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)_____
(рівень вищої освіти)

Програмно-аппаратна реалізація розширеного автоматичного тюнера на основі покращеного алгоритму аналізу Фур'є
(тема)

Виконав: здобувач 2024 року навчання, групи
СКСм-23-2
Різник М.А.
(прізвище, ініціали)
Спеціальність *123 _Комп'ютерна інженерія*
Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма
Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)
Керівник роботи ____Хаханова Г.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _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 _____
(шифр і назва)

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

Освітня програма _____Комп'ютерна інженерія _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«__»_____ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Різнику Максиму Андрійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Програмно-аппаратна реалізація розширеного автоматичного тюнера на основі покращеного алгоритму аналізу Фур'є

затверджена наказом по університету від "_08_"_____11_____2024 р. № _1189 Ст_____

2. Термін подання здобувачем роботи до екзаменаційної комісії _____10.01.2025_____

3. Вихідні дані до роботи (проекту)_____

Плата Arduino UNO_____

Середовище розробки Arduino IDE_____

Мова програмування C/C++_____

4. Перелік питань, що потрібно опрацювати у роботі_____

Аналіз предметної галузі та постановка задачі проектування

Розробка програмної частини системи

Розробка апаратної частини системи

Програмування мікроконтролерного пристрою

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

_____

_____

6. Консультанти розділів роботи (проекту)

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

7. Дата видачі завдання 02.09.2024_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Видача теми проекту, узгодження і затвердження теми	02.09.2024-30.09.2024	
2	Аналіз проблемної галузі, постановка задачі, вибір інструментальних засобів	30.09.2024-08.10.2024	
3	Розробка електричної схеми пристрою	08.10.2024-12.10.2024	
4	Реалізація макету пристрою	12.10.2024-20.10.2024	
5	Розробка програми для мікроконтролера	20.10.2024-25.10.2024	
6	Тестування системи	25.10.2024-20.11.2024	
7	Оформлення пояснювальної записки	20.11.2024-14.12.2024	
8	Перевірка виконаного проекту керівником, допуск до захисту	15.12.2024-24.12.2024	
9	Захист проекту	14.01.2025-31.01.2025	

Здобувач _____ (підпис)

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

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи бакалавра: 61 с., 21 рис., 23 джерел.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КОМПЬЮТЕРНА СИСТЕМА, ГИТАРНИЙ ТЮНЕР, ARDUINO, МУЗИЧНІ ІНСТРУМЕНТИ, АЛГОРИТМ ФУР'Є, ARDUINOFFT.

Метою кваліфікаційної роботи є дослідження алгоритму Швидкого Перетворення Фур'є (ШПФ) та його застосування у мобільних застосунках на базі платформи Arduino, використовуючи бібліотеку ArduinoFFT, знаходження можливостей його покращення.

Метою роботи є аналіз можливостей покращення бібліотеки ArduinoFFT для оптимізації використання пам'яті, зменшення часу обчислень, підвищення точності результатів, розширення функціоналу та підвищення енергоефективності. Робота зосереджена на теоретичному обґрунтуванні та пропозиціях щодо методів та підходів, які можуть бути використані для досягнення зазначених покращень.

Описано, протестовано та реалізовано найкращі методи та техніки для оптимізації використання ресурсів Arduino, включно з ефективним управлінням пам'яттю, скороченням обчислювального навантаження, застосуванням віконних функцій для підвищення точності спектрального аналізу, а також інтеграцією з різноманітними сенсорами для розширення можливостей застосування.

Процес дослідження включав теоретичний аналіз існуючих алгоритмів, їх можливостей та обмежень. На основі цього аналізу було розроблено ефективніший код для швидкого аналізу Фур’є та надано рекомендації для його подальшої оптимізації.

ABSTRACT

Master’s thesis contains 61 pages, 21 figures, 23 sources according to the list of links.

SOFTWARE, COMPUTER SYSTEM, GUITAR TUNER, ARDUINO, MUSICAL INSTRUMENTS, FOURIER ALGORITHM, ARDUINOFFT.

The goal of this qualification project is to investigate the Fast Fourier Transform (FFT) algorithm and its application in mobile applications based on the Arduino platform, using the ArduinoFFT library, and to identify opportunities for its improvement.

The work aims to analyze ways of enhancing the ArduinoFFT library to optimize memory usage, reduce computation time, increase result accuracy, expand functionality, and improve energy efficiency. It focuses on the theoretical justification and proposed methods and approaches that can be employed to achieve these improvements.

The best methods and techniques for optimizing Arduino resource usage have been described, tested, and implemented, including efficient memory management, reduction of computational load, use of window functions to enhance spectral analysis accuracy, as well as integration with various sensors to broaden the range of possible applications.

The research process involved a theoretical analysis of existing algorithms, their capabilities, and limitations. Based on this analysis, more efficient code for rapid Fourier analysis was developed, along with recommendations for further optimization.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE – інтегроване середовище розробки (англ. Integrated Development Environment);

PWM – модуляція ширини імпульсу (англ. Pulse Width Modulation);

USB – універсальна послідовна шина (англ. Universal Serial Bus);

ПЗ – програмне забезпечення.

ВСТУП

Автоматичний гітарний тюнер – комп’ютерна система спеціального призначення, а саме система для зчитування звукових хвиль, перетворення їх на електромагнітні сигнали, обробки та передачі необхідних сигналів на мотор що буде проводити настройку гітари.

Сучасний розвиток електроніки та обчислювальної техніки дозволяє створювати високотехнологічні пристрої для виконання специфічних завдань у різних галузях. Однією з таких галузей є цифрова обробка сигналів, зокрема в контексті музичних технологій. Алгоритми аналізу Фур'є, які використовуються для розкладу сигналів на частотні складові, є базовими інструментами для обробки аудіосигналів, але їхнє використання на мікроконтролерних платформах, таких як Arduino, вимагає оптимізації з урахуванням обмежених ресурсів.

Ця робота спрямована на розробку програмно-апаратної реалізації розширеного автоматичного тюнера, що базується на вдосконаленому алгоритмі аналізу Фур'є. Основна увага приділяється оптимізації бібліотеки ArduinoFFT та її адаптації для роботи з аудіосигналами в реальному часі.

У межах дослідження передбачається теоретичний аналіз існуючих алгоритмів, їх переваг і недоліків, розробка та тестування вдосконаленого програмного забезпечення, а також впровадження нових функцій для підвищення точності й розширення можливостей тюнера. Очікується, що запропоновані методи дозволять створити універсальний інструмент для музикантів і розробників, який поєднає високу продуктивність із мінімальними витратами енергії.

Таким чином, ця робота має на меті не лише вдосконалення програмного забезпечення, але й внесок у розвиток цифрової обробки сигналів та її застосування в портативних пристроях.

1 АНАЛІЗ ЗАВДАННЯ

1.1 Поняття, концепція і основні можливості автоматичного гітарного тюнеру

Автоматичний гітарний тюнер – це пристрій або програмне забезпечення, створене для точного налаштування гітарних струн без активної участі музиканта у процесі налаштування. Його основна мета — забезпечити максимально точну відповідність частот коливань струн до встановлених стандартних частот. Це досягається за рахунок об'єднання електроніки, алгоритмів обробки сигналів і механічних елементів, таких як серводвигуни, які можуть фізично підкручувати або послаблювати струни.

Концепція автоматичного тюнера ґрунтується на аналізі звукового сигналу, отриманого від струни, що коливається. Зазвичай цей сигнал зчитується через мікрофон, п’єзоелектричний датчик або безпосередньо через електрогітарний звукознімач. Потім сигнал проходить через алгоритм швидкого перетворення Фур'є (FFT), що дозволяє розкласти його на частоти, які складають цей звук. Алгоритм ідентифікує частоту основного тону, тобто тієї частоти, яка сприймається як «нота».

Основні можливості автоматичного тюнера включають здатність розпізнавати частоту звучання струни, порівнювати її з еталонною частотою для відповідної ноти, а також автоматично налаштовувати струну до необхідного рівня. Процес налаштування часто передбачає використання серводвигуна, що під’єднаний до колка гітари. Серводвигун коригує натяг струни, доки частота не стане відповідною еталонній.

Однією з ключових переваг автоматичного тюнера є точність, яка досягається завдяки цифровій обробці сигналу. Алгоритми дозволяють з високою роздільною здатністю аналізувати навіть незначні відхилення частоти. Крім того, такий пристрій забезпечує швидкість налаштування, що

особливо цінно для музикантів, які виступають наживо або працюють у студії, де економія часу є критичною.

Важливим аспектом концепції автоматичного гітарного тюнера є інтеграція інтерфейсу для взаємодії з користувачем. На практиці це може бути LED-індикація, що сигналізує про статус налаштування струни, звуковий сигнал, який сповіщає про завершення процесу, або навіть додаток на смартфоні для більш тонкого налаштування параметрів.

Автоматичний тюнер також може працювати в умовах зовнішніх шумів завдяки використанню високоякісних фільтрів, які ізолюють звук конкретної струни від сторонніх звуків. Це робить пристрій ефективним навіть у несприятливих акустичних умовах.

Таким чином, автоматичний гітарний тюнер є сучасним інструментом, який об’єднує принципи цифрової обробки сигналів, інженерні рішення в механіці й інтуїтивно зрозумілий інтерфейс, забезпечуючи музикантам точне, швидке і зручне налаштування їхніх інструментів.

1.2 Апаратна частина

Для створення тюнера була обрана Arduino UNO R3, яка є одним із найпопулярніших рішень для прототипування електронних пристроїв. Ця плата базується на мікроконтролері ATmega328P, що забезпечує високий рівень функціональності та простоту використання для розробників. Вона оснащена 14 цифровими портами, які можна налаштувати як входи або виходи (шість із них підтримують PWM-сигнали), а також 6 аналоговими входами.

У плату інтегрований кварцовий резонатор із частотою 16 МГц, що гарантує стабільну роботу мікроконтролера. Для підключення до комп’ютера передбачено USB-роз'єм, який також можна використовувати для живлення плати. Додатково плата має роз’єм ICSP для прошивки програмного забезпечення, стандартний роз’єм живлення та кнопку для скидання налаштувань.

Для автономного живлення обрано 9-вольтову батарейку типу «Крона». У специфікації зазначено, що плата може працювати з джерелами живлення в діапазоні від 6 до 20 В. Водночас рекомендовано дотримуватись оптимальних значень: при напрузі менше 7 В стабільність роботи може порушуватися, а перевищення 12 В може спричинити перегрівання компонентів, що вимагатиме додаткового охолодження.

Arduino UNO вирізняється широким спектром застосування завдяки підтримці численних бібліотек і доступу до великої спільноти розробників. Це значно спрощує створення та налаштування різноманітних електронних проектів, зводячи до мінімуму необхідність роботи на низькому рівні програмування.

1.3 Програмна частина

Програмна частина розробки є ключовим компонентом проєкту, оскільки саме в ній реалізуються алгоритми аналізу Фур'є, що лежать в основі роботи автоматичного тюнера. Для розробки програмного забезпечення використовуватимуться інструменти, що забезпечують ефективну роботу на платформі Arduino.

Arduino IDE є основним інструментом для написання, тестування та завантаження коду на платформу Arduino. Це інтегроване середовище підтримує мову програмування C/C++ і надає зручний інтерфейс для роботи з бібліотеками, тестування коду та завантаження програм у мікроконтролер. Середовище також забезпечує швидкий доступ до менеджера бібліотек, що дозволяє легко інтегрувати сторонні пакети, такі як ArduinoFFT, до поточного проєкту.

Arduino IDE є зручним для розробників завдяки широкій підтримці апаратного забезпечення, багатоплатформенності та активній спільноті, яка надає готові рішення та документацію. У межах даного проєкту середовище використовується для написання базового коду, налаштування взаємодії між

компонентами системи та тестування розроблених алгоритмів.

Основою для реалізації алгоритму аналізу Фур'є є бібліотека ArduinoFFT, яка широко використовується для роботи з Швидким Перетворенням Фур'є (ШПФ) на платформі Arduino. Ця бібліотека надає базовий набір функцій для обчислення частотних компонентів сигналу, включаючи:

– обробку дискретизованих сигналів;

– обчислення спектру частот із заданою кількістю вибірок;

– аналіз отриманих даних у частотній області.

Вона є компактною бібліотекою, яка оптимізована для використання в умовах обмежених ресурсів мікроконтролерів. Однак, її базова реалізація має певні обмеження, такі як мінімальна підтримка віконних функцій, відсутність оптимізацій для енергоспоживання та порівняно висока складність обчислень для великих вибірок.

У межах цього проєкту бібліотека ArduinoFFT слугуватиме базою для вдосконалення програмної частини. Планується провести оптимізацію її роботи шляхом:

– зменшення обсягу використовуваної оперативної пам’яті;

– впровадження додаткових віконних функцій для покращення точності спектрального аналізу;

– оптимізації алгоритму для підвищення енергоефективності;

– розширення функціоналу, наприклад, шляхом додавання функцій для роботи з шумами чи інтеграції з іншими сенсорами.

Для реалізації програмної частини будуть використовуватися модульні підходи, що забезпечать легкість інтеграції покращень до базової бібліотеки. Структура коду буде побудована так, щоб забезпечити зручність у тестуванні й налаштуванні окремих компонентів. Зокрема, планується створення окремих модулів для:

– зчитування сигналів з мікрофона;

– попередньої обробки даних (фільтрація та нормалізація);

– виконання спектрального аналізу з використанням вдосконаленої версії ArduinoFFT;

Програмна частина розробляється з урахуванням можливостей масштабування та розширення функціональності у майбутньому, що дозволить адаптувати систему для різних завдань спектрального аналізу.

1.4 Огляд існуючих рішень

У сучасному світі алгоритми аналізу Фур'є знаходять широке застосування в різних галузях, таких як обробка аудіосигналів, телекомунікації, медицина та наукові дослідження. Однак їх адаптація для портативних пристроїв, зокрема на базі мікроконтролерів, таких як Arduino, має свої особливості та обмеження. У цьому контексті важливим є аналіз існуючих бібліотек і підходів, які застосовуються для реалізації алгоритмів Фур'є на пристроях із обмеженими ресурсами.

