МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних наук та прикладної математики

"До захисту допущена"
Зав. кафедри комп'ютерних наук та прикладної математики
д.т.н., професор Турбал Ю. В.
«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

«Розробка прототипу IoT-системи моніторингу здоров'я»

Виконала: Ковальчук Карина Ігорівна _____
(підпис)

група ІПЗ-41

Керівник: доц., к.т.н. Климюк Ю. Є. _____
(підпис)

Рівне – 2025

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 57 с., 14 рисунків, 2 таблиці, 11 джерел.

Мета роботи: розробити прототип IoT-системи для моніторингу основних фізіологічних показників здоров'я людини в реальному часі та створити мобільний застосунок для візуалізації отриманих даних.

Об’єкт дослідження – процеси збору, передачі, обробки та візуалізації фізіологічних параметрів людини за допомогою IoT-технологій.

Предмет дослідження – розробка програмно-апаратного комплексу для безперервного моніторингу фізіологічних показників з подальшим відображенням даних у мобільному застосунку.

Методи розробки базуються на використанні мікроконтролерів Arduino, MAX30102, DS18B20, WeMos D1 Wi-Fi, фреймворку React Native, Node.js + Express.js для backend-сервера та бази даних MongoDB.

Розроблено прототип IoT-системи моніторингу здоров'я, що дозволяє безперервно збирати дані про температуру тіла, частоту серцевих скорочень та насиченість крові киснем. Створено власний мобільний застосунок, який забезпечує візуалізацію поточних та історичних даних, а також надсилання сповіщень при критичних відхиленнях. Реалізовано механізми авторизації та захисту даних.

Ключові слова: ІОТ-СИСТЕМА, МОНІТОРИНГ ЗДОРОВ’Я, ФІЗІОЛОГІЧНІ ПАРАМЕТРИ, ARDUINO, REACT NATIVE, МОБІЛЬНИЙ ЗАСТОСУНОК, ТЕЛЕМЕДИЦИНА.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

ЧСС – Частота серцевих скорочень

API – Інтерфейс прикладного програмування

Backend – серверна частина проекту

Frontend – клієнтська частина проекту

IoT – Інтернет речей (Internet of Things)

IoMT – Internet of Medical Things

JSON – формат обміну даними у вигляді тексту

SpO_2 – Насиченість крові киснем

UI – користувацький інтерфейс

UX – досвід користувача

ВСТУП

Швидкий розвиток Інтернету речей (IoT) відкриває нові можливості в охороні здоров'я, зокрема для персоналізованого та безперервного моніторингу фізіологічних параметрів людини. Традиційні методи контролю часто не дозволяють оперативно реагувати на раптові зміни показників, що особливо критично для людей із хронічними захворюваннями, літніх людей, спортсменів, а також у контексті зростаючої потреби в телемедицині та віддаленому нагляді [11].

Сучасні дослідження у сфері IoT для моніторингу здоров'я демонструють значний прогрес у розробці сенсорів, методів передачі даних та мобільних платформ. Однак, залишаються виклики, пов'язані з інтеграцією технологій, забезпеченням надійності, точності та конфіденційності даних, а також розробкою доступних та масштабованих рішень. Існуючі комерційні системи часто дорогі або мають обмежений функціонал. Необхідність у таких системах, що дозволяють отримувати дані в реальному часі та своєчасно реагувати на критичні зміни, підкреслює актуальність обраної теми.

Актуальність теми зумовлена низкою факторів:

1. Зростаюча потреба в безперервному моніторингу: демографічні зміни (старіння населення), зростання поширеності хронічних захворювань та необхідність контролю стану спортсменів вимагають постійного нагляду за життєво важливими показниками.
2. Розвиток телемедицини: пандемія COVID-19 прискорила розвиток телемедичних послуг, де віддалений моніторинг є ключовим компонентом для надання якісної медичної допомоги без фізичного контакту.
3. Персоналізація охорони здоров'я: індивідуалізований підхід до здоров'я, що базується на безперервному зборі даних, дозволяє своєчасно коригувати лікування та підвищувати якість життя.

4. Виявлення аномалій та попередження критичних станів: автоматизований аналіз даних з можливістю виявлення відхилень від норми дозволяє оперативно реагувати на потенційно небезпечні ситуації.

На основі зазначеного, метою кваліфікаційної роботи є розробка прототипу IoT-системи для моніторингу основних фізіологічних показників здоров'я людини в реальному часі та створення відповідного мобільного застосунку для візуалізації даних, збереження історії вимірювань та надсилання сповіщень.

Для досягнення поставленої мети було визначено наступні завдання:

1. Проаналізувати сучасні підходи та технології, що застосовуються для розробки IoT-систем моніторингу здоров'я.
2. Обрати оптимальні апаратні компоненти (сенсори, мікроконтролер, модуль зв'язку) для реалізації системи.
3. Розробити архітектуру програмно-апаратного комплексу IoT-системи моніторингу здоров'я.
4. Створити та реалізувати програмне забезпечення для мікроконтролера для збору та передачі даних.
5. Розробити backend-сервер для прийому, обробки, зберігання даних та забезпечення API-зв'язку з мобільним застосунком.
6. Спроектувати та реалізувати мобільний застосунок для візуалізації показників, збереження історії та надсилання сповіщень.
7. Провести тестування розробленої системи та оцінити її функціональність і ефективність.

Об’єкт дослідження – процеси збору, передачі, обробки та візуалізації фізіологічних параметрів людини за допомогою IoT-технологій.

Предмет дослідження – програмно-апаратний комплекс для безперервного моніторингу фізіологічних показників з подальшим відображенням даних у мобільному застосунку.

Наукова новизна роботи полягає в розробці та програмній реалізації інтегрованого прототипу IoT-системи моніторингу здоров'я. Ця система поєднує доступні апаратні компоненти з власним backend-сервером та кросплатформним мобільним застосунком, забезпечуючи гнучкість та масштабованість. Розроблений мобільний застосунок не лише візуалізує дані, а й інтегрує функціонал збереження історії та своєчасного оповіщення про критичні відхилення, що є ключовим для персоналізованого моніторингу.

Практична значущість результатів кваліфікаційної роботи полягає у створенні готового до впровадження прототипу IoT-системи, який може бути використаний для індивідуального моніторингу здоров'я, віддаленого нагляду за пацієнтами, а також як основа для подальших досліджень у сфері телемедицини та носимих пристроїв. Розроблена система демонструє можливість побудови функціональних та доступних рішень для моніторингу життєво важливих показників без значних фінансових витрат. Програмна реалізація та архітектурні рішення можуть слугувати зразками для розробки аналогічних систем.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ ІСНУЮЧИХ РІШЕНЬ

1.1. Концепція Інтернету речей в охороні здоров'я та її актуальність

Широке впровадження технологій Інтернету речей (IoT) відкриває нові, інноваційні підходи до багатьох сфер життя, зокрема до охорони здоров'я. Сучасна медицина все більше потребує переходу від ручного лікування до проактивного моніторингу та превентивної діагностики. Цей можливо завдяки здатності IoT-пристроїв збирати та передавати великі обсяги фізіологічних даних у реальному часі.

IoT у сфері охорони здоров'я (IoMT – Internet of Medical Things) – це мережа взаємопов'язаних медичних пристроїв, сенсорів, програмного забезпечення та сервісів, які збирають, аналізують і обмінюються даними про стан здоров'я пацієнта. Це дозволяє здійснювати безперервний моніторинг життєво важливих показників поза межами медичних закладів, що є критично важливим для пацієнтів із хронічними захворюваннями, людей похилого віку, спортсменів та в умовах зростаючої популярності телемедицини.

Актуальність розвитку таких систем зумовлена низкою факторів:

- Зростання захворюваності на хронічні хвороби: Безперервний моніторинг дозволяє відстежувати динаміку стану пацієнтів та своєчасно виявляти критичні відхилення.
- Старіння населення: Збільшення частки літніх людей вимагає ефективних рішень для віддаленого догляду та контролю здоров'я.
- Розвиток телемедицини: IoT-системи є основою для надання якісних телемедичних послуг, зменшуючи потребу у фізичних візитах до лікаря.
- Персоналізація медицини: Індивідуальний збір даних дозволяє створювати персоналізовані програми лікування та профілактики.

Традиційні методи моніторингу, що базуються на періодичних вимірюваннях у клінічних умовах, не можуть забезпечити повної картини стану здоров'я пацієнта. Існує нагальна потреба в системах, які можуть постійно збирати дані, аналізувати їх та попереджати про потенційні ризики, що створює проблемну ситуацію, обрану для вивчення в даній роботі.

1.2. Огляд існуючих рішень та технологій для моніторингу здоров'я на базі IoT

Аналіз існуючих рішень у галузі IoT-систем моніторингу здоров'я виявляє різноманіття підходів до реалізації програмно-апаратних комплексів. Більшість таких систем складається з трьох основних компонентів: апаратний модуль для збору даних (сенсори та мікроконтролери), комунікаційний канал для передачі даних та програмний модуль для обробки, зберігання та візуалізації інформації (серверна частина та клієнтський застосунок).

Апаратні компоненти на ринку представлено широким спектром біометричних сенсорів. Для моніторингу пульсу та насиченості крові киснем (SpO_2) популярними є датчики на основі фотоплетизмографії, такі як MAX30102, який зарекомендував себе як компактний та відносно недорогий модуль з високою точністю [9]. Для вимірювання температури тіла часто використовуються цифрові датчики, наприклад, DS18B20 [9], що відрізняється простотою підключення та достатньою точністю для домашнього використання. Як обчислювальні модулі широко застосовуються мікроконтролери платформи Arduino (наприклад, Arduino Uno, Arduino Mega), які забезпечують гнучкість у програмуванні та легкість інтеграції з різними сенсорами [4–7].

Для передачі даних від IoT-пристроїв до хмарних або локальних серверів використовуються різні технології, включаючи Wi-Fi, Bluetooth, Zigbee, LoRaWAN тощо. Wi-Fi плати, такі як WeMos D1 mini (на базі ESP8266), є популярним вибором завдяки їхній доступності, низькій вартості, компактності та можливості прямого підключення до мережі Інтернет. Ці плати поєднують

функціональність мікроконтролера та Wi-Fi модуля, що спрощує розробку та інтеграцію.

