ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему «Інформаційна система контролю стану здоров'я мовою C++»

на здобуття освітнього ступеня бакалавра

зі спеціальності 126 Інформаційні системи та технології

освітньо-професійної програми Інформаційні системи та технології

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

_____ _____
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД-41
Данііл ДОБРОВ
Ім'я, ПРІЗВИЩЕ

Керівник:
науковий ступінь, вчене звання
Валентина ДАНИЛЬЧЕНКО, PhD
Ім'я, ПРІЗВИЩЕ

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

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК

«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Добров Данііл Дмитрович

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Інформаційна система контролю стану здоров'я мовою C++

керівник кваліфікаційної роботи Валентина ДАНИЛЬЧЕНКО, PhD,

(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» 02 2025р. № 56

2. Строк подання кваліфікаційної роботи «30» 05 2025 р.

3. Вихідні дані до кваліфікаційної роботи: офіційна документація C++, науково-технічна література, дані про існуючі системи моніторингу здоров'я, результати українських та закордонних наукових досліджень у сфері медичного моніторингу.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз ефективності існуючих рішень медичного моніторингу
2. Постановка технічного завдання, вибір технологічного стеку, проєктування архітектури, структури, користувацького інтерфейсу системи
3. Розробка системи моніторингу стану здоров'я мовою C++

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «24» 02 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	01.03.2025 р.	
2	Дослідження технічних аспектів та ефективності існуючих рішень медичного моніторингу	15.03.2025 р.	
3	Вибір технологічного стеку, постановка технічного завдання на розробку	17.03.2025 р.	
4	Проєктування структури, архітектури та користувацького інтерфейсу	4.04.2025 р.	
5	Програмна реалізація та тестування роботи системи	16.05.2025 р.	
6	Розробка демонстраційних матеріалів	26.05.2025 р.	
7	Попередній захист дипломної роботи	27.05.2025 р.	
8	Подання роботи в деканат	30.05.2025 р.	

Здобувач(ка) вищої освіти _____ Даніїл ДОБРОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 61 стор., 38 рис., 1 табл., 22 джерела.

Мета роботи – розробити та впровадити мікропроцесорну інформаційну систему моніторингу стану здоров'я з використанням бездротової передачі даних, хмарної платформи обробки інформації та базових сенсорів життєво важливих показників людини.

Об'єкт дослідження – процеси віддаленого медичного моніторингу фізіологічного стану людини на основі вбудованих систем і сенсорних модулів.

Предмет дослідження – апаратні та програмні засоби реалізації систем моніторингу здоров'я, включаючи алгоритми зчитування, обробки та передачі даних із використанням мікроконтролерів, бездротових протоколів та хмарних сервісів.

Короткий зміст роботи: У дипломній роботі проведено огляд сучасного ринку інформаційних систем для моніторингу стану здоров'я, зокрема мобільних додатків та стаціонарних платформ. Визначено переваги й недоліки наявних рішень, що обумовило необхідність розробки власної системи. Сформульовано технічне завдання, обґрунтовано вибір апаратного та програмного забезпечення (ESP8266, DS18B20, Arduino IDE, ThingSpeak, HTTP/HTTPS). Розроблено архітектуру, структурну схему та інтерфейс системи. Детально описано алгоритми роботи, реалізацію програмного забезпечення мовою C++, ручне тестування функціональності та візуалізацію даних у хмарному середовищі. Визначено ефективність створеної системи та окреслено напрями її подальшого розвитку, зокрема інтеграцію з мобільними додатками та розширення кількості сенсорів.

КЛЮЧОВІ СЛОВА: СИСТЕМА МОНІТОРИНГУ, МІКРОКОНТРОЛЕР, ESP8266, DS18B20, ПУЛЬС-СЕНСОР, THINGSPEAK, ХМАРНА ПЛАТФОРМА, C++, ІНФОРМАЦІЙНА СИСТЕМА, МЕДИЧНИЙ МОНІТОРИНГ.

ABSTRACT

Textual part of the qualification work for obtaining a bachelor's degree: 61 pages, 38 figures, 1 table, 22 sources.

The purpose of the work is to develop and implement a microprocessor information system for health monitoring using wireless data transmission, a cloud information processing platform and basic sensors of human vital signs.

The object of research is the processes of remote medical monitoring of the physiological state of a person based on built-in systems and sensor modules.

The subject of the study is hardware and software for the implementation of health monitoring systems, including algorithms for reading, processing and transmitting data using microcontrollers, wireless protocols and cloud services.

Summary of the work: In the thesis, an overview of the modern market of information systems for health monitoring, in particular mobile applications and stationary platforms, is carried out. The advantages and disadvantages of existing solutions are identified, which necessitated the development of their own system. The terms of reference are formulated, the choice of hardware and software (ESP8266, DS18B20, Arduino IDE, ThingSpeak, HTTP/HTTPS is substantiated). The architecture, structural diagram and interface are developed System. The algorithms of work, the implementation of software in the C++ language, manual functionality testing, and data visualization in the cloud environment are described in detail. The effectiveness of the created system is determined and the directions of its further development are outlined, in particular, integration with mobile applications and expansion of the number of sensors.

KEYWORDS: MONITORING SYSTEM, MICROCONTROLLER, ESP8266, DS18B20, PULSE SENSOR, THINGSPEAK, CLOUD PLATFORM, C++, INFORMATION SYSTEM, MEDICAL MONITORING.

ЗМІСТ

ВСТУП

У сучасних умовах, коли хронічні захворювання, серцево-судинні проблеми та вікові зміни здоров’я охоплюють дедалі ширші верстви населення, постає гостра потреба у впровадженні ефективних інформаційних систем моніторингу стану здоров’я. Згідно з прогнозами, глобальний ринок систем моніторингу здоров’я зростатиме в середньому на 7,5% на рік, досягнувши 99,49 млрд дол. США у 2033 році. Віддалений контроль життєвих показників стає ключовим елементом у телемедицині та цифровому догляді за пацієнтами, особливо в умовах стрімкого старіння населення та зростання потреби в індивідуалізованому підході до лікування.

Особливу актуальність мають мініатюрні та енергоефективні IoT-рішення, що забезпечують безперервне збирання, передавання та візуалізацію фізіологічних показників у режимі реального часу. У цьому контексті інформаційні системи, побудовані на базі мікроконтролерів з бездротовими інтерфейсами, виступають ефективними інструментами для інтеграції з хмарними платформами, аналітичними модулями та вебінтерфейсами.

Метою дипломної роботи є розробка та дослідження інформаційної системи для моніторингу стану здоров’я людини, реалізованої на базі мікроконтролера ESP8266 із програмним забезпеченням мовою C++. Система має забезпечити вимірювання фізіологічних параметрів (температури тіла та частоти пульсу), їхнє збереження, передавання та відображення у зручному цифровому вигляді.

Завдання дослідження:

- Проаналізувати сучасний стан та тенденції розвитку систем моніторингу здоров’я.
- Оцінити переваги та недоліки існуючих рішень на основі мобільних додатків та стаціонарних систем.
- Сформулювати технічне завдання для розробки системи.
- Обґрунтувати вибір апаратної та програмної частин проєкту.

- Реалізувати систему вимірювання температури та пульсу з передаванням даних до хмарного сервісу.
- Провести тестування та оцінку ефективності розробленого рішення.

Об'єкт дослідження – інформаційна система моніторингу стану здоров'я.

Предмет дослідження – методи зчитування, обробки, передавання та візуалізації фізіологічних даних за допомогою вбудованих пристроїв та хмарних сервісів.

Методи дослідження – аналіз наукової літератури та ринку, моделювання апаратної частини, експериментальна реалізація прототипу, тестування функціональності, інтерпретація результатів та оцінка ефективності.

Основні результати та підходи, реалізовані у дипломній роботі, були представлені на VII Міжнародній науково-технічній конференції «Сучасний стан та перспективи розвитку IoT» у вигляді тез «Інтелектуальні інформаційні системи для моніторингу, управління та підтримки користувачів».

Наукова новизна полягає у реалізації оптимізованого інформаційного рішення з використанням простих апаратних засобів для високоточних та безперервних вимірювань із можливістю розширення до мультисенсорної системи, що підвищує його придатність до подальшого застосування в телемедицині.

Практична значущість роботи полягає у створенні робочого прототипу системи, яка може бути використана для персонального контролю здоров'я, а також як основа для медичних досліджень та розвитку інтелектуальних рішень у галузі біомедичних технологій.

РОЗДІЛ 1. АНАЛІЗ ЕФЕКТИВНОСТІ ТА ВИВЧЕННЯ ПРИНЦИПУ РОБОТИ ІСНУЮЧИХ РІШЕНЬ МЕДИЧНОГО МОНІТОРИНГУ

1.1 Принципи роботи та класифікація існуючих інформаційних систем моніторингу

Інформаційні системи моніторингу стану здоров'я являють собою складні програмно-апаратні комплекси, які інтегрують різноманітні сенсори (біометричні, фізіологічні, хімічні), вузли збирання даних (edge-комп’ютинг, мікроконтролери), IoT-шлюзи (зазвичай Raspberry Pi або Arduino), локальні сервери та хмарні сервіси для обробки, зберігання та аналізу. Дані, отримані від носимих чи стаціонарних сенсорів, можуть включати температуру, серцевий ритм, рівень насичення киснем, рівень глюкози тощо. Передавання даних здійснюється по бездротових протоколах (Wi-Fi, BLE, Zigbee) на шлюз, що конвертує дані у стандартизовані формати та передає їх на локальний чи хмарний сервер. Це дозволяє лікарям і медичному персоналу отримувати вичерпну інформацію про стан здоров’я пацієнтів у режимі реального часу, інтегрувати дані з іншими медичними системами та формувати рекомендації щодо лікування. Усе це робить інформаційні системи моніторингу стану здоров'я надзвичайно важливими інструментами для сучасної медицини.

Основний принцип роботи інформаційних систем моніторингу стану здоров'я полягає у взаємодії кількох ключових компонентів: сенсорів (біометричних, фізіологічних, хімічних), вузлів збирання даних (edge-комп’ютинг, мікроконтролери з підключенням до мережі), шлюзів IoT (наприклад, Raspberry Pi, Arduino із встановленими брокерами MQTT або протоколами HTTPS), локального сервера з розподіленими базами даних та хмарних середовищ з потужною обчислювальною інфраструктурою для обробки великих обсягів даних. Як зображено на рис. 1.1, носимі сенсори збирають дані про фізіологічний стан людини (температура, пульс, ЕКГ, насиченість киснем, рівень глюкози тощо). Ці дані передаються на вузли безпеки (safe node) та здоров'я (health node), де виконуються первинна обробка, фільтрація та попереднє машинне навчання. Після

цього дані передаються через шлюз IoT (наприклад, з використанням протоколів MQTT, CoAP або HTTP) на локальний сервер та/або у хмарне середовище для розширеної аналітики, прогнозування та візуалізації даних у вигляді графіків та таблиць. Це дозволяє віддалений моніторинг стану пацієнта, виявлення критичних відхилень та своєчасне реагування медичного персоналу [1].

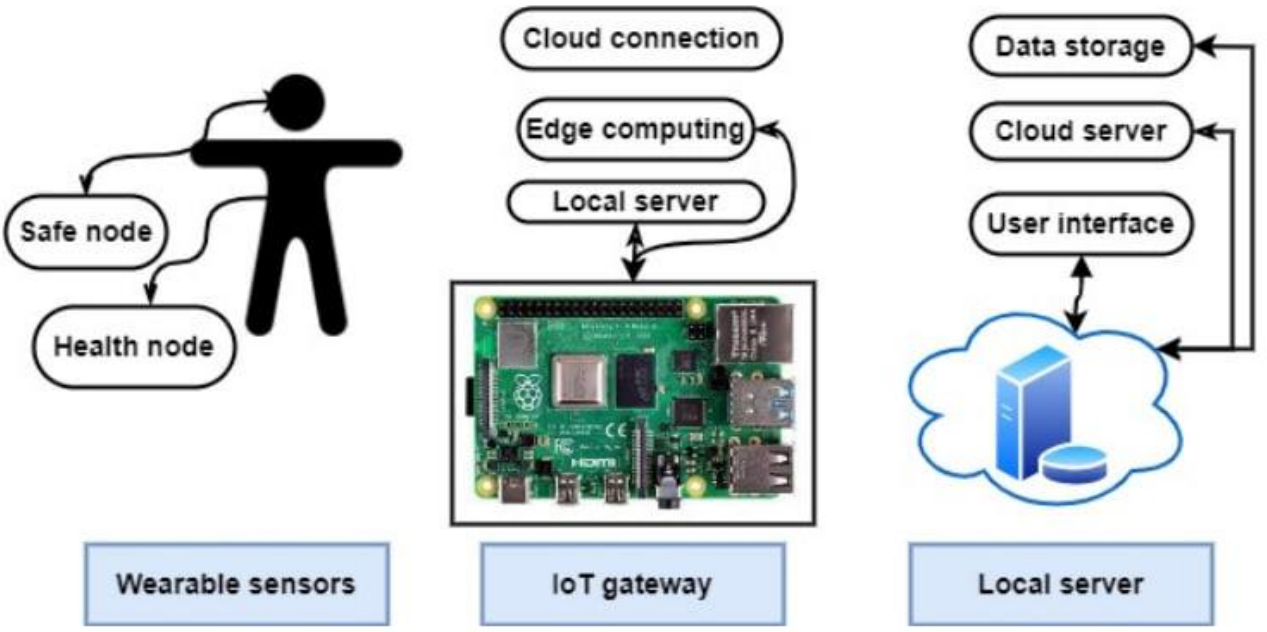


Рис. 1.1 Типова архітектура комплексної системи моніторингу медичних показників [1]

Існують також нетипові архітектури таких систем, зокрема гібридні рішення, де обчислювальні завдання виконуються одночасно на edge-пристроях і у хмарі, а також децентралізовані peer-to-peer системи для обміну даними між кількома вузлами без центрального сервера. Такі підходи дозволяють зменшити затримку передачі даних та підвищити відмовостійкість системи.

Класифікація інформаційних систем моніторингу стану здоров'я ґрунтується на низці технічних критеріїв, які охоплюють тип сенсорних компонентів, функціональні характеристики системи та її архітектурну організацію. З технічної точки зору сенсори поділяються на контактні (наприклад, електрокардіографічні (ЕКГ), електроміографічні (ЕМГ), температурні сенсори, біоімпедансметри), які вимагають прямого контакту з тілом пацієнта, та безконтактні (інфрачервоні

термометри, фотоплетизмографи, радарні сенсори), що здійснюють вимірювання за допомогою відбитого або випромінюваного випромінювання. Детальна ієрархія класифікації сенсорів для моніторингу стану здоров'я представлена на рис. 1.2.

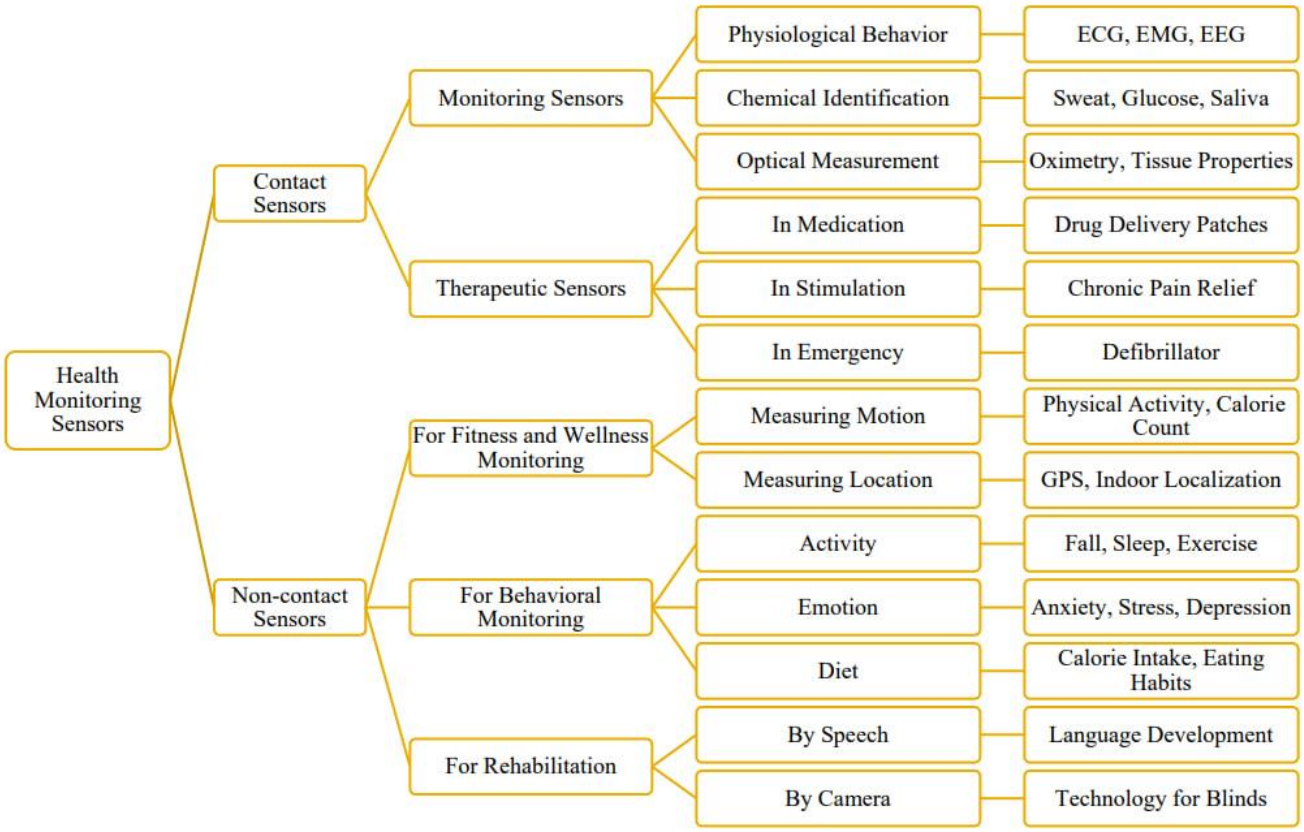


Рис. 1.2 Дерево класифікації сенсорів для моніторингу стану здоров'я [1]

Як видно з діаграми, сенсори можуть бути розподілені на моніторингові та терапевтичні, а також орієнтовані на фітнес, поведінкове спостереження або реабілітацію. В межах цих груп сенсори класифікуються відповідно до їх функціонального призначення: для вимірювання фізіологічних параметрів (наприклад, ЕКГ, ЕЕГ), для хімічної ідентифікації (піт, слина, глюкоза), для оцінки рухової активності, емоційного стану, дієти, мовлення тощо. Такий підхід забезпечує цілісне охоплення різних аспектів фізичного й психоемоційного здоров'я пацієнтів [1].

Додаткову класифікацію сенсорів можна побачити на рис. 1.3, який демонструє приклади конкретних типів сенсорів, що застосовуються в медичних інформаційних системах.

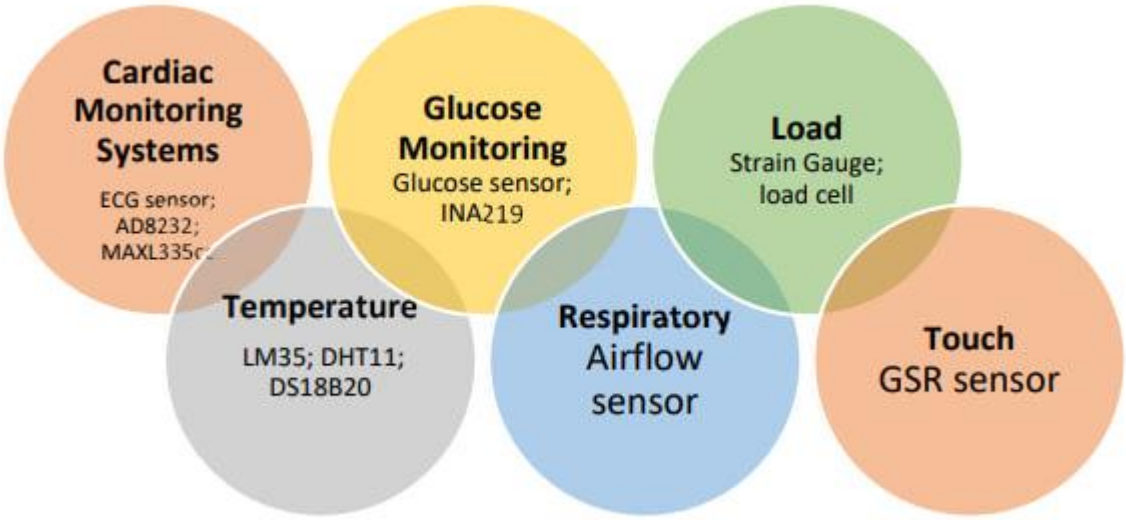


Рис. 1.3 Типи сенсорів, що використовуються в інформаційних системах моніторингу здоров’я [1]

Наприклад, для кардіомоніторингу використовуються сенсори типу AD8232 або MAXL335C, для вимірювання температури — термодатчики LM35, DHT11, DS18B20, для контролю рівня глюкози — сенсор INA219. Також у практиці застосовуються сенсори дотику (GSR), сенсори повітряного потоку (respiratory airflow), тензодатчики навантаження (strain gauge, load cell). Ці сенсорні компоненти відіграють ключову роль у побудові багатофункціональних систем моніторингу, забезпечуючи збір критично важливих біомедичних показників з високою точністю та надійністю.

