МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

Петручик Ілля Ігорович

СТЕГАНОГРАФІЯ В АУДІО МЕТОДОМ ЗМІНИ ФАЗИ ЗВУКОВОГО СИГНАЛУ (PHASE CODING)

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «магістр»

Науковий керівник:
ГОЛОВІН МИКОЛА БОРИСОВИЧ,
кандидат фізико-математичних наук,
доцент кафедри комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри комп'ютерних наук та кібербезпеки
від _____ 2025 р.
Завідувач кафедри

(_____) Гришанович Т. О.

ЛУЦЬК – 2025

ЗМІСТ

ВСТУП

Актуальність теми. Поточний стан цифрових технологій відзначається швидким нарощуванням інформаційних потоків та зростанням вимог до їх захисту. Експансія онлайн-платформ, мультимедіа та хмарних технологій створює попит на надійні способи забезпечення конфіденційності даних. Серед перспективних рішень у цій сфері виділяється стеганографія – дисципліна, що займається маскуванням даних у різноманітних цифрових медіа, включаючи графіку, відеоролики та звукові файли. Стеганографія аудіо сигналів, особливо техніка фазового кодування (Phase Coding), привертає увагу завдяки можливості досягати значної непомітності прихованих даних при мінімальному впливі на оригінальний звуковий ряд. Даний метод інтегрує досягнення цифрової сигнальної обробки з елементами криптографії, що уможливлює створення комунікаційних систем з посиленим захистом. Прогрес у сфері цифрового аудіо, включаючи формати з покращеною якістю та розширеним частотним діапазоном, відкриває додаткові перспективи для розвитку стеганографічних технологій. Існуючі рішення вимагають подальшого вдосконалення, для забезпечення оптимального співвідношення між надійністю маскування, обсягом переданих даних та відсутністю артефактів. У зв'язку з цим, вивчення методів фазової модуляції є важливим науковим завданням, що сприяє підвищенню якості прихованої комунікації через аудіо канали.

Мета роботи – вивчити та втілити метод аудіо стеганографії, заснований на модифікації фазових характеристик звукового сигналу (Phase Coding), з метою забезпечення непомітної передачі інформації без погіршення слухового сприйняття.

Для досягнення поставленої мети потрібно:

- здійснити огляд наявних підходів до стеганографії в аудіо, виявивши їхні переваги та недоліки;
- вивчити механізм функціонування методу фазової модуляції та його математичний апарат;

- створити алгоритмічне рішення для вбудовування та витягування прихованих даних з аудіо матеріалу;
- впровадити програмне втілення методу з використанням Python та відповідних бібліотек для роботи з сигналами;
- виконати тестування для визначення якості відтворення, ефективності маскування та стійкості до деформацій;
- сформулювати практичні рекомендації щодо використання розробленого програмного додатку.

Об’єкт дослідження – методи стеганографії в аудіо сигналах.

Предмет дослідження – метод зміни фази звукового сигналу для приховання інформації в аудіо.

Апробація результатів роботи – результати дослідження планується представити у вигляді наукової статті та продемонструвати у рамках науково-практичної конференції з проблем інформаційної безпеки та цифрових технологій.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ОСНОВИ СТЕГАНОГРАФІЇ В АУДІОСИГНАЛАХ

1.1. Поняття та класифікація методів стеганографії

Стеганографія – це наука, що займається маскуванням процесу передачі даних через їх інтеграцію у різноманітні цифрові контейнери: графічні файли, відеоматеріали, звукові записи чи текстові документи. Відмінність від криптографії полягає у тому, що остання кодує зміст повідомлення, тоді як стеганографія приховує саме існування комунікації. Завдяки цьому, вона стає цінним засобом у сфері захисту даних, надаючи додатковий захисний бар'єр від перехоплення та неавторизованого моніторингу [1].

Сучасні стеганографічні технології ґрунтуються на внесенні змін у ті характеристики цифрових даних, які залишаються непомітними для органів чуття людини. Зокрема, в аудіо стеганографії, модифікації торкаються параметрів звукової хвилі (амплітуда, частотні компоненти, фазові зсуви). Головна суть аудіо стеганографії, полягає в тому, що ці зміни не викликають слухових артефактів. Основна вимога до подібних технік це досягнення рівноваги між трьома критичними показниками:

- стійкістю, здатністю протистояти стисненню та зашумленню;
- пропускною здатністю, обсягом прихованої інформації;
- прихованістю, відсутністю відчутної деградації якості.

Стеганографічні техніки систематизуються за декількома критеріями. Відповідно до природи носія інформації розрізняють [2]:

- графічну стеганографію, що ґрунтується на маніпуляціях із значеннями пікселів;
- відео стеганографію, котра поєднує просторові та темпоральні атрибути відео послідовностей;
- аудіо стеганографію, що експлуатує характеристики акустичних сигналів;
- текстову стеганографію, призначену для маскування даних у літерних

послідовностях;

- мережеву стеганографію, яка використовує специфіку транспортування пакетів у телекомунікаційних системах.

- Відповідно до природи носія інформації та механізму функціонування, стеганографічні техніки поділяються на три основні категорії [3].

Техніки просторового домену (LSB — найменший значущий біт). Ґрунтуються на модифікації найменш впливових бітових позицій, наприклад у амплітудному представленні. Характеризуються простотою впровадження, проте виявляють слабку резистентність до сигнальної обробки.

Техніки частотного домену. Засновані на корекції коефіцієнтів спектральних перетворень. Демонструють підвищену стійкість завдяки операціям у частотному просторі.

Техніки фазового домену (Phase Coding). Оперують фазовими характеристиками сигналу для інтеграції прихованих повідомлень. Мають виняткову непомітність, оскільки слуховий апарат людини малочутливий до фазових модифікацій.

Додатково стеганографічні підходи диференціюють за моделлю доступу до ключової інформації [4]:

- з публічним ключем (екстракція даних вимагає лише загальнодоступних параметрів);
- з приватним ключем (відновлення повідомлення можливе виключно за наявності конфіденційного ключа);
- комбіновані схеми (інтегрують елементи обох моделей).

Стеганографія становить значущий сегмент сучасної інформатики, що уможливлює латентну передачу відомостей у цифрових екосистемах. Її практичне застосування набуває критичного значення в контексті посилених вимог до конфіденційності, а також під час транспортування чутливої або інтелектуальної власності [5].

Цифровий аудіосигнал являє собою дискретизоване відображення звукової хвилі, що формується внаслідок трансформації аналогового сигналу у цифрову структуру через процеси семплювання та квантування. Подібна форма представлення надає можливості надійного збереження, транспортування, обробки та дослідження звукової інформації в електронних системах. У контексті стеганографії цифровий формат звукових даних виявляється особливо придатним, адже він відкриває значні перспективи для зміни окремих характеристик сигналу без суттєвого погіршення його якості. Ключовими характеристиками цифрового аудіосигналу виступають частота семплювання, розрядність квантування, число каналів та швидкість передачі даних[6].

Частота семплювання (sampling rate). Відображає число відліків неперервного сигналу на одиницю часу. Її вимірюють у герцах (Гц), і вона визначає максимальну частоту відтворюваного звуку. Як приклад, частота 44,1 кГц, застосовувана у компакт-дисках, забезпечує відтворення частот до 22,05 кГц, що охоплює діапазон людського слуху.

Розрядність квантування (bit depth). Визначає число бітів для кодування одного семплу. Параметр визначає динамічний діапазон та точність передачі амплітуди. Так, 16-бітове кодування гарантує приблизно 96 дБ динамічного діапазону, а 24-бітове — понад 120 дБ.

Число каналів (channels). Характеризує просторовий аспект аудіо. Монофонічні записи (1 канал) відтворюють звук з одного джерела, а стереофонічні (2 канали) забезпечують ефект просторовості, а багатоканальні конфігурації (5.1, 7.1, тощо) дозволяють реалізувати об'ємне звучання.

Швидкість передачі даних (bitrate). Відображає обсяг інформації, що обробляється за одиницю часу, і служить індикатором якості та рівня стиснення аудіофайлу.

Крім основних характеристик, суттєве значення мають спектральний склад, частотно-амплітудна характеристика та фазові властивості сигналу, що визначають його придатність для стеганографічних застосувань. Саме фазові властивості, на противагу амплітудним, чинять мінімальний вплив на слухове

сприйняття, що робить їх оптимальними для інтеграції прихованих даних методом Phase Coding [7].

У процесі характеризації та дослідження аудіосигналів застосовуються математичні апарати, що базуються на дискретному перетворенні Фур'є (DFT) чи швидкому перетворенні Фур'є (FFT), які уможливлюють вивчення частотної структури сигналу. Ці підходи становлять фундаментальні засоби у створенні алгоритмів обробки та інтеграції даних, забезпечуючи трансформацію між темпоральною та спектральною формами відображення сигналу [8].

Отже, цифровий аудіосигнал представляє собою комплексну сутність з великою кількістю параметрів, що інтегрує фізичні, математичні та інформаційні аспекти. Його властивості визначають потенціал ефективної інтеграції латентної інформації, збереження аудіоякості та резистентності сигналу до наступної обробки.

1.2. Методи приховування інформації в аудіо

Вбудовування даних в аудіофайли є однією з найактуальніших і розвиненіших задач цифрової стеганографії, адже людське слухове сприйняття характеризується обмеженою чутливістю до певних акустичних параметрів. Ця особливість уможливлює модифікацію сигнальних характеристик у спосіб, за якого інтегроване повідомлення лишається неідентифікованим для людини, що прослуховує запис, проте зберігає можливість точного відновлення при декодуванні [9].

Технології аудіостеганографії систематизуються відповідно до способу трансформації звукових властивостей. Основні категорії цих технологій охоплюють [10]:

- методи просторової області;
- методи частотної області;
- методи фазової області;
- методи на основі ехо-затримки;

- комбіновані методи.

Методи просторової області (LSB – Least Significant Bit). Найелементарнішою та найбільш розповсюдженою стратегією інтеграції даних є модифікація найменш значимих розрядів дискретних відліків аудіо. За цього підходу двійкові розряди латентного повідомлення замінюють кінцеві біти амплітудних величин сигналу. Зокрема, коли відлік характеризується значенням 10011010, то субституція останнього розряду на 1 трансформує його в 10011011, що не спричиняє аудіальних змін.

Достоїнствами технології є елементарність впровадження та значна місткість прихованого каналу, а вразливістю — недостатня резистентність до трансформацій, як-от компресія, фільтраційні процеси чи накладання перешкод.

Методи частотної області. У межах цієї категорії, інтеграція реалізується після трансформації сигналу в спектральний простір, зокрема через застосування дискретного перетворення Фур'є (DFT) чи дискретного косинусного перетворення (DCT). Прихована інформація локалізується в специфічних спектральних складових, які характеризуються зниженою помітністю для слуху, наприклад у високочастотних зонах спектра.

Означені технології демонструють підвищену резистентність до обробки, проте, відзначаються складністю імплементації та порівняно обмеженою пропускною спроможністю [11].

Методи фазової області (Phase Coding). Технології трансформації фази базуються на специфіці людської аудіальної системи, яка виявляє низьку реактивність до фазових модуляцій сигналу. Інтеграція даних реалізується через зміну фази певних фрагментів акустичної хвилі відповідно до змісту повідомлення. Для репрезентації «1» та «0» можуть застосовуватись диференційовані кутові модуляції фази, зокрема 0° та 180°.

Ключовою перевагою Phase Coding є значна прихованість і стабільність декодування, оскільки фазові модифікації практично не впливають на сприйняття якості звучання. Серед обмежень, варто відзначити меншу місткість

каналу порівняно з технологіями LSB, оскільки латентна інформація переважно локалізується виключно у початкових фрагментах сигналу [12].

Методи на основі ехо-затримки. У рамках даного підходу приховане повідомлення кодується через короткострокові темпоральні затримки сигналу, ефекти реверберації. Для репрезентації диференційованих бітів застосовуються відмінні величини затримки (зокрема, 1 мс для біта «0» та 2 мс для біта «1»). Оскільки такі модифікації залишаються майже неідентифікованими людським слухом, технологія демонструє високу ефективність та резистентність до стандартних аудіотрансформацій [13].

Основним викликом методів на основні ехо-затримки є прецизійне синхронізоване ідентифікування затримок на стадії декодування.

Комбіновані методи. З метою оптимізації ефективності та резистентності сучасні стеганографічні системи часто інтегрують кілька методологій. Наприклад, інтеграція інформації може відбуватись одночасно у фазовому та спектральному просторах, що уможливлює досягнення балансу між місткістю, латентністю та резистентністю до трансформацій [14].

Технології інтеграції інформації в аудіо характеризуються різноманітністю за механізмом функціонування та результативністю. Вибір оптимальної технології визначається встановленими завданнями і пріоритетом: максимальна резистентність, чи розширена місткість каналу, або ж мінімальна помітність модифікацій. У контексті даної роботи центральна увага концентрується на технології трансформації фази звукового сигналу (Phase Coding) як одній із найперспективніших галузей аудіостеганографії.

Метод зміни фази (Phase Coding). Техніка модифікації фазових характеристик (Phase Coding) належить до традиційних і водночас найбільш затребуваних підходів у галузі аудіостеганографії. Її фундаментальна концепція ґрунтується на експлуатації фазових параметрів звукового сигналу як носія для вбудовування конфіденційної інформації. Ключова ідея полягає у тому, що фазові параметри звукової хвилі можуть зазнавати модифікацій таким способом, аби інтегрувати інформаційні біти без генерації відчутних

акустичних артефактів у результуючому аудіофайлі [15].

Фаза сигналу характеризує часовий зсув коливань щодо точки відліку. Людська слухова система в першу чергу реагує на амплітудні та частотні властивості звукових хвиль, у той час як мінімальні варіації фазових характеристик залишаються поза межами слухового розрізнення. Саме ця особливість перцепції перетворює фазу на досконалий інструмент для імплементації прихованих каналів передачі даних. Алгоритм виконується через послідовність наступних кроків[16].

1. Сегментація вхідного аудіосигналу. Початковий звуковий матеріал піддається розподілу на фрагменти однакової протяжності (зазвичай, блоки з 1024 чи 2048 дискретних відліків). Подальша обробка здійснюється індивідуально для кожного з виділених фрагментів, що дозволяє локалізувати модифікації та забезпечити гнучкість процедури вбудовування.

2. Перетворення розділеного сигналу у частотну область. Кожен виокремлений сегмент підлягає обробці за допомогою швидкого перетворення Фур'є (FFT). Ця математична операція дає змогу репрезентувати часовий сигнал як набір синусоїдальних компонентів, кожна з яких характеризується власною амплітудою та фазою. Таке перетворення сигналу відкриває доступ до маніпуляції окремими частотними складовими без значного впливу на загальну структуру сигналу. Результат перетворення сигналу можна описати як:

$$X(k) = A(k)e^{j\phi(k)}$$

де:

- $A(k)$ – амплітуда;
- $\phi(k)$ – фаза частоти k .

3. Модифікація фазових характеристик початкового сегмента. Саме в межах першого фрагмента здійснюється імплементація повідомлення. Інформаційні біти кодуються шляхом зміни фазових кутів обмеженої кількості спектральних компонентів, переважно низькочастотних, оскільки маніпуляції у

цьому діапазоні спричиняють мінімальні, непомітні, викривлення. Вибір низьких частот обумовлений найменшою чутливістю людського слуху до фазових змін у цій спектральній області. Типова схема кодування може виглядати так:

- біт зі значенням "0" асоціюється з фазовим зміщенням 0°;
- біт зі значенням "1" відповідає фазовому зміщенню 180°.

Внаслідок цієї процедури генерується фазовий дескриптор, який містить зашифровані інформаційні біти. Бінарна опозиція (0° проти 180°) забезпечує максимальну роздільність при декодуванні та підвищує стійкість до незначних спотворень [17].

4. Узгодження фазових характеристик суміжних сегментів. Для згладжування різких фазових розривів на межах між послідовними фрагментами застосовується процедура фазового вирівнювання. Фази всіх наступних сегментів коригуються пропорційно до модифікацій, внесених у перший блок. Ця операція критично важлива для збереження акустичної цілісності результуючого сигналу та запобігання виникненню слухових артефактів на стиках сегментів.

5. Реконструкція часового сигналу через зворотне перетворення. До усієї сукупності оброблених сегментів застосовується інверсне перетворення Фур'є (IFFT). На підставі модифікованих фазових параметрів та незмінних амплітудних характеристик синтезується результуючий аудіосигнал, який містить інтегроване приховане повідомлення. Важливо зазначити, що збереження амплітудного спектру гарантує акустичну подібність з оригіналом.

Для зворотного декодування зашифрованої інформації з аудіосигналу виконується зворотна послідовність операцій [18].

1. Поділ аудіоматеріалу на ідентичні сегменти.

2. Застосування прямого перетворення Фур'є до початкового фрагмента.

3. Аналіз фазових кутів визначеної множини гармонічних складових.

4. Декодування бітів на основі детектованих фазових зміщень (0° чи

180°).

Таким чином, за умови володіння параметрами алгоритму (розмір сегмента, число задіяних гармонік, схема фазового кодування), приймач може з високою точністю реконструювати передане повідомлення. Симетричність процедур вбудовування та вилучення забезпечує точність методу.

Основні переваги методу роблять його привабливим для застосувань, де критичним є баланс між прихованістю та технічною складністю імплементації:

Прихованість. Фазові модифікації практично не впливають на суб'єктивну якість звукового матеріалу.

Стабільність даних. Вбудована інформація демонструє стійкість до типових трансформацій, таких як нормалізація амплітудних характеристик або зміна частоти дискретизації.

Концептуальна доступність. Практична реалізація базується на загальновідомих математичних апаратах спектрального аналізу (FFT/IFFT).

Обмеження визначають специфічні сценарії застосування методу та зумовлюють необхідність ретельного добору параметрів залежно від конкретних вимог до стеганографічної системи. Зокрема, обмежена місткість пояснюється скінченною кількістю спектральних компонентів, модифікація яких не призводить до помітної деградації сигналу, унаслідок чого обсяг вбудованих даних залишається порівняно невеликим. Вразливість до компресії проявляється в тому, що застосування втратних форматів стиснення (MP3, AAC, OGG) може суттєво змінювати фазову структуру сигналу та призводити до втрати прихованої інформації. Крім того, метод є вимогливим до синхронізації, оскільки навіть незначна похибка у визначенні меж сегментів значно ускладнює або робить неможливим коректне декодування прихованих даних.

