Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Метод та апаратно-програмні засоби визначення напрямів звукових сигналів»

08-23.МКР.010.00.000 ПЗ

Виконав: студент 2 курсу, групи 1КІ-20м
напряму підготовки (спеціальності)
123 — «Комп'ютерна інженерія»
Рак М. І.

Керівник: к.т.н., доц.каф. ОТ
Тарновський М.Г.
(прізвище та ініціали)

Опонент: к.т.н., проф. каф. ЗІ
Лужецький В.А.
(прізвище та ініціали)

Допущено до захисту

Завідувач кафедри ОТ
_____д.т.н., проф. Азаров О. Д.
«___» _____2021 р

Вінниця 2021

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень — магістр

Спеціальність 123 — «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

обчислювальної техніки

д.т.н., проф. Азаров О. Д.

_____

«____» _____2021 року

З А В Д А Н Н Я

НА МАГІСТРСЬКУ КВАЛІФІКАЦІЙНУРОБОТУ СТУДЕНТУ

Раку Михайлу Івановичу

1. Тема проекту Метод та апаратно-програмні засоби визначення напрямів звукових сигналів, керівник проекту Тарновський Микола Геннадійович , кандидат технічних наук, доцент кафедри лазерної та оптоелектронної техніки, затверджені наказом Вінницького національного технічного університету від 06.03.2020 року № ___.

2. Строк подання студентом проекту _____

3. Вихідні дані до роботи: призначення — визначення локалізації джерела звукового сигналу за методами пасивної аудіолокації; джерело вхідних сигналів — мікрофонна решітка; частотний діапазон вхідних сигналів від 80 до 20000 Гц; режим роботи — автоматичний; підтримка взаємодії з комп′ютером.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) огляд і аналіз методів локації та параметрів аудіолокаційних систем; аналіз методів та алгоритмів обробки звукових сигналів, розробка апаратних засобів; розробка програмних засобів; економічний

розділ.

.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) загальна структурна схема, блок схеми роботи загального алгоритму, технічні характеристики пристроїв, результат виконання програми.

6. Консультанти розділів проекту приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3	Тарновський М.Г. к.т.н., доцент		
4	Кавецький В. В. к.е.н., доцент		

7 Дата видачі завдання_____

Таблиця 2 — Календарний план

№ етапу	Назва етапу	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд і аналіз джерел інформації	07.09-28.09.21	
2	Розробка технічного завдання	29.09.21	
3	Огляд та аналіз систем аудіолокації	12.10-21.10.21	
4	Розробка структури системи	22.10-25.10.21	
5	Практична реалізація, результати	26.10-28.10.21	
6	Оформлення пояснювальної записки	29.11-15.12.21	

Студент _____ Рак М. І.

Керівник роботи _____Тарновський М. Г.

АНОТАЦІЯ

УДК 681.3

Рак М.І. Метод та апаратно-програмні засоби визначення напрямів звукових сигналів. Магістерськ кваліфікаційна робота зі спеціальності 123 — комаютерна інженерія, освітня програма — компютерна інженерія. Вінниця: ВНТУ, 2021,116 с.

В магістерсікій дипломній роботі було досліджено та проаналізовано програмно–апаратні засоби для виявлення напрямку звуку. У теоретичній частині розглянута класифікація уже відомих систем аудіолокації. Розроблено метод та алгоритми, які дозволяють функціонувати системі вірно. Розроблено програмне забезпечення для обрахунку здвигів фаз при прийманні сигналу на мікрофону решітку. Вона була розроблена модулями. Один з них написаний мовою C++, а інший JavaScript при підключені потужних бібліотек для роботи зі звуком.

Ключові слова: аудіолокація, мікрофонні решітки, фазова модуляція, стереофонія.

ABSTRACT

UDC 681.3

Rak M. I. Method and hardware-software means of determining the directions of sound signals. Master's degree in specialty 123 - commutation engineering, educational program - computer engineering. Vinnytsia: VNTU, 2021,116 p.

In the master's thesis, software and hardware for detecting the direction of sound were researched and analyzed. In the theoretical part the classification of already known audiolocation systems is considered. The method and algorithms that allow the system to function properly have been developed. Software for calculation of phase shifts when receiving a signal on a microphone grating has been developed. It was developed by modules. One of them is written in C ++, and the other JavaScript when powerful libraries are connected to work with sound.

Key words: audiolocation, microphone arrays, phase modulation, stereophony.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АЦП — аналого - цифровий перетворювач

ПК — персональний комп’ютер

ОС — операційна система

ПЗ — програмне забезпечення

ЧЧ — частотна чутливість

ВСТУП

Використання різноманітних сигналів у різних середовищах є невід’ємною ознакою сучасного світу. За рахунок обробки сигналів вирішується широкий спектр задач у різноманітних сферах, включаючи військову, авіаційну та космічну.

Актуальністю теми є один із перспективних напрямів, що невід′ємно пов′язаний з опрацюванням сигналів — це аудіолокації. Аудіолокація заснована на використанні явищ відбивання та розсіювання звукових хвиль і представляє собою засіб, що дозволяє виявляти наявність різноманітних об′єктів, визначати їх взаємне розташування та стан. Найближчим конкурентом аудіолокації є оптична техніка. Основними перевагами засобів аудіолокації за порівнянням з тими, що використовують оптичне випромінювання, є можливість працювати в умовах темряви, більша дальність дії в складних погодних умовах, наприклад, дощу та туману, більша точність визначення напрямку на об'єкт, тощо.

Акустична локалізація об’єктів та отримання інформації про джерело звуку важливі для завдань автоматизованого контролю доступу та захисту територій, в комп’ютерних системах для ідентифікації джерел звуку, в мультимедійних системах для стадіонів, концертних та виставкових залів тощо. Одним з перспективних напрямів у сфері аудіолокації є системи пасивної аудіолокації, призначені для визначення напрямів надходження звукових сигналів на контрольованій місцевості [1,2]. У зв′язку з цим пошук шляхів покращення та вдосконалення основних функціональних характеристик апаратно-програмних засобів для виявлення і розпізнавання об’єктів методом пасивної аудіолокації є актуальним науково-технічним завданням, що має практичне застосування.

Сучасні рішення для обробки акустичних сигналів засновані на застосуванні комп’ютерних інформаційних систем, що забезпечують багатоканальне аналого-цифрове перетворення, з подальшою обробкою сигналів цифровими методами. Вхідні сигнали в подібних системах, як

правило, отримуються за допомогою звукових мікрофонів, що перетворюють акустичні коливанні в електричні сигнали звукової частоти. Найбільш перспективними серед таких систем є такі, що забезпечують синхронне багатоканальне аналого-цифрове перетворення, цифрове підсумування і кореляційний аналіз сигналів, отримуваних за допомогою мікрофонних решіток [3].

Основними проблемами, які потребують вирішення, є забезпечення кроссплатформеності, тобто можливості використовувати програмне забезпечення на різних програмно-апаратних платформах та коректність результатів роботи в умовах старіння елементів, зміни теплового режиму та шуму [4].

Метою магістерської роботи є спрощення апаратної реалізації та розширення можливостей застосування системи аудіолокації на основі синхронної багатоканальної обробки звукових сигналів.

Поставлена мета досягається вирішенням таких задач:

— аналіз сучасних підходів до побудови та організації систем аудіолокації;

— аналіз сучасних методів та алгоритмів багатоканальної обробки звукових сигналів;

— визначення підходів до реалізації апаратних та програмних засобів системи аудіолокації зі спрощеною організацією процесів синхронного багатоканального введення та обробки звукових сигналів.

Об’єктом дослідження є процес багатоканального введення та аналого-цифрового перетворення акустичних сигналів.

Предметом дослідження є методи та засоби пасивної аудіолокації для визначення локалізації джерел звукових сигналів.

Новизною одержаних результатів можна вважати:

— набула подальшого розвитку реалізація методу пасивної аудіолокації на основі визначення часових зсувів між акустичними сигналами, що надходять за кількома каналами, відповідно до якої за рахунок

використання зовнішньої звукової карти та модуля одноплатного комп′ютера спрощується апартно-програмна підтримка процесів введення та обробки звукових сигналів;

— напропоновані підходи до побудови апаратних та програмних засобів системи аудіолокації покращують показники її мобільності, швидкості обробки інформації та спрощують її розгортання та використання на місцевості.

Основні результати роботи отримані автором самостійно.

За результатами магістерської роботи опубліковані тези доповіді на конференції: “Молодь в науці: дослідження, проблеми, перспективи (МН-2022)”.

Обсяг та структура магістерської роботи містить вступ, 5 розділів, висновки, перелік джерел посилань і додатки. Загальний обсяг роботи – 134 сторінок, з яких основний зміст викладено на 88 сторінках друкованого тексту, містить 36 рисунків, 14 таблиць. Перелік джерел посилань містить 50 найменувань.

1 ОГЛЯД І АНАЛІЗ МЕТОДІВ ЛОКАЦІЇ ТА ПАРАМЕТРІВ АУДІОЛОКАЦІЙНИХ СИСТЕМ

1.1 Аналіз сфер застосування систем локації

Використання аудіолокатора в даний час відкриває дуже різноманітні можливості як для цивільних, так і для військових цілей. Тому протягом усього свого існування вони модифікувалися і використовувалися для нових цілей. Аудіолокація заснована на принципах радара.

Однією зі сфер застосування аудіолокації є локалізація джерела звуку — визначення місця розташування джерела звуку за даними вимірювання параметрів звукового поля. Звукове поле може бути описане основним фізичним параметром, таким як звуковий тиск. Вимірюючи цю властивість, можна визначити напрямок джерела звукового сигналу. Цю властивість можна розглянути на прикладі, людського вуха як воно приймає сигнал та опрацьовує його [5].

Локалізація може бути описана в термінах тривимірного положення. Азимут або горизонтальний кут, висота або вертикальний кут, і відстань (для статичних звуків) або швидкість (для переміщення звуків).

Азимут звуку сигнал про різницю в часі прибуття між вухами, відносної амплітудою високочастотних звуків (тіньовий ефектом), і асиметричними спектральними відображеннями від різних частин нашого тіла, в тому числі тулуба, плечей, і вушних раковин. Ці сигнали відстані є втратою амплітуди, втратою високих частот, а відношення прямого сигналу до сигналу, який відбивався [6].

Залежно від того, де знаходиться джерело, наша голова виступає в якості бар'єру для зміни тембру, інтенсивності та спектральні властивостей звуку, що допомагають орієнтуватися мозку,звідки лунає звук. Ці найдрібніші відмінності між двома вухами відомі як інтерауральні репліки.

Більш низькі частоти, з великими довжинами хвиль, розсіюють звук навколо голови змушуючи мозок зосередитися тільки на фазуванні сигналах

від джерела.

Для частот нижче 800 Гц, розміри голови (відстань вуха 21,5 см, що відповідає міжвушному часу затримки 625 мкс) менше, ніж половина довжини хвилі звукових хвиль. Таким чином, слухова система може визначати фазові затримки між обома вухами без плутанини. Відмінності інтерауральних рівня (рисунок 1.1) є дуже низькими в цьому діапазоні частот, особливо нижче приблизно 200 Гц, так що точна оцінка напрямки введення практично неможливо на основі тільки різниць рівнів.

При зниженні частоти нижче 80 Гц стає важко або неможливо використовувати або різницю в часі, або різницю рівнів для визначення бічного джерела звуку, оскільки різниця фаз між вухами стає замалою для спрямованої оцінки [6].

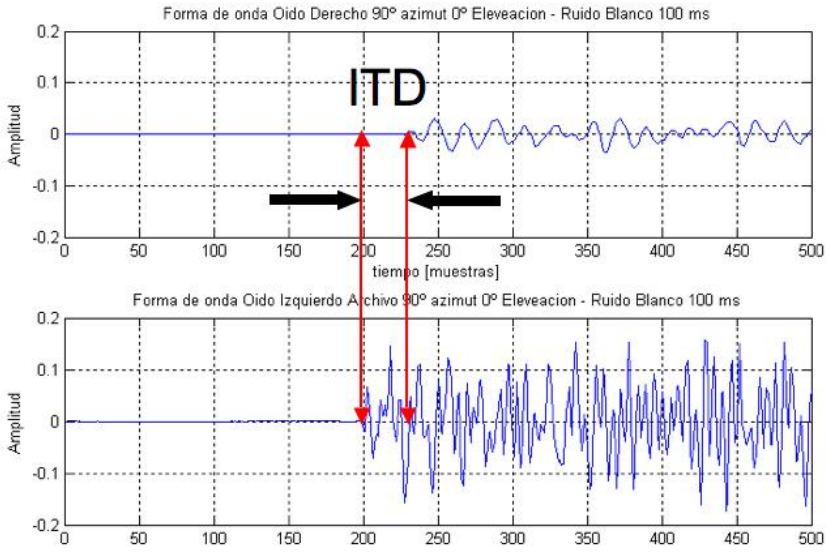

Рисунок 1.1 — Представлення інтерауральних рівнів

В технічних системах зазвичай звуковий тиск вимірюється за допомогою мікрофонів. Вони мають діаграму, яка описує їх чутливість залежно від дії. Багато різних мікрофонів мають всенаправлену діаграму направленості, що означає, що їх чутливі мікрофони існують з іншими полярними моделями, які є більш чутливими за визначенням, ніж задане визначення. Однак це все ще не вирішує проблему локалізації звуку.

Для досягнення точності та ідентичності вимірювальних каналів, крім

самокалібрування АЦП, використовують визначення та встановлення динамічних властивостей електричних вимірювальних каналів [7] та акустичних властивостей мікрофонів. Метод калібрування передбачає, що в каналах вимірювання оптимізовано лише окремі, некоректовані параметри (шум, швидкість). Деякі інші параметри (зміщення, посилення, форма амплітудно–фазової частотної характеристики, різниці між каналами) можна визначити та адаптувати цифровим шляхом. Розширення принципів калібрування та коригування до вимірювальних шляхів системи дає змогу покращити їх властивості порівняно з відомими технічними рішеннями.

Продуктивність класичних багатоканальних алгоритмів електронного сканування обмежена кількістю згенерованих здвигів та отриманням кореляційних сум, і зазвичай у режимі реального часу не виконуюють своїх функцій, і їхні методи не працюють без урахування конкретних сигналів, які відтворюються.

Коли звукові сигнали розпізнаються, вони реєструються в частотному діапазоні, як ефективні лінійні коефіцієнти, кептральні коефіцієнти та спектральні піки частоти [7, 8]. Гауссові зміни в основному використовуються для прийняття рішень. Також спостерігається значне зниження надійності виявлення сигналу, коли рівень шум / сигнал дуже великий.

Первинний аудіолокатор, в цілому, використовується для виявлення цілей, опромінюючи їх звуковою хвилею і потім приймаючи зворотній сигнал від цілі. Так як швидкість звукових хвиль в різних середовищах відома, можу бути можливим визначити відстань до цілі, опираючись на вимірюванні різних параметрів при поширенні сигналу.

У системах пасивного типу для локалізації звуку та ідентифікації об'єктів при прийомі акустичних сигналів застосовуються мікрофонні рішітки, що формують задану діаграму спрямованості акустичних антен. Мікрофона решітка — це вид мікрофонних засобів, що створюють систему у вигляді набору звукових приймачів, які одночасно працюють координативно (у фазі

або із затримкою фази) та мають своє розташування у закритому або відкритому приміщеннях [9].

Застосування мікрофонних решіток дає такі переваги [10]:

— спрямованість прийому звуку;

— пригнічення шумів точкових джерел;

— пригнічення нестаціонарних шумів оточення та досягнення більших відношення сигнал/шум;

— часткове ослаблення реверберації;

— можливість просторової локалізації звуку цільового джерела;

— можливість супроводу джерела, що рухається, і точкового джерела шуму.

Налаштування мікрофонних решіток на прийом сигналу за одним або декількох необхідних кутових напрямках здійснюється шляхом обробки сигналу в каналі кожного мікрофона та подальшого зваженого підсумовування цих сигналів. Сама решітка при цьому залишається нерухомою: вона не повертається до джерела сигналу, а налаштовується на необхідний кут прийому „електронним“ способом [11].

Відповідно до цього принцип роботи багатоканальних мікрофонних систем заснований на цифровій обробці сигналу підсумовування в каналі з введенням часових зсувів, необхідних для розпізнавання звуку. Для цього акустичні сигнали після перетворення у мікрофонах в низкочастотні електричні сигнали піддаються оцифровуванню та передаються для подальшої обробки на персональний комп'ютер. Оцифровка аналогових сигналів відбувається у результаті багатоканального аналого-цифрового перетворення.

1.2 Класифікація систем аудіолокації та характеристика основних технічних рішень

За призначенням системи локації можна класифікувати так [12] :

— виявлення;

— управління і стеження;
— панорамні;
— бічного огляду;
— метеорологічні ;
— цілевказівні;
— контрбатарейної сутички;
— огляду ситуації;
— поліцейський радар.

Віддалене захоплення голосової інформації (RMS) важливе не в останню чергу в області аудіорадарів, технології розпізнавання голосу, телекомунікацій та в багатьох інших технічних рішень.

Питання про використання засобів віддаленого збору голосової інформації широко обговорюються в системах безпеки і часто використовуються для спеціальних цілей. Послуги з різних країн. Значна частина цього заснована на прийомі мікрофонами акустичних звукових коливань з подальшим посиленням в десятки разів. Однак такі системи не в змозі значно збільшити відстань збору мовної інформації в міських умовах, оскільки разом із збільшенням мовного сигналу (МС) збільшується навколишній шум, що є однією з основних причин зниження розбірливості і якості мови.

У тихій місцевості можна отримати MS з відстані приблизно 100 м, але є й інші фактори (розповсюдження кількох променів, розсіювання високочастотних звукових компонентів, вітер тощо), які також обмежують відстань голосової інформації [13]. Ці обмеження значною мірою долаються в системах мікрофонних масивів (MR). MR добре відомий фахівцям у цій галузі. В даний час MR є важливою сферою в області цифрової обробки сигналів. Детальний опис принципів проектування та алгоритмів обробки сигналів МР представлено в серії базових робіт [14].

Сучасні успіхи в області застосування МР опираються на результати, які були досягнуті в галузі дистанційного збору аудіо інформації (ДСАІ)

протягом XX століття. Розглянемо коротко основні етапи розвитку засобів ДСАІ.

Перші системи дистанційного прийому звукової інформації були розроблені на початку 20 століття. Історію віддаленого збору аудіоінформації можна розбити на три фази: фаза зародження або «механічна», фаза аналогової обробки та фаза обробки цифрового сигналу.

На початковому етапі (під час Першої світової війни) дистанційний прийом звуку використовувався для виявлення літаків і визначення положення артилерійських батарей [15]. Ці методи та інструменти розвивалися до Другої світової війни, після чого вони поступилися місцем радарам. Другим етапом розвитку стали засоби обробки аналогових сигналів. Саме в цей момент почалося використання спрямованих підсилювачів мікрофонного сигналу, що дозволило використовувати інструменти DSAI для віддаленого отримання мовної інформації (DSMI). Розроблено ряд систем, детальний огляд яких наведено в [16], [17].

Основними типами ДСМІ на другому етапі були параболічні рефлекторні та лампові мікрофонні системи. Тоді ж були розроблені перші МР, але вони мали обмежені можливості та дуже складну реалізацію [18]. Крім того, на цьому етапі були розроблені основи обробки сигналів в антенних решітках [18]-[20]. Пізніше ці ідеї були розроблені в алгоритмах обробки сигналів МР.

Нарешті, третім етапом розвитку DSMI (з 1990-х років) стало використання цифрової обробки сигналів (ЦОС). З розробкою ЦОС МР використовувався для вирішення широкого спектру завдань, пов'язаних з обробкою мовних сигналів:

— виділення мови (підвищення розбірливості) цільового диктора в шумах;

— віддалений збір мовної інформації;

— дистанційне розпізнавання мовлення (наприклад, для голосового керування пристроями);

— дистанційна ідентифікація динаміків;

— розподіл мови мовця, вибір мови цільового мовця в мовному коктейлі;

— визначення положення та відстеження положення джерел звуку;

— аналіз випромінювання звуку від джерел звуку;

— системи автоматичного розпізнавання мови та голосового керування в автомобілі;

— інформаційні кіоски з мовною службою в громадських місцях;

— слухові апарати

Відзначимо основні особливості МР:

— вони працюють з цифровим звуком (семплуються за амплітудою, часом і простором);

— відтворюють у цифровому вигляді основні методи аналогової обробки ПК (фокусування звуку тощо);

— реалізація великої кількості перетворень сигналів, які недоступні для аналогових систем;

Потужним додатковим імпульсом для розширення сфери застосування МР є розробка та серійне виробництво цифрових MEMS-мікрофонів [21].

Усі технології вилучення мовної інформації в МР нерозривно пов'язані з алгоритмами DSP. Використання DSP значно розширює функціональні можливості засобів віддаленого збору звукової інформації. У майбутньому MR буде означати як систему мікрофонів, так і алгоритми обробки їх сигналів.

Використовуються три види модуляції - амплітудна, частотна і фазова. При амплітудній модуляції амплітуда несучої частоти змінюється з часом разом з амплітудою інформаційного сигналу. Частотна модуляція викликає відхилення в часі (девіацію) несучої частоти з амплітудою корисного сигналу. У разі фазової модуляції це робиться відповідно до фази несучої частоти. Приклад наведено на рисунку 1.2 [22].

Рисунок 1.2 — Приклади різної модуляції та відповідних корисних сигналів

1.3 Аудіолокація

Дистанційний збір даних та аудіолокалізація об’єктів на землі важлива в задачах автоматизованого контролю доступу та охорони територій, у комп’ютерних системах ідентифікації джерел звуку, у мультимедійних системах для стадіонів, концертних залів, виставкових залів тощо.

Як і в усіх системах локації, ми можемо розрізняти пасивне та активне місце розташування аудіо:

— в активному місці генерується тон для створення відлуння, яке потім аналізується для визначення положення об'єкта;

— пасивне розташування включає виявлення шумів або вібрацій створені об'єктом, які виявляють й потім аналізуються для визначення місця розташування даного об'єкта.

У середині 20 століття, з розвитком електроніки, вперше були розроблені мікрофонні масиви для збору інформації, так що цей період можна охарактеризувати як етап обробки аналогового сигналу.

Інструменти цифрової обробки широко використовуються з 1990-х років. Їх використовували для виконання різноманітних завдань: визначення джерела звуку, його аналіз; в слухових апаратах; для дистанційного

розпізнавання гучномовців; Виділення певного тону з шуму. Цей час називають етапом цифрової обробки сигналу.

Мікрофонна решітка в основному використовується для дослідження явища звукового сигналу, оскільки це один із типів спрямованих мікрофонів, реалізованих для прийому звуку і які працюють узгоджено (по фазі або з фазовими затримками) [23].

Функціональний принцип багатоканальних мікрофонних решіток заснований на цифровому підсумовуванні сигналів у каналах з введенням часових зрушень, необхідних для формування шаблону. Акустичний сигнал, що проникає в мікрофонну решітку (МР), перетворюється в цифрову форму за допомогою багатоканальних аналого-цифрових перетворювачів (АЦП) і передається на персональний комп'ютер (ПК), який здійснює необхідну обробку та вилучення корисний сигнал.

Мікрофонний масив складається з кількох аудіоприймачів, які працюють разом. Просторова селективність прийому сигналу може бути визначена шляхом зміни фазової затримки сигналу, який по-різному доходить до різних мікрофонів [23]. Вихід мікрофонного масиву є суматором. Амплітуди і фази сигналів, що надходять від аудіоприймачів до суматора, регулюються ваговими коефіцієнтами (ваговими функціями). Ця опція реалізована в програмному забезпеченні, тому ви можете отримати будь-яку діаграму без зміни дизайну або переміщення сітки.

