Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)
Аналіз та візуалізація спектру вхідного звукового сигналу засобами мікроконтролера Raspberry Pi Pico
(тема)

Виконав: здобувач 2025 року навчання,
групи КІУКІ-21-8
Зінов'єв М. С.
(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник ст. викл. Шевченко О. Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С. В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ___ _____
(підпис)

«07» 05 20 25 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Зінов'єву Михайлу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз та візуалізація вхідного звукового сигналу засобами мікроконтролера Raspberry Pi Pico
затверджена наказом університету від 21 05 2025 р. № 403
2. Термін подання здобувачем роботи до екзаменаційної комісії ____ 06 2025 р.
3. Вихідні дані до роботи
Мікроконтролер Raspberry Pi Pico
Середовище розробки Visual Studio Code
Мова програмування C
Електретний мікрофон
Аналоговий вхід
Світлодіодна матриця 32x8
4. Перелік питань, що потрібно опрацювати в роботі
Аналіз предметної області та постановка задачі проєктування
Розробка структурної схеми пристрою
Вибір компонентів та апаратна реалізація схеми
Розробка програмного забезпечення
Тестування пристрою

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 15 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання, узгодження теми	05.05.2025 – 07.05.2025	
2	Аналіз предметної області, постановка завдання	07.05.2025 – 12.05.2025	
3	Вибір інструментальних засобів та розробка структурної схеми	12.05.2025 – 15.05.2025	
4	Збірка апаратної частини на макетній платі	15.05.2025 – 17.05.2025	
5	Програмування пристрою	17.05.2025 – 23.05.2025	
6	Тестування розробленої системи	23.05.2025 – 26.05.2025	
7	Оформлення пояснювальної записки	26.05.2025 – 10.06.2025	
8	Перевірка виконаного проєкту керівником	10.06.2025 – 12.06.2025	
9	Захист роботи	12.06.2025 – 20.06.2025	

Дата видачі завдання _07_ __травня__2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст.викладач Шевченко О.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 44 сторінок, 15 рисунків, 11 джерел за переліком посилань.

СПЕКТРАЛЬНИЙ АНАЛІЗ, АУДІОСИГНАЛ, МІКРОКОНТРОЛЕР, RASPBERRY PI PICO, МУЗИЧНА ВІЗУАЛІЗАЦІЯ, WI-FI, ВЕБСЕРВЕР, ДИСТАНЦІЙНЕ КЕРУВАННЯ

Метою даної роботи є розробка мікроконтролерної системи для спектрального аналізу та візуалізації звукового сигналу, зокрема музичних композицій.

Основою для реалізації є мікроконтролерна плата Raspberry Pi Pico, побудована на мікроконтролері RP2040.

Побудовано функціональну схему проєкту, розроблене програмне забезпечення для візуалізатора, а також для вебсервера дистанційного керування. Проведено детальне тестування готового пристрою.

ABSTRACT

Bachelor’s thesis contains 44 pages, 15 figures, 11 sources according to the list of links.

SPECTRUM ANALYSIS, AUDIO SIGNAL, MICROCONTROLLER, RASPBERRY PI PICO, MUSIC VISUALIZATION, WI-FI, WEB SERVER, REMOTE CONTROL

The purpose of this work is to develop a microcontroller system for spectral analysis and visualization of an audio signal, in particular musical compositions.

The basis for implementation is the Raspberry Pi Pico microcontroller board, built on the RP2040 microcontroller.

An electrical diagram of the project was built, software was developed for the computing part of the device, as well as for the remote control web server. Detailed testing of the finished device was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МК – мікроконтролер;

ІС – інтегральна схема;

АЦП – аналогово-цифровий перетворювач;

ПДП – прямий доступ до пам’яті;

FFT – швидке перетворення Фур’є;

ПЗ – програмне забезпечення;

API – програмний інтерфейс застосунку;

GPIO – порти введення-виведення загального призначення;

I2C – Inter-Integrated Circuit;

SPI – Serial Peripheral Interface;

UART – Universal Asynchronous Receiver-Transmitter;

ВСТУП

У процесі сприйняття звуку та музики найбільш активно залучені, вочевидь, органи слуху. У внутрішньому вусі людини розташовуються волоскові клітини, які перетворюють коливання від звукових хвиль на електричні сигнали, що по слуховому нерву передаються у мозок. Різні волоскові клітини реагують на різні частоти звуку, тобто наш слух заснований на своєрідній реалізації спектрального аналізу.

Спектральний аналіз звуку — це метод, який дозволяє розкласти звуковий сигнал на складові частоти. Будь-який звук складається з хвиль різної частоти, і аналіз дає змогу побачити, які саме частоти присутні у цьому звуці та з якою інтенсивністю вони звучать.

Результатом аналізу є спектр — графік, на якому по горизонталі відкладаються частоти, а по вертикалі — їх амплітуди. Це дозволяє виявити гармоніки, шуми, тембр і навіть особливості голосу або музичних інструментів. Коли ми чуємо одну і ту саму ноту, зіграну на скрипці та фортепіано, ми розрізняємо їх саме завдяки різниці у спектрах — складових частотах та їхній інтенсивності.

Звук або музика, пов’язані з динамічним зображенням, привертають нашу увагу та збільшують час прослуховування. Довгий час «побачити» музику можна було лише умовно, наприклад у танці, але з приходом цифрових технологій з’явилась можливість ефектно візуалізувати композицію у різних формах, одна з них – спектр сигналу. Окрім реалізації спектрального аналізу на мікропроцесорах комп’ютерів, можливо також реалізувати його у більш компактному вигляді. Сучасні мікроконтролерні плати дозволяють створювати

пристрої, здатні швидко оцифрувати сигнал, проаналізувати його спектр, побудувати відповідне зображення та вивести його на екран.

Спектральний візуалізатор може бути вбудованим у радіоприймач, акустичну систему, музичний центр тощо. Часто його можна побачити у системах автозвуку. Інший варіант реалізації – автономний пристрій, на зразок цифрової рамки для фото, що виступає прикрасою робочого столу або полички.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Візуалізація спектру

Візуалізатор спектру – це пристрій або програма для графічного представлення аудіосигналів у частотній області, яке зазвичай використовується для аналізу сигналів і в розважальних застосуваннях. Поширеною реалізацією є використання вертикальних смуг або стовпців, які відображають амплітуду або спектральну щільність потужності на дискретних частотних інтервалах. Ці смуги динамічно оновлюються в реальному часі, щоб відображати частотний вміст аудіосигналу. Візуальна форма вертикальних смуг забезпечує інтуїтивне відображення зв’язку між частотою та амплітудою, часто з використанням кольору, висоти та анімації для покращення сприйняття та естетичної привабливості. На рисунку 1.1 можна побачити приклад використання спектральної візуалізації

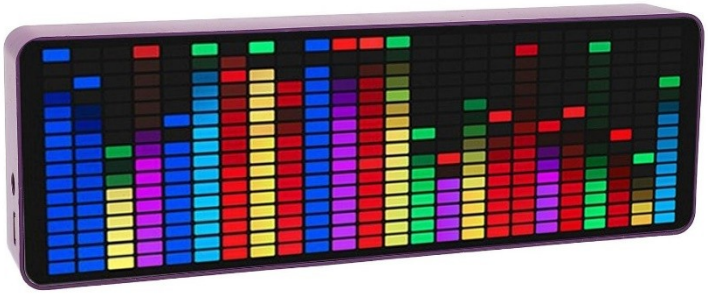

Рисунок 1.1 – Приклад використання спектральної візуалізації

Механізм роботи візуалізатора полягає у перетворенні сигналу з часового діапазону у частотний. Це досягається шляхом застосування швидкого перетворення Фур'є. Модуль кожного комплексного біну обчислюється, як правило, через квадратний корінь із суми квадратів дійсної та уявної частин, що дає амплітудний спектр.

Отримані амплітуди відображаються на дискретні вертикальні смуги, кожна з яких представляє певну частоту. Залежно від роздільної здатності FFT та частоти дискретизації аудіосигналу, частотні смуги можуть бути лінійно або логарифмічно розподілені.

У професійних застосуваннях, – моніторингу мовлення або акустичному аналізі, – спектральні візуалізатори мають високу роздільну здатність, калібровану шкалу амплітуд і точну часову інтеграцію. Вони можуть мати додаткові функції: накладення спектра, маркери гармонік і опорні криві. Споживчі реалізації, присутні у музичних центрах або радіоприймачах, надають перевагу естетичності й візуальним ефектам.

У ході кваліфікаційної роботи буде розроблений пристрій для ефектної візуалізації спектру музичних композицій. Перетворення абстрактної частотної інформації в динамічну графіку створює зв'язок між звуком і зображенням, та заохочує користувачів продовжити прослуховування.

1.2 Алгоритм швидкого перетворення Фур’є

Поширеним способом опису електричного сигналу є його представлення в часовій області, напруга або струм як функція часу. Осцилограф відображає представлення сигналу в часовій області. Інший спосіб опису сигналу - це використання його представлення в частотній області, амплітуди сигналу як функції частоти. Представлення в частотній області повинно включати

інформацію як про величину, так і про фазу, щоб повністю представити сигнал. Теорія Фур'є пов'язує представлення обох областей [12, с.2].

