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 другий (магістерський)

Спеціалізовані комп'ютерні засоби для спектрального аналізу звукових сигналів на базі мікроконтролера Arduino
(тема)

Виконав:

здобувач 2024 року навчання,
групи СКСм-23-2
Українець В.О.
(прізвище, ініціали)

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

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)

Керівник: доц. Хаханова Г.В.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Спеціалізовані комп'ютерні системи
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

"____" _____ 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Українцю Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Спеціалізовані комп'ютерні засоби для спектрального аналізу сигналів на базі мікроконтролера Arduino

затверджена наказом по університету від "08" 11 2024 р. № 1189 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22.01.2025

3. Вихідні дані до роботи

Мікроконтролер Arduino Nano

Мікрофон

Мова програмування C++

Матриця MA

Середовища розробки Arduino IDE

4. Перелік питань, що потрібно опрацювати в роботі

1 Аналіз предметної області та постановка завдання

2 Аналіз поняття спектра звуку та якості аналізу

3 Аналіз технологій аналізу аудіо спектра

4. Розробка апаратної реалізації проекту

5. Розробка програмної реалізації проекту

6. Тестування розробленої системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 17 слайдів

6. Консультанти розділів роботи (заповнюється 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	Термін виконання етапів роботи	Примітка
1	Отримання завдання	02.09.2024-06.09.2024	
2	Аналіз проблемної галузі, постановка завдання	06.09.2024-30.09.2024	
3	Вибір інструментальних засобів та розробка структурної схеми проекту	30.09.2024-08.10.2024	
4	Збірка апаратної частини	08.10.2024-12.10.2024	
5	Програмна реалізація	12.10.2024-20.10.2024	
6	Тестування розробленої системи	20.10.2024-25.10.2024	
7	Оформлення пояснювальної записки	25.10.2024-20.11.2024	
8	Оформлення графічного матеріалу	20.11.2024-14.12.2024	
9	Перевірка виконаного проекту керівником	15.12.2024-24.12.2024	
10	Захист роботи	14.01.2025-31.01.2025	

Дата видачі завдання 02 вересня 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Хаханова Г.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи містить: 58 с., 1табл., 10 рис., 10 джерел посилання.

КОНТРОЛЕР, МІКРОКОНТРОЛЕР НА БАЗІ АРДУІНО, ARDUINO, ФУР'Є-ПЕРЕТВОРЕННЯ, FFT, СПЕКТРАЛЬНИЙ АНАЛІЗ, АУДІО СИГНАЛ, АНАЛІЗ ЗВУКУ, СПЕКТРОГРАМА, ШУМОВЕ ЗАБРУДНЕННЯ, АУДІО АНАЛІЗАТОР, МІКРОФОН, ADC, ОБРОБКА СИГНАЛУ, ВІЗУАЛІЗАЦІЯ, ОБЛАДНАННЯ ДЛЯ АНАЛІЗУ, ТЕОРЕТИЧНИЙ АНАЛІЗ, C++, C.

Метою кваліфікаційної роботи є розробка комп’ютерного засобу для спектрального аналізу звукових сигналів на базі мікроконтролера Arduino.

У ході кваліфікаційної роботи було проаналізовано принципи роботи та готові рішення для систем спектрального аналізу звукових сигналів. На основі проведеного аналізу визначено основні вимоги та параметри, необхідні для розробки системи аналізу аудіоспектра з використанням мікроконтролера Arduino. Особливу увагу було приділено вибору відповідного апаратного та програмного забезпечення для забезпечення точного аналізу та обробки звукових сигналів в режимі реального часу.

ABSTRACT

The explanatory note contains 58 pp., 1 table, 10 figures, 10 sources.

CONTROLLER, MICROCONTROLLER BASED ON ARDUINO, ARDUINO, FOURIER TRANSFORM, FFT, SPECTRAL ANALYSIS, AUDIO SIGNAL, SOUND ANALYSIS, SPECTROGRAM, NOISE POLLUTION, AUDIO ANALYZER, MICROPHONE, ADC, SIGNAL PROCESSING, VISUALIZATION, ANALYSIS EQUIPMENT, THEORETICAL ANALYSIS, C++, C

The purpose of the qualification work is to develop a computer tool for spectral analysis of audio signals based on the Arduino microcontroller.

In the course of the qualification work, the principles of operation and ready-made solutions for spectral analysis systems for audio signals were analysed. Based on the analysis, the main requirements and parameters necessary for the development of an audio spectrum analysis system using the Arduino microcontroller were determined. Particular attention was paid to the selection of appropriate hardware and software to ensure accurate analysis and processing of audio signals in real time.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

FFT – Fast Fourier Transform – Швидке перетворення Фур'є;

AUX – Auxiliary – допоміжний аудіоінтерфейс;

TRRS – Tip-Ring-Ring-Sleeve – роз'єм із 4 контактами (стерео + мікрофон);

TRS – Tip-Ring-Sleeve – роз'єм із 3 контактами (стерео);

TS – Tip-Sleeve – роз'єм із 2 контактами (моно);

OFC – Oxygen-Free Copper – безкиснева мідь для мінімальних втрат сигналу;

IDE – Integrated Development Environment – інтегрована середа розробки.

ВСТУП

Аналіз аудіоспектра звуку відіграє ключову роль у сучасних технологіях і знаходить широке застосування у таких сферах, як музична індустрія, системи екологічного моніторингу, акустична діагностика та безпека. З розвитком технологій зростає потреба у високоточних методах аналізу звукових сигналів, що дозволяють ідентифікувати частотні компоненти, оцінювати їх динамічні зміни в часі та забезпечувати точну і наочну візуалізацію спектру. Такий підхід сприяє більш глибокому розумінню акустичних процесів, які відбуваються у навколишньому середовищі, та дозволяє отримувати критично важливі дані для прийняття обґрунтованих рішень у різних галузях.

Якість аналізу аудіоспектра безпосередньо впливає на точність та надійність отриманих результатів, що має вирішальне значення не лише для професійного аудіообладнання, але й для наукових досліджень, пов’язаних з обробкою звукових сигналів. Високоточний аналіз спектра є основою для створення інноваційних технологій, таких як новітні методи стиснення аудіоданих, розробка адаптивних фільтрів, алгоритмів обробки сигналів або навіть моделювання сучасних музичних інструментів. Завдяки таким рішенням з’являються нові підходи до вирішення проблем у сфері акустики, розширюються можливості сучасних технологій і вдосконалюються методи роботи з аудіосигналами.

У межах даної роботи основний акцент зроблено на розробці системи аналізу аудіоспектра звуку із застосуванням апаратної платформи Arduino. Arduino - це популярна мікроконтролерна платформа, яка завоювала довіру розробників завдяки простоті в освоєнні, доступній вартості та широкій підтримці серед спільноти користувачів. Використання Arduino дозволяє створювати високофункціональні, гнучкі та кастомізовані системи для аналізу звукових сигналів, що адаптуються до різних умов експлуатації. У рамках

дослідження були детально розглянуті принципи функціонування систем аналізу аудіоспектра, обрані оптимальні апаратні компоненти, а також розроблені алгоритми, що забезпечують ефективну та стабільну роботу системи. Такий підхід дозволяє забезпечити надійність і точність аналізу, а також відкриває перспективи для подальшого розвитку в області аудіотехнологій.

1 СПЕКТРАЛЬНИЙ АНАЛІЗ ЗВУКУ: ТЕХНОЛОГІЇ ТА ПІДХОДИ

Спектральний аналіз є одним з ключових методів для вивчення звукових характеристик, який широко використовується в різних галузях: музична індустрія, аудіоінженерія, акустика, наука про звуки, екологічний моніторинг та багато інших. Розуміння спектру звуку дає можливість детальніше аналізувати різні властивості звукових хвиль та їхню природу. систем, проаналізовано методи визначення ідентичності особистостей, визначено мету та постановку завдання.

1.1 Поняття спектру звуку

Спектр звуку – це розподіл інтенсивності звукових хвиль по різних частотах, що дозволяє отримати детальну інформацію про склад звукового сигналу. Він являє собою своєрідну "картину" звуку, яка показує, з яких частотних компонентів складається си

Спектр звуку можна уявити у вигляді графіка, де по горизонтальній осі розташовані частоти (вимірювані в герцах), а по вертикальній осі – амплітуди цих частот. Такі графічні зображення спектрів називаються спектрограмами і використовуються для візуального представлення частотного складу звуку. Кожна спектрограма надає детальну інформацію про те, які частоти присутні у звуковому сигналі і наскільки вони інтенсивні, що дозволяє не тільки чути звук, але й бачити його у вигляді зрозумілих графіків. Спектрограми є важливим інструментом в роботі інженерів і науковців, оскільки дозволяють отримати точну інформацію про звукові процеси, що відбуваються у різних середовищах.

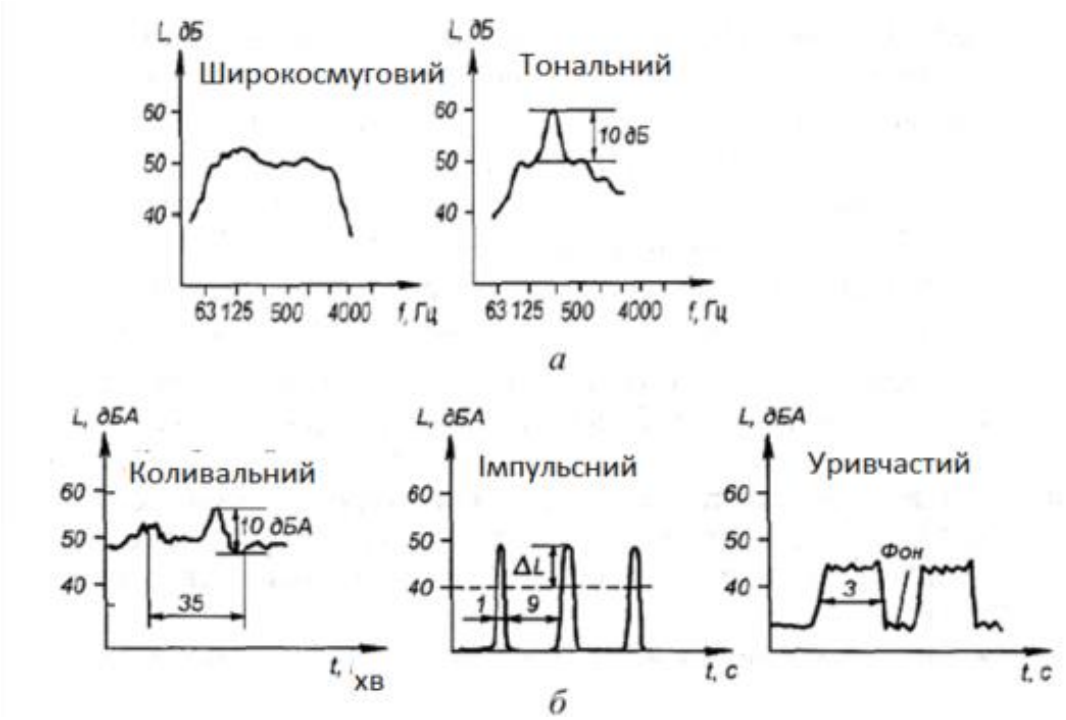

Рисунок 1.1 – Спектрограми за їх видами

Одним з найпоширеніших методів для отримання спектру звуку є швидке перетворення Фур'є (FFT). Цей метод дозволяє перетворити звуковий сигнал з тимчасової області в частотну. Іншими словами, замість аналізу зміни звуку з плином часу, як це робиться в тимчасовій області, FFT дозволяє "розкласти" сигнал на його частотні складові, що значно полегшує аналіз складних звукових сигналів. Швидке перетворення Фур'є широко використовується в аудіоінженерії та наукових дослідженнях, оскільки воно є відносно простим у реалізації і водночас дуже ефективним з точки зору швидкості та точності.