Одним із найпоширеніших рішень для реалізації алгоритму швидкого перетворення Фур'є (ШПФ) на платформах Arduino є бібліотека ArduinoFFT, через її розповсюдженість саме її і було обрано для покращення. Ця бібліотека забезпечує базову функціональність для аналізу частотних компонентів сигналу з використанням ШПФ. Її основними перевагами є простота інтеграції та доступність для розробників. Однак бібліотека має деякі недоліки, зокрема відносно високу обчислювальну складність, обмежену точність і відсутність розширених функцій для покращення спектрального аналізу, таких як застосування віконних функцій чи фільтрація шуму.

Серед альтернативних рішень варто зазначити бібліотеки для інших платформ, таких як CMSIS-DSP для мікроконтролерів ARM Cortex-M. Ця бібліотека забезпечує високу продуктивність і підтримує широкий спектр функцій цифрової обробки сигналів, включаючи оптимізовані реалізації ШПФ. Однак її використання на платформах Arduino є обмеженим через

відмінності в архітектурі та необхідність значних обчислювальних ресурсів.

Для більш потужних мікроконтролерів та комп’ютерів застосовуються бібліотеки, такі як NumPy і SciPy у середовищі Python, а також MATLAB. Вони пропонують широкий функціонал для спектрального аналізу, включаючи можливість застосування різних типів вікон, аналізу сигналів у реальному часі та фільтрації. Проте ці інструменти зазвичай використовуються в стаціонарних умовах і не призначені для роботи на пристроях з обмеженими ресурсами.

У галузі портативних пристроїв та мікроконтролерів реалізація алгоритмів Фур'є часто обмежується базовими потребами, такими як визначення основних частотних компонентів аудіосигналу або фільтрація. Наприклад, у мобільних застосунках для аналізу звуку часто використовуються вбудовані API, такі як Android Audio Framework, які забезпечують доступ до обчислень ШПФ через готові бібліотеки. Однак такі рішення є менш гнучкими та не дозволяють повністю контролювати процес аналізу сигналу.

Таким чином, існуючі рішення для реалізації алгоритмів Фур'є на портативних пристроях мають як переваги, так і обмеження. Це відкриває можливості для подальших досліджень і розробки оптимізованих алгоритмів, які поєднують високу продуктивність, точність та енергоефективність, адаптовану до потреб сучасних мікроконтролерних систем, таких як Arduino.

2 ПОСТАНОВКА ЗАДАЧІ ТА ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Актуальність дослідження

Актуальність дослідження обумовлена широким застосуванням аналізу Фур'є в різних областях, як цифрова обробка сигналів, телекомунікації, аудіоінженерія, медична діагностика та багато інших. Оптимізація цих процесів через розуміння та застосування ДПФ та ШПФ може значно підвищити ефективність обробки сигналів, що має велике практичне та теоретичне значення. В контексті швидко розвиваючихся технологій, здатність точно та швидко аналізувати звукові хвилі є важливим фактором у розвитку нових інноваційних рішень.

2.2 Цілі та завдання дослідження

Основною ціллю цього дослідження є покращення алгоритму аналізу звукових хвиль, зокрема для застосування в мікроконтролерних системах, як Arduino. Це покращення буде зосереджене на оптимізації ефективності та точності існуючих методів обробки звуку, з акцентом на фільтруванні шумів та підвищенні точності визначення тональності сигналів. Ця робота буде базуватися на попередніх дослідженнях та розробках, зокрема на створенні автоматичного гітарного тюнера.

2.3 Теоретичні основи фналізу Фур'є

Аналіз Фур'є є потужним математичним інструментом, який дозволяє розкладати складні сигнали на суму простих періодичних компонентів. В основі цього методу лежить теорія, згідно з якою будь-яку функцію часу, яка задовольняє певним умовам, можна подати у вигляді суми синусів і косинусів з різними частотами, амплітудами та фазами. Цей підхід забезпечує універсальність та гнучкість у дослідженні сигналів, що знаходить застосування у багатьох галузях науки й техніки.

Дискретне перетворення Фур'є (ДПФ) є одним із найважливіших інструментів у цифровій обробці сигналів. Його головна перевага полягає у здатності аналізувати сигнали, які були дискретизовані, тобто перетворені з неперервної форми в послідовність чисел. Завдяки цьому методу стає можливим виявлення частотних компонентів сигналу, що є фундаментальним для розуміння його структури та властивостей.

Швидке перетворення Фур'є (ШПФ) є оптимізованою версією ДПФ [3]. Воно дозволяє суттєво зменшити кількість необхідних обчислень для перетворення, що особливо важливо при обробці великих даних або в реальному часі. Ефективність ШПФ забезпечує можливість швидшого аналізу сигналів без значної втрати інформації, що робить його незамінним інструментом у багатьох прикладних задачах.

Вперше концепція перетворення Фур'є була запропонована Жаном Батистом Жозефом Фур'є у 1807 році. Його ідея полягала в тому, що будь-яку функцію можна представити як суму тригонометричних рядів. Незважаючи на початковий скептицизм наукової спільноти, теорія Фур'є з часом здобула визнання завдяки своїй універсальності та практичній значущості. Від опису розподілу тепла в твердих тілах вона поширилася на фізику, інженерію, обробку сигналів і навіть статистику, ставши ключовим інструментом для аналізу як неперервних, так і дискретних сигналів.

2.4 Принципи дискретного перетворення Фур'є (ДПФ)

Дискретне перетворення Фур'є (ДПФ) є одним із найважливіших інструментів у цифровій обробці сигналів. Воно дозволяє аналізувати сигнали, які представлені у формі послідовностей чисел, отриманих у результаті дискретизації або вимірювань. Центральною ідеєю ДПФ є перетворення сигналу з часової області, де він подається як послідовність значень у часі, у частотну область. У частотній області сигнал представляється як сума синусоїд і косинусоїд із різними частотами, амплітудами та фазами, що дає змогу виявити його основні частотні компоненти.

Математично ДПФ визначається через набір рівнянь, які перетворюють дискретний часовий сигнал у набір коефіцієнтів Фур'є. Ці коефіцієнти характеризують амплітуду і фазу кожної частоти сигналу, дозволяючи отримати повний спектральний опис його структури [4].

ДПФ знаходить широке застосування в аналізі різних типів сигналів, включаючи аудіо- та радіочастотні сигнали [5]. Його головними перевагами є здатність точно визначати частотні компоненти та висока швидкість обчислень, особливо при використанні сучасних комп'ютерних систем. Окрім цього, ДПФ є основою для розробки більш складних методів обробки сигналів, зокрема швидкого перетворення Фур'є (ШПФ), що дозволяє ще більше оптимізувати процес аналізу.

2.5 Виклики та обмеження

Однак, ДПФ має певні обмеження, зокрема відносно розміру вибірки та роздільної здатності. Розмір вибірки впливає на кількість доступних частотних компонентів, тоді як роздільна здатність – на точність визначення цих компонентів. Ці обмеження вимагають відповідного підходу до вибірки та аналізу сигналу для отримання достовірних результатів.

2.6 Принципи швидкого перетворення Фур'є (ШПФ)

Швидке перетворення Фур'є (ШПФ) є вдосконаленою версією дискретного перетворення Фур'є (ДПФ), яка забезпечує значно швидше обчислення частотних компонентів сигналу. Основна перевага ШПФ полягає в суттєвому зменшенні кількості необхідних обчислень, що робить його особливо ефективним при роботі з сигналами, що містять велику кількість даних. Завдяки цьому ШПФ стало одним із ключових інструментів цифрової обробки сигналів, оскільки воно дозволяє виконувати аналіз у реальному часі та знаходить застосування у багатьох сферах.

Математично ШПФ базується на тих самих принципах, що й ДПФ, однак використовує оптимізовані алгоритми для скорочення обчислювальної складності. Цей метод реалізується за допомогою підходу "поділ та володарювання", який передбачає розбиття великого ДПФ на менші, більш ефективні підзадачі, які обробляються рекурсивно. Такий підхід значно зменшує обчислювальну складність – із $O(N^2)$ у ДПФ до $O(N*\log N)$, де N – кількість дискретних точок сигналу. Ця оптимізація робить ШПФ надзвичайно ефективним для обробки великих масивів даних [6].

ШПФ широко застосовується в задачах, де необхідна швидка обробка даних, таких як обробка аудіо- та відеосигналів, телекомунікації, медична візуалізація та багато інших. Його здатність ефективно обробляти сигнали в реальному часі дозволяє створювати системи, що відповідають сучасним вимогам до швидкодії.

Попри всі переваги, ШПФ має свої обмеження. Зокрема, алгоритм демонструє зниження ефективності при роботі з сигналами, які мають нерівномірні часові інтервали або неперіодичні властивості. Крім того, якість результатів ШПФ залежить від точності вихідного дискретизованого сигналу, що накладає певні вимоги на етап попередньої обробки даних. Ці аспекти вимагають додаткової уваги при розробці систем, що використовують ШПФ.

2.7 Математичні основи ДПФ та ШПФ

ДПФ перетворює послідовність N комплексних чисел x0, x1, … xN-1, у іншу послідовність комплексних чисел X0, X1, … XN-1, які представляють амплітуди та фази різних частотних компонентів початкового сигналу. Математично ДПФ визначається наступним чином:

$$X[k] = \sum_{n=1}^{N-1} \left(x[n] * e^{-\frac{i2\pi kn}{N}} \right), \quad (2.1)$$

де:

Xk – k–ий коефіцієнт Фур'є;

xn – n–е значення в часовій послідовності;

N – загальна кількість точок у сигналі;

i – уявна одиниця[7][8].

Ця формула дозволяє перейти від часової області (де сигнал представлений як послідовність значень у часі) до частотної області (де сигнал представлений як сума синусів та косинусів).

ШПФ – це алгоритм для швидкого обчислення ДПФ, який значно зменшує кількість необхідних обчислень. Основна ідея ШПФ полягає у рекурсивному розділенні ДПФ на менші ДПФ. Найпопулярніша форма ШПФ – це радикс–2 алгоритм, який передбачає, що N є ступенем двійки (тобто N=2m =2m для деякого цілого m)[9].

В радикс–2 ШПФ, ДПФ розділяється на дві частини: одна для парних індексів і одна для непарних. Таке розділення дозволяє виконувати обчислення більш ефективно. Математично це можна виразити так:

$$X_k = E_k + e^{-\frac{i2\pi k}{N}} O_k, \quad (2.2)$$

де Ek та Ok – це ДПФ для парних та непарних елементів відповідно.

Ці рівняння застосовуються рекурсивно до тих пір, поки не досягнуться окремі елементи, тобто до тих пір, поки ДПФ не буде розділено до рівня одного елемента.

2.8 Порівняння ДПФ та ШПФ з точки зору ефективності обчислень

Дискретне перетворення Фур'є (ДПФ) є основним методом для переведення сигналу з часової області у частотну. Однак, при виконанні без будь-яких оптимізацій, обчислювальна складність ДПФ становить $O(N^2)$, де N – кількість дискретних точок сигналу. Така квадратична залежність часу обчислень від розміру сигналу є значним обмеженням, особливо для великих обсягів даних. Прикладом може бути сигнал з 1024 точками, для обробки якого ДПФ потребуватиме понад мільйон обчислень [10].

Швидке перетворення Фур'є (ШПФ) є вдосконаленням ДПФ, яке знижує обчислювальну складність до O(N*logN). Завдяки цьому час обчислень зростає лише логарифмічно зі збільшенням розміру сигналу, що робить ШПФ значно ефективнішим. Наприклад, той самий сигнал із 1024 точками ШПФ обробить за приблизно 10 тисяч обчислень, що демонструє колосальну економію ресурсів і часу.

У практичному застосуванні ця обчислювальна ефективність ШПФ відкриває можливості для обробки сигналів у реальному часі, чого неможливо досягти за допомогою традиційного ДПФ. ШПФ використовується у широкому спектрі завдань, включаючи обробку аудіо- та відеосигналів, радіолокацію, сейсмічний аналіз і багато інших. Його здатність швидко й точно обробляти великі обсяги даних робить ШПФ незамінним інструментом у сучасних технологіях, де швидкість і продуктивність відіграють ключову роль.

2.9 Опис процесу аналізу звукових хвиль

Звукові хвилі є вібраційними хвилями, які поширюються через повітря або інші середовища і можуть бути зафіксовані як аналогові або цифрові сигнали. Для їхнього дослідження важливим є перетворення сигналу з часової області, де він представлений як амплітуда в часі, у частотну область. У частотній області сигнал подається як сума синусоїд і косинусоїд із різними частотами, що дозволяє визначити частотні компоненти сигналу. Для цього часто використовуються дискретне перетворення Фур'є (ДПФ) або його вдосконалена версія – швидке перетворення Фур'є (ШПФ).

Процес аналізу звукових хвиль за допомогою ДПФ передбачає кілька етапів. Спочатку звуковий сигнал піддається дискретизації, тобто перетворюється на послідовність цифрових значень, які відображають амплітуду хвилі в різні моменти часу. Після цього застосовується ДПФ, що перетворює часові дані у частотний спектр, дозволяючи виявити основні частотні компоненти сигналу, їх амплітуди і фази. Отриманий спектр надає важливу інформацію про ключові характеристики звуку, такі як висота, гучність і тембр.

Швидке перетворення Фур'є (ШПФ) реалізує аналогічний підхід, але з використанням оптимізованих алгоритмів, які дозволяють виконувати обчислення значно швидше. Це особливо важливо для обробки великих обсягів даних або при роботі в реальному часі. Як і у випадку з ДПФ, аналіз починається з дискретизації сигналу, після чого ШПФ забезпечує швидке перетворення часових даних у частотну область. Це дає змогу отримувати точні результати, які використовуються для ідентифікації звуків, шумів або інших характеристик сигналу.

Аналіз звукових хвиль знаходить широке застосування у різних галузях. У музичній індустрії та аудіоінженерії він використовується для покращення якості звуку, аналізу гармоній та налаштування музичних інструментів. У сфері акустичної діагностики методи аналізу звукових сигналів дозволяють

виявляти дефекти обладнання або аномалії в роботі систем. У медицині аналіз звукових хвиль, наприклад, серцебиття або дихання, допомагає діагностувати різноманітні захворювання. Таким чином, перетворення Фур'є є універсальним інструментом для дослідження звукових сигналів у багатьох сферах.

3 ПОКРАЩЕННЯ АЛГОРИТМУ АНАЛІЗУ ФУР’Є

3.1 Приклади використання ДПФ та ШПФ в аудіоаналітиці

Дискретне перетворення Фур'є (ДПФ) та швидке перетворення Фур'є (ШПФ) відіграють ключову роль у сучасній аудіоаналітиці, оскільки вони є основними інструментами для аналізу та обробки звукових сигналів. Ці методи знаходять застосування в багатьох сферах, зокрема у аудіоінженерії, розпізнаванні мовлення, діагностиці та цифровій обробці сигналів.