Отже, техніка фазової модифікації представляє собою оптимальне рішення для імплементації компактних інформаційних масивів, коли пріоритетними цілями виступають непомітність для слухового сприйняття та стійкість до базових аудіотрансформацій, а не максимізація пропускної

спроможності стеганографічного каналу [19].

1.3. Різновиди методів Phase Coding

Різновиди Phase Coding дозволяють адаптувати метод до особливостей аудіосигналу й вимог конкретного застосування. У подальших підрозділах розглянуто принципи роботи кожного з цих підходів.

Технологія Phase Coding реалізується у кількох варіантах, що різняться за критеріями відбору фрагментів сигналу для інтеграції даних і механізмами маніпуляції фазовими параметрами. Вибір певної стратегії залежить від балансу між прихованістю, резистентністю та ємністю інформаційного каналу. Наведемо ключові варіації Phase Coding [20].

Локальне фазове кодування грунтується на вимірюванні енергетичних характеристик сигналу й інтегрує дані виключно у зони з підвищеною амплітудою. Такий підхід мінімізує аудіо артефакти, адже гучні ділянки менш сприйнятлива до фазових варіацій.

Сегментне фазове кодування поділяє звукову послідовність на темпоральні блоки (сегменти), всередині яких розміщується фрагмент повідомлення. Даний спосіб гарантує організованість, темпоральну синхронізацію й простоту екстракції.

Адаптивне фазове кодування встановлює параметри кодування в реальному часі з урахуванням властивостей сигналу, зокрема енергетичного профілю, спектральної композиції чи частотного діапазону. Це посилює ефективність і маскування за варіативних акустичних сценаріїв.

Кодування за seed-фразою застосовує криптографічний seed-ключ для визначення позицій фреймів чи фазових компонентів, у які вбудовуються інформаційні біти. Це забезпечує підвищений рівень конфіденційності й ускладнює стороннє детектування прихованих даних.

Варіації Phase Coding уможливлюють налаштування методу під специфіку аудіоматеріалу й критерії конкретної імплементації. У наступних

секціях детально описано механізми функціонування кожної з цих технік.

1.3.1. Локальне фазове кодування (Local Phase Coding)

Локалізоване фазове кодування становить одну зі стратегій модифікації фазових параметрів, за якої конфіденційна інформація інтегрується виключно у такі сегменти аудіопотоку, енергетичний рівень яких перевершує встановлене граничне значення. Подібна тактика гарантує максимальну прихованість впровадженого контенту, оскільки фазові трансформації реалізуються у високоамплітудних фрагментах, де їхнє виявлення слуховою системою залишається мінімальним. Енергетична межа застосовується індивідуально для кожного часового фрейму, що надає можливість точно ідентифікувати придатні для вбудовування частини сигналу. Реалізація локалізованого фазового кодування спирається на низку ключових параметрів [21], які визначають ефективність, прихованість і стійкість вбудовування інформації.

Величина фрейму визначає кількість семплів, що аналізуються. Цей параметр безпосередньо впливає на роздільність у часово-частотному просторі й точність фазових варіацій.

Ступінь перекриття вікон регулює міру взаємного накладання послідовних сегментів. Це забезпечує плавність переходів між суміжними блоками та мінімізує спотворення при відновленні сигналу після інверсного FFT.

Енергетичний поріг встановлює мінімальний рівень енергії ділянки, при якому можливе кодування. Забезпечує вибір тільки тих ділянок сигналу, де вставка даних буде непомітною для слуху.

Розмір аналітичного вікна визначає довжину ділянки сигналу, на базі якої визначається рішення щодо вбудовування. Визначає якість оцінювання енергетичних і спектральних властивостей.

Кількість біт на фрейм визначає обсяг інформаційних одиниць, що інтегруються в окремий блок.

Частотний діапазон, це смуга спектра, всередині якої виконуються фазові

трансформації. Дає змогу локалізувати втручання у тих секторах, де зміни приховуються мінімально або характеризуються підвищеною стабільністю до подальшої обробки.

Послідовність операцій локалізованої методики охоплює такі стадії:

- аудіосигнал сегментується на фрейми згідно з визначеним частотним діапазоном.
- для кожного блоку визначається його енергетичний показник;
- у разі перевищення порогового значення застосовується пряме перетворення Фур'є (FFT) з наступною модифікацією фазових кутів спектральних компонентів відповідно до схеми кодування;
- по завершенню фазових корекцій виконується інверсне перетворення (IFFT), а трансформовані фрейми синтезуються у результуючий сигнал із застосуванням механізму компенсації перекриття.

Отже, локальне фазове кодування є ефективним методом стеганографічного вбудовування даних у аудіосигнали, що забезпечує високу непомітність завдяки використанню лише тих фреймів, енергія яких перевищує заданий поріг. Правильний вибір параметрів дозволяє оптимізувати баланс між ємністю каналу і якістю сигналу, забезпечуючи надійне та ефективне кодування при збереженні природності звучання аудіоматеріалу.

1.3.2. Сегментне фазове кодування (Segment-based Phase Coding)

Метод сегментного фазового кодування базується на розбитті звукової послідовності на темпоральні фрагменти фіксованої протяжності, скажімо, по одній секунді кожен, з вкладенням частин повідомлення в ці фрагменти. Така стратегія уможливлює побудову організованої стеганографічної системи, де кожен фрагмент оснащений синхронізаційним ідентифікатором, який спрощує процес декодування та забезпечує стійкість до локальних пошкоджень сигналу. Реалізація сегментного фазового кодування спирається на такі ключові параметри [22]:

- тривалість фрагмента – величина сигнального блоку, що підлягає

комплексній обробці та впливає на точність синхронізації і стабільність процесу кодування;

- кількість бітів на фрагмент – число інформаційних одиниць, інтегрованих в окремий фрагмент, що визначає пропускну здатність прихованого каналу передавання даних;
- коефіцієнт накладання фрагментів – відсоток часткового перекриття суміжних блоків, який забезпечує плавність фазових трансформацій і підвищує надійність кодування;
- синхропослідовність – унікальна бітова структура для маркування початку фрагмента, що використовується з метою точної локалізації меж сегментів і коректного декодування інформації;
- величина фрейму – параметр, аналогічний відповідному показнику локального фазового кодування, який визначає кількість семплів у межах одного аналізованого блока.

Для виконання сегментного фазового кодування використовується наступний алгоритм.

1. Розділення аудіосигналу на сегменти. Сигнал розбивається на часові сегменти заданої тривалості з урахуванням перекриття між сусідніми сегментами.
2. Інтеграція синхропослідовності. На початковій позиції кожного блоку розміщується унікальна бітова послідовність, призначена для точної синхронізації та ідентифікації меж сегмента під час відновлення даних.
3. Декомпозиція сегмента на фрейми. Кожен блок піддається подальшому поділу на дрібніші фрейми згідно з обраним розміром для реалізації фазових трансформацій.
4. Вбудовування повідомлення. Біти конфіденційного повідомлення інтегруються у фазові характеристики окремих фреймів сегмента.
5. Збірка сегментів у сигнал. Після завершення фазових модифікацій усі блоки об’єднуються в єдиний аудіопотік з урахуванням накладання для забезпечення плавних переходів між фрагментами.

Отже, сегментний підхід до фазового кодування створює структуровану інтеграцію прихованої інформації в аудіосигнал, гарантуючи надійну синхронізацію та високу резистентність до локальних деформацій сигналу. Варіативність ключових параметрів надає можливості гнучкої оптимізації співвідношення між пропускною спроможністю прихованого каналу та акустичною якістю результату, забезпечуючи ефективне та стійке кодування при збереженні природної звукової текстури аудіоматеріалу.

1.3.3. Адаптивне фазове кодування (Adaptive Phase Coding)

Адаптивна техніка фазової модуляції принципово відмінна від локалізованого та сегментованого підходів завдяки автоматичному калібруванню параметрів відповідно до властивостей звукового матеріалу [23]:

- енергетичного профілю;
- спектральної структур;
- частотної динаміки.

Така методологія оптимізує процес інтеграції прихованого повідомлення з одночасним збереженням акустичної якості та підвищенням резистентності до деформацій і компресії. Ключовими параметрами адаптивної методики є:

- енергетичний рівень фрейму – показник, що використовується для визначення фрагментів сигналу, оптимальних для кодування; сегменти з високою енергетичною насиченістю обираються як носії даних;
- спектральна композиція – характеристика сигналу, яка визначає частотні зони, у межах яких фазові трансформації залишаються непомітними для слухового сприйняття;
- динамічний об’єм бітів на фрейм – кількість бітів, що інтегруються у фрейм та встановлюються на основі локальних параметрів сигналу, забезпечуючи оптимальне співвідношення між пропускною здатністю і звуковою природністю;
- розмір аналітичного вікна – параметр, що налаштовується динамічно з метою точного визначення енергетичних і спектральних властивостей

сигналу;

- порогові значення адаптивного вибору – сукупність параметрів, згідно з якими фрейм або сегмент кваліфікується як придатний для кодування.

Алгоритм адаптивного фазового кодування:

- характеристичний аналіз сигналу – проводиться оцінювання енергетичних і спектральних параметрів кожного фрейму або сегмента звукового потоку.
- вибір оптимальних фреймів/сегменті – ідентифікуються ділянки сигналу, що задовольняють пороговим критеріям енергії та спектру для безпечної інтеграції інформації.
- адаптивне визначення кількості біт на фрейм – для кожного відібраного фрейма обчислюється оптимальний обсяг біт для інтеграції.
- впровадження фазових трансформацій – модифікуються фазові характеристики обраних компонентів згідно з вибраною схемою кодування та адаптивним бітовим обсягом.
- реконструкція цілісного сигналу – після впровадження змін, усі фрейми об’єднуються, формуючи завершений стеганографічний аудіоматеріал.

Адаптивна фазова модуляція гарантує максимальну прихованість і функціональну ефективність завдяки динамічному налаштуванню параметрів відповідно до характеристик звукового матеріалу. Такий підхід забезпечує оптимальну рівновагу між пропускною спроможністю стеганографічного каналу та акустичною якістю, роблячи методику універсальною для різноманітних аудіоформатів і резистентною до різноманітних форм обробки та компресійних алгоритмів.

1.3.4. Фазове кодування за seed-фразою (Seed-based Phase Coding)

Методика фазового кодування на основі seed-ключа ґрунтується на формуванні псевдовипадкової черги блоків або фрагментів, призначених для інтеграції прихованого повідомлення, використовуючи вихідний seed-ключ як базу. Такий механізм суттєво підвищує захищеність процесу кодування, адже

без доступу до seed-ключа реконструювати структуру вставки даних неможливо. Основні параметри даного підходу [24]:

- Seed-ключ, який слугує базовою основою для генерування псевдовипадкової послідовності та гарантує відтворюваність процесу і захист від неавторизованого втручання;
- величина фрейму, що визначається аналогічно до параметра, описаного для локального фазового кодування;
- кількість бітів на фрейм, яка встановлюється відповідно до принципів, наведених у описі локального фазового кодування;
- тривалість фрагмента, яка задає розмір порцій сигналу для псевдовипадкового визначення позицій інтеграції інформації.

Алгоритм функціонування фазового кодування за seed-фразою включає такі етапи:

- створення псевдовипадкової послідовності на основі seed-ключа формується порядок блоків або фрагментів, призначених для кодування;
- відбір блоків (фрагментів) відповідно до сформованої псевдовипадкової послідовності ідентифікуються ділянки сигналу для інтеграції бітів повідомлення;
- реалізація фазових трансформацій у відібраних блоках або фрагментах здійснюється модифікація фазових параметрів гармонічних компонентів з метою кодування бітів повідомлення;
- об’єднання сигналу модифіковані блоки або фрагменти інтегруються у фінальний аудіопотік із вбудованим прихованим повідомленням.

Фазове кодування на основі seed-ключа гарантує високий рівень захищеності та прихованості інтегрованої інформації, оскільки структура кодування базується на псевдовипадковій черзі. Підхід дозволяє ефективно поєднувати стеганографічні характеристики з інформаційним захистом, що робить його доцільним для застосування у сценаріях, де критичною є конфіденційність даних.

1.3.5. Висновки аналізу методів Phase Coding

Досліджені техніки фазової модуляції ілюструють широкий спектр стратегій для стеганографічної інтеграції повідомлень в аудіопотоки. Результат дослідження відображено в таблиці 1 «Порівняння методів Phase Coding» [25].

Таблиця 1 . Порівняння методів Phase Coding

Метод	Основна ідея	Переваги	Недоліки
Класичне	Кодування фази всього сигналу фрейм за фреймом	Просте, стабільне декодування	Менше прихованих сегментів, помітне при слабкому сигналі
Локальне	Кодування тільки у сильних фреймах (поріг енергії)	Менше артефактів у тихих ділянках, адаптивність	Потрібно налаштовувати поріг енергії, складніше реалізувати
Сегментне	Розбиття сигналу на сегменти і кодування всередині сегмента	Краща прихованість у часі, можна кодувати лише важливі сегменти	Більш складне синхронне декодування, вимагає точного сегментування
Адаптивне кодування	Параметри кодування змінюються в залежності від характеристик сигналу	Оптимізація якості звуку і стійкості декодування	Складна реалізація, потребує додаткового аналізу сигналу
Кодування за сід-фразою	Використання seed-фрази для визначення фазових змін	Захищене від стороннього декодування.	Потрібно точно зберігати seed, складніше синхронізувати

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ДОДАТКУ ДЛЯ РОБОТИ З PHASE CODING

2.1. Постановка задачі, призначення та вимоги до програмного засобу

Програмний засіб повинен надавати широкі можливості в роботі з Phase Coding. Зокрема, забезпечувати можливість повноцінного життєвого циклу кодування і декодування з варіаціями параметрів. Також, варто зазначити, що повноцінна та швидка робота з повинна виконуватись навіть з аудіофайлами великого розміру, а також інтеграцію алгоритмів обробки сигналів. До основних вимог до додатку можна віднести:

- зручність (інтерфейс повинен бути інтуїтивним, з чіткою диференціацією між режимами «Кодувати», «Декодувати» та різних режимів роботи додатку);
- швидкодія (обробка сигналів повинна виконуватись у межах декількох секунд навіть для тривалих аудіозаписів);
- надійність (застосунок повинен стабільно працювати з *.WAV файлами різної частоти дискретизації та числа каналів);
- модульність (логіка кодування, декодування та візуалізації має бути реалізована у вигляді окремих функціональних блоків);
- візуалізація (користувач повинен мати доступ до перегляду часової форми сигналу, частотного спектру та спектрограми);
- масштабованість (рхітектура має дозволяти безперешкодне додавання нових методів вбудовування без реструктуризації основного коду);
- доступність (додаток має працювати на різних операційних системах Windows 10/11, Linux, MacOS).

До основних функцій, що повинні бути реалізовано можна віднести:

- імпорт аудіофайлів формату *.wav із відображенням їхніх базових

параметрів;

- вибір режиму: кодування або декодування;
- підтримка модифікацій методу Phase Coding:
    1. стандартне кодування і декодування;
    2. локалізоване кодування і декодування;
    3. сегментоване кодування і декодування;
    4. адаптивне кодування і декодування;
    5. кодування і декодування на базі seed-ключа.
- введення та збереження текстового повідомлення для інтеграції в аудіосигнал;
- конфігурація параметрів кодування і декодування в залежності від обраного методу;
- програвання вихідного та трансформованого аудіосигналів для аналізу і порівняння.

2.2. Вибір засобу реалізації програмного застосунку

У процесі розробки стеганографічної системи на основі методу Phase Coding було проаналізовано кілька підходів до програмної реалізації, які різняться інструментарієм, технічними характеристиками та складністю впровадження.

Першим із розглянутих варіантів стало застосування мови C++ разом із фреймворком Qt. Цей підхід часто використовується для побудови настільних додатків, що потребують інтенсивної обробки сигналів у режимі реального часу. C++ забезпечує прямий контроль над апаратними ресурсами й пам'яттю, що уможливлює ефективну імплементацію комплексних алгоритмів цифрової обробки сигналів. Фреймворк Qt, у свою чергу, пропонує потужні інструменти для побудови інтерфейсів, керування потоками, взаємодії з файловою системою та мультимедійними компонентами.

Незважаючи на високу швидкодію, робота з C++ вимагає суттєвих

часових ресурсів на розробку й налагодження, а також ускладнює супровід та модернізацію системи. Додатково, для досягнення кросплатформної сумісності необхідна окрема компіляція під кожну цільову операційну систему.

Альтернативою може стати веб-застосунок, що функціонує безпосередньо у веб-браузері. Сучасний стек веб-технологій — HTML5, CSS3 і JavaScript — надає достатню гнучкість для створення динамічних користувацьких інтерфейсів. Для маніпуляції звуковими даними доцільно застосувати Web Audio API, який уможливлює базові операції з аудіопотоками: завантаження, відтворення, фільтрацію, аналіз частотного складу, тощо.

Головною перевагою є відсутність необхідності у локальному встановленні програмного забезпечення, що спрощує доступ кінцевих користувачів. Веб-платформа також є універсальною, оскільки застосунок працюватиме на будь-якому пристрої з актуальним браузером.

Одним з напрямів є створення мобільного застосунку під платформи Android чи iOS. Розробка передбачає використання середовищ Android Studio (Java/Kotlin) або Xcode (Swift). Мобільна версія дала б змогу користувачам виконувати кодування та декодування аудіоповідомлень просто зі смартфона або планшета, що особливо зручно для оперативної комунікації.

Проте такий підхід має свої обмеження: знижена обчислювальна потужність браузерів і обмежений доступ до локальної файлової системи. Це створює труднощі при обробці масивних аудіофайлів та реалізації складних математичних методів стеганографії, які потребують швидких обчислень із залученням спеціалізованих бібліотек.

