КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра радіотехніки та радіоелектронних систем

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Ігор АНІСІМОВ
« __ » червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«WEB-ЗАСТОСУНОК КОМПЛЕКСУ ВИМІРЮВАННЯ ВІДНОСНОГО РІВНЯ МОНООКСИДУ ВУГЛЕЦЮ У ВИДИХУ ЛЮДИНИ»

Виконав:
студент 4-го курсу
денної форми навчання
спеціальності 172 - Телекомунікації та радіотехніка
ОП «Інформаційна безпека телекомунікаційних систем і мереж»
Яшан Ігор Сергійович _____

Науковий керівник:
к.ф.-м.н., доц. Бех Ігор Іванович _____

Рецензент:
к.т.н., м.н.с. Велигоцький Дмитро Володимирович _____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри радіотехніки та радіоелектронних систем від «23» червня 2023 р., протокол № 22.

Завідувач кафедри радіотехніки та радіоелектронних систем,
доктор фіз.-мат. наук, професор
Анісімов Ігор Олексійович _____

Київ – 2023

ЗМІСТ

ВСТУП

В сучасному світі дуже важливо слідкувати за станом здоров`я людини для збереження її високої працездатності та збільшення тривалості життя. І на допомогу нам в побутовому житті приходить багато різноманітних приладів для моніторингу показників життєдіяльності людини. Наприклад пульсометри, тонометри, глюкометри та різні аналізатори газів в організмі людини [1].

Аналіз повітря дихання пацієнта відноситься до неінвазійних методів діагностики, що й викликає підвищений інтерес. З розвитком інформаційних технологій стало легше створювати нові апаратні засоби та методики визначення процентного вмісту різних газів. Один з способів діагностики є аналіз складу видихуваного повітря, яке представляє собою суміш газів різного ендогенного походження з дихальних шляхів [2]. Компоненти видихуваного повітря можуть свідчити про стан здоров’я людини. Для того щоб якось можна було фіксувати та аналізувати маркери певних газів, зокрема, монооксиду вуглецю, створені різноманітні пристрої для діагностики зовнішнього дихання. Загалом, розробка апаратури і методик для віддаленого контролю стану здоров’я людини на основі мікропроцесорної техніки сприяє підвищенню ефективності діагностики і є актуальним напрямком у медицині [2].

Наприклад, у випадках хронічної обструкції легень чи ревматоїдного артриту, куріння хворого є фактором, що впливає на результат проведених аналізів [2]. Тому перед взяттям проби хворих просять утриматися від куріння на термін 8 годин. Але, як показала статистика, частина пацієнтів ігнорує попередження, що впливає на результати. Отже важливо мати прилади, що можуть встановити, чи палили пацієнти перед проведенням процедури.

Ще один приклад використання датчиків для виявлення чадного газу у диханні був представлений в роботі [3]. У ній автор пропонує використовувати сенсори для вимірювання рівня CO у мобільних

смартфонах та персональних комп’ютерах. А саме вводити їх в конструкцію гаджетів. Важливою особливістю цих «втручань» є об'єктивна оцінка стану курця, через частий контроль вмісту чадного газу в диханні. Необхідна об’єктивна оцінка куріння, оскільки курці часто помилково класифікують себе як некурців під час спроб кинути палити. Крім того, деякі дані свідчать про те, що лише моніторинг CO в диханні може сприяти зменшенню куріння [3].

Отже задача створення портативного апаратно-програмного комплексу (з можливістю підключення до мережі Інтернет) для аналізу рівня чадного газу у видиху людини, який буде мати досить хорошу чутливість для аналізу видиху людини і міг би показати коректні данні в реальному часі є актуальною.

Враховуючи все вищезазначене під час виконання дипломної роботи бакалавра переді мною було поставлено завдання розробити веб-застосунок для відображення та первинного аналізу даних, отриманих від апаратної частини комплексу для вимірювання відносного рівня монооксиду вуглецю у видиху людини.

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ

1.1. Актуальність діагностики зовнішнього дихання

Оперативний контроль стану здоров'я та працездатності організму людини за складом видиху повітря полягає, в основному, у вимірюванні відносної концентрації CO_2, O_2 і CO. Більш інформативним є одночасні виміри й інших характеристик організму людини, таких як частота дихання, частота серцевих скорочень, вологість та температура повітря, що видихається. В роботі [2] наведено методику аналізу повітря видиху на наявність маркерних газів і наведено шляхи апаратного вирішення проблеми. Саме моніторинг концентрації CO_2, O_2 і CO, визначення об'ємів цих газів, виділених при видиху, їх співвідношення, має виконуватись в портативному інтелектуальному приладі на основі мікропроцесорної техніки з використанням електрохімічних та інших сенсорів [2].

Основною функцією комплексу, що розробляється, буде вимірювання відносного рівня монооксиду вуглецю у видиху людини. CO, або чадний газ, є одним з найбільш токсичних компонентів продуктів горіння. Чадний газ входить до складу диму і виділяється при тлінні та горінні майже всіх горючих речовин. У невеликих кількостях він навіть утворюється в організмі людини. Підступність чадного газу проявляється у тому, що він легко проходить через бар'єр легень, потрапляючи у кров, легко вступає в контакт з білком гемоглобіном. Найгірше те, що монооксид вуглецю набагато швидше та сильніше може зв'язуватися з гемоглобіном у порівнянні з киснем, витісняючи його і утворюючи досить стійку сполуку — карбоксигемоглобін. Кров при цьому втрачає здатність переносити і правильно використовувати кисень, що пошкоджує мозок та інші органи. В результаті спричиняється кисневе голодування організму в цілому, що при тривалому вдиханні великої концентрації чадного газу призводить до летального наслідку [2].