Розберемо принципи роботи мікрофонного масиву. Якщо мікрофони не спрямовані, вони вловлюють звук з усіх боків. Іншими словами, електричні сигнали від кожного з мікрофонів містять інформацію про звуки, що надходять з усіх боків. Обробляючи ці сигнали разом, ви можете вибрати звук, який надходить з певного напрямку. Таким чином, сам мікрофон не посилає звук з певного напрямку, а обробляє багатоканальний сигнал. Він може вибирати різні джерела звуку, встановлюючи різні параметри для обробки багатоканального сигналу від мікрофонних масивів або записаних багатоканальних фонограм.

Для зручності подальшого використання орієнтаційних шаблонів необхідно провести його аналіз. Напрямок залежить від геометрії мікрофонного масиву (кількість і положення мікрофонів) і алгоритмів обробки сигналу. На рис. 1.3 показана схема рівновіддаленої мікрофонної лінії з вісьмома елементами. Горизонтальна вісь - кут падіння акустичного сигналу, вертикальна - амплітуда вихідного сигналу в мікрофонній решітці в лінійній шкалі [24].

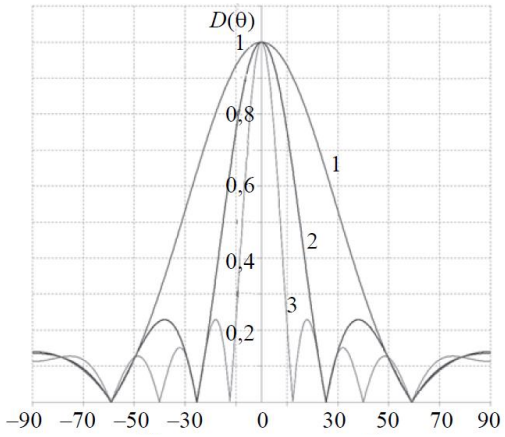

Рисунок 1.3 — Приклад діаграми спрямованості на базі 8 елементного масиву з мікрофонів

Основними параметрами діаграм спрямованості є ширина основної пелюстки (при 3 дБ), рівень бічних пелюсток (або відношення рівня основної пелюстки до рівня бічної) та так званий "індекс напряму", що характеризує здатність мікрофонного масиву генерувати пригнічувач ізотропного шуму. Всі властивості мікрофонної решітки залежать від частоти сигналу.

Аналіз методів проектування та основних параметрів мікрофонної решітки показав, що вони все частіше використовуються в різних системах обробки аудіо інформації, зокрема при віддаленому збиранні аудіо інформації. Її функціональність визначається її геометрією та алгоритмами цифрової обробки сигналів. Основні принципи роботи мікрофонного масиву та алгоритми обробки їх сигналів досі активно розробляються.

Оптимізація геометрії решіток заснована на статистичному моделюванні, аналітичного рішення проблеми немає. Оптимізація геометрії за різними критеріями призводить до різних структур з нееквідистантним розташуванням мікрофонів. Ряд закордонних компаній пропонують системи на базі мікрофонних масивів для вирішення різноманітних проблем, пов'язаних з обробкою голосових та аудіосигналів. Ефективність їх практичного застосування багато в чому визначається наявністю інформації, яка узагальнює досвід його практичного застосування.

Цифрова обробка сигналів, отримуваних з мікрофонної ріщітки, потребує їх перетворення у цифрову форму, що обумовлює необхідність застосування аналого-цифрових перетворювачів. Аналого-цифрові перетворювачі (АЦП) — це пристрої, які отримують аналогові вхідні сигнали і генерують відповідні цифрові сигнали, придатні для обробки мікропроцесорами та іншими цифровими пристроями. Класифікація АЦП наведена на рис. 1.4.

В принципі, не виключається можливість перетворення різних фізичних величин безпосередньо в цифрову форму, але ця проблема рідко може бути вирішена через складність таких перетворювачів. Тому зараз це найефективніший спосіб перетворення величин різної фізичної природи спочатку в функціональні електричні, а потім за допомогою кодово-цифрових перетворювачів напруги. Саме ці перетворювачі маються на увазі, коли згадуються АЦП.

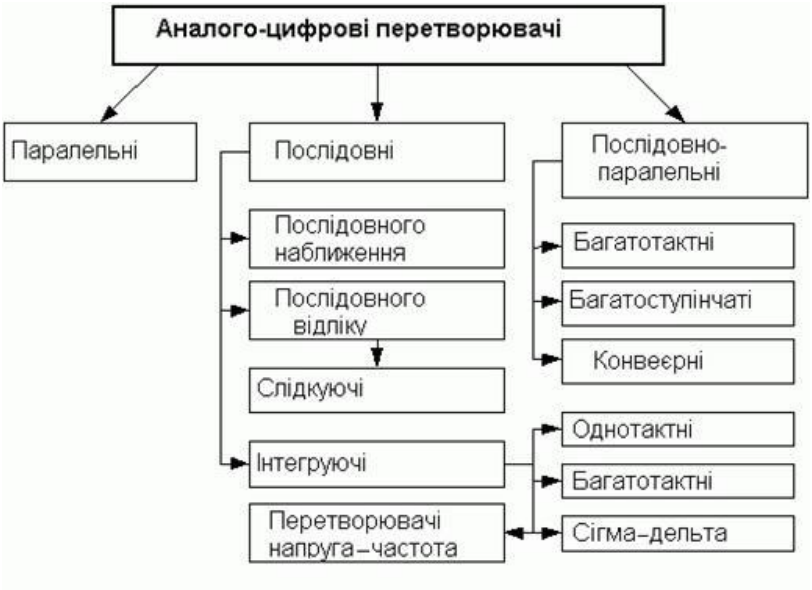

Рисунок 1.4 — Класифікація аналогово-цифрових перетворювачів

Процес аналого-цифрового перетворення безперервних сигналів, який реалізується за допомогою АЦП, являє собою перетворення неперервної в часі функції $U(t)$, яка описує вхідний сигнал, у числову послідовність $\{U(t_j)\}$, j=0,1,2,…, які можна простежити до деяких фіксованих моментів часу. Цей процес можна розбити на дві окремі операції: вибірка та квантування.

Як згадувалося раніше, найпоширенішою формою вибірки є рівномірна вибірка, яка базується на теоремі вибірки. Відповідно до цієї теореми, миттєві значення сигналу $U(t_j)$ у дискретні моменти часу $t_j = j\omega t$ слід використовувати як коефіцієнти a_j, а період дискретизації слід вибрати з умови:

$$t = 1/2F_m$$

де F_m– максимальна частота спектра сигналу, що перетворюється.

Тоді, отримаємо відомий вираз теореми відліків.

$$U(t) = \sum_{j=-\infty}^{\infty} U(j\omega\Delta t)\frac{\sin[2\pi F_m(t - j\Delta t)]}{2\pi F_m(t - j\Delta t)}$$

Для сигналів із суворо обмеженим спектром цей вираз є тотожністю. Проте спектри реальних сигналів лише асимптотично наближаються до нуля. Застосування рівномірного сканування до таких сигналів викликає специфічні високочастотні спотворення в системах обробки інформації через сканування. Щоб зменшити це спотворення, необхідно або збільшити частоту дискретизації, або використовувати додатковий фільтр нижніх частот перед АЦП, який обмежує спектр вхідного сигналу перед його аналого-цифровим перетворенням.

Для введення та аналого-цифрового перетворення звукових сигналів у розроблювані системі використаємо звукову карту. Типова звукова карта містить звуковий чіп, який містить цифро-аналоговий перетворювач, який перетворює записаний або згенерований цифровий звук в аналоговий формат. Вихідний сигнал подається через стандартні з'єднання, зазвичай TRS або Cinch, до підсилювача, навушників або зовнішнього пристрою. Якщо кількість або розмір роз’ємів на задній панелі комп’ютера завеликий, їх можна зробити окремо. Більш просунуті звукові карти містять декілька мікросхем, щоб досягти вищої якості або покращити продуктивність різних операцій одночасно, наприклад, для запису музики в режимі реального часу, важливо, щоб синтез звуків відбувався з мінімальною затримкою процесора [25].

2 МЕТОДИ ТА АЛГОРИТМИ ОБРОБКИ ЗВУКОВИХ СИГНАЛІВ

В даному розділі магістерської кваліфікаційної роботи розглянемо методи та алгоритми, які можуть бути застосовані для створення пасивної аудіолокаційної системи для визначення напрямку звукових сигналів.

2. 1 Аналіз алгоритмів обробки сигналів мікрофонних решіток

Як було зазначено у першому розділі найбільш ефективним рішенням для вирішення задач просторової локалізації джерел звукових сигналів є застосування мікрофонних решіток. Мікрофонні решітки використовуються для вирішення широкого кола практичних завдань, що обумовлює застосування різних алгоритмів обробки сигналів. Вибір того чи іншого алгоритму обробки сигналів ґрунтується на виборах акустичної обстановки:

— вибір близького/віддаленого джерела звуку;

— вибір стаціонарної/динамічної акустичної обстановки;

— обробка в реальному часі / пост обробка.

Обробка мовних сигналів мікрофонними ґратами може відбуватися як у часовій, і у частотної області [26]. У часовій області сигнали кожному мікрофоні проходять через КІХ-фільтр і далі сигнали з усіх мікрофонів об'єднуються в один вихідний сигнал системи.

У частотній області широкосмуговий сигнал розбивається на вузькосмуговий за допомогою короткочасного перетворення Фур'є, які обробляються окремо.

Розглянемо алгоритми обробки сигналів, що набули найбільшого поширення:

— алгоритм затримки та підсумовування;

— алгоритми фільтрації та підсумовування;

— алгоритми обробки сигналів у решітках;

— алгоритми, які мінімізують потужність шуму на виході;

— алгоритми, що ґрунтуються на критерії мінімуму середньоквадратичної помилки;

— алгоритми, що ґрунтуються на критерії максимуму відношення сигнал/шум.

Алгоритм затримки та підсумовування, відомий як delay-and-sum beamforming, є найпростішим алгоритмом формування діаграми спрямованості (рис. 2.1) [27]-[29].

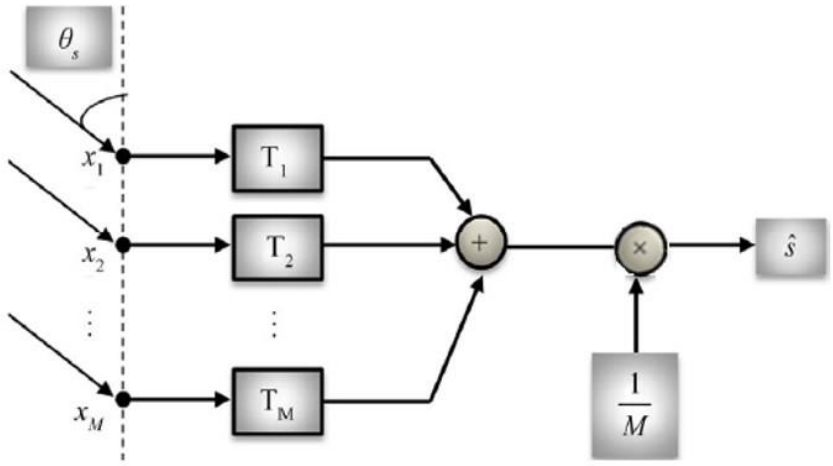

Рисунок 2.1 — Схема Алгоритма затримки та підсумовування

На кожен мікрофон (всього - M) вводиться тимчасова затримка Tm для подальшого підсумовування когерентного сигналів. Таке підсумовування дозволяє посилити корисний сигнал та послабити сигнали перешкод.

Мовний сигнал може бути розкладений на вузькосмугові частотні складові, затримки можуть бути апроксимовані фазовими зсувами кожної смуги частот. Амплітудний множник на виході є елементом усереднення: він обернено пропорційний кількості мікрофонів.

У часовій області вихідний сигнал системи представляється як:

$$S(t) = \frac{1}{M}\sum_{m=1}^{M} x_m(t - T_m)$$

Алгоритм фільтрації та підсумовування найбільш загальний клас алгоритмів. Застосовується для реверберуючих середовищ. Алгоритм фільтрації та підсумовування відрізняється тим, що амплітуда і фаза є частотно залежними параметрами [28, 29]. Сигнал на кожному мікрофоні проходить через нерекурсивний фільтр і потім піддається підсумовування з сигналами сторонніх каналів. Загальна схема алгоритму фільтрації та підсумовування зображена на рис. 2.2, а вихідний сигнал системи в частотній області є сумою частотно-залежних компонент.

$$S_{out}(f) = \sum_{n=1}^{N} w_n(f) x_n(f)$$

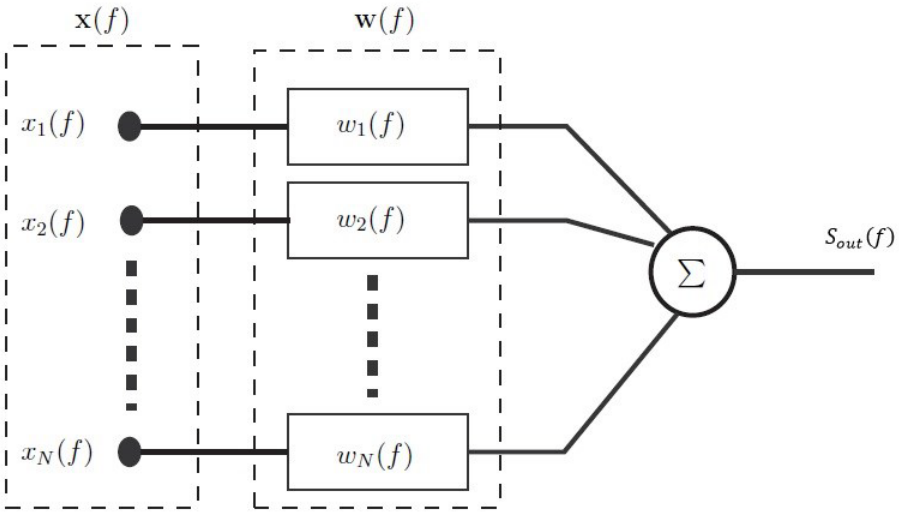

Рисунок 2.2 — Схема алгоритма фільтрації та підсумовування

Алгоритм обробки сигналів у підрешітках. Діаграми спрямованості мікрофонних решіток залежить від частоти прийнятого сигналу. При прийомі широкосмугових сигналів на різних частотах рівень бічних пелюсток перешкод різний, а також різна і ширина основної пелюстки діаграми спрямованості решітки. Щоб забезпечити прийом широкосмугового сигналу, створюється інваріантність діаграми спрямованості по частоті шляхом розбиття масиву елементів на ґрати. Кожна підрешітка є лінійним

еквідистантним масивом приймачів, що приймає сигнал у певній смузі частот. Для того, щоб рівень бічних пелюсток залишався незмінним у кожній підрешітці використовують фіксовану кількість елементів. Кількість елементів мікрофонної решітки може бути скорочено шляхом використання одного і того ж приймача в різних решітках.

Для вилучення вихідного сигналу мікрофонної решітки спочатку формуються S вихідних сигналів решіток різних діапазонів частот, які піддаються операції підсумовування [28]:

$$S_{out}(f) = \sum_{s=1}^{S} \sum_{n=1}^{N} w_n(f) x_n(f)$$

На рис. 2.3 наведена схема обробки широкосмугового сигналу за допомогою мікрофонної решітки з дев'яти елементів, розділених на чотири підрешітки, що обробляє широкосмуговий сигнал відповідно в чотирьох різних діапазонах частот огляді [28].

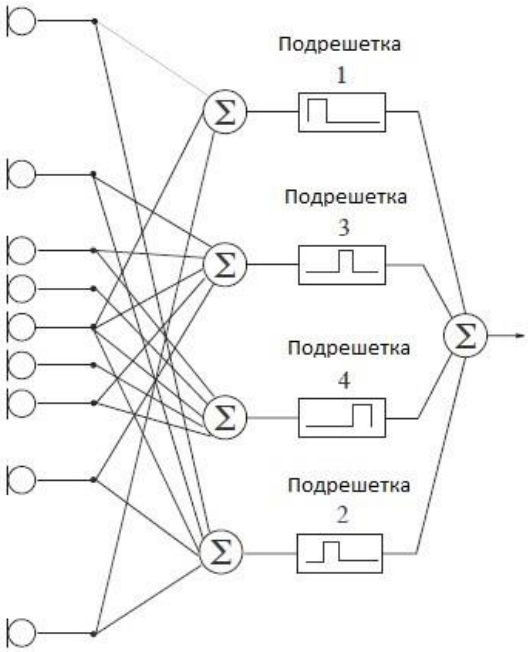

Рисунок 2.3 — Схема обробки широкосмугового сигнал

На рис. 2.4 наведений приклад обробки звукового сигналу з використанням мікрофонної решітки з двадцяти дев'яти елементів, які також розділена на чотири підрешітки для роботи в діапазоні частот від 500 Гц до 8 кГц [30]. Така технічна реалізація дозволяє підтримувати постійну ширину основного пелюстки діаграми спрямованості на всьому аналізованому частотному діапазоні.

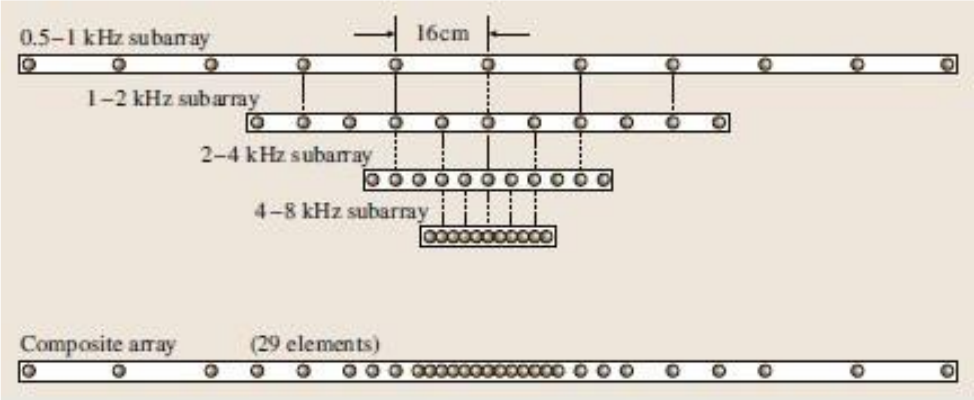

Рисунок 2.4 — Розташування мікрофонів у двадцяти дев'яти елементних ґратах

Розглянуті алгоритми спрямовані на реалізацію бажаного відгуку системи у заданому напрямку. Наприклад, необхідно отримати сигнал, що надходить з певного напрямку, і в цьому випадку бажаний відгук буде прийнятий за одиницю цього напрямку. Або, наприклад, доступна інформація про діючу на певній частоті перешкоді і про напрям з якого вона приходить, тоді бажаний відгук на цій частоті та напрямку дорівнює нулю.

Подальші алгоритми, які будуть розглянуті, ґрунтуються на статистичних властивостях шуканих та інтерференційних сигналів. Дані алгоритми оптимізують деяку функцію за рахунок адаптивної фільтрації вхідних сигналів, виділяючи корисний сигнал і відхиляючи перешкоди, що приходять з інших напрямків. Оптимізація полягає у застосуванні різних критеріїв, таких як максимальне відношення сигнал/шум (MSNR), мінімальна середньоквадратична помилка (MMSE), мінімальна дисперсія шуму (MVDR) та інші.

Алгоритми, що мінімізують потужність шуму на виході. Більшість адаптивних методів покладаються на мінімізацію середньоквадратичної помилки між опорним сигналом та вихідним сигналом. На жаль, алгоритм найменшого середньоквадратичного відхилення (СКО) може погіршити бажаний сигнал, оскільки він спрямований на мінімізацію середньоквадратичної помилки і не вимагає вимог до порога спотворень бажаного сигналу. Адаптивний алгоритм, який враховує цей недолік, називається алгоритмом Фроста [27] (рис.2.5).

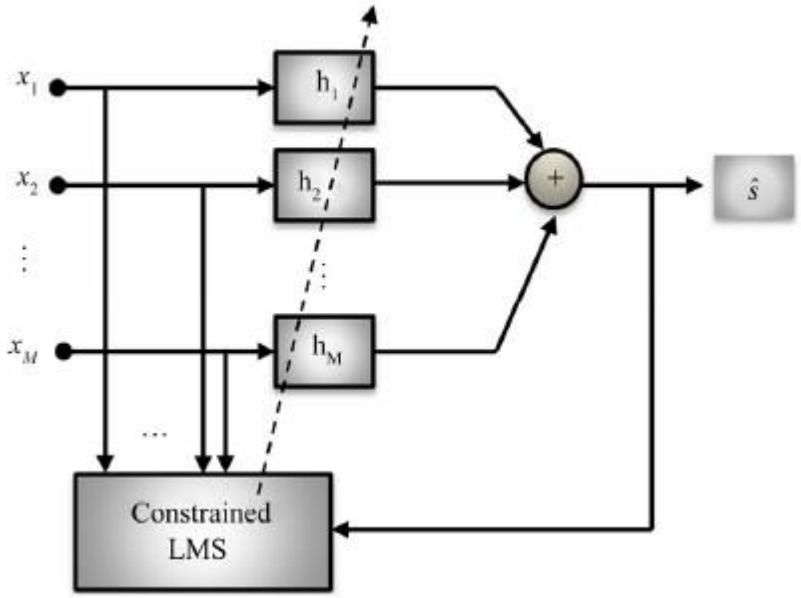

Рисунок 2.5 — Схема алгоритма Фроста

У цій схемі коефіцієнти фільтра адаптуються через використання алгоритму найменшого СКО. Алгоритм використовується для мінімізації потужності шуму на виході при збереженні постійного коефіцієнта посилення та лінійної фази на відгук системи у напрямку корисного джерела.

До цього класу можна віднести алгоритм пригнічення бічних пелюсток (рис. 2.6). Схему алгоритму можна поділити на верхній та нижній тракт. Верхній призначений формування оцінки корисного сигналу. Сигнали вирівнюються за часом і з допомогою формування діаграми спрямованості тракт настроюється корисний сигнал. Нижній тракт забезпечує виключення корисного сигналу за рахунок блокування матриці B [31]. Після блокуючої

матриці сигнали нижнього тракту проходять фільтр a, що адаптивно мінімізує потужність шуму на виході. Вихідний сигнал системи є різницевий сигнал верхнього та нижнього тракту: з оцінки корисного сигналу віднімається загальний сигнал інтерференції.

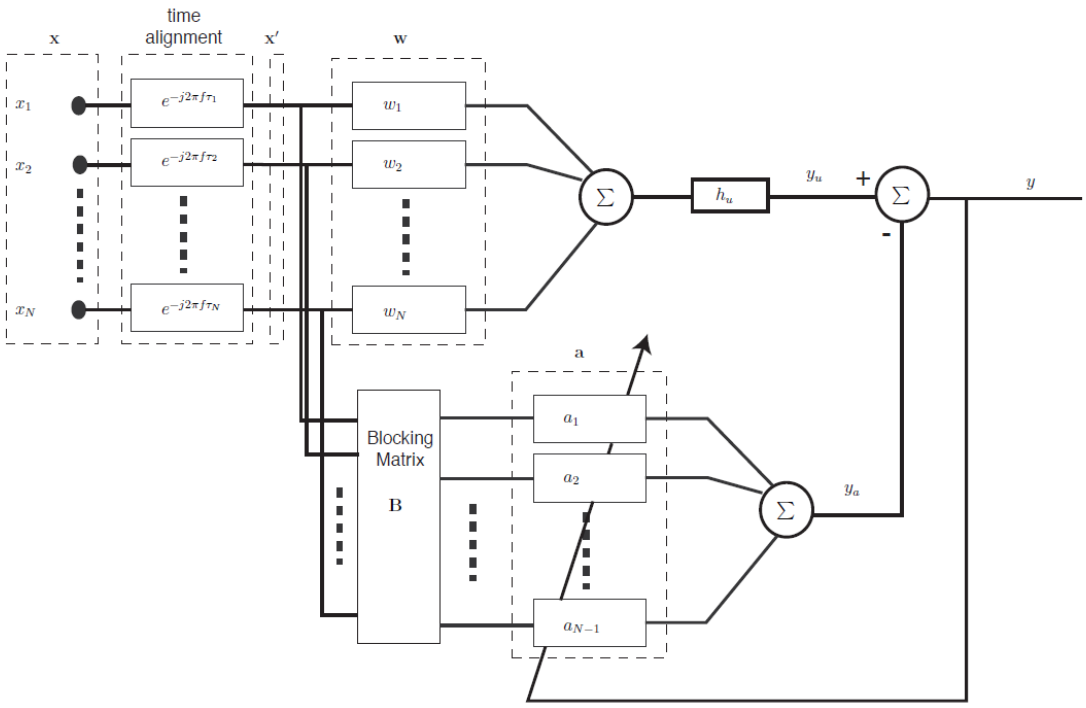

Рисунок 2.6 — Схема алгоритму придушення бічних пелюсток

У роботі [32] описаний алгоритм, що мінімізує потужність шуму на виході для двоелементної диференціальної мікрофонної решітки, що адаптивно формує нулі в напрямку джерел когерентного шуму.

