НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки”

на тему: Виявлення голосової активності в звуковому сигналі

Виконав: студент 4 курсу, групи ТР-52

Скитенко Роман Володимирович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Стативка Юрій Іванович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту
Скитенку Роману Володимировичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Виявлення голосової активності в звуковому сигналі”
керівник роботи доцент, к.т.н. Стативка Юрій Іванович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи графічне відображення результатів виявлення голосової активності в аудопотоці.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати проблематику виявлення голосової активності в звуковому сигналі, провести порівняльний аналіз сучасних програмних рішень виявлення голосової активності, створити та реалізувати алгоритм виявлення голосової активності .
5. Перелік ілюстраційного матеріалу (з точним зазначенням обов’язкових креслень) 1. Постановка задачі та аналіз проблеми виявлення голосової активності, огляд програмних рішень. 2. Засоби розробки та опис програмної реалізації. 3. Інтерфейс користувача. 4. Висновки.

6. Публікації: _____

Дата видачі завдання ”___”_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	Передзахист		
9.	Захист		

Студент _____ Скитенко Р.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стативка Ю.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою роботи була розробка та реалізація алгоритму виявлення голосової активності в звуковому сигналі, що дасть змогу користувачу аналізувати вхідний аудіосигнал в режимі реального часу або завантажити свій аудіофайл і провести його аналіз. Всі обчислення проводяться на основі алгоритмів аналізу енергетичних характеристик сигналу. Після проведення аналізу користувач отримує графічне відображення отриманих результатів. Користувацький додаток представляє собою вікно, за допомогою якого користувач вводить вхідну інформацію та отримує результат.

Ключові слова: виявлення голосової активності, VAD, короткочасні енергетичні характеристики, кількість переходів через нуль.

Записка містить 66 сторінок, 23 рисунки, 2 формули та 25 посилань.

ABSTRACT

The purpose of the work was to develop and implement an algorithm for detecting voice activity in a sound signal, which will allow the user to analyze the incoming audio signal in real time, or download their audio file and analyze it. All calculations are made on the basis of algorithms for analyzing the energy characteristics of the signal. After the analysis, the user receives a graphic representation of the results. The custom application is a window by which the user inputs the input information and receives the result.

Keywords: detection of voice activity, VAD, short-term energy characteristics, number of transitions by zero.

The note contains 66 pages, 23 figures, 2 formulas and 25 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

VAD (від англійської – Voice Activity Detection) – процес виявлення голосової активності в звуковому сигналі.

SNR (від англійської – Signal to Noise Ratio) – відношення сигналу до рівня шуму, поняття, що активно використовується в акустичній електроніці та акустиці.

TASI (від англійської – Time-Assignment Speech Interpolation) – аналогова технологія, що використовується для збільшення потужності передачі голосу на довгих ділянках.

NuGet – менеджер пакетів розроблений компанією Microsoft та застосовуваний в Visual Studio.

ВСТУП

Виявлення голосової активності в звуковому сигналі та виділення ділянок з нею в реальному часі є важливим аспектом обробки записів мови і широко використовується в різноманітних програмах з голосовим інтерфейсом користувача, програмах телекомунікації, системах захисту інформації, програмному забезпеченні для музичних виконавців та багатьох інших областях, пов’язаних з необхідністю виділення в часі ділянок з наявною мовною активністю в аудіосигналах.

Виявлення голосової активності, також відоме як VAD - це процес знаходження голосової активності у вхідному акустичному сигналі для виділення ділянок активного мовлення від фонового шуму або тиші [1]. Сама тема не являється новою: вперше з проблемою стикнулися під час дослідження TASI систем в другій половині минулого сторіччя [3].

Головною метою VAD є розпізнавання та кодування мови. Технологія покликана полегшити процеси обробки мови і використовується для відбору ділянок аудіосигналу чи деактивації деяких процесів на його ділянках під час відсутності мовної активністі. Так, основною перевагою використання методів та алгоритмів виявлення голосової активності в звуковому сигналі є суттєва економія на передачі даних по каналу зв’язку та економія обчислювальних ресурсів, оскільки шум або тиша під час мовлення не кодується і не передається, що призводить до суттєвого зменшення об’єму трафіку, і, через це, до збільшення пропускної здатності каналу зв’язку. При цьому, при виникненні помилок, голос, що інтерпретується як шум може породжувати різноманітні подальші помилки через, так звані, «вирізки з розмови», фоновий шум же, що був інтерпретований як голос, буде призводити до зниження рівня компресії [2]. Такі деталі призводять до певних особливостей алгоритмів виявлення голосової активності, що використовуються як засоби попередньої обробки аудіосигналу в програмах з голосовим інтерфейсом, або телекомукаціях, де повнота висловлювання більш важлива.

Розпізнавання голосової активності є важливою технологією, що дозволяє

використовувати широке коло програм пов’язаних з активним мовленням, тому були розроблені різноманітні алгоритми VAD, які забезпечують різні функції та компроміси між затримкою, чутливістю, точністю та обчислювальними витратами. Деякі алгоритми також забезпечують подальший аналіз, наприклад, визначення емоції диктора, або його ідентифікацію за особливостями мовлення. Виявлення голосової активності зазвичай не залежить від мови.

В той же час, для VAD алгоритмів існує ряд невирішених проблем, серед яких: низька толерантність більшості методів до збільшення рівня навколишнього шуму, необхідна простота алгоритмів, та інші. Також майже невирішеною є проблема відділення музики від голосової активності[1]

За результатами дослідження розроблено алгоритм виявлення голосової активності в звуковому сигналі в реальному часі заснованому на короткочасних енергетичних характеристиках сигналу, та тестуванню програмного коду. Робота містить деталі результатів дослідження, та описує розробку програмного забезпечення, починаючи від розгляду технологій використаних для написання, опису алгоритмів і завершуючи описом використання системи майбутнім користувачем.

Дана робота містить 3 розділи.

У першому розділі описується постановка задачі реалізації алгоритму виявлення голосової активності в звуковому сигналі, її мета, об’єкт і предмет дослідження, завдання дослідження, описуються проблеми та підходи до вирішення проблеми виявлення голосової активності в звуковому сигналі, розглядаються наявні популярні програмні рішення даної проблеми.

У другому розділі вказуються основні засоби розробки даної системи, та описана програмна реалізація алгоритмів виявлення голосової активності.

У третьому розділі описується сценарій взаємодії користувача з програмним продуктом та його недоліки.

1. ПОСТАНОВКА ЗАДАЧІ ТА АНАЛІЗ ПРОБЛЕМИ ВИЯВЛЕНЯ ГОЛОСОВОЇ АКТИВНОСТІ В ЗВУКОВОМУ СИГНАЛІ В РЕАЛЬНОМУ ЧАСІ, ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

За час існування проблеми виявлення голосової активності в звуковому сигналі була напрацьована певна теоретична та практична база вирішення питання, яка є досить важливою для подальшого аналізу проблеми. Далі приводяться постановка задачі та досліження необхідні для вирішення задачі.

1.1 Постановка задачі виявлення голосової активності в звуковому сигналі

Використовуючи теоретичний матеріал, наданий керівником дипломної роботи, розробити та реалізувати методи та алгоритми виявлення голосової активності в звуковому сигналі у вигляді програмної бібліотеки та демонстраційного програмного продукту, за допомогою якої можна проаналізувати вхідний звуковий сигнал на наявність голосу, виявити та виділити окремі ділянки голосової активності. Бібліотека повинна пропонувати для використання декілька методів виявлення голосової активності та можливість легко розширювати та доповнювати її іншими методами та алгоритмами. Програма для демонстрації повинна мати зручний інтерфейс користувача, демонструвати роботу алгоритмів виявлення голосової активності реалізованих в програмній бібліотеці, візуалізувати результати у доступній, інтуїтивно зрозумілій формі, мати можливість працювати як із записаним звуковим файлом, так і з вхідним у реальному часі аудіопотоком.

В результаті розробки програмного продукту, розробнику-користувачу бібліотеки буде надана можливість опрацьовувати різні звукові файли з різною

можливою частотою та каналами запису, виділяти та опрацьовувати ділянки голосової активності. За рахунок цього, програмний продукт розроблений з використанням розробленої бібліотеки зможе вигравати в економії трафіку та швидкодії в порівнянні з аналогами, що не використовують технологію виявлення голосової активності в звуковому сигналі.

Розроблена бібліотека повинна забезпечувати наступні можливості:

— приймати та опрацьовувати передані дані .wav файлу;

— розбивати вхідні дані на фрейми необхідної для роботи довжини, якщо вхідний формат відрізняється від необхідного;

— на основі метаданих вхідного файлу, визначати кількість кадрів та їх величину (кількість байтів у яких зашифрована інформація кадру) для подальшої роботи з ними;

— аналізувати звуковий сигнал на наявність голосової активності та видавати користувачу інформацію про його наявність;

— мати можливість бути легко розширеною без втрати попереднього функціоналу;

— мати можливість бути легко інтегрованою в будь-який програмний продукт, який цього потребує;

Програма для демонстрації повинна мати такі функції:

— можливість працювати з вхідним аудіофайлом в режимі реального часу;

— візуалізувати результати процесу виявлення ділянок з голосовою активністю використовуючи графічне відображення результатів у вигляді графіку або відображення у вигляді числових даних;

— візуалізувати результати роботи кожного з реалізованих методів виявлення голосової активності;

— можливість завантажити .wav файл для аналізу;

— мати можливість відтворити завантажений файл;

— можливість візуалізувати результати виявлення звукової активності на 100мс фреймах завантаженого аудіофайлу;

Для розробки програмної бібліотеки була використана мова програмування C# та платформа .NET Framework 4.7.1. Для представлення графічного інтерфейсу користувача у програмі для демонстації використовувалися засоби Windows Forms.

Метою розробки програмної бібліотеки алгоритмів виявлення голосової активності в звуковому сигналі є створення відкритого програмного продукту, що за своїм функціоналом не буде поступатися комерційним аналогам та буде унікальним прецедентом для платформи .NET Framework.

Мета цього проекту полягає в тому, щоб дати змогу будь-якому розробнику-користувачеві бібліотеки ефективніше та простіше вести розробку програмних продуктів де необхідне виявлення голосової активності на базі .NET Framework, як то, наприклад, контролери основані на голосовому управлінні або засоби аудіокомунікації. Демонстраційний програмний продукт можна використовувати для перевірки записаних аудіофайлів на наявність в них вірогідної голосової активності, виділення ділянок з нею, та аналізу графічного відображення звукового сигналу, що може бути зручним для навчання нейромереж, або виявлення закономірностей залежності відображення фонем та їх звучання за допомогою нейромереж.

1.2 Необхідні критерії детектора голосової активності

Основними критеріями детектора голосової активності в звуковому сигналі вважають надійність, стійкість, точність, адаптивність, простоту, можливість використання в реальному часі, можливість функціонувати без інформації про присутній шум. Відповідно, ідеальний детектор має задовольняти всім вищеперерахованим умовам [4]. Варто зазначити, що ідеального детектора голосової активності не існує.

