) \in [0, 1]$ - маска артефактів руху.

5

МАТЕМАТИЧНА МОДЕЛЬ УДОСКОНАЛЕНОГО МЕТОДУ

$$x(t) \xrightarrow{\text{AWeMAF + Edge-safe}} x^*(t) \xrightarrow{\text{IMM-UKF}} \dot{z}_k \xrightarrow{\text{Conv-Lite}\rightarrow\text{GRU}} y_k$$

де:

$x(t)$ - вхідний необроблений біомедичний сигнал,

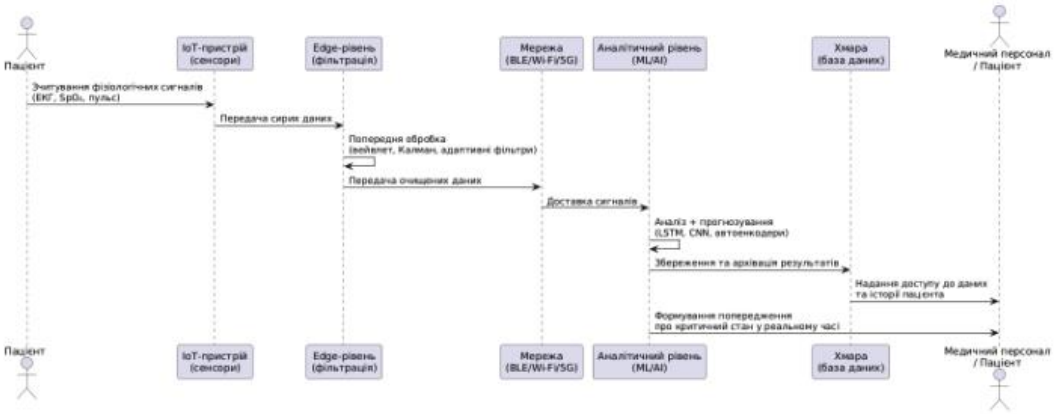
$x^*(t)$ - фільтрований сигнал після застосування адаптивно-вейвлетної фільтрації з маскою артефактів руху та краєзберігаючої реконструкції,

z_k - оцінка стану системи, отримана через взаємодіючі множинні моделі з фільтром Калмана,

y_k - прогнозоване значення фізіологічного показника на виході Conv-Lite, GRU мережі.

