МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,
д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

IoT-система збору та аналізу біометричних даних для оцінки рівня стресу

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія»

Здобувач _____ Дмитро ЧЕРЕДНИЧЕНКО
підпис

«__» _____ 20__ р.

Керівник PhD, доцент,
каф. комп'ютерної інженерії _____ Євген ДАРНАПУК
підпис

«__» _____ 20__ р.

Миколаїв – 2026

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 202__ р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача
Чередниченка Дмитра Олександровича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи

IoT-система збору та аналізу біометричних даних для оцінки рівня стресу.

Затверджена наказом по ЧНУ ім. Петра Могили від 25.11.2025 № 294.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Розробка функціонального прототипу на базі IoT для збору, передавання, збереження та первинного аналізу біомедичних даних із використанням сенсорного модуля MAX30102, плати ESP8266 V3.0 CH340 та серверної частини на базі Python і Flask з відображенням отриманих значень у вебінтерфейсі.

4. Перелік питань, що підлягають розробці:

1) проаналізувати фізіологічні основи стресу та біомедичні показники, придатні для його орієнтовної оцінки;

2) дослідити IoT-технології, апаратні засоби збору даних і методи обробки біомедичних сигналів та визначити вимоги до системи;

3) розробити архітектуру функціонального прототипу на базі IoT для збору, передавання та аналізу біомедичних даних;

4) обґрунтувати вибір апаратних і програмних засобів, розробити схему підключення, алгоритм роботи та структуру серверної частини;

5) реалізувати апаратну, програмну та серверну частини функціонального прототипу;

6) провести експериментальну перевірку прототипу, проаналізувати отримані результати та визначити напрями подальшого вдосконалення.

5. Перелік графічних матеріалів

Презентація.

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Євген ДАРНАПУК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Дмитро ЧЕРЕДНИЧЕНКО
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «____» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: IoT-система збору та аналізу біометричних даних для оцінки рівня стресу.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	01.12.2025	10.12.2025	Виконано
2	Складання календарного плану КБР	11.12.2025	20.12.2025	Виконано
3	Огляд літератури та сучасних рішень	21.12.2025	31.12.2025	Виконано
4	Розробка проєктних рішень IoT-системи	05.01.2026	19.02.2026	Виконано
5	Проєктування та реалізація апаратно-програмної частини	20.02.2026	21.03.2026	Виконано
6	Тестування системи та аналіз результатів	24.03.2026	26.04.2026	Виконано
7	Збір та аналіз матеріалів до КБР	27.04.2026	21.05.2026	Виконано
8	Оформлення КБР, графічних матеріалів, додатків та презентації	27.04.2026	05.06.2026	Виконано
9	Перший передній захист КБР	22.05.2026	22.05.2026	Виконано
10	Другий передній захист КБР	05.06.2026	05.06.2026	Виконано
11	Завершення оформлення КБР та презентації	06.06.2026	08.06.2026	Виконано
12	Перевірка на академічний плагіат, фальсифікацію та списування	09.06.2026	10.06.2026	Виконано
13	Відгук керівника КБР	09.06.2026	10.06.2026	Виконано
14	Рецензування КБР	10.06.2026	13.06.2026	Виконано
15	Подання КБР, її електронної копії та інших документів (відгуку, рецензії)	15.06.2026	17.06.2026	Виконано
16	Захист кваліфікаційної бакалаврської роботи	24.06.2026	24.06.2026	Виконано

Керівник роботи _____ Євген ДАРНАПУК

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Дмитро ЧЕРЕДНИЧЕНКО

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«IoT-система збору та аналізу біометричних даних для оцінки рівня стресу»
Здобувач: Чередниченко Дмитро Олександрович
Керівник: PhD, доцент Дарнапук Євген Сергійович

Кваліфікаційна бакалаврська робота присвячена розробці функціонального прототипу системи на базі IoT для збору, передавання, збереження та первинного аналізу біомедичних даних, які можуть використовуватися для орієнтовної оцінки рівня стресу. Актуальність роботи пов'язана з потребою у технічних засобах, що дають змогу регулярно отримувати біомедичні показники в повсякденних умовах. Традиційні способи оцінювання стресу, зокрема опитування, самооцінка або медичні обстеження, не завжди забезпечують регулярне отримання об'єктивних даних, оскільки залежать від суб'єктивного сприйняття, спеціальних умов або участі фахівців.

Об'єктом дослідження є процес збору, передавання та аналізу біомедичних даних у системі на базі IoT. Предметом дослідження є методи та апаратно-програмні засоби збору, передавання, очищення, обробки й аналізу біомедичних даних для оцінки рівня стресу. Метою роботи є розробка функціонального прототипу IoT-системи, який забезпечує збір, передавання, збереження, очищення, обробку та первинний аналіз біомедичних даних.

Практичне значення роботи полягає у створенні доступного апаратно-програмного прототипу, який поєднує сенсорний модуль MAX30102, плату ESP8266 V3.0 CH340, серверну частину на базі Python/Flask, CSV-файл для збереження результатів і вебінтерфейс для перегляду отриманих значень. Розроблена система може використовуватися як навчальний або дослідний зразок для подальшого вдосконалення засобів попереднього моніторингу психофізіологічного стану людини.

Результати кваліфікаційної роботи апробовано під час XXVIII Всеукраїнської щорічної науково-практичної конференції «Могилянські читання – 2025» (м. Миколаїв, 10–14 листопада 2025 р.) та XXIII Міжнародної наукової конференції «Ольвійський форум – 2026: стратегії країн Причорноморського регіону в геополітичному просторі» (м. Миколаїв, 29 червня – 4 липня 2026 р.). За результатами дослідження підготовлено відповідні публікації матеріалів тез.

Пояснювальна записка складається зі вступу, трьох розділів, висновків, переліку джерел посилання та додатків. У вступі обґрунтовано актуальність теми, визначено мету, об'єкт, предмет, практичне значення та задачі роботи.

У першому розділі розглянуто фізіологічні основи стресу, його вплив на організм людини, біомедичні показники для орієнтовної оцінки стресового стану, IoT-технології, апаратні засоби збору даних, методи обробки сигналів і вимоги до

системи. У другому розділі виконано проєктування апаратно-програмної частини системи. Обґрунтовано вибір сенсорного модуля MAX30102, плати ESP8266 V3.0 CH340, середовища Arduino IDE, мови Python і фреймворку Flask. Сформовано архітектуру системи, логіку взаємодії між сенсором, мікроконтролером і серверною частиною та порядок налагодження прототипу. У третьому розділі реалізовано функціональний прототип системи – зібрано апаратний вузол, розроблено прошивку для ESP8266 V3.0 CH340, створено Flask-сервер, CSV-збереження даних і вебінтерфейс. Під час експериментальної перевірки підтверджено передавання даних від сенсора MAX30102 до мікроконтролера, серверної частини, CSV-файлу та вебінтерфейсу. Значення Stress Score та Stress Level розглядаються як орієнтовна програмна оцінка, а не як медичний висновок.

У висновках узагальнено результати роботи, підтверджено виконання поставлених завдань і досягнення мети. Визначено, що система працює як єдиний апаратно-програмний комплекс, а подальше вдосконалення може бути пов’язане з покращенням фіксації сенсора, розширенням алгоритмів обробки сигналу, додаванням інших біомедичних сенсорів і переходом від CSV-файлу до бази даних.

Кваліфікаційна бакалаврська робота містить 75 с. (без додатків), 38 рис., 16 табл., 27 джерел посилання та 3 додатки.

Ключові слова: *біомедичні дані, IoT-система, рівень стресу, сенсор MAX30102, ESP8266 V3.0 CH340, пульсовий сигнал, ЧСС, Python, Flask, CSV, вебінтерфейс.*

ABSTRACT
of the Bachelor's Thesis
"IoT-based system for biometric data collection and analysis to assess stress levels"
Applicant: Dmytro Cherednychenko
Supervisor: PhD, Associate Professor Yevhen Darnapuk

The bachelor's thesis is devoted to the development of a functional prototype of an IoT-based system for collecting, transmitting, storing and performing primary analysis of biomedical data that can be used for an approximate assessment of stress level. The relevance of the work is related to the need for technical tools that make it possible to regularly obtain biomedical indicators in everyday conditions. Traditional methods of stress assessment, in particular questionnaires, self-assessment or medical examinations, do not always provide regular objective data, as they depend on subjective perception, special conditions or the involvement of specialists.

The object of the study is the process of collecting, transmitting and analyzing biomedical data in an IoT-based system. The subject of the study is the methods, hardware and software tools for collecting, transmitting, cleaning, processing and analyzing biomedical data for stress level assessment. The aim of the work is to develop a functional prototype of an IoT system that provides collection, transmission, storage, cleaning, processing and primary analysis of biomedical data.

The practical significance of the work lies in the creation of an accessible hardware and software prototype that combines the MAX30102 sensor module, the ESP8266 V3.0 CH340 board, the server part based on Python/Flask, a CSV file for storing results and a web interface for viewing the obtained values. The developed system can be used as an educational or research prototype for further improvement of tools for preliminary monitoring of a person's psychophysiological state.

The results of the bachelor's thesis were presented at the XXVIII All-Ukrainian Annual Scientific and Practical Conference "Mohyla Readings – 2025" (Mykolaiv, November 10–14, 2025) and at the XXIII International Scientific Conference "Olvian Forum – 2026: Strategies of the Black Sea Region Countries in the Geopolitical Space" (Mykolaiv, June 29 – July 4, 2026). Based on the results of the study, relevant publications of conference abstracts were prepared.

The explanatory note consists of an introduction, three chapters, conclusions, a list of references and appendices. The introduction substantiates the relevance of the topic and defines the aim, object, subject, practical significance and tasks of the work.

The first chapter examines the physiological basis of stress, its influence on the human body, biomedical indicators for approximate assessment of stress state, IoT technologies, hardware tools for data collection, signal processing methods and system requirements. The second chapter presents the design of the hardware and software part of the system. The choice of the MAX30102 sensor module, the ESP8266 V3.0 CH340

board, the Arduino IDE environment, the Python programming language and the Flask framework is substantiated. The system architecture, the logic of interaction between the sensor, microcontroller and server part, as well as the procedure for setting up the prototype are formed. The third chapter describes the implementation of the functional prototype of the system – the hardware unit was assembled, firmware for the ESP8266 V3.0 CH340 was developed, a Flask server, CSV data storage and a web interface were created. During experimental testing, data transmission from the MAX30102 sensor to the microcontroller, server part, CSV file and web interface was confirmed. The Stress Score and Stress Level values are considered as an approximate software-based assessment, not as a medical conclusion.

The conclusions summarize the results of the work, confirm the completion of the assigned tasks and the achievement of the aim. It is determined that the system operates as a single hardware and software complex, while further improvement may be related to improving sensor fixation, expanding signal processing algorithms, adding other biomedical sensors and moving from a CSV file to a database.

The bachelor's thesis comprises 75 pages (excluding appendices), 38 figures, 16 tables, 27 references, and 3 appendices.

Keywords: *biomedical data, IoT system, stress level, MAX30102 sensor, ESP8266 V3.0 CH340, pulse signal, heart rate, Python, Flask, CSV, web interface.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПС – апаратно-програмна система

ВСР – варіабельність серцевого ритму

ІТ – інформаційні технології

КБР – кваліфікаційна бакалаврська робота

УМБ – універсальна мобільна батарея

ЧСС – частота серцевих скорочень

CSV – Comma-Separated Values

GPIO – General-Purpose Input/Output

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

I2C – Inter-Integrated Circuit

IoT – Internet of Things

IR – Infrared

USB – Universal Serial Bus

ВСТУП

Зростання психоемоційного навантаження та розвиток інформаційних технологій (ІТ) роблять проблему стресу дедалі актуальнішою. Люди щоденно перебувають під впливом різних подразників (тригерів), які впливають на фізичний та емоційний стан. До даних тригерів відносяться інтенсивний режим, нестача відпочинку, тривала напруженість тощо. Через це стрес поступово перетворюється на постійну реакцію організму на різні ситуації. В деяких випадках стрес допомагає швидше пристосуватися до стресової ситуації, проте його довготривалість несе негативний вплив. З цього виникає потреба у засобах, які надають змогу не лише фіксувати його показники, а й використовувати отримані дані для подальшої оцінки рівня стресу. Загальні методи оцінки стресу, різні опитування, анкетування, медичні обстеження надають лише об’єктивні результати без змоги безперервно відстежувати рівень стресу.

Медичні обстеження дають більш обґрунтовані результати, проте вони потребують часу, участі спеціалістів і відповідних умов. У цьому напрямі слід використати апаратно-програмну систему (АПС), яка допоможе організувати збір даних, їх передавати, очищувати та обробляти. У межах кваліфікаційної бакалаврської роботи (КБР) основна увага приділяється створенню системи, яка може збирати біомедичні дані, передавати їх до програмної частини та готувати до подальшого аналізу, оскільки окреме вимірювання не дає повного уявлення про стан людини.

Актуальність теми зумовлена поширенням стресових станів та потребою у своєчасному відстеженні змін психофізіологічного стану людини в повсякденних умовах. Традиційні способи оцінювання стресу, зокрема опитування, самооцінка або разові медичні обстеження, не завжди забезпечують регулярне отримання об’єктивних даних, оскільки залежать від суб’єктивних відповідей, участі фахівців або спеціальних умов проведення. Розв’язання цієї проблеми можливе шляхом розробки апаратно-програмного засобу на базі IoT, який забезпечує збір, передавання та первинний аналіз біомедичних даних для орієнтовної оцінки рівня стресу.

Метою кваліфікаційної роботи є розробка функціонального прототипу на базі IoT, який забезпечує збір, передавання, збереження, очищення, обробку та первинний аналіз біомедичних даних для орієнтовної оцінки рівня стресу.

Для досягнення поставленої мети слід вирішити наступні **задачі**:

– проаналізувати фізіологічні основи стресу та біомедичні показники, придатні для його орієнтовної оцінки;

– дослідити IoT-технології, апаратні засоби збору даних і методи обробки біомедичних сигналів та визначити вимоги до системи;

– розробити архітектуру функціонального прототипу на базі IoT для збору, передавання та аналізу біомедичних даних;

– обґрунтувати вибір апаратних і програмних засобів, розробити схему підключення, алгоритм роботи та структуру серверної частини;

– реалізувати апаратну, програмну та серверну частини функціонального прототипу;

– провести експериментальну перевірку прототипу, проаналізувати отримані результати та визначити напрями подальшого вдосконалення.

Об’єктом дослідження є процес збору, передавання та аналізу біомедичних даних у системі, побудованій на базі IoT.

Предметом дослідження є методи та апаратно-програмні засоби збору, передавання, очищення, обробки й аналізу біомедичних даних для оцінки рівня стресу.

Практичне значення полягає в розробці функціонального прототипу на базі IoT, який забезпечує збір, передавання, збереження та первинну обробку біомедичних даних. Запропоноване рішення може використовуватися для попереднього моніторингу психофізіологічного стану людини в повсякденних умовах, а також як основа для подальшого вдосконалення засобів оцінювання рівня стресу.

Апробація результатів кваліфікаційної роботи відбулася під час XXVIII Всеукраїнської щорічної науково-практичної конференції «Могилянські читання – 2025» (м. Миколаїв, 10–14 листопада 2025 р.) [1]; XXIII Міжнародна наукова конференція «Ольвійський форум – 2026: стратегії країн Причорноморського регіону в геополітичному просторі» (м. Миколаїв, 29 червня – 4 липня 2026 р.) [3].

1 ОСНОВИ ТЕОРЕТИЧНИХ ТА ТЕХНОЛОГІЧНИХ АСПЕКТІВ ОЦІНКИ СТРЕСУ

1.1 Фізіологічні основи стресу та його вплив на організм

Стрес – захисна реакція організму на дію зовнішніх або внутрішніх чинників, які порушують його звичний стан рівноваги та потребують мобілізації фізіологічних ресурсів. Цими чинниками можуть бути емоційне напруження, небезпека, біль, перевтома, нестача сну, інформаційне перевантаження або тривале перебування в умовах невизначеності. У короткочасній формі стрес може мати пристосувальне значення, оскільки допомагає організму швидше реагувати на зміну умов. Водночас тривала або багаторазово повторювана стресова реакція здатна негативно впливати на самопочуття, працездатність і загальний функціональний стан людини [2; 30].

Перед початком оцінки рівня стресу за біомедичними даними слід зрозуміти, які саме фізіологічні процеси в організмі змінюються під дією стресового тригера. Для АПС стрес не може розглядатися лише як суб’єктивне відчуття тривоги або напруження. З інженерної точки зору важливими є прояви, які можна зареєструвати або передати у вигляді цифрових даних для подальшого використання. Незалежно від походження подразника організм намагається адаптуватися до нових умов. В цей час він активізує внутрішні регуляторні механізми. Короткочасна стресова реакція має пристосувальне значення, а в ситуації небезпеки або підвищеного навантаження вона дає змогу швидше мобілізувати ресурси організму:

- підвищення увага;
- активізація серцево-судинної системи;
- посилення дихання;
- зміни м’язового тонусу.

Такий стан допомагає людині своєчасно реагувати на виниклі подразники, однак тривале або багаторазове повторення стресової реакції негативно впливає на організм. Згодом поступово втрачає можливість повноцінно відновлюватися, що може проявлятися у втомі, порушенні сну, зниженні концентрації, емоційній

нестабільності та погіршенні загального самопочуття. Формування самої реакції пов’язане з роботою кількох систем організму. Якщо ситуація потребує відповіді, активізується автономна нервова система, яка функціонує через взаємодію симпатичного та парасимпатичного відділів. Симпатична відповідає за реакцію мобілізації, а парасимпатична – за розслаблення та відновлення [4; 18]. Унаслідок зсуву балансу в бік симпатичної активації організм переходить у режим підвищеної готовності – змінюються частота серцевих скорочень (ЧСС), ритм дихання, тонус судин, активність потових залоз.

Для технічної системи дані зміни мають практичне значення, оскільки частина з них може фіксуватися сенсорними засобами та подаватися у вигляді біомедичних даних. Інформативним значенням є варіабельність серцевого ритму, яка в період мінливості часових інтервалів між послідовними ударами серця напряму відображає ступінь напруження регуляторних систем [4; 15; 24]. Зміни в активності потових залоз дають змогу зафіксувати електрошкірну активність, яка виступає в ролі індикатора емоційного збудження. Наприклад, зміна серцевого ритму або характеру пульсової хвилі свідчить про зміну функціонального стану організму. Однак дані показники не можна використовувати як прямий доказ стресу, оскільки вони залежать і від інших чинників:

- фізичного навантаження;
- температури середовища;
- втоми;
- якості сну;
- індивідуальних особливостей людини.

Окрім нервової регуляції, у стресовій реакції бере участь ендокринна система. Якщо дія подразника тривала, організм підтримує стан напруження за допомогою гормональних механізмів. Одним із гормонів, пов’язаних зі стресом, є кортизол, який бере участь у регуляції енергетичних ресурсів, впливає на рівень глюкози в крові та імунні процеси. У короткочасній перспективі така реакція допомагає організму адаптуватися до навантаження, однак за тривалого стресу постійна гормональна активність може негативно позначатися на функціональному

стані людини. Спираючись на багатогранність цих процесів, для побудови надійних алгоритмів оцінки стану людини доцільно використовувати мультимодальний підхід. Поєднання кількох біомедичних каналів дозволяє ефективніше відфільтровувати вплив сторонніх чинників і підвищує точність ідентифікації стресу апаратно-програмними засобами. Узагальнений опис процесу наведено на рис. 1.1.

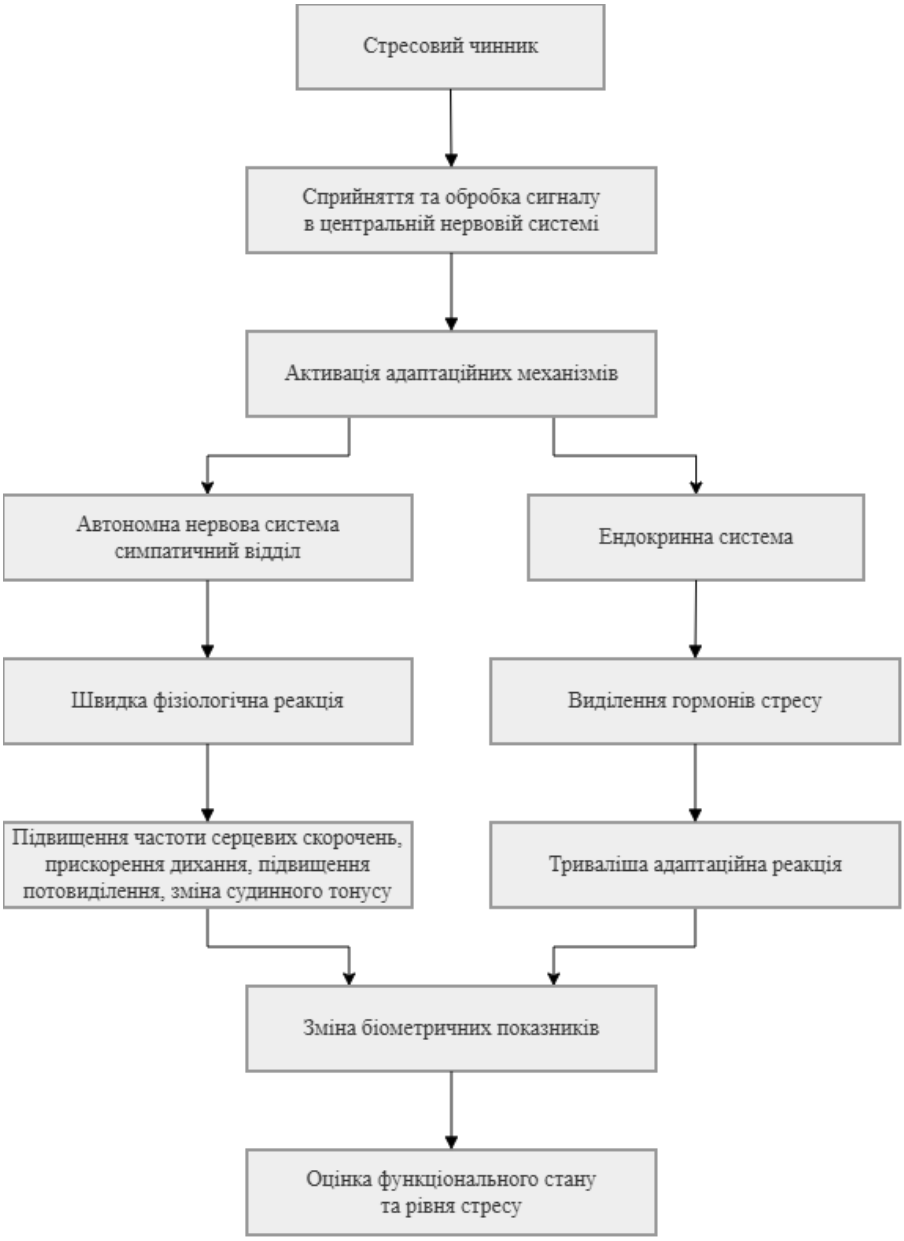


Рисунок 1.1 – Загальна схема фізіологічної реакції організму на стрес

На даному рисунку наведено загальну послідовність фізіологічної реакції організму на стрес. Стресовий чинник сприймається центральною нервовою системою, після чого активізуються автономні та гормональні механізми регуляції.

Їхня дія відображається на роботі серця, судин, дихальної системи та потових залоз. Саме ці зміни можуть бути перетворені на біомедичні показники й використані для подальшої обробки в системі на базі IoT.

Якщо говорити про тривалість стресу, то його поділяють на гострий і хронічний. Гострий стрес виникає швидко, зазвичай у відповідь на конкретну подію або короткочасне навантаження. Після припинення дії подразника організм поступово повертається до звичного стану. Хронічний стрес формується інакше і пов’язаний із тривалим або повторюваним впливом стресових чинників, а його наслідки поступово накопичуються. Порівняння видів стресу наведено у табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика гострого та хронічного стресу

Ознака	**Гострий стрес**	**Хронічний стрес**
Тривалість	Короткочасна реакція на подразник	Тривалий або повторюваний стан
Типові причини	Раптова подія, небезпека, сильне хвилювання	Недосипання, постійне навантаження, тривала невизначеність
Реакція організму	Швидка мобілізація ресурсів	Поступове виснаження адаптаційних можливостей
Можливий вплив	Допомагає швидше реагувати на ситуацію	Частіше погіршує самопочуття та працездатність
Типові прояви	Прискорене серцебиття, підвищена увага, напруження	Втома, порушення сну, зниження концентрації, емоційне виснаження

З результатів, наведених у таблиці, можна побачити, що для оцінки стану людини важливо враховувати не лише зміну окремого показника, а й умови, за яких він виник. Наприклад, підвищення ЧСС може бути наслідком як емоційного напруження, так і фізичної активності, втоми або інших зовнішніх умов. Отримані дані слід розглядати як непрямі ознаки, які спочатку потрібно очистити, порівняти та піддати подальшому аналізу фахівцями. Фізіологічна реакція на стрес створює підстави для використання окремих біомедичних показників у технічних системах. У межах системи на базі IoT сенсорні модулі не визначають стрес безпосередньо, а реєструють доступні для вимірювання сигнали, зокрема пульсову хвилю, частоту

серцевих скорочень або інші параметри, пов’язані зі зміною функціонального стану організму.