У сучасних інформаційних системах моніторингу стану здоров’я спостерігається тенденція до ускладнення архітектурних рішень за рахунок впровадження багатоетапних моделей обробки даних. Як зображено на рис. 1.4, типова інфраструктура включає кілька рівнів обчислювальних вузлів: периферійні (edge nodes), туманні (fog nodes) та хмарні (cloud data centers).

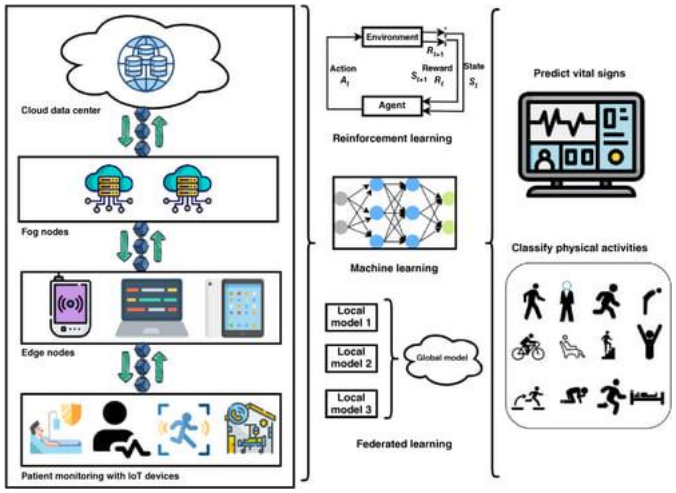


Рис. 1.4 Архітектура багаторівневої системи моніторингу здоров'я з використанням edge/fog/cloud технологій та методів штучного інтелекту [1]

Дані, зібрані з IoT-пристроїв у пацієнта, первинно обробляються на edge-рівні для зменшення затримок і об'єму переданої інформації, далі передаються до fog-вузлів для більш складної агрегації та, при необхідності, до хмари для довготривалого зберігання, візуалізації та глибинного аналізу.

Особливу увагу привертає застосування інтелектуальних технологій, таких як навчання з підкріпленням (reinforcement learning) для динамічного прийняття рішень, класичне машинне навчання (machine learning) для класифікації активностей користувача, а також федеративне навчання (federated learning), що дозволяє тренувати глобальні моделі без прямого обміну персональними даними між локальними пристроями. Цей підхід забезпечує як високий рівень персоналізації, так і дотримання принципів безпеки та конфіденційності медичної інформації [1-4].

1.2 Огляд сучасного ринку систем моніторингу стану здоров'я

Сучасний ринок систем моніторингу стану здоров'я демонструє стрімке зростання завдяки впровадженню цифрових технологій, таких як носимі пристрої, телемедицина, системи штучного інтелекту та інтелектуального аналізу даних. За оцінками аналітичної компанії DataM Intelligence, у 2024 році глобальний ринок оцінювався у 51,26 мільярда доларів США, а до 2033 року прогнозується його

зростання до 99,49 мільярда доларів США зі середньорічним темпом зростання (CAGR) 7,5% (рис. 1.5).

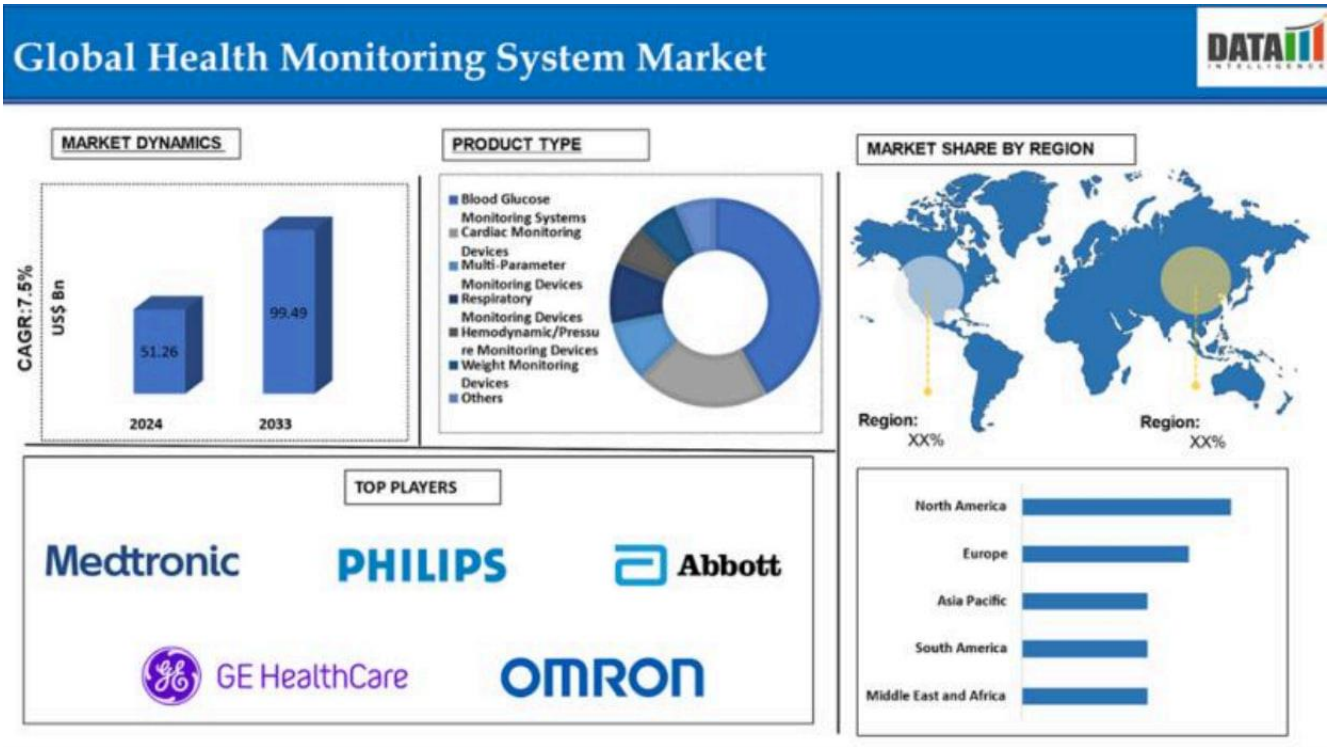


Рис. 1.5 Структура та динаміка глобального ринку систем моніторингу стану здоров'я

Особливу динаміку демонструють такі категорії продуктів, як системи моніторингу рівня глюкози, кардіомоніторингові пристрої, багато параметричні системи моніторингу, сенсори дихання та тиску, а також системи контролю ваги. Ці продукти активно впроваджуються як у клінічних умовах, так і в побуті, що є важливою ознакою децентралізації медичних послуг.

Згідно з останніми даними, найбільшу частку ринку займають Північна Америка та Європа, тоді як Азіатсько-Тихоокеанський регіон демонструє найвищі темпи зростання, що зумовлено як зростанням захворюваності на хронічні хвороби, так і активною цифровізацією медицини в країнах регіону.

На ринку домінують такі гіганти, як Medtronic, Philips, Abbott, GE Healthcare, Omron, які забезпечують розробку, виробництво та підтримку широкого спектра моніторингових пристроїв для професійного та домашнього використання. Додатковим драйвером зростання є впровадження рішень на основі штучного

інтелекту, наприклад, як у випадку запуску інструменту Star Health Face Scan в Індії у 2024 році, що забезпечує оцінку 18 життєво важливих показників протягом однієї хвилини [5].

У 2025 році особливої популярності набули мобільні застосунки для моніторингу стану здоров’я, які користувачі завантажують мільйонами.

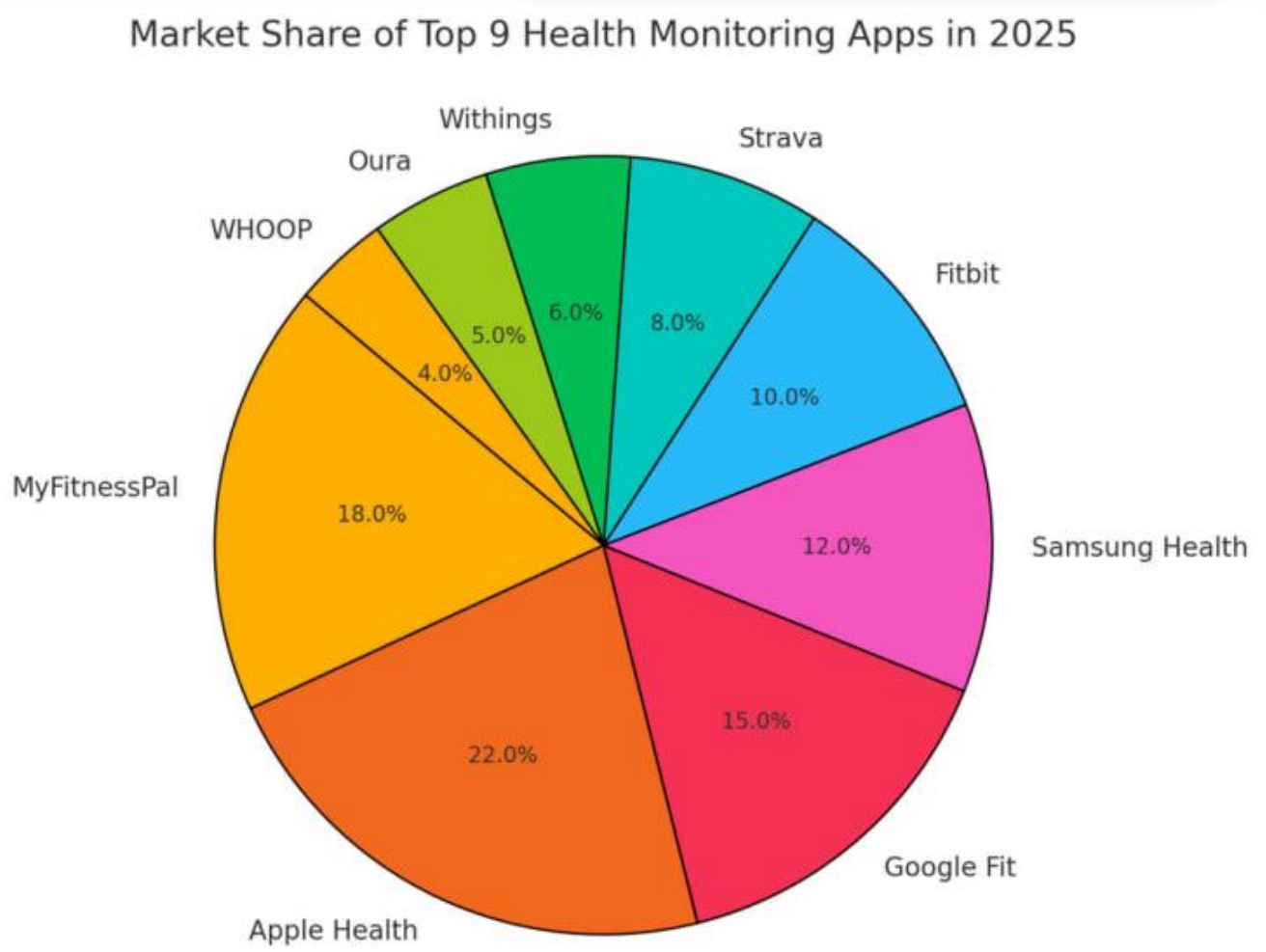


Рис. 1.6 – Популярні рішення мобільного моніторингу здоров’я у 2025 році

Серед них варто виокремити BetterMe Health Coaching, що пропонує понад 3000 варіантів тренувань, індивідуальні плани харчування та моніторинг фізичних і психологічних показників. Додаток Whoop визнано найбільш повнофункціональним серед wellness-трекерів завдяки глибокому аналізу сну, навантаження та відновлення організму. Також популярністю користується Healow, який дозволяє інтегрувати дані з різних медичних джерел і синхронізуватися з клінічними системами.

Ці додатки характеризуються широкими можливостями персоналізації, AI-аналітикою, хмарною синхронізацією та орієнтацією на профілактику й контроль хронічних станів. Завдяки зрозумілому інтерфейсу, багатофункціональності та точності моніторингу, вони стали частиною цифрової культури здоров'я у багатьох країнах світу [6].

BetterMe Health Coaching — це цифрова платформа для покращення фізичного та психоемоційного стану, яка поєднує функціональність фітнес-тренера, дієтолога та ментора зі здорового способу життя в одному мобільному застосунку. Завдяки алгоритмам персоналізації, вбудованим засобам аналітики та модульній структурі, додаток став одним з найуспішніших wellness-рішень у світі, з понад 150 мільйонами завантажень та щомісячним доходом понад $2 мільйони [7].

BetterMe побудований як модульна система з гнучким API-інтерфейсом, який забезпечує зв'язок між фронтендом (мобільним застосунком) та бекенд-сервісами. Серед основних компонентів:

- Модуль фітнесу: включає бібліотеку з понад 3000 відеоуроків та інтерактивних програм тренувань, адаптованих під рівень користувача (від новачків до просунутих). Алгоритми добору базуються на відповідях користувача у вхідному опитуванні та щоденних трекінгах.
- Модуль харчування: дозволяє створювати індивідуальні плани харчування на основі обраної дієти (вегетаріанська, кето, середземноморська тощо) та відслідковувати макроелементи за допомогою вбудованої бази продуктів.
- Модуль ментального здоров'я: містить аудіосесії з медитаціями, дихальними вправами, коучинговими порадами, що сприяють зменшенню стресу та розвитку корисних звичок.
- Трекери життєвих показників: для води, харчування, сну, менструального циклу, ваги, активності (кроків), інтервального голодування тощо.

BetterMe використовує адаптивні рекомендаційні механізми на основі моделей машинного навчання. Кожен користувач проходить початкове опитування, що включає стать, вік, вагу, рівень активності, цілі (наприклад, зниження ваги, поліпшення сну, нарощення м'язів). Зібрані дані формують персоніфіковану

щоденну програму тренувань, харчування та відпочинку. Крім того, користувач отримує push-нагадування та візуальний прогрес за досягненнями [8].

Інтерфейс додатку мінімалістичний і структурований за принципом "все в один дотик": головна панель містить дашборд з рекомендаціями на день, календар активностей, кнопки швидкого доступу до трекерів, а також магазин з можливістю замовити тренажери та одяг. На рис. 1.7 представлений дизайн деяких екранів додатку.

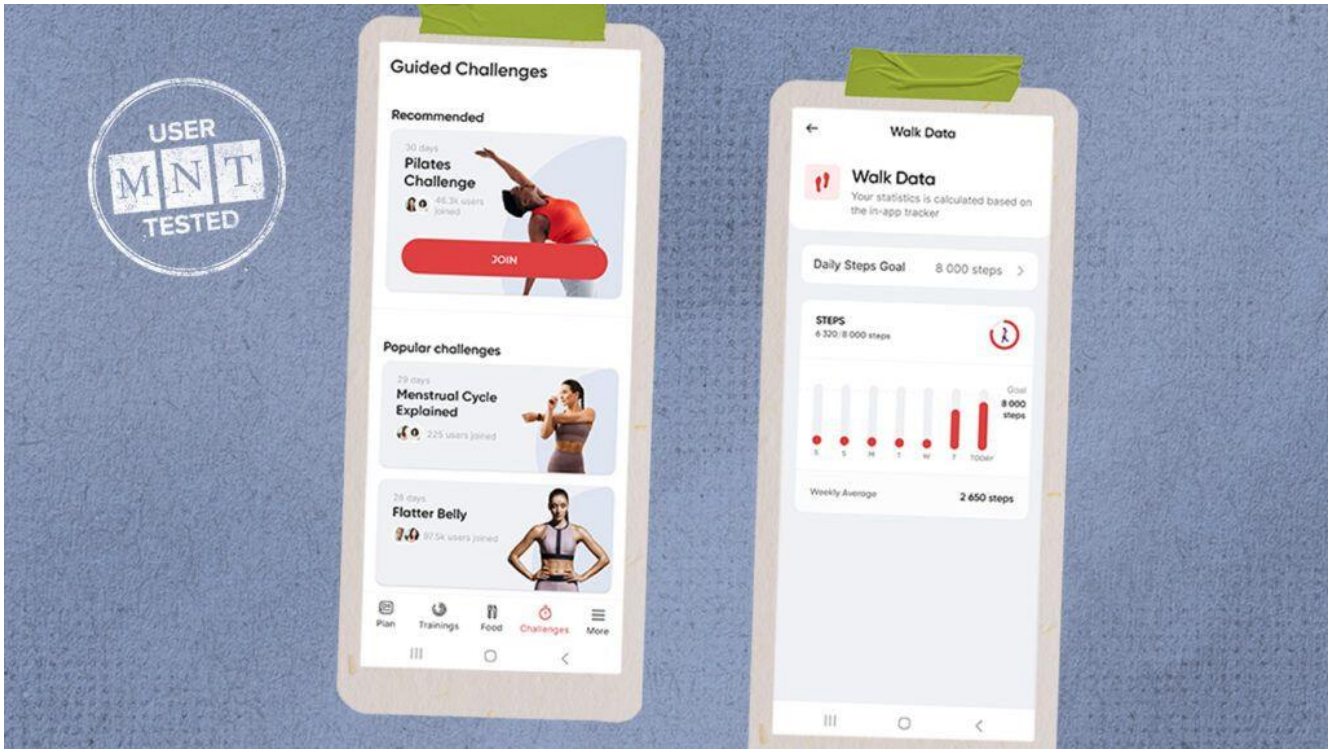


Рис. 1.7 Користувацький інтерфейс BetterMe [8]

Серед основних переваг BetterMe виділяють gростоту використання та інтуїтивно зрозумілий дизайн, високий рівень персоналізації контенту, доступність без додаткового обладнання (лише смартфон), підтримку користувачів з особливими потребами (вправи для людей на візках, з травмами тощо), можливість вибору інтерфейсної мови та кросплатформеність (iOS, Android).

Серед недоліків користувачі відзначають наявність прихованих платежів за доступ до повного функціоналу, стереотипність візуальних моделей тіла, обмежені можливості соціальної взаємодії між користувачами [8].

BetterMe Health Coaching є потужним і доступним інструментом для самостійного оздоровлення, який успішно поєднує тренди цифрової медицини, гейміфікації та персоніфікованої підтримки.

WHOOP — це інноваційна система безекранного фітнес-моніторингу, розроблена для безперервного збору біометричних даних, аналізу відновлення, сну та навантаження з метою оптимізації фізичної активності та запобігання перевтомі. Вона складається з компактного носимого пристрою WHOOP 4.0 та мобільного застосунку з розширеним функціоналом. На відміну від традиційних фітнес-трекерів, WHOOP не має екрану та максимально орієнтований на аналітику, а не моментальну зворотну взаємодію [9].

WHOOP 4.0 — це пристрій розміром меншим за попередню версію (на 33%), із водозахистом за стандартом IP68, можливістю заряджання під час носіння та автономністю до 6 днів. У конструкцію вбудовано:

- 5 світлодіодів та 4 фотодіоди для оптичного вимірювання пульсу (з частотою 100 вимірів на секунду);
- акселерометр для виявлення рухової активності;
- модуль Bluetooth для передавання даних у мобільний застосунок.

Всі показники передаються в реальному часі у застосунок, що дозволяє будувати точну аналітику життєвих параметрів користувача: пульс, варіабельність серцевого ритму (HRV), фази сну, рівень напруги (Strain) та відновлення (Recovery).

Застосунок WHOOP реалізовано для платформ iOS, Android та у вигляді веб-інтерфейсу. Основні розділи [9]:

- Home – дашборд з візуалізацією поточного рівня Strain, Recovery і Sleep, а також прогнозованими рекомендаціями на день;
- Health – розділ із деталізованими біометричними індикаторами, включаючи серцевий ритм, HRV, фазу сну, а також монітор стресу, тиск (на тарифі Life), монітор здоров’я та інші параметри [10];
- Community – система команд та спільнот для колективного аналізу показників;

- WHOOP Live – можливість накладання біометричних даних у реальному часі на відео або зображення користувача.

WHOOP також дозволяє відображати довготривалі тренди показників, використовує щоденні рекомендації щодо фізичної активності, персоналізовані цілі та "рекомендовану напругу дня" залежно від відновлення організму (рис. 1.8). Завдяки цій логіці, система здатна попереджати про ризик перетренованості чи недостатнього відпочинку.

Рис. 1.8 Користувацький інтерфейс Whoop [10]

До переваг WHOOP належать максимально точне оптичне вимірювання завдяки щільному контакту сенсора зі шкірою, безперервне відстеження HRV як ключового індикатора відновлення, заряджання без зняття з тіла (через портативну батарею), глибокий аналіз сну з фазами та рекомендаціями щодо часу засинання, професійна аналітика для спортсменів, медиків та тренерів.