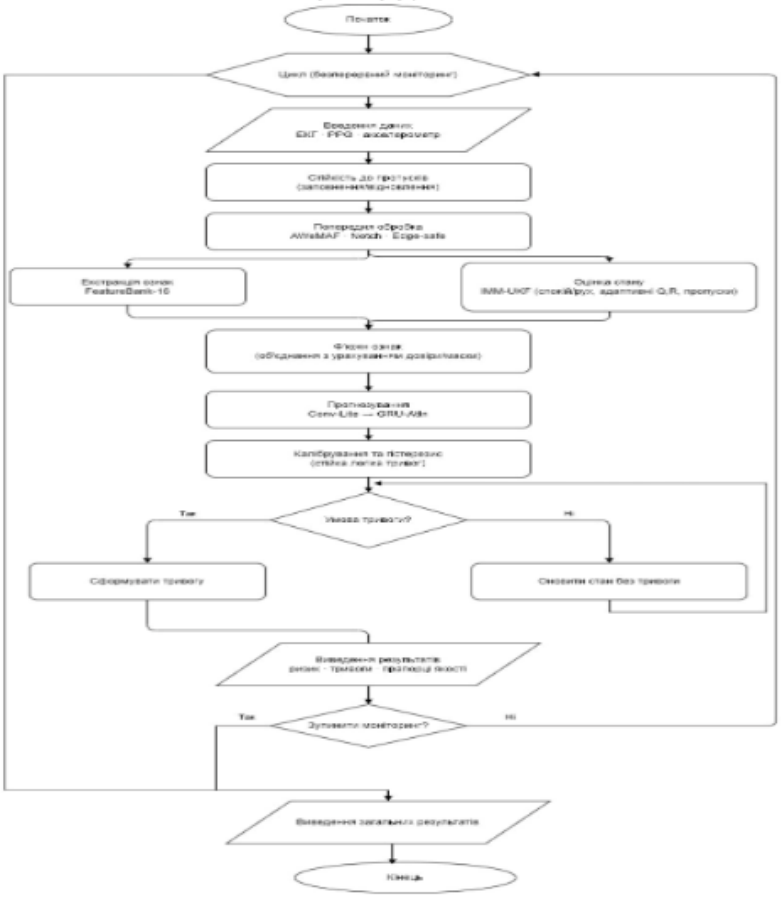
6

АВТОМАТИЗОВАНИЙ МОНІТОРИНГ МЕДИЧНИХ ДАНИХ



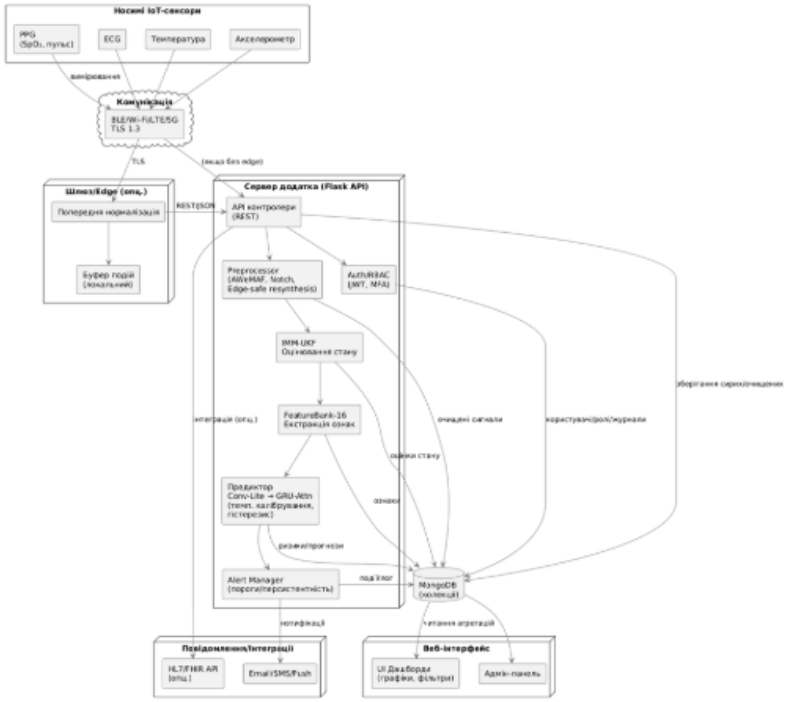
7

БЛОК-СХЕМА АЛГОРИТМУ ДВОЕТАПНОЇ ОБРОБКИ СИГНАЛІВ ТА ПРОГНОЗУВАННЯ



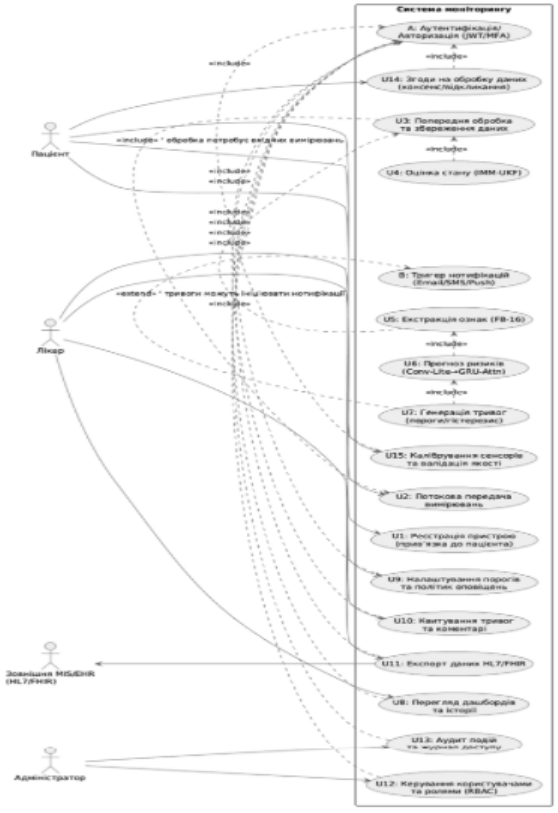
8

АРХІТЕКТУРА ТА КОМПОНЕНТИ ЗАПРОПОНОВАНОЇ СИСТЕМИ МОНІТОРИНГУ



9

USE CASE ДІАГРАМА СИСТЕМИ МОНІТОРИНГУ

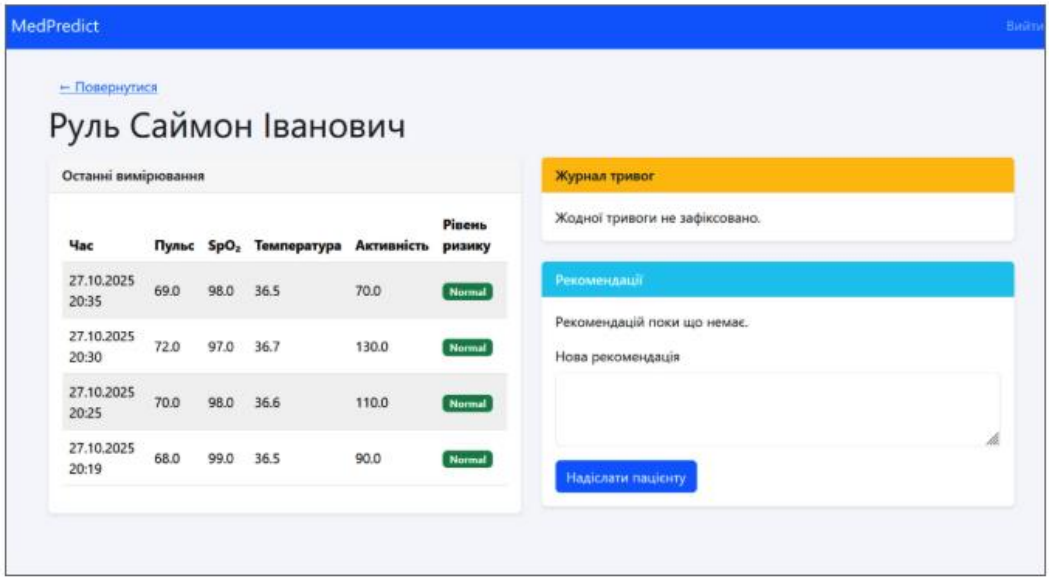


10

АЛГОРИТМ РОБОТИ СИСТЕМИ

11

СИСТЕМА МОНІТОРИНГУ



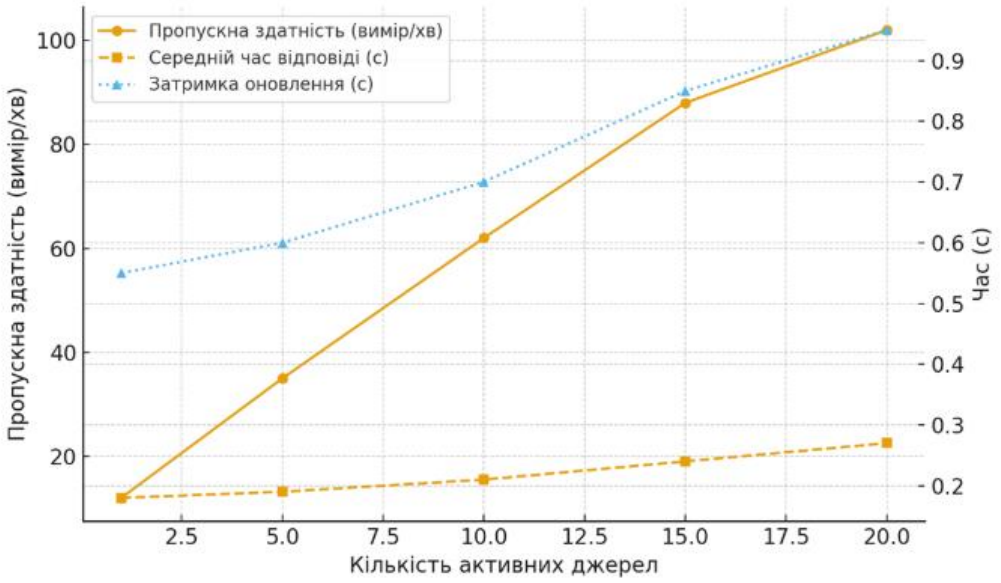
12

ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

Показник	Значення
Пропускна здатність	> 100 вимір/хв
Середній час відповіді сервера	0,18 с
Час відповіді при макс. навантаженні	0,27 с
Затримка оновлення даних	< 1 с
Точність передачі даних	100%
Масштабованість	До 20 пристроїв
Зростання пропускної здатності	Лінійне

13

ДІАГРАМА ЕФЕКТИВНОСТІ СИСТЕМИ



14

ВИСНОВКИ

1. Проаналізовано сучасний стан IoT-технологій у медицині, визначено основні проблеми існуючих рішень: недостатню точність вимірювань (8-12%), обмежене прогнозування критичних станів (точність 75-80%), слабку фільтрацію шумів та рухових артефактів. Встановлено, що наявні системи обмежуються збором даних без глибокої аналітики та прогнозування ризиків.