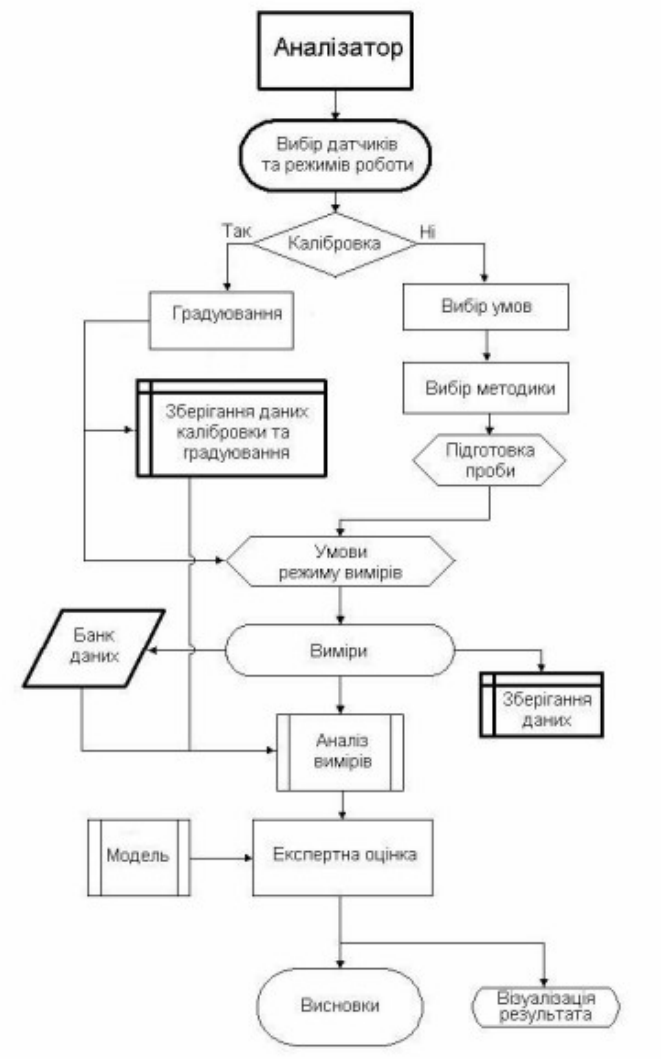

Рис. 1.1. Структурна схема загального алгоритму роботи газоаналізатора [2].

Загальний алгоритм роботи газоаналізатора можна розбити на декілька основних етапів, як це показано на рис. 1.1 [2]:

- вибір датчиків та їх калібрування, враховуючи умови режиму вимірів;
- виміри та зберігання даних;
- аналіз та візуалізація результатів через мережу.

РОЗДІЛ 2. ПРОГРАМНО-АПАРАТНА ЧАСТИНА ВЕБ-ЗАСТОСУНКУ НА БАЗІ ESP-32

2.1. Апаратна частина комплексу

У пристроях для аналізу складу повітря, особливо в галузі контролю якості повітря в медичних пристроях, апаратна частина відіграє надзвичайно важливу роль. Одним із простих рішень при створенні таких пристроїв є використання мікроконтролерів, зокрема ESP-32, та простих газових сенсорів для детектування чадного газу. В даній роботі було заплановано використати датчик MQ-7.

ESP-32 - це потужний мікроконтролер з високою продуктивністю, широкими можливостями. Його енергоефективність та великий набір функцій роблять його ідеальним вибором для реалізації портативних пристроїв для будь-якого аналізу. Головною особливістю ESP32, яка зумовила вибір саме цього мікроконтролера для виконання поставленого в роботі завдання, полягає в тому, що він має вбудований модуль WiFi, а це забезпечує простий шлях підключення до мережі Інтернет. Однак, для точного та надійного вимірювання рівня газів, іноді необхідні додаткові покращення та схемотехнічні рішення.

При створенні апаратної частини в роботі зроблено акцент на візуалізації даних. Для цього за допомогою підключення мікроконтролера до маршрутизатора, у якого є вихід до мережі Інтернет, було створено локальну мережу, що в свою чергу дає змогу для налаштування веб-застосунку, в якому можливо зробити будь-яку візуалізацію даних.

Перед початком створення програми потрібно розуміти основні складові апаратної частини та описати їх налаштування. На рис. 2.1. наведено структурну схему пристрою.

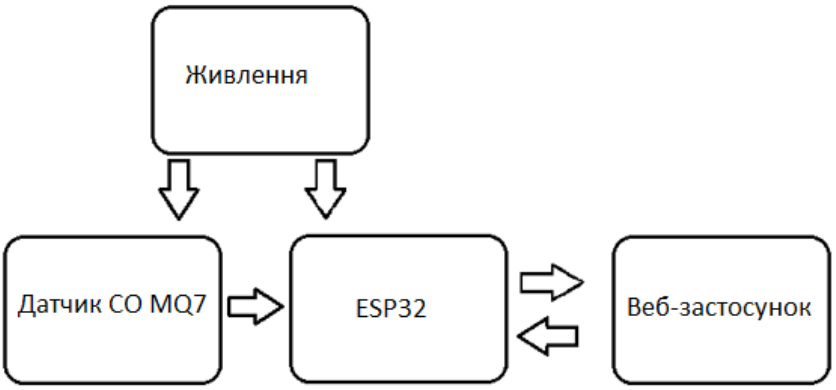

Рис. 2.1. Структурна схема створюваного програмно-апаратного комплексу.

Для створення газоаналізатора було використано мікроконтролер ESP-32 [4], побудований на мікромодулі ESP-WROOM-32 – новому мініатюрному високопродуктивному чіпі, що дозволить знімати показання з датчика чадного газу MQ7 [5] для детектування рівня чадного газу та передавати дані у веб-застосунок через стек протоколів стандартів WIFI 802.11n [6]. Зовнішній вигляд мікроконтролера ESP-32 наведено на рис. 2.2, а основні характеристики – в табл. 1.

Таблиця 1.

