МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра автоматизації та комп'ютерно-інтегрованих технологій

РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНА ЗАПИСКА ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

на тему: Автоматизація моніторингу та діагностики функціонального стану пацієнтів на основі IoT-технологій

Виконав: здобувач вищої освіти
3 курсу, групи Сінж-2023-1-У
напряму підготовки (спеціальності)
174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Шалагін Дмитро Максимович
(прізвище та ініціали)
Керівник Піддубна Л.В., доц. каф. АКІТ
(прізвище та ініціали, наук. ступ., вч. звання)

Рецензент Ківіренко О.Б., начальник виробництва ТОВ «Альфа-Композіт»
(прізвище та ініціали, наук. ступ., вч. звання)

Харків – 2026

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА

Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури

Кафедра автоматизації та комп'ютерно-інтегрованих технологій
Освітньо-кваліфікаційний рівень – бакалавр
Галузь знань 17 «Електроніка, автоматизація та електронні комунікації»
Спеціальність 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

ЗАТВЕРДЖУЮ
Завідувач кафедри АКІТ
_____ БАРАНОВ О.О.
« 19 » червня 2026 року

З А В Д А Н Н Я

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Шалагіна Дмитра Максимовича

1.Тема роботи Автоматизація моніторингу та діагностики функціонального стану пацієнтів на основі IoT-технологій
Затверджена наказом університету від « 22 » травня 2026 року № 440-03.
Керівник роботи Піддубна Л.В., к. філос.н., доц., доцент кафедри АКІТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
2. Строк подання роботи здобувачем вищої освіти «__15__» _червня_ 2026 р.
3. Вихідні дані до роботи Розробка системи моніторингу та діагностики функціонального стану пацієнтів на основі IoT-технологій
4. Зміст розрахунково пояснювальної записки (перелік питань, які потрібно розробити): Вступ. Аналіз об'єкта автоматизації та обгрунтування технічних рішень. Проєктування та моделювання системи телемедичного моніторингу. Технічна реалізація та апробація телемедичної системи. Охорона праці, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація.

6. Консультанти розділів проєкту (роботи)

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Аналіз проблеми	Піддубна Л.В.	11.05.2026	21.05.2026
Основна частина	Піддубна Л.В.	22.05.2026	31.05.2026
Спеціальний розділ	Піддубна Л.В.	01.06.2026	12.06.2026
Охорона праці	Малишева В.В.	05.06.2026	12.06.2026

7. Дата видачі завдання «_11_» __травня_2026_р.

Керівник _____ Піддубна Л.В.
(підпис)

Завдання прийняв до виконання _____ Шалагін Д.М.
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання розділів	Примітка
1	Розробка 1го розділу бакалаврської роботи	11.05.2026 - 21.05.2026	
2	Розробка 2го розділу бакалаврської роботи	22.05.2026 - 31.05.2026	
3	Розробка 3го розділу бакалаврської роботи	01.06.2026 - 07.06.2026	
4	Розробка розділу з охорони праці	08.06.2026 - 14.06.2026	
5	Рецензування бакалаврської роботи	15.06.2026	Ківіренко О.Б.
6	Захист на ДЕК	26.06.2026	

Здобувача вищої освіти _____ Шалагін Д.М.
(підпис)

Керівник _____ Піддубна Л. В.
(підпис)

РЕФЕРАТ

Автоматизація моніторингу та діагностики функціонального стану пацієнтів на основі IoT-технологій – Шалагін Дмитро Максимович, дипломна робота бакалавра, Харків, Харківський національний університет міського господарства імені О.М. Бекетова, кількість сторінок 105, кількість таблиць 28, кількість рисунків 4, кількість джерел літератури 24.

Актуальність теми бакалаврської роботи обумовлена необхідністю дистанційного медичного контролю в умовах навантаження на систему охорони здоров'я, дефіциту лікарів у віддалених та прифронтових районах та потреби в реабілітації пацієнтів поза стаціонаром.

Мета бакалаврської роботи: розробка комп'ютерно-інтегрованої системи для автоматизованого збору та візуалізації медичних показників пацієнта в режимі реального часу.

Об'єкт дослідження: процес дистанційного моніторингу та діагностики основних показників життєдіяльності організму людини (частоти серцевих скорочень, рівня кисню у крові, артеріального тиску, температури).

Предмет дослідження: методи та засоби автоматизації збору, передачі та інтелектуальної обробки біометричних даних за допомогою IoT-технологій.

Реалізація мети бакалаврської роботи потребує вирішення таких завдань:

- Проаналізувати сучасні IoT-рішення для телемедицини.
- Спроєктувати архітектуру системи «сенсор-хмара-інтерфейс».
- Реалізувати алгоритм автоматичного сповіщення про критичні стани.
- Розробити веб-інтерфейс для моніторингу даних лікарем.

Методи дослідження: системний аналіз, методи об'єктно-орієнтованого програмування, метод баз даних та SQL-моделювання, методи тестування програмного забезпечення.

КЛЮЧОВІ СЛОВА: IoT-технології, телемедицина, комп'ютерно-інтегрована система, біометричні дані, система віддаленого моніторингу пацієнтів.

ABSTRACT

Automation of monitoring and diagnostics of patients’functional state based on IoT technologies – Shalagin Dmytro Maksymovych, bachelor's thesis, Kharkiv, Kharkiv National University of Urban Economy named after O.M. Beketov, number of pages 105, number of tables 28, number of figures 4, number of literature sources 24.

The relevance of the topic of the bachelor's thesis is due to the need for remote medical control in conditions of strain on the healthcare system, a shortage of doctors in remote and frontline areas, and the need for outpatient rehabilitation of patients.

The purpose of the bachelor's thesis: development of a computer-integrated system for automated collection and visualization of a patient's medical indicators in real time.

Object of research: the process of remote monitoring and diagnostics of the main vital indicators of the human body (heart rate, blood oxygen level, blood pressure, temperature).

Subject of research: methods and means of automating the collection, transmission and intelligent processing of biometric data using IoT technologies.

The implementation of the goal of the bachelor's thesis requires solving the following tasks:

- Analyze modern IoT solutions for telemedicine.
- Design the architecture of the "sensor-cloud-interface" system.
- Implement an algorithm for automatic notification of critical conditions.
- Develop a web interface for data monitoring by a doctor.

Research methods: systems analysis, object-oriented programming methods, database and SQL modeling methods, software testing methods.KEYWORDS: computer-integrated automated control system, digitalization of production, production subsystems.

KEYWORDS: IoT technologies, biometric data, telemedicine, Remote Patient Monitoring, computer-integrated system.

ЗМІСТ

ВСТУП

Цифрова трансформація медицини (eHealth) вимагає переходу від епізодичного обстеження до безперервного моніторингу стану пацієнта. В умовах поствоєнного відновлення України та високої щільності населення в містах-мільйонниках, таких як Харків, навантаження на амбулаторії зростає. Використання IoT-технологій дозволяє знизити ризик раптових ускладнень (інфаркти, інсульти) за рахунок ранньої діагностики аномалій. Створення вітчизняних доступних систем автоматизації моніторингу є критично важливим для сталого розвитку системи охорони здоров'я.

Мета роботи: Підвищення ефективності та оперативності медичного нагляду за пацієнтами шляхом розробки та впровадження автоматизованої IoT-системи, що забезпечує дистанційний моніторинг функціонального стану з мінімальною участю медичного персоналу в процесі збору даних.

Об’єкт дослідження: Процеси оперативного збору, накопичення та аналізу часових рядів біометричних параметрів пацієнта (частота серцевих скорочень, рівень кисню у крові, артеріальний тиск) у телемедичних мережах.

Предмет дослідження: Комп’ютерно-інтегровані засоби автоматизації (мікроконтролери, сенсори, хмарні сервіси), протоколи бездротового зв’язку (MQTT, HTTP) та алгоритми підтримки прийняття рішень для ідентифікації критичних відхилень у функціональному стані людини.

Реалізація мети бакалаврської роботи потребує вирішення таких завдань:

- Виконати порівняльний аналіз технічних характеристик сучасних біометричних датчиків та протоколів передачі даних для медичних систем.
- Обґрунтувати вибір апаратної бази (мікроконтролерів ESP32 / STM32) та хмарної платформи для реалізації системи.
- Спроєктувати інфологічну модель бази даних для довготривалого зберігання медичної історії пацієнта.

- Розробити алгоритмічне забезпечення для автоматичної пріоритезації сповіщень при виявленні небезпечних трендів у показниках пацієнта.
- Програмно реалізувати клієнт-серверну частину системи (FastAPI + React) та протестувати її на реальних сценаріях роботи.

У процесі виконання дипломної роботи було використано такі методи дослідження, як теорія автоматичного керування та системний аналіз – для побудови загальної структури системи; методи цифрової обробки сигналів – для фільтрації шумів та артефактів із сирих даних сенсорів; методи баз даних та SQL-моделювання – для організації структури збереження інформації; методи імітаційного моделювання – для перевірки стійкості системи при високих навантаженнях на канали зв’язку; експериментальні методи – для оцінки похибки вимірювань сенсорів у порівнянні з еталонними медичними приладами.

РОЗДІЛ 1. АНАЛІЗ ОБ'ЄКТА АВТОМАТИЗАЦІЇ ТА ОБҐРУНТУВАННЯ ТЕХНІЧНИХ РІШЕНЬ

1.1 Сучасний стан та перспективи розвитку телемедичних технологій в Україні

Цивілізаційні виклики останніх років, зокрема пандемія COVID-19 та повномасштабна військова агресія, стали потужними каталізаторами цифрової трансформації української медицини. Сьогодні телемедицина в Україні еволюціонує від простого відеозв'язку між лікарем та пацієнтом до комплексних комп'ютерно-інтегрованих систем, що охоплюють діагностику, моніторинг та реабілітацію.

На державному рівні було затверджено стратегію розвитку телемедицини на 2023 – 2025 роки, яка передбачає впровадження цифрових медичних сервісів, розвиток інфраструктури та інтеграцію з європейськими системами охорони здоров'я.

Сучасний стан телемедицини в Україні характеризується впровадженням електронної системи охорони здоров'я (eHealth); розвитком онлайн-консультацій лікарів; використанням мобільних додатків для пацієнтів; інтеграцією цифрових технологій у медичні процеси.

Водночас існують проблеми, пов'язані з нерівномірним доступом до Інтернету; недостатньою технічною оснащеністю медичних закладів; дефіцитом кваліфікованих кадрів.

Системи віддаленого моніторингу пацієнтів (RPM – Remote Patient Monitoring) є ключовим сегментом телемедицини, що базується на використанні цифрових технологій для збору медичних даних в одному місці (наприклад, вдома у пацієнта) та їх електронної передачі медичним працівникам в інше місце для оцінки та надання рекомендацій. Сучасні RPM-рішення, що впроваджуються в Україні, класифікуються за трьома рівнями. Сенсорний рівень представлений носимими пристроями (smart-watches, патчі) та медичними гаджетами з Bluetooth-інтерфейсом (глюкометри, тонометри,

спірометри). Транспортний рівень утворюють мобільні додатки та IoT-шлюзи, що агрегують дані та передають їх у хмарне середовище за протоколами MQTT або HTTPS. Аналітичний рівень представлено серверним програмним забезпеченням, що використовує алгоритми обробки часових рядів для виявлення аномалій, наприклад, тахікардії або критичного зниження сатурації. Перспективи розвитку RPM в Україні пов'язані з інтеграцією цих систем у єдиний державний реєстр eHealth, що дозволить автоматично оновлювати електронну медичну картку пацієнта даними з його персональних пристроїв моніторингу.

Військові дії призвели до різкого зростання кількості пацієнтів, які потребують тривалої, часто пожиттєвої реабілітації. Телемедичні технології передставлені дистанційною фізіотерапією, моніторингом психоемоційного стану, безперервністю лікування, тощо. Дистанційна фізіотерапія передбачає використання датчиків руху та систем комп'ютерного зору для контролю правильності виконання реабілітаційних вправ удома. Моніторинг психоемоційного стану здійснюється шляхом автоматизованого збору даних про якість сну, варіабельність серцевого ритму та рівень активності для раннього виявлення ознак посттравматичного стресового розладу (ПТСР). Для системи автоматизації ПТСР цікавий тим, що він має чіткі біометричні маркери, які IoT-система може фіксувати в реальному часі. До них відносять варіабельність серцевого ритму (ВСР/HRV), якість та структура сну, шкірно-гальванічна реакція (ШГР). Варіабельність серцевого ритму (ВСР/HRV) є головним показником стану вегетативної нервової системи. При ПТСР спостерігається низька варіабельність, що свідчить про постійний стрес і нездатність організму до релаксації. Важливим показником є якість та структуру сну. Датчики (акселерометри) фіксують переривчастий сон, нічні жахи та рухову активність під час сну, що є класичними симптомами розладу. Стан вегетативної нервової системи можна визначити за шкірно-гальванічною реакцією (ШГР), а саме, зміною електропроводності шкіри при виділенні поту (навіть мікроскопічному) під час спалахів тривоги чи флешбеків.

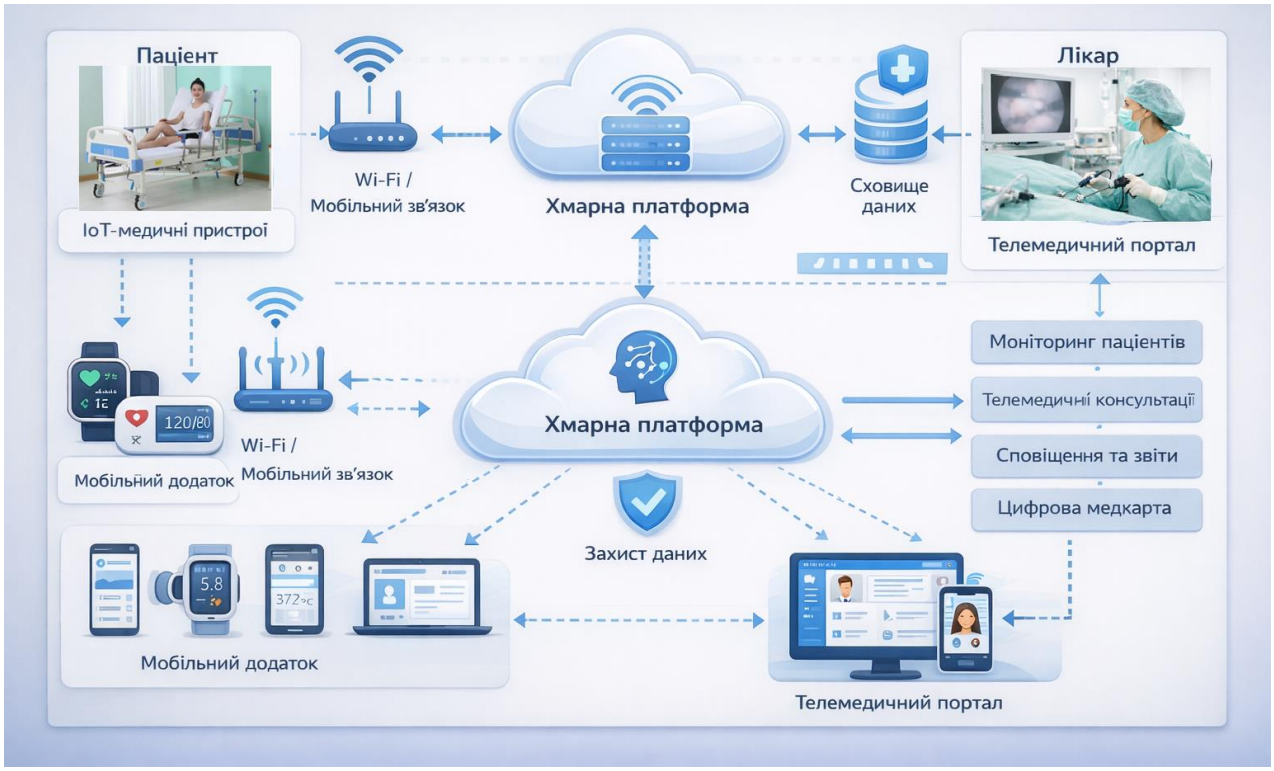


Рисунок 1.1 – Архітектура телемедичної системи

Безперервність лікування реалізується шляхом організації постійного зв'язку з реабілітологом для пацієнтів з ампутаціями або складними мінно-вибуховими травмами, що потребують тривалого корегування терапії. Для деокупованих територій та віддалених сільських громад, де спостерігається критичний дефіцит профільних лікарів та зруйнована медична інфраструктура, телемедицина є практично єдиним способом отримання кваліфікованої допомоги.

Ключовими аспектами впровадження телемедицини у важкодоступних регіонах є мобільні телемедичні комплекси, стійкість зв'язку, асинхронний моніторинг, тощо. Мобільні телемедичні комплекси представлені валізами з набором діагностичного обладнання, які дозволяють сімейному лікарю на місці провести обстеження та передати дані вузькому спеціалісту в обласний центр (наприклад, до Харкова). Стійкість зв'язку реалізується через використання терміналів Starlink для забезпечення стабільного каналу

передачі біометричних даних у зонах із пошкодженими лініями зв'язку. Асинхронний моніторинг здійснюється за допомогою системи «Store-and-forward», де дані збираються локально і відправляються на сервер при появі сигналу мережі, що критично важливо для стабілізації стану пацієнтів у «сірих» зонах.