Швидке перетворення Фур’є (Fast Fourier Transform, FFT) є алгоритмічною оптимізацією дискретного перетворення Фур’є (Discrete Fourier Transform, DFT), що широко застосовується для спектрального аналізу цифрових сигналів, зокрема у звуковій обробці. Основна мета перетворення Фур’є – перехід від часової області представлення сигналу до частотної, що дає змогу розкласти складний сигнал на гармонічні складові. Формально, для послідовності x[n] довжини N, DFT визначається виразом

$$X[k] = \sum_{n=0}^{N-1} \left(x[n] * e^{-i\frac{2\pi}{N}kn} \right), \quad (1.1)$$

де X[k] – спектральна компонента на частоті (k/N)*fs (fs – частота дискретизації), а i – уявна одиниця.

Оскільки обчислювальна складність виразу (1.1) становить $O(N^2)$, застосування DFT до довгих сигналів або в реальному часі є обчислювально неефективним. Алгоритм швидкого перетворення Фур’є знижує складність до O(N*logN), що досягається через рекурсивне декомпонування задачі на менші підзадачі з використанням симетричних властивостей експоненціальних базисних функцій. Найпоширенішою реалізацією FFT є алгоритм Кулі–Тьюкі, ефективний у випадках, коли N є степенем двійки.

У випадку обробки звукового сигналу, що є функцією часу s(t), процес починається з дискретизації цього сигналу з частотою fs, внаслідок чого утворюється скінченна послідовність s[n] = s(nT), де T = 1/fs – період дискретизації. FFT застосовується до такого набору даних для отримання

спектру, в якому кожна компонента X[k] описує амплітуду та фазу синусоїди певної частоти, яка входить до складу сигналу. Результати FFT є комплексними числами, модуль яких визначає амплітуду відповідної частотної складової. Частотна роздільна здатність такого аналізу дорівнює $\Delta f = fs/N$, що встановлює мінімальний відступ між сусідніми частотними бінами.

Під час аналізу реального звукового сигналу, який не є періодичним або нескінченним, необхідно працювати з обмеженим часовим вікном, що призводить до виникнення ефекту спектрального витікання (spectral leakage), викликаного раптовими переходами на межах вікна. Щоб мінімізувати цей ефект, до сигналу застосовують віконну функцію, яка згладжує значення сигналу на межах вікна, зменшуючи високочастотні артефакти в спектрі.

Серед віконних функцій найчастіше використовуються вікна Хеммінга, Ханна, Блекмана та Кайзера. Застосування вікна впливає на частотну роздільну здатність, що вимагає точного підбору віконної функції відповідно до задачі.

Для аналізу нестаціонарних сигналів, до яких належить більшість звуків мовлення, музики чи довкілля, FFT зазвичай застосовується у вигляді короткочасного перетворення Фур’є (Short-Time Fourier Transform, STFT). В цьому випадку сигнал розбивається на перекривані вікна фіксованої довжини, до кожного з яких окремо застосовується FFT. Таким чином формується двовимірне представлення сигналу – спектрограма, де одна вісь відповідає часу, інша – частоті, а значення амплітуди кодуються інтенсивністю або кольором.

FFT є основою багатьох алгоритмів роботи зі звуком, зокрема розпізнавання мовлення, виявлення тону, музичного синтезу, шумозаглушення, а також візуалізації спектру звуку в спектроаналізаторах. На практиці результати FFT часто обробляються додатково, наприклад, через логарифмічне масштабування амплітуди. Важливим також є нормування спектра, компенсація амплітуд віконної функції та оцінка енергетичних характеристик, таких як

середньоквадратичне значення, спектральна щільність потужності або спектральний центр.

1.3 Постановка задачі

Мета роботи полягає у проєктувані, збірці та програмуванні пристрою для візуалізації звукового спектру з дистанційним та недистанційним налаштуванням параметрів. Пріоритетом у роботі пристрою є ефектна візуалізація музичних композицій, а не проведення вимірювань, тож необхідно врахувати відповідний частотний діапазон, зручність використання, час відгуку та обробки натискань.

Для досягнення поставленої мети необхідно виконати наступні кроки:

– детально описати структуру апаратної та програмної частин проєкту;

– визначити оптимальні параметри для дискретизації сигналу та аналізу спектру: частоту дискретизації, кількість семплів, час одного кадру, час утримання значень частоти на дисплеї, швидкість інтерфейсних шин, конфігурацію зсуву аналогового сигналу;

– розробити механізм усунення шумів при візуалізації;

– побудувати вебсторінку з панеллю керування, визначити оптимальну кількість запитів до сервера;

– провести тестування з багатьма ітераціями відлагодження та усунення помилок.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ПРОЄКТУ

2.1 Розробка структурної схеми візуалізатора спектру

При роботі над апаратним забезпеченням необхідно спочатку створити схеми різних типів та ступенів деталізації.

Структурна схема (рисунок 2.1) описує загальні функціональні частини пристрою та взаємозв’язки між ними. Ключовим елементом нового виробу є мікроконтролерна плата, яка буде виконувати функції оцифрування та обробки сигналу, побудови зображення а також керування зовнішніми модулями.

Для спектрального аналізу звукових хвиль необхідно побудувати схеми подання, які перетворять вхідний сигнал у прийнятну для МК форму. Усі способи подання сигналу мають спільне призначення, тож на схемі вони об’єднані у один функціональний блок.

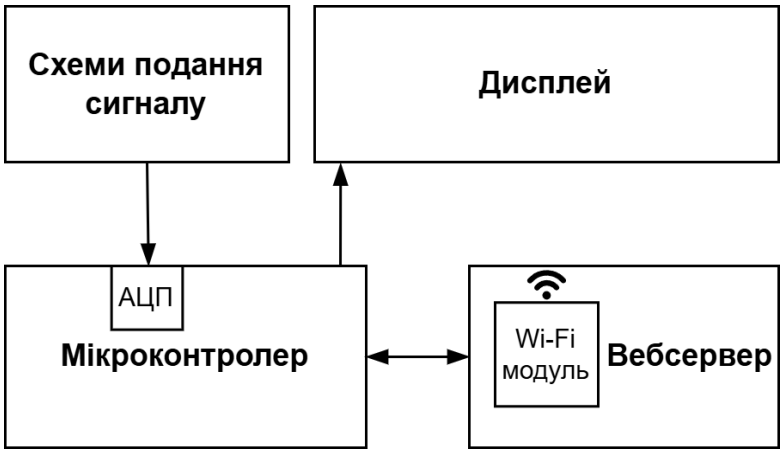

Рисунок 2.1 – Структурна схема пристрою

Сформоване контролером зображення надходить у модуль дисплея, де буде відтворене у вигляді набору точок або пікселів.

Для реалізації дистанційного керування пристроєм засобами вебсторінки, всередині пристрою повинен бути модуль, що забезпечує доступ до вебсайту у локальній мережі та обробляє запити вебклієнтів. Обмін даними між основним МК та вебсервером здійснюється у обидві сторони.

Фізична схема з'єднань пристрою на макетній платі представлена на рисунку 2.2. За даною схемою буде будуватися фізичний макет, втім розташування деяких з'єднань може змінитися. Детальний опис зображених модулів та часткових електричних схем буде наведено далі у цьому розділі.

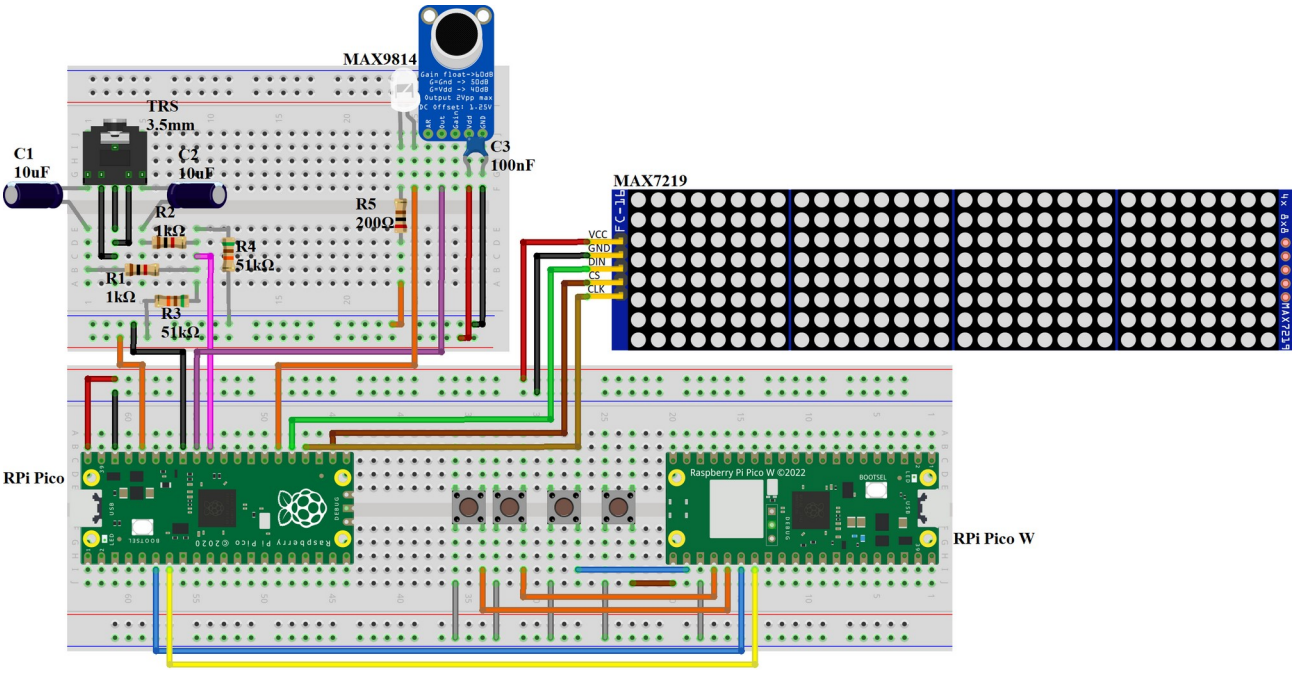

Рисунок 2.2 – Фізична схема з'єднань пристрою

2.2 Raspberry Pi Pico

Ключовою функцією проєктованого пристрою є спектральний аналіз за допомогою FFT у реальному часі, тому важливою задачею розробника є вибір