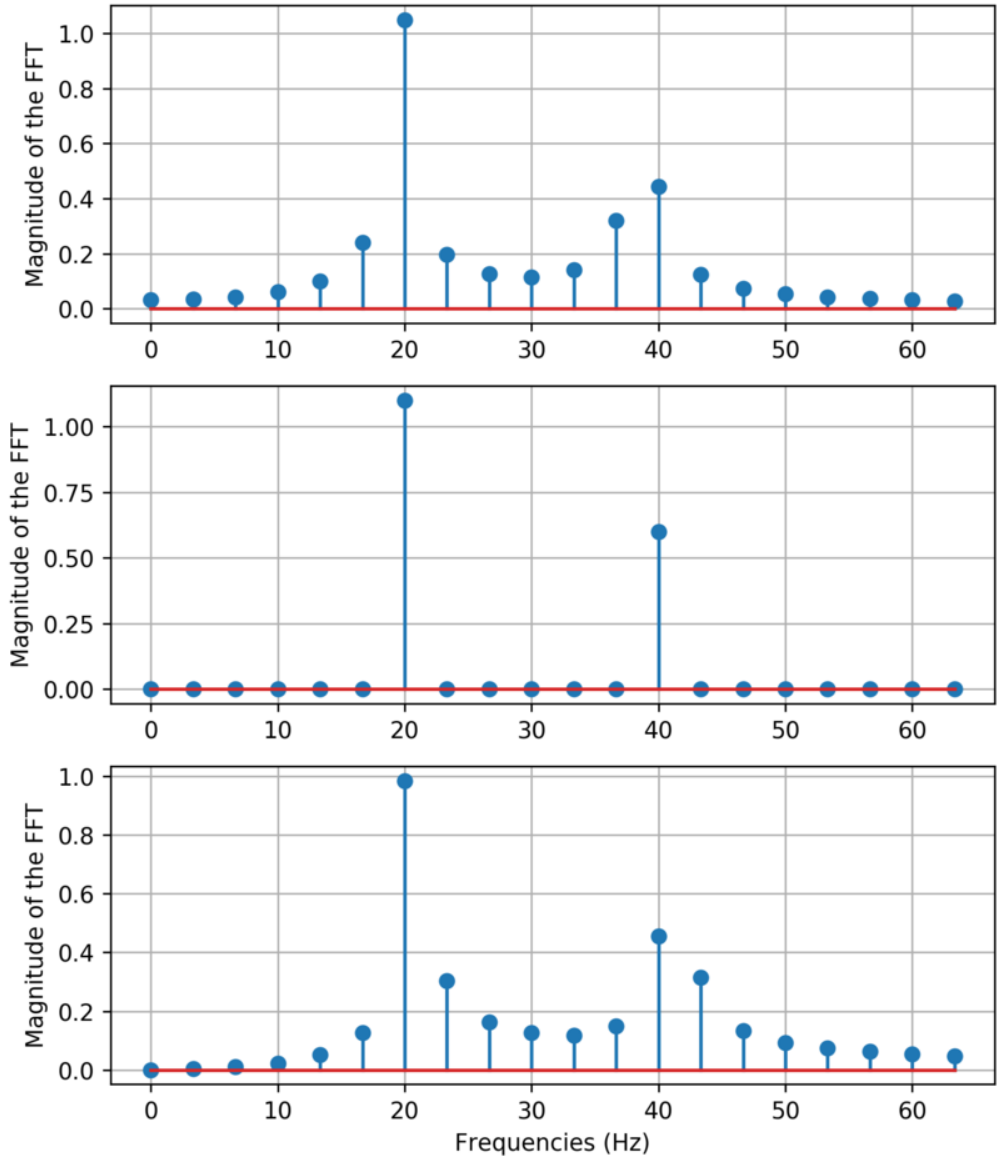

Рисунок 1.2 – Амплітуди швидкого перетворення Фур'є для різної кількості компонентів розкладання

Для більш точного розуміння спектру звуку варто звернути увагу на декілька ключових характеристик:

– частота (Frequency): це основна характеристика звукових хвиль, яка вимірюється в герцах (Гц) і визначає, скільки разів звукова хвиля повторюється за одну секунду. Людське вухо здатне сприймати частоти в діапазоні від приблизно 20 Гц до 20 кГц. Звуки з нижчою частотою вважаються низькими (бас), а з вищою – високими (требл). Наприклад, басові звуки у музиці зазвичай знаходяться в діапазоні від 20 до 200 Гц, тоді як звуки високих частот – в діапазоні від 2 до 20 кГц;

– амплітуда (Amplitude): амплітуда визначає інтенсивність або гучність звуку. Вона відповідає за те, наскільки сильним чи тихим є звук. Звук з великою амплітудою буде гучним, а з малою – тихим. Амплітуда також може використовуватися для визначення різниці між основними та фоновими звуками, що є важливим для процесів аудіообробки і створення якісного звуку;

– фаза (Phase): фаза звукової хвилі визначає її стан у певний момент часу. Цей параметр є особливо важливим при обробці складних звукових сигналів, оскільки зміна фаз різних звукових компонентів може призводити до явищ інтерференції – підсилення або ослаблення певних частот. Правильна фаза сигналів важлива для створення чіткого і чистого звуку, особливо при роботі з кількома джерелами звуку.

Спектральний аналіз надає можливість розділяти звук на окремі частотні компоненти, що робить його незамінним у таких задачах, як виявлення шумів, аналіз голосу та музики, діагностика аудіо систем, дослідження акустичних властивостей матеріалів та моніторинг навколишнього середовища. Для кожної з цих задач спектр звуку може містити різну інформацію, тому важливо підібрати правильні методи і технології для його аналізу.

Для проведення спектрального аналізу використовуються різноманітні інструменти та технології. Це можуть бути як програмні рішення для персональних комп'ютерів (наприклад, MATLAB, Audacity або спеціалізовані аналізатори спектра), так і апаратні пристрої, які використовуються для детального вивчення звукових сигналів. Зокрема, популярними є мікроконтролери, такі як Arduino, які можуть використовуватися для створення простих, але ефективних систем спектрального аналізу звуку. Завдяки своїй доступності та простоті у використанні, Arduino часто стає вибором як для ентузіастів, так і для професіоналів у галузі аудіообробки.

1.2 Історія аналізу аудіо спектру

Історія дослідження аудіоспектру бере свій початок з середини XX століття, коли вчені почали досліджувати математичні та технічні підходи до обробки звукових сигналів. У цей час було зроблено важливе відкриття: звук, який ми чуємо як суцільний потік, насправді складається з комбінацій частот різної величини. Це фундаментальне усвідомлення заклало основу для розробки методів спектрального аналізу. Одним із перших практичних застосувань цього методу стала телекомунікаційна галузь. У телефонному зв'язку аналіз спектру використовували для покращення якості передачі голосу та оптимізації роботи дротових мереж.

Перехід від аналогових до цифрових технологій у середині XX століття призвів до революції в обробці звуку. Основи цієї революції були закладені ще у XIX столітті Жаном-Батистом Фур'є, який розробив метод перетворення звукових сигналів із часової області у частотну. Цей підхід став базовим для сучасних методів роботи з аудіосигналами. Теорія Фур'є стимулювала розробку нових алгоритмів, які дозволили аналізувати структуру звукових хвиль із високою точністю та виділяти їхні частотні компоненти.

У 1960-х і 1970-х роках розвиток цифрової техніки стимулював появу перших алгоритмів для цифрової обробки сигналів. Це стало можливим завдяки зростанню обчислювальних потужностей. Саме тоді аналіз спектру почав використовуватись у таких нових сферах, як музична індустрія, акустичні дослідження та телекомунікації.

У 1965 році було розроблено метод швидкого перетворення Фур'є (FFT), авторами якого стали Кулі та Тьюкі. Ця технологія зробила аналіз спектру значно швидшим, дозволивши працювати з великими обсягами даних у реальному часі. Використання FFT стало критично важливим у багатьох галузях: від медицини, де цей метод застосовували для аналізу серцевих і мозкових сигналів, до науки й технологій, наприклад, у розробці акустичних систем.

Розвиток технологій цифрової обробки сигналів привів до створення стандартів стиснення аудіо та відео, таких як MP3 і AAC. Завдяки їм вдалося зменшити розміри файлів без значної втрати якості звуку. Це стало основою для популярності потокових платформ, як-от Spotify або YouTube Music.

Крім того, аналіз спектру звуку став основою для систем розпізнавання мови. Ці системи, які постійно вдосконалюються, знайшли застосування в голосових асистентах (наприклад, Siri або Google Assistant), системах безпеки, а також у побутових пристроях, таких як "розумні" будинки.

У 1980-х і 1990-х роках, завдяки появі персональних комп'ютерів, аналіз аудіо став доступним для широкої аудиторії. Такі програми, як MATLAB або Audacity, дозволили користувачам проводити детальний аналіз спектру звуку навіть удома. Це стало важливим кроком для музикантів, інженерів та ентузіастів, які могли тепер працювати зі звуком на професійному рівні.

Окрім музики, спектральний аналіз використовувався в індустрії для моніторингу обладнання, наприклад, двигунів і машин. Це дозволило своєчасно виявляти несправності та проводити ремонт.

Сьогодні спектральний аналіз звуку продовжує активно розвиватися. Використання штучного інтелекту, глибинного навчання та хмарних обчислень дозволяє здійснювати аналіз звуку у реальному часі з надзвичайною точністю. Це відкриває нові можливості для створення систем розпізнавання мови, автоматизації процесів керування у "розумних" пристроях та оптимізації акустичних досліджень.

Сучасні дослідження зосереджені на розробці більш точних алгоритмів, здатних працювати з великими масивами даних, а також на вдосконаленні методів, що дозволяють аналізувати складні звукові сигнали у складних акустичних середовищах. Інновації у цій галузі обіцяють зробити аналіз звуку ще більш ефективним у таких сферах, як медицина, кіноіндустрія, аерокосмічна інженерія та акустична екологія.

1.3 Критерії якості аналізу аудіо спектра

Критерії якості аналізу аудіо спектра є вирішальними для досягнення точності, надійності та ефективності в обробці звукових сигналів. Вони визначають здатність системи аналізу виконувати свої завдання на високому рівні, забезпечуючи коректне представлення звукового спектра та надійність результатів. Ось основні критерії, які варто враховувати:

Точність частотного аналізу, який визначає здатність системи точно ідентифікувати частотні компоненти звукового сигналу. Висока точність дозволяє розділяти близькі по частоті звукові хвилі та аналізувати складні сигнали, такі як музичні інструменти, голос або акустичні шуми. Неточність частотного аналізу може призводити до спотворення результатів і втрати важливої інформації про спектр сигналу.

Чутливість визначає здатність системи виявляти дуже слабкі сигнали або сигнали на фоні шумів. Висока чутливість є особливо важливою для аналізу тихих або віддалених джерел звуку, таких як природні звуки, відлуння або слабкі біологічні сигнали (наприклад, серцебиття). Недостатня чутливість може призвести до втрати важливої інформації в процесі аналізу.

Роздільна здатність спектрального аналізу визначає здатність системи розрізняти частоти, які знаходяться близько одна до одної. Чим вища роздільна здатність, тим точніше система може розпізнавати компоненти спектра, що є критично важливим у ситуаціях, коли кілька джерел звуку накладаються одне на одне або існують дуже тонкі відмінності між частотами. Зниження роздільної здатності призводить до "розмиття" частотних компонентів.

Швидкість обробки є важливим критерієм, особливо коли аналіз звукового спектра потрібно проводити в режимі реального часу. Це актуально для таких сфер, як аудіо-візуальні системи, моніторинг середовищ, музичне виробництво та інші задачі, де затримки в аналізі є неприйнятними. Швидкі системи забезпечують миттєву реакцію на зміни в звуковому середовищі, що

критично для динамічних і активних середовищ.

Шумовий поріг визначає, наскільки добре система може відокремлювати корисний сигнал від шуму. Низький рівень шуму у фінальних даних забезпечує більш чисте і точне представлення спектра звуку, особливо в умовах, де рівень навколишнього шуму є високим. Система з високим шумовим порогом може неправильно інтерпретувати звукові сигнали або включати небажані шуми в результати.

Надійність і стабільність системи мають велике значення для аналізу, особливо коли мова йде про довготривалі дослідження або повторювані задачі. Система повинна забезпечувати стабільні результати без втрати точності або ефективності під час тривалого використання. Будь-які збої або нестабільність можуть негативно вплинути на якість аналізу і викликати недовіру до отриманих даних.