- Для прийому, обробки та зберігання даних з IoT-пристроїв необхідний надійний backend-сервер. Популярними технологіями для створення RESTful API є Node.js у поєднанні з фреймворком Express.js [2], що забезпечує високу продуктивність та масштабованість.
- Для зберігання великих обсягів часових рядів даних, характерних для систем моніторингу, часто обирають NoSQL бази даних, такі як MongoDB [3]. Документоорієнтована модель MongoDB гнучко адаптується до різноманітних типів даних та забезпечує високу швидкість запису та читання, що є перевагою для IoT-додатків.
- Для візуалізації даних та взаємодії з користувачем розробляються мобільні або веб-застосунки. React Native є кросплатформним фреймворком [1], що дозволяє створювати нативні мобільні застосунки для iOS та Android з єдиної кодової бази, що значно прискорює розробку та зменшує витрати.

Більшість представлених на ринку та в науковій літературі IoT-систем моніторингу здоров'я зосереджуються на зборі та базовій візуалізації даних. Проте, багато з них мають певні обмеження:

1) доступність та вартість: комерційні рішення часто є дорогими, а багато open-source проєктів можуть бути неоптимізованими або складними для налаштування;

2) інтеграція компонентів: не завжди забезпечується безшовна інтеграція між апаратною частиною, бекендом та мобільним застосунком, що може призводити до затримок або втрати даних;

3) функціональність мобільних застосунків: часто обмежується лише відображенням поточних даних без можливості аналізу історії, налаштування сповіщень чи підтримки користувацьких профілів;

4) безпека даних: недостатня увага приділяється питанням авторизації та захисту персональних медичних даних під час передачі та зберігання.

Враховуючи вищезазначені недоліки, є очевидна потреба в розробці інтегрованої IoT-системи моніторингу здоров'я, яка б поєднувала доступність апаратних компонентів з надійною програмною архітектурою та зручним мобільним застосунком. Особливу увагу варто приділити розробці власного backend-сервера для забезпечення гнучкості у зберіганні та обробці даних, а також використанню кросплатформних фреймворків, таких як React Native, для швидкої розробки клієнтського застосунку з розширеним функціоналом, включаючи історію вимірювань, персоналізовані сповіщення та базові механізми безпеки. Такий підхід дозволить створити прототип, що демонструватиме високу функціональність та потенціал для подальшого масштабування.

1.3. Вимоги до програмного продукту та основні завдання

На основі проведеного аналізу предметної області та існуючих рішень, можна сформулювати ключові вимоги до розроблюваного прототипу IoT-системи моніторингу здоров'я.

Системні вимоги:

1) надійність збору даних: система повинна забезпечувати точний та безперервний збір фізіологічних показників (температура тіла, ЧСС, SpO_2) за допомогою відповідних сенсорів;
2) бездротова передача даних: дані мають передаватися від пристрою до backend-сервера за допомогою Wi-Fi з'єднання;

3) зберігання даних: система повинна забезпечувати надійне зберігання отриманих даних у базі даних з можливістю доступу до історичних записів;
4) візуалізація даних: мобільний застосунок має надавати зручний інтерфейс для відображення поточних та історичних даних у зрозумілому форматі (наприклад, графіки, таблиці);
5) сповіщення: система повинна мати функціонал сповіщень у разі виходу показників за попередньо встановлені критичні межі;
6) авторизація користувачів: має бути реалізований механізм авторизації для забезпечення доступу лише зареєстрованим користувачам;
7) масштабованість: архітектура системи повинна передбачати можливість розширення функціоналу та підключення нових типів сенсорів у майбутньому.

Функціональні завдання, які повинен вирішувати програмний продукт:

– збір даних з сенсорів MAX30102 (пульс, SpO_2) та DS18B20 (температура);
– локальне відображення показників на LCD-екрані;
– передача зібраних даних з мікроконтролера Arduino Mega (через ESP8266) на backend-сервер;
– прийом та обробка даних backend-сервером (Node.js + Express.js);
– збереження даних у базі даних MongoDB;
– реалізація API для взаємодії мобільного застосунку з backend-сервером;
– розробка мобільного застосунку (React Native) для відображення поточних значень показників;
– реалізація функціоналу перегляду історії вимірювань у мобільному застосунку;
– надсилання push-сповіщень або іншого типу оповіщень користувачеві при виявленні критичних відхилень у показниках;
– реалізація базового механізму авторизації/реєстрації користувачів.

1.4. Роль Arduino у IoT-системах моніторингу здоров'я

Arduino є відкритою апаратно-програмною платформою для розробки електронних пристроїв, що базується на простих мікроконтролерах та інтуїтивно зрозумілому середовищі програмування. Ця платформа забезпечує широкий набір інструментів для розробників, дозволяючи створювати різноманітні інтерактивні пристрої, включаючи системи моніторингу здоров'я.

1.4.1. Основні характеристики платформи Arduino

На апаратному рівні платформа Arduino відрізняється наступними ключовими особливостями:

1. Мікроконтролери: Плати Arduino використовують різні мікроконтролери, такі як ATmega328 (Arduino Uno) або ATmega2560 (Arduino Mega), які забезпечують достатню обчислювальну потужність для збору та попередньої обробки фізіологічних даних.
2. Розширюваність: Плати Arduino мають численні порти (цифрові та аналогові) для підключення сенсорів (наприклад, MAX30102, DS18B20), дисплеїв (LCD-модуль) та інших периферійних пристроїв. Це робить їх ідеальними для створення кастомних систем моніторингу, де потрібне інтегрування кількох датчиків.

Ці особливості роблять Arduino надзвичайно гнучкою платформою для розробки прототипів IoT-систем моніторингу здоров'я, адаптованих до конкретних потреб проекту [4-7].

Програмне забезпечення:

1. Arduino IDE: Інтегроване середовище розробки (IDE) Arduino є простим та інтуїтивно зрозумілим інструментом для написання коду на мові програмування, подібній до C++, та завантаження його на плату мікроконтролера.

2. Бібліотеки: Велика кількість офіційних та сторонніх бібліотек дозволяє легко інтегрувати різні компоненти і функції, такі як зчитування даних з сенсорів, керування дисплеями, робота з модулями зв'язку (наприклад, через AT-команди для ESP8266/WeMos D1 mini) [8] [10].

Така багата екосистема програмних інструментів суттєво прискорює процес розробки та дозволяє зосередитися на логіці програми, а не на низькорівневих деталях апаратного забезпечення.

1.4.2. Переваги використання Arduino у системах моніторингу здоров'я

Завдяки своїм ключовим перевагам, Arduino є оптимальним вибором для розробки IoT-систем моніторингу здоров'я, серед яких можна виділити:

1) доступність та простота використання: arduino забезпечує простий і доступний спосіб створення прототипів IoT-систем моніторингу здоров'я навіть для розробників-початківців. Інтуїтивно зрозуміле середовище програмування та наявність великої кількості навчальних матеріалів роблять Arduino популярним вибором серед аматорів та професіоналів;
2) гнучкість та кастомізація: завдяки відкритій архітектурі та підтримці великої кількості додаткових модулів і сенсорів, розробники можуть створювати унікальні та інноваційні системи, які відповідають їхнім специфічним потребам моніторингу. Наприклад, можна додавати нові датчики для розширення функціоналу системи (вимірювання ЕКГ, артеріального тиску тощо);
3) інтеграція з різними платформами: arduino можна легко інтегрувати з різними комунікаційними модулями (Wi-Fi, Bluetooth) для передачі даних на смартфони, веб-сервери або хмарні платформи. Це робить Arduino універсальним інструментом для розробки пристроїв, сумісних з широким спектром програмного забезпечення для обробки даних;

4) спільнота та підтримка: велика та активна спільнота користувачів Arduino забезпечує широкий спектр підтримки та обміну знаннями. На форумах [9], у блогах та на відеоканалах можна знайти багато прикладів проєктів, які можна використовувати як основу для власних розробок, що значно прискорює процес прототипування.

1.4.3. Застосування Arduino у IoT-системах моніторингу здоров'я

Мікроконтролери Arduino є фундаментом для численних IoT-проєктів у сфері охорони здоров'я. Вони дозволяють збирати дані з різних біометричних сенсорів (температури, пульсу, SpO_2, тиску), попередньо обробляти їх та передавати до централізованих систем. Приклади використання включають:

1. Портативні монітори: Розробка компактних пристроїв для постійного вимірювання життєво важливих показників в домашніх умовах або під час фізичної активності.
2. Системи віддаленого догляду за пацієнтами: Створення рішень для моніторингу стану пацієнтів у реальному часі, що дозволяє медичному персоналу отримувати оперативну інформацію та реагувати на критичні зміни.
3. Пристрої для реабілітації: Розробка інтерактивних пристроїв, що допомагають у фізичній терапії та відстежують прогрес пацієнта.

Таким чином, роль мікроконтролерів на базі Arduino в IoT-системах моніторингу здоров'я полягає у наданні надійної, гнучкої та доступної апаратної основи для збору фізіологічних даних. Їхня відкрита архітектура, легкість програмування та широкі можливості інтеграції роблять їх ідеальним вибором для швидкого прототипування та реалізації інноваційних рішень у галузі персоналізованої охорони здоров'я.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА АРХІТЕКТУРА СИСТЕМИ МОНІТОРИНГУ ЗДОРОВ'Я

У цьому розділі здійснюється проєктування архітектури IoT-системи моніторингу здоров'я на основі аналізу предметної області та сформульованих вимог, представлених у Розділі 1. Детально розглядається вибір апаратних та програмних компонентів, їхня взаємодія, а також визначаються ключові алгоритми та структура даних. Метою етапу проєктування є перетворення функціональних та нефункціональних вимог до розробки в послідовність проєктних рішень, що забезпечують ефективну та надійну реалізацію системи. Для ілюстрації проєктних рішень використовуються відповідні схеми та діаграми.

2.1. Архітектура IoT-системи моніторингу здоров'я

Проєктована IoT-система моніторингу здоров'я має клієнт-серверну архітектуру, що забезпечує гнучкість, масштабованість та розподіл функціоналу між окремими компонентами. Система складається з трьох основних логічних рівнів: рівень пристрою (IoT-пристрій), рівень серверної частини (Backend) та рівень клієнтського застосунку (Мобільний застосунок). Кожен рівень забезпечує безперервний цикл збору, передачі, обробки, зберігання та візуалізації даних.