Напруга живлення	5В
WiFi Стандарти	FCC/CE/IC/TELEC/KCC/SRRC/NCC
Мережеві протоколи:	TCP/UDP/HTTP/FTP/MQTT
Частотний діапазон	2.4 ~ 2.5 ГГц
Робочий струм, середній	80 мА
Робочий струм піковий	500 мА
USB-UART конвертер	CP2102
Діапазон робочих температур	-40°C ~ +85°C

Рис. 2.2. Зовнішній вигляд мікроконтролера ESP-32 [4].

MQ7 – нормований датчик для детектування рівня чадного газу, проте він також має досить непогану чутливість до природного газу, побутового газу і різних зріджених попутних нафтових газів (пропан, пропілен, бутан та інші). Схематична будова датчика CO MQ7 наведена на рис. 2.3.

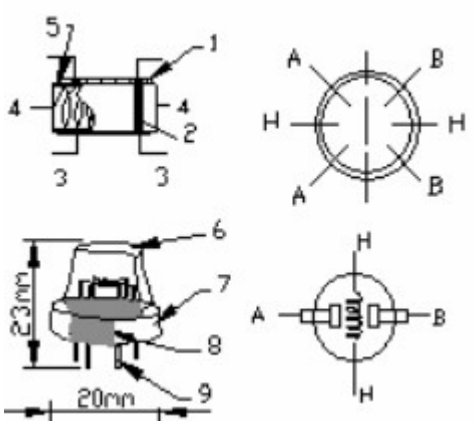

Рис. 2.3. Схематична будова датчика CO MQ7 [5].

1 – Газочутливий шар, 2 – Електрод , 3 – Електродна лінія, 4 – Нагрівальна спіраль, 5 – Керамічна трубка, 6 – Антивибухова сітка, 7 – Затискаюче кільце, 8 – Смоляна основа, 9 – Порт у вигляді трубки.

Основним робочим елементом датчика є нагрівальний елемент (4), за рахунок якого відбувається хімічна реакція, в результаті якої на аналоговий вихід надходить сигнал, пропорційний до концентрації газу [5]. Основні характеристики датчика MQ7 наведені у табл. 2.

Таблиця 2.

Напруга живлення	5 В
Струм споживання	160 мА
Час розжарення нагрівального елементу	60-90 с
Потужність нагрівача	350 мВт
Температурний діапазон	-10 - 50 °C
Діапазон чутливості	10 – 10000 ppm

2.2. Перший варіант веб-застосунку, використання протоколу FTP

За останні роки розвиток технологій підштовхує розробників розробляти свої прилади з урахуванням можливості бути складовою частиною будь-якої мережі, зокрема глобальної мережі Інтернет. У цьому контексті платформа ESP-32 виходить на передній план, завоювавши визнання своєю функціональністю та надійністю. Як уже зазначалося, одна з головних переваг ESP-32 полягає в його малому розмірі та низькому енергоспоживанні, що робить його ідеальним вибором для розробки портативних приладів. ESP-32 має вбудовану підтримку WiFi та Bluetooth, що дозволяє легко підключатись до мережі Інтернет та інших пристроїв. Крім того, ця платформа має потужний мікроконтролер та багатофункціональні порти, що дають можливість взаємодіяти з різними сенсорами, пристроями та сервісами. ESP-32 має надзвичайно широкі можливості для розробки веб-застосунків, які об'єднують в собі повноцінний серверний компонент (бекенд) і візуальний інтерфейс користувача (фронтенд). За допомогою цієї платформи можна легко збирати дані з датчиків, взаємодіяти з веб-серверами, керувати пристроями та аналізувати інформацію в режимі реального часу. У даній роботі використано підхід, при якому ESP-32 має змогу виконувати дві ролі. Перша роль надає можливості мікроконтролеру бути центральною точкою системи локальної мережі. Такий підхід дозволить користувачеві відразу почати комунікацію з програмною

складовою. Друга роль дозволяє бути частиною іншої локальної мережі та підключатись до інших маршрутизаторів , які мають доступ до інтернету.

Також платформа ESP-32 дозволяє розробникам користуватись ефективними інструментами розробки, які заощаджують час та надають готові рішення у написанні драйверів , які потрібні для мікроконтролера ESP-32, а саме всі можливі функції для стартового файлу та функції ініціалізації для певних периферійних пристроїв. Найпоширеніший інструмент розробки є Arduino IDE. Саме в цьому середовищі є безліч можливостей та готових бібліотек , які допомагають зосередитись лише на написанні логіки.

В даній роботі потрібно було зосередитись на розробці веб-додатку, який би мав змогу в режимі реального час відтворювати інформацію у зручному вигляді. Інформація повинна збиратись з сенсорів підключених до мікроконтролера та передаватись на сам веб-застосунок. Також дуже важливою задачею було написання коду для самого веб-додатку, за допомогою веб-технологій таких як html, css та javaScript, та обдумати дизайн, який би був водночас простим для користування та повноцінним для отримання повноти картини з зібраних даних.

Першим кроком до створення будь-якого веб-додатку є написання коду для html сторінки. В даній роботі у першому наближенні було використано примітивний та базовий концепт блочної системи, де кожен блок є складовою більш великого блоку. Також було використано семантичний підхід, що полягає у використані семантичних конструкцій таких як header, footer, form та інші блоки, які формують всю html-сторінку та дозволяє уникати багато повторів однотипних блоків div. Така технологія має назву BEM-технологія, що використовується в сучасних веб-застосунках та допомагає програмісту писати більш чіткий та структурований код, який легко підтримувати та покращувати. На рис. 2.4 зображено приклад, який виділяє головні семантичні html теги.