Основними напрямами перспективного розвитку телемедицини в Україні є інтеграція з IoT (використання носимих пристроїв; автоматичний збір даних), використання штучного інтелекту (аналіз великих даних; прогнозування стану пацієнтів; підтримка лікарських рішень), розвиток хмарних платформ (централізоване зберігання даних; доступ у реальному часі), розширення нормативної бази (стандартизація телемедичних послуг; захист персональних даних), масове впровадження RPM як можливість переходу від реактивної до проактивної медицини.

1.2 Порівняльний аналіз методів зняття та передачі біометричних сигналів

Біометричні сигнали – це фізіологічні параметри організму людини, які використовуються для оцінки стану здоров’я. У телемедичних системах вони відіграють ключову роль, оскільки дозволяють здійснювати віддалений моніторинг пацієнтів. До основних біометричних сигналів належать електрокардіограма (ЕКГ); частота серцевих скорочень; рівень насичення крові киснем (SO_2); температура тіла (таблиця 1.1).

Ефективність телемедичної системи моніторингу критично залежить від точності первинного зняття показників та надійності їх передачі до обчислювального вузла. Розглянемо апаратні рішення для реєстрації фізіологічних параметрів та комунікаційні технології для їх трансляції.

Для автоматизації моніторингу функціонального стану пацієнта, зокрема при детекції симптомів ПТСР або серцево-судинних порушень, ключовими є такі три типи сенсорів, як датчики електрокардіографії для реєстрації електричної активності серця, пульсоксиметричні датчики – пульсоксиметри для реєстрації рівня кисню в крові (SpO_2) та пульсу, датчики температури для вимірювання температури тіла.

Принцип роботи датчиків електрокардіографії засновані на тому, що електроди фіксують електричні імпульси; сигнал обробляється і передається до системи.

У портативних IoT-пристроях часто використовуються одноканальні модулі (наприклад, AD8232). Вони дозволяють виділяти чистий сигнал серцевого ритму (QRS-комплекс) навіть за наявності шумів, спричинених рухом пацієнта. Аналіз варіабельності серцевого ритму (ВСР) є «золотим стандартом» для оцінки стресу та стану вегетативної нервової системи.

Перевагами датчиків ЕКГ є їх висока точність; можливість виявлення серцевих патологій. До недоліків можна віднести складність підключення; чутливість до шумів.

Пульсоксиметри (SpO2 та ЧСС) засновані на фотоплетизмографії (PPG), тобто, вимірюванні поглинання світла різної довжини хвилі (червоного та інфрачервоного) гемоглобіном у капілярах. Для реалізації роботи пульсоксиметрів використовуються інтегровані сенсори типу MAX30102. Вони поєднують світлодіоди та фотоприймач в одному корпусі. Пульсоксиметри дозволяють відстежувати рівень кисню в крові та частоту пульсу, що важливо при моніторингу панічних атак та респіраторних захворювань.

До переваг пульсоксиметрів відносять неінвазивність; простоту використання, до недоліків – залежність від руху; вплив освітлення.

Датчики температури тіла засновані на використанні напівпровідникових терморезисторів (контактні датчики) або інфрачервоних пірометрів (безконтактні датчики). Датчики температури тіла реалізуються на

базі високоточних цифрових сенсорів, наприклад, MLX90614 або MAX30205, які забезпечують похибку не більше ±0.1°C. Датчики температури призначені для виявлення запальних процесів та моніторингу циркадних ритмів, що є важливим при аналізі порушень сну.

Перевагами датчиків температури тіла є їх простота; низька вартість. Недоліками є менша точність у деяких умовах.

Таблиця 1.1 – Порівняння біометричних датчиків

Параметр	ЕКГ	Пульсоксиметр	Температура
Тип сигналу	Електричний	Оптичний	Тепловий
Точність	Висока	Середня	Середня
Інвазивність	Ні	Ні	Ні
Складність	Висока	Низька	Низька
Застосування	Кардіологія	Контроль стану	Загальний моніторинг

Передача даних є важливим етапом у телемедичних системах. Вона забезпечує зв’язок між датчиками та сервером або лікарем. До основних вимог до передачі даних відносять надійність; енергоефективність; безпеку; швидкість передачі. Вибір протоколу передачі даних визначає енергоефективність пристрою (час роботи від батареї), радіус дії та надійність доставки критичних медичних пакетів.

Розглянемо основні протоколи передачі даних (таблиця 1.2).

Протокол Bluetooth Low Energy (BLE) є оптимальним для носимих браслетів. Має найнижче енергоспоживання, але обмежений радіус дії (до 10 – 20 метрів). Потребує смартфона як шлюзу.

Протокол Wi-Fi забезпечує високу швидкість, що важливо для передачі сирих даних ЕКГ у реальному часі, але швидко розряджає акумулятор пристрою.

Протокол LoRaWAN використовується для важкодоступних регіонів та великих відстаней (до 15 км). Має низьку швидкість, що дозволяє передавати лише узагальнені показники, наприклад, пульс раз на 5 хвилин.

Протокол Message Queuing Telemetry Transport (MQTT) – протокол обміну повідомленнями; працює поверх TCP/IP; забезпечує ефективну передачу даних у реальному часі. Він є стандартом для IoT завдяки своїй легкості та роботі в умовах нестабільного зв'язку (Publish/Subscribe модель).

Таблиця 1.2 – Порівняльна характеристика протоколів передачі медичних даних

Характеристика	BLE	Wi-Fi	LoRaWAN	MQTT (над Wi-Fi/GSM)
Енергоспоживання	Дуже низьке	Високе	Мінімальне	Середнє
Дальність дії	10–50 м	30–100 м	2–15 км	Глобальна (через IP)
Пропускна здатність	Низька (до 2 Мбіт/с)	Висока (від 54 Мбіт/с)	Дуже низька (до 50 кбіт/с)	Середня (оптимізована)
Сценарій використання	Фітнес-трекери, патчі	Стаціонарні монітори в лікарні	Віддалені села, зони надзвичайних ситуацій	Передача даних у хмару
Надійність доставки	Середня	Висока	Низька (без підтверджень)	Висока (QoS рівні)

Для телемедичних систем доцільно комбінувати технології, наприклад, BLE використовувати для збору даних з датчиків; Wi-Fi – для передачі в інтернет; MQTT – для обміну між серверами; LoRaWAN – для сільських і віддалених територій. Комбінований підхід забезпечує гнучкість; масштабованість; ефективність, тому ми будемо використовувати його для реалізації бакалаврської роботи. На фізичному рівні зняття даних буде здійснюватися через сенсори AD8232 (ЕКГ) та MAX30105 (SpO_2). На транспортному рівні будемо використовувати мікроконтролер з підтримкою Wi-Fi (ESP32) для прямої передачі даних у локальну мережу. На рівні додатку застосуємо протокол MQTT, оскільки він дозволяє мінімізувати обсяг трафіку та забезпечує гарантовану доставку критичних сповіщень про стан пацієнта навіть при слабкому сигналі інтернету, що актуально для прифронтових територій та Харкова.

1.3 Постановка задачі та формування вимог до комп’ютерно-інтегрованої системи

Метою розробки є створення комп’ютерно-інтегрованої системи для віддаленого моніторингу стану пацієнтів, яка забезпечує збір, передачу, зберігання та аналіз біометричних даних у реальному часі. Виходячи з цього система повинна автоматично зчитувати фізіологічні параметри; передавати дані до хмарного середовища; аналізувати стан пацієнта; формувати сповіщення у разі відхилень. Розробка направлена на підвищення якості медичного контролю при мінімальному навантаженні на медичний персонал. Визначення переліку параметрів моніторингу здійснюється з урахуванням поширених захворювань; можливостей сенсорів; важливості показників для діагностики.

На рисунку 1.2 зображено архітектуру комп’ютерно-інтегрованої системи для віддаленого моніторингу стану пацієнтів, яку ми розробляємо у дипломній бакалаврській роботі.

Рисунок 1.2 – Архітектура IoT-комплексу, призначеного для автоматизованого збору, обробки та передачі даних у реальному часі

Для забезпечення комплексного аналізу функціонального стану пацієнта (зокрема при ПТСР та серцево-судинних патологіях) обрано мінімально достатній набір параметрів моніторингу (таблиця 1.3), які забезпечують комплексний контроль стану пацієнта; раннє виявлення патологій; можливість дистанційної діагностики.

Частота серцевих скорочень (ЧСС) ЧСС є базовим показником фізичного навантаження. Варіабельність серцевого ритму (ВСР) (інтервали між R-зубцями ЕКГ) є об'єктивним індикатором активності симпатичної нервової системи. Низька варіабельність при нормальному пульсі сигналізує про стан хронічного стресу або наближення панічної атаки.

Таблиця 1.3 – Основні параметри моніторингу

№	Параметр	Опис	Значення
1	Частота серцевих скорочень (HR)	Робота серця	60–100 уд/хв
2	ЕКГ	Електрична активність серця	Аналіз ритму
3	SpO_2	Насичення крові киснем	95–100%
4	Температура тіла	Стан організму	36–37°C
5	Артеріальний тиск	Робота судин	120/80 мм рт. ст.

Рівень сатурації кисню в крові (SpO_2) демонструє насичення крові киснем. Критичне зниження сатурації (менше 94 %) може свідчити про респіраторну недостатність або бути соматичним проявом тривожного розладу (гіпервентиляція легень).

Моніторинг температурних трендів дозволяє відрізнити психоемоційні сплески від запальних процесів або інфекційних захворювань.

Рівень рухової активності (акселерометрія) є необхідним показником для верифікації даних. Зростання пульсу на фоні активного руху є нормою, тоді як тахікардія у стані спокою потребує негайного реагування системи.

Комп’ютерно-інтегрована система для віддаленого моніторингу стану пацієнтів повинна збирати дані з датчиків; передавати їх у хмару; зберігати історію вимірювань; аналізувати показники; формувати попередження. До системи моніторингу висуваються функціональні (таблиця 1.4) та нефункціональні вимоги.

Таблиця 1.4 – Функціональні вимоги до системи моніторингу

№	Вимога	Опис
1	Збір даних	Зчитування з сенсорів

№	Вимога	Опис
2	Передача	Безперервна передача
3	Зберігання	Хмарна база
4	Аналіз	Виявлення відхилень
5	Сповіщення	Повідомлення лікаря

Нефункціональні вимоги до системи для віддаленого моніторингу стану пацієнтів стосуються її надійності; енергоефективності; безпеки даних; масштабованості; доступності 24/7.

У якості сформованих технічних вимог до системи можна визначити вимоги до частоти дискретизації ЕКГ, часу автономної роботи, затримки передачі критичного сигналу, типу доступу (таблиця 1.5)

Таблиця 1.5 – Технічні вимоги до системи віддаленого моніторингу стану пацієнтів

№	Вимога	Значення
1	Частота дискретизації ЕКГ	не менше 250 Гц для коректного обчислення ВСР
2	Час автономної роботи	не менше 24 годин у режимі активного моніторингу
3	Затримка передачі критичного сигналу (Latency)	не більше 2 сек
4	Тип доступу	кросплатформний веб-інтерфейс (React) із рольовою моделлю доступу (Пацієнт / Лікар / Адміністратор)

Висновок до розділу 1.

У розділі 1 проведено аналіз стану телемедицини в Україні. Визначено, що телемедицина в Україні перебуває на етапі активного розвитку та є важливим інструментом підвищення доступності медичних послуг. Особливу роль відіграють системи віддаленого моніторингу пацієнтів, які забезпечують безперервний контроль стану здоров’я, знижують навантаження на медичну систему та підвищують ефективність лікування.

В умовах дефіциту медичних кадрів, складної безпекової ситуації та зростання потреб у реабілітації, впровадження RPM-технологій є перспективним напрямом, що сприятиме модернізації системи охорони здоров’я та покращенню якості життя населення.

У розділі 1 також сформульовано задачу створення комп’ютерно-інтегрованої системи моніторингу пацієнтів, визначено перелік основних параметрів, які забезпечують ефективний контроль стану здоров’я. Запропоноване рішення дає можливість забезпечити надійність, масштабованість і ефективність системи, що дозволить використовувати її у сучасних умовах розвитку телемедицини.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ ТЕЛЕМЕДИЧНОГО МОНІТОРИНГУ

2.1 Розробка архітектури IoT-комплексу та схеми автоматизації збору даних

Для створення ефективної, масштабованої та надійної системи, що забезпечує безперервний моніторинг технологічних параметрів, які забезпечують ефективний контроль стану здоров’я пацієнту, потрібно розробити архітектури IoT-комплексу.

Структурна схема комплексу має вигляд: «Датчик → Мікроконтролер → Хмарний сервер → Клієнтський додаток».

Архітектура IoT-системи базується на чотирирівневій моделі, яка включає рівень датчиків (сенсорів), рівень мікроконтролера, хмарний сервер (Cloud-рівень), клієнтський додаток.

На рівні датчиків (сенсорів) здійснюється первинний збір даних з фізичного середовища. Використовуються різні типи датчиків залежно від задачі, такі, як датчики вологості; тиску; вібрації; струму та напруги; температурні датчики. Датчики формують аналогові або цифрові сигнали, що відображають поточний стан об’єкта контролю.

Мікроконтролер виконує функції збору, попередньої обробки та передачі даних. Основними задачами на рівні мікроконтролера є опитування датчиків; фільтрація та нормалізація сигналів; перетворення аналогових сигналів у цифрові; формування пакетів даних; передача інформації через мережеві протоколи (Wi-Fi, Ethernet, LoRa, GSM). Також на цьому рівні можуть реалізовуватися алгоритми локальної логіки (edge computing), наприклад, виявлення аварійних станів; пороговий контроль параметрів; буферизація даних при втраті зв’язку.

Хмарна інфраструктура забезпечує централізовану обробку, зберігання та аналіз даних. Основними функціями хмарного серверу (Cloud-рівня) є прийом даних від пристроїв через API або брокери повідомлень, наприклад,

MQTT; збереження даних у базі; обробка потоків у реальному часі; аналітика та агрегація; формування звітів. Перевагами використання хмарного середовища є масштабованість; доступність; відмовостійкість; можливість інтеграції з іншими сервісами.

Кінцевий користувач взаємодіє із системою через клієнтський інтерфейс (веб або мобільний додаток). Основними можливостями клієнтського додатку є візуалізація даних у вигляді графіків і таблиць; моніторинг у реальному часі; отримання сповіщень про аварії; керування обладнанням (за необхідності).

Запропонована структура забезпечує повний цикл обробки інформації – від фізичного вимірювання до прийняття управлінських рішень.

Для реалізації носимого пристрою (шлюзу) обрано мікроконтролер ESP32 (WROOM-32). Вибір обґрунтовано такими факторами, як інтегровані інтерфейси зв'язку, обчислювальна потужність, енергоефективність, периферія. Обгрунтуємо більш детально.

Інтегровані інтерфейси зв'язку. Наявність вбудованих модулів Wi-Fi та Bluetooth (Dual Mode) дозволяє передавати дані безпосередньо в хмару або через смартфон пацієнта.

Обчислювальна потужність. Двоядерний процесор із частотою до 240 МГц дозволяє виконувати цифрову фільтрацію сигналів ЕКГ безпосередньо «на борту» (Edge Computing), що розвантажує сервер.

Енергоефективністю. Підтримка режимів глибокого сну (Deep Sleep) із споживанням до 10 мкА, що критично для автономних пристроїв із живленням від акумулятора.

Периферія. Велика кількість каналів 12-бітного АЦП (ADC) та підтримка I2C/SPI для підключення медичних сенсорів.

Мікроконтролер ESP32 підтримує Wi-Fi і BLE; має низьке енергоспоживання; забезпечує достатню продуктивність; широко використовується в IoT, і , відповідно до цього, є оптимальним варіантом для телемедичних систем.

Для зберігання та обробки медичної інформації обрано концепцію Hybrid Cloud із використанням таких технологій, як протокол MQTT, серверна логіка за допомогою Python, шифрування даних за протоколом TLS/SS, база даних «PostgreSQL + TimescaleDB».

Протокол передачі Message Queuing Telemetry Transport (MQTT) є оптимальним протоколом для IoT завдяки мінімальному об'єму службових заголовків, що дозволяє системі працювати навіть при низькій швидкості мобільного інтернету і є актуальним для важкодоступних регіонів.

Серверна логіка реалізована FastAPI (Python), забезпечує високу швидкість обробки асинхронних запитів та автоматичну генерацію документації API, що спрощує подальшу інтеграцію з державними системами eHealth.

З метою виконання вимог безпеки здійснюється шифрування даних за протоколом TLS/SSL. Зберігання медичних даних відбувається на серверах із дотриманням вимог GDPR та Закону України «Про захист персональних даних».

Базу даних реалізовано «PostgreSQL + TimescaleDB». Медичні дані – це часові ряди (time-series).

Таблиця 2.1 – Порівняння можливих варіантів бази даних

Тип БД	Переваги	Недоліки
Реляційні (MySQL, PostgreSQL)	простота, універсальність	низька ефективність для великих потоків
NoSQL (MongoDB)	гнучкість структури	не оптимізована для часових рядів
Time-series (InfluxDB, TimescaleDB)	висока продуктивність, спеціалізовані функції	потребує налаштування

Розширення TimescaleDB дозволяє ефективно стискати старі дані та виконувати складні аналітичні запити, наприклад, середній пульс за тиждень, у сотні разів швидше за звичайні реляційні бази. Розглянемо як відбувається вибір архітектури бази даних для часових рядів (Time-series data).