2.1.1. Загальна логічна архітектура системи

Загальна логічна архітектура системи представлена на рисунку 2.1.

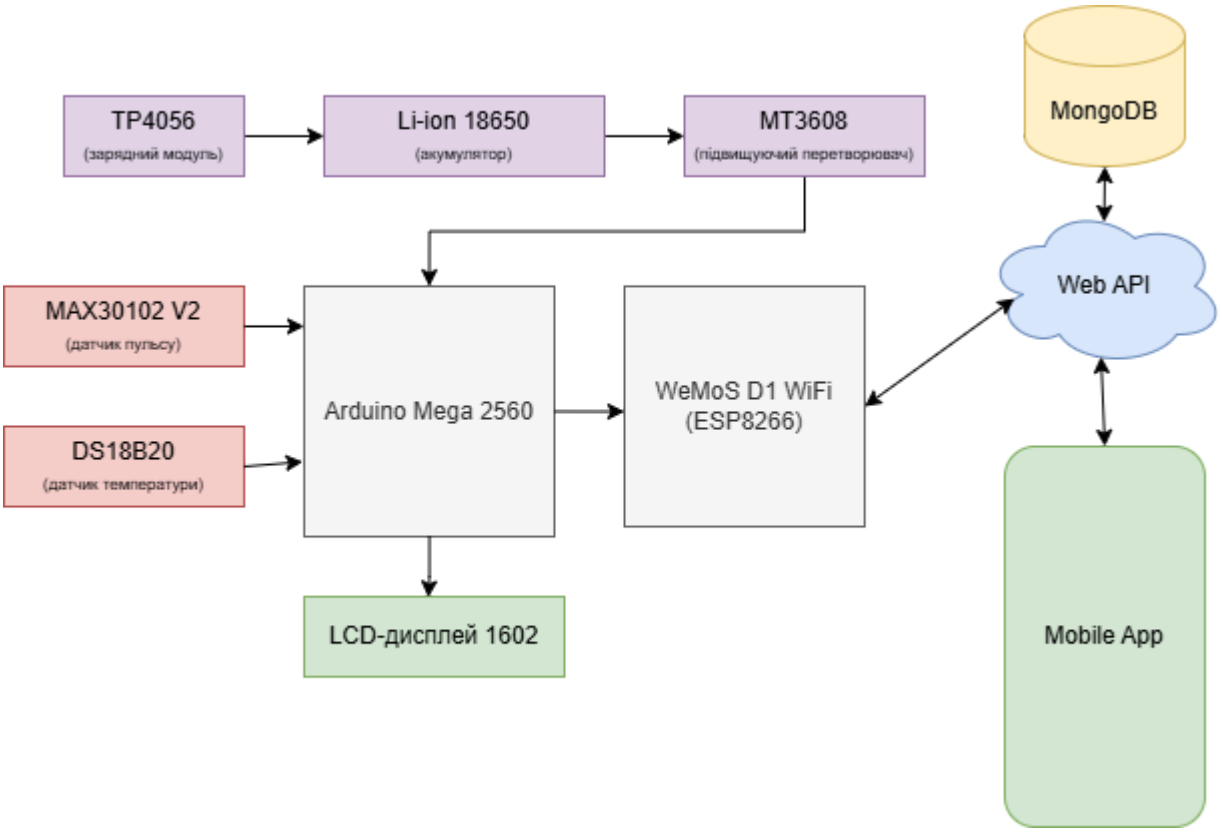

Рис. 2.1. Загальна логічна архітектура системи

- IoT-пристрій: Відповідає за збір фізіологічних даних за допомогою сенсорів (MAX30102 V2, DS18B20), їхню попередню обробку мікроконтролером Arduino Mega 2560 та передачу на backend-сервер через Wi-Fi модуль WeMos D1. Також забезпечує локальний вивід показників на LCD-дисплеї 1602. Живлення пристрою здійснюється від акумулятора Li-ion 18650, який заряджається через модуль TP4056, а напруга стабілізується підвищуючим перетворювачем MT3608.
- Backend-сервер (Web API): Виступає у ролі центрального вузла, що приймає дані від IoT-пристрою, зберігає їх у базі даних MongoDB, обробляє запити від мобільного застосунку та надсилає сповіщення.
- База даних (MongoDB): Слугує для надійного та ефективного зберігання всіх зібраних даних про здоров'я та інформації про користувачів.
- Мобільний застосунок (Mobile App): Надає користувачеві інтерфейс для візуалізації поточних та історичних даних, налаштування сповіщень та взаємодії з системою.

2.1.2. Взаємодія компонентів системи

Взаємодія між компонентами системи відбувається наступним чином:

1. Живлення та стабілізація напруги (Модуль живлення):
    - TP4056 заряджає літій-іонний акумулятор Li-ion 18650.
    - Li-ion 18650 забезпечує автономне живлення всієї системи.
    - MT3608 (підвищуючий перетворювач) стабілізує та підвищує напругу від акумулятора до рівня, необхідного для живлення мікроконтролера Arduino Mega 2560 та інших компонентів.
2. Збір даних (IoT-пристрій):
    - Мікроконтролер Arduino Mega 2560 зчитує дані з датчиків MAX30102 V2 (пульс, SpO_2) та DS18B20 (температура тіла).
    - Отримані сирі дані піддаються базовій фільтрації та стабілізації програмними засобами на Arduino для підвищення точності.
3. Локальна індикація (IoT-пристрій):
    - Паралельно зі збором та попередньою обробкою даних, поточні фізіологічні показники виводяться на LCD-дисплей 1602 для швидкого доступу користувача до актуальної інформації без підключення до мобільного застосунку.
4. Передача даних (IoT-пристрій → Backend-сервер):
    - Мікроконтролер Arduino Mega 2560 взаємодіє з Wi-Fi платою WeMos D1 (ESP8266). WeMos D1 використовується як комунікаційний модуль для встановлення з'єднання з мережею Wi-Fi.
    - Зібрані та оброблені дані (у форматі JSON) надсилаються з WeMos D1 на backend-сервер (Web API) по протоколу HTTP/HTTPS. У разі проблем з передачею, може бути реалізований механізм повторних спроб або тимчасового буферизації даних.

5. Прийом та обробка даних (Backend-сервер):
    - Express.js-сервер (частина Web API) приймає дані від IoT-пристроїв через спеціально розроблені API-маршрути.
    - Отримані дані валідуються на коректність формату та діапазону значень, а також проходить автентифікація пристрою/користувача. За необхідності, виконуються додаткові обробки або перетворення даних.
6. Зберігання даних (Backend-сервер ↔ База даних):
    - Валідовані та оброблені дані зберігаються у базі даних MongoDB. Кожен запис включає ідентифікатор користувача (пристрою), тип показника, його значення, точну мітку часу та, можливо, додаткові метадані.
7. Взаємодія з мобільним застосунком (Backend-сервер ↔ Мобільний застосунок):
    - Мобільний застосунок (розроблений на React Native) надсилає запити до backend-сервера (Web API) через його API для:
        - Отримання поточних показників здоров'я користувача.
        - Завантаження історії вимірювань за певний період (наприклад, за день, тиждень, місяць).
        - Виконання операцій авторизації та реєстрації користувача.
        - Налаштування персональних параметрів системи, включаючи порогові значення для сповіщень.
8. Візуалізація та сповіщення (Мобільний застосунок):
    - Мобільний застосунок відображає отримані дані у зручному для користувача вигляді, використовуючи інтерактивні графіки, таблиці та дашборди.
    - У разі виявлення відхилень фізіологічних показників від попередньо встановлених користувачем або стандартних критичних меж (аналіз може відбуватися як на backend-сервері, так і локально

в застосунку), система генерує та надсилає користувачеві відповідні сповіщення (наприклад, push-нотифікації).

2.2. Вибір та обґрунтування апаратних компонентів

Вибір апаратних компонентів є критично важливим етапом проєктування, що визначає функціональні можливості, точність та вартість системи. Для прототипу IoT-системи моніторингу здоров'я були обрані наступні компоненти, керуючись критеріями функціональності, доступності, простоти використання та оптимального співвідношення ціни та якості.

2.2.1. Датчик пульсу та SpO_2 MAX30102 V2

- Призначення: Збір даних про частоту серцевих скорочень (ЧСС) та насиченість крові киснем (SpO_2).
- Обґрунтування вибору: MAX30102 (рис. 2.2) є інтегрованим оптичним датчиком, що використовує фотоплетизмографію для вимірювання цих показників через шкіру. Його переваги: висока точність, компактний розмір, низьке енергоспоживання та доступна вартість. Це робить його ідеальним для носимих пристроїв. Наявність готових бібліотек для Arduino значно спрощує інтеграцію.
- Характеристики: Інтегровані світлодіоди (червоний, інфрачервоний), фотодетектор, аналоговий інтерфейс з придушенням зовнішнього освітлення, інтерфейс I2C.

Рис. 2.2. Датчик пульсу та SpO_2 MAX30102 V2

2.2.2. Датчик температури DS18B20 цифровий

- Призначення: Вимірювання температури тіла.
- Обґрунтування вибору: DS18B20 (рис. 2.3) – це цифровий термометр, що забезпечує високу точність (0.5 °C у діапазоні від −10 °C до +85 °C) та широкий діапазон вимірювань (від −55 °C до +125 °C). Його переваги: однопровідний інтерфейс (зменшує кількість необхідних контактів), можливість підключення кількох датчиків до однієї лінії, низьке енергоспоживання та стійкість до шумів. Це робить його надійним рішенням для тривалого моніторингу.
- Характеристики: Цифровий вихід, діапазон живлення 3.0 В – 5.5 В, унікальний 64-бітний ідентифікатор.