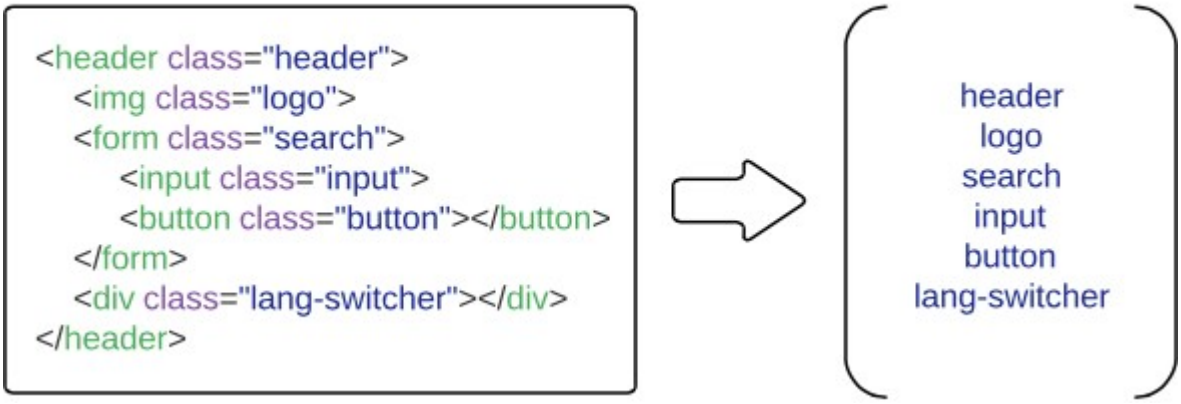

Рис. 2.4. Структура html-тегів у вигляді BEM-технології.

Також при створенні html коду було використано певний css-код, що надав html-сторінці більш чітку структуру. CSS властивості були використані для певних тегів, які зберігались в атрибуті class та мали конкретну назву, що описує сам тег. Оскільки основною задачею є візуалізація даних, постало питання як найкраще їх представити. Було прийнято рішення про створення графіку з двома осями, де одна вісь представлятиме час та інша представлятиме значення, отримані з датчика. Для того щоб створити такий графік та спостерігати його зміну його в реальному часі, потрібно було використати певні бібліотеки, написанні мовою javaScript та модифікувати існуючий код для своїх потреб. В кінцевому результаті вийшла html сторінка, фото якої зображено на рис. 2.5, на якій відображено графік з тестовими точками.

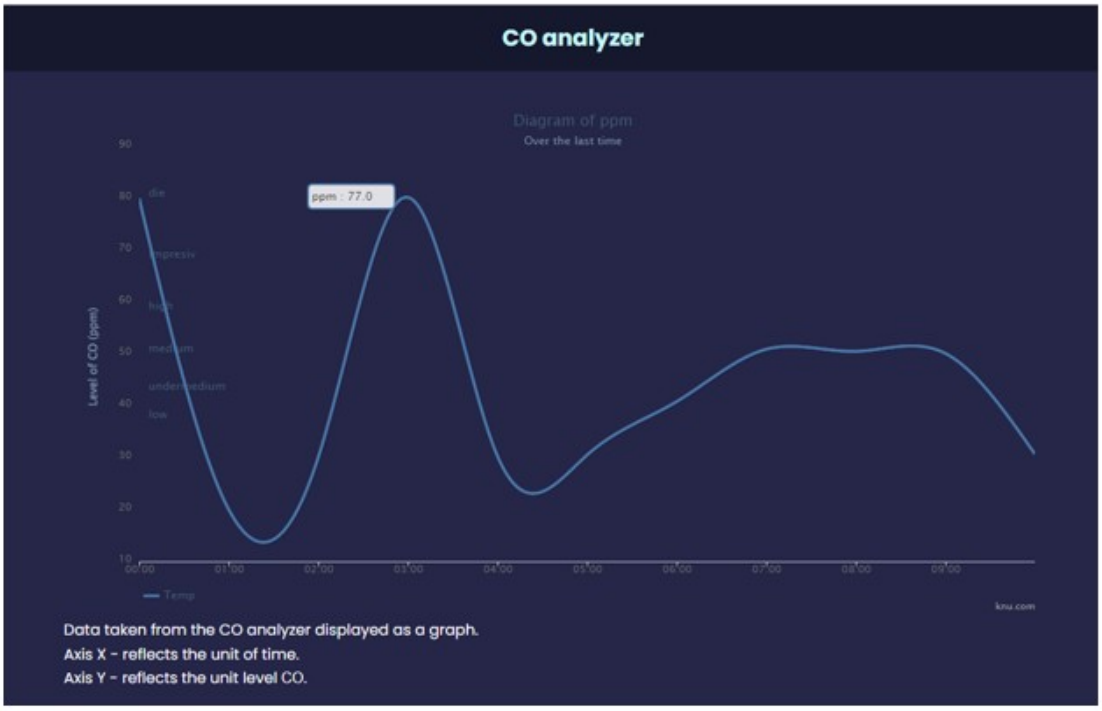

Рис. 2.5. Фото html-сторінки, побудованої із використанням BEM-технології.

Наступним кроком після створення фронтенд частини потрібно було реалізувати серверну логіку та налаштувати вбудований WiFi модуль. Оскільки для обміну даними між апаратною та програмною частинами створюваного пристрою спочатку було обрано протокол FTP, потрібно було реалізувати FTP сервер та обрати сторонній клієнт, який би надав можливість транспортувати файли та отримувати їх на боці клієнта. Код, який потрібно було написати для підтримки WiFi модуля, є досить простим та однотипним, його основне завдання полягає в тому, щоб використати функцію ініціалізації та задати пароль та назву точки доступу. Єдина проблема, яка постала в такій реалізації – це обмін даними між апаратною та програмною частинами пристрою в реальному часі, бо протокол FTP – це протокол, основною задачею якого є передачі файлів по мережі, дуже складно реалізувати канал зв’язку який би передавав інформацію досить швидко та без зайвих накладних процесів, таких як обробка цих даних та створення певного

функціоналу для обробки цих даних на боці клієнту. Тому було прийнято рішення використати протокол HTTP.