1.3 Основні теоретичні підходи до вирішення проблеми

За час існування проблеми виявлення голосової активності в аудіопотоці було представлено суттєву кількість алгоритмів чи методів вирішення, кожен спеціалізований під певну проблему і зі своїми перевагами та недоліками. Більшість методик можна поділити на три умовні групи:

— методи засновані на опрацюванні виключно статичних даних аудіопотоку після їх обробки та можливого перетворення (G.729, CMU Sphinx VAD), переважно, мають низький час роботи та невелику довжину вхідного фрейму;

— методи та алгоритми засновані на аналізі статистичних моделей сигналів та шумів (WebRTC VAD);

— застосування навчених нейронних мереж для виявлення голосової активності в звуковому сигналі;

Найбільше поширення здобули алгоритми основані на різноманітних характеристиках сигналу через сукупність своїх характеристик (переважна необхідна простота, достатня надійність та невеликий необхідний розмір вхідного фрейму). Основними відмінностями такого роду детекторів є використовувані характеристики, найчастіше використовувані це:

— короткочасові енергетичні характеристики (short-term energy);
— спектральні характеристики (spectrum based);
— функції автокореляції;
— характеристики потужності на вузькосмугових відрізках;
— характеристики кепстральних коефіцієнтів тональної частоти;

Всі методи вимагають дискретизований звуковий сигнал для аналізу, і в тій чи іншій мірі використовують характеристики сигналу для аналізу, тому всі методи можна вважати похідними від методів заснованих на опрацюванні характеристик сигналу. Вартим уваги є те, що нейронні мережі та деякі просунуті статистичні моделі забезпечують досить високі показники вірного розпізнавання (92.29 – 98.59 вірних показників при +- 5дБ від навчального рівня в середньому для систем заснованих на нейронних мережах), але при цьому використовують фрейми досить великої

довжини, на приклад від 2 до 15 секунд та мають довгий час аналізу, що унеможливлює використання даних методів в режимі реального часу [5].

1.4 Проблеми та актуальність

У проблемі VAD алгоритмів досі існує ряд невирішених питань, основним з яких є стійкість до шуму. Так, в умовах високого SNR (signal-to-noise ratio) навіть найпростіші VAD працюють коректно, однак всі без виключень відомі алгоритми сильно деградують при його (параметра) зниженні, через це зайві 5дБ шуму можуть призвести до серйозних дефектів роботи, а інколи і повної неефективності алгоритмів. Для боротьби з проблемою використовують функції автокореляції та/або приглушення шумів, що допомагає підвищити толерантність алгоритмів виявлення голосової активності до рівня навколишнього шуму та перепадів SNR однак, подібні алгоритми передбачають наявність фонових шумів і тому демонструють свою недієздатність в умовах опрацювання чистого сигналу, через це подібні алгоритми не проходять умову необхідності роботи без попередньої інформації про навколишній шум, та не можуть бути названими універсальними. Як і всі методи, методи автокореляції не можуть аналізувати сигнал із SNR < 1.

Іншою проблемою є необхідна простота алгоритмів, що випливає із вимоги працездатності в режимі реального часу. Тобто, метод має бути досить простим, щоб його алгоритми не вимагали обсягу вираховувальних можливостей (розмір буферу, кількість операцій), які б унеможливлювали роботу із можливою затримкою, яка перевищує поріг реального часу. Відповідно, через це алгоритми з більшою надійністю (деякі статистично-орієнтовані методи, навчені нейронні мережі, тощо), неможливо використовувати у програмному забезпеченні пов’язаному з голосовими інтерфейсами користувача або телекомунікації. В більш загальному вигляді, цю проблему можна охарактеризувати як проблему недостатньої швидкості алгоритмів.

У своєму дослідженні я зосередився на проблемі швидкості VAD алгоритмів реального часу, оскільки такі алгоритми використовуються для попереднього

відділення ділянок з активним мовленням від ділянок шуму та тиші та широко використовується в пристроях з голосовим інтерфейсом керування, програмах шифрування та захисту голосової інформації та інших областях, пов’язаних з необхідністю виділення в часі ділянок в сигналах з мовною активністю, і є досить актуальною темою. Основною їх проблемою є високий рівень фальшивих спрацьовувань та/або високий рівень деградації при зниженні відношення шум/сигнал[8]. Перевагами ж є переважно низький час опрацювання фреймів (зазвичай 5 – 125 мс) [8] та їх невеликий розмір (10 – 100 мс).

1.5 Огляд загальних підходів програмної реалізації виявлення голосової акивності в звуковому сигналі

Популярними нині рішеннями при написанні проектів які вимагають виявлення голосової активності є використання вкладених бібліотек VAD аналізаторів різних відкритих звукових кодеків, як то GSM.729 або Skype SILK, чи бібліотек відкритих програмних продуктів, що спеціалізуються на опрацюванні звуку, таких як CMU Sphinx. Це спричиняє ряд незручностей для розробників, оскільки подібні VAD детектори подеколи орієнтовані на спеціалізований звуковий сигнал у певному діапазоні та певному рівні дискретизації, що не завжди є прийнятним. Іншим же негативним фактором подібної практики є необхідність наявності потрібних кодеків на комп’ютерах кінцевого користувача чи підвищення вимог до програмного продукту за рахунок інтеграції кодеків напряму в програмний продукт.

1.6 CMU Sphinx

CMU Sphinx, який також коротко називають Sphinx (Сфінкс), є загальним терміном, що описує групу систем розпізнавання мови, розроблених в Університеті Карнегі-Меллона. До них відносяться ряд розпізнавальних мов (Sphinx 2 - 4) і тренажер акустичної моделі (SphinxTrain). У 2000 році група Сфінкс в Карнегі-

Меллоні зобов'язалася відкрити декілька компонентів розпізнавання мовлення, включаючи Sphinx 2 і пізніше Sphinx 3 (у 2001 році). Декодери мовлення надходять з акустичними моделями та зразками додатків. Додаткові ресурси включають додаткове програмне забезпечення для навчання акустичної моделі, компіляції мовної моделі та словника з вимовою у вільному доступі, cmudict.

Методи детектування голосової активності реалізовані у сфінксі використовують алгоритм Брента Шмідта Нільсена. Кожен раз, коли звук надходить, середній рівень сигналу та рівень фонового шуму оновлюються, використовуючи рівень сигналу поточного звуку. Якщо середній рівень сигналу перевищує рівень фонового шуму на певне граничне значення (конфігурується), то поточний звук позначається як мова.

Відкритість коду допомагає легко інтегрувати бібліотеку алгоритмів виявлення голосової активності в звуковому сигналі Sphinx у власний код (рисунок 3.1.).

```
SpeechClassifier s = new SpeechClassifier();

s.setPredecessor(dataSource);
Data d = s.getData();

while(d != null) {
    if(s.isSpeech()) {
        System.out.println("Speech is detected");
    }
    else {
        System.out.println("Speech has not been detected");
    }

    System.out.println();
    d = s.getData();
}
```

Рисунок 3.1. – Приклад інтеграція VAD бібліотеки Sphinx в код проекту.

Суттєвим же недоліком потрібного методу, окрім описаних вище, є низька якість реалізації детектору голосу у наявній збірці, оскільки сам продукт використовується для дещо інших цілей, і алгоритм детектора голосової активності не є для нього пріоритетним у розробці та модернізації.

1.7 GSM.729

GSM.729 - це алгоритм стиснення аудіоданих на основі вузькосмугового кодера голосу, вільний від роялті, який використовує довжину кадру 10 мілісекунд. Офіційно описується як кодер мови на рівні 8 кбіт / с, використовуючи кодове збудження лінійного передбачення мовного кодування (CS-ACELP). Через низькі вимоги до пропускної здатності, GSM.729 в основному використовується в голосових протоколах (VoIP), коли смуга пропускання повинна бути збережена, наприклад, для конференц-дзвінків. Стандарт GSM.729 працює з бітрейтом 8 кбіт / с, але розширення забезпечують швидкості 6,4 кбіт/с і 11,8 кбіт/с для гіршої і кращої якості мовлення, відповідно.

Для виявлення голосової активності, алгоритми GSM.729 використовують лінійний спектр пари частот, full-band energy та low-band energy, zero-crossing rate і використовує класифікатор з використанням фіксованих меж в обмеженому просторі [6].

Недоліками використання вищеописані, окрім вказаних, варто зазначити необхідність повністю інтегрувати кодек у програмний продукт, що, через його обсяг, збільшить необхідні для роботи характеристики комп’ютера.

1.8 WebRTC VAD

Технологія WebRTC (англ. real-time communications — комунікація в реальному часі) — інтернет-протокол із відкритим кодом, призначений для організації голосового та відеозв'язку через інтернет у режимі реального часу розроблений компанією Google та розповсюджуваний за ліцензією BSD-3. Він включає в себе основні складові для високоякісної комунікації в Інтернеті, такі як різноманітні мережеві, аудіо- та відеокомпоненти, що використовуються в голосових і відео-чатах.

Протокол включає в себе відкриту бібліотеку VAD методів, яка на момент написання цієї роботи вважається однією з найкращих відкритих. Методи виявлення голосової активності у вхідному акустичному сигналі, що використовуються в цьому протоколі, засновані на аналізі сигнала за допомогою змішаної моделі Гаусіана (Gaussian mixture model) з автокореляцією після кожного фрейму та працюють з проміжками до 30 мс (рисунок 3.2.).

```
262
263       // Calculate local speech probabilities used later when updating the GMM.
264       h1 = (int16_t) (h1_test >> 12);  // Q15
265       if (h1 > 0) {
266         // High probability of speech. Assign conditional probabilities for each
267         // Gaussian in the GMM. Otherwise use the initialized values, i.e., 0.
268         tmp1_s32 = (speech_probability[0] & 0xFFFFF000) << 2;  // Q29
269         sgprvec[channel] = (int16_t) WebRtcSpl_DivW32W16(tmp1_s32, h1);  // Q14
270         sgprvec[channel + kNumChannels] = 16384 - sgprvec[channel];
271       }
272     }
273
274     // Make a global VAD decision.
275     vadflag |= (sum_log_likelihood_ratios >= totalTest);
276
277     // Update the model parameters.
278     maxspe = 12800;
279     for (channel = 0; channel < kNumChannels; channel++) {
280
```

Рисунок 3.2. – Приклад коду з кореляцією результатів для наступної ітерації, та винесенням остаточного рішення по перевіряємому фреймі методом WebRTC VAD.

Недоліками вищеописаного детектора голосової активності є необхідність встановлення всього пакету WebRTC API, який має досить великий обсяг та високі вимоги до характеристик комп’ютера, що унеможливлює його викоритсання у прикладних десктопних програмах, або як локального детектора голосової активності у програмах з голосовим інтерфейсом користувача.

1.9 Opus codec VAD

Аудіокодек Opus — вільний та відкритий звуковий кодек, прийнятий у вересні 2012 Internet Engineering Task Force (IETF) як стандартний аудіо-кодек для інтернет-застосунків [7].

VAD методи, реалізовані в кодеку, працюють за наступним алгоритмом: для кожного фрейму вираховується рівень активності голосу, спектральний нахил і SNR. Для цього кожен фрейм розбивається на діапазонах частот 0 - Fs / 16, Fs / 16 - Fs / 8, Fs / 8 - Fs / 4, а Fs / 4 - Fs / 2, де Fs - частота дискретизації (8, 12, 16) , 24 кГц). Діапазон 0 - Fs / 16 фільтруется фільтром (передаточна функція H (z) = 1-z ^ (- 1) - Moving Average) для зменшення енергії на найнижчих частотах (рисунок 3.3.). На кожному діапазоні вираховується енергія. На основі цієї енергії рахуємо наступні характеристики: SNR - середньостатистичні діапазони, рівень активності голосу - оснований на SNR і середньостатистичній сумі енергетичних діапазонів, спектральних навантажень - середній діапазон значень SNR. Вираховується та виконується автокореляція і на її підстави і підставі вирахуваних раніше значень і відбувається остаточна класифікація.