Одним із головних напрямків застосування ДПФ та ШПФ є аудіоінженерія. Ці методи використовуються для аналізу частотного спектру музичних треків, що дозволяє точно налаштовувати еквалізацію для покращення якості звучання. Також ШПФ застосовується для реставрації звуку – видалення шумів і артефактів зі старих записів, дозволяючи ідентифікувати та відокремлювати небажані частоти.

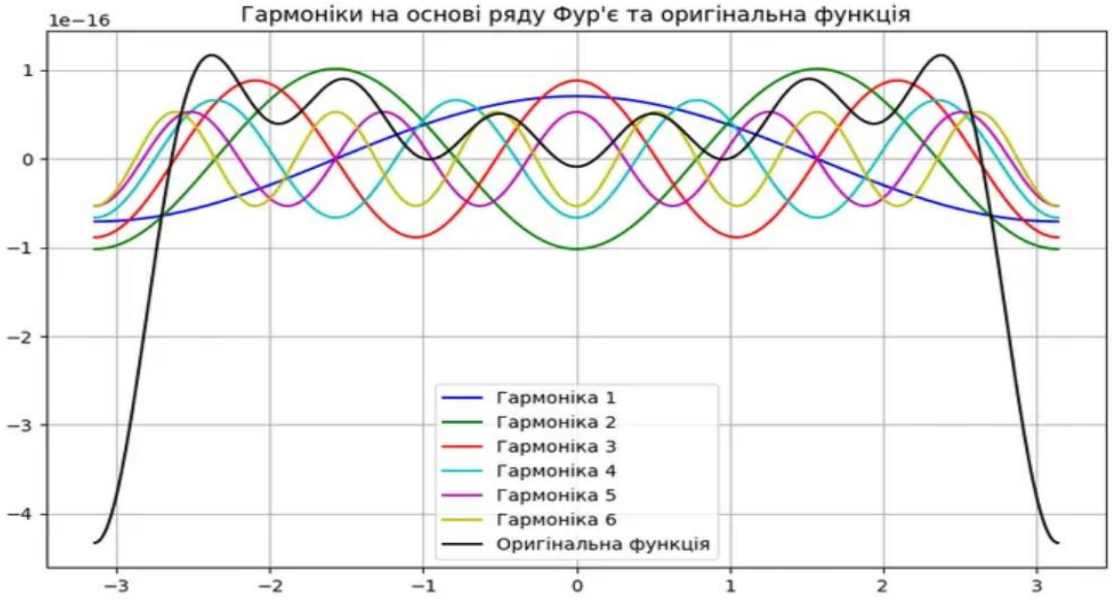

Рисунок 3.1 – Візуальна демонстрація аналізу Фур'є звукових хвиль

Методи ДПФ та ШПФ використовуються у системах розпізнавання мовлення для перетворення мовних сигналів у частотний спектр. Це важливо для розпізнавання слів, інтонацій та навіть емоцій. ШПФ також застосовується для швидкого аналізу мовлення в реальному часі, що є критичним для роботи голосових асистентів та інтерактивних IVR-систем.

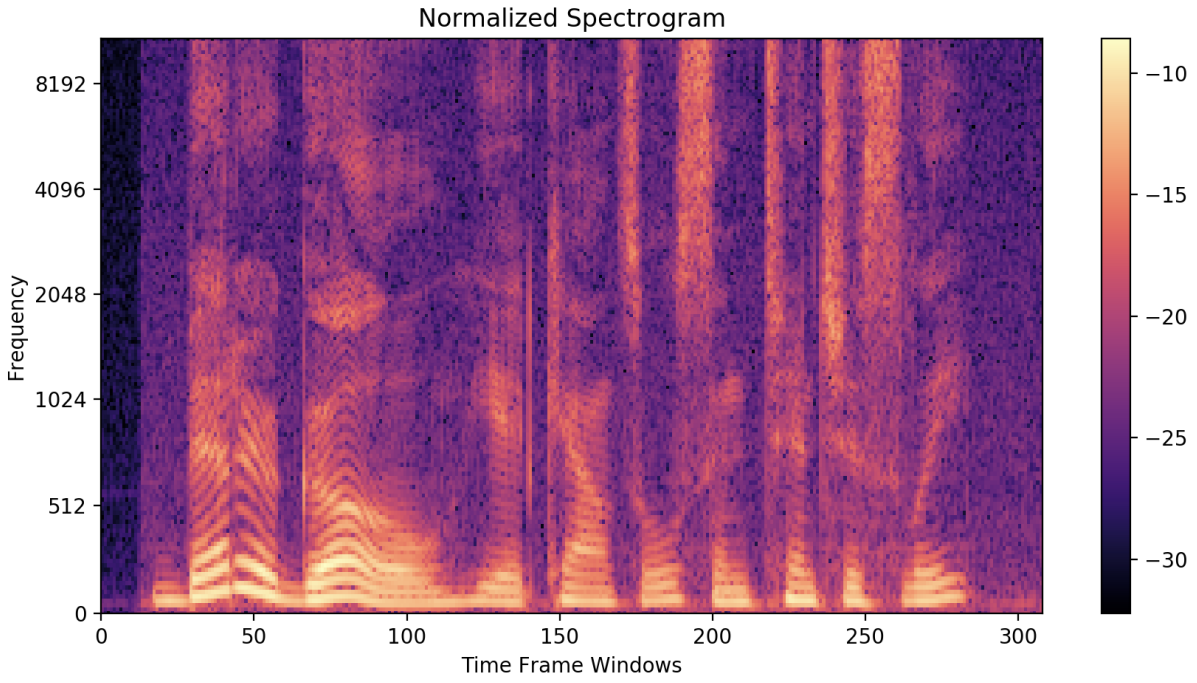

Рисунок 3.2 – Нормалізована спектограмма аналізом Фур'є для розпізнавання голосу

ДПФ та ШПФ знаходять широке застосування у сфері діагностики. Наприклад, у медицині вони використовуються для аналізу звукових хвиль, які виникають у тілі пацієнта, таких як серцебиття, для діагностики різних захворювань [11]. У промислових умовах ці методи допомагають в акустичному моніторингу, дозволяючи виявляти несправності обладнання на ранніх стадіях [12].

Швидке перетворення Фур'є є важливим інструментом у телекомунікаціях, де його використовують для ефективного кодування та декодування аудіосигналів [13]. Крім того, ДПФ та ШПФ застосовуються для аналізу радіочастотних сигналів, що дозволяє визначити доступний частотний діапазон і виявити завади [14].

У сфері музичних технологій методи ДПФ та ШПФ дозволяють автоматично створювати музику, аналізуючи стилі та генеруючи композиції [15] [16]. Крім того, вони є основою для розробки автоматичних гітарних тюнерів, які з високою точністю визначають висоту звуку гітарних струн [17].

Ці приклади демонструють широкий спектр застосувань ДПФ та ШПФ у реальних задачах, що робить їх універсальними інструментами для аналізу та обробки звукових сигналів.

3.2 Застосування ШПФ в застосунках на базі Arduino

Arduino-пристрої, незважаючи на обмежені обчислювальні ресурси, є ідеальною платформою для реалізації швидкого перетворення Фур'є (ШПФ) завдяки своїй адаптивності та доступності. Використання ШПФ на Arduino дозволяє здійснювати аналіз аудіосигналів із мікрофонів у реальному часі, що відкриває широкі можливості для розробки мобільних пристроїв у різних сферах.

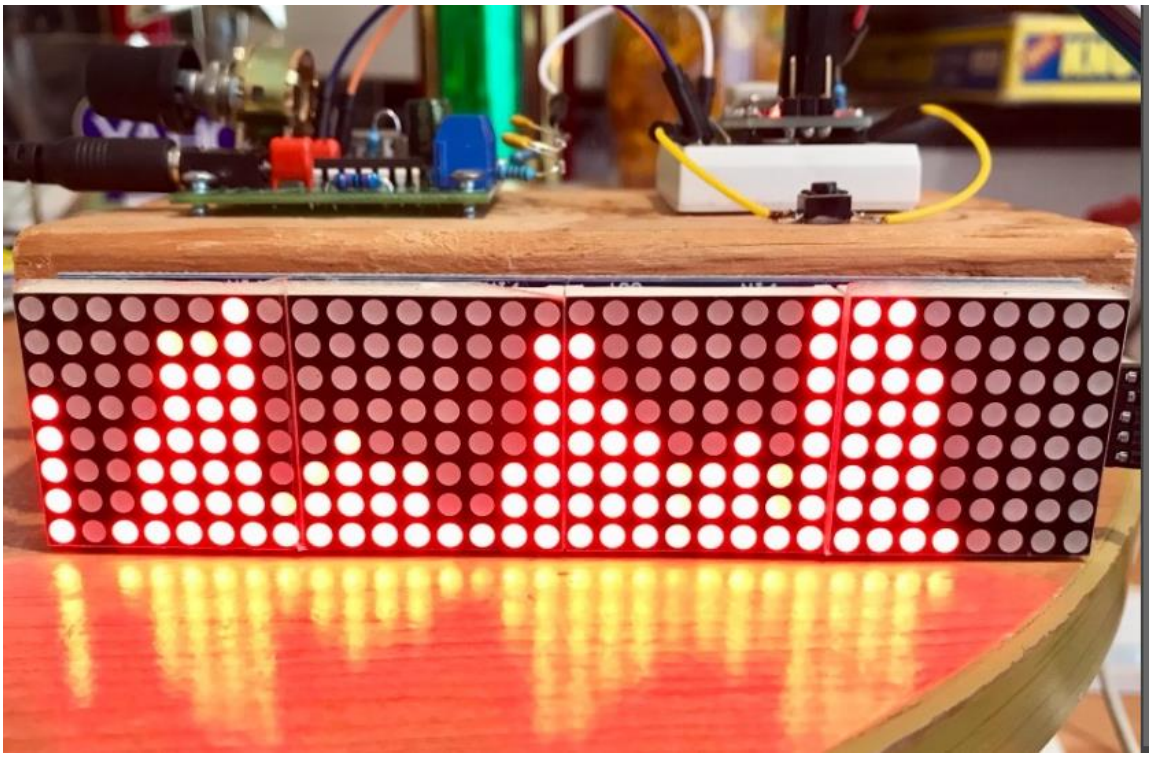

Рисунок 3.4 – Система аналізу звуку на основі плати Arduio

Одним із ключових напрямків використання ШПФ на Arduino є створення портативних аудіоаналітичних пристроїв, які можуть виконувати моніторинг рівнів шуму або аналізувати музичні тони та гармонії. Наприклад, такі пристрої можуть бути використані для оцінки шумового забруднення у міських або промислових зонах, допомагаючи визначати рівень шуму в реальному часі. Завдяки можливості інтеграції сенсорів, як-от мікрофони та датчики звуку, Arduino здатен обробляти зібрані дані за допомогою ШПФ для активізації механізмів відгуку, таких як попереджувальні сигнали або автоматизовані системи.

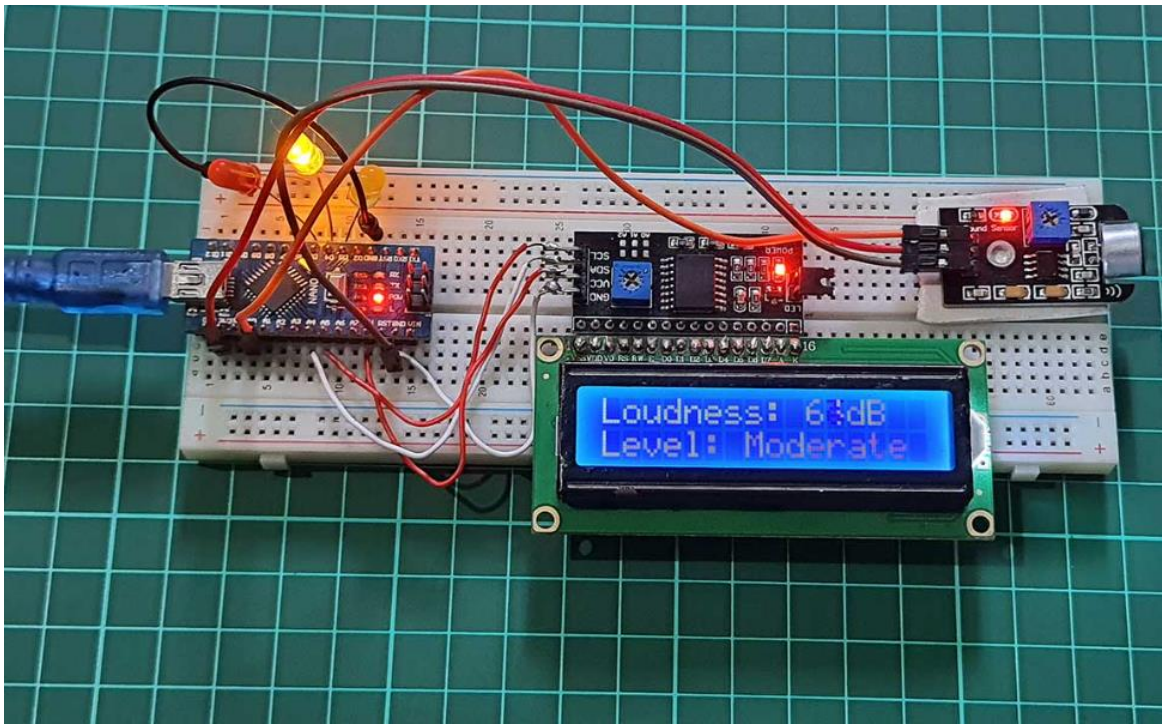

Рисунок 3.5 – Система аналізу шуму на основі плати Arduio

Для музикантів Arduino може слугувати основою для створення компактних пристроїв, що дозволяють аналізувати музичні гармонії та частотні характеристики інструментів. Завдяки ШПФ ці пристрої можуть забезпечити точне налаштування інструментів або допомогти коригувати звукові сигнали перед записом чи виконанням. Ця функціональність особливо корисна в мобільних умовах, наприклад, під час концертів або вуличних виступів.

