МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження алгоритмів аналізу і обробки моделей представлення слухового сприйняття інформації

(тема)

Виконав:
Студент 2 курсу, групи ІПЗм-20-4
Джміль І.В.
(прізвище, ініціали)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник проф. Шостак І.В.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

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

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав.кафедри _____
(підпис)
« ____ » _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Джміль Ігоря Вячеславовича
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження алгоритмів аналізу і обробки моделей представлення слухового сприйняття інформації

затверджена наказом університету від 24.03.2022 р. № 412 Ст

2. Термін подання роботи до екзаменаційної комісії 10 05 2022р.
3. Вихідні дані до роботи *теорія лінгвістики, теорія автоматів, онтології, акустичні моделі сприйняття інформації*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, опис запропонованих варіантів оптимізації, використовувані методи та алгоритми, опис розробленої програмної системи, опис застосованих програмних рішень, аналіз можливих застосувань*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	31 березня 2022	*виконано*
2.	Огляд існуючих методів	10 квітня 2022.	*виконано*
3.	Розробка алгоритмів, проектування та розробка ПЗ	15 квітня 2022	*виконано*
4.	Підготовка пояснювальної записки	20 квітня 2022	*виконано*
5.	Спецчастина	28 квітня 2022	*виконано*
6.	Підготовка презентації та доповіді	03 травня 2022	*виконано*
7.	Попередній захист	05 травня 2022	*виконано*
8.	Нормоконтроль, рецензування	07 травня 2022.	*виконано*
9.	Занесення роботи в електронний архів	08 травня 2022.	*виконано*
10.	Допуск до захисту в зав. кафедри	11 травня 2022	*виконано*

Дата видачі завдання 28 березня 2022р.

Студент _____
(підпис)

Керівник роботи _____ проф. Шостак І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 95 с., 26 рис., 5 табл., 31 джерело.

ЗАКОН ТАЛБОТА, АСИНХРОННИЙ ЧАСТОТНО-ІМПУЛЬСНИЙ КОД ЗВУКУ, КРИТИЧНА ЧАСТОТА СИНХРОНІЗАЦІЇ

Об'єкт дослідження – частотно-імпульсне кодування звуку.

Методи дослідження – психоакустика, теорія дискретизації, методи дискретизації, синхронізації й кодування звуку,.

Мета роботи – розробка програмної системи, призначеної для вивчення та тестування методу частотно-імпульсного кодування звуку.

Система може бути використана для визначення параметрів частотно-імпульсного кодування звуку, при якому забезпечується якісний і ощадливий формат звукової інформації. Практичне застосування системи передбачається в цифрових пристроях запису й відтворення звуку, у системах вокодерного зв'язку та системах розпізнавання звукової мови.

TALBOT'S LAW, ASYNCHRONOUS FREQUENCY PULSE SOUND CODE, CRITICAL SYNCHRONIZATION FREQUENCY

The object of study – frequency-pulse coding of sound.

Research methods – psychoacoustics, sampling theory, sampling methods, synchronization and sound coding.

The purpose of the work is to develop a software system designed to study and test the method of frequency-pulse sound coding.

The system can be used to determine the parameters of frequency-pulse coding of sound, which provides a high-quality and economical format of audio information. Practical application of the system is provided in digital devices of recording and reproduction of sound, in systems of vocoder communication and systems of recognition of sound language.

Умови публікації пояснювальної записки

Я, _____Джміль Ігор Вячеславович_____
(прізвище, ім'я, по батькові)

студент групи ___ІПЗм-20-4_здобувач вищої освіти на другому (магістерському) рівні

кафедра програмної інженерії,
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему ___ Дослідження методів аналізу скінченних темпоральних автоматів ___,
(назва роботи)

що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У зв'язку з високими темпами розвитку, обчислювальна техніка стала найважливішим засобом автоматизації. Її застосування множить можливості людини, різко збільшує ефект від його діяльності. Області застосування ЕОМ швидко розширюються.

Однак бурхливий розвиток обчислювальної техніки саме породжує проблеми, що вимагають невідкладного рішення. Росте армія програмістів. Складання й налагодження програм, діалог з машиною – усе це трудомісткі й стомлюючі операції. Внаслідок цього спілкування людини й ЕОМ утруднене [1]. Щоб скористатися послугами обчислювальної техніки, людей сам повинен затратити масу праці. Через цей багато фахівців не застосовують обчислювальну техніку зовсім або використовують її в недостатньому ступені, і на цьому суспільство терпить величезні втрати.

Іншим немаловажним фактором, що впливають на недостатнє використання ЕОМ є наступний: тверда конкурентна боротьба на комп'ютерному ринку й ринку програмного забезпечення приводить до того, що його виробники прагнуть добитися максимальної якості звуку й зображення, а через недостатню вивченість властивостей і механізмів роботи відповідних людських органів це робиться далеко не найефективнішими й економічними методами, що приводить до значних і, найчастіше, невиправданим витратам пам'яті ЕОМ і інших її ресурсів, і, як наслідок – до значного збільшення вартості використовуваних апаратних і програмних продуктів і, отже, їхньої меншої доступності [2].

Важливим напрямком удосконалювання способів спілкування людини й ЕОМ є створення спеціальної апаратури й програм, що забезпечують ефективний обмін інформацією між ЕОМ і її користувачем за допомогою природної мови [3]. Мова – це найлегший і зручний спосіб спілкування, найбільше часто використовуваний людиною. Лист, набір тексту на комп'ютері – значно менш мобільні й оперативні способи спілкування. Мова є найшвидшим способом

зв'язку, вона ефективно діє на відстані, у темряві, вона легко сполучається з іншими діями людини. Але ефективне рішення проблеми автоматичного мовного спілкування людину й ЕОМ неможливо без вирішення проблем, пов'язаних з дискретизацією акустичних коливань, їх відділення від різних шумів і перешкод і їх кодування таким чином, щоб мати можливість розпізнавати окремі фонетичні одиниці.

Виходячи із усього сказаного вище, можна зробити однозначний висновок: вивчення властивостей слухового сприйняття людини є однієї з найважливіших завдань, рішення якої приведе до значного полегшення спілкування людину й ЕОМ і розширенню сфери застосування засобів обчислювальної техніки.

В області вивчення властивостей слухового сприйняття людини є множина завдань, що чекають свого рішення. Однієї з них є з'ясування принципів і математичний опис закономірностей, на основі яких людина сприймає безперервні коливання повітря й перетворює їх у дискретну форму [4].

Дана робота присвячена вивченню властивостей і механізмів сприйняття слуховим аналізатором людини дискретизованих акустичних коливань і розробки програмної системи кодування звуку, яка, опираючись на відповідні наукові дослідження, дозволяла б брати до уваги й кодувати той мінімальний обсяг інформації, який необхідний слуховому аналізатору людини для сприйняття коду звуку, як повноцінної картини вихідного звукового сигналу.

При введенні в ЕОМ, осцилограма звуку кодується за допомогою АЦ-перетворювачів. При цьому діаграма звукового тиску заміняється ґратчастою функцією. Чисельні значення окремих ліній ґратчастої функції запам'ятовуються обчислювальною машиною. Такий спосіб кодування звуку набув повсюдного застосування [5]. Тим часом, дослідження механізму слухового аналізатора людини показує, що природа використовує зовсім інший спосіб кодування звуку, а саме – частотно-імпульсний. Якщо за допомогою мікроелектронної техніки визначити, яка інформація передається по окремих волокнах слухового нерва, то виявиться, що по них випливають одиночні електрохімічні імпульси, причому тем рідше, чим нижче гучність сигналів і навпаки. Кожний імпульс виникає в момент

нагромадження деякої фіксованої порції площі під кривою звукового тиску, відлічуваної від моменту виникнення попереднього імпульсу. Дослідження показують, що частотно-імпульсний метод кодування звуку вигідно відрізняється від застосовуваного повсюдно в цей час АЦП – як у змісті підвищеної завадостійкості, так і в змісті ощадливості при зберіганні в пам'яті ЕОМ. Установлене, що частотно-імпульсне кодування звуку забезпечує, як мінімум, семиразову економію пам'яті при зберіганні коду звуку в порівнянні з АЦП-кодуванням. Значна економія досягається також за рахунок зменшення трудомісткості обробки звукових кодів [6].

Метою кваліфікаційної роботи є розробка моделей програмної системи кодування й відтворення звуку, а також методу формування частотно-імпульсного коду на базі узагальненого закону Талбота. Ця програмна система має бути призначена для тестування запропонованого методу частотно-імпульсного кодування звуку й визначення оптимальних параметрів цього кодування для аналізу слухового сприйняття інформації.

1 АНАЛІЗ СТАНУ РОЗВ'ЯЗАННЯ ПРОБЛЕМИ ТА ОБҐРУНТУВАННЯ ЦІЛЕЙ ДОСЛІДЖЕННЯ

1.1 Аналіз досліджень в області дискретизації акустичних коливань

У більшості сучасних систем розпізнавання мови найпершим перетворенням електричного сигналу, що надходить із виходу мікрофона, є його дискретизація. Загальноприйняте в цьому випадку використовувати аналого-цифровий перетворювач (АЦП) [1]. Звичайно формується 8-12 розрядний двійковий код ординати ґратчастої функції акустичного коливання, що відповідає її виміру приблизно із двома-трьома знаками точності. Інтервал між сусідніми крапками ухвалюється, як правило, у межах від 100 мкс до 1 мс. Акустичне відтворення подібного коду забезпечує більш-менш розбірливу мову. У ряді випадків використовують набагато більш точні коди. Так, наприклад, у [2] описане застосування 16нрозрядних двійкових кодів при частоті дискретизації 68 кГц. Синтезований по такому коду звук майже не відрізнимо на слух від вихідного звуку.

Широке практичне використання дискретизації мовних сигналів висуває для наукового пророоблення низка актуальних питань:

– чи можна строго обґрунтувати правомірність дискретизації звуку;

– чи є використання аналого-цифрового перетворювача стандартного типу найкращим способом дискретизації звукового сигналу;

– з якою точністю й частотою слід проводити дискретизацію й на чому засновувати вибір чисельних значень параметрів дискретизації?