Алгоритми, що ґрунтуються на критерії мінімуму середньоквадратичної помилки використовуються за наявності достатньої інформації про корисний сигнал. Якщо ця умова виконується, є можливість сформувати опорний сигнал q(t). Сигнал помилки f(t) відображає різницю між бажаною реакцією мікрофонної решітки та вихідним сигналом решітки [33]. Алгоритми, засновані на мінімізації середньоквадратичної помилки прагнуть виключити цю різницю.

$$f(t) = q(t) - W^H(S + X(t))$$

де W – ваговий вектор;

S – вектор корисного сигналу,

X – вектор інтерференції.

На рис. 2.6 показана схема реалізації алгоритму мінімізації СКО.

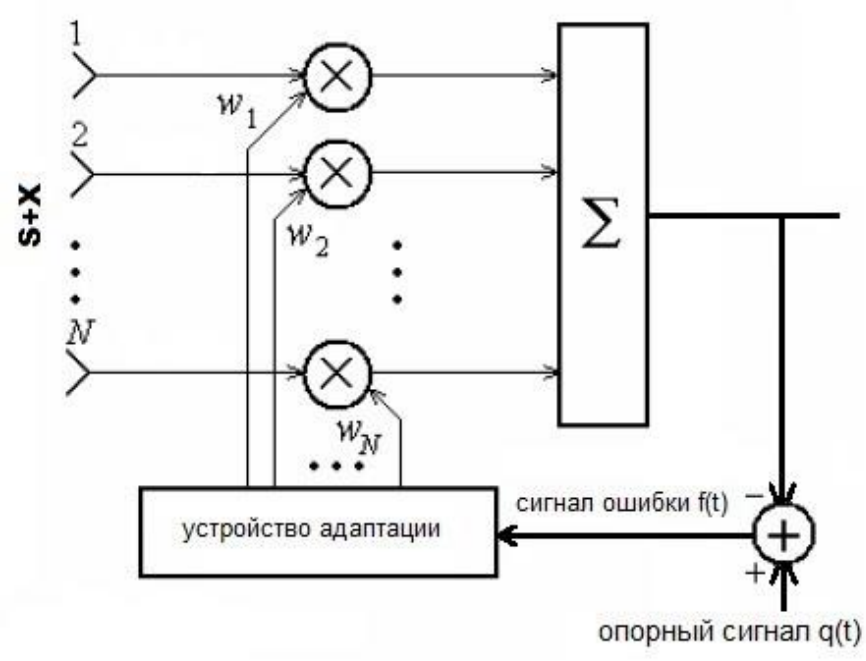

Рисунок 2.6 — Схема алгоритма мінімалізації СКО

Мінімальне середньоквадратичне відхилення за такої постановки завдання:

$$< |f|^2 >=< |q|^2 - R^H M^{-1} R$$

а вираз для оптимальних вагових коефіцієнтів за критерієм мінімізації середньоквадратичної помилки можна знайти з умови:

$$W = M^{-1} R$$

де R – кореляційний вектор.

$$R =< (S + X(t))q(t) >$$

За даним критерієм працюють алгоритми постфільтрації [27, 28]: алгоритм фільтрації та підсумовування з додаванням фільтра до виходу системи (рис. 2.7).

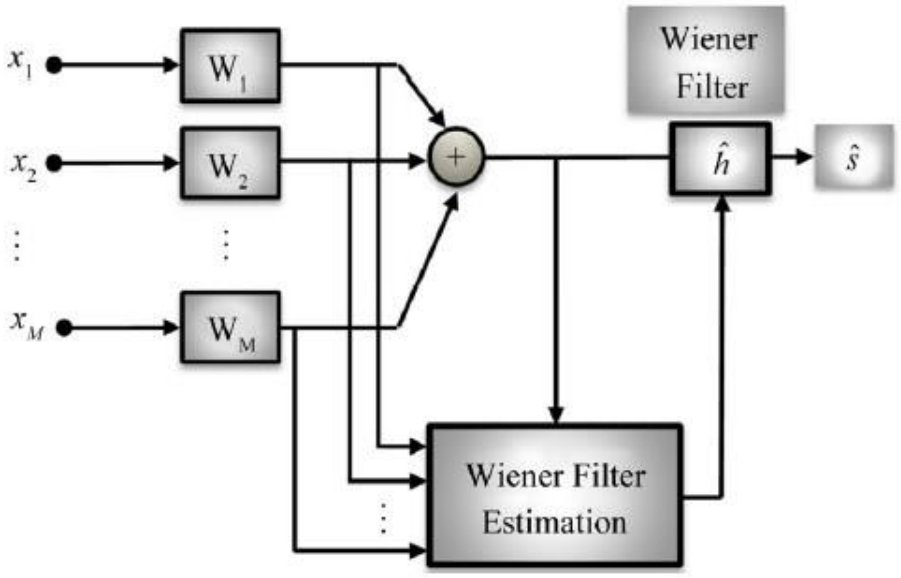

Рисунок 2.7 — Схема алгоритму постфільтрації Вінера

Алгоритми, що базуються на критерії максимуму відношення сигнал/шум приймається відношення середньої потужності корисного сигналу до середньої потужності перешкоди (власного шуму приймачів та зовнішніх джерел перешкод).

Відношення сигнал/шум на виході мікрофонних ґрат можна записати в матричній формі [46],[33]:

$$\frac{W^H M_s W}{W^H M W}$$

де M — кореляційна матриця перешкод, Ms — кореляційна матриця корисного сигналу.

Оптимальний ваговий вектор, що забезпечує максимізацію цього відношення, можна знайти з умови:

$$W = M^{-1}S$$

При цьому максимальне значення відношення сигнал/шум дорівнює:

$$\Lambda_{max} = \varsigma S^H M^{-1} S_\varsigma$$

де ς- відношення потужності корисного сигналу до потужності власного шуму в окремому елементі решітки.

2.2 Розробка алгоритму роботи системи визначення напрямів звукових хвиль

З метою зниження вимог до продуктивності апаратних засобів та спрощення програмного забезпечення та процесів розробки було прийнято рішення розділити програму на два модулі. Перший модуль для обробки звукових сигналів, другий модуль для відображення графічних даних. Це дозволяє ефективно використовувати пам'ять, зменшує появу різноманітних неточностей у розрахунках і значно прискорює роботу лише обох модулів. Загальний алгоритм роботи наведено у додатках.

Загальний алгоритм роботи створено на основі роботи цифрових антенних сіток. Вони являють собою пасивні або активні системи, тобто аналогово-цифрові (цифрово-аналогові) канали із загальним фазовим центром, в яких формування структури відбувається в цифровому вигляді без використання фазообертів..

Обробка аудіосигналу є одним з найважливіших напрямків дослідження в обробці сигналів. Йдеться про електронні маніпуляції звуковими сигналами. Оскільки аудіосигнали можуть бути представлені в будь-якому цифровому

або аналоговому форматі, обробка може здійснюватися в будь-якій області. Аналогові процесори діють безпосередньо на електричний сигнал, тоді як цифрові процесори працюють математично на цифрове представлення цього сигналу.

Цифрове представлення звукового сигналу розпізнається як послідовність символів, зазвичай двійкових чисел. Це дозволяє обробляти сигнали за допомогою цифрових схем, таких як цифрові сигнальні процесори, мікропроцесори та комп'ютери загального призначення. Більшість сучасних аудіосистем використовують цифровий підхід; методи обробки цифрових сигналів набагато потужніші та ефективніші, ніж обробка аналогових сигналів.

Методи та програми обробки включають зберігання даних, стиснення, вилучення музики, обробку мовлення, локалізацію, акустичне розпізнавання, передачу, шумозаглушення, акустичний відбиток пальців, розпізнавання тону, синтез і посилення (наприклад, вирівнювання, фільтрацію, стиснення рівня, луна) та погашення реверберації або додати.

2.2.1 Розробка алгоритму роботи звукового модуля

Цей модуль бере на себе основну роботу по представленню діаграм спрямованості, тому що без швидкого і якісного розрахунку великого потоку даних нічого б не відбулося, тому при виборі мов ООП вибір був зроблений на C++ :

— програма зчитує аргументи, передані в командний рядок, а саме IP-адресу, порт і кількість використаних потоків;

— за допомогою бібліотеки PortAudio програма ініціалізує аудіоінтерфейс операційної системи та визначає пристрій запису, встановлений за замовчуванням;

— програма формує потік запису та обробки аудіосигналу, який надходить на пристрій запису, визначений на кроці 2;

— за допомогою аргументів командного рядка, введених користувачем, програма створює сервер і розподіляє його навантаження між заданою користувачем кількістю потоків;

— програма очікує на підключення клієнта;

— кожному заданому інтервалі (100 мс) програма зчитує потік даних, що надходить на записуючий пристрій;

— обробка отриманих даних;

— оброблені дані надсилаються всім підключеним клієнтам через сокети;

— повернутися на 3 кроки назад.

2.2.2 Розробка алгоритму роботи графічного модуля.

З використанням одноплатного комп’ютера виникла ідея перетворити його на сервер і надсилати інформацію клієнтам через розетки. Алгоритм цієї системи можна пояснити наступним чином. Архітектура клієнт/сервер – це обчислювальна архітектура виробник/споживач, в якій сервер діє як виробник, а клієнт – як споживач. Ці послуги можуть включати доступ до програм, зберігання даних, спільний доступ до файлів, доступ до принтера або прямий доступ до необмеженої обчислювальної потужності сервера.

Архітектура клієнт/сервер працює, коли клієнтський комп’ютер надсилає запит на ресурс або процес на сервер через мережеве з’єднання, який потім обробляється та передається клієнту. Серверний комп’ютер може одночасно керувати кількома клієнтами, тоді як один клієнт може бути підключений до кількох серверів одночасно, кожен з яких надає різні послуги. У найпростішій формі Інтернет також базується на архітектурі клієнт/сервер, у якій веб-сервери одночасно обслуговують багато користувачів даними веб-сайтів. [33].

Сокет — це програмний об’єкт, який діє як кінцева точка та створює двосторонній мережевий зв’язок між сервером і програмою-клієнтом.

URL-адреси та їх з’єднання здебільшого використовуються для доступу до Інтернету, але іноді програми вимагають простого з’єднання між клієнтом програми та стороною сервера.

Коли клієнт спілкується із сервером, наприклад, надсилаючи запити до бази даних, довірений сервер і клієнтське з’єднання встановлюються через канал зв’язку TCP. За допомогою цього типу зв’язку клієнт і сервер можуть читати або записувати в сокети, які прив’язані до певного каналу зв’язку.

Розетки в основному діляться на два типи: активні і пасивні. Активні розетки підключаються до віддалених розеток через відкрите підключення для передачі даних. Якщо це з’єднання закрито, активні сокети на кожній кінцевій точці будуть знищені. Пасивні сокети не підключені; Замість цього вони чекають на вхідне з'єднання, яке генерує новий активний сокет.

Хоча між сокетом і портом існує тісне з'єднання, насправді сокет не є портом. Кожен порт може мати пасивний сокет, який очікує на вхідні з’єднання, і кілька активних сокетів, кожен з яких відповідає відкритому з’єднанню на порту.

Оскільки модуль є сторінкою в браузері, його робота починається після завантаження веб-сайту:

— після завантаження сторінки відтворюється інтерфейс програми, в якому клієнту пропонується ввести IP-адресу сервера модуля обробки аудіоданих;

— після введення IP-адреси встановлюється з'єднання сокету з сервером;

— очікування даних;

— отримані дані конвертуються у відповідну структуру для успішного відображення бібліотекою CanvasJS;

— презентація розкладу;

— повернутися на 3 кроки назад;

2.3 Розробка загального алгоритму роботи системи

Основний алгоритм роботи, завдяки його обширним математичним розрахункам, здійснюється програмною частиною проекту, тому опис цієї схеми є лише оглядовим.

Після встановлення мікрофонної стійки та підключення до зовнішніх звукових карт. Реалізується наступний алгоритм:

— мікрофонна решітка приймає сигнал;

— сигнал надходить до звукової карти та оцифровується;

— записати цей сигнал в масив даних Raspberry Pi і створити на його основі сервер;

— масиви даних надходять в ПК, який являється клієнтом системи;

— відбуваються математичні розрахунки за допомогою програмної частини проекту;

— результат програми рендерінг графіка в в вікні браузера.

Коли аудіосигнал зустрічає мікрофони XLR з фантомним живленням, які використовують технологію фазової стерефонії, вони сприймають сигнал по-різному через різну відстань між ними. Після оцифровки сигналу за допомогою АЦП він посилюється і подається мікропроцесорний модуль, де записується в масивах частот і кількості вибірок, які одразу зміщуються в елементи каналів і обчислюється середньоквадратичне значення рівнів сигналу, після чого ці масиви відправляються на ПК і діляться на канали і формують масив вихідних даних, який відображається у вигляді шаблону в вікно браузера в кінці. Алгоритм роботи системи наведено в додатку.

Програмна частина буде здійснювати моніторинг акустичної системи в процесі клієнт/сервер, не перебуваючи безпосередньо на місці розташування об’єкта, що охороняється. Це, в свою чергу, мінімізує ризик для оператора, якщо він намагається ввійти в зону без дозволу, що підвищує безпеку для обслуговуючого персоналу.

Основна перевага запропонованого підходу полягає можливість проведення контролю на великій площі з мінімальними витратами. Дві мікрофонні решітки дозволять контролювати площу в один квадратний кілометр. Основним проблемами, які потрібно вирішувати, є боротьба з фоновим шумом, який буде впливати на коректність кінцевих результатів.

Акустичний шум — це випадковий процес, викликаний рухом, коливаннями внутрішніх структур пружного середовища під впливом зовнішніх і внутрішніх фізичних, хімічних, біологічних факторів. Ці хвилі можуть нести корисну інформацію, тоді їх називають шумовим сигналом. Наприклад, шум вітру, шум літаків, вібрації від машин і транспортних засобів, навколишній шум і природні явища можуть істотно вплинути на роботу перевіреного алгоритму в ідеальних умовах. Шуми, які не містять корисної інформації та погіршують вимірювальні властивості пристрою локації, класифікуються як перешкоди. За походженням акустичний шум поділяється на пасивний і активний. Активний шум (інтерференція) — це сигнали, які викликані внутрішніми фізичними, хімічними, біологічними процесами в пружних середовищах. В Світовому океані джерелами такого шуму є, наприклад, поверхневі хвилі, підводні течії, шум прибою, дощ, біологічна активність океану тощо.

Пасивний шум викликається впливом акустичного сигналу на пружне середовище — це відомо як реверберація. Пасивний шум зумовлений розсіюванням акустичних хвиль на неоднорідностях середовища, поперечний розмір якої становить більше 1/100 довжини хвилі, а також відбиттям акустичних хвиль від шарів і кордонів сусідніх середовищ. Це перехідні процеси, параметри яких змінюються з часом. Запровадження регулювання посилення в часі зміщує процес з нестійкого до стаціонарного.

З огляду на випадковість шуму, спотворення параметрів сигналу в акустичному комплексі є найбільш придатним для опису та визначення характеристик випадкового процесу математичним апаратом теорії ймовірностей. Часто відомі деякі параметри акустичного сигналу (тиск,

швидкість вібрації), на основі яких можна оцінити статистичні властивості сигналу. Такий підхід може дати хороший результат при дотриманні умов ергодичності. Відомо, що випадковий стаціонарний процес є ергодічним, якщо всі його статистичні властивості можна знайти на основі реалізації цього процесу достатньо великої тривалості. До таких ВП належать стаціонарні акустичні шуми, перешкоди та деякі сигнали. Отже, властивості акустичних сигналів виявляються з аналізу реалізації процесу за більш тривалий період часу.

Ще один недолік — розміри акустичної системи. Його розміри не великі в межах цільової області, але загалом буде важко переміщати цю систему від об’єкта до об’єкта. Використання аудіолокатора розраховане на стаціонарне використання з мінімальною коригуванням обслуговуючого персоналу.

Продуктивність класичних багатоканальних електронних алгоритмів дискретизації була обмежена використанням звичайних компенсацій і вивченням кореляцій сигналів і часто не працюють в реальному часі, а їх методи передаються незалежно від конкретних сигналів, що обробляються.

Завдяки циклічному використанню багатоканальних сигналів у системі аудіолокації для кожної із зазначених дискретних завдань і координатора об’єктів для багатоканального завдання сканування може бути забезпечено різними графічними процесорами (за допомогою адаптерів PC-GPU).

У відповідних пристроях, які містять похибки, з найбільшими коефіцієнтами та перекриттям об’єктів, узгодженими джерелами, сигнали від сусідніх дискретних можуть бути нечіткими. Щоб уточнити координати дискретного об’єкта на місцевості, можна створити логічний вузол інверсного виходу, який використовується для розв’язування нечітких рівнянь систем. Однак переважна більшість методів розкриті в аналітичних системах, які обмежуються наявністю точних розмов, які зазвичай не характерні для аудіопозицій.

Усуваючи недоліки, слід зробити захисний екран для мікрофонів, оскільки більш чутлива конструкція конденсаторного мікрофона не

функціонує належним чином під впливом води. Навіть невелика кількість вологи може пошкодити конденсаторний елемент. Мікрофони зі знімними ґратами та лобове скло можна почистити, як описано вище. Щоб почистити мікрофони з жорсткою сіткою, використовуйте суху зубну щітку з м’якою щетиною та очистіть сітку.

3. РОЗРОБКА АПАРАТНИХ ЗАСОБІВ

В даному розділі магістерської кваліфікаційної роботи проведемо розробку апаратної частини розглядуваної системи. Розробку розпочнемо з вибору елементної бази під час якого перевагу будемо надавати закінченим функціональним модулям.

3.1 Вибір апаратного забезпечення.

Одним з основних елементів системи є мікрофон. Основними характеристиками мікрофонів є такі:

— чутливість;

— частотна чутливість;

— акустичні властивості;

— особливості орієнтації;

— власний рівень шуму мікрофона.

. Чутливість мікрофона визначається відношенням напруги на виході мікрофона до звукового тиску P0, як правило, у вільному звуковому полі, тобто без впливу відбиваючих поверхонь. Коли синусоїдна звукова хвиля поширюється в напрямку робочої осі мікрофона, цей напрямок називається осьовою чутливістю і вимірюється в мВ/Па:

$$M0 = U/P_0$$

Частотна чутливість (ВЧ) — це залежність осьової чутливості мікрофона від частоти звукових коливань у вільному полі. Нерівномірність ФЧ зазвичай вимірюється в децибелах у вигляді двадцяти логарифмів (на основі 10), чутливість мікрофона на певній частоті до чутливості на еталонній частоті (зазвичай 1 кГц) [34].

Вплив звукового поля мікрофона визначається акустичною характеристикою, яка визначається відношенням сили, що діє на мембрану мікрофона, і звукового тиску у вільному звуковому полі: A = F / P, а чутливість мікрофона M = U / P можна представити у вигляді U / P = U / F • F / P і виразити в A. Тоді отримаємо: M = A • U / F. Відношення напруги на виході мікрофона до сили, що діє на мембрані U/F характеризує мікрофон як електромеханічний перетворювач. Акустичні властивості визначають спрямованість мікрофона.

Характеристикою напрямку є залежність чутливості мікрофона від напрямку падіння звукової хвилі відносно осі мікрофона. Він визначається відношенням чутливості Mα, якщо звукова хвиля падає під кутом α до акустичної осі мікрофона, до його осьової чутливості:

$$\varphi = M\alpha / M_0$$

Характеристика спрямованості описує, як мікрофони вловлюють звук, де мікрофони «чують» просторово і які позиції заблоковані. Вибір мікрофона з потрібною спрямованістю дозволить виділити потрібний звук і мінімізувати небажаний шум [35]. За характеристиками спрямованості мікрофони поділяються на : лінійні, «вісімкові», всеспрямовані та кардіоїдні.

Лінійні мікрофони мають трубчасту конструкцію, яка дозволяє отримати їх високу спрямованість (рис. 3.1). Чутлива капсула мікрофона знаходиться на кінці інтерференційної трубки, яка усуває звук з боків шляхом придушення фази. Така конструкція забезпечує більш жорстку характеристику спрямованості спереду передньої частини з більшим діапазоном лову.

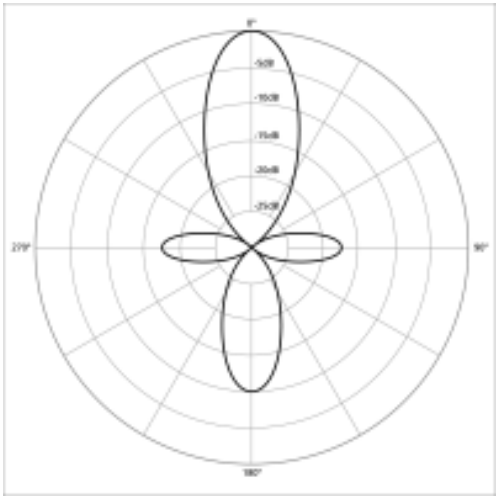

Рисунок 3.1 — Спрямованість лінійного мікрофона

Діаграма спрямованості «вісімкового» мікрофона нагадує цифру 8 (рис. 3.2). З діаграми спрямованості витікає, , що такі мікрофони вловлюють звук як спереду, так і позаду них і відхиляють дві сторони. Ця передня та задня чутливість робить їх ідеальними для запису стерео та для запису двох або більше інструментів. Вони по суті схожі на всеспрямовані мікрофони, але мають відхилення звуку з двох сторін. Хоча «вісімкові» мікрофони не такій популярні, таку діаграму спрямованості реалізують в стрічкових та конденсаторних мікрофонах.

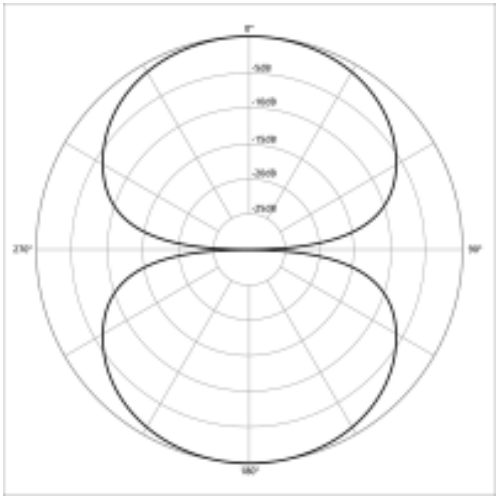

Рисунок 3.2 — Спрямованість “Вісімкового” мікрофона

Діаграма спрямованосі всеспрямованого мікрофона має форму

кола (рис. 3.3). Такий мікрофон однаково сприймає звук з усіх боків. Завдяки своїй непрямій конструкції та нульовому відхиленню ці мікрофони дозволяють краще захоплювати сприймають звук. Головним недоліком всеспрямованих мікрофонів є те, що вони, як правило, контролюють зворотний зв’язок, що робить їх непридатними для гучних і галасливих місць.