надійних компонентів достатньої потужності. Якщо при виборі аудіокомпонентів головним фактором є висока чутливість при мінімальному значенні шумів на виході, то для плати мікроконтролера існує низка вимог до характеристик:

- відносна дешевизна;
- високочастотне ядро (або ядра);
- наявність АЦП з високою роздільною здатністю;
- реалізація механізму ПДП для швидшого оцифрування сигналу;
- наявність шинних інтерфейсів для взаємодії із зовнішніми пристроями.

У результаті дослідження та порівняння наявних на ринку МК, у своєму ціновому сегменті кращим рішенням виявився контролер Raspberry Pi Pico.

Raspberry Pi Pico — це мікроконтролерний модуль, заснований на мікросхемі RP2040, розробленій Raspberry Pi Foundation. Чип RP2040 побудований на двоядерному процесорі ARM Cortex-M0+ з тактовою частотою до 133 МГц. Мікросхема має 264 КБ статичної оперативної пам’яті (SRAM) та не містить вбудованої флеш-пам’яті, натомість використовує зовнішній модуль обсягом 2 МБ.

12-бітний АЦП у Raspberry Pi Pico здатний працювати з напругою від 0 до 3.3 В. Він має чотири входи, з яких три (ADC0, ADC1, ADC2) виведені на зовнішні піни, а один (ADC4) підключений до внутрішнього температурного сенсора. Теоретична роздільна здатність становить 4096 рівнів квантування, що дозволяє проводити точні вимірювання аналогових сигналів для потреб моніторингу, керування (рисунок 2.3).

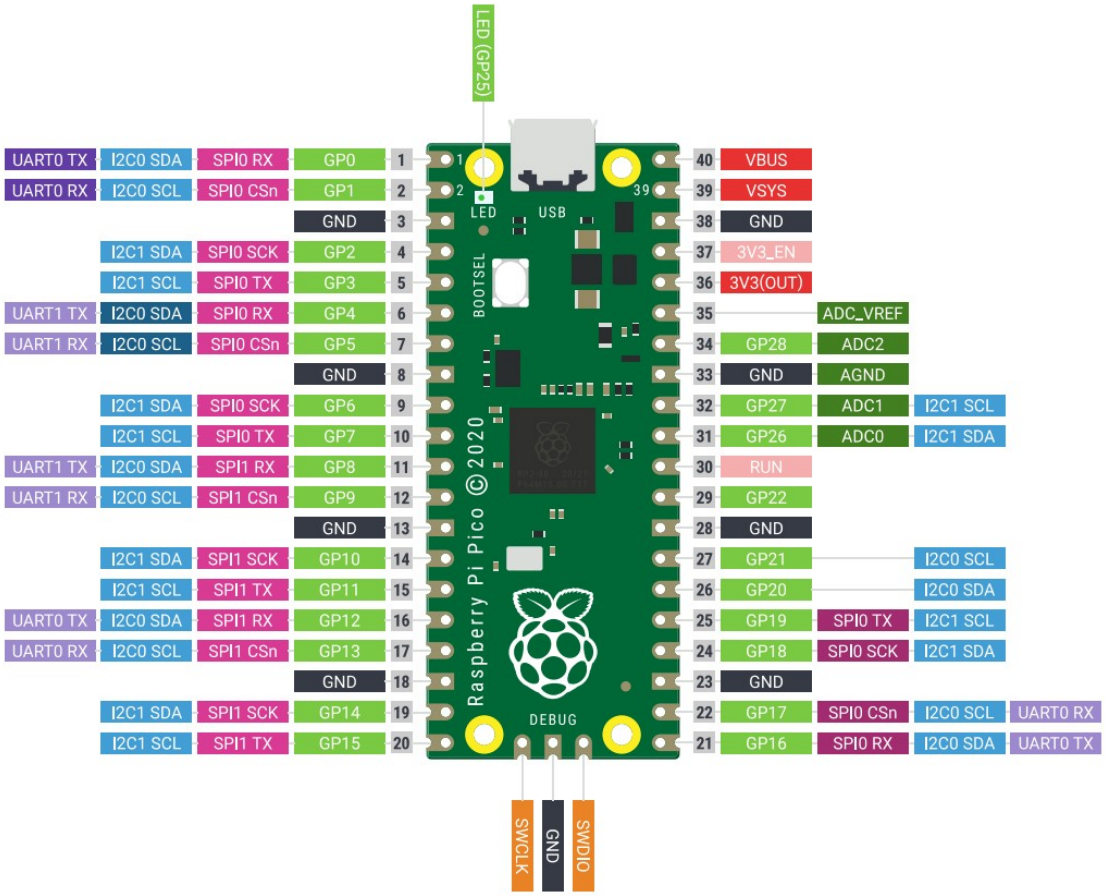

Рисунок 2.3 – МК Raspberry Pi Pico [9]

RPi Pico має розвинену систему периферійних шинних інтерфейсів. Контролер RP2040 підтримує інтерфейси SPI, I²C та UART, по два контролери на кожен інтерфейс. SPI працює в режимах master/slave та підтримує апаратну передачу даних зі швидкістю до 62.5 Мбіт/с, що дає змогу використовувати його для високошвидкісного обміну з дисплеями, зовнішніми ЦАП, сенсорами тощо. Інтерфейс I²C має 7-розрядну адресацію, що дозволяє підключати до 127 периферійних пристроїв: модулів RTC, пам’яті EEPROM, LCD-дисплеїв та інших. UART реалізує класичний двонаправлений асинхронний обмін даними, з підтримкою апаратного контролю потоку та програмованою швидкістю

передавання. Можливе використання UART для виведення відлагоджувальної інформації.

Крім стандартної периферії, мікроконтролер має два блоки PIO (Programmable I/O), кожен з чотирма програмованими машинами станів. Ці блоки дозволяють реалізовувати нестандартні протоколи, генерувати точні часові сигнали. PIO суттєво розширює можливості мікроконтролера щодо взаємодії з апаратурою, оскільки працює незалежно від основного процесора, розвантажуючи ядро від обробки задач, час виконання яких є критичним.

2.3 Модуль дисплея

Модуль з 4-х точкових дисплеїв зі світлодіодами червоного кольору, побудований на базі мікросхеми MAX7219, має загальну роздільну здатність 32x8 точок.

MAX7219/MAX7221 — це компактні драйвери дисплеїв із послідовним введенням/виведенням, які створюють інтерфейс між мікроконтролером та 7-сегментними числовими дисплеями, гістограмними дисплеями або 64 окремими світлодіодами[6]. Вбудовані мікросхеми включають дешифратор двійково-десяткового коду, схему мультиплексного сканування, драйвери сегментів та розрядів, а також статичну оперативну пам'ять 8x8, яка зберігає кожен розряд. За рахунок каскадування декількох драйверів можливо збільшити роздільну здатність модуля (рисунок 2.4).

Інформацію в кожному розряді можна оновити без необхідності оновлення всіх розрядів матриці. Мікросхема має режим сну із запам'ятовуванням інформації, аналогове та цифрове управління яскравістю підключених індикаторів та тестовий режим, що вмикає всі LED-сегменти.

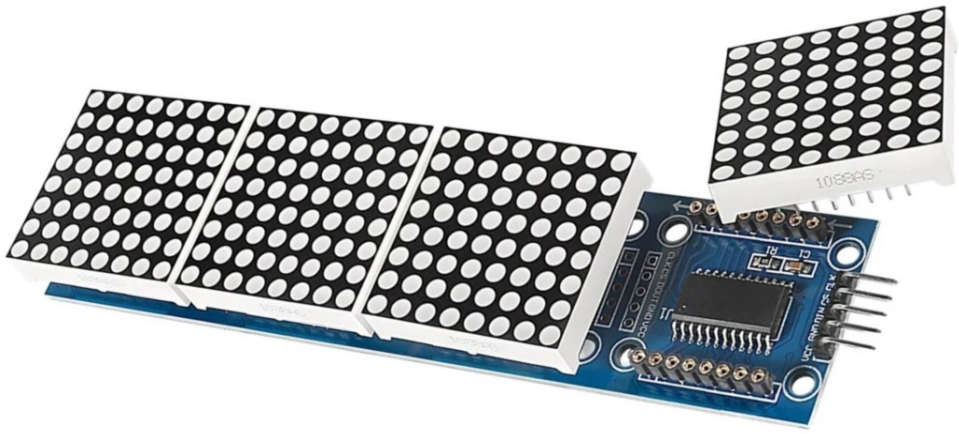

Рисунок 2.4 – Модуль світлодіодної матриці

2.4 Модуль мікрофона та аналоговий вхід

У якості стандартного засобу введення звуку використовується вбудований модуль мікрофона. MAX9814 – це недорогий, високоякісний мікрофонний підсилювач з автоматичним регулюванням підсилення (АРП) та малошумним зміщенням мікрофона. Пристрій оснащений попереднім підсилювачем, підсилювачем зі змінним коефіцієнтом посилення (VGA), вихідним підсилювачем, генератором напруги зміщення мікрофона та схемою керування АРП (рисунок 2.5)[7].