Недоліки цього методу ідентичні до недоліків WebRTC VAD, окрім цього архітектура програмного рішення ускладнює доступ до потрібних бібліотек, що може стати ще одним мінусом у використанні даного рішення в програмному продукті.

```
139     /*****************************************/
140     /* HP filter on lowest band (differentiator) */
141     /*****************************************/
142     X[ decimated_framelength - 1 ] = silk_RSHIFT( X[ decimated_framelength - 1 ], 1 );
143     HPstateTmp = X[ decimated_framelength - 1 ];
144     for( i = decimated_framelength - 1; i > 0; i-- ) {
145         X[ i - 1 ]  = silk_RSHIFT( X[ i - 1 ], 1 );
146         X[ i ]     -= X[ i - 1 ];
147     }
148     X[ 0 ] -= psSilk_VAD->HPstate;
149     psSilk_VAD->HPstate = HPstateTmp;
150
```

Рисунок 3.3. – Приклад коду з реалзіацією фільтрації низьких частот в Opus codec VAD.

Незважаючи на недоліки, даний детектор вважається одним з найкращих відкритих програмних рішень, і широко використовується у некомерційних програмних продуктах.

1.10 Висновки до розділу

У даному розділі поставлена задача роботи, було наведено проблему виявлення голосової активності в звуковому сигналі, наведено головні критерії яким має відповідати ідеальний детектор голосової активності, оглянуто основні теоретичні шляхи та методи вирішення поставленого питання, розглянуто існуючі проблеми в даній темі та обгрунтовано підхід у виборі реалізовуваного набору алгоритмів для створення VAD методу, було розглянуто найпопулярніші відкриті програмні рішення для виявлення голосової активності у вхідному акустичному сигналі, описано алгоритми даних програмних рішень, принципи їх роботи, позитивні та негативні аспекти їх використання для побудови власних програм з використанням технології виявлення голосової активності.

2. ЗАСОБИ РОЗРОБКИ ТА ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ВИЯВЛЕННЯ ГОЛОСОВОЇ АКТИВНОСТІ В ЗВУКОВОМУ СИГНАЛІ

Важливим чинником, під час розробки програмного продукту, є вибір засобів програмної реалізації та технологій. Середовищем розробки інформаційної системи було обрано Microsoft Visual Studio Community 2017. Для створення графічного інтерфейсу для демонстрації програмної бібліотеки використовувався інтерфейс програмування додатків Windows Forms, що є частиною Microsoft .NET Framework.

Для розробки алгоритмів використовувалась об’єктно-орієнтована мова програмування C#.

Для зчитування вхідної інформації в систему та розбиття вхідних звукових потоків на фрейми потрібної довжини використовувалася відкрита бібліотека NAudio завантажена за допомогою NuGet.

Програмна реалізація представляє собою виконуваний файл додатку, що забезпечує відповідний функціонал, та API у вигляді DLL бібліотеки з класами необхідними для виявлення голосової активності в звуковому сигналі.

2.1 Середовище розробки Visual Studio 2017

Середовище розробки Visual Studio 2017 дозволяє ефективно, якісно і швидко писати код, при цьому не втрачаючи уваги з контексту поточного файлу. За допомогою Visual Studio можливо легко переглянути структуру виклику та пов’язані функції, повернені значенні та стан тестування. Також можна провести рефакторинг коду, знайти помилки та отримати варіанти їх усунення[9].

На сьогоднішній день продукти Microsoft з сімейства Visual Studio містять в собі багатий інструментарій. У сімействі Visual Studio 2017 є IDE,

мультиплатформний редактор коду Visual Studio Code, який доступний для систем на базі операційних систем Windows, Mac та Linux, зручний сервіс для організації командної роботи – Visual Studio Team Services. Також ведеться розробка рішення для користувачів під платформою Mac OS – Visual Studio for Mac.

З кожним роком інструментарій Visual Studio оновлюється і постійно розширюється. А компанія Microsoft постійно намагається покращити зручність середовища розробки, з урахуванням побажань розробників з усього світу, які користуються Visual Studio. В результаті понад 30 мільйонів завантажень на сьогоднішній день, що робить цей продукт одним із найпопулярніших серед розробників додатків.

Кожен з розробників може завантажити собі 60-денну безкоштовну версію Visual Studio 2017 і повністю ознайомитись з повним функціоналом середовища розробки, протестувати його на власному досвіді і зробити для себе певні висновки. А потім, в залежності від потреб, лишитися на версії Community, якщо інструментарій використовується лише для освітніх цілей, або перейти на платну версію в тому разі, коли розроблений програмістом додаток буде використовуватись як комерційний продукт.

Варто виділити і модульний підхід до процесу установки інструменту. Серед великої кількості різних модулів, що дозволяють програмісті розробляти додатки різного рівня і для різних платформ, тепер можна обрати лише ті, що потрібні конкретно для реалізації даної задачі. Це прискорює процес установки, а отже і загальний час, що витрачається на розробку програмного продукту загалом.

Ще одним важливим нововведенням для сімейства Microsoft Visual Studio стала інтеграція з хмарними сервісами платформи Azure. Ця інтеграція дозволяє полегшити створення, налаштування і публікацію розроблених програмних рішень в хмарі Azure напряму із середовища IDE.

У листопаді 2016 року почалася розробка Visual Studio for Mac. На даний момент вже відбулося декілька preview-випусків цього середовища розробки для Mac OS від Microsoft. Це середовище характеризується розробкою додатків для

смартфонів, хмарних рішень, а також програмних рішень для операційної системи Mac OS. Компанією ведеться постійна робота над оптимізацією та виправленням помилок, з урахуванням побажань розробників. Вже були додані підтримка NuGet та .NET Core.

2.2 Windows Forms

Windows Forms – інтерфейс програмування додатків, який відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. За допомогою цього інтерфейсу, в середині середовища розробки ми можемо отримати доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого додатку в керованому коді. А за рахунок того, що керований код – це класи, що реалізують додаток Windows Forms і вони не залежать від мови розробки, програміст може спокійно використовувати Windows Forms при підготовці програмного забезпечення як на C++, C#, так і на J# або VisualBasic.Net та інших мовах [10].

В наш час Windows Forms використовується як замінник старої і складнішої бібліотеки MFC, що була написана на мові програмування C++. Але розглядаючи Windows Forms варто зазначити, що вона не пропонує парадигму, як це робила MFC.

Виправити цю ситуацію покликані сторонні бібліотеки, які існують у Windows Forms. Головною є бібліотека User Interface Process Application Block, що використовується найчастіше серед розробників додатків на базі Windows Forms. Вона була випущена групою програмістів Microsoft, які займаються саме прикладами та рекомендаціями. User Interface Process Application Block безкоштовна та доступна для скачування з офіційного сайту. Разом з нею можна знайти вихідний код та приклади, які покращать розуміння бібліотеки та прискорять навчання.

Додаток Windows Forms – дієво-орієнтовний, а отже виконання програми будується на певних подіях, які підтримуються Microsoft .NET Framework. Це можуть бути дії користувача, повідомлення від інших додатків і програм, події операційної

системи та інше. Це спрощує розуміння логіки вже готового програмного продукту, а отже і процес роботи з програмним продуктом взагалі, що є плюсом для користувачів.

2.3 NAudio

Бібліотека NAudio - це API з відкритим вихідним кодом для .NET, написаний на мові C # Марком Хітом. Він призначений для надання повного набору корисних класів утиліт, для побудови звукових додатоків. NAudio ліцензовано під ліцензією Microsoft Public License (Ms-PL), що дозволяє використовувати її в будь-якому проекті, включаючи комерційні розробки.

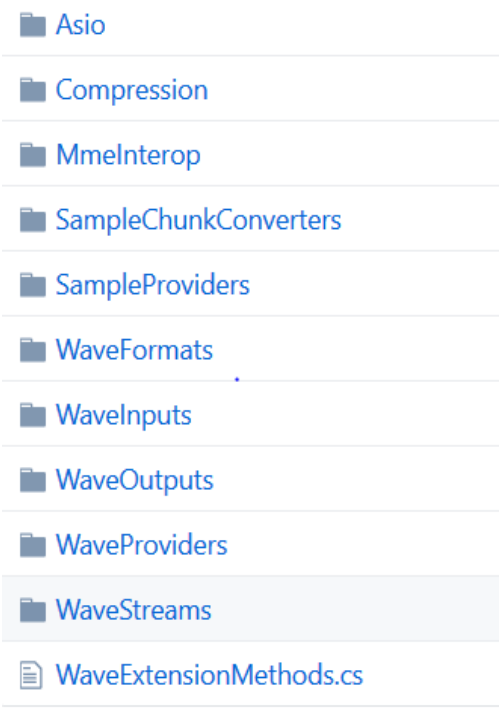

Рисунок 2.1. – Структура простору імен NAudio.Wave бібліотеки NAudio.

При написанні програмного продукту методи бібліотека використовувалися для організації запису та відтворення звукових файлів, що аналізуються, або захоплення вхідного звукового потоку. Для цього були використані методи класів NAudio.Wave (рисунок 2.1.), що відповідає за роботу з аудіоданими формату .wav, який був обраний за гнучкість у питанні запису звуку та відсутність стиснення файлу.

Окрім того, бібліотека надає можливості відтворення різноманітних аудоформатів за допомогою вбудованого API (WaveOut, DirectSound, ASIO, WASAPI), читання аудіофайлів з багатьох популярних форматів (WAV, AIFF, MP3, G.711, ADPCM, WMA, AAC, MP4), перетворення між різними формами нестисненого звуку, кодування аудіо за допомогою кодеків ACM встановлених на комп’ютері.

2.4 Опис функціональності програмного забезпечення

Можливості додатку демонструються на діаграмі прецендентів (use case діаграмі) (рисунок 2.2.).