Кардіоїдні мікрофони, які захоплюють все, що знаходиться перед собою, і блокує все інше. Це дозволяє нам направляти мікрофон на джерело звуку та ізолювати його від небажаного зовнішнього шуму, що робить його ідеальним для живих та інших ситуацій, де потрібне придушення шуму та зворотного зв’язку. Діаграма спрямованості такого мікрофону показана на рис. 3.4.

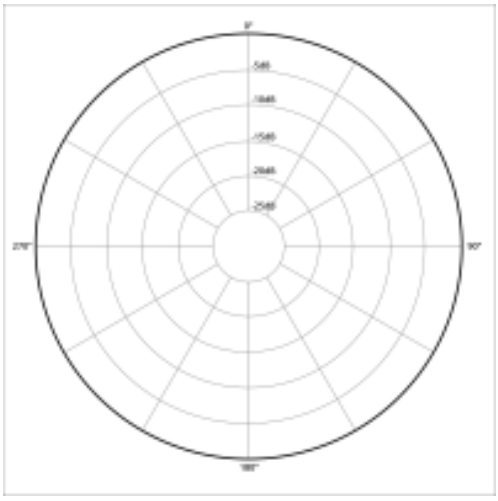

Рисунок 3.3 — Спрямованість всенаправленого мікрофона

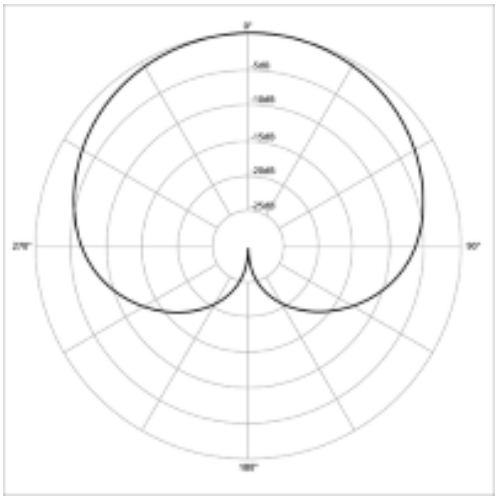

Рисунок 3.4 — Спрямованість кардіоїдного мікрофона

Кардіоїдні мікрофони значно перевершують всі інші за популярністю і широко використовуються в живих виступах, від караоке до великих концертів на арені. Іншим поширеним використанням є гучні мікрофонні інструменти, такі як барабани та гітарні колонки. Зауважте, що ці типи мікрофонів додають тонкий колір звуку, коли джерело знаходиться поза віссю. Тому положення мікрофона дуже важливе під час розмови та співу [36].

Згідно з міжнародними стандартами, рівень власного шуму мікрофона визначається як рівень звукового тиску, який створює на виході мікрофона напругу, що відповідає напрузі, що створюється в ньому лише його внутрішнім шумом без звукового сигналу.

Основні типи мікрофонів та їх основні характеристики представлені у таблиці 1.

Таблиця 1— Порівняння типів мікрофонів

Мікрофон	Гц	дБ	мВ/Па
Вугільний	301—3400	21	1002
Динамічний	101—10000	13	0,5~1,1
Конденсаторний	31—15000	6	6
Електродинамічний смугового типу	71—15000	11	1,6

Динамічний мікрофон (рис. 3.5) використовує діафрагму, звукову котушку та магніт. Звукова котушка розташована в магнітному полі і створюється в задній частині мембрани. При переміщенні звукової котушки в магнітному полі виникає електричний сигнал, що відповідає отриманому звуку.

Динамічні мікрофони відносно прості, недорогі та мають високий рівень стабільності.

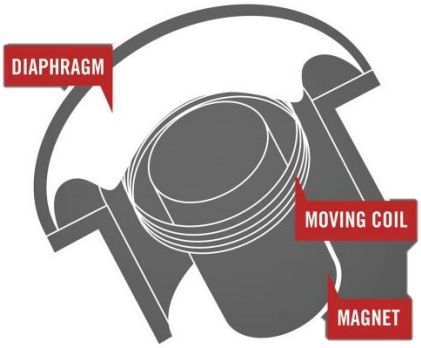

Рисунок 3.5 — Динамічний мікрофон

Ці мікрофони дають змогу працювати з часто високими рівнями звукового тиску і в основному нечутливі до екстремальних температур і вологості.

Смуговий мікрофон (рис. 3.6) — це тип динамічного мікрофона, який використовує тонкий електронний радіозв'язок між полюсами магніту. Лінійні мікрофони зазвичай двонаправлені. Вони вловлюють звуки, які знаходяться спереду або ззаду, але не з боків.

Основним елементом конденсаторних мікрофонів (рис. 3.7) є конструкція електрично зарядженої мембрани (backplate), яка є звукочутливим конденсатором. Коли діафрагма приводиться в дію звуком, зазор між діафрагмою і задньою пластиною змінюється, і, як наслідок, змінюється ємність конденсатора.

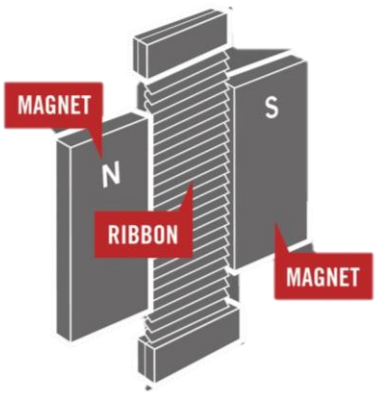

Рисунок 3.6 — Смуговий мікрофон

Конденсаторні мікрофони є більш чутливими і забезпечують більш плавний і естетичний звук, особливо на високих частотах [37].

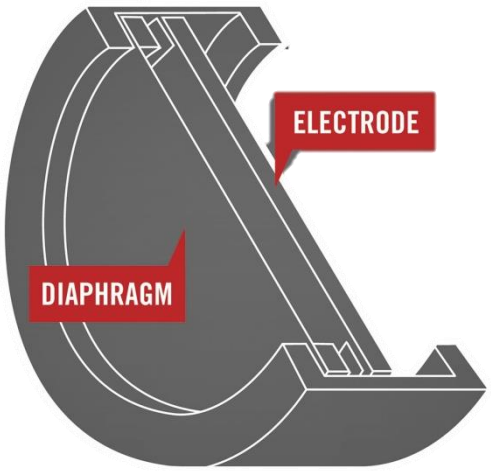

Рисунок 3.7 — Конденсаторний мікрофон

Виходячи з матеріалу, описаного вище, вибір припав на конденсаторний мікрофон Behringer C1.

Іншим головним елементом системи є мікропроцесорний модуль. Його можна реалізувати на мікроконтролері – цифровому мікропроцесорному пристрій, що містить на одному кристалі процесорне ядро та всі необхідні периферійні пристрої для реалізації спеціалізованого мікрокомп'ютера, спрямованого на розв’язування задач контролю/управляння. Різноманітні периферійні пристрої, якими обладнані сучасні мікроконтролери, дозволяють без додаткових апаратних витрат реалізувати взаємодію з іншим зовнішнім обладнанням, досягаючи максимальної ефективності у вирішенні задач контролю та управління.

Іншою альтернативою використання окремого мікроконтролера є застосування готових мікроконтролерних модулів – одноплатних комп′ютерів. Одноплатні комп′ютери є повнофункціональними мікрокомп′ютерами, які містять на одній платі усе необхідне для швидкої реалізації потрібних технічних рішень. Як правило, це процесор, графічне ядро, USB і мережеві інтерфейси, як провідні, так і безпровідні. Оскільки одноплатні комп′ютери є

максимально завершеними, вони дозволяють мінімізувати витрати на розробку. Враховуючи викладене вище, для даної розробки будемо використовувати одноплатний комп′ютер.

Відносно новим одноплатним комп’ютером на ринку є Tinker Board від Asus, одного з провідних світових постачальників якісної електроніки. Комп′ютер Tinker Board побудований на 4-ядерному процесорі Cortex-A17 з частотою 1,8 ГГц, підтримує можливість підключення відеоінтерфейсу HDMI, який може відтворювати зображення в якості 4K, має 2 ГБ пам'яті під вимоги користувача, слот для зберігання даних у вигляді SD. карта до 16 ГБ, наявність мережевих інтерфейсів, а саме Gigabit Ethernet і Wi-Fi, 4 порти USB 2.0, наявність аудіокодека, підтримка Android [38]. Схематичне розташування всіх деталей показано на рис. 3.8.

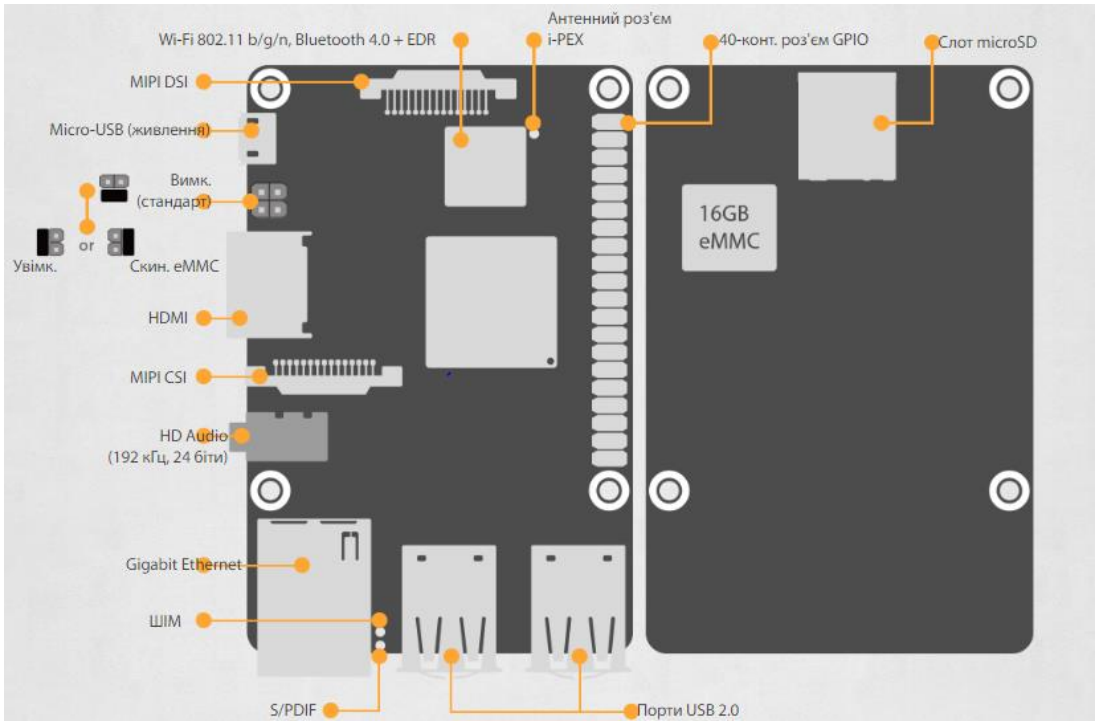

Рисунок 3.8 — Схема одноплатного комп’ютера Asus Tinker Board

Ще одним з варіантів є одноплатний комп′ютер BeagleBone (рис. 3.9), що пропонується компанією BeagleBone , яка на відміну від Asus, виробляє

широкий спектр електроніки, вона високоспеціалізована і використовує весь свій потенціал для створення власних одноплатних комп’ютерів.

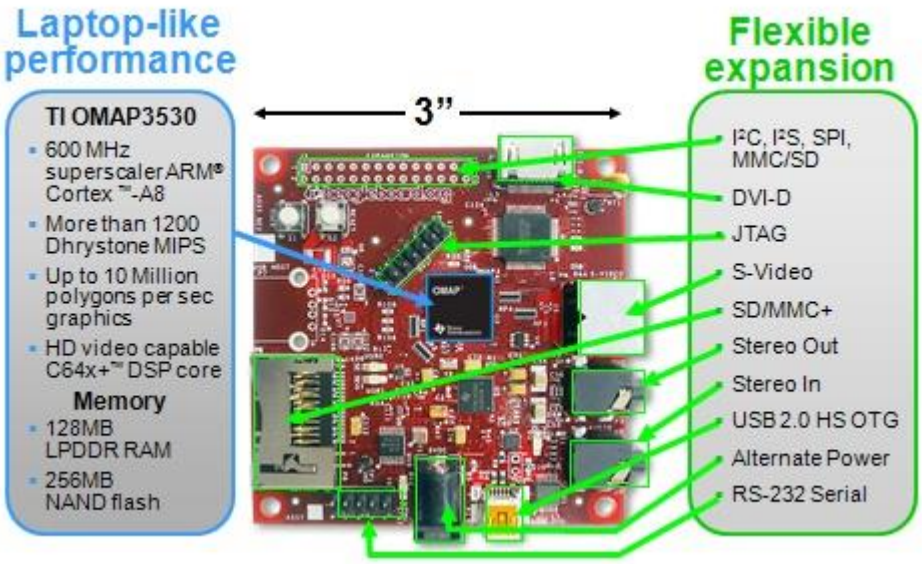

Рисунок 3.9 — Складові BeagleBone

Комп′ютер BeagleBone побудований на базі процесора TI OMAP3530 з ядром ARM Cortex-A8 з тактовою 720 МГц A8, має порти USB і SD, оперативну та постійну пам'яті об′ємом 256 МБ, а також порт для підключення камера. Для цієї плати можливий вибір операційної системи вже може бути між Windows CE і Linux. Основним недоліком міні-комп’ютера є те, що його можна завантажити лише з SD-карти, що, у свою чергу, має тенденцію тримати руки користувача дещо закованими в наручники [39].

Найкращим варіантом для вирішення поставлених задач є одноплатний комп′ютер Raspberry PI 3. Основною відмінністю від попередньої версії є використання потужнішого процесора з частотою 1,4 ГГц (4-ядерний ARM Cortex-A53), дводіапазонного WiFi модуль з 802.11ac, модуля Bluetooth 4.2, більш швидкодіючого модуля Ethernet з можливістю живлення через нього, покращеною можливістю завантаження PXE/USB накопичувачів, покращеною температурна продуктивністю.

У моделі Raspberry Pi 3 наявний 40-піновий роз'єм GPIO, серед контактів якого є інтерфейси SPI, I²C, UART. Крім того, на платі є інтерфейси дисплея

MIPI DSI (Display Serial Interface), відеокамери 15-pin MIPI CSI-2 (Camera Serial Interface), HDMI, комбінований 3.5 мм аудіо роз'єм і композитне відео. Живлення Raspberry Pi 3 Model B здійснюється від 5-вольтного адаптера через роз'єм micro-USB або роз′єма живлення. Рекомендується використовувати джерело живлення з силою струму не менше 2 А.

Raspberry Pi працює в основному на операційних системах, заснованих на Linux ядрі: Raspbian, Ubuntu MATE, Ubuntu Core, Android, Arch Linux ARM, openSUSE, Gentoo Linux, CentOS, Fedora, Kali Linux. Також можлива установка Windows 10 IOT, RISC OS Pi, FreeBSD, NetBSD і ін.. [40].

На рис.3.10 наведений зовнішній вигляд одноплатного комп'ютера Raspberry Pi 3 . Основні характеристики представлені у табл. 2 [40].

Рисунок 3.10 — Raspberry Pi 3B+

Таблиця 2 — Основні характеристики одноплатного комп'ютера Raspberry Pi 3 Model B+

Характеристика	Raspberry Pi 3 Model B+
Основний чіп	Broadcom BCM2837
Архітектура ядра	64-бітний чотирьохядерний ARMv8 Cortex-A53 процесор з тактовою частотою 1.2 ГГц;
Графічний співпроцесор	2-ядерний Video Core IV® Multimedia
Оперативна пам’ять	1ГБ LPDDR2 SDRAM (900 MHz)
Постійна пам'ять	слот Micro-SD

Продовження таблиці 2

Інтерфейси	Ethernet, WIFI 802.11n + Bluetooth 4.1 Low Energy (BLE), MiniHDMI, 4×Micro-USB 2.0, Композитный RCA, 15 MIPI Camera Serial Interface (CSI-2), 40 контактів портів введення/виведення;
Габаритні розміри	85 мм x 56 мм x 17 мм

Перетворення аналогових сигналів, отримуваних за допомогою мікрофонних решіток, у цифровий потік даних в розглядуваній аудіолокаційної системи доцільно використати зовнішню звукову карту. Кожен з типів відеокарт розрахований на задоволення потреб певного «типу» взаємодії. Оскільки аудіоінтерфейси мають так багато функцій, важко сказати, які з них важливі, а які ні. При виборі високоякісної зовнішньої звукової карти слід враховувати 5 основних характеристик:

— сумісність DAW(digital audio workstation);

— інтерфейсні роз'єми;

— кількість вводу / виводу (input/output);

— типи вхідних каналів;

— формовий фактор.

Розглянемо ці характеристики більш детально.

Загалом більшість DAW працюватимуть з більшістю інтерфейсів, але не завжди. Якщо у вас ще немає певної DAW, не хвилюйтеся. Тому що 90% найкращих DAW сумісні з кожним вибраним інтерфейсом. Однак якщо у вас уже є DAW, який ви хочете продовжувати використовувати, перевірте сумісність веб-сайту компанії. Іноді цю інформацію часто важко знайти.

Інтерфейсні підключення. При підключенні аудіоінтерфейсу до комп’ютера зазвичай є чотири варіанти кабелю, показані на рис. 3.11: USB зазвичай спостерігається на більш дешевих інтерфейсах домашньої студії та пропонує найповільнішу швидкість передачі даних.

— Firewire використовується на дорожчих інтерфейсах домашньої студії та пропонує набагато вищу швидкість передачі даних (сьогодні вони стають все рідкішими).

— Thunderbolt останнім часом став популярним завдяки останнім напівпрофесійним інтерфейсам і набагато швидший, ніж USB або Firewire.

— PCIE вже давно є стандартним підключенням для професійних інтерфейсів, оскільки він пропонує додаткову обчислювальну потужність і надзвичайно швидку передачу даних.

Рисунок 3.11 — Типи з’єднань.

Кількість входів/виходів (вхід/вихід). У типовому інтерфейсі кількість вводу-виводу може змінюватися від 1-2 до понад і в основному залежить від кількості потоків, які можна записати або відтворити одночасно:

Підраховуючи вхідні канали аудіоінтерфейсу, виробники можуть використовувати будь-яку кількість різних типів входу. Проте майже у всіх випадках він містить комбінацію трьох основних, які показані на рис. 3.12:

— мікрофонний вхід дозволяє підключити мікрофон безпосередньо до інтерфейсу;

— лінійний вхід вимагає додавання підсилювача верхнього мікрофона

для використання в якості мікрофонного каналу;

— оптичний вхід – це тип «цифрового» входу, який вимагає додавання підсилювача верхнього мікрофона, а в якості мікрофонного каналу використовується цифровий перетворювач без оптичного виходу.

Якщо ви хочете використовувати звуковий інтерфейс, не додаючи багатоканальний мікрофонний підсилювач, у вас може бути менше доступних входів.

Рисунок 3.12 — Типи входів(з ліва на право: мікрофонний, лінійний, оптичний)

Інтерфейси часто мають 16 або більше загальних вхідних каналів, але лише 2-8 для підключення мікрофонів. Таким чином, справжня кількість входів в аудіоінтерфейсі без додаткової передачі – це кількість попередніх мікрофонних підсилювачів, а не кількість входів.

Форм-фактор — це фізичний розмір і форма інтерфейсу, які мають два варіанти (рис. 3.13):

— інтерфейси для робочого столу – менші та поміщаються на столі поруч із комп’ютером;

— інтерфейси, що встановлюються в стійку, більші і встановлюються в стандартну стійку.

Рисунок 3.13 — Типи інтерфейсів

Для розглядуваної системи найбільш підходить перший варіант інтерфейсу, оскільки він дешевий, простий у використанні і не вимагає спеціальної збірки чи периферійних пристроїв.

Для реалізації багатоканального введення аналогових сигналів у розроблюваній системі вибираємо зовнішню звукову карту Behringer U-PHORIA UMC404HD (рис 3.14) цифрові дані з якої можна отримати через USB інтерфейс [41]. Даний вибір обумовлений тим, що Behringer U-PHORIA UMC404HD має 2 входи для мікрофонів XLR, дозволяє отримати високу швидкість перетворенням звуку у цифровий та підтримує фантомне живлення.

Рисунок 3.14 — Зовнішня звукова карта

Behringer U-PHORIA UMC404HD є професійним аудіоінтерфейсом USB з високоякісними компонентами, великою кількістю вхідних і вихідних з'єднань та 24 бітними 192 кГц. АЦП /ЦАП. Behringer U-PHORIA UMC404HD має чотири комбінованих входи на передній панелі (мікрофонний / інструментальний / лінійний). Кожен з входів оснащений мікрофонним підсилювачем Midas і фантомним живленням 48 В для конденсаторних мікрофонів. Кожен вхід має окремі регулятори посилення, перемикачі типу, а також індикатори сигналу та відсікання. Загальні елементи керування на правій стороні аудіоінтерфейсу представлені MIX (вихідний сигнал з обробленим мікшуванням), MAIN OUT (встановлення основного виходу) і керуванням виходом для навушників. У цьому сегменті також є дві кнопки STEREO / MONO і MONITOR A / B: остання перемикає відразу між двома

каналами прямого моніторингу (які підключені до роз'ємів PLAYBACK OUTPUTS на задній панелі). Моніторинг відбувається без затримок.

На задній стороні звукової карти є 4 аналогових виходи на cinch (несбалансований) і 1/4 "TRS jack (балансний) на додаток до основних виходів на тому ж гнізда 1/4" TRS в поєднанні з двома виходами XLR-M Плюс є 4 вставки для підключення зовнішніх пристроїв обробки сигналів (наприклад, процесорів ефектів). На задній панелі ви також знайдете MIDI-вхід і вихід для підключення зовнішніх MIDI-контролерів і модулів. Підключення до комп'ютера здійснюється за допомогою USB-кабелю з роз'ємом типу B. Світлодіод на передній панелі вказує, що USB-з'єднання існує. Підтримує ПК та Mac, попередня інсталяція драйвера не потрібна [42].

2.2 Розробка структурної схеми системи пасивної аудіолокації

Перед тим як розробляти саму схему потрібно знайти всі недоліки такої системи, та вже на їх основі роботи робочий варінт системи пасиввної аудіолокації.

Основна перевага роботи – несучість великих площ, що в свою чергу мінімізує витрати на інші засоби захисту. Нам знадобиться лише дві мікрофонні сітки на квадратний кілометр, а це означає, що зі збільшенням розміру кількість акустичних станцій подвоюється. При цьому його використання спричиняє високі витрати та руйнує економічну складову всього проекту.

Основним недоліком є фоновий шум, який заважає надійній звуковій обробці місцевості. В акустиці шум — це випадкове коливання, яке характеризується зміною частоти та амплітуди.

Акустичний шум — це випадковий процес, викликаний рухом, коливаннями внутрішніх структур пружного середовища під впливом зовнішніх і внутрішніх фізичних, хімічних, біологічних факторів. Ці хвилі можуть нести корисну інформацію, тоді їх називають шумовим сигналом.

Наприклад, шум вітру, шум літаків, вібрації від машин і транспортних засобів, навколишній шум і природні явища можуть істотно вплинути на роботу перевіреного алгоритму в ідеальних умовах. Шуми, які не містять корисної інформації та погіршують вимірювальні властивості локаційного пристрою, класифікуються як перешкоди. За походженням акустичний шум поділяється на пасивний і активний.