Серед обмежень системи виділяють відсутність екрану та моментального інтерфейсу, відсутність GPS і традиційних функцій фітнес-трекера (кількість кроків, повідомлення тощо), висока вартість підписки (від $27/міс), не повністю локалізований інтерфейс для неангломовних регіонів [10].

Загалом, WHOOP позиціонується не як "гаджет", а як біоаналітична система для поглибленого самоспостереження й оптимізації навантажень на основі науково обґрунтованих показників.

Разом із тим, ринок систем медичного моніторингу не обмежується лише мобільними застосунками чи носимими пристроями.

Значну частку ринку займають десктопні або веборієнтовані програмні рішення, які інтегруються безпосередньо в інфраструктуру медичних установ. Такі системи орієнтовані на клінічне використання: вони підтримують підключення до медичного обладнання (наприклад, ЕКГ-моніторів, пульсоксиметрів, апаратів для вимірювання тиску), забезпечують збереження результатів у структурованій базі даних, надають інструменти для тривалого відстеження стану пацієнта та інтегруються з електронною медичною документацією (EHR).

Ключовими характеристиками подібних платформ є: висока масштабованість, можливість створення облікових записів для медичного персоналу з різним рівнем доступу, централізований контроль, а також аналітичні модулі для виявлення ризиків та автоматичного формування клінічних звітів. Прикладами є такі системи, як Philips IntelliVue Guardian, GE CARESCAPE та інші спеціалізовані рішення, що відповідають вимогам стандартів безпеки та сумісності HL7, FHIR, DICOM.

Philips IntelliVue Guardian — це високотехнологічна система медичного моніторингу, яка призначена для виявлення ранніх ознак клінічного погіршення у пацієнтів на загальних стаціонарах та у відділеннях невідкладної допомоги. Система забезпечує автоматизований збір життєвих показників, обчислення індексу раннього попередження (Early Warning Score, EWS), та надсилає попередження клінічному персоналу для забезпечення своєчасного реагування [11].

Philips IntelliVue Guardian реалізована як модульне рішення, яке може інтегруватися в існуючу інфраструктуру лікарні. Основні компоненти включають:

- Монітори серії IntelliVue MP5SC — пристрої для спотового та безперервного моніторингу життєвих показників, які мають вбудовану функцію обчислення EWS;
- IntelliVue Cableless Measurements — бездротові сенсори для вимірювання частоти серцевих скорочень, сатурації, дихання, що дозволяють зберегти мобільність пацієнтів;
- IntelliVue Guardian Software — центральне програмне ядро, яке приймає дані від моніторів, обробляє їх, обчислює індекси та передає повідомлення через IntelliSpace Event Management;
- Інтеграція з електронною медичною документацією (EHR) — через модуль IntelliBridge Enterprise.

Рис. 1.9 демонструє сучасну архітектуру мобільної медичної екосистеми з інтеграцією персональних сенсорів, мобільного застосунку, хмарних PHR/EMR серверів та взаємодією з медичними установами — як можливе майбутнє доповнення до IntelliVue Guardian у межах концепції mHealth.

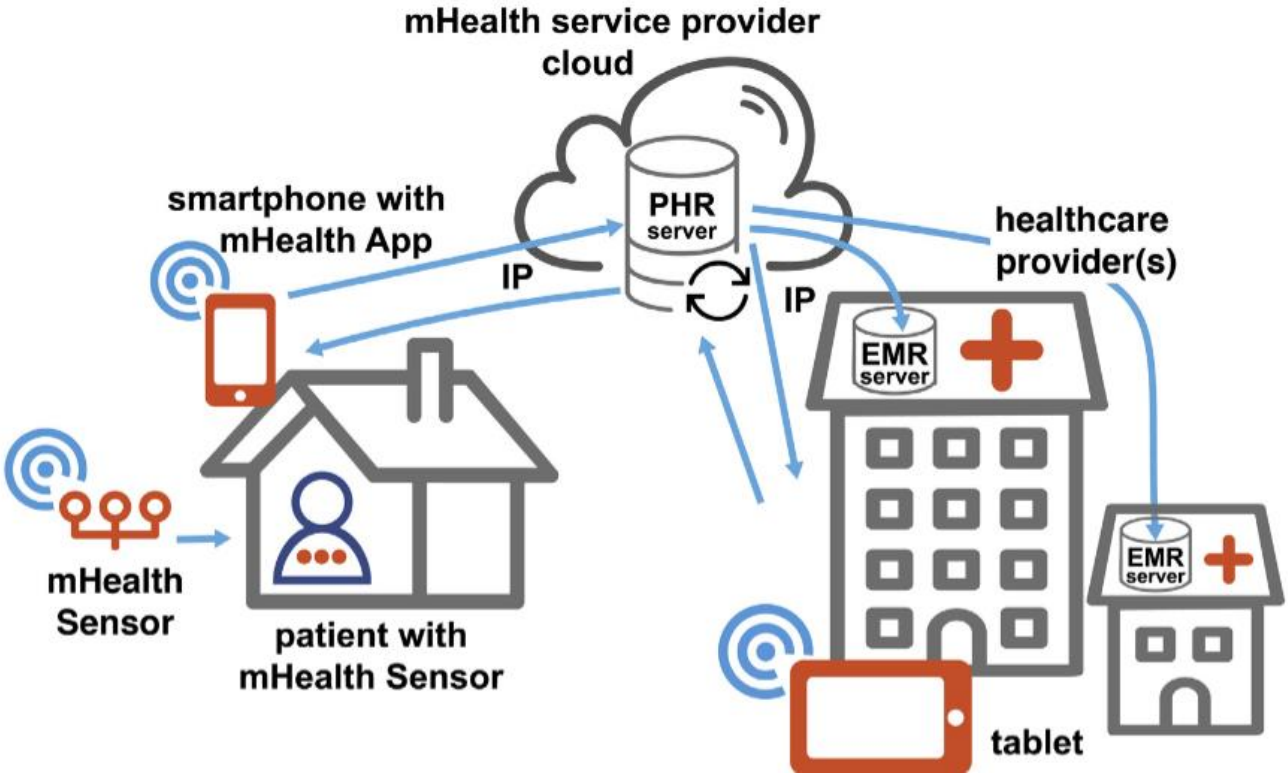


Рис. 1.9 – Концептуальна схема Philips IntelliVue Guardian [12]

Система працює відповідно до лікарняних протоколів спостереження. Наприклад, при реєстрації EWS=2 — не вживається дій; при EWS=4 — активується повідомлення медсестрі та частіше відстеження показників; при EWS=5 — ініціюється виклик групи швидкого реагування (Rapid Response Team), без необхідності залишати ліжко пацієнта [11]. Усі дані автоматично записуються в систему, а персонал отримує сповіщення через смартфон, планшет чи пейджер.

На рис. 1.10 представлено послідовність взаємодії медичного персоналу з моніторинговою системою IntelliVue Guardian на основі обчисленого EWS — від реєстрації показників до клінічного реагування. Кожен етап позначено відповідним числом, що вказує на рівень тривожності.

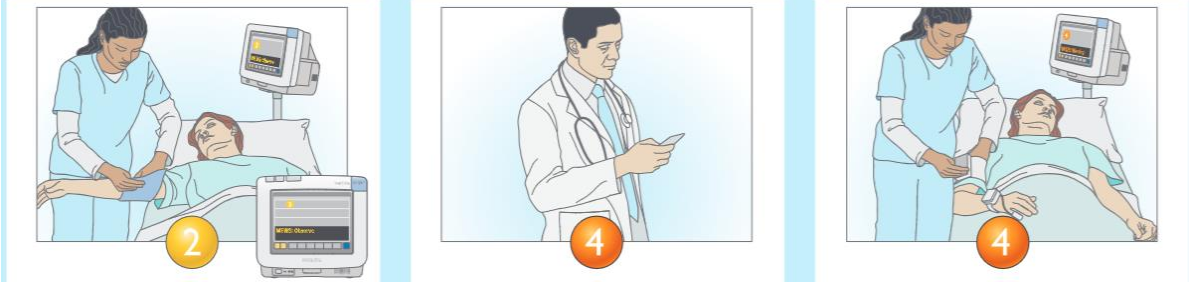


Рис. 1.10 Послідовність роботи із системою [11]

Рис. 1.11 ілюструє різні сценарії прийняття клінічних рішень відповідно до динаміки життєвих показників та рівня попередження, а також розподіл відповідальності між медсестрою, лікарем і командою швидкого реагування.

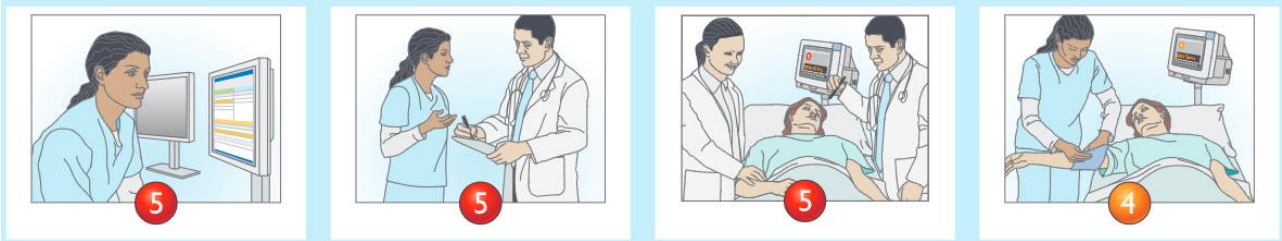


Рис. 1.11 Послідовність роботи із системою [11]

Особливістю IntelliVue Guardian є можливість конфігурування логіки реагування та інтервалів вимірювання, що дозволяє адаптувати систему до конкретного медичного закладу. Згідно з дослідженнями, застосування автоматизованих систем спостереження типу IntelliVue сприяє зниженню кількості несподіваних переведень в реанімацію та смертності серед пацієнтів загальних відділень [12].

Технічні особливості та переваги

- Висока точність завдяки багаторазовим перевіркам даних та об'єднанню джерел вимірювання;
- Повна автоматизація: збір, аналіз і передача даних без ручного втручання;
- Підтримка мобільності пацієнтів через бездротові сенсори;
- Можливість масштабування та гнучкої конфігурації (від окремих палат до цілих лікарень);
- Відповідність міжнародним стандартам безпеки та сумісності (HL7, DICOM, FHIR);
- Покращення комунікації між медперсоналом завдяки централізованому повідомленню через Event Management.

На рис. 1.12 показано апаратний інтерфейс монітора IntelliVue MP5SC, включаючи ключові параметри, які відображаються на екрані в реальному часі:

пульс, SpO_2, артеріальний тиск, частота дихання та повідомлення про необхідність втручання.

Рис. 1.12 – Апаратний інтерфейс монітора IntelliVue MP5SC [11]

Philips IntelliVue Guardian є надійною платформою для впровадження доказово обґрунтованих практик раннього втручання та стандартизації догляду в умовах лікарні [11, 12].

GE CARESCAPE — це високопродуктивна модульна система клінічного моніторингу, розроблена для використання в широкому спектрі лікарняних умов — від операційних до відділень інтенсивної терапії. Зокрема, модель CARESCAPE B650 поєднує точність медичних вимірювань, розширену модульність та зручну інтеграцію з інформаційними системами охорони здоров’я [13].

Система GE CARESCAPE складається з моніторів, багатофункціональних модулів і інтерфейсів, що дозволяють інтегруватися з іншими пристроями. Серед ключових компонентів:

- 15-дюймовий сенсорний дисплей з адаптивним інтерфейсом і можливістю відображення до 14 графіків;
- CARESCAPE Patient Data Module — портативний модуль збору даних із пам’яттю на 24 години та підтримкою до 36 подій аритмії, 10 ST-сегментів та 20 серцевих розрахунків;
- EK-Pro алгоритм — аналіз аритмій у режимі реального часу з підтримкою 4 відведень;

- Entropy та NMT модулі — для моніторингу глибини анестезії та нейром’язової блокади;
- Доступ до MUSE — система інтегрується з кардіологічною інформаційною платформою MUSE для двостороннього обміну даними ЕКГ [13].

На відміну від багатьох конкурентів, GE CARESCAPE дозволяє медичному персоналу адаптувати профілі моніторингу відповідно до клінічного сценарію: анестезіологія, інтенсивна терапія, післяопераційний догляд, неонатологія тощо. Це забезпечується через "Pages & Profiles" — функціональність швидкого перемикання сценаріїв.

Інтерфейс монітора CARESCAPE B650 є інтуїтивно зрозумілим і високоінформативним. Він підтримує гнучке налаштування відображення кривих і параметрів, дозволяючи медичному персоналу одночасно спостерігати за до 14 фізіологічними сигналами на одному екрані. Основне меню забезпечує швидкий доступ до історії пацієнта, налаштувань сигналів тривоги, візуалізації трендів і параметрів підключених модулів. Значна увага приділена ергономіці та адаптації до конкретного клінічного профілю, що значно зменшує час на перенавчання персоналу. На рис. 1.13 представлено типове відображення інтерфейсу CARESCAPE з розміткою ділянок керування, тривожних індикаторів і параметрів життєдіяльності пацієнта.

Рис. 1.13 Інтерфейс монітора GE CARESCAPE B650 з візуалізацією кривих ЕКГ, дихання, SpO_2, артеріального тиску та панеллю керування параметрами [14]

Клінічні можливості та технічні особливості GE CARESCAPE охоплюють широкий спектр медичних напрямів. У кардіології система забезпечує підтримку 12SL аналізу електрокардіограми з високою діагностичною точністю, а також безперервний моніторинг інтервалів QT/QTc. У сфері нейрофізіології CARESCAPE використовує модулі BIS, EEG та AEP для комплексного контролю за активністю головного мозку. Для дихальної підтримки система пропонує респіраторні модулі, які дозволяють моніторити газообмін, параметри спірометрії та забезпечують повний контроль роботи вентилятора. У напрямку гемодинаміки CARESCAPE використовує модулі E-COP, PiCCO та SvO_2 для оцінювання серцевого викиду, рівня насичення крові киснем та артеріального тиску. В акушерстві та неонатології реалізовано точне налаштування сигналів тривоги відповідно до віку та стану пацієнта, а також зменшення кількості хибних спрацьовувань завдяки використанню EK-Pro версії 13 та алгоритму IntelliRate [14].

GE CARESCAPE підтримує підключення до електронної медичної документації, систем PACS, а також інших пристроїв за допомогою Unity Network ID. Система розроблена з урахуванням масштабування — її можна адаптувати до потреб як невеликих відділень, так і великих лікарень. Крім того, можливість віддаленого обслуговування зменшує витрати на технічну підтримку.

CARESCAPE — це гнучке, конфігуроване рішення з глибокою спеціалізацією, яке дозволяє реалізувати принцип персоналізованого моніторингу та забезпечити безперервний доступ до клінічно значущої інформації в режимі реального часу [13, 14].

Крім основних функціональних переваг, GE CARESCAPE вирізняється високим рівнем безпеки завдяки сегментованій архітектурі мережі. Комунікація між моніторами пацієнтів та іншими медичними пристроями здійснюється через спеціалізовані VLAN-мережі: одна для критичних даних у реальному часі (MC VLAN) і одна для другорядного обміну (IX VLAN). Це дозволяє фізично розмежувати трафік медичних пристроїв від загальнолікарняного, підвищуючи надійність і захист від кібератак [15].

Система також підтримує інтеграцію з рішеннями кібербезпеки, такими як платформа Armis, яка дозволяє пасивно моніторити всі пристрої CARESCAPE в мережі, включаючи відстеження їх розташування, типу, моделі, FDA-класифікації та поведінкових аномалій (рис. 1.14).

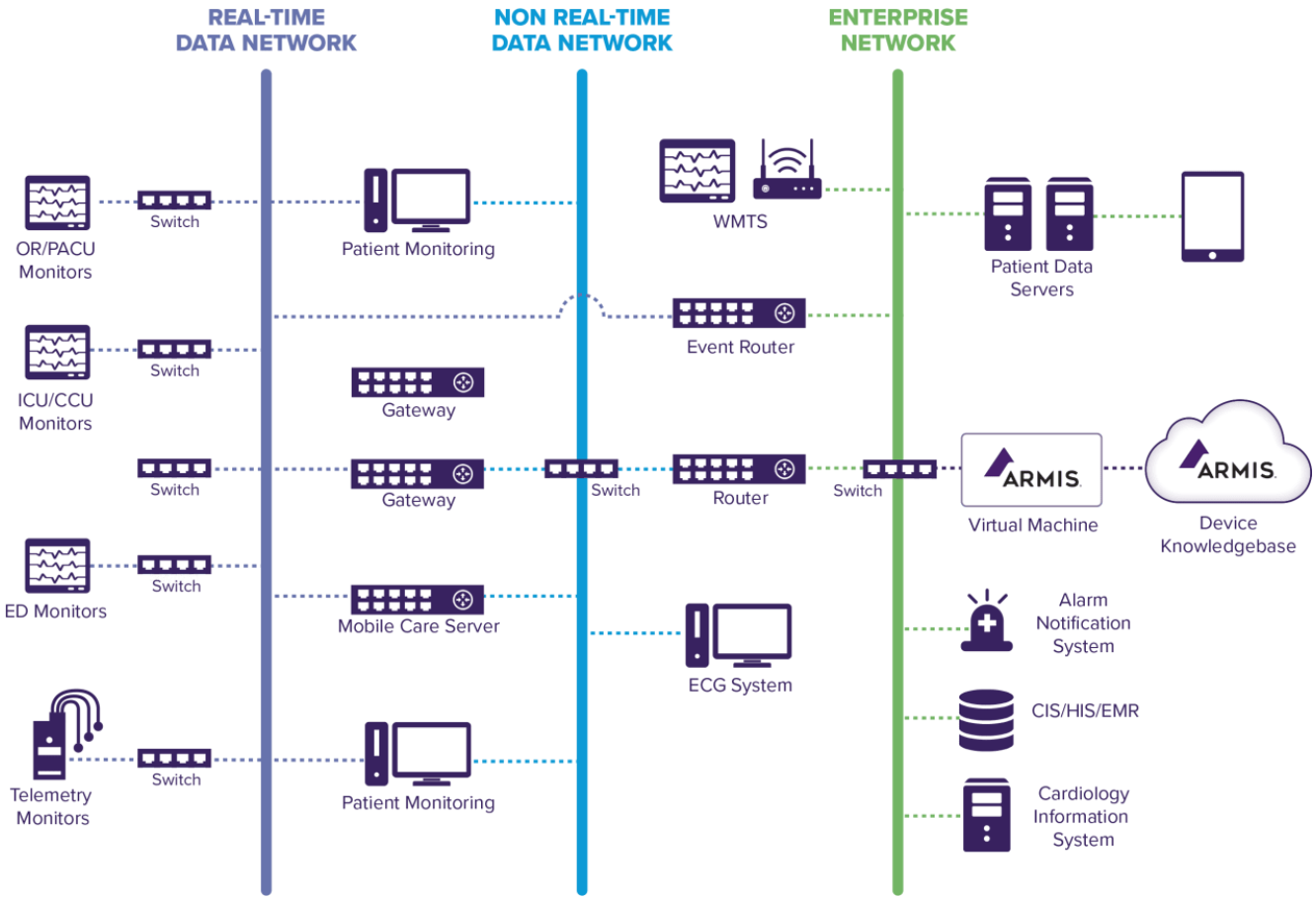


Рис. 1.14 Мережна архітектура системи GE CARESCAPE з розподілом на підмережі реального часу, нерегламентованого трафіку та корпоративного сегменту з інтеграцією кібербезпекового рішення Armis [15]

Armis надає IT-відділам повну видимість активів без потреби встановлення агентів або запуску активного сканування, що критично важливо для безперервної роботи медичного обладнання [15].

GE CARESCAPE є не лише функціонально гнучкою системою клінічного моніторингу, а й передовим прикладом безпечної цифрової екосистеми, адаптованої до умов сучасної лікарні.

Ринок систем моніторингу стану здоров’я перебуває на етапі стрімкого розвитку та диверсифікації, охоплюючи як мобільні застосунки для

індивідуального користування, так і складні програмно-апаратні комплекси для клінічного застосування. Спостерігається тенденція до інтеграції рішень із хмарними платформами, електронними медичними картками, системами штучного інтелекту та засобами кібербезпеки. У результаті це створює нову парадигму цифрової медицини, в якій моніторинг стану здоров'я стає безперервним, персоналізованим і мультиплатформенним, забезпечуючи не лише своєчасне виявлення критичних змін у стані пацієнта, а й активну участь користувачів у власному оздоровленні. Ця еволюція систем охорони здоров'я закладає фундамент для нових підходів у клінічній практиці, профілактиці хронічних захворювань і загальному підвищенні якості медичних послуг.

1.3 Аналіз переваг і недоліків існуючих рішень

Сучасні системи моніторингу стану здоров'я демонструють значне різноманіття як у функціональному, так і в архітектурному плані. Вони охоплюють як персоналізовані мобільні застосунки (типу BetterMe Health Coaching, WHOOP), так і масштабовані клінічні рішення (Philips IntelliVue Guardian, GE CARESCAPE), кожне з яких має свої переваги, обмеження та сферу доцільного використання.