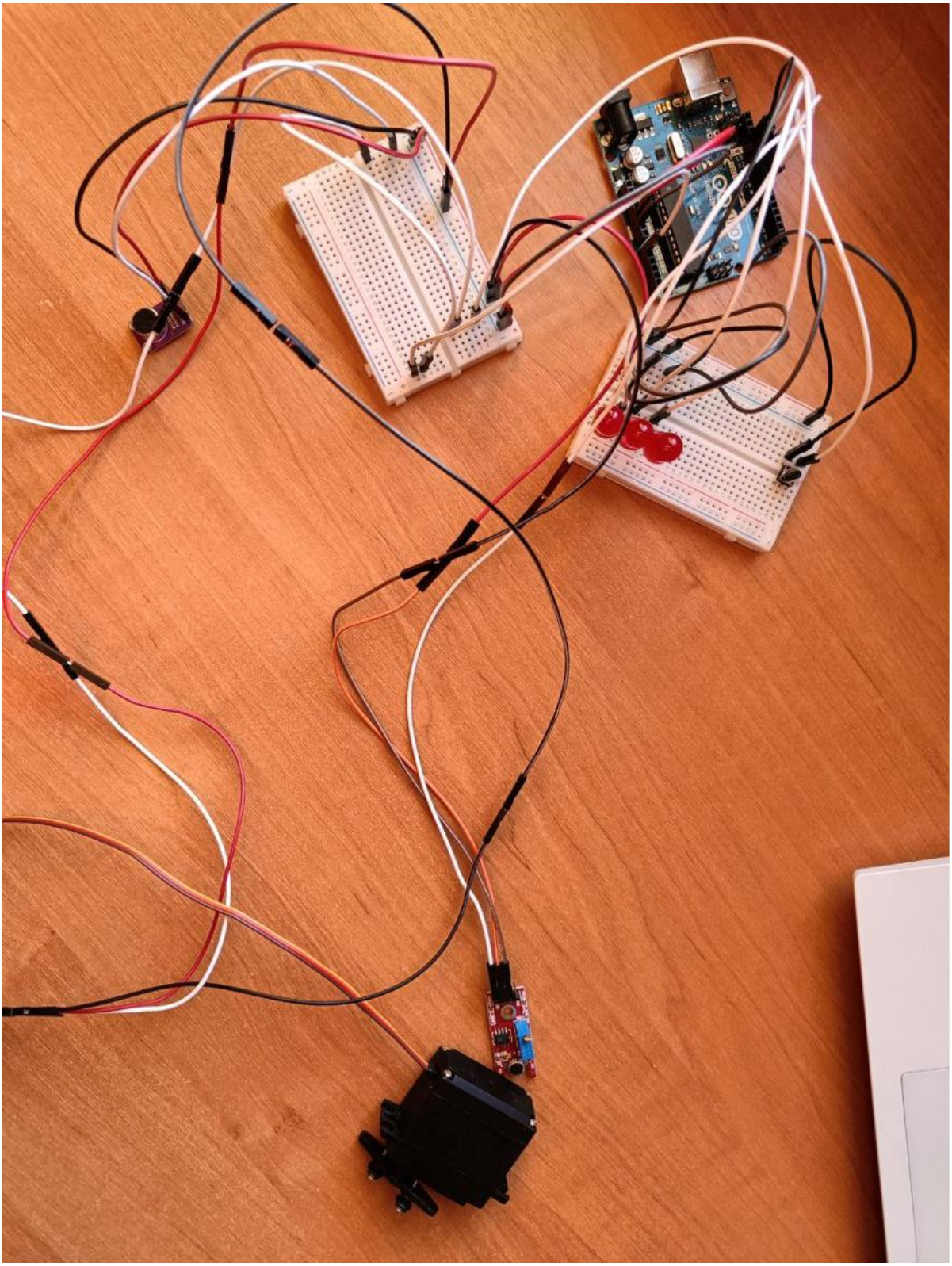

Рисунок 3.6 – Схема гітарного автотюнеру

Ще однією важливою сферою застосування є інтерактивність у фізичному просторі. Пристрої на базі Arduino, використовуючи ШПФ, можуть ідентифікувати звукові патерни, які активують інші механічні або електронні системи. Наприклад, вони можуть відкривати двері, вмикати світло або запускати автоматизовані системи на основі звуків, що надходять із середовища. Це також може бути використано для налаштування або навіть гри на музичних інструментах, створюючи нові можливості для інтерактивної музичної творчості.

Незважаючи на численні переваги, впровадження ШПФ на платформі Arduino супроводжується певними викликами. Обмежена пам’ять та потужність обробки вимагають оптимізації алгоритмів, що використовуються. Тут важливу роль відіграють бібліотеки, як-от ArduinoFFT, які дозволяють зменшити використання ресурсів, зберігаючи при цьому високу точність і швидкість обробки.

З розвитком технологій Інтернету речей (IoT) та удосконаленням мікроконтролерів, можливості використання ШПФ на Arduino продовжують зростати. Новітні досягнення у виробництві мікроконтролерів відкривають перспективи створення більш складних і ефективних застосунків, які можуть знайти своє місце в сучасних мобільних технологіях і автоматизованих системах.

3.3 Програмне Забезпечення для аналізу Фур'є

Аналіз Фур'є є однією з ключових операцій у багатьох завданнях цифрової обробки сигналів, і для його реалізації існує широкий спектр програмного забезпечення. Кожна з цих програм пропонує унікальні функції, що дозволяють реалізувати дискретне та швидке перетворення Фур'є залежно від потреб користувача.

MATLAB є потужним інструментом для чисельних обчислень, візуалізації та програмування, який включає в себе інструменти для реалізації ДПФ та ШПФ. Завдяки своїм функціональним можливостям MATLAB активно використовується у наукових дослідженнях, інженерії та навчальних програмах [18][19].

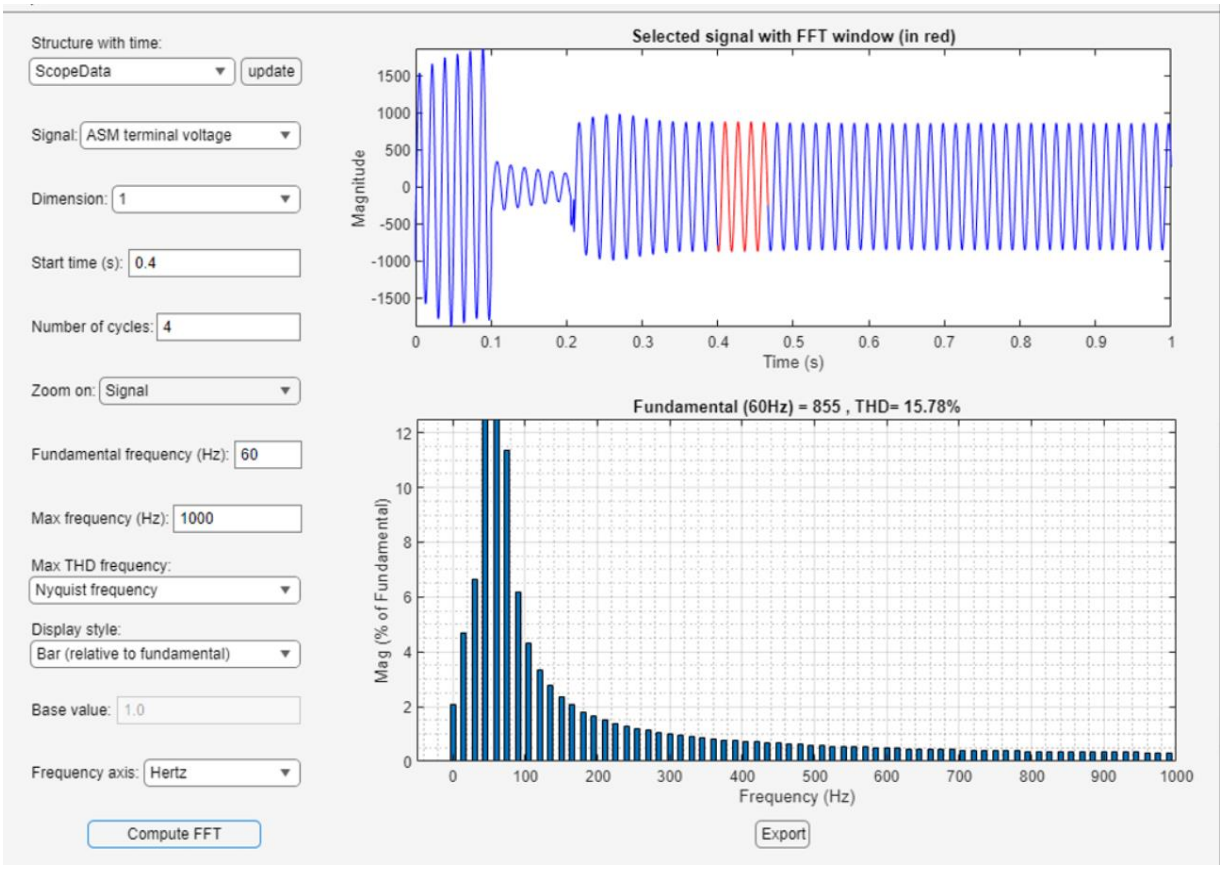

Рисунок 3.7 – Використання MATLAB для аналізу Фур'є

GNU Octave, схоже за функціоналом на MATLAB, пропонує багато аналогічних можливостей, включаючи підтримку функцій для ДПФ та ШПФ. Це програмне забезпечення є безкоштовною альтернативою MATLAB і широко застосовується в академічних середовищах [20].

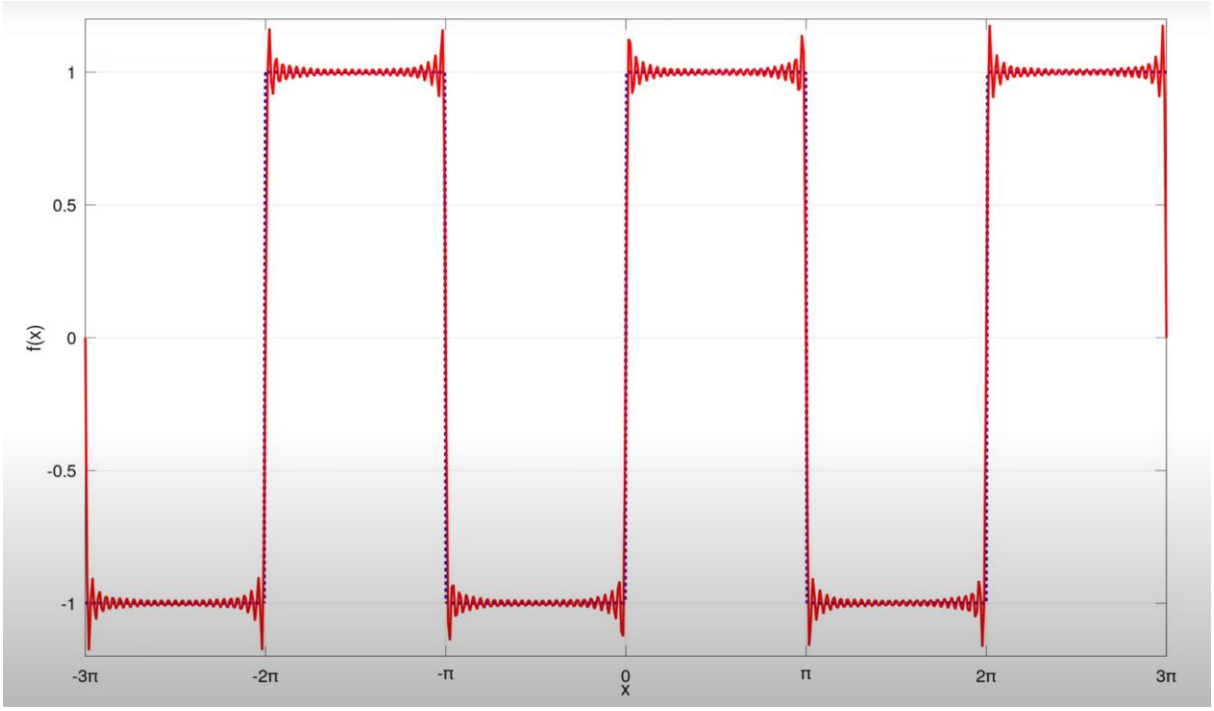

Рисунок 3.8 – Використання GNU Octave для аналізу Фур'є

Python з бібліотеками NumPy та SciPy є ще одним популярним інструментом для аналізу Фур'є. Ці бібліотеки надають потужні функції для виконання ДПФ та ШПФ, а також забезпечують засоби для візуалізації даних. Python активно використовується у сфері обробки даних, машинного навчання та наукових досліджень [21][22].