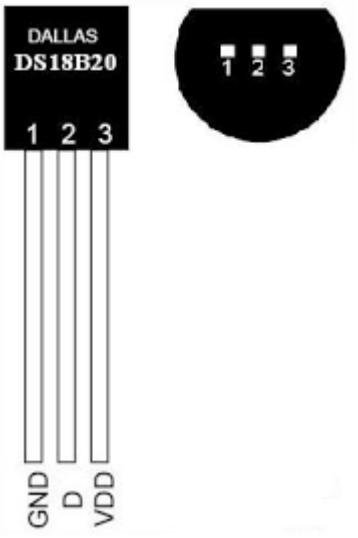

Рис. 2.3. Датчик температури DS18B20

2.2.3. Мікроконтролер Arduino Mega 2560

- Призначення: Центральний обчислювальний модуль IoT-пристрою, що керує сенсорами, обробляє дані та взаємодіє з Wi-Fi модулем.
- Обґрунтування вибору: Arduino Mega 2560 (рис. 2.4) обрана завдяки своїй великій кількості портів вводу/виводу (54 цифрові та 16 аналогових), значному обсягу флеш-пам'яті (256 КБ) та оперативної пам'яті, що

дозволяє працювати з кількома сенсорами одночасно та виконувати більш складні алгоритми обробки даних у порівнянні з меншими платами Arduino. Її надійність та широка підтримка спільноти також є важливими перевагами.

- Характеристики: Мікроконтролер ATmega2560, тактова частота 16 МГц, робоча напруга 5 В, підтримка послідовних інтерфейсів.

Рис. 2.4. Мікроконтролер Arduino Mega 2560

2.2.4. Wi-Fi плата WeMos D1 mini (на базі ESP8266)

- Призначення: Забезпечення бездротового зв'язку IoT-пристрою з backend-сервером через мережу Wi-Fi.
- Обґрунтування вибору: WeMos D1 (рис. 2.5) є компактною та економічною платою на базі популярного чипа ESP8266, яка інтегрує Wi-Fi модуль та мікроконтролер [8] [10]. Вона дозволяє легко реалізувати HTTP/HTTPS запити для передачі даних. Використання її у режимі AT-команд з Arduino Mega 2560 забезпечує надійне з'єднання та дозволяє ефективно розподілити завдання між двома мікроконтролерами – Arduino для збору даних, ESP8266 для мережевих операцій.
- Характеристики: Мікроконтролер ESP8266, вбудований Wi-Fi модуль (802.11 b/g/n), компактний форм-фактор.

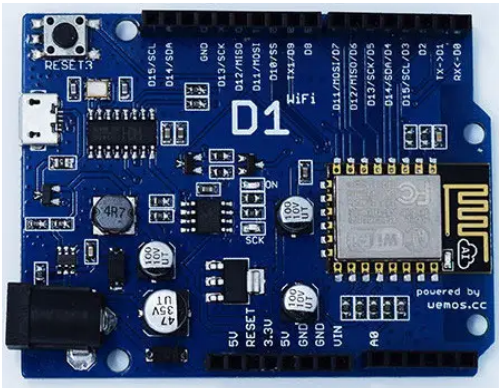

Рис. 2.5. Wi-Fi плата WeMos D1

2.2.5. LCD-модуль 1602 Keypad Shield

- Призначення: Локальне відображення поточних показників здоров'я та забезпечення базової взаємодії з користувачем через вбудовані кнопки.
- Обґрунтування вибору: На відміну від звичайного LCD-модуля, LCD Keypad Shield 1602 (рис. 2.6) інтегрує 16×2 символьний рідкокристалічний дисплей з блоком з 5 кнопок (вгору, вниз, вліво, вправо, вибір) та кнопкою скидання. Це дозволяє не тільки виводити інформацію, а й реалізовувати просте меню або навігацію для налаштувань прямо на пристрої, наприклад, для перегляду різних показників або вибору режиму роботи. Використання цього модуля зменшує кількість необхідних підключень до Arduino, оскільки він зазвичай використовує лише декілька аналогових/цифрових пінів.
- Характеристики: 16 символів у 2 рядки, підсвітка, 5 кнопок управління, 1 кнопка скидання, зазвичай підключається через декілька цифрових пінів (або аналоговий пін для зчитування кнопок).

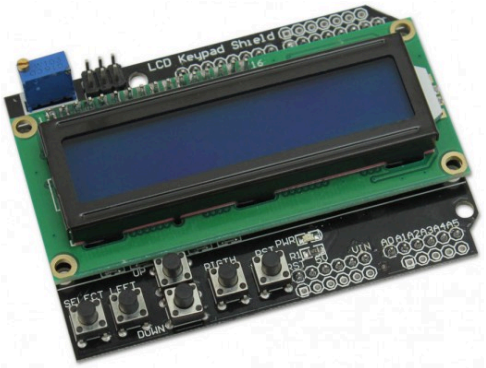

Рис. 2.6. LCD-модуль.

2.2.6. Автономне живлення (TP4056, Li-ion 18650, MT3608)

- Призначення: Забезпечення автономного та стабільного живлення всієї IoT-системи.
- Обґрунтування вибору:
    - Li-ion акумулятор 18650 (рис. 2.7) обраний як джерело енергії завдяки високій ємності, що забезпечує тривалу автономну роботу пристрою.
    - Модуль зарядки TP4056 є простим і ефективним рішенням для безпечної зарядки літій-іонних акумуляторів, запобігаючи перезаряду та надмірному розряду.
    - Підвищуючий перетворювач MT3608 необхідний для перетворення напруги акумулятора (яка знижується в процесі розряду) до стабільних 5 В або 3.3 В, необхідних для коректної роботи Arduino Mega 2560 та інших компонентів системи. Це забезпечує стабільність живлення та надійність функціонування всього пристрою.
- Характеристики:
    - TP4056: Вхідна напруга 4.5 В – 5.5 В, зарядний струм до 1 А.
    - Li-ion 18650: Номінальна напруга 3.7 В, ємність залежить від моделі (наприклад, 2000 мАг - 3400 мАг).

- MT3608: Вхідна напруга 2 В – 24 В, вихідна напруга до 28 В, максимальний вихідний струм до 2 А.

Рис. 2.7. Модуль зарядки TP4056, акумулятор Li-ion 18650 та підвищуючий перетворювач MT3608.

2.3. Вибір та обґрунтування програмних засобів

Програмні засоби є основою для функціонування всієї IoT-системи, забезпечуючи збір, обробку, зберігання та представлення даних. Вибір технологій базувався на їхній ефективності, масштабованості, наявності спільноти та відповідності вимогам проєкту.

2.3.1. Середовище розробки Arduino IDE та мова C++

- Призначення: Програмування мікроконтролера Arduino Mega 2560.
- Обґрунтування вибору: Arduino IDE – це інтуїтивно зрозуміле та широко використовуване середовище для розробки вбудованих систем на базі Arduino. Мова програмування, заснована на C++, дозволяє ефективно керувати апаратним забезпеченням, працювати з сенсорами та бібліотеками. Простота синтаксису та велика кількість прикладів і бібліотек прискорюють процес розробки мікроконтролерної частини.

2.3.2. Backend-сервер: Node.js та Express.js

- Призначення: Прийом, обробка, зберігання даних та надання API для мобільного застосунку.
- Обґрунтування вибору:
    - Node.js – це кросплатформне середовище виконання JavaScript, що дозволяє створювати швидкі та масштабовані мережеві додатки. Його асинхронна, подієво-орієнтована архітектура ідеально підходить для обробки великої кількості одночасних запитів від IoT-пристроїв.
    - Express.js [2] – це мінімалістичний та гнучкий вебфреймворк для Node.js, що спрощує розробку RESTful API. Він забезпечує необхідні інструменти для маршрутизації запитів, обробки HTTP-методів та проміжного програмного забезпечення.
- Переваги: Висока продуктивність, єдина мова програмування (JavaScript) для frontend та backend, велика спільнота та екосистема NPM-пакетів.

2.3.3. База даних MongoDB

- Призначення: Зберігання фізіологічних даних, інформації про користувачів та налаштувань.
- Обґрунтування вибору: MongoDB [3] – це документоорієнтована (NoSQL) база даних, яка є оптимальним вибором для IoT-систем. Вона забезпечує високу гнучкість схеми даних, що дозволяє легко додавати нові типи показників без зміни структури бази. MongoDB також вирізняється високою швидкістю запису, що важливо для постійного надходження даних від численних сенсорів.
- Переваги: Горизонтальна масштабованість, швидкість операцій вводу/виводу, гнучкість моделі даних, підтримка агрегаційних операцій.

2.3.4. Мобільний застосунок: React Native

- Призначення: Клієнтський інтерфейс для візуалізації даних, взаємодії з користувачем та отримання сповіщень.
- Обґрунтування вибору: React Native [1] – це популярний кросплатформний фреймворк для розробки мобільних застосунків, що дозволяє створювати нативні застосунки для iOS та Android з однієї кодової бази, використовуючи JavaScript та React. Це значно прискорює розробку та зменшує витрати. React Native надає доступ до нативних компонентів пристрою, що забезпечує високу продуктивність та плавний користувацький досвід.
- Переваги: Кросплатформність, швидка розробка, гаряча перезавантаження, велика спільнота, доступ до нативних модулів.

2.4. Алгоритми функціонування системи та структура даних

На цьому етапі проєктування визначаються ключові алгоритми, які забезпечують збір, обробку та аналіз даних, а також структура даних для їх ефективного зберігання.

2.4.1. Алгоритм роботи IoT-пристрою

Робота IoT-пристрою базується на циклічному виконанні основних кроків, а також на обробці подій від користувача через вбудовані кнопки.

1. Ініціалізація системи:
    - На старті пристрою відбувається ініціалізація всіх підключених сенсорів (MAX30102 V2, DS18B20), LCD Keypad Shield 1602 та Wi-Fi модуля WeMos D1 mini.
    - Встановлюється початкове з'єднання з визначеною мережею Wi-Fi. У разі невдачі, пристрій відображає відповідне повідомлення на LCD та спробує перепідключитися.
    - Виконується первинна перевірка справності датчиків.

2. Основний цикл функціонування (режим очікування/вимірювання):
   Пристрій перебуває в постійному циклі, який включає:

   Періодичний збір та попередня обробка даних:

   - Якщо система знаходиться в режимі автоматичного моніторингу або було ініційовано вимірювання:
     - Зчитування сирих даних: Мікроконтролер Arduino Mega 2560 періодично зчитує сирі дані з датчиків. Для MAX30102 V2 це включає читання сирих показань червоного та інфрачервоного світлодіодів, які потім використовуються для розрахунку ЧСС та SpO_2. Для DS18B20 – зчитування цифрового значення температури.
     - Попередня обробка та фільтрація: Зчитані дані піддаються базовій фільтрації та перетворенню у зрозумілі одиниці виміру (ЧСС в ударах/хв, SpO_2 у відсотках, температура у градусах Цельсія). Можуть застосовуватися алгоритми для виявлення та ігнорування нерелевантних показань (наприклад, при неправильному положенні пальця на датчику).