Для втілення програмного рішення було вибрано мову Python, яка вирізняється зрозумілим синтаксисом, багатим набором бібліотек для наукових розрахунків і великою спільнотою фахівців. Python є оптимальним інструментом для задач, пов'язаних із цифровою обробкою сигналів, машинним навчанням, створенням графічних інтерфейсів та візуалізацією даних.

До основних переваг Python належать кросплатформеність, що дозволяє запускати додатки на Windows, Linux і macOS без суттєвих змін у коді.

Широкий асортимент бібліотек, зокрема numpy, scipy, soundfile та matplotlib, забезпечує готові рішення для обробки аудіо, прискорює виконання математичних операцій та побудову графіків. Висока швидкість розробки зумовлена інтуїтивним синтаксисом та модульною архітектурою, що полегшує оперативне написання коду та інтеграцію нового функціоналу. Розвинена екосистема включає велику базу прикладів, документації та відкритих проєктів, що суттєво полегшує процес розробки.

До сильних сторін мобільного варіанта належать висока доступність, зручність у використанні та тісна інтеграція із системними можливостями девайсу. Водночас обмежені апаратні ресурси (RAM, CPU) і складність роботи з великими WAV-файлами зменшують придатність мобільного рішення для повноцінного дослідження методу Phase Coding. Крім того, оптимізація алгоритмів ЦОС під мобільні платформи вимагає додаткових зусиль, що подовжує цикл розробки.

Для побудови графічного інтерфейсу користувача обрано бібліотеку PyQt5, що є Python оболонкою над фреймворком Qt. Вона надає можливість створення сучасних, інтуїтивних і багатофункціональних графічних інтерфейсів. До основних інструментів Python, які будуть використовуватись:

- інтеграція з Qt Designer, який дозволяє швидко проєктувати вікна й елементи інтерфейсу через візуальний редактор;
- механізм зв'язування елементів інтерфейсу з бізнес-логікою через сигнали;
- повноцінна підтримка сучасних компонентів управління, як-от кнопка (QPushButton), слайдер (QSlider), поле (QLineEdit), числове поле (QSpinBox), тощо;
- вбудовані можливості для роботи з графікою, медіа та обробкою подій;
- гнучка стилізація зовнішнього вигляду через CSS-подібні таблиці стилів (Qt Style Sheets), що дає можливість швидко створити професійний дизайн для програми.

Окрім PyQt5, для виконання роботи буде використовуватись наступні

бібліотеки:

- Wave, використовується для програвання *.wav файлів. Дозволяє керувати гучністю та часом програвання звуку;
- Numpy, використовується для роботи з масивами даних, виконання FFT;
- Matplotlib, класичний інструмент для побудови графіків. В даній роботі використовується для побудови спектрограми і водоспаду.

2.3. Загальний опис проєкту

Програмна архітектура додатку ґрунтується на модульній побудові, що гарантує адаптивність, масштабованість і легкість супроводу коду. У створеному додатку можна визначити декілька ключових підсистем, які комунікують одна з одною через чіткі програмні інтерфейси.

Модуль графічної оболонки створено із застосуванням бібліотеки PyQt5, що надає можливість формування сучасних елементів управління. Інтерфейс включає наступні ключові елементи: Окрім PyQt5, для виконання роботи буде використовуватись наступні бібліотеки.

Панель імпорту звукового файлу формату *.wav, на якій демонструються базові параметри сигналу:

- частота дискретизації;
- кількість каналів;
- кількість семплів;
- ширина семплу;
- тривалість.

Елементи вибору режиму роботи програми:

- декодувати;
- закодувати.

Елементи вибору методу Phase Coding:

- класичне;
- локальне;

- сегментне;
- адаптивне;
- за seed фразою.

Поля для введення налаштувань, що змінюються в залежності від обраного режиму та методу.

Блок керування програванням звуку:

- слайдер медіапрогравача;
- керування програванням;
- керування звуком програвання.

В свою чергу, компонент цифрової обробки втілює центральну функціональність здійснення стеганографічних операцій вбудовування та вилучення інформації. До його складу входять:

- компонент читання сигналу;
- компонент трансформації сигналу;
- компонент кодування та декодування сигналу для кожного доступного в додатку метода;
- компонент фазової модуляції;
- компонент збереження модифікованого сигналу.

2.4. Створення графічного інтерфейсу програмного додатку

Побудову програмного застосунку вирішено було почати з створення користувацького інтерфейсу. Для цього, було використано окремий застосунок Qt Designer.

Спершу, було створено базову, MVP, версію інтерфейсу, для того, щоб зрозуміти його основні недоліки та спробувати в використанні під час розробки.

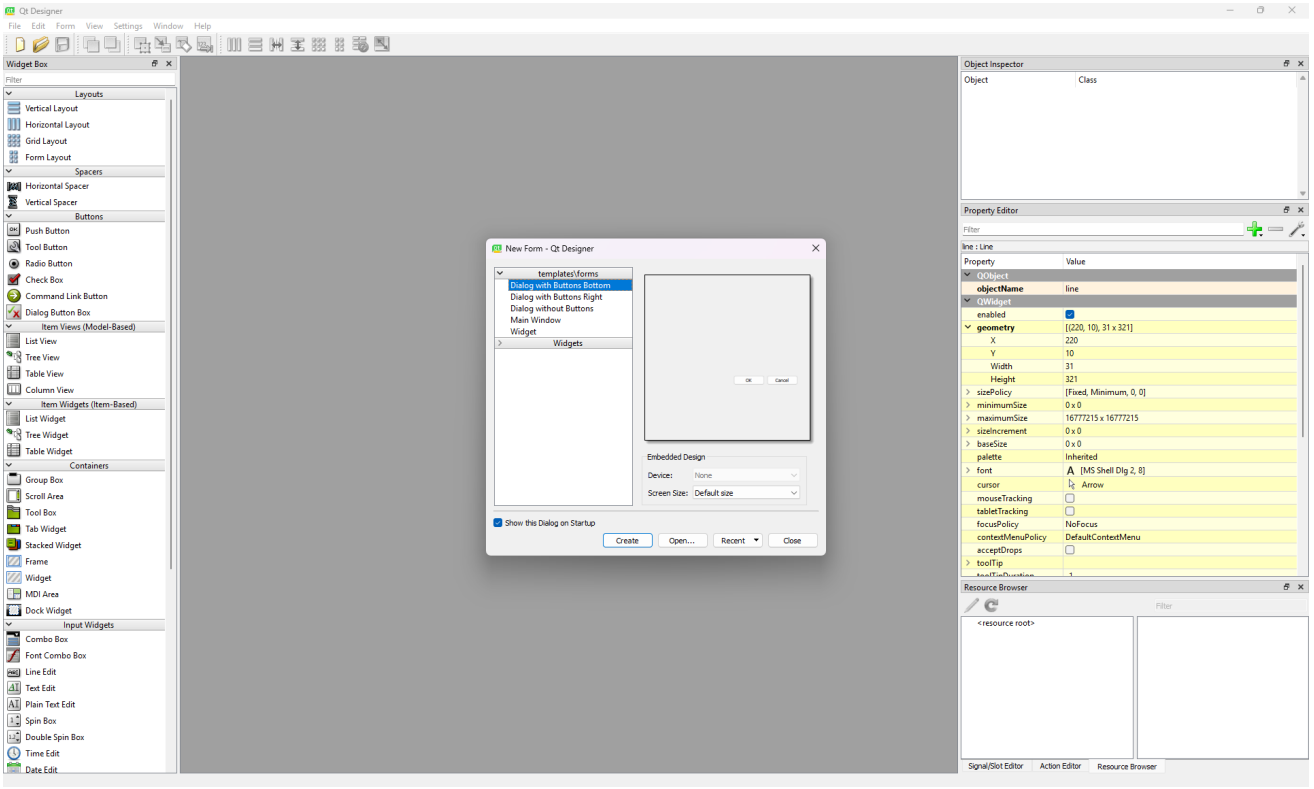

Рисунок 2.1. Інтерфейс Qt Designer

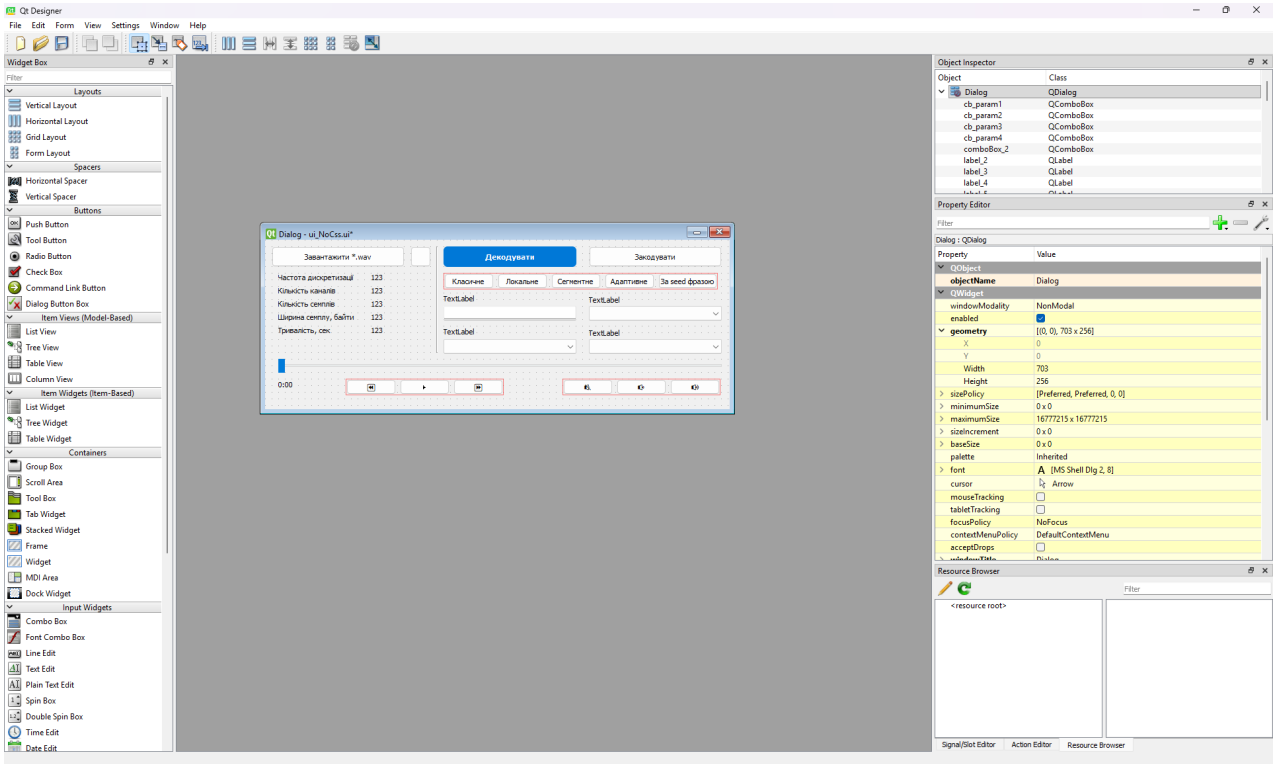

Рисунок 2.2. Перша версія інтерфейсу

При створенні додатку, активно використовувалась функція попереднього перегляду інтерфейсу в Qt Designer. Для її активації було використано сполучення клавіш «Cntrl» та «R».

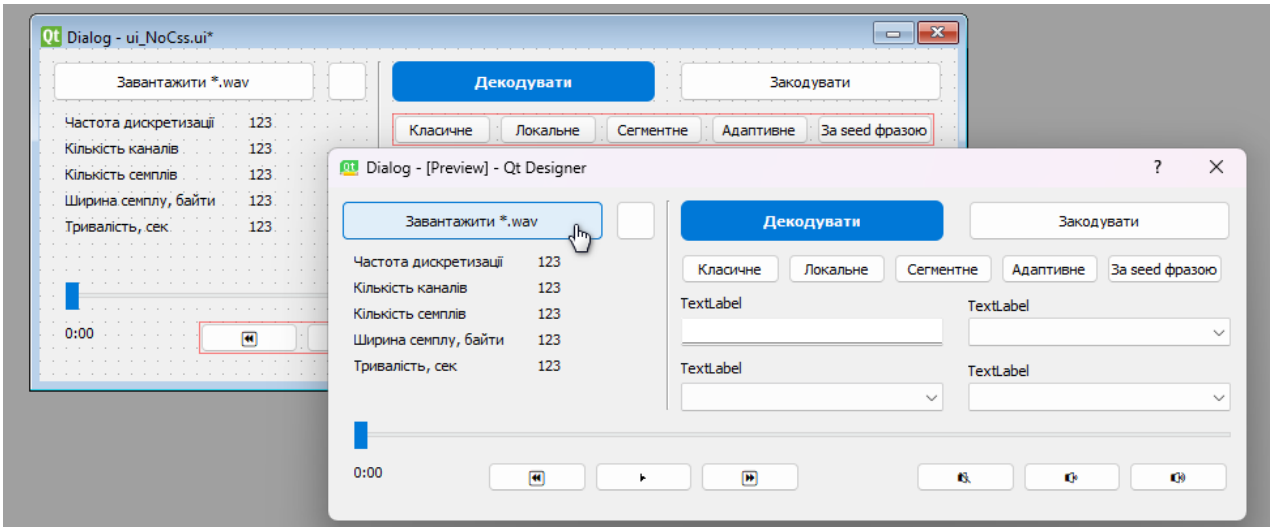

Рисунок 2.3. Режим попереднього перегляду інтерфейсу

Після створення інтерфейсу було реалізовано основні функції інтерфейсу, після чого, додаток було стилізовано та модифіковано для зручного використання.

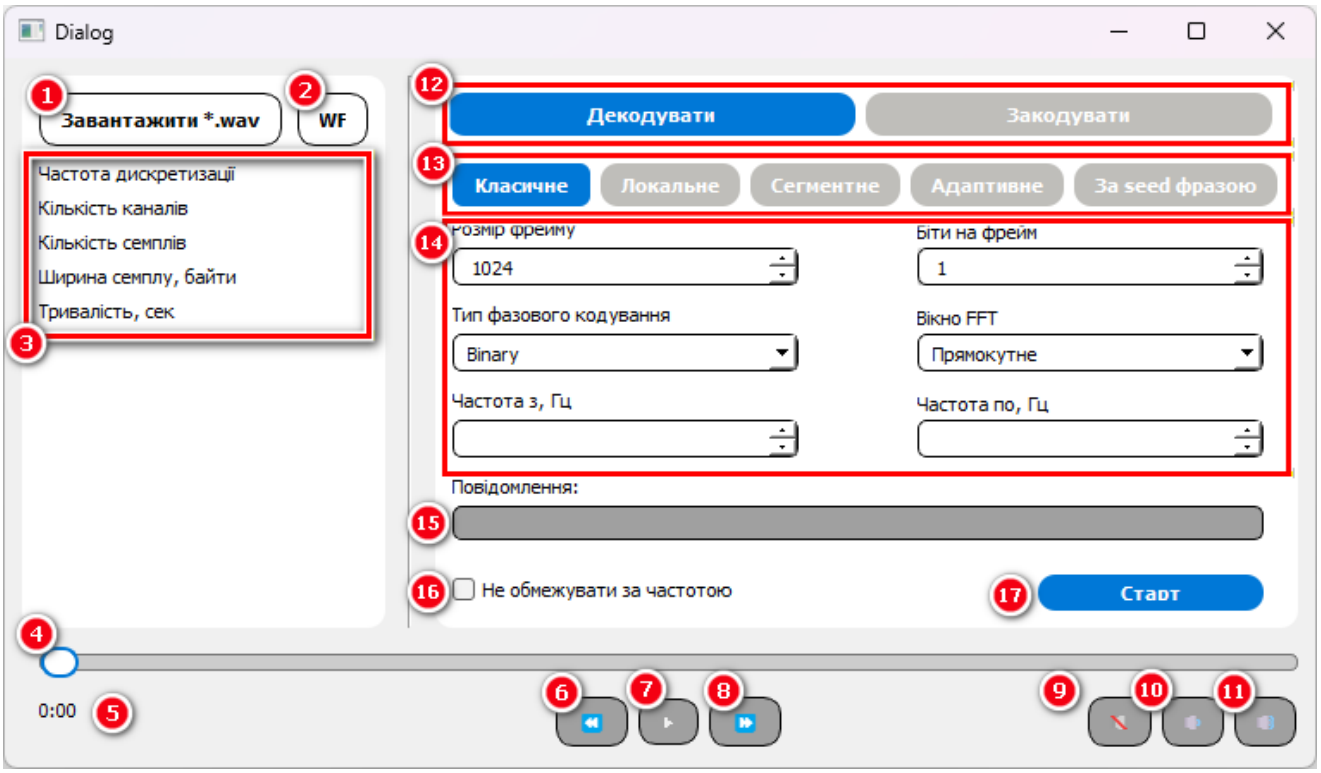

Рисунок 2.4. Фінальний інтерфейс програмного додатку

Фінальна версія програмного застосунку складається з 17 основних елементів.

1. Кнопка «Завантажити *.wav». При натисканні, відкривається стандартний інструмент операційної системи для роботи з файловою системою (наприклад, провідник).

