Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри,
д-р техн. наук, проф.
_____ І. М. Журавська
« __ » _____ 2023 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Програмно-апаратний комплекс моніторингу та аналізу фізіологічних показників стану людини на базі модуля ESP32

Спеціальність «Комп'ютерна інженерія»
123 – КМР.1 – 605.221710526

Студентка _____ І. В. Сімакова
підпис
«__» _____ 2023 р.

Керівник к.т.н., доцент _____ Я. М. Крайник
підпис
«__» _____ 2023 р.

Миколаїв 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АЦП – аналогово-цифровий перетворювач

ІЧ – інфрачервоний

УЗД – Ультразвукова діагностика

ARM – Advanced RISC Machine

EEPROM – Electrically Erasable Programmable Read-Only Memory

GPIO – General-purpose input/output

FIFO – first in, first out (перший прийшов перший вийшов), принцип накопичення та обробки завдань

PPG – photoplethysmography (фотоплетізмографія)

I2C – Inter-Integrated Circuit (послідовна шина даних для зв'язку інтегральних схем)

Mbyte – Мегабайт

Mbps – Мегабіти за секунду

Kbyte – Кілобайти

Kbps – Кілобіти за секунду

LCD – Liquid crystal display (рідкокристалічний дисплей)

SpO2 – Сатурація кисню та пульс

SaO2 – Насичення артеріальної крові киснем

ВСТУП

Станом на кінець 2022 року, пандемія коронавірусної хвороби SARS-COVID-19 все ще досі поширюється і завдає шкоди багатьом людям своїм перебігом та наслідками перенесеної хвороби як окремо людині, так і державам загалом. Одним з основних симптомів того, що дана хвороба спричинить пневмонію є зниження рівня кисневої сатурації крові пацієнта відносно норми. Після експериментального підтвердження цього симптому, пристрої виміру сатурації та пульсу стали набагато частіше куплятися простими людьми, модулі з сенсорами пульсоксиметрії стали обов’язковими в більшості фітнес-браслетів, а вимірювання сатурації стало ледь не обов’язковим регулярним медичним дослідженням для хворих на пневмонію

Актуальність цієї роботи обумовлена малою кількістю на ринку медичних та спортивних пристроїв, які якісно відслідковують та оброблюють фізіологічні показники життєдіяльності людини, такі як пульс, оксигенація крові та температура тіла.

Мета роботи: створення апаратно-програмного комплексу моніторингу та аналізу фізіологічних показників стану людини, який може бути застосованим як для ефективного добового моніторингу стану пацієнта хворого на COVID-19 і його контрольованого реабілітаційного відновлення, так і для базового моніторингу фізіологічних показників спортсмена або простого користувача комплексу.

Об’єкт дослідження: процеси збору, обробки та представлення інформації щодо отриманих фізіологічних показників людини.

Предмет дослідження: методи, засоби та технології збору та обробки фізіологічних показників людини, таких як серцевий ритм, киснева сатурація крові та температура для подальшого аналізу та представлення отриманих даних.

Методи дослідження

1) метод «порівняння» – для порівняння використано декілька систем з подібним функціоналом для визначення спільного функціоналу, переваг та недоліків подібних систем;

2) метод «індукція» – сформовано ряд індукційних висновків стосовно можливості використання певного апаратного та програмного забезпечення для реалізації комплексу збору та обробки даних певних фізіологічних показників людини;

3) методи «Аналіз» та «Синтез» – застосовано для декомпозиції окремих функцій пристроїв збору та обробки серцевого ритму, кисневої сатурації крові та температури для визначення апаратних та програмних складових, що необхідні для їх реалізації;

4) метод «Моделювання» – використано для побудови програмного забезпечення на початковому етапі буде розроблено модель архітектури, що відображатиме взаємозв’язок модулів.

Завдання, які необхідно вирішити для досягнення мети:

– дослідити наявні системи збору та обробки фізіологічних показників людини;

– провести порівняльний аналіз наявних систем для визначення їх переваг та недоліків;

– визначити апаратні та програмні компоненти для побудови апаратно-програмного комплексу

– розробити та здійснити апробацію методу первинної обробки отриманих даних сенсором MAX30102, який працює за принципом фотоплетизмографії;

– розробити та здійснити апробацію методів передачі даних поміж пристроєм збору показників та серверним програмним забезпеченням з використанням алгоритмів шифрування даних;

– розробити та здійснити апробацію методів обробки, фільтрації та візуалізації отриманих показників стану людини;

– розробити апаратний модуль збору фізіологічних показників людини;

– розробити програмне забезпечення для апаратного модулю, серверної та клієнтської частини апаратно-програмного комплексу;

– порівняти результати роботи створеного апаратно-програмного комплексу з комерційними аналогами

Гіпотеза – передбачається що застосування алгоритмів активної діагностики та пропоновані можливості обробки, накопичення, протоколювання даних і доступу до них лікарів для подальшого аналізу активного відгуку та формування діагностичних заключень є інноваційним кроком, що суттєво скорочує трудовитрати пов’язані з витратами допоміжного (на доставку лікаря до пацієнта та зворотно) та основного часу лікаря, оскільки зменшується необхідність витрат часу на спостереження, проводяться у наслідок автоматизації постійно з деталізованим моніторингом функціонування.

Наукова новизна. ґрунтується на створенні та використанні традиційних та модернізованих методів збору, обробки та візуалізації фізіологічних показників людини з використанням компонентів, що здешевлюють ціну пристроїв збору та первинної обробки отриманих даних таких як ESP32 та сенсор пульсоксиметрії MAX30102 а також додають можливість до подальшого удосконалення пристрою через гнучкість та відкритість платформи ESP32.

Сам створений комплекс надає можливість як користувачеві так і спеціалісту отримати потрібний йому оброблений та візуалізований набір даних для його використання в тих чи інших цілях, наприклад, для аналізу перебігу стану хвороби пацієнта на COVID-19.

Практичне значення: проєкт передбачає розробку апаратно-програмного комплексу, що складається з трьох компонентів: пристрою збору, первинної обробки даних та їх передачі на сервер; серверного програмного забезпечення для глибинної обробки, аналізу та генерації представлень отриманих даних; клієнтського програмного забезпечення, що надає користувачеві представлення оброблених даних.

Кваліфікаційна робота пройшла апробацію під час конференцій [1]–[3].

1 АНАЛІТИЧНА ЧАСТИНА. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ СИСТЕМ ЗБОРУ ТА АНАЛІЗУ ФІЗІОЛОГІЧНИХ ПОКАЗНИКІВ ЛЮДИНИ

1.1 Загальний огляд проблеми збору та аналізу фізіологічних показників людини

Збір медичних показників є важливим елементом моніторингу стану людини як для діагностування хвороби, визначенням патологій, протіканням анамнезу хвороби, так і для моніторингу певних показників задля коригування свого повсякденного життя. До медичних показників можна віднести частоту серцевих скорочень, артеріальний тиск, рівень оксигенації крові, електрокардіограми, рівень цукру і т.д. Для вирішення першої проблеми, а саме – отримання певних медичних показників організму, існують величезна кількість пристроїв. Наприклад, електрокардіографи, пульсоксиметри, пульсометри, УЗД сканери, магнітно-резонансні томографи, тощо. Дані, отримані з цих пристроїв, повинні мати достатню точність для формування тих чи інших висновків про стан людини. Для задоволення інших проблем, як наприклад, корегування способу життя заради збільшення свого персонального комфорту використовуються портативні фітнес-трекери, що мають розмір з наручний годинник. Дуже часто такий тип пристроїв має додатковий функціонал а також меншу точність отриманих показників фізичного.

У залежності від типу даних, що треба зібрати, сучасний ринок пропонує величезну кількість моделей пристроїв різної цінової категорії та застосування. Для повсякденного рутинного моніторингу певних базових показників, як от, наприклад, вимірювання пульсу, кількості кроків, оксигенації крові, електрокардіограм, можна використовувати такі пристрої як Apple Watch (починаючи з серії 6 містить можливість отримання електрокардіограми) Xiaomi Mi Band (з версії 5 підтримується вимірювання рівня оксигенації крові), Garmin Venu 2 Plus і т. д. Недоліком всіх цих наявних на ринку пристроїв є або закритість платформи, або важкість

отримання зібраних даних в потрібному для подальшої обробки форматі, або відсутність можливості створення надбудов для додаткового аналізу отриманих даних.

1.2 Огляд існуючих рішень на ринку

Станом на кінець 2022 року, існує досить велика кількість пристроїв та комплексів, що частково або повністю реалізують подібний функціонал кваліфікаційної роботи. Наприклад, Apple Watch Ultra (рис.1.1) має можливість збору та первинної обробки даних температури тіла, серцевого ритму, рівня кров’яної сатурації киснем, електрокардіограми. Також заявлена виробником автономність роботи пристрою може перевищувати 180 годин та має високий рівень захисту супроти пилу, води, ударів, вібрації та перепадів температури. Але, на жаль, виробник заявляє, що пристрій не призначений для використання в медичній діагностиці, лікуванні та медичних цілях. [4]

Рисунок 1.1 – Apple Watch Ultra

Подібний функціонал має і Samsung Galaxy Watch 4 (рис.1.2), але окрім того має можливість знімати показники артеріального тиску. Як заявляється виробником, для цього в пристрої використовується PPG сенсор.[5]

Рисунок 1.2 – Samsung Galaxy Watch 4

Трохи менший функціонал має наступний пристрій – Xiaomi Mi Band 7 Pro (рис. 1.3). У порівнянні з попередніми двома пристроями, у нього відсутня можливість виміру артеріального тиску та електроміографії, також нема точного опису рівня захисту пристрою[6]. Але в той же час його ціна в рази менша за аналоги, тож для персонального не медичного відслідковування певних фізіологічних показників цей пристрій підходить.

Рисунок 1.3 – Xiaomi Mi Band 7 Pro

Загалом, спрощена структура пристроїв такого типу складається з таких компонентів:

1. Центральний контролер
2. Фотоплетизмографічні сенсори (від одного до декількох десятків)
3. Гіроскоп або акселерометр
4. Модуль передачі даних (Bluetooth або WiFi)
5. Екран для виведення інформації.

1.3 Формування вимог до програмно-апаратного комплексу

Internet-of-Things (IoT) рішення є одним зі способів вирішення проблем наявних на ринку пристроїв. Сама концепція IoT складається із взаємопов’язаних фізичних пристроїв з програмним забезпеченням, що здійснюють обмін даними між зовнішнім світом та комп’ютерними системами за допомогою стандартних протоколів зв’язку. А мікроконтролер ESP32 [7], [8] є одним з головних інструментів у вивченні IoT. Серія цих мікроконтролерів відноситься до типу «Система на кристалі», мають інтегровані контролери Wi-Fi та Bluetooth, низьке енергоспоживання та надають можливість використання повноцінної Linux системи на цьому малому чіпі і все це за відносно низької ціни контролеру. ESP32 підключає до себе сенсори та актуатори за допомогою GPIO пінів. Поєднання ESP32 та IoT надає новий поштовх в розвитку технологій в системі охорони здоров’я [7].

Рисунок 1.4 – ESP32 мікроконтролер

Мікроконтролер використовує мікропроцесор Tensilica Xtensa LX6 в двоядерних та одноядерних варіаціях та включає вбудовані антенні перемикачі, підсилювач потужності, приймач з низьким рівнем шумів, фільтри та модулі керування живленням. ESP32 можна використовувати як у якості головного мікроконтролеру для окремо взятого девайсу, так і у якості периферійного пристрою для будь-якого іншого мікроконтролеру. ESP32 може комунікувати з іншими Wi-Fi або Bluetooth пристроями за допомогою SPI/SDIO або I2C/UART інтерфейсів [9].Функціональна діаграма ESP32 наведена на рис. 1.5.

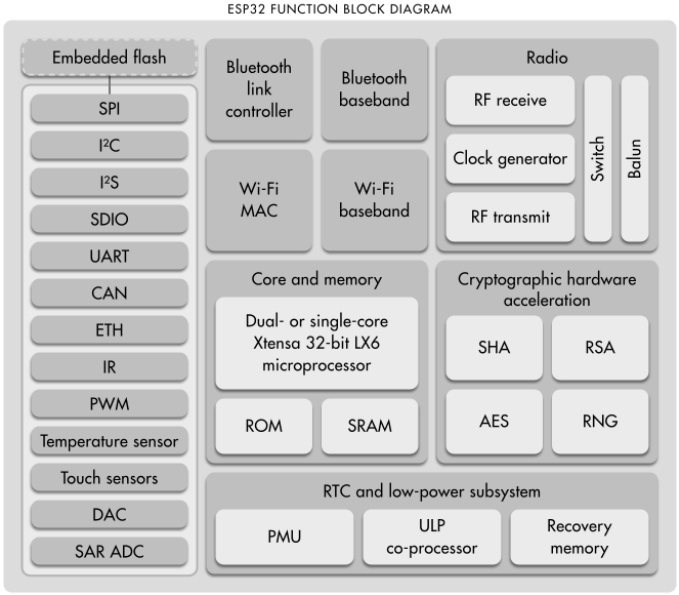


Рисунок 1.5 – Функціональна діаграма ESP32

З метою розробки пристрою для отримання медичних показників до мікроконтролеру ESP32 можна під'єднати сенсори визначення частоти серцевих скорочень та кров'яної сатурації (наприклад, MAX30102), сенсори моніторингу руху (10-DOF IMU Sensor) [7] та інші.

Модуль MAX30102 [10], [11] – інтегральний датчик пульсу і насичення крові киснем. Він включає оптимізовану оптику, два світлодіоди, фотодетектор, високоточний аналоговий підсилювач і перетворювач, цифровий обробник і інтерфейсний модуль. Для підключення датчика до контролера використовується послідовний інтерфейс I2C. На рис. 1.6 показано зовнішній вигляд датчику MAX30102.

Рисунок 1.6 – Сенсор MAX30102

У корпусі MAX30102 реалізована повнофункціональна схема сенсорного модуля для створення портативних систем пульсоксиметрії с високими вимогами до точності вимірювань. Пристрій має мініатюрні розміри, домогтися яких вдалося без шкоди для оптичних або електричних характеристик. Для інтеграції в повнофункціональну переносну вимірювальну систему потрібно мінімум додаткових зовнішніх компонентів.

Управління роботою MAX30102 здійснюється через внутрішні програмні регістри. Цифрові вихідні дані можуть бути збережені в 32-бітному буфері FIFO, який дозволяє через загальну шину послідовно передавати цифровий потік на зовнішній контролер.

MAX30102 працює від одного джерела живлення 1,8 В та окремого джерела живлення 3,3 В для внутрішніх світлодіодів. Зв'язок з мікроконтролерами здійснюється через стандартний I2C-сумісний інтерфейс. Принципова схема сенсору наведена на рис. 1.7.

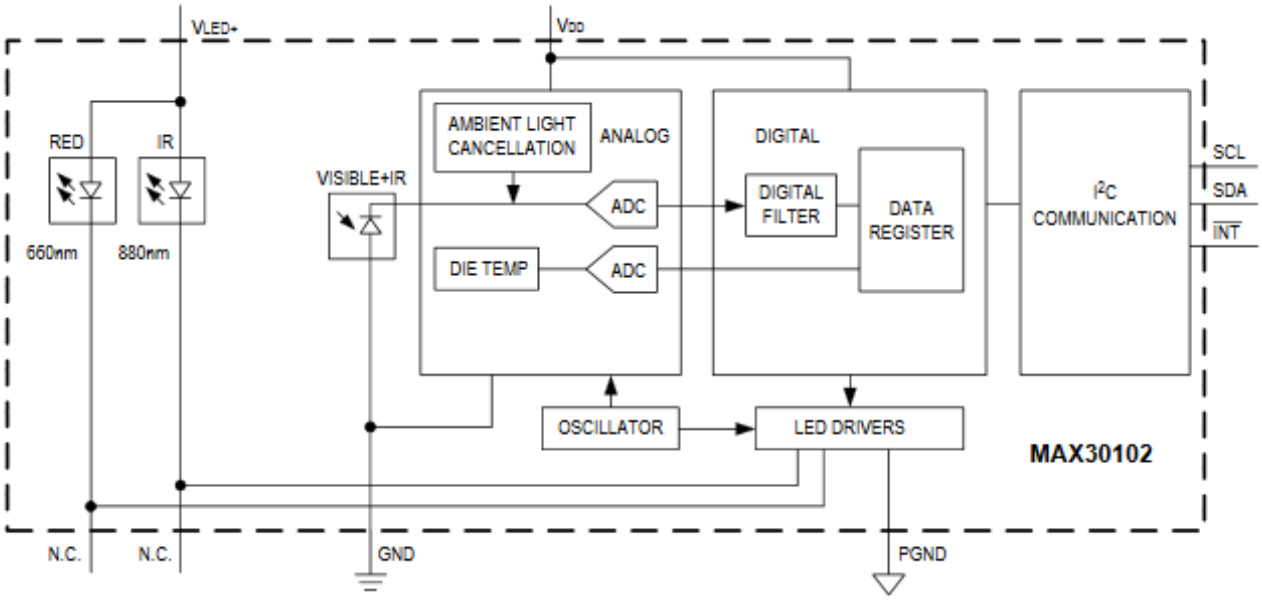


Рисунок 1.7 – Принципова схема сенсору MAX30102

Основними характеристиками даного модулю є:

– малий розмір (5.6мм x 3.3мм x 1.55мм 14-піновий оптичний модуль)

– вбудоване покривне скло для оптимальної та надійної роботи

– напруга живлення 3,3V (внутрішній стабілізатор на 1,8V)

– струм у режимі вимірювання 1.2mA

– струм у режимі сну до 10μA

– інтерфейс підключення I2C

– максимальна частота інтерфейсу 400kHz

У порівнянні з попередньою версією модуля, а саме з MAX30100[12], було підвищено точність вимірювання пульсу та оксигенації без зміни форми та розташування пінів даного модулю. Це дозволяє використовувати схеми, зібрані під модуль MAX30100, до MAX30102 не вносячи ніяких змін. Схема розпіновки модулю наведена на рис. 1.8. У таблиці 1 наведено опис пінів цього сенсору.