1.2 Біомедичні показники та методи оцінки рівня стресу

Після розгляду фізіологічної реакції організму на стрес потрібно визначити, які саме показники можна використати для її фіксації. АПС-система працює не з абстрактним поняттям напруження, а з конкретними параметрами, які вимірюються, передаються, очищуються від похибок і використовуються для подальшого аналізу. В системах такого типу увага приділяється біомедичним показникам, що відображають зміни у роботі серцево-судинних, нервових, дихальних та інших систем організму.

Одним з найбільш інформативних показників виступає ЧСС, яка показує, наскільки активно працює серце в певний момент часу, адже під час стресу організм переходить у режим підвищеної готовності – тоді серцевий ритм прискорюється. Цей показник зручний тим, що його можна отримати неінвазивно та без складного медичного обладнання. Використання лише ЧСС для оцінки стресу недостатньо, оскільки пульс може зростати не тільки через емоційне напруження, а й після фізичного навантаження, втомлення, підвищення температури навколишнього середовища, вживання кофеїну або інших чинників. Інформативним показником є варіабельність серцевого ритму (ВСР), яка характеризує не просто кількість серцевих скорочень, а зміну часових інтервалів між ними. У стані відносного спокою серце не працює абсолютно рівномірно – між окремими ударами є невеликі природні відхилення, що свідчать про гнучкість регуляції з боку автономної нервової системи. Коли наш організм відчуває стрес, симпатичний відділ нервової системи активізується сильніше, і серцевий ритм стає більш жорстким і менш варіабельним.

Для отримання інформації про роботу серця у компактних системах зазвичай використовується пульсова хвиля. Вона демонструє зміну кровонаповнення судин під час кожного серцевого скорочення. Такий сигнал можна реєструвати оптичним способом, коли сенсор випромінює світло на ділянку шкіри та фіксує зміну його

відбиття або поглинання. На основі цих даних формується графік, у якому можна визначити піки пульсової хвилі, а вже з них – ЧСС і часові інтервали між ударами. Для системи на базі IoT це оптимальний варіант, оскільки він дозволяє отримувати важливі дані без використання громіздкого обладнання [16; 18; 21]. Загальний принцип формування пульсового сигналу та отримання з нього параметрів показаний на рис. 1.2.

Рисунок 1.2 – Формування пульсового сигналу

На якість вимірювання впливають рухи користувача, неправильне розміщення сенсора, зовнішнє освітлення, слабкий контакт зі шкірою або індивідуальні особливості кровообігу [9; 18]. Отриманий первинний сигнал не можна одразу використовувати для оцінювання стану людини. Його потрібно попередньо очистити, відфільтрувати випадкові коливання, визначити коректні піки та лише після цього розрахувати потрібні параметри.

Крім серцевих показників, для оцінки стресу використовуються й інші показники – електродермальна активність, яка пов’язана з роботою потових залоз, оскільки під час емоційного напруження потовиділення може посилюватися навіть без фізичного навантаження. Температура шкіри може змінюватися через перерозподіл кровотоку та зміну тонусу судин. Показники дихання відображають частоту і характер дихальних рухів, які під час стресу стають швидшими або менш рівномірними. Однак велика кількість показників не завжди покращує систему. Кожен додатковий сенсор ускладнює апаратну частину, підвищує вимоги до живлення, збільшує обсяг даних і потребує окремої обробки. Для системи на базі

IoT слід використовувати показники, які пов’язані з серцевим ритмом і пульсовим сигналом, оскільки вони добре поєднують доступність вимірювання з практичною цінністю для аналізу. За допомогою апаратного підходу система автоматично збирає дані, які зберігаються в цифровому вигляді та обробляються за допомогою програмних засобів.

Біомедичні показники – базові для технічного оцінювання рівня стресу, але їх не можна розглядати як прямий і однозначний доказ певного стану. Найбільш придатними для компактної системи на базі IoT є параметри, які можна отримати неінвазивно, стабільно та з мінімальним дискомфортом для користувача. Подальша якість оцінювання залежить не лише від самого сенсора, а й від правильного очищення сигналу, виділення потрібних параметрів і коректного аналізу отриманих даних.

1.3 IoT-технології та апаратні засоби збору біомедичних даних

Після визначення біомедичних показників їх можна отримати, передати та підготувати до подальшої обробки. У межах такої системи сенсор не повинен працювати ізольовано. Він є лише початковою ланкою, яка фіксує фізіологічний сигнал, далі цей сигнал має бути переданий до програмної частини, де з нього можна сформувати набір даних. Система на базі IoT поєднує фізичний пристрій, засоби передавання даних і програмне середовище.

Під час збору біомедичних даних сигнал, отриманий від тіла людини, не залишається лише на рівні сенсора або мікроконтролера. Він передається далі – на вебсайт, серверну частину або іншу програмну платформу, де очищується, зберігається та використовується для оцінювання стану. У спрощеному вигляді така система працює послідовно – сенсор реєструє фізіологічний сигнал, мікроконтролер приймає отримані значення, формує дані для передавання, після чого вони надходять до вебінтерфейсу. У вебінтерфейсі відбуваються очищення, аналіз і візуалізація отриманих результатів. У цьому випадку апаратна частина відповідає за збір і передавання, а програмна – за роботу з отриманим датасетом [1; 14]. Структурна схема збору біомедичних даних продемонстрована на рис. 1.3.

Рисунок 1.3 – Структура системи на базі IoT для збору біомедичних даних

У цій схемі основним джерелом даних виступає сенсорний модуль. Для оцінювання рівня стресу можна використати різні типи апаратних засобів: оптичні сенсори пульсу, ЕКГ-сенсори, датчики електродермальної активності, температурні датчики та сенсори дихання. Кожен із цих варіантів має власне призначення, але не кожен підходить для компактної системи на базі IoT:

1) ЕКГ-сенсори надають точні дані про електричну активність серця, що корисно для медичних досліджень, однак у портативній системі цей підхід незручний. Потрібні електроди, правильне розміщення на тілі, стабільний контакт зі шкірою;

2) датчики електродермальної активності можуть фіксувати зміни провідності шкіри, що пов’язані з роботою потових залоз. Під час емоційного напруження цей показник змінюється, але він залежить від умов вимірювання – вологості, температури, стану шкіри, місця встановлення електродів;

3) температурні сенсори простіші, але температура шкіри не дає достатньої інформації про рівень стресу;

4) сенсори дихання потребують окремого вузла або складнішого розміщення, їх використання у цій системі не є доцільним;

5) для компактної системи на базі IoT більш зручним є оптичний спосіб реєстрації пульсового сигналу. Його головна перевага в тому, що вимірювання

виконується неінвазивно і не потребує електродів чи складного медичного обладнання;

6) оптичний сенсор фіксує зміну кровонаповнення тканин під час серцевих скорочень. На основі такого сигналу отримується пульсова хвиля, визначається ЧСС і виконується подальший розрахунок інтервалів між піками.

З огляду на практичну реалізацію системи, оптичний сенсорний модуль добре поєднується з мікроконтролером, підходить для невеликого пристрою і дозволяє отримувати показники, пов’язані з роботою серцево-судинної системи. Порівняння основних апаратних засобів збору біомедичних даних наведено в табл. 1.2.

Таблиця 1.2 – Порівняння апаратних засобів збору біомедичних даних для оцінювання рівня стресу

Апаратний компонент	**Тип отримуваних даних**	**Переваги**	**Обмеження**	**Рекомендації до використання у спеціалізованих проєктах**
Оптичний сенсор пульсу	Пульсова хвиля, частота серцевих скорочень, інтервали між ударами	Неінвазивність, компактність, зручність інтеграції з мікроконтролером	Чутливість до рухів, освітлення та контакту зі шкірою	Дає змогу отримати основні серцево-судинні показники у компактному пристрої
ЕКГ-сенсор	Електрична активність серця, серцевий ритм	Висока точність вимірювання серцевої діяльності	Потребує електродів, правильного розміщення та стабільного контакту	Підходить для медичних або лабораторних умов
Датчик електродермальної активності	Зміну електропровідності шкіри	Добре реагує на емоційне збудження	Залежить від вологості, температури та стану шкіри	Може бути корисним як додатковий сенсор
Температурний сенсор	Температуру шкіри або тіла	Простота вимірювання, низьке енергоспоживання	Значна залежність від зовнішніх умов	Не доцільний як основний показник, але може бути допоміжним
Сенсор дихання	Частоту та ритм дихання	Відображає зміну дихальної реакції при напруженні	Потребує додаткового обладнання або складнішого розміщення	Доцільний у складніших системах моніторингу

З таблиці видно, що універсального сенсора для оцінювання немає. ЕКГ-сенсор надає точний сигнал, але ускладнює використання пристрою. Датчики електродермальної активності та температури можуть доповнювати аналіз, однак значною мірою залежать від зовнішніх умов. Оптичний сенсор пульсу виглядає найбільш доцільним для компактної системи на базі IoT, оскільки дозволяє отримувати важливі серцево-судинні показники. У цій системі мікроконтролер виступає в ролі приймача даних від сенсора, готує їх до передавання і забезпечує зв’язок із програмною частиною. Від його можливостей залежить стабільність роботи сенсора, швидкість передачі даних і надійність системи. Для портативного забезпечення енергоживлення слід врахувати частоту передачі даних, тип датчика і сам мікроконтролер.

У практиці існує багато пристроїв, які частково виконують схожі функції. Найбільш вживані – смарт-годинники та фітнес-браслети, оскільки вони компактні, зручні для щоденного використання і можуть постійно відстежувати окремі показники стану людини. Наприклад, Apple Watch Series 9 (рис. 1.4) має оптичний сенсор серця, електричний сенсор серця та датчик температури, що робить його прикладом багатофункціонального персонального пристрою моніторингу.

Рисунок 1.4 – Приклад смарт-годинника Apple Watch Series 9

Фітнес-браслети мають схожу логіку використання, але простіші та доступніші – Fitbit Charge 6 (рис. 1.5) підтримує відстеження ЧСС і SpO_2 та має електричні сенсори. Цей пристрій добре підходить для повсякденного контролю

активності, але для розробки власної системи його можливості обмежені, оскільки користувач переважно працює з вже обробленими даними.

Рисунок 1.5 – Приклад фітнес-браслета Fitbit Charge 6

Окрему групу становлять спортивні смарт-годинники, які орієнтовані на контроль тренувань, відновлення та загального навантаження. Garmin Venu 3 (рис. 1.6) постійно відстежує ЧСС із зап’ястя та використовує дані серцевого ритму для оцінки стану організму, зокрема показників стресу та варіабельності серцевого ритму під час сну. Такі пристрої зручні для користувача, але вони також залишаються закритими комерційними системами, у яких неможливо повністю простежити логіку обробки сигналу.

Рисунок 1.6 – Приклад спортивного смарт-годинника Garmin Venu 3

Для дослідницьких задач існують спеціалізовані носимі пристрої. Одним із таких прикладів є Empatica E4 (рис. 1.7), який поєднує PPG, EDA, акселерометр і датчик температури шкіри.

Рисунок 1.7 – Приклад дослідницького носимого пристрою Empatica E4

Медичні моніторингові системи мають інший рівень точності та надійності. Philips IntelliVue X3 (рис. 1.8) належить до професійних моніторів пацієнта і може використовуватися для контролю таких параметрів, як ECG, SpO_2, неінвазивний артеріальний тиск і дихання. Дані системи працюють у контрольованих умовах і більше підходять для клінічного застосування.

Рисунок 1.8 – Приклад медичної моніторингової системи Philips IntelliVue X3

Узагальнення наведених прикладів продемонстровано в табл. 1.3.

Таблиця 1.3 – Порівняння технологій моніторингу

Технологія / приклад	Переваги	Недоліки
Apple Watch Series 9	1) компактність і зручність щоденного використання; 2) підтримка додаткових датчиків, зокрема температурних.	1) висока вартість; 2) закриті алгоритми обробки; 3) обмежений доступ до первинного сигналу.
Fitbit Charge 6	1) доступність і компактність; 2) відстеження ЧСС, SpO_2 та активності.	1) користувач працює переважно з готовими показниками; 2) обмежений доступ до сирих даних; 3) менша придатність для власної обробки сигналу.
Garmin Venu 3	1) постійний моніторинг серцевого ритму; 2) зручність для спортивного й повсякденного використання.	1) комерційна закрита система; 2) обмежений доступ до алгоритмів аналізу; 3) орієнтація більше на користувацький результат, ніж на дослідницьку обробку.
Empatica E4	1) орієнтація на дослідницьке використання; 2) підтримка PPG, EDA, температури та рухових даних.	1) вища вартість; 2) спеціалізоване призначення; 3) залежність від дослідницької платформи.
Philips IntelliVue X3	1) професійний клінічний моніторинг; 2) висока надійність у контрольованих умовах.	1) висока вартість; 2) складність використання; 3) потреба у медичному середовищі та кваліфікованому персоналі.
Розроблювана система на базі IoT	1) відкритий доступ до отриманих даних; 2) передавання сигналу до власного вебінтерфейсу; 3) можливість власного очищення й аналізу; 4) гнучкість для подальшого вдосконалення.	1) нижча ергономічність на етапі прототипу; 2) залежність від якості сенсорного сигналу та стабільності передавання.

IoT-підхід дозволяє поєднати сенсор, мікроконтролер, бездротову передачу та вебінтерфейс в одну систему. Для збору біомедичних даних у межах такої системи неінвазивний оптичний спосіб реєстрації пульсового сигналу забезпечує достатню інформативність, не ускладнює конструкцію пристрою і створює основу для подальшого очищення та аналізу даних у програмній частині.

1.4 Методи обробки та аналізу біомедичних сигналів

Збір біомедичних даних – початковий етап роботи всієї системи. Сигнал, одержаний від сенсора, не надає готової оцінки рівня стресу. Для мікроконтролера

або вебсистеми – послідовні числові значення, в яких наявні шуми або випадкові стрибки, які не придатні для аналізу. Задача обробки полягає у відокремленні частин сигналів від спотворень і підготовці даних до подальшої роботи. У портативних системах біомедичні сигнали не бувають стабільними від початку вимірювання до його завершення. На якість даних впливають рухи користувача, зміна положення сенсора, слабкий або нестабільний контакт зі шкірою, зовнішнє освітлення, електричні перешкоди та індивідуальні особливості кровообігу. Через це на графіку з’являються різкі перепади, пропуски, дрібні коливання або ділянки, які не відповідають реальному серцевому ритму. Якщо ці дані використовувати для оцінювання, система може неправильно визначити ЧСС або часові інтервали між ударами [9; 21].

Очищення – головний етап роботи з отриманими сигналами, метою якого є зменшення впливу випадкових похибок та шумів без втрати форми сигналу. Основними джерелами завад під час реєстрації біомедичних даних виступають артефакти руху, зміни зовнішнього освітлення та низькочастотний дрейф ізолінії. Щоб уникнути цих впливів, застосовують цифрову фільтрацію, згладжування та відкидання некоректних значень. У випадку пульсової хвилі слід не просто зробити графік рівнішим, а зберегти ділянки, за якими потім визначаються піки серцевих скорочень. Однак надлишкове згладжування зміщує та приглушує максимуми отриманого сигналу. Після очищення дані потрібно привести до вигляду, зручного для аналізу: за допомогою мінімаксного нормування значення приводяться до певного діапазону. Це корисно, коли вимірювання виконуються в різні моменти часу або за різних умов, оскільки амплітуда пульсової хвилі може змінюватися через силу притискання сенсора, особливості кровонаповнення тканин або положення руки.

Визначення піків пульсової хвилі – найважливіший етап у даній системі, оскільки кожен коректно знайдений систолічний максимум відповідає окремому серцевому скороченню. Однією з основних складностей алгоритмічного пошуку екстремумів – наявність дикротичного зубця на спаді хвилі, який система може хибно інтерпретувати як додаткове скорочення. Для уникнення цього застосовують

методи адаптивного порогу або аналіз похідної сигналу. Помилки на цьому етапі мають критичне значення – пропущений пік, подвійне спрацювання або хибний максимум змінюють часовий ряд і спотворюють результати подальших обчислень ВСР [9; 16]. Структурна схема послідовності обробки сигналу наведена на рис. 1.9.

Рисунок 1.9 – Послідовність обробки біомедичного сигналу

Дана блок-схема демонструє, що між моментом вимірювання та кінцевим результатом існує кілька етапів. Від якості кожного з яких залежить, наскільки коректною буде подальша оцінка стану користувача.

ВСР характеризує нерівномірність інтервалів між ударами серця. У стані спокою серцевий ритм має природні коливання, у той час як під час стресу вплив симпатичної нервової системи посилюється і ритм може ставати одноманітним. Зниження варіабельності – одна з ознак підвищеного напруження організму. Для аналізу отриманих параметрів можна застосовувати різні підходи. Найпростіший у застосуванні – пороговий аналіз, коли значення порівнюються з наперед заданими межами. Наприклад, підвищення ЧСС у поєднанні зі зменшенням варіабельності вказує на напружений стан.

Цей метод легко реалізується, однак він не враховує індивідуальні особливості користувача. Практичним є порівняння показників із базовим станом конкретної людини – система орієнтується не лише на загальні межі, а й на значення, які є типовими для користувача у стані відносного спокою. Це дозволяє

точніше визначати відхилення, оскільки нормальний пульс і ВСР можуть суттєво відрізнятися у різних людей. Статистичні методи дають можливість оцінювати не окреме значення, а поведінку сигналу в певному часовому інтервалі. Можна аналізувати середнє значення інтервалів між ударами, їх розкид, зміну показників у динаміці. Цей підхід краще відповідає задачі оцінювання стресового навантаження, оскільки стрес проявляється не одним різким стрибком, а поступовою зміною параметрів.

У складніших системах можна застосувати методи машинного навчання. Вони дозволяють працювати з набором ознак, а не з одним показником. Наприклад, модель може враховувати ЧСС, інтервали між піками, амплітуду сигналу, стабільність пульсової хвилі тощо. На основі такого набору можна класифікувати стан користувача як нормальний, помірно напружений або такий, що має ознаки підвищеного стресового навантаження. Проте використання машинного навчання потребує датасетів, попередньої розмітки даних і перевірки точності моделі [5; 19]. Порівняння основних методів обробки та аналізу біомедичних сигналів надано в табл. 1.4.

Таблиця 1.4 – Методи обробки та аналізу біомедичних сигналів

Метод	**Призначення**	**Переваги**	**Обмеження**
Фільтрація сигналу	Усунення шумів і небажаних коливань	Підвищує якість первинного сигналу	Потребує правильного вибору параметрів
Згладжування	Зменшення випадкових дрібних коливань	Спрощує подальше визначення піків	Може спотворювати форму сигналу при надмірному застосуванні
Нормалізація	Приведення значень до зручного діапазону	Полегшує порівняння даних	Не усуває фізичні причини похибок
Визначення піків	Пошук моментів серцевих скорочень	Дає основу для розрахунку ЧСС і ВСР	Чутливе до шумів і артефактів
Пороговий аналіз	Порівняння показників із заданими межами	Простий у реалізації та поясненні	Не завжди враховує індивідуальні особливості
Статистичний аналіз	Оцінка зміни показників у часовому проміжку	Дозволяє аналізувати динаміку стану	Потребує достатнього обсягу даних
Машинне навчання	Класифікація стану за набором ознак	Може враховувати складні залежності	Потребує якісного датасету та навчання моделі

Частина з наведених у таблиці методів потрібна для підготовки сигналу, інші – для виділення показників або формування оцінки. Для запропонованої системи першочерговими є методи, які забезпечують стабільну роботу з пульсовим сигналом. Окремо потрібно враховувати якість вихідних даних: якщо сигнал містить багато шуму, пропущені піки або хибні максимуми, то навіть складний метод аналізу не забезпечить достовірного результату.

Практична реалізація починається з контролю якості сигналу, під час якого непридатні для аналізу фрагменти потрібно відкидати або позначати як ненадійні. Для запропонованої системи на базі IoT розподіл обробки між пристроєм і вебінтерфейсом має виконуватися для базових операцій, пов’язаних зі зчитуванням і попередньою підготовкою даних. Більш детальне очищення, аналіз, збереження та візуалізацію виконуються у програмній частині, куди передається отриманий датасет. Це зменшує навантаження на мікроконтролер і залишає можливість надалі змінювати або вдосконалювати алгоритми без суттєвої зміни апаратної частини. Обробка біомедичних сигналів – проміжна ланка між апаратним вимірюванням і оцінюванням рівня стресу. Сенсор лише реєструє фізіологічний сигнал, але корисною інформацією він стає після очищення, виділення інформативних ознак і подальшого аналізу.

1.5 Визначення технічних та функціональних вимог до системи

Після аналізу фізіологічних основ стресу, біомедичних показників, апаратних засобів збору даних та методів їх обробки необхідно визначити вимоги до системи, що розробляється. На даному етапі формується опис підключення та характеристик, яким має відповідати система на базі IoT. Вона має забезпечувати повний цикл роботи з біомедичними даними:

- зчитування пульсового сигналу;
- передавання отриманих значень до програмного середовища;
- очищення від похибок;
- обробку;
- аналіз;

– подання результатів у зрозумілому вигляді.

Водночас така система не повинна розглядатися як медичний діагностичний пристрій, а її призначення полягає у попередньому технічному моніторингу показників, які можуть бути пов’язані зі зміною функціонального стану людини. Загальна структура системи включає сенсорний вузол, мікроконтролерний блок, канал бездротового передавання даних та програмне середовище для приймання, збереження, обробки й візуалізації отриманих даних. Сенсорний вузол відповідає за реєстрацію пульсового сигналу, мікроконтролер – за приймання й передавання даних, а програмна частина – за очищення, збереження, аналіз і візуалізацію отриманого датасету. Живлення системи має бути організоване так, щоб пристрій стабільно працював протягом усього сеансу тестування. Основні функціональні вимоги наведено в табл. 1.5.

Таблиця 1.5 – Специфікація необхідних вимог

Параметр	**Необхідне або рекомендоване значення**	**Призначення**
Тип вимірювання	Неінвазивна реєстрація пульсового сигналу	Отримання біомедичних даних без електродів і складної підготовки
Частота дискретизації	Не менше 50 Гц, рекомендовано 50–100 Гц	Фіксація форми пульсової хвилі та визначення її піків
Діапазон ЧСС	40–180 уд/хв	Робота з типовими значеннями серцевого ритму
Інтервали між піками	Орієнтовно 300–2000 мс	Виявлення помилкових або фізіологічно некоректних значень
Точність визначення ЧСС	Орієнтовно 3–5 уд/хв у стабільних умовах	Достатньо для попереднього моніторингу
Точність визначення інтервалів	Не гірше 20–30 мс у тестових умовах	Потрібна для аналізу ритмічності серцевих скорочень
Рекомендована тривалість вимірювання	2–5 хв	Формування стабільнішого набору даних
Передавання даних	Бездротове передавання через Wi-Fi, з інтервалом оновлення 1–5 с	Передавання значень до програмного середовища
Формат даних	Структурований формат із можливістю збереження у CSV	Зручність накопичення, перегляду та подальшої обробки отриманих біомедичних даних
Живлення	Через USB-підключення від ноутбука або УМБ	Забезпечення стабільної роботи прототипу під час тестування
Робочі умови	Температура 10–40 °C, вологість до 80 %	Типові умови використання прототипу в приміщенні
Програмне середовище	Власний вебінтерфейс або Flask-сервер	Приймання, очищення, збереження, аналіз і візуалізація даних

Окрім технічних характеристик, система повинна бути зручною для тестування, щоб запуск збору даних виконувався у кілька простих кроків, а результат відображався у зрозумілому вигляді – у формі графіка, числових значень чи короткого узагальненого результату. При цьому система повинна залишатися відкритою для подальшого вдосконалення, наприклад, додавання нових алгоритмів обробки або розширення набору показників.