До беззаперечних переваг таких рішень належить можливість безперервного або періодичного збору фізіологічних показників у реальному часі, що забезпечує раннє виявлення погіршень стану пацієнта. Також спостерігається активний розвиток автоматизованих систем оцінки ризиків (як-от Early Warning Score), широке застосування бездротових сенсорів, хмарних сервісів зберігання даних, інтероперабельності з електронними медичними системами, що дозволяє інтегрувати моніторинг в загальний клінічний робочий процес. У мобільному сегменті фітнес-додатки, орієнтовані на покращення загального самопочуття користувача, збагачені персоналізованими алгоритмами планування фізичних навантажень, дієтичних рекомендацій, когнітивної поведінкової терапії тощо.

Разом з тим, наявна низка обмежень, які стримують універсальне впровадження таких систем. По-перше, відсутність єдиних протоколів передачі,

зберігання та обробки даних призводить до фрагментації інформації між різними платформами. По-друге, значна частина мобільних застосунків працює за принципом самозвітування, що знижує об’єктивність отриманих даних. У клінічному контексті проблемними залишаються питання інтеграції в інфраструктуру лікарні, захисту даних, конфігураційної складності, потреби в навчанні персоналу та значної вартості повного розгортання системи. Не менш критичною є відсутність зворотного зв’язку між персональними гаджетами користувача та медичними інформаційними системами.

Результати, представлені на рисунку 1.15, ґрунтуються на узагальненому аналізі функціональних характеристик трьох основних категорій систем моніторингу стану здоров’я: мобільних додатків, портативних IoT-пристроїв та клінічних моніторингових систем. Кожен тип системи було оцінено за п’ятьма ключовими критеріями, що відображають як технічні, так і експлуатаційні властивості: рівень доступності, об’єктивність зібраних даних, ступінь інтеграції з електронними медичними записами (EHR), точність фізіологічних вимірювань та гнучкість/масштабованість.

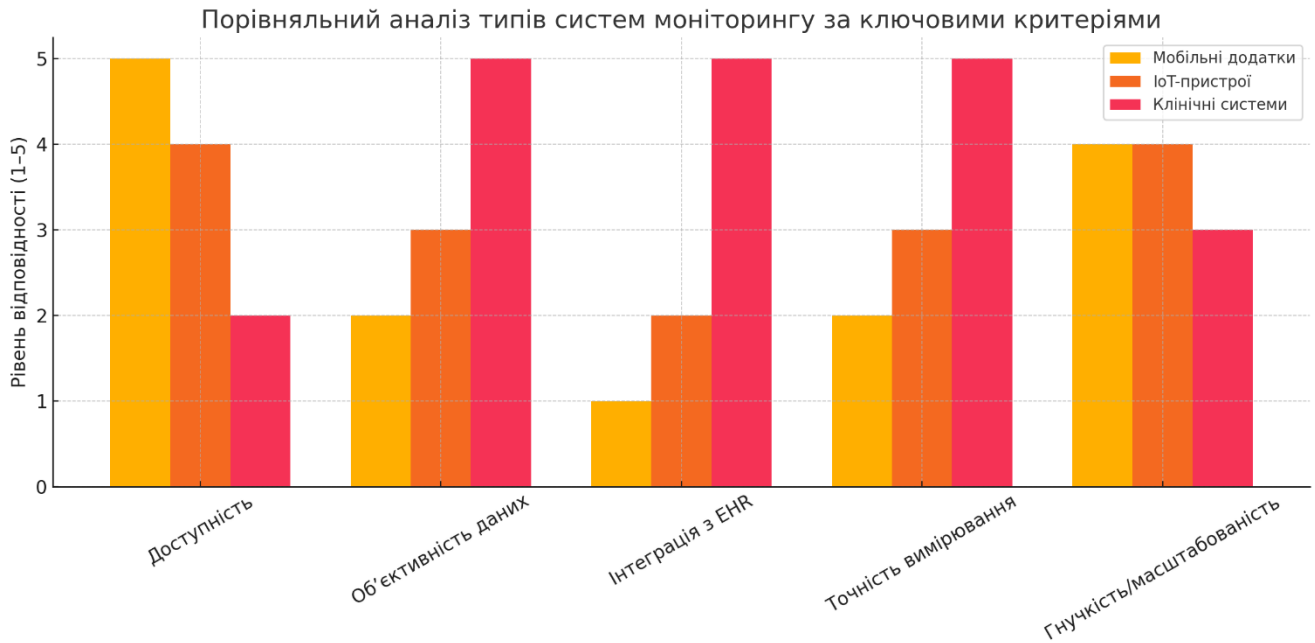


Рис. 1.15 - Порівняльний аналіз типів систем моніторингу стану здоров’я за ключовими критеріями функціональності

Як видно з графіка, мобільні фітнес-додатки демонструють найвищу доступність (оцінка 5 із 5), завдяки простоті інсталяції, низькому порогу входу для користувачів і широкому розповсюдженню через цифрові платформи. Водночас ці рішення мають низький рівень об’єктивності (2) та фактичну відсутність інтеграції з EHR-системами (1), що значно обмежує їхню клінічну придатність. Точність вимірювань також залишається помірною (2), оскільки більшість таких систем покладається на самозвітування або базові датчики без валідації результатів.

IoT-пристрої для моніторингу мають кращий баланс між доступністю (4) та об’єктивністю (3), оскільки забезпечують автоматичний збір показників життєдіяльності через сенсори. Вони частково інтегруються з медичними платформами (оцінка 2) і забезпечують прийнятний рівень точності (3), але все ще поступаються в цій категорії професійним системам. Їхня гнучкість оцінюється на рівні 4 — завдяки підтримці бездротової передачі даних і мобільності.

Клінічні системи, як Philips IntelliVue Guardian чи GE CARESCAPE, демонструють найвищі показники у сфері точності (5), інтеграції з EHR (5) та об’єктивності даних (5), оскільки працюють із сертифікованими сенсорними модулями, мають алгоритми виявлення ризиків і забезпечують безперервний контроль стану пацієнта. Проте їхня доступність є низькою (2), враховуючи високу вартість і необхідність спеціалізованого обслуговування. Гнучкість таких систем також дещо нижча (3), оскільки масштабування потребує складної технічної конфігурації.

Візуалізовані дані підкреслюють потребу в гібридному рішенні, яке б поєднувало технічну точність і достовірність клінічних систем із гнучкістю, доступністю та зручністю мобільних платформ.

РОЗДІЛ 2. ПОСТАНОВКА ТЕХНІЧНОГО ЗАВДАННЯ, ВИБІР ТЕХНОЛОГІЧНОГО СТЕКУ ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Постановка технічного завдання та визначення функціональних вимог

Метою розробки є створення інформаційної системи моніторингу стану здоров’я в реальному часі, орієнтованої на персональне використання з можливістю віддаленого доступу до зібраних біомедичних даних. Система повинна забезпечувати неперервне вимірювання фізіологічних показників, їх обробку, візуалізацію та передавання у хмарне сховище для подальшого аналізу. Важливою вимогою є підтримка роботи з мінімальним втручанням користувача та можливість автономного функціонування.

Основними об’єктами спостереження в рамках системи є температура тіла та частота серцевих скорочень користувача. Для цього передбачається використання цифрового термодатчика на основі протоколу 1-Wire (наприклад, DS18B20) та аналогового пульс-сенсора, підключених до мікроконтролера ESP8266. Останній виконує функції обробки сигналів, попереднього аналізу даних, організації бездротового з’єднання з мережею Wi-Fi і передавання показників до платформи дистанційного моніторингу.

Для збереження та віддаленого перегляду медичних даних має використовуватися сервіс ThingSpeak, який дозволяє публікувати та візуалізувати значення температури тіла та пульсу у вигляді часових графіків. Дані повинні надсилатися з періодичністю, що забезпечує актуальність показників, без надмірного навантаження на мережу та енергоспоживання. Система повинна бути енергоефективною, працювати стабільно в умовах побутового бездротового з’єднання та не потребувати втручання оператора після початкової конфігурації.

Функціональні вимоги до системи формулюються таким чином:

- здійснювати безперервне зчитування температури тіла та частоти серцевих скорочень;

- обробляти аналоговий сигнал пульсу для визначення миттєвих значень ЧСС (у вигляді BPM);
- виявляти пульсові імпульси та інтервали між ними (IBI);
- виконувати попередню фільтрацію та нормалізацію даних на мікроконтролері;
- забезпечити підключення до мережі Wi-Fi і автентифікацію за вказаними параметрами;
- надсилати зібрані показники до сервісу ThingSpeak із вказаним API-ключем;
- формувати у хмарі відповідні графіки для температури та пульсу;
- працювати у фоновому режимі без взаємодії з користувачем після запуску;
- мати можливість масштабування або модифікації під додаткові сенсори у майбутньому.

Система має бути орієнтована на використання у домашніх умовах або в амбулаторному середовищі, де необхідне періодичне дистанційне спостереження за базовими параметрами життєдіяльності людини. Очікується, що реалізація таких вимог дозволить підвищити ефективність самостійного контролю здоров’я, а також забезпечити підготовку до подальшої інтеграції з телемедичними сервісами.

2.2 Обґрунтування вибору технологічного стеку та інструментальних засобів

Побудова ефективної та енергоощадної інформаційної системи моніторингу стану здоров’я потребує ретельного вибору апаратних компонентів, здатних забезпечити точність, стабільність і зручність експлуатації в умовах обмежених ресурсів.

Мікроконтролер ESP8266 (NodeMCU) є центральним обчислювальним вузлом системи, відповідальним за обробку даних з сенсорів, керування логікою взаємодії та бездротову передачу інформації. Завдяки вбудованому Wi-Fi-модулю,

ESP8266 дозволяє реалізувати з'єднання з хмарними сервісами без потреби у додатковому мережевому обладнанні.

На рис. 2.1 зображено функціональне призначення кожного виводу (GPIO) мікроконтролера ESP8266 NodeMCU.

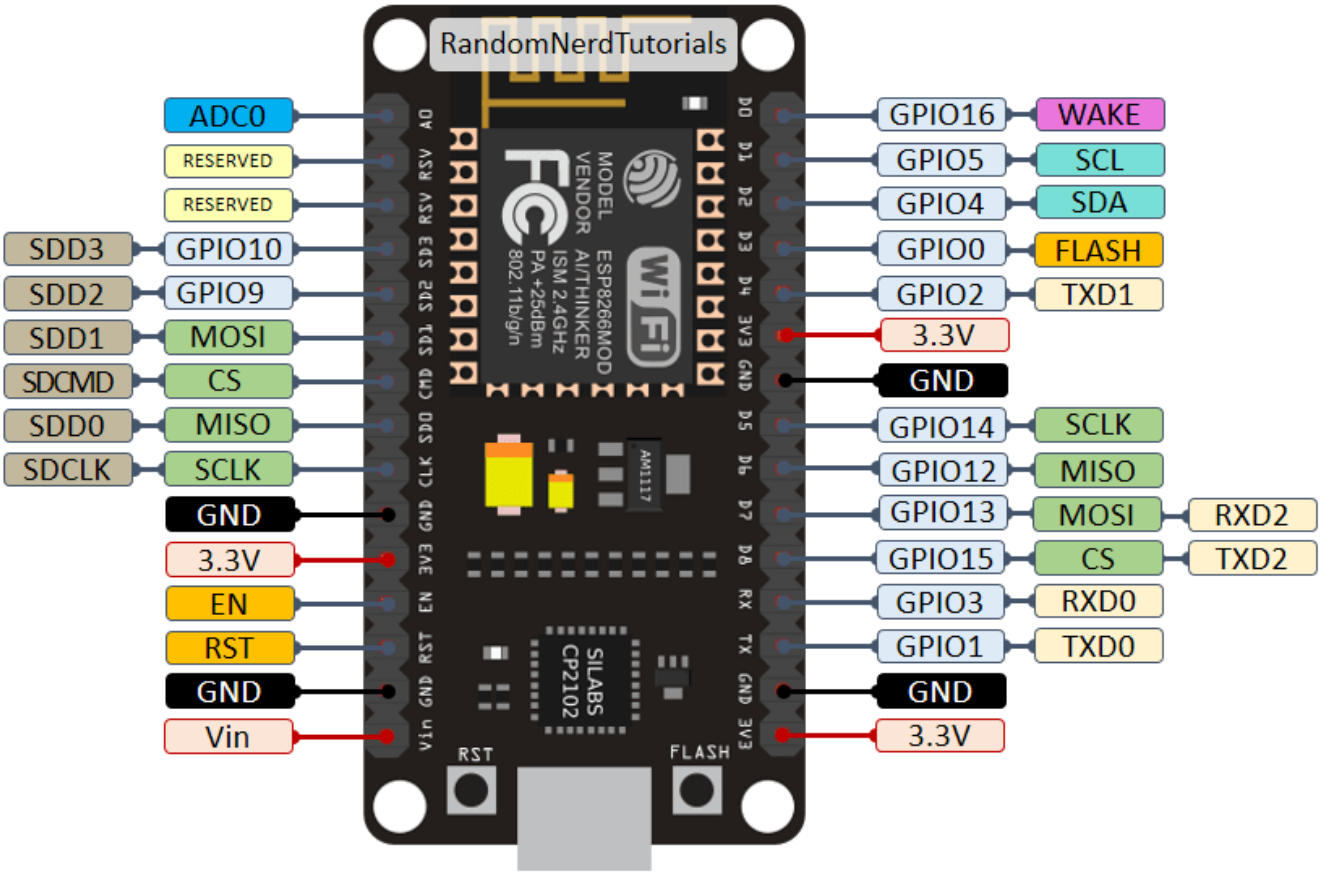


Рис. 2.1 Порти мікроконтролера NodeMCU [16]

Загального призначення цифрові входи/виходи (GPIO) використовуються для зчитування логічного стану або передачі керувальних сигналів. Порти з підтримкою протоколів UART, SPI та I²C забезпечують комунікацію з зовнішніми пристроями та сенсорами. Зокрема, порти TXD0/RXD0 використовуються для серійного обміну з комп'ютером під час розробки і налагодження, SDA/SCL — для взаємодії з модулями через шину I²C, а MOSI, MISO, SCLK — для реалізації SPI-зв'язку. Окрему функцію виконують лінії живлення (3.3V, GND, Vin), які слугують джерелом стабілізованої напруги для підключених компонентів. Наявність одного аналогового входу ADC0 дозволяє зчитувати аналоговий сигнал з пульс-сенсора.

Завдяки такому функціональному розмаїттю пінів, NodeMCU забезпечує гнучке компонування схеми та мінімізує обмеження при розширенні системи [16].

Його підтримка програмування у середовищі Arduino IDE, низьке енергоспоживання та достатня обчислювальна потужність роблять його оптимальним вибором для задач вбудованого моніторингу.

Цифровий датчик температури DS18B20 вибрано через його високу точність (до ±0,5°C), цифровий протокол передачі даних (1-Wire), можливість підключення кількох сенсорів до однієї шини та надійність у широкому діапазоні температур (від −55°C до +125°C).

На рис. 2.2 зображено типові варіанти конструктивного виконання цифрового датчика температури DS18B20 у трьох корпусах: TO-92, SO (150 mils) та µSOP.

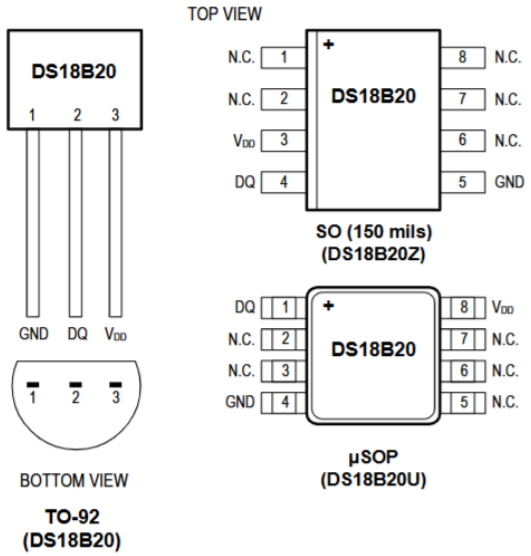


Рис. 2.2 Конфігурація виводів цифрового датчика температури DS18B20 [17]

У межах кожного з них представлено призначення основних виводів: живлення (V_{DD}), заземлення (GND) та лінії даних (DQ). Вивід DQ забезпечує одночасно передачу цифрових даних і, в разі роботи у паразитному режимі, живлення сенсора. Така реалізація дозволяє мінімізувати кількість необхідних з’єднань та спрощує інтеграцію датчика в мікроконтролерну систему. Завдяки технології 1-Wire, всі обмінні операції виконуються по одному сигнальному проводу, що дозволяє створювати компактні та масштабовані мережі

температурного моніторингу. У конфігурації TO-92, що найчастіше застосовується в практичних реалізаціях, виводи мають наступне призначення: 1 – GND, 2 – DQ, 3 – V_{DD}. Для режиму паразитного живлення вивід живлення може бути підключений до GND, а живлення здійснюється безпосередньо через DQ-провід. Така архітектура забезпечує зниження апаратної складності і є зручною для реалізації систем з великою кількістю датчиків у складних умовах експлуатації.

Інтеграція із бібліотеками OneWire і DallasTemperature дозволяє спростити реалізацію програмної взаємодії з пристроєм, забезпечуючи стабільне отримання температурних значень.

Аналоговий пульс-сенсор використовується для вимірювання частоти серцевих скорочень за допомогою фотоплетизмографії. Датчик реєструє зміни інтенсивності світлового відбиття від шкіри, що корелюють із пульсовими хвилями. Завдяки використанню відповідних фільтрів і алгоритмів згладжування можливо в реальному часі обчислювати BPM (ударів за хвилину) та IBI (інтервал між ударами), що забезпечує високу інформативність при мінімальних апаратних витратах.

На рис. 2.3 представлено комплексне зображення конструкції, принципу дії та електронної структури оптичного пульс-сенсора, що базується на методі фотоплетизмографії.

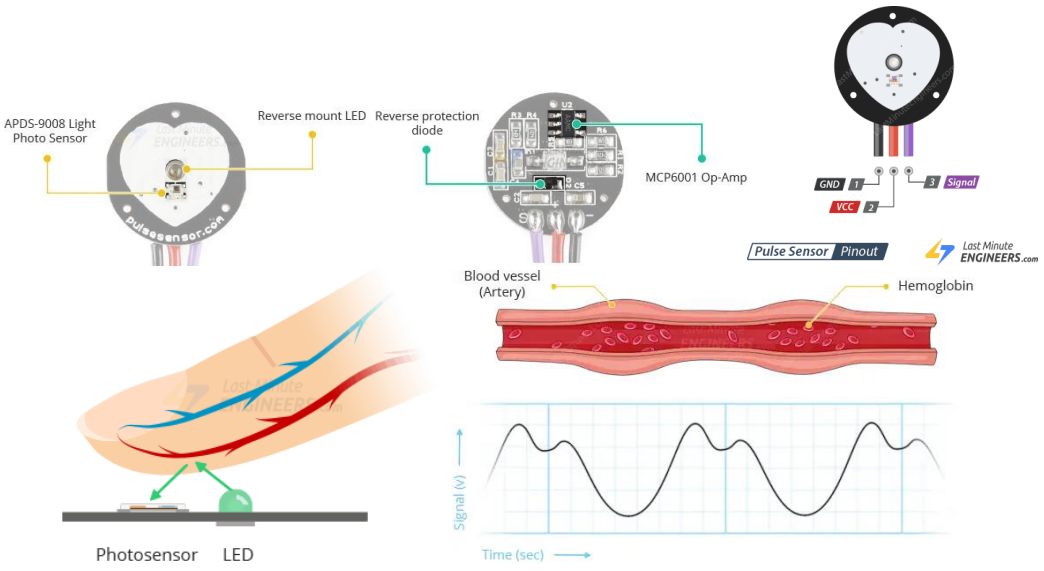


Рис. 2.3 Конструкція, принцип роботи та структура пульс-сенсора на основі фотоплетизмографії (PPG) [19]

Центральним елементом є світлодіод з тильним монтажем, який випромінює світло в діапазоні видимого спектру у напрямку шкіри пацієнта. Розсіяне світло частково поглинається гемоглобіном у кровоносних судинах, а решта – реєструється високочутливим фотодатчиком APDS-9008, розташованим поруч. На основі змін інтенсивності відбитого світла, викликаних пульсаціями крові, формується аналоговий сигнал, який надалі піддається фільтрації та посиленню за допомогою операційного підсилювача MCP6001. Сформований сигнал є періодичним і містить характерну амплітудну модуляцію, що відображає серцевий ритм користувача.

Конструктивно сенсор реалізований у вигляді трипровідного модуля, де передбачено підключення до живлення (VCC), землі (GND) та аналогового виходу (Signal), з чітким маркуванням на платі. Присутність захисного діода зворотного струму підвищує надійність схеми при нестабільному живленні. Механізм взаємодії сенсора зі шкірою демонструє розташування пальця безпосередньо над парою світлодіод-фотодіод, що дозволяє забезпечити максимально стабільний контакт і мінімізувати зовнішні світлові завади. У нижній частині рис. 2.3 візуалізовано типову форму аналогового сигналу на виході сенсора, який відповідає пульсовим хвилям та використовується для обчислення частоти серцевих скорочень (BPM). Така структура дає змогу реалізувати простий, але ефективний метод моніторингу пульсу у портативних та автономних системах [19].