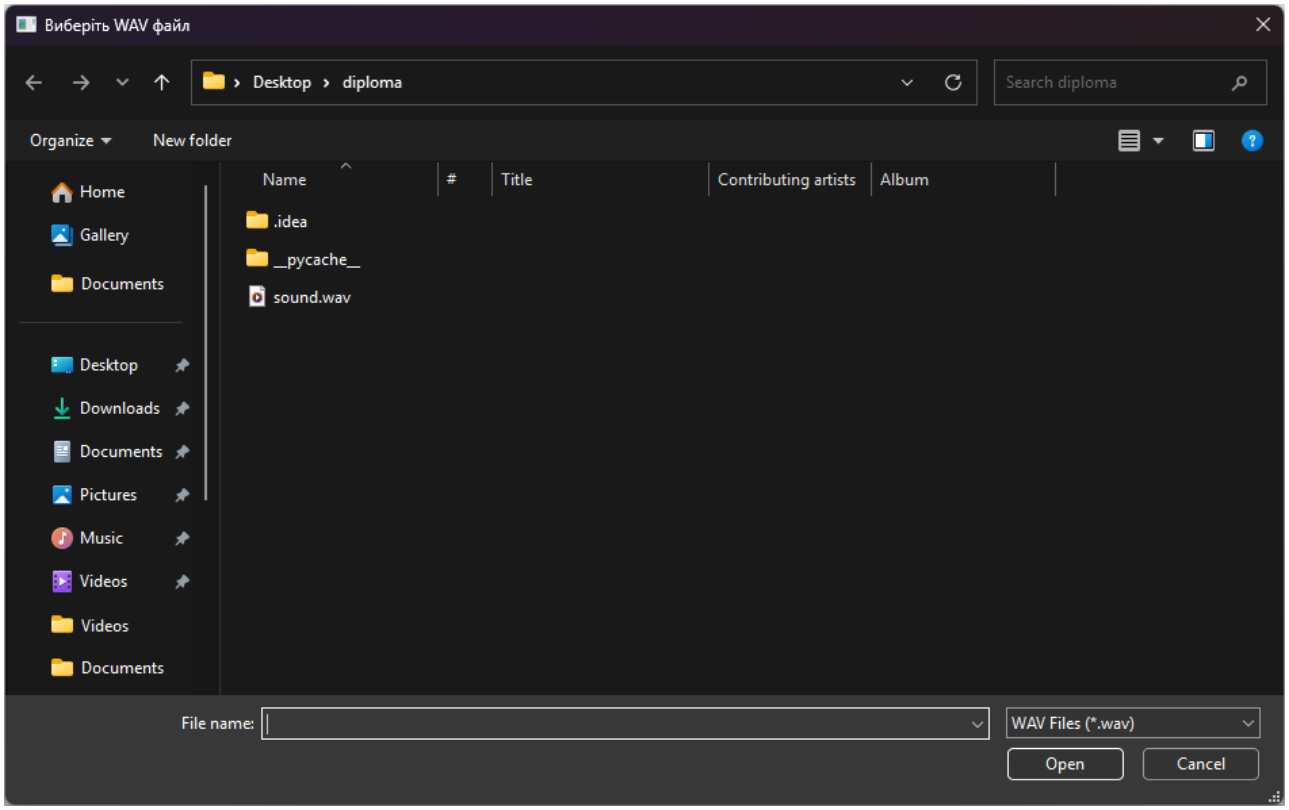

Рисунок 2.5. Вибір *.wav файлу

До вибору доступні тільки файли формату *.wav, оскільки програмний застосунок наразі підтримує тільки такий формат. Після успішного вибору файлу, його буде завантажено та отримано основні характеристики сигналу. Після завантаження, назва кнопки «Завантажити *.wav» зміниться на шлях до завантаженого файлу в системі (за потреби, цей шлях буде обрізано), а параметри сигналу будуть заповнені значеннями.

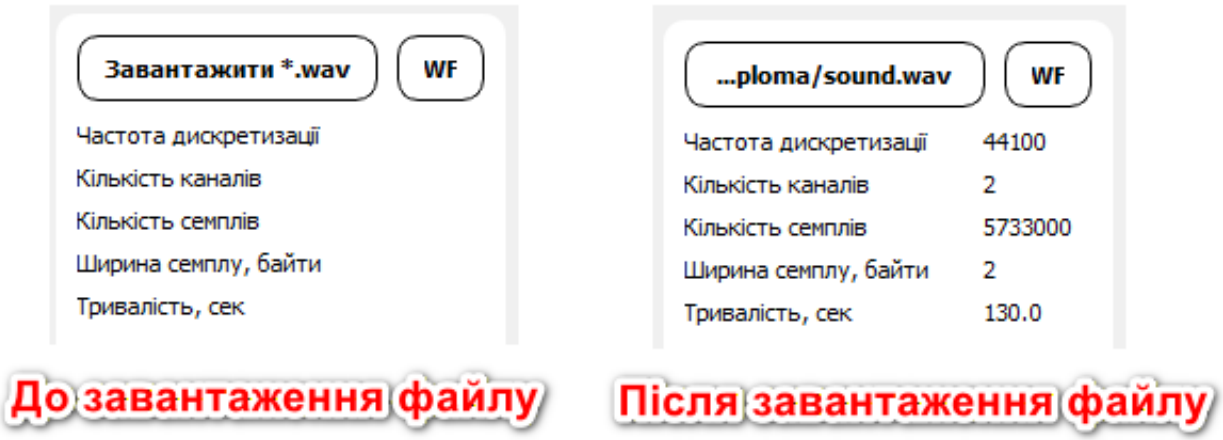

Рисунок 2.6. Зміни в інтерфейсі після завантаження файлу

2. Кнопка «WF». Кнопка доступна до натискання, тільки за умови завантаженого файлу (через кнопку «Завантажити *.wav»). При натисканні, відкривається вікно для порівняння водоспадів двох сигналів. Перший з водоспадів автоматично формується відповідно до поточно завантаженого файлу. Дане вікно є окремим та може бути відкрито скільки завгодно раз.

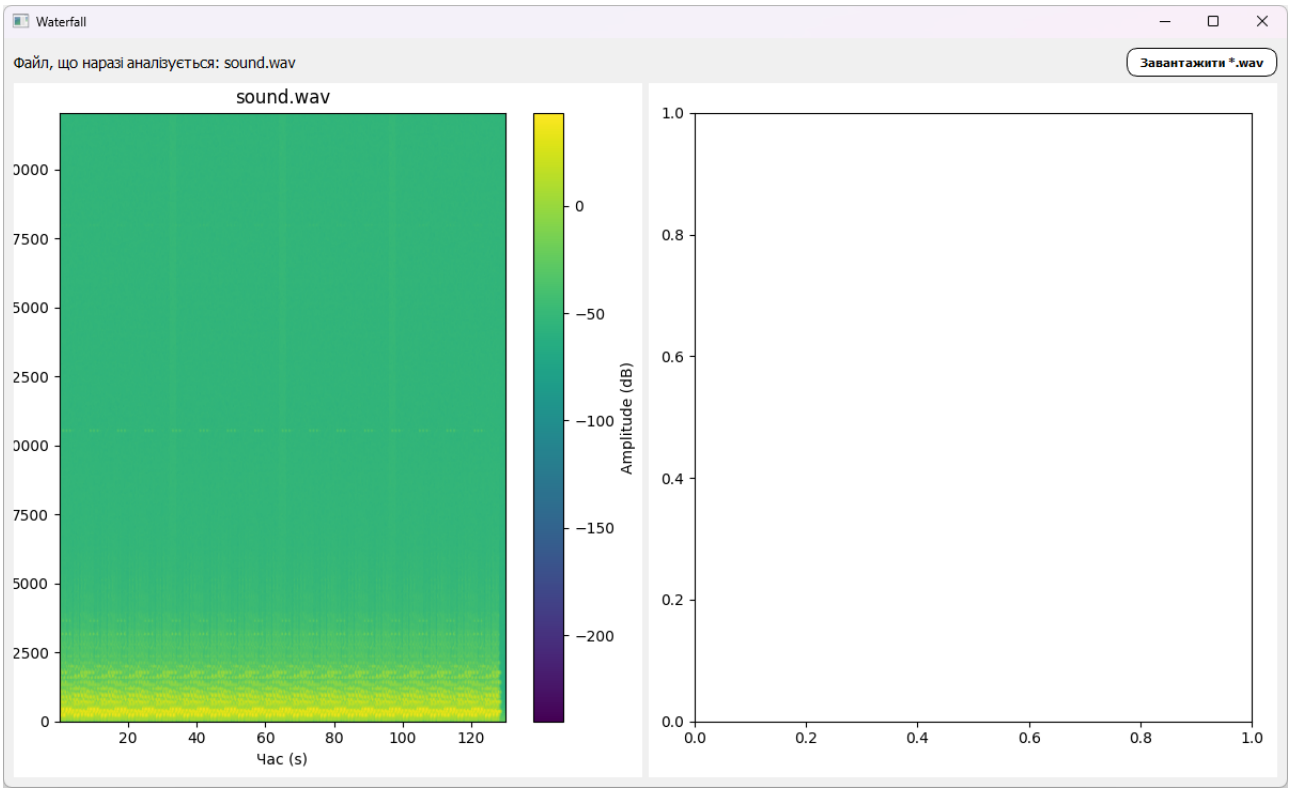

Рисунок 2.7. Вікно «Водоспад»

Сторінка складається з :

- рядка, що визначає назву поточного файлу, що аналізується;
- кнопки «Завантажити *.wav» (логіка роботи кнопки аналогічна до однойменної кнопки «Завантажити *.wav» на головному екрані; після успішного завантаження файлу, другий водоспад буде намальовано);
- два графіки з водоспадом (дівий графік презентує водоспад того сигналу, що наразі аналізується та будується автоматично, правий графік будується після того, як користувач завантажить файл).

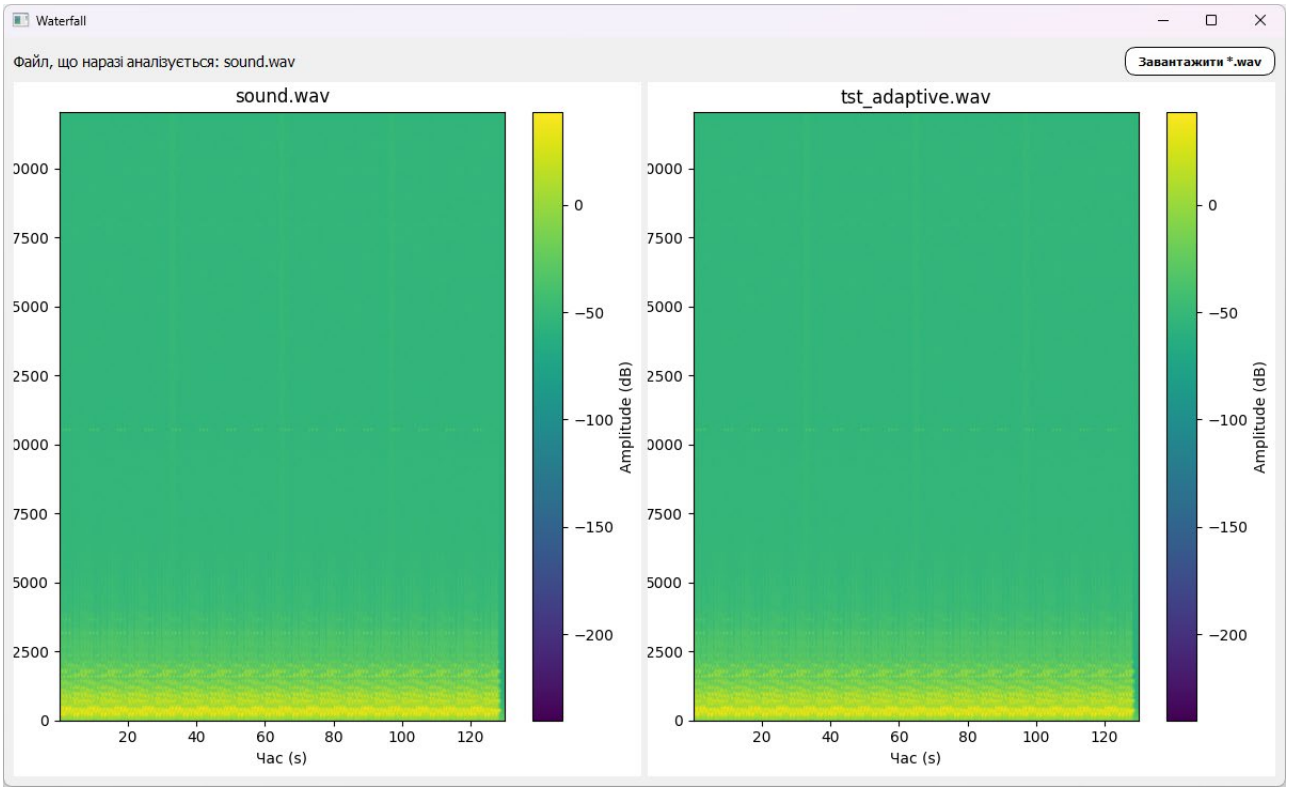

Рисунок 2.8. Порівняння двох водоспадів (зліва – оригінальний, справа – водоспад, після адаптивного кодування phase coding)

3. Параметри сигналу. Відображає параметри сигналу, що наразі аналізується. При запуску програми, параметри відсутні, оскільки сигнал не обрано. Після вибору сигналу за допомогою кнопки «Завантажити *.wav», значення заповнюються (приклад на рисунку 2.6). До параметрів сигналу входять:

- частота дискретизації;

- кількість каналів;
- кількість семплів;
- ширина семплу, в байтах;
- тривалість, в секундах.

4. Індикатор прогресу відтворення аудіо. При запуску програми, індикатор не доступний до редагування. Після завантаження аудіо, користувач, перетягуючи вказівник індикатора, може змінити час програвання звуку. При програванні звуку, індикатор прогресу відтворення аудіо поступово заповнюється. При перемотці – відповідно змінює своє положення.

5. Часовий індикатор прогресу відтворення аудіо. Числовий показник того, яка частина аудіофайлу була прослухана в форматі «<MM>: <SS> / <MM>: SS». При запуску програмного додатку, значення тексту рівне «0:00». Після завантаження, формат змінюється на зазначений вище.

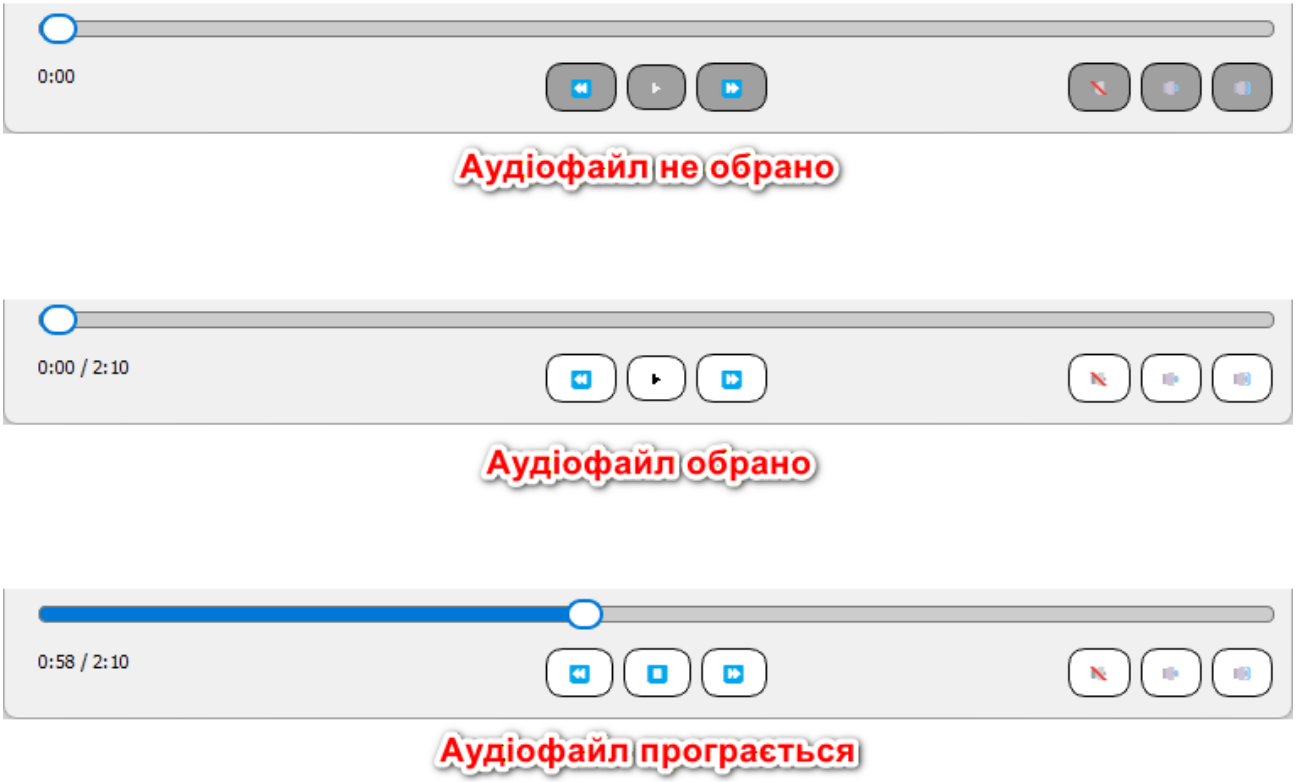

Рисунок 2.9. Приклад роботи індикаторів прогресу програвання аудіо

6. Кнопка «Крок назад». При запуску програми, кнопка не доступна до натискання. Після завантаження аудіо, кнопка стає доступною до

натискання, при якому час програвання аудіо змінюється на 10 секунд назад.

7. Кнопка «Пуск» («Пауза»). При запуску програми, кнопка не доступна до натискання. Після завантаження аудіо, кнопка стає доступною до натискання, при якому аудіо починає програватись і іконка кнопки змінюється на «Пауза» (відображено на рисунку 2.9). При повторному натисканні, аудіо зупиняється. Під час того, як аудіо програється, індикатори програвання аудіо (4, 5) відповідно змінюються.

8. Кнопка «Крок вперед». При запуску програми, кнопка не доступна до натискання. Після завантаження аудіо, кнопка стає доступною до натискання, при якому час програвання аудіо змінюється на 10 секунд вперед.

9. Кнопка «Вимкнути звук» («Ввімкнути звук»). При запуску програми, кнопка не доступна до натискання. Після завантаження аудіо, кнопка стає доступною до натискання, при якому звук аудіо вимикається. При повторному натисканні,

10. Кнопка «Зменшити звук». При запуску програми, кнопка не доступна до натискання. Після завантаження аудіо, кнопка стає доступною до натискання, при якому гучність відтворюваного аудіо зменшується на 10%.

11. Кнопка «Збільшити звук». При запуску програми, кнопка не доступна до натискання. Після завантаження аудіо, кнопка стає доступною до натискання, при якому гучність відтворюваного аудіо збільшується на 10%.