У портативних умовах вимірювання можуть виникати рухові артефакти, короткочасні пропуски, різкі викиди або нестабільний контакт сенсора зі шкірою. Такі ділянки не можна автоматично використовувати для оцінювання, тому програмна частина повинна виявляти некоректні значення і не враховувати їх під час подальшого аналізу. Це особливо важливо для пульсового сигналу, оскільки помилка у визначенні піка впливає не лише на поточне значення ЧСС, а й на розрахунок інтервалів між ударами. Для перевірки виконання вимог доцільно використовувати декілька простих критеріїв, які охоплюють повний цикл роботи системи від отримання сигналу до подання результату, наведено в табл. 1.6.

Таблиця 1.6 – Критерії перевірки працездатності розроблюваної системи

Етап перевірки	Що перевіряється	Очікуваний результат
Отримання сигналу	Наявність пульсової хвилі протягом не менше 60 с	Сигнал має форму, у якій можна визначити піки
Розрахунок ЧСС	Визначення ЧСС	Значення перебуває в межах 40–180 уд/хв
Аналіз інтервалів	Розрахунок часу між сусідніми піками	Інтервали не мають очевидно помилкових значень менше 300 мс або більше 2000 мс
Передавання даних	Надходження значень до програмного середовища	Дані оновлюються з інтервалом не більше 5 с
Очищення сигналу	Виявлення шумів, пропусків і різких викидів	Некоректні фрагменти не використовуються для формування результату
Робота вебінтерфейсу	Приймання, збереження та відображення даних	Користувач бачить графік, числові значення або узагальнений результат
Повний цикл роботи	Взаємодія апаратної та програмної частин	Виконується послідовність

Сформульовані вимоги визначають межі подальшої розробки системи на базі IoT, які демонструють функції, що повинні бути реалізовані та значення, які є достатніми для прототипу та за якими критеріями перевіряється його

працездатність. Основна увага під час подальшої реалізації має бути спрямована на стабільне отримання пульсового сигналу, його передавання до вебінтерфейсу, очищення від похибок і підготовку до аналізу. Окремо слід передбачити можливість збереження отриманих значень у CSV-форматі, оскільки це спрощує накопичення результатів вимірювання, їх перегляд і подальшу первинну обробку біомедичних даних. Саме ці вимоги будуть використані як основа для побудови архітектури, вибору апаратних засобів і реалізації програмної частини системи.

Висновки до розділу 1

У першому розділі було розглянуто теоретичні та технологічні основи розробки системи на базі IoT для збору й первинного аналізу біомедичних даних з метою орієнтовної оцінки рівня стресу. Визначено, що стрес є не лише психоемоційним станом, а й фізіологічною реакцією організму, яка супроводжується змінами в роботі серцево-судинної, нервової, дихальної та ендокринної систем. Саме ці зміни створюють підстави для використання окремих біомедичних показників у технічних системах, однак їх не можна розглядати як прямий і однозначний доказ стресу. Було встановлено, що для компактного функціонального прототипу найбільш доцільними є показники, пов’язані з серцевим ритмом і пульсовою хвилею. Частота серцевих скорочень, інтервали між піками пульсової хвилі та варіабельність серцевого ритму можуть відображати зміну функціонального стану людини, проте на них впливають і сторонні чинники – фізичне навантаження, втома, температура середовища, якість сну та індивідуальні особливості користувача. Через це отримані дані потребують попереднього очищення, перевірки якості та подальшого аналізу.

Огляд апаратних засобів збору біомедичних даних показав, що для системи на базі IoT доцільно використовувати неінвазивний оптичний спосіб реєстрації пульсового сигналу. Порівняння оптичних сенсорів, ЕКГ-сенсорів, датчиків електродермальної активності, температурних сенсорів і сенсорів дихання підтвердило, що оптичний сенсор пульсу краще відповідає вимогам компактності, простоти підключення та зручності використання в умовах прототипу. При цьому

комерційні носимі пристрої та медичні моніторингові системи мають свої переваги, але часто є закритими, дорогими або орієнтованими на клінічне середовище, що обмежує можливість власної обробки первинних даних. Проаналізовано основні методи обробки та аналізу біомедичних сигналів та встановлено, що первинний сигнал, отриманий від сенсора, не є готовим результатом для оцінювання стану людини. Його необхідно очистити від шумів, згладити, нормалізувати, визначити коректні піки пульсової хвилі та лише після цього використовувати для розрахунку інформативних параметрів.

У розділі сформульовано технічні та функціональні вимоги до розроблюваної системи на базі IoT. Визначено, що прототип має забезпечувати неінвазивне зчитування пульсового сигналу, передавання даних через бездротовий канал, їх збереження, очищення, первинну обробку та відображення результатів у вебінтерфейсі. Окремо враховано вимоги до частоти дискретизації, допустимого діапазону ЧСС, інтервалів між піками, стабільності живлення під час тестування та формату збереження даних. Використання CSV-формату є доцільним для накопичення результатів вимірювання, їх перегляду та подальшої первинної обробки без ускладнення системи додатковими засобами зберігання.

Проведений аналіз дав змогу перейти від загального теоретичного опису стресу до визначення практичних вимог до майбутнього прототипу. На основі результатів першого розділу в подальшому буде обґрунтовано вибір апаратних компонентів, розроблено архітектуру системи на базі IoT, описано підключення сенсорного модуля до мікроконтролера, логіку передавання даних і структуру програмної частини для їх приймання, збереження та відображення.

2 ПРОЄКТУВАННЯ СИСТЕМИ ЗБОРУ ТА ПЕРЕДАВАННЯ БІОМЕДИЧНИХ ДАНИХ НА БАЗІ IOT

2.1 Загальна концепція та принцип роботи системи на базі IoT

Основна ідея побудови системи на базі IoT полягає в розподілі задач між апаратною та програмною частинами. Апаратна частина відповідає за отримання біомедичного сигналу та його передавання, а програмна – за приймання даних, збереження, попередню обробку і подання результатів у зручному вигляді. Джерелом первинного сигналу в системі виступає сенсор MAX30102 (рис. 2.1), що використовується для оптичної реєстрації пульсового сигналу, який виникає внаслідок зміни кровонаповнення тканин під час серцевих скорочень. У результаті роботи сенсора формуються значення IR та RED каналів, які надалі можуть бути використані для побудови пульсової хвилі, визначення її піків і розрахунку показників, пов’язаних із роботою серцево-судинної системи [6; 21].

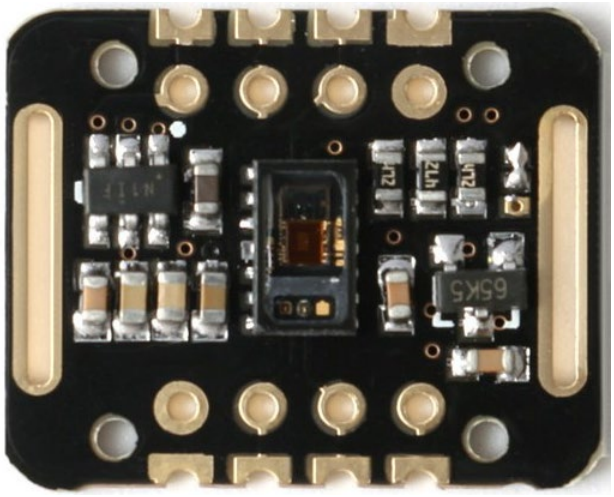


Рисунок 2.1 – Зображення сенсорного модулю MAX30102

Для запропонованої системи сенсорний модуль обирався не лише за принципом доступності, а й за можливістю практичного використання в компактній системі на базі IoT. Сенсор має бути невеликим, не вимагати електродів або складного закріплення, підтримувати підключення до мікроконтролерної плати та формувати дані, які можна надалі обробляти програмно. Порівняння можливих сенсорних модулів наведено в табл. 2.1.

Таблиця 2.1 – Порівняння сенсорних модулів для збору біомедичних даних

Характеристика	MAX30102	Pulse Sensor	AD8232
Тип сигналу	Оптичний пульсовий сигнал	Оптичний пульсовий сигнал	ЕКГ-сигнал
Вихідні дані	IR та RED значення	Аналогова пульсова хвиля	Аналоговий ЕКГ-сигнал
Метод вимірювання	Фотоплетизмографія	Фотоплетизмографія	Електрокардіографія
Інтерфейс підключення	I2C	Аналоговий вихід	Аналоговий вихід
Напруга живлення	3,3 В	3–5 В	2,0–3,5 В
Основні канали / електроди	RED та IR LED	1 оптичний канал	2–3 електроди
Частота дискретизації / вихід	50–3200 виб./с	Аналогове значення АЦП	Аналоговий вихід після підсилення
Споживання струму	Залежить від режиму LED	4 мА	170 мкА
Практичне обмеження	Чутливість до рухів і зовнішнього світла	Менше даних для подальшого аналізу	Потребує електродів і правильного розміщення

З таблиці 2.1 видно, що сенсор MAX30102 дозволяє отримувати пульсовий сигнал оптичним способом, не потребує електродів і може бути підключений до мікроконтролерної плати через I2C. Для задачі збору біомедичних даних у компактній системі цього достатньо, оскільки основна подальша робота виконується у програмному середовищі після збереження отриманих значень [6; 16; 21].

Керуючим вузлом системи виступає ESP8266 V3.0 CH340 (рис. 2.2), який приймає значення від сенсора MAX30102, формує пакет даних і передає його до вебінтерфейсу через Wi-Fi, а наявність модуля CH340 спрощує підключення плати до комп’ютера під час програмування та налагодження. Передавання даних у системі доцільно виконувати саме через Wi-Fi, оскільки вебсайт або серверна частина повинні приймати дані без постійного фізичного підключення пристрою до комп’ютера. ESP8266 V3.0 CH340 відповідає основним вимогам системи на базі IoT, плата має вбудований Wi-Fi, що дозволяє передавати дані до серверної частини без додаткового модуля зв’язку, підтримує I2C, що дає змогу підключити сенсор MAX30102 без складної схеми [11; 29].

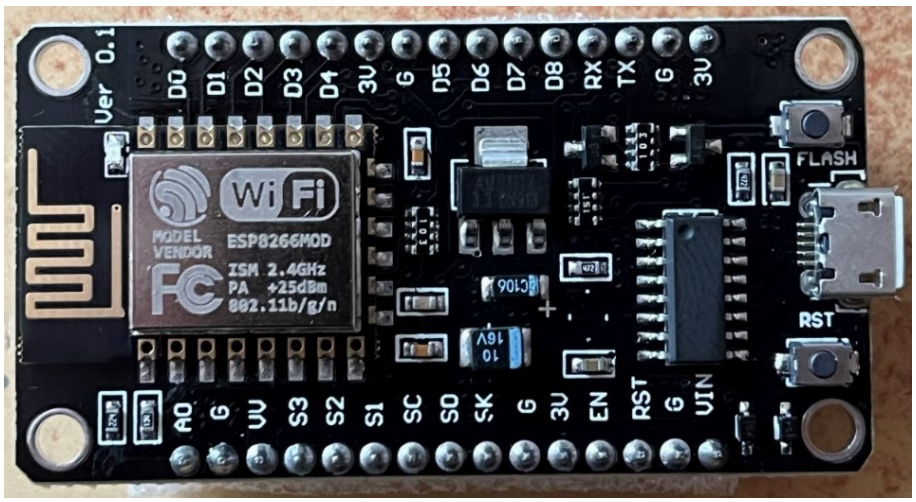


Рисунок 2.2 – Зображення плати ESP8266 V3.0 CH340

Вибір мікроконтролерної плати пов’язаний із тим, що пристрій повинен мати бездротовий зв’язок, підтримувати підключення сенсора через I2C і залишатися достатньо простим для реалізації в межах прототипу. Для цієї системи достатньо стабільно зчитувати значення з сенсора MAX30102 та передавати їх на сервер, де вже виконуються збереження, очищення, аналіз і візуалізація. Порівняння можливих мікроконтролерних плат наведено в табл. 2.2.

Таблиця 2.2 – Порівняння мікроконтролерних плат для IoT-системи

Характеристика	**ESP8266 V3.0 CH340**	**Arduino Uno**	**Raspberry Pi Pico W**
Мікроконтролер / процесор	ESP8266EX	ATmega328P	RP2040
Архітектура	32-bit Tensilica L106	8-bit AVR	Dual-core Arm Cortex-M0+
Тактова частота	80 / 160 МГц	16 МГц	133 МГц
Робоча логіка	3,3 В	5 В	3,3 В
Бездротовий зв’язок	Wi-Fi 802.11 b/g/n	Немає вбудованого Wi-Fi	Wi-Fi
GPIO	11	14	26
Аналогові входи	1	6	3
Підтримка I2C	Так	Так	Так
Пам’ять	4 МБ Flash, 64 КБ SRAM	32 КБ Flash, 2 КБ SRAM	2 МБ Flash, 264 КБ SRAM
Підключення до ПК	USB через CH340	USB	USB
Середовище програмування	Arduino IDE	Arduino IDE	MicroPython / C/C++
Особливість для цієї системи	Достатній для збору й передавання даних	Потребує WiFi-модуля	Потребує іншої логіки програмування

З таблиці 2.2 видно, що ESP8266 V3.0 CH340 відповідає поставленим вимогам системи. ESP32 у даній роботі не розглядалася як основний варіант керуючого вузла, оскільки фактична реалізація прототипу виконується на базі ESP8266 V3.0 CH340. ESP32 має ширші технічні можливості, зокрема більшу продуктивність, більшу кількість GPIO та підтримку Bluetooth, однак для поставленої задачі ці можливості не обов’язкові. У запропонованій системі мікроконтролерна плата виконує не повну обробку біомедичного сигналу, а зчитування значень із сенсора MAX30102 та передавання їх через Wi-Fi до серверної частини, тому використання ESP32 може ускладнити опис апаратної реалізації без суттєвої переваги для обраної архітектури. Основна обробка, очищення, збереження та візуалізація даних виконуються на стороні Python + Flask, тому ресурсів ESP8266 V3.0 CH340 достатньо для реалізації пристрою збору біомедичних даних.

Серверна частина системи передбачається на базі Python та фреймворку Flask. Python добре підходить для роботи з числовими даними та біомедичними сигналами, а Flask дозволяє швидко створити серверну частину без зайвої складності структури проєкту. Для розроблюваної системи важливо, щоб сервер міг приймати HTTP-запити від мікроконтролера, обробляти дані у форматі CSV, зберігати отримані значення та передавати їх на сторінку візуалізації. Flask має просту маршрутизацію, не потребує складного налаштування і добре поєднується з бібліотеками Python для обробки даних [16]. Формат CSV легко відкривається в табличних редакторах, добре обробляється засобами Python і зберігає часові мітки, значення пульсового сигналу та додаткові параметри, які надалі будуть використані для аналізу.

Загальний рух даних у системі виконується як послідовність – сенсор MAX30102 реєструє пульсовий сигнал, ESP8266 V3.0 CH340 зчитує отримані значення, після чого через Wi-Fi передає їх на Flask-сервер. На серверній частині дані приймаються, записуються у CSV-файл, проходять попередню перевірку, очищення та подаються у вебінтерфейсі у вигляді графіка чи числових значень. Взаємодія апаратної та програмної частин системи наведено на рис. 2.3.

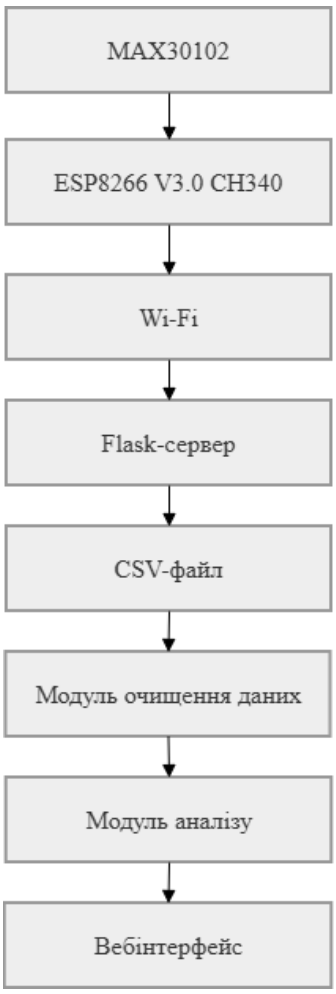


Рисунок 2.3 – Структурна схема системи на базі IoT

Дана структурна схема демонструє два рівні роботи системи:

1) Апаратний рівень включає сенсор MAX30102 та плату ESP8266 V3.0 CH340, функція якого полягає у реєстрації первинного сигналу, зчитуванні цифрових значень та передаванні цих значень до серверної частини.

2) Програмний рівень включає Flask-сервер, файл збереження даних у форматі CSV, модуль очищення, модуль аналізу та вебінтерфейс для відображення результатів. Цей розподіл дає змогу відокремити етап збору даних від етапу їх програмної обробки. На ESP8266 V3.0 CH340 виконуються лише операції, необхідні для роботи пристрою збору – зчитування значень, формування даних і передавання їх на сервер. Очищення сигналу, перевірка коректності значень, аналіз і візуалізація виконуються вже у вебінтерфейсі та серверній частині, що зменшує навантаження на мікроконтролерну плату і спрощує подальше вдосконалення алгоритмів.

Обрана концепція підходить для реалізації прототипу IoT-системи збору біомедичних даних, в якій сенсор MAX30102 забезпечує отримання первинного пульсового сигналу, ESP8266 V3.0 CH340 виконує роль керуючого вузла та засобу бездротового передавання, а серверна частина на базі Python + Flask приймає, зберігає і готує дані до подальшої обробки [6; 11; 20].

2.2 Апаратне підключення та конфігурація пристрою

Підключення сенсора MAX30102 до ESP8266 V3.0 CH340 виконується через цифровий інтерфейс I2C, використовуються дві сигнальні лінії – SCL і SDA, а також живлення та спільна земля. У типовому варіанті для ESP8266 лінія SCL підключається до контакту D1 / GPIO5, а лінія SDA – до контакту D2 / GPIO4. Цей варіант зручний для програмування в Arduino IDE, оскільки ці контакти використовуються як стандартні для I2C-з’єднання на платах ESP8266. Живлення сенсорного модуля подається від стабілізованої напруги 3,3 В, оскільки ESP8266 працює з 3,3-вольтовими логічними рівнями. Спільна земля обов’язково з’єднується між усіма вузлами системи, якщо земля сенсора, плати та живлення не буде спільною, обмін даними може бути нестабільним або сенсор взагалі не буде визначатися під час ініціалізації [6; 11–12]. Відповідність контактів підключення наведено в табл. 2.3.

Таблиця 2.3 – Підключення сенсора MAX30102 до ESP8266 V3.0 CH340

Контакт MAX30102	Контакт ESP8266 V3.0 CH340	Призначення
VIN / VCC	3V3	Живлення сенсорного модуля
GND	GND	Спільна земля системи
SCL	D1 / GPIO5	Лінія тактування інтерфейсу I2C
SDA	D2 / GPIO4	Лінія передавання даних інтерфейсу I2C
INT	Вільний GPIO	Застосовується для переривань за потреби

Принципову схему підключення сенсорного модуля MAX30102 до ESP8266 V3.0 CH340 наведено на рис. 2.4.

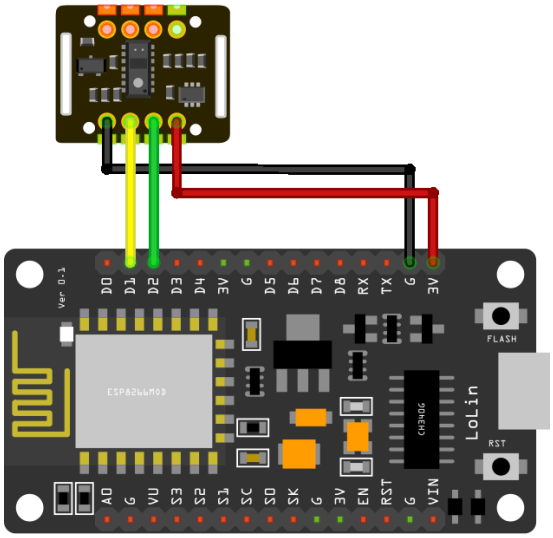


Рисунок 2.4 – Принципова схема підключення сенсора MAX30102 до ESP8266 V3.0 CH340

На рисунку 2.4 показано підключення сенсорного модуля MAX30102 до плати ESP8266 V3.0 CH340. Для обміну даними використовується інтерфейс I2C: лінія SCL підключається до контакту D1 / GPIO5, а лінія SDA – до контакту D2 / GPIO4. Живлення сенсора подається від контакту 3V3, а контакт GND з’єднується зі спільною землею плати. Робота інтерфейсу I2C у цій системі зручна тим, що для обміну даними потрібно лише дві сигнальні лінії, що спрощує схему підключення і залишає інші контакти плати вільними для можливого подальшого розширення. I2C-з’єднання потребує акуратного монтажу, бо нестійкий контакт на лініях SCL або SDA може призвести до того, що сенсор не буде визначатися програмою або передаватиме некоректні значення [10].

Плата ESP8266 V3.0 CH340 під час активного WiFi-з’єднання може споживати більше струму, ніж у режимі очікування. Якщо живлення буде нестабільним, можливі перезавантаження плати, втрата WiFi-з’єднання або переривання передавання даних [9]. Для сенсора MAX30102 стабільність живлення також важлива, оскільки коливання напруги можуть впливати на якість отриманого пульсового сигналу. На етапі тестування живлення пристрою виконується через USB-підключення від ноутбука або від універсальної мобільної батареї (УМБ). Живлення від ноутбука зручне під час програмування плати, завантаження прошивки та перевірки роботи через Serial Monitor, а використання УМБ дозволяє

перевірити роботу прототипу без постійного підключення до комп’ютера. ESP8266 під час передавання даних через Wi-Fi може створювати короткочасні піки споживання струму. Якщо джерело живлення або USB-кабель не забезпечують достатньої стабільності, плата може працювати нестабільно навіть за правильної схеми підключення, що проявляється у самовільному перезавантаженні, втраті підключення до мережі або нерегулярному передаванні даних.

На платі ESP8266 V3.0 CH340 передбачено вбудований вузол стабілізації живлення, який забезпечує формування напруги 3,3 В для роботи мікроконтролера ESP8266. У разі живлення плати через USB вхідна напруга 5 В перетворюється до рівня, необхідного для роботи ESP8266.

Якість пульсового сигналу залежить не тільки від електричної схеми, а й від розміщення сенсора. Палець або інша ділянка шкіри повинні щільно прилягати до чутливої частини модуля, але без надмірного натиску, якщо контакт нестабільний, на графіку можуть з’являтися різкі викиди, пропуски або ділянки, які не відповідають реальній пульсовій хвилі. На якість вимірювання впливає і зовнішнє освітлення, оскільки сенсор MAX30102 працює за оптичним принципом, якщо на чутливу частину сенсора потрапляє зайве світло, показники можуть ставати менш стабільними. Під час тестування сенсор розміщується так, щоб контакт зі шкірою був рівномірним, а зовнішнє світло не потрапляло безпосередньо на фотоприймач [6; 9; 21].

Під час макетного складання важливе значення має довжина та надійність провідників, оскільки занадто довгі або погано зафіксовані з’єднання можуть спричиняти короткочасні збої в обміні по I2C або нестабільну роботу живлення. Найбільш чутливими виступають лінії SCL, SDA, живлення та спільна земля.

2.3 Програмне середовище та алгоритм роботи

Для програмування плати обрано середовище Arduino IDE, яке зручне, оскільки підтримує плати на базі ESP8266, дозволяє швидко завантажувати прошивку через USB-підключення та перевіряти роботу пристрою через послідовний монітор. Наявність модуля CH340 на платі спрощує підключення до

комп’ютера, однак для коректної роботи може знадобитися встановлення відповідного драйвера. Після цього в Arduino IDE обирається потрібна плата ESP8266, порт підключення та завантажується програма. Для роботи пристрою необхідні бібліотеки, які забезпечують обмін даними із сенсором, роботу з Wi-Fi та передавання HTTP-запитів [7; 12]. Основні програмні засоби наведено в табл. 2.4.

Таблиця 2.4 – Програмні засоби для роботи пристрою збору даних

Засіб / бібліотека	**Призначення**
Arduino IDE	Написання, компіляція та завантаження програми на ESP8266 V3.0 CH340
ESP8266 board package	Підтримка плат ESP8266 у середовищі Arduino IDE
Wire.h	Робота з інтерфейсом I2C для обміну даними з MAX30102
MAX30105.h / бібліотека MAX3010x	Ініціалізація сенсора MAX30102 та зчитування IR і RED значень
ESP8266WiFi.h	Підключення плати ESP8266 до WiFi-мережі
ESP8266HTTPClient.h	Формування та надсилання HTTP-запитів на Flask-сервер
Serial Monitor	Налагодження роботи пристрою та перегляд службових повідомлень

У програмі плата ESP8266 V3.0 CH340 виконує роль керуючого вузла, після запуску якого, вона ініціалізує послідовний порт, підключається до WiFi-мережі, перевіряє доступність сенсора MAX30102 через I2C і лише після цього переходить до зчитування даних. Цей порядок важливий, оскільки передавання значень на сервер має сенс тільки тоді, коли пристрій має мережеве з’єднання, а сенсор коректно визначається програмою.