Аналіз літературних даних показує, що ні на один із цих питань поки немає вичерпних відповідей.

Звичайно для обґрунтування правомірності дискретизації мовного сигналу посилаються на теорему дискретизації [3]. Ця теорема затверджує, що у випадку, коли спектр J(ω) функції $f(t)$, заданої на всій речовинній осі, тотожно дорівнює нулю при частотах $\omega \geq \omega_0$, то за точними значенням функції $f(t)$, знайденим у

крапках $t = 0, \pm 2\pi / \omega_0, \pm (2\pi / \omega_0) * 2, \ldots$, можна повністю відновити вид функції $f(t)$. Ґрунтуючись на теоремі дискретизації, часто роблять наступний висновок: оскільки вухо не чує звуків при частотах вище 20 кГц, те для одержання повноцінного коду звуку можна скористатися його цифровим кодом, отриманим при подвоєній частоті дискретизації, рівної 40 кГц [4].

Цей висновок представляється не цілком переконливим. По-перше, у теоремі дискретизації говориться про речовинні, тобто про ідеально точні значення функції $f(t)$. На запитання про те, чи можна замінити ці значення їх кінцевими цифровими кодами й скільки знаків треба обрати в цих кодах, теорема дискретизації взагалі не дає якої-небудь відповіді. Тим часом ясно, що ідеально точно виміряти значення функції $f(t)$ при її дискретизації практично неможливо. По-друге, з того факту, що вухо не чує звуків на частотах вище 20 кГц, зовсім не випливає, що у звуках, сприйманих слуховим аналізатором, немає частот вище 20 кГц. Напроти, точно відомо, що вони там є. Більше того, установлене, що частина із цих частот суттєво впливає на слухове сприйняття звуку. Так, наприклад, широко відомо, що виключення з акустичного коливання музичного добутку частот у діапазоні 20-40 кГц помітно збіднює звучання цього добутку.

Посилаються також на те, що людський слух, як і будь-який інший вимірювальний прилад, має кінцеву чутливість і кінцевою смугою пропущення частот, і тому для нього повинен існувати який те кінцевий поріг як за рівнем сигналу, так і за часом. Отже, при виборі досить великої частоти дискретизації й достатньо високої точності при вимірі значень акустичної діаграми буде отриманий код мовного повідомлення, що містить у собі всю інформацію, сприйману людським вухом. У принципі, це, звичайно, правильне твердження [6], однак воно носить занадто загальний характер і не може служити достатнім практичним керівництвом при виборі оптимального способу дискретизації мовного сигналу й установленні її раціональних параметрів. Вище говорилося, що використання 16нрозрядних двійкових кодів при частоті дискретизації 68 кГц забезпечує майже ідеальну натуральність відтворення закодованого звуку. Таке кодування вимагає щосекундної передачі більш мільйона біт інформації. Нам не

вдалося виявити літературних даних про той мінімальний щосекундний обсяг дискретної інформації, який би забезпечив ідеальне по слуховому сприйняттю відтворення коду звуку. Можна змінювати цей обсяг інформації без істотного збитку для якісного звучання і якщо так, те на скільки і якими методами. Впливає спосіб кодування звуку на мінімально припустимий обсяг переданої інформації. На ці й багато інші подібні питання наука поки не дає вичерпних відповідей [5]. Їхнє вирішення створило би наукову основу для істотного поліпшення технічних характеристик акустичної апаратури.

Очевидно, що відповідь на всі ці питання може бути отриманий лише на шляху психофізичного вивчення властивостей слухового сприйняття. Аналіз літературних даних показує, що до самого останнього часу розробки в слуху в цій області не були цілком успішними. У теорії зору, однак, удалося зробити набагато більше, чим у теорії слуху. В 1834 році Талбот [6] сформулював закон, що носить його ім'я, який згодом був застосований для розробки й обґрунтування практичних процедур дискретизації зорових сигналів. Згідно із законом Талбота будь-який сигнал з немінливою яскравістю B_0 можна замінити таким спеціальним (у тому числі й дискретним) періодично мінливим у часі (тобто мерехтливим) сигналом $B(t)$, що спостерігач не помітить яких-небудь тимчасових змін у своєму відчутті цього сигналу й сприйме його точно таким же, як сигнал B_0 .

Для візуальної рівності сигналів $B(t)$ і B_0 необхідно, щоб середня яскравість сигналу $B(t)$ збіглася з яскравістю B_0, тобто щоб мала місце рівність [7]:

$$B_0 = \frac{\int_0^T B(t)dt}{T}. \qquad (1.1)$$

Потрібно також, щоб період T сигналу $B(t)$ був досить малий. Аналогічний закон був сформульований також для зорових картин, розгорнутих у поле зору [8].

Наскільки швидкими повинні бути світлові мелькання, щоб їх не розділяло око людини – на це питання дають відповідь численні й ретельно виконані праці по визначенню критичної частоти мелькань [9]. Аналогічні дослідження були проведені й по визначенню критичного розміру дрібних елементів зорової картини для випадку просторового сприйняття [10]. Ці результати виявилися досить корисними для вибору й обґрунтування конкретних режимів дискретизації зорових картин. Теорія зору, заснована на законі Талбота, у цей час широке застосовується у всіх випадках, де потрібно часткова або повна дискретизація зорового сигналу (наприклад, у техніку кіно при виборі числа зміни кадрів у секунду, у техніку телебачення при призначенні густоти рядків у зображенні, у поліграфії при формуванні крапкового растра півтонового зображення).

У [11] було отримане узагальнення закону Талбота на випадок зорових картин, довільним образом мінливих у просторі й у часі. Тут нас буде цікавити узагальнення закону Талбота лише на випадок зорових картин, що довільно змінюються в часі [11]. Нехай довільно обраний зоровий стимул $B_0(t)$ і задана нескінченна послідовність зорових стимулів $\{BN(t)\}_{N\in(1,\infty)}$, що задовольняє умові:

$$\lim_{N\to\infty}\int_{t_1}^{t_2} B_N(t)dt = \int_{t_1}^{t_2} B(t)dt. \qquad (1.2)$$

де t_1 і t_2 – довільні фіксовані моменти часу.

Узагальнений закон Талбота говорить, що відчуття, викликуване зоровою картиною $BN(t)$, у межі при $N \to \infty$ буде неотличимо від зорового відчуття, викликуваного картиною $B(t)$. Умова (1.2) означає, що послідовність функцій $B_1(t)$, $B_2(t)$,... сходиться до функції $B(t)$ у середньому. Таким чином, згідно з узагальненим законом Талбота, зоровий аналізатор як би здійснює лінійне згладжування швидких коливань у діаграмі зорового сигналу.

В 1977 році в роботі [13] була вперше продемонстрована застосовність до слуху узагальненого закону Талбота у формі (1.2). Сигнали $B_0(t)$ і $BN(t)$ інтерпретуються як функції залежності звукового тиску від часу. В експериментах довільно обирані звуки $B_0(t)$ (шуми, дикторська мова, музичні добутки) періодично переривалися з різною шпаруватістю λ і з періодом T. Одночасно із цим для задоволення умови (1.2) переривчастий сигнал підсилювався в $1/\lambda$ раз. Сформований описаним способом переривчастий сигнал приймався в якості слухового стимулу $BN(t)$, де під N мається на увазі частота переривання сигналу $N = 1 / T$. Досвіди показали, що при $N \geq 30$ кГц будь-який звук $B_0(t)$ не відрізнимий від відповідного йому переривчастого звуку $BN(t)$.

1.2 Цифровий звук і частоти дискретизації

Звук являє собою безперервний у часі й по амплітуді процес, тобто тиск повітря змінюється в часі плавно, а не перестрибує від одного значення до іншого. Звук може бути перетворений в електричний сигнал за допомогою мікрофона, який залежно від зміни тиску повітря змінює створюване їм на виході електрична напруга. Після перекладу акустичного звуку в електричний сигнал безперервність у часі й по амплітуді зберігається: напруга сигналу змінюється аналогічно зміні тиску повітря, от чому отриманий на виході звук називають аналоговим. Ми можемо записати електричний сигнал на магнітну стрічку й перетворити його знову у звук за допомогою динаміка, який працює як «мікрофон навпаки» – переміщає повітря відповідно до змін напруги. Відповідно зберігається й згадана безперервність сигналу.

Незважаючи на те, що аналоговий електричний сигнал справно служить людству протягом десятиліть, згодом стало ясно, що аналогові сигнал і магнітний запис – не кращі способи передачі й зберігання звукової інформації, оскільки й при передачі, і при зберіганні даних відбувається неминуча втрата інформації,

тобто погіршення звуку. У той же час передачу й зберігання даних у комп'ютерах, що оперують винятково цифровими даними, можливо робити без яких-небудь втрат. Питання тільки в тому, як перевести аналоговий звук у цифровий і назад.

Для рішення першого завдання існують спеціальні обладнання, відомі як аналого-цифрові перетворювачі (АЦП). Ці обладнання здатні перетворити безперервний аналоговий сигнал у послідовність окремих чисел, тобто зробити його дискретним. Перетворення відбувається в такий спосіб: обладнання багато раз у секунду вимірює амплітуду аналогового сигналу й видає результати вимірів у вигляді чисел. Результат дискретизації не є точним аналогом безперервного електричного сигналу. Наскільки всеж-таки відповідає цифровий звук аналоговому? Очевидно, що ця відповідність буде тем повніше, чим частіше відбуваються виміру й чому вони точніше. Частота, з якої проводяться виміри, називається частотою дискретизації. А на точність вимірів амплітуди вказує число біт вимірів, що використовуються для вистави результату. Цей параметр називають розрядністю.