Існує кілька опцій підсилення, що встановлюються шляхом замикання контакту Gain, проте це не використовується у даній роботі.

Між пристроями, з’єднаними дротом, аудіосигнал зазвичай передається у формі змінного струму, що коливається відповідно до звукової хвилі. АЦП Raspberry Pi Pico працює у діапазоні напруги 0В - 3.3В, сигнал що виходить за його межі може пошкодити пристрій.

Рисунок 2.5 – Електретний мікрофон з підсилювачем MAX9814

Щоб запобігти цьому та отримати коректні дані після дискретизації, потрібно «змістити» хвилю напруги. Для цього використовується наступна схема:

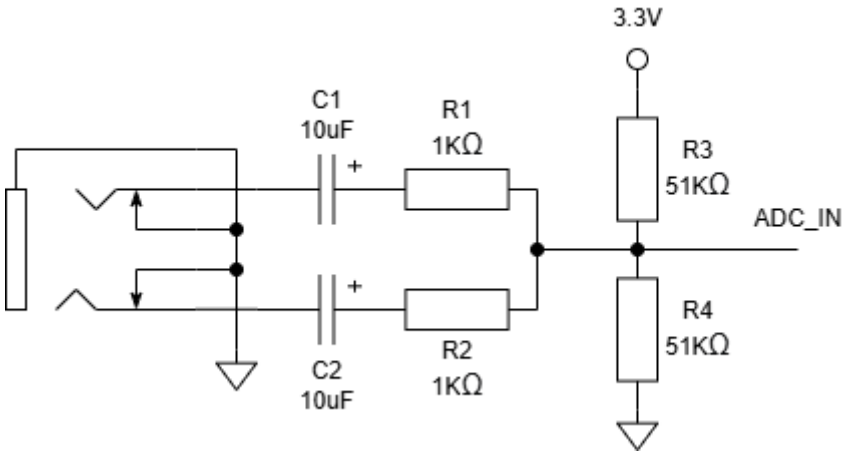

Рисунок 2.6 – Схема постійного зміщення вхідного сигналу

Вхідний сигнал на рисунку 2.6 забезпечений портом TRS 3.5мм, обидва канали об’єднується, результуючий сигнал подається на дільник напруги. Конденсатори C1, C2 та резистори R1, R2 у цій схемі захищають канали один

від одного та від джерела постійного струму, блокуючи потік струму у зворотньому напрямі, тобто до джерела аудіосигналу.

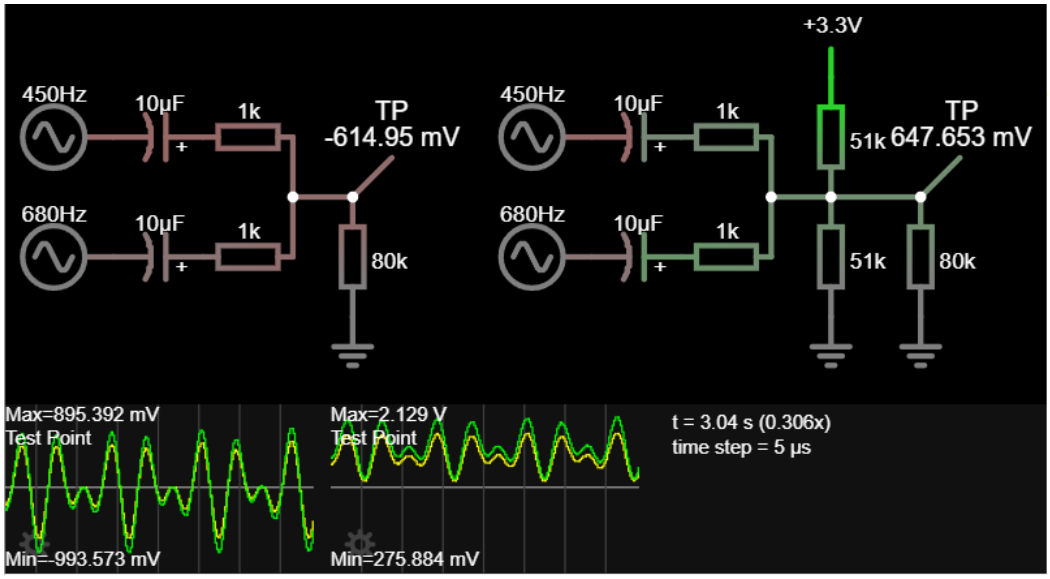

Рисунок 2.7 – Сигнали зі зміщенням (праворуч) та без нього

На рисунку 2.7. представлено сигнали зі зміщенням (праворуч) та без нього Обидва способи подання звукового сигналу працюють незалежно на основі різних входів АЦП. Для перемикання між ними можна скористатись спеціальною фізичною кнопкою або кнопкою на вебсторінці панелі керування.

2.5 Вебсервер для керування пристроєм

За початковим планом, реалізація базових функцій візуалізатора та керування ним, у тому числі дистанційного, будувалася на основі однієї плати мікроконтролера Raspberry Pi Pico W, що відрізняється від базової версії апаратною підтримкою бездротових технологій Wi-Fi та Bluetooth. У ході декількох тестів, була виявлена критична проблема – вебсервер, що працював

на другому фізичному ядрі МК RP2040, після серії запитів від веббраузера повністю блокував роботу пристрою. Можлива причина такої поведінки полягає у нестачі пам’яті для двох ядер. Перевірка роботи окремо серверного та обчислювального забезпечення не виявила таких проблем.

Процеси обробки та візуалізації були перенесені на інший контролер сімейства Pico першого покоління, не оснащений модулями бездротового зв’язку. Обробка запитів до вебсервера та подій натискання на фізичні кнопки залишилася на Pico W. Мультиядерність МК не була використана у проєкті. Обмін даними про оновлені параметри побудований на основі шини I2C. Зосередження обробки усього керування зі сторони користувача на одному пристрої дозволяє легко вирішити новоявлену проблему синхронізації між основним пристроєм та вебсервером. Суть цієї проблеми, а також алгоритму взаємодії двох МК буде детально розкрита у наступному розділі.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Більша частина складності даного проєкту полягає у програмуванні пристрою. Необхідно написати високоефективну програму мовою С з ручним управлінням пам’яттю, здатну швидко та безпомилково реагувати на запити користувача і, водночас, здійснювати обробку та візуалізацію спектру сигналу з високою частотою оновлення.

Для досягнення максимального контролю над системою, було вирішено використовувати лише низькорівневі програмні бібліотеки від постачальника плати МК. Це значно збільшить час розробки, проте дозволить удосконалювати та розширювати проєкт без втрати гнучкості програмних інтерфейсів.

Робота над програмою велася переважно шляхом багатьох ітерацій над кожною підзадачею, без підготовленої схеми алгоритму. Такий підхід найкраще підходить для створення проєктів з унікальним набором функцій, у яких за відсутністю аналогів не може бути документації, де чітко описані та обґрунтовані усі інженерні рішення.

За результатами розробки та аналізу мікроконтролерної програми сформована загальна схема алгоритму роботи основного МК (рисунок 3.1).

Рисунок 3.1 – Схема алгоритму роботи візуалізатора

3.1 Ініціалізація програми

Етап ініціалізації програми полягає у оголошенні масивів та окремих змінних, що знадобляться для роботи основного циклу. Після ініціалізації контексту параметрів за замовчуванням (яскравість дисплея, режим відображення та спосіб введення сигналу), він надсилається до серверу єдиний раз за увесь цикл роботи пристрою, потім дані будуть лише надходити.

При вході у основний цикл програми, першочерговим етапом є порівняння поточних параметрів із контекстом, що оновлюється сервером. Якщо один з параметрів відрізняється, виконується послідовність дій, що модифікують внутрішні змінні, у разі потреби надсилають команди на дисплей та вмикають індикацію відповідного параметру для сповіщення користувача.