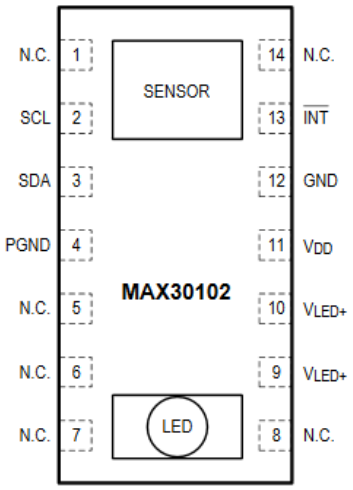


Рисунок 1.8 – Розташування пінів модулю MAX30102

Таблиця 1 – Опис пінів сенсору MAX30102

Pin	Назва	Функціонал піну
1, 7, 8, 14	N.C.	Немає з'єднання, використовується для стабільного під'єднання модулю до плати
2	SCL	I2C Clock Input
3	SDA	I2C Clock data, двустороння, відкритий сток
4	PGND	Заземлення для блоку LED драйверів
5	IR_DRV	Інфрачервоний LED катод та точка підключення LED драйверу.
6	R_DRV	Червоний LED катод та точка підключення LED драйверу.
9	R_LED+	Живлення для червоного LED (Анодне підключення)
10	IR_LED+	Живлення для інфрачервоного LED (Анодне підключення)
11	VDD	Аналогове живлення
12	GDN	Аналогове заземлення
13	INT	Пін переривання (відкритий сток)

Підсистема SpO2 в сенсорі MAX30102 містить систему приглушення навколишнього світла (ALC), сигма-дельта-АЦП безперервного часу та власний дискретний часовий фільтр [13]. ALC має внутрішню схему відстеження та утримання для скасування навколишнього світла та збільшення ефективного динамічного діапазону. АЦП SpO2 має програмовані повномасштабні діапазони від 2μA до 16μA. ALC може скасувати до 200 мкА струму навколишнього середовища. Внутрішній АЦП – це перетворювач сигма-дельта безперервного перевитрати часу з 18-

бітовою роздільною здатністю. Частота дискретизації АЦП становить 10,24 МГц. Швидкість вихідних даних АЦП може бути запрограмована від 50 в/с (вибірок в секунду) до 3200 в/с.

MAX30102 має вбудований датчик температури для калібрування температурної залежності підсистеми SpO2 [13]. Температурний датчик має роздільну здатність 0,0625 °C. Вихідні дані пристрою відносно не чутливі до довжини хвилі ІЧ-світлодіода, де довжина хвилі червоного світлодіода є критичною для правильної інтерпретації даних. Алгоритм SpO2, що використовується з вихідним сигналом MAX30102, може компенсувати відповідну помилку SpO2 із зміною температури навколишнього середовища.

Сенсор має інтегровані червоні та інфрачервоні світлодіодні драйвери для модуляції світлодіодних імпульсів, яке потрібне для вимірювання SpO2 та серцевого ритму. Подачу струму на світлодіоди можна запрограмувати в діапазоні від 0 до 50 мА при достатній напрузі живлення. Ширину світлодіодного імпульсу можна запрограмувати в діапазоні від 69 мкс до 411 мкс, щоб алгоритм оптимізував точність SpO2 та серцевого ритму та споживання енергії на основі випадків використання. На рис 1.9 зображено принцип роботи сенсору MAX30102, який працює за принципом фотоплетізмографії [14].

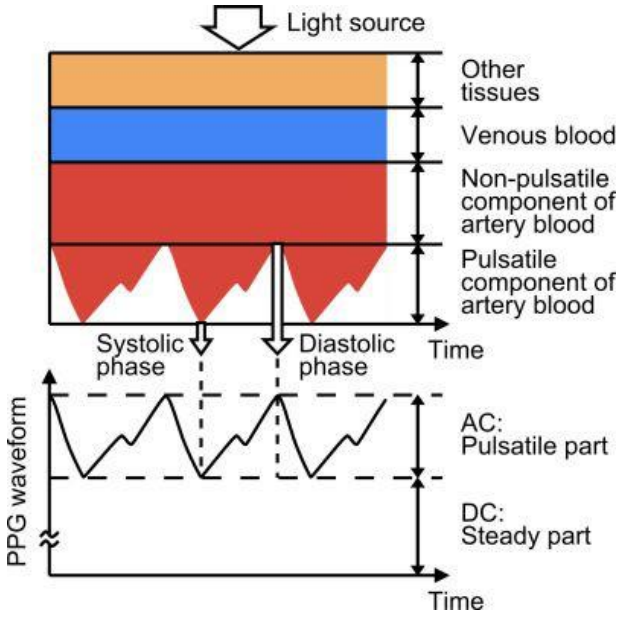


Рисунок 1.9 – Принцип роботи сенсору MAX30102

Узагальнену структуру пристрою та всього комплексу наведено на рис. 1.10. Пристрій на базі ESP32 первинно оброблює отримані дані з сенсорів пульсоксиметрії, температури та руху. Далі вже первинно оброблені дані до IoT серверу передаються або за допомогою прямого Wi-Fi підключення, або через мобільний пристрій на базі Android або iOS за допомогою прив’язки через Bluetooth. Після цього IoT сервер оброблює отримані дані та надсилає їх на мобільний або веб-застосунок, що виводить потрібну користувачеві інформацію у вигляді графіків, дата-сетів, повідомлень щодо відхилень певних даних від норми і т. д. Подібна система описана у [15].

Також на дисплей самого пристрою виводяться усереднені дані щодо частоти серцевих скорочень та рівня кров’яної сатурації за певний період часу. При реалізації цього пристрою, усереднені дані будуть виводитися на LCD дисплей за 5-хвилинний період.

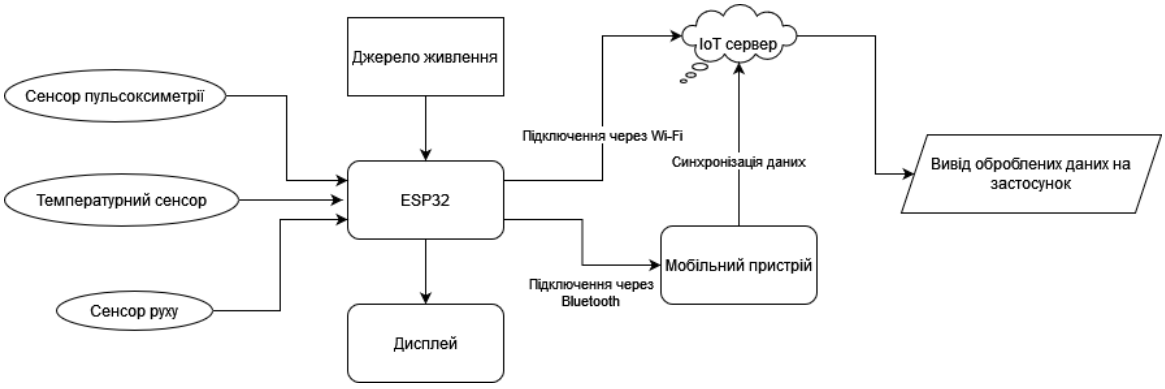


Рисунок 1.10 – Узагальнена схема IoT пристрою на базі мікроконтролеру ESP32

Виходячи з наведеного концепту можна зробити висновок, що на базі мікроконтролеру ESP32 можна побудувати IoT пристрій для збору медичних показників людини, який буде попередньо оброблювати отримані дані та відправляти їх на хмарний сервер, який буде проводити подальшу обробку та візуалізацію отриманих даних.

1.4. Апробація результатів дослідження

Дослідження апробовано в наступних роботах [2], [3], [16]Кожна з цих робіт містить частину, що пов’язана з певним етапом розробки програмно-апаратного комплексу моніторингу та аналізу фізіологічних показників людини.

В роботі [2] досліджено нову модель взаємодії в IoT-інфраструктурі для набору пристроїв. Для продукту кінцевого користувача, що складається з групи пристроїв, розглянуто проблеми інтеграції в бездротову мережу на основі WiFi-технології, ідентифікації програмного забезпечення, обміну даними та оновлення програмного забезпечення. Встановлене рішення використовує призначення ролей сервера та клієнта пристроям у наборі. Сервер розгортає тимчасову мережу, щоб пристрій користувача міг отримати до неї доступ і ввести необхідну інформацію про мережу для підключення. Також були розглянуті механізми зв’язку в мережі клієнта для набору пристроїв. Використовуючи запропоновані моделі, клієнт може керувати набором пристроїв, надсилати та отримувати дані (розглядається окремий випадок для аудіоданих), оновлювати прошивку та ресурси пристроїв безпосередньо зі свого пристрою за допомогою спеціально розробленого програмного забезпечення. В роботі було використано набір плат ESP8266 в форм-факторі NodeMCU для реалізації та тестування розробленої моделі. Розроблений прототип системи здатний отримувати конфігурацію мережі користувача, отримувати дані з клієнтського пристрою та відображати їх. Цю

систему підтвердження концепції можна додатково розширити для нових функцій.

В роботі [16]представлена система моніторингу фізичної активності. Система, що розробляється, складається з апаратної частини (розумний годинник і смартфон) і програмної частини (набір бібліотек і розроблений додаток). Цільовою платформою для запуску програмного забезпечення є смартфони Apple, які отримують інформацію про фізичну активність користувача з годинника. Розроблений додаток у своїй роботі спирається на набір бібліотек, які були відібрані для розробки. Програма для моніторингу - це гібридна програма, яка запускається з вікна браузера. У статті подано повний опис архітектури програмного забезпечення з усіма використовуваними компонентами та взаємозв’язками між ними. Основною функцією розробленого програмного забезпечення є моніторинг фізичної активності користувача, що дозволяє збирати дані про тренування, планувати та підсумовувати агреговані їх результати. Також надається детальна інформація про кожне тренування. Додаток було протестовано з трьома різними типами тренувань (ходьба, біг і їзда на велосипеді) і показало, що його можна використовувати для моніторингу тренувального процесу. Його також можна розширити, щоб надати графічну інформацію про фізичну активність.

В роботі [3] наведено концепт IoT пристрою на базі ESP32 для збору та обробки медичних даних людини. Показано основні характеристики контролеру ESP32, що дозволяють його використання в IoT системах подібного типу а також наведено алгоритм роботи складових пристрою.

Висновки до розділу 1

В цьому розділі опановано та проаналізовано матеріал щодо різних систем, технологій та методів роботи систем моніторингу та аналізу фізіологічних показників стану людини. Досліджено найбільш поширені пристрої з подібним функціоналом, наукову та патентну інформацію для

подальшої розробки програмно-апаратного комплексу моніторингу та аналізу фізіологічних показників стану людини.

Визначено, що для мінімального функціоналу подібного роду пристроїв, вони мають складатися з наступних компонентів: контролер управління, сенсор фотоплетизмографії, модуль безпровідної передачі даних. Обрано ESP32 в якості контролера управління, первинної обробки та фільтрації а також в якості модуля безпровідної передачі даних. Досліджено його функціонал, можливості та структуру. Також в якості сенсору фотоплетизмографії обрано MAX30102, завдяки якому можна отримати дані щодо серцевого ритму та капілярної сатурації крові.

Також розроблено узагальнену структурну схему IoT системи на базі ESP32 для відслідковування фізіологічних показників людини.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

2.1 Проектування програмно-апаратного комплексу моніторингу та аналізу фізіологічних показників стану людини на базі модулю ESP32

Як було зазначено у попередньому розділі, узагальнена структура комплексу складається з трьох головних компонент. Це – пристрій збору даних, IoT сервер та пристрій виводу (рис. 1.10). Пристрій збору складається з контролеру управління ESP32, який одночасно і працює в якості контролеру передачі даних до IoT серверу, сенсору фотоплетизмографії, що вимірює пульс, капілярну оксигенацію крові та температуру тіла, та акселерометру, призначеного для вимірювання кількості кроків. IoT сервер та і загалом серверна частина складається з програмних модулів для отримання «сирих» даних за допомогою прийому HTTP пакетів, що попередньо зашифровані, модулів обробки та фільтрації даних, та їх візуалізації.

Блок-схема підключення модулів пристрою наведена на рис. 2.1.

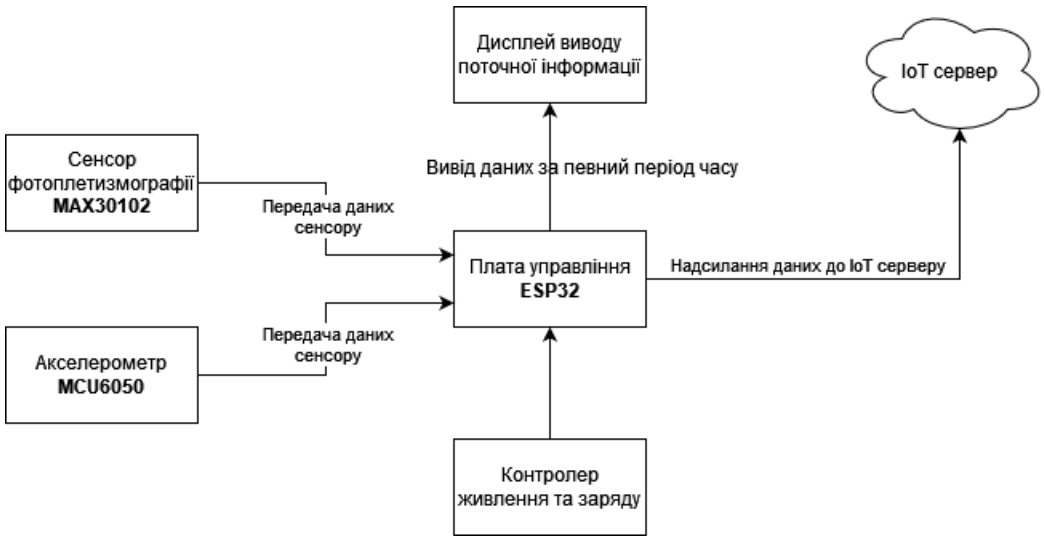


Рисунок 2.1 – Блок-схема підключення модулів пристрою збору фізіологічних показників стану людини

Завдяки тому, що ESP32 має декілька ядер, є можливість реалізації багатопотоковості програми на цьому контролері. А це значить, що на збір та

обробку даних з модулю фотоплетизмографії та акселерометру не потрібно додатково застосовувати ще один модуль ESP32.

2.2 Алгоритм отримання даних серцевого ритму та оксигенації крові

Для вимірювання серцевого ритму використовується модуль MAX30102, що за допомогою методу фотоплетизмографії дозволяє отримувати дані пульсу та рівня капілярної кисневої сатурації (SpO2). Структура сенсору MAX30102 наведено на рис. 2.2.

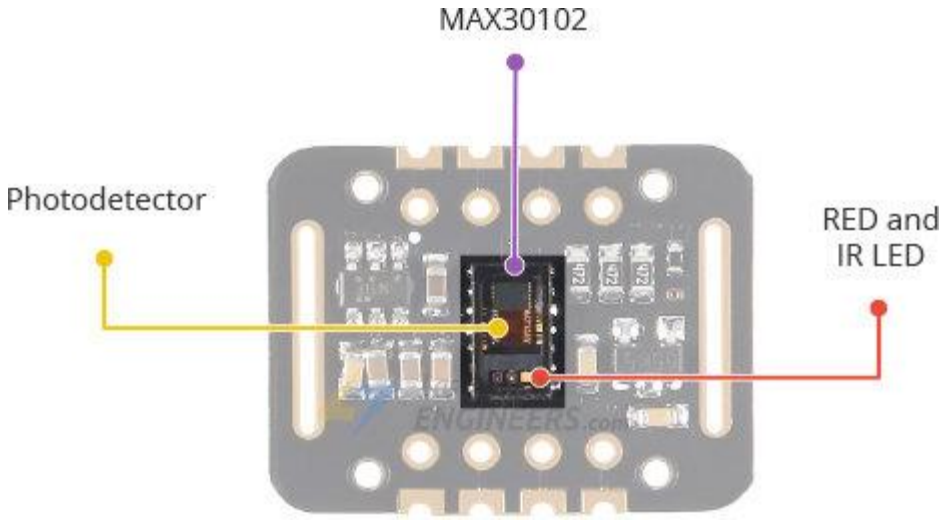


Рисунок 2.2 – Структура сенсору MAX30102

Хоча в крові є різні типи сполук гемоглобіну, для розрахунків SpO2 передбачається, що оксигенований гемоглобін та дезоксигенований гемоглобін є єдиними важливими факторами. У режимі відбивної пульсоксиметрії світлодіоди висвітлюють тканини шкіри, відбитий сигнал якого отримується фотодіодом. Даний відбитий сигнал містить світло, що оптично модульоване об'ємними змінами артерій та капілярів. Цей сигнал фотоплетизмографії (PPG) надзвичайно важливий для визначення частоти серцевих скорочень та рівня SpO2. Сигнали PPG мають постійні та змінні компоненти, що показано на рис. 2.3.