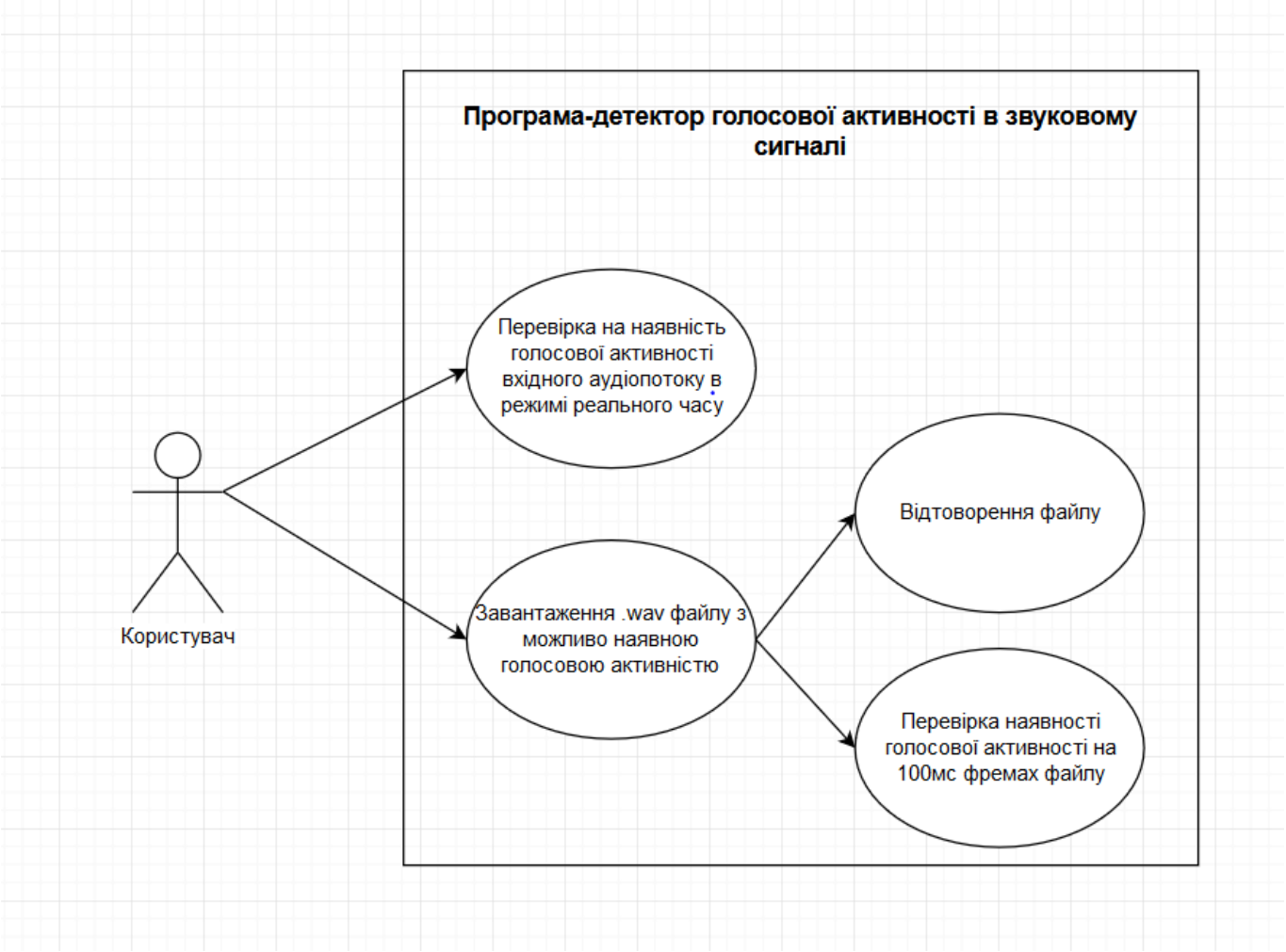

Рисунок 2.2. – Діаграма прецендентів.

Діграма прецендентів описує відношення між акторами та прецендентами в

системі. Суть діаграми полягає в представленні проектованої системи у вигляді сутностей чи акторів, що взаємодіють із системою за допомогою можливих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Тобто, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором [11].

2.5 Реалізація взаємодії із вхідними аудіоданими

Людським організмом звук сприймається як неперервний потік, але для його обробки, звуковий потік необхідно дискретизувати, відповідно частота дискретизації, або, іншими словами, кількість замірів за одиницю часу є однією з найважливіших характеристик аудіофайлу. При обробці вхідного аудіопотоку в реальному часі частота дискретизації відповідає за розмір буфера, що необхідно буде опрацювати.

Як зазначалося вище, оптимальним для аналізу на наявність голосової активності був обраний формат звукового файлу .wav, у ньому дані звукового потоку зберігаються в масиві даних формату byte, розмір якого залежить від частоти дискретизації, кількості каналів запису (один або два, відповідно, 2 або 4 байти на 1 кадр), та, власне, тривалості файлу (рисунок 2.3.). Окрім власне даних звуку, .wav-файл має, так названі, дані заголовку, що зберігають в собі інформацію про формат, кількість каналів запису, частоту дискретизації, кількість байтів в масиві даних та інше, загальним обсягом завжди 44 байти (рисунок 2.3.).

При опрацюванні завчасно записаного зразку, з обраного файлу формату .wav, шляхом відкидання даних заголовку байти, знаходиться масив з основною інформацією про звук цього файлу, який потім, на основі вказаної в заголовках інформації про частоту дискретизації та кількість каналів запису, розбивається на масиви інформації фреймів в 100 мс, що більш зручно для графічної демонстрації аніж необхідні для роботи методів фрейми довжиною 10 мс (рисунок 2.4.).

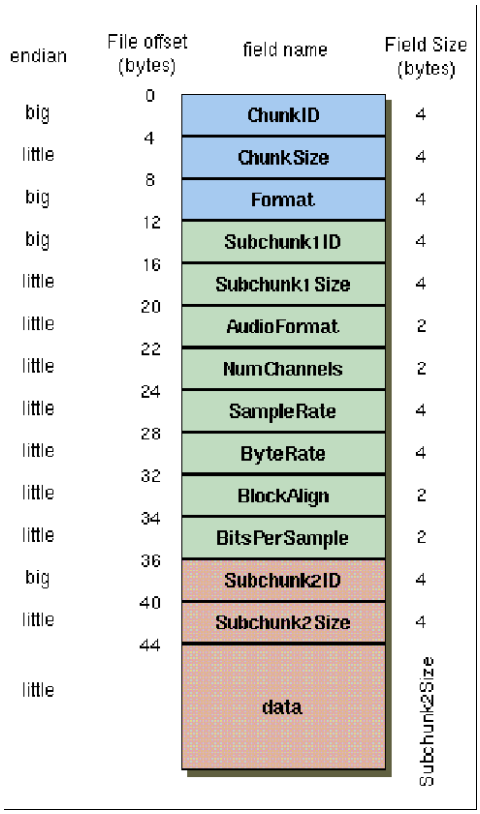

Рисунок 2.3. – Формат даних .wav файлу.

```
OpenFileDialog opf = new OpenFileDialog();
            if (opf.ShowDialog() == DialogResult.OK)
            {
                filePath = opf.FileName;
                byte[] bytes = File.ReadAllBytes(filePath);
                var b = bytes.ToList();
                b.RemoveRange(0, 44);
                bytes = b.ToArray();
                int counter = 0;
                bytes_arr = bytes.GroupBy(_ => counter++ /
1600).Select(elem => elem.ToArray()).ToArray();
                framesDictionary = new Dictionary<string,
byte[]>();
                counter = 0;
                foreach(var i in bytes_arr)
                {
   framesComboBox.Items.Add($"Frame{counter}");
   framesDictionary.Add($"Frame{counter++}", i);
                }
             }
```

Рисунок 2.4. – Приведення до необіхдного вигляду .wav файлу з частотою дискретизації 8000 Гц.

При опрацюванні вхідного аудіопотоку в реальному часі, загальний алгоритм лишається тим же самим, але, реалізовується простіше, оскільки методи бібліотеки NAudio дозволяють напряму вказати необхідний розмір буфера та отримувати необхідного розміру масив байтів напряму через аргумент обробника подій (рисунок 2.5.).

```
void Data_Avaible(object sender, WaveInEventArgs e)
        {
            sted.fraim = e;
            stzcd.fraim = e;

            bool enVoice = sted.VoiceIsPresent;
            bool stVoice = stzcd.VoiceIsPresent;

            if (enVoice || stVoice) boolPanel.BackColor =
Color.Green;
            else boolPanel.BackColor = Color.Red;

            stedArrays = sted.energyArraysList;

            for(int i = 0; i < 10; i++)
            {
                labels[i].Text =
    stzcd.arr[i].count.ToString();
                if (stzcd.arr[i].result) labels[i].BackColor
= Color.Green;
                else labels[i].BackColor = Color.Red;
            }

            Panel.Refresh();
         }
```

Рисунок 2.5. – Метод, що приймає аргумент обробника подій (WaveInEventArgs), що містить в собі масив байтів буфера вказаної розмірності.

Процес дискретизації звукового потоку є першим кроком до процесу будь-якої обробки звуку за допомогою комп’ютера, і виконується в будь-яких програмах, що працюють зі звуком.

2.6 Знаходження енергії кадрів фрейму

Наступним кроком після дискретизації та вичленення даних із файлу звуку або вхідного аудіопотоку є знаходження енергії кожного кадру вхідного фрейму, що здійснюється за допомогою операції об’єднання байтів, які зберігають інформацію про цей кадр (рисунок 2.5.).

```
short first = (short)(( buffer[1] << 8) |  buffer[0]);
```

Рисунок 2.6. – Знаходження енергії кадру закодованого у двохбайтному форматі.

Процес знаходження енергії кадрів фрейму звукового потоку оснований на метаданих цього аудіопотоку і винесений в абстрактний клас детектора (рисунок 2.7.).

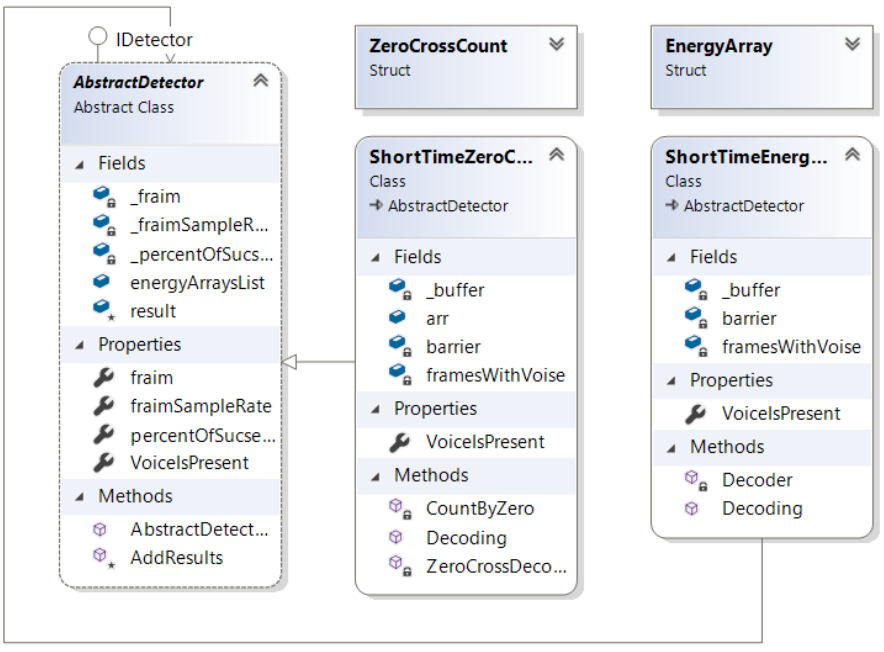

Рисунок 2.7. – Структура класів-детекторів голосової активності.

Знаходження енергії кадрів аудіосигналу необхідий крок для подальшої обробки фрейму методами класів-детекторів голосової активності.

2.7 Метод короткочасних енергетичних характеристик сигналу

Основним методом виявлення голосової активності в даній роботі є метод короткочасних енергетичних характеристик сигналу, який поєднує в собі два популярні методи, такі як виявлення активності шляхом порівняння енергії кадру зі сталою, та метод короткочасної енергії фрейму сигналу.

Метод порівняння енергії кожного кадру зі сталою є одним з найпростіших методів виявлення активності в звуковому сигналі, і заключається в порівнянні енергії кожного конкретного кадру аналізованого фрейму з початково заданим пороговим значенням. Через свою надмірну просто, не може бути використаним як самостійний метод тому використовується для додаткової перевірки методу короткочасної енергії фрейму сигналу (рисунок 2.8.).

```
bool Decoder(byte[] _buffer, double barrier)
          { double Sum2 = 0;
            bool Tr = false;
            for (int index = 0; index < _buffer.Length;
    index += 2)
            {
                double Tmp = (short)((_buffer[index + 1] <<
8) | _buffer[index]);
                energyArray ._energyList.Add((short)Tmp);
                Tmp /= 32768.0;
                Sum2 += Tmp * Tmp;
                if (Tmp > barrier) Tr = true;
            }
            Sum2 /= _buffer.Length / 2;

            if (Tr || Sum2 > barrier)
                return true;
            else
                return false; }}
```

Рисунок 2.8. – Метод-декодер.

Метод короткочасної енергії фрейму сигналу заснований на знаходженні сумарної енергії фрейму за допомогою формули квадратичного перетворення:

$$\varepsilon = \sum_{n=0}^{N-1} (x(n)^2)$$

Де N – кількість кадрів у фреймі, а n – порядковий номер кадру. Знайдене значення енергії фрейму ділиться на максимальне можливе значення для отримання відносного показника енергії фрейму (рисунок 2.8.), отримане значення порівнюється з пороговим, яке задається експериментально для кожного окремого середовища, на основі порівняння і виноситься остаточне рішення про наявність голосової активності на даному сегменті (рисунок 2.9.).