Лістинг 3.1 – Обробка вхідних параметрів

```
while (1) { // початок основного циклу
    if (brightness != spv_context.brt) {
      brightness = spv_context.brt;
      set_display_brightness(brightness);

      overlay_frames = OVL_FRAMES;
      clear_buffer(overlay);
      draw_char(overlay, brightness, 26, 1);
    }

    if (render_mode != spv_context.mode) {
      render_mode = spv_context.mode;

      overlay_frames = OVL_FRAMES;
      clear_buffer(overlay);
      draw_char(overlay, 'a' + render_mode, 26, 1);
    }

    if (input_type != spv_context.inp) {
      input_type = spv_context.inp;
      if (input_type == INPUT_EXTERNAL) {
        adc_select_input(CAPTURE_CHANNEL1);
```

```
      } else {
        adc_select_input(CAPTURE_CHANNEL0);
      }
      gpio_put(INTMODE_LED_PIN, (input_type ==
INPUT_INTERNAL));
    }
...
```

3.2 Перетворення сигналу на спектр

За результатами оцифрування сигналу з використанням механізму ПДП отримуємо масив беззнакових восьмирозрядних значень, де сигнал тиші, враховуючи шуми, коливається навкого середнього значення 128. Параметром функції для перетворення Фур'є повинен бути масив дійсних значень сигналу, де середня амплітуда дорівнює нулю, тому ми зміщуємо сигнал шляхом віднімання середьного арифметичного від кожного оцифрованого семплу, після чого домножуємо на попередньо обчислений коефіцієнт вікна Ханна.

Лістинг 3.2 – Оцифрування сигналу та побудова смуг спектру

```
sample(cap_buf);
uint16_t sum = 0;
for (int i = 0; i < NSAMP; i++) {
  sum += cap_buf[i];
}
float avg = (float)sum / NSAMP;
for (int i = 0; i < NSAMP; i++) {
  fft_in[i] = ((float)cap_buf[i] - avg) * window[i];
  float amplitude = fabs(fft_in[i]);
  if (max_amplitude < amplitude) {
    max_amplitude = amplitude;
  }
}
if (timer_fired) {
  printf("MAX_AMPL = %f;\n", max_amplitude);
  is_paused = (max_amplitude < MIN_AMPLITUDE);
  max_amplitude = 0;
  timer_fired = false;
}
```

```
clear_buffer(offscreen);
if (!is_paused) {
  // compute fast fourier transform
  kiss_fftr(cfg, fft_in, fft_out);
  // any frequency bin over NSAMP/2 is aliased (nyquist
sampling theorum)
  memset(bands, 0, sizeof(bands));
  for (int i = 1; i < NSAMP / 2; i++) {
    // complex to magnitude
    fft_out[i].r =
        sqrtf(fft_out[i].r * fft_out[i].r + fft_out[i].i *
fft_out[i].i);
#if MAGN_FILTERING
    if (fft_out[i].r < MIN_MAGN) {
      fft_out[i].r = 0.0f;
    }
#endif
    bands[bin_to_band(i)] += fft_out[i].r;
  }
...
```

Для можливостей оптимізації, зменшення впливу шумів та покращення користувацького досвіду розроблена система з таймером, яка перевіряє максимальну зчитану амплітуду протягом 3 секунд. Якщо ця амплітуда не перевищує певний поріг, подальші етапи перетворення Фур’є та відображення спектру не виконуються, пристрій переходить у режим паузи.

3.3 Виведення зображення

Базові принципи візуалізації у пристрої схожі з тими, які використовуються у сучасній комп’ютерній графіці. Кадри оновлюються з високою частотою і кожен з них є унікальним, тобто будується спочатку і не містить у собі інформації з попереднього кадру. Масив з 32 байтів, кожен біт якого характеризує стан відповідної точки на матриці, побайтно передається на дисплей по шині SPI лише після повної побудови кадру, тобто увесь спектр спочатку будується у пам’яті МК і тільки потім відтворюється на екрані.

Лістинг 3.4 – Базові функції малювання на екрані

```
typedef struct Display_Buffer Display_Buffer;
struct Display_Buffer {
  uint8_t *bytes;
  int width;
  int height;
};
...
void clear_buffer(Display_Buffer *buffer) {
  memset(buffer->bytes, 0, buffer->width * buffer->height);
}
void render_buffer(Display_Buffer *buffer) {
  uint8_t spi_packet[2];
  for (int i = 0; i < buffer->height; i++) {
    cs_select();

    spi_packet[0] = CMD_DIGIT0 + i;
    for (int j = 0; j < buffer->width; j++) {
      spi_packet[1] = buffer->bytes[i * NUM_MODULES + j];
      spi_write_blocking(spi_default, spi_packet, 2);
    }
    cs_deselect();
  }
}
```

Після компонування масиву дійсних чисел bands, який репрезентує щойно зчитаний спектр смугами частот, ці смуги потрібно конвертувати у вигляд, зручний для візуалізації на матриці висотою 8 точок. Обчислюємо максимальне значення масиву смуг та довжини смуги на дисплеї. PEAK_BAND використовується для того, щоб звуки невеликої амплітуди відтворювалися короткими смугами на спектрі. Якщо найбільша смуга у масиві перевищує значення PEAK_BAND, весь спектр буде побудовано відносно нового максимуму. Використовуючи отримані дані, перекладаємо значення усіх смуг з діапазону [0; max_band] на діапазон [0; max_height] та зберігаємо їх у масиві display_bands.

Лістинг 3.5 – Побудова зображення спектру

```
float max_band = 0;
for (int i = 0; i < NUM_BANDS; i++) {
  max_band = MAX(max_band, bands[i]);
}
max_band = MAX(max_band, PEAK_BAND);
uint8_t max_height =
    (render_mode == RENDER_INWARD || render_mode ==
RENDER_OUTWARD)
        ? (MODULE_SIZE / 2)
        : MODULE_SIZE;
for (int i = 0; i < NUM_BANDS; i++) {
  uint8_t height = map(bands[i], 0, max_band, 0, max_height);
  if (display_bands[i] <= height) {
    display_bands[i] = height;
    hold_counters[i] = HOLD_FRAMES;
  } else if (hold_counters[i] > 0) {
    hold_counters[i] -= 1;
  } else if (display_bands[i] > 0) {
    display_bands[i] -= 1;
    hold_counters[i] = HOLD_FRAMES;
  }
}
```

Якщо кожен кадр буде побудований на основі нового спектру, не враховуючи дані попереднього, візуалізація динамічної музики буде надто різкою, проявлятися мерехтінням різних частин дисплею. Це погіршує візуальний ефект та враження користувача, тому бажано було надати анімації плавності. Для вирішення цієї проблеми використовується система з лічильниками кадрів (рисунок 3.2), кожен з яких відповідає своїй частотній смузі. Коли значення лічильника стає нульовим, додатні значення смуг попереднього кадру зменшуються на одиницю (смуги стають коротшими на одну точку) а лічильник перезапускається. При цьому, якщо попередні значення спектру менші за відповідні нові, вони замінюються новими значеннями, і лічильник модифікованої смуги знову перезапускається.

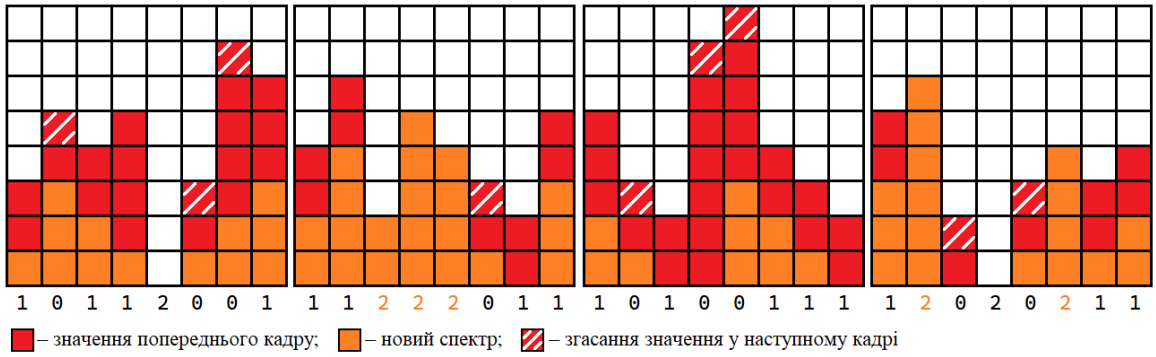

Рисунок 3.2 – Демонстрація системи плавного згасання смуг

У кінці ітерації основного циклу відбувається запис зображення у пам’ять та передача на модуль дисплея. У режимі паузи замість спектру відтворюється спеціальний символ паузи, що складається з двох вертикальних ліній. Незалежно від режиму роботи пристрою, за потреби, останнім у пам’ять записується частина так званого оверлея (від англ. “overlay” – накладка). У оверлей записуються зображення літер або цифр, що відповідають режиму відображення або рівню яскравості дисплея, після чого у основний буфер зображення копіюється потрібна частина оверлею, у даному випадку це 4-ий модуль 8x8 біт. За допомогою раніше описаної функції render_buffer увесь побудований кадр передається на дисплей з IC MAX7219. Перед початком нової ітерації циклу застосовуємо затримку для контролю частоти оновлення спектру та зображення.

Лістинг 3.6 – Кінець ітерації циклу

```
while (1) {
  ...
  if (!is_paused) {
    draw_spectrum(offscreen, display_bands, render_mode);
  } else {
    draw_pause(offscreen);
```

```
  }
  if (overlay_frames > 0) {
    overlay_frames -= 1;
    copy_display_module(offscreen, overlay, 3);
  }
  render_buffer(offscreen);
  sleep_ms(1000 / FPS);
}
```

Одним з ключових параметрів розробленої візуалізації є режим відображення. Визначено 5 режимів відображення спектру: стандартний(default), інвертований(inverse), точковий(dots), спрямований всередину(inward) та спрямований назовні(outward). Різниця між ними полягає у примітивах, що використовуються для малювання смуг спектру(лінії або точки), значенні активної смуги(1 або 0) та використанні симетрії при побудові зображення.