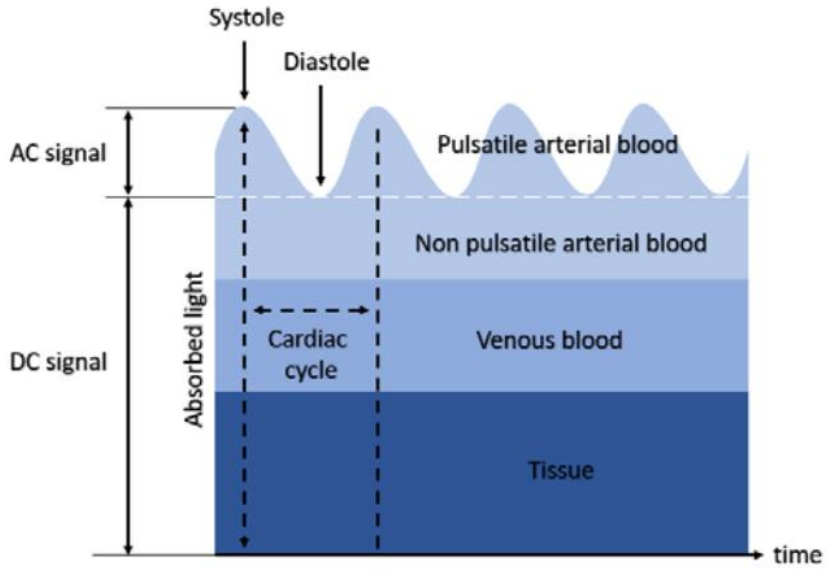


Рисунок 2.3 – Постійні та змінні компоненти сигналу фотоплетізмографії

Постійна компонента пояснюється поглинанням світла неімпульсної тканини: венозної капілярної та артеріальної крові. З іншого боку, змінна компонента зумовлена пульсуючим характером артеріальної крові. Оскільки артерії мають безпосередній зв'язок із серцем, артеріальна кров пульсує, коли пульсує серце. Серцевий ритм можна обчислити, вимірявши час між послідовними систолічними піками. Тут важливо відзначити, що частоту серцевих скорочень можна виміряти, використовуючи лише один світлодіод, наприклад, червоний, оскільки компонент змінного струму є єдиним необхідним сигналом.

Для розрахування значення SpO_2 використовуються два світлодіоди з різною довжиною хвилі для визначення співвідношення оксигенованого гемоглобіну до дезоксигенованого гемоглобіну. Червоний та ІЧ-світлодіоди використовуються для визначення окремих сигналів PPG. Оскільки постійні та змінні компоненти двох світлодіодів мають різну амплітуду, їх потрібно нормалізувати для корисного порівняння. Для цього порівняння визначається відношення «R», яке прямо пропорційне SpO2. Перш ніж вводити рівняння для R та SpO2, слід зазначити, що SpO2 наближається до сатурації артеріального кисню (SaO2). Наступне рівняння для R відоме як «відношення коефіцієнтів:»

$$R = \frac{\frac{AC_{red}}{DC_{red}}}{\frac{AC_{infrared}}{DC_{infrared}}} \quad (2.1)$$

Після того, як коефіцієнт R визначено, для визначення оцінки SpO2 можна використовувати криволінійне наближення або пошукову таблицю. Ці дані зазвичай збираються емпіричними методами, де використовуються численні предмети. Вік, тон шкіри, загальний стан здоров'я та медичні умови можуть впливати на точність вимірювання SpO2. Нижче наведено приклад лінійного наближення, отриманий із найкращого прямолінійного наближення даних SpO2 проти R між діапазоном R 0,4-3,4:

$$SpO_2 = 104 - 17R \quad (2.2)$$

2.3 Алгоритм отримання даних температури тіла

Окрім можливості отримання значень серцевого ритму та пульсу, сенсор MAX30102 містить температурний сенсор, який дозволяє вимірювати температуру тіла. Цей сенсор є досить точний та може вимірювати температуру тіла в діапазоні від -40˚C до +85˚C з точністю ±1˚C.

Сам алгоритм отримання значень температури є наступним:

1. Конфігуруємо регістр температури для отримання одного семплу;
2. У випадку, якщо переривач температурного регістру налаштований, то процес зчитування даних температури завершено;
3. Зчитування значень температурного регістру
4. Розрахунок значення температури

Цей алгоритм реалізовано в бібліотеці MAX30105.h від Sparkfun, і тому він є достатньо легким для використання з обраним сенсором.

2.4 Алгоритм отримання кількості кроків за допомогою сенсору MPU-6050

Сенсор MPU-6050 є поєднанням 3-осьового гіроскопу та 3-осьового акселерометру на одному кремнієвому кристалі разом із вбудованим цифровим процесором руху, який обробляє складні 6-осьові алгоритми MotionFusion [17]–[20]. Пристрій може отримати доступ до зовнішніх магнітометрів або інших датчиків через допоміжну головну шину I2C, що дозволяє пристроям збирати повний набір даних сенсору без втручання системного процесора.

В структурі пристрою збору фізіологічних показників стану людини, цей сенсор використовується для визначення кількості кроків людини. Ці дані можна отримати за допомогою датчика гіроскопа, використовуючи граничне значення порівняння (поріг) на осі даних X. Коли його підключено, сенсор гіроскопу зчитує дані X, Y і Z, потім беруться дані з усіх трьох осей. Один кут — це вісь X. Дані X порівнюються за допомогою верхнього граничного значення та нижнього граничного значення для виявлення кроків.

Дані, які зчитуються для підрахунку кількості кроків, мають форму значень кутів X, Y і Z. Значення кута перетворюється на графік синусоїди, щоб можна було легко визначити порогові верхні та нижні значення кута, який формує значення хвилі. У цій системі треба взяти значення одного з кутів, який використовується для виявлення кроків і обчислення відстані, тобто осі X. Під час руху вісь X зчитує дані з руху руки, якщо дані гіроскопа менші за -20 градусів, і це слугує маркером того, що дані досягли найнижчої точки і потім датчик знову зчитує дані гіроскопа. Якщо дані гіроскопа вже вище 20° і дані маркера вже зчитувались з найнижчої точки, до кількості кроків додається ще один.

2.5. Алгоритм роботи апаратної частини комплексу

Алгоритм роботи апаратної частини комплексу моніторингу та аналізу фізіологічних показників стану людини в спрощеному вигляді виглядає наступним чином (рис. 2.4):

1. Ініціалізація модулю.
2. Перевірка підключення та роботи модулю MAX30102.
3. У випадку, якщо плату не знайдено – то передається повідомлення про відсутність плати MAX30102 і пристрій переходить в режим очікування до наступної перевірки.
4. Перевірка підключення та роботи модулю MPU6050
5. У випадку, якщо модуль не знайдено – відбувається передача повідомлення про відсутність плати MPU6050 та перехід в режим очікування до наступної перевірки
6. Перевірка підключення до WiFi мережі та встановлення з’єднання з віддаленим IoT сервером
7. У випадку відсутності підключення, пристрій переходить в режим очікування до наступної спроби встановлення з’єднання зі сервером
8. Отримання «сирих» даних з сенсору.
9. Первинна обробка отриманих даних
10. Шифрування даних
11. Відправка даних до серверу.

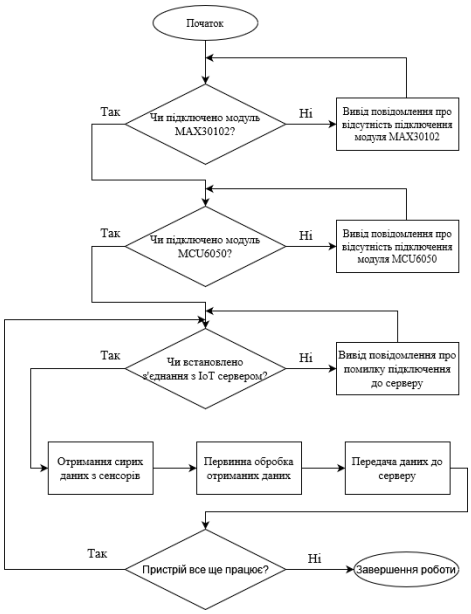


Рисунок. 2.4 – Блок-схема роботи програми на мікроконтролері

Розглянемо більш детально цей алгоритм. Ініціалізація пристрою проходить після затискання кнопки пристрою або по таймеру. Сам процес ініціалізації відбувається за такою схемою, яку наведено на рис. 2.5.

Після запуску відбувається перевірка наявності даних налаштувань в постійній енергонезалежній пам’яті. У випадку, якщо налаштувань немає – то відбувається завантаження стандартних налаштувань роботи пристрою. Далі відбувається перевірка підключень компонентів до контролеру та наявності збережених Bluetooth та WiFi мереж. Після чого відбувається запуск безпровідних інтерфейсів та встановлення підключень. У випадку неможливості встановлення з’єднання з Wi-Fi мережею, пристрій переходить до режиму очікування та через певний період часу намагається перепідключитися за допомогою Bluetooth або Wi-Fi з’єднання.

Після блоку ініціалізації пристрою відбувається перевірка наявності або здатності до роботи його основних елементів, а саме – сенсору MAX30102 та MPU6050. У випадку, якщо один з сенсорів чи обидва відсутні або відключені, то пристрій повторно проходить процес ініціалізації або відключається в залежності від налашт увань.

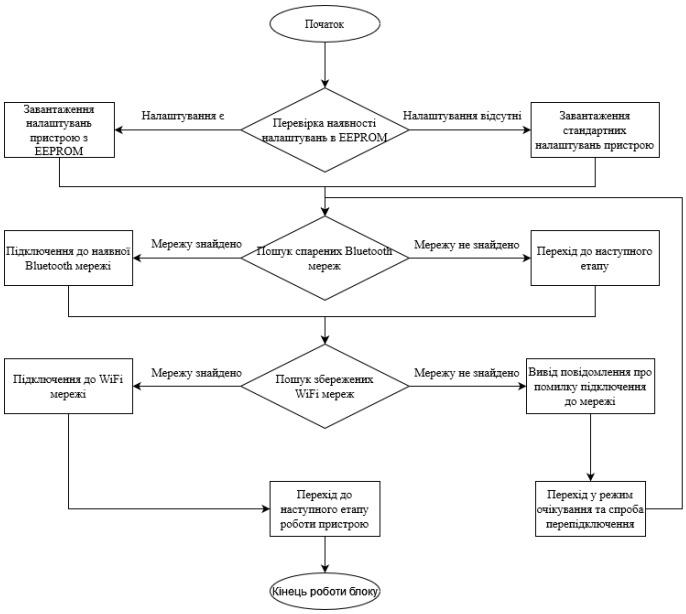


Рисунок 2.5 – Блок-схема процесу ініціалізації пристрою

Наступним етапом роботи програми є отримання «сирих» даних з сенсорів та їх первинна обробка (рис. 2.6). В цьому випадку розуміється отримання значень фотоплетизмографічних сенсорів, сенсору температури, гіроскопу та акселерометру.

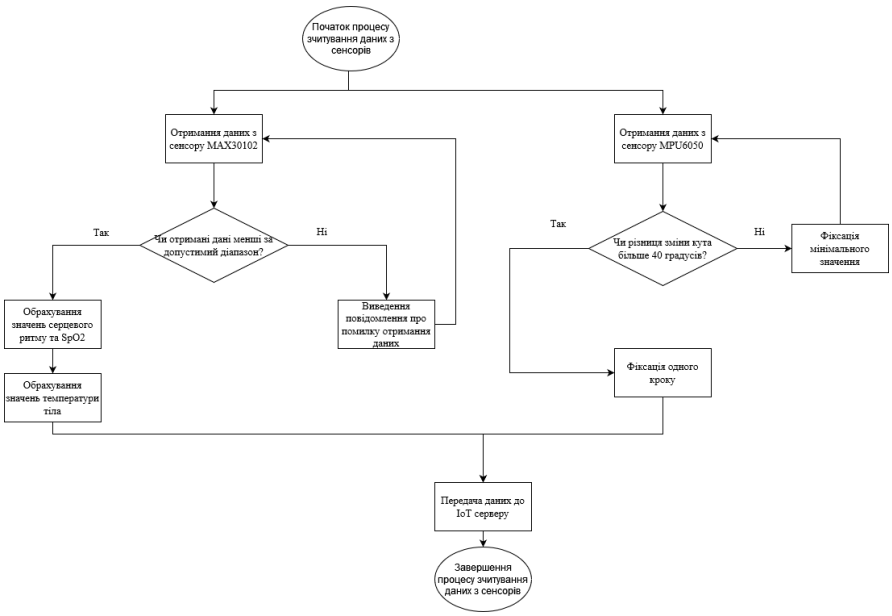


Рисунок 2.6 – Блок-схема процесу обробки «сирих» даних з сенсорів

Після отримання даних з сенсорів, вони перевіряються на входження до потрібного діапазону даних. У випадку, якщо значення масиву даних температури, серцевого ритму та/або рівня кров’яної сатурації є нижчим за порогове – користувач отримує повідомлення про помилку в отриманні

даних. У випадку підрахунку кроків вираховується значення кута по осі X. Після чого відбувається вивід поточних даних на екран пристрою та передача попередньо оброблених та зашифрованих даних до IoT серверу за допомогою HTTP протоколу.

2.6. Запис даних до EEPROM

EEPROM – це постійний запам’ятовувач, що програмується та очищується за допомогою електрики і є одним з видів енергонезалежної пам’яті.

Принцип роботи EEPROM базується на зміні та реєстрації електричного сигналу в ізольованій області (кишені) напівпровідникової структури.

Зміна заряду («запис» та «стирання») виконується поданням між затвором і витоком великого потенціалу, щоб напруженість електричного поля в тонкому діелектрику між каналом транзистора і кишенею виявилася достатньою для виникнення тунельного ефекту. Для посилення ефекту тунелювання електронів у кишеню при записі застосовується невелике прискорення електронів шляхом пропускання струму через канал польового транзистора. Читання виконується польовим транзистором, для якого кишеня виконує роль затвора. Потенціал плавного затвору змінює порогові характеристики транзистора, що і реєструється ланцюгами читання.

Основна особливість класичного осередку EEPROM – наявність другого транзистора, який допомагає керувати режимами запису і стирання. Деякі реалізації виконувалися у вигляді одного тризатворного польового транзистора (один затвор плавний і два звичайних). Ця конструкція забезпечується елементами, які дозволяють їй працювати у великому масиві таких же осередків. З'єднання виконується у вигляді двовимірної матриці, в якій на перетині стовпців і рядків розташовується одна клітинка. Оскільки осередок EEPROM має третій затвор, то крім підкладки до кожної клітинки підходять 3 провідники (один провідник стовпців і 2 провідники рядків).

Існує два види обмежень інформації, що зберігається – це кількість циклів перезапису та тривалість зберігання. Під час перезапису, підзатворний окислений шар плавучого затвору транзисторів поступово накопичує в собі електрони. Електричне поле нерухомих електронів додається до поля електронів в плавучому утворенні, спускаючи вікно між пороговими напруженнями. Після досягнення достатньої кількості циклів перезаписів, різниця буде занадто малою, щоб бути помітною. Тому подальший запис інформації на пристрій перестає бути можливий та відбувається відмова цього обладнання. Виробники зазвичай використовують максимальну кількість перезаписів в 1 млн. циклів перезапису.

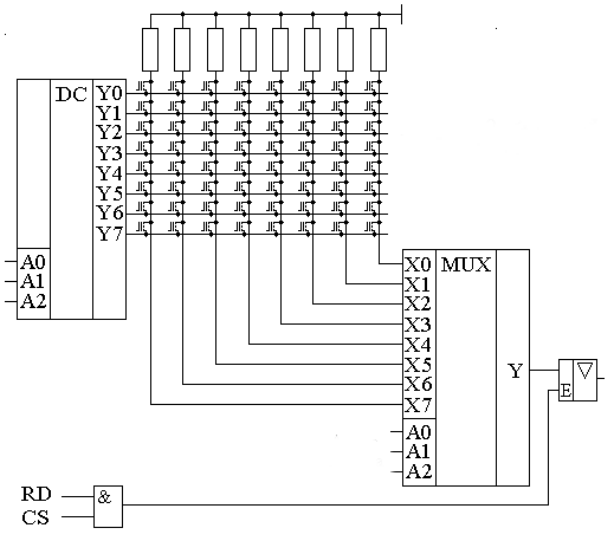


Рисунок 2.7 – Схема EEPROM

Другий вид обмежень – тривалість зберігання обумовлена тим, що під час зберігання електрони, що знаходяться в плаваючому затворі, можуть пройти крізь ізолятор, особливо за підвищеної температури та викликати втрату заряду, повертаючи затвор у запертий стан. Виробники зазвичай гарантують термін збереження даних до 10 років і більше.