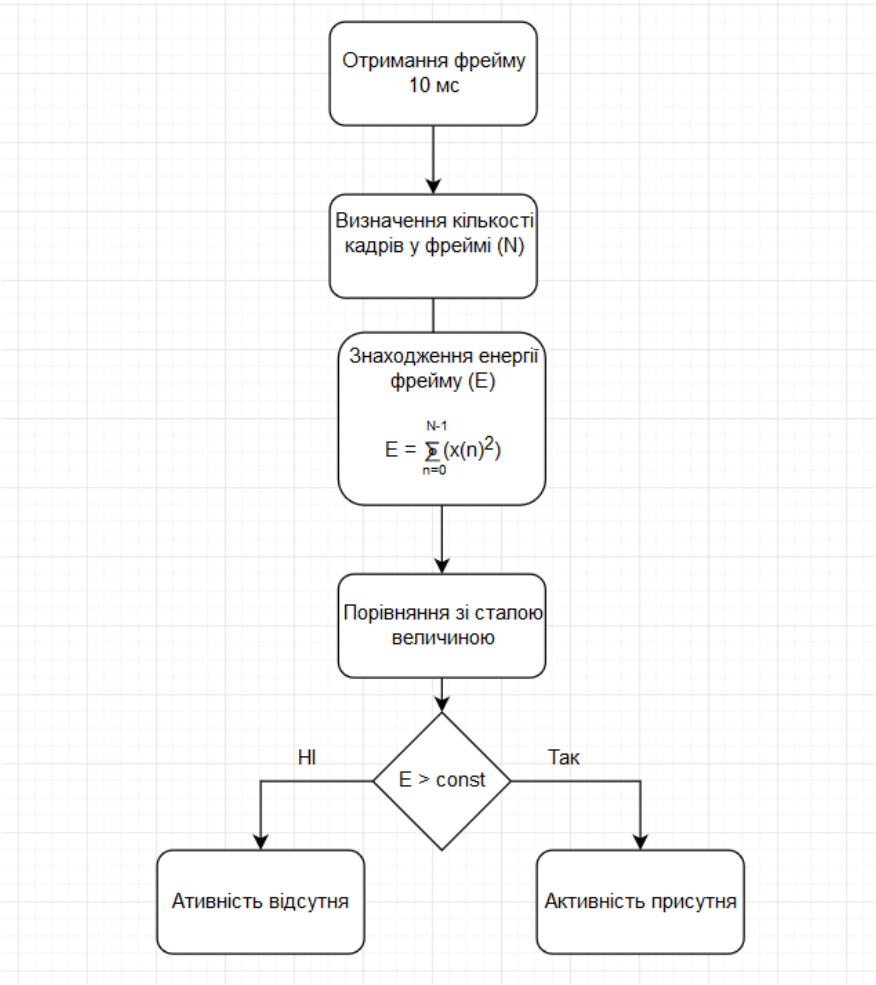

Рисунок 2.9. – Алгоритм методу енергії фрейму.

Як зрозуміло з назви метод працює з фреймами довжиною в 10мс. Сам по собі метод не є досить точним, але через роботу з відносно короткими фреймами надзвичайно гнучкий до модернізації, що дозволяє значно зменшити рівень фальшивих спрацьовувань. Недоліками методу є сильна деградація при зменшенні

відношення сигнал/шум та можливі фальшиві спрацьовування на довготривалі (більше 40 мс) голосні шуми (на приклад, оплески). Також вагомим недоліком даного методу є відсутність чутливості до низькоенергетичних звуків людського мовлення (шиплячі та свистячі звуки), для вирішення цієї проблеми застосовується метод кількості переходів через нуль.

2.8 Метод кількості переходів через нуль

Підрахунок кількості переходів через нуль представляє собою широко застосовуваний метод виявлення голосової активності, що базується на підрахунку кількості переходів енергетичних значень кадрів фрейму через нульову відмітку. Виходячи із статистичних особливостей людського мовлення і особливостей звучання шиплячих та свистячих звуків (які є низькоенергетичними та високочастотними), даний метод чудово підходить для виявлення сам цих аспектів людського голосу.

Підрахунок кількості переходів через нуль відбувається за наступною формулою:

$$S = \sum_{n=1}^{N-1} \left| \operatorname{sgn}\left(x(n)\right) - \operatorname{sgn}\left(x(n-1)\right) \right|$$

Де N – кількість кадрів у фреймі, n – порядковий номер кадру, x(n) – значення енергії кадру n. Отримане значення порівнюється із пороговим, яке задається на основі особливостей людського мовлення, та дорівнює близько 30 переходів через нуль на 10мс фрейм для задачі виявлення високочастотних звуків діапазону людського мовлення, на основі порівняння виноситься рішення про наявність такого роду звуків у звуковому ряді, а отже і вірогідного людського мовлення (рисунок 2.10).

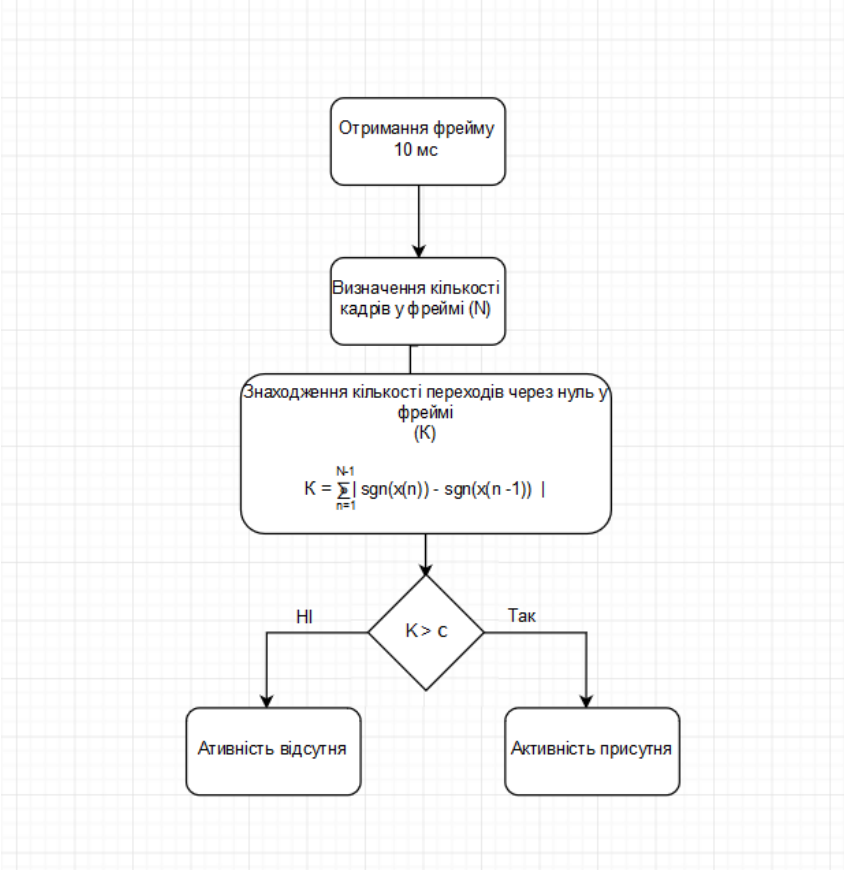

Рисунок 2.10. – Алгоритм методу підрахунку кількості переходів через нуль.

Використання методів короткочасних енергетичних характеристик та короткочасних переходів через нуль дозволяє з високою швидкістю ідентифікувати ділянки з вірогідною голосовою активністю (рисунок 2.11.).

```
            sted.fraim = e;
            stzcd.fraim = e;

            bool enVoice = sted.VoiceIsPresent;
            bool stVoice = stzcd.VoiceIsPresent;

            if (enVoice || stVoice) boolPanel.BackColor =
Color.Green;
                else boolPanel.BackColor = Color.Red;
```

Рисунок 2.11. – Прийняття рішення на основі результатів двох детекторів.

Рішення про наявність голосової активності на фреймі вирішується шляхом диз’юнкції результатів двох методів класів-детекторів голосової активності.

2.9 Додаткові покращення методу

Зважаючи на гнучкість методу короткочасних характеристик сигналу, та його вразливості до високоенергетичних короткочасних шумів та особливостями людської мови було вирішено покращити метод додатковим параметром перевірки тривалості активності (рисунок 2.12).

```
protected void AddResults(short count)
    {
        if (count >= FramesWithVoiceActivityRow)
            result = true;
    }
```

Рисунок 2.12. – Фільтрація надто коротких вірогідних ділянок голосової активності.

Таким чином введена фільтрація ділянок активності які сумарно менші по протяжності за 50мс, оскільки голосова активність не може бути настільки короткою. Таке рішення дозволяє ігнорувати деякі високоенергетичні, але короткі по протяжності шуми, що в цілому сприяє надійності та ефективності системи.

2.10 Висновки до розділу

У даному розділі було описано основне програмне забезпечення, програмні додатки та бібліотека програмного коду, що були використані для реалізації програмної частини роботи, а також аргументовані причини їх вибору, проілюстрована функціональність системи, наведена архітектура бібліотеки детекторів голосової активності, описані особливості дискретизації та передобробки звуку, розібрані алгоритми класів-детекторів голосової активності, та наведена реалізація додаткової постобробки інформації наданої з детекторів.

3. ВИПРОБУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

Використовуваний алгоритм було розроблено з метою максималізації швидкості обробки вхідного сигналу у порівнянні з аналогічним програмним забезпеченням. Основною відмінністю від аналогів є саме швидкість опрацювання даних (рисунок 3.1.). Додаток було розроблено з акцентом на простоту та зручність у використанні клієнтом, та візуалізацію отриманих результатів для зручності користувача.