Лістинг 3.7 – Режими візуалізації

```
static inline void draw_spectrum(Display_Buffer *buffer,
uint8_t *bands, Render_Mode mode) {
  switch (mode) {
    case RENDER_DEFAULT: {
      for (int i = 0; i < NUM_BANDS; i++) {
        if (bands[i] > 0) {
          draw_vert_line(buffer, i,
                         (MODULE_SIZE - bands[i]),
                         (MODULE_SIZE - 1));
        }
      }
    } break;
    case RENDER_INVERSE: {
      for (int i = 0; i < NUM_BANDS; i++) {
        if (bands[i] < MODULE_SIZE) {
          draw_vert_line(buffer, i, 0,
                         (MODULE_SIZE - 1) - bands[i]);
        }
      }
    } break;
    case RENDER_DOTS: {
```

```
      for (int i = 0; i < NUM_BANDS; i++) {
        if (bands[i] < MODULE_SIZE) {
          draw_pixel(buffer, i,
                     (MODULE_SIZE - 1) - bands[i]);
        }
      }
    } break;
    case RENDER_INWARD: {
      for (int i = 0; i < NUM_BANDS; i++) {
        if (bands[i] > 0) {
          draw_vert_line(buffer, i, 0, bands[i] - 1);
          draw_vert_line(buffer, i,
                         (MODULE_SIZE - bands[i]),
                         (MODULE_SIZE - 1));
        }
      }
    } break;
    case RENDER_OUTWARD: {
      for (int i = 0; i < NUM_BANDS; i++) {
        if (bands[i] > 0) {
          draw_vert_line(buffer, i,
                         ((MODULE_SIZE / 2) - bands[i]),
                         ((MODULE_SIZE / 2) - 1));
          draw_vert_line(buffer, i, (MODULE_SIZE / 2),
                         (MODULE_SIZE / 2) + bands[i] - 1);
        }
      }
    } break;
  }
}
```

3.4 Взаємодія з вебсервером

Розглянемо взаємодію основного пристрою з вебсервером за шинним протоколом I2C. Ведучим пристроєм у цій системі є саме вебсервер, який надсилає запити до веденого МК, де вони обробляються з використанням механізму переривань. Контекст параметрів – це структура даних, однаково визначена у програмах обох МК, у якій зберігаються значення параметрів системи.

Лістинг 3.8 – Обробка переривання I2C на веденому пристрої

```
struct {
    int8_t brt;
    int8_t mode;
    int8_t inp;
} spv_context;

static void i2c_slave_handler(i2c_inst_t *i2c,
i2c_slave_event_t event) {
  switch (event) {
    case I2C_SLAVE_RECEIVE: {
      i2c_read_raw_blocking(i2c, (uint8_t *)&spv_context,
sizeof(spv_context));
    } break;
    case I2C_SLAVE_REQUEST: {
      i2c_write_raw_blocking(i2c, (uint8_t *)&spv_context,
sizeof(spv_context));
    } break;
    case I2C_SLAVE_FINISH: {
    } break;
    default: {
    } break;
  }
}
```

Як вже було зазначено, ведучий МК отримує дані з веденого лише один раз на початку роботи пристрою. Це робиться для зручності розробника, щоб не визначати параметри за замовчуванням окремо на кожному контролері. Після цієї першої передачі, модифікація параметрів відбувається виключно на сервері, який узагальнено можна назвати центром керування. Якщо фізичні кнопки обробляються на основному МК, дані після модифікації потрібно надіслати на вебсервер, інакше контекст на ньому буде неактуальним. Така конфігурація системи потребує більше програмного коду, крім того з’явилася б проблема синхронізації параметрів: якщо фізична та віртуальна кнопки натиснуті за короткий інтервал часу, модифіковані дані з МК Pico W будуть відправлені на

основний МК Pico і навпаки. Таким чином, актуальність інформації на сервері знаходилася б у залежності від порядку прибуття пакетів даних.

Під час ініціалізації центру керування пристрій під'єднується до локальної бездротової мережі Wi-Fi за назвою та паролем. Фізичні натискання обробляються послідовно у нескінченному циклі із затримкою між ітераціями, а запити до вебсервера обробляються за допомогою переривань.

Центральним елементом розробленої вебсторінки панелі керування є елемент HTML5 canvas. Він забезпечує умовне зображення одного модуля дисплея, де відтворений поточний режим відображення та рівень яскравості. Крім цього, панель складається із заголовка, трьох текстових полів (тільки для читання), та трьох пар кнопок для модифікації кожного параметра (рисунок 3.3).

Лістинг 3.7 – Структура HTML-документу

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 <title>SpecVis Console</title>
</head>
<body>
 <div id="data" style="display: none;">
  <div id="brt-data"><!--#brt--></div>
  <div id="mode-data"><!--#mode--></div>
  <div id="inp-data"><!--#inp--></div>
 </div>
 <div id="main-content">
  <h1 style="text-align: center;">SpecVis</h1>
  <div id="inp-text">INPUT: </div>
  <canvas width="320" height="320" style="background-color:
black;"></canvas>
  <div>
   <a href="/cmd.cgi?brt=0"><button>-</button></a>
   <input id="brt-text" readonly>
   <a href="/cmd.cgi?brt=1"><button>+</button></a>
  </div>
```

```
  <div>
   <a href="/cmd.cgi?mode=p"><button>PREV</button></a>
   <input id="mode-text" readonly>
   <a href="/cmd.cgi?mode=n"><button>NEXT</button></a>
  </div>
  <div>
   <a href="/cmd.cgi?inp=0"><button>INTERNAL</button></a>
   <a href="/cmd.cgi?inp=1"><button>EXTERNAL</button></a>
  </div>
 </div>
 <style>...</style>
 <script>...</script>
</body>
</html>
```

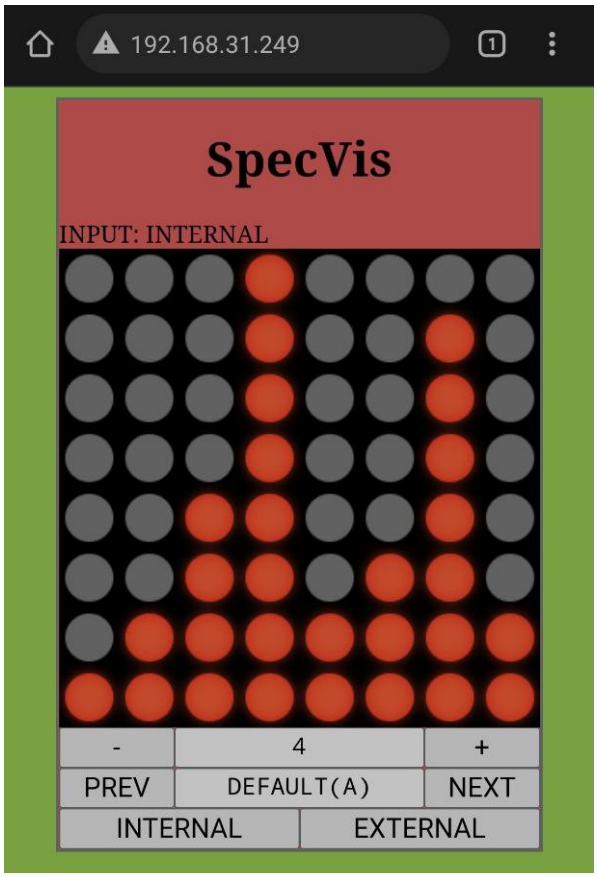

Рисунок 3.3 – Створена вебсторінка

Динамічність та інтерактивність вебсторінки забезпечена технологіями Server-Side Includes(SSI) та Common Gateway Interface(CGI). SSI працює з

модифікованими файлами HTML-розмітки та дозволяє позначити спеціальними тегами місця на сторінці, у які будуть вставлені потрібні дані з сервера. Таким чином користувачу буде доступна актуальна інформація про параметри системи.

Лістинг 3.7 – Обробка SSI на сервері

```
const char * ssi_tags[] = {"brt","mode","inp"};
u16_t ssi_handler(int iIndex, char *pcInsert, int iInsertLen)
{
  size_t printed;
  switch (iIndex) {
    case 0: { // brt
      printed = snprintf(pcInsert, iInsertLen, "%d",
server_context.brt);
    } break;
    case 1: { // mode
      printed = snprintf(pcInsert, iInsertLen, "%d",
server_context.mode);
    } break;
    case 2: { // inp
      printed = snprintf(pcInsert, iInsertLen, "%d",
server_context.inp);
    } break;
    default: {
      printed = 0;
    } break;
  }
  return (u16_t)printed;
}
```

За допомогою запитів CGI реалізований механізм керування пристроєм з боку вебклієнтів. Створена одна програма CGI, яка приймає 3 параметри, що відповідають параметрам системи. Кожен з цих параметрів має 2 значення, тому кожній кнопці інтерфейсу відповідає пара параметр-значення.

Лістинг 3.8 – Обробка запитів CGI