Зручний і інтуїтивний інтерфейс – важливий аспект для програмного забезпечення та апаратних засобів аналізу спектра. Користувачам необхідно мати можливість легко взаємодіяти з системою, швидко налаштовувати параметри та інтерпретувати результати аналізу. Наявність графічних інтерфейсів для візуалізації спектрограм може значно полегшити процес аналізу, особливо для тих, хто не є експертами у технічних деталях.

Система повинна бути здатною адаптуватися до різних об'ємів даних та умов експлуатації. Це особливо важливо для застосувань, де потрібна обробка великих масивів аудіо даних або постійне збільшення вимог до швидкості та точності аналізу. Масштабованість забезпечує гнучкість системи, дозволяючи їй працювати як з малими, так і з великими задачами без значної втрати продуктивності.

1.4 Мета та постановка завдання

Метою кваліфікаційної роботи є розробка спеціалізованого комп’ютерного засобу для спектрального аналізу звукових сигналів.

Згідно з поставленою метою визначено наступні завдання:

- аналіз поняття спектра звуку, історія аналізу та критерії якості;
- аналіз та вибір мікроконтролеру і компонентів, що входять складовими до графічного аналізатору аудіо спектра звуку;
- розробка структурної схеми проекту;
- розробка функціональної схеми проекту;
- аналіз, вибір мов програмування та середовища розробки мікроконтролера;
- розробка програмного забезпечення мікроконтролера;
- тестування розробленого графічного аналізатора аудіо спектра звуку.

2 ТЕХНОЛОГІЇ АНАЛІЗУ АУДІО СПЕКТРУ МІКРОКОНТРОЛЕРНИХ ПРИСТРОЇВ

У другому розділі розглянуто технології аналізу аудіо спектра на базі мікроконтролерних пристроїв.

2.1 Перетворення аудіо сигналу

Перетворення аналогового аудіо сигналу у цифровий формат є ключовим етапом у обробці звукових даних. Мікроконтролери, такі як Arduino, відіграють важливу роль у цьому процесі, оскільки дозволяють ефективно обробляти звук на базовому рівні за допомогою спеціалізованих алгоритмів. Для точного відтворення та обробки аудіосигналу необхідно виконати кілька послідовних операцій, серед яких семплування, квантування, кодування та збереження даних у цифровому вигляді.

Семплування є першочерговим етапом при роботі з аналоговими сигналами. У ході цього процесу аналоговий сигнал перетворюється у набір відліків - семплів, які фіксують амплітуду звуку в певний момент часу. Чим частіше виконується семплування, тим точніше цифровий сигнал буде відображати оригінальний аналоговий сигнал. Частота семплування зазвичай вимірюється в герцах (Гц) і вказує, скільки разів за секунду знімається значення амплітуди. Наприклад, для обробки аудіо на рівні CD якість становить 44,1 кГц, що означає 44100 семплів на секунду.

Однак для мікроконтролерів, таких як Arduino, частота семплування обмежена апаратними ресурсами. У стандартних моделях Arduino частота семплування зазвичай коливається в межах від 8 кГц до 10 кГц. Це означає, що при обробці звуку з такими параметрами виникає певна втрата якості порівняно з більш потужними пристроями, однак для багатьох базових

застосувань, таких як обробка голосових сигналів чи виявлення шуму, цього достатньо.

Після семплування кожен отриманий семпл повинен бути перетворений у цифрове значення. Це відбувається через процес квантування, який полягає в округленні амплітуди сигналу до найближчого значення у визначеному діапазоні. Кількість можливих значень визначається бітовою глибиною, або розрядністю. Наприклад, 8-бітове квантування дозволяє кодувати амплітуду кожного семпла у вигляді одного з 256 можливих значень (від 0 до 255). У той час як 16-бітове квантування, яке застосовується у більш професійних аудіосистемах, дозволяє записувати кожен семпл із значно вищою точністю – 65536 можливих рівнів.

У контексті мікроконтролера Arduino типовою бітовою глибиною є 10 біт, що дає 1024 рівні квантування. Це дозволяє отримати більш точне відображення амплітуди сигналу, ніж при використанні 8-бітової глибини, проте все ще є компромісом між якістю звуку і обмеженими ресурсами мікроконтролера.

Кодування є наступним етапом, який полягає у перетворенні заквантуваних значень у послідовність бітів, зручну для зберігання та подальшої обробки. Найбільш поширеним методом кодування є імпульсно-кодовий формат (PCM - Pulse Code Modulation). У цьому форматі кожен семпл аудіосигналу представляється у вигляді двійкових значень і може бути збережений або переданий на наступний етап обробки.

Важливо враховувати, що різні алгоритми кодування можуть використовуватися для різних цілей. Наприклад, при передачі сигналу через мережі часто використовуються компресійні методи кодування, такі як MP3 або AAC, які дозволяють зменшити розмір даних за рахунок незначної втрати якості.

Остаточним етапом процесу перетворення є збереження аудіосигналу у цифровому вигляді для подальшого аналізу або обробки. Цей процес залежить від можливостей апаратної платформи. У випадку з Arduino, збереження даних

може відбуватися у внутрішній пам'яті, на зовнішніх носіях, таких як SD-карти, або безпосередньо передаватися на інші пристрої через серійні інтерфейси, такі як I2C або SPI.

2.2 Розбиття сигналу на фрейми

Розбиття аудіо сигналу на фрейми є невід'ємною частиною будь-якого процесу обробки звукових даних. Це дозволяє розглядати сигнал не як суцільну безперервну хвилю, а як послідовність коротких сегментів, що значно полегшує аналіз та обробку звукової інформації. Кожен фрейм представляє собою відрізок сигналу, який може бути окремо опрацьований для подальших перетворень, таких як спектральний аналіз або розпізнавання аудіо.

Основні параметри, що визначають розбиття сигналу на фрейми, включають розмір фрейму (вікна) та крок між фреймами (частота оновлення або "overlap"). Розмір фрейму зазвичай вибирається в межах від 20 до 40 мс. Цей діапазон обумовлений тим, що такий інтервал часу є достатньо малим для аналізу звукового сигналу, але при цьому достатнім для того, щоб включити необхідну кількість циклів звукової хвилі, що дозволяє зробити точний аналіз частотних характеристик.

Частота оновлення визначає, наскільки близько розташовані фрейми один до одного. Зазвичай використовується перекриття фреймів, що дозволяє зменшити втрати інформації на межах фреймів. Наприклад, якщо розмір фрейму складає 25 мс, то крок між фреймами може становити 10 мс, що забезпечує перекриття у 15 мс. Таке перекриття необхідне для збереження безперервності звукового сигналу при його обробці, оскільки без нього можна втратити важливу інформацію на стиках фреймів.

Один із ключових аспектів розбиття сигналу на фрейми — це можливість представлення кожного фрейму у вигляді вектора або матриці для подальшого математичного аналізу. Кожен фрейм може бути оброблений за

допомогою таких методів, як перетворення Фур'є або вейвлет-перетворення, для отримання частотних характеристик або інших параметрів сигналу.

Це дозволяє виконувати більш глибокий аналіз звукового матеріалу на основі кожного окремого фрейму. Наприклад, у спектральному аналізі кожен фрейм перетворюється на набір частотних компонентів, що дає змогу дослідити, як змінюються частотні характеристики звуку з часом.

2.3 Функція вікна

Під час обробки аудіосигналів застосовується метод віконування, де звукові дані розділяються на фрагменти або вікна для подальшого аналізу. Функція вікна впливає на кінцевий результат такого розділення. Вона дозволяє контролювати внесок кожного фрагменту у власне аналізований сигнал, вирівнює артефакти та допомагає уникнути появи спотворень на краях вікна, які можуть виникнути в результаті різниці в амплітуді на початку та кінці фрагменту. Ця функція зазвичай представляє собою математичну формулу або послідовність чисел, які застосовуються до кожного сегмента сигналу для його згладжування та оптимізації для подальшого аналізу.

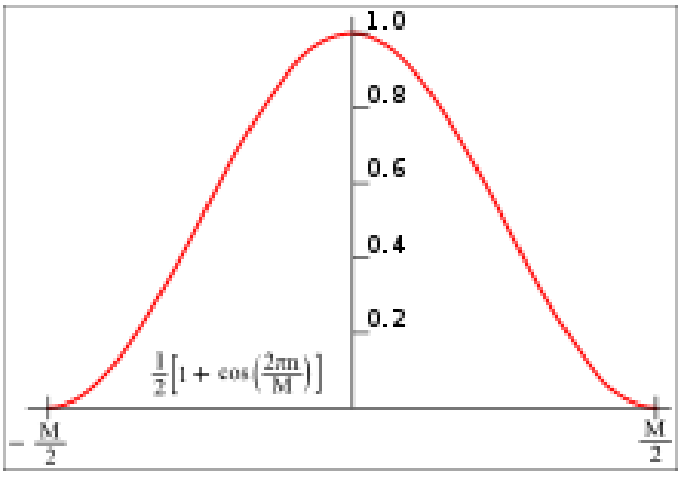

Рисунок 2.1– Графічне зображення віконної функції

Під час обробки аудіосигналів Дискретне Перетворення Фур'є (ДПФ) використовується для перетворення сигналу з часової області у частотну.

ДПФ визначається наступною формулою:

$$x_{(k)} = \sum_{n=0}^{N-1} x\,(n) \cdot e^{2\pi k^{-i} n/N}\,, \qquad (2.1)$$

де X(k) - це значення ДПФ на частоті k, x(n) - це вхідний сигнал, N-кількість відліків у сигналі, k - частотп у дискретному часі n, i – уявна одиниця.

Після виконання ДПФ отримуємо спектр сигналу, де амплітуда |X(k)| вказує на величину сигналу на певній частоті k, а фаза ∠X(k) показує фазовий зміщення цієї частоти.

Для аналізу аудіосигналів використовується розбиття на невеликі часові фрагменти, або фрейми, і на кожен з них застосовується ДПФ. Цей процес називається короткочасним перетворенням Фур'є (КПФ):

$$x_{(m,\omega)} = \sum_{n=0}^{N-1} x\,(n + mH) \cdot \omega(n) \cdot e^{-i\omega m}, \qquad (2.2)$$

де X(m,ω) – представлення КПФ сигналу, x(n+mH) - вхідний сигнал у вікні w(n), w(n) – функція вікна, ω - частота у континуальному часі.

Цей процес дозволяє отримати спектрограму, що візуалізує частотні компоненти сигналу в залежності від часу.

Один з методів аналізу звукових хвиль полягає у використанні Дискретного Перетворення Фур'є (ДПФ). ДПФ є математичним методом, який дозволяє перетворювати сигнал з часової області у частотну. ФормулаДПФ виглядає наступним чином:

$$x_{(k)} = \sum_{n=0}^{N-1} x\,(n) \cdot e^{2\pi k^{-i} n/N}\,. \qquad (2.3)$$

Крім того, для аналізу звукових сигналів використовується процес розбиття на невеликі часові фрагменти, які називаються фреймами. Далі до кожного фрейму застосовується ДПФ, що дозволяє отримати інформацію про складові частоти на дискретних часових інтервалах. Отримані результати можуть бути представлені у вигляді спектрограми, яка візуалізує частотні характеристики сигналу в залежності від часу.

2.4 Обчислення спектральної щільності

Обчислення спектральної щільності є важливою частиною аналізу звукових сигналів. Це процес визначення відносної сили сигналу на різних частотах. Спектральна щільність може бути обчислена шляхом використання квадрату амплітуди Дискретного Перетворення Фур'є (ДПФ) від сигналу.

Формула для обчислення спектральної щільності S(f) виглядає наступним чином:

$$S(f)=|X(f)|^2, \tag{2.4}$$

де S(f) - спектральна щільність, а X(f) - ДПФ сигналу на частоті f. Квадрат модуля ДПФ визначає спектральну щільність сигналу на певних частотах.

Обчислення спектральної щільності дозволяє отримати інформацію про потужність сигналу на різних частотах, що є корисним для різноманітних додатків, таких як аналіз звуку, обробка сигналів у реальному часі та визначення властивостей акустичних сигналів.