2.4. Другий варіант веб-застосунку, використання протоколу HTTP

В наступному варіанті фронтенд частина зазнала незначних змін та в більшій мірі просто виступала як експериментальна реалізація інтерфейсу, щоб зрозуміти, як користувач буде реагувати на певні кольори та іншу додану інформацію. Щодо налаштування WiFi все залишилось без змін. А от стосовно серверної частини, то змінився підхід до реалізації самого серверу. В більшості випадків веб-застосунки використовують бекенд на основі http-сервера, тому було потрібно використовувати специфічні методи розробки для http-сервера, наприклад, створення http-запитів. Під час написання першого варіанту http-серверу був використаний підхід з використанням підключенням клієнту та обробкою запиту через один стандартний потік символів, ціла html-сторінка генерувалась як один монолітний текст для відправки в браузер, тобто код для фронтенду формувався через аргумент функції client.println(), яка приймає лише текст як аргумент. Такий підхід для написання коду веб-застосунку унеможливлював підтримку програми в довгостроковій перспективі та робив практично неможливим додавання складного функціоналу, що зображено у лістингу 2.1. Проте цього разу отримано чітке розуміння про взаємодію клієнта з сервером та отримано комунікацію в реальному часі.

```
while (client.connected()) {
    if (client.available()) {
      char c = client.read();
      Serial.write(c);
      header += c;
      if (c == '\n') {
        if (currentLine.length() == 0) {
```

```
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-
width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            client.println("<style>html { font-family: Helvetica; display: inline-block;
margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #4CAF50; border:  none; color:
white; padding: 16px 40px;");

              // ....
                  }
                   }
                     }
```

Лістинг 2.1. Формування сторінки за допомогою client.println.

РОЗДІЛ 3. ФІНАЛЬНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

3.1. Складові веб-додатку на базі ESP-32

В останній ітерації проекту було покращено практично все, крім написання коду для модуля WiFi, він залишився таким самим. Фронтенд та бекенд частини зазнали більших змін, бо серверна частина, яка була розроблена цього разу, мала декілька http-запитів та більш гнучкі можливості щодо самої логіки роботи, а сам інтерфейс став більш інформаційним.

Для того, щоб розібрати програмну частину, а також як саме вона синхронізується з апаратною частиною, потрібно розібрати всі складові програмної частини, яка більшою мірою представлена у вигляді веб-застосунку. Веб-застосунок представлений в даній дипломній роботі має такі складові: файлова система для зберігання інформації, серверна частина та фронтенд частина веб-застосунку.

Насамперед розглянемо програмну частину, яка використовується для забезпечення безперервного зв'язку та обробки даних. Одним із потужних інструментів для програмування ESP-32 та створення веб-додатків є бібліотека ESPAsyncWebServer, яка може одночасно підтримувати веб-серверні можливості та асинхронне програмування, що забезпечує стабільне підключення та обробку одночасно багатьох запитів. Варто зазначити, що веб-застосунок в базовому налаштуванні має такий функціонал: модуль WiFi підключається до маршрутизатора з виходом до мережі Інтернет, для того щоб мати змогу користуватись глобальними бібліотеками, які допоможуть покращити вигляд та функціональність додатку. Дані надходять з датчиків до мікроконтролера, проходять певну обробку та за допомогою функціоналу http-серверу надсилаються до клієнта, яким слугує веб-браузер. Дуже важливо зазначити, що на цьому етапі потрібно щоб маршрутизатор мав доступ до інтернету, так як використанні бібліотеки фронтенд частини не будуть працювати. На рис. 3.1 зображено структуру локальної мережі з

двома клієнтами, сервером та маршрутизатором, який має доступ до мережі Інтернет.

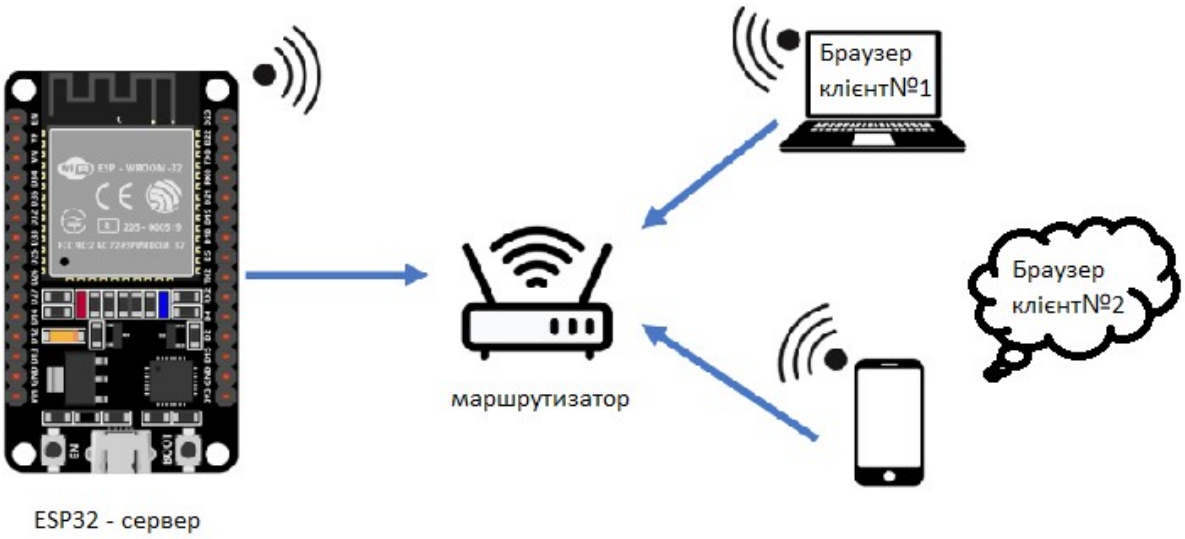

Рис. 3.1. Структура локальної мережі.