```
const char * cgi_cmd_handler(int iIndex, int iNumParams, char
*pcParam[], char *pcValue[])
{
    if (strcmp(pcParam[0] , "brt") == 0) {
      if(strcmp(pcValue[0], "0") == 0) {
        server_context.brt -= 1;
        if (server_context.brt <= 0) {
          server_context.brt = MAX_BRT;
        }
      } else if(strcmp(pcValue[0], "1") == 0) {
        server_context.brt += 1;
        if (server_context.brt > MAX_BRT) {
          server_context.brt = 1 ;
        }
      }
    } else if (strcmp(pcParam[0], "mode") == 0) {
      if(strcmp(pcValue[0], "p") == 0) {
        server_context.mode -= 1;
        if (server_context.mode < 0) {
          server_context.mode = MAX_MODE;
        }
      } else if(strcmp(pcValue[0], "n") == 0) {
        server_context.mode += 1;
        if (server_context.mode > MAX_MODE) {
          server_context.mode = 0;
        }
      }
    } else if (strcmp(pcParam[0] , "inp") == 0) {
      if(strcmp(pcValue[0], "0") == 0) {
        server_context.inp = 0;
      } else if(strcmp(pcValue[0], "1") == 0) {
        server_context.inp = 1;
      }
    }
    send_to_slave_i2c((uint8_t *)&server_context,
sizeof(server_context));
    return "/index.shtml";
}
```

Реакція на дії користувача для обох способів взаємодії однакова: змінюється відповідний параметр, за потреби його значення порівнюється з межами дозволеного діапазону. Якщо параметр яскравості або режиму відображення виходить за ці межі, він «здійснює оберт» до протилежної межі діапазону. Після цього набір параметрів відправляється на основний пристрій.

4 ТЕСТУВАННЯ ПРИСТРОЮ

Продемонструвати роботу пристрою для динамічної візуалізації за допомогою фотографій досить складно, враховуючи кількість можливих конфігурацій параметрів та музичних композицій. У даному розділі буде описано та показано ключові етапи у роботі пристрою, на основі яких можна сформувати загальне уявлення про нього. Почнемо з рисунку 4.1, на якому зображено знеструмлений візуалізатор.

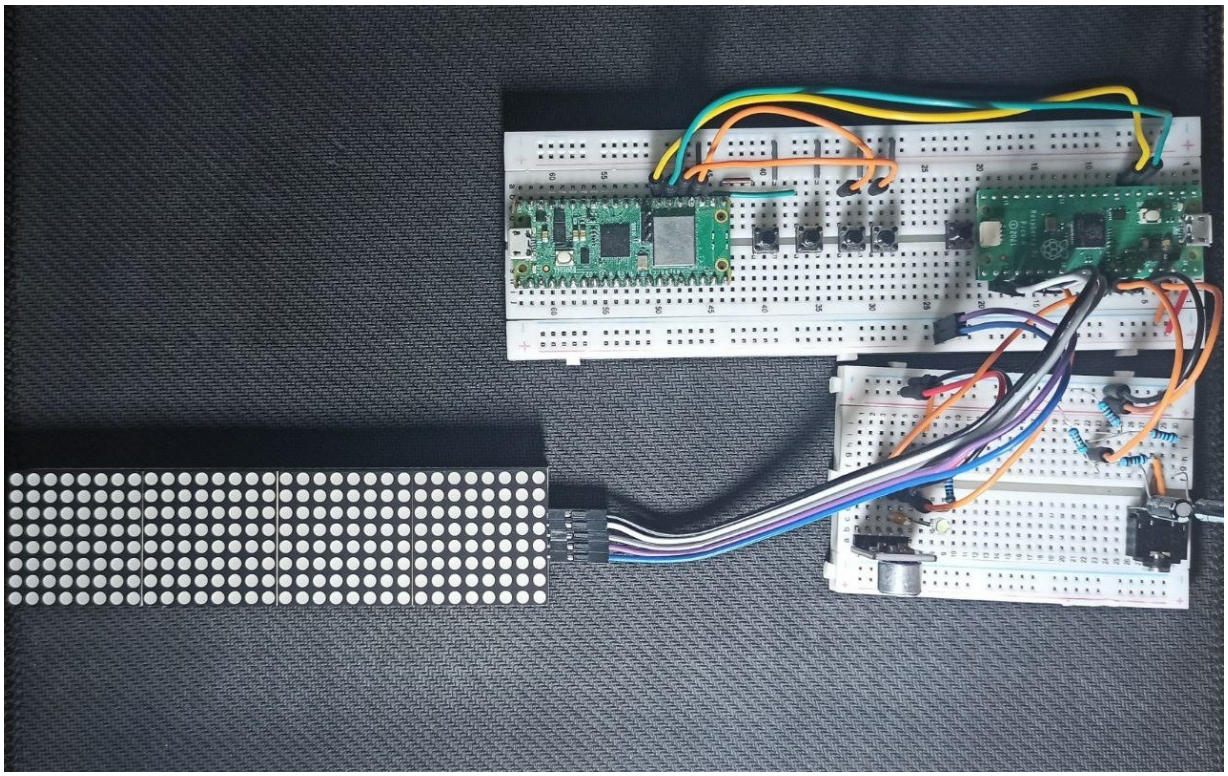

Рисунок 4.1 – Знеструмлений спектральний візуалізатор

Для увімкнення пристрою спочатку потрібно подати крізь кабель micro-USB живлення на основний мікроконтролер, після чого тим же чином

під’єднати центр керування. Загоряння вбудованого світлодіода плати МК Pico W означає успішне підключення центру до локальної бездротової мережі. Загоряння світлодіода поруч з модулем мікрофона означає, що цей модуль зараз є активним способом подання сигналу.

Якщо протягом 3 секунд пристрій не зафіксує сигнал гучніше порогового значення, він перейде у режим паузи. Перемикання та індикація усіх параметрів у цьому режимі все ще доступні, проте візуалізація вимкнена допоки амплітуда сигналу не перевищить межу активації. Стан пристрою після підключення до джерела живлення та переходу у режим паузи показаний на рисунку 4.2.

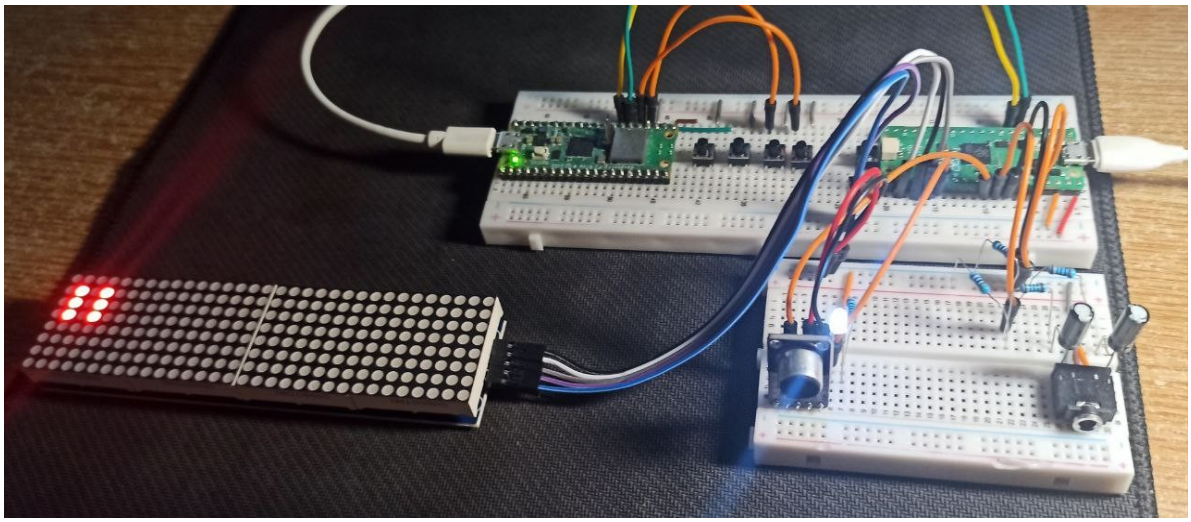

Рисунок 4.2 – Увімкнений пристрій

Підключення крізь кабель AUX зовнішніх динаміків або навушників підтримується більшістю музичних програвачів. Розроблений візуалізатор спектру у цьому плані відрізняється від зовнішніх пристроїв тільки тим, що не може випромінювати поданий звуковий сигнал у навколишній простір. Для вирішення цієї проблеми можна використовувати розгалужувач AUX, який

дозволяє під’єднати до одного джерела сигналу два пристрої, проте для спрощення демонстрації будемо використовувати пряме з’єднання.

На рисунку 4.3 показано візуалізатор, з’єднаний кабелем зі смартфоном, у активному режимі роботи. Після натискання на фізичні або віртуальні кнопки, що відповідають за контекст режиму відображення або рівня яскравості, на екрані з’явиться оверлей і залишиться там на деякий час. На ньому буде показано нове значення модифікованого параметра.

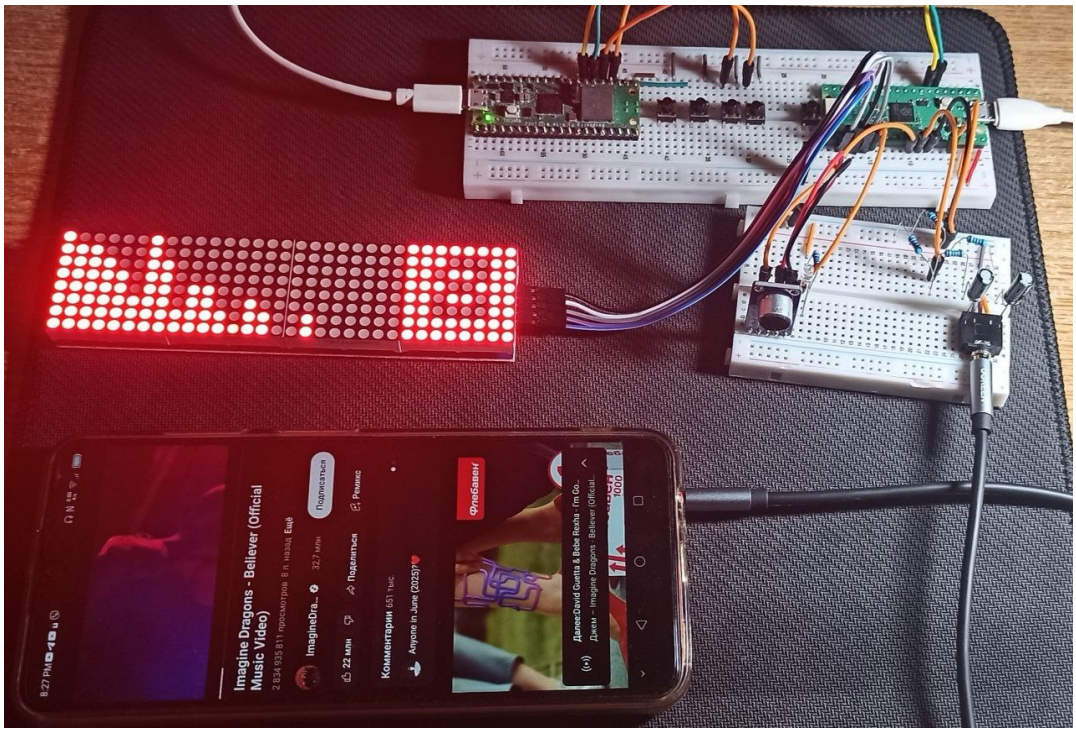