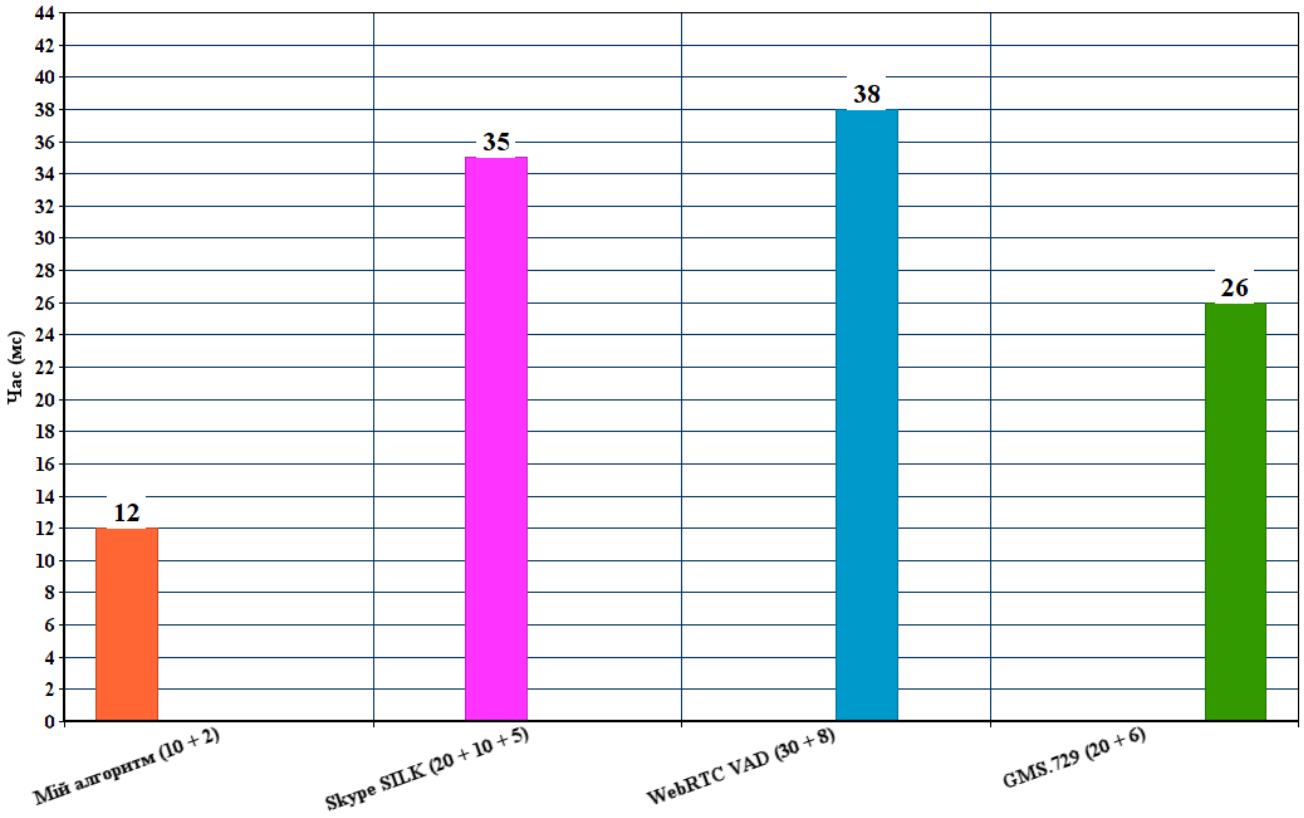

Рисунок 3.1. – Діаграма порівняння часу роботи алгоритму з доступними аналогами.

Відповідно, алгоритм через його окремі недоліки доцільно використовувати як перший фільтруючий елемент, передаючи виявлені ним ділянки голосової активності на більш точні, але повільні алгоритми для повторного аналізу, що дозволить знизити час і збільшити точність роботи такої системи в цілому.

3.1 Сценарій взаємодія користувача з системою

При відкритті додатку перед користувачем з’явиться вікно у якому буде три кнопки: «Start», «Get .wav file» та «Play» (рисунок 3.2.).

Рисунок 3.2. – Вікно додатку

При натисненні на кнопку «Start», вона зміниться на кнопку «Stop», додаток почне зчитувати вхідний аудіопотік з мікрофона та аналізувати його на наявність голосової активності (рисунок 3.3.). При цьому на графіку у лівій частині вікна демонструється графічне відображення поточного звукового ряду тривалістю в 100мс (тобто 10 фреймів по 10мс) для зручності відображення та перегляду, 10 полів з числовими значеннями в нижній частині екрану демонструють відповідно кількість переходів через нуль у цих 10 фреймах відображених на графіку, для зручності сприйняття, змінюючи колір на зелений, коли кількість переходів більша за необхідну для ідентифікації сегменту як такого, що містить голосову активність, і, відповідно, червоний, якщо кількості переходів через нуль недостатньо.

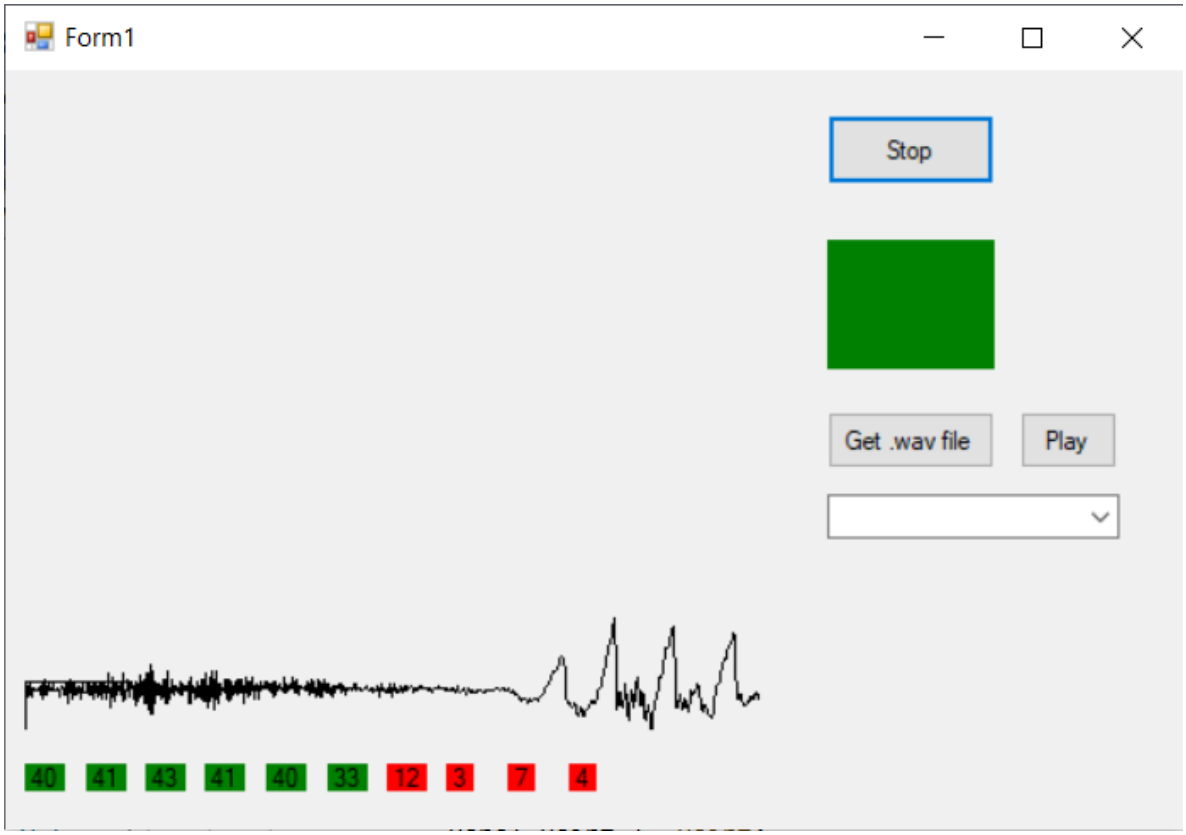

Рисунок 3.3. – Процес аналізу вхідного потоку.

Кольорова панель під кнопкою «Stop» демонструє наявність голосової активності на поточному сегменті, відображаючи цей процес кольором: зелений означає, що голосова активність присутня, червоний – відсутня.

Для графічного відображення спрацьовувань методу короткочасної енергії сигналу на графік нанесена шкала-орієнтир яка залежить від заданого порогового значення (рисунок 3.4.). Перехід шкали графіком означатиме високу вірогідність інтерпретації фрейму таким, що має голосу активність.

При натисненні кнопки «Get .wav file» користувач побачить діалогове вікно, через яке може задати .wav файл для аналізу (рисунок 3.5.).

Після вибору файлу з’являється можливість його відтворення за допомогою кнопки «Play» (до вибору файлу натиснення на «Play» нічого б не дало, оскільки вона була не активною), та заповнюється стамілісекундними фреймами ListBox під цими двома кнопками (рисунок 3.7.).

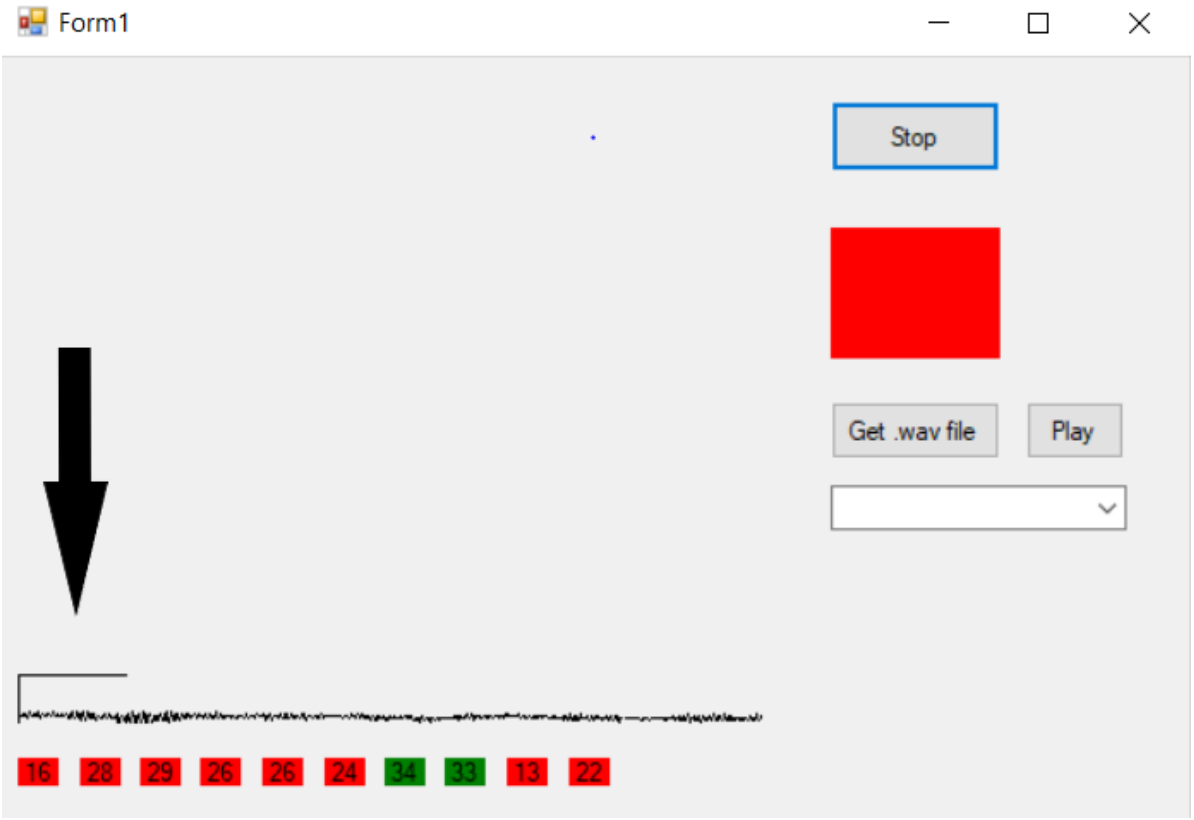

Рисунок 3.4. – Шкала-орієнтир.