Будь-яка звукова хвиля, яка поширюється в просторі, виявляє на перешкоди, що зустрічаються, у тому числі й на наші барабанні перетинки, якийсь тиск. Ми суб'єктивно сприймаємо зміну тиску звукових хвиль як зміна гучності звуку. Максимальна зміна тиску в повітрі при поширенні звукових хвиль у порівнянні з тиском при відсутності хвиль називається звуковим тиском. Як і будь-який інший, звуковий тиск вимірюється в паскалях (Па). Але при оцінці інтенсивності звукових хвиль частіше застосовується інше поняття – сила звуку. Вона визначає потік звукової енергії, який щосекунди проходить через квадратний сантиметр умовної площини, розташованої перпендикулярно напрямку поширення хвилі. Звуковий тиск і сила звуку перебувають у квадратичній залежності (сила звуку прямо пропорційна звуковому тиску у квадраті). Сила звуку описує енергетичні властивості звукової хвилі й виміряється у ватах на квадратний сантиметр (Вт/см^2). Така одиниця буває дуже зручна при розрахунках – це єдина причина її введення.

Для того щоб ми змогли почути той або інший звук, його сила повинна бути більше певного рівня. Цей рівень називається порогом чутності: якщо звукова хвиля має малу інтенсивність – нижче цього порога, то ми просто не сприймаємо її й нам здається, що навколо коштує повна тиша, хоча насправді повітря навколо коливається. Точно також інша справа й зі звуками великої інтенсивності: ми чуємо звук тільки до певного рівня, який називається болючим порогом. Якщо сила звуку більше цього рівня, то ми випробовуємо біль у вухах. Різниця між рівнями болючого порога й порога чутності називається динамічним діапазоном слуху. Наш слуховий апарат улаштований таким чином, що лінійна зміна сили звуку (або звукового тиску) не сприймається нами як лінійна зміна гучності. Гучність звуку і його сила зв'язані між собою більш складною залежністю. Збільшення гучності в 2 рази відповідає збільшенню сили звуку в 100 раз (звукового тиску – в 10 раз), збільшення гучності в 4 рази відповідає збільшенню сили звуку вже в 10000 раз (звукового тиску – в 100 раз), а збільшення гучності в 8 раз відповідає зміні сили звуку в 100_000_000 раз (звукового тиску – в 10000 раз)

Перетворення аналогового сигналу в цифровий складається із двох етапів: дискретизації за часом і квантування по амплітуді. Дискретизація за часом означає, що сигнал представляється поруч відліків (семплів), узятих через рівні проміжки часу. Наприклад, коли ми говоримо, що частота дискретизації 44,1 кГц, те це значить, що сигнал виміряється 44 100 разів протягом однієї секунди. Основне питання на першому етапі перетворення аналогового сигналу в цифровий (оцифрування) полягає у виборі частоти дискретизації аналогового сигналу. Чим більше частота, тем точніше відповідає цифровий сигнал аналоговому. Однак пропорційно збільшенню частоти зростають:

– інтенсивність потоку цифрових даних, а пропускні можливості інтерфейсів не безмежні, особливо якщо записується/відтворюється одночасно кілька каналів;

– обчислювальне навантаження на цифрові процесори, а їх обчислювальні можливості також обмежені;

– обсяг пам'яті, необхідної для зберігання цифрового сигналу.

Очевидно, що необхідний компроміс. Від вибору частоти дискретизації залежить частотний діапазон отриманого цифрового звуку й максимальна частота аналогового сигналу, правильно представлена в цифровому. Уважається, що людина чує частоти в діапазоні від 20 до 20 000 Гц. Згідно з теоремою Найквиста, для того, щоб аналоговий (безперервний за часом) сигнал можна було точно відновити по його відлікам, частота дискретизації повинна бути як мінімум удвічі більше максимальної звукової частоти. Звукова частота, рівна половині частоти дискретизації, називається частотою Найквиста і є максимальною частотою, яку дана цифрова система може правильно зберегти й відтворити. Таким чином, якщо реальний аналоговий сигнал, який ми збираємося перетворити в цифрову форму, містить частотні компоненти від 0 до 20 кГц, те частота дискретизації такого сигналу повинна бути не менш 40 кГц. Сьогодні найпоширенішими частотами дискретизації є 44,1 кГц (CD) і 48 кГц (DAT).

Така залежність називається логарифмічної, і саме через таку особливість нашого сприйняття зміна рівня (гучності) звуку прийнято вимірювати в логарифмічних одиницях – Белах (Б). Відмінність величин сили звуку в белах обчислюється по формулі: N=lg 11/12 (lg – десятковий логарифм), N – зміна рівня звуку, а 11 і 12 – верхня й нижня границі сили звуку. Десятикратне збільшення сили звуку відповідає 1 Белу (lgl0=l), а сторазове збільшення відповідає двом белам (lgl00=2) і т.д.

Логарифмічна шкала повністю відповідає особливостям нашого слуху. На практиці виявляється, що білий – це занадто більша величина для зміни рівня. Тому частіше застосовується децибел (дБ) – десята частина біла. Тоді зміна рівня в децибелах буде обчислюватися по формулі: $N=10 \lg I_1/I_2$. Мінімальний перепад рівня, який здатне сприйняти наше вухо, саме рівний одному децибелу. Це одна з головних причин уведення такої системи виміру рівня. А весь динамічний діапазон слуху становить 120 дБ. Зміна рівня звуку звичайно оцінюється в децибелах щодо порога чутності. Коли говорять, що рівень звуку в колонках

рівний ста децибелам, мають на увазі, що стовпчика працюють на рівні перевищуючому поріг чутності на 100 дБ.

Цифрова вистава звуку коштовна насамперед можливістю нескінченного зберігання й тиражування без втрати якості, однак перетворення з аналогової форми в цифрову й назад все-таки неминуче приводить до часткової його втрати. Найбільш неприємні для слуху викривлення, внесені на етапі оцифрування – гранулярний шум, що виникає при квантуванні сигналу за рівнем через округлення амплітуди до найближчого дискретного значення.

1.3 Огляд методів відтворення звуку

Гранулярний шум сильно корелюваний із сигналом (залежить від нього) і являє собою гармоніки сигналу, викривлення від яких найбільш помітні у верхній частині спектра. Прояву гранулярного шуму і його зв'язок із сигналом легко визначити, прослухавши синусоїдальний сигнал із частотою близько 0,1 – 5 Гц (гранулярний шум у цьому випадку проявляється у вигляді мінливого по висоті паразитного тону, частота якого залежить від частоти, форми й максимальної амплітуди корисного сигналу). Потужність гранулярного шуму обернено пропорційна кількості щаблів квантування, однак через логарифмічну характеристику слуху при лінійнім квантуванні (постійна величина щабля) на тихі звуки доводиться менше щаблів квантування, чому на голосні, і в результаті основна щільність нелінійних викривлень припадає на область тихих звуків. Під час відновленні звуку із цифрової форми в аналогову виникає проблема згладжування східчастої форми сигналу й придушення гармонік, внесених частотою дискретизації. Через неідеальність амплітудно-частотної характеристики може відбуватися або недостатнє придушення цих перешкод, або надлишкове ослаблення корисних високочастотних складових. Погано подавлені

гармоніки спотворюють форму аналогового сигналу (особливо в області високих частот), що створює враження «шорсткуватого», «брудного» звуку.

Крім запису й відтворення звуків із зовнішніх джерел можливий чисто комп'ютерний синтез звуку. Зараз таких методів два: WT і FM. WT (Wavetable – таблиця хвиль) – відтворення заздалегідь записаних у цифровому виді звучань – семплів. Інструменти з малою тривалістю звучання звичайно записуються повністю, а для інших може записуватися лише початок/кінець звуку й невелика «середня» частина, яка потім програється в циклі протягом потрібного часу. Для зміни висоти звуку оцифровка програється з різною швидкістю, а щоб при цьому сильно не змінювався характер звучання, семпли інструментів складаються з декількох фрагментів для різних діапазонів нот. У складних синтезаторах використовується паралельне програвання декількох семплів на одну ноту й додаткова обробка звуку(модуляція, фільтрування, різні «оживляючі» ефекти й т.п.). Більшість плат містить вбудований набір інструментів у ПЗУ, деякі плати дозволяють додатково завантажувати власні інструменти в ОЗУ плати. Деякі моделі Pci-плат дозволяють використовувати для завантаження інструментів загальне ОЗУ комп'ютера. Переваги цього методу – гранична реалістичність звучання класичних інструментів і простота одержання звуку. Недоліки – наявність твердого набору заздалегідь підготовлених тембрів, багато параметрів яких не можна змінювати в реальному часі, більші обсяги пам'яті для семплів (іноді до декількох мегабайт на інструмент), відмінності у звучаннях синтезаторів. FM (частотна модуляція) – синтез за допомогою декількох генераторів сигналу (звичайно синусоїдального) із взаємною модуляцією. Кожний генератор забезпечується схемою керування частотою й амплітудою сигналу й утворює «оператор» – базову одиницю синтезу. Найчастіше у звукових картах застосовується 2-операторний (OPL2) синтез і іноді – 4-операторний (OPL3). Незважаючи на те, що більшість карт підтримує режим OPL3, стандартне програмне забезпечення для сумісності використовує їх у режимі OPL2. Схема з'єднання операторів (алгоритм) і параметри кожного оператора (частота, амплітуда й закон їх зміни в часі) визначають тембр звучання; кількість

операторів і ступінь тонкощі керування ними визначають гранична кількість синтезованих тембрів. Гідності цього методу – відсутність заздалегідь записаних звуків і пам'яті для них, велика різноманітність одержуваних звучань, повторюваність тембрів на різних картах із сумісними синтезаторами. Недоліки – дуже мала кількість «благозвучних» тембрів у всім можливому діапазоні звучань, відсутність якого-небудь алгоритму для їхнього пошуку, украй груба імітація звучання реальних інструментів, складність реалізації тонкого керування операторами, через що у звукових картах використовується сильно спрощена схема зі значно меншим діапазоном можливих звучань. При використанні звуків реальних інструментів для синтезу найкраще підходить метод WT; для створення ж нових тембрів більш зручний FM, хоча можливості Fm-синтезаторів звукових карт сильно обмежені через свою простоту.