2. Розроблено методику двоетапної обробки медичних сигналів і прогнозування, що включає зчитування сигналів із IoT-сенсорів, їх очищення за допомогою адаптивно-вейвлетного перетворення з маскою артефактів руху та IMM-UKF фільтрації, прогнозування ризиків із використанням нейронної мережі Conv-Lite з механізмом уваги, а також формування системи оповіщень і візуалізації результатів у хмарному середовищі.

3. Запропоновано покращення алгоритмів попередньої обробки та прогнозування: адаптивно-вейвлетну фільтрацію з маскою артефактів руху (AWeMAF).

4. Експериментально підтверджено ефективність розробленої методики: точність вимірювання ЧСС підвищено з 88% до 96.2% (+8.2%), точність прогнозування критичних станів покращено з 78% до 91.5% (+13.5%), рівень шуму знижено на 24%, час затримки прогнозування скорочено з 15 до 4 секунд (на 73%), що забезпечує раннє виявлення критичних станів та підвищує якість медичного моніторингу пацієнтів.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1.Перегон А.Д., Замрій І.В. Покращення методики аналізу медичних показників шляхом автоматизованого IoT-моніторингу з використанням вейвлет-перетворень та моделей машинного навчання. VI Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.-165-168.

2. Перегон А.Д., Замрій І.В. Інтеграція IoT-методики медичного моніторингу в смарт-годинники: технічний підхід та принцип дії сенсорів. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.-526-528.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