Враховуючи те, що на мікроконтролерній платі EEPROM має дуже незначний та обмежений обсяг пам’яті, пристрій не має можливості зберігати абсолютно всі зібрані дані в собі. Структура даних в EEPROM наведено на рис. 2.8.

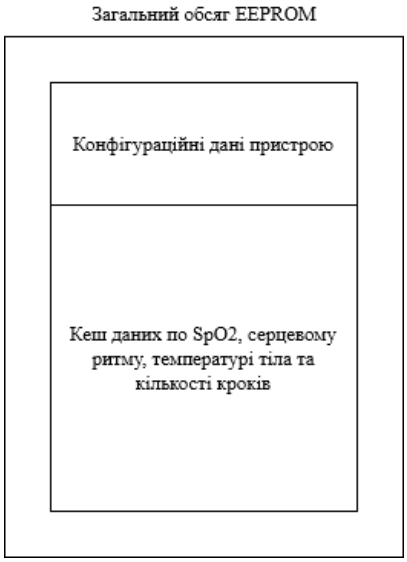


Рисунок 2.8 – Структура даних всередині EEPROM

Дані в EEPROM розподіляються на дві частини – на область даних з конфігурацією пристрою та область зібраних даних з користувача. Конфігураційні дані містять в собі дані по підключенню до зовнішніх клієнтських пристроїв за допомогою Wi-Fi та/або Bluetooth 2.0, періодичність вимірювань, режим роботи мережевих інтерфейсів та опцію увімкнення або вимкнення енергозбереження пристрою. У випадку, якщо блок пам’яті, що містить конфігурацію є порожнім або дані не зчитуються – то застосовується стандартна конфігурація, що є частиною програмного забезпечення модуля. Блок-схеми роботи з блоком конфігурації наведено на рис. 2.9 та рис 2.10.

Рисунок 2.9 – Блок-схема алгоритму застосування збереженої конфігурації

Рисунок 2.10 – Блок-схема процесу запису нової конфігурації пристрою в секцію EEPROM

Процес запису оновлень конфігурації пристрою складається з декількох етапів. Першим є отримання нових даних налаштувань пристрою за допомогою підключення мережевого інтерфейсу до клієнтського пристрою або Wi-Fi мережі. Дані отримуються та конвертуються в формат, оптимальний для збереження даних в EEPROM. Наступним кроком є перевірка доступу до блоку енергонезалежної пам’яті і якщо доступу немає – то пристрій виводить повідомлення про помилку доступу. У випадку, якщо доступ до модуля пам’яті є – відбувається запис та перезавантаження пристрою.

2.7. Алгоритм роботи серверної частини комплексу

Узагальнено, серверна частина комплексу моніторингу та аналізу фізіологічних показників стану людини поділяється на дві основні частини:

- Блок прийому, обробки та збереження даних користувача
- Блок передачі та візуалізації оброблених даних
- Блок прийому, обробки та збереження даних користувача працює наступним чином:
- Отримання шифрованих HTTP-пакетів
- Дешифрування отриманих даних

- Фільтрація значень-«викидів» отриманих даних та їх обробка
- Запис відфільтрованих значень до бази даних

Блок передачі та візуалізації оброблених даних може працювати в декількох режимах – або як самостійний веб-застосунок, що генерує графіки та показує статистику даних користувача за певний період часу, або як API, що повертає JSON відповідь з певним набором даних до мобільного чи десктопного застосунку, для подальшої роботи з даними.

2.8 Алгоритми обробки та представлення зібраних даних

2.8.1 Обробка отриманих даних з використанням вейвлет-перетворень

Застосування методів цифрової обробки сигналів і зокрема, методів цифрової фільтрації, є поширеним і використовується у багатьох важливих галузях досліджень, як-от: обробка мовних сигналів, цифрова телефонія і цифровий зв'язок, обробка фототелеграфних і телевізійних зображень, радіо- та гідро-локаційні системи, космічні дослідницькі та розвідувальні системи, геологічна розвідка і таке інше. Біологія і медицина не є виключеннями, цифрова фільтрація сигналів – один з сучасних способів їх обробки.

Основні переваги цифрової обробки сигналів полягають в:

1. Можливості реалізації складних методів та алгоритмів обробки сигналів та зображень, недоступних для аналогових пристроїв;
2. забезпеченні високої точності обробки;
3. гнучкості і універсальності засобів, розвиненому користувацькому інтерфейсові тощо.

З другої половини минулого сторіччя виник і з успіхом розвивається новий та важливий напрямок теорії та технології обробки сигналів, зображень та часових послідовностей, який отримав назву вейвлет-перетворень. Вони напрочуд добре пристосовані для вивчення структури неоднорідних, нестаціонарних процесів, тобто таких, якими є більшість медичних сигналів.

Більшість традиційних медичних діагностичних сигналів, як-от: електрокардіограми, електроенцефалограми, плетизмограми, пульсограми тощо, по-перше одновимірні, по-друге, суттєво нестаціонарні, і найчастіше цифрові, послідовності, задані у часовому представленні. Вейвлети – це потужний сучасний інструментарій, який використовують для обробки таких сигналів та стискання отриманих даних [21]–[24].

Вейвлети являють собою особливі функції у вигляді коротких хвиль (сплесків) з нульовим інтегральним значенням по всій осі незалежної змінної (найчастіше координати, або часу). Ці функції припускають зсув по осі змінної та масштабування (розтягання або стискання). За рахунок зміни масштабів вейвлети здатні виявляти відмінності в характеристиках процесу на різних шкалах, а за рахунок зсувів по осі змінної можна аналізувати властивості процесу в різних точках інтервалу, змінної, який досліджується. Будь-який з відомих вейвлетів породжує повну систему ортогональних функцій, в базисі яких можна розкладати будь-яку функцію. Повнота такої системи гарантує її відновлення (реконструкцію, синтез) шляхом зворотного вейвлет-перетворення після вейвлет-аналізу (декомпозиції сигналу в ряд по вейвлетах).

Вейвлети – це перенесені в часі (b) та масштабовані (a) копії, які також називають «дочірніми вейвлетами»

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}} \Psi(\frac{t-b}{a}) \tag{2.3}$$

деякої швидко змінної функції – так званого «материнського вейвлета»: $\Psi(t)$

.

2.8.2 Генерація графіків представлення даних

Оброблені дані представляються користувачу у вигляді графіку, де віссю X є дати замірів, а віссю Y є рівень серцевого ритму або кров'яної оксигенації. Представлення даних відбувається наступним чином: сервер, отримавши запит на відображення графіки у веб-застосунку, отримує зібрані

дані по одному пристрою, обирає їх кількість (наприклад – останні 30, 60, 90 або всі можливі записи) та виводить їх на графік.

2.9. RSA–шифрування

RSA–алгоритм – це алгоритм підпису з використанням відкритого ключа, розроблений Роном Рівестом, Аді Шаміром і Леонардом Адлеманом в 1977 році [25]–[27]. Цей асиметричний алгоритм використовує логарифмічні функції для захисту від «вичерпного пошуку» та є достатньо простим для реалізації та розгортання. На рис. 2.11 показана перевірка цифрових підписів за допомогою RSA.

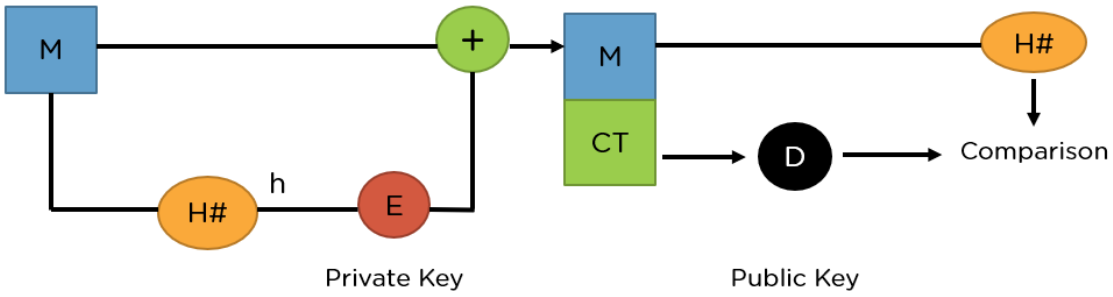


Рисунок 2.11 – Перевірка цифрового підпису за допомогою RSA

Коли RSA використовується для шифрування та дешифрування загальних даних, набір ключів шифрування протилежно змінюється. На відміну від перевірки підпису, для шифрування даних використовується відкритий ключ одержувача, а для дешифрування даних – закритий ключ одержувача. Таким чином, у цьому сценарії немає необхідності обмінюватися ключами.

RSA-криптографія містить два основних компоненти:

– Генерація ключів: генерація ключів, які будуть використовуватися для шифрування та дешифрування даних для обміну.

– Функція шифрування/дешифрування: кроки, які необхідно виконати під час шифрування та відновлення даних.

Ключ генерується наступним чином:

1. Обирається два великих простих числа p та q довжиною 512/1024/2048 біт кожне

2. Обчислюється їх добуток $n = pq$
3. Обчислення функції Ейлера $\varphi(n) = (p-1)(q-1)$
4. Обирається таке число e, що $1 < e < \varphi(n)$ та e є взаємно простим з $\varphi(n)$
5. Використовуючи розширений алгоритм Евкліда знаходиться число d таке, що e$d \equiv 1(mod\ \varphi(n))$

Згенерувавши ключі, вони передаються як параметри до функцій, які обчислюють зашифрований і відкритий текст за допомогою відповідного ключа.

Якщо відкритий текст є m, то тоді шифрований текст є $c = m \cdot e \cdot mod(n)$. І навпаки, якщо шифрований текст є c, то тоді $m = c \cdot d \cdot mod(n)$

2.10 Структура бази даних серверної частини

База даних має складатися з таблиць, що містять список зареєстрованих пристроїв, та даних, що зібрані з кожного апаратно-програмного модулю. У випадку, якщо реєструється новий пристрій – йому надається унікальний ідентифікатор, за яким йде відслідковування та запис до БД

База даних складається з чотирьох таблиць – DeviceList, UserList, Data, SettingsTable. Таблиця DeviceList складається з трьох полів, а саме:

– uuid – унікальний ідентифікатор пристрою;
– hardwareId – ідентифікатор пристрою;
– deviceDescription – опис пристрою.

Ця таблиця відповідає за збереження даних про всі наявні пристрої, що мають можливість підключення до серверу. Кожен пристрій має свій унікальний ідентифікатор, який присутній в EEPROM з моменту першого запуску модулю. Цей ідентифікатор разом з описом пристрою, що також знаходиться в EEPROM в блоці налаштувань, передається на сервер. Після чого сервер присвоює свій унікальний ідентифікатор (uuid) для подальшого зв'язування записів в базі даних, що відносяться до цього модулю дослідження та відслідковування фізичного стану людини.

Завданням таблиці UserList є зберігання даних про користувачів, їх паролі та пристрої. Таблиця складається з чотирьох полів:

– userID – унікальний ідентифікатор користувача;

– username – ім’я користувача;

– passwd – пароль користувача, зашифрований за допомогою алгоритму SHA-256;

– deviceID – унікальний ідентифікатор пристрою, воно пов’язане з полем uuid таблиці DeviceList.

Після реєстрації користувача відбувається прив’язка пристрою до його аккаунту. Максимальна кількість пристроїв, що може бути прив’язана до аккаунту – один, так як на цьому етапі розвитку проекту кількість готових пристроїв є обмеженою.

Таблиця Data вміщає в собі всі записи та виміри, зроблені пристроями та переданими на сервер. Вона складається з чотирьох полів:

– deviceID – унікальний ідентифікатор пристрою, воно пов’язане з полем uuid таблиці DeviceList;

– timestamp – час та дата заміру;

– hr – кількість серцевих скорочень за хвилину;

– spo2 – відсотковий рівень оксигенації крові;

– temp – температура тіла

– steps – кількість кроків на момент часу з початку доби (UTC час)

Отримані дані з модулю потрапляють на сервер, де проходять первинну фільтрацію від шумових даних. Після чого вони записуються до цієї таблиці з прив’язкою до пристрою.

Таблиця SettingsTable слугує місцем для збереження налаштувань модулів. Вона складається з таких полів:

– deviceID – унікальний ідентифікатор пристрою, воно пов’язане з полем uuid таблиці DeviceList;

– timestamp – час збереження налаштувань пристрою;

- ssid – збережений ідентифікатор Wi-Fi мережі
- password – зашифрований пароль за допомогою алгоритму SHA-256
- bluetoothStatus – стан Bluetooth підключення (True або False)
- frequency – частота регулярних замірів показників;

В таблицю SettingsTable записуються дані налаштувань кожного пристрою, збережені та передані на сервер за весь час роботи модулю відслідковування фізичного стану людини. Налаштування модулю відбувається як через мобільний застосунок, так і через веб-інтерфейс серверного застосунку. Стандартним значенням ssid та дешифрованим значенням поля password є SpO2Device та PasswordSpO2 відповідно, тому під час першого налаштування модулю рекомендовано створити Wi-Fi мережу з WP2-Personal шифруванням та наведеними даними.

Висновки до розділу 2

В цьому розділі наведено алгоритми визначення таких фізіологічних показників людини, як серцевий ритм, кров’яна сатурація, температура тіла та кількість кроків за певний період часу. Розроблено та представлено алгоритм роботи апаратно-програмного модулю для дослідження та відслідковування стану людини а саме: запис налаштувань до EEPROM, алгоритм збереження та очищення даних всередині енергонезалежної пам’яті, підключення та взаємодії модулю поміж ним та клієнтським та/або серверним обладнанням. Також модифіковано алгоритм обробки «сирих» даних на пристрої для збільшення точності обрахунків та створено алгоритм фільтрування отриманих даних та відсіювання «шумових» сигналів.

3 РЕАЛІЗАЦІЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

3.1 Компонентна база, що використовується в апаратній частині проекту

Апаратна частина комплексу моніторингу та аналізу фізіологічних показників стану людини складається з наступних компонентів:

- ESP32, який слугує центральним контролером та контролером обміну даних
- MAX30102, який є модулем збору даних серцевого ритму, кисневої сатурації капілярної крові та температури тіла
- MPU6050, акселерометр та гіроскоп для підрахунку кількості кроків
- TP4056, модуль керування живленням
- літій-полімерної батареї
- корпусу.

В якості контролера управління обрано ESP32-WROOM-32D[28]–[30] (рис. 3.1). Особливістю цього модуля є більш економічна робота в активному режимі та ультранизьке споживання в сплячому режимі та режимі очікування. Модуль має розведені користувацькі виходи GPIO, що ще більше розширює можливості побудови IoT-систем на його базі.

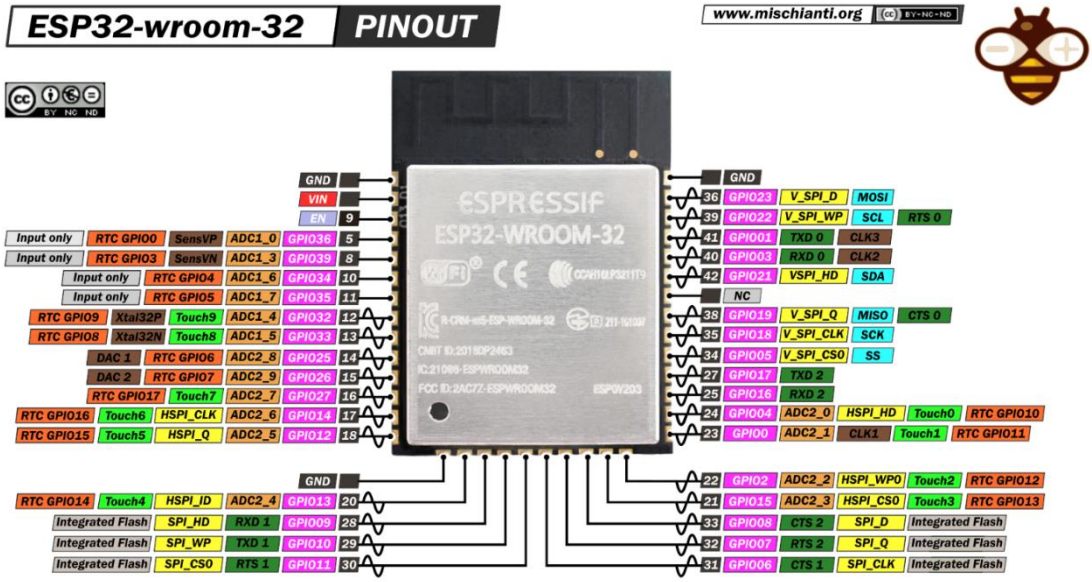


Рисунок 3.1 – ESP32-WROOM-32D pinout

Для пристрою також обрано літій-полімерний акумулятор формату 651620 (рис. 3.2) ємністю 100 мАг з контролером, номінальною напругою 3.7В (4.2В при повній зарядці).

Рисунок 3.2 – Літій-полімерний акумулятор UC 651620 3.7 V 100 mAh

Він містить в собі контролер заряду, що захищає його від повного розрядження або перезаряду. Напругою відсічки є 2,5 В – це значить, що після досягнення даної напруги на батареї контролер відключить навантаження. Розмір самої батареї є 16×20×6,5 мм, і він є оптимальним для прототипу пристрою.