Обрані апаратні компоненти повністю відповідають технічним та функціональним вимогам системи. Вони дозволяють забезпечити точний моніторинг фізіологічних параметрів, безперервну роботу в автономному режимі, а також адаптивність до майбутнього масштабування проєкту.

Програмна складова системи також відіграє ключову роль у забезпеченні стабільної роботи, масштабованості та інтеграції із зовнішніми сервісами.

Мова програмування C++ (Arduino-style) обрана як основна для реалізації прошивки мікроконтролера. Її перевагами є висока швидкодія, низький рівень абстракції, гнучкий доступ до апаратного рівня пристрою та широке поширення у вбудованих системах. Стилізація під Arduino дозволяє використовувати численні

відкриті бібліотеки, що значно спрощує взаємодію з периферією та реалізацію мережевих функцій.

Мова програмування C++ для Arduino є адаптованим варіантом класичного C++, орієнтованим на розробку прошивок для мікроконтролерів. Вона підтримує ключові особливості, такі як об'єктно-орієнтоване програмування, роботу з класами, шаблонами, указівниками та апаратно орієнтованими структурами. У контексті платформи Arduino синтаксис C++ спрощено завдяки використанню попередньо визначених функцій setup() та loop(), які формують основну логіку програми.

Особливістю стилю Arduino C++ є орієнтація на читабельність та модульність. Рекомендується структуровано розділяти код на функціональні блоки, уникати надмірної вкладеності, використовувати зрозумілі імена змінних, констант та функцій. Широко застосовуються стандартні бібліотеки та API для взаємодії з периферією (цифрові/аналогові порти, UART, I²C, PWM тощо), що дозволяє скоротити час розробки. У середовищі Arduino IDE компіляція, завантаження та налагодження програм реалізується в один клік, що робить C++ надзвичайно зручним для розробки вбудованих систем [20].

Arduino IDE, інтерфейс якого показано на рис. 2.4, виступає як інтегроване середовище розробки, що забезпечує підтримку компіляції, завантаження прошивки, серійного моніторингу та зручну бібліотечну екосистему. Його простота, кросплатформеність і сумісність із мікроконтролером ESP8266 сприяють скороченню часу розробки та полегшенню налагодження коду.

Рис. 2.4 Інтерфейс середовища Arduino IDE [21]

Wi-Fi (802.11 b/g/n) використовується як основний бездротовий протокол для підключення системи до локальної мережі та Інтернету. Інтеграція з Wi-Fi дозволяє у реальному часі передавати зібрані дані на хмарні сервіси або вебінтерфейс, а також забезпечує можливість масштабування проєкту для роботи в IoT-мережах.

На рис. 2.5 наведено графічне порівняння основних специфікацій бездротових протоколів передачі даних IEEE 802.11b, 802.11g та 802.11n, які використовуються в мережах Wi-Fi. Протокол 802.11n, який підтримується мікроконтролером ESP8266, є найбільш сучасним серед зазначених і забезпечує максимальну швидкість передачі до 300 Мбіт/с, що в кілька разів перевищує пропускну здатність попередніх стандартів: 802.11g (54 Мбіт/с) та 802.11b (11 Мбіт/с). Крім того, він має збільшену зону покриття та вищу стійкість до завад завдяки використанню технології множинного антенування (MIMO) та ширшого спектра частот. Завдяки цим перевагам, стандарт 802.11n є оптимальним для систем медичного моніторингу, які потребують стабільного з’єднання з хмарними платформами.

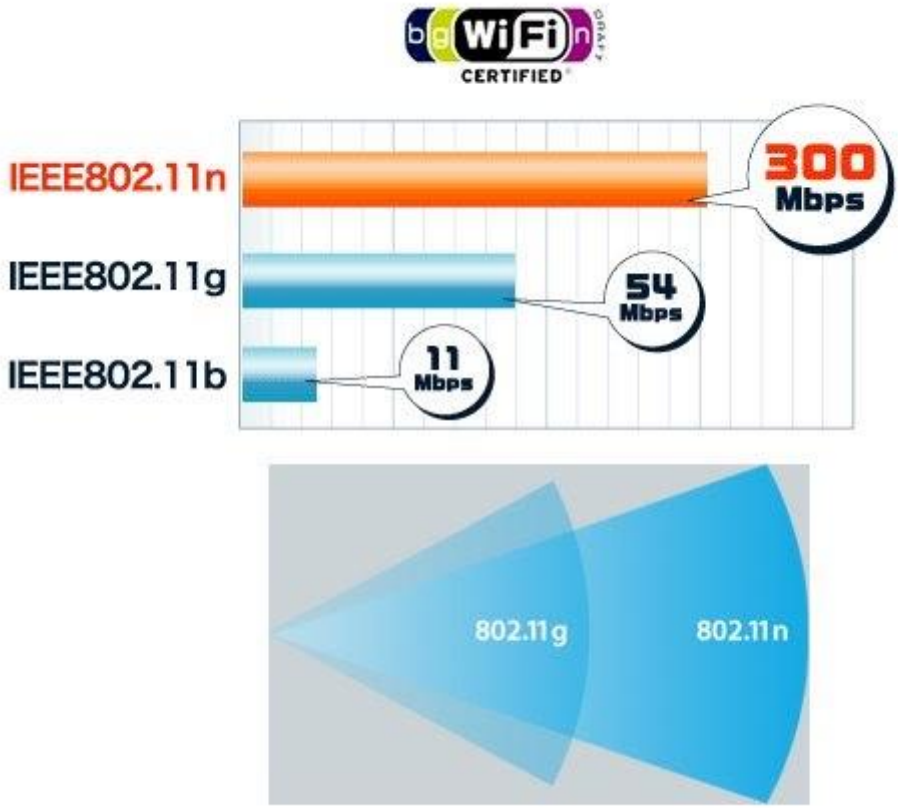


Рис. 2.5 Порівняння швидкісних характеристик бездротових стандартів IEEE 802.11b/g/n [22]

Хмарна платформа ThingSpeak слугує засобом зберігання, обробки та візуалізації даних у режимі реального часу. Вона підтримує RESTful API, автоматичне оновлення графіків та можливість аналітики без потреби у власному сервері. Такий підхід дозволяє реалізувати централізований моніторинг фізіологічних показників без складної серверної інфраструктури.

Як проілюстровано на рис. 2.6, ThingSpeak функціонує як MQTT-сервер, до якого підключаються мікроконтролери (наприклад, NodeMCU) у ролі MQTT-клієнтів, передаючи дані з різних сенсорів, зокрема температури, вологості, струму тощо. Зібрана інформація надходить у хмару, де вона зберігається, обробляється та виводиться для подальшого аналізу користувачем [23].

Рис. 2.6 Схема інтеграції сенсорних пристроїв з хмарною платформою ThingSpeak у режимі MQTT-клієнтів [23]

Інтернет-протокол HTTP/HTTPS застосовується для встановлення зв'язку між мікроконтролером та хмарною платформою.

На рис. 2.7 представлено модель обміну повідомленнями між клієнтською системою та брокером повідомлень, що працює в хмарному або серверному середовищі, із залученням захищеного з'єднання через HTTP/HTTPS-проксі.

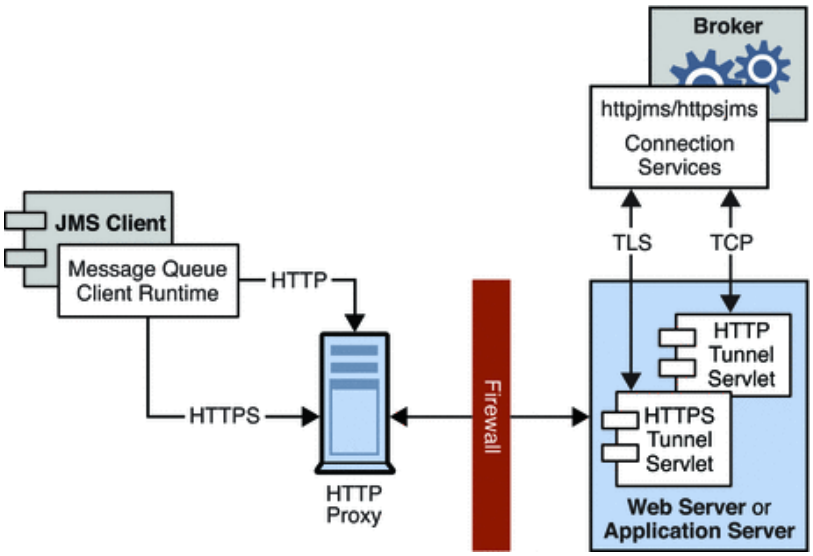


Рис. 2.7 Архітектура взаємодії клієнта з брокером через HTTP/HTTPS проксі у захищеному середовищі [24]

Клієнтська частина (JMS Client) реалізує підключення до брокера за допомогою стандартних сервісів з'єднання, які підтримують передачу повідомлень через протоколи HTTP або HTTPS. Проксі-сервер виконує роль шлюзу між внутрішньою мережею клієнта та зовнішнім сервером, забезпечуючи додатковий рівень безпеки шляхом контролю доступу через міжмережевий екран (firewall). На стороні сервера, повідомлення обробляються відповідними сервлетами HTTP/HTTPS тунелю, які передають їх далі брокеру, використовуючи безпечні канали зв'язку (TCP або TLS). Така архітектура дозволяє забезпечити стійкість до мережевих загроз, шифрування трафіку та можливість масштабування IoT-рішень, які взаємодіють із хмарною платформою через стандартні веб-протоколи.

Надійність протоколу, підтримка захищених з'єднань та універсальність реалізації роблять його доцільним вибором для інтеграції з більшістю сучасних вебсервісів.

Вибір програмних інструментів забезпечує баланс між функціональністю, продуктивністю та простотою реалізації, що дозволяє системі ефективно виконувати свої завдання в контексті моніторингу стану здоров'я користувача.

На рис. 2.8 зображено узагальнену мережеву структуру компонентів, які становлять основу розроблюваної системи моніторингу стану здоров'я. Схема розділена на дві логічні гілки: апаратну та програмну частину. До апаратної частини належать мікроконтролер ESP8266 (NodeMCU), цифровий температурний сенсор DS18B20, аналоговий пульс-сенсор, макетна плата та з'єднувальні провідники, які забезпечують зчитування та передавання біомедичних сигналів. Програмна частина включає мову програмування C++ в стилі Arduino, середовище розробки Arduino IDE, бездротовий зв'язок за протоколом Wi-Fi (802.11 b/g/n), хмарну платформу ThingSpeak для візуалізації даних та HTTP/HTTPS протоколи для реалізації API-запитів. Така інтеграція апаратних і програмних компонентів гарантує ефективну, стабільну та масштабовану роботу системи в умовах реального часу.

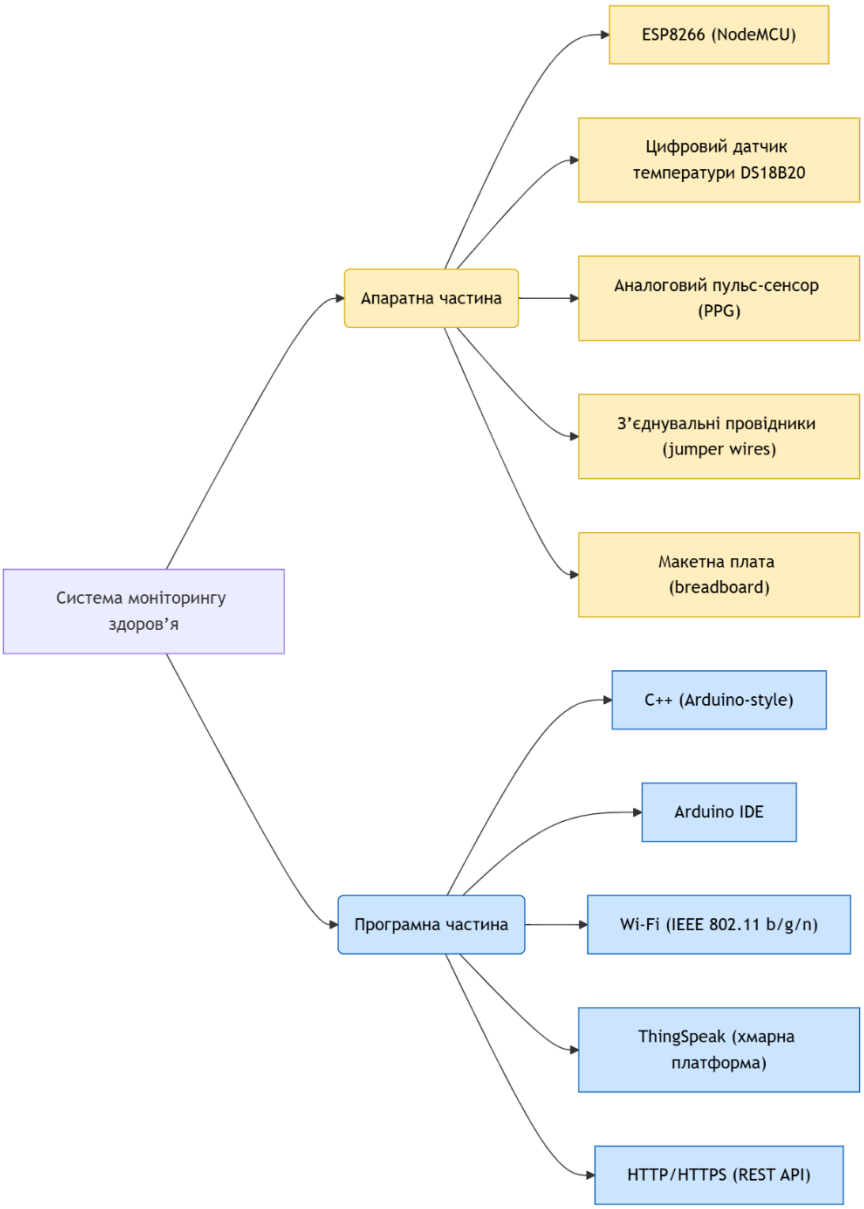


Рис. 2.8 Структурна схема обраного технологічного стеку апаратного та програмного забезпечення системи

У сукупності апаратні та програмні засоби, обрані для реалізації системи, забезпечують оптимальне поєднання надійності, гнучкості та простоти інтеграції. Застосування відкритих апаратних платформ, доступних сенсорів та інструментів розробки з відкритим кодом дозволяє не лише скоротити витрати на розробку, а й забезпечити масштабованість проєкту в майбутньому. Весь стек технологій орієнтований на досягнення високої функціональної ефективності системи в умовах обмежених ресурсів вбудованих пристроїв.

2.3 Проєктування архітектури, структури та користувацького інтерфейсу системи

На рис. 2.9 представлено логічну архітектуру функціонування системи, яка дозволяє реалізувати повноцінне зчитування, передавання, зберігання та візуалізацію медичних показників у режимі реального часу.

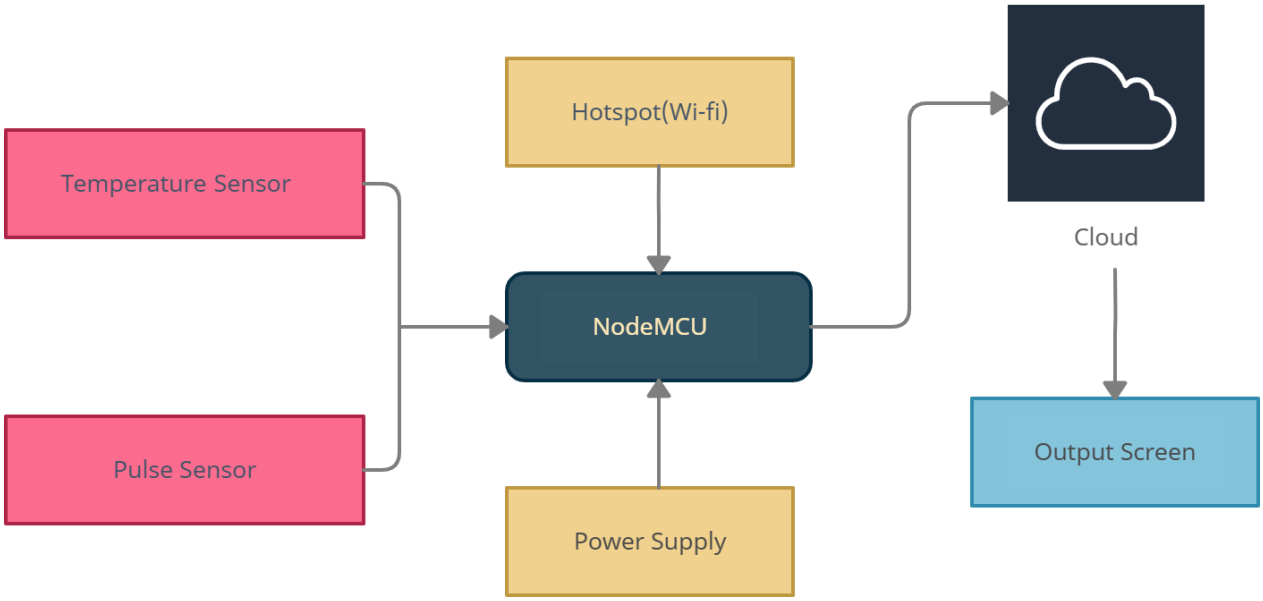


Рис. 2.9 Структурна архітектура запропонованої системи моніторингу стану здоров'я

Центральним елементом є мікроконтролер NodeMCU, побудований на базі модуля ESP8266, що виконує роль обчислювального ядра, комунікаційного інтерфейсу та вузла агрегації даних.

До NodeMCU підключені два основних сенсорних пристрої: цифровий температурний сенсор DS18B20, який передає значення температури тіла, та аналоговий пульс-сенсор типу PPG, що вимірює частоту серцевих скорочень на основі оптичної фотоплетизмографії. Всі ці дані зчитуються через відповідні GPIO-піни плати, обробляються у прошивці, написаній мовою C++ у середовищі Arduino IDE, та інтерпретуються в цифрову форму, придатну для подальшого передавання.

Для забезпечення бездротового зв'язку NodeMCU підключається до локальної мережі за допомогою Wi-Fi-модуля, який підтримує стандарти IEEE

802.11 b/g/n. Це дозволяє у реальному часі відправляти дані на хмарний сервіс — у даному випадку на платформу ThingSpeak, яка забезпечує функції зберігання, обробки та візуалізації результатів через web-інтерфейс.

Після обробки дані можуть бути відображені на вивідному екрані (Output Screen), доступному користувачеві через браузер або інший web-клієнт. Архітектура системи реалізує повний цикл збору і доставки критично важливої медичної інформації — від фізіологічних сенсорів до віддаленого користувача або медичного персоналу.

Живлення системи забезпечується окремим модулем живлення (Power Supply), який підтримує стабільну роботу сенсорів та мікроконтролера протягом тривалого періоду. Загальна архітектура проєкту є модульною, масштабованою та оптимізованою для застосування в умовах портативного використання або медичних закладів.

На рис. 2.10 зображено апаратну реалізацію фізичних з’єднань основних сенсорних компонентів системи моніторингу стану здоров’я з мікроконтролером NodeMCU. У верхній частині схеми представлено підключення аналогового пульс-сенсора, який використовує фотоплетизмографічний принцип дії. Живлення сенсора подається з 3.3 V (пін 3V3) на платі NodeMCU до пінів VCC сенсора, тоді як GND сенсора з’єднано з піном GND мікроконтролера. Сигнальний пін сенсора під’єднується до аналогового входу A0, через який зчитується амплітуда оптичного сигналу, що корелює з серцевими скороченнями користувача.