1.4 Обґрунтування цілей дослідження

Завданням даного проекту є вивчення й математичний опис закономірності перетворення безперервного слухового стимулу (акустичної діаграми) у дискретний образ. Завдання полягає в тому, щоб з'ясувати, при яких параметрах перетворення аналогового сигналу в цифровий код досягається ідентичність слухового сприйняття вихідного звуку і його дискретного образа. Відправним пунктом цього дослідження з'явилося використання ефекту згладжування в слуху. Воно привело до висновку про те, що природньою формою вистави дискретного слухового образа звуку може служити частотно-імпульсний код акустичної діаграми. До розробки висуваються наступні два питання:

– при яких умовах забезпечується ідентичність відчуттів натурального слухового стимулу і його асинхронного частотно-імпульсного коду;

– як позначається на слуховій рівності звуку синхронізація частотно-імпульсного коду.

Щоб розв'язати поставлені питання, довелося розробити теоретичні основи побудови асинхронних і синхронізованих частотно-імпульсних кодів, що збігаються по слуховому відчуттю з вихідним звуком; розробити програмну систему для перетворення діаграми звуку в коди необхідного виду, що допускає регулювання в широких межах режимів перетворення; виконати спеціальні досвіди по встановленню рівності або нерівності слухових відчуттів, порушуваних натуральними звуками і їх дискретними еквівалентами [14, 15].

Програмна система складається із двох модулів. Перший з них створює wav-файли, другий (основний) модуль підсумує їх у різних режимах і відтворює з метою перевірки акустичної ідентичності й відсутності викривлень.

Вхідними даними першого модуля є звукові файли у форматі .wav і побайтная десятковий запис звукових кодів, а даними основної програми є параметри підсумовування й відтворення звукових діаграм. Ці дані йдуть у блок редагування вихідних звукових діаграм, потім у блок підсумовування звукових діаграм.

Вихідними даними розробленої системи є озвучені вихідні й сумарні звукові діаграми, що дозволяє експериментально обґрунтувати параметричні характеристики запропонованого методу частотно-імпульсного кодування звуку.

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1 Аналіз алгоритмів аналізу частотної інформації

Одними з найбільш точних приладів для виміру НВЧ трактів є векторні аналізатори ланцюгів (ВАЛ) [8]. Ланцюги, які можуть бути виміряні за допомогою ВАЛ, мають широкий діапазон застосування, починаючи простим обладнанням, таким як фільтри й підсилювачі, і закінчуючи складними модулями, що використовуються в системах телекомунікації. ВАЛ представляє собою одну з найбільш складних частин багатоцільового іспитового обладнання в області високочастотної й надвисокочастотної радіотехніки. Він призначений для вимірів комплексних коефіцієнтів передачі й відображення (елементів матриці розсіювання) багатополюсників [9]. На рисунку 2.1 наведена типова структурна схема ВАЛ, що має один вимірювальний порт [9]. Однією з основних функцій ВАЛ є вимірювання комплексного коефіцієнту відбиття.

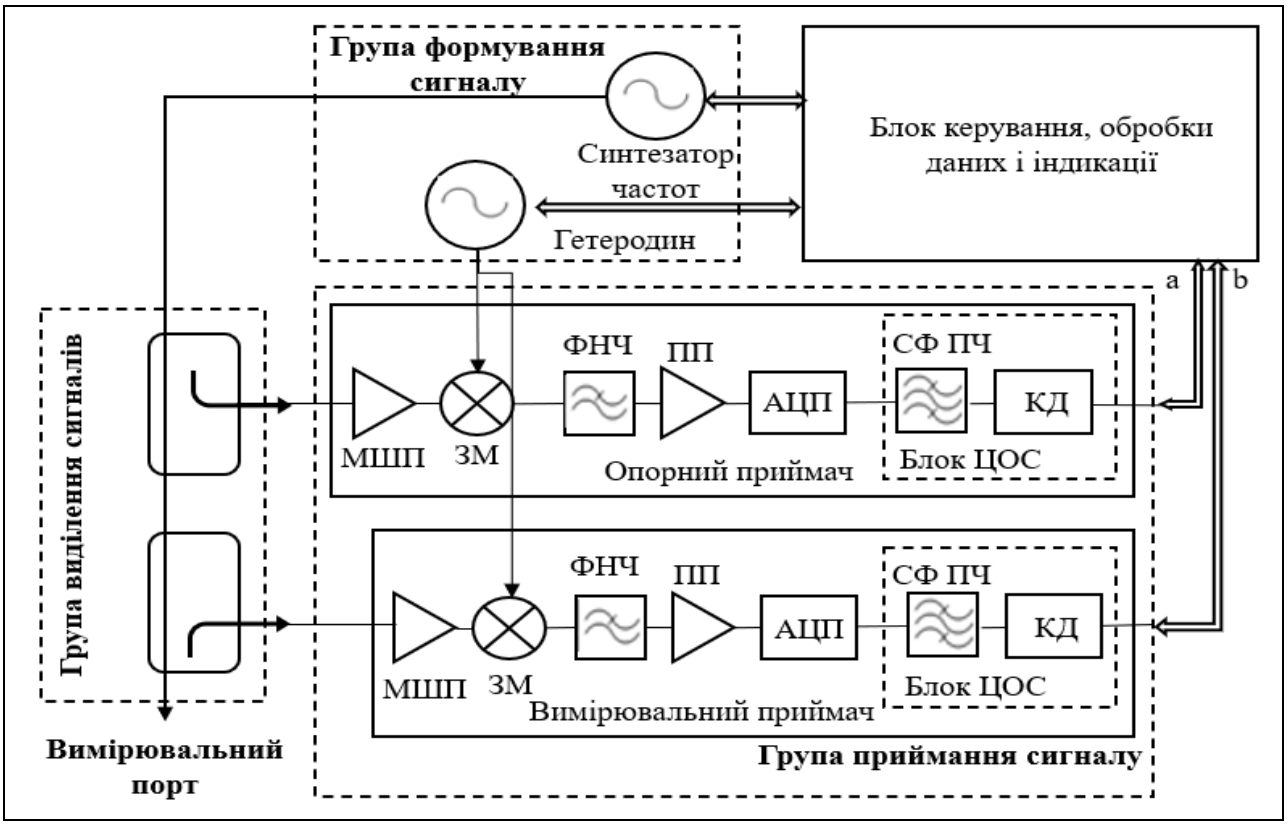

Рисунок 2.1 – Структурна схема рефлектометра [9]

У групу формування сигналу входять два джерела: синтезатор частот і гетеродин. Джерела синхронізовані по частоті від одного опорного генератора. Синтезатор частот призначений для формування зондуючого сигналу в діапазоні робочих частот ВАЛ. Сигнал гетеродину, зміщений по частоті щодо зондувального на величину проміжної частоти, необхідний для перетворення (зниження) частоти.

Група виділення сигналів призначена для одержання падаючої й відбитої від досліджуваного обладнання хвиль. Для виділення хвиль використовують спрямовані відгалужувачі або мости. Для виділення падаючої хвилі також застосовують дільники потужності.

Група приймання сигналу складається із двох ідентичних приймачів: опорного й вимірювального. Для радіотехнічних вимірів, як правило, приймачі будують за супергетеродинною схемою. Вони здійснюють перетворення сигналів на більш низьку проміжну частоту, посилення й фільтрацію. До складу приймачів входять наступні основні елементи: малошумливий підсилювач, змішувач, фільтр нижніх частот, аналого-цифровий перетворювач і блок цифрової обробки сигналів:

– малошумливий підсилювач призначений для приймання й посилення сигналів з низьким рівнем викривлень;

– змішувач служить для перетворення на проміжну частоту;

– фільтр нижніх частот призначений для фільтрації (виділення) сигналів проміжної частоти на виході змішувача;

– підсилювач потужності призначений для основного посилення сигналів у приймачі;

– аналогово-цифровий перетворювач служить для перетворення сигналу в цифровий вигляд;

– блок цифрової обробки сигналів, основними елементами якого є смуговий фільтр проміжної частоти й квадратурний демодулятор, призначений для фінальної фільтрації й одержання на виході приймачів вимірювальних

сигналів «a» і «b» у комплексному виді. Сигнали «a» і «b» надалі використовуються для рішення вимірювального завдання – вимірювання комплексного коефіцієнта відбиття.

До вимірювального порту підключаються однопортові досліджувані обладнання. До однопортових обладнань (двополюсників) відносяться узгоджені й неузгоджені навантаження, короткозамкнені навантаження й навантаження холостого ходу (у тому числі заходу хвильового й повного опорів). Крім цього, до даного класу обладнань можна віднести багатопортове обладнання. При цьому один з портів такого обладнання повинен підключатися до ВАЛ для тестування, а ті, що залишилися – навантажуватися певним чином" [10].

"Блок керування, обробки даних і індикації відповідає за синхронну роботу всіх блоків, вирішує вимірювальне завдання й відображає результат в обраних користувачем одиницях виміру на екрані ВАЛ. Частина функцій блоку може бути реалізована на зовнішньому персональному комп'ютері" [11].

Для двопортового обладнання (чотириполюсників) характеристика віддзеркалення (відбиття) від першого порту називається коефіцієнтом відбиття S_{11}, характеристика передачі в прямому напрямку називається коефіцієнтом передачі S_{21}, характеристика передачі у зворотному напрямку називається коефіцієнтом передачі S_{12}, і характеристика відбиття від другого порту називається коефіцієнтом відбиття S_{22}.

Вимірювані параметри представлені на рис. 2.2.

Кожний S-параметр містить амплітудно-частотну й фазо-частотну характеристики обладнання, що тестується у відповідному напрямку. Існує багато стандартних способів відображення обмірюваних S-параметрів на екрані ВАЛ. Можна вибирати, у якому вигляді переглядати результати: у вигляді графіку КСХ або зворотних втрат від частоти, діаграми Сміта, амплітуди, фази, внесеного загасання або посилення, групової затримки й ін.