ESP-32 та інші пристрої підключається до зовнішнього маршрутизатора, таким чином створюється локальна мережа, до якої можуть підключатись різні пристрої та за локальною адресою відкривати веб-застосунок одночасно.

Не менш важливим покращенням програмної частини є використання найпростішої файлової системи SPIFFS. В цій файловій системі будуть зберігатись файли з розширенням html, які будуть передаватись із серверу на запити від клієнтів по локальній мережі. Це дозволить писати код фронтенд частини в окремому файлі, як це робилось в першому варіанті додатку при використання протоколу FTP. Файлова система представлена у вигляді додаткової папки data в середині проекту та щоб почати її використовувати, потрібно ініціалізувати її (як це показано, наприклад, в Лістингу 3.1) та виконати просте налаштування в середовищі Arduino IDE.

```
#include <SPIFFS.h>
...
```

```
if(!SPIFFS.begin()){
  Serial.println(" Error occurred :SPIFFS");
  return;
 }
```

Лістинг 3.1. Ініціалізація файлової системи SPIFFS.

Розглянемо серверну частину. Серверна частина є невід'ємною частиною веб-застосунку. Потрібно розуміти принципи роботи http-серверу, які дозволять якісно побудувати архітектуру веб-додатку. В першу чергу сама архітектурна будова будь-якого веб-додатку ділиться на різні типи, але в даній роботі використана архітектура сервер-клієнт, яка є однією з найпоширеніших архітектурних моделей у розробці бекенду. Структурна схема архітектури сервер-клієнт зображена на рис. 3.2. Вона базується на розподіленій системі, де сервер та клієнти взаємодіють між собою через мережу, надаючи послуги та отримуючи відповідні запити. У архітектурі сервер-клієнт, сервер виступає як централізована точка, яка надає послуги та обробляє запити від клієнтів. Клієнти, з свого боку, звертаються до сервера для отримання певних послуг або обробки даних. Також важливим є використання певного каналу зв'язку для комунікації між сервером і клієнтом, який може бути реалізований за допомогою різних протоколів, таких як FTP, HTTP, HTTPS та інші. У даній роботі каналом зв'язку виступає

захищений протокол гіпертекстової передачі даних – HTTPS.

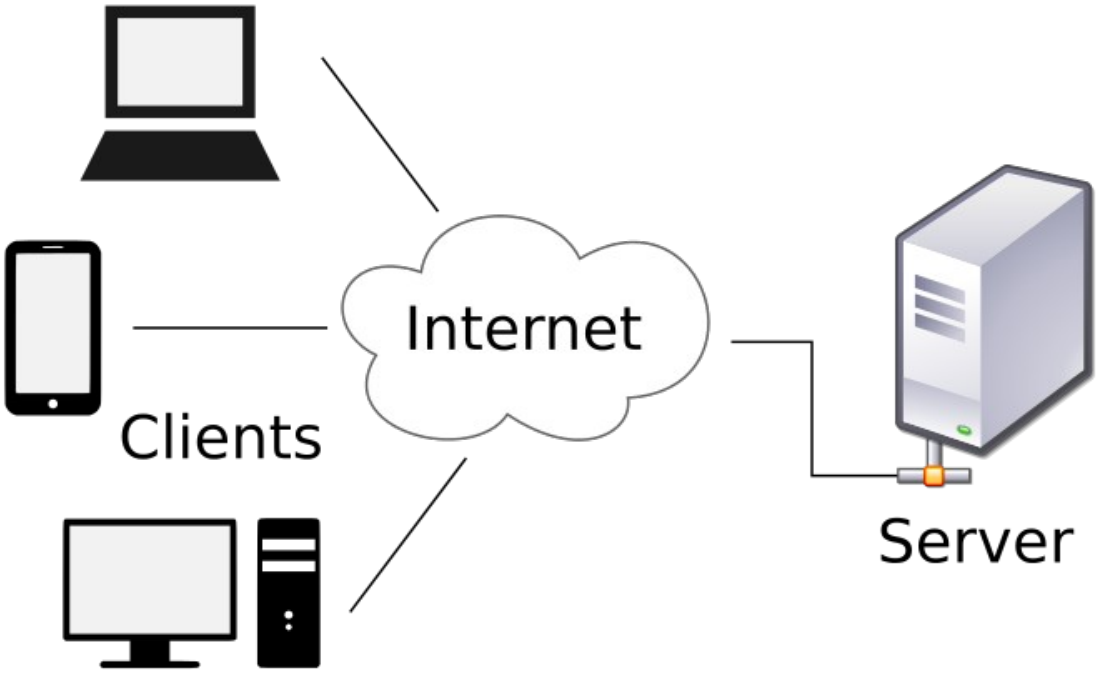

Рис. 3.2. Структурна схема архітектури сервер-клієнт.

Для того щоб клієнт мав змогу звертатись до серверу потрібно описати правила доступу до серверу, так звані http-запити, приклад яких наведено в Лістингу 3.2. Основні типи http-запитів:

- GET-запит використовується для отримання ресурсу з сервера. Клієнт надсилає запит на сервер із вказаною адресою ресурсу, і сервер повертає цей ресурс у відповіді;
- POST-запит використовується для надсилання даних на сервер для обробки. Цей метод використовується, наприклад, при відправці форми на сервер або при створенні нового ресурсу на сервері;

- PUT-запит використовується для оновлення існуючого на сервері ресурсу. Клієнт надсилає дані, які потрібно оновити, за вказаною адресою ресурсу, і сервер виконує відповідні зміни;
- DELETE-запит використовується для видалення ресурсу на сервері. Клієнт надсилає запит на видалення ресурсу за вказаною адресою, і сервер виконує відповідні дії.