12. Вибір режиму роботи програми. До вибору можливо:

- Декодувати (режим для розшифровки повідомлення в середині звукового сигналу. В даному режимі поле «Повідомлення» не доступне для вводу);
- Закодувати (режим для зашифровки повідомлення в аудіофайлі. При виборі даного режиму поле «Повідомлення» є обов’язковим до заповнення).

13. Вибір методу роботи програми. В залежності від вибору режиму роботи параметри операції (14) буде змінено. До вибору можливо класичний, локальний, сегментний, адаптивний та за seed фразою режими.

Класичний. Параметри, визначені в цьому режимі:

- Розмір фрейму (доступні лише числа від 2 до 8192, причому кінцеве значення має бути парним);
- Біти на фрейм (доступні лише числа від 1 до 8);
- Тип фазового кодування (вибір здійснюється зі списку доступних значень: бінарна, квадратурна, багаторівнева (2), багаторівнева (3), багаторівнева (4), багаторівнева (5));
- Вікно FFT з випадним списком з вибором значень: прямокутне, Хеммінга, Ханна;
- Частота з, Гц (до вводу доступні тільки цифри і значення поля повинно бути в межах від 0 до частоти дискретизації, діленої на 2);
- Частота по, Гц (до вводу доступні тільки цифри, значення поля повинно бути в межах від 0 до частоти дискретизації, діленої на 2 і значення повинно бути меншим, за значення в полі «Частота з, Гц).

Локальний. Параметри, визначені в цьому режимі:

- Розмір фрейму (аналогічно до режиму «Класичний»);
- Біти на фрейм (аналогічно до режиму «Класичний»);
- Поріг енергії, % (до вводу доступні тільки числа і значення поля повинно бути в межах від 1 до 100);
- Довжина вікна аналізу (до вводу доступні тільки числа і значення поля повинно бути в межах від 10 до 500);
- Частота з, Гц (аналогічно до режиму «Класичний»);
- Частота по, Гц (Аналогічно до режиму «Класичний»).

Сегментний. Параметри, визначені в цьому режимі:

- Тривалість сегменту, мс. (до вводу доступні тільки числа і значення поля повинно бути в межах від 10 до 10 000);
- Біти на сегмент (аналогічно полю «Біти на фрейм» до режиму «Класичний»);
- Перекриття, % (до вводу доступні тільки числа і значення поля повинно

бути в межах від 0 до 100);

- Синхропослідовність (до вводу доступні тільки латинські літери, кількість символів в полі повинно бути від 1 до 16);
- Розмір фрейму (аналогічно до режиму «Класичний»).

Адаптивний. Параметри, визначені в цьому режимі:

- Тип адаптації (випадний список з вибором значень: За енергією, За частотою, За ентропією);
- Біти на блок (аналогічно полю «Біти на фрейм» до режиму «Класичний»);
- Чутливість адаптації, % (до вводу доступні тільки числа. Значення поля повинно бути в межах від 1 до 100);
- Частота з, Гц (аналогічно до режиму «Класичний»);
- Частота по, Гц (аналогічно до режиму «Класичний»);

За seed фразою. Параметри, визначені в цьому режимі:

- Seed фраза (до вводу доступні тільки латинські літери, кількість символів в полі повинно бути від 1 до 16);
- Алгоритм PRNG (випадний список з вибором значень: Mersenne Twister, XORShift, ChaCha2);
- Перемішування частот (випадний список з вибором значень: Так, Ні.);
- За ентропією.
- Випадковий порядок сегментів (ипадний список з вибором значень: Так,Ні);
- Розмір фрейму (аналогічно полю в режимі «Класичний»).

14. Параметри режиму. Змінюються відповідно до вибору режиму.

15. Поле «Повідомлення». Визначає повідомлення, що повинно бути закодовано в аудіосигнал. Не доступне до редагування, якщо обрано режим «Декодувати».

Після успішного декодування, в полі буде зазначено декодоване

повідомлення.

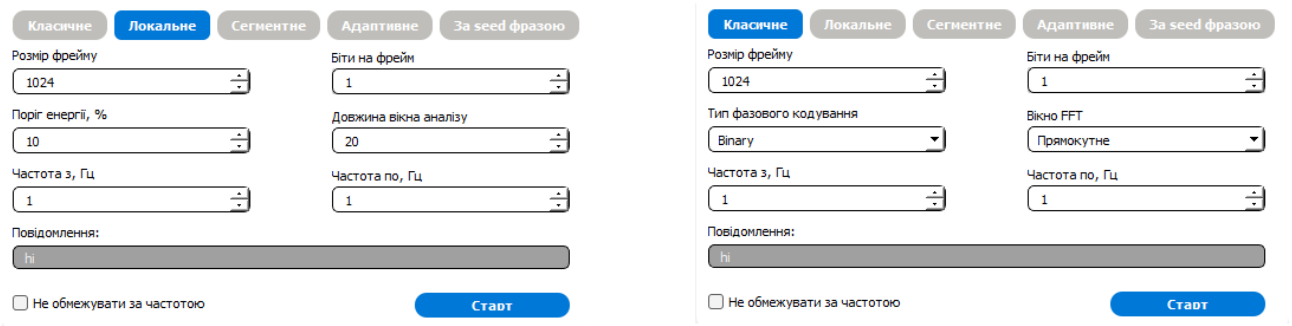

Рисунок 2. 10. Приклад зміни параметрів в залежності від обраного режиму

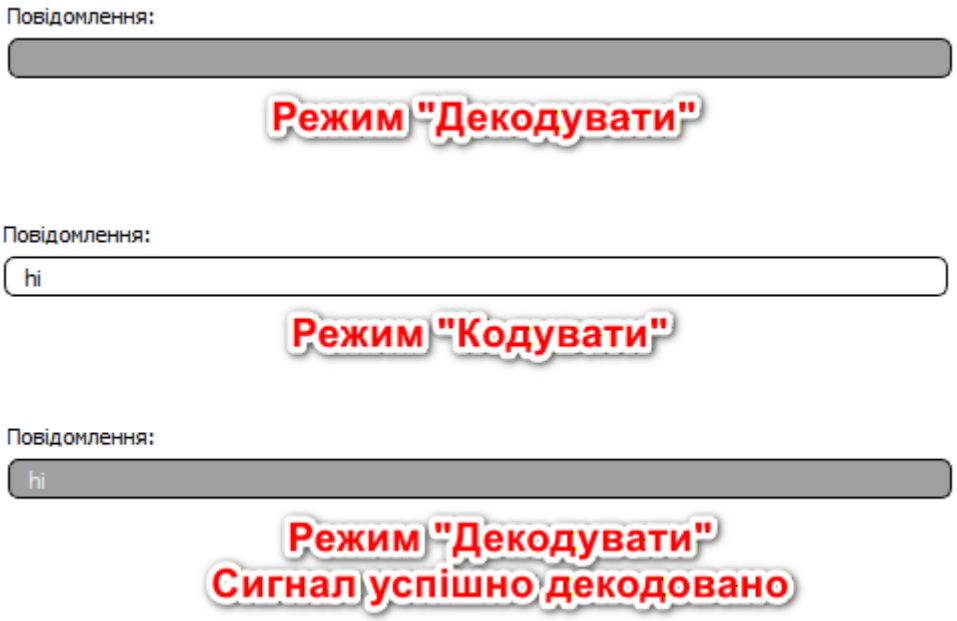

Рисунок 2.11. Поле «Повідомлення» в залежності від режиму роботи програми

16. Прапорець «Не обмежувати за частотою». Якщо прапорець активовано, поля «Частота з, Гц» і «Частота по, Гц» стають недоступними до редагування. Поле відображається тільки в тих режимах роботи програми, де присутні поля «Частота з, Гц» і «Частота по, Гц», а саме:

- Класичний;
- Локальний;
- Адаптивний.

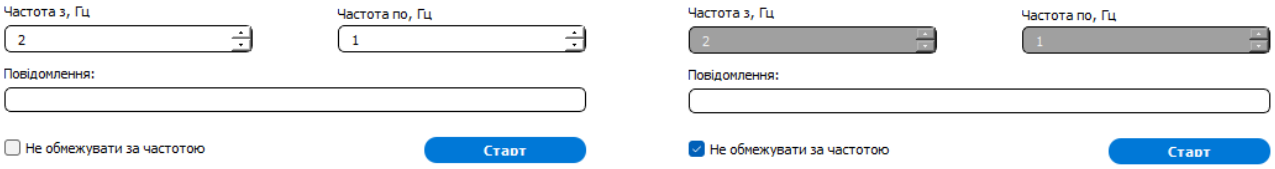

Рисунок 2.12. Робота прапорця «Не обмежувати за частотою»

17. Кнопка «Старт». При натисканні на кнопку, виконується перевірка правильності вводу полів обраного режиму. Якщо перевірка не було пройдено, на екран виводиться повідомлення про помилки в відповідних полях.

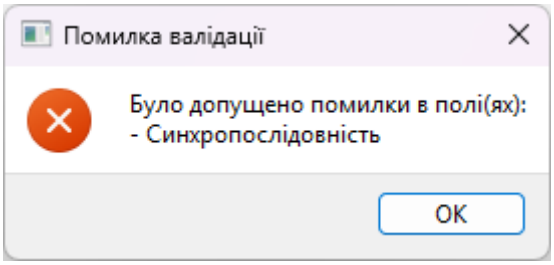

Рисунок 2.13. Помилка валідації

Якщо перевірку полів та кодування було виконано успішно, на екрані буде виведено повідомлення про успіх операції.

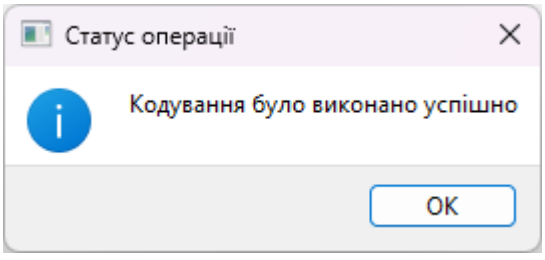

Рисунок 2.14. Повідомлення про успішне виконання операції

Якщо ж було обрано режим «Декодування», після натискання на кнопку виконається перевірка полів (приклад зображено на рисунку 2.13) і виконано декодування. Якщо декодування буде виконано успішно, на екран буде

виведено відповідне інформаційне повідомлення, а результат декодування буде додатково записано в поле «Повідомлення».

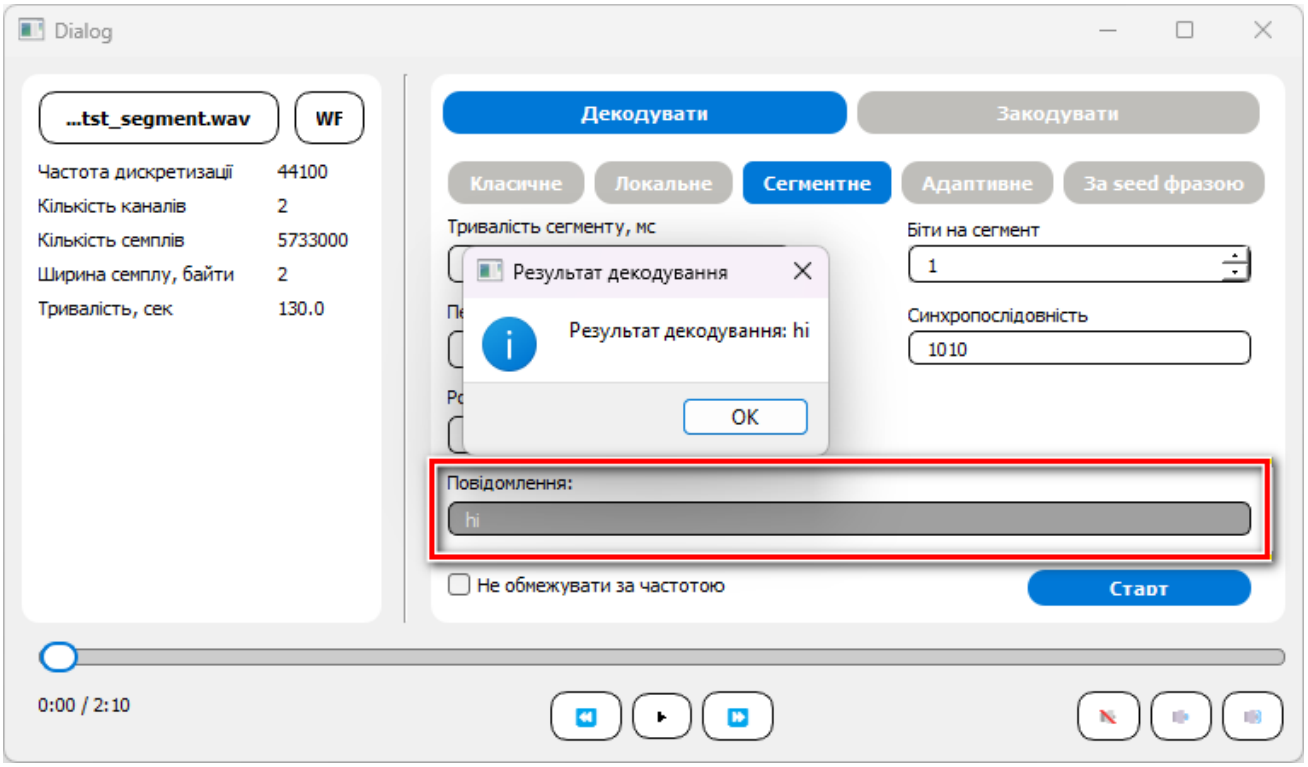

Рисунок 2.14. Результат декодування

Отже, в цьому розділі було розглянуто інтерфейс програмного додатку.

2.5. Архітектура програмного додатку

Архітектура програмного додатку є важливою, оскільки, в правильно закладеній архітектурі, кількість часу для впровадження нового функціоналу набагато менша. Для створення цього програмного додатку, було виділено три основні сутності:

1. Клас «Main Window».
2. Клас «Sound».
3. Клас «Waterfall».

Основним класом програмного застосунку є клас «Main Window», що

реалізує собою основний інтерфейс програмного додатку та всю логіку інтерфейсу. Об’єкт класу «Main Window» вміщує в собі об’єкти класів «Sound» та «Waterfall».

Клас «Sound» являє собою аудіофайл і містить методи для:

- відкриття та збереження аудіофайлу;
- вичитки інформації про аудіофайл;
- керування програванням і звуком;
- кодування і декодування повідомлення для всіх режимів роботи програмного додатку.

Останнім наявним класом програмного застосунку є клас «Waterfall». Він відповідає за вікно «Водоспад». Його представлено окремим класом через специфіку роботи з PyQt5.

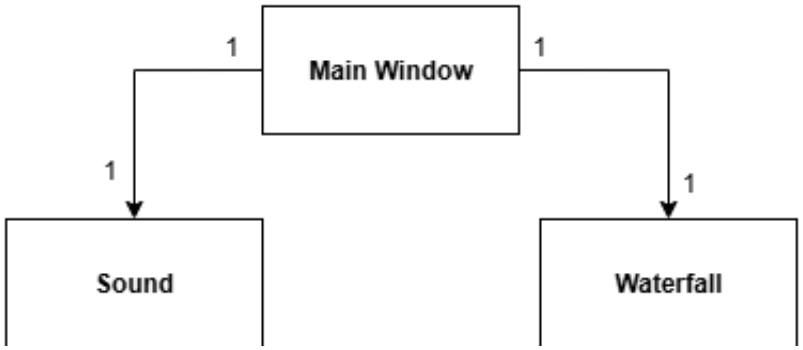

Рисунок 2.15. Спрощена архітектурна схема додатку

2.6. Особливості реалізації програмного застосунку

При реалізації програмного застосунку, спершу було реалізовано клас «Sound». В конструкторі класу винесемо основні параметри аудіофайлу, технічні змінні для керування звуком та відтворенням файлу. Також, додано автоматичний виклик завантаження файлу, якщо шлях до файлу вказано при створенні об’єкту.

```
class Sound:  2 usages
    def __init__(self, filename):
        self.filename = filename        # назва файлу

        self.sound = None               # об'єкт wave

        self.data = None                # список аудіо даних
        self.sample_rate = None         # частота дискретизації
        self.num_channels = None        # кількість каналів
        self.duration_sec = None        # тривалість у секундах
        self.num_frames = None          # загальна кількість семплів
        self.sample_width_byte = None   # ширина семплу в байтах

        self.isPlaying = False          # ознака, чи грає звук
        self.volume = 100               # поточні налаштування звуку

        self.player = QMediaPlayer()    # об'єкт плеєра

        if filename is not None:        # автоматична вичитка файлу, якщо
            self.read_sound()           # передано при створенні об'єкту
```

Рисунок 2.16. Конструктор класу «Sound»

Наступним кроком, перейдемо до завантаження аудіофайлу. Для цього, використовується вбудована в Python бібліотека Wave. На початку функції, виконується перевірка на наявність шляху до аудіофайлу, за потреби виводиться повідомлення про необхідність надати шлях до аудіофайлу. Якщо шлях присутній, виконується вичитка аудіофайлу та виконується отримання основних параметрів звуку:

- Параметр «sample_rate». Частота дискретизації.
- Параметр «num_channels». Кількість каналів
- Параметр «num_frames». Кількість фреймів.
- Параметр «duration_sec». Тривалість звуку.
- Параметр «sample_width_byte». Глибина дискретизації.

- Параметр «data». Визначає масив даних самого аудіофайлу, що необхідний для подальшої роботи. Сам масив представляє з себе потік сирих байтів, що потребують подальшої обробки за допомогою Numpy.

```
def read_sound(self, filename=None):  2 usages
    if filename is not None:
        self.filename = filename
    elif self.filename is None:
        print('Надайте шлях до .wav файлу')
        return

    self.sound = wave.open(self.filename,  mode: 'r')

    self.sample_rate = self.sound.getframerate()
    self.num_channels = self.sound.getnchannels()
    self.num_frames = self.sound.getnframes()
    self.duration_sec = self.num_frames / float(self.sample_rate)
    self.sample_width_byte = self.sound.getsampwidth()
    self.data = self.sound.readframes(self.num_frames)
```

Рисунок 2.17. Вичитка аудіофайлу

Надалі, виконаємо реалізацію функцій, що керують програванням аудіофайлу.