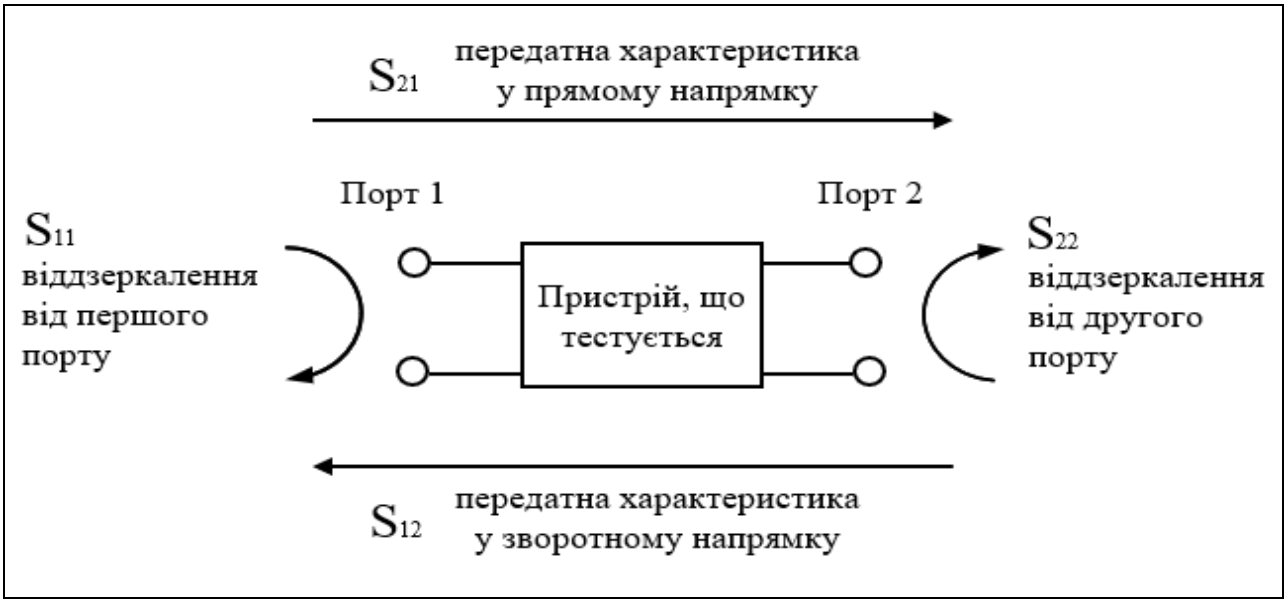

Рисунок 2.2 – Графічне представлення вимірюваних параметрів [11]

Як приклад, на рисунку 2.3 показаний екран векторного аналізатора Anritsu VNA Master серії MS20xxb з результатами виміру характеристик смугового фільтра.

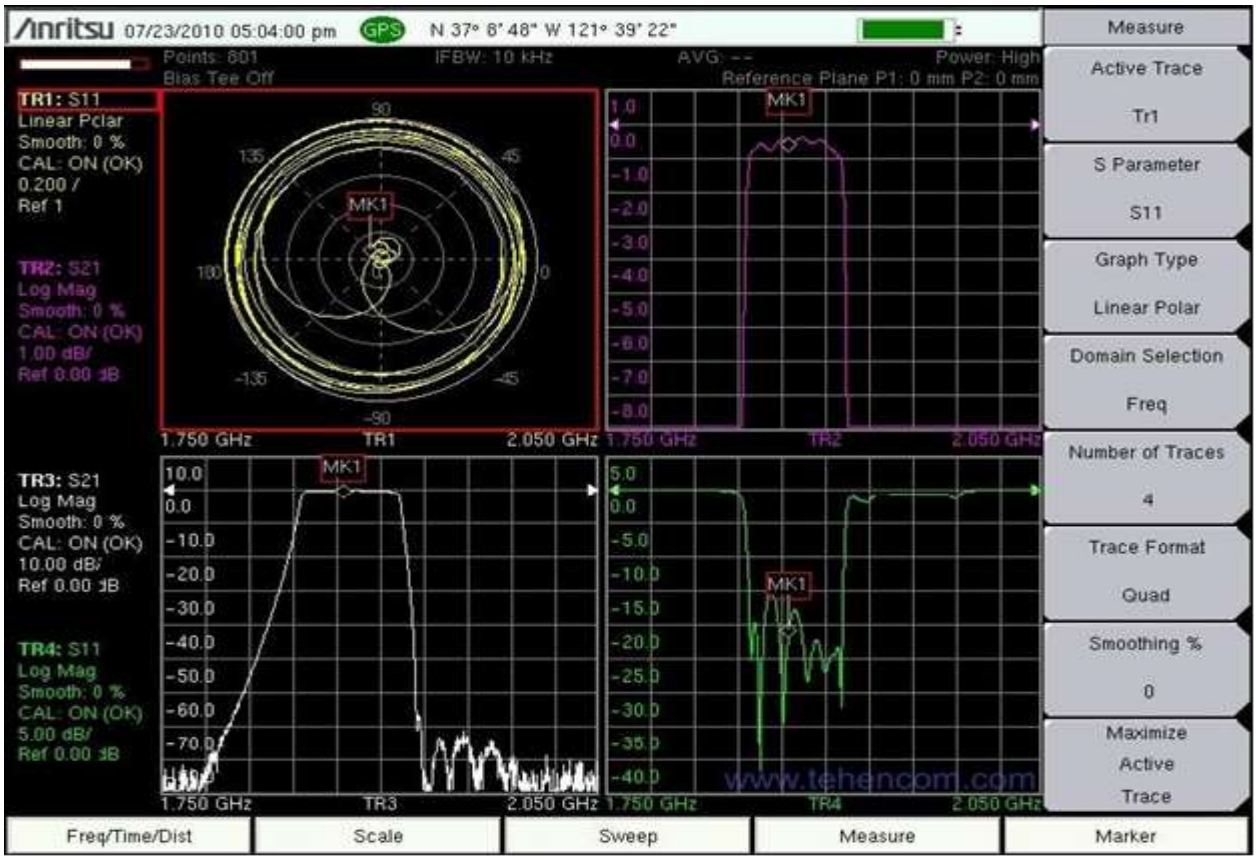

Рисунок 2.3 – Зображення екрана векторного аналізатора ланцюгів [12]

Основні параметри фільтра (S_{11} і S_{21}) представлені на чотирьох детальних графіках.

Також використовується пристрій DIY Vector Network Analyzer UVNA-63 [10]. Дане обладнання є комплектом для самостійного складання. Воно складається з набору кабелів, калібрувального набору, плати для обробки отриманих даних з портів обладнання, різних перехідників. Модулі даного обладнання закріплюються на металевій підкладці. Блок обробки даних має шість портів і живиться від USB порту комп'ютера. По цьому ж інтерфейсу відбувається приймання даних і керування блоком. Вид зібраного обладнання представлено на рис. 2.4.

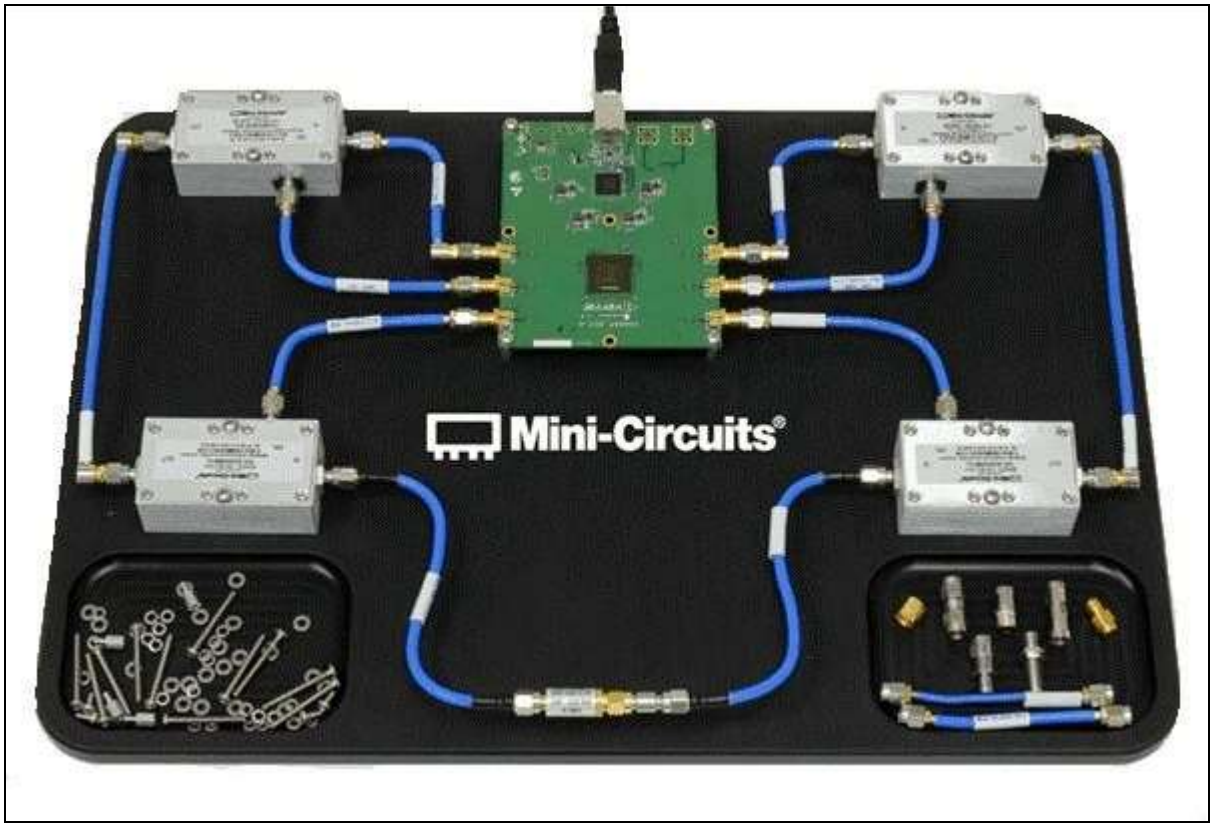

Рисунок2.4 – Зібраний модульний векторний аналізатор ланцюгів [14]

Даний апарат має два порти для виміру, відповідно на ньому можна вимірювати як комплексний коефіцієнт відбиття, так і коефіцієнт передачі. Структурна схема обладнання показано на рисунку 2.5.