2.5 Оцінка відповідності звуків

Оцінка відповідності звуків полягає у визначенні амплітуди частот та обчисленні спектральної щільності для кожного фрейму звукового сигналу.

Для досягнення цієї мети проводиться аналіз Дискретного Перетворення Фур'є (ДПФ) для кожного фрейму сигналу з метою отримання спектральних характеристик.

Спочатку, звуковий сигнал розбивається на невеликі фрагменти, або фрейми. На кожен фрейм застосовується ДПФ для перетворення сигналу з часової області у частотну. Це дозволяє отримати інформацію про розподіл енергії сигналу на різних частотах.

Після отримання ДПФ для кожного фрейму, амплітуда частоти визначається як модуль комплексного числа, отриманого від ДПФ для кожної частоти. Формула для обчислення амплітуди частот A може бути виражена наступним чином:

$$A = \sqrt{\mathrm{Re}^2 + Im^2}, \qquad (2.5)$$

де Re^2 – реальна частина комплексного числа, а Im^2 – уявна частина комплексного числа, що отримано від ДПФ. Визначивши амплітуду кожної частоти для кожного фрейму, можна обчислити спектральну щільність, яка вказує на потужність сигналу на різних частотах у кожному фреймі. Це допомагає в оцінці подібності або відмінності між різними звуками та ідентифікації їх характеристик на основі їх спектрального складу.

2.6 Аналіз отриманих спектрів для виявлення акустичних паттернів

Аналіз спектрів звукових сигналів дозволяє виявити характерні акустичні патерни, які відображають основні властивості звуку. Цей процес полягає в дослідженні спектральних характеристик кожного фрейму аудіосигналу для виявлення типових закономірностей, що можуть бути використані для ідентифікації або класифікації звуків. Завдяки спектральному аналізу можна розрізняти звуки за частотними, амплітудними та часовими характеристиками, що є важливою основою для аналізу звукових сигналів у

різних застосуваннях.

Акустичні патерни представляють собою специфічні набори частотних і амплітудних компонентів, що характеризують певні звуки. Наприклад, для музичних інструментів це можуть бути основні частоти нот і гармоніки, для мовлення - характерні частоти звуків мови, а для техногенних звуків - спектральні особливості шуму або механічних звуків.

Аналіз спектрів допомагає виявити ці патерни через детальний розгляд спектральних компонентів звукового сигналу. Це дозволяє знаходити повторювані звукові структури, що можуть бути використані для подальшої ідентифікації акустичних подій. Основним інструментом для цього є спектрограми, які дають візуальне відображення частотних компонентів звуку в часі, дозволяючи аналізувати зміни в спектрі сигналу.

Кожен звук має свої унікальні спектральні характеристики, такі як основна частота (фундаментальний тон), гармоніки, амплітудні та часові параметри. Основна частота визначає висоту звуку, тоді як гармоніки визначають тембр або забарвлення звуку. Амплітудні характеристики сигналу дозволяють визначити гучність у кожен момент часу, а часові параметри дають можливість відстежувати зміну звукових характеристик протягом часу.

Такі характеристики важливі для класифікації звуків або ідентифікації їх джерел. Наприклад, ударні інструменти мають специфічний широкосмуговий спектр, тоді як струнні або духові інструменти відрізняються чіткими гармонійними структурами. Спектральний аналіз дозволяє легко розпізнавати ці відмінності та ідентифікувати джерела звуку.

Аналіз отриманих спектрів дозволяє здійснювати класифікацію звуків на основі їх акустичних характеристик. У музичних додатках, наприклад, це дозволяє розпізнавати ноти або визначати тип музичного інструменту за його спектральним профілем. Аналіз частотного спектру також дозволяє ідентифікувати різні звукові події, такі як шум або тональні звуки.

2.7 Синтез результатів аналізу

Синтез результатів аналізу спектру звукових сигналів є заключним етапом в обробці аудіоінформації, де всі отримані дані з окремих фреймів поєднуються для отримання цілісної картини аудіосигналу. Цей процес полягає в об'єднанні інформації, отриманої під час аналізу кожного фрейму, таких як амплітуди частот, спектральні компоненти, амплітудно-частотні характеристики та інші параметри. Це дає можливість побудувати повну спектральну картину сигналу, яка може бути використана для подальшого аналізу або обробки.

Основною метою синтезу результатів є поєднання спектральних характеристик, які були виділені під час обробки фреймів. Для кожного фрейму спектральний аналіз дозволяє визначити частоти та амплітуди, що домінують у певний момент часу. Синтезуючи ці дані, ми можемо отримати динамічну картину зміни частотних компонентів сигналу у часі. Це особливо важливо для аналізу складних аудіосигналів, де частоти змінюються з плином часу, як це відбувається, наприклад, у музичних творах або мовленні.

Після синтезу результатів аналізу дані зазвичай представляються у вигляді графіків або інших візуальних інструментів для полегшення інтерпретації та подальшого аналізу. Візуалізація може бути представлена у різних формах залежно від завдань аналізу та виду аудіосигналу:

– спектрограми один з найпоширеніших способів візуалізації аудіосигналу, що показує зміни частотного спектра сигналу у часі. Спектрограма дає змогу чітко бачити, як змінюються частотні компоненти, що корисно для виявлення періодичних чи специфічних звукових подій;

– амплітудно-частотні характеристики графіки, що показують амплітуди окремих частот у кожен момент часу, дають можливість оцінити домінуючі частоти в аудіосигналі;

– частотні діаграми вони дозволяють зобразити спектральний склад сигналу в узагальненому вигляді, фокусуючись на частотних діапазонах, де

відбувається найбільша енергія звукового сигналу.

Інші візуальні представлення: у випадках, коли спектральний аналіз поєднується з іншими методами обробки звуку, можуть бути використані й інші види візуалізації, такі як вейвлет-спектрограми або часово-частотні графіки.

2.8 Відображення графіків спектрів для кожного фрейму

Відображення графіків спектрів для кожного фрейму є одним із ключових етапів аналізу звукових сигналів, що дозволяє візуалізувати їх спектральні характеристики. Візуалізація спектрів дає змогу наочно відобразити частотні компоненти звуку, розподіл енергії між частотами та зміни амплітуд у часі. Це полегшує розуміння структури аудіосигналу та дає можливість аналізувати його особливості на різних етапах обробки.

Графіки спектрів, створені для кожного фрейму, дозволяють побачити розподіл частот у межах невеликого часово-частотного вікна. Для таких графіків характерними є осі частоти та амплітуди, де частота показує, які частотні компоненти присутні в звуці, а амплітуда - їхню інтенсивність. Окрім цього, можна створити об'єднані графіки для більш тривалих відрізків звукового сигналу, які відображають загальну структуру спектра протягом усього аудіо. Аналіз графіків спектрів дозволяє виявляти важливі акустичні особливості, які не завжди помітні в сирих аудіоданих. До основних завдань такого аналізу відносяться:

– пікові частоти вказують на найбільш інтенсивні частотні компоненти сигналу, які можуть бути важливими для ідентифікації звуків або для класифікації;

– аналіз змін амплітуд частотних компонентів допомагає зрозуміти, як розподіляється енергія сигналу в різні моменти часу, що може бути корисним для розпізнавання динаміки звуку;

– вивчення таких характеристик, як ширина смуг, симетричність

спектра або наявність гармонік, допомагає детальніше зрозуміти природу звукового сигналу.

2.9 Фільтрація шумів

Фільтрація шумів є важливим етапом в обробці аудіосигналів, метою якого є зменшення або повне видалення небажаних шумів, що можуть спотворювати корисний сигнал. Цей процес дозволяє покращити якість звуку та полегшити подальший аналіз аудіоданих, зокрема, коли мова йде про важливі частотні компоненти. Видалення шуму дозволяє зосередитися на корисній інформації, що міститься у звуковому сигналі, і значно підвищує роздільну здатність під час аналізу частот та амплітуд.

Основна мета фільтрації шумів полягає у виділенні корисних частотних компонентів звуку та усуненні тих, що не містять важливої інформації або є продуктом сторонніх джерел шуму. Для досягнення цього використовуються різні типи фільтрів. Наприклад, фільтри нижньої частоти блокують високочастотні шуми, залишаючи низькочастотні корисні компоненти, а фільтри верхньої частоти, навпаки, усувають низькочастотні шуми, зберігаючи високочастотну частину сигналу. Пасові фільтри дозволяють пропускати сигнал лише у визначеному частотному діапазоні, що є корисним для виділення важливих частот, а ріжучі фільтри можуть усувати вузькі смуги шуму, що зазвичай виникають від стабільних перешкод.

Для ефективної фільтрації застосовуються цифрові фільтри та спеціалізовані алгоритми, які адаптуються до характеристик шуму та сигналу. Цифрові фільтри виконують обчислення, що дозволяють налаштовувати фільтрацію на конкретні частотні компоненти. Адаптивні фільтри здатні змінювати свої параметри в реальному часі, підлаштовуючись до зміни умов шуму під час запису або передавання звуку. Окрім цього, фільтр Калмана, широко використовуваний у динамічних системах, дозволяє прогнозувати зміни сигналу і усувати шумові компоненти на основі цих прогнозів.

3 АПАРАТНА РЕАЛІЗАЦІЯ ПРОЕКТУ

У розділі описано обрані компоненти для системи аналізу аудіо спектру звуку, їх технічні характеристики, розроблено структурну та функціональну схеми проекту.

3.1 Технічні характеристики компонентів аналізатора аудіо спектра звуку

Для розробки прототипу використано наступні компоненти:

- мікроконтролер Arduino NANOv3;
- адресна матриця MAX7219;
- інтерфейс AUX;
- 3.5мм кабель мініджек.

3.1.1 Мікроконтролер Arduino NANOv3

Arduino Nano - це високофункціональна мікроконтролерна плата компактного розміру, спеціально розроблена для зручного використання в малогабаритних електронних проектах. Основою роботи Arduino Nano є мікроконтролер ATmega328 (у деяких версіях - ATmega168), що забезпечує високу продуктивність при мінімальних апаратних вимогах. Завдяки широким функціональним можливостям і підтримці стандартної екосистеми Arduino, ця плата стала популярним вибором для прототипування та розробки як аматорських, так і професійних проектів.

Arduino Nano має схожий набір функцій із популярною платою Arduino Uno, включаючи кількість цифрових і аналогових входів/виходів, підтримку програмування через середовище Arduino IDE, а також широкий спектр бібліотек для різних датчиків і периферійних пристроїв. Однак, на відміну від

Arduino Uno, Nano виконана в значно компактнішому форм-факторі, що робить її ідеальним вибором для інтеграції в обмежених просторових умовах або портативних пристроях.

Плата підтримує живлення через USB-роз'єм або зовнішнє джерело напруги (від 6 до 12 В), що забезпечує її універсальність у застосуванні. Окрім того, Arduino Nano оснащена вбудованим стабілізатором напруги, що дозволяє використовувати нестабільні джерела живлення. Завдяки компактності, низькому енергоспоживанню та сумісності з численними периферійними модулями, Arduino Nano широко використовується в проектах автоматизації, робототехніки, створення сенсорних систем і пристроїв інтернету речей (IoT).