На першому етапі в коді задаються параметри WiFi-мережі – назва мережі та пароль. Плата намагається встановити з’єднання і виводить службові повідомлення в послідовний монітор. Якщо підключення не виконано, пристрій повторює спробу. Після підключення до Wi-Fi виконується ініціалізація сенсора MAX30102. Програма звертається до сенсора через I2C і перевіряє, чи відповідає він на запит. Якщо сенсор не визначається, причина може бути в неправильному підключенні ліній SCL і SDA, відсутності спільної землі, нестабільному живленні або помилці в налаштуванні бібліотеки. Для налагодження в такому випадку доцільно виводити повідомлення про помилку в Serial Monitor. Після успішної ініціалізації

починається зчитування значень IR та RED каналів. Водночас RED-значення також зберігаються, тому що вони можуть бути корисними для подальшої перевірки якості сигналу або розширення алгоритму аналізу. На цьому етапі мікроконтролер не виконує складної обробки сигналу, а лише отримує первинні числові значення від сенсора.

Передавання даних на сервер виконується у форматі CSV-рядка. Один рядок даних може містити часову мітку або номер вимірювання, значення інфрачервоного каналу та значення червоного каналу Плата зчитує значення з сенсора, перетворює їх у текстовий рядок і надсилає на сервер через HTTP-запит, а алгоритм роботи пристрою має циклічний характер. Після запуску та успішної ініціалізації система не завершує роботу після одного вимірювання, а продовжує зчитувати нові значення з сенсора і надсилати їх на сервер [12; 20]. Це дозволяє накопичувати послідовність даних, з якої надалі можна побудувати графік пульсової хвилі та виконати попередню обробку. Блок-схему алгоритму функціонування пристрою збору біомедичних даних наведено в *додатку А.1*.

Основна послідовність роботи пристрою така:

1) запуск ESP8266 V3.0 CH340;
2) підключення до WiFi-мережі;
3) ініціалізація сенсора MAX30102;
4) перевірка доступності сенсора;
5) зчитування IR та RED значень;
6) формування CSV-рядка;
7) передавання даних на Flask-сервер;
8) перевірка успішності передавання;
9) перехід до наступного циклу вимірювання.

Якщо WiFi-з’єднання не встановлено, плата не переходить до передавання даних, оскільки сервер буде недоступним, якщо сенсор MAX30102 не відповідає, зупиняється зчитування або повторюється ініціалізація, якщо HTTP-запит не переданий успішно, система повторює спробу надсилання або переходить до наступного вимірювання з фіксацією помилки в послідовному моніторі.

Ці перевірки потрібні для того, щоб під час тестування було зрозуміло, на якому етапі виникла проблема. У програмній реалізації сенсор зчитує значення частіше, ніж вони надсилаються до Flask-сервера, що зменшує навантаження на мережеву частину і не створює надмірної кількості HTTP-запитів. Остаточні параметри залежать від стабільності WiFi-з’єднання, швидкості роботи сервера та необхідної деталізації сигналу.

Під час налагодження важливо використовувати послідовний монітор Arduino IDE через який перевіряється підключення до Wi-Fi, наявність відповіді від сенсора MAX30102, поточні значення IR та RED каналів і результат HTTP-запиту, що дозволяє окремо перевірити кожен етап роботи пристрою – апаратне підключення, зчитування сигналу та передавання даних. Якщо значення сенсора змінюються при контакті пальця з модулем, а сервер отримує нові рядки – пристрій збору даних працює коректно. Розроблений алгоритм забезпечує роботу апаратного вузла системи – ESP8266 V3.0 CH340 приймає дані від сенсора MAX30102, формує їх у простому форматі та передає на сервер через Wi-Fi.

2.4 Проєктування серверної частини та вебінтерфейсу

Для реалізації серверної частини обрано мову програмування Python та фреймворк Flask. Цей вибір пов’язаний із тим, що розроблювана система працює не лише як вебінтерфейс для перегляду результатів, а як програмне середовище для приймання, збереження та подальшої обробки числових даних. Значення, отримані від ESP8266 V3.0 CH340, потребують перевірки, накопичення, візуалізації та подальшого аналізу, тому важливо використовувати мову, яка зручна саме для роботи з датасетами й біомедичними сигналами.

Python добре підходить для цієї задачі, оскільки має простий синтаксис, зручні засоби роботи з файлами та велику кількість бібліотек для обробки даних, побудови графіків і математичних обчислень. У подальших етапах роботи це дозволить використовувати одне програмне середовище не тільки для запуску вебсервера, а й для очищення сигналу, фільтрації, визначення піків та розрахунку

необхідних показників. Flask у цій системі виконує роль легкого вебфреймворку, за допомогою якого можна швидко створити маршрути для приймання даних і сторінку для їх перегляду [20]. Порівняння мов програмування наведено в табл. 2.5.

Таблиця 2.5 – Порівняння мов програмування для реалізації серверної частини

Характеристика	**Python**	**JavaScript / Node.js**	**PHP**	**Java**
Приклад вебфреймворку	Flask, FastAPI, Django	Express.js, NestJS	Laravel, Symfony	Spring Boot
Запуск простого локального сервера	Можливий у межах одного файлу app.py	Потребує Node.js-проєкту та пакетів npm	Можливий через PHP-сервер або фреймворк	Потребує складнішої структури проєкту
Робота з CSV-файлами	Вбудований модуль csv, бібліотека pandas	Пакети csv-parser, fast-csv	Функції fgetcsv, fputcsv	OpenCSV, Apache Commons CSV
Обробка числових даних	NumPy, pandas, SciPy	Додаткові пакети, наприклад danfo.js, mathjs	Обмежена без додаткових бібліотек	Можлива, але потребує більше коду
Побудова графіків	matplotlib, Plotly	Chart.js, D3.js	Chart.js через вебсторінку	JavaFX або передавання у вебінтерфейс
Приймання запитів від ESP8266	Через маршрут Flask, наприклад /upload	Через маршрут Express.js	Через PHP-скрипт або фреймворк	Через контролер Spring Boot
Доцільність для запропонованої системи	Приймання даних, CSV, графік, подальша обробка сигналу	Зручний для вебчастини, але менш зручний для аналізу	Підходить для простого сайту, але слабший для обробки сигналів	Надлишковий для локального прототипу

Порівняння, наведене в табл. 2.5, показує, що екосистема Python містить необхідні засоби для приймання HTTP-запитів, роботи з CSV-файлами, обробки числових даних, побудови графіків і подальшого аналізу біомедичних сигналів. Для прототипу це важливо, оскільки серверна частина повинна не лише отримати значення від ESP8266 V3.0 CH340, а й перевірити їх, зберегти у структурованому вигляді, підготувати до візуалізації та подальшої обробки. JavaScript / Node.js зручне середовище для веблогіки, однак для задач аналізу числових даних потребує додаткових бібліотек. PHP орієнтований на формування вебсторінок і просту серверну обробку, а Java використовується для масштабніших серверних рішень зі

складнішою структурою проєкту. Для локального функціонального прототипу на базі IoT використання Python і Flask зберігає просту структуру серверної частини та водночас залишає можливість подальшого розширення алгоритмів обробки даних.

Серверна частина повинна приймати дані від пристрою збору, перевіряти наявність необхідних значень, записувати їх у файл, передавати у вебінтерфейс і забезпечувати перегляд результатів вимірювання. Орієнтовну структуру серверної частини наведено в табл. 2.6.

Таблиця 2.6 – Орієнтовна структура серверної частини та вебінтерфейсу

Елемент проєкту	**Призначення**
app.py	Основний файл Flask-застосунку, у якому описуються маршрути приймання, збереження та передавання даних
data/	Папка для збереження файлів із результатами вимірювань
sensor_data.csv	Файл, у який записуються дані, отримані від пристрою збору
templates/	Папка з HTML-шаблонами вебсторінок
index.html	Головна сторінка вебінтерфейсу
static/	Папка для файлів оформлення та клієнтських скриптів
style.css	Файл стилів для оформлення сторінки
script.js	Файл для оновлення даних на сторінці та побудови графіка

У такій структурі файл *app.py* – основний елемент серверної частини через який запускається Flask-сервер, задаються маршрути, приймаються запити від пристрою збору даних і передаються значення у вебінтерфейс. Папка *data* використовується для збереження результатів вимірювань, оскільки окреме збереження даних зручне під час тестування, а файл можна переглянути, очистити або використати для подальшого аналізу. Папки *templates* і *static* відповідають за візуальну частину – HTML-сторінку, стилі та скрипти оновлення даних.

Робота Flask-сервера будується на маршрутах, де один маршрут відповідає за відкриття головної сторінки, інший – за приймання даних від ESP8266 V3.0 CH340, ще один – за передавання збережених значень у вебінтерфейс. Під час тестування

передбачається маршрут для очищення файлу з попередніми вимірюваннями, щоб нові дані не змішувалися зі старими. Приклади маршрутів наведено в табл. 2.7.

Таблиця 2.7 – Приклад основних маршрутів Flask-сервера

Маршрут	Призначення	Результат виконання
/main	Відкриття головної сторінки вебінтерфейсу	Відображається сторінка з поточними значеннями та службовою інформацією
/upload	Приймання даних від ESP8266 V3.0 CH340	Отримані значення перевіряються та записуються у файл
/data	Передавання збережених значень у вебінтерфейс	Сторінка отримує дані для оновлення графіка
/clear	Очищення файлу з даними під час тестування	Попередні вимірювання видаляються перед новим сеансом

Маршрут /main використовується для відкриття головної сторінки вебінтерфейсу, на якій відображаються графік отриманого сигналу, поточні значення IR та RED каналів, кількість записів у файлі та службова інформація про стан приймання даних. Маршрут */upload* основний для взаємодії серверної частини з пристроєм збору даних. Після надходження запиту сервер перевіряє, чи отримано необхідні значення, якщо запит порожній або має неправильну структуру, такі дані не повинні записуватися у файл. Це потрібно для того, щоб у датасеті не з’являлися порожні рядки або випадкові службові повідомлення, які надалі можуть заважати побудові графіка та обробці сигналу. Маршрут */data* використовується вебсторінкою – через нього інтерфейс отримує збережені значення та оновлює графік, що дозволяє переглядати зміну сигналу без ручного відкривання файлу з даними.

Вебінтерфейс не повинен бути перевантаженим зайвими елементами, основне призначення якого показати, що система працює коректно – дані надходять, записуються та можуть бути представлені у зрозумілому вигляді. На головній сторінці розміщується графік сигналу, блок останніх прийнятих значень, кількість записів у файлі та кнопку очищення даних перед новим вимірюванням. Основні елементи вебінтерфейсу наведено в табл. 2.8.

Таблиця 2.8 – Елементи вебінтерфейсу системи

Елемент вебінтерфейсу	Призначення
Графік сигналу	Відображення зміни отриманих значень у часі
Блок поточних значень	Показ останніх прийнятих IR та RED значень
Лічильник записів	Контроль кількості рядків, збережених у файлі
Стан приймання даних	Відображення інформації про надходження нових значень
Кнопка очищення	Видалення попередніх даних перед новим тестуванням

Наявність графіка для первинної перевірки системи допоможе зрозуміти, що якщо сигнал змінюється під час контакту пальця із сенсором – апаратний вузол передає дані, а сервер їх приймає. Якщо графік не оновлюється або значення залишаються незмінними, це свідчить про проблему на одному з етапів – підключення сенсора, передавання через Wi-Fi, роботу маршруту приймання або оновлення сторінки. Логіка роботи серверної частини та вебінтерфейсу наведену на рис. 2.5, який відображає послідовність приймання даних від ESP8266 V3.0 CH340, обробку запиту Flask-сервером, перевірку коректності отриманих значень, запис у CSV-файл і подальше оновлення графіка та числових показників у вебінтерфейсі.

Рисунок 2.5 – Блок-схема алгоритму роботи серверної частини та вебінтерфейсу

Блок-схема демонструє, що серверна частина працює циклічно, оскільки після отримання даних Flask-сервер перевіряє їхню коректність і якщо значення некоректні, сервер повертається до очікування нового запиту. Однак якщо дані відповідають потрібній структурі, вони записуються у CSV-файл, після чого передаються у вебінтерфейс для оновлення графіка та числових показників. Дана логіка не змішує помилкові запити з коректними вимірюваннями і підтримує стабільну роботу системи під час тестування.

Після запуску сервер має перебувати в одній локальній мережі з пристроєм збору даних. Це потрібно для того, щоб ESP8266 V3.0 CH340 міг звернутися до IP-адреси комп’ютера або іншого пристрою, на якому запущено Flask і якщо сервер не запущено або IP-адресу вказано неправильно, дані не будуть прийняті. На стороні Flask можна переглядати службові повідомлення в консолі – факт надходження запиту, отримані значення, результат запису у файл або повідомлення про помилку, що дає змогу швидко зрозуміти, чи проблема виникає на стороні пристрою збору, у WiFi-з’єднанні чи вже на рівні серверної логіки. Серверна частина на даному етапі не виконує остаточну оцінку рівня стресу, а створює стабільний канал приймання та збереження даних.

2.5 Порядок налагодження прототипу

Для системи на базі IoT недостатньо лише правильно з’єднати компоненти та написати програму для мікроконтролера. Система повинна працювати як єдиний ланцюг, у якому сенсор формує первинні значення, ESP8266 V3.0 CH340 приймає їх і передає через Wi-Fi, а серверна частина зберігає дані та відображає їх у вебінтерфейсі, якщо хоча б один із цих етапів працює нестабільно, кінцевий результат не можна вважати придатним для подальшої обробки. Налагодження слід виконувати не відразу для всієї системи, а послідовно. Спочатку перевіряється апаратний рівень, потім програмна частина мікроконтролера, після цього мережеве передавання і наприкінці – робота Flask-сервера та вебінтерфейсу. Цей порядок дозволяє швидше визначити місце помилки, наприклад, якщо дані не з’являються

на вебсторінці, причина може бути не у Flask-сервері, а в неправильному підключенні сенсора, нестабільному живленні або відсутності WiFi-з’єднання.

Першим етапом є перевірка живлення та підключення апаратних компонентів. На цьому етапі контролюється, чи подається напруга на ESP8266 V3.0 CH340 і сенсор MAX30102, чи правильно з’єднані контакти GND, 3V3, SCL та SDA. Особливу увагу потрібно звернути на спільну землю, оскільки без неї обмін даними через I2C може бути нестабільним [6; 9; 20].

Другий етап – перевірка роботи MAX30102 через Arduino IDE. Для цього використовується простий тестовий скетч, який зчитує IR та RED значення і виводить їх у Serial Monitor, якщо при контакті пальця із сенсором значення змінюються, це означає, що сенсор працює і мікроконтролер отримує від нього дані, а якщо значення не змінюються або залишаються нульовими, слід перевірити підключення, живлення, бібліотеку для сенсору MAX30102 і правильність вибору контактів I2C [6; 7].

Після перевірки сенсора виконується перевірка WiFi-з’єднання – ESP8266 V3.0 CH340 повинен підключитися до локальної мережі та отримати можливість передавати дані на Flask-сервер. На цьому етапі важливо перевіряється правильність назви мережі, пароля, IP-адреси сервера та доступність комп’ютера, на якому запущено серверну частину і якщо плата не підключається до мережі, передавання даних буде неможливим навіть за правильної роботи сенсора. Сервер має приймати запити від ESP8266 V3.0 CH340, перевіряти отримані значення і записувати їх у файл. Під час налагодження слід виводити службові повідомлення в консоль, щоб бачити, чи надходять запити, які саме значення отримано і чи виконано запис у CSV-файл, що надає змогу швидко визначити, чи проблема виникає на стороні пристрою збору даних чи вже на рівні серверної частини [9–10; 16].

Після цього перевіряється робота вебінтерфейсу, де CSV-файл повинен оновлюватися, але графік не змінюється, то проблема пов’язана з маршрутом передавання даних у вебінтерфейс або з клієнтським скриптом, який відповідає за оновлення графіка. Якщо графік змінюється разом із новими значеннями, основний

цикл роботи системи виконується правильно – сенсор зчитує сигнал, мікроконтролер передає дані, сервер їх зберігає, а вебінтерфейс відображає результат. Основні етапи налагодження системи наведено в табл. 2.9.

Таблиця 2.9 – Етапи налагодження системи на базі IoT

Етап налагодження	Що перевіряється	Очікуваний результат
Перевірка живлення	Подача напруги на ESP8266 V3.0 CH340 і сенсор MAX30102	Плата та сенсор запускаються без перезавантажень
Перевірка I2C-з’єднання	Підключення SCL, SDA, GND і 3V3	Сенсор визначається програмою
Перевірка MAX30102	Зчитування IR та RED значень	Значення змінюються при контакті пальця із сенсором
Перевірка Wi-Fi	Підключення ESP8266 до локальної мережі	Плата отримує доступ до Flask-сервера
Перевірка Flask-сервера	Приймання HTTP-запитів	Сервер отримує дані та записує їх у файл
Перевірка CSV-файлу	Наявність нових записів	Дані зберігаються у правильній структурі
Перевірка вебінтерфейсу	Оновлення графіка і числових значень	Користувач бачить отримані дані на сторінці

Цей порядок налагодження дозволяє перевіряти систему не хаотично, а за послідовністю проходження даних. Якщо помилка виникає на певному етапі, не потрібно одразу змінювати всю програму або схему підключення достатньо перевірити попередній етап і визначити, чи дійшли дані до потрібної частини системи. Наприклад, якщо у CSV-файлі немає нових рядків, але в Serial Monitor видно, що ESP8266 зчитує значення, тоді проблема може бути у WiFi-з’єднанні або HTTP-запиті. Якщо у Serial Monitor немає зміни IR та RED значень, тоді причина в сенсорі або його підключенні. Для зручності під час тестування слід використовувати кілька контрольних ознак:

– поява службових повідомлень у Serial Monitor;
– оновлення CSV-файлу;
– зміна графіка у вебінтерфейсі.

Під час налагодження можливі типові помилки, які пов’язані як з апаратною, так і з програмною частиною. Частина з них виникає через неправильне підключення або нестабільне живлення, інша частина – через некоректні налаштування Wi-Fi, IP-адреси сервера або маршруту приймання даних. Окремо слід врахувати, що під час налагодження не варто одразу оцінювати рівень стресу,

скільки на цьому етапі основне завдання – перевірка технічної працездатності системи. Тобто потрібно переконатися, що сигнал надходить, дані передаються, файл оновлюється, а вебінтерфейс відображає отримані значення. Лише після цього слід переходити до більш детальної обробки сигналу, визначення піків, розрахунку ЧСС та аналізу показників.

Налагодження системи дозволяє визначити обмеження прототипу, наприклад, якщо користувач рухає пальцем або сенсор прилягає нестабільно, дані можуть містити різкі стрибки. Якщо WiFi-з’єднання нестабільне, частина запитів може не доходити до сервера, а ці обмеження не свідчать про неправильну концепцію системи, але їх потрібно враховувати під час подальшої експериментальної перевірки. Для завершення етапу налагодження потрібно отримати мінімальний працездатний результат – стабільне визначення сенсора MAX30102, зміна IR та RED значень при контакті пальця із сенсором, підключення ESP8266 V3.0 CH340 до Wi-Fi, надходження даних на Flask-сервер, запис у CSV-файл і відображення сигналу у вебінтерфейсі. Якщо всі ці умови виконуються, системи готова до подальшого етапу – обробки даних та експериментальної перевірки [20; 21].

Висновки до розділу 2

У другому розділі було виконано проєктування апаратно-програмної частини системи збору та передавання біомедичних даних на базі IoT. Розроблено загальну концепцію системи, у якій апаратний рівень відповідає за реєстрацію первинного пульсового сигналу, а програмний рівень забезпечує приймання, збереження, відображення та підготовку даних до подальшої обробки. Цей розподіл дозволяє не перевантажувати мікроконтролер складними обчисленнями, а основну роботу з очищення, аналізу та візуалізації виконувати на серверній частині. Як основний сенсорний модуль для збору біомедичних даних обрано сенсор MAX30102, який працює за принципом фотоплетизмографії та формує значення IR і RED каналів. Його використання доцільне для цієї системи, оскільки модуль – компактний, не потребує електродів і може бути підключений до мікроконтролерної плати через

інтерфейс I2C. Серед можливих варіантів керуючого вузла обрано ESP8266 V3.0 CH340, оскільки ця плата має вбудований Wi-Fi, підтримує I2C-з’єднання та забезпечує достатні можливості для зчитування й передавання даних на сервер.

У межах розділу описано апаратне підключення компонентів системи – підключення сенсора MAX30102 до ESP8266 V3.0 CH340 виконується через лінії SCL і SDA, із використанням живлення 3,3 В та спільної землі. Враховано стабільність живлення, можливі піки споживання під час WiFi-передавання, якість контакту сенсора зі шкірою, вплив зовнішнього освітлення та надійність з’єднувальних провідників.

Для програмування пристрою збору даних обрано середовище Arduino IDE. У роботі визначено основні програмні засоби, необхідні для взаємодії з сенсором, підключення до Wi-Fi та передавання даних на сервер – Wire.h, MAX30105.h / MAX3010x, ESP8266WiFi.h та ESP8266HTTPClient.h. Серверну частину системи спроєктовано на базі мови програмування Python та фреймворку Flask. Python обрано через зручність роботи з числовими даними, CSV-файлами, графіками та алгоритмами обробки біомедичних сигналів. Flask використовується для приймання HTTP-запитів від ESP8266 V3.0 CH340, запису отриманих значень у CSV-файл і передавання даних у вебінтерфейс. Визначено порядок налагодження прототипу, який передбачає послідовну перевірку живлення, I2C-з’єднання, роботи сенсора MAX30102, підключення ESP8266 V3.0 CH340 до Wi-Fi, приймання даних Flask-сервером, оновлення CSV-файлу та відображення результатів у вебінтерфейсі. Цей порядок дозволяє перевіряти систему не хаотично, а за логікою проходження даних від сенсора до вебінтерфейсу, що спрощує виявлення можливих помилок на етапі макетної реалізації.

Підібрані апаратні компоненти, визначене програмне середовище та розроблена логіка взаємодії між сенсором, мікроконтролером і серверною частиною сформували основу для подальшої реалізації системи на базі IoT у третьому розділі.

3 ОБРОБКА БІОМЕДИЧНИХ ДАНИХ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА СИСТЕМИ НА БАЗІ IOT

3.1 Реалізація апаратної частини пристрою збору даних

Апаратна частина функціонального прототипу на базі IoT призначена для реєстрації первинного пульсового сигналу та передавання отриманих значень до програмної частини системи. На цьому етапі основна увага приділяється практичному складанню вузла, перевірці фізичного підключення та підтвердженню того, що сенсорний модуль реагує на контакт із пальцем і формує числові значення для подальшої обробки. До складу апаратної частини входять плата ESP8266 V3.0 CH340, сенсорний модуль MAX30102, макетна плата, з’єднувальні дроти та USB-кабель для живлення й налагодження. ESP8266 V3.0 CH340 виконує роль керуючого вузла, який приймає дані від сенсора та передає їх далі через Wi-Fi, а сенсор MAX30102 – як джерело первинних біомедичних даних, зокрема значень IR та RED каналів. Ці значення ще не готова оцінка рівня стресу, а початковий сигнал, який надалі зберігається, візуалізується та обробляється у програмній частині. Основні компоненти, використані для складання апаратного вузла, наведено на рис. 3.1.