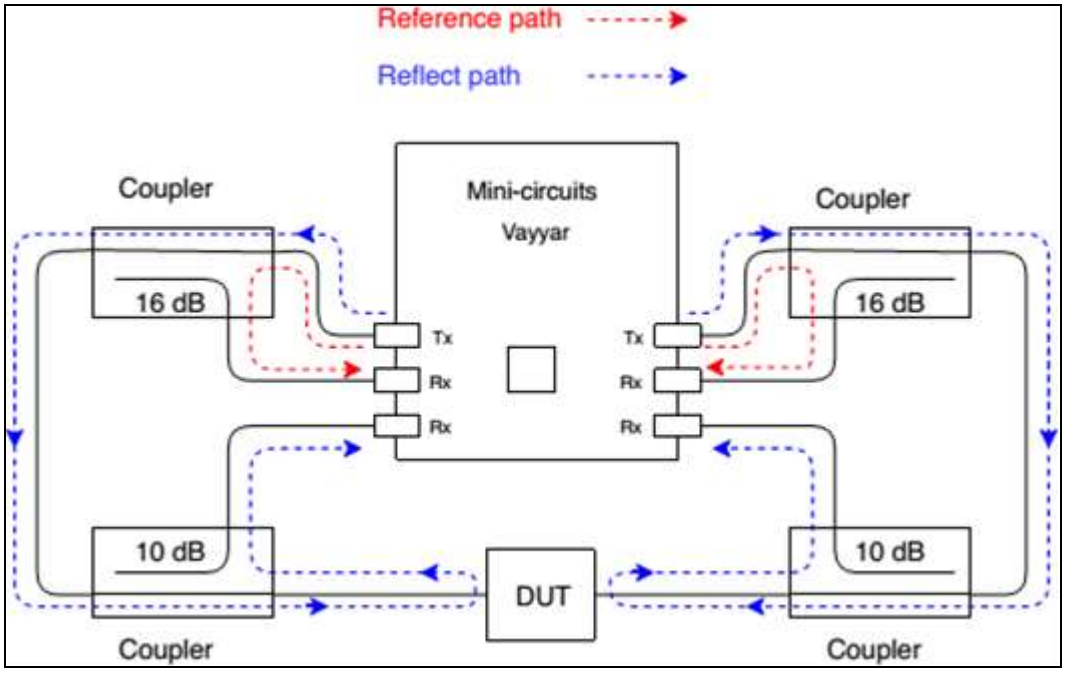

Рисунок 2.5 – Вимірювальна установка з відображенням опорного (червоний) і відбитого (синій) шляхів [15]

При вимірі обладнань на вихід основної плати надходить інформація у вигляді матриці значень.

2.2 Опис методів частотно-імпульсного кодування звуку

Розглянуто деякі властивості слуху, на основі яких розроблений спосіб досить ощадливої дискретної вистави звуків мови. У якості відправного пункту дослідження використовується узагальнений закон Талбота.

Узагальнений закон Талбота встановлює достатню умову слухової рівності довільних звуків. Випробуваний порівнює свої відчуття звуку $A(t)$ і одного зі звуків нескінченного ряду $B_1(t)$, $B_2(t)$, ... Закон говорить, що при виконанні умови (1.2) у міру просування вправо уздовж ряду $\{B_i\}$, $i \in (1,\infty)$ відчуття звуку $B_i(t)$ буде наближатися до відчуття звуку $A(t)$, і в межі при $i \to \infty$ порівнювані звуки будуть сприйматися на слух як зовсім ідентичні.

У психоакустиці протягом багатьох десятиліть уживали наполегливі, але безуспішні спроби відшукати слуховий аналог закону Талбота [12]. При цьому з'ясувалося, що якщо за аналогією із зором під сигналом $B(t)$ розуміти мінливий у часі енергетичний рівень звукового сигналу, то в застосуванні до слуху закон Талбота виявляється невірним. Якщо ж під сигналом $B(t)$ розуміти діаграму звукового тиску, то закон Талбота у формі (1.1), якщо він застосований до слуху, хоча й підтверджується на практиці, однак являє собою зовсім беззмістовне твердження. Справа в тому, що діаграма звуку B_0 з постійним рівнем звукового тиску не сприймається вухом і вона не є чутною. Нечутним виявляється також і будь-який періодичний звуковий сигнал $B(t)$, що задовольняє умові (1.1), якщо зробити досить малим його період (при $T \leq 50$ мкс).

Важливо відзначити, що у формулюванні узагальненого закону Талбота є істотний недолік. Справа в тому, що в ній ніяк не бере участь поняття критичної частоти слуху. Це – закон граничний, ідеалізований. Тому узагальнений закон Талбота не дає відповіді на запитання про те, за яким параметром оцінювати ступінь близькості звуку $BN(t)$ до звуку $B_0(t)$ і при якому значенні цього параметра відчуття звуків $BN(t)$ і $B_0(t)$ збіжаться. Представляється цікавим, як у теоретичному плані, так і із практичної точки зору, спробувати так підсилити формулювання узагальненого закону Талбота, щоб у ній безпосередньо з'явився слуховий поріг. Це дозволило б підійти до розробки наукових основ розрахунку раціональних параметрів дискретизації акустичного сигналу й вибору методу дискретизації. Це також дозволило б підійти до вивчення експериментальними методами конкретних значень величин порогів у слуху при різних умовах слухового сприйняття.

Будь-який звук будемо формально характеризувати його акустичною діаграмою, тобто залежністю тиску повітря (звукового тиску) від часу, заданої на інтервалі $[0,T_0]$. Потрібно розглядати звуки, діаграми яких плавно (безупинно) змінюються в часі. Такі звуки будемо позначати символом $A(t)$. До подібних звуків належить більшість звуків природного походження, у тому числі й звуки мови. Крім безперервних звуків будемо використовувати деякі звуки, що штучно

сформовані і діаграми яких мають розриви (перегони) і нескінченні викиди (імпульси).

$$\int_{t_1}^{t_2} B_i(t)\,dt < \infty, \ (i = 1, 2, \ldots). \qquad (2.1)$$

Такі звуки зазвичай позначають символами $B_1(t)$, $B_2(t)$, ... До діаграм цього виду висуватимуть єдину вимогу: вони повинні бути локально суммируемыми. Це означає, що при будь-яких межах t_1 і t_2, що належать інтервалу часу $[0,T_0]$, інтеграл від акустичної діаграми $Bi(t)$ повинен бути кінцевим. Ясно, що вимога (2.1) для будь-яких звуків, які можна практично сформувати, виконується.

Нижче приводиться трохи видозмінена (несуттєво) формулювання узагальненого закону Талбота, яка більш зручна для наших цілей. Нехай $A(t)$ - діаграма деякого звуку й $\{B_\omega(t)\}\omega\in(0,\infty)$ – нескінченна множина акустичних діаграм, де ω – дійсне позитивне число. Тоді узагальнений закон Талбота запишеться в наступній формі.

Якщо при будь-яких t_1 і t_2

$$\lim_{\omega\to\infty} \int_{t_1}^{t_2} B_\omega(t)dt = \int_{t_1}^{t_2} A(t)dt, \qquad (2.2)$$

то в межі при необмеженім збільшенні числа ω відчуття звуку $B_\omega(t)$ збіжиться з відчуттям звуку $A(t)$. Тільки що записане формулювання узагальненого закону Талбота відрізняється від вихідного лише тим, що в ній натуральне число N замінене позитивним речовинним числом ω.

Експериментальна перевірка узагальненого закону Талбота показує [13], що при виконанні умови (2.2) відчуття звуку $A(t)$ дійсно наближається до відчуття звуку $B_\omega(t)$, якщо необмежено збільшувати число ω. Виявляється навіть щось більше: відчуття звуків $A(t)$ і $B\omega(t)$ стають ідентичними не тільки в межі, але вже

при підході до нього. Це означає, що, збільшуючи в експерименті число ω, ми рано або пізно приходимо до такого його значення ω_0, коли при кожному $\omega > \omega_0$ буде спостерігатися слухова рівність звуків $A(t)$ і $B_\omega(t)$. Таким чином, факти показують, що є можливість трохи підсилити узагальнений закон Талбота, перетворивши його із граничного твердження про звук $A(t)$ і нескінченної послідовності звуків $\{B_\omega(t)\}$, $\omega \in (0,\infty)$ у висловлення про слухової рівності двох звуків $A(t)$ і $B_\omega(t)$, де $\omega > \omega_0$.

Викладені моделі приводять до наступного формулювання посиленого узагальненого закону Талбота.

Якщо звук $A(t)$ і множина звуків $\{B_\omega(t)\}$ $\omega \in (0,\infty)$ задовольняють умові (2.2), то найдеться таке число $\omega_0 > 0$, що для кожного $\omega > \omega_0$ відчуття звуку $B_\infty(t)$ збіжиться з відчуттям звуку $A(t)$.

Припустивши, що посилений узагальнений закон Талбота невірний, тоді, яке б велике число ω_0 було б не обране, завжди знайдеться таке число $\omega > \omega_0$, для якого звуки $B_\omega(t)$ і $A(t)$ будуть сприйматися як різні. У цьому випадку, щоб переконатися в істинності вихідного узагальненого закону Талбота, довелося б збільшити число (нескінченно, що практично неможливо зробити. Таким чином, при невиконанні посиленого узагальненого закону Талбота істинність вихідного закону було б неможливо перевірити в експерименті. У результаті буде зроблено висновок: якщо узагальнений закон Талбота підтверджується на практиці, то повинен бути вірний також і посилений узагальнений закон Талбота.