Для зарядки акумулятора використовується мікросхема TP4056 (рис. 3.3), що дозволяє заряджати його за допомогою micro-USB кабелю. Ця мікросхема дозволяє заряджати літій-полімерні батареї до напруги 4,2 В струмом 1 А. Її розмір 27,75×17,25 мм, і тому цю мікросхему можна застосовувати в побудові прототипу пристрою.

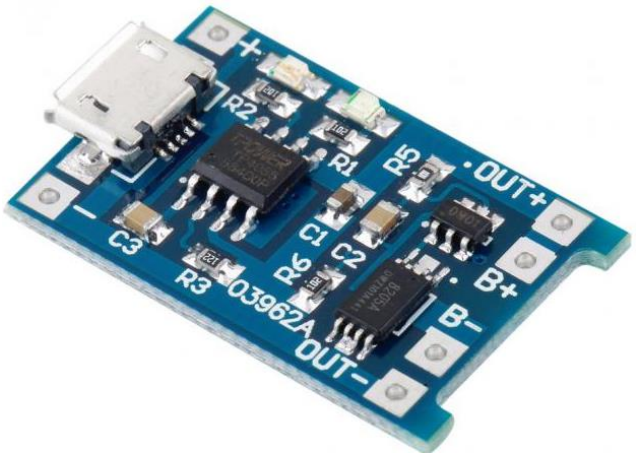


Рисунок 3.3 – Мікросхема TP4056

3.2. Розробка прототипу апаратної частини проекту

На основі наявних компонентів створено принципову та монтажну схему пристрою. Розробка принципової схеми проводилась в середовищі Fritzing (рис. 3.4). Fritzing – це застосунок з відкритим кодом для розробки аматорського та професійного забезпечення САПР для проектування електронного обладнання. Принципова схема наведена на рис. 3.5. Для кращого візуального розуміння підключення компонентів до контролера ESP32-WROOM-32D. Монтажна схема наведена на рис. 3.6.

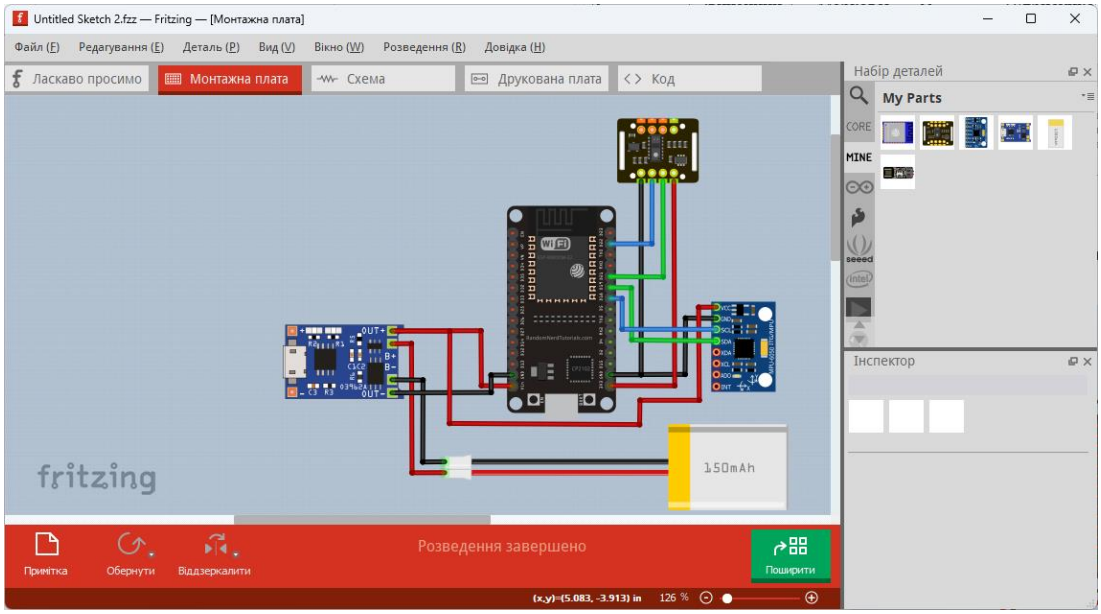


Рисунок 3.4 – Створення принципової схеми в середовищі Fritzing

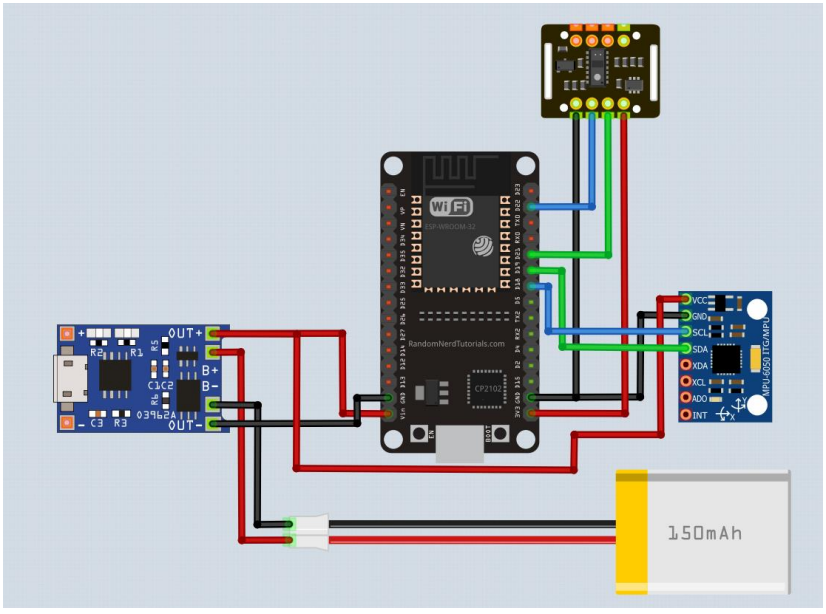


Рисунок 3.5 – Принципова схема апаратної частини комплексу

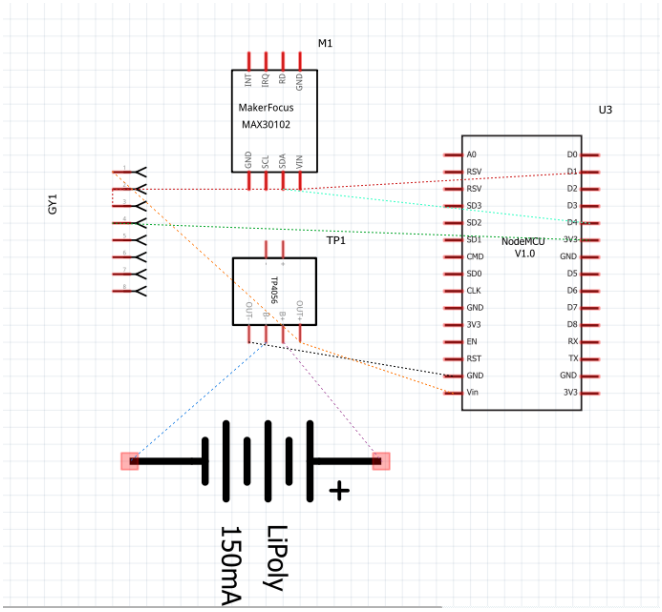
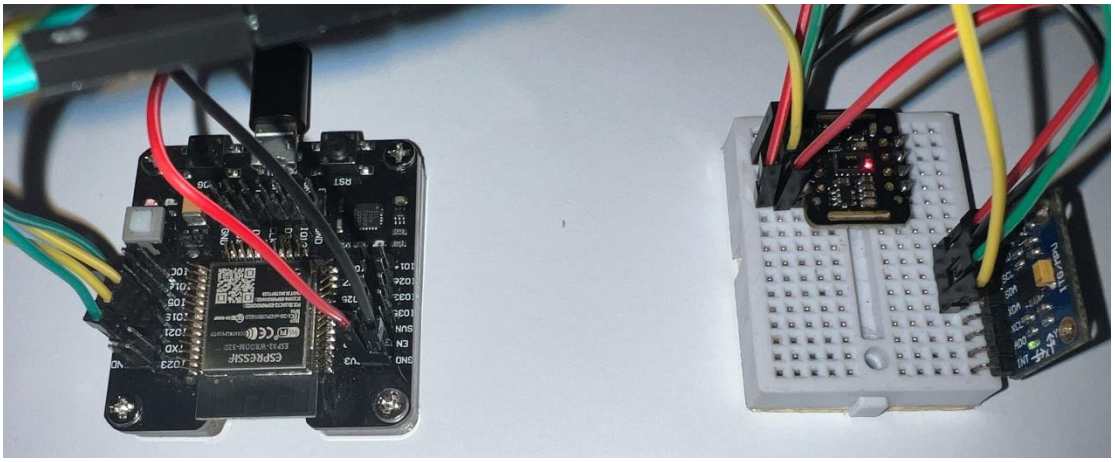


Рисунок 3.6 – Монтажна схема апаратної частини комплексу

Підключення компонентів наведено на рис. 3.7. Зверніть увагу – так як MPU 6050 та MAX30102 використовують I2C для підключення даних, в коді програми прописується дві I2C шини для обміну даних. Сам контролер ESP32-WROOM-32D підключено до модулю-адаптеру, що дозволяє не використовувати для програмування плати в форм-факторі NodeMCU та завдяки цьому зменшуючи розмір прототипу.



Модель для пристрою створено за допомогою онлайн-сервісу Autodesk Tinkercad, що дозволяє створювати нові та модифікувати наявні 3D-моделі. Вона складається з корпусу, де вміщуються електронні компоненти пристрою та ремінця для носіння його на руці (рис. 3.7). Розмір в розкладеному вигляді сягає 189 мм в довжину та 28 мм в ширину. Сам

корпус, де знаходиться контролер, плати та акумулятор, має розміри 30x28x25 мм.

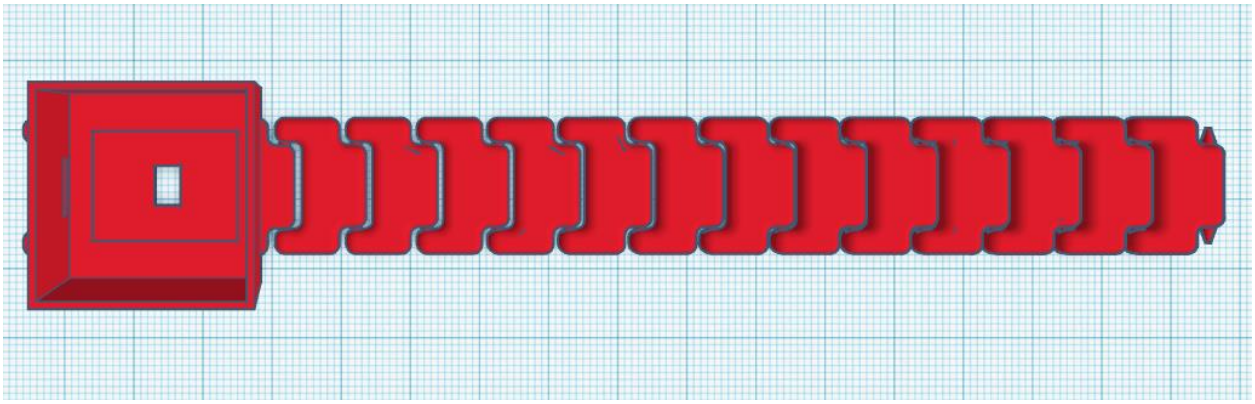


Рисунок 3.7 – 3D-модель корпусу та ремінця в зібраному вигляді для апаратної частини комплексу моніторингу фізіологічних показників людини

За допомогою програмного забезпечення Ultimaker Cura .stl файл з 3D-моделлю трансльовано в G-Code формат та роздруковано на 3D-принтері з використанням PLA пластику. Вигляд корпусу після друку наведено на рис. 3.8.

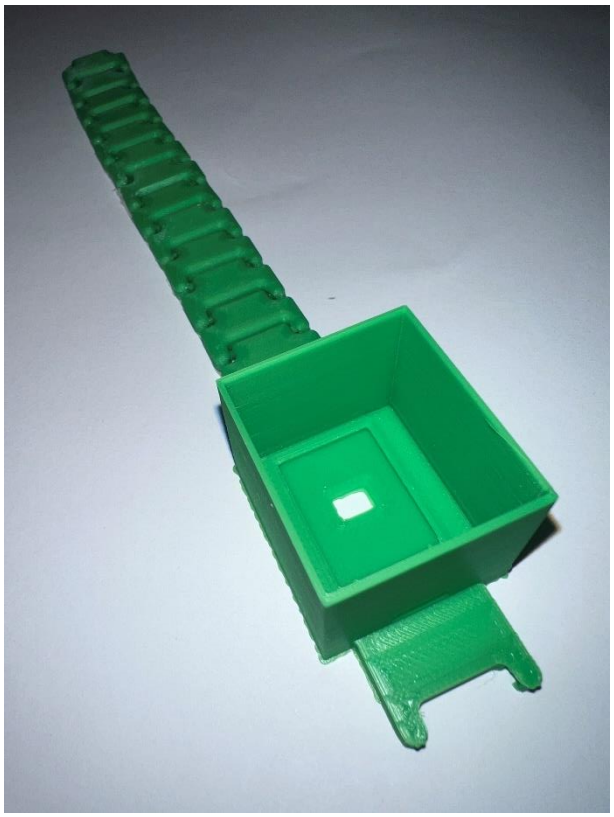


Рисунок 3.8 – Вигляд корпусу після друку (без верхньої кришки панелі)

Монтажну схему можна розбити на декілька частин – секція живлення контроллерної плати та проектної частини, де відбувається зняття даних та передача їх до серверу. Прототип блоку живлення модулю наведено на рис. 3.9 і він складається з модулю керування живленням TP4056, літій-полімерної батареї на 150 мАг подальшого живлення компонентів схеми.

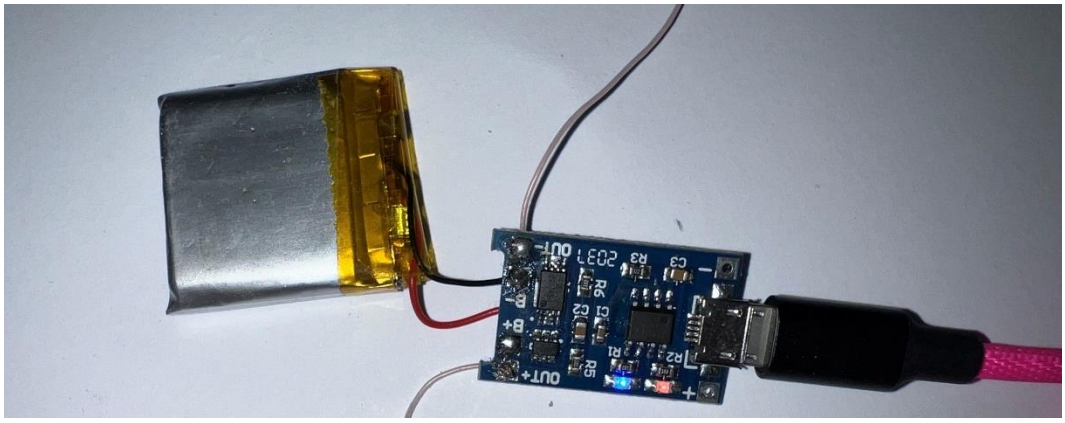


Рисунок 3.9 – Частина модулю, що відповідає за живлення пристрою

Сам прототип пристрою збору фізіологічних показників стану людини наведено на рис. 3.10. Через те, що електричні компоненти пристрою не знаходяться на одній платі – корпус, де розмішуються всі компоненти, є відносно високим.

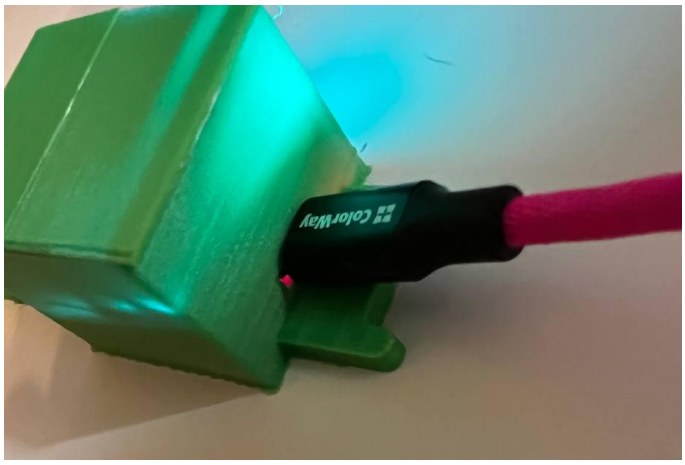


Рисунок 3.10 – Прототип пристрою збору фізіологічних показників стану людини

3.3. Розробка програмного забезпечення до апаратного модулю

Програмне забезпечення до апаратного модулю узагальнено складається з наступних частин:

– Налаштування двох I2C інтерфейсів

– Блок підключення до WiFi мережі

– Блок шифрування та надсилання даних

– Блок збору даних серцевого ритму, кров’яної оксигенації та температури тіла

– Блок підрахунку кількості кроків

Реалізація налаштування двох I2C інтерфейсів починається з визначення пінів, які будуть використані для цього. В даному випадку для першого I2C інтерфейсу обрано в якості SDA піну GPIO21, в якості SCL – GPIO 22. Другий інтерфейс – SDA це GPIO19, SCL – GPIO18. Наступним кроком є створення об’єктів класу TwoWire для оголошення першої та другої I2C шин. Після того в setup() за допомогою методу begin() запускаємо I2C інтерфейси та в loop() використовуємо режим прийому-передачі даних для кожного з інтерфейсів. Приклад коду з налаштуванням I2C інтерфейсів для ESP32 наведено на рис. 3.11.