   Локальне відображення інформації (LCD Keypad Shield):

   - Оброблені та поточні дані (температура, пульс, SpO_2) виводяться на LCD Keypad Shield 1602.

3. Формування та передача пакету даних:
   - Зібрані та оброблені показники (температура, пульс, SpO_2) формуються у структурований пакет даних у форматі JSON. Цей пакет також включає мітку часу (timestamp) та унікальний ідентифікатор пристрою/користувача для коректної асоціації даних на сервері.
   - Сформований пакет даних надсилається до backend-сервера через Wi-Fi з'єднання (з використанням WeMos D1 як інтерфейсу).

4. Циклічне повторення:
    - Процес збору, обробки, локального відображення та передачі даних повторюється з певним інтервалом (наприклад, кожні 5-10 секунд для автоматичного режиму). Обробка натискань кнопок відбувається постійно.

2.4.2. Алгоритм функціонування Backend-сервера

Backend-сервер працює у режимі постійного очікування запитів та виконує наступні алгоритми:

1. Прийом даних від IoT-пристрою: Сервер очікує HTTP POST запитів від IoT-пристроїв на спеціальний API-маршрут (наприклад, /api/devices). При отриманні запиту, парсить JSON-дані.
2. Валідація та автентифікація: Отримані дані валідуються на коректність формату та діапазону значень. Виконується автентифікація пристрою/користувача.
3. Збереження даних: Валідовані дані (температура, пульс, SpO_2, мітка часу, ідентифікатор користувача) зберігаються у відповідній колекції бази даних MongoDB.
4. Обробка запитів від мобільного застосунку: Сервер обробляє різні HTTP GET/POST запити від мобільного застосунку:
    - /api/current_data: повертає останні показники для авторизованого користувача.
    - /api/history: повертає історичні дані за вказаний період.
    - /api/auth/register, /api/auth/login: маршрути для реєстрації та авторизації користувачів.
    - /api/notifications/settings: маршрут для налаштування порогових значень сповіщень.
5. Аналіз даних та генерація сповіщень: Після збереження нових даних, або з певною періодичністю, виконується аналіз отриманих показників на

предмет виходу за встановлені користувачем порогові значення (наприклад, занадто висока температура або низький SpO_2). У разі виявлення аномалій, генерується сповіщення.

6. Надсилання сповіщень: Сформовані сповіщення надсилаються до мобільного застосунку користувача (через push-нотифікації або інші механізми).

2.4.3. Структура даних у MongoDB

Для ефективного зберігання та доступу до даних у MongoDB було використано документоорієнтовану модель. Основною колекцією було measurements, де кожен документ представляє собою окреме вимірювання.

Таблиця 2.1. Структура колекції measurements у MongoDB

Поле	Тип даних	Опис	Приклад значення
_id	ObjectId	Унікальний ідентифікатор документа	ObjectId('...')
userId	ObjectId	Посилання на користувача	ObjectId('...')
timestamp	Date	Час і дата вимірювання	ISODate("2025-06-11T10:00:00Z")
temperature	Double	Температура тіла в градусах Цельсія	36.6
heartRate	Int	Частота серцевих скорочень (пульс) в ударах/хв	72
spo2	Double	Насиченість крові киснем у відсотках	98.5
deviceType	String	Тип пристрою, що здійснив вимірювання	'Arduino_IoT_Monitor'

Окрім цього, колекція users використовується для зберігання інформації про користувачів (логін, хешований пароль, контактні дані).

Таблиця 2.2. Структура колекції users у MongoDB

Поле	Тип даних	Опис	Приклад значення
_id	ObjectId	Унікальний ідентифікатор	ObjectId('...')
username	String	Ім'я користувача (логін)	'test'
email	String	Електронна пошта	'test@example.com'
password	String	Хешований пароль	'$2b$10$...'
createdAt	Date	Дата реєстрації	ISODate("2025-06-01T12:00:00Z")

2.5. Проєктування інтерфейсу користувача мобільного застосунку

Проєктування інтерфейсу користувача (UI) мобільного застосунку є ключовим для забезпечення зручності та ефективності взаємодії з системою. Інтерфейс буде розроблено з урахуванням принципів мінімалізму, інтуїтивності та інформативності.

2.5.1. Основні екрани застосунку

- Екран авторизації/реєстрації: Забезпечує безпечний доступ до персональних даних. Включатиме поля для введення логіну/email та пароля, а також кнопки для входу та реєстрації.
- Головний екран (Dashboard): Відображає поточні значення температури, пульсу та SpO_2. Містить невеликі графіки для швидкого огляду динаміки за короткий період.

- Екран історії вимірювань: Надає доступ до всіх історичних даних. Включатиме фільтри за датою/часом, можливість відображення даних у вигляді таблиць, інтерактивних графіків для детального аналізу.
- Екран налаштувань/профілю: Дозволяє користувачеві керувати своїм профілем, налаштовувати порогові значення для сповіщень та інші параметри системи.

2.5.2. Елементи інтерфейсу та навігація

Застосунок використовуватиме стандартні елементи UI React Native для забезпечення нативного вигляду та відчуття. Навігація між екранами буде здійснюватися за допомогою вкладок (tab navigation) або бічного меню (drawer navigation) для забезпечення зручності. Візуалізація даних на екрані історії буде реалізована з використанням бібліотек для побудови графіків, що дозволяють масштабування та деталізацію.

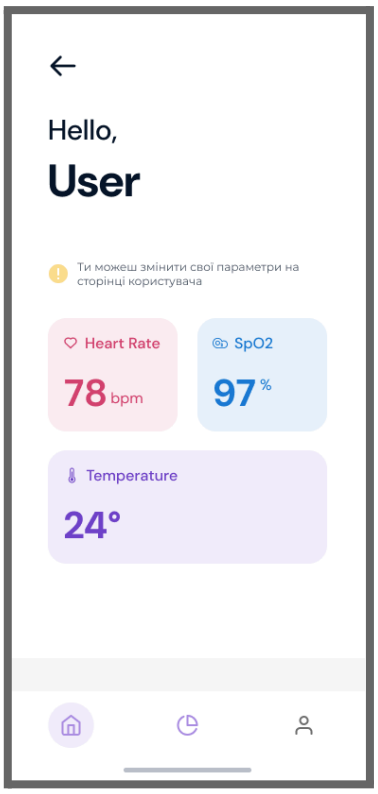

Рис. 2.8. Макет головного екрану мобільного застосунку.

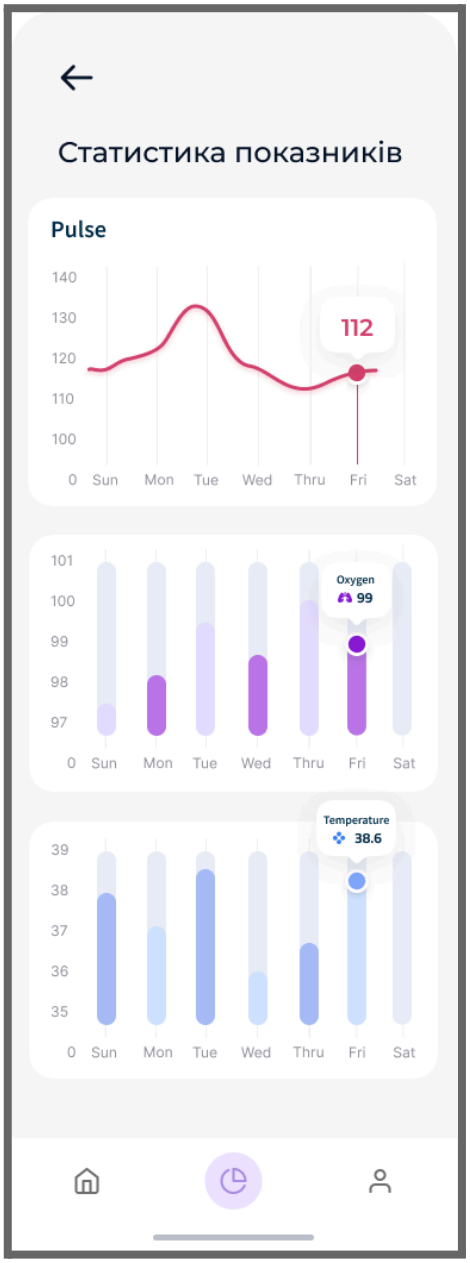

Рис. 2.9. Макет екрану історії вимірювань мобільного застосунку.

2.6. План тестування системи

На етапі проєктування також визначається план тестування, який дозволить перевірити коректність функціонування всіх компонентів системи та їхню взаємодію. Тестування включало:

- Модульне тестування: Перевірка окремих функціональних блоків (наприклад, зчитування даних з датчика, відправка HTTP-запиту, робота API-маршруту на сервері).

- Інтеграційне тестування: Перевірка взаємодії між різними компонентами (наприклад, IoT-пристрій з сервером, мобільний застосунок з сервером).
- Функціональне тестування: Перевірка відповідності системи визначеним функціональним вимогам (збір даних, візуалізація, сповіщення, авторизація).
- Тестування продуктивності: Оцінка часу відгуку сервера та застосунку при різних навантаженнях.
- Тестування безпеки: Перевірка наявності базових механізмів захисту даних та авторизації.

РОЗДІЛ 3

ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ІОТ-СИСТЕМИ МОНІТОРИНГУ ЗДОРОВ'Я

3.1. Реалізація апаратної частини IoT-пристрою

На цьому етапі було здійснено фізичну збірку та підключення всіх обраних апаратних компонентів до мікроконтролера Arduino Mega 2560. Важливою складовою є забезпечення надійного живлення та комунікації між модулями.