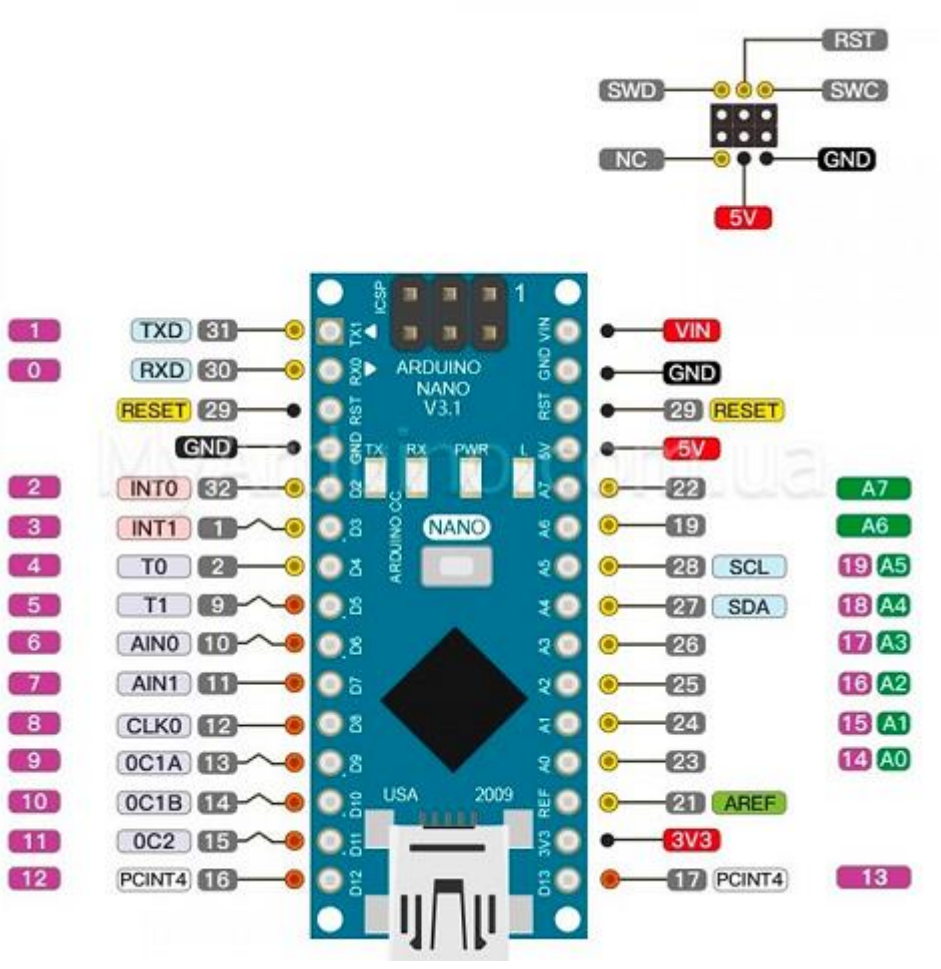

Рисунок 3.1 – Зовнішній вигляд та опис пінів Arduino NANO

Технічні характеристики мікроконтролера Arduino NANO наведено у таблиці 3.1.

Таблиця 3.1 – Технічні характеристики мікроконтролера Arduino NANO

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog IN Pins	8
EEPROM	1 KB
DC Current per I/O Pins	20 mA (I/O Pins)
Input Voltage	7-12V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g
Product Code	A000005

3.1.2 Адресна матриця MAX7219

MAX7219 - це модуль керування світлодіодними матрицями або 7-сегментними індикаторами, який дозволяє зручно управляти до 64 індивідуальними світлодіодами через послідовний інтерфейс. Модуль широко використовується в пристроях виводу інформації, таких як дисплеї годинників, текстові табло, індикатори стану та інші LED-дисплеї. Завдяки вбудованому мікроконтролеру, MAX7219 забезпечує просте підключення до мікроконтролерів і мікропроцесорів через інтерфейс SPI.

Із особливостей можна відокремити: вбудований декодер для 7-сегментних індикаторів, підтримку прямого керування світлодіодами через регістри та вбудовану пам’ять дисплея для збереження стану пікселів

Технічні характеристики адресної матриці MAX7219:

- діапазон напруги живлення від 4,0–5,5 В;
- робоча напруга дисплея 5–6 В (регульована через зовнішній резистор);
- піковий струм 320 мА (залежить від кількості активних світлодіодів);
- кількість керованих світлодіодів до 64 (8×8 матриця або до 8×7-сегментних індикаторів із крапкою);
- інтерфейс SPI-сумісний (DIN, CLK, CS);
- яскравість світлодіодів програмовані 16 рівнів;
- швидкість передачі даних підтримує швидкі SPI-з'єднання для роботи в реальному часі;
- каскадування дозволяє з’єднувати кілька модулів для створення більших дисплеїв;
- розміри матриці (для 8×8) стандартно 32 мм × 32 мм (залежить від конструкції модуля);
- робоча температура від -40 ℃ до 85 ℃.

Зовнішній вигляд матриці MAX7219 приведено на рис. 3.2

Рисунок 3.2 – Зовнішній вигляд матриці MAX7219

3.1.3 Вхідний інтерфейс AUX

AUX - це універсальний інтерфейс для передачі аналогового аудіосигналу між різними пристроями, який широко використовується у побутовій та професійній аудіотехніці. AUX забезпечує просте підключення аудіопристроїв, таких як смартфони, плеєри, комп’ютери, телевізори та аудіосистеми, для відтворення звуку через зовнішні динаміки або навушники. Основною перевагою цього інтерфейсу є його сумісність із широким спектром пристроїв і стандартів аудіовиводу.

Із особливостей можна відокремити: вбудований декодер для 7-сегментних індикаторів, підтримку прямого керування світлодіодами через регістри та вбудовану пам’ять дисплея для збереження стану пікселів

Технічні характеристики інтерфейсу AUX:

- аналоговий тип сигналу з лінійним рівнем;
- стандартний 3.5 мм TRS роз’єм, також підтримуються варіанти 2.5 мм і 6.35 мм залежно від обладнання;
- 2 стерео каналів;
- напруга сигналу 0.5-2В;
- опір стандартно 10–100 кОм для джерела і навантаження, що забезпечує високу сумісність із більшістю аудіопристроїв;

- частотний діапазон від 20 Гц до 20 кГц, що відповідає стандартному діапазону чутності людини;
- рівень сигнал/шум до 80 дБ залежно від підключеного обладнання;
- сумісний з аналоговими і цифровими джерелами звуку через попереднє перетворення сиганалу.

Зовнішній вигляд роз’єму AUX приведено на рис. 3.3

Рисунок 3.3 – Роз’єм AUX

3.1.4 Кабель 3.5 мм мініджек

3.5 мм мініджек-кабель — це універсальний аудіокабель із роз'ємом стандарту 3.5 мм TRS (Tip-Ring-Sleeve), який використовується для передачі аналогового аудіосигналу між пристроями. Завдяки своїй сумісності, компактності та простоті використання, мініджек-кабель є одним із найбільш розповсюджених засобів з’єднання аудіопристроїв у побутових і професійних аудіосистемах. Кабель забезпечує передачу стереозвуку, а також можливість використання для моно- або багатоканального аудіо залежно від типу роз'єму (TRRS, TRS, TS).

Технічні характеристики кабелю 3.5 мм мініджек:

- стандартний 3.5 мм TRS роз’єм, підтримуються також варіанти

TRRS для мікрофонів або багатоканального аудіо;

- матеріал провідника мідь високої чистоти (OFC) для мінімізації втрат сигналу;
- довжина кабелю від 0.2 до 3 м, з можливістю використання довших варіантів;
- опір провідника ≤ 0.2 Ом/м, що забезпечує стабільність сигналу;
- частотний діапазон від 20 Гц до 20 кГц, що відповідає стандартному аудіодіапазону;
- екранування одинарне або подвійне (фольга/оплетення) для захисту від електромагнітних перешкод.

Зовнішній вигляд 3.5 мм мініджек-кабелю на рис. 3.4.

Рисунок 3.4 – 3.5 мм мініджек-кабель

3.2 Структурна та функціональна схеми з’єднання

У процесі розробки кваліфікаційної роботи було виконано побудову та графічне представлення структурної та функціональної схем системи. Структурна схема (рис. 3.3) наочно демонструє логічні зв’язки між основними компонентами системи, такими як адресна світлодіодна матриця, мікроконтролер Arduino Nano та аудіовхід типу мініджек. Вона слугує базою для розуміння взаємодії компонентів на рівні їхніх функціональних завдань та ролей у системі. У свою чергу, функціональна схема описує фізичні з’єднання між компонентами системи, включаючи деталі електричних з’єднань і розташування окремих елементів.

Склад системи включає кілька ключових компонентів. Адресна матриця відповідає за візуалізацію аудіосигналу у вигляді графічного подання спектрів. Мікроконтролер Arduino Nano виконує функцію аналізу сигналу, отриманого через аудіовхід, та передачу обробленої інформації на матрицю для її відображення. Аудіовхід, реалізований на основі роз'єму типу мініджек (3.5 мм TRS), забезпечує підключення до зовнішнього джерела аудіосигналу, такого як смартфон, комп’ютер або інший аудіопристрій.

Після підключення системи до джерела аудіосигналу через роз’єм типу мініджек мікроконтролер Arduino Nano розпочинає свою роботу. Він виконує зчитування вхідного аналогового сигналу через відповідний канал, після чого здійснює аналіз сигналу на основі його спектрального складу. Розбитий на частотні діапазони сигнал передається до адресної матриці, яка відображає амплітуду кожного спектрального компонента у вигляді відповідних світлових індикаторів. Такий підхід дозволяє в реальному часі візуалізувати акустичний сигнал, що значно розширює функціональні можливості системи як інструмента для моніторингу та аналізу звуку.

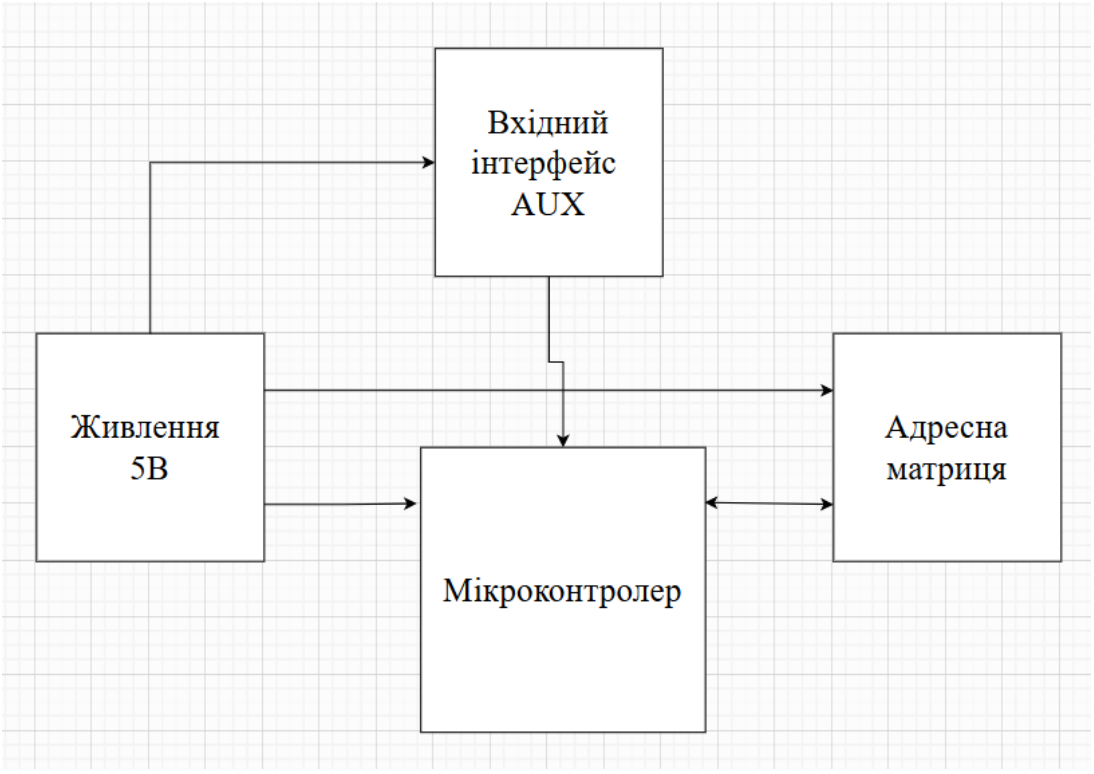

Рисунок 3.5 – Структурна схема проекту

3.3 Схеми з`єднання компонентів

Розглянемо схему підключення складових проекту більш детально.

Для підключення адресної матриці MAX7219 потрібно п’ять дротів. Для коректного підключення матриці до мікроконтролера необхідно знати призначення пінів на матриці (рис. 3.6).