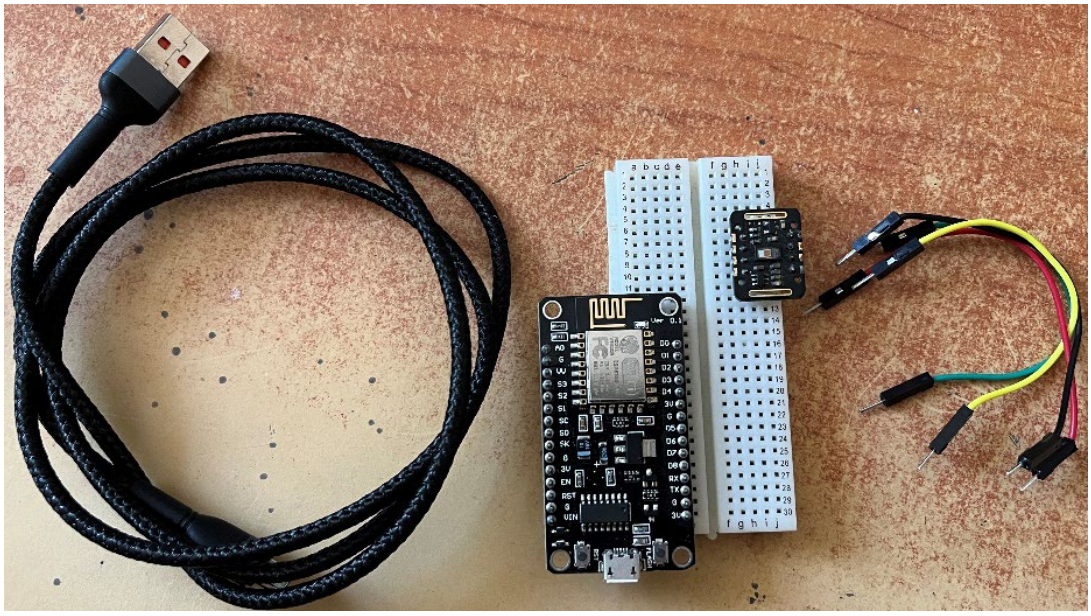


Рисунок 3.1 – Плата ESP8266 V3.0 CH340, сенсорний модуль MAX30102 та з’єднувальні елементи для складання прототипу

На першому етапі компоненти розміщуються на макетній платі. Цей спосіб складання зручний для тестового прототипу, оскільки дає змогу швидко перевірити правильність з’єднань, доступ до контактів і реакцію системи на зміну підключення без остаточного монтажу. Це особливо важливо для етапу налагодження, коли потрібно окремо перевіряти живлення, лінії обміну даними та стабільність роботи сенсора. Підключення сенсорного модуля MAX30102 до ESP8266 V3.0 CH340 реалізовано через інтерфейс I2C. Для роботи вузла використовуються лінії передавання даних, живлення 3,3 В і спільна земля. Практична реалізація підключення через макетну плату наведена на рис. 3.2.

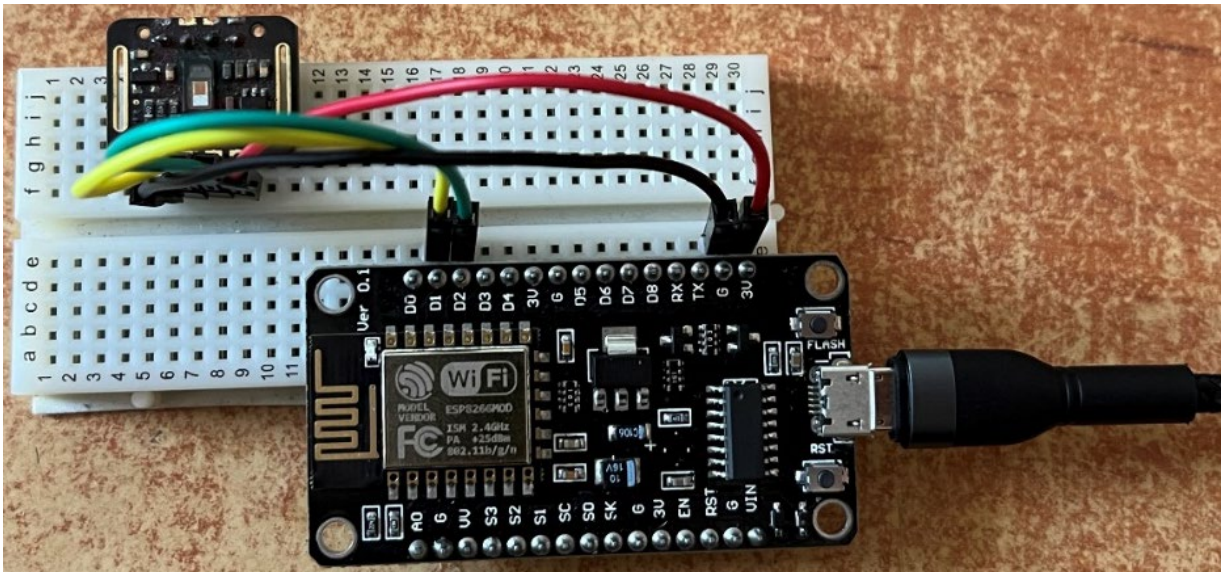


Рисунок 3.2 – Підключення сенсорного модуля MAX30102
до плати ESP8266 V3.0 CH340 через макетну плату

Після складання апаратного вузла перевіряється реакція сенсора на контакт із пальцем. Сенсор MAX30102 працює за оптичним принципом, тому під час вимірювання важливе значення має стабільне положення пальця на чутливій ділянці модуля. Сенсор фіксує зміну відбиття світла від тканин, що пов’язано зі зміною кровонаповнення під час серцевих скорочень. У межах прототипу ці зміни подаються у вигляді числових значень IR та RED каналів, які надалі передаються до програмної частини системи на базі IoT, якщо контакт недостатній або палець зміщується, отримані значення можуть змінюватися різко, що ускладнює побудову графіка та подальший аналіз сигналу. Через це під час первинної перевірки важливо

не лише переконатися, що сенсор визначається платою, а й простежити зміну показників під час реального контакту з пальцем. Цей етап дозволяє оцінити, чи апаратний вузол формує придатний первинний сигнал, а не лише передає випадкові або нестабільні значення. Процес тестового зчитування пульсового сигналу наведено на рис. 3.3.

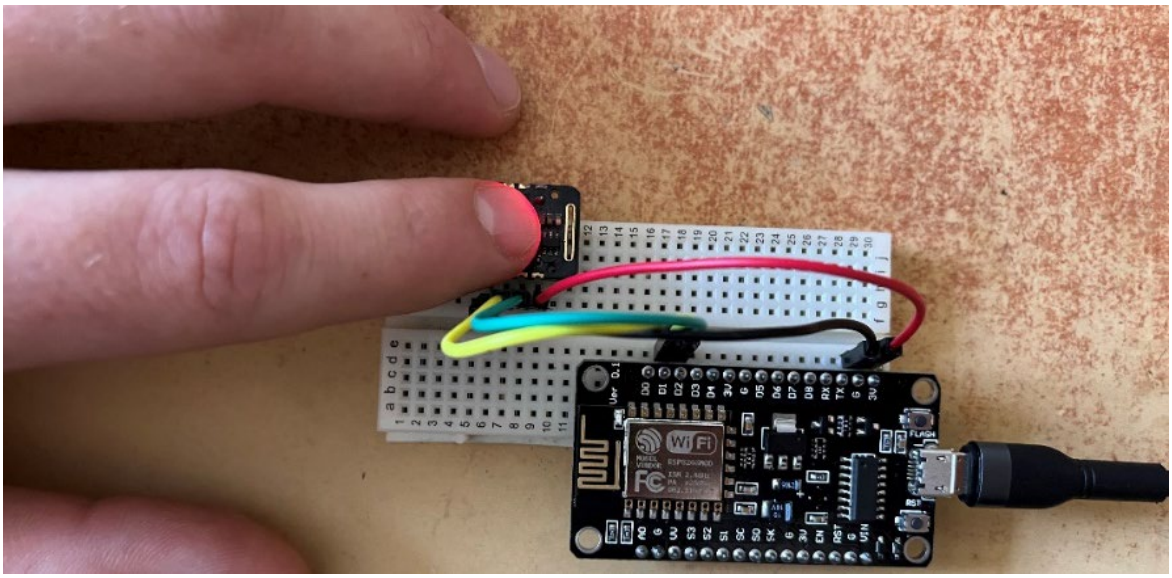


Рисунок 3.3 – Зчитування пульсового сигналу сенсором MAX30102 у складі прототипу

Живлення прототипу під час тестування здійснюється через USB-підключення, що дозволяє одночасно живити плату, завантажувати програму та контролювати службові повідомлення під час налагодження. Для фактичної перевірки апаратної частини важливо, щоб плата запускалася без перезавантажень, сенсор визначався програмою, а значення IR та RED каналів змінювалися під час контакту пальця із сенсорним модулем. Наступним етапом є перевірка взаємодії апаратного вузла з програмною частиною, в якому після зчитування значень ESP8266 V3.0 CH340 передає їх до серверної частини, де дані відображаються у вебінтерфейсі у вигляді числових показників і графіка сигналу. Ця перевірка показує, що працює не лише окремий сенсор, а весь шлях проходження даних – від фізичного вимірювання до візуального подання результату на сторінці. Приклад тестування передавання біомедичних даних до вебінтерфейсу наведено на рис. 3.4.

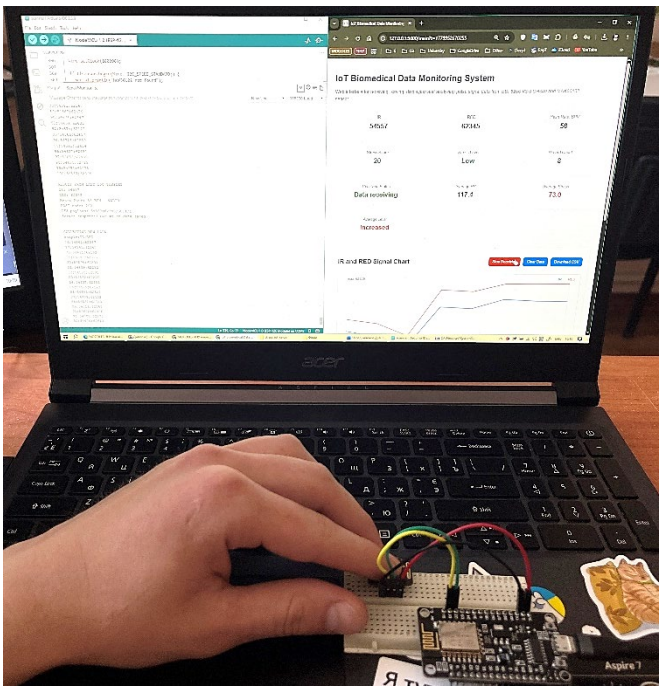


Рисунок 3.4 – Тестування передавання біомедичних даних

У результаті апаратна частина прототипу забезпечує зчитування первинного пульсового сигналу за допомогою сенсорного модуля MAX30102, приймання отриманих значень платою ESP8266 V3.0 CH340 та подальше передавання даних до серверної частини системи. Зібраний вузол виконує функцію початкової ланки всієї системи на базі IoT, оскільки саме на цьому етапі фізіологічний сигнал перетворюється на числові значення, придатні для збереження, візуалізації та подальшої обробки. Цей підхід дозволяє відокремити процес фізичного вимірювання від програмної частини, де вже виконуються накопичення, перегляд і підготовка даних до аналізу.

Практична перевірка показує, що апаратний вузол реагує на контакт пальця із сенсором, формує IR та RED значення і може передавати їх у програмне середовище. При цьому отримані значення ще не остаточний результат оцінювання рівня стресу, а виступають вхідними даними для наступних етапів роботи – програмної обробки, збереження у файлі, побудови графіка та експериментальної перевірки. Важливим результатом цього етапу є підтвердження того, що всі основні апаратні компоненти взаємодіють між собою: сенсорний модуль реєструє сигнал, мікроконтролерна плата приймає дані, а підключення через USB забезпечує стабільну роботу під час тестування.

Реалізована апаратна частина забезпечує зв’язок між сенсорним модулем і програмною частиною, дозволяє перевірити стабільність зчитування пульсового сигналу та підготувати біомедичні дані для наступного етапу – реалізації програмного забезпечення мікроконтролера. Надалі саме від коректності роботи цього вузла залежить якість передавання значень, формування датасету та можливість подальшого аналізу отриманого сигналу в серверній частині системи.

3.2 Реалізація програмного забезпечення ESP8266 V3.0 CH340

Програмне забезпечення мікроконтролера забезпечує роботу апаратного вузла як частини системи на базі IoT. Плата ESP8266 V3.0 CH340 не лише отримує значення від сенсорного модуля MAX30102, а й виконує підключення до WiFi-мережі, формує дані у визначеному форматі та передає їх до серверної частини. На цьому етапі апаратний сигнал перетворюється на послідовність цифрових значень, які вже можуть бути прийняті Flask-сервером, збережені у CSV-файлі та відображені у вебінтерфейсі. Для роботи з платою ESP8266 V3.0 CH340 використано середовище Arduino IDE. Перед завантаженням прошивки необхідно забезпечити коректне визначення плати комп’ютером, оскільки модуль CH340 використовується для USB-підключення та передавання програмного коду на мікроконтролер. Після встановлення драйвера CH340 (рис. 3.5) плата розпізнається операційною системою як послідовний порт, що дозволяє обрати відповідний COM-порт у середовищі Arduino IDE та виконати завантаження програми.

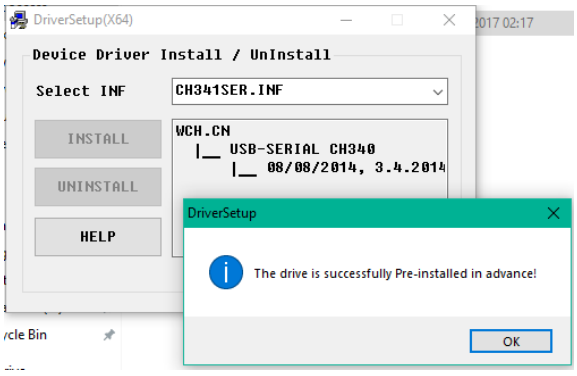


Рисунок 3.5 – Встановлення драйвера CH340 для підключення плати ESP8266 V3.0 CH340 до комп’ютера

Після налаштування драйвера в Arduino IDE встановлено пакет підтримки плат ESP8266 (рис. 3.6), щоб середовище програмування мало доступ до налаштувань компіляції, завантаження прошивки, параметрів flash-пам’яті та бібліотек, необхідних для роботи з WiFi-модулем. У менеджері плат Arduino IDE обрано пакет ESP8266 by ESP8266 Community, що забезпечує підтримку плат на базі ESP8266.

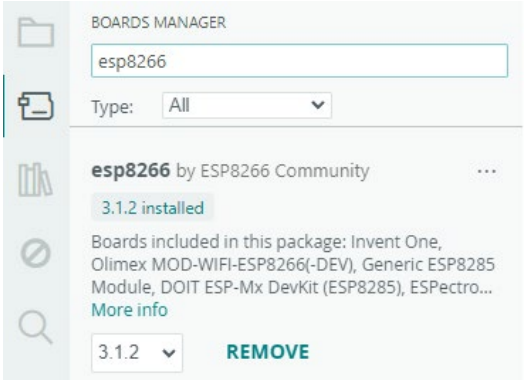


Рисунок 3.6 – Встановлений пакет плат ESP8266 у середовищі Arduino IDE

Після підготовки середовища створено прошивку для керування роботою пристрою збору біомедичних даних. У початковій частині програми підключаються бібліотеки для роботи з інтерфейсом I2C, сенсором MAX30102, WiFi-з’єднанням і HTTP-запитами (рис. 3.7). Окремо задаються контакти SDA та SCL, параметри локальної WiFi-мережі, адреса Flask-сервера, розмір буфера даних і частота зчитування сигналу. Саме ці налаштування визначають, з яким сенсором працює мікроконтролер, через які контакти здійснюється обмін даними та куди будуть передаватися результати вимірювання.

```
#include <Wire.h>
#include "MAX30105.h"

#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <math.h>

#define SDA_PIN D2
#define SCL_PIN D1

MAX30105 sensor;

const char* ssid = "iPhone 13";
const char* password = "*******";
const char* serverUrl = "http://172.20.10.7:5000/upload";

const int DATA_BUFFER_SIZE = 100;
const int UPDATE_SAMPLES = 25;
const int SAMPLE_RATE = 50;
```

Рисунок 3.7 – Підключення бібліотек і задання основних параметрів

У програмі передбачено окрему функцію для підключення ESP8266 V3.0 CH340 до WiFi-мережі. Після запуску плата переходить у режим клієнта, підключається до заданої мережі та виводить службові повідомлення в Serial Monitor. Це дозволяє під час налагодження перевірити наявність з’єднання та IP-адресу плати, якщо WiFi-з’єднання відсутнє, передавання даних до Flask-сервера неможливе, тому ця перевірка є одним із перших етапів роботи прошивки.

Ініціалізація сенсора MAX30102 виконується у функції *setup()*. Спочатку запускається послідовний порт, після чого налаштовується інтерфейс I2C через контакти D2 та D1. Далі програма перевіряє, чи відповідає сенсор на запит. Якщо сенсор MAX30102 не визначається, виконання зупиняється, а в Serial Monitor виводиться повідомлення про помилку. Після успішного визначення сенсора задаються основні параметри його роботи: яскравість світлодіодів, усереднення вибірок, режим RED та IR каналів, частота дискретизації, ширина імпульсу й діапазон АЦП. Ці налаштування впливають на стабільність отриманого сигналу під час реєстрації пульсової хвилі. Фрагмент коду ініціалізації наведено на рис. 3.8.

```
void setup() {
  Serial.begin(115200);
  delay(1000);

  Serial.println();
  Serial.println("ESP8266 MAX30102 FLASK CSV SYSTEM STARTED");

  Wire.begin(SDA_PIN, SCL_PIN);
  Wire.setClock(100000);

  if (!sensor.begin(Wire, I2C_SPEED_STANDARD)) {
    Serial.println("MAX30102 not found");
    while (1) {
      delay(1000);
    }
  }

  Serial.println("MAX30102 found");

  byte ledBrightness = 0x3F;
  byte sampleAverage = 4;
  byte ledMode = 2;
  int sampleRate = 50;
  int pulseWidth = 411;
  int adcRange = 16384;

  sensor.setup(
    ledBrightness,
    sampleAverage,
    ledMode,
    sampleRate,
    pulseWidth,
    adcRange
  );

  sensor.setPulseAmplitudeRed(0x3F);
  sensor.setPulseAmplitudeIR(0x3F);
  sensor.setPulseAmplitudeGreen(0);
```

Рисунок 3.8 – Ініціалізація сенсора MAX30102 та WiFi-з’єднання

Основне зчитування біомедичних даних виконується через окрему функцію (рис. 3.9), яка очікує наявність нового зразка від сенсора, після чого зберігає значення RED та IR каналів у відповідні буфери. Ця структура зручна для

подальшої обробки, оскільки програма працює не з одиничним випадковим значенням, а з послідовністю вимірювань. У межах прошивки використовується буфер на 100 зразків, що дозволяє аналізувати короткий фрагмент сигналу та визначати зміну показників у часі.

```
void readSample(int index) {
  while (sensor.available() == false) {
    sensor.check();
  }

  redBuffer[index] = sensor.getRed();
  irBuffer[index] = sensor.getIR();

  sensor.nextSample();
}
```

Рисунок 3.9 – Фрагмент коду зчитування IR та RED значень із сенсора MAX30102

Значення IR та RED використовуються як первинні дані сенсора. IR-канал у програмі застосовується для визначення наявності пальця на сенсорі та орієнтовного розрахунку частоти серцевих скорочень. Якщо значення IR нижче встановленого порогу, програма вважає, що палець не виявлено, і передає на сервер відповідну ознаку. Це допомагає відокремити реальні вимірювання від ситуацій, коли сенсор працює без контакту зі шкірою. Для розрахунку частоти серцевих скорочень у прошивці використовується аналіз останніх 100 зразків IR-сигналу. Спочатку обчислюється середнє значення сигналу та його відхилення, після чого формується порогове значення для пошуку піків. Далі програма визначає локальні максимуми, перевіряє відстань між ними та обчислює орієнтовне значення ЧСС у BPM. Результат додатково перевіряється на допустимий діапазон. Якщо обчислене значення виходить за межі 40–180 BPM або кількість коректних піків недостатня, результат позначається як недійсний.

Перед відправленням даних на Flask-сервер програма перевіряє стан WiFi-з'єднання, якщо підключення втрачено, плата повторно виконує підключення до мережі. Після цього створюється HTTP-запит, а дані формуються у вигляді CSV-рядка – один рядок містить значення IR, RED, розраховану ЧСС, ознаку

коректності ЧСС та ознаку наявності пальця на сенсорі. Фрагмент коду формування даних та їх передавання наведено на рис. 3.10.

```
void sendCsvToServer(long ir, long red, int hr, bool hrValid, bool finger) {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Wi-Fi disconnected. Reconnecting...");
    connectToWiFi();
    return;
  }

  WiFiClient client;
  HTTPClient http;

  if (!http.begin(client, serverUrl)) {
    Serial.println("HTTP connection failed");
    return;
  }

  http.addHeader("Content-Type", "text/csv");

  String csvPayload = "";
  csvPayload += String(ir);
  csvPayload += ";";
  csvPayload += String(red);
  csvPayload += ";";
  csvPayload += String(hr);
  csvPayload += ";";
  csvPayload += (hrValid ? "1" : "0");
  csvPayload += ";";
  csvPayload += (finger ? "1" : "0");

  int httpCode = http.POST(csvPayload);

  Serial.print("POST code: ");
  Serial.println(httpCode);

  Serial.print("CSV payload: ");
  Serial.println(csvPayload);
```

Рисунок 3.10 – Формування CSV-рядка та передавання даних на Flask-сервер

Під час виконання HTTP-запиту ESP8266 V3.0 CH340 надсилає сформований CSV-рядок на маршрут */upload*, який обробляється Flask-сервером. Після відправлення у Serial Monitor виводиться код відповіді сервера та сам рядок даних, що дозволяє під час тестування перевірити, чи дійшов запит до серверної частини, які саме значення були передані та чи виникла помилка під час HTTP-з’єднання. Логіка роботи основного циклу програми побудована на постійному оновленні буфера даних. Спочатку виконується збирання початкових 100 зразків, після чого результат обробляється та передається на сервер. Далі програма зсуває буфер, додає нові зразки й повторює передавання. Завдяки цьому система працює безперервно, а сервер отримує не одиничне вимірювання, а послідовність значень, придатну для побудови графіка сигналу у вебінтерфейсі.

Після завершення написання програми виконано її компіляцію та завантаження на плату ESP8266 V3.0 CH340. У вікні виводу Arduino IDE відображається процес запису прошивки у flash-пам’ять мікроконтролера. Після завершення запису середовище повідомляє про перевірку даних і скидання плати, що підтверджує успішне завантаження програмного коду на пристрій (рис. 3.11).

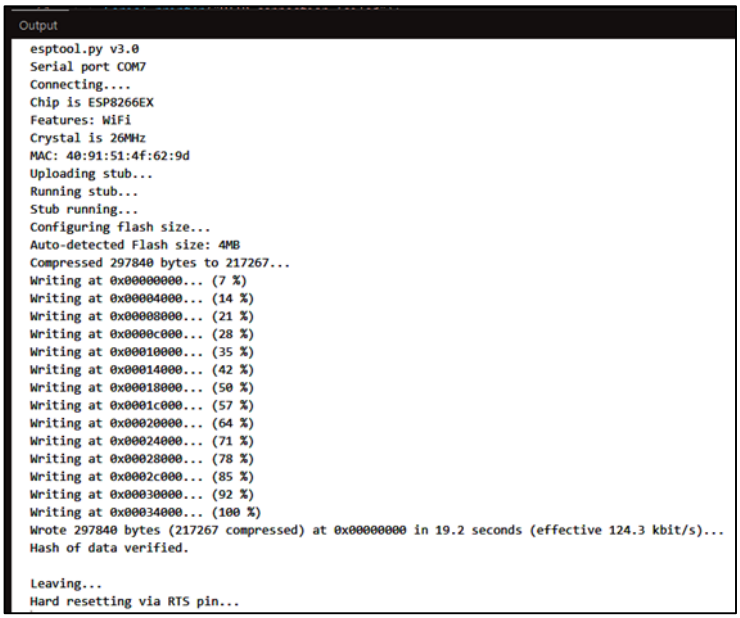


Рисунок 3.11 – Успішне завантаження прошивки

Після завантаження прошивки плата переходить до виконання програми, а через Serial Monitor можна контролювати основні етапи роботи:

– запуск системи;
– підключення до Wi-Fi;
– визначення сенсора MAX30102;
– зчитування IR та RED значень;
– розрахунок ЧСС;
– результат HTTP-запиту до Flask-сервера.

Наявність таких службових повідомлень спрощує перевірку, оскільки дозволяє окремо оцінити роботу кожної частини – сенсора, мікроконтролера, мережевого з’єднання та передавання даних на сервер. Реалізоване програмне забезпечення ESP8266 V3.0 CH340 забезпечує повний цикл роботи пристрою збору біомедичних даних – ініціалізацію сенсора MAX30102, зчитування IR та RED каналів, перевірку наявності пальця, орієнтовний розрахунок ЧСС, формування CSV-рядка та передавання даних на Flask-сервер через Wi-Fi. Ця прошивка дає змогу апаратній частині працювати не як окремий сенсорний модуль, а як вузол системи на базі IoT, який формує потік даних для подальшого збереження, візуалізації та аналізу в серверній частині. Повний лістинг коду прошивки наведено в *додатку Б.1*.