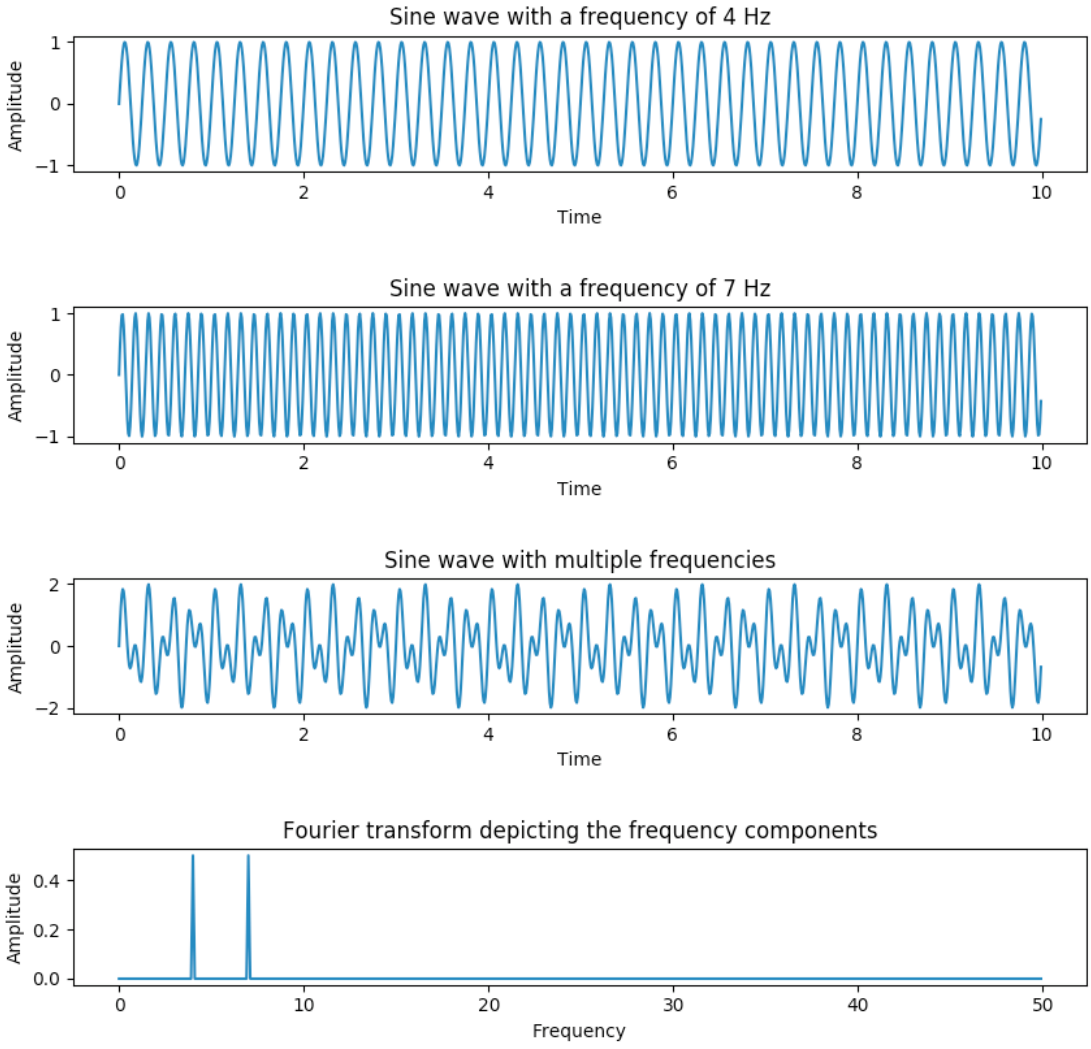

Рисунок 3.9 – Використання NumPy для аналізу Фур'є

Audacity є безкоштовним аудіоредактором із відкритим кодом, який забезпечує базові інструменти для аналізу та візуалізації частотних спектрів аудіосигналів. Ця програма підходить як для аматорів, так і для професійної аудіообробки, дозволяючи виконувати прості операції з аудіоаналізу.

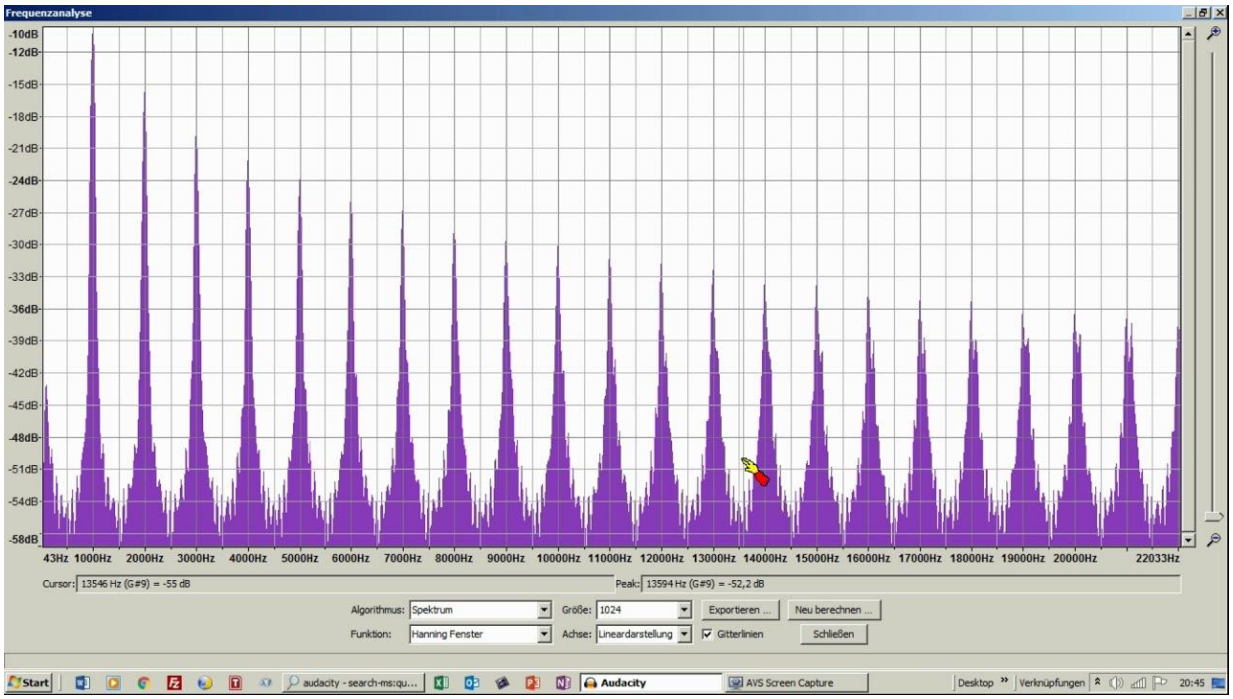

Рисунок 3.10 – Використання Audacity для аналізу Фур'є

Adobe Audition є професійним програмним забезпеченням для обробки аудіо, яке надає розширені інструменти для аналізу, включаючи ДПФ та ШПФ. Це програмне забезпечення широко використовується у професійній аудіоіндустрії для редагування, міксування та обробки звуку.

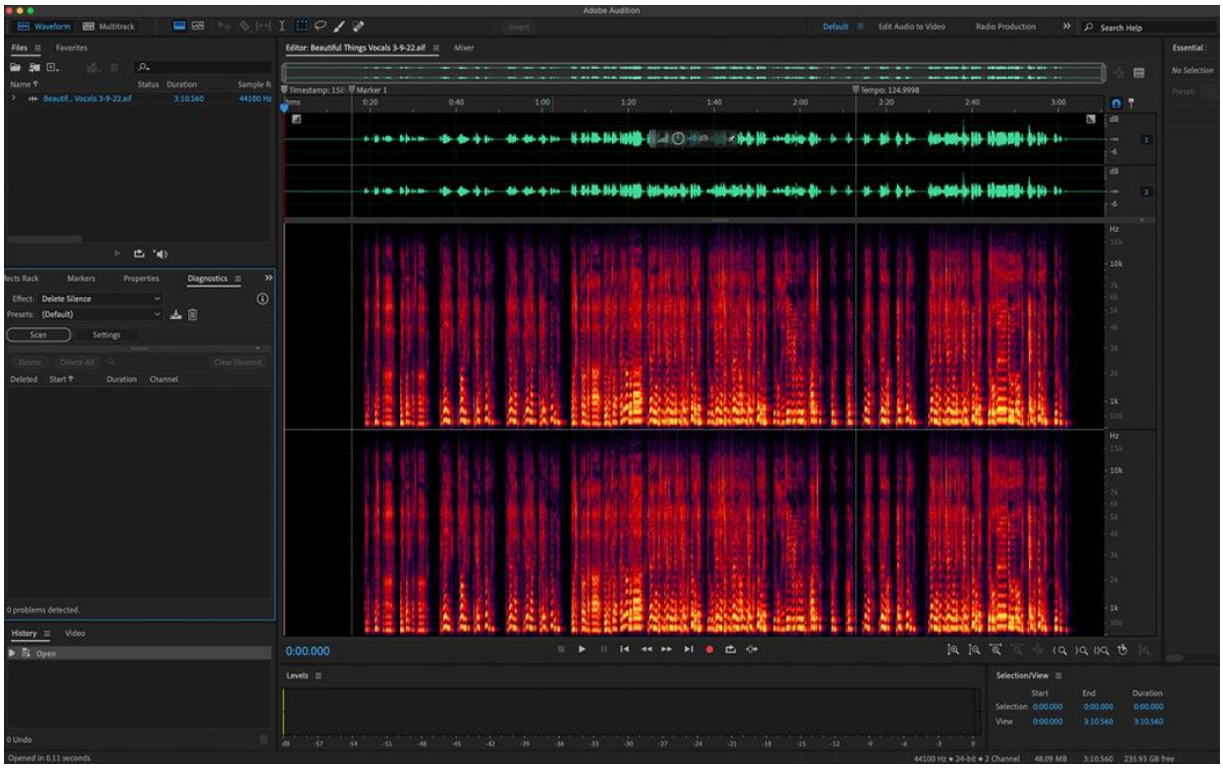

Рисунок 3.11 – Використання Adobe Audition для аналізу Фур'є

Ці програмні рішення демонструють широкий спектр можливостей для реалізації аналізу Фур'є, дозволяючи вибирати найбільш підходяще рішення залежно від специфічних вимог завдання.

3.4 Аналіз проблем алгоритму ArduinoFFT

ArduinoFFT є бібліотекою для Arduino, яка дозволяє виконувати швидке перетворення Фур'є (ШПФ). Однак, як і будь–яка бібліотека, вона може мати певні обмеження, особливо у випадках, де потрібна висока точність або ефективність. Ось декілька шляхів покращення алгоритму для ArduinoFFT:

Оптимізація використання пам'яті.

Проблема: Arduino має обмежену кількість пам'яті, що може обмежувати розмір вибірки ШПФ.[23]

Покращення: Ефективніше використання пам'яті, можливо через використання більш компактних форматів даних або оптимізацію алгоритмів зберігання.

Зменшення часу обчислень.

Проблема: Час обчислення може бути критичним для додатків у реальному часі.

Покращення: Впровадження більш ефективних алгоритмів ШПФ або використання методів апроксимації для зменшення кількості обчислень.

Підвищення точності.

Проблема: Точність може страждати через обмеження апаратного забезпечення Arduino.

Покращення: Калібрування алгоритму для компенсації апаратних неточностей або застосування методів згладжування для усунення шуму.

Розширення функціоналу.

Проблема: ArduinoFFT може не підтримувати деякі спеціалізовані функції.

Покращення: Додавання нових функцій, таких як виявлення специфічних частотних компонентів або інтеграція з іншими сенсорами для розширеного аналізу.

Енергоефективність.

Проблема: Arduino часто використовується в мобільних або віддалених додатках, де енергоефективність є важливою.

Покращення: Оптимізація алгоритму для зменшення споживання енергії, наприклад, через зниження тактової частоти або застосування режимів сну.

3.5 Аналіз вирішень проблем бібліотеки ArduinoFFT

ArduinoFFT є потужним інструментом для реалізації Швидкого Перетворення Фур'є (ШПФ) на платформі Arduino, але ефективне використання цієї бібліотеки вимагає оптимізації, спрямованої на подолання обмежень апаратних ресурсів, таких як пам'ять, швидкість обчислень, точність, функціональність і енергоефективність.

Оптимізація використання пам'яті є ключовим аспектом, оскільки ресурси Arduino дуже обмежені. Одним із підходів є зменшення розміру вхідних даних шляхом використання методів передобробки сигналу, таких як видалення несуттєвих частин або вибіркове зчитування даних. Інший підхід включає використання in-place алгоритмів, які зменшують потребу у тимчасових масивах, дозволяючи обчислення без використання додаткової пам’яті. Управління життєвим циклом змінних також є критичним: створення і звільнення об’єктів слід здійснювати лише тоді, коли це необхідно. Така оптимізація дозволяє значно підвищити надійність і точність аудіоаналізу, зменшуючи ризик переповнення пам'яті.

Зменшення часу обчислень є необхідним для обробки даних у реальному часі. Метод "подвійного буферування" дозволяє Arduino паралельно працювати з одним набором даних, тоді як інший зчитується. Використання оптимізованих математичних бібліотек і методів алгоритмічної оптимізації, таких як зменшення ітерацій у циклах, також сприяє підвищенню продуктивності. Застосування асинхронних методів і переривань дозволяє

виконувати обчислення, не перериваючи інші задачі, що значно підвищує ефективність.

Підвищення точності аналізу є важливим для забезпечення якості обробки сигналів. Це досягається використанням віконних функцій, таких як Хеммінга чи Блекмана, які зменшують "спектральні витоки" і покращують спектральний аналіз. Крім того, калібрування датчиків і оновлення бібліотеки до останньої версії сприяють зменшенню похибок і покращенню точності результатів.

Розширення функціоналу бібліотеки ArduinoFFT може включати додавання алгоритмів виявлення специфічних частотних компонентів, що корисно в акустичній діагностиці та системах безпеки. Інтеграція з іншими сенсорами, такими як акселерометри чи екологічні датчики, дозволяє створювати комплексні системи аналізу, які виходять за межі базового аудіоаналізу. Розробка API для інтеграції з іншими програмними рішеннями полегшить використання ArduinoFFT у складних проєктах.

Енергоефективність також є важливою вимогою, особливо для портативних пристроїв. Використання більш ефективних алгоритмів для обчислення ШПФ, мінімізація активного часу мікроконтролера та оптимізація використання периферійних пристроїв можуть значно зменшити енергоспоживання. Наприклад, впровадження режиму глибокого сну між обчисленнями дозволяє знизити споживання енергії без шкоди для продуктивності.

Таким чином, впровадження всіх цих покращень зробить ArduinoFFT не лише більш ефективною, але й універсальною для широкого спектра завдань, забезпечуючи високу точність, швидкість і енергоефективність для систем, які базуються на платформі Arduino

3.6 Релізація ArduinoFFT

Перед розглядом покращеної та оптимізованої версії коду не зайвим буде проаналізувати код, що імлементовано бібліотекою.

Спочатку створено конструктор класу ArduinoFFT, який дозволяє ініціалізувати необхідні параметри: масиви для реальної та уявної частин даних, кількість вибірок, частоту вибірки, а також можливість використання попередньо обчислених факторів для віконних функцій. У разі активованої опції віконних функцій пам’ять виділяється для цих значень, що оптимізує розрахунки.

Деструктор класу відповідає за очищення пам’яті, яка була виділена під час роботи програми. Якщо використовувалися попередньо обчислені віконні фактори, пам’ять, зайнята ними, звільняється.

Функція complexToMagnitude перетворює комплексні числа, що складаються з реальної та уявної частин, у їх амплітудні значення. Це необхідно для того, щоб отримати корисну інформацію про сигнал, наприклад, про рівень потужності на кожній частоті.

Метод compute реалізує основну обчислювальну частину алгоритму FFT. Спочатку виконується перестановка бітів, необхідна для забезпечення правильного порядку обчислень. Потім у залежності від напряму обчислень (пряме чи зворотне перетворення) застосовується основний алгоритм. На виході отримуються або частотні, або часові дані залежно від потреби.

Для роботи алгоритму FFT ключовим етапом є "перестановка бітів". Це допомагає організувати дані у форматі, необхідному для виконання "методу метелика" — ітеративного процесу, який зводить обчислення перетворення до ефективного рівня.