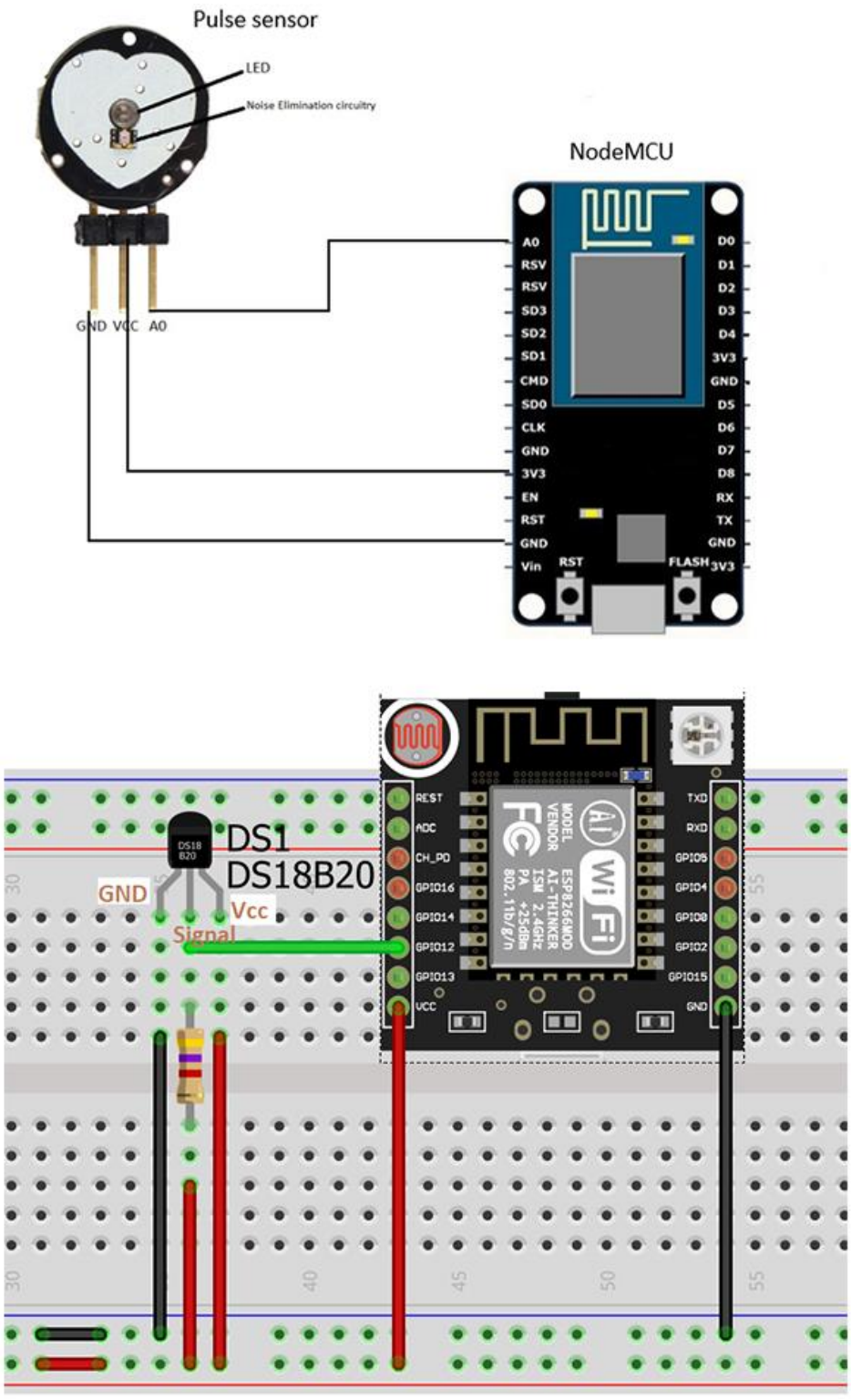


Рис. 2.10 Схема підключення сенсорів до плати NodeMCU

У нижній частині наведено підключення цифрового температурного сенсора DS18B20. Для його живлення використано пін 3.3V та GND, відповідно до стандартної конфігурації. Сигнальний пін (жовтий дріт або пін "Signal") з’єднано з цифровим входом GPIO12 (D6) мікроконтролера. Для забезпечення стабільності

передавання сигналу на шині 1-Wire, між лінією живлення (Vcc) та сигнальною лінією встановлено підтягувальний резистор номіналом 4.7 кОм, який дозволяє уникнути спотворень даних і забезпечує коректну ініціалізацію пристрою.

Ця конфігурація забезпечує коректне функціонування двох незалежних сенсорних каналів: аналогового (пульс) і цифрового (температура), що обробляються NodeMCU для подальшої передачі на хмарну платформу через Wi-Fi.

Реалізація користувацького інтерфейсу в розробленій системі моніторингу стану здоров’я здійснюється за допомогою хмарної платформи **ThingSpeak**, яка забезпечує відображення зібраних біометричних даних у вигляді інтерактивних графіків. Як зображено на рис. 2.11, вебінтерфейс містить окремі ділянки для кожного каналу даних, що відповідає конкретному сенсору.

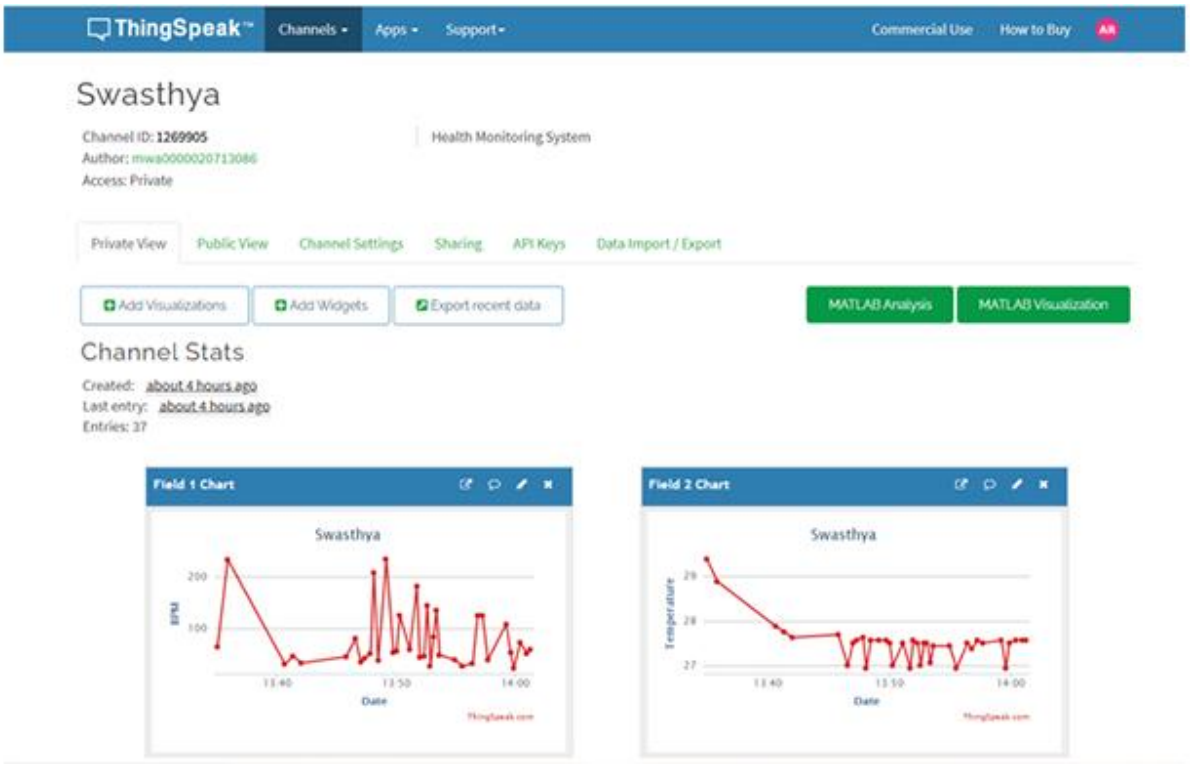


Рис. 2.11 Інтерфейс виводу даних на платформі ThingSpeak

У даному випадку канали Field 1 і Field 2 візуалізують динаміку зміни частоти серцевих скорочень (BPM) та температури тіла відповідно, з прив’язкою до часових міток.

Інтерфейс автоматично оновлюється в реальному часі, використовуючи механізми HTTP/HTTPS запитів. Користувач може змінювати параметри виводу, налаштовувати типи графіків, інтервали оновлення, колірні схеми, масштаб осей тощо. Також доступні засоби експорту, API-ключі для інтеграції з іншими застосунками та блоки MATLAB для подальшої обробки даних. Гнучкість візуалізації дозволяє не лише спостерігати за поточними значеннями, а й виявляти тренди або аномалії в історичних даних, що є важливою складовою у задачах персонального або клінічного моніторингу.

Платформа ThingSpeak виступає не лише як сховище для телеметрії, а й як ефективний засіб реалізації інтерфейсу користувача без потреби в окремому розробленні фронтенд-компонентів.

РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ СТАНУ ЗДОРОВ'Я МОВОЮ C++

3.1. Реалізація програмної частини системи мовою C++

На рис. 3.1 показано загальний алгоритм роботи розробленого програмного забезпечення для системи моніторингу стану здоров'я.

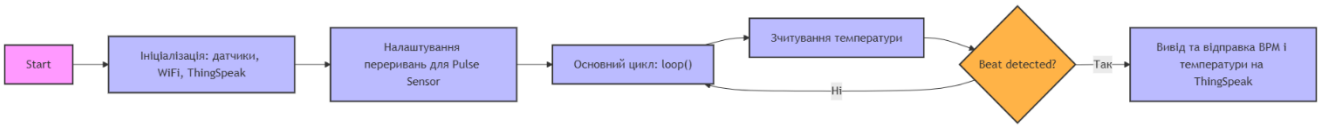


Рис. 3.1 Блок-схема загального алгоритму роботи системи моніторингу стану здоров'я

Алгоритм функціонування системи розпочинається з етапу ініціалізації, під час якого здійснюється налаштування апаратних компонентів, зокрема датчиків температури та пульсу, а також встановлення з'єднання з бездротовою мережею Wi-Fi та конфігурація параметрів взаємодії з хмарною платформою ThingSpeak. На цьому етапі визначаються параметри каналів для передачі фізіологічних даних, зчитуються відповідні ключі API та встановлюється стабільне з'єднання з мережею передачі даних.

Далі виконується налаштування апаратного переривання для пульсового сенсора, що дозволяє реагувати на події реєстрації серцевих скорочень у реальному часі. Переривання прив'язується до певного цифрового входу контролера та асоціюється з обробником, який реагує на кожне детектоване серцебиття, забезпечуючи точність вимірювання частоти пульсу (BPM — beats per minute).

Після завершення етапу ініціалізації система переходить у основний цикл виконання (loop()), в межах якого реалізується повторюване опитування сенсорів і логіка передачі даних. На кожній ітерації циклу здійснюється зчитування поточного значення температури тіла з цифрового датчика DS18B20. Отримані значення обробляються, приводяться до необхідного формату, після чого визначається, чи зафіксовано пульсовим сенсором факт серцевого скорочення. У разі виявлення події

серцебиття ініціюється процес обчислення актуального значення пульсу (BPM) та формування відповідного повідомлення для передачі.

У випадку, якщо подія пульсу зафіксована, система виконує відправлення зібраних значень температури та частоти пульсу на платформу ThingSpeak за допомогою інтернет-протоколу HTTP. Ці дані зберігаються у відповідних каналах, відображаються у вигляді графіків та можуть бути проаналізовані користувачем у режимі реального часу.

Програмна логіка поєднує переривання на апаратному рівні з циклічним збором даних і хмарною передачею інформації, що забезпечує ефективне, надійне та масштабоване рішення для моніторингу фізіологічного стану користувача.

У фрагменті коду, наведеному на рис. 3.2, здійснюється підключення бібліотек, необхідних для реалізації функціональності вбудованої системи моніторингу стану здоров'я. Кожна з них відіграє ключову роль у забезпеченні взаємодії програмної частини з апаратними компонентами та зовнішніми сервісами.

```
#include <DallasTemperature.h>
#include <OneWire.h>
#include <ESP8266WiFi.h>
#include <Ticker.h>
#include "ThingSpeak.h"
```

Рис. 3.2 Підключення бібліотек для реалізації основної функціональності програми

Бібліотека <DallasTemperature.h> відповідає за взаємодію з цифровим температурним сенсором DS18B20. Вона надає високорівневі функції для ініціалізації датчика, зчитування температури, керування режимами роботи тощо. У поєднанні з OneWire ця бібліотека дозволяє використовувати кілька температурних сенсорів на одній шині передачі даних.

Бібліотека <OneWire.h> реалізує протокол OneWire, що використовується для комунікації з пристроями, які підтримують цю шину, зокрема сенсорами DS18B20.

OneWire дозволяє підключати численні пристрої до одного цифрового порту мікроконтролера, ідентифікуючи їх за унікальними адресами.

Бібліотека <ESP8266WiFi.h> надає засоби для підключення мікроконтролера NodeMCU на базі чипа ESP8266 до бездротової мережі Wi-Fi. Вона включає функції налаштування точки доступу, підключення до наявної мережі, керування параметрами мережевого інтерфейсу та діагностики з'єднання.

Бібліотека <Ticker.h> реалізує механізм таймерів зворотного виклику (callback), які дозволяють запускати функції з певною періодичністю незалежно від основного циклу програми. Це забезпечує ефективну обробку переривань, наприклад, у випадку підрахунку серцевих скорочень або регулярного зчитування показників сенсорів.

Нарешті, бібліотека "ThingSpeak.h" забезпечує взаємодію з хмарною платформою ThingSpeak, яка слугує для зберігання, обробки та візуалізації телеметричних даних. Вона містить API-запити для надсилання даних на сервер, керування каналами та параметрами з'єднання, що дозволяє передавати зібрані з датчиків значення безпосередньо у вебінтерфейс користувача.

У фрагменті коду, наведеному на рис. 3.3, оголошуються константи за допомогою директиви препроцесора #define, що дозволяє прив'язати зрозумілі ідентифікатори до конкретних апаратних або конфігураційних параметрів системи.

```
#define pulsePin A0
#define ONE_WIRE_BUS D2

#define WIFI_SSID "   "
#define WIFI_PASSWORD "   "
```

Рис. 3.3 Оголошення констант для конфігурації пінів мікроконтролера та підключення до Wi-Fi-мережі

Константа pulsePin визначена як A0, що відповідає аналоговому входу мікроконтролера NodeMCU. До цього входу підключено аналоговий пульс-сенсор (PPG-модуль), який генерує сигнал напруги, пропорційний змінам

кровонаповнення тканин. Порт A0 слугує точкою зчитування аналогових значень серцевого ритму.

Константа ONE_WIRE_BUS встановлена як D2, тобто цифровий пін GPIO4. Це той пін, через який здійснюється комунікація з цифровим температурним сенсором DS18B20 за протоколом OneWire. Встановлення окремої шини OneWire дозволяє оптимізувати взаємодію з датчиком температури, забезпечуючи точність і стабільність передачі даних.

Наступні дві директиви — WIFI_SSID та WIFI_PASSWORD — використовуються для збереження імені бездротової мережі Wi-Fi та відповідного паролю. Ці значення необхідні для ініціалізації мережевого з'єднання через модуль ESP8266. Надалі вони підставляються у функції з бібліотеки ESP8266WiFi, що дозволяє програмно підключитися до локальної мережі і, відповідно, передавати дані до хмарної платформи ThingSpeak.

Фрагмент програми, поданий на рис. 3.4, містить ключові оголошення глобальних змінних, об'єктів бібліотек та параметрів, які забезпечують базову ініціалізацію і подальше функціонування системи моніторингу фізіологічних параметрів. Усі ці змінні використовуються в різних частинах алгоритму для обробки даних із сенсорів, аналізу сигналів та передавання результатів на хмарну платформу ThingSpeak.

Змінні myChannelNumber та myWriteAPIKey використовуються для встановлення ідентифікатора каналу та відповідного ключа автентифікації, необхідного для передавання даних на ThingSpeak. WiFiClient client створює екземпляр клієнта для роботи з мережею, а Ticker flipper є об'єктом бібліотеки Ticker, що дозволяє реалізувати асинхронні виклики функцій (наприклад, для обробки переривань або таймерів).

Оголошення OneWire oneWire(ONE_WIRE_BUS); ініціалізує об'єкт шини OneWire, яка дозволяє здійснювати зв'язок з цифровим температурним датчиком DS18B20 через заданий пін. Далі DallasTemperature sensors(&oneWire); створює об'єкт для роботи з бібліотекою DallasTemperature, що забезпечує високорівневий інтерфейс для зчитування температури.

```
unsigned long myChannelNumber = ;
const char * myWriteAPIKey = " ";

int keyIndex = 0;
WiFiClient  client;

Ticker flipper;

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);
volatile int BPM;
volatile int Signal;
volatile int IBI = 600;
volatile boolean Pulse = false;
volatile boolean QS = false;
volatile int rate[10];
volatile unsigned long sampleCounter = 0;
volatile unsigned long lastBeatTime = 0;
volatile unsigned long current;
volatile int P = 512;
volatile int T = 512;
volatile int thresh = 560;
volatile int amp = 0;
volatile boolean firstBeat = true;
volatile boolean secondBeat = false;
volatile unsigned long lastMillis = 0;
volatile float tempSignal=0;
volatile int msTime = 0;
float t;
```

Рис. 3.4 Глобальні змінні та ініціалізація об’єктів для зчитування і обробки температури та пульсу

Більшість наступних змінних оголошено як volatile, що сигналізує компілятору про можливість їх змінення поза межами основного потоку виконання (наприклад, у функціях переривання). Це особливо важливо у випадку аналізу серцевого ритму, який ґрунтується на точному відстеженні часу та амплітуди сигналу.

Зокрема, BPM (Beats Per Minute) є значенням частоти серцевих скорочень, Signal — поточне аналогове значення, зчитане з пульс-сенсора. IBI (Inter-Beat Interval) задає початковий інтервал між ударами (у мс), а Pulse — логічний прапорець, що сигналізує про виявлений удар. Змінна QS (Quantified Self) вказує, чи був виявлений новий пульсовий пік.

Масив rate[10] використовується для зберігання десяти останніх значень інтервалів між ударами, що дозволяє розрахувати усереднене значення ЧСС. Параметри sampleCounter, lastBeatTime та lastMillis фіксують часові мітки для обробки даних у реальному часі. Змінні P, T, thresh, amp, firstBeat, secondBeat, tempSignal та msTime беруть участь у фільтрації, виявленні піків і стабілізації вимірювань. float t — змінна для збереження результату температурного вимірювання.

Представлений на рис. 3.5 фрагмент коду реалізує функцію setup(), яка виконується один раз під час запуску мікроконтролера і відповідає за початкову ініціалізацію апаратних і програмних компонентів системи моніторингу.

```
void setup()
{
  sensors.begin();
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to ");
  Serial.print("WIFI_SSID");
    while (WiFi.status() !=WL_CONNECTED)
          {
            Serial.print(".");
            delay(500);
          }
  Serial.println();
  Serial.print("Connected");
  Serial.println("IP Address:");
  Serial.println(WiFi.localIP());

 ThingSpeak.begin(client);
  pinMode(A0,INPUT);
  interruptSetup();
  lastMillis=millis();
}
```

Рис. 2.5 Реалізація функції setup(): початкова ініціалізація датчиків, Wi-Fi-з’єднання та ThingSpeak

На початку викликається метод sensors.begin(), який ініціалізує об’єкт DallasTemperature та готує цифровий датчик температури DS18B20 до зчитування значень через шину OneWire. Далі за допомогою Serial.begin(115200) активується

послідовний інтерфейс із швидкістю 115200 бод для можливості виведення діагностичної інформації через монітор порту.

Підключення до бездротової мережі здійснюється викликом WiFi.begin(WIFI_SSID, WIFI_PASSWORD), де використовуються попередньо оголошені змінні для SSID та пароля. Після цього система виводить повідомлення "Connecting to" з іменем мережі і входить у цикл while (WiFi.status() != WL_CONNECTED), який триває доти, поки пристрій не встановить з’єднання з мережею Wi-Fi. У цьому циклі кожні 500 мілісекунд виводиться крапка, що індикує процес очікування. Коли з’єднання встановлено, виконується послідовність повідомлень "Connected" та "IP Address", після чого відображається локальна IP-адреса модуля, отримана методом WiFi.localIP().

Після встановлення зв’язку із мережею виконується ініціалізація клієнта для хмарної платформи за допомогою виклику ThingSpeak.begin(client), що дозволяє надалі передавати дані до ThingSpeak через протокол HTTP. Вхідний пін A0, до якого підключено аналоговий пульс-сенсор, оголошується як вхідний за допомогою функції pinMode(A0, INPUT).

Функція interruptSetup() активує таймерне переривання, необхідне для періодичної вибірки сигналу пульсу з високою частотою дискретизації. Нарешті, змінній lastMillis присвоюється значення системного таймера millis(), що позначає початкову часову точку для подальших обчислень інтервалів між ударами серця.

На рис. 3.6 представлено реалізацію основного циклу loop() в Arduino-програмі, що відповідає за безперервне зчитування даних з датчиків температури та пульсу, а також за подальший їх вивід і передачу до хмарної платформи ThingSpeak.

На початку виконання циклу викликається функція sensors.requestTemperatures(), яка ініціює процес зчитування температури з цифрового сенсора DS18B20, підключеного через шину OneWire. Після цього метод sensors.getTempCByIndex(0) повертає значення температури у градусах Цельсія, яке зберігається у змінній t. Зчитування відбувається для першого пристрою на шині (індекс 0), що відповідає єдиному підключеному датчику.

```
void loop()
{
    sensors.requestTemperatures();
    t = sensors.getTempCByIndex(0);
        if(msTime>10000)
          {
            noInterrupts();
            interrupts();
            msTime=0;
          }

        if(QS == true)
          {
           // A Heartbeat Was Found
           serialOutputWhenBeatHappens();
           QS = false;
          }
}
```

Рис. 3.6 Основний цикл обробки даних сенсорів температури та серцевого ритму

Далі йде перевірка умови if (msTime > 10000), яка дозволяє раз на 10 секунд (10000 мс) виконати певні дії. У випадку виконання цієї умови спочатку викликається noInterrupts() для тимчасового вимкнення всіх апаратних переривань — це забезпечує узгодженість доступу до змінних, що можуть змінюватися в обробнику переривань. Потім одразу активуються переривання командою interrupts(), і значення таймера msTime обнуляється. Така конструкція може слугувати для синхронізації або очищення накопичених проміжних даних, запобігаючи їх перевантаженню.