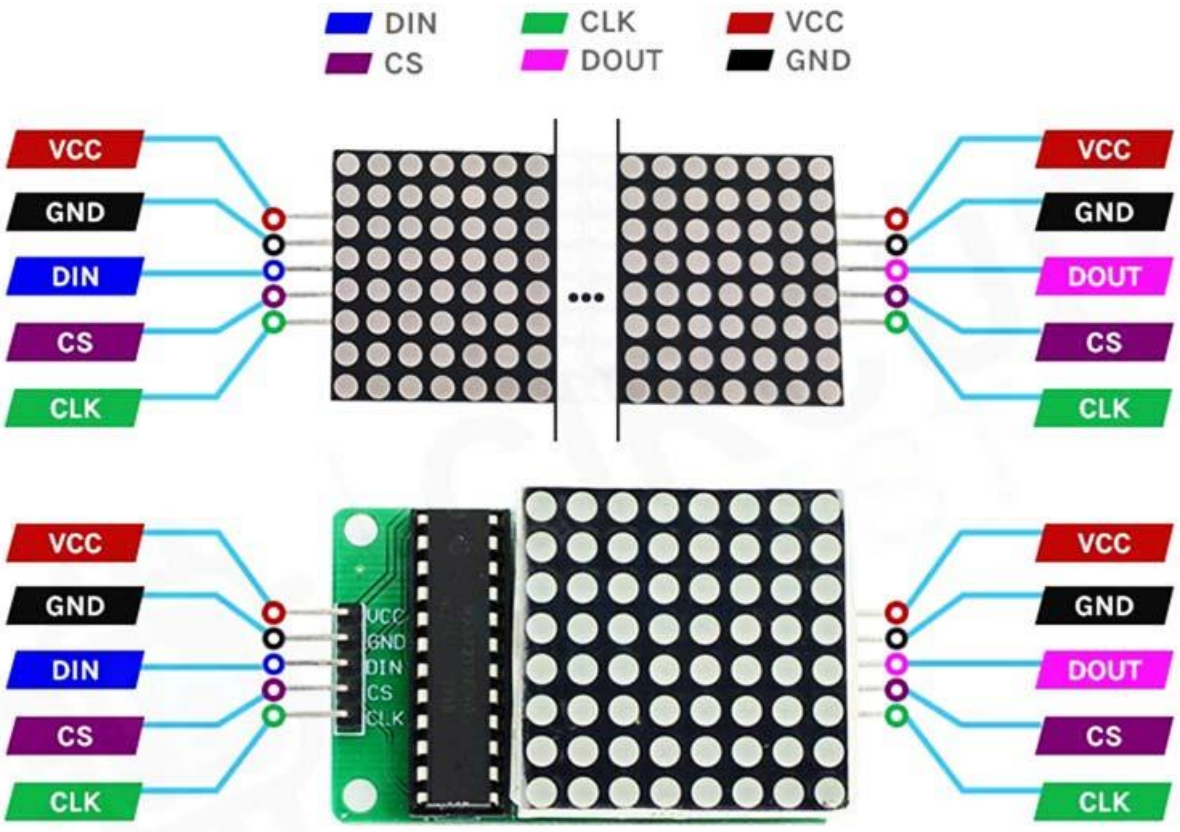

Рисунок 3.6 – Піни підключення адресної матриці

У даній схемі підключення модулів світлодіодних матриць MAX7219 реалізовано з використанням мікроконтролерів Arduino Nano. Для забезпечення живлення матриць їхні контакти VCC підключені до пінів 5V Arduino Nano, які, у свою чергу, паралельно під’єднані до зовнішнього джерела живлення для стабільного забезпечення необхідної напруги. Контакти GND матриць також підключаються до пінів GND Arduino Nano, які

аналогічно паралельно з’єднані із землею зовнішнього джерела живлення для забезпечення загального нульового потенціалу.

Дані для керування матрицями передаються через послідовний інтерфейс SPI. Контакти DIN модулів матриць підключаються до пінів D11 Arduino, які відповідають за передачу даних. Контакти CS (Chip Select), що активують модулі, під’єднані до пінів D10, а контакти CLK (Clock) для синхронізації передачі даних - до пінів D13. Така схема з’єднання забезпечує коректну роботу SPI-протоколу для керування матрицями.

Для зчитування аудіосигналу використовується аналоговий вхід A0 на Arduino Nano. Сигнал із джерела аудіо підключається через конденсатор ємністю 10 нанофарад, що слугує для фільтрації постійної складової сигналу, забезпечуючи передачу лише змінної складової. Одночасно для правильного налаштування опорної напруги аналогового входу використовується пін REF, який підключений через резистор номіналом 4.7 кОм до джерела живлення. Пін 3V3 з'єднується з сигналом через резистор на 10 кОм, що дозволяє створити дільник напруги для належного узгодження рівнів сигналу з вхідними характеристиками мікроконтролера. Такий підхід дозволяє точно зчитувати амплітуду аудіосигналу та відображати її на світлодіодних матрицях.

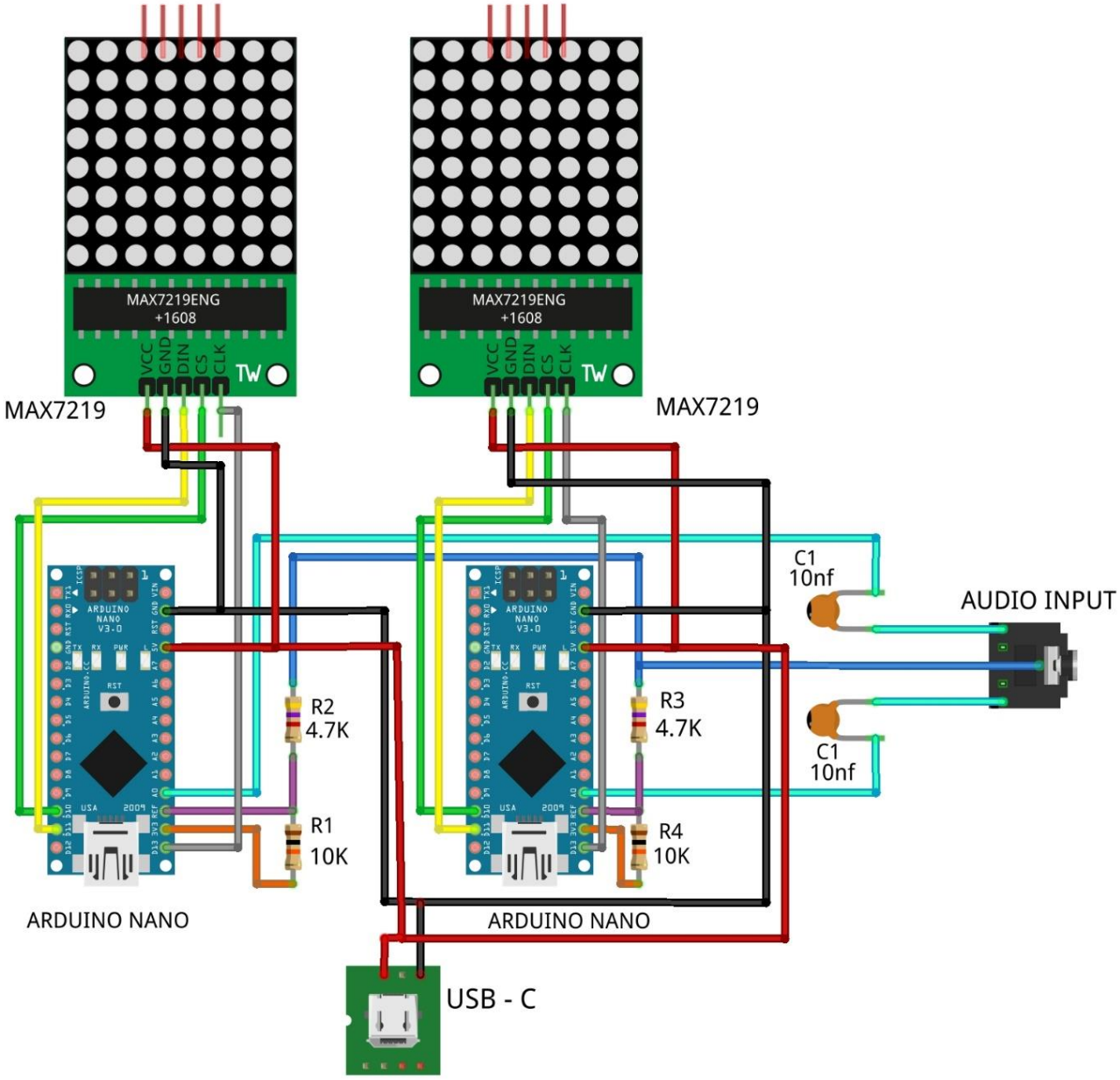

Рисунок 3.6 – Функціональна схема проекту

У представленій схемі реалізовано підключення двох модулів MAX7219 до двох окремих контролерів Arduino Nano для візуалізації сигналів лівого та правого каналів стереозвуку. Однією з особливостей архітектури Arduino Nano є наявність лише одного входу AREF, який слугує для вимірювання опорної напруги або для налаштування аналогового входу під час зчитування сигналів. Оскільки стереосигнал складається з двох незалежних каналів (лівого і правого), для їх одночасного аналізу потрібні два окремі входи AREF, які Arduino Nano не може забезпечити через обмеження апаратної структури. Для

вирішення цієї проблеми у схемі використано два окремих мікроконтролери Arduino Nano: кожен з них відповідає за обробку сигналу лише одного з каналів (лівого або правого). Це дозволяє одночасно зчитувати та обробляти стереосигнал, відображаючи його на окремих світлодіодних матрицях для кожного каналу. Подібне розділення забезпечує точне та незалежне відображення амплітуди сигналу для кожного каналу, але вимагає додаткового апаратного забезпечення у вигляді другого мікроконтролера.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

У розділі приведено обґрунтування вибору мов програмування, опис середовища розробки програми та опис розробленого коду.

4.1 Вибір мови програмування та середовища розробки для мікроконтролера Arduino Nano

Для написання коду для мікроконтролера Arduino Nano існує декілька основних середовищ програмування:

– Arduino IDE;
– PlatformIO;
– Visuino;
– Mbed Studio

4.1.1 Середовище розробки Arduino IDE

Arduino IDE (Integrated Development Environment) є офіційним інтегрованим середовищем розробки для платформи Arduino, призначеним для написання, компіляції та завантаження програмного коду на мікроконтролерні плати. Це середовище базується на мові Wiring, що є спрощеною формою C++, і забезпечує користувачам інтуїтивно зрозумілий інтерфейс, який особливо зручний для новачків у програмуванні. Arduino IDE підтримує широкий спектр плат Arduino, таких як Uno, Mega, Nano та інші, дозволяючи легко адаптувати код для різних апаратних платформ.

Однією з ключових переваг Arduino IDE є вбудований менеджер бібліотек, який спрощує додавання зовнішніх бібліотек для підключення датчиків, дисплеїв, модулів зв'язку та інших периферійних пристроїв. Середовище забезпечує автоматичну перевірку коду на наявність помилок під

час компіляції та пропонує зручний механізм завантаження програм через USB-інтерфейс. Крім того, користувачі мають доступ до великої спільноти та відкритих ресурсів, що сприяють швидкому розвитку ідей та їх реалізації як у навчальних, так і в професійних проєктах. Таким чином, Arduino IDE є універсальним інструментом для розробників, що поєднує простоту використання з гнучкістю та широкими функціональними можливостями.

4.1.2 Середовище розробки PlatformIO

Arduino IDE являє собою інтегроване середовище розробки, що PlatformIO є потужним кросплатформеним середовищем розробки для вбудованих систем, яке забезпечує розробників широким спектром інструментів для створення програмного забезпечення для мікроконтролерів. Це середовище інтегрується з популярними редакторами коду, такими як Visual Studio Code та Atom, і підтримує безліч апаратних платформ, включаючи Arduino, ESP32, STM32 та інші. PlatformIO виділяється своєю модульною структурою, яка дозволяє розробникам гнучко налаштовувати середовище відповідно до конкретних потреб проєкту.