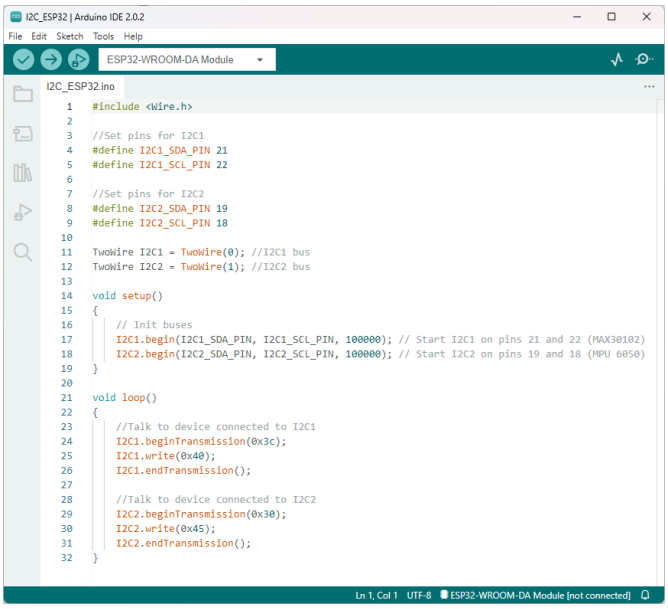


Рисунок 3.11 – Приклад налаштування двох I2C інтерфейсів для роботи з MAX30102 та MPU 6050

Для підключення до WiFi мережі контролеру ESP32 використовується бібліотека WiFi.h. Саме підключення відбувається наступним чином: за допомогою методу WiFi.begin() створюється нове підключення до WiFi мережі. В якості аргументів цей метод приймає ssid та пароль від локальної мережі. У випадку, якщо WiFi.status() не повертає значення WL_CONNECTED, то пристрій намагається підключитися знову. Приклад реалізації підключення до WiFi мережі наведено на рис. 3.12.

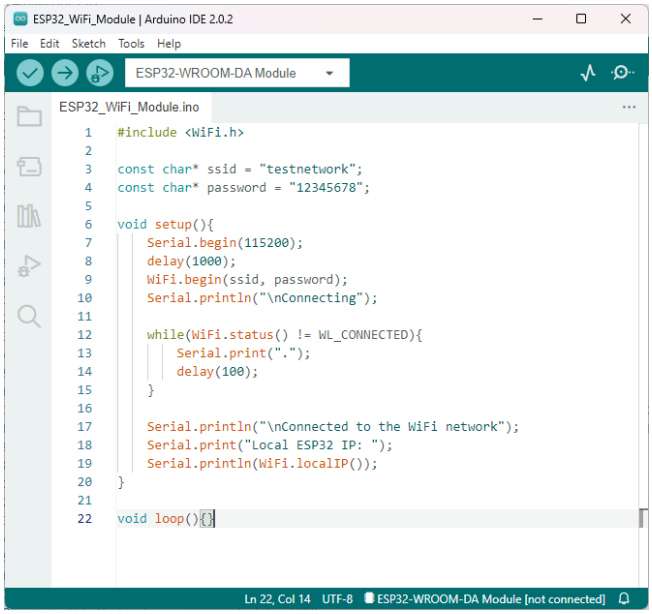


Рисунок 3.12 – Приклад блоку коду для підключення до WiFi мережі

За допомогою модулю MAX30102 можна отримати значення температури тіла людини, серцевого ритму, рівня оксигенації. В розробці програмного коду для отримання цих даних використано бібліотеки «Wire.h», «MAX30105.h», «heartRate.h», «spo2_algorithm.h» від SparkFun. Так як одночасно нема можливості робити виміри пульсу, SpO2 та температури, реалізується перемикання модулю в різні режими виміру, як на рис. 3.13. Кожне перемикання відбувається через 1000 замірів, що відбуваються за мілісекунди, один повний цикл заміру даних з модулю MAX30102 зазвичай займає 2-4 секунди.

```
void setup_hr_mode()
{
  particleSensor.setup(); //Configure sensor with default settings
  particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate sensor is running
  particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
}

void setup_temp_mode()
{
  particleSensor.setup(0); //Configure sensor. Turn off LEDs
  particleSensor.enableDIETEMPRDY(); //Enable the temp ready interrupt. This is required.
}

void setup_spo2_mode()
{
  byte ledBrightness = 60; //Options: 0=Off to 255=50mA
  byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
  byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
  byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
  int pulseWidth = 411; //Options: 69, 118, 215, 411
  int adcRange = 4096; //Options: 2048, 4096, 8192, 16384
  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange); //Configure sensor with these settings
}
```

Рисунок 3.13 – Налаштування кожного з режимів роботи MAX30102

Повний код програми з реалізацією всіх замірів пульсу, оксигенації та температури наведено в додатку А. Модуль підрахунку кроків, частину коду якого показано на рис. 3.14, так само розміщено в додатку А.

```
void step_counter()
{
    mpu6050_acceleration_t accel;
    mpu6050_rotation_t gyro;
    uint64_t now_time = 0, prev_time = 0;
    int16_t temp;
    int8_t range = 0;
    float accel_g_x, accel_g_y, accel_g_z;
    float accel_int_x, accel_int_y, accel_int_z;
    float gyro_ds_x, gyro_ds_y, gyro_ds_z, accel_res, gyro_res, temp_c;
    int accel_x_avg_buff[AVG_BUFF_SIZE];
    int accel_y_avg_buff[AVG_BUFF_SIZE];
    int accel_z_avg_buff[AVG_BUFF_SIZE];
    int accel_x_avg_buff_count = 0;
    int accel_y_avg_buff_count = 0;
    int accel_z_avg_buff_count = 0;
    int accel_x_avg, accel_y_avg, accel_z_avg;
    int min_reg_accel_x = 0, min_reg_accel_y = 0, min_reg_accel_z = 0;
    int max_reg_accel_x = 0, max_reg_accel_y = 0, max_reg_accel_z = 0;
    int min_curr_accel_x, min_curr_accel_y, min_curr_accel_z;
    int max_curr_accel_x, max_curr_accel_y, max_curr_accel_z;
    int dy_thres_accel_x = 0, dy_thres_accel_y = 0, dy_thres_accel_z = 0;
    int dy_chan_accel_x, dy_chan_accel_y, dy_chan_accel_z;
    int sample_new = 0, sample_old = 0;
    int step_size = 200, step_count = 0;
    int active_axis = 0, interval = 500000;
    int step_changed = 0;
```

Рисунок 3.14 – Частина коду реалізації підрахунку кроків

3.4 Розробка програмного забезпечення для серверної частини проекту

Серверна частина реалізована за допомогою мови програмування Python версії 3.11 та з використанням веб-фреймворку Django. Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Він підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі

на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Django – високорівневий відкритий Python-фреймворк для розробки веб-систем та сервісів. Його архітектура подібна на архітектуру «Модель-представлення-контролер» (MVC) Однак, те що називається «контролером» в класичній моделі MVC, в Django називається «вид», а те, що мало б бути «видом», називається «шаблон». Таким чином, MVC в Django називають MTV («Модель-Шаблон-Вигляд»).

Структура проекту серверного застосунку наведена на рис. 3.15.

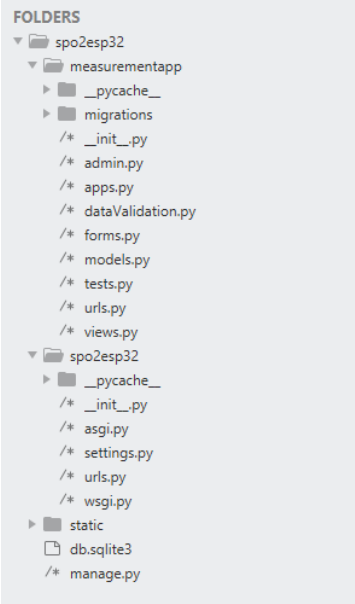


Рисунок 3.15 – Структура серверної частини проекту

Файл manage.py є файлом-консольною утилітою, що допомагає в створенні, розгортанні та управління проектом. Файл settings.py містить в собі налаштування проекту такі як режим роботи, вивід інформації по відлагоджуванню, підключені додатки та інше. В urls.py описані шляхи до кожного застосунку, наприклад, на рис. 3.16 зображено зв’язування застосунку з кореневим шляхом проекту.

```
urlpatterns = [
    path('', include('measurementapp.urls')),
    path('admin/', admin.site.urls),
]
```

Рисунок 3.16 – Налаштування кореневого шляху та прив'язка до застосунку

Папка measurementapp – це є папка самого застосунку, який містить в собі опис бази даних (models.py), опис форм для парольного вводу (forms.py), опис шляхів всередині застосунку (urls.py), опис шаблонів застосунку (views.py) та реалізація функціоналу для фільтрації та валідації отриманих даних з пристрою (dataValidation.py). Папка templates містить в собі html-шаблони сторінок для кожного з представлень застосунку. Вміст всіх файлів проекту наведено в додатку Б.

Структура бази даних серверного застосунку описана раніше в другому розділі цієї роботи. Сама ж база даних програмно описана в файлі models.py (рис. 3.17) та за допомогою команди `python manage.py makemigrations` доповнена таблицями з префіксом measurementapp_. Структура бази даних наведена на рис. 3.18.

```
from django.db import models

class DeviceList(models.Model):
    uuid = models.CharField(max_length=200, unique=True)
    hardwareId = models.CharField(max_length=200, unique=True)
    deviceDescription = models.CharField(max_length=200)

class UserList(models.Model):
    userID = models.CharField(max_length=200, unique=True)
    username = models.CharField(max_length=200)
    passwd = models.CharField(max_length=50)
    deviceID = models.CharField(max_length=200)

class Data(models.Model):
    uuid = models.CharField(max_length=200)
    timestamp = models.DateTimeField(auto_now_add=True)
    hr = models.IntegerField()
    spo2 = models.FloatField()
    temp = models.FloatField()
    steps = models.IntegerField()

class SettingsTable(models.Model):
    deviceID = models.CharField(max_length=200)
    timestamp = models.DateTimeField()
    ssid = models.CharField(max_length=200)
    password = models.CharField(max_length=50)
    bluetoothStatus = models.BooleanField(default = True)
    frequency = models.IntegerField(default = 10)
```

Рисунок 3.17 – Вміст models.py

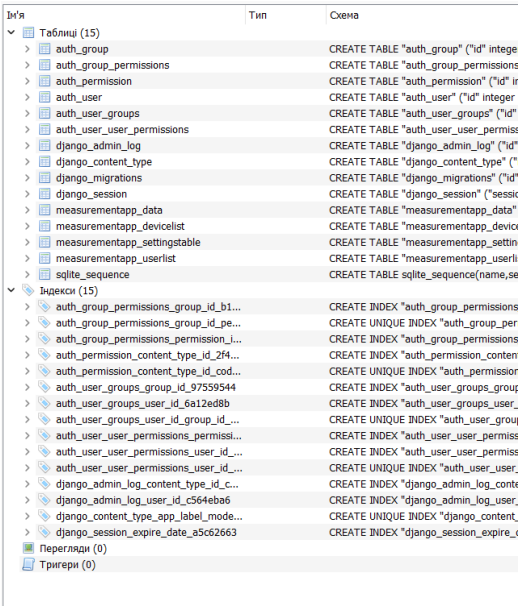


Рисунок 3.18 – Структура бази даних серверного застосунку

Висновки до розділу 3

В цьому розділі виконано реалізацію програмно-апаратного комплексу моніторингу та аналізу фізіологічних показників стану людини на базі модулю ESP32. Було підібрано потрібну елементну базу, створено 3D-модель корпусу пристрою та зібрано два робочих прототипи пристрою. Також було розроблено клієнтське та серверне програмне забезпечення для представлення та обробки зібраних даних по серцевому ритму, пульсоксиметрії, температури та кількості кроків за добу.

4 ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Огляд та аналіз отриманих даних

Для аналізу отриманих даних зроблена вибірка даних температури, пульсу, оксигенації крові, отриманих 14.02.2023 в період з 11:30 до 20:50 UTC+2. Скріншот сторінки з результатами заміру наведено на рис. 4.1. Відповідно до рекомендацій, для більш-точного вимірювання серцевого ритму, рівня оксигенації крові та температури, рука, на якій знаходиться пристрій, має знаходитися на столі та бути розташована навпроти серця. Також бажано, щоб під час вимірювань пульсу, SpO2 та температури знаходилась в сидячому положенні.

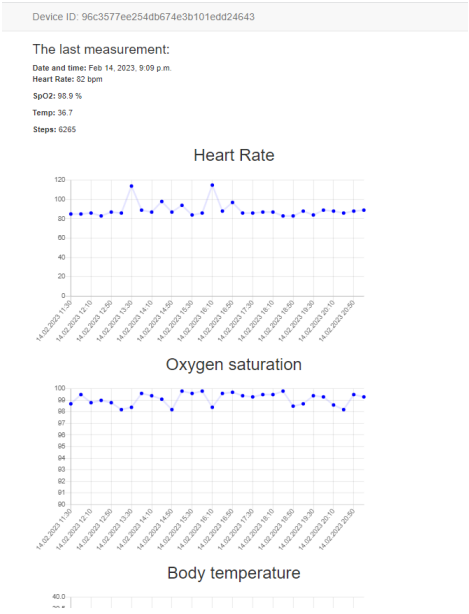


Рисунок 4.1 – Скріншот отриманих даних з пристрою

Розглянемо більш детальніше отримані дані. Кожна точка на графіку показує значення одного з показників (серцевого ритму,SpO2, температури тіла, сумарної кількості кроків за добу) кожні 20 хвилин виміру. Наприклад, в графіку значень серцевого ритму 14.02.2023 о 12:10 показник пульсу був 86 ударів за хвилину (рис. 4.2).

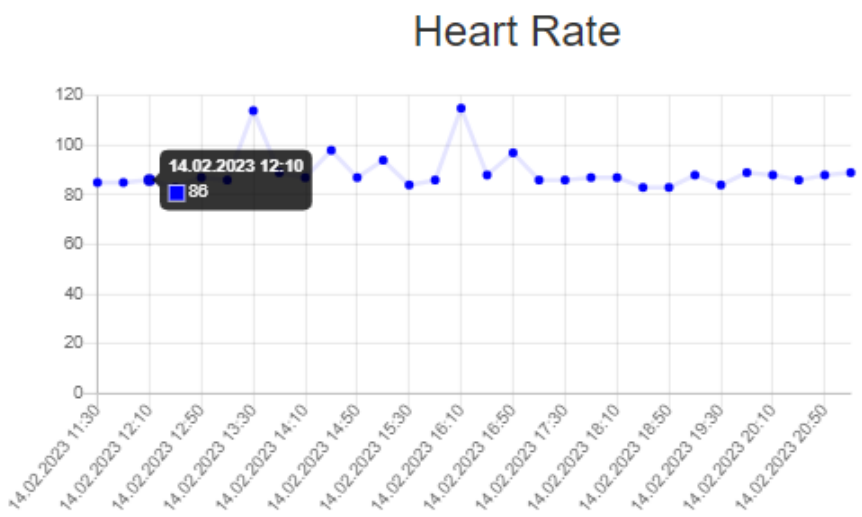


Рисунок 4.2 – Графік значень серцевого ритму, отриманий за допомогою прототипу пристрою

На рис. 4.3. показано вибірку значень оксигенації крові (SpO2), отриманих за допомогою розробленого пристрою. Наприклад, значення цього показника о 14:50 становило 98,5%, що є на рівні норми.

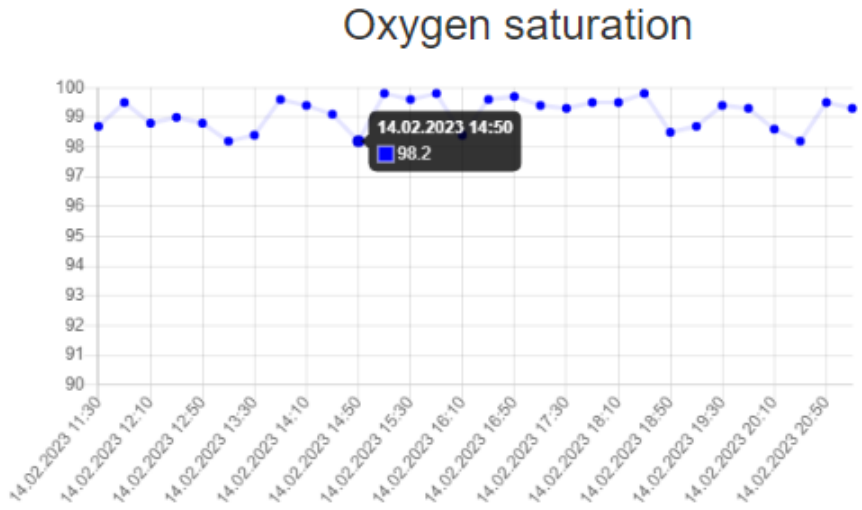


Рисунок 4.3 – Показники оксигенації крові, отримані за допомогою розробленого пристрою

Вибірка отриманих значень температури тіла наведена на рис. 4.4. Загалом, значення температури тіла коливалась від 36,4 до 37,1. Тому можна припустити, що значення температури тіла є на рівні норми.