У наступному блоці if (QS == true) відбувається перевірка наявності нового виявленого серцевого скорочення. Змінна QS (скорочення від "Quantified Self") виставляється у значення true у функції переривання, коли пульс-сенсор фіксує пік у сигналу. Якщо така подія зафіксована, викликається функція serialOutputWhenBeatHappens(), яка відповідає за обробку та передачу інформації про ритм серця. Після цього прапорець QS скидається в false для уникнення повторної обробки тієї ж події у наступних ітераціях циклу.

Функція loop() забезпечує циклічне зчитування даних із сенсорів, обробку серцевих скорочень і підтримку часової логіки, необхідної для стабільної роботи системи моніторингу фізіологічних параметрів.

На рис. 3.7 зображено реалізацію функції serialOutputWhenBeatHappens(), яка викликається у випадку виявлення серцевого скорочення (пульсу) та виконує серію дій з виведення та передачі даних на платформу ThingSpeak. Це один із ключових елементів алгоритму системи моніторингу, що відповідає за комунікацію з хмарною інфраструктурою.

```
void serialOutputWhenBeatHappens()
{
    sendDataToSerial('B',BPM);
    Serial.print("Temperature is: ");
    Serial.println(t);
    ThingSpeak.setField(1, BPM);
    ThingSpeak.setField(2, t);
    int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
            if(x == 200)
              {
                Serial.println("Channel update successful.");
              }
            else
              {
                Serial.println("Problem updating channel. HTTP error code " + String(x));
              }

}
```

Рис. 3.7 Передача частоти серцевих скорочень і температури на хмарну платформу ThingSpeak

На початку функції викликається допоміжна функція sendDataToSerial('B', BPM), яка, ймовірно, форматує та передає значення частоти серцевих скорочень (BPM, beats per minute) до серійного монітора. Далі реалізовано безпосередній вивід значення температури тіла: за допомогою Serial.print() та Serial.println(t) значення температурного сенсора (що зберігається у змінній t) виводиться у консоль для моніторингу в режимі реального часу.

Наступні дві інструкції ThingSpeak.setField(1, BPM) та ThingSpeak.setField(2, t) відповідають за заповнення відповідних полів каналу на платформі ThingSpeak.

Значення частоти серцевих скорочень записується у перше поле, а температура — у друге. Згодом викликається метод ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey), який надсилає всі встановлені поля на вказаний канал, автентифікуючись за допомогою API-ключа.

Результат виконання запису зберігається у змінну x. Якщо значення x дорівнює 200, це означає успішне оновлення каналу (HTTP-статус 200 — OK). У такому разі виводиться повідомлення "Channel update successful." у консоль. Якщо ж повернутий інший код (наприклад, 400 або 500 серії), то виконується альтернативна гілка else, в якій виводиться повідомлення про помилку разом із відповідним кодом (HTTP error code ...). Цей механізм дозволяє налагоджувати з’єднання та оперативно виявляти проблеми з мережею чи автентифікацією на хмарному сервісі.

Загалом, дана функція є центральним модулем передачі даних системи до інтернету та служить мостом між локальним збором біомедичних показників і їхнім представленням в онлайн-середовищі.

На рис. 3.8 представлено реалізацію двох допоміжних функцій, що забезпечують вивід даних у серійний монітор і налаштування періодичних переривань, необхідних для коректного вимірювання частоти серцевих скорочень у реальному часі.

```
void sendDataToSerial(char symbol, int data )
{
    Serial.print(symbol);
    Serial.println(data);
}

void interruptSetup()
{

    flipper.attach_ms(2, ISRTr);
}
```

Рис. 3.8 Реалізація функцій виводу у серійний монітор та налаштування таймерного переривання

Функція sendDataToSerial(char symbol, int data) призначена для зручного виведення пар значень у серійний монітор. Першим параметром є символьний маркер symbol, який дозволяє позначити тип або джерело даних (наприклад, 'B' для BPM), а другим — числове значення data, яке відображає виміряний параметр. Таким чином, функція дозволяє стандартизовано виводити результат, наприклад, у форматі B: 72, що спрощує зчитування інформації під час налагодження.

Друга функція, interruptSetup(), відповідає за конфігурацію таймерного переривання, яке необхідне для обробки сигналу з аналогового пульс-сенсора. У цьому випадку використовується метод attach_ms() об’єкта flipper (який, згідно з попереднім кодом, є екземпляром класу Ticker). Цей метод налаштовує виклик функції ISRT кожні 2 мілісекунди. Таким чином забезпечується періодичне опитування аналогового сигналу та виявлення максимумів, мінімумів і ритмічних змін, необхідних для розрахунку BPM.

Використання таймерного переривання дозволяє забезпечити стабільність та незалежність обробки пульс-сигналу від основного циклу виконання програми, що особливо важливо для досягнення точної і стабільної частоти оновлення показників.

Як показано на рис. 3.9, обробка серцевого ритму в системі реалізується у вигляді послідовного логічного алгоритму, що починається з читання аналогового сигналу з пульс-сенсора. На наступному етапі виконується перевірка: чи є сигнал меншим за попереднє порогове значення — у цьому разі відбувається оновлення змінної T, що позначає локальний мінімум сигналу. Якщо ж навпаки — сигнал перевищує попередній максимум P, його значення оновлюється. Потім виконується перевірка умови виявлення пульсу: вона враховує перевищення сигналу порогу, значну амплітуду, відсутність активного пульсу (Pulse = false) і дотримання мінімального інтервалу часу (N > IBI * 3/5).

У разі виявлення пульсу розраховується IBI — інтервал між ударами — та обчислюється BPM (частота серцевих скорочень). Значення IBI зберігається у масиві rate[], що дозволяє усереднити значення BPM на основі останніх 10 зразків. Також оновлюється змінна QS, яка слугує індикатором того, що виявлено нове

скорочення. Якщо протягом понад 2,5 секунд не виявлено пульсу, система скидає ключові змінні до значень за замовчуванням для уникнення накопичення похибок.

Рис. 3.9 демонструє структурну логіку, яка потім реалізується безпосередньо у коді функції ISRT(), представленої на рис. 3.10. Ця зв'язка дозволяє ефективно поєднати концептуальний рівень проектування алгоритму із його безпосередньою реалізацією в прошивці пристрою.

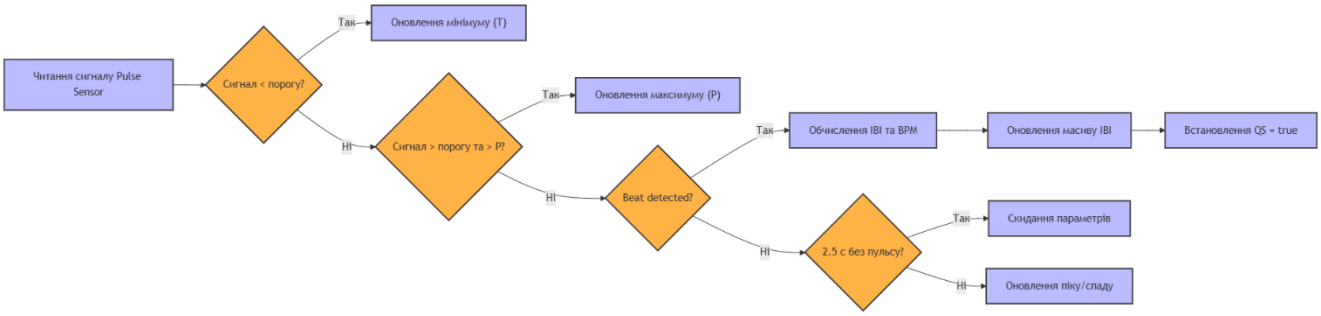


Рис. 3.9 Блок-схема алгоритму обробки сигналу серцевого ритму в обробнику переривань

У коді, наведеному на рис. 3.10, реалізується обробник переривань ISRT(), що виконує зчитування аналогового сигналу з пульс-сенсора, визначення моменту серцевого скорочення та обчислення значення частоти серцевих скорочень (BPM). Алгоритм цієї функції докладно ілюстрований на рис. 3.9. Робота функції починається з відключення переривань (noInterrupts()), щоб уникнути конфліктів під час критичних операцій над змінними.

Далі зчитується значення аналогового сигналу (analogRead(pulsePin)) і поточний момент часу у мілісекундах. На основі різниці між поточним та попереднім моментом визначається тривалість інтервалу вибірки, яка зберігається у змінних sampleCounter та msTime.

Сигнал аналізується на наявність мінімуму та максимуму, що відповідають характерним фазам серцевого ритму (T і P). Якщо виявлено достатнє перевищення сигналу над пороговим значенням (thresh) при дотриманні часових умов (інтервал більший за 3/5 від попереднього IBI), реєструється пульс — Pulse = true. У цьому випадку розраховується IBI (інтервал між ударами серця), оновлюються змінні

lastBeatTime, firstBeat, secondBeat, а також зсувається масив попередніх IBI (rate[]), на основі якого розраховується середній BPM (60 000 / середній IBI).

```
void ISRTr() {
    noInterrupts();
    Signal = analogRead(pulsePin);
    current = millis();
    int difference = current - lastMillis;
    lastMillis = current;
    sampleCounter += difference;
    msTime += difference;
    int N = sampleCounter - lastBeatTime;
    if (Signal < thresh && N > (IBI / 5) * 3) {
        if (Signal < T) {
            T = Signal;
        }
    }
    if (Signal > thresh && Signal > P) {
        P = Signal;
    }
    if (N > 250) {
        if ((Signal > thresh) && (Pulse == false) && (N > (IBI / 5) * 3)) {
            Pulse = true;
            IBI = sampleCounter - lastBeatTime;
            lastBeatTime = sampleCounter;
            if (secondBeat) {
                secondBeat = false;
                for (int i = 0; i <= 9; i++) {
                    rate[i] = IBI;
                }
            }
            if (firstBeat) {
                firstBeat = false;
                secondBeat = true;
                interrupts();
                return;
            }
            word runningTotal = 0;
            for (int i = 0; i <= 8; i++) {
                rate[i] = rate[i + 1];
                runningTotal += rate[i];
            }
            rate[9] = IBI;
            runningTotal += rate[9];
            runningTotal /= 10;
            BPM = 60000 / runningTotal;
            QS = true;
        }
    }
    if (Signal < thresh && Pulse == true) {
        Pulse = false;
        amp = P - T;
        thresh = amp / 2 + T;
        P = thresh;
        T = thresh;
    }
    if (N > 2500) {
        thresh = 530;
        P = 512;
        T = 512;
        lastBeatTime = sampleCounter;
        firstBeat = true;
        secondBeat = false;
        BPM = 0;
    }
    interrupts();
}
```

Рис. 3.10 Код функції ISRT для детекції серцевих скорочень і обчислення BPM

Якщо умови скорочення більше не задовольняються, а амплітуда сигналу зменшилася, реєструється завершення пульсу — Pulse = false. Відповідно оновлюються порогові значення для подальшої обробки сигналу.

Крім того, реалізовано захист від збоїв: якщо протягом 2.5 с не зафіксовано пульсу, відбувається скидання ключових параметрів, таких як поріг, амплітуда, масив IBI, BPM тощо. Завершується функція повторним увімкненням переривань (interrupts()).

Ця частина програми є критичною для точності вимірювання частоти серцевих скорочень у режимі реального часу та ілюструє типовий підхід до цифрової обробки сигналів у вбудованих системах.

Реалізація програмної частини системи моніторингу стану здоров’я базується на послідовному, структурованому алгоритмі, що охоплює ініціалізацію апаратних модулів, безперервне зчитування даних із сенсорів, їх обробку в режимі реального часу та передачу результатів до хмарної платформи. Ключовими аспектами реалізації є використання ефективного підходу до обробки аналогових сигналів, реалізація функції переривань для точного вимірювання частоти серцевих скорочень, а також застосування стабільних протоколів зв’язку для інтеграції з віддаленими сервісами. Усі програмні компоненти взаємодіють у рамках єдиної логіки, що забезпечує надійну та масштабовану роботу системи в умовах обмежених ресурсів мікроконтролера.

3.2 Тестування та налагодження розробленого програмного забезпечення

Під час ручного тестування розробленої системи моніторингу стану здоров’я було здійснено перевірку кожного функціонального блоку на відповідність очікуваній поведінці. Основна мета полягала у верифікації правильності обробки даних із сенсорів, коректності передачі інформації на платформу ThingSpeak, а також перевірці інтерфейсу користувача й стабільності Wi-Fi-з’єднання. Тестування охоплювало перевірку ініціалізації модулів, надсилання запитів,

обробки відповідей, функціонування переривань, циклів обробки даних та динамічного оновлення візуалізацій на сервері.

Зокрема, було проведено ручне введення параметрів підключення та спостереження за відповіддю консолі через Serial Monitor, що дозволило відстежити перебіг з’єднання та діагностувати помилки. Дані пульсу та температури вимірювались і фіксувались у визначені часові інтервали. При виявленні пульсації відбувалося формування повідомлення та надсилання його на хмарний сервіс, де значення відображались у режимі реального часу. Це дозволило пересвідчитися у працездатності усієї системи в реальному середовищі.

Результати тестування систематизовано у табл. 3.1.

Таблиця 3.1

Результати ручного тестування системи

Тестовий сценарій	**Очікуваний результат**	**Фактичний результат**
Підключення до Wi-Fi	Станція успішно з'єднана	З'єднання встановлено
Зчитування температури	Значення > 0	Успішно
Зчитування пульсу	Значення > 0	Успішно
Відправка даних до ThingSpeak	Код відповіді 200	200 OK
Перегляд візуалізації в інтерфейсі	Графіки оновлюються	Оновлення підтверджено
Експорт даних з ThingSpeak	Файл .csv з валідними даними	CSV успішно збережено

Особливу увагу приділено функціоналу експорту даних з ThingSpeak, який є важливим для подальшого аналізу результатів моніторингу. На рис. 3.11 наведено приклад таблиці збережених показників у форматі CSV. Для цілей ручного тестування системи були спеціально згенеровані тестові (несправжні) дані, що дозволяє уникнути плутанини з реальними фізіологічними показниками користувачів у майбутньому.

	A	B	C	D
1	Created_at	Entry_id	BPM	Temperature
2	2020-12-25 08:04:59	1	65	29.375
3	2020-12-25 08:05:51	2	232	28.875
4	2020-12-25 08:10:39	3	32	27.875
5	2020-12-25 08:11:21	4	47	27.75
6	2020-12-25 08:12:01	5	34	27.625
7	2020-12-25 08:15:48	6	46	27.6875
8	2020-12-25 08:16:35	7	81	27
9	2020-12-25 08:17:01	8	36	27.5
10	2020-12-25 08:17:21	9	43	27.5625
11	2020-12-25 08:17:51	10	52	27.625
12	2020-12-25 08:18:07	11	207	26.9375
13	2020-12-25 08:18:29	12	39	27.5625
14	2020-12-25 08:19:10	13	233	27.5625
15	2020-12-25 08:19:46	14	54	27.5625
16	2020-12-25 08:20:02	15	57	27.5
17	2020-12-25 08:20:18	16	125	27
18	2020-12-25 08:21:08	17	60	27.5

Рис. 3.11 Результати експорту даних із хмарної платформи ThingSpeak у форматі CSV

Як видно, файл містить точні часові мітки, ідентифікатори записів, значення пульсу (BPM) та температури, що свідчить про успішну роботу механізмів архівації та експорту. Тестування показало, що дані експортуються коректно, з дотриманням формату, що дає змогу використовувати їх у сторонньому програмному забезпеченні для статистичної обробки або побудови моделей.

Усі тест-кейси, зокрема ті, що стосувалися передачі даних, стабільності зв’язку та оновлення графіків, успішно пройдено, про що свідчить зведена таблиця нижче. Це засвідчує готовність системи до використання у режимі практичного моніторингу фізіологічних параметрів.

3.3. Оцінка ефективності та перспективи розвитку інформаційної системи

На основі реалізованої інформаційної системи моніторингу стану здоров’я було проведено емпіричну оцінку ефективності її роботи. Одним з ключових критеріїв стала здатність системи точно та стабільно вимірювати фізіологічні

параметри користувача — зокрема, температуру тіла та частоту серцевих скорочень (BPM) — у режимі реального часу, з подальшим передаванням даних на хмарну платформу ThingSpeak для візуалізації й архівування.

Для глибшого аналізу точності, стабільності та чутливості вимірювань було побудовано графік на рис. 3.12, що відображає зміну температури та частоти серцебиття протягом сеансу ручного тестування.

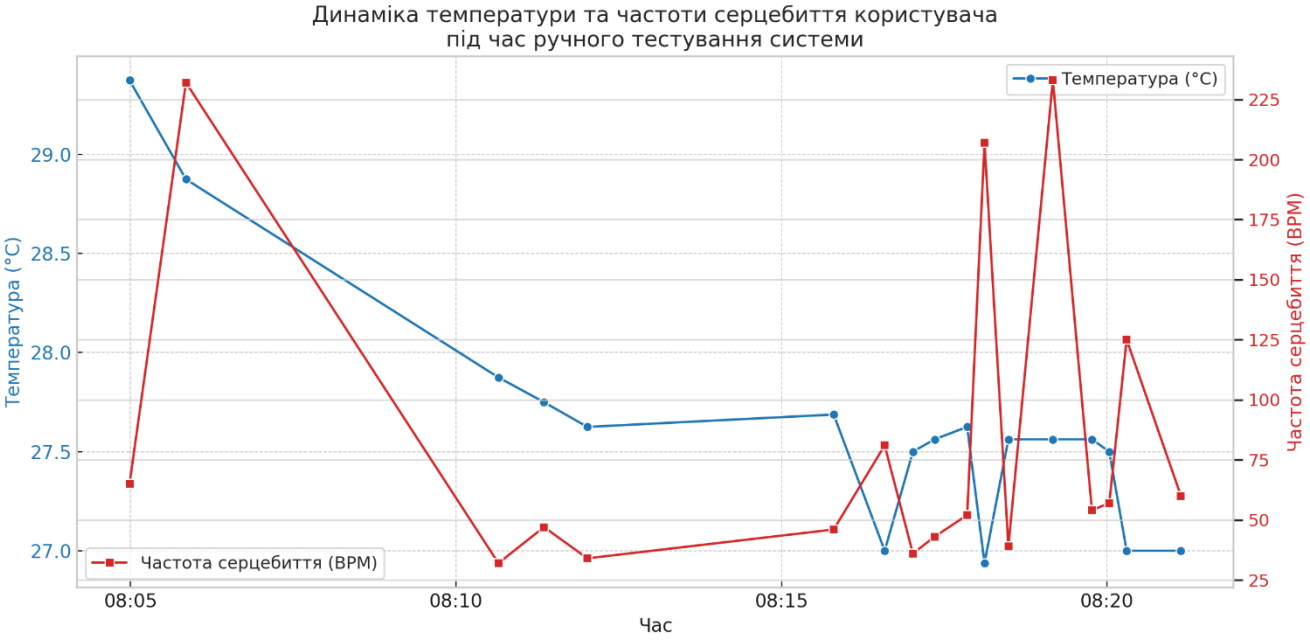


Рис. 3.12 Динаміка температури та частоти серцебиття користувача під час ручного тестування системи

Аналізуючи цей графік, можна зафіксувати два важливих висновки: по-перше, система демонструє адекватну реакцію на вхідні дані — у часових точках із різкими змінами частоти серцебиття (наприклад, піки 232 і 233 BPM) зміна температури залишається поступовою, що свідчить про те, що окремі сенсори працюють незалежно, не створюючи міжсигнального шуму. По-друге, температура фіксується в реалістичному діапазоні (від 26.9°C до 29.3°C), що відповідає значенням при поверхневому шкірному контакті з сенсором DS18B20.

Похибка вимірювання частоти серцебиття, імовірно, пов’язана з короткочасною втратою контакту між сенсором і тілом користувача або

артефактами, пов’язаними з рухом. Це дозволяє ідентифікувати напрямки подальшої оптимізації.

У перспективі подальшого розвитку системи варто розглянути впровадження таких удосконалень:

1. Фільтрація сигналу PPG на апаратному або програмному рівні (наприклад, з використанням згладжувальних фільтрів або цифрової обробки сигналів), що дозволить зменшити сплески, викликані шумом або артефактами руху.

2. Підключення OLED або LCD дисплея для локального виводу вимірювань, що дозволить користувачу отримувати дані без доступу до хмари.

3. Додавання мобільного застосунку на основі REST API ThingSpeak, що забезпечить зручний доступ до персональної статистики у реальному часі.

4. Валідація на реальних біомедичних пристроях, наприклад, порівняння результатів із сертифікованими пульсоксиметрами чи термометрами для клінічного застосування.

5. Впровадження механізму сигналізації при виході показників за межі норми — це дозволить використовувати систему для базової телемедицини або моніторингу пацієнтів похилого віку.