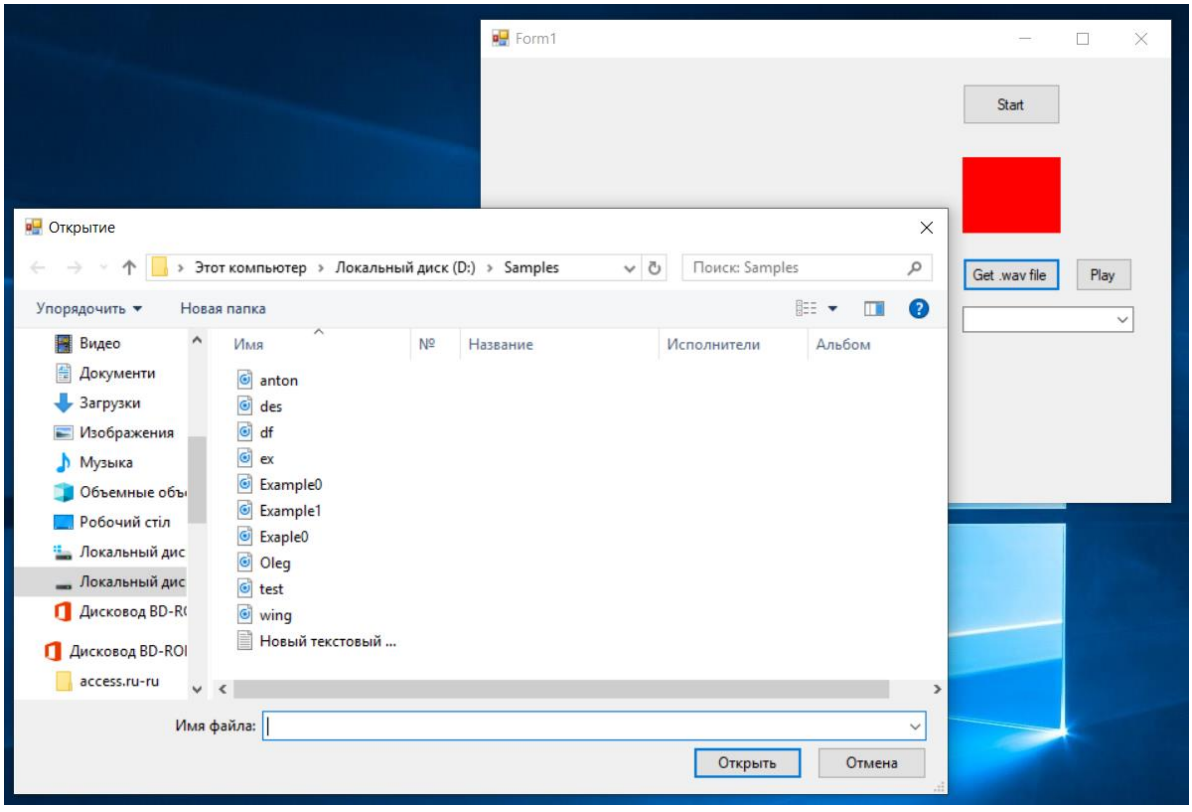

Рисунок 3.5. – Діалогове вікно.

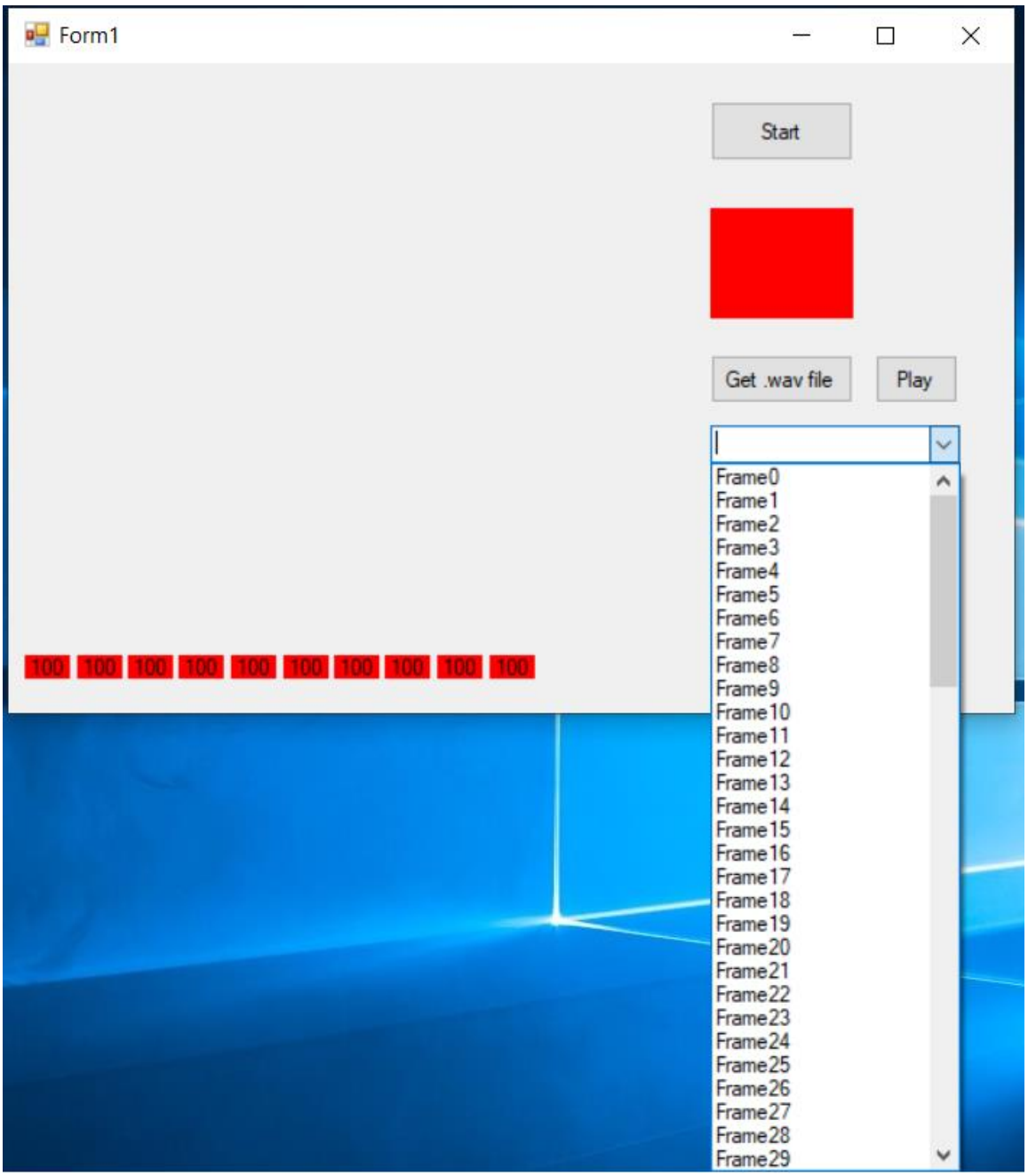

Рисунок 3.7. – ListBox з фреймами обраного файлу.

При виборі довільного фрейму можна відобразити його в графічному режимі для перегляну на наявність голосвої активності (рисунок 3.8.).

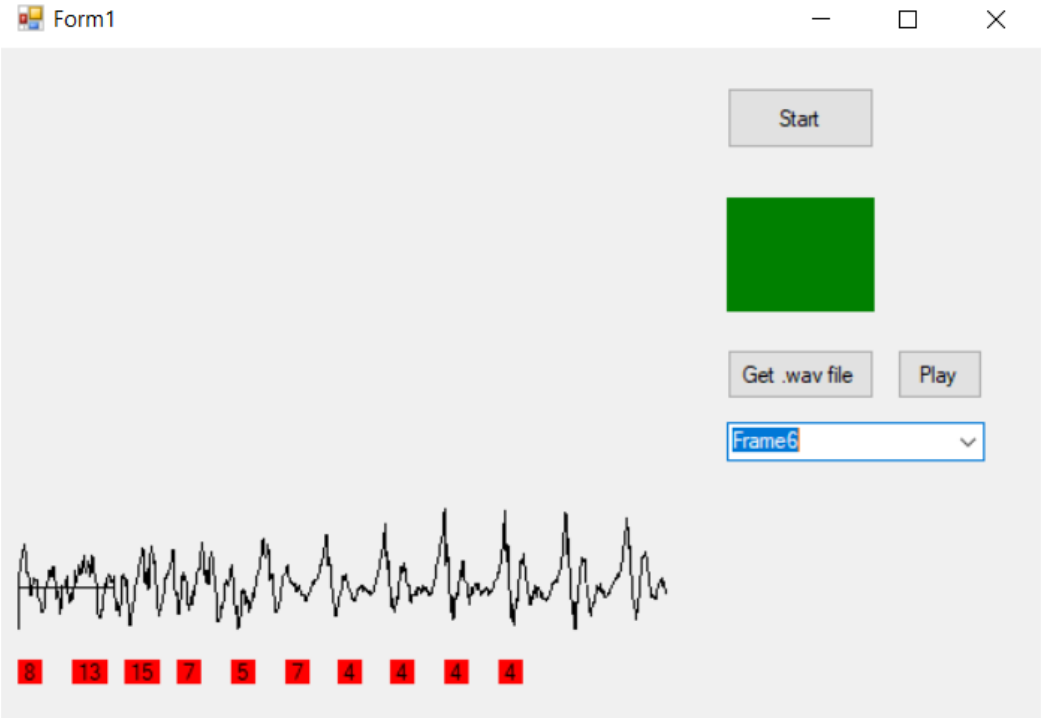

Рисунок 3.8. – Перегляд відображення обраного фрейму.

3.2 Недоліки системи

Серед основних недоліків додатку можна виділити наступні:

1. Аналіз проводиться лише для .wav файлі.

2. Недоліки алгоритму (висока чутливість до зниження SNR, невірні інтерпретації) повністю переносяться на додаток.

3. Додаток створювався для роботи в тихій кімнаті, відповідно, в інших середовищах результати роботи погіршуються.

3.3 Висновки до розділу

У даному розділі було продемонстровано сценарій взаємодії користувача з додатком та наведений порівняльний аналіз швидкості роботи алгоритму.

ВИСНОВКИ

Дана робота була присвячена актуальному питанню виявлення голосової активності в звуковому сигналі. При вирішенні поставлених задач були отримані наступні результати:

1. Аналіз проблематики виявлення голосової активності в звуковому сигналі показав зростання рівня використання алгоритмів виявлення голосу в повсякденному житті, що пов’язано зі зростаючою популярністю на ринку пристроїв та програмного забезпечення з голосовим інтерфейсом користувача (системи розумного будинку, побутові пристрої, голосові помічники, тощо). Однак також було виявлено відсутність універсальних чи близьких до універсальних алгоритмів.

2. При аналізі відкритих рішень були виявлені їх позитивні (висока ефективність роботи в умовах для яких створені, постійна модифікація) та негативні (надто вузька спеціалізація, подеколи мала швидкість роботи) сторони. Виявлено загальні для всіх алгоритмів проблеми деградації при зниженні SNR, та недостатньої швидкодії алгоритмів заснованих на нейронних мережах.

3. Для виявлення голосової активності у реальному часі було вирішено розробити та реалізувати алгоритм, що вирішував би проблему швидкодії, реалізовано швидкий метод виявлення голосової активності, який має потенціал використання з іншими більш повільними алгоритмами. На основі створеного алгоритму був написаний користувацький застосунок, що дозволяє більш детально ознайомитися з алгоритмом, та провести свої дослідження та експерименти.

Дана прикладна програма реалізована за допомогою мови C# та WindowsForms, що зручно для десктопного використання кінцевим користувачем.

Для роботи з додатком необхідно мати встановленими виконуваний файл застосування і .NET Framework 4.7.1, підключений мікрофон та/або звуковий файл у форматі .wav

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ramirez, J.; J. M. Gorriz; J. C. Segura Voice Activity Detection. Fundamentals and Speech Recognition System Robustness. In M. Grimm and K. Kroschel (ed.). Robust Speech Recognition and Understanding, 2007. — 22 с.

2. M. Y. Appiah, M. Sasikath, R. Makrickaite, M. Gusaite, Robust Voice Activity Detection and Noise Reduction Mechanism — Institute of Electronics Systems, Aalborg University, 2015. — 81 с.

3. Ravi Ramachandran; Richard Mammone (6 December 2012). Modern Methods of Speech Processing. Springer Science & Business Media. 102 p.