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/index.html");
});

server.on("/ppm", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", read_ppm().c_str());
});
```

Лістинг 3.2. Http-запити для відправки інформації на клієнт.

В даній роботі створено два GET-запити для того, щоб відсилати всі потрібні ресурси на клієнт. Перший запит відправляє повністю html сторінку з файлової системи ESP-32, яка буде відображатись в браузері. Другий запит на відправлення даних, які обробляються в функції read_ppm. Далі потрібно буде організувати обробку даних з сенсорів та налаштувати сервер на клієнт-серверну архітектуру з асинхронним режимом. Асинхронний режим надасть змогу серверу опрацьовувати відразу декілька запитів одночасно.

Фронтенд частина веб-застосунку в даній дипломній роботі включає в себе декілька основних компонентів: HTML використовується для створення структури та розмітки веб-сторінок. Він визначає елементи такі як заголовки, параграфи, списки, таблиці та інші, що використовуються для відображення контенту на веб-сторінці. CSS відповідає за зовнішній вигляд елементів на

веб-сторінці. Він використовується для задання кольорів, шрифтів, розмірів, меж, фонів та інших стилів, що додають веб-сторінці привабливість та дизайн. Також в сучасних додатках використовується JavaScript, який дозволяє додати функціональність, динаміку до сайту та допомагає взаємодіяти з бекендом.

Розглянемо особливості фронтенд частини. Насамперед варто відзначити чітку структуру html сторінки, використання семантичних тегів та сучасного підходу до стилізації самої сторінки. За стилізацію сторінки відповідає технологія bootstrap, яка надає ряд корисних можливостей та покращень (див. Лістинг 3.3). Для того щоб почати використовувати bootstrap зверху html коду необхідно підключити певні посилання. Надалі все, що потрібно знати – це набір правил css-властивостей та скорочень цих же правил.

```
<div class="wrapper d-flex flex-column min-vh-100" >
        <header>
                <nav class="navbar navbar-expand-lg navbar-dark bg-dark ">
                <h1 class="navbar-brand mx-auto fs-1">ESP CO Analyzer</h1>
                </nav>
        </header>
</div>
```

Лістинг 3.3. Використання bootstrap технології в коді.

За допомогою такого підходу не потрібно писати всі властивості в різних файлах. Також це дозволяє відразу бачити головні css властивості певних блоків. Проте це не є єдиним правильним рішенням. Головною мотивацією використання bootstrap полягає у тому, що він має дещо покращені та готові блоки властивостей, що полегшує написання коду. Після закінчення частини роботи з структурою та стилізацією сторінки. Потрібно перейти до інтерактивності. Щоб додати інтерактивності на сайтах використовують

можливості мови програмування javaScript, яка вбудована майже у всі браузери. В розробленому веб-додатку реалізовано створення графіку, який отримує данні з серверної частини та оновлює їх згідно з заданим інтервалом часу. Розглянемо код який відповідає за комунікацію з сервером (Лістинг 3.4).

```
function createGraf() {
                const request  = new XMLHttpRequest();
                request.open('GET', "/ppm", true);
                request.send();

                request.addEventListener('load', () => {
                        if (request.status === 200) {
                                const x = getTimeZone(),
                                        y = parseFloat(request.responseText);
                                        myArray.push(y);
                                if(chartT.series[0].data.length > 40) {
                                        chartT.series[0].addPoint([x, y], true, true,
true);
                                } else {
                                        chartT.series[0].addPoint([x, y], true,
false, true);
                                }}