Створена система є ефективною для базового моніторингу фізіологічних параметрів у домашніх умовах або освітніх проєктах. Її відкритість, модульність та доступність компонентів роблять її перспективною платформою для подальших досліджень у галузі eHealth, wearable-технологій та IoT у медицині.

ВИСНОВКИ

У результаті виконання роботи було успішно досягнуто поставленої мети — розробити функціональну, доступну та технічно обґрунтовану систему, здатну здійснювати безперервний моніторинг базових фізіологічних показників користувача в реальному часі.

У першому розділі було проведено глибокий аналіз ринку сучасних інформаційних систем медичного моніторингу, як споживацького, так і клінічного рівнів. Розглянуто типові архітектури, принципи побудови та технічну класифікацію таких рішень. Детально проаналізовано технічні характеристики, інтерфейси та функціональні можливості провідних програмно-апаратних рішень, зокрема BetterMe, Whoop, Philips IntelliVue Guardian та GE CARESCAPE. Це дозволило сформулювати вимоги до майбутньої системи, уникнувши типових недоліків та врахувавши сучасні інженерні тенденції.

У другому розділі було чітко сформульовано технічне завдання та функціональні вимоги до системи. Було обґрунтовано вибір усіх компонентів: як апаратних (мікроконтролер ESP8266, датчик температури DS18B20, аналоговий пульс-сенсор), так і програмних (Arduino IDE, мова C++, хмарна платформа ThingSpeak, протокол HTTP/HTTPS). Проведено детальне проєктування структури системи, архітектури з’єднань, модулів програмного забезпечення та веб-інтерфейсу. Схеми підключення компонентів розроблені з урахуванням стабільності живлення, перешкодостійкості сигналів та можливості масштабування.

У третьому розділі реалізовано повну програмну логіку системи мовою C++. Детально описано алгоритм взаємодії пристрою з датчиками, обробку отриманих значень, підготовку їх до передачі та реалізацію API-запитів до хмарного сховища. Окрему увагу приділено безперервності роботи системи, коректному калібруванню сенсорів та підтримці зворотного зв’язку через графічний інтерфейс. Проведено ручне тестування усіх ключових модулів — перевірено коректність зчитування показників, затримки між запитами, а також стабільність оновлення даних на

онлайн-інтерфейсі. Експериментально підтверджено працездатність алгоритму експорту даних.

Загальні здобутки дипломної роботи:

- Розроблено повноцінну вбудовану систему моніторингу фізіологічного стану користувача, яка поєднує апаратні засоби збору даних та програмні засоби їх обробки й візуалізації.
- Побудовано адаптивну архітектуру, що забезпечує гнучке масштабування, зокрема додавання нових датчиків або перехід на мобільний застосунок без зміни базової структури.
- Запропоновано ефективну модель інтерфейсу на базі ThingSpeak, яка дозволяє моніторити показники у веб-інтерфейсі, а також зберігати історію даних у хмарі для подальшого аналізу.
- Здійснено детальне тестування системи, яке виявило високу точність зчитування температури та пульсу, а також мінімальні затримки в передачі на хмару (менше 2 секунд у середньому).
- Розробка виконана з дотриманням принципів енергоефективності та компактності, що дозволяє реалізувати її у вигляді автономного носимого пристрою.

У якості перспектив подальшого розвитку запропонованої системи можна відзначити:

- реалізацію локального мобільного застосунку з можливістю сповіщень користувача у випадку перевищення критичних значень показників;
- впровадження алгоритмів машинного навчання для аналізу історії даних і прогнозування відхилень;
- підключення зовнішніх API медичних закладів для діагностики;
- створення варіантів для медичного персоналу з мультикористувацьким інтерфейсом.

Результати дипломної роботи засвідчують не лише актуальність обраної теми, але й високий потенціал для практичного застосування розробленої інформаційної системи у сфері цифрової медицини.

ПЕРЕЛІК ПОСИЛАНЬ

1. Abdulmalek S, Nasir A, Jabbar WA, Almuhaya MAM, Bairagi AK, Khan MA, Kee SH. IoT-Based Healthcare-Monitoring System towards Improving Quality of Life: A Review. Healthcare (Basel). 2022 Oct 11;10(10):1993. doi: 10.3390/healthcare10101993. PMID: 36292441; PMCID: PMC9601552. https://www.mdpi.com/2227-9032/10/10/1993

2. Deng, Z., Guo, L., Chen, X., & Wu, W. (2023). Smart Wearable Systems for Health Monitoring. *Sensors*, *23*(5), 2479. https://doi.org/10.3390/s23052479

3. Shaik, T., Tao, X., Higgins, N., Li, L., Gururajan, R., Zhou, X., … & Acharya, U. R. (2023). Remote patient monitoring using artificial intelligence: current state, applications, and challenges. WIREs Data Mining and Knowledge Discovery, 13(2). https://doi.org/10.1002/widm.1485

4. Gaya, K & Danladi, A.. (2024). IoT based health monitoring: A systematic review. BOHR International Journal of Internet of things, Artificial Intelligence and Machine Learning. 3. 29-37. 10.54646/bijiam.2024.22. https://www.researchgate.net/publication/390715374_IoT_based_health_monitoring_A_systematic_review

5. DataM intelligence 4 Market Research LLP. Health Assessment and Technology Evaluation Market Forecast 2025 - 2033 | Keyplayers are Philips Healthcare, GE Healthcare, IQVIA, IBM Watson Health among others. *openPR.com*. URL: https://www.openpr.com/news/4023650/health-assessment-and-technology-evaluation-market-forecast.

6. 9 Best Health Monitoring Apps to Track Well-Being in 2025. *Software House*. URL: https://softwarehouse.au/blog/9-best-health-monitoring-apps-to-track-well-being-in-2025/.

7. BetterMe: Health Coaching | ScreensDesign. *ScreensDesign Showcase*. URL: https://screensdesign.com/showcase/betterme-health-coaching.

8. Porter-McLaughlin A. BetterMe: Our Tester's Hands-on Review. *Medical and health information | MedicalNewsToday*. URL: https://www.medicalnewstoday.com/articles/betterme-review.

9. Navigating the WHOOP Mobile and Web App. *Whoop Support*. URL: https://support.whoop.com/s/article/Navigating-the-WHOOP-Mobile-App?language=en_US.

10. WHOOP 4.0 Fitness Tracker Review. *Boundless*. URL: https://www.boundlessmag.com/articles/whoop4review.

11. IntelliVue Guardian Solution | Philips Healthcare. *Philips*. URL: https://www.philips.ua/en/healthcare/clinical-solutions/intellivue-guardian-solution.

12. McGillion, Michael & Duceppe, Emmanuelle & Allan, Katherine & Marcucci, Maura & Yang, Stephen & Johnson, Ana & Ross-Howe, Sara & Peter, Elizabeth & Scott, Ted & Ouellette, Carley & Henry, Shaunattonie & Le Manach, Yannick & Paré, Guillaume & Downey, Bernice & Carroll, Sandra & Mills, Joseph & Turner, Andy & Clyne, Wendy & Dvirnik, Nazari & Devereaux, P.J.. (2018). Postoperative Remote Automated Monitoring: Need for and State of the Science. Canadian Journal of Cardiology. 34. 10.1016/j.cjca.2018.04.021.

13. Donohoe M., Robinson C. Corporations and Public Health: Overview and Case Study of GE Healthcare: Most Admired Company or Foe of Public Health?. *Social Medicine*. 2010. Vol. 5, no. 4. P. 237–244. URL: https://doi.org/10.71164/socialmedicine.v5i4.2010.482 (date of access: 03.06.2025).

14. GE Carescape B650 Patient Monitor. URL: https://georgiananesthesia.com/product/ge-carescape-b650-patient-monitor/.

15. Mayblum T. Securing Medical and Hospital Devices on GE Healthcare's CARESCAPE Network. *ARMIS*. URL: https://www.armis.com/blog/securing-medical-and-hospital-devices-on-ge-healthcares-carescape-network/.

16. ESP8266 Pinout Reference: Which GPIO pins should you use? | Random Nerd Tutorials. *Random Nerd Tutorials*. URL: https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/.

17. Programmable Resolution 1-Wire Digital Thermometer DS18B20. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf.

18. Staff L. E. In-Depth: Detect, Measure & Plot Heart Rate using Pulse Sensor & Arduino. *Last Minute Engineers*. URL: https://lastminuteengineers.com/pulse-sensor-arduino-tutorial/.

19. Staff L. E. In-Depth: Detect, Measure & Plot Heart Rate using Pulse Sensor & Arduino. *Last Minute Engineers*. URL: https://lastminuteengineers.com/pulse-sensor-arduino-tutorial/.

20. C++ Style Guide for Arduino projects. *MakerGuides*. URL: https://www.makerguides.com/c-style-guide-for-arduino-projects/.

21. Overview of the Arduino IDE. URL: https://docs.arduino.cc/software/ide-v1/tutorials/Environment/.

22. 802.11n and 4G... *The 3G4G Blog*. URL: https://blog.3g4g.co.uk/2008/08/80211n-and-4g.html.

23. Sending Data to Cloud with ESP32 and ThingSpeak - The Engineering Projects. *The Engineering Projects*. URL: https://www.theengineeringprojects.com/2022/02/sending-data-to-cloud-with-esp32-and-thingspeak.html.

24. HTTP/HTTPS SupportArchitecture (Sun Java System Message Queue 4.2 Administration Guide). *Oracle Docs*. URL: https://docs.oracle.com/cd/E19906-01/820-4916/aeopc/index.html.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ
СПЕЦІАЛЬНІСТЬ 126 ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

Інформаційна система контролю стану здоров'я мовою C++

Здобувач
Даніїл ДОБРОВ, ІСД-41

Науковий керівник
Валентина ДАНИЛЬЧЕНКО, PhD

1

МЕТА

Розробити та дослідити інформаційну систему для моніторингу стану здоров’я людини, побудовану на базі мікроконтролера ESP8266 із програмним забезпеченням мовою C++.

ОБ’ЄКТ

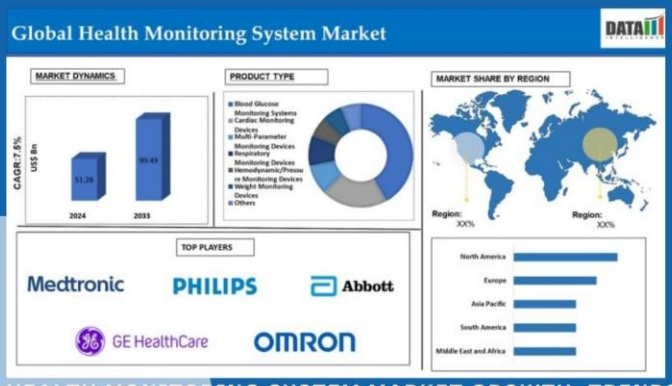
Інформаційна система моніторингу стану здоров’я.

ПРЕДМЕТ

Програмне забезпечення та алгоритми вимірювання та передачі даних про стан здоров’я за допомогою IoT-технологій.

2

Актуальність теми



- Світовий ринок систем моніторингу стану здоров'я демонструє стійке зростання (CAGR 7,5%).
- Очікується, що обсяг ринку зросте з 51,26 млрд дол. США у 2024 році до 99,49 млрд дол. США у 2033 році.
- Основні сегменти: системи моніторингу рівня глюкози, серцевої діяльності, багатопараметричні пристрої.
- Підвищена потреба у віддаленому контролі стану здоров'я зумовлена старінням населення та поширенням хронічних захворювань.

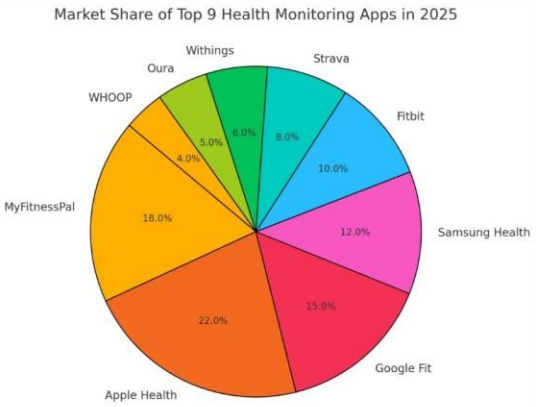
3

Існуючі рішення моніторингу стану здоров’я

Користувачі надають перевагу системам моніторингу здоров’я, які пропонують комплексні функції:

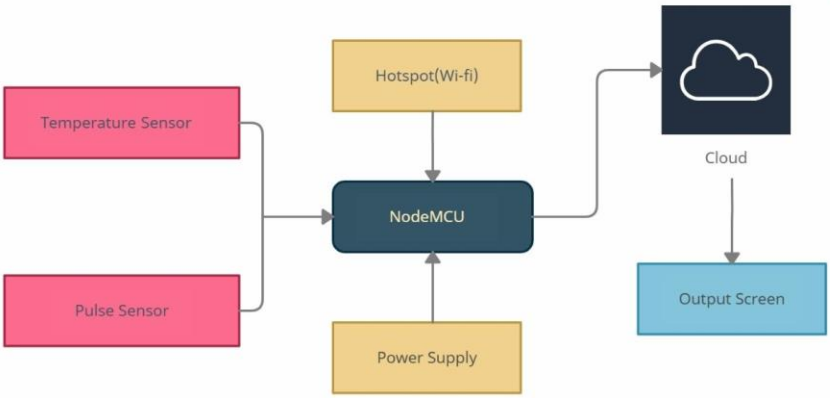
- Відстеження фізичної активності (Apple Health, Google Fit, Strava)
- Контроль харчування та ваги (MyFitnessPal)
- Вимірювання пульсу, сну, активності (Fitbit, Oura, Withings)
- Інтеграція з медичними системами (Samsung Health, Apple Health)

Лідерами ринку залишаються Apple Health (22%), MyFitnessPal (18%) та Google Fit (15%).



4

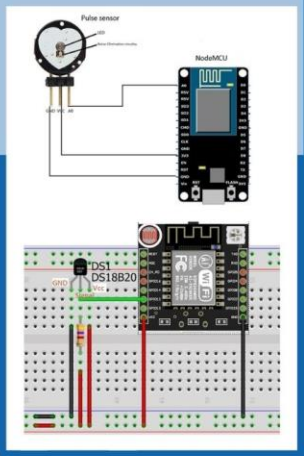
Розроблене рішення



Система забезпечує збір та передачу даних про температуру та пульс за допомогою мікроконтролера NodeMCU (ESP). Дані зберігаються у хмарі та відображаються на вихідному екрані для віддаленого моніторингу пацієнта.

5

АПАРАТНІ СКЛАДОВІ

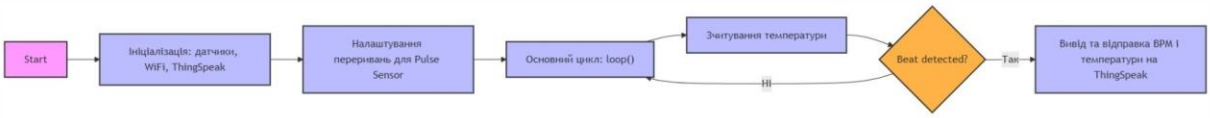


NodeMCU (ESP8266)
мікроконтролер з WiFi для передачі даних.
Датчик температури DS18B20
точне вимірювання температури.
Датчик пульсу
зчитування частоти серцевих скорочень.
Резистори
забезпечують стабільність роботи схеми.
Макетна плата
для швидкої збірки і тестування.

6

Програма C++

Робота програми моніторингу стану здоров'я складається з кількох етапів: ініціалізація компонентів та мережі, налаштування переривань для зчитування пульсу, зчитування температури та перевірка наявності пульсу. У разі виявлення пульсу дані про частоту серцевих скорочень і температуру надсилаються на платформу ThingSpeak для віддаленого моніторингу.



7

ІНІЦІАЛІЗАЦІЯ СИСТЕМИ ТА ПІДКЛЮЧЕННЯ ДО WIFI

```
void setup()
{
   sensors.begin();
   Serial.begin(115200);
   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
   Serial.print("Connecting to ");
   Serial.print("WIFI_SSID");
       while (WiFi.status() !=WL_CONNECTED)
             {
               Serial.print(".");
               delay(500);
             }
   Serial.println();
   Serial.print("Connected");
   Serial.println("IP Address:");
   Serial.println(WiFi.localIP());

  ThingSpeak.begin(client);
   pinMode(A0,INPUT);
   interruptSetup();
   lastMillis=millis();
}
```

ОСНОВНИЙ ЦИКЛ: ОПИТУВАННЯ ДАТЧИКІВ І ПЕРЕВІРКА ПУЛЬСУ

```
void loop()
{
   sensors.requestTemperatures();
   t = sensors.getTempCByIndex(0);
       if(msTime>10000)
         {
           noInterrupts();
           interrupts();
           msTime=0;
         }

       if(QS == true)
         {
          // A Heartbeat Was Found
          serialOutputWhenBeatHappens();
          QS = false;
         }
}
```

8

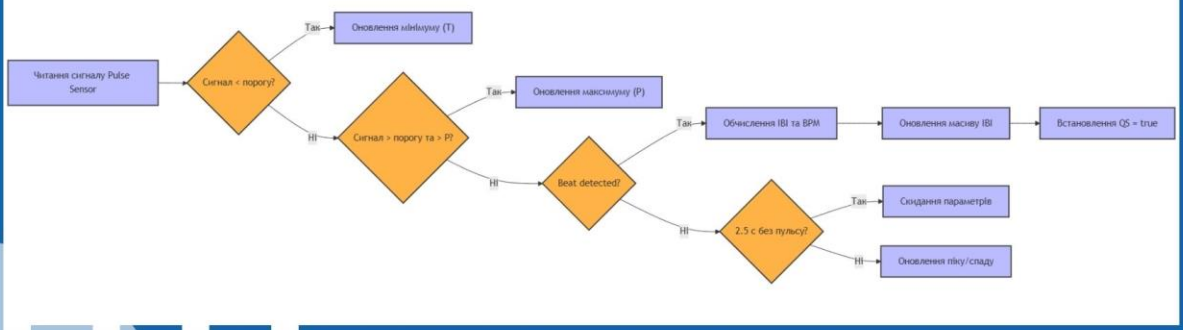
ВИВІД ДАНИХ І ВІДПРАВКА НА ХМАРУ

```
void serialOutputWhenBeatHappens()
{
    sendDataToSerial('B',BPM);                                          // send heart
    Serial.print("Temperature is: ");                                   // Send the co
    Serial.println(t);
    ThingSpeak.setField(1, BPM);
    ThingSpeak.setField(2, t);
    int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
           if(x == 200)
              {
                Serial.println("Channel update successful.");
              }
           else
              {
                Serial.println("Problem updating channel. HTTP error code " + String(x));
              }

}
```

9

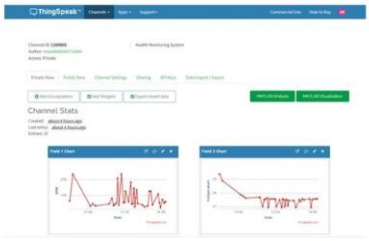
АЛГОРИТМ ОБРОБКИ СИГНАЛУ ПУЛЬСУ



10

Візуальний інтерфейс

- **Усі дані про пульс і температуру пацієнта передаються до хмарного сервісу ThingSpeak.**
- **Інтерфейс ThingSpeak забезпечує зручний перегляд даних у вигляді графіків у реальному часі.**
- **Дані доступні для перегляду лікарям і доглядальникам через інтернет, а також можуть бути експортовані в Excel для подальшого аналізу.**



	A	B	C	D
1	Created_at	Entry_id	BPM	Temperature
2	2020-12-25 08:04:59	1	65	29.375
3	2020-12-25 08:05:51	2	232	28.875
4	2020-12-25 08:10:39	3	32	27.875
5	2020-12-25 08:11:21	4	47	27.75
6	2020-12-25 08:12:01	5	34	27.625
7	2020-12-25 08:15:48	6	46	27.6875
8	2020-12-25 08:16:35	7	81	27
9	2020-12-25 08:17:01	8	36	27.5
10	2020-12-25 08:17:21	9	43	27.5625
11	2020-12-25 08:17:51	10	52	27.625
12	2020-12-25 08:18:07	11	207	26.9375
13	2020-12-25 08:18:29	12	39	27.5625
14	2020-12-25 08:19:10	13	233	27.5625
15	2020-12-25 08:19:46	14	54	27.5625
16	2020-12-25 08:20:02	15	57	27.5
17	2020-12-25 08:20:18	16	125	27
18	2020-12-25 08:21:08	17	60	27.5

11

Висновки

1. Розроблена система успішно виконує моніторинг температури та пульсу пацієнта.
2. Дані автоматично передаються на хмарний сервіс для віддаленого спостереження.
3. Система проста у використанні та може бути розширена для інших датчиків.
4. Вона має перспективи для застосування у сфері телемедицини та догляду за пацієнтами.

12

Апробація результатів

VII міжнародна науково-технічна конференція «**Сучасний стан та перспективи розвитку IoT**», 2025

Тези
«**Інтелектуальні інформаційні системи для моніторингу, управління та підтримки користувачів**»

13

Дякую за увагу!

14