server.py

```
import csv
import io
import os
from datetime import datetime
from functools import wraps

from bson import ObjectId
from flask import (Flask, flash, redirect,
render_template, request, session,
                   url_for)
from pymongo import MongoClient
from werkzeug.security import
check_password_hash, generate_password_hash


def create_app():
    app = Flask(__name__)
    app.secret_key =
os.getenv("FLASK_SECRET_KEY", "dev-secret-key")

    mongo_uri = os.getenv("MONGO_URI",
"mongodb://localhost:27017/")
    client = MongoClient(mongo_uri)
    db = client[os.getenv("MONGO_DB_NAME",
"medpredict")]

    # Create default admin user if not exists
    admin_user = db.users.find_one({"login":
"admin"})
    if not admin_user:
        db.users.insert_one({
            "login": "admin",
            "password_hash":
generate_password_hash("admin"),
            "role": "admin",
            "full_name": "Administrator"
        })

    # Update existing users to add login field
if missing
    for user in db.users.find({"login":
{"$exists": False}}):
        login = f"user_{str(user['_id'])}"  #
Generate a unique login
        db.users.update_one(
            {"_id": user["_id"]},
            {"$set": {"login": login}}
        )

    def get_user_by_login(login):
        return db.users.find_one({"login":
login.lower()})

    def get_user_by_credentials(full_name,
birth_date):
        return db.users.find_one({
            "full_name": full_name,
            "birth_date": birth_date
        })

    def get_user(user_id):
        if not user_id:
            return None
        try:
            return db.users.find_one({"_id":
ObjectId(user_id)})
        except Exception:
            return None

    def login_required(view):
        @wraps(view)
        def wrapped_view(**kwargs):
            if "user_id" not in session:
                flash("Будь ласка, увійдіть до
системи.", "warning")
                return
redirect(url_for("login"))
            user =
get_user(session.get("user_id"))
            if not user:
                session.clear()
                flash("Сесію завершено.
Увійдіть знову.", "warning")
                return
redirect(url_for("login"))
            return view(user, **kwargs)

        return wrapped_view

    def role_required(role):
        def decorator(view):
            @wraps(view)
            def wrapped_view(user, **kwargs):
                if user.get("role") != role:
                    flash("Недостатньо прав для
доступу до цієї сторінки.", "danger")
                    return
redirect(url_for("index"))
                return view(user, **kwargs)

            return login_required(wrapped_view)

        return decorator

    def compute_risk(measurement):
        risk = "normal"
        reasons = []
        heart_rate =
measurement.get("heart_rate")
        spo2 = measurement.get("spo2")
        temperature =
measurement.get("temperature")

        if heart_rate is not None and
(heart_rate < 50 or heart_rate > 110):
            risk = "warning"
            reasons.append("Небезпечний пульс")
        if spo2 is not None and spo2 < 92:
            risk = "warning"
```

```
            reasons.append("Низький рівень
SpO₂")
        if temperature is not None and
temperature >= 38:
            risk = "critical"
            reasons.append("Висока
температура")
        if heart_rate is not None and
heart_rate >= 130:
            risk = "critical"
            reasons.append("Критично високий
пульс")
        if spo2 is not None and spo2 < 88:
            risk = "critical"
            reasons.append("Критично низький
SpO₂")

        return risk, reasons

    def
latest_measurement_for_patient(patient_id):
        return db.measurements.find_one(
            {"patient_id":
ObjectId(patient_id)}, sort=[("timestamp", -1)]
        )

    def measurement_summary(measurements):
        summary = {
            "heart_rate": None,
            "spo2": None,
            "temperature": None,
            "activity": None,
            "timestamp": None,
            "risk_level": "normal",
            "reasons": [],
        }
        if not measurements:
            return summary
        latest = measurements[0]
        summary.update(
            {
                "heart_rate":
latest.get("heart_rate"),
                "spo2": latest.get("spo2"),
                "temperature":
latest.get("temperature"),
                "activity":
latest.get("activity"),
                "timestamp":
latest.get("timestamp"),
                "risk_level":
latest.get("risk_level", "normal"),
                "reasons":
latest.get("reasons", []),
            }
        )
        return summary

    @app.route("/")
    def index():
        user_id = session.get("user_id")
        if not user_id:
            return redirect(url_for("login"))
        user = get_user(user_id)
        if not user:
            session.clear()
            return redirect(url_for("login"))
        role = user.get("role")
        if role == "patient":
            return
redirect(url_for("patient_dashboard"))
        if role == "doctor":
            return
redirect(url_for("doctor_dashboard"))
        if role == "admin":
            return
redirect(url_for("admin_dashboard"))
        session.clear()
        flash("Невідома роль користувача.",
"danger")
        return redirect(url_for("login"))

    @app.route("/login", methods=["GET",
"POST"])
    def login():
        if request.method == "POST":
            login = request.form.get("login",
"").lower()
            password =
request.form.get("password", "")

            user = get_user_by_login(login)
            if user and
check_password_hash(user["password_hash"],
password):
                session["user_id"] =
str(user["_id"])
                flash("Вітаємо у MedPredict!",
"success")
                return
redirect(url_for("index"))
            flash("Невірний логін або пароль.",
"danger")
        return render_template("login.html")

    @app.route("/register", methods=["GET",
"POST"])
    def register():
        if request.method == "POST":
            full_name =
request.form.get("full_name", "").strip()
            birth_date =
request.form.get("birth_date", "")
            login = request.form.get("login",
"").lower()
            password =
request.form.get("password", "")