Функція «play_sound». Ініціалізує відтворення аудіофайлу.

Функція «stop_sound». Ініціалізує зупинку відтворення аудіо.

Функція «play_forward». Зміщує програвання аудіо на 10 мілісекунд вперед.

Функція «play_backward». Зміщує програвання аудіо на 10 мілісекунд назад.

Функція «get_volume». Повертає поточний рівень гучності аудіоплеєра.

Функція «mute». Ініціалізує вимикання звуку, зберігаючи поточний рівень гучності.

Функція «unmute». Відновлює гучність після вимкнення звуку.

```
def play_sound(self):  2 usages
    self.player.play()
    self.isPlaying = True

def stop_sound(self):  1 usage
    self.player.stop()
    self.isPlaying = False

def play_forward(self):  1 usage
    self.player.setPosition(self.player.position() + 10)

def play_backward(self):  1 usage
    self.player.setPosition(self.player.position() - 10)

def get_volume(self):  2 usages
    return self.player.volume()

def mute(self):  1 usage
    self.volume = self.player.volume()
    self.player.setVolume(0)

def unmute(self):  1 usage
    self.player.setVolume(self.volume)

def plus_volume(self):  1 usage
    if self.get_volume() + 10 < 101:
        self.player.setVolume(self.volume + 10)
        self.volume = self.player.volume()

def minus_volume(self):  1 usage
    if self.get_volume() - 10 > -1:
        self.player.setVolume(self.volume - 10)
        self.volume = self.player.volume()
```

Рисунок 2.18. Функції керування аудіо і звуком

Функція «plus_volume». Підвищує рівень гучності на 10 одиниць, якщо

він не перевищує максимальне значення.

Функція «minus_volume». Зменшує рівень гучності на 10 одиниць, якщо він не менший за мінімальне значення.

Тепер можна переходити до створення функцій, що відповідають за кодування та декодування повідомлення. Для прикладу, буде наведено тільки функцію класичного кодування, оскільки вона буде самою наочною для пояснення роботи алгоритму.

Розглянемо процес кодування повідомлення. Спершу, система перевіряє присутність завантажених аудіофайлів і у разі їх відсутності, операція припиняється з виведенням відповідного повідомлення до консолі. Далі, відбувається попередня обробка текстової інформації, призначеної для приховування в звуковому сигналі.

Текст конвертується у байтове представлення через кодування UTF-8, після чого обчислюється його розмір. Розмір повідомлення фіксується як 16-бітний заголовок, що додається на початок основного бінарного масиву. Текстове повідомлення трансформується у бінарну послідовність, де кожен символ представлено 8-бітним двійковим кодом. Таким чином створюється цілісний бітовий масив, що містить як заголовок, так і закодовану інформацію, який у подальшому інтегрується у фазу аудіосигналу.

Після формування бінарної послідовності відбувається визначення способу фазового кодування. За умови активації режиму «multi-level» із текстового параметра зчитується числове значення, що задає кількість фазових рівнів модуляції, тобто число бітів, які можливо закодувати в межах одного фрейму. Якщо обрано тип «quadrature», число бітів на фрейм фіксується на рівні двох, тоді як для стандартного режиму «binary» на рівні одного біта.

```
def encode_message_classic(self, message, frame_size=2048, bits_per_frame=1,phase_type='binary', window_type='rectangular',
                           freq_from=None, freq_to=None, output_filename='encoded_sound.wav',):
    if self.data is None:
        print('Спочатку завантажте аудіо файл')
        return False

    message_bytes = message.encode('utf-8')
    msg_length = len(message_bytes)
    header_bits = format(msg_length, '016b')
    payload_bits = ''.join(format(b, '08b') for b in message_bytes)
    bitstream = header_bits + payload_bits

    pt = phase_type.strip().lower()
    if pt.startswith('multi-level') or pt.startswith('multi level'):
        import re
        m = re.search( pattern: r'\((\d)\)', phase_type)
        if m:
            inferred_bits = int(m.group(1))
        else:
            inferred_bits = bits_per_frame
    elif pt.startswith('quadrature'):
        inferred_bits = 2
    else:
        inferred_bits = 1
```

Рисунок 2.19. Перша частина функції кодування

У наступній частині функції виконується синхронізація налаштувань фазової модуляції з реальними характеристиками, а також опрацювання звукових даних перед інтеграцією повідомлення. На початку відбувається контроль узгодженості числа бітів у межах одного фрейму «bits_per_frame» із раніше визначеним варіантом фазового кодування.

Потім розраховується параметр «M», що визначає, на скільки частин буде поділено фазове коло. На підставі параметру, генерується вектор «phase_levels», котрий включає рівномірно розташовані фазові значення в інтервалі від $-\pi$ до π, що відповідають обраній схемі модуляції.

Далі, формується сукупність символів для подальшого кодування. Оскільки загальна кількість бітів у повідомленні не завжди ділиться без залишку на число бітів у фреймі, в кінець бітової послідовності вставляються додаткові нулі. Після цього бітова послідовність сегментується на блоки. Кожен блок містить «bits_per_frame» біт і визначає окремий символ, який надалі отримає відповідне фазове значення в частотному просторі.

Залежно від параметра глибини квантування «sample_width_byte» здійснюється трансформація вихідних байтових даних у чисельний масив NumPy (int16 або uint8), що надає можливість виконувати математичні маніпуляції із сигналом. Крім того, проводиться верифікація розрядності аудіофайлу.

```
if bits_per_frame != inferred_bits:
    print(f"Увага: 'біти на фрейм' змінено з {bits_per_frame} до {inferred_bits} відповідно до типу '{phase_type}'.")
    bits_per_frame = inferred_bits

if bits_per_frame < 1 or bits_per_frame > 5:
    print('Підтримуються 1..5 біт на фрейм')
    return False

M = 1 << bits_per_frame
phase_levels = np.linspace(-np.pi, np.pi, M, endpoint=False)

pad_len = (-len(bitstream)) % bits_per_frame
if pad_len:
    bitstream += '0' * pad_len
symbols = [bitstream[i:i + bits_per_frame] for i in range(0, len(bitstream), bits_per_frame)]

if self.sample_width_byte == 2:
    audio_array = np.frombuffer(self.data, dtype=np.int16)
elif self.sample_width_byte == 1:
    audio_array = np.frombuffer(self.data, dtype=np.uint8)
else:
    print(f'Непідтримувана ширина семплу: {self.sample_width_byte} байт')
    return False
```

Рисунок 2.20. Друга частина функції кодування

В наступному етапі виконується попередня обробка звукового матеріалу перед застосуванням фазової модуляції і аналіз його потенціалу щодо розміщення прихованих даних. Коли аудіозапис містить два канали, алгоритм оперує виключно лівим каналом, оскільки, такий підхід спрощує процедуру обробки та знижує ймовірність виникнення артефактів. Система розраховує загальну кількість сегментів, що підлягають виокремленню із звукової послідовності, після чого здійснює верифікацію достатності доступного обсягу для інтеграції повідомлення.

На наступному кроці, конструюється вагова функція для кожного

часового сегмента згідно з визначеним різновидом (прямокутна, Хеммінга чи Ханна), що сприяє плавному переходу на границях фреймів і мінімізації ефекту спектрального розсіювання. Встановлюється частотна зона, призначена для імплементації інформації, з обов'язковою валідацією меж задля збереження значень у прийнятному частотному коридорі.

```
if self.num_channels == 2:
    audio_array = audio_array[::2]

total_frames = len(audio_array) // frame_size
capacity_symbols = total_frames
if len(symbols) > capacity_symbols:
    print(f'Повідомлення занадто довге: потрібно {len(symbols)} фреймів, доступно {capacity_symbols}.')
    print(f'Максимум {capacity_symbols * bits_per_frame} біт ({capacity_symbols * bits_per_frame // 8} байт)')
    return False

print(f'Кодування phase coding: frame={frame_size}, bits/frame={bits_per_frame}, M={M}, window={window_type}')
print(f'Символів до кодування: {len(symbols)} (разом {len(bitstream)} біт)')

wt = window_type.strip().lower()
if wt in ('прямокутне', 'rectangular', 'rectangle', 'boxcar'):
    window = np.ones(frame_size, dtype=np.float64)
elif wt in ('хеммінга', 'хемінга', 'hamming'):
    window = np.hamming(frame_size).astype(np.float64)
elif wt in ('ханна', 'hann', 'hanning'):
    window = np.hanning(frame_size).astype(np.float64)
else:
    print(f"Невідоме вікно '{window_type}', використано прямокутне")
    window = np.ones(frame_size, dtype=np.float64)

fs = self.sample_rate
N = frame_size
def hz_to_bin(f_hz):
    return int(np.clip(round(f_hz * N / fs),  a_min: 1, N // 2 - 1))

if freq_from is not None and freq_to is not None and freq_to > freq_from:
    b_start = hz_to_bin(freq_from)
    b_end = hz_to_bin(freq_to)
else:
    b_start = N // 8
    b_end = N // 4
b_start = max(1, min(b_start, N // 2 - 1))
b_end = max(b_start + 1, min(b_end, N // 2 - 1))
```

Рисунок 2.21.Третя частина функції кодування

Надалі, відбувається пряма інтеграція інформаційних бітів у фазову структуру звукового матеріалу. У випадку 8-бітного PCM аудіо виконується

нормалізація значень відносно нульової точки, що гарантує адекватну фазову трансформацію, водночас для альтернативних форматів сигнал трансформується безпосередньо в формат float64. Звуковий потік сегментується на часові відрізки, що кореспондують із символами повідомлення, причому на кожен відрізок накладається вагова функція для усунення розривів між суміжними блоками.

Кожний часовий відрізок піддається процедурі FFT з метою отримання частотного розкладу, амплітудних та фазових характеристик. Для актуального символу визначається необхідне фазове зміщення, котре імплементується в обраній частотній зоні, а задля збереження симетричності спектра застосовується принцип комплексного спряження. Після корекції фазових параметрів частотне подання трансформується назад у часовий домен через IFFT, і фрагмент знову множиться на віконну функцію для прибирання артефактів і різких переходів на стиках сегментів. Оброблені фрагменти інтегруються в початковий масив даних, створюючи аудіопотік із закодованою інформацією.

На фінальній стадії, змінений сигнал конвертується у цілочисельне представлення, що забезпечує можливість його запису у *.Wav. Для 16-бітного аудіо значення обрізаються до допустимого діапазону і конвертуються в int16, тоді як для 8-бітного PCM додається офсет 128, щоб повернути дані в беззнаковий формат uint8.

У випадку, коли вихідний аудіофайл мав двоканальну структуру, відбувається реконструкція стереофонічного звучання через дублювання обробленого сигналу в обидва аудіоканали. Згодом, отриманий масив експортується у *.Wav із дотриманням початкових параметрів: числа каналів, розрядності квантування та частоти семплювання.

```
if self.sample_width_byte == 1:
    audio_float = audio_array.astype(np.float64).copy() - 128.0
else:
    audio_float = audio_array.astype(np.float64).copy()

for i, sym_bits in enumerate(symbols):
    start = i * frame_size
    end = start + frame_size
    if end > len(audio_float):
        break

    segment = audio_float[start:end].copy()
    seg_win = segment * window

    spec = np.fft.fft(seg_win)
    mag = np.abs(spec)
    phase = np.angle(spec)

    sym_val = int(sym_bits, 2)
    target_phase = phase_levels[sym_val]

    phase[b_start:b_end] = target_phase
    for k in range(b_start, b_end):
        phase[-k] = -phase[k]

    spec_mod = mag * np.exp(1j * phase)
    seg_rec = np.fft.ifft(spec_mod).real

    seg_rec *= window

    audio_float[start:end] = seg_rec
```

Рисунок 2.22. Четверта частина функції кодування

```
if self.sample_width_byte == 2:
    encoded_audio = np.clip(audio_float, -32768,  a_max: 32767).astype(np.int16)
else:
    encoded_audio = np.clip(audio_float + 128.0,  a_min: 0,  a_max: 255).astype(np.uint8)

if self.num_channels == 2:
    stereo = np.zeros(len(encoded_audio) * 2, dtype=encoded_audio.dtype)
    stereo[::2] = encoded_audio
    stereo[1::2] = encoded_audio
    encoded_audio = stereo

with wave.open(output_filename,  mode: 'w') as out_wav:
    out_wav.setnchannels(self.num_channels)
    out_wav.setsampwidth(self.sample_width_byte)
    out_wav.setframerate(self.sample_rate)
    out_wav.writeframes(encoded_audio.tobytes())

print(f"Повідомлення успішно закодоване в '{output_filename}'")
return True
```

Рисунок 2.23. Фінальна частина кодування

Функція decode_message_classic() здійснює вилучення захованого тексту з аудіоструктури, що була опрацьована технікою фазової модуляції, і багато в чому дублює попередні операції, що виконувались при кодуванні. На початковому етапі відбувається верифікація присутності аудіо даних та синхронізація налаштувань декодування (bits_per_frame, phase_type) відповідно до застосованого типу модифікації. Надалі, формується набір фазових значень, створюється масив аудіоданих залежно від кількості каналів і розрядності семплів, а також визначається тип віконної обробки та частотні границі для спектрального аналізу, що повністю співпадає з підготовчою стадією під час вбудовування.

Ключова різниця міститься у логіці обробки звукової послідовності. На противагу модифікації фазових компонент, тут для кожного часового сегмента розраховується усереднена фаза в межах обраного частотного інтервалу з урахуванням амплітудних характеристик. Після цього, отримана середня фаза співставляється з референтними фазовими позначками для ідентифікації захованого символу, який конвертується у бінарний вигляд. Після отримання всієї бітової послідовності виділяється заголовок (16 біт) для встановлення фактичної довжини повідомлення, і з наступних інформаційних бітів конструюється байтовий вектор, який підлягає декодуванню у текстовий формат UTF-8.

В результаті, функція виводить та повертає декодоване повідомлення або повідомляє про помилки, якщо файл занадто короткий, параметри невірні або виникає проблема з UTF-8 декодуванням.

В формуванні класів «» та «» не виникало особливостей, оскільки код цих класів формується шаблонно, для PyQt5 додатку.

```
bitstream = []
for i in range(total_frames):
    start = i * frame_size
    end = start + frame_size
    segment = audio_float[start:end]
    if len(segment) < frame_size:
        break
    seg_win = segment * window
    spec = np.fft.fft(seg_win)
    band = spec[b_start:b_end]
    mag = np.abs(band)
    if mag.sum() == 0:
        avg_phase = 0.0
    else:
        unit = np.exp(1j * np.angle(band))
        complex_mean = np.sum(unit * mag) / (mag.sum() + 1e-12)
        avg_phase = np.angle(complex_mean)
    diffs = np.angle(np.exp(1j * (avg_phase - phase_levels)))
    sym_idx = int(np.argmin(np.abs(diffs)))
    bits = format(sym_idx, f'0{bits_per_frame}b')
    bitstream.append(bits)
```

Рисунок 2.24. Алгоритм декодування повідомлення

```
def click_start(self):  1 usage
    if self.active_mode == 'encode':
        match self.active_type:
            case 'classic':
                frame_size = self.ui.sp_param1.value()
                bits_per_frame = self.ui.sp_param2.value()
                phase_type = self.ui.cb_param3.currentText()
                window_type = self.ui.cb_param4.currentText()
                message = self.ui.le_message.text()
                if self.ui.checkBox.isChecked():
                    freq_from = None
                    freq_to = None
                else:
                    freq_from = self.ui.sp_param5.value()
                    freq_to = self.ui.sp_param6.value()

                self.sound.encode_message_classic(message, frame_size, bits_per_frame, phase_type, window_type, freq_from, freq_to,  output_filename: 'tst_encoded.wav')
            case 'local':...
            case 'segment':...
            case 'adaptive':...
            case 'seed':...
    else:
        match self.active_type:
            case 'classic':
                frame_size = self.ui.sp_param1.value()
                bits_per_frame = self.ui.sp_param2.value()
                phase_type = self.ui.cb_param3.currentText()
                window_type = self.ui.cb_param4.currentText()
                if self.ui.checkBox.isChecked():
                    freq_from = None
                    freq_to = None
                else:
                    freq_from = self.ui.sp_param5.value()
                    freq_to = self.ui.sp_param6.value()

                res = self.sound.decode_message_classic(frame_size, bits_per_frame, phase_type, window_type, freq_from, freq_to)
                self.ui.le_message.setText(res if res is not None else '')
                msg = QMessageBox()
                msg.setIcon(QMessageBox.Information)
                msg.setWindowTitle('Результат декодування')
                msg.setText(f'Результат декодування: {res}')
                msg.setStandardButtons(QMessageBox.Ok)
                msg.exec_()
```

Рисунок 2.25. Ініціалізація кодування та декодування в додатку

Для побудови графіків водоспаду, використовувалась вбудована функція «spectrogram» з бібліотеки Scipy.

```
def plot_spectrogram(self, wav_path, name):  2 usages
    self.ax.clear()
    if not os.path.exists(wav_path):
        self.ax.text( x: 0.5,  y: 0.5,  s: f"Файл не знайдено:\n{wav_path}", ha='center', va='center', transform=self.ax.transAxes)
        self.draw()
        return

    samplerate, data = wavfile.read(wav_path)
    if data.ndim > 1:
        data = data.mean(axis=1)
    if np.issubdtype(data.dtype, np.integer):
        max_val = np.iinfo(data.dtype).max
        data = data.astype(np.float32) / max_val
    else:
        data = data.astype(np.float32)

    f, t, Sxx = signal.spectrogram(data, fs=samplerate, window='hann', nperseg=NPERSEG, noverlap=NOVERLAP, detrend=False, scaling='density', mode='magnitude')

    Sxx_db = 20.0 * np.log10(Sxx + 1e-12)
    im = self.ax.imshow(Sxx_db, aspect='auto', origin='lower', extent=[t.min(), t.max(), f.min(), f.max()])
    self.ax.set_xlabel('Час (s)')
    self.ax.set_ylabel('Частота (Hz)')
    self.ax.set_title(name)
    self.figure.colorbar(im, ax=self.ax, label='Amplitude (dB)')
    self.draw()
```

Рисунок 2.25. Функція для побудови водоспаду

2.7. Організація тестування програмного засобу

Основним завданням випробування створеного програмного додатку є підтвердження правильності імплементації механізмів приховування та вилучення даних, перевірка стабільності функціонування додатку, а також аналіз ергономічності та надійності графічного інтерфейсу користувача. Процес випробування становить критично важливий етап розробки, і надає можливості для виявлення потенційних дефектів, визначення ефективності системи та підтвердження відповідності програмного додатку встановленим вимогам.