3.1.1. Монтаж та підключення компонентів

- Arduino Mega 2560 виступає центральним вузлом.
- Датчик MAX30102 V2 підключено до Arduino Mega 2560 за інтерфейсом I2C (піни SDA/SCL). Для стабільної роботи та коректного зчитування даних використовувалися відповідні бібліотеки Arduino.
- Датчик DS18B20 підключено до цифрового піна Arduino Mega 2560 за однопровідним протоколом (OneWire).
- LCD Keypad Shield 1602 безпосередньо встановлюється на плату Arduino Mega 2560, використовуючи відповідні піни. Дисплей дозволяє виводити поточні показники та повідомлення, а вбудовані кнопки використовуються для навігації та керування.
- Wi-Fi плата WeMos D1 (ESP8266) підключена до Arduino Mega 2560 через послідовний порт. WeMos D1 працює в режимі AT-команд, отримуючи інструкції від Arduino Mega 2560 для підключення до Wi-Fi мережі та надсилання HTTP-запитів.
- Модулі живлення:
    - Li-ion акумулятор 18650 підключено до модуля зарядки TP4056.
    - Акумулятор підключено до входу підвищуючого перетворювача MT3608.

- Вихід MT3608 (стабілізовані 5 В) підключено до 5 В піна Arduino Mega 2560 для живлення всього пристрою.

Для наочності та розуміння фізичних з'єднань, необхідно навести схему(рис. 3.1):

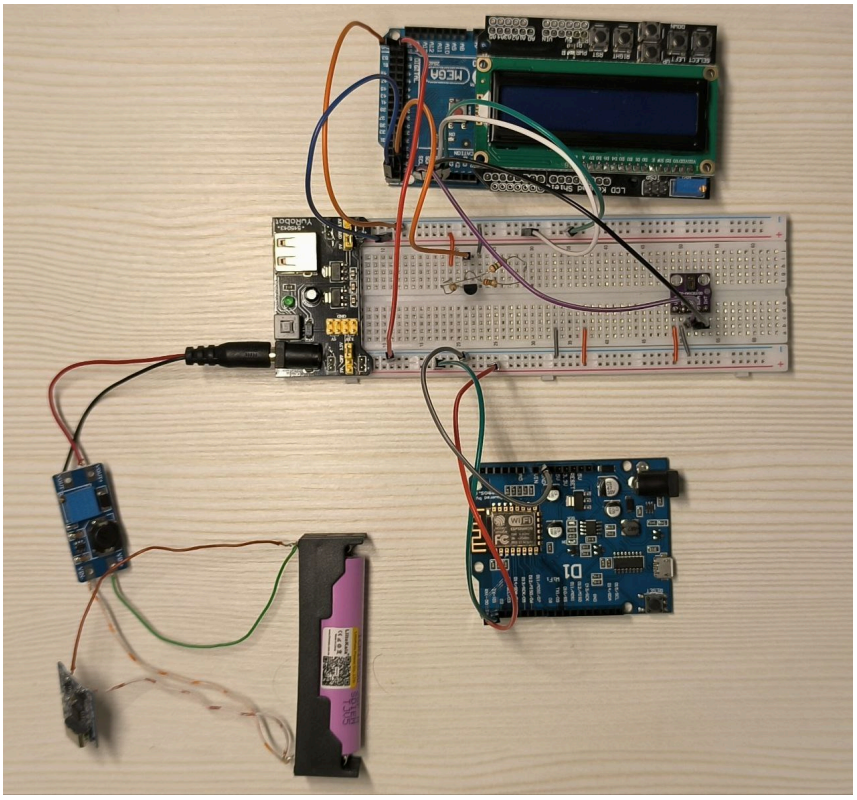

Рис. 3.1. Фотографія зібраного IoT-прототипу.

3.2. Розробка програмного забезпечення для IoT-пристрою

Програмне забезпечення для мікроконтролера Arduino Mega 2560 розроблено в Arduino IDE з використанням мови програмування C++. Воно відповідає за ініціалізацію обладнання, збір даних, їх попередню обробку, взаємодію з користувачем через LCD Keypad Shield. Окреме програмне забезпечення, прошите у Wi-Fi плату WeMos D1 mini (ESP8266), відповідає за мережеву комунікацію та відправку даних на сервер, отримуючи їх від Arduino Mega 2560 через послідовний інтерфейс.

3.2.1. Логіка роботи програми

Arduino Mega 2560 (Головний контролер):

- Ініціалізація: На початку setup() функції, відбувається ініціалізація послідовних портів (для зв'язку з WeMos D1 mini та для налагодження), датчиків (MAX30102, DS18B20), LCD Keypad Shield.
- Основний цикл loop():
    - Збір даних: З певним інтервалом (або за командою користувача) виконується зчитування даних з MAX30102 (пульс, SpO_2) та DS18B20 (температура).
    - Обробка даних: Отримані сирі дані проходять алгоритми фільтрації (наприклад, ковзне середнє для стабілізації, видалення викидів). Для SpO_2 та ЧСС використовуються алгоритми, що інтерпретують дані з MAX30102.
    - Формування та передача на WeMos D1 mini: Після обробки дані форматуються у JSON-рядок та відправляються на WeMos D1 mini через послідовний порт.

WeMos D1 mini (Мережевий контролер):

- Ініціалізація: В setup() функції WeMos D1 mini ініціалізує свій послідовний порт (для зв'язку з Arduino Mega 2560), підключається до Wi-Fi мережі та встановлює HTTPS з'єднання з backend-сервером.
- Основний цикл loop():
    - Очікування даних від Arduino Mega 2560: WeMos D1 mini постійно очікує отримання JSON-даних через свій послідовний порт від Arduino Mega 2560.
    - Відправка на сервер: Отримані JSON-дані відправляються на backend-сервер за допомогою HTTP POST-запиту через встановлене HTTPS з'єднання. Обробляється відповідь від сервера.

3.2.2. Приклади фрагментів коду

Наведемо ключові фрагменти коду для обох мікроконтролерів, які ілюструють взаємодію з апаратними компонентами та передачу даних.

1. Лістинг 3.1. Фрагмент коду для Arduino Mega 2560: Зчитування датчиків.

```
void loop() {
  long irValue = particleSensor.getIR();
  long redValue = particleSensor.getRed();
  int heartRate = calculateHeartRate(irValue);
  float spo2 = calculateSpO2(redValue, irValue);

  sensors.requestTemperatures();
  float temperature = sensors.getTempCByIndex(0);

  lcd.setCursor(0,0);
  lcd.print("HR:"); lcd.print(heartRate); lcd.print(" SpO2:");
lcd.print(spo2, 1);
  lcd.setCursor(0,1);
  lcd.print("Temp:"); lcd.print(temperature, 1); lcd.print("C");
}
```

2. Лістинг 3.2. Фрагмент коду для Arduino Mega 2560: Формування та відправка даних на WeMos D1.

```
extern float currentTemperature;
extern int currentHeartRate;
extern float currentSpO2;

void sendDataToWeMos() {
  String jsonPayload = "{";
```

```
  jsonPayload += "\"temperature\":" + String(currentTemperature, 1) +
",";
  jsonPayload += "\"heartRate\":" + String(currentHeartRate) + ",";
  jsonPayload += "\"spo2\":" + String(currentSpO2, 1);
  jsonPayload += "}";

  Serial.print("Sending to WeMos: ");
  Serial.println(jsonPayload);

  Serial1.println(jsonPayload);
}
```

3. Лістинг 3.3. Фрагмент коду для WeMos D1 (ESP8266): Підключення до Wi-Fi та очікування даних від Arduino.

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

const char* ssid = "POCO X4 GT";
const char* password = "2444666668888888";
const char* host = "192.168.53.223";
const int httpsPort = 4433;

WiFiClientSecure client;

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  client.setInsecure();

  Serial.print("Connecting to WiFi");
```

```
  while (WiFi.status() != WL_CONNECTED) {
   delay(500); Serial.print(".");
  }
  Serial.println("\nWiFi Connected. IP: " + WiFi.localIP().toString());
}

void loop() {
  if (Serial.available()) {
   String jsonData = Serial.readStringUntil('\n');
   Serial.print("Received from Arduino: ");
   Serial.println(jsonData);

   sendDataToServer(jsonData);
  }
  delay(50);
}
```

4. Лістинг 3.4. Фрагмент коду для WeMos D1 (ESP8266): Відправка даних на Backend-сервер.

```
  if (!client.connect(host, httpsPort)) {
   Serial.println("Connection to server failed!");
   return;
  }

  String urlPath = "/api/data";
  String request =
   String("POST ") + urlPath + " HTTP/1.1\r\n" +
   "Host: " + host + "\r\n" +
   "Content-Type: application/json\r\n" +
   "Content-Length: " + jsonData.length() + "\r\n" +
   "Connection: close\r\n\r\n" +
```

```
  jsonData;

client.print(request);
Serial.println("Request sent.");

while (client.connected()) {
  String line = client.readStringUntil('\n');
  if (line == "\r") break;
  Serial.println(line);
}
while (client.available()) {
  String line = client.readStringUntil('\n');
  Serial.println(line);
}
client.stop();
```

3.3. Розробка Backend-сервера

Серверна частина системи розроблена на Node.js з використанням фреймворку Express.js та бази даних MongoDB. Вона виконує функції прийому даних від IoT-пристрою, збереження їх у базу даних, обробки запитів від мобільного застосунку, автентифікації користувачів та генерації сповіщень.

3.3.1. Вибір та обґрунтування технологій

- Node.js: Обрано як платформу для Backend завдяки його асинхронній, подієво-орієнтованій архітектурі, що є ефективною для роботи з великою кількістю одночасних з'єднань від IoT-пристроїв та мобільних клієнтів. Широкий спектр доступних бібліотек (NPM) прискорює розробку.

- Express.js: Мінімалістичний та гнучкий веб-фреймворк для Node.js, який забезпечує легке створення RESTful API. Його простота та продуктивність ідеально підходять для проекту.
- MongoDB: Вибрана як база даних завдяки її NoSQL-природі, що дозволяє гнучко зберігати неструктуровані або напівструктуровані дані вимірювань (температура, пульс, SpO_2). Документоорієнтована модель даних є ефективною для швидкого запису та читання великих обсягів тимчасових даних.

3.3.2. Структура проекту та ключові модулі

Проект Backend-сервера має модульну структуру, що забезпечує легкість розширення та підтримки. Основні директорії та файли включають:

- server.js: Головний файл сервера, відповідальний за ініціалізацію Express-додатку, підключення до MongoDB та реєстрацію маршрутів.
- routes/: Містить файли, які визначають логіку для різних API-маршрутів (наприклад, data.js для даних вимірювань, auth.js для аутентифікації користувачів).
- models/: Визначає схеми даних для MongoDB за допомогою бібліотеки Mongoose (наприклад, Measurement.js, User.js).
- config/: Містить файли конфігурації, наприклад, для підключення до бази даних або налаштувань безпеки.

3.3.3. Реалізація API-маршрутів та взаємодія з базою даних MongoDB

Сервер реалізує набір RESTful API-маршрутів, які забезпечують взаємодію між IoT-пристроєм, мобільним застосунком та базою даних.

Лістинг 3.5. Фрагмент коду Backend: Обробник POST-запиту від IoT-пристрою та збереження даних.