Дані, що генеруються IoT-системою, мають часову природу, оскільки кожне значення прив’язане до конкретного моменту часу. Тому для їх ефективного зберігання доцільно використовувати бази даних часових рядів (Time-Series Database, TSDB). Основними вимогами до бази даних є висока швидкість запису; ефективне зберігання великих обсягів даних; підтримка агрегацій (середнє, максимум, мінімум); можливість роботи з часовими інтервалами; оптимізація для потокових даних.

Для даної системи доцільно використовувати Time-series базу даних, оскільки вона оптимізована для роботи з потоковими даними; забезпечує швидкий запис великих обсягів інформації; дозволяє легко виконувати аналіз часових залежностей; підтримує політики збереження даних (retention policies). Наприклад, використання TimescaleDB (розширення PostgreSQL) дозволяє поєднати переваги реляційної моделі та спеціалізованих механізмів обробки часових рядів.

Хмарна платформа забезпечує централізоване зберігання даних; доступ у реальному часі; масштабованість системи.

Для практичної реалізації запропонованої архітектури доцільно використати сучасні доступні компоненти, що забезпечують надійність, масштабованість та простоту інтеграції.

Таблиця 2.2 – Основні компоненти IoT-системи

Рівень системи	Компонент	Приклад	Основні характеристики	Призначення
Датчики	Датчик температури та вологості	DHT22	Температура: -40…+80°C, вологість: 0 – 100 %	Контроль мікроклімату

Рівень системи	Компонент	Приклад	Основні характеристики	Призначення
Датчики	Датчик температури	DS18B20	Цифровий, точність ±0.5°C	Вимірювання температури
Датчики	Датчик тиску	BMP280	Вимірювання атмосферного тиску	Моніторинг середовища
Датчики	Датчик струму	ACS712	Вимірювання змінного/постійного струму	Контроль електроспоживання
Мікроконтролер	Контролер	ESP32	Wi-Fi, Bluetooth, 240 МГц, ADC	Обробка та передача даних
Мікроконтролер	Альтернатива	Arduino Uno	Простота використання	Базові IoT-рішення
Комунікація	Протокол передачі	MQTT	Легкий, publish/subscribe	Передача даних у реальному часі
Комунікація	Протокол передачі	HTTP/HTTPS	Універсальний	Взаємодія з API
Комунікація	Бездротовий зв’язок	Wi-Fi	Висока швидкість передачі	Локальні мережі
Комунікація	Бездротовий зв’язок	LoRa	Велика дальність, низька швидкість	Віддалені об’єкти
Сервер	Хмарна платформа	AWS IoT / Azure IoT	Масштабованість, аналітика	Обробка та зберігання
Сервер	База даних	InfluxDB	Оптимізація під time-series	Зберігання даних
Сервер	База даних	TimescaleDB	SQL + time-series	Аналітика даних
Клієнт	Веб-додаток	React / Angular	Інтерактивний інтерфейс	Візуалізація даних
Клієнт	Мобільний додаток	Flutter	Кросплатформеність	Доступ користувача

Використання мікроконтролера ESP32 є обґрунтованим завдяки наявності вбудованих модулів Wi-Fi та Bluetooth, що значно спрощує реалізацію бездротової передачі даних без додаткового обладнання. Крім того, він має достатню обчислювальну потужність для реалізації алгоритмів попередньої обробки даних.

Серед датчиків обрано популярні та перевірені рішення (DHT22, DS18B20, BMP280), які забезпечують достатню точність і легко інтегруються з мікроконтролерами.

Для передачі даних використовувати протокол MQTT, оскільки він є легким, ефективним і широко застосовується в IoT-системах. У випадках, коли потрібна сумісність із веб-сервісами, доцільно використовувати HTTP/HTTPS.

Як базу даних обрано InfluxDB або TimescaleDB, оскільки вони спеціалізуються на обробці часових рядів і дозволяють ефективно працювати з великими обсягами телеметричних даних.

Клієнтська частина може бути реалізована як у вигляді веб-додатку, так і мобільного застосунку, що забезпечує гнучкість доступу до системи.

2.2 Розробка алгоритмів цифрової фільтрації та попередньої обробки біосигналів

У сучасних IoT-системах медичного та біометричного призначення важливим етапом є попередня обробка біосигналів. Біосигнали (пульс, ЕКГ, температура, рівень кисню в крові тощо) характеризуються високою зашумленістю та нестабільністю, що обумовлено впливом зовнішніх факторів, руховими артефактами та особливостями вимірювальних пристроїв. Тому для забезпечення достовірності даних необхідно застосовувати алгоритми цифрової фільтрації та інтелектуальної обробки. Попередня обробка сигналів включає такі етапи, як усунення шумів і перешкод; згладжування сигналу; нормалізація значень; виявлення аномалій; підготовка даних для подальшого аналізу.

Джерелами шумів у біосигналах можуть бути електромагнітні перешкоди; рухи користувача; нестабільність контактів сенсора; внутрішні шуми електроніки.

Якість діагностики у телемедичних системах безпосередньо залежить від чистоти вхідного сигналу. Біометричні сигнали, особливо ЕКГ, мають

низьку амплітуду (мілівольти) і схильні до впливу перешкод: мережевих наводок (50 Гц), м’язових артефактів та дрейфу ізолінії.

Для очищення сигналу в системі реалізовано каскад цифрових фільтрів, що працюють у режимі реального часу на стороні мікроконтролера (Edge Computing).

Фільтр високих частот (ФВЧ / High-Pass Filter) призначений для усунення дрейфу ізолінії, спричиненого диханням пацієнта або нещільним контактом електродів. Частота зрізу f_c = 0,5 Гц дозволяє відсікти низькочастотні коливання, зберігаючи при цьому морфологію зусця P та сегмента ST. Один із підходів – використання різницевого фільтра:

$$y[n] = x[n] - x[n-1] \tag{2.1}$$

Метод використання різницевого фільтра дозволяє виділити швидкі зміни сигналу, що є важливими, наприклад, при аналізі ЕКГ. Перевагами методу є ефективне видалення повільних трендів; підвищення чутливості до змін; недоліками – підсилення шумів; потреба у додатковому згладжуванні.

Фільтр низьких частот (ФНЧ / Low-Pass Filter) використовується для усунення високочастотних м’язових шумів (електроміографічних перешкод) та радіочастотних наводок, які не несуть корисної інформації. Частота зрізу f_c = 40 Гц. Оскільки основна енергія ЕКГ-сигналу зосереджена в діапазоні до 35-40 Гц, таке обмеження не спотворює QRS-комплекс.

Фільтр низьких частот використовується для високочастотних шумів. Такий фільтр пропускає повільні зміни сигналу та згладжує різкі коливання. Найпростішим варіантом є ковзне середнє:

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i] \tag{2.2}$$

де $x[n]$ – вхідний сигнал;

$y[n]$ – відфільтрований сигнал;

N – розмір вікна згладжування.

Перевагами методу ковзне середнє є простота реалізації; низькі обчислювальні витрати; недоліками – затримка сигналу; згладжування пікових значень.

Режекторний (Notch) фільтр слугує для видалення вузькосмугової перешкоди від побутової електромережі (50 Гц). Використовується рекурсивний фільтр з глибоким загородженням на частоті 50 Гц.

У практичних системах часто застосовується послідовне використання фільтрів, а саме, фільтр високих частот використовується для усунення дрейфу; фільтр низьких частот – для згладжування шумів. Такий підхід дозволяє сформувати смуговий фільтр, який пропускає лише корисний діапазон частот біосигналу.

Для програмної реалізації на ESP32 використовується різницеве рівняння фільтра першого порядку:

$$y[n] = \alpha \cdot x[n] + (1 - \alpha) \cdot y[n-1] \tag{2.3}$$

де α – коефіцієнт згладжування, що розраховується виходячи з частоти дискретизації.

Загальна логіка обробки біосигналу може бути представлена у вигляді наступної послідовності: Зчитування даних із сенсора. → Перевірка коректності значень (фільтрація викидів). → Застосування фільтра високих частот. → Застосування фільтра низьких частот. → Нормалізація сигналу. → Передача оброблених даних на сервер або в систему аналізу. Для підвищення функціональності системи передбачено механізм інтелектуальних тривожних

сповіщень (Smart Alarms), який реагує на відхилення біометричних показників від допустимих норм. Система Smart Alarms базується на встановленні граничних значень параметрів; аналізі динаміки змін; урахуванні тривалості відхилення; фільтрації хибних спрацювань.

Використовується порогова модель, яка полягає у тому, що для кожного параметра задаються допустимі межі: мінімальне значення X_{min}; максимальне значення X_{max}. Умова спрацювання тривоги:

$$X < X_{min} \text{ , або } X > X_{max} \quad (2.4)$$

Однак проста порогова логіка може призводити до частих хибних спрацювань, що призводить до втоми медичного персоналу та ігнорування критичних станів, тому застосовуються додаткові механізми. Для підвищення надійності використовуються такі підходи, як часова затримка – сигнал повинен залишатися за межами норми певний час; ковзне середнє – аналізується не миттєве значення, а усереднене; гістерезис – різні пороги для включення та вимкнення тривоги. Система може реагувати не тільки на перевищення порогів, а й на небезпечну динаміку змін, а саме, різке зростання або падіння показника; стабільне відхилення від норми; нестабільність сигналу. Це дозволяє виявляти критичні стани ще до досягнення граничних значень.

Алгоритм роботи Smart Alarms можна представити наступним чином:

Отримання обробленого сигналу. → Порівняння значення з допустимими межами. → Перевірка тривалості відхилення. → Аналіз тренду зміни сигналу. → Формування події тривоги. → Надсилання сповіщення користувачу (push, SMS, веб-інтерфейс).

Алгоритм прийняття рішення про сповіщення складається з таких основних блоків, як валідація через акселерометр, багатокритеріальний аналіз (ПТСР-детектор), критичні пороги (Hard Limits).

Валідація через акселерометр. Якщо ЧСС > 120 уд/хв, система перевіряє дані з акселерометра. Якщо зафіксовано високу рухову активність – стан інтерпретується як фізичне навантаження (тривога не надсилається). Якщо активність низька (спокій) – стан класифікується як тахікардія або панічна атака (активація сповіщення).

Багатокритеріальний аналіз (ПТСР-детектор). Система відстежує одночасну зміну трьох показників: різке падіння варіабельності серцевого ритму (ВСР) + зростання ЧСС + зміна провідності шкіри. При виявленні такого патерну система ініціює «М'яке сповіщення» пацієнту (пропозиція дихальної вправи) та запис 30-секундної ділянки ЕКГ для лікаря.

Критичні пороги (Hard Limits). Незалежно від активності, сповіщення надсилається миттєво при досягненні критичних значень: $SpO_2 < 90$,
ЧСС > 180 уд/хв або ЧСС < 40 уд/хв.

Таблиця 2.3 – Рівні пріоритетності сповіщень

Рівень	Стан	Дія системи
Низький	Відхилення тренду, наприклад погіршення сну	Запис у щоденник, звіт лікарю в кінці дня.
Середній	Ознаки стресу / Панічна атака	Пуш-повідомлення пацієнту, помітка в Dashboard.
Високий	Критична аритмія / Гіпоксія	Звуковий сигнал, SMS довіреній особі, Alert для лікаря.

Запропонована комбінація методів цифрової фільтрації та багатокритеріальної логіки Smart Alarms дозволяє мінімізувати кількість

хибних спрацювань та забезпечити високу достовірність автоматизованої діагностики функціонального стану пацієнта в домашніх умовах.

2.3 Математичне моделювання процесу передачі даних у реальному часі

У системах моніторингу біосигналів одним із ключових аспектів є забезпечення надійної та своєчасної передачі даних від сенсорних пристроїв до серверної частини. Особливого значення це набуває у випадках обробки критичної медичної інформації, де навіть незначні затримки або втрати пакетів можуть призвести до некоректних рішень або запізнілої реакції системи. Математичне моделювання процесу передачі даних дозволяє оцінити продуктивність системи, виявити вузькі місця та обґрунтувати вибір технологій зв'язку.

Процес передачі даних у IoT-системі можна представити як послідовність джерело даних → канал зв'язку → приймач даних. Основними параметрами цієї моделі є швидкість генерації даних; пропускна здатність каналу; затримка передачі; ймовірність втрати пакетів.

Пропускна здатність каналу визначає максимальний обсяг даних, який може бути переданий за одиницю часу, і є критичним параметром для систем реального часу. Математично пропускна здатність визначається як:

$$C = \frac{V}{T} \tag{2.5}$$

де C – пропускна здатність (біт/с); V – обсяг переданих даних (біт); T – час передачі (с).

Для IoT-систем важливо забезпечити умову:

$$C \geq R \tag{2.6}$$

де R – швидкість генерації даних сенсорами.

Нехай система передає, наприклад, 10 вимірювань за секунду; кожне вимірювання – 32 біти. Тоді $R=10 \cdot 32=320$ біт/с.

З урахуванням службових заголовків та протоколів (приблизно +50%) $R_{total} \approx 480$ біт/с. Таким чином, навіть низькошвидкісні канали можуть забезпечити передачу таких даних.

Таблиця 2.4 – Порівняння каналів зв'язку

Технологія	Пропускна здатність	Затримка	Область застосування
Wi-Fi	до 100 Мбіт/с	низька	локальні мережі
GSM (4G)	до 10–50 Мбіт/с	середня	мобільні системи
LoRa	до 50 кбіт/с	висока	віддалені сенсори
Bluetooth	до 2 Мбіт/с	низька	персональні пристрої

Для коректної передачі сигналу ЕКГ необхідно забезпечити достатню частоту дискретизації (fs). Згідно з теоремою Котельникова (Найквіста-Шеннона), частота дискретизації повинна бути принаймні вдвічі більшою за максимальну частоту спектра сигналу. Для діагностичного ЕКГ fmax ≈ 100 Гц, тому обираємо fs = 250 Гц.

Потік даних від одного каналу ЕКГ при 12-бітній глибині АЦП (n = 12 біт) розраховується як:

$$R_{ECG} = f_s \, n = 250 \, 12 = 3000 \text{ біт/с} \approx 3 \text{ кбіт/с} \tag{2.7}$$

Враховуючи передачу додаткових параметрів (SpO_2, температура, дані акселерометра) та оверхед (службові заголовки) протоколу MQTT/TCP/IP, сумарний потік даних від одного пацієнта становить:

$$R_{total} = (R_{ECG} + R_{other})\, k_{overhead} \approx (3 + 1)\, 1{,}2 = 4{,}8 \text{ кбіт/с} \quad (2.8)$$

Це значення є мізерним для сучасних мереж Wi-Fi/4G, що дозволяє системі стабільно працювати навіть при значному погіршенні якості зв'язку (Edge/GPRS).

Вибір каналу зв’язку визначається компромісом між швидкістю, енергоспоживанням та затримками.

Затримка передачі є критичним параметром для систем реального часу. Загальна затримка визначається як сума окремих складових:

$$T_{total} = T_{tx} + T_{prop} + T_{proc} + T_{queue} \quad (2.9)$$

де T_{tx} – час передачі пакету в каналах зв'язку; T_{prop} – час розповсюдження сигналу в мережі.; T_{proc} – час обробки та фільтрації на ESP32; T_{queue} – час очікування в чергах маршрутизаторів.

Для медичних систем реального часу критичне значення $T_{total} \leq 250$ мс. При використанні протоколу MQTT з брокером, розташованим у хмарі, середня затримка T_{total} коливається в межах 80 – 150 мс, що повністю відповідає медичним вимогам для дистанційного моніторингу.

T_{tx} – час передачі (Transmission delay) залежить від розміру пакета та пропускної здатності каналу:

$$T_{tx} = \frac{L}{C} \quad (2.10)$$

де L – довжина пакета.

T_{prop} – час поширення (Propagation delay) залежить від відстані та швидкості сигналу.

T_{proc} – час обробки (Processing delay) включає обробку на мікроконтролері та сервері.

T_{queue} – час очікування в черзі (Queue delay), черги виникають при перевантаженні мережі.

Для медичних IoT-систем допустимі значення затримки зазвичай становлять до 100 – 200 мс – для критичних сигналів (ЕКГ); до 1 с – для моніторингових даних.

Передача медичних даних вимагає високого рівня надійності та захисту від помилок, оскільки втрата «R-зубця» на ЕКГ може призвести до помилкової діагностики асистолії.

Для звичайного моніторингу пульсу використовується QoS 0 (Quality of Service) в MQTT (доставка «якнайшвидше»). Для передачі ЕКГ та тривожних сповіщень (Smart Alarms) використовується QoS 1 (гарантована доставка хоча б один раз із підтвердженням PUBACK).

Для компенсації джиттера (нерівномірності затримок) на стороні клієнтського додатка (React) реалізовано адаптивний буфер (Jitter Buffer). Він накопичує 500 мс сигналу перед візуалізацією, що забезпечує плавність відображення кривої ЕКГ навіть при короткочасних «латах» мережі.

Кожен пакет містить контрольну суму (CRC-32) та порядковий номер. Якщо пакет втрачено, сервер ініціює запит на повторну передачу або маркує ділянку графіка як «дані втрачені», щоб не вводити лікаря в оману.