Однією з ключових переваг PlatformIO є його розширені функції управління бібліотеками та залежностями, що забезпечують автоматичне встановлення необхідних модулів без потреби в ручному пошуку. Також середовище підтримує вбудовані інструменти для тестування та налагодження коду, включаючи емуляцію та моніторинг роботи мікроконтролера в реальному часі. Завдяки інтеграції з системами управління версіями, такими як Git, PlatformIO є чудовим вибором для командної роботи та складних проєктів. Це середовище особливо цінується професійними розробниками за його гнучкість, кросплатформеність і зручність для створення сучасних рішень у сфері IoT, автоматизації та робототехніки.

4.1.3 Середовище розробки Visuino

Visuino - це інноваційне інтегроване середовище розробки для платформи Arduino, яке базується на методології візуального програмування. Це середовище дозволяє створювати складні програми для мікроконтролерів без необхідності писати код вручну. Замість цього розробник працює з блоками, що представляють функціональні компоненти, такі як датчики, модулі зв'язку або алгоритмічні конструкції, які можна легко з'єднувати між собою за допомогою графічного інтерфейсу.

Visuino особливо підходить для новачків, які не мають досвіду програмування, а також для освітніх цілей, допомагаючи швидко освоїти основи створення проектів з використанням Arduino. Крім того, середовище підтримує велику кількість мікроконтролерних платформ, таких як Arduino Uno, Nano, Mega, ESP32 та інші, що робить його універсальним рішенням для різноманітних задач. Visuino має вбудовані інструменти для автоматичної генерації та компіляції коду, а також для завантаження його безпосередньо на плату. Середовище виділяється своєю простотою, але при цьому дозволяє реалізовувати складні алгоритми, інтегруючи підтримку численних периферійних пристроїв, таких як дисплеї, мотори, сенсори та модулі зв'язку. Завдяки інтуїтивному підходу до створення проектів, Visuino допомагає суттєво скоротити час розробки, зберігаючи при цьому гнучкість для реалізації складних інженерних ідей.

4.1.4 Обґрунтування вибору мови програмування для мікроконтролера

Arduino IDE було обрано для розробки проєкту через його ключові переваги, що безпосередньо впливають на ефективність роботи. По-перше, середовище забезпечує мінімальну складність налаштувань: інсталяція, підключення плати та завантаження коду виконуються швидко й інтуїтивно.

Це дозволяє зосередитися на написанні програми, не витрачаючи час на складну конфігурацію середовища.

По-друге, Arduino IDE має потужний менеджер бібліотек, який дозволяє легко інтегрувати сторонні модулі для периферійних пристроїв, таких як датчики або дисплеї. Це особливо важливо для проєктів із використанням кількох апаратних компонентів. Крім того, завдяки широкій підтримці апаратних платформ Arduino, ця IDE гарантує сумісність із будь-якою платою, що дає змогу швидко масштабувати або адаптувати проєкт.

Третім вагомим фактором є її популярність серед розробників. Arduino IDE забезпечує доступ до великої кількості документації, прикладів та технічної підтримки від спільноти. Це значно скорочує час на вирішення технічних проблем і дозволяє знайти готові рішення для стандартних завдань. Таким чином, вибір Arduino IDE зумовлений її ефективністю, простотою та доступністю, що є критично важливими для швидкої розробки проєктів.

4.2 Налаштування середовища розробки Arduino IDE та драйверів

Для початку роботи з платою Arduino Nano у середовищі Arduino IDE необхідно виконати низку налаштувань, а також переконатися, що на комп’ютері встановлено відповідний драйвер для роботи з USB-інтерфейсом. Arduino Nano зазвичай використовує чіп CH340 для USB-з'єднання, тому перед початком роботи необхідно завантажити та встановити драйвер CH340. Завантажити його можна з офіційного джерела або перевірених сайтів для підтримки драйверів. Після встановлення драйвера необхідно перезавантажити комп’ютер, щоб зміни набули чинності. Для підключення плати Arduino Nano до комп’ютера використовується стандартний USB-кабель із роз’ємом mini-USB. Важливо переконатися, що кабель підтримує передачу даних, оскільки деякі кабелі призначені лише для зарядки і не забезпечують зв’язок із платою.

4.3 Підключення необхідних бібліотек в Arduino IDE

Перед початком роботи з проектом для мікроконтролера Arduino Nano необхідно визначити основні бібліотеки для його підтримки та програмування. Для цього необхідно спочатку встановити плату Arduino Nano через менеджер плат середовища розробки Arduino IDE. Вибір відповідної плати здійснюється через меню "Інструменти" -> "Плата" -> "Менеджер плат", де після завантаження пакету для Arduino Nano, автоматично встановлюються базові бібліотеки та драйвера для налаштування зв'язку з мікроконтролером.

Додатково, для розширення функціональності проекту, можуть бути використані сторонні бібліотеки, такі як, наприклад, для роботи з датчиками або модулями зв'язку. Для їх підключення слід перейти в меню "Інструменти" -> "Керувати бібліотеками", де можна знайти необхідну бібліотеку через пошук за назвою. Після цього користувач має можливість завантажити актуальну версію бібліотеки.

У разі наявності бібліотеки у форматі ZIP архіву, її можна підключити вручну через меню "Скетч" -> "Підключити бібліотеку" -> "Додати .ZIP бібліотеку", після чого слід вибрати відповідний архів на локальному пристрої. Це дозволяє інтегрувати сторонні бібліотеки, що не входять до стандартного набору

.Для використання бібліотеки у написанні коду мовами C/C++ їх необхідно підключити у файлі основного коду за допомогою директиви препроцесора #include:

```
#include <FHT.h>

#include <SPI.h>

#include <Adafruit_GFX.h>

#include <Max72xxPanel.h>
```

FHT.h – ця бібліотека реалізує алгоритм перетворення Фур'є в

дискретному вигляді, але замість стандартного перетворення Фур'є (FFT) вона використовує перетворення Хартлі. Вибір Хартлі, а не Фур'є, обумовлений тим, що перетворення Хартлі використовує тільки дійсні числа, що дозволяє зменшити складність обчислень та заощадити ресурси мікроконтролера, особливо коли потрібно обробляти сигнали в реальному часі на пристроях з обмеженими можливостями. Це робить бібліотеку FHT більш підходящою для мікроконтролерів, таких як Arduino, для задач, пов'язаних з аналізом спектрів сигналів, зокрема для аудіоаналізу чи обробки даних з сенсорів.

SPI.h – бібліотека для роботи з інтерфейсом SPI (Serial Peripheral Interface), який дозволяє обмінюватися даними між мікроконтролером та периферійними пристроями, такими як сенсори, дисплеї, мікросхеми пам'яті, радіомодулі тощо. SPI забезпечує високу швидкість передачі даних і використовується для синхронної передачі інформації.

Adafruit_GFX.h – універсальна бібліотека для роботи з графічними дисплеями, підтримує різні типи екранних модулів, такі як OLED, TFT, LCD. Вона надає набір функцій для малювання простих геометричних форм, тексту та зображень на екрані, що робить її дуже корисною для відображення інформації на дисплеях в різних проектах на базі Arduino.

Max72xxPanel.h – бібліотека для керування матрицями світлодіодів, зокрема для чіпів MAX7219 або MAX7221, які використовуються для створення панелей з світлодіодними екранами. Вона дозволяє легко керувати великими матрицями світлодіодів, створювати анімації, відображати текст і зображення, що робить її корисною для проектів з візуальними інтерфейсами, в тому числі для виведення графічної інформації або повідомлень на багаторазових дисплеях.

4.4 Написання програмного коду мовами C/C++

Після підключення всіх необхідних бібліотек директивою #include, оголошуються макроси, що задають параметри налаштувань для роботи

системи. У цьому випадку значення макросів використовуються для конфігурації характеристик роботи матриці, звукових сигналів та анімацій:

```
#define BRIGHTNESS 15
```

Оголошення макросу, що задає яскравість матриці. Значення макросу варіюється від 0 до 15, що дає можливість регулювати інтенсивність світіння світлодіодів на матриці. Це важливо для налаштування візуальних ефектів під різні умови освітлення.

```
#define INPUT_GAIN 1.5
```

Макрос, що визначає коефіцієнт підсилення для вхідного аудіосигналу. Значення 1.5 вказує на підсилення звукового сигналу на 50%. Це дозволяє налаштовувати чутливість системи до звуку.

```
#define LOW_PASS 35
```

Макрос, що задає нижній поріг чутливості до шумів. Це значення використовується для фільтрації незначних коливань звукового сигналу, що дозволяє уникнути помилкових сплесків на відсутність звуку.

```
#define MAX_COEF 1.1
```

Макрос, що визначає коефіцієнт для зменшення максимальних піків сигналу, щоб зробити їх більш приємними для сприйняття. Задаючи значення 1.1, ми трохи зменшуємо амплітуду максимальних піків, покращуючи візуальний ефект.

```
#define NORMALIZE 0
```

Макрос, що визначає, чи буде виконуватися нормалізація пік звукового сигналу. Якщо значення дорівнює 1, то піки низьких і високих частот будуть рівними за висотою при однаковій гучності. Якщо значення 0, нормалізація не проводиться.

```
#define SMOOTH 0.4
```

Макрос, що визначає ступінь плавності руху стовпчиків на екрані. Значення 0.4 надає помірну плавність руху, що дозволяє зробити анімацію більш м'якою і приємною для сприйняття.

```
#define DELAY 4
```

Макрос, який задає затримку між оновленнями матриці, в мілісекундах. Затримка в 4 мс визначає швидкість оновлення графічного відображення.

```
#define DEF_GAIN 100
```

Макрос, що визначає максимальний поріг посилення за замовчуванням. Це значення використовується, коли інші методи налаштування посилення (ручний чи автоматичний) не активовані.

```
#define MANUAL_GAIN 0
```

Макрос для активації або деактивації ручного налаштування гучності за допомогою потенціометра. Якщо значення дорівнює 1, то гучність можна налаштовувати вручну, а якщо 0 — автоматичне регулювання.

```
#define AUTO_GAIN 1
```

Макрос для активації або деактивації автоматичного регулювання гучності. Встановлення значення 1 дозволяє активувати цю функцію, а 0 — вимикає.

```
#define MAX_DOTS 1
```

Макрос, що визначає, чи будуть відображатися точки максимальних піків на екрані. Значення 1 включає відображення точок, а значення 0 - вимикає.

```
#define FALL_DELAY 50
```

Макрос, що задає затримку падіння точок максимальних піків на екрані в мілісекундах. Це дозволяє налаштовувати швидкість анімації для кращого візуального сприйняття.

```
#define FALL_PAUSE 700
```

Макрос, що визначає паузу перед падінням точок максимальних піків, що дає можливість створити ефект "затримки" перед відновленням анімаці

```
byte posOffset[33] = {
  2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 32, 36, 40, 44, 48, 52, 57, 62, 67, 72, 78, 84, 90, 96, 102, 108, 120
};
```

Масив, що містить значення для розподілу тонів, орієнтованих на параболічну шкалу частот. Ці значення використовуються для представлення звукових частот від 80 Гц до 16 кГц.

```
#define AUDIO_IN 0
```

Макрос, що вказує пін мікроконтролера, до якого підключено вхідний аудіосигнал.

```
#define POT_PIN 7
```

Макрос, що визначає пін мікроконтролера, до якого підключений потенціометр для ручного регулювання гучності.

```
#define MATR_NUM 4
```

Макрос, який задає кількість матриць, підключених послідовно. У цьому випадку це чотири матриці.

```
#define FHT_N 256
```

Макрос, що визначає ширину спектра для аналізу сигналу. Значення 256 є вдвічі більшим за фактичну ширину спектра, яка дорівнює 128 точок.

```
#define LOG_OUT 1
```

Макрос для активації або деактивації виведення логів для розробників. Якщо значення 1, логи будуть виводитися, а якщо 0 — виведення вимикається.