Нове формулювання закону слуху є в логічному змісті більш сильним твердженням, ніж вихідне. Для доказу цього положення досить формально вивести колишнє формулювання закону з нового і показати, що зворотний висновок, загалом, неможливий. Припустимо, що виконується посилений узагальнений закон Талбота. Тоді з умови (2.2) випливає існування числа $\omega_0 > 0$ такого, що для кожного $\omega > \omega_0$ відчуття звуку $B_\omega(t)$ збіжиться з відчуттям звуку $B(t)$. Звідси безпосередньо випливає, що й у межі при $\omega \to \infty$, відчуття звуку $B_\omega(t)$ збіжиться з відчуттям звуку $A(t)$. Разом з тим ясно, що, якщо з умови (2.2)

випливає рівність відчуттів звуків $B_\omega(t)$ і $A(t)$ у межі при $\omega \to \infty$, то звідси, загалом кажучи, зовсім не випливає логічно, що найдеться таке число ω_0, що при кожному $\omega > \omega_0$ буде мати місце рівність відчуттів звуків $B_\omega(t)$ і $A(t)$. Цією обставиною виправдовується використання слова «посилений» у назві нового формулювання закону слуху.

У посиленому узагальненому законі Талбота мова йде про існування деякого числа $\omega_0 > 0$. Вибір цього числа не є єдиним. Насправді, якщо в якості ω_0 підходить деяке число α, те в цій же ролі підійде також і будь-яке інше число $\beta > \alpha$. Очевидно, однак, що множина M усіх чисел, кожне з яких можна використовувати в ролі числа ω_0, обмежене знизу. Нижню границю множини M позначити символом $\omega_{кр}$ і назвати критичним значенням параметра ω. Важливо звернути увагу на те, що величину $\omega_{кр}$ було б неправильно апріорі вважати ні від чого не залежної (тобто абсолютної) константою. Логічно можливо, щоб число $\omega_{кр}$ залежало від звуку $A(t)$ і від множини звуків $\{B_\omega(t)\}$, $\omega \in (0,\infty)$. Нижче буде виконано експериментальне визначення величини $\omega_{кр}$ для випадків, коли $A(t)$ є звук мови, а $B_\omega(t)$ є тимчасовою послідовністю стандартних імпульсів, що фізично реалізує той або інший варіант частотно-імпульсного коду цього звуку.

2.3 Алгоритм формування асинхронних кодів звуків

У цьому підрозділі для звуку $A(t)$ будується спеціальне сімейство $\{B_\omega(t)\}_{\omega \in (0,\infty)}$ частотно-імпульсних кодів, що задовольняє умові (2.2) Нехай $A(t)$ – діаграма довільно обраного звуку мови, ординати якої на всім інтервалі завдання $[0, T_0]$ обмежені за абсолютним значенням фіксованим позитивним числом α:

$$-\alpha \le A(t) \le \alpha, \ (0 \le t \le T_0). \qquad (2.3)$$

Для будь-яких реальних звуків мови умова (2.3) може бути практично виконаною при підходящому виборі числа. Число ухвалюється рівнянням:

$$\alpha = \max_{0 \le t \le T_0} \left| A(t) \right| . \tag{2.4}$$

Вибрати деяке речовинне число α_0, що задовольняє умові:

$$\alpha_0 > \alpha. \tag{2.5}$$

Уведемо позитивний речовинний параметр (і пов'язаний з ним параметр (, що змінюється на інтервалі (0,() і обумовлений формулою:

$$\omega = \frac{\alpha_0 - \alpha}{\varepsilon} . \tag{2.6}$$

Будується послідовність моментів часу θ_1, θ_2, ..., θ_m, задаючи останні рівностями:

$$\int_0^{\theta_1} (A(t) + \alpha_0) dt = \varepsilon, \int_{\theta_1}^{\theta_2} (A(t) + \alpha_0) dt = \varepsilon, ..., \int_{\theta_{m-1}}^{\theta_m} (A(t) + \alpha_0) dt = \varepsilon . \tag{2.7}$$

У якості числа m приймається найбільше з натуральних чисел, що задовольняє умові

$$\theta_m \le T_0 \tag{2.8}$$

У моменти часу θ_1, θ_2, ..., θ_m формуються короткі стандартні імпульси, кожний з яких охоплює площу ε. Отриману послідовність імпульсів, після її переміщення вниз на величину α_0, прийняти в якості діаграми звуку $B_\omega(t)$.

Діаграму $B_\omega(t)$ можна з достатньою точністю представити аналітично в наступному виді:

$$B_\omega(t) = -\alpha_0 + \varepsilon\sum_{i=1}^{m}\delta(t-\theta_i)\,. \qquad (2.9)$$

Тут $\delta(t\text{-}\theta_i)$ – функція Дірака. Вона задає імпульс зневажливо малої тривалості, що виникає в момент часу θ_i, який охоплює одиничну площу. Акустичну діаграму $B_\omega(t)$ назвемо асинхронним частотно-імпульсним кодом звуку $A(t)$.

Тільки що описаний спосіб формування асинхронного частотно-імпульсного коду $B_\omega(t)$ звуку $A(t)$ проілюстровано за допомогою рис. 2.6. На рис. 2.6, а) зображений приклад діаграми звуку $A(t)$, а на рис. 2.6, б) – асинхронний частотно-імпульсний код $B_\omega(t)$ звуку $A(t)$.

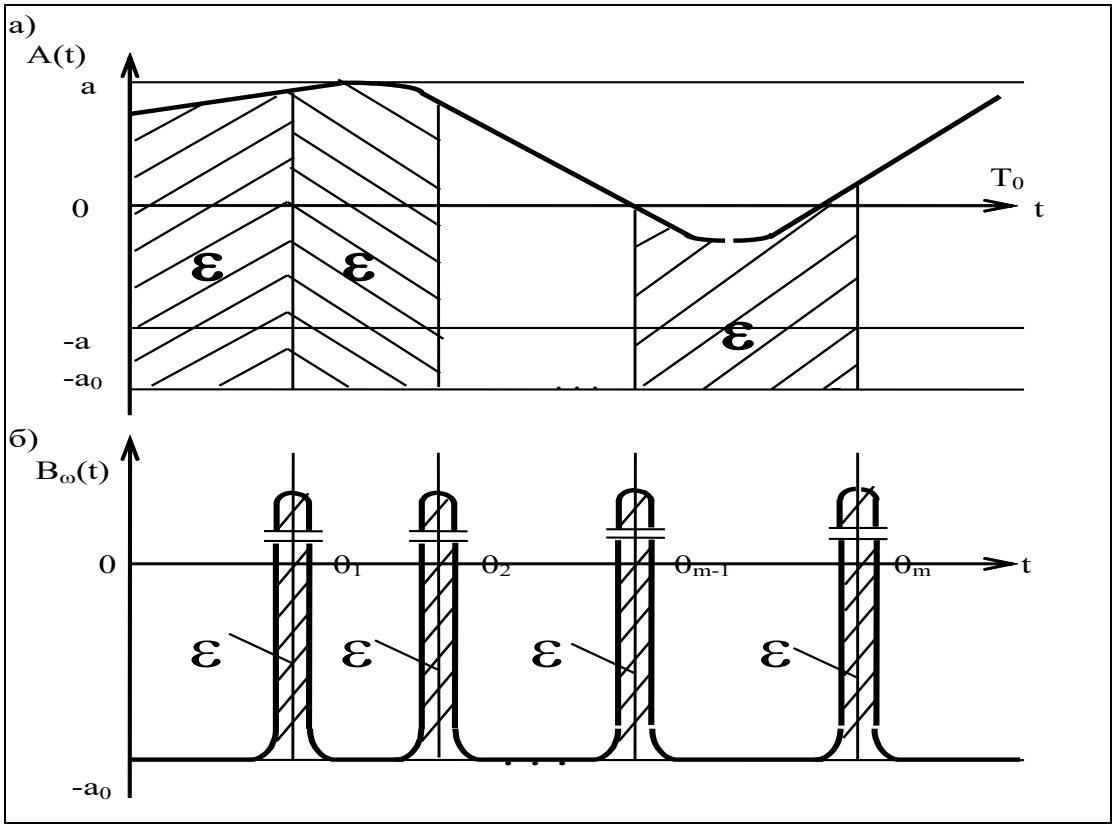

Рисунок 2.6 – Спосіб формування асинхронного частотно-імпульсного коду звуку

Черговий імпульс виробляється в той момент, коли під кривій $A(t)+\alpha_0$ накопляється площа величини ε. Після появи імпульсу процес нагромадження площі починається знову.

Якби було $A(t) \equiv -\alpha$, тобто якби рівень звукового тиску на всім інтервалі завдання акустичної діаграми $A(t)$ був рівний своєму найменшому можливому значенню, то всі інтервали між сусідніми імпульсами асинхронного частотно-імпульсного коду $B_\omega(t)$ звуку $A(t)$ були б однаковими й досягали своєї максимально можливої величини

$$T = \frac{\varepsilon}{\alpha_0 - \alpha}. \qquad (2.10)$$

Остання рівність безпосередньо випливає з рівнянь (2.7) при підстановці в них $A(t)=-\alpha$ і обліку того факту, що $T=\theta_i-\theta_{i-1}$, $(1 \le i \le m)$. Тут мається на увазі, що $\theta_0=0$. Використовуючи (2.6), одержано $\omega=1/T$.

Таким чином, фізичний зміст параметра ω наступний. Це є мінімальна частота проходження імпульсів в асинхронному частотно-імпульсному коді $B_\omega(t)$ звуку $A(t)$. Частоту (можна регулювати практично, змінюючи величину площі (, охоплюваної кожним стандартним імпульсом. Згідно з формулою (2.6) зменшення величини (веде до збільшення частоти).