Під час тестування функціональності застосунку було здійснено наступні категорії перевірок.

Перевірка функціональності. Тестування роботи ключових варіантів кодування: класичного, локального, сегментованого, адаптивного та кодування на основі ключової фрази. Контролювалась правильність інтеграції

повідомлення у звуковий сигнал, точність його відновлення, а також узгодженість результуючого сигналу з очікуваними характеристиками.

Перевірка інтерфейсу. Аналіз ергономічності взаємодії користувача з інтерфейсом додатку, правильність візуалізації компонентів, інтуїтивність організації налаштувань та чіткість маркування полів введення. Окрему увагу сфокусовано на логіці вибору режимів кодування та налаштування відображуваних полів відповідно обраного режиму.

Перевірка ефективності. Визначення швидкодії під час опрацювання аудіофайлів різноманітної тривалості та роздільності (від 44.1 до 192 кГц). Контролювалась тривалість виконання операцій кодування та декодування, реакція графічного інтерфейсу при відображенні спектрограм і водоспадних діаграм.

Протягом процесу випробування не було зафіксовано критичних дефектів у функціонуванні програми. Кілька незначних недоліків у відображенні інтерфейсу та валідації введень було усунуто. Програма засвідчила надійне функціонування навіть за умов обробки довготривалих аудіозаписів і значних параметрів FFT.

За результатами проведеного випробування було верифіковано, що програмний інструмент належно реалізує алгоритми фазового кодування, гарантує точне відновлення повідомлень та ергономічну взаємодію користувача з усім функціональним спектром можливостей. Застосунок задовольняє критерії стабільності, продуктивності та зручності, що надає можливість використовувати його застосування як в освітніх, так і в науково-дослідних напрямках.

2.8. Рекомендації по використанню та впровадженню програмного засобу

Створений програмний додаток знаходить застосування як у наукових дослідженнях, так і в освітніх програмах, що стосуються цифрової обробки

сигналів, стеганографії та захисту інформації. Його можливості дають змогу не тільки ілюструвати механізм фазового кодування, але й детально вивчати вплив різних параметрів на ефективність вбудовування та надійність відновлення прихованих даних.

Додаток рекомендується впроваджувати в освітній процес при викладанні курсів, що охоплюють цифрову обробку сигналів, телекомунікації, акустику, теорію інформації, криптографію та стеганографію. Він надає студентам можливість на практиці ознайомитися з функціонуванням алгоритмів фазового кодування, вивчити різноманітні режими роботи і здійснити їх порівняльний аналіз за критеріями сигналу. Сфери практичного застосування програмного продукту включають:

- використання в навчальних лабораторіях для ілюстрації основ аудіостеганографії;
- залучення до науково-дослідних робіт для апробації інноваційних методів приховування даних в аудіоматеріалах;
- впровадження у системи захисту інформації, де критичною є забезпечення конфіденційності чи верифікації справжності аудіо контенту.

Експлуатація програми не вимагає спеціалізованого устаткування, оскільки, програму було конвертовано в *.exe файл, що містить всі залежності та не вимагає наявності Python та сторонніх бібліотек. Завдяки графічному інтерфейсу на базі PyQt5, користувач отримує можливість легко регулювати налаштування кодування, відстежувати трансформації спектру сигналу та аналізувати результати вбудовування інформації.

Перспективні напрямки вдосконалення програмного рішення:

- впровадження пакетного режиму для паралельної обробки кількох аудіофайлів;
- розробка компонента статистичного аналізу для визначення рівня прихованості інформації;
- створення веб-застосунку або мобільного додатку, що забезпечить

доступність для ширшої аудиторії.

Отже, представлений програмний продукт виступає не лише як освітній засіб, а й як платформа для майбутніх досліджень і розвитку методів стеганографії на основі фазового кодування.

ВИСНОВКИ

У роботі досліджено сучасні підходи до стеганографії аудіосигналів, зокрема, технологію фазового кодування (Phase Coding), що забезпечує надійне приховування даних при незначних модифікаціях звукової хвилі. Вивчено ключові механізми роботи фазового кодування, його сильні та слабкі сторони у порівнянні з альтернативними технологіями. Досліджено механізм сегментного та адаптивного фазового кодування, який дозволяє оптимізувати процес інтеграції повідомлення залежно від характеристик аудіосигналу.

Визначено ключові вимоги до програмного додатку для імплементації технології фазового кодування. Проведено аналіз інструментів та форматів створення програмного забезпечення, зокрема десктопних, браузерних та мобільних варіантів. Аргументовано вибір мови Python та бібліотеки PyQt5 як ключових технологій розробки завдяки їхній універсальності, підтримці різних платформ, широкому спектру інструментів для аналізу сигналів та побудови користувацьких інтерфейсів.

Створено програмний додаток для кодування та декодування повідомлень із аудіофайлів за допомогою методу фазового кодування. У програмі впроваджено функціонал вибору варіанту кодування (класичний, локальний, сегментний, адаптивний, за ключовою фразою), визначення основних параметрів звукового сигналу. Користувацький інтерфейс забезпечує інтуїтивну роботу, включаючи графічне відображення сигналу, демонстрацію спектральних характеристик та налаштування параметрів кодування.

Здійснено випробування ключових функціональних режимів програми, що засвідчило її надійність, правильність алгоритмічних рішень та результативність фазового кодування за різноманітних параметрів аудіосигналу.

Завдяки виконаній роботі було засвоєно технології цифрового аналізу сигналів, методи фазового кодування та засади побудови графічних програм на Python із застосуванням PyQt5. Створене програмне рішення придатне для

вивчення стеганографічних технологій у освітніх та дослідницьких проєктах, а також слугує фундаментом для майбутнього розвитку функціональності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yadnya M.S., Kanata B., Anwar M.K. Using Phase Coding Method for Audio Steganography with the Stream Cipher Encrypt Technique. *Proceedings of the First Mandalika International Multi-Conference on Science and Engineering (MIMSE 2022)*.

URL: https://www.atlantis-press.com/proceedings/mimse-i-c-22/125980143 (дата звернення: 12.11.2025).

2. Souvik Bhattacharyya, Indradip Banerjee, Gautam Sanyal. *A Survey of Steganography and Steganalysis Technique in Image, Text, Audio and Video as Cover Carrier.*

URL: https://www.rroij.com/open-access/a-survey-of-steganography-and-steganalysis-technique-in-image-text-audio-and-video-as-cover-carrier-1-16.php?aid=37026 (дата звернення: 12.11.2025).

3. A.A. AlSabhany. *Digital audio steganography: Systematic review ...* (2020)

URL: https://www.sciencedirect.com/science/article/abs/pii/S1574013720304160 (дата звернення: 12.11.2025).

4. ТРУХАНСЬКА, В. О.; ХМЕЛІВСЬКИЙ, Ю. С. Огляд напрямів прихованої передачі даних у відеофайлах. *Комп'ютерні технології обробки даних*, 2024, 113-115.

5. УМАНСЬКИЙ, О. Г. Огляд стеганографічних методів та їх застосування у системах забезпечення безпеки банківських інформаційних ресурсів. *Èlektronnoe Modelirovanie*, 2024, 46.4.

6. International Organization for Standardization. ISO/IEC 10149:1995 Information technology – Discs with recorded digital audio data – Physical and file-format characteristics.

7. RASTEGAYEV, Roman, et al. REVIEW OF METHODS FOR

EMBEDDING DIGITAL WATERMARKS FOR AUDIO FILE PROTECTION. *Системи управління, навігації та зв'язку. Збірник наукових праць*, 2024, 4.78: 132-138.

8. ДРОБАХІН, Олег Олегович. Розвиток в Україні методів вимірювання в мікрохвильовому і терагерцовому діапазонах електромагнітних хвиль (огляд). Вісті вищих учбових закладів. Радіоелектроніка, 2024.

9. Kovtunenko, Arkady A. *Review of the Main Methods of Digital Watermarking and Embedding Watermarks into Audio Signals.*

10. ФОКІН, Денис Геннадійович; ЄВДОКИМЕНКО, Марина Олександрівна. ВИКОРИСТАННЯ ЦИФРОВОЇ СТЕГАНОГРАФІЇ В СУЧАСНОМУ КІБЕРПРОСТОР. *Збірник матеріалів Міжнародної науково-технічної конференції «ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ»*, 2025, 153-155.

11. ТРУХАНСЬКА, В. О.; ХМЕЛІВСЬКИЙ, Ю. С. Огляд напрямів прихованої передачі даних у відеофайлах. *Комп'ютерні технології обробки даних*, 2024, 113-115.

12. Mohamad, F.S., Yasin, N.S.M., & Iqtait, M. *Information Hiding Based on Audio Steganography using Least Significant Bit.* International Journal of Engineering and Technology, 7(3.28), 334-336, 2018. DOI:10.14419/ijet.v7i3.28.28427.

13. AlSabhany, A.A., Ali, A.H., Ridzuan, F.H.M., Ab Halim, A.H., & Mokhtar, M.R. *Digital audio steganography: systematic review, classification, and analysis of the current state of the art.* Computer Science Review, 2020. DOI:10.1016/j.cosrev.2020.100316.

14. Bah l, J., & Ramakishore, D. *LSB Technique and Its Variations Used in Audio Steganography: A Survey.* IJERT, Vol.2 Issue 4 (April 2013).

15. Стружко, В. Р.; Антоненко, С. В. "Порівняння характеристик методів та алгоритмів стеганографії в зображеннях та звукових сигналах." *Actual Problems of Applied Information Technology*, 2024. URL: https://actualproblems.dp.ua/index.php/APAIT/article/view/269

16. Світловський, Є. "Стеганографічні підходи до оброблення

мультимедіа-контенту.” *Вісник Криворізького державного університету. Серія: Комп’ютерні науки та управління*, 2023. URL: https://visnikkrnu.kdu.edu.ua/statti/2023_3_2023_3_185.pdf

17. Zorilo, B. B. “Стеганографія та криптографія: нові напрями розвитку.” *Інформаційна безпека*, 2020, №3-4. DOI:10.15276/imms.v10.no3-4.136.

URL: https://immm.op.edu.ua/files/archive/n3-4_v10_2020/2020_3-4(2).pdf

18. Ziubina R., Teliushchenko V., Veselska O., Petrenko A. „Evaluating the Steganographic Integrity of the Phase Coding Method for Concealing Confidential Information within Audio Files.” *International Journal of Emerging Technologies*, 2024.

URL: https://ijet.pl/index.php/ijet/article/view/10.24425-ijet.2024.152088/2929

19. Yang G. „An Improved Phase Coding Audio Steganography Algorithm.” *Journal of The Colloquium for Information Systems Security Education*, Vol 12 No 1 (2025). DOI:10.53735/cisse.v12i1.195

20. Поліщук А. О. “Методи комп’ютерної стеганографії для аудіо контейнерів.” Магістерська дисертація, Київ, 2018. URL: https://ela.kpi.ua/handle/123456789/23834

21. Ковальчук С. О., Кухарська Н. П. “Використання методу фазового кодування для приховування конфіденційної інформації в аудіофайлах.”, Львівський державний університет безпеки життєдіяльності, 20XX. URL: https://sci.ldubgd.edu.ua/bitstream/123456789/758/1/19.pdf

22. GUAÑA-MOYA, Javier, et al. Information security vulnerabilities using steganography as the art of hiding information. In: *International Conference on Information Technology & Systems*. Cham: Springer Nature Switzerland, 2024. p. 107-116.

23. Шудра В. Ю. Вдосконалення стеганографічного методу фазового кодування аудіофайлів за рахунок зміни фази звукового сегменту. Магістерська кваліфікаційна робота. Вінницький національний технічний університет. 2018. URL: http://ir.lib.vntu.edu.ua//handle/123456789/25735

24. BOHRA, Shresty, et al. Advancements in Modern Steganography Techniques for Enhanced Data Security: A Comprehensive Review. In: *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*.

25. IEEE, 2024. p. 941-944.APAU, Richard, et al. Image steganography techniques for resisting statistical steganalysis attacks: A systematic literature review. *PloS one*, 2024, 19.9: e0308807.

ДОДАТОК А
Технічне завдання

1 Призначення розробки

Роботи з розроблення додатку для кодування та декодування інформації в аудіо методами Phase Coding здійснюються на підставі написання кваліфікаційної роботи.

2 Мета додатку

Програмний додаток призначений для кодування та декодування повідомлення в аудіо за допомогою різних типів роботи Phase Coding програми. Також, додаток дозволяє відтворювати аудіо, переглядати основні параметри та порівнювати графіки водоспаду сигналу.

3 Вимоги до програмного продукту

2.1.1. Функціональні вимоги

Функціональні вимоги до додатку включають:

1. Імпорт і первинна обробка звукових файлів.

- Застосунок має підтримувати імпорт аудіоматеріалів у форматах WAV.

- Після імпорту системі слід відображати ключові характеристики сигналу: частоту семплювання, загальну тривалість та кількість аудіоканалів.

- Необхідно забезпечити функціонал попереднього відтворення завантаженого файлу.

2. Вибір методу фазової модуляції.

- Система має надавати можливість обрання одного з наявних методів кодування:

  - Класичний.
  - Локальний.

- Сегментний.
- Адаптивний.
- За ключовою фразою.

- При виборі конкретного методу мають автоматично з'являтися відповідні поля налаштувань.

3. Кодування повідомлення.

- Система має забезпечувати кодування текстової інформації у вибраному аудіосигналі через модифікацію фазових характеристик.

- Після завершення кодування модифікований аудіофайл із закодованими даними експортується, а на екрані з'являється сповіщення про успішне виконання операції.

4. Декодування повідомлення.

- Система має здійснювати розшифрування та відображати відновлене повідомлення у текстовому вигляді.

- Програма повинна відпрацьовувати успішно, за умови того, що

5. Графічне представлення сигналу.

- Застосунок має забезпечувати візуалізацію сигналу у форматі водоспаду.

- Виконання порівняльного аналізу спектрів вихідного та закодованого сигналів.

6. Управління інтерфейсом програми.

- Всі базові функціональні можливості мають бути реалізовані через графічний інтерфейс.

- Інтерфейс має характеризуватись інтуїтивністю та логічною організацією елементів керування.

- Забезпечити відображення інформативних повідомлень про виявлені помилки чи некоректні параметри.

7. Стабільність та захист від збоїв.

- Програма має належним чином обробляти виняткові ситуації (зокрема, непідтримуваний формат або відсутність тексту для кодування).

2.1.2. Нефункціональні вимоги

Нефункціональні вимоги до додатку включають:

1. Вимоги щодо швидкодії.

- Розроблений програмний додаток має забезпечувати опрацювання звукових файлів у режимі реального часу або близькому до нього для записів тривалістю до 10 хвилин.

- Тривалість процесу кодування чи декодування інформації не повинна перевищувати 5 секунд для аудіоматеріалів тривалістю до 1 хвилини (за стандартної частоти дискретизації 44,1 кГц).

- Споживання оперативної пам'яті має залишатися в межах 500 МБ під час опрацювання аудіофайлів середньої довжини.

2. Вимоги щодо стабільності роботи.

- Застосунок має коректно функціонувати при опрацюванні звукових файлів формату WAV, інформуючи користувача про виявлені помилки у разі невідповідного формату.

- При виникненні збоїв під час процесу кодування чи декодування система має видавати зрозуміле повідомлення з описом причини проблеми.

3. Вимоги щодо ергономічності.

- Користувацький інтерфейс має бути зрозумілим інтуїтивно навіть для осіб без спеціалізованих знань у галузі цифрового опрацювання сигналів.

- Інтерфейс застосунку має бути локалізовано українською мовою.

- Елементи управління (кнопки, меню вибору, поля введення) повинні бути розташовані логічними групами відповідно до їх призначення.

4. Вимоги щодо сумісності та мобільності.

- Розроблений застосунок має коректно працювати на провідних операційних платформах Windows, Linux і macOS (за допомогою інтерпретатора Python).

- Для функціонування системи на macOS необхідний інтерпретатор **Python 3.10** або пізнішої версії. Функціонування програми на Windows та Linux не потребує додаткових умов.

- Застосунок має бути реалізований як кросплатформенне рішення, що не потребує окремої компіляції для різних операційних систем.

5. Вимоги щодо захисту даних.

- Система не повинна зберігати приховані текстові дані або звукові файли користувача без його явного дозволу.

- Інформація, що підлягає кодуванню, має опрацьовуватись виключно в оперативній пам'яті без створення тимчасових копій на носії.

- Програма не повинна здійснювати жодних мережевих з'єднань або передачі інформації без прямої згоди користувача.

4 Вимоги до надійності

Програмний продукт має забезпечувати коефіцієнт доступності системи більше 0,95 (обчислюється як частка від ділення середнього часу безперебійної роботи на загальну суму середнього часу безперебійної роботи і середньої тривалості відновлення після збою);

5 Умови експлуатації

Програма може використовуватись на пристроях з операційною системою Windows, Linux чи macOS (вимагається встановлення Python).

6 Вимоги до інформаційної і програмної сумісності

Програмний додаток повинен успішно працювати на всіх пристроях, що використовують операційну систему Windows 10, Linux Ubuntu 20.04 та вище. Додаток має бути сумісним з різними моделями ПК що підтримують ці версії ОС.

7 Вимоги до інформаційної і програмної сумісності

До набору необхідної документації входить:

- кваліфікаційна робота;
- технічне завдання;
- інструкція користувачу.

8 Стадії і етапи розробки

1. Попереднє дослідження. Проєктування, розробка і тестування програмного забезпечення

2. Створення технічної специфікації. Сформовано набір функціональних і нефункціональних вимог до програмного додатку, деталізовано ключові функції програми.

3. Розробка системної архітектури. Визначено архітектуру застосунку, механізми взаємодії між компонентами, принципи обробки цифрових сигналів і здійснено добір технологічних засобів.