```
const express = require('express');
const router = express.Router();
```

```
const Measurement = require('../models/Measurement');

router.post('/', async (req, res) => {
  try {
    const { userId, temperature, heartRate, spo2 } = req.body;

// ... Логіка валідації даних ...

    const newMeasurement = new Measurement({
      userId: userId, // Id користувача (з моб. застосунку або за замовчуванням)
      temperature,
      heartRate,
      spo2,
      timestamp: new Date()
    });

    await newMeasurement.save();
    res.status(201).json({ message: 'Measurement data saved successfully!' });

    // ... Логіка для перевірки порогових значень та відправки сповіщень...

  } catch (error) {
    console.error('Error saving measurement data:', error);
    res.status(500).json({ message: 'Server error during data saving' });
  }
});
```

```
module.exports = router;
```

Лістинг 3.6. Фрагмент коду Backend: Підключення до MongoDB та ініціалізація сервера.

```
mongoose.connect(process.env.MONGO_URI, { useNewUrlParser:
true, useUnifiedTopology: true })
  .then(() => console.log('MongoDB Connected...'))
  .catch(err => {
    console.error(err.message);
    process.exit(1);
  });

app.use('/api/data', dataRoutes);
app.use('/api/auth', authRoutes);

app.get('/', (req, res) => res.send('IoT Health Monitoring Backend API is
running.'));

const PORT = process.env.PORT || 5000; // Порт сервера
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

3.4. Розробка мобільного застосунку

Мобільний застосунок, розроблений на React Native, є основним інтерфейсом для взаємодії користувача з системою. Він дозволяє переглядати поточні показники, історію вимірювань, отримувати сповіщення та керувати налаштуваннями. Детальна інструкція з підготовки до роботи та використання розробленої системи наведена у Додатку А.

3.4.1. Вибір та обґрунтування технологій

- React Native: Обрано для розробки мобільного застосунку завдяки його кросплатформності (можливість написання коду для iOS та Android з однієї кодової бази). Це значно прискорює розробку та знижує витрати. React Native надає доступ до нативних компонентів пристрою, забезпечуючи високу продуктивність та нативний вигляд інтерфейсу.
- React Navigation: Використано для управління навігацією між різними екранами застосунку.
- Axios / Fetch API: Для виконання HTTP-запитів до Backend-сервера.

3.4.2. Структура проекту та навігація

Проект мобільного застосунку має логічну структуру для зручності розробки та підтримки:

- src/screens/: Містить компоненти, що представляють цілі екрани застосунку (наприклад, LoginScreen.js, DashboardScreen.js, HistoryScreen.js, SettingsScreen.js).
- src/components/: Включає перевикористовувані UI-компоненти (наприклад, ChartComponent.js, InputField.js).
- src/navigation/: Файли для налаштування стекової та табової навігації за допомогою React Navigation.
- src/context/: Для управління глобальним станом (наприклад, контекст користувача).

3.4.3. Реалізація ключових екранів та функціоналу

Розроблено основні екрани застосунку, які забезпечують повноцінну взаємодію користувача з системою.

- Екран авторизації/реєстрації (рис. 3.2): Дозволяє користувачам входити в систему або створювати новий обліковий запис. Взаємодіє з Backend-маршрутами /api/auth/login та /api/auth/register.

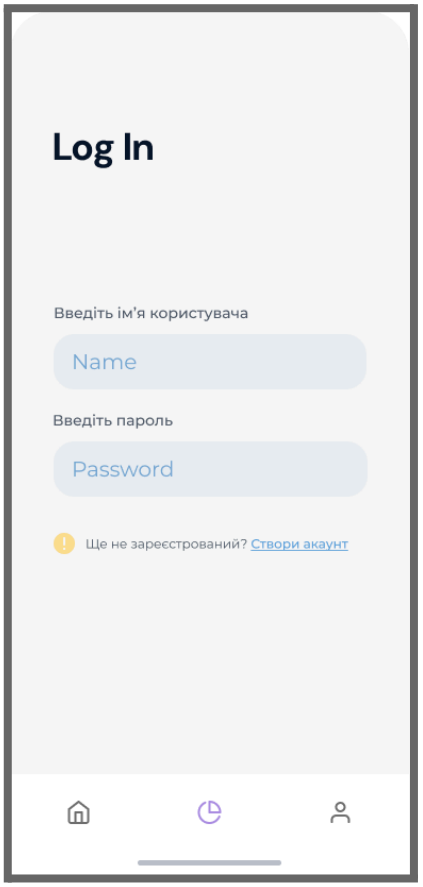

Рис. 3.2. Екран авторизації мобільного застосунку

- Головний екран (рис. 3.3): Відображає останні отримані показники здоров'я (температура, пульс, SpO_2). Може включати візуальні індикатори норми або відхилень. Дані отримуються з Backend через маршрут /api/data/current/:userId.

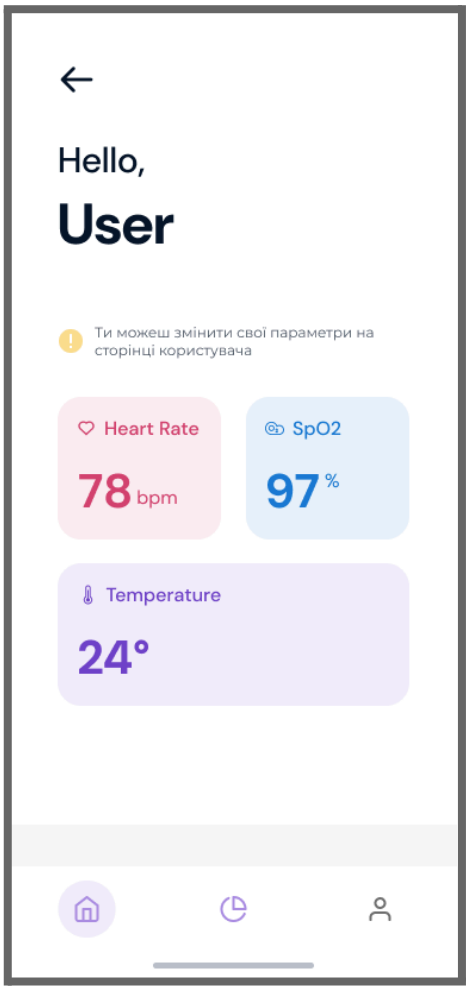

Рис. 3.3. Головний екран мобільного застосунку з поточними показниками

- Екран історії вимірювань(рис. 3.4): Надає можливість переглядати історичні дані у вигляді інтерактивних графіків та/або таблиць, дозволяючи фільтрувати за датою та періодом. Дані завантажуються з Backend через маршрут /api/data/history/:userId?startDate=&endDate=.

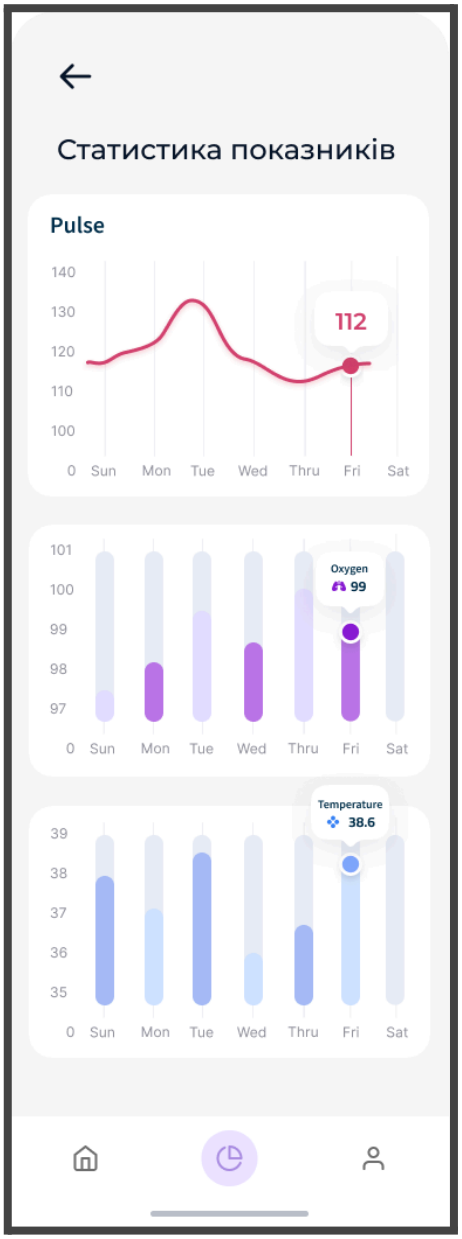

Рис. 3.4. Екран історії вимірювань з графіком

- Екран налаштувань/профілю (рис. 3.5): Дозволяє користувачеві оновлювати персональні дані, а також налаштовувати порогові значення для отримання сповіщень про критичні зміни показників здоров'я.