            if not full_name:
                flash("ПІБ є обов'язковим
полем.", "warning")
                return
render_template("register.html")

            if not birth_date:
                flash("Дата народження є
обов'язковим полем.", "warning")
                return
render_template("register.html")

            if not login:
                flash("Логін є обов'язковим
полем.", "warning")
                return
render_template("register.html")

            if get_user_by_login(login):
```

```
                flash("Користувач з таким
логіном вже існує.", "warning")
                return
render_template("register.html")

            if
get_user_by_credentials(full_name, birth_date):
                flash("Пацієнт з таким ПІБ та
датою народження вже існує.", "warning")
                return
render_template("register.html")

            password_hash =
generate_password_hash(password)
            result = db.users.insert_one(
                {
                    "full_name": full_name,
                    "birth_date": birth_date,
                    "login": login,
                    "password_hash":
password_hash,
                    "role": "patient",
                }
            )
            session["user_id"] =
str(result.inserted_id)
            flash("Реєстрація успішна!",
"success")
            return
redirect(url_for("patient_dashboard"))
        return render_template("register.html")

    @app.route("/logout")
    def logout():
        session.clear()
        flash("Ви вийшли з системи.", "info")
        return redirect(url_for("login"))

    @app.route("/patient/dashboard")
    @role_required("patient")
    def patient_dashboard(user):
        measurements = list(
            db.measurements.find({"patient_id":
user["_id"]}).sort("timestamp", -1)
        )
        summary =
measurement_summary(measurements)
        alerts = db.alerts.find({"patient_id":
user["_id"]}).sort("timestamp", -1)
        recommendations =
db.recommendations.find({"patient_id":
user["_id"]}).sort(
            "timestamp", -1
        )
        return render_template(
            "patient_dashboard.html",
            user=user,
            summary=summary,
            measurements=measurements[:10],
            alerts=list(alerts),
            recommendations=list(recommendation
s),
        )

    @app.route("/doctor/dashboard")
    @role_required("doctor")
    def doctor_dashboard(user):
        patients = list(db.users.find({"role":
"patient"}))
        patient_cards = []
        for patient in patients:
            # Перевірка наявності обов'язкових
полів
            if not patient.get("full_name"):
                continue  # Пропускаємо
пацієнтів без ПІБ

            latest =
latest_measurement_for_patient(patient["_id"])
            reasons = []
            risk_class = "normal"
            risk_label = "Немає даних"
            if latest:
                risk_class =
latest.get("risk_level", "normal")
                risk_label =
risk_class.capitalize()
                reasons = latest.get("reasons",
[])
            patient_cards.append(
                {
                    "id": str(patient["_id"]),
                    "full_name":
patient["full_name"],
                    "latest": latest,
                    "risk_class": risk_class,
                    "risk_label": risk_label,
                    "reasons": reasons,
                }
            )
        return render_template(
            "doctor_dashboard.html", user=user,
patients=patient_cards
        )

    @app.route("/doctor/patient/<patient_id>")
    @role_required("doctor")
    def doctor_patient_detail(user,
patient_id):
        patient = get_user(patient_id)
        if not patient or patient.get("role")
!= "patient":
            flash("Пацієнта не знайдено.",
"danger")
            return
redirect(url_for("doctor_dashboard"))
        measurements = list(
            db.measurements.find({"patient_id":
ObjectId(patient_id)}).sort(
                "timestamp", -1
            )
        )
        alerts = list(
            db.alerts.find({"patient_id":
ObjectId(patient_id)}).sort(
                "timestamp", -1
            )
        )
        recommendations = list(
            db.recommendations.find({"patient_i
d": ObjectId(patient_id)}).sort(
                "timestamp", -1
            )
        )
        return render_template(
            "doctor_patient_detail.html",
            user=user,
            patient=patient,
```

```
            patient_id=str(patient["_id"]),
            measurements=measurements,
            alerts=alerts,
            recommendations=recommendations,
        )

    @app.route("/doctor/patient/<patient_id>/recommend", methods=["POST"])
    @role_required("doctor")
    def doctor_add_recommendation(user, patient_id):
        patient = get_user(patient_id)
        if not patient or patient.get("role") != "patient":
            flash("Пацієнта не знайдено.", "danger")
            return redirect(url_for("doctor_dashboard"))
        message = request.form.get("message", "").strip()
        if not message:
            flash("Введіть текст рекомендації.", "warning")
        else:
            db.recommendations.insert_one(
                {
                    "patient_id": ObjectId(patient_id),
                    "doctor_id": user["_id"],
                    "message": message,
                    "timestamp": datetime.utcnow(),
                }
            )
            flash("Рекомендація збережена.", "success")
        return redirect(url_for("doctor_patient_detail", patient_id=patient_id))

    @app.route("/admin/dashboard")
    @role_required("admin")
    def admin_dashboard(user):
        doctors = []
        for doctor in db.users.find({"role": "doctor"}):
            doctors.append(
                {"id": str(doctor["_id"]), "full_name": doctor["full_name"], "login": doctor["login"]}
            )
        patients = []
        for patient in db.users.find({"role": "patient"}):
            patients.append(
                {"id": str(patient["_id"]), "full_name": patient["full_name"], "login": patient["login"]}
            )
        devices = []
        for device in db.devices.find():
            patient_name = None
            patient_id = device.get("patient_id")
            if patient_id:
                patient_doc = db.users.find_one({"_id": patient_id})
                if patient_doc:
                    patient_name = patient_doc.get("full_name")
            devices.append(
                {"device_id": device.get("device_id"), "updated_at": device.get("updated_at"), "patient_name": patient_name}
            )
        return render_template(
            "admin_dashboard.html",
            user=user,
            doctors=doctors,
            patients=patients,
            devices=devices,
        )

    @app.route("/admin/create_user", methods=["POST"])
    @role_required("admin")
    def admin_create_user(user):
        full_name = request.form.get("full_name", "").strip()
        birth_date = request.form.get("birth_date", "")
        login = request.form.get("login", "").lower()
        password = request.form.get("password", "")
        role = request.form.get("role")

        if not full_name:
            flash("ПІБ є обов'язковим полем.", "warning")
            return redirect(url_for("admin_dashboard"))

        if not login:
            flash("Логін є обов'язковим полем.", "warning")
            return redirect(url_for("admin_dashboard"))

        if role == "patient" and not birth_date:
            flash("Дата народження є обов'язковою для пацієнта.", "warning")
            return redirect(url_for("admin_dashboard"))

        if role not in {"doctor", "patient"}:
            flash("Некоректна роль користувача.", "danger")
            return redirect(url_for("admin_dashboard"))

        if get_user_by_login(login):
            flash("Користувач з таким логіном вже існує.", "warning")
            return redirect(url_for("admin_dashboard"))

        if role == "patient" and get_user_by_credentials(full_name, birth_date):
            flash("Пацієнт з таким ПІБ та датою народження вже існує.", "warning")
            return redirect(url_for("admin_dashboard"))

        user_data = {
```

```
            "full_name": full_name,
            "login": login,
            "password_hash":
generate_password_hash(password),
            "role": role,
        }

        if role == "patient":
            user_data["birth_date"] =
birth_date

        db.users.insert_one(user_data)
        flash("Користувача створено.",
"success")
        return
redirect(url_for("admin_dashboard"))

    @app.route("/admin/assign_device",
methods=["POST"])
    @role_required("admin")
    def admin_assign_device(user):
        device_id =
request.form.get("device_id", "").strip()
        patient_id =
request.form.get("patient_id")
        if not device_id or not patient_id:
            flash("Заповніть всі поля.",
"warning")
            return
redirect(url_for("admin_dashboard"))
        patient = get_user(patient_id)
        if not patient or patient.get("role")
!= "patient":
            flash("Пацієнта не знайдено.",
"danger")
            return
redirect(url_for("admin_dashboard"))
        db.devices.update_one(
            {"device_id": device_id},
            {
                "$set": {
                    "device_id": device_id,
                    "patient_id":
patient["_id"],
                    "updated_at":
datetime.utcnow(),
                }
            },
            upsert=True,
        )
        flash("Пристрій прив'язано до
пацієнта.", "success")
        return
redirect(url_for("admin_dashboard"))

    def parse_float(value):
        try:
            return float(value)
        except (TypeError, ValueError):
            return None

    def parse_timestamp(value):
        if not value:
            return datetime.utcnow()
        for fmt in ("%Y-%m-%dT%H:%M:%S", "%Y-
%m-%d %H:%M:%S", "%d.%m.%Y %H:%M"):
            try:
                return datetime.strptime(value,
fmt)
            except ValueError:
                continue
        try:
            return
datetime.fromisoformat(value)
        except ValueError:
            return datetime.utcnow()

    @app.route("/admin/upload", methods=["GET",
"POST"])
    @role_required("admin")
    def admin_upload(user):
        if request.method == "POST":
            file = request.files.get("file")
            if not file or not
file.filename.endswith(".csv"):
                flash("Оберіть CSV-файл.",
"warning")
                return
redirect(url_for("admin_upload"))
            stream =
io.StringIO(file.stream.read().decode("utf-8"))
            reader = csv.DictReader(stream)
            inserted = 0
            for row in reader:
                device_id =
row.get("device_id") or row.get("device")
                device =
db.devices.find_one({"device_id": device_id})
                if not device:
                    continue
                patient_id =
device.get("patient_id")
                if not patient_id:
                    continue
                measurement = {
                    "patient_id": patient_id,
                    "device_id": device_id,
                    "timestamp":
parse_timestamp(row.get("timestamp")),
                    "heart_rate":
parse_float(row.get("heart_rate")),
                    "spo2":
parse_float(row.get("spo2")),
                    "temperature":
parse_float(row.get("temperature")),
                    "activity":
parse_float(row.get("activity")),
                }
                risk_level, reasons =
compute_risk(measurement)
                measurement["risk_level"] =
risk_level
                measurement["reasons"] =
reasons
                db.measurements.insert_one(meas
urement)
                if risk_level in {"warning",
"critical"}:
                    db.alerts.insert_one(
                        {
                            "patient_id":
patient_id,
                            "risk_level":
risk_level,
                            "reasons": reasons,
                            "timestamp":
measurement["timestamp"],
```

```
                            "device_id":
device_id,
                        }
                    )
                inserted += 1
            flash(f"Імпортовано записів:
{inserted}.", "success")
            return
redirect(url_for("admin_dashboard"))
        return
render_template("admin_upload.html", user=user)

    return app


app = create_app()


if __name__ == "__main__":
    app.run(host="0.0.0.0",
port=int(os.getenv("PORT", 5000)), debug=True)
```