                                else {
                                console.log( 'Error...');
                        }
                })
        }
```

Лістинг 3.4. Використання javaScript для створення динамічного графіку.

Функція createGraph виконує запит до сервера на отримання даних та реалізовує побудову графіка. Також для того, щоб створити запит на сервер використовується XMLHttp запит, що є застарілим підходом, проте надійним з точки зору працездатності. В той час як триває запит на отримання даних, виконується обробка об’єкту ChartT, який знаходиться в бібліотеці hartsJs. За допомогою такого об’єкта дуже просто створювати певні графічні моделі. У випадку розробленого застосунку найкращим вибором було створити графік залежності значення концентрації чадного газу у ppm від часу вимірювання.

Функціонал даного веб-застосунку для проведення вимірювань передбачає кнопки, які відповідатимуть за старт, зупинку та припинення вимірювання. В кінці кожного вимірювання буде з’являтись додаткова інформація щодо поточного вимірювання: максимальне та мінімальне значення концентрації чадного газу, середнє значення за проміжок вимірювання та за останні 10% часу. Після написання всього коду отримано веб-сторінку, яка зображена на рис. 3.3, з динамічною зміною графіка та інформацією про сам графік.

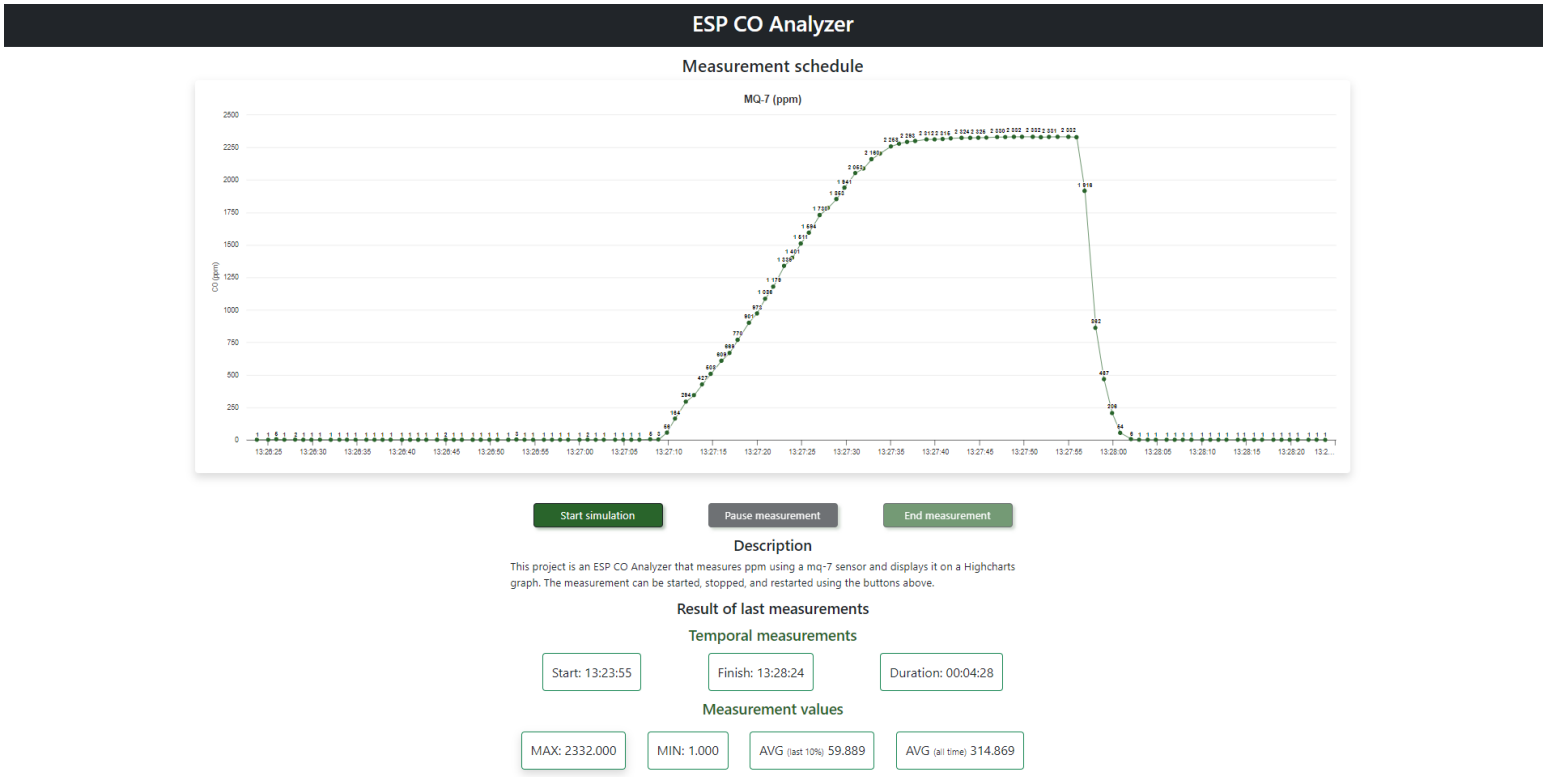

Рис. 3.3. Фінальний вигляд веб-сторінки.

ВИСНОВКИ

В результаті виконання завдання, поставленого в даній роботі, створено повноцінний, закінчений та робочий веб-застосунок для відображення та первинного аналізу даних, отриманих від апаратної частини комплексу для вимірювання відносного рівня монооксиду вуглецю у видиху людини на базі контролера ESP32.

Для подальшого розвитку проекту необхідно конкретизувати вимоги як до апаратної частини пристрою, так і до програмної, оскільки специфічні задачі можуть вимагати специфічних рішень. Наприклад, веб-застосунок дозволяв би отримувати дані з будь-якої точки планети, а не тільки в конкретній локальній мережі. Або ж написання більш стабільного та захищеного банку даних чи представлення даних у певних форматах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ходаковський М.І., Будник М.М., Кобзар Т.А., Крячок Т.В. Використання мультисенсорного газоаналізатора для тестування складу повітря при диханні люди. *Системи керування та комп'ютери*. 2019. № 3. С. 60-63с.
2. Лукаш С.І., Вакал Л.П. Розробка методики вимірів маркерних газів у повітрі дихання. *Комп'ютерні засоби, мережі та системи*. № 11. 2012. С. 1-4с.
3. Annsofi Sandberg, Johan Grunewald Assessing Recent Smoking Status by Measuring Exhaled Carbon Monoxide Levels. *National Center for Biotechnology Information*. 2011. 3-5с.
4. Технічна документація до мікроконтролера ESP32 URL: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf. (дата звернення 10.01.2023)
5. Технічна документація датчик детектування рівня чадного газу MQ7 URL: https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf. (дата звернення 10.11.2022)
6. Лукаш С.І., Будник М.М., Фролов Ю.О., Вакал Л.П., Лукаш Л.Л. Прилад для діагностики зовнішнього дихання. *Комп'ютерні засоби, мережі та системи*. 2016. № 15. С. 24-25.
7. Andrew S. Tanenbaum. «Computer Networks». 960с.
8. What is HTTP URL: https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/#:~:text=HTTPS%20is%20HTTP%20with%20encryption,far%20more%20secure%20than%20HTTP (дата звернення 22.04.2023)
9. Марійн Гавербеке. « Eloquent JavaScript, 3rd Edition: A Modern Introduction JS». 451с.

ДОДАТОК 1

```
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>

const char* ssid_name = "";
const char* password_name = "";
AsyncWebServer server(80);

String read_ppm() {
  long t = analogRead(2);

  if (isnan(t)) {
    Serial.println("Failed read sensor!");
    return "";
  }
  else {
    Serial.println(t);
    return String(t);
  }
}


void setup(){

  Serial.begin(115200);

  bool status;

  // Initialize FS
  if(!SPIFFS.begin()){
```

```
  Serial.println(" Error  occurred while mounting SPIFFS");
  return;
 }

 // Initialize  WiFi
 WiFi.begin(ssid_name, password_name);
 while (WiFi.status() != WL_CONNECTED) {
  delay(2000);
  Serial.println("Connecting to WiFi..");
 }

 //IP Address
 Serial.println(WiFi.localIP());

 server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/index.html");
 });

 server.on("/ppm", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", read_ppm().c_str());
 });

 server.begin();
}

void loop(){}
```