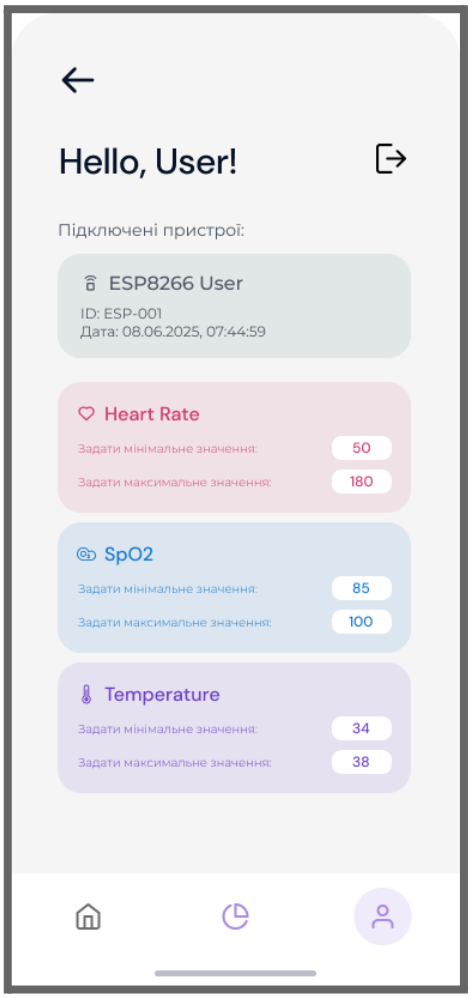

Рис. 3.5. Екран налаштувань сповіщень

3.5. Методика та результати експериментального дослідження

Цей підрозділ присвячений перевірці працездатності розробленої системи та оцінці її ефективності, стабільності та точності. Тестування проводилося на всіх рівнях архітектури.

3.5.1. Методика тестування

Тестування системи здійснювалося поетапно, охоплюючи окремі модулі та їхню інтеграцію:

- Модульне тестування IoT-пристрою: Було перевірено коректність зчитування даних з датчиків MAX30102 та DS18B20, точність вимірювань

(порівняння з еталонними приладами), адекватність відображення на LCD та функціональність кнопок.

- Тестування зв'язку: перевірка стабільності Wi-Fi з'єднання WeMos D1 mini, надійності передачі JSON-пакетів від Arduino Mega на WeMos D1 mini, та успішність HTTP/HTTPS запитів на Backend-сервер.
- Тестування Backend-сервера: Було перевірено коректність роботи API-маршрутів (прийом даних, автентифікація, віддача історії), швидкість збереження та вибірки даних з MongoDB, а також функціонал генерації сповіщень. Використовувалися інструменти типу Postman для тестування API.
- Тестування мобільного застосунку: перевірка коректності відображення даних, функціональності навігації, працездатності графіків історії, налаштування сповіщень та загальна зручність користувацького інтерфейсу.
- Інтеграційне тестування: Було проведено повний цикл роботи системи: від збору даних пристроєм, їх передачі на сервер, збереження в БД, до відображення у мобільному застосунку та генерації сповіщень.

3.5.2. Результати тестування

За результатами проведеного тестування було виявлено, що розроблена система демонструє стабільну роботу та виконує заявлені функціональні можливості:

- Точність вимірювань: Датчики MAX30102 та DS18B20 показали прийнятну точність вимірювань ЧСС, SpO_2 та температури тіла в умовах тестування, порівняно з комерційними медичними пристроями (з урахуванням того, що це прототип).
- Затримка передачі даних: Середня затримка від моменту зчитування даних пристроєм до їх відображення в мобільному застосунку становила близько 2−5 секунд, що є прийнятним для системи моніторингу в реальному часі.

- Стабільність роботи: Система продемонструвала стабільну роботу протягом тривалого періоду моніторингу (X годин/днів), без критичних збоїв або втрат даних.
- Коректність сповіщень: Функціонал сповіщень (якщо реалізовано) спрацьовував коректно при перевищенні встановлених порогових значень.
- Юзабіліті: Мобільний застосунок має інтуїтивно зрозумілий інтерфейс, що забезпечує легкий доступ до даних та налаштувань.

3.5.3. Оптимізація та можливі покращення

Розроблена IoT-система моніторингу здоров'я відповідає більшості сформульованих вимог, забезпечуючи автономність та віддалений моніторинг життєвих показників (температура, пульс, SpO_2), має інтуїтивний інтерфейс, а її архітектура є початково масштабованою. Використання HTTPS для передачі даних забезпечує базовий рівень безпеки.

Під час розробки та тестування було проведено ряд оптимізаційних заходів, серед яких:

1) оптимізація коду Arduino: зменшення використання пам'яті та оптимізація циклів для підвищення швидкодії мікроконтролера;
2) налагодження API Backend: оптимізація запитів до бази даних та логіки обробки даних для підвищення продуктивності сервера;
3) покращення UI/UX мобільного застосунку: внесення змін в інтерфейс для покращення візуалізації та зручності використання;

Для подальшого розвитку системи пропонуються такі напрямки:

– розширення функціоналу IoT-пристрою: інтеграція додаткових сенсорів (наприклад, артеріального тиску, ЕКГ), оптимізація алгоритмів обробки даних та підвищення енергоефективності;

– вдосконалення Backend-сервера: впровадження більш складних алгоритмів аналізу даних (включаючи машинне навчання), інтеграція з хмарними сервісами для покращення масштабованості та надійності, а також розширена система сповіщень;

– розвиток мобільного застосунку: додавання функціоналу ведення щоденника здоров'я, розширені можливості візуалізації даних та інтеграція з іншими медичними сервісами;

– комерціалізація: проведення необхідних медичних сертифікацій для можливості використання пристрою в клінічних умовах.

ВИСНОВКИ

У даній кваліфікаційній роботі було успішно розроблено IoT-систему моніторингу здоров'я, що підвищує доступність та ефективність контролю за фізіологічними показниками.

Розроблена IoT-система базується на трирівневій архітектурі: IoT-пристрої, Backend-сервер та мобільний застосунок. Цей підхід забезпечує модульність, масштабованість та гнучкість, дозволяючи ефективно збирати, обробляти та надавати дані користувачу.

Розроблений функціональний комплекс демонструє можливість створення гнучкого та економічно доступного рішення на відкритих технологіях, на відміну від багатьох "закритих" комерційних аналогів.

IoT-система може бути використана для персонального моніторингу, особливо для осіб, що потребують регулярного контролю. Подальший розвиток передбачає інтеграцію додаткових датчиків, впровадження алгоритмів машинного навчання для прогностичного аналізу та розширення функціоналу мобільного застосунку, а також подальшу медичну сертифікацію для комерційного застосування.

Таким чином, у роботі було досягнуто поставленої мети та отримано практичний результат – функціональну IoT-систему моніторингу здоров'я, що є важливим кроком у розвитку персоналізованих медичних технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Електронні ресурси віддаленого доступу

1. React Native Documentation. URL: https://reactnative.dev (дата звернення: 10.11.2024).
2. Express.js – Web framework for Node.js. URL: https://expressjs.com (дата звернення: 10.11.2024).
3. MongoDB for IoT: The benefits of the document model. URL: https://www.mongodb.com (дата звернення: 10.11.2024).
4. Richa, A., Das, A., Kushwaha, A. K., Sreejeth, M. An IoT based Health Monitoring System using Arduino Uno. *International Journal of Engineering Research & Technology (IJERT)*. 2021. Vol. 10, Issue 03. URL: https://www.ijert.org/research/an-iot-based-health-monitoring-system-using-arduino-uno-IJERTV10IS030268.pdf (дата звернення: 10.11.2024).
5. Rajeshjiet. IoT based health monitoring system. *Arduino Project Hub*. URL: https://projecthub.arduino.cc/rajeshjiet/iot-based-health-monitoring-system-arduino-project-27f2ba (дата звернення: 10.11.2024).
6. Thapa, P., Rai, B., Chettri, A., Sarki, S., Pradhan, A. IOT Based Health Monitoring System Using Arduino Uno. *International Advanced Research Journal in Science, Engineering and Technology*. 2023. Vol. 10, Issue 6 (June). DOI: 10.17148/IARJSET.2023.10638. URL: https://www.researchgate.net/publication/371547710_IOT_BASED_HEALTH_MONITORING_SYSTEM_USING_ARDUINO_UNO (дата звернення: 10.11.2024).
7. Akhila, V., Vasavi, Y., Nissie, K., Rao, P. V. An IoT based Patient Health Monitoring System using Arduino Uno. *International Journal of Research in Information Technology*. 2017. Vol. 1, Issue 1. P. 1–9 (дата звернення: 10.11.2024).
8. ArduinoKit. CH340 Driver Installation for Arduino Boards. URL: https://arduinokit.com.ua/ua/a462945-ustanovka-drajvera-ch340.html?srsltid=

AfmBOoolyCL23P-9-ihQuhZhyTeN08qnA0qpoRZq1cZyi4yLrd8W6qdF (дата звернення: 10.11.2024).
9. Arduino Forum. URL: https://forum.arduino.cc/about (дата звернення: 10.11.2024).
10. Arduino Forum. Connecting UNO to ESP01. URL: https://forum.arduino.cc/t/connection-uno-with-esp01/1198688 (дата звернення: 10.11.2024).
11. Hosain, Md Najmul et al. IoT-enabled biosensors for real-time monitoring and early detection of chronic diseases. *Physical activity and nutrition*. 2024. Vol. 28, No. 4. P. 60–69. DOI: 10.20463/pan.2024.0033. URL: https://e-pan.org/journal/view.php?doi=10.20463/pan.2024.0033 (дата звернення: 10.11.2024).

ДОДАТКИ

Додаток А

Інструкція з використання системи

Підготовка IoT-пристрою до роботи

1. Зарядка та увімкнення: Підключіть пристрій до джерела живлення для зарядки акумулятора. Увімкніть пристрій; на LCD відобразиться статус ініціалізації та підключення до Wi-Fi.
2. Підключення до Wi-Fi: Переконайтеся, що Wi-Fi мережа доступна, і параметри підключення (SSID, пароль), прошиті у WeMos D1, коректні.

Використання IoT-пристрою для вимірювань

1. Початок вимірювання: Розмістіть один палець на сенсорі MAX30102, а інший на датчику температури DS18B20.
2. Відображення даних: На LCD Keypad Shield почнуть відображатися поточні значення пульсу, SpO_2 та температури.

Встановлення та використання мобільного застосунку

1. Встановлення: Завантажте та встановіть мобільний застосунок.
2. Реєстрація / Авторизація: Зареєструйтеся або увійдіть до існуючого облікового запису.
3. Перегляд даних: На головному екрані (Dashboard) відобразяться останні отримані дані з IoT-пристрою. На екрані "Історія" доступні графіки та таблиці з динамікою показників.
4. Налаштування: У розділі "Налаштування" можна задати порогові значення для отримання сповіщень.