3.3 Реалізація серверної частини та вебінтерфейсу

Серверна частина системи виконує роль проміжної ланки між пристроєм збору біомедичних даних і вебінтерфейсом. Після зчитування значень із сенсорного модуля MAX30102 плата ESP8266 V3.0 CH340 передає сформований CSV-рядок через Wi-Fi на Flask-сервер. Сервер приймає ці дані, перевіряє їхню структуру, відкидає некоректні вимірювання, розраховує орієнтовний показник рівня стресу, зберігає результат у CSV-файл і передає актуальні значення на сторінку вебінтерфейсу. У початковій частині коду створюється Flask-застосунок, підключається підтримка CORS, визначається папка для збереження результатів, назва CSV-файлу, роздільник даних і структура майбутніх записів. CSV-файл містить часову мітку, значення IR та RED каналів, ЧСС, ознаки коректності вимірювання, умовний бал стресу, текстовий рівень стресу та середні значення за групою записів. Фрагмент коду створення Flask-застосунку та налаштування CSV-файлу наведено на рис. 3.12.

```
app = Flask(__name__)
CORS(app)

DATA_DIR = "data"
CSV_FILE = os.path.join(DATA_DIR, "sensor_data.csv")
CSV_DELIMITER = ";"

receiving_enabled = True

CSV_HEADER = [
    "timestamp",
    "ir",
    "red",
    "heart_rate",
    "valid_hr",
    "finger",
    "stress_score",
    "stress_level",
    "average_heart_rate_5",
    "average_stress_score_5",
    "average_stress_level_5"
]
```

Рисунок 3.12 – Створення Flask-застосунку та налаштування CSV-файлу

Після підготовки основного файлу сервер запускається з командного рядка у віртуальному середовищі проєкту. Команда *python app.py* активує Flask-сервер, після чого в консолі відображаються службові повідомлення про режим роботи, локальну адресу 127.0.0.1:5000 і мережеву адресу комп’ютера. Саме мережева адреса використовується для взаємодії з ESP8266 V3.0 CH340, оскільки

мікроконтролер має передавати дані на комп’ютер, де запущена серверна частина. Запуск Flask-сервера наведено на рис. 3.13.

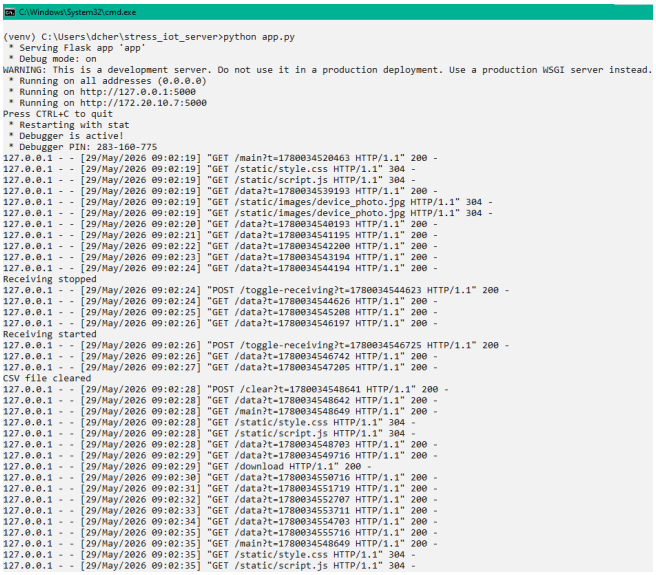


Рисунок 3.13 – Результат запуску Flask-серверу

Після запуску сервера вебінтерфейс відкривається через маршрут */main*. У початковому стані сторінка ще не містить прийнятих вимірювань, тому в блоках IR, RED, Heart Rate, Stress Score і Stress Level відображаються порожні значення. Лічильник записів дорівнює нулю, а стан приймання перебуває в режимі очікування. Цей вигляд підтверджує, що сторінка вебінтерфейсу завантажується коректно, однак дані від пристрою збору ще не надходили. Початковий стан вебінтерфейсу наведено на рис. 3.14.

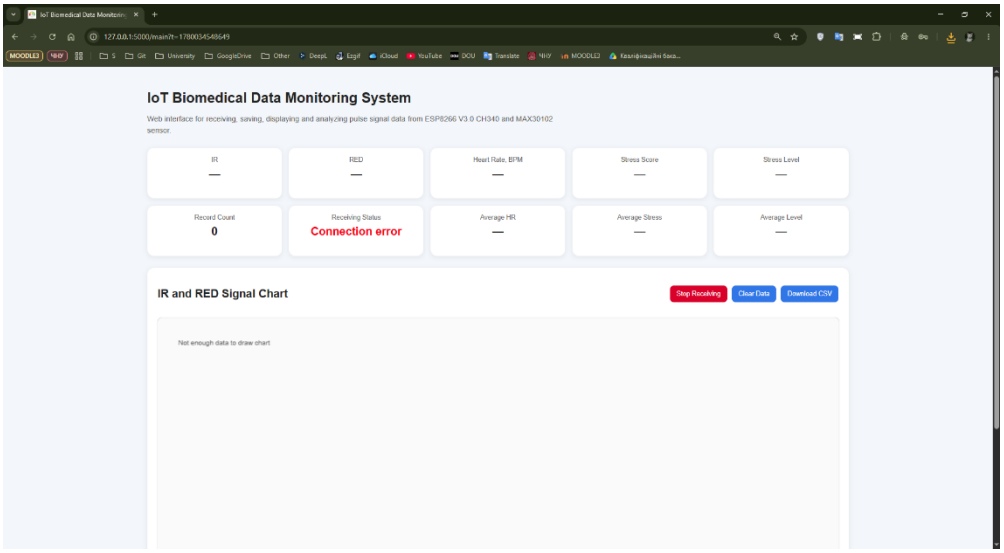


Рисунок 3.14 – Вигляд готового вебінтерфейсу

Для приймання даних від ESP8266 V3.0 CH340 у серверній частині реалізовано маршрут */upload*, який працює методом *POST*. На цей маршрут мікроконтролер надсилає CSV-рядок із п’яти основних значень:

– IR;
– RED;
– ЧСС;
– ознака коректності ЧСС;
– ознака наявності пальця на сенсорі.

Сервер спочатку перевіряє, чи не зупинене приймання даних, після чого отримує тіло запиту, очищує його від зайвих символів і передає на розбір як CSV-рядок. Якщо дані відсутні або мають неправильну структуру, сервер повертає повідомлення про помилку й не додає такий запис до файлу. Фрагмент коду приймання CSV-даних від ESP8266 V3.0 CH340 наведено на рис. 3.15.

```
@app.route( rule: "/upload", methods=["POST"])  Dmytro Cherednyche
def upload_data():
    global receiving_enabled

    if not receiving_enabled:
        print("Receiving is stopped. Incoming data skipped.")
        return "PAUSED: receiving stopped", 200

    raw_data = request.get_data(as_text=True).strip()

    if not raw_data:
        return "ERROR: CSV data not received", 400

    try:
        csv_row = parse_incoming_csv(raw_data)
    except Exception:
        return "ERROR: CSV row cannot be parsed", 400

    if len(csv_row) < 5:
        return "ERROR: CSV row has wrong structure", 400
```

Рисунок 3.15 – Приймання CSV-даних від ESP8266 V3.0 CH340

Для фільтрація некоректних вимірювань використовується функція, яка перевіряє три умови:

1) чи виявлено палець на сенсорі;
2) чи є значення ЧСС коректним;
3) чи перебуває воно в допустимому діапазоні.

Якщо хоча б одна з цих умов не виконується, вимірювання не використовується для подальшого збереження. Ця перевірка потрібна через особливості оптичного сенсора – нестабільний контакт пальця, зміщення руки або відсутність реального пульсового сигналу можуть створювати випадкові значення, які не слід записувати як коректні дані. Фрагмент коду фільтрації некоректних біомедичних даних наведено на рис. 3.16.

```
def is_valid_measurement(heart_rate :{__lt__, __gt__} , valid_hr, finger):
    if not finger:
        return False

    if not valid_hr:
        return False

    if heart_rate < 40 or heart_rate > 180:
        return False

    return True
```

Рисунок 3.16 – Фільтрація некоректних біомедичних даних

Після проходження первинної перевірки сервер перетворює прийняті значення у потрібні типи даних. Значення IR, RED і ЧСС обробляються як числові параметри, а ознаки *valid_hr* і *finger* – як логічні. Далі виконується розрахунок умовного показника стресу та визначається текстовий рівень:

– Low;
– Normal;
– Moderate;
– Increased;
– High.

Розраховані значення *stress_score* та *stress_level* використовуються як умовна програмна інтерпретація отриманих біомедичних даних. Вони не розглядаються як медичний висновок або діагностичний показник, а потрібні для перевірки того, як серверна частина обробляє прийняті значення, формує результат і передає його у вебінтерфейс. Після цього формується запис із часовою міткою, біомедичними

показниками й розрахованими параметрами, який додається до CSV-файлу. Фрагмент коду обробки та збереження біомедичних даних наведено на рис. 3.17.

```
def upload_data():
        heart_rate = int(csv_row[2])
        valid_hr = parse_bool(csv_row[3])
        finger = parse_bool(csv_row[4])

        if not is_valid_measurement(heart_rate, valid_hr, finger):
            print("Received CSV:", raw_data)
            print("Skipped invalid measurement")
            return "SKIPPED: invalid measurement", 200

        stress_score, stress_level = calculate_stress_level(
            heart_rate,
            valid_hr,
            finger
        )

        record = {
            "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            "ir": ir,
            "red": red,
            "heart_rate": heart_rate,
            "valid_hr": valid_hr,
            "finger": finger,
            "stress_score": stress_score,
            "stress_level": stress_level,
            "average_heart_rate_5": "",
            "average_stress_score_5": "",
            "average_stress_level_5": ""
        }

        existing_records = read_csv_records()
        record = calculate_averages_for_each_five_records(existing_records, record)

    except ValueError:
        return "ERROR: CSV contains incorrect values", 400

    append_to_csv(record)

    print("Received CSV:", raw_data)
    print("Saved valid record:", record)

    return "OK: valid data saved", 200
```

Рисунок 3.17 – Обробка та збереження біомедичних даних у CSV-файл

Для відображення актуальних даних у вебінтерфейсі використовується маршрут */data*, який зчитує записи з CSV-файлу, визначає загальну кількість збережених вимірювань, останній прийнятий запис, а також останні розраховані середні значення. Це дає змогу не відкривати CSV-файл вручну після кожного нового вимірювання. Вебінтерфейс автоматично звертається до серверної частини, отримує оновлені значення та відображає їх у зрозумілому для користувача вигляді, що важливо під час тестування прототипу, оскільки дозволяє одразу бачити, чи надходять дані від ESP8266 V3.0 CH340, чи збільшується кількість записів і чи змінюється сигнал на графіку. Фрагмент коду передавання збережених даних у вебінтерфейс наведено на рис. 3.18.

```python
@app.route( rule: "/data", methods=["GET"])  Dmytro Ch
def get_data():
    global receiving_enabled

    records = read_csv_records()
    latest_average = get_latest_average(records)

    response = jsonify({
        "receiving_enabled": receiving_enabled,
        "count": len(records),
        "latest": records[-1] if records else None,
        "latest_average": latest_average,
        "records": records[-100:]
    })

    return no_cache_response(response)
```

Рисунок 3.18 – Передавання збережених даних у вебінтерфейс

У серверній частині передбачено й допоміжні маршрути для керування процесом роботи:

1. Маршрут */toggle-receiving* дозволяє тимчасово зупинити або відновити приймання нових даних.

2. Маршрут */clear* очищує CSV-файл перед новим вимірюванням, залишаючи тільки заголовки стовпців.

3. Маршрут */download* забезпечує завантаження файлу *sensor_data.csv* для подальшого перегляду або аналізу.

Повний лістинг програмного коду серверної частини наведено в *додатку Б.2*.

Завдяки цьому серверна частина не лише приймає дані від мікроконтролера, а й забезпечує базове керування накопиченням результатів. У результаті реалізована серверна частина приймає CSV-дані від ESP8266 V3.0 CH340, перевіряє структуру отриманого рядка, фільтрує некоректні вимірювання, формує запис із розрахованими параметрами, зберігає його у CSV-файлі та передає актуальні значення у вебінтерфейс. Блок-схему алгоритму коду обробки біомедичних даних наведено в *додатку А.2*. Створений вебінтерфейс забезпечує відображення поточних показників, стану приймання, кількості записів і даних для побудови графіка.

3.4 Перевірка роботи системи та аналіз отриманих даних

Експериментальна перевірка проводилася для підтвердження працездатності розробленої системи на базі IoT у повному циклі роботи: зчитування сигналу сенсором MAX30102, передавання даних платою ESP8266 V3.0 CH340, приймання CSV-рядка Flask-сервером, збереження результатів у файл і відображення показників у вебінтерфейсі.

На першому етапі контролювалася робота прошивки ESP8266 V3.0 CH340 через Serial Monitor. У вікні виводу відображаються значення IR та RED каналів, розрахована ЧСС, ознака коректності результату, сформований CSV-рядок, код відповіді сервера та повідомлення про успішне збереження даних. Це підтверджує, що мікроконтролер не лише зчитує значення із сенсора, а й передає їх до серверної частини. Приклад передавання результатів вимірювання наведено на рис. 3.19.

```
IR: 54908
RED: 63042
Heart Rate: 102 BPM | VALID
POST code: 200
CSV payload: 54908;63042;102;1;1
Server response: OK: valid data saved
------------------------------

COLLECTING NEW DATA
sample;IR;RED
76;54889;63037
77;54884;63027
78;54880;63047
79;54889;63050
80;54896;63058
81;54895;63059
82;54892;63064
83;54888;63039
84;54878;63049
85;54892;63065
86;54878;63060
87;54892;63059
88;54882;63075
89;54874;63062
90;54862;63065
91;54870;63050
92;54878;63066
93;54878;63080
94;54861;63086
95;54856;63050
96;54843;63056
97;54853;63049
98;54842;63056
99;54852;63061
100;54847;63072

RESULT FROM LAST 100 SAMPLES
IR: 54847
RED: 63072
Heart Rate: 102 BPM | VALID
POST code: 200
CSV payload: 54847;63072;102;1;1
Server response: OK: valid data saved
------------------------------
```

Рисунок 3.19 – Передавання результатів вимірювання

Наступним етапом перевірялася робота Flask-сервера. У консолі сервера відображаються вхідні POST-запити на маршрут */upload*, прийняті CSV-рядки, повідомлення про пропуск некоректних вимірювань і збереження коректних записів. Наявність таких повідомлень показує, що серверна частина приймає дані від ESP8266 V3.0 CH340, виконує фільтрацію та записує лише ті значення, які відповідають заданим умовам. Приймання та збереження CSV-даних у консолі Flask-сервера наведено на рис. 3.20.

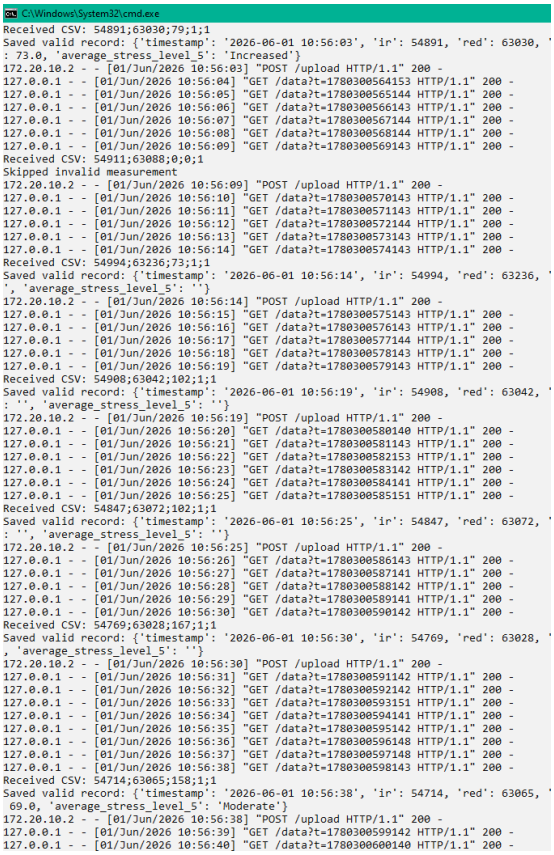


Рисунок 3.20 – Приймання та збереження CSV-даних

Після приймання даних сервер передає їх у вебінтерфейс. На сторінці відображаються поточні значення IR, RED, ЧСС, умовний показник Stress Score, рівень Stress Level, кількість збережених записів, стан приймання даних, а також середні значення за групою вимірювань. Завдяки цьому дає змогу швидко оцінити чи працює вся система як єдиний ланцюг, якщо дані надходять коректно, користувач бачить зміну поточних показників, збільшення кількості записів і оновлення середніх значень. Це дозволяє контролювати роботу системи без відкривання CSV-файлу вручну та без постійного перегляду службових

повідомлень у консолі. Відображення поточних результатів вимірювання у вебінтерфейсі наведено на рис. 3.21.

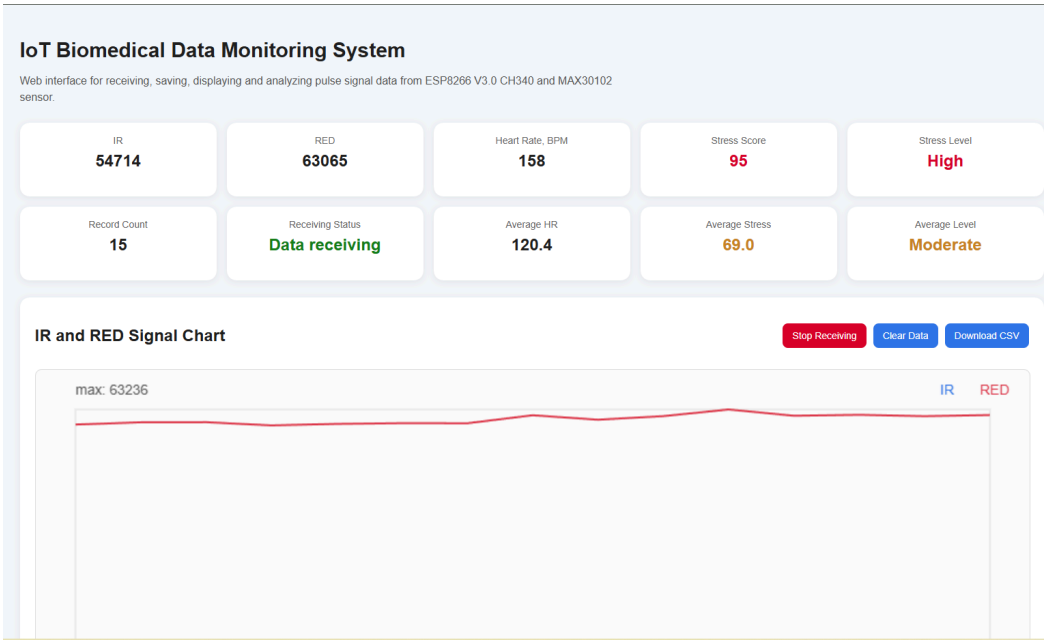


Рисунок 3.21 – Відображення поточних результатів вимірювання

Окремо у вебінтерфейсі формується графік IR та RED сигналів (рис. 3.22), який використовується для візуального контролю надходження даних і зміни значень у часі. На графіку можна побачити, що дані передаються не одиничним вимірюванням, а послідовністю записів, які оновлюються під час роботи системи.

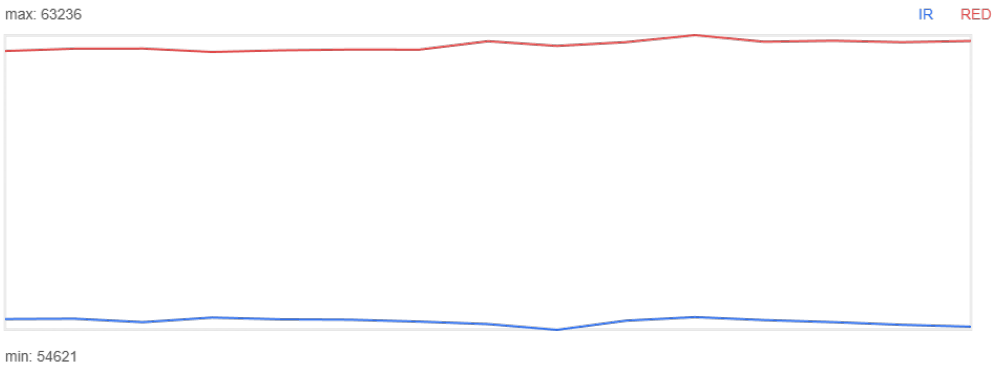


Рисунок 3.22 – Графік IR та RED сигналів у вебінтерфейсі системи

Для зручності контролю у вебінтерфейсі передбачено таблицю останніх прийнятих записів. У ній відображаються час вимірювання, значення IR та RED, ЧСС, ознаки коректності вимірювання, наявність пальця на сенсорі, Stress Score, Stress Level і середні значення. Ця таблиця дозволяє швидко переглянути, які саме

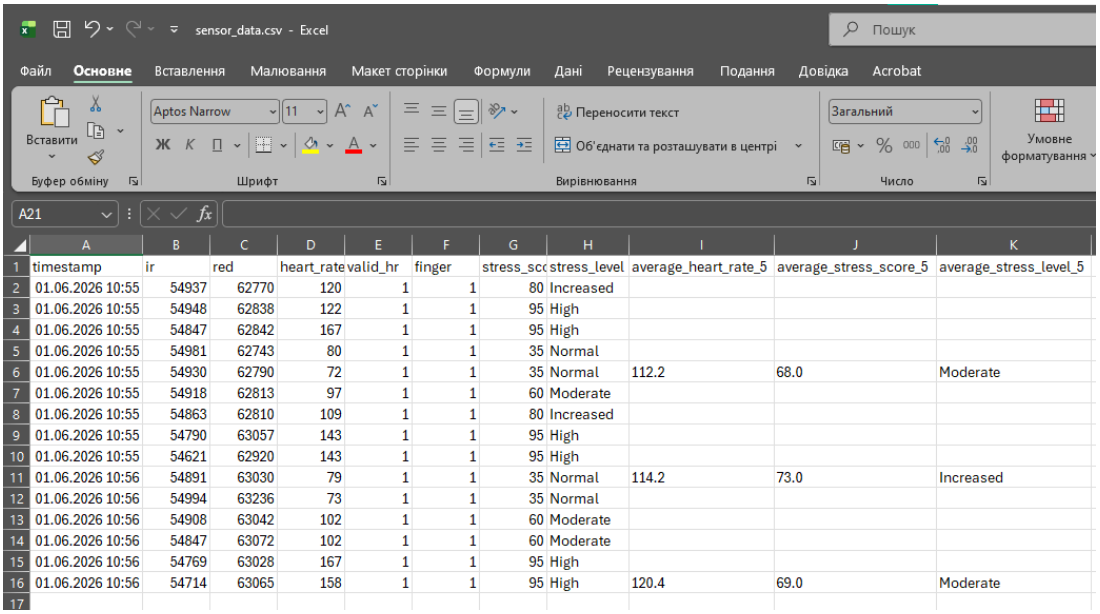
записи потрапили до системи після фільтрації. Таблицю останніх прийнятих записів наведено на рис. 3.23.

Latest Records

Time	IR	RED	Heart Rate	HR Valid	Finger	Stress Score	Stress Level	Avg HR	Avg Stress	Avg Level
2026-06-01 10:56:38	54714	63065	158	Yes	Yes	95	High	120.4	69.0	Moderate
2026-06-01 10:56:30	54769	63028	167	Yes	Yes	95	High	—	—	—
2026-06-01 10:56:25	54847	63072	102	Yes	Yes	60	Moderate	—	—	—
2026-06-01 10:56:19	54908	63042	102	Yes	Yes	60	Moderate	—	—	—
2026-06-01 10:56:14	54994	63236	73	Yes	Yes	35	Normal	—	—	—
2026-06-01 10:56:03	54891	63030	79	Yes	Yes	35	Normal	114.2	73.0	Increased
2026-06-01 10:55:58	54621	62920	143	Yes	Yes	95	High	—	—	—
2026-06-01 10:55:50	54790	63057	143	Yes	Yes	95	High	—	—	—
2026-06-01 10:55:42	54863	62810	109	Yes	Yes	80	Increased	—	—	—
2026-06-01 10:55:37	54918	62813	97	Yes	Yes	60	Moderate	—	—	—
2026-06-01 10:55:32	54930	62790	72	Yes	Yes	35	Normal	112.2	68.0	Moderate
2026-06-01 10:55:27	54981	62743	80	Yes	Yes	35	Normal	—	—	—
2026-06-01 10:55:22	54847	62842	167	Yes	Yes	95	High	—	—	—
2026-06-01 10:55:17	54948	62838	122	Yes	Yes	95	High	—	—	—
2026-06-01 10:55:11	54937	62770	120	Yes	Yes	80	Increased	—	—	—

Рисунок 3.23 – Таблиця останніх прийнятих записів

Збереження результатів перевірялося через відкриття CSV-файлу. У файлі дані подані у структурованому вигляді за стовпцями – *timestamp, ir, red, heart_rate, valid_hr, finger, stress_score, stress_level, average_heart_rate_5, average_stress_score_5, average_stress_level_5*. Це підтверджує, що результати вимірювання не лише відображаються на сторінці, а й накопичуються для подальшого перегляду та аналізу. Збереження результатів експериментальної перевірки у CSV-файлі наведено на рис. 3.24.