Після цього викликається функція setup(). У функції setup() відбувається ініціалізація основних параметрів для налаштування мікроконтролера та периферійних пристроїв. Спочатку встановлюється висока частота дискретизації для аналогового порту, що забезпечує точне вимірювання вхідного сигналу, використовуючи маніпуляції з регістрами контролера, які збільшують тактову частоту АЦП. Це дозволяє досягти максимальної частоти дискретизації 19 кГц, що відповідає теоремі Найквиста. Далі налаштовується зовнішнє джерело опорного напруги для АЦП, що забезпечує більш точне зчитування аналогових значень. Ініціалізація послідовного порту здійснюється через команду Serial.begin(9600), що дозволяє встановити комунікацію з комп'ютером на швидкості 9600 біт/с. Після цього здійснюється налаштування матриць: за допомогою команди matrix.setIntensity(BRIGHTNESS) встановлюється яскравість екрану, а в циклі for кожній матриці задається орієнтація для коректного відображення інформації, оскільки матриці розташовані неправильно. Після цього очищується екран матриці командою matrix.fillScreen(LOW), що заповнює екран чорним кольором, створюючи чистий фон для подальшого відображення. Завершує налаштування запис змін на матриці за допомогою команди matrix.write(), що актуалізує всі попередні налаштування і готує екран до подальшої роботи.

Лістинг 4.1 – Код функції setup()

```
void setup() {
  sbi(ADCSRA, ADPS2);
  cbi(ADCSRA, ADPS1);
  sbi(ADCSRA, ADPS0);

  analogReference(EXTERNAL);

  Serial.begin(9600);
  matrix.setIntensity(BRIGHTNESS);
  for (byte i = 0; i < MATR_NUM; i++) {
    matrix.setRotation(i, 1);
  }
  matrix.fillScreen(LOW);
```

```
  matrix.write();
}
```

Надалі в коді оголошується функція analyzeAudio() Функція analyzeAudio() виконує обробку аудіосигналу для подальшого аналізу спектра частот. Вона здійснює кілька етапів, що включають зчитування аналогових даних, обробку за допомогою дискретного перетворення Фур'є та підготовку результатів для подальшого використання.

На першому етапі функція здійснює цикл, що охоплює всі елементи масиву fht_input, визначеного розміром FHT_N. Для кожного індексу масиву зчитується аналогове значення з порту, до якого підключений вхідний аудіосигнал, через команду analogRead(AUDIO_IN). Це значення зберігається у масиві fht_input, що дозволяє накопичувати реальні дані для подальшої обробки.

Після збору даних наступним етапом є функція fht_window(), яка застосовує вікно до зібраних даних. Процес вікнування необхідний для того, щоб зменшити ефекти артефактів, які можуть виникати при використанні дискретного перетворення Фур'є. Вікно дозволяє отримати кращу відповідність частотної характеристики сигналу.

Далі виконується виклик функції fht_reorder(), що перерозподіляє порядок елементів у масиві даних для того, щоб привести їх до формату, необхідного для проведення самого перетворення Фур'є. Це є необхідною підготовкою для коректного виконання математичних операцій у наступному кроці.

Після цього запускається основний процес обробки даних через функцію fht_run(), що виконує дискретне перетворення Фур'є для оброблених даних, перетворюючи сигнал із часової області в частотну. Це дозволяє отримати інформацію про різні частоти, присутні в аудіосигналі.

Останній крок - це виклик функції fht_mag_log(), яка обчислює логарифмічну магнітуду частот, отриманих в результаті перетворення Фур'є. Логарифмічна шкала є корисною для обробки широкого діапазону значень

амплітуд, що дозволяє зручніше працювати з ними, особливо коли значення варіюються на кілька порядків.

Лістинг 4.2 – Код функції analyzeAudio()

```
void analyzeAudio() {
  for (int i = 0 ; i < FHT_N ; i++) {
    int sample = analogRead(AUDIO_IN);
    fht_input[i] = sample;
  }
  fht_window();
  fht_reorder();
  fht_run();
  fht_mag_log();
}
```

Далі оголошується основна функція loop() в якій викликається функція аналізу звуку і вивід результатів на матриці.

На початку функції перевіряється, чи пройшов основний інтервал часу між ітераціями циклу, за допомогою методу millis(), який повертає кількість мілісекунд з моменту старту програми. Після цього запускається процес аналізу аудіосигналу через виклик функції analyzeAudio(), яка отримує спектральні дані і заносить їх у масив fht_log_out[]. Далі здійснюється фільтрація спектра: кожен елемент масиву проходить через кілька етапів обробки.

Спочатку відбувається фільтрація низьких частот за допомогою порогу LOW_PASS, якщо значення елемента спектра менше цього порогу, воно встановлюється в нуль. Потім виконується підсилення сигналу за допомогою коефіцієнта INPUT_GAIN, що множить значення спектра на відповідний множник. Якщо увімкнена нормалізація (NORMALIZE), рівні частот коригуються пропорційно частотам, щоб зберегти баланс між низькими та високими частотами.

Після обробки спектра починається робота з графічним відображенням даних на матричній панелі. Для кожної колонки матриці обчислюється рівень сигналу, залежно від даних спектра, при цьому враховується плавність зміни

стовпців через фільтрацію. Далі рівні спектра конвертуються в значення, що відповідають діапазону, який відображається на матриці (від 0 до 7), і стовпці малюються на матриці з урахуванням їх висоти.

Також відбувається обробка точок максимуму для кожної частоти: якщо значення частоти перевищує певний поріг, то малюється піксель, що вказує на точку максимуму. Ці пікселі поступово "падають", якщо значення частоти стабілізувалося, і оновлюються з урахуванням затримки, заданої параметрами FALL_DELAY та FALL_PAUSE.

Після виконання всіх операцій оновлюється відображення на матриці за допомогою методу matrix.write(), а також скидається прапор падіння, щоб підготувати систему до наступного циклу.

Крім того, передбачено дві опції для керування рівнем гучності: вручну через потенціометр за допомогою піну POT_PIN, якщо активовано режим MANUAL_GAIN, або автоматично, якщо увімкнено режим AUTO_GAIN. У автоматичному режимі рівень гучності регулюється на основі максимального значення спектра, з використанням коефіцієнта MAX_COEF для коригування гучності в залежності від інтенсивності аудіосигналу.

Лістинг 4.3 – Код функції loop()

```
void loop() {
  if (millis() - mainDelay > DELAY - 2) {
    mainDelay = millis();
    analyzeAudio();
    for (int i = 0; i < 128; i ++) {
      if (fht_log_out[i] < LOW_PASS) fht_log_out[i] = 0;
      fht_log_out[i] = (float)fht_log_out[i] * INPUT_GAIN;
      if (NORMALIZE) fht_log_out[i] = (float)fht_log_out[i]
/ ((float)1 + (float)i / 128);
    }
    maxValue = 0;
    matrix.fillScreen(LOW);
    delay(2);
    for (byte pos = 0; pos < MATR_NUM * 8; pos++) {
      int posLevel = fht_log_out[posOffset[pos]];
      byte linesBetween;
      if (pos > 0 && pos < MATR_NUM * 8) {
        linesBetween = posOffset[pos] - posOffset[pos - 1];
```

```
            for (byte i = 0; i < linesBetween; i++) {
              posLevel += (float) ((float)i / linesBetween) *
fht_log_out[posOffset[pos] - linesBetween + i];
            }
            linesBetween = posOffset[pos + 1] - posOffset[pos];
            for (byte i = 0; i < linesBetween; i++) {
              posLevel += (float) ((float)i / linesBetween) *
fht_log_out[posOffset[pos] + linesBetween - i];
            }
          }
          if (posLevel > maxValue) maxValue = posLevel;

          posLevel = posLevel_old[pos] * SMOOTH + posLevel * (1
- SMOOTH);
          posLevel_old[pos] = posLevel;

          posLevel = map(posLevel, LOW_PASS, gain, 0, 7);
          posLevel = constrain(posLevel, 0, 7);

          if (posLevel > 0) matrix.drawLine(pos, 7, pos, 7 -
posLevel, HIGH);
          if (posLevel > 0 && posLevel > maxLevel[pos]) {
            maxLevel[pos] = posLevel;
            timeLevel[pos] = millis();
          }
          if     (MAX_DOTS     &&     maxLevel[pos]     >=     0)
matrix.drawPixel(pos, 7 - maxLevel[pos], HIGH);
          if (fallFlag) {
            if ((long)millis() - timeLevel[pos] > FALL_PAUSE) {
              if (maxLevel[pos] >= 0) maxLevel[pos]--;
            }
          }
        }
        matrix.write();

        fallFlag = 0;
        if (millis() - fallTimer > FALL_DELAY) {
          fallFlag = 1;
          fallTimer = millis();
        }
        if (MANUAL_GAIN) gain = map(analogRead(POT_PIN), 0, 1023,
0, 150);
        if (AUTO_GAIN) {
          if (millis() - gainTimer > 10) {
            maxValue_f = maxValue * k + maxValue_f * (1 - k);
            if (maxValue_f > LOW_PASS) gain = (float) MAX_COEF *
maxValue_f;
            else gain = 100;
            gainTimer = millis();
          }
        }
      }
    }
```

ВИСНОВКИ

У ході кваліфікаційної роботи було розроблено систему аналізу аудіоспектра звуку на основі мікроконтролера Arduino, яка дозволяє здійснювати спектральний аналіз звукових сигналів у реальному часі. Було реалізовано апаратно-програмний комплекс, що включає мікроконтролер Arduino Nano v3, адресну матрицю MAX7219 для візуалізації спектру, інтерфейс AUX та 3,5 мм кабель мініджек для передачі звукового сигналу.

Під час реалізації проекту було здійснено вибір необхідних компонентів з урахуванням їх технічних характеристик, а також розроблено структурну та функціональну схеми системи. Особливу увагу приділено оптимізації алгоритмів обробки сигналів для забезпечення точного та стабільного аналізу аудіоспектра. Програмна частина системи була створена у середовищі розробки Arduino IDE з використанням мови програмування C++.

Запропонований прототип системи демонструє високу функціональність, гнучкість та низьку вартість, що робить його доступним для широкого спектра застосувань, таких як акустична діагностика, навчальні проекти або інтеграція у більш складні аудіосистеми. Візуалізація спектральних компонентів звукового сигналу у реальному часі значно спрощує аналіз та розширює можливості системи у навчальних та дослідницьких цілях.

Прототип може бути вдосконалений у майбутньому шляхом інтеграції більш потужних мікроконтролерів, використання сенсорів з більшою частотною роздільною здатністю або додавання нових функцій, таких як автоматичне визначення характеристик сигналу. Крім того, можлива оптимізація енергоспоживання системи, що дозволить використовувати її у портативних пристроях. Створена система відкриває перспективи для подальшого розвитку та впровадження новітніх технологій у сфері аудіообробки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дмітрієв А.В. Спектральний аналіз сигналів: теорія і методи / А.В. Дмітрієв. – Київ: Наукова думка, 2010. – 412 с.

2. Мікроконтролери в обробці сигналів: практичний підхід / П. І. Ковальов, А. С. Іванов. – Харків: Видавництво ХНУ, 2018. – 256 с.

3. Цифрова обробка сигналів: основи та застосування / О.М. Бондаренко, В.П. Лісовський. – Львів: Вид-во ЛНУ ім. І. Франка, 2015. – 384 с.

4. Smith S. W. Digital Signal Processing: A Practical Guide for Engineers and Scientists / S. W. Smith. – Newnes, 2002. – 656 p.

5. Lyons R. G. Understanding Digital Signal Processing / R. G. Lyons. – Pearson Education, 2010. – 992 p.

6. https://github.com/AlexGyver/FHTSpectrumAnalyzer/blob/master/Firmware/spertrumMatrix_MAX7219/spertrumMatrix_MAX7219.ino

7. Arduino-Based Audio Projects: Build High-Quality Sound Effects with Arduino / J. Evans. – Apress, 2018. – 284 p.

8. Українець В.О. Дослідження аналізу аудіо спектра звуку на базі мікроконтролера Arduino / В.О. Українець // Матеріали тез 28-го міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» - м. Харків, травень 2024. - С. 126-128.

9. Tooley M. Electronic applications and the Arduino. Electronic Circuits. 2019. С. 371–399. URL: https://doi.org/10.1201/9780367822651-19 .

10. Arduino.cc / Arduino® Nano/ ATmega328 Product Reference Manual. – SKU: A000005 Режим доступу : www/ URL: https://docs.arduino.cc/static/54c63bab542e47478925401ea9f0eb28/A000005 datasheet.pdf .