Математичне моделювання підтверджує, що обрана архітектура на базі ESP32 + MQTT + FastAPI , базується на аналізі ключових параметрів системи: пропускної здатності, затримок передачі та надійності доставки даних та забезпечує необхідний баланс між швидкістю передачі та надійністю. Низька вимога до пропускної здатності (4,8 кбіт/с) робить систему ідеальною для використання у важкодоступних регіонах, де якість інтернету може бути нестабільною.

Мікроконтролер ESP32 формує потік даних із певною швидкістю R. Як було показано в моделі:

$$C \geq R, \tag{2.11}$$

де C – пропускна здатність каналу.

ESP32 здатен передавати дані через Wi-Fi зі швидкістю, що значно перевищує потреби біосигналів (кілобіти/с проти мегабіт/с). Протокол MQTT мінімізує службовий трафік (малий розмір пакетів), що зменшує навантаження на канал. Серверна частина на FastAPI обробляє запити асинхронно, що дозволяє обслуговувати багато пристроїв без затримок.

Математично виконується умова:

$$C_{real} \geq R_{system} \tag{2.12}$$

Для виявлення помилок використовуються контрольні суми (Checksum); циклічний надлишковий код (CRC). Ідея полягає у додаванні до пакета спеціального коду, що дозволяє перевірити його цілісність на стороні приймача. У разі виявлення помилки або втрати пакета застосовується механізм повторної передачі автоматичний запит повтору (ARQ); підтвердження доставки (ACK/NACK). Для критичних даних використовується дублювання пакетів; передача з перекриттям (overlapping data); буферизація на стороні пристрою. Для передачі медичних даних доцільно використовувати MQTT з QoS (рівні гарантії доставки); HTTPS (захищена передача); TCP (гарантована доставка).

Ймовірність успішної передачі пакета можна оцінити як:

$$P_{success} = (1 - P_{error})^n \tag{2.13}$$

де P_{error} – ймовірність помилки в одному біті; n – кількість бітів у пакеті.

Зменшення розміру пакета або використання корекції помилок дозволяє підвищити надійність передачі.

Загальна затримка системи формула (2.9) для обраної архітектури ESP32 забезпечує малий час обробки (Tproc ≈ кілька мс); MQTT працює поверх TCP і використовує легкий механізм доставки → мінімальний T_{tx}; FastAPI (асинхронний сервер) зменшує T_{queue} навіть при великій кількості клієнтів.

Практично затримка передачі становить десятки мілісекунд, що відповідає вимогам реального часу для біосигналів.

Надійність оцінюється через ймовірність успішної доставки формула (2.13). Архітектура підвищує це значення завдяки MQTT QoS, використанню TCP (контроль помилок і повторна передача); можливості буферизації даних на ESP32, що забезпечує $P_{success} \to 1$ навіть у нестабільних мережах.

QoS 0 – швидко, без гарантій;

QoS 1 – доставка гарантована (мінімум один раз);

QoS 2 – доставка без дублювання.

Таблиця 2.5 – Обрана архітектура на базі ESP32 + MQTT + FastAPI є оптимальною:

Критерій	Як забезпечується
Висока швидкість	Wi-Fi + легкий протокол MQTT
Низька затримка	асинхронна обробка FastAPI
Надійність	MQTT QoS + TCP
Масштабованість	серверна обробка запитів

Математичне моделювання параметрів передачі даних (пропускної здатності, затримок та ймовірності успішної доставки) показало, що використання архітектури на базі ESP32, протоколу MQTT та серверної платформи FastAPI забезпечує виконання умов реального часу ($C \geq R$, $T_{total} < T_{critical}$) при високій надійності передачі ($P_{success} \to 1$). Це свідчить про

досягнення оптимального балансу між швидкістю обміну даними та гарантіями їх доставки.

Висновок до розділу 2.

У розділі 2 було розроблено архітектуру IoT-комплексу, яка базується на чотирирівневій моделі: датчики, мікроконтролер, хмарний сервер та клієнтський додаток. Така структура забезпечує ефективний збір, передачу та обробку даних у реальному часі. Обґрунтовано використання бази даних часових рядів, що дозволяє підвищити продуктивність системи та забезпечити ефективний аналіз накопичених даних.

Запропоновано математичну модель процесу передачі даних у реальному часі, що дозволяє оцінити ефективність функціонування IoT-системи. Проведено аналіз пропускної здатності каналів зв’язку та визначено умови їх достатності для передачі біосигналів. Розглянуто структуру затримок та встановлено їх допустимі межі для медичних застосувань. Запропоновано методи забезпечення цілісності даних, включаючи контроль помилок, повторну передачу та використання надійних протоколів, що підвищує достовірність і безпеку передачі критичної інформації.

РОЗДІЛ 3. ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА АПРОБАЦІЯ ТЕЛЕМЕДИЧНОЇ СИСТЕМИ

3.1 Програмна реалізація серверної частини та інтеграція з медичними API

Основною метою розробки програмного забезпечення сервера є забезпечення надійного прийому, зберігання, обробки та захищеного доступу до даних у режимі реального часу. Серверна частина виконує роль центрального вузла системи та забезпечує прийом даних від IoT-пристроїв; агрегацію та обробку інформації; збереження у базі даних; надання доступу клієнтським додаткам; інтеграцію з медичними інформаційними системами. Архітектура реалізується за принципом RESTful-сервісу з використанням сучасних веб-технологій.

Для реалізації серверної логіки обрано фреймворк FastAPI, який забезпечує високу продуктивність, асинхронність та зручність розробки. Використання FastAPI є доцільним з огляду на такі переваги, як асинхронна обробка запитів (async/await); висока швидкість роботи (порівнянна з Node.js); автоматична генерація API-документації; вбудована валідація даних (Pydantic); простота інтеграції з іншими сервісами. Зазначені перваги є особливо важливими для IoT-систем, де необхідно обробляти потоки даних від великої кількості пристроїв.

Основними функціональними модулями серверної частини є API прийому даних, модуль агрегації, модуль збереження, API доступу до даних. Модуль API прийому даних забезпечує прийом даних від ESP32 через HTTP або MQTT; перевірку формату повідомлення; первинну обробку. Модуль агрегації дозволяє об’єднати дані з різних сенсорів; зробити усереднення значень; формування часових рядів. Модуль збереження робить запис у базу даних (InfluxDB або TimescaleDB); оптимізацію структури зберігання. API доступу до даних призначено для отримання історичних даних; фільтрації за часом; передачі даних клієнтам. Приклад структури API надано у тблиці 3.1.

Таблиця 3.1 – Приклад структури API

Метод	Endpoint	Опис
POST	/api/data	Надсилання даних із сенсора
GET	/api/data	Отримання даних
GET	/api/data/{id}	Дані конкретного пристрою
GET	/api/stats	Агреговані показники

Розглянемо на прикладі логіки обробки запиту. Пристрій надсилає JSON-пакет:

```
{
  "device_id": "sensor_1",
  "heart_rate": 75,
  "temperature": 36.6,
  "timestamp": 1710000000
}
```

Сервер перевіряє структуру; фільтрує некоректні значення; записує дані у базу; повертає відповідь про успішну обробку. Оскільки система працює з медичною інформацією, необхідно забезпечити високий рівень безпеки та конфіденційності даних.

JSON Web Token (JWT) це стандартний механізм автентифікації, який дозволяє безпечно передавати інформацію між клієнтом і сервером у вигляді підписаного токена. Він широко використовується в веб-сервісах та IoT-системах, зокрема для захисту доступу до медичних даних. Пояснимо принцип його роботи. Користувач проходить авторизацію (логін/пароль). Сервер перевіряє дані і, якщо все правильно, то генерує JWT-токен. Клієнт додає токен до кожного запиту. Сервер перевіряє токен перед наданням доступу.

JWT складається з трьох частин

HEADER.PAYLOAD.SIGNATURE

Header (заголовок) містить інформацію про тип токена і алгоритм підпису.

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Payload (корисні дані) містить інформацію про користувача:

```
{
  "user_id": "12345",
  "role": "doctor",
  "exp": 1710000000
}
```

user_id – ідентифікатор користувача; role – роль (лікар, пацієнт); exp – час завершення дії токена.

Signature (підпис) формується шляхом шифрування Header і Payload за допомогою секретного ключа:

Signature=HMACSHA256(base64Url(Header)+"."+base64Url(Payload),SecretKey)Signature = HMACSHA256(base64Url(Header) + "." + base64Url(Payload), SecretKey)Signature=HMACSHA256(base64Url(Header)+"."+base64Url(Payload), SecretKey)

Це гарантує, що токен не був змінений.

Пояснимо принцип роботи JWT. Користувач проходить авторизацію (логін/пароль). Сервер перевіряє дані і, якщо все правильно, то генерує JWT-токен. Клієнт зберігає токен (у браузері або додатку). Клієнт додає токен до кожного запиту:

Authorization: Bearer <JWT>

Сервер перевіряє підпис токена. Якщо токен валідний – надає доступ до даних.

Перевагами JWT є відсутність збереження сесій на сервері (stateless); висока швидкість перевірки; масштабованість системи; можливість передачі ролей і прав доступу.

Розглянемо практичний сценарій для системи моніторингу стану пацієнта, наприклад, доступ лікаря до даних пацієнта.

Крок 1. Авторизація. Лікар входить у систему:

POST /api/login

Сервер генерує токен:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer"
}
```

Крок 2. Отримання даних із сенсорів. Лікар робить запит:

GET /api/data?sensor_id=patient_1

Authorization: Bearer <JWT>

Крок 3. Перевірка на сервері

Сервер декодує JWT; перевіряє підпис; перевіряє термін дії (exp); визначає роль (doctor).

Крок 4. Надання доступу. Якщо користувач має права – сервер повертає дані:

```
{
  "heart_rate": 78,
  "temperature": 36.7,
  "timestamp": 1710000000
}
```

Розглянемо сценарій на прикладі для IoT-пристрою (ESP32)

Пристрій також може використовувати JWT:

POST /api/data

Authorization: Bearer <device_token>

Токен містить:

```
{
  "device_id": "sensor_1",
  "type": "iot_device"
}
```

Це дозволяє перевірити, що дані надсилає саме дозволений пристрій; запобігти підробці даних.

JWT особливо важливий у медичних IoT-системах, оскільки забезпечує контроль доступу до конфіденційних даних; дозволяє реалізувати рольову модель (лікар/пацієнт); захищає API від несанкціонованих запитів.

Розглянемо сценарій на прикладі FastAPI + JWT.

Встановлення залежностей

```
pip install fastapi uvicorn python-jose passlib[bcrypt]
```

Основний код серверу на Python:

```
from fastapi import FastAPI, Depends, HTTPException, status
from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm
from jose import JWTError, jwt
from datetime import datetime, timedelta

# Секретний ключ (у реальному проєкті зберігати в .env)
SECRET_KEY = "mysecretkey"
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30

app = FastAPI()

# Фейкова база користувачів
fake_users_db = {
```

```
    "doctor1": {
        "username": "doctor1",
        "password": "1234",
        "role": "doctor"
    }
}

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")


# Функція створення JWT
def create_access_token(data: dict, expires_delta: timedelta = None):
    to_encode = data.copy()
    expire = datetime.utcnow() + (expires_delta or timedelta(minutes=15))
    to_encode.update({"exp": expire})
    return jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)


# Авторизація користувача
@app.post("/token")
def login(form_data: OAuth2PasswordRequestForm = Depends()):
    user = fake_users_db.get(form_data.username)
    if not user or user["password"] != form_data.password:
        raise HTTPException(status_code=401, detail="Invalid credentials")

    access_token = create_access_token(
        data={"sub": user["username"], "role": user["role"]}
    )
    return {"access_token": access_token, "token_type": "bearer"}
```

```
# Перевірка токена
def get_current_user(token: str = Depends(oauth2_scheme)):
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        username: str = payload.get("sub")
        role: str = payload.get("role")
        if username is None:
            raise HTTPException(status_code=401, detail="Invalid token")
        return {"username": username, "role": role}
    except JWTError:
        raise HTTPException(status_code=401, detail="Invalid token")


# Захищений endpoint (наприклад, дані біосигналів)
@app.get("/api/data")
def get_data(current_user: dict = Depends(get_current_user)):
    return {
        "user": current_user["username"],
        "heart_rate": 76,
        "temperature": 36.6
    }
```

POST /token

Користувач (лікар) вводить логін/пароль → отримує JWT.

GET /api/data

Клієнт додає токен:

Authorization: Bearer <JWT>

Сервер перевіряє токен; визначає користувача; надає доступ до даних.

У контексті IoT-системи ESP32 або клієнтський додаток надсилає запити до API; JWT гарантує, що доступ мають лише авторизовані користувачі або пристрої; можна легко реалізувати ролі: лікар / пацієнт / пристрій (таблиця 3.2).

У системах моніторингу біосигналів необхідно чітко розмежувати права доступу між різними типами користувачів. Це забезпечує безпеку медичних даних та коректну роботу системи. Реалізація ролей виконується за допомогою механізму Role-Based Access Control (RBAC) із використанням JWT.

Таблиця 3.2 – Концепція ролей у системі

Роль	Опис	Права доступу
Лікар	Медичний персонал	Перегляд усіх даних, аналітика
Пацієнт	Користувач системи	Перегляд власних даних
Пристрій	IoT-сенсор (ESP32)	Надсилання даних

Опрацюємо збереження ролі у JWT.

При створенні токена в нього додається поле role:

```
data={"sub": user["username"], "role": user["role"]}
```

Приклад payload

```
{
  "sub": "doctor1",
  "role": "doctor",
  "exp": 1710000000
}
```

Це дозволяє серверу визначити права доступу без звернення до бази даних.

Розглянемо як здійснюється реалізація перевірки ролей.

Функція перевірки користувача

```
def get_current_user(token: str = Depends(oauth2_scheme)):
    payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
    return {
        "username": payload.get("sub"),
        "role": payload.get("role")
    }
```

Функція перевірки ролі

```
from fastapi import Depends, HTTPException

def require_role(required_roles: list):
    def role_checker(user: dict = Depends(get_current_user)):
        if user["role"] not in required_roles:
            raise HTTPException(status_code=403, detail="Access denied")
        return user
    return role_checker
```

Розпишемо як реалізується захист API за ролями. Наприклад, доступ лише для лікаря

```
@app.get("/api/doctor/data")
def get_all_patients(user=Depends(require_role(["doctor"]))):
    return {"message": "Доступ до всіх даних пацієнтів"}
```

Доступ для пацієнта (тільки свої дані)

```
@app.get("/api/patient/data")
def get_own_data(user=Depends(require_role(["patient"]))):
   return {"message": f"Дані пацієнта {user['username']}"}
```

Доступ для пристрою (надсилання даних)

```
@app.post("/api/device/data")
def send_data(user=Depends(require_role(["device"]))):
   return {"message": "Дані прийнято від пристрою"}
```

Опишемо особливості реалізації ролі «пристрій»

Для IoT-пристроїв (наприклад, ESP32) доцільно створювати окремі токени; не використовувати логін/пароль у класичному вигляді; прив’язувати токен до device_id.

Приклад payload для пристрою:

```
{
  "sub": "sensor_1",
  "role": "device"
}
```

Це дозволяє ідентифікувати конкретний сенсор; контролювати, які пристрої мають доступ до API; запобігти підробці даних.

Проаналізуємо додаткове розмежування доступу.

Для більш гнучкого контролю можна додати:

1) Прив’язку пацієнта до лікаря, ураховуючи, що лікар бачить тільки «своїх» пацієнтів;

2) Прив’язку пристрою до пацієнта, зважаючи, що сенсор передає дані лише для конкретного користувача;

3) Ролі з розширеними правами, наприклад, адміністратор; аналітик.

Користувач або пристрій отримує JWT. У кожному запиті передає токен. Сервер перевіряє токен; визначає роль; перевіряє права доступу. Якщо роль дозволена – виконується запит.

Реалізація ролей «лікар / пацієнт / пристрій» на основі JWT дозволяє ефективно розмежувати доступ до даних у системі моніторингу біосигналів. Використання Role-Based Access Control забезпечує високий рівень безпеки, гнучкість управління правами та масштабованість системи, що є критично важливим для обробки медичної інформації.

JSON Web Token забезпечує ефективний механізм автентифікації та авторизації в IoT-системах моніторингу біосигналів. Його використання дозволяє організувати безпечний доступ до медичних даних без перевантаження серверної частини, що є критично важливим для систем реального часу.

Розглянемо процес використання OAuth 2.0 для авторизації користувачів через сторонні сервіси (наприклад, медичні платформи); надання доступу до даних без передачі паролів; інтеграції з електронними медичними системами (EHR); розмежування прав доступу через scopes.

Таблиця 3.3 – Основні ролі в OAuth 2.0

Роль	Опис	Приклад у системі
Resource Owner	Власник даних	Пацієнт
Client	Додаток	Веб або мобільний додаток
Authorization Server	Сервер авторизації	Google / медичний API
Resource Server	Сервер ресурсів	Backend (FastAPI)

Для медичних систем найбільш безпечним є сценарій Authorization Code Flow, оскільки токени не передаються напряму через браузер; використовується серверна перевірка; підтримується високий рівень безпеки.

Крок 1. Запит авторизації. Користувач (пацієнт або лікар) переходить за посиланням:

GET /authorize?client_id=CLIENT_ID&redirect_uri=URL&scope=read_data

Система перенаправляє його на сервер авторизації.

Крок 2. Авторизація користувача. Користувач входить у систему (наприклад, медичний сервіс); підтверджує доступ до своїх даних.

Крок 3. Отримання authorization code. Після успішного входу сервер повертає:

GET /callback?code=AUTH_CODE

Крок 4. Обмін коду на токен. Backend-сервер (на базі FastAPI) виконує запит:

POST /token

і отримує:

```
{
  "access_token": "abc123",
  "refresh_token": "xyz456",
  "expires_in": 3600
}
```

Крок 5. Доступ до медичних даних. Сервер використовує access_token для запитів:

GET /medical/data

Authorization: Bearer abc123

Реалізація на FastAPI (спрощений приклад). Налаштування OAuth клієнта

```
from fastapi import FastAPI
import requests

app = FastAPI()

CLIENT_ID = "your_client_id"
```

```
CLIENT_SECRET = "your_client_secret"
REDIRECT_URI = "http://localhost:8000/callback"
Перенаправлення на сервер авторизації
@app.get("/login")
def login():
    auth_url = (
        f"https://provider.com/oauth/authorize?"

f"client_id={CLIENT_ID}&redirect_uri={REDIRECT_URI}&response_type=code
"
    )
    return {"auth_url": auth_url}

Обробка callback

@app.get("/callback")
def callback(code: str):
    token_url = "https://provider.com/oauth/token"

    response = requests.post(token_url, data={
        "client_id": CLIENT_ID,
        "client_secret": CLIENT_SECRET,
        "code": code,
        "redirect_uri": REDIRECT_URI,
        "grant_type": "authorization_code"
    })

    return response.json()
```

Простежимо процес використання OAuth 2.0 у системі моніторингу.

Сценарій 1: Лікар отримує доступ до даних пацієнта, який реалізується за наступними кроками: лікар авторизується через OAuth; отримує access_token; переглядає біосигнали пацієнта через API.

Сценарій 2: Інтеграція з медичною системою втілюється наступним чином: система отримує доступ до EHR; зберігає історію пацієнта; синхронізує дані з IoT-сенсорами.

Сценарій 3: Мобільний додаток працює наступним чином: користувач входить через Google або медичний сервіс; додаток отримує токен; відображає дані з серверу.

Перевагами використання OAuth 2.0 є відсутність передачі паролів; централізована авторизація; можливість інтеграції з зовнішніми сервісами; гнучке управління правами доступу (scopes); підвищений рівень безпеки.

Реалізація OAuth 2.0 у системі моніторингу біосигналів дозволяє забезпечити безпечний доступ до медичних даних та інтеграцію із зовнішніми інформаційними системами. Використання сценарію Authorization Code Flow гарантує високий рівень захисту, а поєднання з JWT забезпечує ефективну роботу внутрішнього API, що робить архітектуру системи гнучкою, масштабованою та придатною для використання у медичних застосуваннях.

Для забезпечення безпеки даних реалізуються такі механізми, як використання HTTPS (шифрування трафіку); перевірка прав доступу; обмеження частоти запитів (rate limiting); логування доступу; розмежування ролей (користувач, лікар, адміністратор).

3.2 Розробка та реалізація підсистеми автентифікації і авторизації користувачів у IoT-системі моніторингу біосигналів

У сучасних інформаційних системах, зокрема в IoT-рішеннях медичного призначення, питання безпеки доступу до даних є критично важливим.

Біосигнали (частота серцевих скорочень, температура тіла, рівень кисню тощо) належать до категорії конфіденційної інформації, тому їх обробка повинна відповідати вимогам захисту даних, цілісності та доступності. Підсистема автентифікації та авторизації є ключовим компонентом, що забезпечує контроль доступу до ресурсів системи та запобігає несанкціонованому використанню даних.

У контексті інформаційних систем необхідно чітко розмежовувати такі два базові процеси, як автентифікація та авторизація. Автентифікація – процес перевірки особи користувача або пристрою (відповідь на питання: хто це?). Авторизація – процес визначення прав доступу (відповідь на питання: що дозволено?). У IoT-системах ці процеси застосовуються як до користувачів (лікарів, пацієнтів), так і до пристроїв (сенсорів, мікроконтролерів).

IoT-системи мають ряд специфічних особливостей, що ускладнюють забезпечення безпеки, а саме, велика кількість підключених пристроїв; обмежені обчислювальні ресурси мікроконтролерів; передача даних через відкриті мережі; необхідність роботи в режимі реального часу; підвищені вимоги до конфіденційності медичних даних. Зазначені особливості обумовлюють необхідність використання легких, але надійних механізмів автентифікації та авторизації.

Для організації доступу до ресурсів системи використовуються такі моделі, як дискреційна модель, мандатна модель, рольова модель.

У дискреційній моделі (DAC) доступ визначається власником ресурсу. Недоліком є складність управління при великій кількості користувачів. У мандатній моделі (MAC) доступ обумовлений політикою безпеки. Використовується у високозахищених системах. Для рольової моделі (RBAC) доступ визначається роллю користувача. RBAC модель є найбільш доцільною для IoT-медичних систем. У нашій дипломній роботі використовується саме рольова модель, яка передбачає такі ролі, як лікар; пацієнт; пристрій. У сучасних веб- та IoT-системах широко застосовується токен-орієнтований

підхід, який базується на використанні спеціальних маркерів доступу (токенів).

Одним із найпоширеніших стандартів є JSON Web Token, основними властивостями якого є компактність і самодостатність; можливість зберігання даних про користувача; криптографічний підпис; відсутність необхідності зберігання сесій на сервері. Це дозволяє ефективно використовувати JWT у розподілених системах із великою кількістю клієнтів, а конкретно, медичних системах.

Для забезпечення безпечної авторизації через сторонні сервіси використовується OAuth 2.0, що дозволяє надавати доступ до ресурсів без передачі пароля; делегувати повноваження стороннім сервісам; реалізувати контроль доступу через scopes. У медичних системах це особливо важливо, оскільки користувачі можуть авторизуватися через надійні провайдери; зменшується ризик витоку облікових даних; забезпечується інтеграція з медичними інформаційними системами.

Найбільш безпечним сценарієм використання OAuth 2.0 є Authorization Code Flow, який передбачає перенаправлення користувача на сервер авторизації; підтвердження доступу; отримання authorization code; обмін коду на access token; використання токена для доступу до ресурсів. Цей підхід забезпечує високий рівень захисту, оскільки токени не передаються безпосередньо через клієнт. У практичних системах часто використовується комбінований підхід, в рамках якого OAuth 2.0 застосовується для автентифікації користувача; JWT – для подальшої авторизації в системі. Такий підхід дозволяє делегувати автентифікацію зовнішнім сервісам; зберігати контроль доступу на стороні власного серверу; зменшити навантаження на систему.

Ми використовуємо OAuth 2.0 для входу через Google, а після цього генеруємо власний JWT для роботи API. Google (OAuth) → підтверджує особу. FastAPI → видає JWT. API → працює через JWT. Розглянемо як реалізуються ролі (таблиця 3.4).

Таблиця 3.4 – Ролі в системі

Компонент	Роль
Google	Сервер авторизації
FastAPI	Сервер ресурсів
Користувач	Лікар / пацієнт
Клієнт	Веб/мобільний додаток

КРОК 1 – Запуск авторизації

```
@app.get("/login")
async def login(request: Request):
    redirect_uri = request.url_for("auth_callback")
    return await oauth.google.authorize_redirect(request, redirect_uri)
```

Користувач заходить на /login. FastAPI формує URL для Google, відбувається redirect на Google. Користувач бачить: «Увійти через Google».

КРОК 2 – Авторизація в Google.

Користувач вводить логін/пароль Google, підтверджує доступ, Google запам’ятовує: хто користувач, які дані дозволено передати

КРОК 3 – Повернення на сервер (callback).

```
@app.get("/auth/callback")
async def auth_callback(request: Request):
    token = await oauth.google.authorize_access_token(request)
```

Google перенаправляє на:

/auth/callback?code=XXXXX

FastAPI отримує code, обмінює його на токен. Це і є Authorization Code Flow.

КРОК 4 – Отримання даних користувача.

```
user_info = token.get("userinfo")
```

Google повертає:

```
{
  "email": "user@gmail.com",
  "name": "Ivan Ivanov"
}
```

Зауважимо, що пароль НЕ передається, а передається тільки базова інформація.

КРОК 5 – Генерація власного JWT

```
def create_jwt(user_info: dict):
    payload = {
        "sub": user_info["email"],
        "name": user_info.get("name"),
        "role": "doctor",
        "exp": datetime.utcnow() + timedelta(hours=1)
    }
```

Таблиця 3.5 – Внутрішня система доступу

Поле	Значення
sub	email користувача
role	роль (лікар/пацієнт)
exp	час життя

КРОК 6 – Повернення JWT клієнту. Клієнт отримує дані користувача, JWT

```
return {
   "user": user_info,
   "jwt": jwt_token
}
```

КРОК 7 – Доступ до API. Клієнт відправляє JWT. Сервер перевіряє підпис, перевіряє час життя, якщо все добре → дає доступ.

```
@app.get("/api/data")
def get_data(token: str):
   payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
```

У дипломній роботі було реалізованоряд сценарієв.

Сценарій 4: Лікар дивиться дані пацієнта. Сценарій реалізується наступним чином: Лікар натискає «Увійти через Google». Проходить авторизацію. Отримує JWT. Робить запит:

GET /api/data

Authorization: Bearer JWT

Сервер повертає:

{

"heart_rate": 78,
"temperature": 36.7
}

При реалізації Сценарію: пацієнт

Пацієнт входить через Google, отримує роль «patient», бачить тільки свої дані.

У системі використано OAuth 2.0 для автентифікації користувачів через Google, що дозволяє уникнути зберігання паролів і підвищує рівень безпеки. Після успішного входу сервер генерує власний JWT-токен, який використовується для подальшого доступу до API системи. Це дозволяє розділити процес автентифікації та авторизації та забезпечити ефективну роботу системи в реальному часі.

3.3 Розробка інтерфейсу користувача для візуалізації клінічних показників

Інтерфейс телемедичної системи є критично важливим елементом, оскільки він має забезпечувати швидку інтерпретацію складних біометричних даних як лікарем, так і пацієнтом. Візуалізація біосигналів є важливим елементом медичних інформаційних систем, оскільки дозволяє швидко оцінити стан пацієнта; виявити аномалії; аналізувати динаміку змін; приймати клінічні рішення.

Для реалізації фронтенд-частини обрано фреймворк React.js, що дозволяє створювати динамічні інтерфейси з оновленням даних у реальному часі без перезавантаження сторінки.

Таблиця 3.6 – Основні вимоги до інтерфейсу

Вимога	Опис
Інформативність	відображення ключових параметрів
Реальний час	оновлення без затримок
Зрозумілість	прості графіки та індикатори
Надійність	стабільна робота при потоці даних
Адаптивність	підтримка різних пристроїв

Таблиця 3.7 – Типи візуалізації медичних даних

Тип даних	Спосіб відображення
ЕКГ	лінійний графік
Артеріальний тиск	часовий графік
Пульс	числовий індикатор + графік
Температура	трендова лінія

Інтерфейс розділено на дві функціональні зони згідно з ролями користувачів: кабінет пацієнта та робоче місце лікаря.

Кабінет пацієнта (Мобільний вигляд) є максимально простотим та зрозумілим. Елементами кабінету є поточні значення ЧСС та SpO2, індикатор заряду IoT-пристрою, кнопка швидкого зв'язку з лікарем та розділ із рекомендаціями, наприклад, дихальні вправи при виявленні стресу (рисунок 3.1).

Кабінет пацієнта є адаптивним (mobile-first) веб-додатком на React. У дипломі реалізовано: ЧСС, SpO_2, заряд пристрою, швидкий зв’язок із лікарем, рекомендації, а також оновлення даних у реальному часі через WebSocket (додаток В).

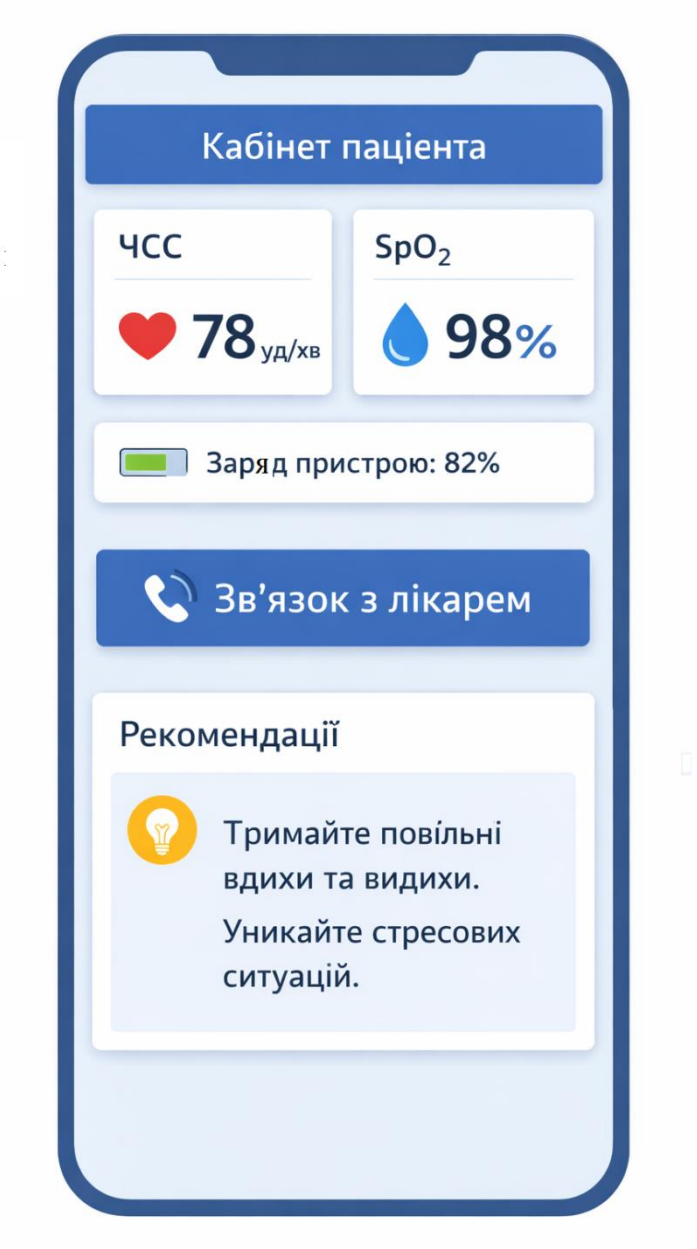


Рисунок 3.1 – Кабінет пацієнта (Мобільний вигляд)

Мобільний інтерфейс пацієнта реалізовано з використанням адаптивного підходу (mobile-first). Додаток забезпечує відображення основних клінічних показників (ЧСС, SpO_2), індикатора стану IoT-пристрою та рекомендацій у разі відхилень. Передача даних здійснюється у реальному часі через WebSocket-з'єднання з сервером на базі FastAPI. Це дозволяє забезпечити оперативне інформування пацієнта та підвищити ефективність моніторингу стану здоров'я.

Інтерфейс лікаря є найбільш функціонально насиченим і призначений для детального аналізу стану пацієнта (рисунок 3.2).

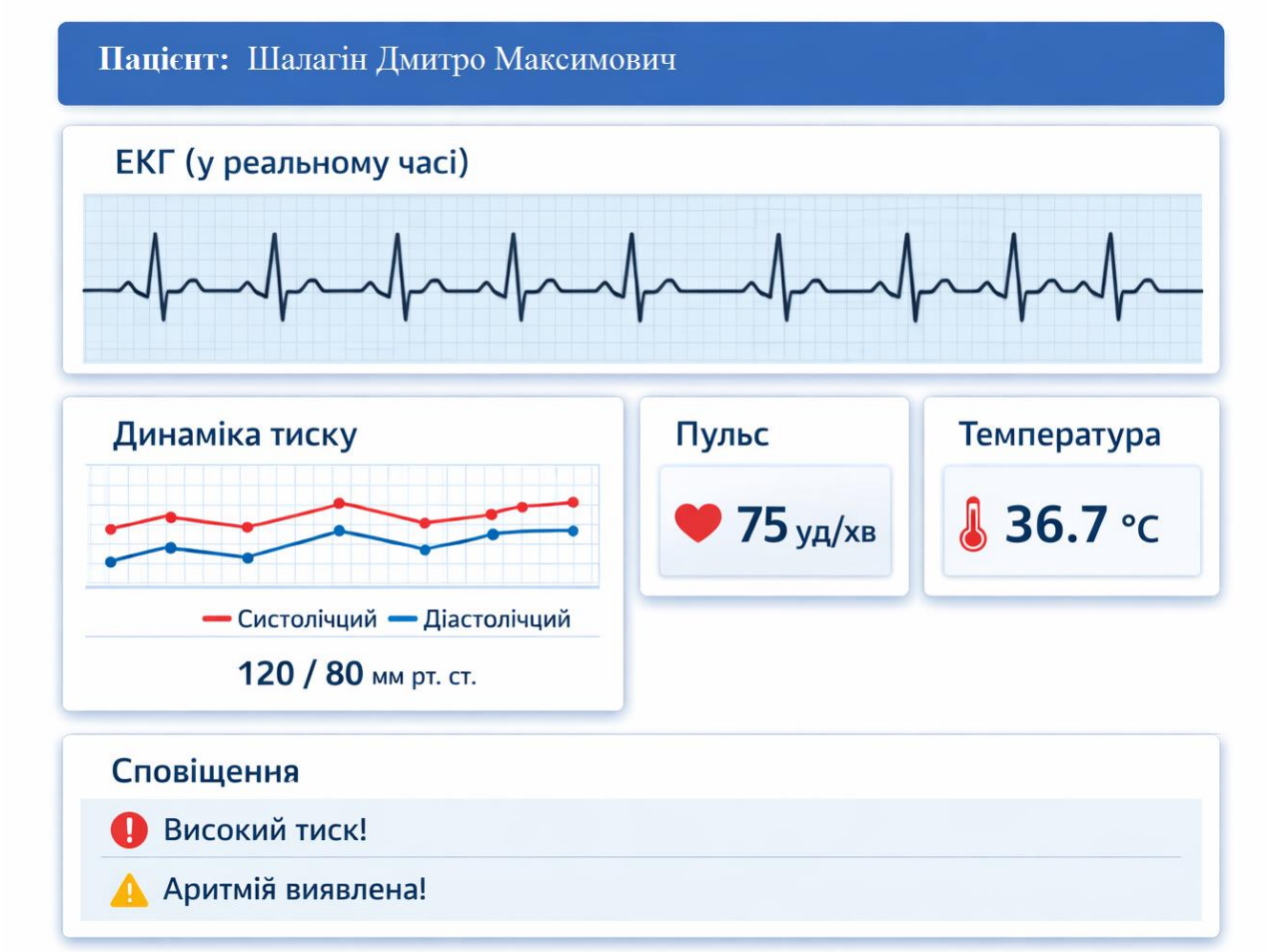


Рисунок 3.2 – Структура Dashboard лікаря

Програмна реалізація Dashboard лікаря базується на клієнт-серверній архітектурі:

ESP32 → Backend (FastAPI) → WebSocket/Socket.io → Frontend (React Dashboard)

Таблиця 3.8 – Основні компоненти Dashboard лікаря

Компонент	Технологія	Функція
IoT-пристрій	ESP32	збір біосигналів
Backend	FastAPI	обробка даних

Компонент	Технологія	Функція
Реальний час	Socket.IO	передача даних
Frontend	React	інтерфейс лікаря
Візуалізація	Recharts	графіки

Сенсори передають дані на сервер (HTTP/MQTT). Backend обробляє дані; зберігає в БД; передає через WebSocket. Frontend отримує дані; оновлює графіки.

Backend відповідає за генерацію потоку даних:

```
from fastapi import FastAPI
from fastapi.responses import JSONResponse
import random

app = FastAPI()

@app.get("/api/data")
def get_data():
    return {
        "heart_rate": random.randint(60, 100),
        "pressure_sys": random.randint(110, 130),
        "pressure_dia": random.randint(70, 90)
    }
```

WebSocket для реального часу:

```
from fastapi import WebSocket
```

```
@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
await websocket.accept()
while True:
data = {
"ecg": [random.random() for _ in range(50)],
"heart_rate": random.randint(60, 100)
}
await websocket.send_json(data)
```

Frontend (React Dashboard)

Підключення до WebSocket

```
import { useEffect, useState } from "react";

function useWebSocket() {
const [data, setData] = useState(null);

useEffect(() => {
const ws = new WebSocket("ws://localhost:8000/ws");

ws.onmessage = (event) => {
setData(JSON.parse(event.data));
};

return () => ws.close();
}, []);
```

```
return data;
}

Відображення ЕКГ (Recharts)

import { LineChart, Line, XAxis, YAxis } from "recharts";

function ECGChart({ ecg }) {
  const chartData = ecg.map((value, index) => ({
    time: index,
    value: value
  }));

  return (
    <LineChart width={600} height={200} data={chartData}>
      <XAxis dataKey="time" />
      <YAxis />
      <Line type="monotone" dataKey="value" />
    </LineChart>
  );
}

Відображення тиску

function PressureChart({ sys, dia }) {
  return (
    <div>
      <p>Систолічний: {sys}</p>
      <p>Діастолічний: {dia}</p>
    </div>
```

```
  );
}
```

Dashboard використовує WebSocket / Socket.IO, React state (useState), автоматичний рендер компонентів, що забезпечує мінімальну затримку; плавне оновлення графіків; відсутність перезавантаження сторінки.

Передбачено обробку тривожних станів. Backend або Frontend перевіряє:

```
if heart_rate > 100:
alert = "Тахікардія"
```

Frontend відображає:

```
if (data.heart_rate > 100) {
alert("Увага! Високий пульс");
}
```

Таблиця 3.9 – Структура компонентів Dashboard

Компонент	Функція
ECGChart	графік ЕКГ
PressureChart	тиск
VitalCard	пульс, температура
AlertsPanel	тривоги

Для відображення динамічних кривих (ЕКГ, PPG) використано бібліотеку Recharts у поєднанні з протоколом WebSockets. Оскільки дані надходять із частотою 250 Гц, для запобігання мерехтінню інтерфейсу застосовано метод «ковзного вікна». На графіку одночасно відображається останні 5 секунд запису.

Для відображення варіабельності серцевого ритму (ВСР) протягом дня використовуються лінійні діаграми зі згладжуванням (Area Charts), що дозволяє лікарю візуально оцінити періоди найбільшого стресу пацієнта.

Для мінімізації пропуску критичних подій в інтерфейсі реалізовано систему Push-сповіщень та візуальних алертів. При виявленні панічної атаки або аритмії відповідний віджет пацієнта в Dashboard лікаря починає пульсувати червоним кольором. При натисканні на сповіщення лікар миттєво отримує доступ до ділянки ЕКГ, зафіксованої за 10 секунд до та після спрацювання алгориту «Smart Alarm», що дозволяє верифікувати подію та відрізнити артефакт від патології.

Таблиця 3.10. Основні компоненти інтерфейсу та їх функції

Компонент інтерфейсу	Технологія	Функціональне призначення
ECG Live Stream	Canvas / WebSockets	Візуалізація кривої ЕКГ у реальному часі без затримок.
Health Metric Cards	React Memo Components	Відображення числових значень пульсу, сатурації та температури.
History Explorer	TimescaleDB API / Recharts	Перегляд архівних даних та аналіз трендів за обраний період.
Alarm Notification Center	Toastify / Web Push API	Миттєве інформування про критичні стани пацієнтів.

Програмна реалізація Dashboard для лікаря базується на сучасній клієнт-серверній архітектурі з використанням FastAPI, WebSocket та React. Застосування бібліотек Socket.IO та Recharts дозволяє забезпечити відображення біосигналів у реальному часі з високою точністю та швидкістю. Це створює ефективний інструмент для моніторингу стану пацієнта та підтримки прийняття медичних рішень (додаток А).

Висновок до розділу 3.

У розділі 3 було здійснено комплексну технічну реалізацію телемедичної системи моніторингу біосигналів, а також проведено її апробацію в умовах, наближених до реальних.

У підрозділі 3.1 реалізовано серверну частину системи на базі сучасного асинхронного фреймворку, що забезпечує ефективну обробку потоків даних від IoT-пристроїв. Використання REST API та механізмів обміну даними в реальному часі дозволило організувати надійну передачу, агрегацію та збереження біомедичних показників. Інтеграція з медичними API та підтримка стандартів обміну даними забезпечують можливість подальшого розширення системи та її сумісність із зовнішніми інформаційними ресурсами.

У підрозділі 3.2 розроблено підсистему автентифікації та авторизації користувачів, що базується на сучасних підходах до забезпечення безпеки, зокрема використанні JWT та OAuth 2.0. Реалізовано рольову модель доступу (лікар, пацієнт, пристрій), що дозволяє ефективно розмежувати права користувачів та гарантує захист конфіденційної медичної інформації. Такий підхід підвищує рівень безпеки системи та забезпечує її відповідність сучасним вимогам до обробки персональних даних.

У підрозділі 3.3 розроблено інтерфейси користувача для різних категорій користувачів. Створено Dashboard для лікаря з можливістю візуалізації біосигналів у реальному часі (ЕКГ, артеріальний тиск, пульс), а також мобільний інтерфейс для пацієнта, який відображає ключові показники стану організму та надає рекомендації. Використання сучасних бібліотек візуалізації та технологій реального часу забезпечило високу інформативність, інтерактивність і зручність користування системою.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1 Організаційно-правові основи забезпечення безпеки праці

Охорона праці є важливою складовою діяльності будь-якої організації та спрямована на збереження життя, здоров’я і працездатності працівників у процесі трудової діяльності. Під час розробки та експлуатації систем автоматизації моніторингу та діагностики функціонального стану пацієнтів на основі IoT-технологій особливого значення набуває забезпечення безпечних умов праці для розробників програмного забезпечення, системних адміністраторів, операторів інформаційних систем та медичного персоналу.

Правову основу охорони праці в Україні становлять Конституція України, Кодекс законів про працю України, Закон України «Про охорону праці», Закон України «Основи законодавства України про охорону здоров’я», Закон України «Про захист персональних даних», Закон України «Про пожежну безпеку», а також державні санітарні норми, правила та інші нормативно-правові акти.

Відповідно до чинного законодавства роботодавець зобов’язаний забезпечити безпечні та нешкідливі умови праці, організувати навчання працівників з питань охорони праці, проводити інструктажі та контролювати дотримання встановлених вимог безпеки.

4.2 Характеристика об’єкта та виявлення потенційних небезпек

У межах виконання дипломної роботи, що присвячена розробці телемедичної системи моніторингу біосигналів, важливим етапом є забезпечення належного рівня охорони праці та безпеки як під час розробки програмно-апаратного комплексу, так і в процесі його експлуатації. Це обумовлено використанням електронного обладнання, комп’ютерної техніки,

мережевих технологій та можливим впливом технічних і природних факторів на людину.

Основною метою даного підрозділу є аналіз умов праці під час розробки та експлуатації телемедичної системи; виявлення потенційно небезпечних і шкідливих виробничих факторів; оцінка їх впливу на здоров'я людини; визначення заходів щодо зниження або усунення негативного впливу.

Об'єктом дослідження є робоче місце розробника програмного забезпечення та середовище функціонування IoT-системи моніторингу біосигналів.

Предметом дослідження виступають умови праці, технічні засоби та фактори, що можуть спричинити небезпеку для життя і здоров'я людини.

Робота над проєктом виконується в умовах офісного або домашнього робочого середовища з використанням персонального комп'ютера, периферійного обладнання та мережевих засобів зв'язку. Основні особливості таких умов тривала робота за комп'ютером; статичне навантаження на опорно-руховий апарат; зорове навантаження; використання електричного обладнання; робота в приміщенні з обмеженим простором.

Потенційні небезпеки, що можуть виникати під час розробки та експлуатації системи, поділяються на кілька груп (таблиця 4.1).

Таблиця 4.1 – Фізичні фактори

Небезпечний фактор	Джерело	Можливі наслідки
Електричний струм	ПК, блоки живлення, IoT-пристрої	ураження електричним струмом
Електромагнітне випромінювання	монітори, мережеве обладнання	вплив на нервову систему
Недостатнє освітлення	робоче місце	погіршення зору
Підвищений рівень шуму	вентилятори ПК	втома, зниження

Небезпечний фактор	Джерело	Можливі наслідки
		концентрації

Таблиця 4.2 – Ергономічні фактори

Фактор	Причина	Наслідки
Неправильна поза	невідповідне робоче місце	біль у спині, шиї
Тривала робота за ПК	відсутність перерв	перевтома
Невідповідність меблів	неправильна висота столу/крісла	порушення постави

Таблиця 4.3 – Психофізіологічні фактори

Фактор	Наслідки
Інформаційне перевантаження	стрес
Висока концентрація уваги	швидка втомлюваність
Монотонність роботи	зниження продуктивності

Таблиця 4.4 – Техногенні та аварійні фактори

Фактор	Причина	Наслідки
Пожежа	коротке замикання	пошкодження обладнання, загроза життю
Відмова обладнання	перевантаження системи	втрата даних
Збій мережі	нестабільне з’єднання	порушення роботи системи

Таблиця 4.5 – Небезпеки при експлуатації IoT-пристроїв

Фактор	Опис	Наслідки
Некоректна робота сенсорів	похибка вимірювань	неправильна діагностика
Перегрів пристрою	тривала робота	пошкодження
Нестабільне живлення	низька якість електромережі	відмова системи

Ідентифіковані небезпеки можуть мати різний ступінь впливу – від незначного дискомфорту до серйозної загрози здоров'ю. Найбільш критичними є ураження електричним струмом; пожежна небезпека; тривале статичне навантаження; помилки в роботі медичних сенсорів.

4.3 Дослідження ризику реалізації потенційних небезпек на об'єкті проєктування та розробка заходів щодо їх попередження

У процесі розробки та експлуатації телемедичної системи моніторингу біосигналів важливим етапом є проведення аналізу та оцінки ризиків, що дозволяє визначити ймовірність виникнення небезпечних ситуацій, ступінь їх впливу на здоров'я користувачів та функціонування системи, а також розробити ефективні заходи щодо їх мінімізації.

Основною метою аналізу ризиків є визначення рівня небезпеки ідентифікованих факторів; оцінка ймовірності їх виникнення; визначення тяжкості можливих наслідків; формування пріоритетів щодо впровадження захисних заходів.

Для оцінки ризиків використовується якісний метод, що базується на визначенні ймовірності виникнення події (P): низька (1); середня (2); висока (3) і також тяжкості наслідків (S): незначні (1); середні (2); критичні (3).

Рівень ризику визначається як добуток:

$$R = P \times S \qquad 4.1$$

де P – ймовірність виникнення події, S – тяжкість наслідків.

Таблиця 4.6 – Класифікація рівнів ризику

Значення R	Рівень ризику	Характеристика
1–2	Низький	допустимий
3–4	Середній	потребує контролю
6–9	Високий	потребує негайних заходів

Таблиця 4.7 – Оцінка ризиків для системи

№	Небезпека	Ймовірність (P)	Наслідки (S)	Ризик (R)	Рівень
1	Ураження електричним струмом	2	3	6	Високий
2	Пожежа (коротке замикання)	1	3	3	Середній
3	Перевтома оператора	3	2	6	Високий
4	Погіршення зору	3	2	6	Високий
5	Помилка сенсорів	2	3	6	Високий
6	Втрата даних	2	2	4	Середній

№	Небезпека	Ймовірність (P)	Наслідки (S)	Ризик (R)	Рівень
7	Відмова системи	2	3	6	Високий
8	Перегрів пристрою	2	2	4	Середній

До основних ризиків можна віднести електричну небезпеку, психофізіологічні навантаження, помилки в роботі сенсорів, відмову системи, пожежну небезпеку.

Електрична небезпека. Ймовірність ураження електричним струмом оцінюється як середня, оскільки використовується стандартне електрообладнання, однак можливі порушення ізоляції або неправильне підключення. Наслідки є критичними, що обумовлює високий рівень ризику.

Психофізіологічні навантаження. Тривала робота за комп’ютером призводить до перевтоми, зниження концентрації та погіршення зору. Висока ймовірність цих факторів обумовлює їх значний ризик, хоча наслідки не є критичними.

Помилки в роботі сенсорів. Неточність вимірювань або збої в роботі IoT-пристроїв можуть призвести до неправильних медичних висновків. Незважаючи на середню ймовірність, наслідки можуть бути критичними, що підвищує рівень ризику.

Відмова системи може бути спричинена програмними помилками або проблемами з мережею. Це критично для систем реального часу, оскільки може призвести до втрати контролю за станом пацієнта.

Пожежна небезпека. Ймовірність виникнення пожежі є низькою за умови дотримання правил експлуатації, проте наслідки можуть бути серйозними.

Аналіз ризиків показав, що найбільш небезпечними є електричні ризики; помилки в роботі сенсорів; відмова системи; перевтома користувача. Ці ризики потребують першочергового впровадження заходів безпеки.

Проведений аналіз та оцінка ризиків дозволили визначити ключові небезпечні фактори, що можуть впливати на безпеку праці та функціонування телемедичної системи. Отримані результати є основою для розробки комплексу організаційних і технічних заходів, спрямованих на зниження рівня ризику до допустимого рівня, що буде розглянуто в наступних підрозділах.

У процесі розробки та експлуатації телемедичної системи моніторингу біосигналів важливим аспектом є забезпечення безпеки в умовах надзвичайних ситуацій (НС). Надзвичайні ситуації можуть мати як техногенний, так і природний характер і здатні суттєво вплинути на функціонування системи, безпеку користувачів та цілісність даних.

У межах дипломного проєкту доцільно виділити такі типи надзвичайних ситуацій, як техногенні надзвичайні ситуації (пожежа в приміщенні; коротке замикання електромережі; відмова серверного обладнання; збій у роботі мережевої інфраструктури; витік або втрата даних), природні надзвичайні ситуації (відключення електроенергії; стихійні лиха (буревії, повені тощо); температурні впливи на обладнання), соціальні та кіберзагрози (несанкціонований доступ до системи; кібератаки; порушення конфіденційності медичних даних).

Таблиця 4.8 – Надзвичайні ситуації можуть призвести до таких наслідків:

Вид НС	Наслідки
Пожежа	пошкодження обладнання, загроза життю
Відключення живлення	зупинка системи
Збій мережі	втрата зв’язку з пристроями
Кібератака	витік даних, порушення роботи
Відмова серверу	недоступність сервісів

Особливо критичним є вплив надзвичайних ситуацій на безперервність моніторингу стану пацієнта, що може призвести до несвоєчасного виявлення небезпечних змін у біосигналах.

До технічних заходів безпеки можна віднести захист електроживлення, протипожежні заходи, резервування даних, забезпечення мережевої надійності, тощо.

Захист електроживлення передбачає використання джерел безперебійного живлення (UPS); застосування стабілізаторів напруги; захист від короткого замикання.

Протипожежні заходи включають використання негорючих матеріалів; встановлення систем пожежної сигналізації; наявність вогнегасників; дотримання правил експлуатації електрообладнання.

Резервування даних потребує регулярного створення резервних копій; зберігання даних у хмарних сервісах; використання RAID-масивів (за необхідності).

Забезпечення мережевої надійності передбачає дублювання каналів зв’язку; використання захищених протоколів (HTTPS); моніторинг стану мережі.

Організаційні заходи направлені на розробку планів евакуації; проведення інструктажів з техніки безпеки; обмеження доступу до серверного обладнання; контроль за дотриманням правил експлуатації.

Для запобігання кіберзагроз та несанкціонованому доступу застосовуються автентифікація користувачів (JWT, OAuth 2.0); шифрування даних; контроль доступу за ролями; використання антивірусного програмного забезпечення; регулярне оновлення програмного забезпечення.

У разі виникнення надзвичайної ситуації є певний комплекс дій, які обумовлені типом НС. Наприклад, при пожежі потрібно негайно відключити електроживлення; повідомити відповідні служби; евакуювати персонал; використати засоби пожежогасіння. При відмові системи необхідно перевести систему на резервні ресурси; повідомити адміністратора; провести діагностику

та відновлення. При втраті даних треба використати резервні копії; перевірити цілісність інформації; відновити працездатність системи.

Для телемедицини характерні підвищені вимоги до безпеки, оскільки система працює в режимі реального часу; дані мають критичне значення для здоров’я; відмова системи може мати серйозні наслідки. Тому необхідно забезпечити безперервний моніторинг; автоматичні сповіщення про збої; резервні канали передачі даних.

Забезпечення безпеки у надзвичайних ситуаціях є важливою складовою функціонування телемедичної системи моніторингу біосигналів. Реалізація комплексу технічних та організаційних заходів дозволяє знизити ризики виникнення небезпечних ситуацій, мінімізувати їх наслідки та забезпечити безперервність роботи системи. Це підвищує надійність, безпеку та ефективність використання розробленого рішення в умовах реальної експлуатації.

Висновок до розділу 4.

У розділі 4 дипломної роботи розглянуто питання забезпечення охорони праці та безпеки в надзвичайних ситуаціях при розробці та експлуатації телемедичної системи моніторингу біосигналів. Виконано постановку завдання та проведено ідентифікацію потенційних небезпек, характерних для робочого місця розробника та середовища функціонування IoT-системи. Виявлено основні групи небезпечних і шкідливих факторів, зокрема фізичні, ергономічні, психофізіологічні та техногенні, які можуть впливати на здоров’я людини та надійність функціонування системи. Здійснено аналіз та оцінку ризиків із використанням якісного підходу, що враховує ймовірність виникнення небезпечних ситуацій та тяжкість їх наслідків. Визначено найбільш критичні ризики, серед яких електрична небезпека, перевтома користувача, можливі збої в роботі сенсорів та відмова системи. Обґрунтовано необхідність впровадження відповідних заходів для їх зниження до

допустимого рівня. Розглянуто питання забезпечення безпеки в умовах надзвичайних ситуацій, зокрема техногенного, природного та кібернетичного характеру. Запропоновано комплекс технічних та організаційних заходів, спрямованих на підвищення надійності системи, збереження даних та захист користувачів. Особливу увагу приділено забезпеченню безперервності роботи телемедичної системи та швидкому відновленню її функціонування у разі виникнення аварійних ситуацій.

У цілому, проведений аналіз підтверджує, що дотримання вимог охорони праці та впровадження ефективних заходів безпеки дозволяє значно знизити рівень ризиків, забезпечити безпечні умови праці та підвищити надійність функціонування розробленої телемедичної системи. Це є важливою передумовою її практичного застосування та подальшого впровадження в медичну практику.

ЗАГАЛЬНІ ВИСНОВКИ

У дипломній роботі вирішено актуальне науково-практичне завдання розробки комп’ютерно-інтегрованої телемедичної системи моніторингу біосигналів, що забезпечує збір, передачу, обробку та візуалізацію медичних даних у режимі реального часу.

У першому розділі проведено аналіз сучасного стану телемедичних технологій в Україні та світі, визначено основні тенденції їх розвитку та перспективи впровадження. Виконано порівняльний аналіз методів зняття та передачі біометричних сигналів, що дозволило обґрунтувати вибір ефективних технічних рішень. На основі проведеного аналізу сформульовано постановку задачі та визначено основні вимоги до системи, зокрема щодо точності вимірювань, швидкодії, надійності та безпеки.

У другому розділі здійснено проєктування та моделювання системи телемедичного моніторингу. Розроблено архітектуру IoT-комплексу, що включає сенсорний рівень, рівень обробки даних та клієнтський рівень. Запропоновано алгоритми цифрової фільтрації та попередньої обробки біосигналів, які дозволяють підвищити точність вимірювань та зменшити вплив шумів. Проведено математичне моделювання процесу передачі даних у реальному часі, що підтвердило ефективність обраної архітектури та її здатність забезпечити необхідну пропускну здатність і допустимий рівень затримок.

У третьому розділі реалізовано програмну частину системи, включаючи серверний модуль, підсистему автентифікації та авторизації, а також користувацькі інтерфейси. Розроблено backend-сервер для обробки даних із сенсорів та інтеграції з медичними API. Реалізовано захищений доступ до системи на основі сучасних технологій автентифікації та авторизації. Створено інтерфейс Dashboard для лікаря з можливістю візуалізації біосигналів у реальному часі та мобільний інтерфейс для пацієнта.

У четвертому розділі розглянуто питання охорони праці та безпеки в надзвичайних ситуаціях. Проведено ідентифікацію потенційних небезпек, виконано аналіз та оцінку ризиків, а також запропоновано комплекс заходів щодо їх мінімізації. Розроблено рекомендації щодо забезпечення безпечних умов праці та стабільного функціонування системи в умовах можливих аварійних ситуацій.

У результаті виконання дипломної роботи досягнуто поставленої мети та вирішено всі поставлені задачі. Розроблена телемедична система характеризується високим рівнем функціональності, надійності та безпеки, що робить її перспективною для практичного застосування у сфері дистанційного моніторингу стану здоров’я пацієнтів.

Практична значущість роботи полягає у можливості використання запропонованих рішень для створення сучасних медичних інформаційних систем, що сприятимуть підвищенню якості медичного обслуговування, оперативності прийняття рішень та доступності медичної допомоги.

Перспективи подальших досліджень можуть бути пов’язані з розширенням функціональності системи, інтеграцією з національними медичними реєстрами, використанням технологій штучного інтелекту для аналізу біосигналів та впровадженням мобільних і хмарних рішень для масштабування системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python Software Foundation. SpeechRecognition Library for Python : електронний ресурс. – 2024. – Режим доступу: https://pypi.org/project/SpeechRecognition/
2. Закон України Про охорону праці (Відомості Верховної Ради України (ВВР), 1992, № 49, ст.668). https://zakon.rada.gov.ua
3. Методичні вказівки до виконання розділу «Охорона праці та безпека в надзвичайних ситуаціях». Харків : ХНУМГ ім. О. М. Бекетова, 2021.
4. ДСТУ 4297:2004 Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги
5. Digital Health in Ukraine 2023: Annual Report. – Київ : МОЗ України, 2023. – 78 с. – Режим доступу: https://moz.gov.ua
6. World Health Organization. Global strategy on digital health 2023–2030. – Geneva: WHO, 2023. – Режим доступу: https://www.who.int/publications/i/item/9789240020924
7. European Commission. Telemedicine and Digital Health in the EU 2024. – Brussels, 2024. – Режим доступу: https://health.ec.europa.eu
8. Smith J., Brown T. Internet of Medical Things (IoMT): Applications and Challenges // IEEE Access. – 2023. – Vol. 11. – P. 45210–45225. – Режим доступу: https://ieeexplore.ieee.org
9. Kumar P., Singh R. Real-Time Health Monitoring Using IoT Devices // Sensors. – 2024. – Vol. 24(3). – Режим доступу: https://www.mdpi.com/1424-8220/24/3/1120
10. FastAPI Documentation. – 2025. – Режим доступу: https://fastapi.tiangolo.com
11. MQTT Version 5.0 Specification. – OASIS, 2023. – Режим доступу: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
12. JSON Web Token (JWT) – RFC 7519. – IETF, 2023. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519

13. OAuth 2.0 Authorization Framework – RFC 6749. – IETF, 2023. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6749

14. React Documentation. – 2025. – Режим доступу: https://react.dev

15. Recharts Documentation. – 2025. – Режим доступу: https://recharts.org

16. Socket.IO Documentation. – 2025. – Режим доступу: https://socket.io/docs

17. Zhang Y., Liu H. ECG Signal Processing Using Digital Filters // Biomedical Signal Processing. – 2023. – Vol. 82. – Режим доступу: https://www.sciencedirect.com

18. Patel S., Wang L. Wearable Sensors for Health Monitoring // IEEE Sensors Journal. – 2024. – Vol. 24. – P. 11234–11250. – Режим доступу: https://ieeexplore.ieee.org

19. European Society of Cardiology. Guidelines for cardiovascular monitoring 2023. – Режим доступу: https://www.escardio.org

20. Міністерство охорони здоров’я України. Електронна система охорони здоров’я (eHealth). – 2024. – Режим доступу: https://ehealth.gov.ua

21. National Institute of Standards and Technology. Cybersecurity Framework 2.0. – 2024. – Режим доступу: https://www.nist.gov/cyberframework

22. Google Identity Platform. OAuth 2.0 for Web Applications. – 2025. – Режим доступу: https://developers.google.com/identity/protocols/oauth2

23. OpenAPI Specification 3.1. – 2023. – Режим доступу: https://spec.openapis.org/oas/latest.html

24. Khan A., Ali M. Secure Telemedicine Systems Using Cloud Computing // Journal of Medical Systems. – 2024. – Vol. 48(2). – Режим доступу: https://link.springer.com

ДОДАТОК А

Приклад сторінки Dashboard для лікаря як міні-проєкт (Backend + Frontend)

1. Структура проєкту

medical-dashboard/