admin_dashboard.html

```
{% extends 'base.html' %}
{% block content %}
<h1 class="mb-4">Адмін-панель MedPredict</h1>
<div class="row g-4">
    <div class="col-md-6">
        <div class="card shadow-sm">
            <div class="card-header">Створення
користувачів</div>
            <div class="card-body">
                <form method="post" action="{{
url_for('admin_create_user') }}">
                    <div class="mb-3">
                        <label class="form-
label" for="full_name">ПІБ</label>
                        <input class="form-
control" type="text" name="full_name"
id="full_name" required>
                    </div>
                    <div class="mb-3">
                        <label class="form-
label" for="login">Логін</label>
                        <input class="form-
control" type="text" name="login" id="login"
required>
                    </div>
                    <div class="mb-3">
                        <label class="form-
label" for="birth_date">Дата народження</label>
                        <input class="form-
control" type="date" name="birth_date"
id="birth_date">
                        <small class="form-text
text-muted">Обов'язково для пацієнтів</small>
                    </div>
                    <div class="mb-3">
                        <label class="form-
label" for="password">Пароль</label>
                        <input class="form-
control" type="password" name="password"
id="password" minlength="6" required>
                    </div>
                    <div class="mb-3">
                        <label class="form-
label">Роль</label>
                        <select class="form-
select" name="role" required>
                            <option
value="doctor">Лікар</option>
                            <option
value="patient">Пацієнт</option>
                        </select>
                    </div>
                    <button class="btn btn-
primary" type="submit">Створити
користувача</button>
                </form>
            </div>
        </div>
    </div>
    <div class="col-md-6">
        <div class="card shadow-sm h-100">
            <div class="card-header">Прив'язка
пристроїв</div>
            <div class="card-body">
                <form method="post" action="{{
url_for('admin_assign_device') }}">
                    <div class="mb-3">
                        <label class="form-
label" for="device_id">ID пристрою</label>
                        <input class="form-
control" type="text" name="device_id"
id="device_id" placeholder="device01" required>
                    </div>
                    <div class="mb-3">
                        <label class="form-
label" for="patient_id">Пацієнт</label>
                        <select class="form-
select" name="patient_id" id="patient_id"
required>
                            <option
value="">Оберіть пацієнта</option>
                            {% for patient in
patients %}
                                <option
value="{{ patient.id }}">{{ patient.full_name
}}</option>
                            {% endfor %}
                        </select>
                    </div>
                    <button class="btn btn-
success" type="submit">Прив'язати</button>
                </form>
                <hr>
                <h5>Активні пристрої</h5>
                {% if devices %}
                    <ul class="list-group list-
group-flush">
                        {% for device in
devices %}
                            <li class="list-
group-item d-flex justify-content-between
align-items-center">
                                <span>{{
device.device_id }}</span><small class="ms-
2">{{ device.patient_name or 'без пацієнта'
}}</small>
                                <small
class="text-muted">{{
device.updated_at.strftime('%d.%m.%Y %H:%M') if
device.updated_at else '' }}</small>
                            </li>
```

```
                    {% endfor %}
                </ul>
            {% else %}
                <p class="mb-0">Пристроїв ще не прив'язано.</p>
            {% endif %}
        </div>
    </div>
</div>

{% include 'admin_users_list.html' %}

<div class="card shadow-sm mt-4">
    <div class="card-header">Швидка навігація</div>
    <div class="card-body">
        <a class="btn btn-outline-secondary" href="{{ url_for('admin_upload') }}">Імпорт даних з CSV</a>
    </div>
</div>
{% endblock %}
```

base.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>{{ title or 'MedPredict' }}</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-primary mb-4">
    <div class="container-fluid">
        <a class="navbar-brand" href="{{ url_for('index') }}">MedPredict</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ms-auto">
                {% if session.get('user_id') %}
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('logout') }}">Вийти</a>
                    </li>
                {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('login') }}">Увійти</a>
                    </li>
                {% endif %}
            </ul>
        </div>
    </div>
</nav>
<main class="container">
    {% with messages = get_flashed_messages(with_categories=true) %}
        {% if messages %}
            <div class="mt-2">
                {% for category, message in messages %}
                    <div class="alert alert-{{ category }}">{{ message }}</div>
                {% endfor %}
            </div>
        {% endif %}
    {% endwith %}
    {% block content %}{% endblock %}
</main>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```