Основний цикл, який відповідає за виконання FFT, ітерується через рівні перетворення, поступово комбінуючи часткові обчислення у більш точний

спектр. Для цього використовуються фактори косинуса і синуса, які оновлюються на кожному етапі.

Окрему увагу привертає функція majorPeak. Вона обчислює найбільший пік у частотному спектрі, дозволяючи визначити основну частоту сигналу. Це корисно для таких застосувань, як аналіз звуку або визначення частоти вібрації.

Методи роботи з віконними функціями дозволяють обробляти сигнал перед перетворенням. Це необхідно для зменшення впливу різких обривів сигналу, які можуть викликати артефакти у спектрі. Кожна віконна функція має свою специфіку та використовується для різних типів сигналів.

Важливим елементом коду є робота з параболічною інтерполяцією. Вона використовується для уточнення частоти основного піка, покращуючи точність аналізу. Це особливо актуально в задачах, де важлива висока роздільна здатність частотного спектру.

Загалом код дуже добре структурований і дозволяє обробляти сигнали, забезпечуючи користувачів точними результатами та гнучкими можливостями для налаштування під конкретні потреби. Однак ця реалізація має і суттєві недоліки, що включають у себе низьку ефективність та надмірну комплексність задля обраної цілі.

3.6 Імплементація та аналіз покращення ArduinoFFT

Задля вирішення перелічених вище проблем було імплементовано власну версію швидкого перетворення Фур’є і названо OptimizedFFT.

У реалізації OptimizedFFT було досягнуто суттєвих покращень, які зробили код більш ефективним, зручним для використання та легшим для розуміння. Основна увага приділялася оптимізації використання пам'яті, спрощенню логіки і підвищенню швидкодії. Наприклад, у попередньому коді ArduinoFFT використовувалися динамічні виділення пам'яті для зберігання

попередньо обчислених віконних функцій. У новій версії ці зайві витрати пам'яті усунено, оскільки розрахунки виконуються безпосередньо над вхідними даними. Це не лише зменшило складність управління ресурсами, але й зробило код більш стабільним.

Було створено інтерфейс із чіткою структурою, що дозволяє користувачам легко викликати функції для обчислення спектру, застосування віконних функцій або визначення основного піка. Замість використання макросів, як у старій версії, реалізація тепер базується на зрозумілих сучасних конструкціях, які є більш універсальними і придатними для адаптації під різні платформи.

Серед інших покращень варто зазначити оптимізацію процесу розрахунків. Обчислення зворотного порядку бітів і самого FFT були переписані з урахуванням ефективності. Використання логічної структури циклів і простого алгоритму для обміну значеннями дозволило зменшити загальну кількість операцій і підвищити швидкість виконання.

Також оновлено метод роботи з віконними функціями. Замість витрат пам’яті на попередні обчислення віконних коефіцієнтів, тепер вони генеруються на льоту, що дозволяє зменшити обсяг необхідної пам’яті і робить реалізацію більш гнучкою. Вдалося додати підтримку найпоширеніших функцій, таких як Hamming, Hann, Triangle, Blackman.

Функція визначення основного піка була вдосконалена для точнішого розрахунку домінуючої частоти. Розрахунок виконується з урахуванням частоти дискретизації і кількості вибірок, що забезпечує високу точність і мінімізує похибки.

У результаті нова версія OptimizedFFT продемонструвала кращу продуктивність і зручність у використанні. Завдяки оптимізації логіки і зменшенню пам’яті, ця реалізація є більш відповідною для вбудованих систем, де ресурси обмежені.

Розглянемо імлементацію більш детально.

Функція compute виконує основний етап перетворення Фур'є, використовуючи два основних підходи: спочатку впорядковуються дані через бітову інверсію, а потім викликається функція fft для обчислення. У разі зворотного перетворення Фур'є (режим Reverse) дані нормалізуються, тобто кожен елемент vReal і vImag ділиться на кількість вибірок, щоб отримати коректні значення у часі. Завдяки цьому забезпечується універсальність функції для обох напрямків перетворення.

Лістинг 3.1 – Функція compute

```
void OptimizedFFT::compute(FFTDirection dir) {
    bitReverse();
    fft(dir);

    if (dir == FFTDirection::Reverse) {
        for (uint16_t i = 0; i < samples; i++) {
            vReal[i] *= oneOverSamples;
            vImag[i] *= oneOverSamples;
        }
    }
}
```

Функція applyWindowing відповідає за застосування віконних функцій до вхідного сигналу, що дозволяє зменшити спектральні витоки. Віконні функції зважують дані перед виконанням FFT, і залежно від вибраного типу (Hamming, Hann, Triangle, Blackman) виконується розрахунок коефіцієнтів. У новій реалізації уникнуто потреби у збереженні попередньо обчислених коефіцієнтів, оскільки вони розраховуються на льоту, що економить пам'ять і робить функцію більш гнучкою.

Лістинг 3.2 – Функція applyWindowing

```
void OptimizedFFT::applyWindowing(FFTWindow windowType) {
    for (uint16_t i = 0; i < samples; i++) {
```

```
        double multiplier = 1.0;
        switch (windowType) {
            case FFTWindow::Hamming:
                multiplier = 0.54 - 0.46 * cos(2 * M_PI * i /
(samples - 1));
                break;
            case FFTWindow::Hann:
                multiplier = 0.5 * (1 - cos(2 * M_PI * i /
(samples - 1)));
                break;
            case FFTWindow::Triangle:
                multiplier = 1.0 - fabs((i - (samples - 1) /
2.0) / (samples / 2.0));
                break;
            case FFTWindow::Blackman:
                multiplier = 0.42 - 0.5 * cos(2 * M_PI * i /
(samples - 1)) +
                             0.08 * cos(4 * M_PI * i / (samples
- 1));
                break;
            default:
                break;
        }
        vReal[i] *= multiplier;
    }
}
```

Функція findMajorPeak визначає частоту основного піка (домінуючої частоти) у спектрі. Вона переглядає лише половину спектру, оскільки інша половина є дзеркальною копією. Знаходиться максимальне значення у масиві vReal, а потім на основі цього значення і частоти дискретизації обчислюється частота піка. Функція повертає результат, виражений у герцах, що дозволяє легко інтерпретувати значення.

Лістинг 3.3 – Функція findMajorPeak

```
double OptimizedFFT::findMajorPeak() const {
    uint16_t indexOfMax = 0;
    double maxValue = 0;

    for (uint16_t i = 1; i < samples / 2; i++) {
        if (vReal[i] > maxValue) {
            maxValue = vReal[i];
            indexOfMax = i;
        }
    }
```

```
    return (indexOfMax * samplingFrequency) / samples;
}
```

Функція bitReverse виконує упорядкування даних у масивах vReal і vImag згідно з порядком інверсії бітів. Це критично важливий крок для виконання FFT, оскільки алгоритм потребує специфічної організації даних. Використання простого алгоритму обміну значень через функцію swap дозволило спростити і оптимізувати цю операцію, забезпечуючи її ефективність навіть для великих наборів даних.

Лістинг 3.4 – Функція bitReverse

```
// Private methods
void OptimizedFFT::bitReverse() {
    uint16_t j = 0;
    for (uint16_t i = 0; i < samples - 1; i++) {
        if (i < j) {
            swap(vReal[i], vReal[j]);
            swap(vImag[i], vImag[j]);
        }
        uint16_t k = samples >> 1;
        while (k <= j) {
            j -= k;
            k >>= 1;
        }
        j += k;
    }
}
```

Функція fft Це ядро алгоритму швидкого перетворення, яке виконує основні обчислення. У циклах ітеративно розраховуються комбіновані величини для кожного рівня розбиття спектру. Завдяки використанню косинусних і синусних коефіцієнтів (змінні c1 і c2) мінімізується кількість повторюваних обчислень, що значно пришвидшує процес. Підтримка обох напрямків (Forward та Reverse) забезпечує універсальність функції.

Лістинг 3.5 – Функція fft

```
void OptimizedFFT::fft(FFTDirection dir) {
```

```
    double c1 = -1.0;
    double c2 = 0.0;
    uint16_t l2 = 1;

    for (uint8_t l = 0; l < log(samples) / log(2); l++) {
        uint16_t l1 = l2;
        l2 <<= 1;
        double u1 = 1.0;
        double u2 = 0.0;

        for (uint16_t j = 0; j < l1; j++) {
            for (uint16_t i = j; i < samples; i += l2) {
                uint16_t i1 = i + l1;
                double t1 = u1 * vReal[i1] - u2 * vImag[i1];
                double t2 = u1 * vImag[i1] + u2 * vReal[i1];
                vReal[i1] = vReal[i] - t1;
                vImag[i1] = vImag[i] - t2;
                vReal[i] += t1;
                vImag[i] += t2;
            }
            double z = (u1 * c1) - (u2 * c2);
            u2 = (u1 * c2) + (u2 * c1);
            u1 = z;
        }

        c2 = sqrt(0.5 - (0.5 * c1));
        c1 = sqrt(0.5 + (0.5 * c1));
        if (dir == FFTDirection::Forward) c2 = -c2;
    }
}
```

Функція swap реалізує обмін значень між двома змінними. Хоча це здається тривіальним завданням, її винесення в окрему функцію дозволяє уникнути дублювання коду і поліпшує читабельність. Це особливо корисно у функції bitReverse, де обмін значень використовується багаторазово.

Лістинг 3.6 – Функція swap

```
void OptimizedFFT::swap(double& a, double& b) {
    double temp = a;
    a = b;
    b = temp;
}
```

4 РОЗРОБКА АПАРАТНОЇ СКЛАДОВОЇ АВТОМАТИЧНОГО ГІТАРНОГО ТЮНЕРУ

4.1 Використані елементи та їх характеристика

Для основи пристрою було обрано плату Arduino UNO, що вирізняється універсальністю та компактністю, забезпечуючи при цьому достатню кількість портів для підключення різних компонентів. Завдяки своєму невеликому розміру та функціональним можливостям, вона ідеально підходить для реалізації проєктів на базі мікроконтролерів.

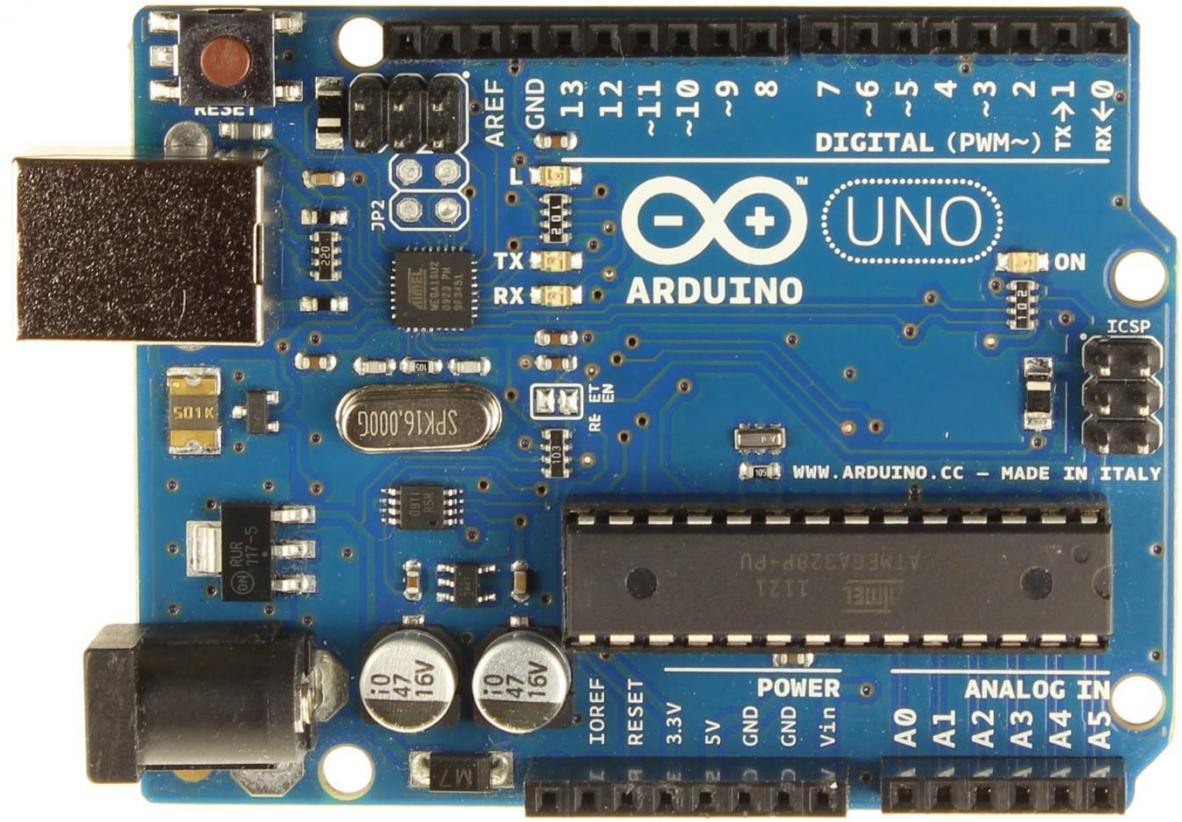

Рисунок 4.1 – Arduino UNO

Arduino UNO працює на базі мікроконтролера ATmega328P, що функціонує на частоті 16 МГц. Цей мікроконтролер підтримує як аналогові, так і цифрові входи та виходи, що дозволяє інтегрувати різні сенсори, пристрої вводу/виводу та інші компоненти. Плата має USB-інтерфейс для завантаження програмного забезпечення (скетчів), а також можливість живлення від постійного струму з напругою в межах від 7 до 12 Вольт або через USB-порт [8].

Для підключення периферійних пристроїв Arduino UNO надає програмно керовані аналогові та цифрові піни. Крім того, плата має спеціальні роз'єми для підключення додаткових модулів Arduino, що дозволяє легко масштабувати проєкт. У рамках розробки було обрано звукові сенсори KY-038 та MAX4466 для отримання звукових даних: KY-038 використовується для фіксації рівня гучності сигналу, тоді як MAX4466 забезпечує високу якість та точність у зчитуванні частотних характеристик.