	A	B	C	D	E	F	G	H	I	J	K
1	timestamp	ir	red	heart_rate	valid_hr	finger	stress_sc	stress_level	average_heart_rate_5	average_stress_score_5	average_stress_level_5
2	01.06.2026 10:55	54937	62770	120	1	1	80	Increased			
3	01.06.2026 10:55	54948	62838	122	1	1	95	High			
4	01.06.2026 10:55	54847	62842	167	1	1	95	High			
5	01.06.2026 10:55	54981	62743	80	1	1	35	Normal			
6	01.06.2026 10:55	54930	62790	72	1	1	35	Normal	112.2	68.0	Moderate
7	01.06.2026 10:55	54918	62813	97	1	1	60	Moderate			
8	01.06.2026 10:55	54863	62810	109	1	1	80	Increased			
9	01.06.2026 10:55	54790	63057	143	1	1	95	High			
10	01.06.2026 10:55	54621	62920	143	1	1	95	High			
11	01.06.2026 10:56	54891	63030	79	1	1	35	Normal	114.2	73.0	Increased
12	01.06.2026 10:56	54994	63236	73	1	1	35	Normal			
13	01.06.2026 10:56	54908	63042	102	1	1	60	Moderate			
14	01.06.2026 10:56	54847	63072	102	1	1	60	Moderate			
15	01.06.2026 10:56	54769	63028	167	1	1	95	High			
16	01.06.2026 10:56	54714	63065	158	1	1	95	High	120.4	69.0	Moderate
17											

Рисунок 3.24 – Результати експериментальної перевірки у CSV-файлі

Для перевірки роботи системи виконано три окремі вимірювання. Дані учасників подано анонімно, оскільки в межах цієї роботи оцінюється працездатність технічного прототипу, а не індивідуальний медичний стан людини. Для кожного учасника враховувалися лише коректні записи, у яких сервер підтвердив наявність пальця на сенсорі та коректність значення ЧСС. Результати тестування наведено в табл. 3.1.

Таблиця 3.1 – Результати експериментальної перевірки системи на трьох учасниках

Учасник	**Тривалість реєстрації, с**	**Кількість коректних записів**	**Діапазон ЧСС, уд/хв**	**Середня ЧСС, уд/хв**	**Середній Stress Score**	**Останній середній рівень**
Учасник 1	87	15	72–167	115,6	70,0	Moderate
Учасник 2	100	16	67–90	88,9	50,6	Moderate
Учасник 3	80	15	71–176	117,3	75,3	Increased

За результатами трьох вимірювань система успішно сформувала записи для кожного учасника, зберегла їх у CSV-файли та відобразила у вебінтерфейсі. Кількість коректних записів становила 15, 16 і 15 відповідно, що підтверджує здатність прототипу накопичувати послідовність біомедичних даних. У всіх трьох випадках сервер приймав CSV-рядки, перевіряв їхню структуру, відкидав некоректні вимірювання та записував валідні значення у файл. Порівняння трьох вимірювань показує, що система однаково виконує основні функції для різних учасників:

1) приймає сигнал;
2) формує записи;
3) розраховує поточні й середні показники;
4) оновлює графік та таблицю у вебінтерфейсі.

У результаті експериментальна перевірка підтвердила працездатність розробленого прототипу. Система забезпечує зчитування біомедичних даних із сенсора MAX30102, передавання значень через ESP8266 V3.0 CH340, приймання та фільтрацію даних на сервері, збереження результатів у CSV-файлі та відображення поточних показників у вебінтерфейсі. Отримані результати

створюють основу для подальшої оцінки переваг, обмежень і можливих напрямів удосконалення системи.

3.5 Переваги, обмеження та напрями вдосконалення розробленої системи

Розроблена система на базі IoT виконує поставлену задачу – сенсор MAX30102 зчитує первинні біомедичні дані, плата ESP8266 V3.0 CH340 передає їх через Wi-Fi, а серверна частина на Python/Flask приймає, зберігає й відображає результати у вебінтерфейсі. У роботі використано доступні компоненти, які не потребують складного монтажу. ESP8266 V3.0 CH340 має вбудований Wi-Fi, тому для передавання даних не потрібно додавати окремий модуль зв’язку, сенсор MAX30102 працює за оптичним принципом і дає змогу отримувати пульсовий сигнал без електродів, що спрощує тестування, хоча й вимагає правильного розміщення пальця на сенсорі. Мікроконтролер не виконує всю обробку самостійно на ньому залишаються основні дії:

1) зчитування значень;
2) формування CSV-рядка;
3) передавання на сервер.

Перевірка, фільтрація, запис у файл і підготовка даних для вебінтерфейсу виконуються на серверній частині. Цей розподіл зручний для доопрацювання, бо змінювати алгоритми в Python простіше, ніж постійно перепрошивати мікроконтролер. Для збереження результатів використано CSV-файл, оскільки дані мають зрозумілу структуру, їх можна відкрити в табличному редакторі, переглянути окремі записи або використати для подальшої обробки. Вебінтерфейс доповнює цю частину, бо під час тестування не потрібно щоразу відкривати файл вручну. Поточні значення, графік сигналу, кількість записів і умовний рівень стресу видно одразу на сторінці.

Основне обмеження прототипу пов’язане з якістю сигналу. Сенсор MAX30102 чутливий до положення пальця, сили притискання, рухів руки та зовнішнього освітлення. Якщо контакт нестабільний, значення можуть різко

змінюватися або ставати некоректними. Серверна частина відкидає частину таких вимірювань, але повністю прибрати вплив рухів на рівні макетного зразка складно.

Показники Stress Score та Stress Level у цій роботі мають орієнтовне значення та не є медичним висновком і не можуть використовуватися для точної діагностики стресу. Їхня роль у прототипі полягає в тому, щоб показати, як отримані біомедичні дані можуть бути програмно оброблені й подані користувачу у зрозумілій формі. Для більш точного оцінювання стану людини одного пульсового сигналу недостатньо, оскільки на ЧСС впливають фізичне навантаження, втома, температура середовища, емоційний стан і індивідуальні особливості.

Головне покращення для сенсора MAX30102 – стабілізація вимірювання, шляхом створення простого корпус або фіксатору, який утримуватиме палець в одному положенні, що зменшить кількість випадкових коливань сигналу й зробить вимірювання більш рівномірним. Інший напрям розвитку – додавання інших сенсорів – датчик електродермальної активності, температури шкіри, дихання або акселерометром. Наприклад, акселерометр допоміг би фіксувати рух руки й позначати такі ділянки сигналу як менш надійні. Додаткові сенсори дали б змогу аналізувати не один показник, а кілька біомедичних сигналів одночасно, що зробило б оцінювання стану більш змістовним.

CSV-файлу достатньо для прототипу, але для тривалішого використання системи зручнішою буде база даних. Вона дала б змогу зберігати історію вимірювань, працювати з різними користувачами та порівнювати результати за окремі періоди. Розроблений прототип показує, така система може працювати, але для підвищення точності потрібні стабільніше вимірювання, більша кількість біомедичних показників і ширша експериментальна база.

Висновки до розділу 3

У третьому розділі реалізовано функціональний прототип системи на базі IoT для збору, передавання, збереження та первинного аналізу біомедичних даних. Основну увагу приділено практичному складанню апаратної частини, розробці прошивки ESP8266 V3.0 CH340, створенню серверної частини на базі Python/Flask

та перевірці роботи системи в реальних умовах тестування. У межах апаратної реалізації було зібрано пристрій збору даних на основі плати ESP8266 V3.0 CH340 та сенсорного модуля MAX30102. Сенсорний модуль використовується для реєстрації первинного пульсового сигналу, а мікроконтролерна плата виконує зчитування значень IR та RED каналів і передавання даних через Wi-Fi. Під час складання прототипу перевірено фізичне підключення компонентів, стабільність живлення через USB, реакцію сенсора на контакт пальця та можливість отримання числових значень, придатних для подальшої обробки.

Для роботи пристрою збору даних розроблено прошивку ESP8266 V3.0 CH340 у середовищі Arduino IDE. У програмі реалізовано ініціалізацію інтерфейсу I2C, запуск сенсора MAX30102, підключення до WiFi-мережі, зчитування IR та RED значень, визначення наявності пальця на сенсорі, орієнтовний розрахунок ЧСС і формування CSV-рядка для передавання на сервер. Серверну частину реалізовано мовою Python із використанням фреймворку Flask. Сервер приймає CSV-дані від ESP8266 V3.0 CH340, перевіряє структуру отриманого рядка, фільтрує некоректні вимірювання, розраховує умовний показник Stress Score і текстовий рівень Stress Level, після чого зберігає записи у CSV-файлі. Вебінтерфейс забезпечує перегляд поточних значень IR, RED, ЧСС, умовного рівня стресу, кількості записів, середніх значень і графіка сигналу. Ця структура дозволяє контролювати роботу системи без ручного відкривання файлу з даними під час кожного вимірювання.

Результати експериментальної перевірки підтвердили, що розроблена система на базі IoT працює як єдиний апаратно-програмний комплекс. Зібраний прототип забезпечує зчитування біомедичних даних, їх передавання на сервер, фільтрацію, збереження та відображення у вебінтерфейсі. Подальше вдосконалення системи має бути пов’язане зі стабілізацією положення сенсора, покращенням алгоритмів обробки сигналу, додаванням інших біомедичних сенсорів і накопиченням більшої кількості експериментальних даних.

ВИСНОВКИ

У КБР розроблено прототип системи на базі IoT для збору та первинного аналізу біомедичних даних, які можуть використовуватися для орієнтовної оцінки рівня стресу. У роботі поєднано апаратну частину, програму для мікроконтролера, сервер на Python/Flask і вебінтерфейс.

Розглянуто, які фізіологічні зміни виникають під час стресової реакції та які біомедичні показники можуть бути корисними для технічного аналізу такого стану. Найбільше значення для цієї роботи мав пульсовий сигнал і ЧСС, оскільки ці дані можна отримати неінвазивно за допомогою компактного сенсора. При цьому враховано, що один показник не може точно визначати рівень стресу, тому результат системи розглядається як попередня програмна оцінка, а не як медичний висновок. Для апаратної частини обрано сенсор MAX30102 і плату ESP8266 V3.0 CH340. Сенсор використано для реєстрації IR та RED значень, а плата – для зчитування цих даних і передавання їх через Wi-Fi. Цей набір компонентів виявився компактним та не потребує складної схеми підключення й дозволяє перевірити основну ідею роботи на практиці.

У процесі проєктування сформовано логіку роботи системи – сенсор зчитує первинний сигнал, мікроконтролер формує CSV-рядок і надсилає його на сервер, після чого серверна частина перевіряє отримані значення, відкидає некоректні записи, зберігає дані у CSV-файлі та передає їх у вебінтерфейс. Цей розподіл зручний для ESP8266 V3.0 CH340, яка виконує тільки базові дії, а основна обробка і відображення результатів виконуються на стороні Python/Flask.

Практична реалізація підтвердила, що обрана архітектура може працювати як єдина система. Було зібрано апаратний вузол, перевірено підключення сенсору MAX30102 до ESP8266 V3.0 CH340, налаштовано роботу сенсора, WiFi-з’єднання та передавання даних на сервер. Для мікроконтролера підготовлено прошивку в Arduino IDE, а серверна частина отримала функції приймання, фільтрації, збереження та передавання даних у вебінтерфейс.

Під час експериментальної перевірки дані проходили повний шлях від сенсора до CSV-файлу та вебінтерфейсу. У вебінтерфейсі відображалися поточні

значення IR, RED, ЧСС, умовний показник Stress Score, рівень Stress Level, кількість записів і графік сигналу. Перевірка на трьох учасниках дала змогу отримати послідовність вимірювань і переконатися, що прототип здатний працювати в умовах звичайного макетного тестування. Під час перевірки стали помітними обмеження системи – сенсор MAX30102 досить чутливий до положення пальця, сили притискання, рухів руки та зовнішнього освітлення. Через це частина значень може різко змінюватися або потребувати відкидання на серверній частині.

Поставлені задачі вирішено в повному обсязі, а саме:

– проаналізовано фізіологічні основи стресу та біомедичні показники, придатні для його орієнтовної оцінки;

– досліджено іот-технології, апаратні засоби збору даних і методи обробки біомедичних сигналів та визначено вимоги до системи;

– розроблено архітектуру функціонального прототипу на базі іот для збору, передавання та аналізу біомедичних даних;

– обґрунтовано вибір апаратних і програмних засобів, розроблено схему підключення, алгоритм роботи та структуру серверної частини;

– реалізовано апаратну, програмну та серверну частини функціонального прототипу;

– проведено експериментальну перевірку прототипу, проаналізовано отримані результати та визначено напрями подальшого вдосконалення.

Проведена перевірка показала, що система може збирати, передавати, зберігати й відображати отримані біомедичні дані. Розроблений прототип підтвердив можливість створення доступної системи для збору та первинного аналізу біомедичних даних на базі IoT. Надалі його можна вдосконалити шляхом покращення кріплення сенсора, розрахунку показників варіабельності серцевого ритму, додавання інших сенсорів і переходу від CSV-файлу до бази даних, що дозволить зробити систему стабільнішою та придатнішою для подальших досліджень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дарнапук Є. С., Чередниченко Д. О. IoT-система збору та аналізу біометричних даних для оцінки рівня стресу. *Могилянські читання – 2025* : тези доп. XXVIII Всеукр. наук.-практ. конф., Миколаїв, 10–14 листоп. 2025 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2025. С. 106–109.

2. Стрес як ресурс: інструменти для підтримки психічного здоров'я. *Центр громадського здоров'я МОЗ України*. URL: https://phc.org.ua/news/stres-yak-resurs-instrumenti-dlya-pidtrimki-psikhichnogo-zdorovya (дата звернення: 28.04.2026).

3. Чередниченко Д., Дарнапук Є. Методи оцінювання стресового стану за допомогою IoT-системи на базі ESP8266 V3.0 CH340. *Ольвійський форум – 2026: стратегії країн Причорноморського регіону в геополітичному просторі* : тези доп. XXIII Міжнар. наук. конф., Миколаїв, 29 червня – 4 липня 2026 р. Миколаїв : ЧНУ ім. Петра Могили, 2026 *(подано до друку)*.

4. Agorastos A., Mansueto A. C., Hager T., Pappi E., Gardikioti A., Stiedl O. Heart Rate Variability as a Translational Dynamic Biomarker of Altered Autonomic Function in Health and Psychiatric Disease. *Biomedicines.* 2023. Vol. 11. №. 6:1591. DOI: 10.3390/biomedicines11061591.

5. Al-Atawi A. A., Alyahyan S., Alatawi M. N., Sadad T., Manzoor T., Farooq-i-Azam M., Khan Z. H. Stress Monitoring Using Machine Learning, IoT and Wearable Sensors. *Sensors.* 2023. Vol. 23. №. 21:8875. DOI: 10.3390/s23218875.

6. MAX30102. High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health : datasheet. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/max30102.pdf (Last accessed: 28.04.2026).

7. Getting Started with Arduino IDE 2. *ArduinoDocs*. URL: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2 (Last accessed: 28.04.2026).

8. Wire. *ArduinoDocs*. URL: https://www.arduino.cc/en/Reference/Wire (Last accessed: 01.06.2026).

9. Armañac-Julián P., Kontaxis S., Rapalis A., Marozas V., Laguna P., Bailón R., Gil E., Lázaro J. Reliability of Pulse Photoplethysmography Sensors: Coverage Using

Different Setups and Body Locations. *Frontiers in Electronics.* 2022. Vol. 3. P 1–11. DOI: 10.3389/felec.2022.906324.

10. Chart.js. *Chart.js Documentation.* URL: https://www.chartjs.org/docs/ (Last accessed: 01.06.2026).

11. ESP8266EX Datasheet. Version 7.1. URL: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf (Last accessed: 28.04.2026).

12. ESP8266 Arduino Core Documentation. URL: https://arduino-esp8266.readthedocs.io/ (Last accessed: 28.04.2026).

13. Flask-CORS. URL: https://flask-cors.readthedocs.io/en/latest/ (Last accessed: 01.06.2026).

14. Hendryani A., Gunawan D., Rizkinia M., Hidayati R. N., Hermawan F. Y. Real-Time Stress Detection and Monitoring System Using IoT-Based Physiological Signals. *Bulletin of Electrical Engineering and Informatics.* 2023. Vol. 12, №. 5. P. 2807–2815. DOI: 10.11591/eei.v12i5.5132.

15. Hernandez R., Schneider S., de Vries H. J., Fanning J., Ehrmann D., Jin H., Moore R. C., Juengst S., Stone A. A. Resting Heart Rate Variability Measured by Consumer Wearables and Its Associations with Diverse Health Domains in Five Longitudinal Studies. *Sensors.* 2025. Vol. 25. №. 23:7147. DOI: 10.3390/s25237147.

16. Kim K. B., Baek H. J. Photoplethysmography in Wearable Devices: A Comprehensive Review of Technological Advances, Current Challenges, and Future Directions. *Electronics.* 2023. Vol. 12. №. 13:2923. DOI: 10.3390/electronics12132923.

17. Fetch API. *MDN Web Docs.* URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (Last accessed: 01.06.2026).

18. Namvari M., Lipoth J., Knight S., Jamali A. A., Hedayati M., Spiteri R. J., Syed-Abdul S. Photoplethysmography Enabled Wearable Devices and Stress Detection: A Scoping Review. *Journal of Personalized Medicine.* 2022. Vol. 12. №. 11:1792. DOI: 10.3390/jpm12111792.

19. Nechyporenko A., Frohme M., Strelchuk Y., Omelchenko V., Gargin V., Ishchenko L., Alekseeva V. Galvanic Skin Response and Photoplethysmography for

Stress Recognition Using Machine Learning and Wearable Sensors. *Applied Sciences.* 2024. Vol. 14. №. 24:11997. DOI: 10.3390/app142411997.

20. Pallets. *Flask Documentation.* Version 3.1.x. URL: https://flask.palletsprojects.com/ (Last accessed: 28.04.2026).

21. Park J., Seok H. S., Kim S.-S., Shin H. Photoplethysmogram Analysis and Applications: An Integrative Review. *Frontiers in Physiology.* 2022. Vol. 12:808451. DOI: 10.3389/fphys.2021.808451.

22. Python Software Foundation. csv – CSV File Reading and Writing. Python 3 Documentation. URL: https://docs.python.org/3/library/csv.html (Last accessed: 01.06.2026).

23. Sammito S., Thielmann B., Böckelmann I. Update: factors influencing heart rate variability – a narrative review. *Frontiers in Physiology.* 2024. Vol. 15:1430458. DOI: 10.3389/fphys.2024.1430458.

24. SparkFun Electronics. *SparkFun MAX3010x Pulse and Proximity Sensor Library* : software library. URL: https://github.com/sparkfun/sparkfun_max3010x_sensor_library (Last accessed: 01.06.2026).

25. Talaat F. M., El-Balka R. M. Stress Monitoring Using Wearable Sensors: IoT Techniques in Medical Field. *Neural Computing and Applications.* 2023. Vol. 35. P. 18571–18584. DOI: 10.1007/s00521-023-08681-z.

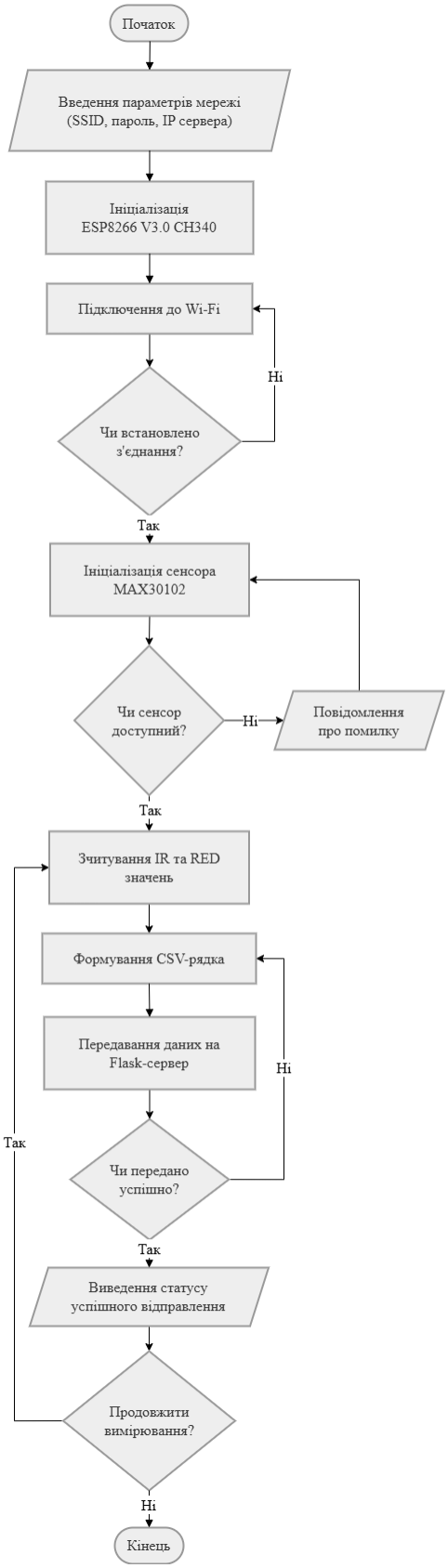
26. WCH. *USB to Serial Chip CH340* : datasheet. URL: https://www.wch-ic.com/downloads/CH340DS1_PDF.html (Last accessed: 28.04.2026).

27. World Health Organization. Stress. *Questions and answers.* 2026. URL: https://www.who.int/news-room/questions-and-answers/item/stress (Last accessed: 28.04.2026).