```
│
├── backend/
│   └── main.py
│
├── frontend/
│   ├── package.json
│   └── src/
│       ├── App.js
│       ├── components/
│       │   ├── ECGChart.js
│       │   ├── VitalCard.js
│       │   └── AlertsPanel.js
│       └── index.js
```

2. BACKEND (FastAPI + WebSocket)

backend/main.py

```
from fastapi import FastAPI, WebSocket
import random
import asyncio
```

```
app = FastAPI()

@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    while True:
        data = {
            "ecg": [random.uniform(-1, 1) for _ in range(50)],
            "heart_rate": random.randint(60, 120),
            "pressure_sys": random.randint(110, 140),
            "pressure_dia": random.randint(70, 90),
            "temperature": round(random.uniform(36.0, 37.5), 1)
        }
        await websocket.send_json(data)
        await asyncio.sleep(1)
```

Запуск backend:

uvicorn main:app –reload

3. FRONTEND (React Dashboard)

frontend/package.json

```
{
  "name": "dashboard",
  "version": "1.0.0",
  "dependencies": {
    "react": "^18.0.0",
    "react-dom": "^18.0.0",
    "recharts": "^2.5.0"
  },
  "scripts": {
    "start": "react-scripts start"
  }
}
```

frontend/src/index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

frontend/src/App.js

```
import React, { useEffect, useState } from "react";
import ECGChart from "./components/ECGChart";
import VitalCard from "./components/VitalCard";
import AlertsPanel from "./components/AlertsPanel";

function App() {
  const [data, setData] = useState(null);

  useEffect(() => {
    const ws = new WebSocket("ws://localhost:8000/ws");

    ws.onmessage = (event) => {
      setData(JSON.parse(event.data));
    };

    return () => ws.close();
  }, []);

  if (!data) return <div>Loading...</div>;

  return (
    <div style={{ padding: "20px", fontFamily: "Arial" }}>
      <h2>Dashboard лікаря</h2>

      <ECGChart ecg={data.ecg} />

      <div style={{ display: "flex", gap: "20px" }}>
        <VitalCard title="Пульс" value={data.heart_rate} unit="bpm" />
        <VitalCard title="Температура" value={data.temperature} unit="°C" />
      </div>

      <div style={{ marginTop: "20px" }}>
        <p>Тиск: {data.pressure_sys}/{data.pressure_dia}</p>
      </div>
```

```
      <AlertsPanel heartRate={data.heart_rate} />
    </div>
  );
}

export default App;
```

frontend/src/components/ECGChart.js

```
import React from "react";
import { LineChart, Line, XAxis, YAxis } from "recharts";

function ECGChart({ ecg }) {
  const chartData = ecg.map((v, i) => ({
    time: i,
    value: v
  }));

  return (
    <div>
      <h3>ΕΚΓ</h3>
      <LineChart width={600} height={200} data={chartData}>
        <XAxis dataKey="time" />
        <YAxis />
        <Line type="monotone" dataKey="value" dot={false} />
      </LineChart>
    </div>
  );
}

export default ECGChart;
```

frontend/src/components/VitalCard.js

```
import React from "react";

function VitalCard({ title, value, unit }) {
  return (
    <div style={{
      border: "1px solid #ccc",
      padding: "15px",
```

```
      borderRadius: "10px",
      width: "150px"
    }}>
      <h4>{title}</h4>
      <p style={{ fontSize: "24px" }}>
        {value} {unit}
      </p>
    </div>
  );
}

export default VitalCard;
```

frontend/src/components/AlertsPanel.js

```
import React from "react";

function AlertsPanel({ heartRate }) {
  let alert = null;

  if (heartRate > 100) {
    alert = "! Тахікардія!";
  } else if (heartRate < 60) {
    alert = "! Брадикардія!";
  }

  return (
    <div style={{ marginTop: "20px" }}>
      <h3>Сповіщення</h3>
      {alert ? <p>{alert}</p> : <p>Норма</p>}
    </div>
  );
}

export default AlertsPanel;
```

Для запуску Backend:

```
cd backend

uvicorn main:app –reload
```

Для запуску Frontend:

```
cd frontend
```

npm install
npm start

Відкрити:

http://localhost:3000

ДОДАТОК В

Приклад додатку Кабінет пацієнта (мобільний вигляд)

1. Структура проєкту

```
patient-app/
│
├── backend/
│   └── main.py
│
└── frontend/
    ├── package.json
    └── src/
        ├── index.js
        ├── App.js
        ├── styles.css
        └── components/
            ├── VitalCard.js
            ├── BatteryIndicator.js
            ├── CallDoctorButton.js
            └── Recommendations.js
```

2. BACKEND (FastAPI + WebSocket)

backend/main.py

```
from fastapi import FastAPI, WebSocket
import random
import asyncio

app = FastAPI()

@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    while True:
        data = {
            "heart_rate": random.randint(60, 110),
            "spo2": random.randint(94, 100),
            "battery": random.randint(20, 100)
        }
        await websocket.send_json(data)
        await asyncio.sleep(2)
```

Запуск

```
cd backend
```

uvicorn main:app –reload

3. FRONTEND (React Mobile UI)

frontend/package.json

```
{
  "name": "patient-app",
  "version": "1.0.0",
  "dependencies": {
    "react": "^18.0.0",
    "react-dom": "^18.0.0"
  },
  "scripts": {
    "start": "react-scripts start"
  }
}
```

frontend/src/index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import "./styles.css";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

frontend/src/styles.css

```
body {
  margin: 0;
  font-family: Arial;
  background: #f5f7fa;
}

.container {
  max-width: 400px;
  margin: auto;
  padding: 15px;
}

.card {
  background: white;
```

```
  border-radius: 15px;
  padding: 15px;
  margin-bottom: 15px;
  text-align: center;
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
}

.button {
  background: #007bff;
  color: white;
  border: none;
  padding: 15px;
  width: 100%;
  border-radius: 10px;
  font-size: 16px;
}

.alert {
  background: #ffe6e6;
  color: red;
}
```

frontend/src/App.js

```
import React, { useEffect, useState } from "react";
import VitalCard from "./components/VitalCard";
import BatteryIndicator from "./components/BatteryIndicator";
import CallDoctorButton from "./components/CallDoctorButton";
import Recommendations from "./components/Recommendations";

function App() {
  const [data, setData] = useState({
    heart_rate: 0,
    spo2: 0,
    battery: 0
  });

  useEffect(() => {
    const ws = new WebSocket("ws://localhost:8000/ws");

    ws.onmessage = (event) => {
      setData(JSON.parse(event.data));
    };

    return () => ws.close();
```

```
  }, []);

  return (
    <div className="container">
      <h2>Кабінет пацієнта</h2>

      <VitalCard title="ЧСС" value={data.heart_rate} unit="bpm" />
      <VitalCard title="SpO₂" value={data.spo2} unit="%" />