Пасивний шум викликається впливом акустичного сигналу на пружне середовище — це відомо як реверберація. Пасивний шум зумовлений розсіюванням акустичних хвиль на неоднорідностях середовища, поперечний розмір якої становить більше 1/100 довжини хвилі, а також відбиттям акустичних хвиль від шарів і кордонів сусідніх середовищ. Це перехідні процеси, параметри яких змінюються з часом. Запровадження регулювання посилення в часі зміщує процес з нестійкого до стаціонарного.

Враховуючи випадковість шуму, спотворення параметрів сигналу в акустичному комплексі найбільш придатне для опису та визначення властивостей випадкового процесу математичним апаратом теорії ймовірностей. Часто відомі деякі параметри акустичного сигналу (тиск, швидкість вібрації), на основі яких можна оцінити статистичні властивості сигналу. Такий підхід може дати хороший результат при дотриманні умов ергодичності. Відомо, що випадковий стаціонарний процес є ергодічним, якщо всі його статистичні властивості можна знайти на основі реалізації цього процесу достатньо великої тривалості.

Проаналізувавши основні недоліки таких систем, було прийнято створити систему пасивної аудіолокації на основі мікрофоних масивів, які підєднані до зовнішньої звукової плати, для покращення обробки сигналу та його посилення.

Після приймання сигналу, наступною задачею є обрахунок через програмну частину у одноплатному компютері. Потім всі обрахунку надходять в ноутбук, та виводяться оброблені дані. А саме, у вигляді діаграми

направленності у вікні браузера. Структурну схему можна побачити на рисунку 3.15, та наведено у додатку.

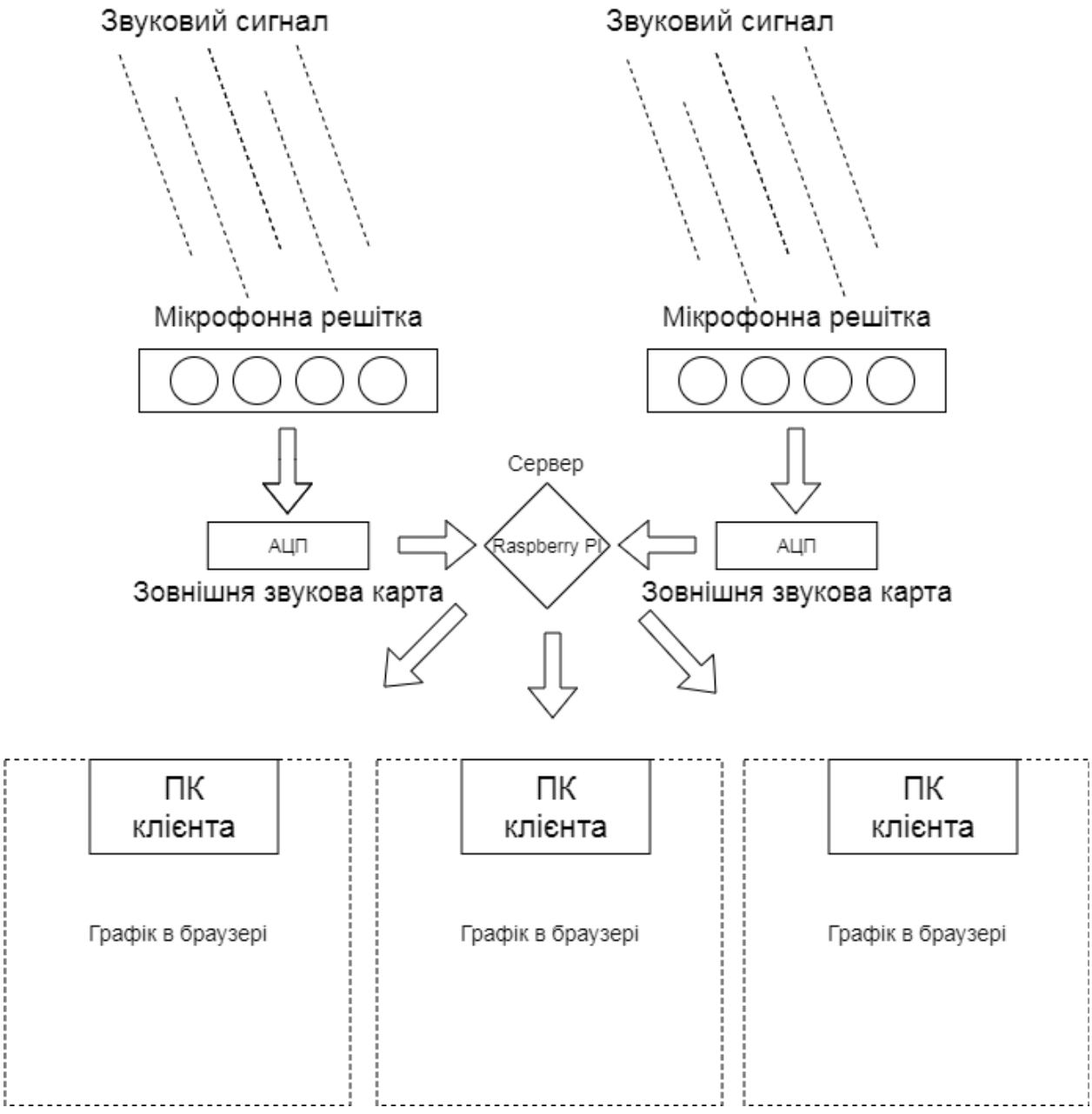

Рисунок 3.15 — Загальна структурна схема

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз доступних засобів розробки програмного забезпечення

При створенні надійного програмного забезпечення, головним питанням для програміста або звичайного розробника є вибір мови програмування. В даному підрозділі розглянемо програмного забезпечення, яке буде вибране для створення унікального пакету прикладних програм для досягнення мети магістерської роботи.

Існує багато типів мов програмування. На рис. 4.1 наведено класифікацію типів мов програмування.

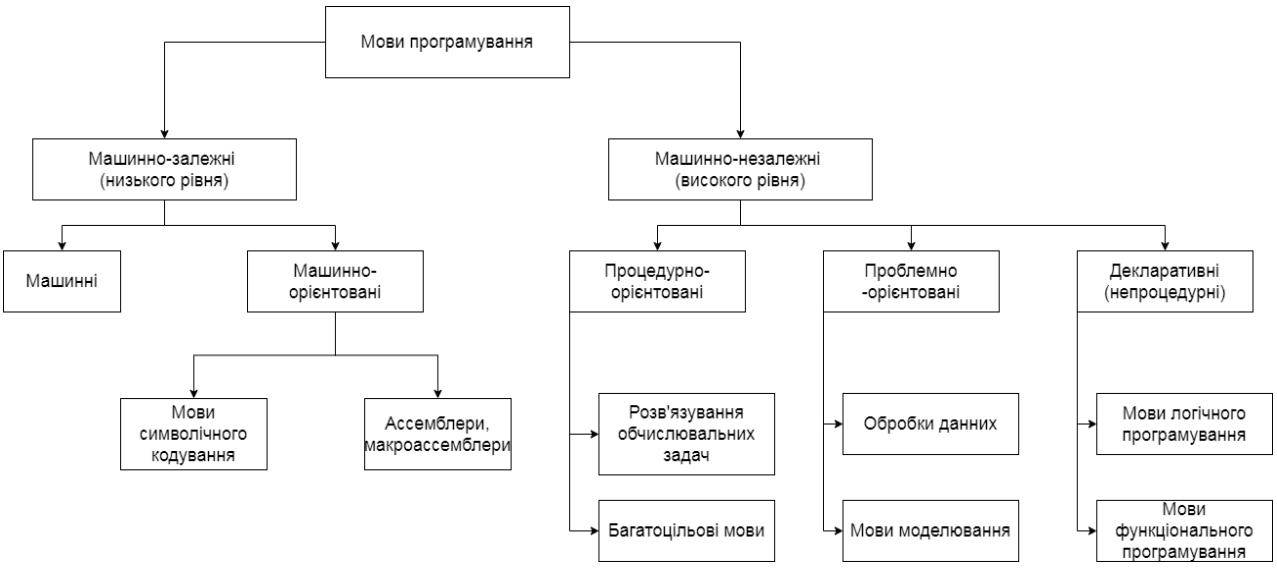

Рисунок 4.1–Типи мов програмування

Різні мови дозволяють використовувати різні стилі програмування. Ось декілька різних типів парадигм мови кодування:

— процедурне програмування;

— функціональне програмування;

— об’єктно-орієнтовне програмування.

Мови процедурного програмування використовуються давно і більш зрозумілі, ніж інші типи мов. Тому вони користуються високою популярністю серед розробників. Основний підхід при роботі з процедурними мовами заснований на розбитті програми на окремі процедури, які, у свою чергу,

також можуть бути багатократно поділені на більш дрібні підпроцедури доки не будуть найменші елементи керування.

Основними перевагами процедурного програмування є простота у використанні, низькі вимоги до пам'яті, проста структура, що полегшує поєднання з інтерпретаторами та компіляторами. Однак процедурні мови не характеризуються низькою захищеністю даних. Крім того, процес пошуку та виправлення помилок є важким та трудомістким, оскільки вимагає порядкового аналізу всієї програми [43].

Основним аспектом процедурної мови є спосіб роботи з вашими пристроями. Використовуючи процедурну мову програмування, ви даєте конкретні вказівки безпосередньо своєму комп'ютеру та повідомляєте, як досягти своїх цілей за допомогою логіки та покрокових процесів. Це тактика, яка підходить до завдань з точки зору зверху вниз. Процедурні мови розглядають дані як окремі від процедур, і це впливає на те, як розробники використовують їх. Приклади процедурних мов програмування включають C, Java та Pascal [43].

На додачу до об'єктно-орієнтованого програмування, функціональне програмування є однією з кількох фундаментальних парадигм програмування. Коротше кажучи, функціональне програмування зосереджується на чисто математичних функціях і незмінних даних, тобто даних, які неможливо змінити після їх створення. У ньому немає стану, а це означає, що змінюється лише введення даних у функціональній програмі.

Оскільки у функціональному програмуванні, на відміну від процедурного програмування, немає об'єктів, які можна замінити тими ж об'єктами, можна концептуально змінити порядок коду і залишити той самий вихід. Це як помножити вісім чисел разом, як би ви це не зробили, ви все одно отримаєте той самий результат.

У цьому сенсі функціональне програмування відноситься до потоку. Вхід збільшується, а результати падають. Це на відміну від об'єктно-орієнтованого

програмування, яке більше є зміною стану машини з унікальними та змінними об’єктами.

Функціональне програмування було трохи довшим, ніж об’єктно-орієнтоване програмування, яке дуже далеко від машини Тьюринга. За останні кілька поколінь він зменшився, але останнім часом він досить значний повернувся в JavaScript, який є парадигмальним, але вважається більш функціональним, ніж об’єктно-орієнтована мова[43].

Він заснований на кількох видатних принципах, які також є основою всіх мов, написаних за допомогою цієї методології.

Він заснований на кількох видатних принципах, які також є основою всіх мов, написаних за допомогою цієї методології.

Основний принцип – незмінність. Це той випадок, коли ви встановлюєте значення для змінної або функції, але воно не змінюється. Це усуне побічні ефекти або несподівані результати, оскільки програма не залежить від захворювання. Таким чином, функції завжди працюють однаково щоразу, коли вони виконуються.

Функціональне програмування має дисциплінований стан, в якому деякі стани можуть змінюватися, щоб створити нову цінність, але це процес, який глибоко контролюється. Функціональне програмування намагається уникнути умов і змін рамок. Коли статус жорстко контролюється, його легше масштабувати та налагоджувати, і ви отримаєте менше несподіваних результатів, коли змінна несподівано змінюється.

Одним із принципів є прозорість посилання, вона виникає в результаті поєднання чистих функцій і незмінності. Оскільки наші функції чисті та передбачувані, ми можемо використовувати їх для заміни змінних, зменшуючи кількість завдань, які ми визначаємо. Якщо результат функції дорівнює змінній, оскільки наші результати передбачувані, ми можемо просто замінити змінну цією функцією.

Функціональне програмування приділяє велику увагу певним функціям, першокласним функціям. Тому він має намір передавати цілі функції одна

одній так само легко, як і інші мови зі змінними. У функціональному програмуванні ці функції можна розглядати як значення або дані [43].

Основною перевагою функціонального програмування є гарна читаність. Оскільки все доведено до функціонального рівня, код легко читається, бо функції прості та зрозумілі. Функції не можуть змінюватися несподівано, а тому більшість даних не можуть бути змінені помилково. Завдяки усьому цьому спрощується налагодження. Крім того,функціональне програмування забезпечує високу стабільність, оскільки дозволяє строго контролювати код, уникаючи несподіваних помилок та непередбачуваних ситуацій.

Проста структура функціонального програмування дуже стабільна, легко продовжувати будувати і покладатися на безпеку свого фундаменту. Також легше досягти високого рівня абстракції за допомогою функцій, які дозволяють безпосередньо розпоряджатися більшою частиною рутинного коду як ітераційні програми, створюючи коротший і стабільніший код.

Проте функціонального програмування є доволі надмірним. Багаті можливості функціонального програмування сприяють створенню великих ланцюгів абстракцій, що вимагає перегляду значної множини функцій для розуміння того, що робить конкретна функція.

Функціональне програмування є більш академічним і суворим, ніж об'єктно-орієнтоване програмування. Це також вимагає кращого розуміння алгебри, лямбда-числювань і теорії категорій, на які воно в значній мірі покладається. Існують менш доступні навчальні матеріали з функціонального програмування, і для цього потрібна інша логіка, ніж те, до якої ми зазвичай звикли.

Наведених вище принципів важко дотримуватися, особливо коли зміна стану значно спрощує програмування. Функціональне програмування часто займає довгий шлях, тому написати окрему функцію легко, а повноцінну програму створити важко.

Використання кількох фіксованих значень означає, що потрібно більше пам’яті або більше продуктивності, оскільки зміна значення вимагає

створення нової інформації або виконання функції для отримання результату кожного разу, коли вона викликається.

Функціональне програмування набагато старше об’єктно-орієнтованого програмування і набуло популярності в останні роки. Це повернення зазвичай пояснюється такими мовами:

JavaScript - хоча і не є чисто функціональною мовою, вона має багато функціональних функцій програмування. Через це програмісти частіше, ніж раніше, використовують функціональне програмування в JavaScript, оскільки воно може бути кориснішим і стабільнішим у певних сценаріях для об’єктно-орієнтованих мов.

Clojure не є загальноприйнятою назвою для JavaScript, але це надійна, функціональна мова програмування, побудована на основі Lsip, яка існує з 1950-х років. Це означає, що він працює до кореня. Він працює на платформі Java і компілюється в байт-код JVM.

Інша функціональна мова, Haskell, призначена для вирішення реальних проблем, а не академічних. Він не такий старий, як Лісп; він був створений у 90-х роках. Він використовується для кількох поширених проектів, таких як віконний менеджер Xmonad.

Об’єктно-орієнтоване програмування базується на об’єктах, структурах даних, які містять як дані (властивості чи атрибути), так і код (процедури чи методи). Об’єкти можуть змінюватися самі, і в більшості мов ООП майже все є об’єктом, який може бути як значущим, так і виконуваним кодом. Кожен об’єкт унікальний, і хоча він може бути копією іншого об’єкта, його змінні можуть відрізнятися від змінних будь-якого іншого об’єкта [44].

Об'єкти в об'єктно-орієнтованому програмному розглядаються як фактичні об'єкти, що мають певні властивості та може виконувати певні функції, впливаючи на себе та інші об′єкти.

Об’єктно-орієнтовані мови мають чотири принципи, які визначають їх і роблять їх набагато ефективнішими у використанні. Деякі люди називають їх чотирма стовпами об’єктно-орієнтованого програмування.

Інкапсуляція — це концепція зв'язування даних з функціями, яка служить функцією безпеки для зберігання цих даних. Наприклад, багато мов ООП не дозволяють нікому, крім певних методів класу, які мають дані, отримати доступ до цих даних. це часто робиться безпосередньо для зберігання певних даних. Він також працює на функціональному рівні, оскільки не дозволяє коду за межами об’єкта втручатися в код і дані, які мають бути в об’єкті, і навпаки [44].

Абстрагування — це ідея, що якщо ви робите щось занадто часто, це має бути його власною суттю. Коли програміст переписує ту саму функцію для різних об’єктів, цю функцію можна абстрагувати, щоб стати власною сутністю і помістити на будь-який об’єкт, який цього потребує.

Наслідування — це принцип, який дозволяє певним класам перебувати під егідою інших класів і успадковувати інформацію та функціональні можливості цього класу, а також розширювати цю функціональність. Це дозволяє нам повторно використовувати код, який ми вже ввели в інших класах [44].

Поліморфізм — це принцип, коли щось може приймати більше ніж одну форму і, у розумінні мов ООП, відноситься до здатності обробляти об'єкти по-різному залежно від їх природи. Це дозволяє нам визначити різні методи обробки об’єктів на основі їх похідного класу.

Об’єктно-орієнтований код надзвичайно модульний, а поліморфізм і абстракція дозволяють створити функцію, яку можна використовувати знову і знову, або для імітації інформації та функцій, які вже були написані. Це економить час, зменшує складність, економить місце, полегшує кодування та розвантажує наші пальці.

Після чотирьох частин, коли основні класи визначені, є достатньо основ, щоб частини програми можна було розробляти та функціонувати окремо одна від одної. Це значно полегшує метод паралельної розробки для великих команд розробників.

Оскільки весь од або його значна частина знаходиться в одному місці, його можна повторно використати. Замість того, щоб проходити через сотню різних випадків, коли функція викликається і виправляється по одній, ми можемо виправити модульну та поліморфну функцію, яка викликається сто разів.

Хоча більшість мов мають певний захист, об’єктно-орієнтовані мови стають у нагоді, оскільки безпека вбудована в інкапсуляцію. За замовчуванням інші методи та класи не мають доступу до приватних даних, а програми, написані мовами ООП, є більш безпечними.

При об'єктно-орієнтованому програмуванні здійснюється багато налаштувань, відповідно до чого програма може функціонувати по різному. Існує багато методологій програмування, які не працюють з іншими методологіями, або є неефективними, або неможливими для використання.

Оскільки об'єктно-орієнтовані мови програмування є модульними та масштабованими, отримання потрібного кінцевого результату можливе лише за умови чітко зробленого дизайну, вимагає чіткого плану. Потрібне чітке розуміння того, що робити для отримання гарно масштабованої, ефективної та потужної програми відповідно до усіх тих можливостей, що надає об'єктно–орієнтована мова.

Оскільки об’єкти та функції можуть діяти незалежно, поглинати інформацію та (переважно) повертати достовірні результати, вони можуть бути заплутаною областю, де не завжди очевидно, що вони роблять. Хоча програміст, ймовірно, створив цей об’єкт і знає, що він робить, мови ООП просто не такі прозорі, як інші мови. У функціональних мовах їм представлений весь код, а не розгорнутий і прихований, як в об'єктно-орієнтованих мовах.

Об’єктно-орієнтовані мови часто переконливі завдяки своїй ефективності. Програми, створені мовами ООП, часто більші та вимагають більше обчислювальних зусиль, ніж функціональні мови. C ++ — це мова ООП, але це одна з найшвидших доступних мов. З тієї ж причини швидкість

не завжди важлива, і різниця в швидкості стає очевидною лише при обробці великих чи складних обчислень або у випадках, коли потрібна надзвичайна швидкість. В іншому випадку більшість мов програмування, очевидно, працюють з однаковою швидкістю залежно від їх функції.

Тепер, коли ми трохи зрозуміли, як працюють об’єктно-орієнтовані мови та де вони використовуються, давайте подивимося на деякі з найпопулярніших об’єктно-орієнтованих мов програмування.

Java є всюди і є однією з найпоширеніших мов усіх часів. Девіз Java — «Напишіть один раз, запускайте де завгодно», і це відображається в кількості платформ, на яких вона працює, і де вона використовується.

Python є мовою загального призначення і використовується в багатьох місцях. Однак Python в значній мірі покладається на машинне навчання та науку про дані і є однією з найкращих мов для цієї нової і постійно зростаючої галузі.

C ++ має швидкість C з функціональністю класу та об'єктно-орієнтованою парадигмою. Це складна, швидка, надійна та потужна мова, яка використовується в багатьох програмах і навіть використовується для створення компіляторів та інтерпретаторів для інших мов.

Для написання програми звукового модуля були проаналізовані технічні характеристики комп’ютера, на якому він створений, тому вибір припав на мову C++ OOP, описану вище.

JavaScript був обраний для написання програми графічного движка після аналізу запропонованого вибору зі списку функціональних мов, переваги та недоліки яких були описані вище.

4.2 Вибір бібліотек

Стандартна бібліотека шаблонів C ++ (STL) — це загальна бібліотека на основі шаблонів, яка пропонує узагальнені контейнери. Замість того, щоб мати справу з динамічними масивами, зв’язаними списками, бінарними деревами або хеш-таблицями, програмісти можуть просто використовувати алгоритм,

наданий STL.

STL структурований відповідно до контейнерів, ітераторів та алгоритмів та їхніх ролей:

— контейнери – їх основна роль полягає в управлінні колекцією об’єктів певного типу, наприклад, масивами цілих чисел або пов’язаними списками рядків;

— ітератори – основне завдання – ітерація по елементу колекцій;

— стандартні алгоритми - основне завдання - обробити елемент колекції.

Щоб написати надійне програмне забезпечення для обчислення великих обсягів інформації, було б розумно використовувати нестандартну бібліотеку, таку як Boost.

Boost надає безкоштовні, рецензовані портативні вихідні бібліотеки C ++. Вони добре працюють зі стандартною бібліотекою C++. Бібліотеки Boost розроблені для широкого використання та використання в різних програмах. Ліцензія Boost сприяє використанню бібліотек Boost всіма користувачами з мінімальними обмеженнями.

Компанія, яка продає бібліотеку Boost, гарантує, що всі функції підходять для можливої стандартизації. Починаючи з десяти бібліотек Boost, включених до Бібліотечного технічного звіту (TR1), і продовжуючи випуск кожного видання стандарту ISO C ++ з 2011 року, Комітет зі стандартів C ++ продовжує покладатися на Boost як на цінне джерело для доповнень до C. + Бібліотечний стандарт + [45].

Boost та ширша спільнота Boost підтримують дослідження та навчання, щоб отримати максимальну віддачу від C ++ та бібліотек, розроблених для нього, включаючи, але не обмежуючись, ті, що входять до бібліотеки Boost.

Організація та спільнота підтримують списки розсилки та чати, щоб дізнатися про найкращі методи та передові технології для бібліотек Boost та користувачів C ++ загалом.

З 2006 року в Аспені, штат Колорадо, проводиться закрита щорічна

конференція з підвищення рівня C++ Now. Конференція – це можливість навчання, яка зосереджена на передовому C ++. З 2007 року Boost є учасником щорічного Google Summer of Code, де студенти розвивають свої навички під час розробки бібліотеки Boost.

Boost працює майже на кожній сучасній операційній системі, включаючи UNIX і Windows. Популярні дистрибутиви Linux і Unix, такі як Fedora, Debian і NetBSD, містять попередньо вбудовані пакети розширення.

Використання бібліотек продуктивності підвищує продуктивність програми. Крім того, застосування такого підходу дозволяє отримати такі переваги:

— відкритий вихідний код, що дозволяє лего модифікувати його за потреби;

— наявнсть ліцензії дозволяє нам розробляти проекти як з відкритим, так і з закритим кодом; отримується можливість отримати прибуток від продажу програмногозабезпечення;

— гарна за документованість, що полегшує пошук усієї потрібної інформації та прикладів програмного коду,що доступні на офіційному веб-сайті;

— підтримка майже всіх сучасні операційні системи, таких як Windows та Linux;

— доповнення до STL замість заміни, використання підвищувальних бібліотек полегшить ті процеси програмування, які ще не пов’язані з STL.