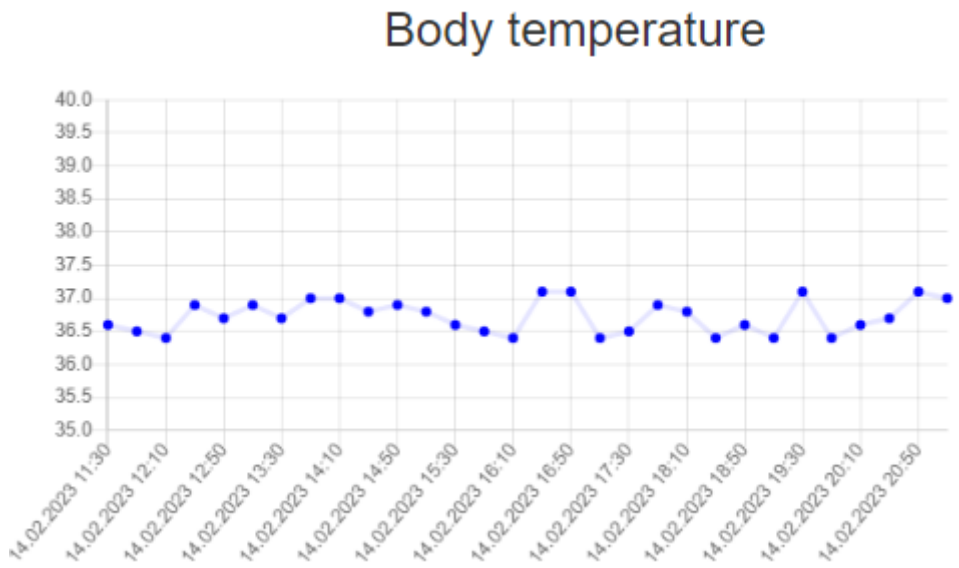


Рисунок 4.4 – Вибірка значень температури тіла

Значення кількості пройдених кроків наведено на рис. 4.5. Загальна кількість кроків на кінець заміру становить 6265 кроків.

Рисунок 4.5 – Кількість пройдених кроків з початку доби

4.2 Аналіз отриманих результатів

Дані серцевого ритму та SpO2, отримані сенсором MAX30102 та які після обробки вже не містять значень-викидів, мають точність 97,11% для пульсу та 98.84% для SpO2 відповідно [31]. Подібні сенсори використовуються не тільки в медичних пульсоксиметрах, а і в розумних годинниках різної цінової категорії. Для порівняння, на рис. 4.6 наведено

добовий графік серцевого ритму за 14.02.2023, що згенеровано застосунком Zepp Life, який збирає та оброблює дані з розумних годинників виробництва Xiaomi. Ці дані пульсу отримані з пристрою Mi Band 7, і на часовому проміжку вибірки даних з рис. 4.2 ці дані відрізняються в середньому на 2-5 ударів за хвилину.

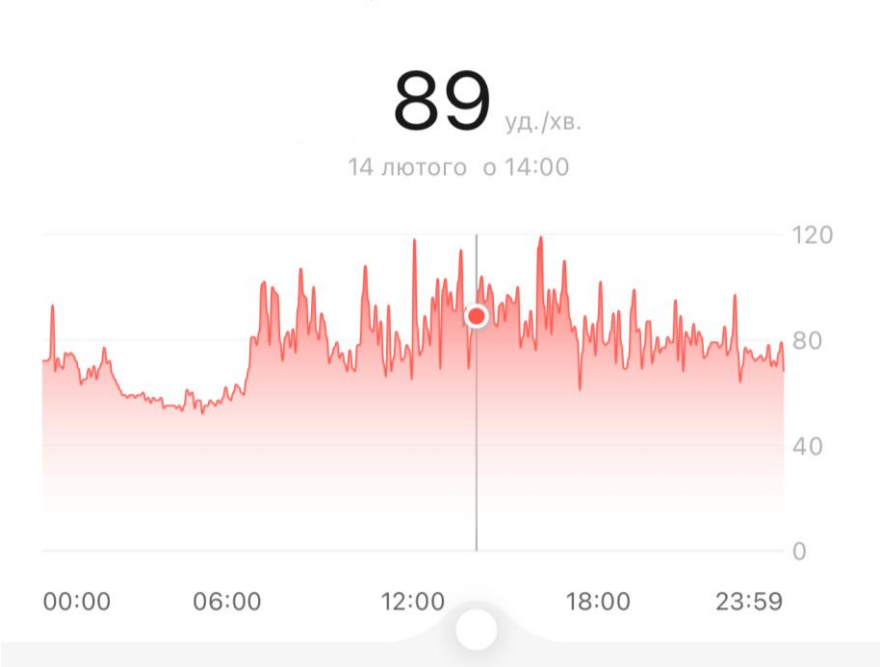


Рисунок 4.6 – Дані серцевого ритму, отримані з пристрою Mi Band 7

На відміну від серцевого ритму, підрахунок значень кількості пройдених кроків за добу частково відрізняється від отриманих з Mi Band 7 (рис. 4.7). Це може бути пов’язано як з надмірною чутливістю MPU 6050, так і нестачею даних щодо зросту та ваги під час обрахунку кількості кроків на пристрої.

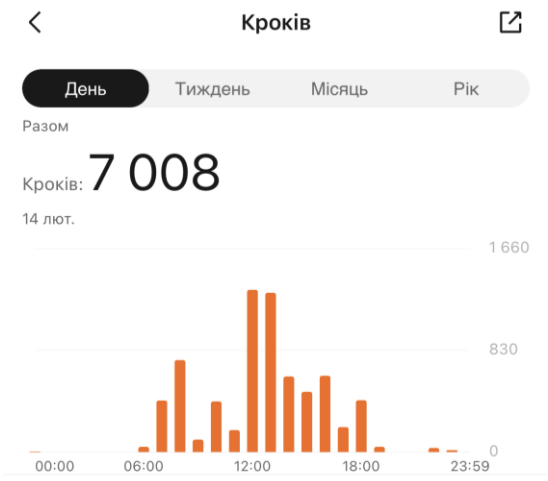


Рисунок 4.7 – Кількість кроків, виявлених пристроєм Mi Band 7

Висновки до розділу 4

В цьому розділі виконано аналіз та порівняння отриманих даних з пристрою моніторингу та аналізу фізіологічних показників стану людини на базі модулю ESP32. Ці дані порівнювались з даними серцевого ритму та кількості кроків, отриманих з Mi Band 7 в той же самий часовий період. Підтверджено результат дослідження [31]щодо похибки у вимірюванні серцевого ритму.

Також виконано порівняння даних виявленої кількості кроків, отриманих як зі створеного пристрою, так і з Mi Band 7. Дані частково відрізняються, але це пов’язано з певною неточністю при виявлені кількості кроків за допомогою MPU 6050, так як для отримання більш точних даних потрібно враховувати зріст та вагу особи.

ВИСНОВКИ

У представленій кваліфікаційній роботі магістра спроектовано та розроблено програмно-апаратний модуль моніторингу та аналізу фізіологічних показників стану людини. Проект складається з трьох компонентів: пристрою збору, обробки даних та їх передачі на сервер; серверного програмного забезпечення для глибинної обробки, аналізу та генерації представлень отриманих даних; клієнтського програмного забезпечення, що надає користувачеві представлення оброблених даних. Кваліфікаційна робота пройшла апробацію під час конференцій [1]–[3].

Результатами цієї роботи є виконання поставлених задач, а саме:

– Досліджено наявні системи збору та обробки фізіологічних показників людини та проведено порівняльний аналіз наявних систем для визначення їх переваг та недоліків;

– Визначено апаратні та програмні компоненти для побудови апаратно-програмного комплексу

– Розроблено та здійснено апробацію методу первинної обробки отриманих даних сенсором MAX30102, який працює за принципом фотоплетизмографії;

– Розроблено та здійснено апробацію методів передачі даних поміж пристроєм збору показників та серверним програмним забезпеченням з використанням алгоритмів шифрування даних;

– Розроблено та здійснено апробацію методів обробки, фільтрації та візуалізації отриманих показників стану людини;

– Розроблено апаратний модуль збору фізіологічних показників людини;

– Розроблено програмне забезпечення для апаратного модулю, серверної та клієнтської частини апаратно-програмного комплексу;

– Порівняно результати роботи створеного апаратно-програмного комплексу з комерційними аналогами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

[1] Y. Krainyk, Y. Darnapuk, and I. Simakova, "Software System for Physical Activity Monitoring: Smart Watch Case," in *2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*, Sep. 2020, pp. 1–4. doi: 10.1109/IDAACS-SWS50031.2020.9297093.

[2] Y. Krainyk, A. Razzhyvin, O. Bondarenko, and I. Simakova, "Internet-of-things device set configuration for connection to wireless local area network," in *CEUR Workshop Proceedings*, 2019, vol. 2353, pp. 885–896.

[3] І. Сімакова, Я. Крайник, and Є. Дарнапук, "ESP32 як основа для IoT-пристроїв збору медичних показників людини," in *Могилянські читання – 2022,: Досвід та тенденції розвитку суспільства в Україні : глобальний, національний та регіональний аспекти : XXV Всеукр. наук.-практ. конф. : тези доповідей : Комп'ютерні науки. Технічні науки.*, 2022, pp. 85–88.

[4] "Apple Watch Ultra – Apple (Україна)." https://www.apple.com/ua/apple-watch-ultra/ (accessed Dec. 15, 2022).

[5] "Купити Samsung Galaxy Watch 4 Classic Black SM-R880NZKASEK | Samsung Україна." https://www.samsung.com/ua/watches/galaxy-watch/galaxy-watch4-classic-black-42-sm-r880nzkasek/ (accessed Dec. 17, 2022).

[6] "Xiaomi Smart Band 7 Pro - Xiaomi UK." https://www.mi.com/global/product/xiaomi-smart-band-7-pro/ (accessed Dec. 17, 2022).

[7] V. Sujitha, R. Sudarmani, B. Aishwarya, V. Vishnu Sanjana, and P. Vigneswari, "IoT based Healthcare Monitoring and Tracking System for Soldiers using ESP32," in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, Mar. 2022, pp. 377–381. doi: 10.1109/ICCMC53470.2022.9754076.

[8] "The Internet of Things with ESP32." http://esp32.net/ (accessed Oct. 15, 2022).

[9] D. Anandhavalli, N. Shanmuga Sundari, S. Aparna Tharshini, P. Pavithra, and G. Sona, "Patient Health Monitoring System Using IoT Techniques," *SSRN Electronic Journal*, 2021, doi: 10.2139/ssrn.3852046.

[10] Maxim Integrated, "High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health MAX30102," *Maxim Integrated*, pp. 1–32, 2015, [Online]. Available: https://www.maximintegrated.com/en/products/sensors/MAX30102.html

[11] S. Pham *et al.*, "Wearable Sensor System to Monitor Physical Activity and the Physiological Effects of Heat Exposure," *Sensors*, vol. 20, no. 3, p. 855, Feb. 2020, doi: 10.3390/s20030855.

[12] Maxim Integrated, "Max30100," *Lecture Notes in Energy*, 2014.

[13] "US20150201853A1 - Wearable heart rate monitor - Google Patents." https://patents.google.com/patent/US20150201853A1/en (accessed Dec. 17, 2022).

[14] A. A. Alian and K. H. Shelley, "Photoplethysmography," *Best Pract Res Clin Anaesthesiol*, vol. 28, no. 4, pp. 395–406, Dec. 2014, doi: 10.1016/j.bpa.2014.08.006.

[15] G. Chuiko, Y. Krainyk, O. Dvornik, and Y. Darnapuk, *IoT in Provenance Management of Medical Data*, vol. 941. 2021. doi: 10.1007/978-3-030-64619-6_15.

[16] Y. Krainyk, Y. Darnapuk, and I. Simakova, "Software System for Physical Activity Monitoring: Smart Watch Case," in *IDAACS-SWS 2020 - 5th IEEE International Symposium on Smart and Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems, Proceedings*, 2020. doi: 10.1109/IDAACS-SWS50031.2020.9297093.

[17] L. Srinivasan, Venkatesh, G. Pranay Deepak Reddy, B. S. Sai Pramath, and J. A. Trivedh, "IoT Based Low-Cost Robotic Agent Design for Covid-19 affected people," in *Proceedings of the International Conference on Electronics and Renewable Systems, ICEARS 2022*, 2022, pp. 468–473. doi: 10.1109/ICEARS53579.2022.9751926.

[18] T. Franco *et al.*, "Motion Sensors for Knee Angle Recognition in Muscle Rehabilitation Solutions," *Sensors*, vol. 22, no. 19, 2022, doi: 10.3390/s22197605.

[19] D. Jing, "Design and Implementation of a Batteryless Pedometer based on a Motion Tracking Sensor," in *ACM International Conference Proceeding Series*, 2022, pp. 19–22. doi: 10.1145/3548608.3561131.

[20] P. Bhattacharjee and S. Biswas, "Smart walking assistant (SWA) for elderly care using an intelligent realtime hybrid model," *Evolving Systems*, vol. 13, no. 2, pp. 265–279, 2022, doi: 10.1007/s12530-021-09382-5.

[21] D. Gozal *et al.*, "Wavelet analysis of oximetry recordings to assist in the automated detection of moderate-to-severe pediatric sleep apnea-hypopnea syndrome," *PLoS One*, vol. 13, no. 12, 2018, doi: 10.1371/journal.pone.0208502.

[22] Y.-S. Lin, Y.-P. Wu, Y.-C. Wu, P.-L. Lee, and C.-H. Yang, "Achieving Accurate Automatic Sleep Apnea/Hypopnea Syndrome Assessment Using Nasal Pressure Signal," *IEEE J Biomed Health Inform*, vol. 26, no. 11, pp. 5473–5481, 2022, doi: 10.1109/JBHI.2022.3199454.

[23] J. Guo, B. Wan, S. Zheng, A. Song, and W. Huang, "A Teenager Physical Fitness Evaluation Model Based on 1D-CNN with LSTM and Wearable Running PPG Recordings," *Biosensors (Basel)*, vol. 12, no. 4, 2022, doi: 10.3390/bios12040202.

[24] M. Santos *et al.*, "The Use of Wearable Pulse Oximeters in the Prompt Detection of Hypoxemia and During Movement: Diagnostic Accuracy Study," *J Med Internet Res*, vol. 24, no. 2, 2022, doi: 10.2196/28890.

[25] R. Podder and R. K. Barai, "Hybrid Encryption Algorithm for the Data Security of ESP32 based IoT-enabled Robots," in *2021 Innovations in Energy Management and Renewable Resources, IEMRE 2021*, 2021. doi: 10.1109/IEMRE52042.2021.9386824.

[26] A. Anand, A. Galletta, A. Celesti, M. Fazio, and M. Villari, "A secure inter-domain communication for IoT devices," in *Proceedings - 2019 IEEE International Conference on Cloud Engineering, IC2E 2019*, 2019, pp. 235–240. doi: 10.1109/IC2E.2019.00038.

[27] M. Suarez-Albela, T. M. Fernandez-Carames, P. Fraga-Lamas, and L. Castedo, "A practical performance comparison of ECC and RSA for resource-constrained IoT devices," in *2018 Global Internet of Things Summit, GIoTS 2018*, 2018. doi: 10.1109/GIOTS.2018.8534575.

[28] "The Internet of Things with ESP32." http://esp32.net/ (accessed Feb. 15, 2023).

[29] O. Barybin, E. Zaitseva, and V. Brazhnyi, "TESTING THE SECURITY ESP32 INTERNET OF THINGS DEVICES," *Cybersecurity: Education, Science, Technique*, vol. 2, no. 6, pp. 71–81, 2019, doi: 10.28925/2663-4023.2019.6.7181.

[30] S. V, Sudarmani. R, A. B, V. S. V, and P. Vigneswari, "IoT based Healthcare Monitoring and Tracking System for Soldiers using ESP32," in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, Mar. 2022, pp. 377–381. doi: 10.1109/ICCMC53470.2022.9754076.

[31] N. N. Sari, M. N. Gani, R. A. M. Yusuf, and R. Firmando, "Telemedicine for silent hypoxia: Improving the reliability and accuracy of Max30100-based system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 3, p. 1419, Jun. 2021, doi: 10.11591/ijeecs.v22.i3.pp1419-1426.