      <BatteryIndicator level={data.battery} />

      <CallDoctorButton />

      <Recommendations heartRate={data.heart_rate} />
    </div>
  );
}

export default App;
```

frontend/src/components/VitalCard.js

```
import React from "react";

function VitalCard({ title, value, unit }) {
  return (
    <div className="card">
      <h3>{title}</h3>
      <p style={{ fontSize: "28px" }}>
        {value} {unit}
      </p>
    </div>
  );
}

export default VitalCard;
```

frontend/src/components/BatteryIndicator.js

```
import React from "react";

function BatteryIndicator({ level }) {
  let color = "green";
  if (level < 30) color = "red";
```

```
  return (
    <div className="card">
      <h3>Заряд пристрою</h3>
      <div style={{
        height: "10px",
        background: "#ddd",
        borderRadius: "5px"
      }}>
        <div style={{
          width: level + "%",
          height: "100%",
          background: color
        }} />
      </div>
      <p>{level}%</p>
    </div>
  );
}

export default BatteryIndicator;
```

frontend/src/components/CallDoctorButton.js

```
import React from "react";

function CallDoctorButton() {
  const handleClick = () => {
    alert("Виклик лікаря...");
  };

  return (
    <button className="button" onClick={handleClick}>
      Зв’язатися з лікарем
    </button>
  );
}

export default CallDoctorButton;
```

frontend/src/components/Recommendations.js

```
import React from "react";

function Recommendations({ heartRate }) {
  let text = "Стан у нормі";
```

```
  if (heartRate > 100) {
    text = "Рекомендація: виконайте дихальні вправи";
  }

  return (
    <div className="card">
      <h3>Рекомендації</h3>
      <p>{text}</p>
    </div>
  );
}

export default Recommendations;
```

Для запуску Backend:

```
cd backend
uvicorn main:app –reload
```

Для запуску Frontend:

```
cd frontend
npm install
npm start
```

Відкрити:

http://localhost:3000

ДОДАТОК С

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА
імені О. М. БЕКЕТОВА

Навчально-науковий інститут енергетичної, інформаційної та транспортної інфраструктури

Кафедра автоматизації та комп'ютерно-інтегрованих технологій

Автоматизація моніторингу та діагностики функціонального стану пацієнтів на основі IoT-технологій

здобувач вищої освіти групи СІНЖ 2023-1-у
Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Шалагін Дмитро Максимович

Керівник Піддубна Л.В., к.філос.н., доцент, доцент кафедри АКІТ

МЕТА

Мета бакалаврської роботи: розробка комп'ютерно-інтегрованої системи для автоматизованого збору та візуалізації медичних показників пацієнта в режимі реального часу.

Реалізація мети бакалаврської роботи потребує вирішення таких завдань:

- Проаналізувати сучасні IoT-рішення для телемедицини.
- Спроєктувати архітектуру системи «сенсор-хмара-інтерфейс».
- Реалізувати алгоритм автоматичного сповіщення про критичні стани.
- Розробити веб-інтерфейс для моніторингу даних лікарем.

2

Об’єкт і предмет дослідження

Об’єкт дослідження: Процеси оперативного збору, накопичення та аналізу часових рядів біометричних параметрів пацієнта (частота серцевих скорочень, рівень кисню у крові, артеріальний тиск) у телемедичних мережах.

Предмет дослідження: Комп’ютерно-інтегровані засоби автоматизації (мікроконтролери, сенсори, хмарні сервіси), протоколи бездротового зв’язку (MQTT, HTTP) та алгоритми підтримки прийняття рішень для ідентифікації критичних відхилень у функціональному стані людини.

3

Методи дослідження

У процесі виконання дипломної роботи було використано такі методи дослідження, як теорія автоматичного керування та системний аналіз – для побудови загальної структури системи; методи цифрової обробки сигналів – для фільтрації шумів та артефактів із сирих даних сенсорів; методи баз даних та SQL-моделювання – для організації структури збереження інформації; методи імітаційного моделювання – для перевірки стійкості системи при високих навантаженнях на канали зв’язку; експериментальні методи – для оцінки похибки вимірювань сенсорів у порівнянні з еталонними медичними приладами.

4

АКТУАЛЬНІСТЬ

Цифрова трансформація медицини вимагає переходу від епізодичного обстеження до безперервного моніторингу стану пацієнта. В умовах поствоєнного відновлення України та високої щільності населення в містах-мільйонниках, таких як Харків, навантаження на амбулаторії зростає. Використання IoT-технологій дозволяє знизити ризик раптових ускладнень (інфаркти, інсульти) за рахунок ранньої діагностики аномалій. Створення вітчизняних доступних систем автоматизації моніторингу є критично важливим для сталого розвитку системи охорони здоров'я.

5

Системи віддаленого моніторингу пацієнтів Remote Patient Monitoring (RPM)

Сучасні RPM-рішення, що впроваджуються в Україні, класифікуються за трьома рівнями.

- Сенсорний рівень представлений пристроями, які носяться (smart-watches, патчі), та медичними гаджетами з Bluetooth-інтерфейсом (глюкометри, тонометри, спірометри).
- Транспортний рівень утворюють мобільні додатки та IoT-шлюзи, що агрегують дані та передають їх у хмарне середовище за протоколами MQTT або HTTPS.
- Аналітичний рівень представлено серверним програмним забезпеченням, що використовує алгоритми обробки часових рядів для виявлення аномалій, наприклад, тахікардії або критичного зниження сатурації.

Перспективи розвитку RPM в Україні пов'язані з інтеграцією цих систем у єдиний державний реєстр eHealth з метою автоматичного оновлення електронної медичної картки пацієнта даними з його персональних пристроїв моніторингу.

6

Архітектура телемедичної системи

7

Порівняння біометричних датчиків

Параметр¤	ЕКГ¤	Пульсоксиметр¤	Температура¤
Тип·сигналу¤	Електричний¤	Оптичний¤	Тепловий¤
Точність¤	Висока¤	Середня¤	Середня¤
Інвазивність¤	Ні¤	Ні¤	Ні¤
Складність¤	Висока¤	Низька¤	Низька¤
Застосування¤	Кардіологія¤	Контроль·стану¤	Загальний·моніторинг¤

8

Основні протоколи передачі даних

Характеристика	BLE	Wi-Fi	LoRaWAN	MQTT (над Wi-Fi/GSM)
Енергоспоживання	Дуже низьке	Високе	Мінімальне	Середнє
Дальність дії	10–50 м	30–100 м	2–15 км	Глобальна (через IP)
Пропускна здатність	Низька (до 2 Мбіт/с)	Висока (від 54 Мбіт/с)	Дуже низька (до 50 кбіт/с)	Середня (оптимізована)
Сценарій використання	Фітнес-трекери, патчі	Стаціонарні монітори в лікарні	Віддалені села, зони надзвичайних ситуацій	Передача даних у хмару
Надійність доставки	Середня	Висока	Низька (без підтверджень)	Висока (QoS рівні)

9

Архітектура IoT-комплексу, призначеного для автоматизованого збору, обробки та передачі даних у реальному часі

10

Основні параметри моніторингу

№	Параметр	Опис	Значення
1	Частота серцевих скорочень (HR)	Робота серця	60–100 уд/хв
2	ЕКГ	Електрична активність серця	Аналіз ритму
3	SpO_2	Насичення крові киснем	95–100%
4	Температура тіла	Стан організму	36–37°C
5	Артеріальний тиск	Робота судин	120/80 мм рт. ст.

11

Функціональні вимоги до системи моніторингу

№	Вимога	Опис
1	Збір даних	Зчитування з сенсорів
2	Передача	Безперервна передача
3	Зберігання	Хмарна база
4	Аналіз	Виявлення відхилень
5	Сповіщення	Повідомлення лікаря

Нефункціональні вимоги до системи моніторингу

Нефункціональні вимоги до системи для віддаленого моніторингу стану пацієнтів стосуються її надійності; енергоефективності; безпеки даних; масштабованості; доступності 24/7.

12

Технічні вимоги до системи віддаленого моніторингу стану пацієнтів

№	Вимога	Значення
1	Частота дискретизації ЕКГ	не менше 250 Гц для коректного обчислення ВСР
2	Час автономної роботи	не менше 24 годин у режимі активного моніторингу
3	Затримка передачі критичного сигналу (Latency)	не більше 2 сек
4	Тип доступу	кросплатформний веб-інтерфейс (React) із рольовою моделлю доступу (Пацієнт / Лікар / Адміністратор)

13

Архітектура IoT-системи базується на чотирирівневій моделі

- На рівні датчиків (сенсорів) здійснюється первинний збір даних з фізичного середовища. Використовуються різні типи датчиків залежно від задачі, наприклад, датчики вологості; тиску; вібрації; струму та напруги; температурні датчики. Датчики формують аналогові або цифрові сигнали, що відображають поточний стан об'єкта контролю.
- Мікроконтролер виконує функції збору, попередньої обробки та передачі даних. Основними задачами на рівні мікроконтролера є опитування датчиків; фільтрація та нормалізація сигналів; перетворення аналогових сигналів у цифрові; формування пакетів даних; передача інформації через мережеві протоколи (Wi-Fi, Ethernet, LoRa, GSM). На цьому рівні можуть реалізовуватися алгоритми локальної логіки (edge computing), наприклад, виявлення аварійних станів; пороговий контроль параметрів; буферизація даних при втраті зв'язку.
- Хмарна інфраструктура забезпечує централізовану обробку, зберігання та аналіз даних. Основними функціями хмарного серверу є прийом даних від пристроїв через API або брокери повідомлень, наприклад, MQTT; збереження даних у базі; обробка потоків у реальному часі; аналітика та агрегація; формування звітів.
- Клієнтський додаток забезпечує візуалізацію даних у вигляді графіків і таблиць; моніторинг у реальному часі; отримання сповіщень про аварії; керування обладнанням (за необхідності).

14

Основні компоненти IoT-системи

Рівень системи	Компонент	Приклад	Основні характеристики	Призначення
Датчики	Датчик температури та вологості	DHT22	Температура: -40…+80°C, вологість: 0 – 100 %	Контроль мікроклімату
Датчики	Датчик температури	DS18B20	Цифровий, точність ±0.5°C	Вимірювання температури
Датчики	Датчик тиску	BMP280	Вимірювання атмосферного тиску	Моніторинг середовища
Датчики	Датчик струму	ACS712	Вимірювання змінного/постійного струму	Контроль електроспоживання
Мікроконтролер	Контролер	ESP32	Wi-Fi, Bluetooth, 240 МГц, ADC	Обробка та передача даних
Мікроконтролер	Альтернатива	Arduino Uno	Простота використання	Базові IoT-рішення
Комунікація	Протокол передачі	MQTT	Легкий, publish/subscribe	Передача даних у реальному часі
Комунікація	Протокол передачі	HTTP/HTTPS	Універсальний	Взаємодія з API
Комунікація	Бездротовий зв'язок	Wi-Fi	Висока швидкість передачі	Локальні мережі
Комунікація	Бездротовий зв'язок	LoRa	Велика дальність, низька швидкість	Віддалені об'єкти
Сервер	Хмарна платформа	AWS IoT / Azure IoT	Масштабованість, аналітика	Обробка та зберігання
Сервер	База даних	InfluxDB	Оптимізація під time-series	Зберігання даних
Сервер	База даних	TimescaleDB	SQL + time-series	Аналітика даних
Клієнт	Веб-додаток	React / Angular	Інтерактивний інтерфейс	Візуалізація даних
Клієнт	Мобільний додаток	Flutter	Кросплатформеність	Доступ користувача

15

Приклад структури API

Метод	Endpoint	Опис
POST	/api/data	Надсилання даних із сенсора
GET	/api/data	Отримання даних
GET	/api/data/{id}	Дані конкретного пристрою
GET	/api/stats	Агреговані показники

16

Концепція ролей у системі

Роль	Опис	Права доступу
Лікар	Медичний персонал	Перегляд усіх даних, аналітика
Пацієнт	Користувач системи	Перегляд власних даних
Пристрій	IoT-сенсор (ESP32)	Надсилання даних

17

Процес використання OAuth 2.0 у системі моніторингу

Сценарій 1: Лікар отримує доступ до даних пацієнта, який реалізується за наступними кроками: лікар авторизується через OAuth; отримує access_token; переглядає біосигнали пацієнта через API.

Сценарій 2: Інтеграція з медичною системою втілюється наступним чином: система отримує доступ до EHR; зберігає історію пацієнта; синхронізує дані з IoT-сенсорами.

Сценарій 3: Мобільний додаток працює наступним чином: користувач входить через Google або медичний сервіс; додаток отримує токен; відображає дані з серверу.

Перевагами використання OAuth 2.0 є відсутність передачі паролів; централізована авторизація; можливість інтеграції з зовнішніми сервісами; гнучке управління правами доступу (scopes); підвищений рівень безпеки.

18

Ролі в системі

Компонент	Роль
Google	Сервер авторизації
FastAPI	Сервер ресурсів
Користувач	Лікар / пацієнт
Клієнт	Веб/мобільний додаток

19

Основні вимоги до інтерфейсу

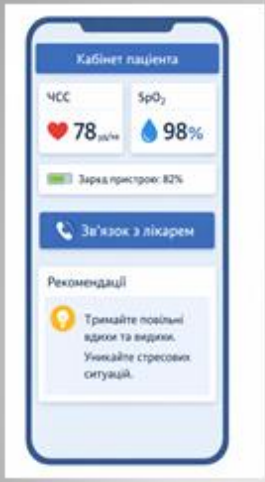
Вимога	Опис
Інформативність	відображення ключових параметрів
Реальний час	оновлення без затримок
Зрозумілість	прості графіки та індикатори
Надійність	стабільна робота при потоці даних
Адаптивність	підтримка різних пристроїв

Типи візуалізації медичних даних

Тип даних	Спосіб відображення
ЕКГ	лінійний графік
Артеріальний тиск	часовий графік
Пульс	числовий індикатор + графік
Температура	трендова лінія

20

Кабінет пацієнта (Мобільний вигляд)



Мобільний інтерфейс пацієнта реалізовано з використанням адаптивного підходу. Додаток забезпечує відображення основних клінічних показників (ЧСС, SpO_2), індикатора стану IoT-пристрою та рекомендацій у разі відхилень. Передача даних здійснюється у реальному часі через WebSocket-з'єднання з сервером на базі FastAPI, що дозволяє забезпечити оперативне інформування пацієнта та підвищити ефективність моніторингу стану здоров'я.

21

Структура Dashboard лікаря

Основні компоненти Dashboard лікаря

Компонент	Технологія	Функція
IoT-пристрій	ESP32	збір біосигналів
Backend	FastAPI	обробка даних
Реальний час	Socket.IO	передача даних
Frontend	React	інтерфейс лікаря
Візуалізація	Recharts	графіки

22

Структура компонентів Dashboard

Компонент	Функція
ECGChart	графік ЕКГ
PressureChart	тиск
VitalCard	пульс, температура
AlertsPanel	тривоги

Основні компоненти інтерфейсу та їх функції

Компонент інтерфейсу	Технологія	Функціональне призначення
ECG Live Stream	Canvas / WebSockets	Візуалізація кривої ЕКГ у реальному часі без затримок.
Health Metric Cards	React Memo Components	Відображення числових значень пульсу, сатурації та температури.
History Explorer	TimescaleDB API / Recharts	Перегляд архівних даних та аналіз трендів за обраний період.
Alarm Notification Center	Toastify / Web Push API	Миттєве інформування про критичні стани пацієнтів.

23

ЗАГАЛЬНІ ВИСНОВКИ

У першому розділі проведено аналіз сучасного стану телемедичних технологій в Україні та світі, визначено основні тенденції їх розвитку та перспективи впровадження. Виконано порівняльний аналіз методів зняття та передачі біометричних сигналів, що дозволило обґрунтувати вибір ефективних технічних рішень. На основі проведеного аналізу сформульовано постановку задачі та визначено основні вимоги до системи, зокрема щодо точності вимірювань, швидкодії, надійності та безпеки.

У другому розділі здійснено проектування та моделювання системи телемедичного моніторингу. Розроблено архітектуру IoT-комплексу, що включає сенсорний рівень, рівень обробки даних та клієнтський рівень. Запропоновано алгоритми цифрової фільтрації та попередньої обробки біосигналів, які дозволяють підвищити точність вимірювань та зменшити вплив шумів.

У третьому розділі реалізовано програмну частину системи, включаючи серверний модуль, підсистему автентифікації та авторизації, а також користувацькі інтерфейси. Створено інтерфейс Dashboard для лікаря з можливістю візуалізації біосигналів у реальному часі та мобільний інтерфейс для пацієнта.

У четвертому розділі розглянуто питання охорони праці та безпеки в надзвичайних ситуаціях. Проведено ідентифікацію потенційних небезпек, виконано аналіз та оцінку ризиків, а також запропоновано комплекс заходів щодо їх мінімізації. Розроблено рекомендації щодо забезпечення безпечних умов праці та стабільного функціонування системи в умовах можливих аварійних ситуацій.

24

Дякую за увагу!