PortAudio було обрано з багатьох бібліотек, які можна використовувати для взаємодії з аудіо. З його використанням створено безліч додатків для взаємодії з будь-якими сигналами і звуками.

PortAudio — це безкоштовна міжплатформна бібліотека з відкритим доступом для введення та виведення аудіо. Він дозволяє писати прості аудіопрограми на «C» або C ++, які можна компілювати та виконувати на багатьох платформах, включаючи Windows, Macintosh OS X та Unix (OSS / ALSA). Він призначений для полегшення обміну аудіо програмним

забезпеченням між розробниками на різних платформах.

Багато програм використовують PortAudio для аудіо вводу-виводу [46]:

— Audacity - безкоштовний загальнодоступний аудіоредактор;

— MoreAmp - аудіоплеєр, транскодер і рипер компакт-дисків;

— Oreka – відкрита, кросплатформна система запису та пошуку аудіо;

— Pyo - модульний API синтезу для Python;

— WireShark - аналізатор мережевих протоколів використовує PortAudio для відтворення RTP-потоків;

— Zep — експериментальна контрольна програма для експериментальної психології.

Якщо ви використовуєте JavaScript як функціональну мову в цьому документі, вам потрібно буде проаналізувати бібліотеки, необхідні для написання графічного модуля.

За останні роки React.js став популярною базою для розробки ефективних веб-додатків.

React.js, спочатку розроблений Facebook, є бібліотекою JavaScript для створення інтерфейсів користувача. React полегшує розробникам, щоб вони виглядали естетично та інтерактивно для користувачів їхніх веб-сайтів [47].

Коли ви кодуєте в JavaScript, ви знаєте, що для того, щоб додати функцію на веб-сайт, вам потрібно вручну написати код. Хоча ви можете створювати менші функції для свого сайту, створюючи більше функцій, ваш код може повторюватися, і ви можете витратити багато часу на написання коду, який ви, можливо, вже написали в іншому місці свого проекту.

Бібліотеку JavaScript, як-от React.js, можна знайти тут, щоб зробити ваш процес кодування більш ефективним, посилаючись на готові фрагменти коду, які можна повторно використовувати у своєму коді.

Оскільки популярність React продовжує зростати, доступні ресурси, що спрощує розпочати роботу. Насправді, React відомий своїми дослідженнями, які показують хорошу ефективність.

Щоб побудувати основну частину веб-сторінки, а саме таблицю, рекомендується використовувати найпотужнішу і незалежну бібліотеку під назвою Canvas.js [48]. Все, що вам потрібно знати, щоб створити графік за допомогою цієї бібліотеки, — це основи JavaScript. і HTML-код.

CanvasJS — це проста у використанні бібліотека діаграм HTML5 і Javascript. Він працює на таких пристроях, як iPhone, iPad, Android, Windows Phone, Microsoft Surface, настільні комп'ютери тощо. Це дозволяє створювати розширені інформаційні панелі, які запускаються на різноманітних пристроях без шкоди для обслуговування або функціональності.

Ми рекомендуємо використовувати цю бібліотеку, оскільки вона проста в освоєнні та має інтуїтивно зрозумілий інтерфейс користувача для взаємодії з розробником. Вона також є високопродуктивною та кросплатформною, а головною перевагою цієї бібліотеки є її незалежність від усіх інших похідних.

4.3 Визначення основних функцій звукового модуля

Після запуску програми перевіряються аргументи командного рядка, а саме IP-адреса порту та кількість потоків, які буде використовуватися програмою, показано на рис. 4.2.

```
// Check command line arguments.
if (argc != 4)
{
    std::cerr <<
    "Usage: SoundDirectionServer <address> <port> <threads>\n" <<
    "Example:\n" <<
    "    SoundDirectionServer 0.0.0.0 8080 1\n";
    return EXIT_FAILURE;
}
auto const address = net::ip::make_address(argv[1]);
auto const port = static_cast<unsigned short>(std::atoi(argv[2]));
auto const threads = std::max<int>(1, std::atoi(argv[3]));
```

Рисунок 4.2 — Процес перевірки

Наступним кроком відбувається ініціалізація потужної бібліотеки PortAudio та звукового приладу встановленого в системі за замовчуванням відображено на рис. 4.3.

```
/* PortAudio start */

PaError error = Pa_Initialize();
if(error != paNoError) {
    std::cerr << Pa_GetErrorText(error) << std::endl;
    Pa_Terminate();
} else std::cout << "Pa_Initialize\n";

/* Input device parameners */
inputParameters.device = Pa_GetDefaultInputDevice();
if(inputParameters.device == paNoDevice) {
    Pa_Terminate();
    std::cerr << "No input devices.\n";
    return -1;
}

deviceInfo = Pa_GetDeviceInfo(Pa_GetDefaultInputDevice());

inputParameters.channelCount = deviceInfo->maxInputChannels;
inputParameters.sampleFormat = paFloat32;
inputParameters.suggestedLatency = deviceInfo->defaultLowInputLatency;
inputParameters.hostApiSpecificStreamInfo = nullptr;

std::cout << "Device is " <<deviceInfo->name <<std::endl;
```

Рисунок 4.3 — Опис функції

Далі ініціалізується потік аудіозапису раніше ініціалізованого пристрою. Він використовується для запису звукової інформації, яка надходить до мікрофонів аудіосітки. За допомогою цього потоку кожні 100 мс викликається функція recordCallback, яка приймає буфер з аудіоданими у своїх параметрах, який потім використовується для зберігання, обробки та надсилання даних для відображення шаблону клієнту. Показано на рис. 4.4. Після ініціалізації аудіопотоку сокети сервера запускаються за IP-адресою та портом, зазначеними користувачем. Для кожного нового клієнта на цьому сервері створюється окрема сесія, яка описана в програмі з класом Session.

Кожен раз, коли викликається функція recordCallback, перевіряється кількість фрагментів буфера, записаних нею раніше. Коли цих фрагментів достатньо для опису 1-секундного аудіосигналу, дані перетворюються та надсилаються всім підключеним клієнтам.

```
if(stream == nullptr || !Pa_IsStreamActive(stream)) {
    // Opening stream
    data.maxFrameIndex = deviceInfo->defaultSampleRate; /* Record for a few seconds. */
    data.frameIndex = 0;
    data.channels = inputParameters.channelCount;

    data.dataLeft.resize(data.maxFrameIndex);
    data.dataRight.resize(data.maxFrameIndex);

    error = Pa_OpenStream(&stream, &inputParameters, nullptr,
                          deviceInfo->defaultSampleRate,
                          512,
                          paClipOff, // we won't output out of range samples so don't bother clipping them
                          recordCallback,
                          &data);
    if(error != paNoError) {
        Pa_Terminate();
        std::cerr << Pa_GetErrorText(error) << std::endl;
        return -1;
    }
    error = Pa_StartStream( stream );
    if(error != paNoError) {
        Pa_Terminate();
        std::cerr << Pa_GetErrorText(error) << std::endl;
        return -1;
    }

    std::cout << "Recording...\n";
}
```

Рисунок 4.4 — Ініціалізації потоку запису

Основною функцією програми звукового модуля є write_data класу сесії. Він викликається, коли буфер заповнюється однією секундою аудіоінформації для перетворення даних і подальшої відправки кожному клієнту.

Перетворення даних здійснюється методом часткового зсуву двох каналів стереосигналу відносно один одного. Щоб пришвидшити обчислення та спростити обробку інформації, було вирішено зсунути канали на 44 фрагмента кожен.

Отримані дані для шаблону можна розділити на три частини. Формування даних для лівої частини графіка шаблону показано на рисунку 4.5. Для цього один канал зміщується вліво відносно іншого.

Максимальна точка графіка шаблону розраховується шляхом розрахунку середньоквадратичного значення двох каналів без зміщення, як показано на рис. 4.6.

Формування даних для правої частини графіка шаблону показано на рис. 4.7. Для цього один канал зміщується вправо відносно іншого.

```
// Left
for(int t = 0; t < ticks; t++) {
    std::vector<float> left;
    for (auto i = t * N; i < data->dataLeft.size(); ++i) {
        left.push_back(data->dataLeft.at(i));
    }
    std::vector<float> right;
    for (auto i = 0; i < data->dataRight.size() - t * N; ++i) {
        left[i] += data->dataRight.at(i);
    }

    value = 0.0;
    for (auto i = 0; i < left.size(); ++i) {
        value += std::pow(left.at(i) * N, 2.0);
    }

    value /= left.size();
    value = std::sqrt(value);

    d.push_back(value);

    left.clear();
    right.clear();
}
```

Рисунок 4.5— Зсув лівого каналу

```
// Center
value = 0.0;
for (auto i = 0; i < std::min(data->dataLeft.size(), data->dataRight.size()); ++i) {
    value += std::pow((data->dataLeft.at(i) + data->dataRight.at(i)) * N, 2.0); // TODO: Среднеквадротическое
}
```

Рисунок 4.6 — Обрахунок середньо квадратичного значення

```
// Right
for(int t = 0; t < ticks; t++) {
    std::vector<float> left;
    for (auto i = 0; i < data->dataLeft.size() - t * N; ++i) {
        left.push_back(data->dataLeft.at(i));
    }
    std::vector<float> right(data->maxFrameIndex - t * 100);
    for (auto i = t * N; i < data->dataRight.size(); ++i) {
        left[i - t * N] += data->dataRight.at(i);
    }

    value = 0.0;
    for (auto i = 0; i < left.size(); ++i) {
        value += std::pow(left.at(i) * N, 2.0);
    }

    value /= left.size();
    value = std::sqrt(value);

    d.push_back(value);

    left.clear();
    right.clear();
}
```

Рисунок 4.7 — Зсув правого каналу

Оброблені дані надсилаються кожному клієнту у форматі JSON. Цей формат найбільш зручний під час обміну інформацією з клієнтом сокету JavaScript.

При виконанні всіх функцій програма в браузері виводить діаграму направленості, наведена в додатках.

4.4 Визначення основних функцій графічного модуля

Після ініціалізації програми відкривається вікно браузера з ділянкою для вводу IP адреси сервера та ділянка для представленням графіка зображеного на рис. 4.8.

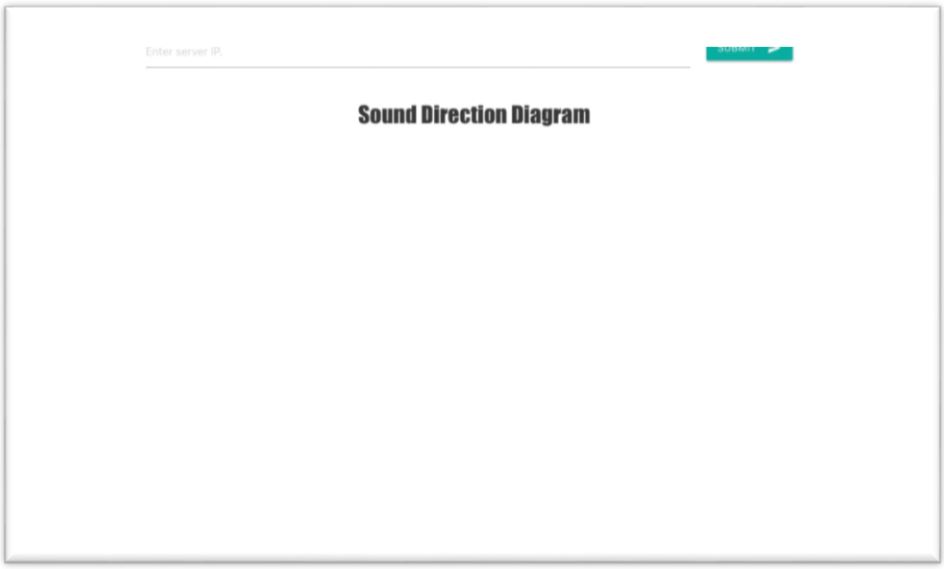

Рисунок 4.8— Вікно браузера

Після введення IP-адреси та натискання кнопки Submit функція handleIPchange видаляється, яка перевіряє правильність IP-адреси за допомогою регулярного виразу у функції ValidateIPaddress. Функція обробника кнопки підтвердження показана на рис. 4.9.

Після підтвердження правильності IP-адреси вона передається як параметр до компонента діаграми. Після зміни параметра, який відповідає за зміну сервера, виконується ініціалізація та підключення сокетів.

Після ініціалізації функція обробника подій налаштована на отримання повідомлення від звукового сервера. Надалі отримані дані будуть конвертовані у відповідний формат для коректного відображення розкладу відповідно до вимог бібліотеки CanvasJSChart.

Тоді засоби цієї бібліотеки показують візерунок у вигляді стовпчастої діаграми на рис. 4.10. Надалі перетворення даних і оновлення графіки будуть повторюватися з кожним новим повідомленням, отриманим аудіосервером.

```
function handleIPChange(e) {
    e.persist();
    if(ip == null) {
        const val = input.current.value;
        if(ValidateIPaddress(val)) {
            setIP(val);
        } else {
            alert("Plese, enter a valid IP address.");
        }
    } else {
        setIP(null);
    }
}
```

Рисунок 4.9 — Опис функції

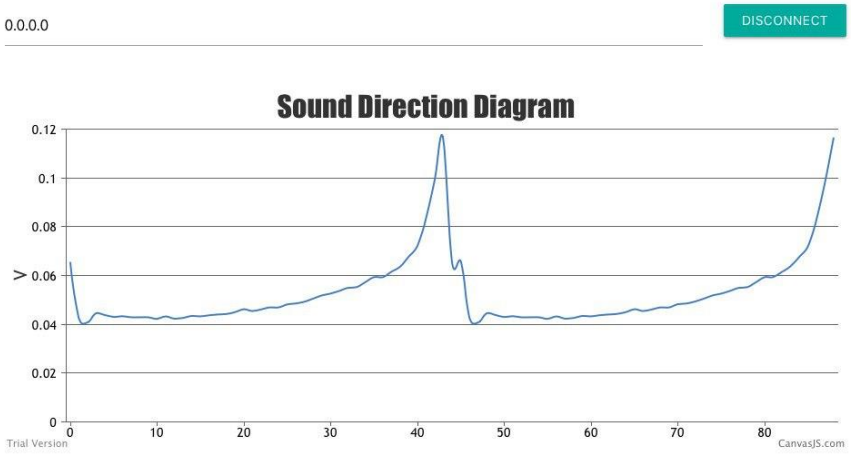

Рисунок 4.10— Діаграма направленості

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота на тему «Методи та апаратно-програмні засоби визначення напрямів звукових сигналів» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

— проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

— розраховано витрати на здійснення науково-технічної розробки;

— розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою « Метод та апаратно-програмні засоби визначення напрямів звукових сигналів» є оцінювання науково-технічного рівня та рівня

комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [49]

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1

	Бали (за 5-ти бальною шкалою)				
	0	1	2	3	4
	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
0	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
1	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
2	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	4	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	3
6. Ринкові перспективи (розмір ринку)	3	4	4
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	2
10. Практична здійсненність (необхідність нових матеріалів)	2	2	2
11. Практична здійсненність (термін реалізації)	4	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	39	38	39
Середньоарифметична сума балів $СБ_с$	38,7		

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів» становить 38,7 бала, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою [49]:

$$q_i = \frac{P_i}{P_{базі}}. \qquad (5.1)$$

де q_i — одиничний параметричний індекс, розрахований за i-м параметром;

P_i — значення i-го параметра виробу;

$P_{базі}$ — аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.4.

Таблиця 5.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показники (параметри)	Одиниця вимірювання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість модулів підключення	шт	2	4	2	0,2
Кількість мікрофонних блоків	шт	4	16	4	0,1
Підсилення	Дб	4	5	1,25	0,25

Продовження таблиці 5.4

Показнкик (параметри)	Одиниця вимірювання	Аналог	Проектуваний пристрій	Відношення параметрів нової розробки до анлога	Питома вага показника
Точність позиціонування	%	90	95	1,06	0,15
Мінімальний рівень звукового сигналу	Дб	10	5	2	0,3
Експлуатаційні витрати	грн	25	15	0,6	0,45
Вартість	грн	11600	10800	0,93	0,55

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 — пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [49]:

$$I_{НП} = \prod_{i=1}^{n} q_i \ , \qquad (5.2)$$

де $I_{нп}$ — загальний показник конкурентоспроможності за нормативними параметрами;

q_i — одиничний (частинний) показник за i-м нормативним параметром;

n — кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [49]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ , \qquad (5.3)$$

де $I_{тп}$ — груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i — одиничний параметричний показник i-го параметра;

α_i — вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n — кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 5.4.

$$I_{mn} = 2 \cdot 0{,}2 + 4 \cdot 0{,}1 + 1{,}25 \cdot 0{,}25 + 1{,}06 \cdot 0{,}15 + 2 \cdot 0{,}3 = 1{,}87.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [49] :

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \ , \qquad (5.4)$$

де $I_{ЕП}$ — груповий параметричний індекс за економічними показниками;

q_i — економічний параметр i-го виду;

β_i — частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m — кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП} = 0{,}6 \cdot 0{,}45 + 0{,}93 \cdot 0{,}55 = 0{,}78.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [49] **:**

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{5.5}$$

$$K_{ІНТ} = 1 \cdot 1{,}87 / 0{,}78 = 2{,}39.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [49]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \tag{5.6}$$

де k — кількість посад дослідників залучених до процесу досліджень;

M_{ni} — місячний посадовий оклад конкретного дослідника, грн;

t_i — число днів роботи конкретного дослідника, дн.;

T_p — середнє число робочих днів в місяці, T_p=22 дні.

$$З_o = 12350{,}00 \cdot 22 / 22 = 12350{,}00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник	12350,00	561,36	22	12350,00
Інженер-розробник електронної апаратури	10200,00	463,64	11	5100,00
Інженер-програміст 1-ї кат.	11600,00	527,27	10	5272,73
Технік	7450,00	338,64	6	2031,82
Всього				24754,55

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_p$) за відповідними найменуваннями робіт НДР на тему « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів» розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (5.7)$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.8)$$

де M_M — розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=2379,00 грн;

K_i — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [49];

K_c — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p — середнє число робочих днів в місяці, приблизно T_p = 22 дн;

$t_{зм}$ — тривалість зміни, год.

$C_1 = 2379{,}00 \cdot 1{,}10 \cdot 1{,}65 / (22 \cdot 8) = 24{,}53$ *грн.*

$З_{р1} = 24{,}53 \cdot 6{,}00 = 147{,}20$ *грн.*

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (24754{,}55 + 1229{,}46) \cdot 11 / 100\% = 2858{,}24$ *грн.*

Таблиця 5.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обчислювального обладнання для проведення досліджень	6,00	2	1,10	24,53	147,20
Підготовка робочого місця інженера-програміста	4,50	3	1,35	30,11	135,49
Підготовка робочого місця інженера-розробника електронної апаратури	5,00	4	1,50	33,45	167,27
Інсталяція програмного забезпечення для моделювання та розробки	5,00	4	1,50	33,45	167,27
Контроль вхідних компонентів	1,50	4	1,50	33,45	50,18
Монтаж експериментальної моделі	10,50	4	1,50	33,45	351,27
Контроль ходу експериментів	7,00	3	1,35	30,11	210,76
Всього					1229,46

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \qquad (5.10)$$

де $Н_{зп}$ — норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (24754,55 + 1229,46 + 2858,24) · 22 / 100% = 6345,29 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{в\,j}\,, \qquad (5.11)$$

де H_j — норма витрат матеріалу *j*-го найменування, кг;

n — кількість видів матеріалів;

$Ц_j$ — вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ — коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

B_j — маса відходів *j*-го найменування, кг;

$Ц_{вj}$ — вартість відходів *j*-го найменування, грн/кг.

$М_1$ = *2,00 · 114,00 · 1,11 - 0,000 · 0,00 = 253,08 грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папірFAXE-500 A4	114,00	2,00	-	-	253,08
Папір для записів AXE 70 A5-250	47,00	5,00	-	-	260,85
Органайзер офісний OFFICE 30/50	220,00	2,00	-	-	488,40
Набір офісний DATA XOD	205,00	3,00	-	-	682,65
Картридж для принтера	1010,00	1,00	-	-	1121,10
Диск оптичний OPTIMA CD	15,50	3,00	-	-	51,62
Flesh-пам'ять GOODRAM 64 C10A	420,00	1,00	-	-	466,20
Чорнило для плотера	500,00	0,15	-	-	83,25
Всього					3407,15

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Методи та апаратно-програмні засоби визначення напрямів звукових сигналів», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (5.12)$$

де H_j — кількість комплектуючих j-го виду, шт.;

$Ц_j$ — покупна ціна комплектуючих j-го виду, грн;

$К_j$ — коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$).

$К_в = 1 \cdot 860,00 \cdot 1,11 = 954,60$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Зовнішня звукова карта Behringer U-PHORIA UMC404HD	1	860,00	954,60
Одноплатний комп'ютер Raspberry Pi 3B+	1	6785,00	7531,35
Мікрофон Behringer C1	1	68,00	75,48
Всього			8561,43

5.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (5.13)$$

де $Ц_i$ — ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ — кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i — коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10 \ldots 1,12$);

k — кількість найменувань устаткування.

$B_{спец} = 3200,00 \cdot 1 \cdot 1,11 = 3552,00$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Звуковий генератор ЛУНА-24	1	3200,00	3552,00
Ноутбук HP Laptop 15s-eq2037ua	1	30650,00	34021,50
Всього			37573,50

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.14)$$

де $Ц_{iпрг}$ — ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ — кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i — коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k — кількість найменувань програмних засобів.

$B_{прг} = 6670,00 \cdot 1 \cdot 1,11 = 7403,70$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Прикладний пакет обробки та аналізу звукових сигналів	1	6670,00	7403,70
Всього			7403,70

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.15)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (21100{,}00 \cdot 1) / (2 \cdot 12) = 879{,}17$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн

Персональний комп'ютер проектувальника EVEREST E5000	21100,00	2	1	879,17
Обчислювальна система проектування	23800,00	2	1	991,67
Робоче місце інженера-програміста	6500,00	5	1	108,33
Робоче місце розробника електронної апаратури	9580,00	5	1	159,67
Пристрій виводу текстової інформації	6800,00	4	1	141,67
Оргтехніка	8925,00	4	1	185,94
Приміщення лабораторії	290500,00	25	1	968,33
ОС Windows 11	7400,00	3	1	205,56
Прикладний пакет Microsoft Office 2019	6700,00	3	1	186,11
Всього				3826,44

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (5.16)$$

де W_{yi} — встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 4,50 грн;

$K_{вni}$ — коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i — коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,25 \cdot 164,0 \cdot 4,50 \cdot 0,95 / 0,97 = 184,50$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.12 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер проектувальника EVEREST E5000	0,25	164,0	184,50
Обчислювальна система проектування	0,40	130,0	234,00
Робоче місце інженера-програміста	0,05	120,0	27,00
Робоче місце розробника електронної апаратури	0,20	120,0	108,00
Пристрій виводу текстової інформації	0,60	2,5	6,75
Оргтехніка	0,65	7,5	21,94
Звуковий генератор ЛУНА-24	0,04	15,0	2,70
Ноутбук HP Laptop 15s-eq2037ua	0,05	65,0	14,63
Всього			599,51

5.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів» належать витрати на відрядження штатних працівників тощо. Витрати за статтею «Службові відрядження» відсутні.

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.17)$$

де $Н_{сп}$ — норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (24754,55 + 1229,46) · 30 / 100% = 7795,20 грн.

5.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.18)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 55%.

$І_в$ = (24754,55 + 1229,46) · 55 / 100% = 14291,20 грн.