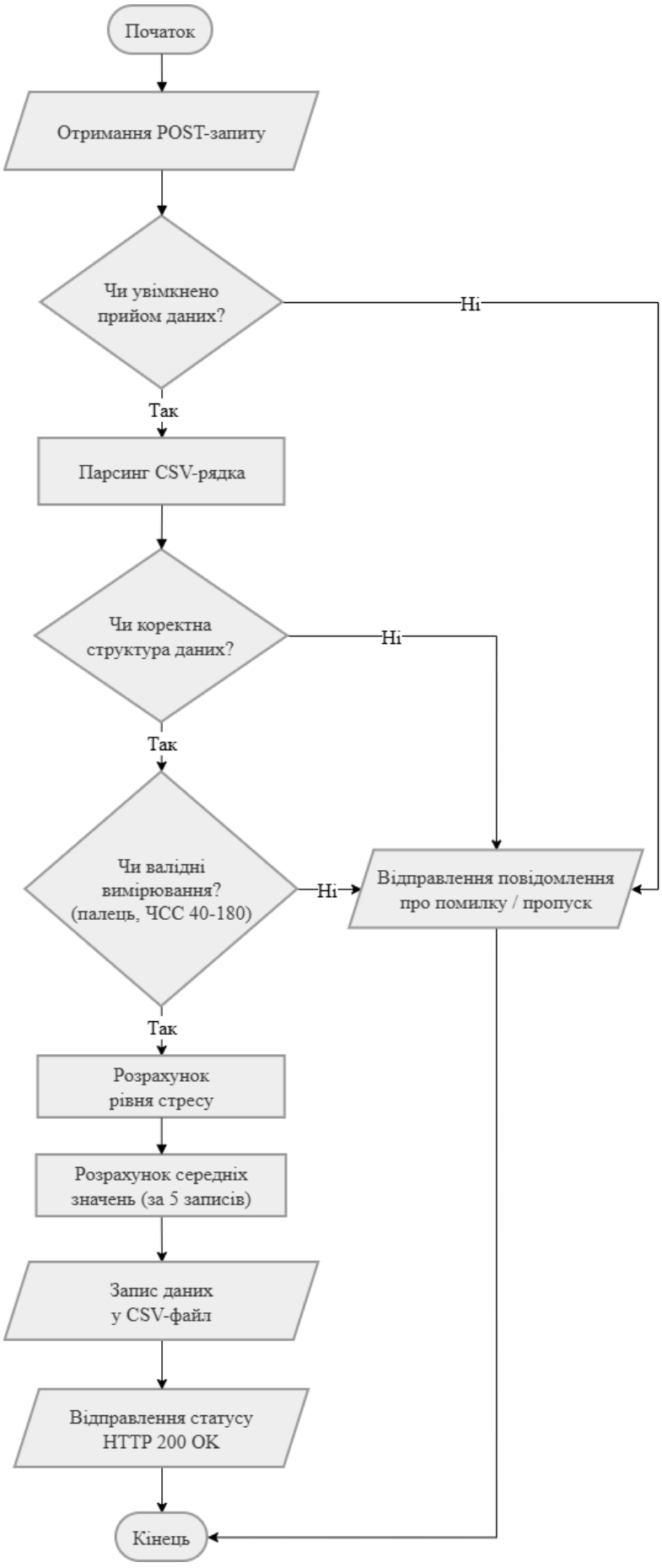
ДОДАТОК А

Блок-схеми апаратно-програмної ІоТ-системи

А.1 Блок-схема алгоритму функціонування ІоТ-системи



А.2 Блок-схема алгоритму коду обробки біомедичних даних



ДОДАТОК Б

Програмні коди для IoT-системи

Б.1 Код прошивки ESP8266 V3 CH340

```
#include <Wire.h>
#include "MAX30105.h"

#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <math.h>

#define SDA_PIN D2
#define SCL_PIN D1

MAX30105 sensor;

const char* ssid = "iPhone 13";
const char* password = "30005071";
const char* serverUrl = "http://172.20.10.7:5000/upload";

const int DATA_BUFFER_SIZE = 100;
const int UPDATE_SAMPLES = 25;
const int SAMPLE_RATE = 50;

uint32_t irBuffer[DATA_BUFFER_SIZE];
uint32_t redBuffer[DATA_BUFFER_SIZE];

void connectToWiFi() {
  Serial.print("Connecting to Wi-Fi: ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  int attempts = 0;

  while (WiFi.status() != WL_CONNECTED && attempts < 40) {
    delay(500);
    Serial.print(".");
    attempts++;
  }

  Serial.println();

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("Wi-Fi connected");
    Serial.print("ESP8266 IP address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("Wi-Fi connection failed");
  }
}

void sendCsvToServer(long ir, long red, int hr, bool hrValid, bool finger) {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Wi-Fi disconnected. Reconnecting...");
    connectToWiFi();
    return;
  }

  WiFiClient client;
  HTTPClient http;

  if (!http.begin(client, serverUrl)) {
    Serial.println("HTTP connection failed");
    return;
  }

  http.addHeader("Content-Type", "text/csv");

  String csvPayload = "";
```

```
  csvPayload += String(ir);
  csvPayload += ";";
  csvPayload += String(red);
  csvPayload += ";";
  csvPayload += String(hr);
  csvPayload += ";";
  csvPayload += (hrValid ? "1" : "0");
  csvPayload += ";";
  csvPayload += (finger ? "1" : "0");

  int httpCode = http.POST(csvPayload);

  Serial.print("POST code: ");
  Serial.println(httpCode);

  Serial.print("CSV payload: ");
  Serial.println(csvPayload);

  if (httpCode > 0) {
    String response = http.getString();
    Serial.print("Server response: ");
    Serial.println(response);
  } else {
    Serial.print("HTTP error: ");
    Serial.println(http.errorToString(httpCode));
  }

  http.end();
}

void readSample(int index) {
  while (sensor.available() == false) {
    sensor.check();
  }

  redBuffer[index] = sensor.getRed();
  irBuffer[index] = sensor.getIR();

  sensor.nextSample();
}

float calculateMeanIR() {
  double sum = 0;

  for (int i = 0; i < DATA_BUFFER_SIZE; i++) {
    sum += irBuffer[i];
  }

  return sum / DATA_BUFFER_SIZE;
}

float calculateDeviationIR(float meanValue) {
  double sum = 0;

  for (int i = 0; i < DATA_BUFFER_SIZE; i++) {
    float difference = irBuffer[i] - meanValue;
    sum += difference * difference;
  }

  return sqrt(sum / DATA_BUFFER_SIZE);
}

int calculateHeartRateFromIR(bool &validResult) {
  validResult = false;

  float meanValue = calculateMeanIR();
  float deviationValue = calculateDeviationIR(meanValue);

  if (deviationValue < 50) {
    return 0;
  }

  float threshold = meanValue + deviationValue * 0.25;

  int peaks[20];
  int peakCount = 0;
```

```
  int minPeakDistance = 16;
  int maxPeakDistance = 75;

  for (int i = 1; i < DATA_BUFFER_SIZE - 1; i++) {
    bool isPeak = irBuffer[i] > threshold &&
                  irBuffer[i] > irBuffer[i - 1] &&
                  irBuffer[i] >= irBuffer[i + 1];

    if (isPeak) {
      if (peakCount == 0 || i - peaks[peakCount - 1] >= minPeakDistance) {
        peaks[peakCount] = i;
        peakCount++;
      } else {
        if (irBuffer[i] > irBuffer[peaks[peakCount - 1]]) {
          peaks[peakCount - 1] = i;
        }
      }
    }

    if (peakCount >= 20) {
      break;
    }
  }

  if (peakCount < 2) {
    return 0;
  }

  int totalInterval = 0;
  int validIntervals = 0;

  for (int i = 1; i < peakCount; i++) {
    int interval = peaks[i] - peaks[i - 1];

    if (interval >= minPeakDistance && interval <= maxPeakDistance) {
      totalInterval += interval;
      validIntervals++;
    }
  }

  if (validIntervals < 1) {
    return 0;
  }

  float averageInterval = (float)totalInterval / validIntervals;
  int bpm = round((60.0 * SAMPLE_RATE) / averageInterval);

  if (bpm < 40 || bpm > 180) {
    return 0;
  }

  validResult = true;
  return bpm;
}

void calculatePrintAndSendResult() {
  long lastIR = irBuffer[DATA_BUFFER_SIZE - 1];
  long lastRED = redBuffer[DATA_BUFFER_SIZE - 1];

  bool fingerDetected = lastIR > 10000;

  if (!fingerDetected) {
    Serial.println();
    Serial.println("RESULT: NO FINGER DETECTED");

    sendCsvToServer(
      lastIR,
      lastRED,
      0,
      false,
      false
    );

    Serial.println("----------------------------");
    return;
  }
```

```
  bool heartRateValid = false;
  int calculatedHeartRate = calculateHeartRateFromIR(heartRateValid);

  Serial.println();
  Serial.println("RESULT FROM LAST 100 SAMPLES");

  Serial.print("IR: ");
  Serial.println(lastIR);

  Serial.print("RED: ");
  Serial.println(lastRED);

  Serial.print("Heart Rate: ");
  Serial.print(calculatedHeartRate);
  Serial.print(" BPM");

  if (heartRateValid) {
    Serial.println(" | VALID");
  } else {
    Serial.println(" | NOT VALID");
  }

  sendCsvToServer(
    lastIR,
    lastRED,
    calculatedHeartRate,
    heartRateValid,
    fingerDetected
  );

  Serial.println("------------------------------");
}

void collectInitialSamples() {
  Serial.println();
  Serial.println("COLLECTING INITIAL DATA");
  Serial.println("sample;IR;RED");

  for (int i = 0; i < DATA_BUFFER_SIZE; i++) {
    readSample(i);

    Serial.print(i + 1);
    Serial.print(";");
    Serial.print(irBuffer[i]);
    Serial.print(";");
    Serial.println(redBuffer[i]);
  }

  calculatePrintAndSendResult();
}

void shiftBuffer() {
  for (int i = UPDATE_SAMPLES; i < DATA_BUFFER_SIZE; i++) {
    redBuffer[i - UPDATE_SAMPLES] = redBuffer[i];
    irBuffer[i - UPDATE_SAMPLES] = irBuffer[i];
  }
}

void collectNewSamples() {
  Serial.println();
  Serial.println("COLLECTING NEW DATA");
  Serial.println("sample;IR;RED");

  for (int i = DATA_BUFFER_SIZE - UPDATE_SAMPLES; i < DATA_BUFFER_SIZE; i++) {
    readSample(i);

    Serial.print(i + 1);
    Serial.print(";");
    Serial.print(irBuffer[i]);
    Serial.print(";");
    Serial.println(redBuffer[i]);
  }

  calculatePrintAndSendResult();
}

void setup() {
```

```
  Serial.begin(115200);
  delay(1000);

  Serial.println();
  Serial.println("ESP8266 MAX30102 FLASK CSV SYSTEM STARTED");

  Wire.begin(SDA_PIN, SCL_PIN);
  Wire.setClock(100000);

  if (!sensor.begin(Wire, I2C_SPEED_STANDARD)) {
    Serial.println("MAX30102 not found");
    while (1) {
      delay(1000);
    }
  }

  Serial.println("MAX30102 found");

  byte ledBrightness = 0x3F;
  byte sampleAverage = 4;
  byte ledMode = 2;
  int sampleRate = 50;
  int pulseWidth = 411;
  int adcRange = 16384;

  sensor.setup(
    ledBrightness,
    sampleAverage,
    ledMode,
    sampleRate,
    pulseWidth,
    adcRange
  );

  sensor.setPulseAmplitudeRed(0x3F);
  sensor.setPulseAmplitudeIR(0x3F);
  sensor.setPulseAmplitudeGreen(0);

  connectToWiFi();

  Serial.println("Place your finger on the sensor and keep it still");
}

void loop() {
  collectInitialSamples();

  while (true) {
    shiftBuffer();
    collectNewSamples();
  }
}
```

Б.2 Програмний код обробки біомедичних даних

```
from flask import Flask, request, jsonify, render_template, send_file, redirect, make_response
from flask_cors import CORS
from datetime import datetime
import os
import csv

app = Flask(__name__)
CORS(app)

DATA_DIR = "data"
CSV_FILE = os.path.join(DATA_DIR, "sensor_data.csv")
CSV_DELIMITER = ";"

receiving_enabled = True

CSV_HEADER = [
    "timestamp",
    "ir",
    "red",
    "heart_rate",
    "valid_hr",
    "finger",
    "stress_score",
    "stress_level",
    "average_heart_rate_5",
    "average_stress_score_5",
    "average_stress_level_5"
]

def parse_bool(value):
    value = str(value).strip().lower()
    return value in ["1", "true", "yes"]

def calculate_stress_level(heart_rate, valid_hr, finger):
    if not finger or not valid_hr or heart_rate <= 0:
        return 0, "No valid data"

    if heart_rate < 60:
        return 20, "Low"

    if 60 <= heart_rate <= 90:
        return 35, "Normal"

    if 91 <= heart_rate <= 105:
        return 60, "Moderate"

    if 106 <= heart_rate <= 120:
        return 80, "Increased"

    return 95, "High"

def calculate_average_stress_level(average_stress_score):
    if average_stress_score <= 0:
        return "No valid data"

    if average_stress_score < 30:
        return "Low"

    if 30 <= average_stress_score < 50:
        return "Normal"

    if 50 <= average_stress_score < 70:
        return "Moderate"

    if 70 <= average_stress_score < 90:
        return "Increased"

    return "High"

def is_valid_measurement(heart_rate, valid_hr, finger):
    if not finger:
        return False
```

```
    if not valid_hr:
        return False

    if heart_rate < 40 or heart_rate > 180:
        return False

    return True

def init_storage():
    os.makedirs(DATA_DIR, exist_ok=True)

    if not os.path.exists(CSV_FILE):
        with open(CSV_FILE, mode="w", newline="", encoding="utf-8-sig") as file:
            writer = csv.writer(file, delimiter=CSV_DELIMITER)
            writer.writerow(CSV_HEADER)
        return

    with open(CSV_FILE, mode="r", newline="", encoding="utf-8-sig") as file:
        reader = csv.reader(file, delimiter=CSV_DELIMITER)
        current_header = next(reader, [])

    if current_header != CSV_HEADER:
        with open(CSV_FILE, mode="w", newline="", encoding="utf-8-sig") as file:
            writer = csv.writer(file, delimiter=CSV_DELIMITER)
            writer.writerow(CSV_HEADER)

def append_to_csv(record):
    with open(CSV_FILE, mode="a", newline="", encoding="utf-8-sig") as file:
        writer = csv.writer(file, delimiter=CSV_DELIMITER)
        writer.writerow([
            record["timestamp"],
            record["ir"],
            record["red"],
            record["heart_rate"],
            1 if record["valid_hr"] else 0,
            1 if record["finger"] else 0,
            record["stress_score"],
            record["stress_level"],
            record["average_heart_rate_5"],
            record["average_stress_score_5"],
            record["average_stress_level_5"]
        ])

def read_csv_records():
    if not os.path.exists(CSV_FILE):
        return []

    records = []

    with open(CSV_FILE, mode="r", newline="", encoding="utf-8-sig") as file:
        reader = csv.DictReader(file, delimiter=CSV_DELIMITER)

        for row in reader:
            try:
                records.append({
                    "timestamp": row.get("timestamp", ""),
                    "ir": int(row.get("ir", 0)),
                    "red": int(row.get("red", 0)),
                    "heart_rate": int(row.get("heart_rate", 0)),
                    "valid_hr": parse_bool(row.get("valid_hr", "0")),
                    "finger": parse_bool(row.get("finger", "0")),
                    "stress_score": int(row.get("stress_score", 0)),
                    "stress_level": row.get("stress_level", "No valid data"),
                    "average_heart_rate_5": row.get("average_heart_rate_5", ""),
                    "average_stress_score_5": row.get("average_stress_score_5", ""),
                    "average_stress_level_5": row.get("average_stress_level_5", "")
                })
            except ValueError:
                continue

    return records

def calculate_averages_for_each_five_records(records, current_record):
    temporary_records = records + [current_record]
    record_count = len(temporary_records)

    if record_count % 5 != 0:
```

```
        current_record["average_heart_rate_5"] = ""
        current_record["average_stress_score_5"] = ""
        current_record["average_stress_level_5"] = ""
        return current_record

    last_five_records = temporary_records[-5:]

    average_heart_rate = round(
        sum(record["heart_rate"] for record in last_five_records) / 5,
        1
    )

    average_stress_score = round(
        sum(record["stress_score"] for record in last_five_records) / 5,
        1
    )

    average_stress_level = calculate_average_stress_level(average_stress_score)

    current_record["average_heart_rate_5"] = average_heart_rate
    current_record["average_stress_score_5"] = average_stress_score
    current_record["average_stress_level_5"] = average_stress_level

    return current_record

def get_latest_average(records):
    for record in reversed(records):
        if record.get("average_heart_rate_5") not in ["", None]:
            return {
                "average_heart_rate_5": record.get("average_heart_rate_5", ""),
                "average_stress_score_5": record.get("average_stress_score_5", ""),
                "average_stress_level_5": record.get("average_stress_level_5", "")
            }

    return {
        "average_heart_rate_5": "",
        "average_stress_score_5": "",
        "average_stress_level_5": ""
    }

def parse_incoming_csv(raw_data):
    if ";" in raw_data:
        delimiter = ";"
    else:
        delimiter = ","

    return next(csv.reader([raw_data], delimiter=delimiter))

def no_cache_response(response):
    response.headers["Cache-Control"] = "no-store, no-cache, must-revalidate, max-age=0"
    response.headers["Pragma"] = "no-cache"
    response.headers["Expires"] = "0"
    return response

@app.route("/")
def root():
    return redirect("/main")

@app.route("/main")
def main_page():
    response = make_response(render_template("index.html"))
    return no_cache_response(response)

@app.route("/upload", methods=["POST"])
def upload_data():
    global receiving_enabled

    if not receiving_enabled:
        print("Receiving is stopped. Incoming data skipped.")
        return "PAUSED: receiving stopped", 200

    raw_data = request.get_data(as_text=True).strip()

    if not raw_data:
        return "ERROR: CSV data not received", 400

    try:
```

```
        csv_row = parse_incoming_csv(raw_data)
    except Exception:
        return "ERROR: CSV row cannot be parsed", 400

    if len(csv_row) < 5:
        return "ERROR: CSV row has wrong structure", 400

    try:
        ir = int(csv_row[0])
        red = int(csv_row[1])
        heart_rate = int(csv_row[2])
        valid_hr = parse_bool(csv_row[3])
        finger = parse_bool(csv_row[4])

        if not is_valid_measurement(heart_rate, valid_hr, finger):
            print("Received CSV:", raw_data)
            print("Skipped invalid measurement")
            return "SKIPPED: invalid measurement", 200

        stress_score, stress_level = calculate_stress_level(
            heart_rate,
            valid_hr,
            finger
        )

        record = {
            "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            "ir": ir,
            "red": red,
            "heart_rate": heart_rate,
            "valid_hr": valid_hr,
            "finger": finger,
            "stress_score": stress_score,
            "stress_level": stress_level,
            "average_heart_rate_5": "",
            "average_stress_score_5": "",
            "average_stress_level_5": ""
        }

        existing_records = read_csv_records()
        record = calculate_averages_for_each_five_records(existing_records, record)

    except ValueError:
        return "ERROR: CSV contains incorrect values", 400

    append_to_csv(record)

    print("Received CSV:", raw_data)
    print("Saved valid record:", record)

    return "OK: valid data saved", 200

@app.route("/data", methods=["GET"])
def get_data():
    global receiving_enabled

    records = read_csv_records()
    latest_average = get_latest_average(records)

    response = jsonify({
        "receiving_enabled": receiving_enabled,
        "count": len(records),
        "latest": records[-1] if records else None,
        "latest_average": latest_average,
        "records": records[-100:]
    })

    return no_cache_response(response)

@app.route("/toggle-receiving", methods=["POST"])
def toggle_receiving():
    global receiving_enabled

    receiving_enabled = not receiving_enabled

    if receiving_enabled:
        message = "Receiving started"
```

```
    else:
        message = "Receiving stopped"

    print(message)

    response = jsonify({
        "receiving_enabled": receiving_enabled,
        "message": message
    })

    return no_cache_response(response)

@app.route("/clear", methods=["POST", "GET"])
def clear_data():
    with open(CSV_FILE, mode="w", newline="", encoding="utf-8-sig") as file:
        writer = csv.writer(file, delimiter=CSV_DELIMITER)
        writer.writerow(CSV_HEADER)

    print("CSV file cleared")

    response = jsonify({
        "status": "success",
        "message": "Data cleared"
    })

    return no_cache_response(response)

@app.route("/download", methods=["GET"])
def download_csv():
    if not os.path.exists(CSV_FILE):
        return "ERROR: CSV file not found", 404

    response = send_file(
        CSV_FILE,
        as_attachment=True,
        download_name="sensor_data.csv",
        mimetype="text/csv",
        max_age=0
    )

    return no_cache_response(response)

if __name__ == "__main__":
    init_storage()

    app.run(
        host="0.0.0.0",
        port=5000,
        debug=True
    )
```

ДОДАТОК В

Матеріали апробації роботи

В.1 XXVIII Всеукраїнська науково-практична конференція «Могилянські читання – 2025»

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2025:
досвід та тенденції розвитку суспільства в Україні: глобальний,
національний та регіональний аспекти»**

XXVIII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 10–14 листопада 2025 року

Миколаїв – 2025

УДК 004.382:004.775:61

Дарнапук Є. С.,
PhD, доцент (б. в. з.) кафедри комп'ютерної інженерії,
Чередниченко Д. О.,
бакалаврант кафедри комп'ютерної інженерії,
Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна

IOT-СИСТЕМА ЗБОРУ ТА АНАЛІЗУ БІОМЕТРИЧНИХ ДАНИХ ДЛЯ ОЦІНКИ РІВНЯ СТРЕСУ

Стрес є природною реакцією організму на зміни середовища, проте його хронічні форми можуть призводити до порушень функціонування нервової, серцево-судинної та ендокринної систем, а також до зниження когнітивної продуктивності та якості життя людини. Проблема виявлення стресових станів має міждисциплінарний характер, поєднуючи психологічні, медичні та інженерні аспекти.

Традиційні методи оцінки рівня стресу (анкетування, опитувальники, лабораторні дослідження) мають обмежену об'єктивність і не забезпечують можливості постійного моніторингу у реальному часі. Нові можливості відкривають технології Інтернету речей (англ. Internet of Things, IoT), що дозволяють збирати та передавати біометричні сигнали через сенсорні пристрої, мікроконтролери та аналітичні модулі.

Метою роботи є створення інтелектуальної IoT-системи збору та аналізу біометричних даних для автоматизованої оцінки рівня стресу користувача, здатної функціонувати у реальному часі. Основна ідея полягає у поєднанні фізіологічних вимірювань, цифрової обробки сигналів та алгоритмів аналітичного оцінювання.

Запропонований підхід ґрунтується на інтеграції фізіологічних маркерів автономної нервової системи – частоти серцевих скорочень (HR), варіабельності серцевого ритму (HRV), шкірно-гальванічної реакції (GSR) та температури шкіри. Дані параметри є найбільш інформативними для оцінювання психоемоційного стану людини [1].

Розвиток напрямів Internet of Medical Things (IoMT) і сенсорних технологій дозволив створювати портативні пристрої для автономного моніторингу здоров'я. Сучасні мікроконтролери, зокрема ESP32 (рис. 1), забезпечують достатню обчислювальну потужність і енергоефективність для реалізації систем збору, аналізу та передавання даних без потреби у дорогих серверних ресурсах [4; 5].

В.2 XXIII Міжнародна наукова конференція «Ольвійський форум – 2026: стратегії країн Причорноморського регіону в геополітичному просторі»

DOI

Дмитро ЧЕРЕДНИЧЕНКО
Євген ДАРНАПУК

МЕТОДИ ОЦІНЮВАННЯ СТРЕСОВОГО СТАНУ ЗА ДОПОМОГОЮ IOT-СИСТЕМИ НА БАЗІ ESP8266 V3.0 CH340

Стрімкий ритм сучасного існування вимагає точного виявлення емоційного та психологічного тиску, що ставить розробку дієвих засобів спостереження за стресом. Проведено вивчення фізіологічних процесів, що лежать в основі стресових відгуків, і обґрунтовано вибір визначальних біометричних індикаторів, серед яких виділяється варіабельність серцевого ритму (ВСР), задля їхньої точної верифікації. Вагоме місце у роботі займає впровадження рішень Інтернету Речей (IoT), які дають змогу безперервно збирати та передавати інформацію в режимі, наближеному до дійсного. Представлена цілісна структура системи нагляду, збудована на основі мікроконтролера та спеціалізованих датчиків. Докладно описано стадії опрацювання біометричних сигналів, виокремлення найбільш змістовних характеристик та використання навчальних алгоритмів для стійкого розрізнення ступенів навантаження. Демонструється, що поєднання передових сенсорних рішень та розумних методів аналізу відкриває свіжі обрії для формування автоматизованих комплексів контролю за психофізіологічним станом індивідуально для кожного.

Ключові слова: *стрес, HRV, IoT, біометричні дані, сенсори, машинне навчання, моніторинг.*

Стрес визначається як неспецифічна адаптаційна реакція організму на дію зовнішніх або внутрішніх подразників. У фізіологічному аспекті він пов'язаний з активацією гіпоталамо-гіпофізарно-наднирникової системи та симпатичного відділу вегетативної нервової системи [1]. У відповідь на стресовий фактор в організмі підвищується рівень адреналіну та кортизолу, що призводить до змін серцевої діяльності, судинного тонусу, потовиділення та температурної регуляції. Найбільш інформативними показниками для оцінювання стресового стану є частота серцевих скорочень (HR), варіабельність серцевого ритму (HRV), шкірно-гальванічна реакція (GSR), температура тіла та електрокардіографічні параметри. Зокрема, під час стресу зазвичай спостерігається підвищення HR, зниження HRV та зростання електропровідності шкіри. Варіабельність серцевого ритму є особливо цінним показником, оскільки вона відображає баланс між симпатичною та парасимпатичною регуляцією серцевої діяльності та широко використовується у сучасних дослідженнях стресу [1–2].

Найбільш інформативними показниками для оцінювання стресового стану є частота серцевих скорочень (HR), варіабельність серцевого ритму (HRV), шкірно-гальванічна реакція (GSR), температура тіла та електрокардіографічні параметри. Зокрема, під час стресу зазвичай спостерігається підвищення HR, зниження HRV та зростання електропровідності шкіри. Варіабельність серцевого ритму є особливо цінним показником, оскільки вона відображає баланс між симпатичною та парасимпатичною регуляцією серцевої діяльності.

Таблиця 1

Основні біометричні показники для оцінювання стресу

Показник	Фізіологічна природа	Зміна при стресі	Практична значущість
HR	Серцева активність	Підвищення	Швидка оцінка стану
HRV	Баланс ВНС	Зниження	Висока діагностична цінність
GSR	Активність потових залоз	Підвищення	Маркер емоційної реакції