Рисунок 4.3 – Робочий стан пристрою після зміни яскравості

Подача сигналу зі вбудованого мікрофона, а також точковий режим відображення продемонстровані на рисунку 4.4. За рахунок використання підсилювача MAX9814, зображення спектру сигналу з мікрофона є більш об’ємним та чутливим до звуків малої амплітуди, у порівнянні з дротовим підключенням.

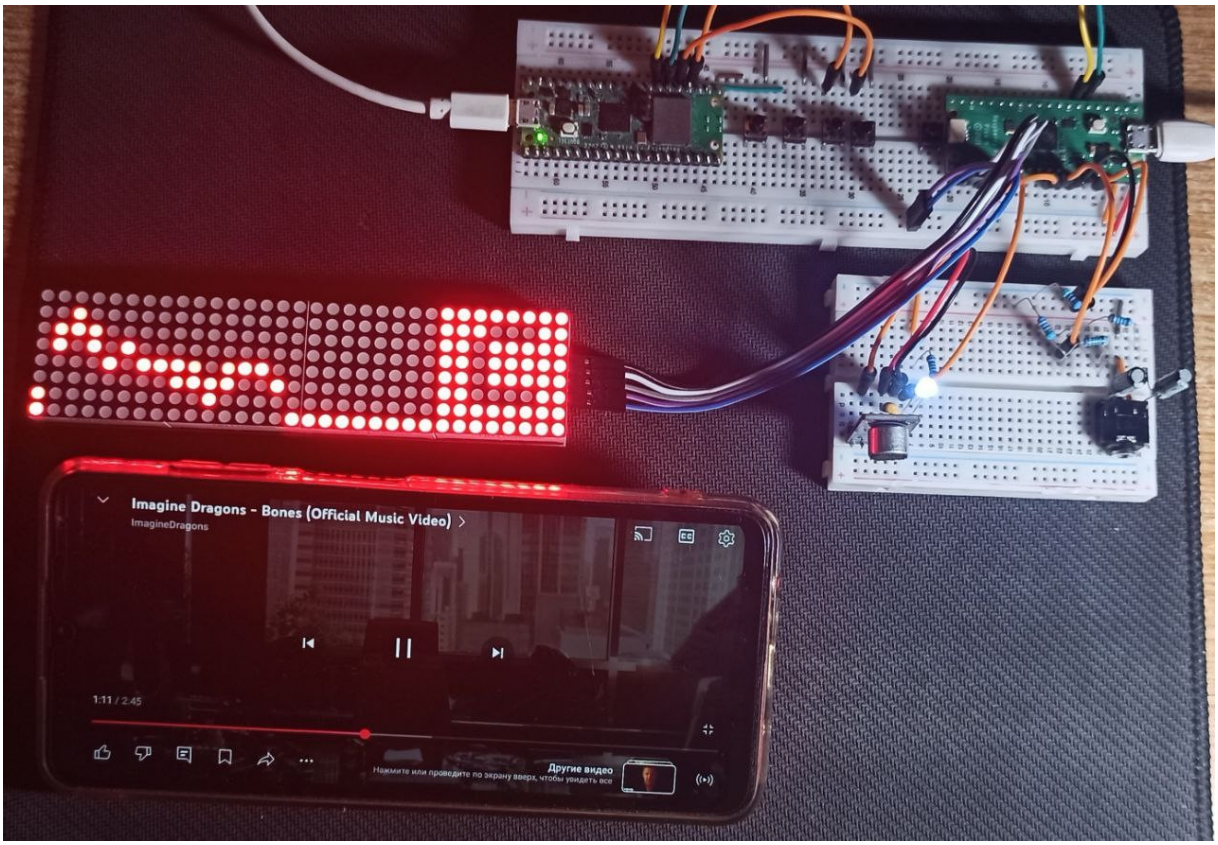

Рисунок 4.4 – Робочий стан пристрою після зміни режиму відображення

Серед помітних недоліків реалізації можна виділити нерегулярні короткочасні шуми від мікрофонного модуля, амплітуди яких іноді достатньо для виходу візуалізатора з режиму паузи. Одна з можливих причин такої поведінки – нестабільне джерело живлення.

За результатами великої кількості тестових сесій, критичних проблем та помилок, що погіршують досвід користування або унеможливлюють подальшу роботу з пристроєм, не виявлено.

ВИСНОВКИ

У кваліфікаційній роботі представлено пристрій для спектрального аналізу та візуалізації вхідного звуку у реальному часі на базі мікроконтролерної плати Raspberry Pi Pico.

Застосовані у роботі програмні структури та алгоритми можуть бути задіяні у розробці вимірювальних приладів, або більш комплексних візуалізаторів музичних композицій. Більша частина програмного забезпечення побудована на основі низькорівневих функцій та структур, тому може бути ефективно абстрагована та виділена у окремі високорівневі бібліотеки.

Створений прототип відрізняється від аналогічних проєктів високою швидкістю обчислень, широким частотним діапазоном спектру та загальною зручністю користування. Однією з ключових особливостей пристрою є можливість дистанційного керування параметрами обчислень. Реалізовано зручний користувацький інтерфейс на вебсторінці, що дозволяє керувати режимом відображення, рівнем яскравості і перемикатися між сигналами з мікрофонного модуля та з аналогового входу.

Серед можливих покращень та модифікацій пристрою можна виділити оптимізацію енергоспоживання, збільшення роздільної здатності дисплея та використання кольорової матриці. Крім того, додавання спільного підсилювача для усіх схем подання сигналу зменшить різницю у спектрах між ними та додасть візуалізації об’єму. Для економії апаратних ресурсів знадобиться змінити систему дистанційного керування, наприклад створити окремий центр керування для усіх пристроїв розумного будинку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Залманзон Л. А. Преобразование Фурье, Уолша, Хаара и их применение в управлении, связи и других областях. Москва : Наука, 1989. 493 с.

2. Наливайко Н. Я. Інформатика : навч. посіб. [для студентів ВНЗ]. Київ : Центр навч. літ., 2011. 576 с.

3. Шпак З. Я. Програмування мовою С : Навч. посіб. для студ. вищ. навч. закл. Львів : Оріяна-Н., 2006. 432 с.

4. Freeman E., Robson E. Head First HTML5 Programming: Building Web Apps with JavaScript. O'Reilly Media, Incorporated, 2011.

5. Fulton J., Fulton S. HTML5 Canvas. O'Reilly Media, Incorporated, 2011.

6. MAX7219/MAX7221 - Serially Interfaced, 8-Digit LED Display Drivers. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/max7219-max7221.pdf (дата звернення: 16.05.2025).

7. MAX9814 - Microphone Amplifier with AGC and Low-Noise Microphone Bias. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/max9814.pdf (дата звернення: 18.05.2025).

8. Ramirez R. W. The FFT, fundamentals and concepts. Englewood Cliffs, N.J : Prentice-Hall, 1985. 178 p.

9. Raspberry Pi Pico Datasheet: An RP2040-based microcontroller board. URL: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf (дата звернення: 15.05.2025).

10. Ritchie D. M., Kernighan B. W. The C Programming Language. Prentice Hall, 1990. 272 p.

11. Witte R. A. Spectrum and Network Measurements. SciTech Publishing, Incorporated, 2014.