5.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \qquad (5.19)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$$B_{нзв} = (24754,55 + 1229,46) \cdot 100 / 100\% = 25984,01 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи на тему « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв} \quad (5.20)$$

$$B_{заг}=24754,55+1229,46+2858,24+6345,294075+3407,15+8561,43+37573,50+7403,70+3826,44+599,51+0,00+7795,20+14291,20+25984,0=144629,67 \text{грн}$$

Загальні витрати $ЗВ$ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \qquad (5.21)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 144629,67 / 0,95 = 152241,76 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN — збільшення кількості споживачів пристрою, у періоди часу, що аналізуються, від покращення його певних характеристик;

N — кількість споживачів які використовували аналогічний пристрій у році до впровадження результатів нової науково-технічної розробки, приймемо 5500 осіб;

$Ц_б$ — вартість пристрою у році до впровадження результатів розробки, приймемо 9000,00 грн;

$\pm\Delta Ц_о$ ——зміна вартості пристрою від впровадження результатів науково-технічної розробки, приймемо 1843,50 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [49]**:**

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.22)$$

де λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =35%;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році ϑ =18%;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(1843,50·5500,00+10843,50·250)·0,83·0,35·(1-0,18/100%)=3061028,28 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(1843,50·5500,00+10843,50·850)·0,83·0,35·(1-0,18/100%)=4610846,36 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(1843,50·5500,00+10843,50·1650)·0,83·0,35·(1-0,18/100%)=6677270,47 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(1843,50·5500,00+10843,50·2400)·0,83·0,35·(1-0,18/100%)=8614543,07 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.23)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,14;

t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 3061028{,}28/(1+0{,}14)^1+4610846{,}36/(1+0{,}14)^2+6677270{,}47/(1+0{,}14)^3+$$
$$+8614543{,}07/(1+0{,}14)^4=2685112{,}52+3547896{,}55+4506967{,}37+5100501{,}05=$$
$$=15840477{,}49 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (5.24)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 152241,76 грн.

$$PV = k_{інв} \cdot ЗВ = 2 \cdot 152241{,}76 = 304483{,}53 \text{ грн.}$$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (5.25)$$

де $ПП$ — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 15840477,49 грн;

PV — теперішня вартість початкових інвестицій, 304483,53 грн.

$E_{абс} = ПП - PV$ = 15840477,49 - 304483,53 = 15535993,97 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.26)$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, 15535993,97 грн;

PV — теперішня вартість початкових інвестицій, 304483,53 грн;

$T_{ж}$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+15535993,97/304483,53)^{1/4} = 1,69.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.27)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d =0,11;

f — показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,15.

$\tau_{мін}$ = 0,11+0,15 = 0,26 < 1,69 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.28}$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,69 = 0,59 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів» становить 38,7 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,39 рази.

Також термін окупності становить 0,59 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою « Методи та апаратно-програмні засоби визначення напрямів звукових сигналів».

ВИСНОВКИ

Аналіз сучасних підходів до побудови та організації систем пасивної аудіолокації показав, що найбільш ефективні серед них засновані на використанні мікрофонних решіток. Використовуючи різні алгоритми обробки сигналів, отримуваних за допомогою мікрофонної решітки, можна сформувати потрібну діаграму спрямованості без змін геометричних параметрів, просторового положення та орієнтації решітки.

Аналіз сучасних методів та алгоритмів багатоканальної обробки звукових сигналів показав, що обробка сигналів з мікрофонних решіток може здійснюватися як у часовій, так і у частотній області. При цьому найбільш простим та ефективним алгоритмом є алгоритм, відповідно до якого для сигналу з кожного мікрофона вводиться часова затримка, що при подальшому зваженому підсумовуванні дозволяє підвищити відношення сигнал/шум.

Часові затримки у каналах можуть бути сформовані за рахунок фазових зсувів вузькосмугових частотних складових, отримуваних при розкладанні звукового сигналу методами цифрової обробки.

Функції багатоканального введення та аналого-цифрового перетворення звукових сигналів запропоновано реалізовувати за допомогою звукової карти Behringer U-PHORIA UMC404HD, а цифрову обробку сигналів з її виходу здійснювати за допомогою одноплатного комп′ютера Raspberry Pi3b+ з подальшим відображенням результатів на екрані ноутбука, що спрощує апаратну реалізацію системи та покращує її мобільність.

Розроблене програмне забезпечення дозволяє у графічній формі подавати розташування джерела звуку на діаграмі направленності у вікні браузера.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Крупельницький Л.В.Характеристики і структури багатоканальних АЦ-систем, що самокорегуються, для аналізу аудіо сигналів / Л.В.Крупельницький // Тези доповідей П’ятої Міжнародної науково-практичної конференції "Методи та засоби кодування, захисту й ущільнення інформації". Україна, Вінниця, 19-21 квітня 2016 р. – Вінниця: ВНТУ, 2016. – С.129-133.

2. Методи та засоби для визначення напрямку та для ідентифікації джерел звуків на місцевості / Ткаченко О. М.; Крупельницький Л. В.; Дерев'яга, Б. С.; Зінчук Р. С. // Матеріали XLV Науково-технічної конференції ВНТУ, Вінниця, 23-24 березня 2016 р. - Електрон. текст. дані. - 2016. - Режим доступу : http://ir.lib.vntu.edu.ua/handle/123456789/10934 4. Хлуденьков В. Система управления освещением – идеальная и оптимальная / В. Хлуденьков //Полупроводниковая светотехника, № 5, 2010, с. 78-81.

3. Азаров О. Д., Крупельницький Л. В., Куш Я. Ю., Структура багатоканальної аналого-цифрової системи, що самокоригується, для синхронного перетворення і опрацювання аудіо сигналів // Вісник Вінницького політехнічного інституту. — 2013. — № 2. — С. 66—72

4. Методи та апаратно-програмні засоби визначення напрямів звукових сигналів / Тарновський М. Г. / Рак М. І. / Матеріали конференції “Молодь в науці ” ВНТУ / Вінниця, 6 грудня 2021. Точка доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/view/14135

5. Азаров О. Д., Дудник О. В., Крупельницький Л. В. Багаторозрядні АЦП слідкувального типу з ваговою надлишковістю, що самокалібруються // Вісник Вінницького політехнічного інституту. — 2013. — № 2. — С. 66—72

6. Український міліарний портал [Електронний ресурс]. Точка доступу : mil.in.ua/uk/blogs/kontrbatarejni-rls-ukrayinskoyi-armiyi/

7. Файловий архів студента [Електронний ресурс]. Точка доступу : https://studfile.net/preview/5729308/

8. Столбов М.Б. Применение микрофонных решеток для дистанционного сбора речевой информации // Научно-технический вестник информационных технологий, механики и оптики. – 2015. – Т. 15. – № 4. – С. 661–675.

9. Діаграма - спрямованість – антена. Технічна енциклопедія TechTrend [Електронний ресурс]. Точка доступу : http://techtrend.com.ua/index.php?newsid=1780

10. М.Б. Столбо Применение микрофонных решеток для дистанционного сбора речевой информации / Столбо М.Б. // Научно-технический вестник информационных технологий, механики и оптики. 2015. том 15. № 4. С. 661- 675

11. Кривошейкин А. В., Перелыгин С. В. Микрофонная решетка для реализации направленной акустической антенны / А. В. Кривошейкин, С. В. Перелыгин // Изв. вузов. Приборостроение. 2015. Т. 58. № 3С. 221- 225.].

12. Файловий архів студента [Електронний ресурс]. Точка доступу : https://studfile.net/preview/5729308/

13. Гатчин Ю.А., Карпик А.П., Ткачев К.О., Чиков К.Н., Шлишевский В.Б. Теоретические основы защиты информации от утечки по акустическим каналам. Учеб. пособие. Новосибирск: СГГА, 2008. 194 с.

14. Van Trees H.L. Detection, Estimation and Modulation Theory, Part IV: Optimum Array Processing. NY: Wiley, 2002. 1470 p.

15. Звукометрия. Пособие для артиллерийских училищ РККА. Москва, государственное военное изда-тельство Наркомата обороны СССР, 1938. 222 с.

16. Хорев А.А. Средства акустической разведки: направленные микрофоны и лазерные акустические сис-темы разведки // Спецтехника и связь. 2008. № 3. С. 34–43.

17. Гатчин Ю.А., Карпик А.П., Ткачев К.О., Чиков К.Н., Шлишевский В.Б. Теоретические основы защиты информации от утечки по акустическим каналам. Учеб. пособие. Новосибирск: СГГА, 2008. 194 с.

18. Науменко И., Кизима В. Узконаправленная фазированная антенная решетка для дистанционного про-слушивания источников речевой информации [Електронний ресурс]. Режим доступу: http://www.bnti.ru/dbtexts/ipks/old/analmat/1/fazresh.pdf

19. Schelkunoff S.A. A mathematical theory of linear arrays // Bell System Technical Journal. 1943. V. 22. P. 80–107.

20. Frost O.L. An algorithm for linearly constrained adaptive array processing // Proceedings of the IEEE. 1972.

21. . Lewis J. Analog and Digital MEMS Microphone Design Considerations. Technical Article MS-2472. Analog Devices, 2013, 4 p.

22. Гольденберг Л. М. Цифровая обработка сигналов: учебное пособие для вузов / Гольденберг Л. М. – М. : Радио и связь, 1990. – 256 с.

23. Столбов М.Б. Применение микрофонных решеток для дистанционного сбора речевой информации // Научно-технический вестник информационных технологий, механики и оптики. – 2015. – Т. 15. – № 4. – С. 661–675.

24. Діаграма - спрямованість – антена. Технічна енциклопедія TechTrend [Електронний ресурс]. Точка доступу : http://techtrend.com.ua/index.php?newsid=1780

25. Інформаційний портал Вікіпедія. [Електронний ресурс] . Точка доступу : uk.wikipedia.org/wiki/Звукова_плата

26. Structure Client/server [Електронний ресурс]. Режим доступу: https://www.techopedia.com/definition/438/clientserver-architecture

27. Микрофоны: амплитудно – частотная характеристика [Електронний ресурс] Точка доступу: https://www.shure.ru/musicians/discover/educational/frequehttps://www.shure.ru/musicians/discover/educational/frequency-responsency-response

28. Основи звукозапису. Частина 4: Вибираємо конденсаторний мікрофон для домашньої студії [Електронний ресурс]. Точка доступу:

https://muztorg.ua/uk/index.php?route=information/aridius_news&aridius_news_id=143

29. The Different Types Of Mics And Their Uses [Електронний ресурс]. Точка доступу: https://www.gearank.com/articles/types-of-mics

30. Микрофоны: типы преобразователей [Електронний ресурс]. Точка доступу: https://www.shure.ru/musicians/discover/educational/transducer-types

31. Asus Tinker Board [Електронний ресурс]. Точка доступу: https://www.asus.com/ua-ua/Single-Board-Computer/Tinker-Board-S/

32. BeagleBone [Електронний ресурс]. Точка доступу: https://beagleboard.org/bone

33. Raspberry Pi [Електронний ресурс]. Точка доступу: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

34. The 5 Key Features to Look For [Електронний ресурс]. Точка доступу: https://ehomerecordingstudio.com/best-audio-interfaces/

35. Behringer U-Phoria UMC404HD [Електронний ресурс]. Точка доступу:http://prosound.ixbt.com/news/2015/march/12/behringer-umc404hd.shtml

36. Функціональна мова програмування [Електронний ресурс]. Режим доступу: https://careerkarma.com/blog/functional-programming-languages/

37. Обєктно орієнтоване програмування [Електронний ресурс]. Режим доступу: https://careerkarma.com/blog/object-oriented-languages/

38. Бібліотека Boost [Електронний ресурс]. Режим доступу: https://www.boost.org/

39. Біблітека PortAudio [Електронний ресурс]. Режим доступу: http://www.portaudio.com/

40. Бібліотека ReatcJS [Електронний ресурс]. Режим доступу: https://reactjs.org/docs/getting-started.html

41. Бібліотека CanvasJS [Електронний ресурс]. Режим доступу: https://canvasjs.com/

42. Hidri, A. About multichannel speech signal extraction and separation techniques / A. Hidri, S. Meddeb, H. Amiri // Journal of Signal and Information Processing. – 2012. – V. 3. – №2. – P. 238–247.

43. McCowan, I. Microphone arrays: a tutorial [Електронний ресурс]. Режим доступу: http://www.aplu.ch/home/download/microphone_array.pdf (дата обращения: 12.12.2019).

44. Beamforming Techniques for Multichannel audio Signal Separation [Електронний ресурс]. Режим доступу: https://arxiv.org/ftp/arxiv/papers/1212/1212.6080.pdf (дата обращения: 12.12.2019).

45. Springer handbook of speech processing / J. Benesti, J. M. Sondhi, Y. Huang (Eds.). – Springer, 2008. – 1159 p.

46. Bourgeois, J. Time-domain Beamforming and Blind Source Separation / J. Bourgeois, W. Minker. – Springer, 2009. – 228 p.

47. Столбов, М. Б. Исследование двухканального алгоритма MVDR для выделения речи из когерентного шума / М. Б. Столбов, Ч. Т. Куан // Научно-технический вестник информационных технологий, механики и оптики. – 2019. – Т. 19. – № 1. – С. 180–183.

48. Ермолаев, В. Т. Современные методы пространственной обработки сигналов в информационных системах с антенными решетками: учебно-методический материал по программе повышения квалификации / В. Т. Ермолаев, А. Г. Флаксман. – Нижний Новгород, 2007. – 99 с.

49. Методичні вказівки до виконання економічної частини магістерськихкваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

50. Методичні вказівки до виконання економічної частини магістерськихкваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТОК А

Міністерство освіти і науки України

Вінницький національний технічний університет

Інститут інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри__ОТ__

проф., д.т.н.. Азаров О.Д.

(наук. ст., вч. зв., ініц. та прізв.) (підпис)

" _____ " _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи

Метод та апаратно-програмні засоби визначення напрямків звукових сигналів

08-23.МКР.0010.00.003.ТЗ

Науковий керівник: доцент к.т.н.

_____ Тарновський М. Г.

(підпис)

«___»_____2021 р.

Виконав: студент групи 1КІ-20м

_____ Рак М. І.

(підпис)

«___»_____2021 р.

Вінниця 2021 р.

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР):

— акустична локалізація об’єктів та отримання інформації про джерело звуку важливі для завдань автоматизованого контролю доступу та захисту територій, в комп’ютерних системах для ідентифікації джерел звуку, в мультимедійних системах для стадіонів, концертних та виставкових залів, у військовій сфері тощо

— наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР:

— мета проекту — спрощення апаратної реалізації та розширення можливостей застосування системи аудіолокації на основі синхронної багатоканальної обробки звукових сигналів;

— призначення розробки — визначення підходів до побудови апаратно-програмних засобів системи пасивної аудіолокації для виявлення звукових джерел на контрольованій місцевості та визначення їх положення.

3. Вихідні дані для виконання МКР:

— призначення – визначення локалізації джерела звукового сигналу за методами пасивної аудіолокації;

— джерело вхідних сигналів – мікрофонна решітка;

— частотний діапазон вхідних сигналів – від 80 до 20000 Гц;

— кількість вхідних каналів – 2;

— підтримка взаємодії з комп′ютером

4 Вимоги до виконання МКР:

— провести техніко-економічне обґрунтування доцільності розробки;

— провести аналіз сучасних методів та алгоритмів обробки звукових сигналів в задачах пасивної аудіолокації;

— визначення підходів до реалізації апаратних та програмних засобів системи аудіолокації зі спрощеною організацією процесів синхронного багатоканального введення та обробки звукових сигналів;

— оцінити комерційний потенціал розробки.

5 Етапи МКР та очікувані результати наведено в таблиці 1.

Таблиця 1 — етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд і аналіз існуючих методів та рішень побудови систем аудіолокації	07.09.21	09.09.21	Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Аналіз методів та алгоритмів обробки звукових сигналів	10.09.21	28.09.21	розділ 2
3	Розробка апаратних засобів системи пасивної аудіолокації	29.09.21	10.11.21	розділ 3
4	Розробка програмних засобів системи пасивної аудіолокації	11.11.21	16.11.21	розділ 4
5	Оцінка комерційного потенціалу розробки	17.11.21	30.11.21	розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	01.12.21	19.12.21	пояснювальна записка, графічний матеріал і презентація

7 Матеріали, що подаються до захисту МКР: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

8 Порядок контролю виконання та захисту МКР є виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

9 Вимоги до оформлення МКР

Вимоги викладені в ДСТУ 3008 –2015 та Положенні про МКР 2021.

ДОДАТОК Б

Загальна структурна схема

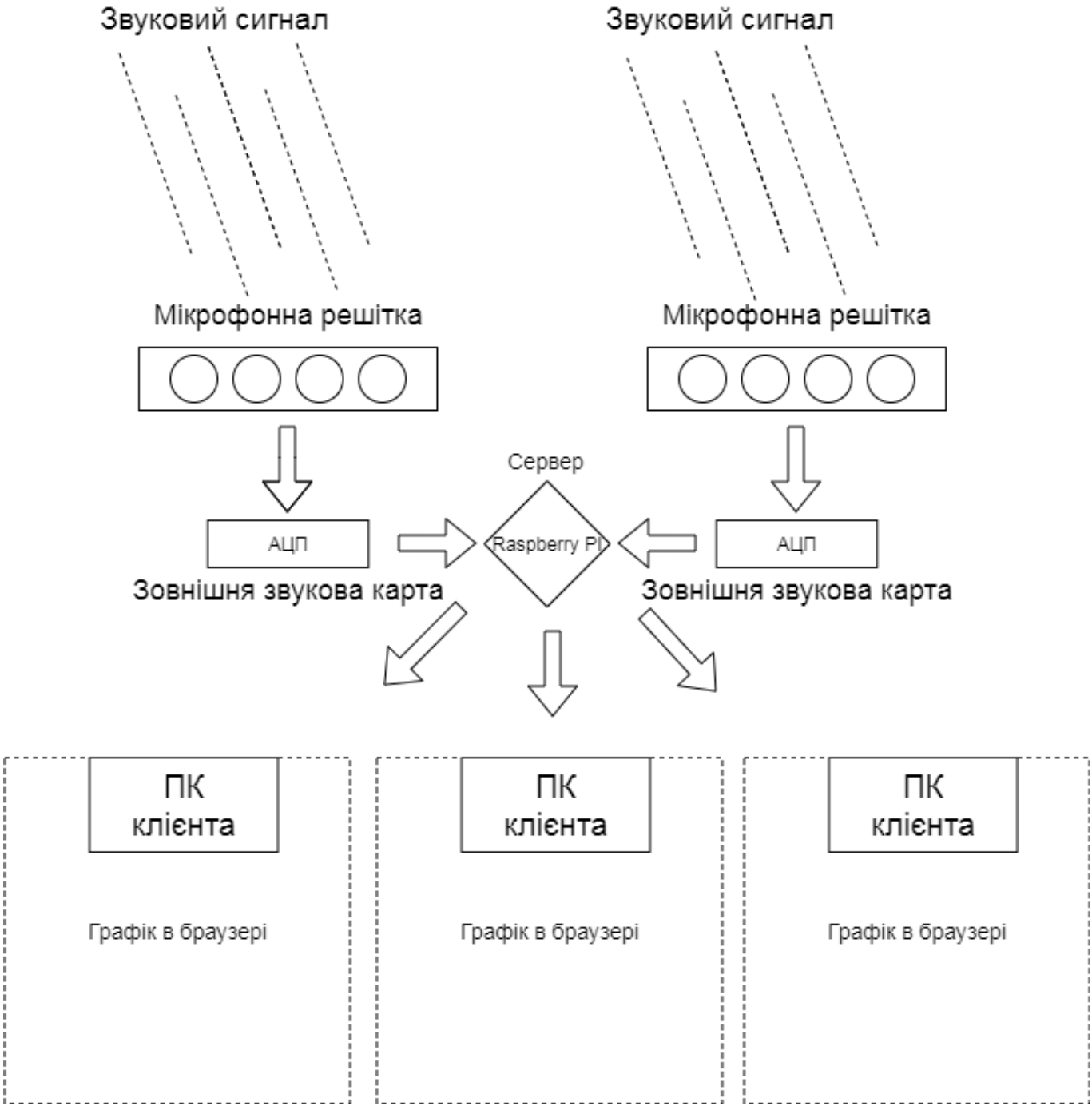

Рисунок Б.1—Загальна структурна схема

ДОДАТОК В

Технічні характеристики звукової карти

Behringer U-Phoria UMC404HD

Інтерфейс — USB.

Розрядність ЦАП — 24 біт.

Максимальна частота ЦАП (стерео) — 192 кГц.

Підтримка технологій:

— ASIO;

— CoreAudio.

Входи:

— XLR Combo,

— Аналоговий вхід,

— MIDI-вхід;

Виходи: 4 — 1/4 "TRS, 4 — RCA, 2 — XLR, 1 — MIDI-вихід, 1 х Вихід для навушників.

Тип — зовнішній.

Додаткові характеристики — Мікрофонні XLR-роз'єми з фантомним живленням +48 В і підсилювачем MIDAS, LED-індикатори наявності сигналу і кліппірованія для входів.

Роз'єм —USB.

Роз'єм — Kensington Lock.

ДОДАТОК Г

Технічні характеристики ноутбука MacBook Air 2017

Процесор — Intel Core i5-5350U.

Графічний адаптер — Intel HD Graphics 6000.

Оперативна пам'ять — 8192 Мбайт, 1600 MHz LPDDR3 onboard.

Дисплей — 13.3 дюйм. 16:10, 1440 x 900 пікс. 128 точок / дюйм, TN LED глянцеве покриття.

Материнська плата Intel Broadwell-U PCH-LP (Premium)

Зберігання даних: Apple SSD SM0128G, 128 Гбайт

Звукова плата: Intel Broadwell PCH-LP High Definition Audio Controller

Інтерфейси — USB 3.0 / 3.1 Gen , Thunderbolt, 1 DisplayPort,

Аудіороз'єм — 3.5mm Headphone.

Картрідер — SDXC Cardreader, датчик освітленості.

Коммунікації Broadcom 802.11ac (a / b / g / n = Wi-Fi 4 / ac = Wi-Fi 5). Bluetooth — 4.0.

Габарити (висота x ширина x глибина) — 17 x 325 x 227 (мм).

Аккумулятор — 54 Вт·ч літій-полімерн.

Операційна система — Apple macOS 10.12 Sierra.

Камера — 720p FaceTime Camera.

Додаткова акустична система: Stereo Speaker.

Клавіатура — Chiclet, підсвічування клавіатури.

Вага — 1.35 Кг.

Адаптер живлення — 200 г.

ДОДАТОК Д

Технічні характеристики мікрофона

Микрофон Behringer C1.

Частотний діапазон — Гц 40-20000.

Призначення — студійний.

Спрямованість — кардіоїда.

Спосіб підключення — дротової.

Тип перетворювача — конденсаторний.

Номінальний опір — Ом 100.

Вихідний штекер — XLR.

Додатково в комплекті адаптер для мікрофонной стійки, кейс.

Колір — сріблястий.

Розміри, мм — 54 х 169.

Вага, г — 450.