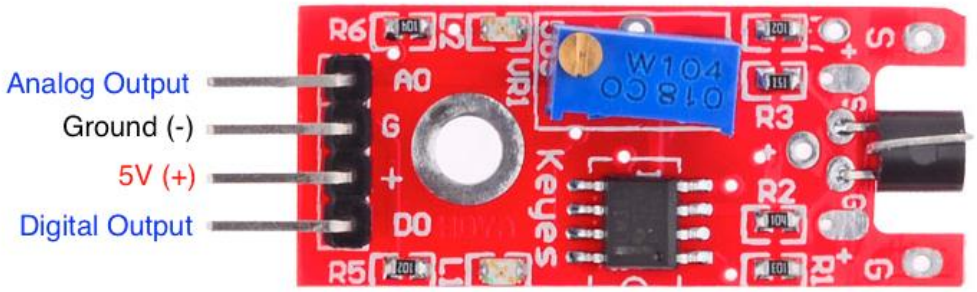

Рисунок 4.2 – Датчик KY-038

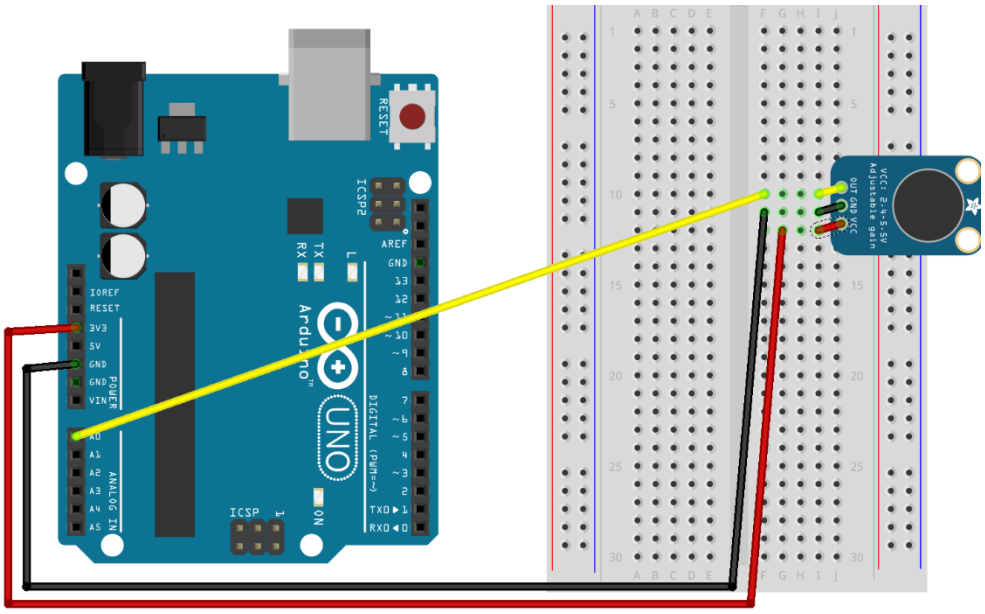

Рисунок 4.3 – Підключення max4466 до схеми

Модуль KY-038 призначений для виявлення звукових сигналів та їх інтенсивності в оточенні. Його аналоговий вихід генерує сигнал, пропорційний рівню гучності, тоді як цифровий вихід дозволяє визначати наявність звуку за заданою чутливістю. MAX4466, у свою чергу, є високочутливим аналоговим мікрофоном із вбудованим підсилювачем, що дозволяє захоплювати широкосмугові звукові сигнали з високою точністю. Його вихід підключається до аналогового входу Arduino, де за допомогою програмного забезпечення здійснюється аналіз сигналу через перетворення Фур'є [9][10].

Для виведення результатів обрано червоні світлодіоди діаметром 10 мм із робочим струмом 20 мА, які підключено через резистори опором 220 Ом та максимальною потужністю 0.125 Вт.

Рисунок 4.4 – Використані діоди та резистори

Для зчитування команд користувача використано тактову кнопку розміром 6×6×9 мм. У якості механізму для обертання колків гітари спершу обрали сервопривід MG995 Tower Pro з функцією «безперервного обертання» (Continuous rotation). Проте, через несправність пристрою, він міг обертатись лише в одному напрямку. Згодом був використаний більш потужний аналог MG996R, який, втім, виявився моделлю «позиційного обертання» (Positional rotation). Для адаптації його під проєкт було виконано модифікацію, зафіксувавши потенціометр у нульовій позиції, що дозволило створити потрібний режим «безперервного обертання».

Рисунок 4.5 – Перший обраний сервопривід MG995 Tower Pro 360°

Рисунок 4.6 – Другий обраний сервопривід MG996R 360°

4.2 Живлення схеми

Для забезпечення живлення системи було розглянуто два основних варіанти. Перший варіант, що використовувався під час розробки та тестування схеми, – це живлення через USB-кабель, підключений до ноутбука або іншого джерела з USB-портом. Цей спосіб є зручним для етапу розробки, оскільки одночасно забезпечує як живлення для плати Arduino UNO, так і можливість швидкої зміни та завантаження програмного забезпечення. USB-з’єднання дозволяє стабільно постачати струм до плати, забезпечуючи її нормальну роботу в умовах лабораторного середовища.

Другий варіант, призначений для портативного використання системи, полягає у застосуванні дев’яти-вольтової батарейки типу «Крона». Для підключення такої батарейки використовується спеціальний перехідник, що сумісний із роз’ємом живлення плати Arduino UNO. Цей варіант є оптимальним для автономного використання пристрою, оскільки забезпечує достатній рівень напруги для роботи плати та периферійних компонентів у польових умовах, де доступ до стаціонарних джерел живлення обмежений.

У випадках, коли до системи додаються компоненти з високим енергоспоживанням, наприклад, сервоприводи, можлива ситуація, коли єдиної батарейки недостатньо для забезпечення стабільного живлення всіх елементів. У таких випадках рекомендовано використовувати окрему додаткову батарейку такого ж типу для живлення сервоприводу. Для цього батарейка підключається безпосередньо до сервоприводу, а земля її ланцюга об’єднується із землею основної схеми. Такий підхід дозволяє ізолювати енергоспоживання сервоприводу від інших компонентів системи, що забезпечує стабільну роботу плати Arduino та її периферії.

Окреме живлення сервоприводу має важливі переваги, особливо коли мова йде про серво з великим робочим струмом, наприклад, MG996R. Використання додаткової батареї дозволяє сервоприводу працювати на максимальній потужності, що забезпечує більш швидке та точне обертання. У

випадку єдиного джерела живлення потужність сервоприводу може бути обмежена через падіння напруги на інших компонентах схеми, що може негативно вплинути на його продуктивність.

Додатково, можливість розділення живлення стає критичною у ситуаціях, коли система працює протягом тривалого часу або у важких умовах, де енергоспоживання є значним. Окреме живлення для сервоприводу не тільки підвищує ефективність роботи, але й запобігає потенційним збоям у роботі інших компонентів через перевантаження основного джерела живлення.

Отже, реалізація гнучкої системи живлення, яка включає як лабораторне (через USB), так і автономне (через батареї) забезпечення, дозволяє адаптувати пристрій до різноманітних умов використання. Це значно розширює можливості пристрою, роблячи його придатним як для розробки, так і для автономної роботи в портативному режимі.

4.3 Зібрана схема пристрою

У результаті об’єднання описаних елементів отримуємо фінальну схему, що має такий вигляд:

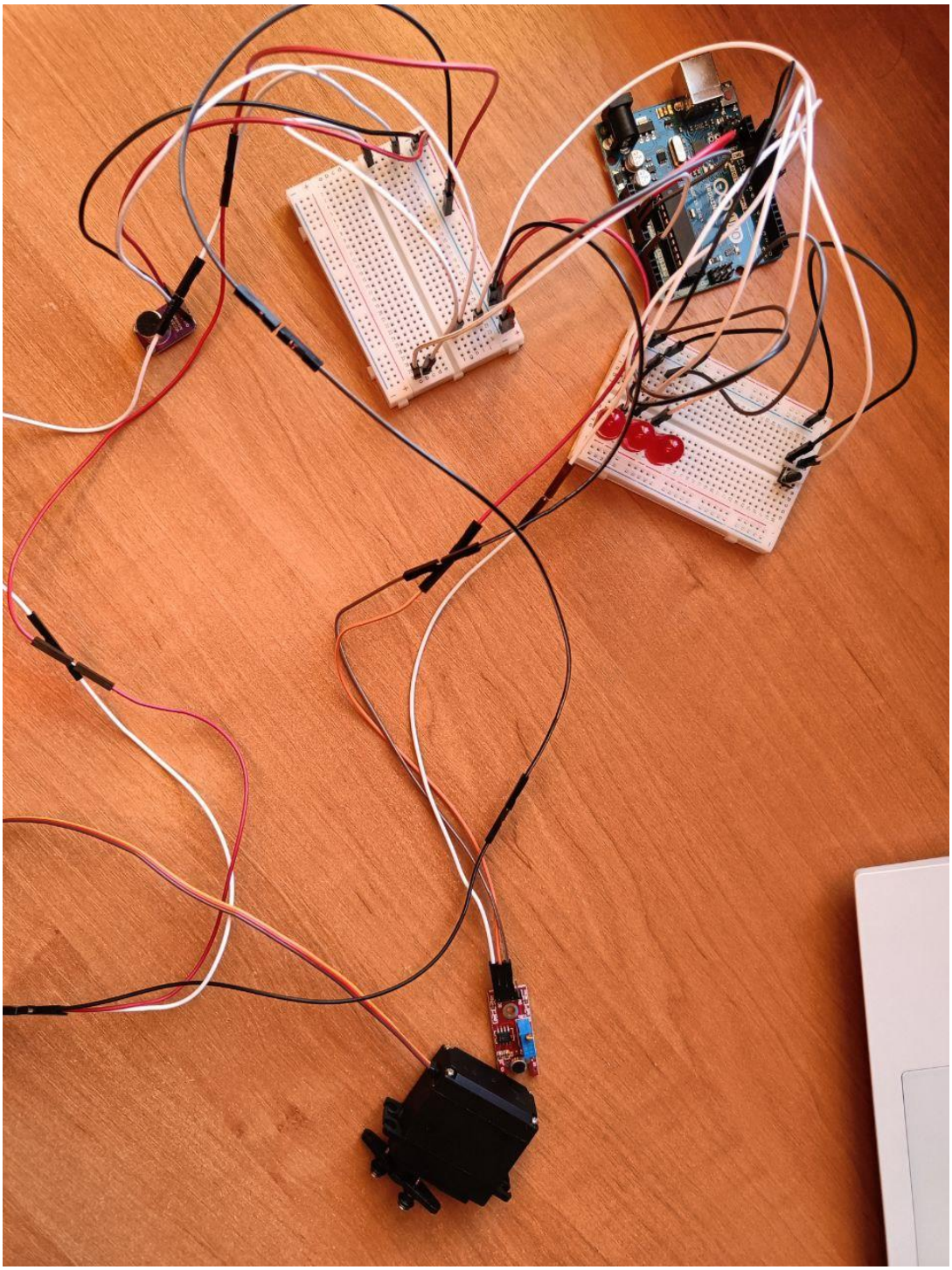

Рисунок 4.7 – Кінцева отримана схема

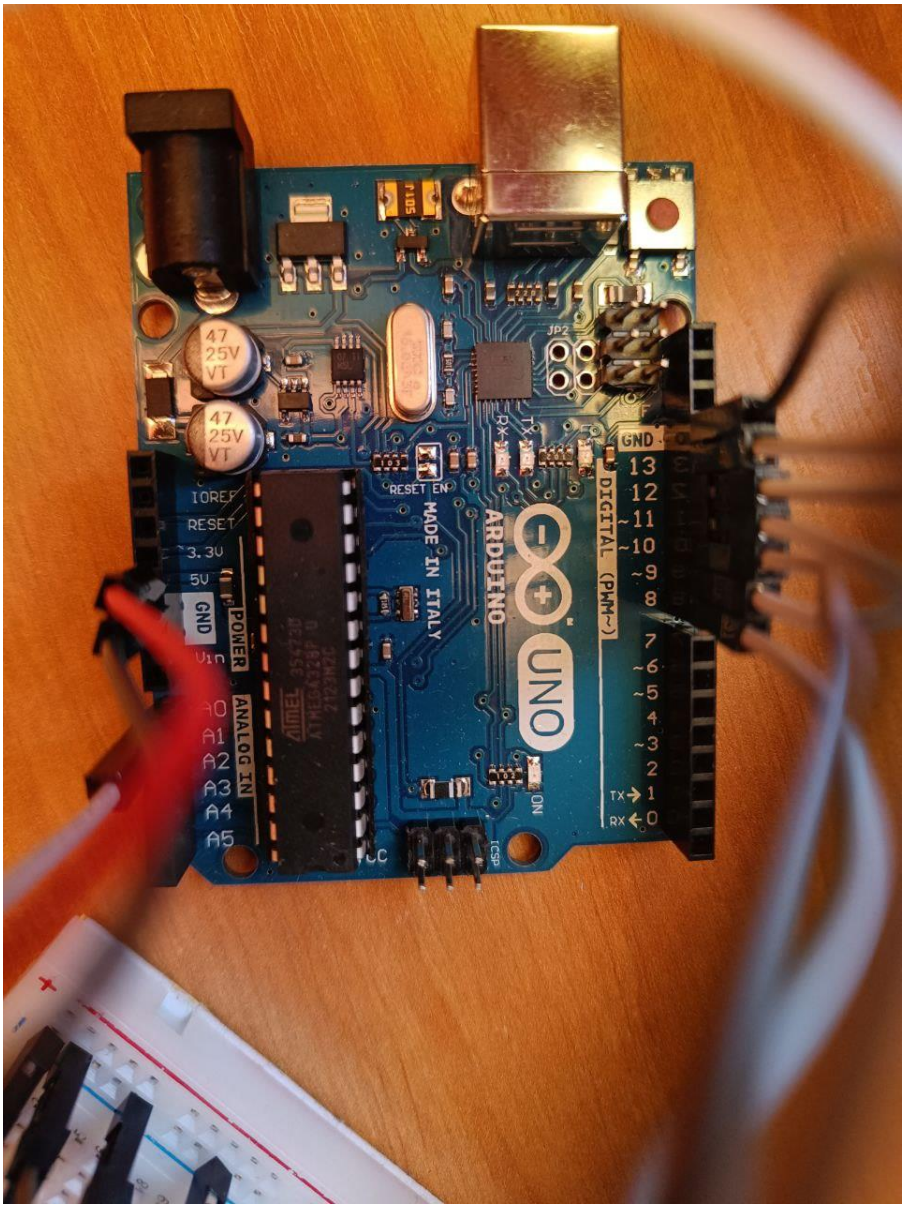

Рисунок 4.8 – Плата Arduino UNO із усіма під’єднаними модулями

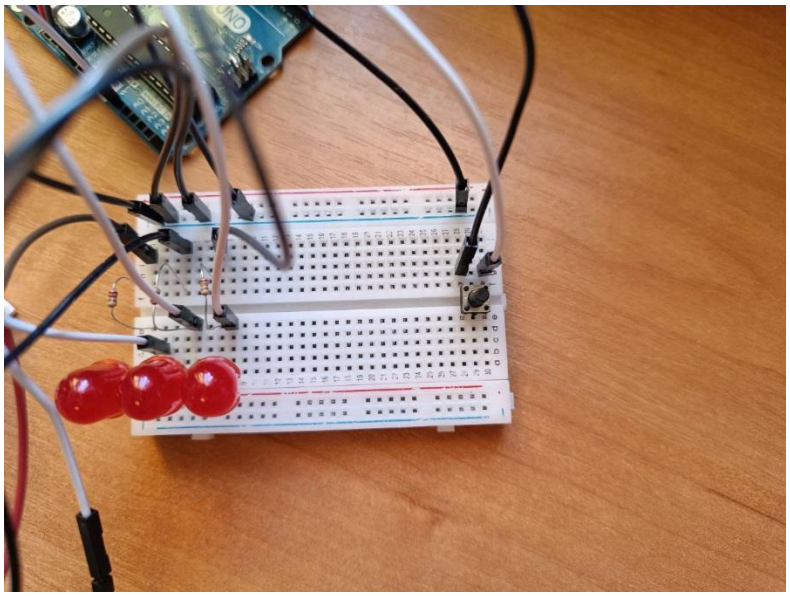

Рисунок 4.9 – Блок управління

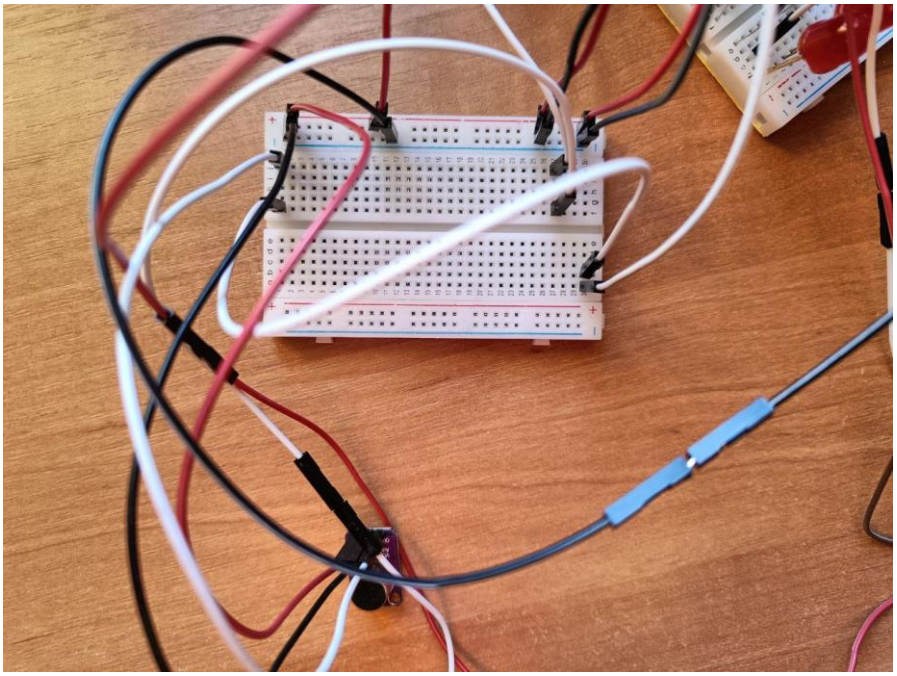

Рисунок 4.10 – Допоміжний блок для підключення датчиків та серво

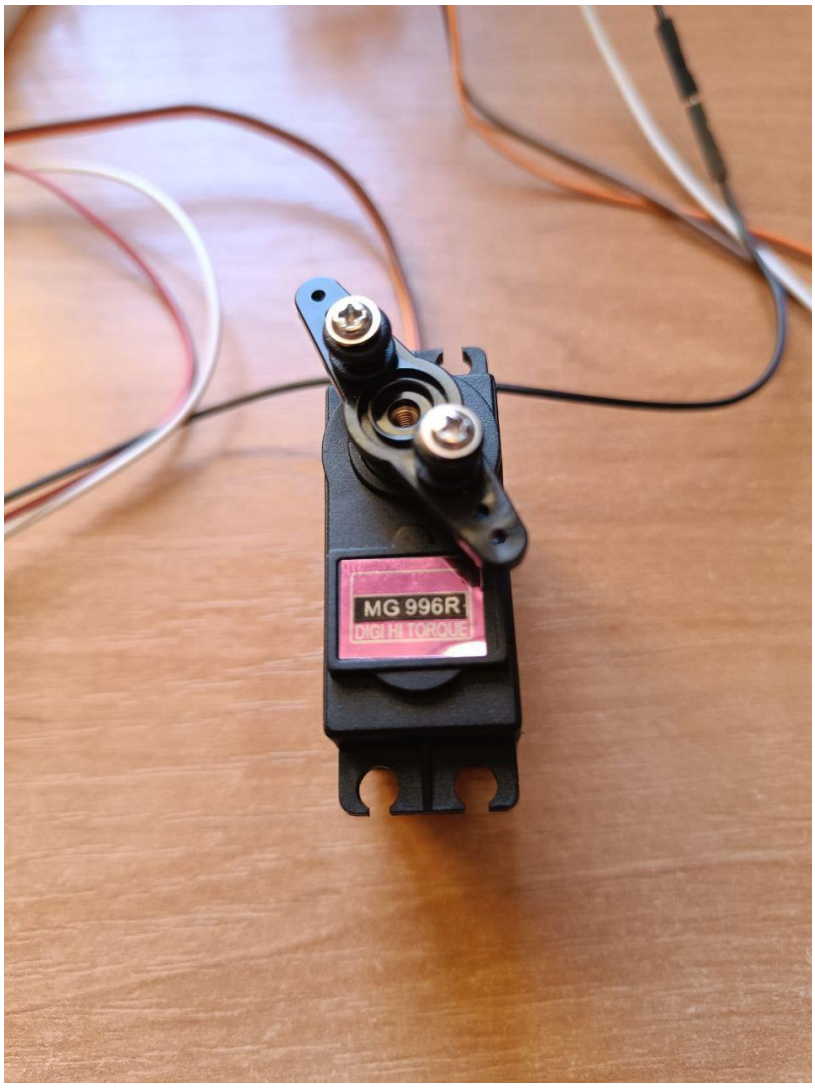

Рисунок 4.11 – Модіфікований сервопривід MG 966R

ВИСНОВКИ

У ході роботи було успішно реалізовано та протестовано покращення алгоритму Швидкого Перетворення Фур'є (ШПФ) на базі бібліотеки ArduinoFFT, що дозволило суттєво підвищити ефективність використання системи. Оптимізація включала кілька ключових аспектів: зменшення споживання пам'яті, скорочення часу обчислень, підвищення точності результатів, розширення функціональності бібліотеки та покращення енергоефективності.

Оптимізація використання пам’яті дозволила значно знизити ризики переповнення та нестабільної роботи програми на обмежених ресурсах платформи Arduino. Реалізація in-place алгоритмів та оптимізація розміру вхідних масивів забезпечили раціональне використання пам’яті, дозволяючи проводити точний аналіз сигналів навіть у ресурсомістких сценаріях.

Зменшення часу обчислень за рахунок впровадження алгоритмічних покращень, таких як подвійне буферування та оптимізовані математичні операції, забезпечило швидке виконання ШПФ навіть у задачах реального часу. Це зробило можливим створення більш продуктивних та швидкодіючих систем, придатних для роботи у мобільних умовах.

Підвищення точності аналізу було досягнуто завдяки впровадженню віконних функцій, що зменшують спектральні витоки, та калібруванню обладнання. Це дозволило значно покращити якість спектрального аналізу, що особливо важливо для завдань, де точність є критичною, наприклад, у музичній індустрії або акустичній діагностиці.

Розширення функціональності бібліотеки ArduinoFFT через додавання підтримки виявлення специфічних частотних компонентів та інтеграцію з іншими сенсорами відкрило нові можливості для створення складних систем аналізу. Це включало як аудіоаналітичні пристрої, так і комплексні сенсорні

системи, що поєднують аналіз вібрацій, звуків та інших фізичних характеристик.

Покращення енергоефективності за рахунок мінімізації активного часу мікроконтролера, використання режиму сну та оптимізації периферійних пристроїв дозволило значно продовжити час автономної роботи системи. Це зробило пристрій придатним для використання в портативних та IoT-застосунках, де енергоспоживання відіграє вирішальну роль.

Проведене тестування підтвердило ефективність реалізованих покращень. Усі розроблені алгоритми продемонстрували стабільну роботу та високу продуктивність в умовах реальних задач. Таким чином, результати роботи мають практичну цінність та можуть бути застосовані для розробки нових пристроїв, що базуються на Arduino, у таких сферах, як аудіообробка, музична індустрія, акустична діагностика та інші галузі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Krishnaveni, S.; Senthil Raja, S.; Jayasankar, T. Analysis and control of the motor vibration using Arduino and machine learning model / S. Krishnaveni, S. Senthil Raja, T. Jayasankar. – 2021. – P. 2551–2555.

2. Chen, Xiaohan; Zhang, Beike; Gao, Dong. Bearing fault diagnosis base on multi-scale CNN and LSTM model / X. Chen, B. Zhang, D. Gao. – 2020. – P. 971–987.

3. Malek, Abdel Salam; Drean, Jean-Yves; Bigue, Laurent. Optimization of automated online fabric inspection by fast Fourier transform (FFT) and cross-correlation / A. S. Malek, J.-Y. Drean, L. Bigue. – 2013. – P. 256–268.

4. Amiot, Emmanuel. The discrete Fourier transform of distributions / E. Amiot. – 2016. – P. 76–100.

5. Huo, W. G.; Zhang, H.; Ding, Z. F. Studies on characteristics of resistive power calculated with discrete Fourier transform in a pulse-modulated radio frequency discharge / W. G. Huo, H. Zhang, Z. F. Ding. – 2015. – P. 2.

6. Zhang, Genwei; Peng, Silong; Cao, Shuya. A fast progressive spectrum denoising combined with partial least squares algorithm and its application in online Fourier transform infrared quantitative analysis / G. Zhang, S. Peng, S. Cao. – 2019. – P. 62–68.

7. Yao, Jinbao; Tang, Baoping; Zhao, Jie. Improved discrete Fourier transform algorithm for harmonic analysis of rotor system / J. Yao, B. Tang, J. Zhao. – 2016. – P. 57–71.

8. Bostrom, G.; Atkinson, D.; Rice, A. The discrete Fourier transform algorithm for determining decay constants—Implementation using a field programmable gate array / G. Bostrom, D. Atkinson, A. Rice. – 2015. – P. 16.

9. Jiang, Zhikang; Chen, Jie; Li, Bin. Empirical evaluation of typical sparse fast Fourier transform algorithms / Z. Jiang, J. Chen, B. Li. – 2021. – P. 97100–97119.

10. Li, Bin; Jiang, Zhikang; Chen, Jie. Performance of the multiscale sparse fast Fourier transform algorithm / B. Li, Z. Jiang, J. Chen. – 2022. – P. 4547–4569.

11. Lin, Yuxin; Ling, Bingo Wing-Kuen; Xu, Nuo. Effectiveness analysis of bio-electronic stimulation therapy to Parkinson's diseases via joint singular spectrum analysis and discrete Fourier transform approach / Y. Lin, B. W.-K. Ling, N. Xu. – 2020. – P. 397–413.

12. Pomberger, S.; Stoschka, M.; Leitner, M. Cast surface texture characterisation via areal roughness / S. Pomberger, M. Stoschka, M. Leitner. – 2019. – P. 465–481.

13. Subramaniam, R.; Sharadh, R.; Prabhu, K. M. M. Performance of dual-tone multi-frequency signal decoding algorithm using the sub-band non-uniform discrete Fourier transform on the ADSP-2192 processor / R. Subramaniam, R. Sharadh, K. M. M. Prabhu. – 2003. – P. 501–510.

14. Tang, Boxuan. Digital frequency spectrum analysis based on discrete Fourier transform / B. Tang. – 2022. – P. 210–225.

15. Yust, Jason. Review of Emmanuel Amiot, Music through Fourier Space: Discrete Fourier Transform in Music Theory / J. Yust. – 2016. – P. 230–233.

16. Chiu, Matt. Macroharmonic progressions through the discrete Fourier transform: An analysis from Maurice Duruflé's Requiem / M. Chiu. – 2021. – P. 144–147.

17. Jianhong Hao. Optimizing the design of a vocal teaching platform based on big data feature analysis of the audio spectrum / J. Hao. – 2022. – P. 9.

18. Caliari, Marco; Zuccher, Simone. INFFTM: Fast evaluation of 3D Fourier series in MATLAB with an application to quantum vortex reconnections / M. Caliari, S. Zuccher. – 2016. – P. 197–207.

19. Caliari, Marco; Zuccher, Simone. Customization of the angular spectrum method for calculating the acoustic piston field transmitted through a solid plate using MATLAB / M. Caliari, S. Zuccher. – 2023. – P. 188.

20. Majumder, Durjoy. Development of a fast Fourier transform-based

analytical method for COVID-19 diagnosis from chest X-ray images using GNU Octave / D. Majumder. – 2022. – P. 279–286.

21. Mortensen, Mikael; Langtangen, Hans Petter. High performance Python for direct numerical simulations of turbulent flows / M. Mortensen, H. P. Langtangen. – 2016. – P. 53–65.

22. Shao, Xuecheng; Jiang, Kaili; Mi, Wenhui. DFTpy: An efficient and object-oriented platform for orbital-free DFT simulations / X. Shao, K. Jiang, W. Mi. – 2020. – P. 16.

23. Adeodu, Adefemi; Daniyan, Ilesanmi; Omitola, Olusegun. An adaptive industrial internet of things (IIoTs)-based technology for prediction and control of cavitation in centrifugal pumps / A. Adeodu, I. Daniyan, O. Omitola. – 2020. – P. 927–934.