4. M. H. Savoji, «A robust algorithm for accurate end pointing of speech,» Speech Communication, pp. 45–60, 1989.

5. X.-L. Zhang and D. L. Wang, “Boosted deep neural networks andmulti-resolution cochleagram features for voice activity detection,”inINTERSPEECH, 2014, p.1538

6. Prabhat.K. Gupta, Shrirang Jangi, Allah B. Lamkin et. al. Voice activity detector for speech signals invariable background noise / Патент США №5649055

7. Network Working Group (2011-07-04). RTP Payload Format and File Storage Format for Opus Speech and Audio Codec. Opus codec.

8. Douglas J. Nelson, David C. Smith, Jeffrey L. Townsend. Voice activity detector / Патент США №6556967

9. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualstudio.com/.

10. Windows Forms [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/winforms/windows-forms-overview/.

11. Буч Градди. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд. / Буч Градди, Максимчук Роберт А., Энгл Майкл

12. X.L. Liu, Y. Liang, Y.H. Lou, H. Li, B.S. Shan. Noise-Robust Voice Activity Detector Based on Hidden Semi-Markov Models, Proc.ICPR`10, 88 p.

13. DMA minimum performance standarts for discontinuous transmission operation of mobile station TIA doc. and database IS-727, June 1998.

14. Paul Alexander Barret. Voice activity detector. Паттерн США №6061647.

15. Чистович Л.А., Венцов Л.А., Гранстрем М.П.. Физиология речи. Восприятие речи человеком. Ленинград, 1976.

16. Sergei N. Gramnitskiy. Method and system for distinguishing speech from music in digital audio signal in real time/ Патент США №7191128

17. Huang, Xuedong. Spoken language processing: a guide to theory, algorithm and system development/ Prentice Hall PTR, 2001.

18. Alex Acerd. Natural language processing (Computer science)/ Prentice Hall PTR, 2001.

19. R. Venkatesha Prasad, Abjeet Sangwan, Vishal Guarar. Comprarsion of Voice Activity Detection Algorithms for VoIP./ CEDT. Indian Institute of Science, 2002 6 p.

20. Гольдштейн Б.С. IP-телефония./Радио и связь. Москва, 2001, 335 с.

ДОДАТОК А

Система виявлення голосової активності в звуковому сигналі

Специфікація

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б	VAD.exe	Виконуваний файл програмної частини
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б	*.wav	Файл із записаною звуковою доріжкою

ДОДАТОК Б

Система виявлення голосової активності в звуковому сигналі

Текст програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 7

Київ 2019

Абстрактний клас детектора.

```
    public abstract class AbstractDetector : IDetector
    {
        WaveInEventArgs _fraim;
        public WaveInEventArgs fraim //100ms .wav sound
freim
        {
            get { return _fraim; }
            set
            {
                energyArraysList = new List<EnergyArray>();
                result = false;
                _fraim = value ?? throw new
ArgumentNullException();
            }
        }
        int _fraimSampleRate;
        public int fraimSampleRate
        {
            get { return _fraimSampleRate; }
            set
            {
                result = false;
                _fraimSampleRate = value;
            }
        }
        int _percentOfSucsess;
        public int percentOfSucsess
        {
            get { return _percentOfSucsess; }
            set
            {
                result = false;
                _percentOfSucsess = value;
            }
        }

        protected bool result;

        public AbstractDetector(WaveInEventArgs fraim, int
fraimSampleRate, int percentOfSucsess)
        {
            if (fraimSampleRate <= 0)
                throw new ArgumentOutOfRangeException();
```

```
            if (percentOfSucsess < 0 || percentOfSucsess >
100)
                throw new ArgumentOutOfRangeException();

            this.fraim = fraim ?? throw new
ArgumentNullException();
            this.fraimSampleRate = fraimSampleRate;
            this.percentOfSucsess = percentOfSucsess;
        }
        public AbstractDetector(WaveInEventArgs fraim, int
percentOfSucsess) : this(fraim, 8000, percentOfSucsess) { }
        public AbstractDetector(WaveInEventArgs fraim) :
this(fraim, 8000, 30) { }
        public AbstractDetector()
        {
            fraimSampleRate = 8000;
            percentOfSucsess = 40;
        }

        short FramesWithVoiceActivityRow;
        protected void AddResults(int count)
        {
            if (count >= percentOfSucsess / 10)
                result = true;
        }

        protected void AddResults(short count)
        {
            if (count >= FramesWithVoiceActivityRow)
                result = true;
        }

        virtual public bool VoiceIsPresent => result;

        public List<EnergyArray> energyArraysList = new
List<EnergyArray>();

    }
public struct EnergyArray
    {
        public List<short> _energyList;
        public bool _haveVoice;
    }
```

Клас реалізації алгоритму кількості переходів через нуль.

```
public class ShortTimeZeroCrossDetector : AbstractDetector
    {
        int barrier = 30;
        int framesWithVoise;

        public List<ZeroCrossCount> arr;

        byte[][] _buffer;

        public bool Decoding()
        {
            arr = new List<ZeroCrossCount>();

            framesWithVoise = 0;
            try
            {
                int counter = 0;
                /*
                 * Break 100ms frame to ten 10ms frames
                 * Multiply by 2 because information about 1
sempl writes in 2 byte
                 */
                _buffer = fraim.Buffer.GroupBy(_ =>
counter++ / (fraimSampleRate / 100 * 2)).Select(elem =>
elem.ToArray()).ToArray();

                foreach (var i in _buffer)
                    if (ZeroCrossDecoder(i, barrier))
                        framesWithVoise++;

                return true;
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine(e.StackTrace);
                return false;
            }
        }

        bool ZeroCrossDecoder(byte[] _buffer, int
crossBarrier)
        {
```

```
            bool _result = false;

            int sum = 0;

            short first = (short)((_buffer[1] << 8) |
_buffer[0]);

            for (int index = 2; index < _buffer.Length;
index += 2)
            {
                short Tmp = (short)((_buffer[index + 1] <<
8) | _buffer[index]);
                sum += Math.Abs(CountByZero(Tmp) -
CountByZero(first));
                first = Tmp;
            }
            if (sum/2 >= crossBarrier) _result = true;
            arr.Add(new ZeroCrossCount { count = sum/2,
result = _result });

            return _result;
        }

        private int CountByZero(short tmp)
        {
            if (tmp >= 1)
                return 1;
            else
                return -1;
        }

        public override bool VoiceIsPresent
        {
            get
            {
                this.Decoding();
                this.AddResults(framesWithVoise);
                return result;
            }
        }
    }

    public struct ZeroCrossCount
    {
        public int count;
```

```
        public bool result;
    }
```

Клас реалізації алгоритму короткочасних енергетичних характеристик сигналу.

```
public class ShortTimeEnergyDetector : AbstractDetector
    {
        double barrier = 0.07; //Get by empiric
        int framesWithVoise;

        byte[][] _buffer;

        public bool Decoding()
        {
            framesWithVoise = 0;
            try
            {
                int counter = 0;
                /*
                 * Break 100ms frame to ten 10ms frames
                 * Multiply by 2 because information about 1
sempl writes in 2 byte
                 */
                _buffer = fraim.Buffer.GroupBy(_ =>
counter++ / (fraimSampleRate / 100 * 2)).Select(elem =>
elem.ToArray()).ToArray();

                foreach (var i in _buffer)
                    if (Decoder(i, barrier))
                        framesWithVoise++;

                return true;
            }
            catch(Exception e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine(e.StackTrace);
                return false;
            }
        }

        bool Decoder(byte[] _buffer, double barrier)
        {
            double Sum2 = 0;
            bool Tr = false;
            EnergyArray energyArray = new EnergyArray();
```

```
            energyArray._energyList = new List<short>();

            for (int index = 0; index < _buffer.Length;
index += 2)
            {
                double Tmp = (short)((_buffer[index + 1] <<
8) | _buffer[index]);
                energyArray ._energyList.Add((short)Tmp);
                Tmp /= 32768.0;
                Sum2 += Tmp * Tmp;
                if (Tmp > barrier)
                    Tr = true;
            }
            Sum2 /= _buffer.Length / 2;

            if (Tr || Sum2 > barrier)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
```

ДОДАТОК В

Система виявлення голосової активності в звуковому сигналі

Опис програмного коду

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 8

Київ 2019

АНОТАЦІЯ

Даний додаток містить опис системи виявлення голосової активності в звуковому сигналі. Створений програмний продукт визначає голосову активність у вхідному акустичному сигналі, візуалізує результати та виконує такі завдання:

– виявлення голосової активності методом короткочасних енергетичних характеристик;

– виявлення голосової активності методом кількості переходів через нуль;

– візуалізація роботи методів;

– можливість завантаження для аналізу та відтворення завчасно записаного файлу у форматі .wav;

Даний застосунок було написано на платформі .NET Framework, використовуючи бібліотеку NAudio, мову C# та Windows Forms.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис системи виявлення голосової активності в звуковому сигналі. У додатку Б міститься програмний код головних частин розробленої системи.

Для роботи з розробленим додатком необхідно мати встановлений виконуваний файл застосунку, .NET Framework 4.7.1, мікрофон та/або записаний аудіофайл у форматі .wav.

При розробці програмного продукту використовувалась платформа .NET Framework, мова програмування C# та середовище розробки Visual Studio 2017.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблені компоненти виконують завдання виявлення голосової активності у вхідному аудіопотоці та візуалізації результатів. Це відбувається за рахунок об’єднання методів короткочасної енергії сигналу та підрахунку кількості переходів через нуль.

Розроблений API може використовуватися у програмному забезпеченні, що потребує виявлення голосової активності, як перший шар виявлення голосу.

Функціональні обмеження на використання додатку полягають лише у форматі даних записаних звукових файлів для аналізу та деградації працездатності в умовах низького SNR .

3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Для забезпечення повноцінної роботи та досягнення високої точності та зручності роботи системи виявлення голосової активності було обрано платформу .NET Framework, використання якої дозволяє максимально просто реалізувати необхідні алгоритми та легко організувати роботу із засобами запису звуку (мікрофоном), а також має досить швидку обробку вхідних запитів користувача.

Також для роботи зі звуком використовувалася додаткова а NAudio.

Розроблений додаток працює в операційній системі Windows, як десктопна програма. Для роботи з ним користувачу необхідно мати мікрофон та/або записаний аудіофайл у форматі .wav, який необхідно проаналізувати .

4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для реалізації хмарного сервісу швидкого прототипування було розроблено алгоритм заснований на поєднанні методів порівняння енергії кадру сигналу та енергії фрейму сигналу, що були для зручності з’єднані в метод короткочасних енергетичних характеристик сигналу, та метод підрахунку кількості переходів через нуль. Всі алгоритми будуються на аналізі вхідного сигналу на основі його енергетичних характеристик.

Загалом, всі обчислення дискретизації вхідного звукового сигналу, та подальшої його обробки алгоримами, з подальшою візуалізацією результатів.

Під час запуску сервісу перед користувачем з’являється вікно для вибору записаного аудіофайлу, або початку роботи зі звуковим поток, що фіксується мікрофоном. Після того, вхідний сигнал розбивається на фрейми в 10мс для режиму роботи в реальному часі, або 100мс для режиму роботи аналізу передчасно записаного аудіофайлу. При проведеному аналізі результати візуалізуються у вигляді графіку звукового потоку, побудованомуна основі енергії кадрів сигналу та текстових формах у яких фіксується кількість переходів через нуль фреймів.

5. ВИКЛИК І ЗАВАНТАЖЕНЯ

Для початку роботи необхідно виклики виконуваний файл додатку.

Після запуску користувач бачить вікно сервісу де і вибирає необхідний йому варіант роботи.

6. ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для розробленого додатку є фіксований мікрофоном аудіопотік, або аудіофайл у форматі .wav.

Вихідними даними є графічне відображення результатів аналізу вхідних даних.