ДОДАТОК Е

Конденсаторний мікрофон. Схема електрична принципова

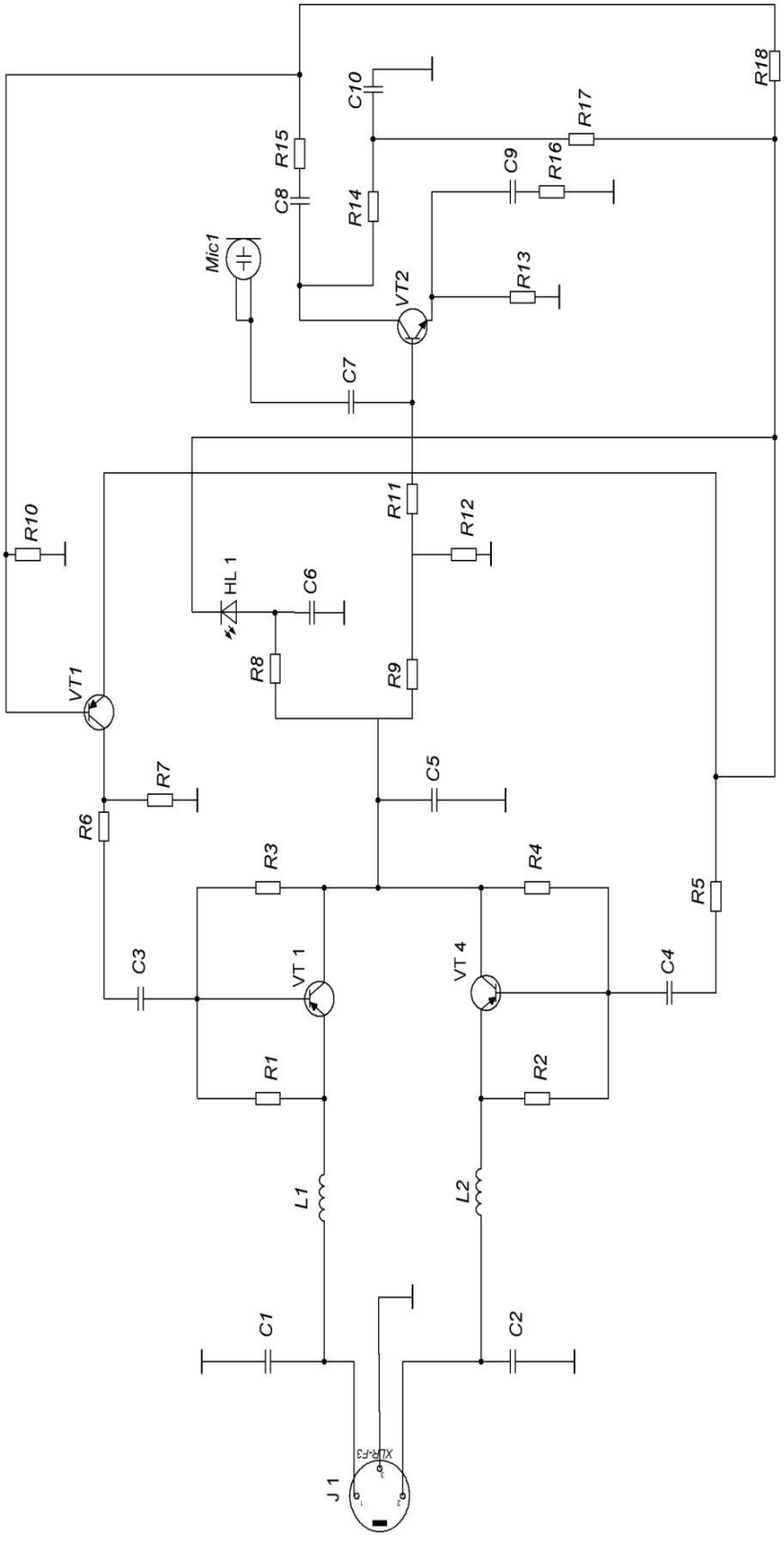

Рисунок Е.1 — Схема електрична принципова

Поз. познач.	Найменування	Кіл.	Примітка
L1, L2	Катушка індуктивності ДМ-0,6-200 мкГн	2	
C1…C10	Конденсатор К50-6 25В 100мкФ.464.031.ТУ	10	
HL1	Світлодіод SMD 2835 3,2V	1	
R1…R18	Резистор МЛТ-0,125-200 Ом+5%	18	
T1…T4	Транзистор КТ315В-ЖКЗ.365.200 ТУ	4	
J1	Роз'єм XLR-F3	1	
Mic1	Мікрофон KY-037	1	

Змн.	Арк.	№ докум.	Підпис	Дат	08-23.МКР.010.00.000 ПЕ3		
Розроб.		Рак М.І.			Конденсаторний мікрофон	Літ.	Арк. / Аркушів
Перевір.		Тарновський.					
Реценз.		Лужецький			Перелік елементів	ВНТУ 1КІ-20м	
Н. Контр.		Швець С. І.					
Затверд.							

ДОДАТОК Ж

Загальний алгоритм роботи

Рисунок Ж 1 — Загальний алгоритм роботи

ДОДАТОК К

Лістинг програми

Лістинг програми звукового модуля

```
#include <boost/beast/core.hpp>
#include <boost/beast/websocket.hpp>
#include <boost/asio/dispatch.hpp>
#include <boost/asio/strand.hpp>
#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>
#include <boost/algorithm/string/split.hpp>
#include <algorithm>
#include <cstdlib>
#include <functional>
#include <iostream>
#include <memory>
#include <string>
#include <thread>
#include <vector>
#include <portaudio.h>
#include <math.h>
#include <nlohmann/json.hpp>
using json = nlohmann::json;
namespace beast = boost::beast;         // from <boost/beast.hpp>
namespace http = beast::http;           // from <boost/beast/http.hpp>
namespace websocket = beast::websocket; // from
<boost/beast/websocket.hpp>
namespace net = boost::asio;            // from <boost/asio.hpp>
namespace ublas = boost::numeric::ublas;
using tcp = boost::asio::ip::tcp;       // from <boost/asio/ip/tcp.hpp>
typedef struct
{
    int             frameIndex;  /* Index into sample array. */
    int             maxFrameIndex;
    int             channels;
    std::vector<float>  dataLeft;
    std::vector<float>  dataRight;
```

```
    } paData;
    void fail(beast::error_code ec, char const* what)
    {
       std::cerr << what << ": " << ec.message() << "\n";
    }
    // Echoes back all received WebSocket messages
    class session : public std::enable_shared_from_this<session>
    {
       websocket::stream<beast::tcp_stream> ws_;
       beast::flat_buffer buffer_;
    public:
       // Take ownership of the socket
       explicit session(tcp::socket&& socket) : ws_(std::move(socket)) {
       }
       bool is_open() { return ws_.is_open(); }
       // Get on the correct executor
       void run() {
          // We need to be executing within a strand to perform async operations
          // on the I/O objects in this session. Although not strictly necessary
          // for single-threaded contexts, this example code is written to be
          // thread-safe by default.
          net::dispatch(ws_.get_executor(),
beast::bind_front_handler(&session::on_run, shared_from_this()));
       }
       // Start the asynchronous operation
       void
       on_run() {
    ws_.set_option(websocket::stream_base::timeout::suggested(beast::role_type:
:server));
          // Set a decorator to change the Server of the handshake
          ws_.set_option(websocket::stream_base::decorator(
             [](websocket::response_type& res) {
                res.set(http::field::server,
                   std::string(BOOST_BEAST_VERSION_STRING) +
                   " SoundDirectionServer");
          }));
          // Accept the websocket handshake
```

```
            ws_.async_accept(beast::bind_front_handler(&session::on_accept,
shared_from_this()));
        }
        void
        on_accept(beast::error_code ec) {
            if(ec)
                return fail(ec, "accept");
            // Read a message
            do_read();
        }
        void
        do_read() {
            // Read a message into our buffer
            ws_.async_read(buffer_,    beast::bind_front_handler(&session::on_read,
shared_from_this()));
        }
        void on_read(beast::error_code ec, std::size_t bytes_transferred) {
            boost::ignore_unused(bytes_transferred);
            // This indicates that the session was closed
            if(ec == websocket::error::closed)
                return;
            if(ec)
                fail(ec, "read");
            if(buffer_.data().size() > 0) {
                std::string cmd = beast::buffers_to_string(buffer_.data());
                if(cmd == "quit") {
                    buffer_.clear();
                    ws_.close(websocket::close_reason(1000));
                } else {
                    std::cout                <<                sockaddr().sa_data                <<
beast::buffers_to_string(buffer_.data()) << std::endl;
                    do_read();
                }
            } else {
                std::cout                <<                sockaddr().sa_data                <<
beast::buffers_to_string(buffer_.data()) << std::endl;
                do_read();
            }
```

```
        }
        void write_data(const paData *data) {
            const int N = 1000;
            const int ticks = data->maxFrameIndex / N;
            std::vector<float> d;

            float value = 0.0;
            for(int t = 0; t < ticks; t++) {
                std::vector<float> left;
                for (auto i = t * N; i < data->dataLeft.size(); ++i) {
                    left.push_back(data->dataLeft.at(i));
                }
                std::vector<float> right;
                for (auto i = 0; i < data->dataRight.size() - t * N; ++i) {
                    left[i] += data->dataRight.at(i);
                }
                value = 0.0;
                for (auto i = 0; i < left.size(); ++i) {
                    value += std::pow(left.at(i) * N, 2.0);
                }
                value /= left.size();
                value = std::sqrt(value);
                d.push_back(value);
                left.clear();
                right.clear();
            }
            value = 0.0;
            for (auto i = 0; i < std::min(data->dataLeft.size(), data->dataRight.size());
++i) {
                value += std::pow((data->dataLeft.at(i) + data->dataRight.at(i)) * N,
2.0);
            }
            value /= std::min(data->dataLeft.size(), data->dataRight.size());
            value = std::sqrt(value);
            d.push_back(value);
            // Right
            for(int t = 0; t < ticks; t++) {
                std::vector<float> left;
```

```
                for (auto i = 0; i < data->dataLeft.size() - t * N; ++i) {
                    left.push_back(data->dataLeft.at(i));
                }
                std::vector<float> right(data->maxFrameIndex - t * 100);
                for (auto i = t * N; i < data->dataRight.size(); ++i) {
                    left[i - t * N] += data->dataRight.at(i);
                }
                value = 0.0;
                for (auto i = 0; i < left.size(); ++i) {
                    value += std::pow(left.at(i) * N, 2.0);
                }
                value /= left.size();
                value = std::sqrt(value);
                d.push_back(value);
                left.clear();
                right.clear();
            }
            //std::vector<float> d = {};
            json rdata;
            rdata["data"] = json(d).dump();
            rdata["index"] = data->frameIndex;
            rdata["maxIndex"] = data->maxFrameIndex;
            rdata["channels"] = data->channels;
            std::string response = rdata.dump();
            auto b = boost::asio::dynamic_buffer(response);
            ws_.async_write(b.data(), beast::bind_front_handler(&session::on_write,
shared_from_this()));
        }
        void on_write(beast::error_code ec, std::size_t bytes_transferred) {
            boost::ignore_unused(bytes_transferred);
            if(ec)
                return fail(ec, "write");
            // Clear the buffer
            buffer_.consume(buffer_.size());
            // Do another read
            // do_read();
        }
    };
```

```
typedef std::shared_ptr<session> session_;
std::vector<session_> sessions;
// Accepts incoming connections and launches the sessions
class listener : public std::enable_shared_from_this<listener>
{
   net::io_context& ioc_;
   tcp::acceptor acceptor_;
public:
   listener(net::io_context& ioc, tcp::endpoint endpoint) : ioc_(ioc), acceptor_(ioc) {
      beast::error_code ec;

      // Open the acceptor
      acceptor_.open(endpoint.protocol(), ec);
      if(ec)
      {
         fail(ec, "open");
         return;
      }
      // Allow address reuse
      acceptor_.set_option(net::socket_base::reuse_address(true), ec);
      if(ec)
      {
         fail(ec, "set_option");
         return;
      }

      // Bind to the server address
      acceptor_.bind(endpoint, ec);
      if(ec)
      {
         fail(ec, "bind");
         return;
      }
      // Start listening for connections
      acceptor_.listen(net::socket_base::max_listen_connections, ec);
      if(ec)
      {
```

```
            fail(ec, "listen");
            return;
        }
    }
    void
    run() {
        std::cout << "Listening for incoming connections...\n";
        do_accept();
    }
private:
    void
    do_accept() {
        // The new connection gets its own strand
        acceptor_.async_accept(net::make_strand(ioc_),
beast::bind_front_handler(&listener::on_accept, shared_from_this()));
    }
    void
    on_accept(beast::error_code ec, tcp::socket socket) {
        if(ec)
        {
            fail(ec, "accept");
        }
        else
        {
            // Create the session and run it
            session_ s = std::make_shared<session>(std::move(socket));
            sessions.push_back(s);
            s->run();
        }

        // Accept another connection
        do_accept();
    }
};

const PaDeviceInfo *deviceInfo;
PaStreamParameters inputParameters;
PaStream *stream = nullptr;
```

```
        paData data;
        float *buff = nullptr;
        static int recordCallback( const void *inputBuffer, void *outputBuffer,
                                   unsigned long framesPerBuffer,
                                   const PaStreamCallbackTimeInfo* timeInfo,
                                   PaStreamCallbackFlags statusFlags,
                                   void *userData )
        {
           paData *data = (paData*)userData;
           const float *rptr = (const float*)inputBuffer;
           int finished;
           long framesToCalc;
           unsigned long framesLeft = data->maxFrameIndex - data->frameIndex;
           if( framesLeft < framesPerBuffer ) {
              framesToCalc = framesLeft;
              finished = paComplete;
           } else
           {
              framesToCalc = framesPerBuffer;
              finished = paContinue;
           }

           if(inputBuffer != NULL) {
              for( int i = 0; i < framesToCalc; i++ ) {
                 //std::cout << *rptr << " ";
                 //std::cout << data->dataLeft.size() << " " << data->frameIndex + i <<
std::endl;
                 data->dataLeft.push_back(*rptr++);  /* left */
                 if(data->channels == 2) {
                    //std::cout << *rptr << std:: endl;
                    //std::cout << data->dataRight.size() << " " << data->frameIndex + i
<< std::endl;
                    data->dataRight.push_back(*rptr++);  /* right */
                 }
              }
           }
           data->frameIndex += framesToCalc;
           if(finished == paComplete) {
```

```
            std::for_each(sessions.begin(), sessions.end(), [&](session_ &s){
                if(s->is_open()) {
                    s->write_data(data);
                } else {
                    sessions.erase(std::remove(sessions.begin(), sessions.end(), s),
sessions.end());
                }
            });

            data->dataLeft.clear();
            data->dataRight.clear();

            finished = paContinue;
            data->frameIndex = 0;
        }

        return finished;
    }
    int main(int argc, char* argv[])
    {
        // Check command line arguments.
        if (argc != 4)
        {
            std::cerr <<
            "Usage: SoundDirectionServer <address> <port> <threads>\n" <<
            "Example:\n" <<
            "    SoundDirectionServer 0.0.0.0 8080 1\n";
            return EXIT_FAILURE;
        }
        auto const address = net::ip::make_address(argv[1]);
        auto const port = static_cast<unsigned short>(std::atoi(argv[2]));
        auto const threads = std::max<int>(1, std::atoi(argv[3]));
        PaError error = Pa_Initialize();
        if(error != paNoError) {
            std::cerr << Pa_GetErrorText(error) << std::endl;
            Pa_Terminate();
        } else std::cout << "Pa_Initialize\n";
        /* Input device parameners */
```

```
        inputParameters.device = Pa_GetDefaultInputDevice();
        if(inputParameters.device == paNoDevice) {
           Pa_Terminate();
           std::cerr << "No input devices.\n";
           return -1;
        }
        deviceInfo = Pa_GetDeviceInfo(Pa_GetDefaultInputDevice());
        inputParameters.channelCount = deviceInfo->maxInputChannels;
        inputParameters.sampleFormat = paFloat32;
        inputParameters.suggestedLatency = deviceInfo->defaultLowInputLatency;
        inputParameters.hostApiSpecificStreamInfo = nullptr;
        std::cout << "Device is " <<deviceInfo->name <<std::endl;
        if(stream == nullptr || !Pa_IsStreamActive(stream)) {
           // Opening stream
           data.maxFrameIndex = deviceInfo->defaultSampleRate; /* Record for a
few seconds. */
           data.frameIndex = 0;
           data.channels = inputParameters.channelCount;
           data.dataLeft.resize(data.maxFrameIndex);
           data.dataRight.resize(data.maxFrameIndex);

           error = Pa_OpenStream(&stream, &inputParameters, nullptr,
                        deviceInfo->defaultSampleRate,
                        512,
                        paClipOff, // we won't output out of range samples so don't
bother clipping them
                        recordCallback,
                        &data);
           if(error != paNoError) {
              Pa_Terminate();
              std::cerr << Pa_GetErrorText(error) << std::endl;
              return -1;
           }
           error = Pa_StartStream( stream );
           if(error != paNoError) {
              Pa_Terminate();
              std::cerr << Pa_GetErrorText(error) << std::endl;
              return -1;
```

```
            }
            std::cout << "Recording...\n";
        }
        std::cout << "Starting server...\n";
        // The io_context is required for all I/O
        net::io_context ioc{threads};
        // Create and launch a listening port
        std::make_shared<listener>(ioc, tcp::endpoint{address, port})->run();
        // Run the I/O service on the requested number of threads
        std::vector<std::thread> v;
        v.reserve(threads - 1);
        for(auto i = threads - 1; i > 0; --i)
            v.emplace_back([&ioc] {
                ioc.run();
            });
        ioc.run();
        /* PortAudio finish */
        free(buff);
        Pa_Terminate();
        /* PortAudio finish */
        return EXIT_SUCCESS;
    }
```

Лістинг модуля графічного відображення

```
    import React, {useState, useRef} from 'react';
    import './App.css';
    import Chart from './Chart';
    function ValidateIPaddress(ipaddress) {
        return /^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-
9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-
9]|[01]?[0-9][0-9]?)$/.test(ipaddress);
    }
    function App() {
        const input = useRef(null);
        const [ip, setIP] = useState(null);
        function handleIPChange(e) {
            e.persist();
            if(ip == null) {
                const val = input.current.value;
```

```
            if(ValidateIPaddress(val)) {
               setIP(val);
            } else {
               alert("Plese, enter a valid IP address.");
            }
         } else {
            setIP(null);
         }
      }

      return (
         <div className="container">
            <div className="col s12">
               <div className="row">
                  <div className="input-field col s10">
                     <input placeholder="Enter server IP." id="ip" type="text" 
className="validate" ref={input}/>
                  </div>
                  <div className="input-field col s2">
                     <button className="btn waves-effect waves-light" 
type="submit" name="submit_ip" onClick={handleIPChange}> {ip == null ? 
"Submit" : "Disconnect"}
                        <i className="material-icons right">send</i>
                     </button>
                  </div>
               </div>
            </div>
            <div className="col s12">
               <Chart ip={ip}/>
            </div>
         </div>
      );
   }
   export default App;
```

Лістинг програми для відображення графіків

```
   var CanvasJSChart = CanvasJSReact.CanvasJSChart;
   const defaultOptions = {
      animationEnabled: false,
```

```
    title:{
        text: "Sound Direction Diagram"
    },
    axisY: {
        title: "V",
        //prefix: "$",
        //includeZero: true
    },
    data: []
};
function Chart(props) {
    const [socket, setSocket] = useState(null);
    const [options, setOptions] = useState(defaultOptions);
    useEffect(() => {
        if(socket != null) {
            socket.send("quit");
            setSocket(null);
            setOptions(defaultOptions);
        }
        const ip = props.ip;
        if(ip != null) {
            let s = new WebSocket("ws://" + ip + ":8080");
            s.onopen = function(e) {
                console.log(`Opened on ${ip}`);
            };
            s.onmessage = function(event) {
                const response = JSON.parse(event.data);
                const data = JSON.parse(response.data);
                const optData = [];
                data.forEach((val, index) => {
                    optData.push({
                        x: index,
                        y: val
                    });
                });
                setOptions({
                    ...defaultOptions,
                    data: [{
```

```
                    type: "spline",
                    dataPoints: optData
                }]
            });
            //chart.current.render();
        };
        s.onclose = function(event) {
            if (event.wasClean) {
                alert(`Соединение закрыто, код: ${event.code}`);
                console.log(event);
            } else {
                // например, сервер убил процесс или сеть недоступна
                // обычно в этом случае event.code 1006
                alert('Соединение прервано');
            }
        };
        s.onerror = function(error) {
            alert("Disconnected.");
            console.log(error);
            setSocket(null);
        }
        setSocket(s);
    }
  }, [props.ip])
  return <CanvasJSChart options = {options}
    //ref={chart}
  />;
}
export default Chart;
```

ДОДАТОК Л

Результат роботи пристроя

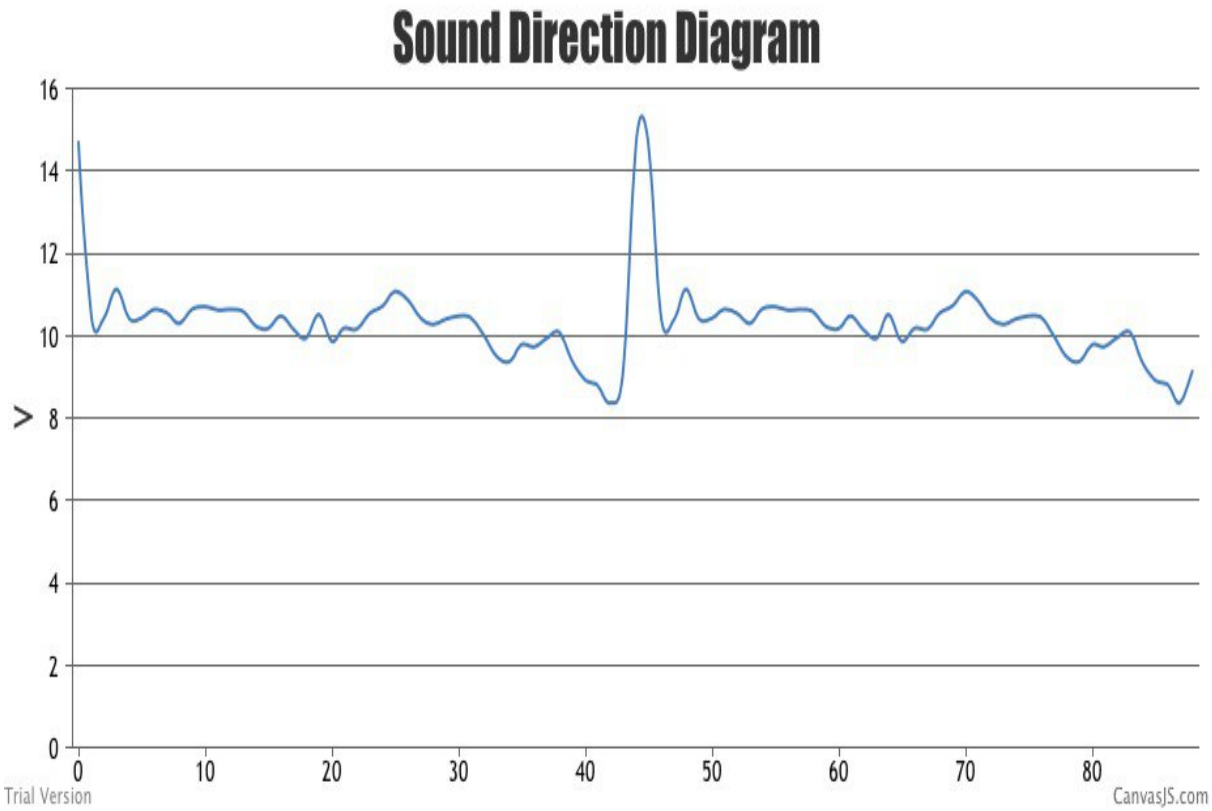

Рисунок Л. 1— Результат роботи пристроя

ДОДАТОК М

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Метод та апаратно-програмні засоби визначення напрямів звуку.

Тип роботи: Магістерська кваліфікаційна робота.

Підрозділ: кафедра Обчислювальної техніки ,Факультет інформаційних технологій та комп'ютерної інженерії, група 1КІ-20м.

Науковий керівник кандидат технічних наук, доцент кафедри ОТ, Тарновський М.Г.

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1		Оригінальність	
КП2			
Тривога/Білі знаки	/	Схожість	

Аналіз звіту подібності (відмінити подібне)

- ☐ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності і відсутності самостійності її автора. Робот направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається):

Автор _____ _____

Опис прийнятого рішення

_____

_____

Особа, відповідальна за перевірку_____ _____

Експерт _____ _____