4. Імплементація програмного рішення. Впроваджено алгоритмічні механізми кодування та декодування, побудовано користувацький інтерфейс, забезпечено візуалізацію.

5. Верифікація і налагодження. Здійснено контроль правильності

виконання ключових операцій, стійкості функціонування, точності розрахунків.

9 Порядок контролю і приймання

Оцінювання результатів виконаної роботи відбувається під час захисту кваліфікаційної роботи.

ДОДАТОК Б
Інструкція користувача

1. Загальні відомості

Настільний програмний додаток для виконання стеганографії.

2. Функціональне призначення

Програмний додаток призначений для освітніх та дослідницьких робіт з аудіо стеганографії.

3. Умови застосування програми

Програма призначена для використання на пристроях з операційною системою Windows, Linux, macOS. Для роботи на операційній системі macOS вимагається встановлення Python.

4. Повідомлення оператору

Натиснути на іконку додатку для запуску.

5. Опис роботи програми

Після запуску додатку, користувачу необхідно завантажити аудіо сигнал формату *.wav. Для виконання цієї дії, необхідно натиснути на кнопку «Завантажити *.wav» та обрати аудіофайл в провіднику чи іншому додатку для роботи з файловою системою (для інших операційних систем).

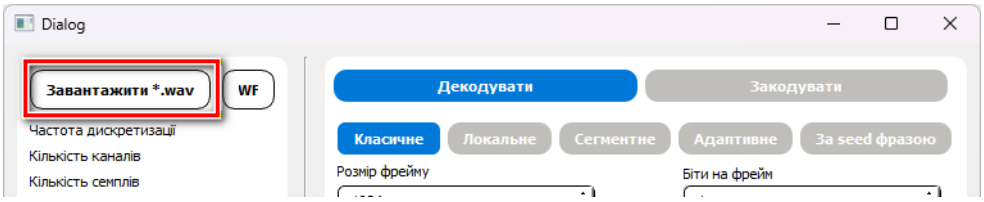

Рисунок Б.1. Кнопка для завантаження аудіофайлу

Після успішного завантаження аудіофайлу, будуть відображені основні параметри сигналу.

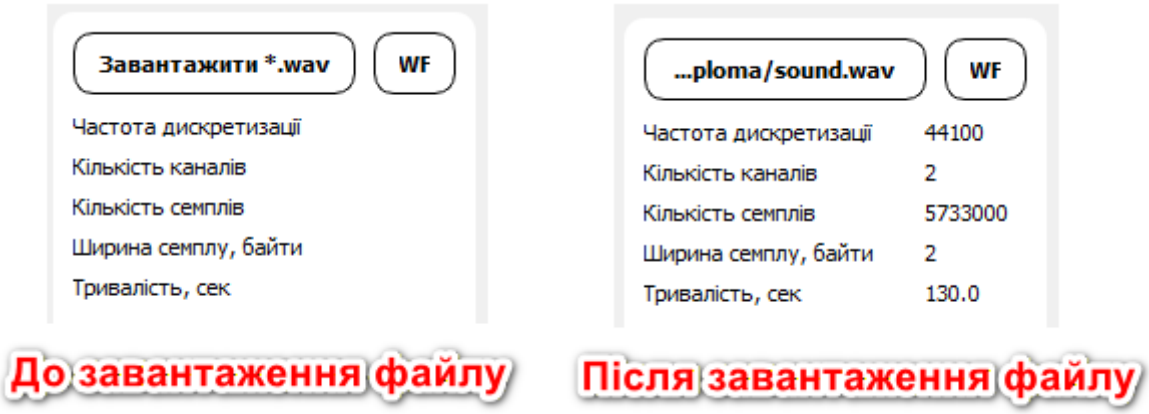

Рисунок Б.2. Результат завантаження файлу

Після завантаження аудіофайлу, можна виконати кодування. Для цього, необхідно обрати режим «Закодувати» та один з методів кодування. Для прикладу, буде обрано класичний метод.

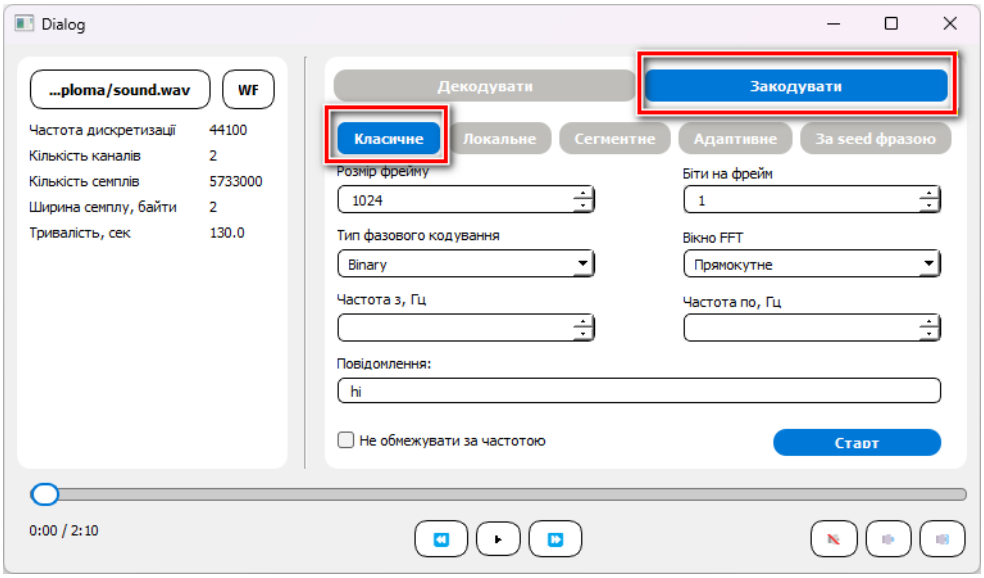

Рисунок Б.3. Вибір режиму та методу для кодування

Після вибору режиму та методу, до редагування стануть доступними поля методу та поле «Повідомлення». Необхідно заповнити ці поля. Для прикладу, буде закодовано повідомлення «test» на налаштуваннях за замовчуванням. Обмеження за частотою не буде виконуватись.

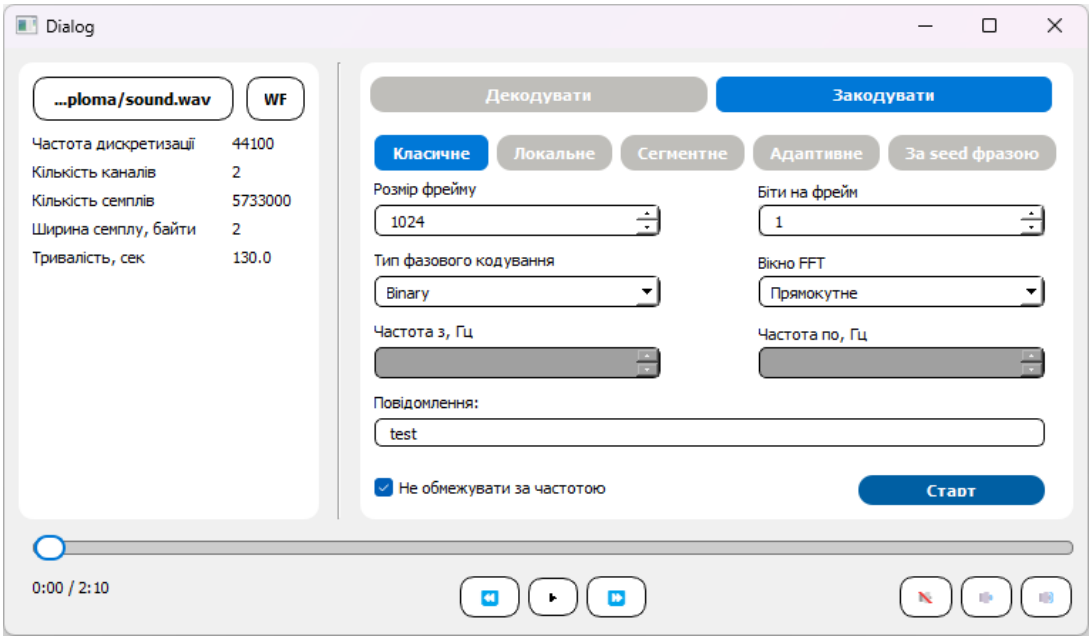

Рисунок Б.4. Кодування повідомлення

Після завершення кодування, буде виведено відповідне інформаційне повідомлення і автоматично буде збережено файл *_encoded.wav. Тепер, необхідно обрати новозгенерований файл, аналогічно тому, як було обрано першочерговий файл. Після його відкриття, змінимо режим на «Декодувати», не змінюючи параметрів.

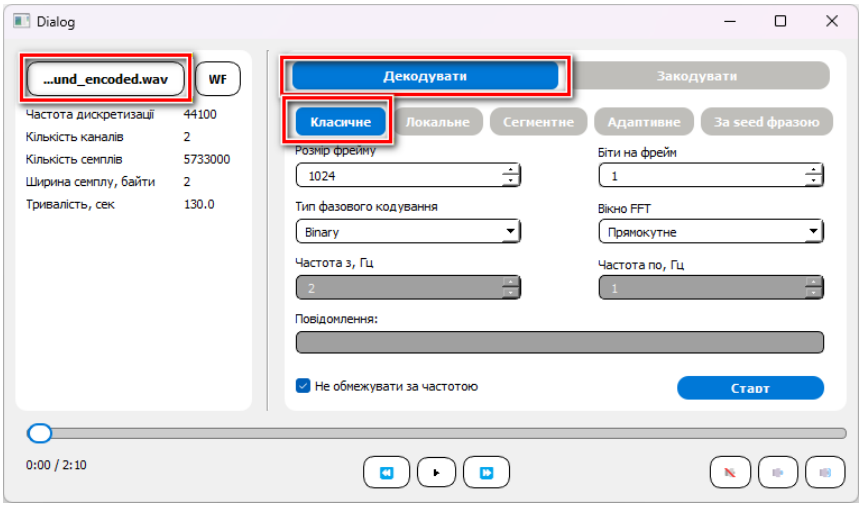

Рисунок Б.5. Встановлення параметрів та декодування

Після завантаження файлу, необхідно натиснути на кнопку «Старт» для початку декодування. Після завершення процесу на екран буде виведено інформаційне повідомлення з результатом декодування.

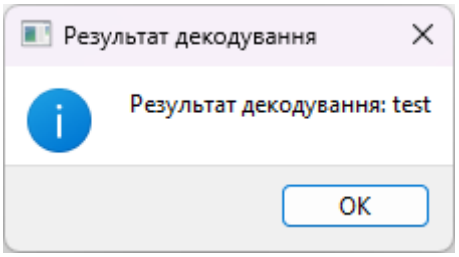

Рисунок Б.6. Результат декодування

Після декодування, можна перейти до порівняння графіків водоспаду. Для цього, необхідно натиснути на кнопку «WF», в результаті чого буде відображено вікно порівняння.

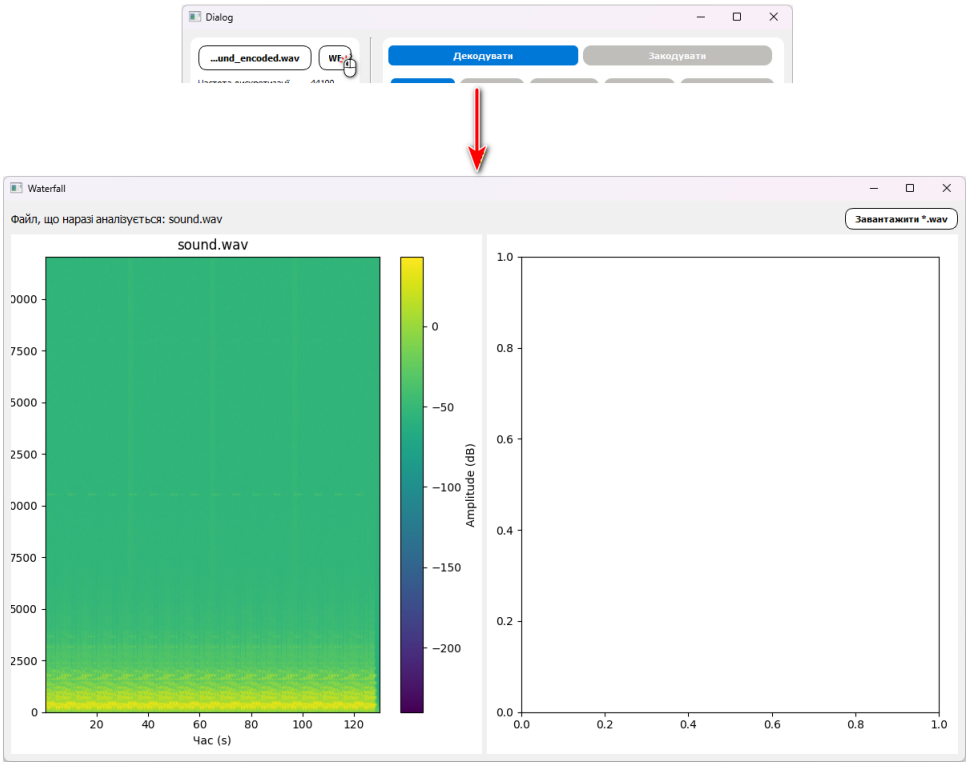

Рисунок Б.7. Відкриття вікна порівняння водоспадів

В лівій частині буде водоспад вже завантаженого сигналу. Для того, щоб відобразити водоспад в правій частині екрану необхідно натиснути на кнопку «Завантажити *.wav» і обрати файл, з яким необхідно виконати порівняння.

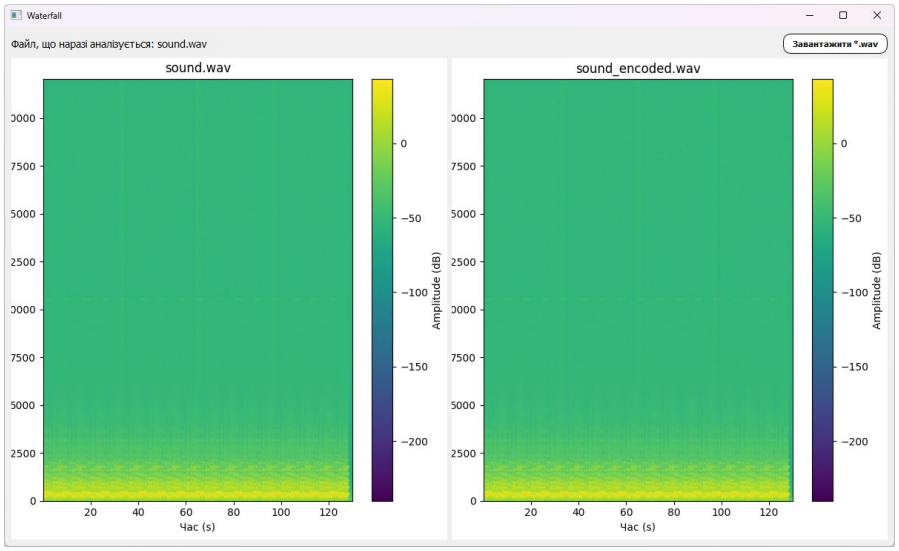

Рисунок Б.8. Порівняння водоспадів

АНОТАЦІЯ

Петручик І.І. Стеганографія в аудіо методом зміни фази звукового сигналу (Phase Coding). Рукопис

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 122 Комп'ютерні науки. Волинський національний університет імені Лесі Українки, Луцьк, 2025 р.

Робота присвячена дослідженню методів стеганографії аудіосигналів та практичній розробці настільного програмного засобу для реалізації методу Phase Coding. У роботі проаналізовано сучасні підходи до приховування інформації в аудіо, зокрема класичне, локальне, сегментне та адаптивне кодування, а також кодування за ключовою фразою. Розглянуто принципи обробки аудіосигналів за допомогою FFT, параметри частотного діапазону та способи підвищення стійкості повідомлення до шумів і перетворень.

Для реалізації програмного засобу обрано мову Python та бібліотеку PyQt5, що дозволило створити кросплатформений графічний інтерфейс користувача. У програмі реалізовано основні функції: кодування та декодування повідомлень у аудіофайлах різними режимами Phase Coding, відображення спектру сигналу, налаштування параметрів сегментації, енергетичних порогів та частотного діапазону.

Результатом роботи є повнофункціональний програмний застосунок, що забезпечує зручне налаштування параметрів стеганографічного процесу, наочну візуалізацію результатів та стабільну роботу з аудіосигналами. Застосунок може бути використаний у навчальних і дослідницьких цілях для аналізу ефективності методів аудіостеганографії. Застосунок експортовано в виконуваний файл для зручності запуску на ПК без Python.

Ключові слова: стеганографія, аудіосигнал, Phase Coding, PyQt5, Python, приховування інформації, FFT.

ABSTRACT

Petruchyk I.I. Steganography in Audio Using Phase Modulation of the Sound Signal (Phase Coding). Manuscript.

Qualification work for obtaining the degree of "Master" in the specialty 122 Computer Science. Lesya Ukrainka Volyn National University, Lutsk, 2025

The thesis is devoted to the study of audio steganography methods and the practical development of a desktop software application implementing the Phase Coding method. The work analyzes modern approaches to hiding information in audio, including classical, local, segmental, adaptive encoding, and encoding using a key phrase. It also examines the principles of audio signal processing using FFT, parameters of the frequency range, and methods to improve message robustness against noise and transformations.

The software implementation was developed using the Python programming language and the PyQt5 library, which enabled the creation of a cross-platform graphical user interface. The program implements the main functions: encoding and decoding messages in audio files using various Phase Coding modes, displaying the signal spectrum, and configuring segmentation parameters, energy thresholds, and frequency ranges.

As a result, a fully functional software application was developed, providing convenient configuration of steganographic parameters, clear visualization of results, and stable operation with audio signals. The application can be used for educational and research purposes to analyze the efficiency of audio steganography methods. The program has been exported as an executable file for convenient use on PCs without the need for a Python environment.

Keywords: steganography, audio signal, Phase Coding, PyQt5, Python, information hiding, FFT.