Додаток А

Код апаратної частини комплексу

```
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"
#include "spo2_algorithm.h"

MAX30105 particleSensor;
//Set pins for I2C1
#define I2C1_SDA_PIN 21
#define I2C1_SCL_PIN 22

//Set pins for I2C2
#define I2C2_SDA_PIN 19
#define I2C2_SCL_PIN 18

TwoWire I2C1 = TwoWire(0); //I2C1 bus
TwoWire I2C2 = TwoWire(1); //I2C2 bus

#define MAX_BRIGHTNESS 255
uint32_t irBuffer[100]; //infrared LED sensor data
uint32_t redBuffer[100];  //red LED sensor data
int32_t bufferLength; //data length
int32_t spo2; //SPO2 value
int8_t validSPO2; //indicator to show if the SPO2 calculation is valid
int32_t heartRate_spo2; //heart rate value for spo2 procedure
int8_t validHeartRate_spo2; //indicator to show if the heart rate calculation is
valid

const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is good.
byte rates[RATE_SIZE]; //Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred

float beatsPerMinute;
int beatAvg;
int appmode = 0; // 0 - heart rate, 1 - temperature, 2 - SpO2
int HRCounter = 0;
float currentTemp = 0;
long Temp = 0;
float averageTemp = 0;
float TempCounter = 0;
long irValue = 0;
void setup()
{
  Serial.begin(115200);
  Serial.println("Initializing...");
```

```
  // Initialize sensor
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port, 400kHz
speed
  {
    Serial.println("MAX30105 was not found. Please check wiring/power. ");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with steady pressure.");
  setup_hr_mode();
}

void setup_hr_mode()
{
  particleSensor.setup(); //Configure sensor with default settings
  particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate
sensor is running
  particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
}

void setup_temp_mode()
{
  particleSensor.setup(0); //Configure sensor. Turn off LEDs
  particleSensor.enableDIETEMPRDY(); //Enable the temp ready interrupt. This is
required.
}

void setup_spo2_mode()
{
  byte ledBrightness = 60; //Options: 0=Off to 255=50mA
  byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
  byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
  byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
  int pulseWidth = 411; //Options: 69, 118, 215, 411
  int adcRange = 4096; //Options: 2048, 4096, 8192, 16384
  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate,
pulseWidth, adcRange); //Configure sensor with these settings
}


float check_temperature()
{
  return particleSensor.readTemperature();
}

void check_hr_value()
{
  irValue = particleSensor.getIR();

  if (checkForBeat(irValue) == true)
  {
```

```
    //We sensed a beat!
    long delta = millis() - lastBeat;
    lastBeat = millis();

    beatsPerMinute = 60 / (delta / 1000.0);

    if (beatsPerMinute < 255 && beatsPerMinute > 20)
    {
      rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in the array
      rateSpot %= RATE_SIZE; //Wrap variable

      //Take average of readings
      beatAvg = 0;
      for (byte x = 0 ; x < RATE_SIZE ; x++)
        beatAvg += rates[x];
      beatAvg /= RATE_SIZE;
    }
  }

}

void step_counter()
{
    mpu6050_acceleration_t accel;
    mpu6050_rotation_t gyro;
    uint64_t now_time = 0, prev_time = 0;
    int16_t temp;
    int8_t range = 0;
    float accel_g_x, accel_g_y, accel_g_z;
    float accel_int_x, accel_int_y, accel_int_z;
    float gyro_ds_x, gyro_ds_y, gyro_ds_z, accel_res, gyro_res, temp_c;
    int accel_x_avg_buff[AVG_BUFF_SIZE];
    int accel_y_avg_buff[AVG_BUFF_SIZE];
    int accel_z_avg_buff[AVG_BUFF_SIZE];
    int accel_x_avg_buff_count = 0;
    int accel_y_avg_buff_count = 0;
    int accel_z_avg_buff_count = 0;
    int accel_x_avg, accel_y_avg, accel_z_avg;
    int min_reg_accel_x = 0, min_reg_accel_y = 0, min_reg_accel_z = 0;
    int max_reg_accel_x = 0, max_reg_accel_y = 0, max_reg_accel_z = 0;
    int min_curr_accel_x, min_curr_accel_y, min_curr_accel_z;
    int max_curr_accel_x, max_curr_accel_y, max_curr_accel_z;
    int dy_thres_accel_x = 0, dy_thres_accel_y = 0, dy_thres_accel_z = 0;
    int dy_chan_accel_x, dy_chan_accel_y, dy_chan_accel_z;
    int sample_new = 0, sample_old = 0;
    int step_size = 200, step_count = 0;
    int active_axis = 0, interval = 500000;
    int step_changed = 0;

    while (true) {
```

```
        if (!mpu6050_get_int_dmp_status()) {
            mpu6050_get_acceleration(&accel);

            range = mpu6050_get_full_scale_accel_range();
            accel_res = mpu6050_get_accel_res(range);

            accel_g_x = (float) accel.accel_x * accel_res - accel_bias[0];
            accel_g_y = (float) accel.accel_y * accel_res - accel_bias[1];
            accel_g_z = (float) accel.accel_z * accel_res - accel_bias[2];

            if (accel_g_x < 0)
                accel_g_x *= -1;
            if (accel_g_y < 0)
                accel_g_y *= -1;
            if (accel_g_z < 0)
                accel_g_z *= -1;

            mpu6050_get_rotation(&gyro);

            range = mpu6050_get_full_scale_gyro_range();
            gyro_res = mpu6050_get_gyro_res(range);

            gyro_ds_x = (float) gyro.gyro_x * gyro_res - gyro_bias[0];
            gyro_ds_y = (float) gyro.gyro_y * gyro_res - gyro_bias[1];
            gyro_ds_z = (float) gyro.gyro_z * gyro_res - gyro_bias[2];

            temp = mpu6050_get_temperature();
            temp_c = (float) temp / 340.0 + 36.53;

            mpu6050_madgwick_quaternion_update
            (
                accel_g_x,
                accel_g_y,
                accel_g_z,
                gyro_ds_x * PI / 180.0f,
                gyro_ds_y * PI / 180.0f,
                gyro_ds_z * PI / 180.0f
            );

            accel_int_x = 1000 * accel_g_x;
            accel_int_y = 1000 * accel_g_y;
            accel_int_z = 1000 * accel_g_z;

            accel_x_avg_buff[accel_x_avg_buff_count] = accel_int_x;
            accel_x_avg_buff_count++;
            accel_x_avg_buff_count %= AVG_BUFF_SIZE;
            accel_x_avg = 0;

            for (int i = 0; i < AVG_BUFF_SIZE; i++)
                accel_x_avg += accel_x_avg_buff[i];
```

```
accel_x_avg /= AVG_BUFF_SIZE;

accel_y_avg_buff[accel_y_avg_buff_count] = accel_int_y;
accel_y_avg_buff_count++;
accel_y_avg_buff_count %= AVG_BUFF_SIZE;
accel_y_avg = 0;

for (int i = 0; i < AVG_BUFF_SIZE; i++)
    accel_y_avg += accel_y_avg_buff[i];

accel_y_avg /= AVG_BUFF_SIZE;

accel_z_avg_buff[accel_z_avg_buff_count] = accel_int_z;
accel_z_avg_buff_count++;
accel_z_avg_buff_count %= AVG_BUFF_SIZE;
accel_z_avg = 0;

for (int i = 0; i < AVG_BUFF_SIZE; i++)
    accel_z_avg += accel_z_avg_buff[i];

accel_z_avg /= AVG_BUFF_SIZE;

now_time = esp_timer_get_time();

if (now_time - prev_time >= interval) {
    prev_time = now_time;

    min_curr_accel_x = min_reg_accel_x;
    max_curr_accel_x = max_reg_accel_x;
    dy_thres_accel_x = (min_curr_accel_x + max_curr_accel_x) / 2;
    dy_chan_accel_x = (max_curr_accel_x - min_curr_accel_x);
    min_reg_accel_x = accel_x_avg;
    max_reg_accel_x = accel_x_avg;
    min_curr_accel_y = min_reg_accel_y;
    max_curr_accel_y = max_reg_accel_y;
    dy_thres_accel_y = (min_curr_accel_y + max_curr_accel_y) / 2;
    dy_chan_accel_y = (max_curr_accel_y - min_curr_accel_y);
    min_reg_accel_y = accel_y_avg;
    max_reg_accel_y = accel_y_avg;
    min_curr_accel_z = min_reg_accel_z;
    max_curr_accel_z = max_reg_accel_z;
    dy_thres_accel_z = (min_curr_accel_z + max_curr_accel_z) / 2;
    dy_chan_accel_z = (max_curr_accel_z - min_curr_accel_z);
    min_reg_accel_z = accel_z_avg;
    max_reg_accel_z = accel_z_avg;

    if (dy_chan_accel_x >= dy_chan_accel_y &&
        dy_chan_accel_x >= dy_chan_accel_z) {
        if (active_axis != 0) {
```

```
                sample_old = 0;
                sample_new = accel_x_avg;
            }
            active_axis = 0;
        } else if (dy_chan_accel_y >= dy_chan_accel_x &&
            dy_chan_accel_y >= dy_chan_accel_z) {
                if (active_axis != 1) {
                sample_old = 0;
                sample_new = accel_y_avg;
            }
            active_axis = 1;
        } else {
            if (active_axis != 2) {
                sample_old = 0;
                sample_new = accel_z_avg;
            }
            active_axis = 2;
        }

    } else if (now_time < 500) {
        if (min_reg_accel_x > accel_x_avg)
            min_reg_accel_x = accel_x_avg;
        if (max_reg_accel_x < accel_x_avg)
            max_reg_accel_x = accel_x_avg;
        if (min_reg_accel_y > accel_y_avg)
            min_reg_accel_y = accel_y_avg;
        if (max_reg_accel_y < accel_y_avg)
            max_reg_accel_y = accel_y_avg;
        if (min_reg_accel_z > accel_z_avg)
            min_reg_accel_z = accel_z_avg;
        if (max_reg_accel_z < accel_z_avg)
            max_reg_accel_z = accel_z_avg;
    }

    sample_old = sample_new;
    switch (active_axis) {
        case 0:
            if (accel_x_avg - sample_old > step_size ||
                accel_x_avg - sample_old < -step_size) {
                sample_new = accel_x_avg;
                if (sample_old > dy_thres_accel_x &&
                    sample_new < dy_thres_accel_x) {
                    step_count++;
                    step_changed = 1;
                }
            }
            break;
        case 1:
            if (accel_y_avg - sample_old > step_size ||
                accel_y_avg - sample_old < -step_size) {
```

```
                        sample_new = accel_y_avg;
                        if (sample_old > dy_thres_accel_y &&
                            sample_new < dy_thres_accel_y) {
                            step_count++;
                            step_changed = 1;
                        }
                    }
                    break;
                case 2:
                    if (accel_z_avg - sample_old > step_size ||
                        accel_z_avg - sample_old < -step_size) {
                        sample_new = accel_z_avg;
                        if (sample_old > dy_thres_accel_z &&
                            sample_new < dy_thres_accel_z) {
                            step_count++;
                            step_changed = 1;
                        }
                    }
                    break;
            }

            if (step_changed) {
                ESP_LOGI(mpu6050_get_tag(), "Step Counter: %d", step_count);

                FILE* file_stepcount = fopen("/spiffs/stepcount.csv", "wa");
                if (file_stepcount == NULL) {
                    ESP_LOGE(mpu6050_get_tag(), "Failed to open stepcount.csv.");
                    return;
                }

                fprintf(file_stepcount, "%d,", step_count);
                fclose(file_stepcount);
                step_changed = 0;
            }

            vTaskDelay(10 / portTICK_PERIOD_MS);
        }
    }
}


void check_spo2()
{
  bufferLength = 100; //buffer length of 100 stores 4 seconds of samples running
at 25sps

  //read the first 100 samples, and determine the signal range
  for (byte i = 0 ; i < bufferLength ; i++)
  {
    while (particleSensor.available() == false) //do we have new data?
```

```
    particleSensor.check(); //Check the sensor for new data

    redBuffer[i] = particleSensor.getRed();
    irBuffer[i] = particleSensor.getIR();
    particleSensor.nextSample(); //We're finished with this sample so move to
next sample
  }

  //calculate heart rate and SpO2 after first 100 samples (first 4 seconds of
samples)
  maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer,
&spo2, &validSPO2, &heartRate_spo2, &validHeartRate_spo2);

  //Continuously taking samples from MAX30102.  Heart rate and SpO2 are
calculated every 1 second
  while (1)
  {
    //dumping the first 25 sets of samples in the memory and shift the last 75
sets of samples to the top
    for (byte i = 25; i < 100; i++)
    {
      redBuffer[i - 25] = redBuffer[i];
      irBuffer[i - 25] = irBuffer[i];
    }

    //take 25 sets of samples before calculating the heart rate.
    for (byte i = 75; i < 100; i++)
    {
      while (particleSensor.available() == false) //do we have new data?
        particleSensor.check(); //Check the sensor for new data


      redBuffer[i] = particleSensor.getRed();
      irBuffer[i] = particleSensor.getIR();
      particleSensor.nextSample(); //We're finished with this sample so move to
next sample
      Serial.println();
      Serial.print(F(", SPO2="));
      Serial.print(spo2, DEC);
      Serial.print(F(", SPO2Valid="));
      Serial.println(validSPO2, DEC);
    }

    //After gathering 25 new samples recalculate HR and SP02
    maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer,
&spo2, &validSPO2, &heartRate_spo2, &validHeartRate_spo2);

  }
}
void loop()
```

```
{
  if (appmode == 0)
  {
    setup_hr_mode();
    while (HRCounter < 1000)
    {
      check_hr_value();
      HRCounter++;
    }
    if (HRCounter == 1000)
    {
      HRCounter = 0;
      appmode++;
        Serial.print("The last IR value=");
        Serial.print(irValue);
        Serial.print(", the last BPM=");
        Serial.print(beatsPerMinute);
        Serial.print(", Avg BPM=");
        Serial.print(beatAvg);
        if (irValue < 50000)
          Serial.print(" No finger, can't measure heart rate ");
        Serial.println();
    }
    if (appmode == 1)
    {
      setup_temp_mode();
      while (TempCounter < 31)
      {
        TempCounter++;
        currentTemp = check_temperature();
        Temp = Temp+currentTemp;
        averageTemp = Temp/TempCounter;
        //Serial.println(averageTemp);
      }
      if (TempCounter == 31)
      {

        Serial.print("Temperature = ");
        Serial.println(averageTemp);
        Temp = 0;
        TempCounter = 0;
        appmode++;
      }
    }
    if (appmode == 2)
    {
      setup_spo2_mode();
      check_spo2();
      appmode = 0;
    }  }}
```

Додаток Б

Код серверної частини комплексу

```
"""
Django settings for spo2esp32 project.
Generated by 'django-admin startproject' using Django 4.1.1.
For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/
For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/
"""
from pathlib import Path
import os
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY    =    'django-insecure-xc6t9ec%rk55x!0zv@1e*#ky1bw_y$*0cwn-
6vd(bwj&6=ytfa'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'measurementapp.apps.MeasurementappConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
CRISPY_TEMPLATE_PACK = 'bootstrap5'
STATIC_ROOT = os.path.join(BASE_DIR,'static')
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
```

```
        'django.middleware.clickjacking.XFrameOptionsMiddleware',
    ]
    ROOT_URLCONF = 'spo2esp32.urls'
    TEMPLATES = [
        {
            'BACKEND': 'django.template.backends.django.DjangoTemplates',
            'DIRS': [],
            'APP_DIRS': True,
            'OPTIONS': {
                'context_processors': [
                    'django.template.context_processors.debug',
                    'django.template.context_processors.request',
                    'django.contrib.auth.context_processors.auth',
                    'django.contrib.messages.context_processors.messages',
                ],
            },
        },
    ]
    WSGI_APPLICATION = 'spo2esp32.wsgi.application'
    # Database
    # https://docs.djangoproject.com/en/4.1/ref/settings/#databases
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': BASE_DIR / 'db.sqlite3',
        }
    }
    # Password validation
    #     https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-
    validators
    AUTH_PASSWORD_VALIDATORS = [
        {
            'NAME':
    'django.contrib.auth.password_validation.UserAttributeSimilarityValidato
    r',
        },
        {
            'NAME':
    'django.contrib.auth.password_validation.MinimumLengthValidator',
        },
        {
```

```
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/
STATIC_URL = 'static/'
# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'#urls.py
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
    path('/', include('spo2hrserver.urls')),
    path('admin/', admin.site.urls),
]
# measurementapp/models.py
from django.db import models

class DeviceList(models.Model):
    uuid = models.CharField(max_length=200, unique=True)
    hardwareId = models.CharField(max_length=200, unique=True)
    deviceDescription = models.CharField(max_length=200)

class UserList(models.Model):
    userID = models.CharField(max_length=200, unique=True)
    username = models.CharField(max_length=200)
    passwd = models.CharField(max_length=50)
    deviceID = models.CharField(max_length=200)

class Data(models.Model):
    uuid = models.CharField(max_length=200)
```

```
    timestamp = models.DateTimeField(auto_now_add=True)
    hr = models.IntegerField()
    spo2 = models.FloatField()
    temp = models.FloatField()
    steps = models.IntegerField()
class SettingsTable(models.Model):
    deviceID = models.CharField(max_length=200)
    timestamp = models.DateTimeField()
    ssid = models.CharField(max_length=200)
    password = models.CharField(max_length=50)
    bluetoothStatus = models.BooleanField(default = True)
    frequency = models.IntegerField(default = 10)
#measurementapp/urls.py
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
    path('/settings',views.settings, name='Settings')
    path('/getData',views.getData, name='getData')
    path('/login',views.login, name="Login")
    path('/logout',views.logout, name="Logout")

]
```