Доведемо тепер, що побудоване описаним вище способом множина $\{B_\omega(t)\}_{\omega\in(0,\infty)}$ асинхронних частотно-імпульсних кодів звуку $A(t)$ задовольняє умові (2.2). Вибирається на інтервалі $[0,T_0]$ довільним образом моменти часу t_1 і t_2, починаючи, з такого, що задовольняє умові $t_1<t_2$. Коли $t_1=t_2$, то умова (2.2), має виконуватися. Позначено через p таке найменше натуральне число й через q таке найбільше натуральне число, для яких виконуються співвідношення

$$t_1 < \theta_p,\ \theta_q \le t_2,\ (1 \le p \le q \le m). \qquad (2.11)$$

Легко бачити, що при досить великому ω числа p і q завжди існують. Згідно (2.9):

$$\int_{t_1}^{t_2} B_\omega(t)dt = -\int_{t_1}^{t_2} \alpha_0 dt + \int_{t_1}^{t_2} \varepsilon \sum_{i=1}^{m} \delta(t-\theta_i)dt =$$

$$= \alpha_0(t_1 - t_2) + \varepsilon \sum_{i=1}^{q} \int_{t_1}^{t_2} \delta(t-\theta_i)dt = \alpha_0(t_1 - t_2) + \varepsilon(q - p + 1)\,.$$

З використанням залежності (2.7):

$$\int_{t_1}^{t_2} A(t)dt = \int_{t_1}^{t_2} (A(t) + \alpha_0)dt - \int_{t_1}^{t_2} \alpha_0 dt =$$

$$= \int_{t_1}^{\theta_{p-1}} (A(t) + \alpha_0)dt + \int_{\theta_{p-1}}^{\theta_q} (A(t) + \alpha_0)dt + \int_{\theta_q}^{t_2} (A(t) + \alpha_0)dt + \alpha_0(t_1 - t_2) =$$

$$= -\int_{\theta_{p-1}}^{t_1} (A(t) + \alpha_0)dt + \varepsilon(q - p + 1) + \int_{\theta_q}^{t_2} (A(t) + \alpha_0)dt + \alpha_0(t_1 - t_2).$$

Після цього:

$$\left| \int_{t_1}^{t_2} B_\omega(t)dt - \int_{t_1}^{t_2} A(t)dt \right| = \left| \int_{\theta_{p-1}}^{t_1} (A(t) + \alpha_0)dt - \int_{\theta_q}^{t_2} (A(t) + \alpha_0)dt \right|.$$

Виконані викладення приводять до наступного результату:

$$\leq \int_{\theta_{p-1}}^{t_1} (A(t) + \alpha_0)dt + \int_{\theta_q}^{t_2} (A(t) + \alpha_0)dt \leq \int_{\theta_{p-1}}^{t_1} (\alpha + \alpha_0)dt + \int_{\theta_q}^{t_2} (\alpha + \alpha_0)dt \leq$$

$$\leq (t_1 - \theta_{p-1})(\alpha + \alpha_0) + (t_2 - \theta_q)(\alpha + \alpha_0) \leq T(\alpha + \alpha_0) + T(\alpha + \alpha_0) = 2\frac{\alpha + \alpha_0}{\omega}.$$

Це означає, що побудована множина $\{B_\omega(t)\}_{\omega \in (0,\infty)}$ асинхронних частотно-імпульсних кодів звуку A(t) задовольняє умові (2.2).

$$\lim_{\omega\to\infty}\left|\int_{t_1}^{t_2} B_\omega(t)dt - \int_{t_1}^{t_2} A(t)dt\right| = 0. \qquad (2.12)$$

2.4 Визначення критичної частоти асинхронного коду звуку

Розглянутий спосіб формування сімейства $\{B_\omega(t)\}_{\omega\in(0,\infty)}$ асинхронних частотно-імпульсних кодів звуку A(t), що задовольняє умові (2.2). Відповідно посиленому узагальненому закону Талбота, для цього сімейства повинне існувати критичне значення $\omega_{кр}$ параметра ω. Величину $\omega_{кр}$ називають критичною частотою проходження імпульсів в асинхронному частотно-імпульсному коді $B_\omega(t)$ звуку $A(t)$.

Відповідно до визначення критичної частоти, якщо на всім інтервалі [0, T_0] завдання звуків частота проходження імпульсів у звуці $B_\omega(t)$ перевищує критичну, тобто $\omega>\omega_{кр}$, те звуки $B_\omega(t)$ і $A(t)$ повинні звучати однаково. Разом з тим, слухові відчуття звуків $B_\omega(t)$ і $A(t)$ при $\omega = \omega_{кр}$ повинні різнитися.

У цьому підрозділі описується блокова схема обладнання для визначення критичної частоти $\omega_{кр}$ різних звуків $A(t)$ і їх асинхронних частотно-імпульсних кодів $B_\omega(t)$ (див. рис. 2.7). Безперервний звук $A(t)$ сприймається мікрофоном 1, який перетворює його в коливання електричної напруги $U(t)$.

Мікрофон має передатний коефіцієнт до$_1$. Напруга $U(t)$, після посилення в p раз і зсув нагору на величину b блоком 2, надходить у вигляді напруги $V(t)$ на граничний елемент 3.

Граничний елемент виробляє деяку послідовність стандартних імпульсів, кожний з яких охоплює площа g. Отримана послідовність імпульсів $P(t)$ підсилюється в q раз і зміщається вниз на величину із блоком 4. У результаті формується сигнал $Q(t)$, який надходить через перемикач 5 на телефон 6 і прослуховується випробуваним як звук $B_\omega(t)$.

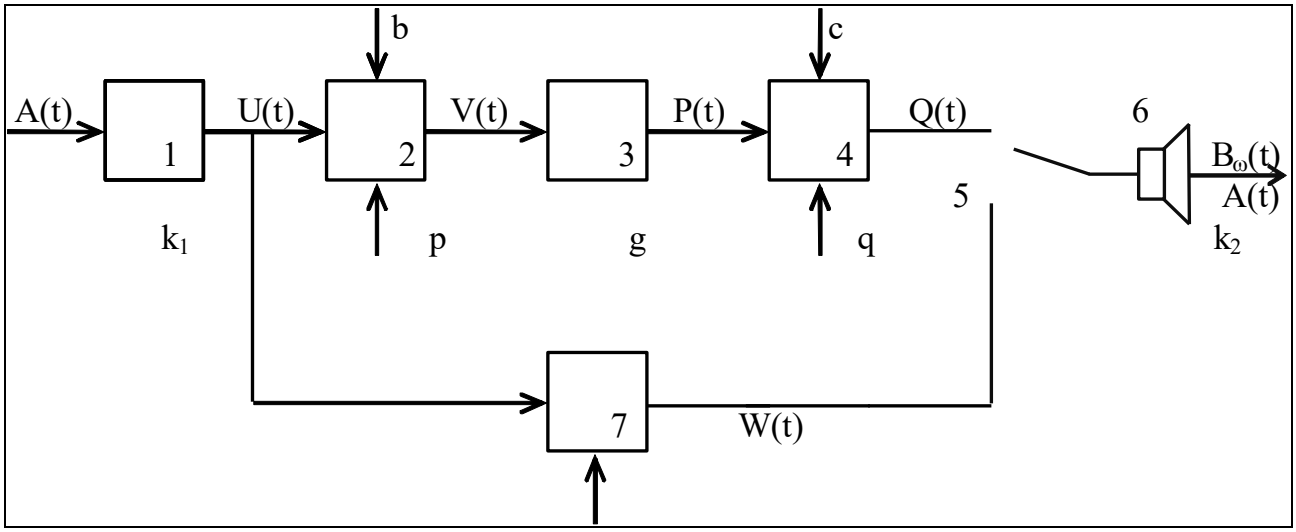

Рисунок 2.7 – Блокова схема квантователя звуку

Через той же перемикач на телефон надходить сигнал $W(t)$, що представляє собою посилений в r раз блоком 7 сигнал $U(t)$. Сигнал $W(t)$ після проходження через телефон, що має передатний коефіцієнт до$_2$, прослуховується випробуваним у ролі звуку $A(t)$.

Математична модель функціонування окремих блоків квантователя звуку. Перетворенням сигналів у блоках 1 і 7 відповідають рівняння:

$$U(t)=k_1A(t), \tag{2.13}$$

$$W(t)=ru(t). \tag{2.14}$$

Робота блоку 6 відповідно при верхньому й нижньому положенні ключа 5 описується рівняннями:

$$B_\omega(t)=k_2Q(t), \tag{2.15}$$

$$A(t)=k_2W(t). \tag{2.16}$$

Для збігу вхідного й вихідного сигналів тракту 1-7-6 необхідно прийняти:

$$r=1/k_1k_2. \tag{2.17}$$

Перетворення сигналів блоками 2 і 4 описуються наступними рівностями:

$$V(t)=pu(t)+b, \tag{2.18}$$

$$Q(t)=qp(t)\text{-с}. \tag{2.19}$$

Послідовність імпульсів, вироблювану блоком 3, математично описуємо вираженням

$$P(t)=\text{g}\sum_{\text{i=1}}^{\text{m}}\delta(\text{t}-\theta_{\text{i}}). \tag{2.20}$$

$$\int_0^{\theta_1}V(t)dt=\text{g},\ \int_{\theta_1}^{\theta_2}V(t)dt=\text{g},\ \ldots,\ \int_{\theta_{\text{m-1}}}^{\theta_{\text{m}}}V(t)dt=\text{g}. \tag{2.21}$$

Моменти часу θ_1, θ_2, ..., θ_m, у які виробляються імпульси, визначається рівняннями.

Модель включає вирішення питання про вибір параметрів настроювання *b*, *c*, *p* і *q* квантователю, при якім звуки $B_\omega(t)$ і $A(t)$ будуть задовольняти умові (2.2). Вибір величини параметра *r* квантователю вже здійснений: він визначається формулою (2.17). Регулювання параметрів k_1, k_2 і *g*, на відміну від тільки що згаданих параметрів, у квантователі не передбачене.

Виведено формули для визначення параметрів настроювання *p* и *b* блоку 2. Із цією метою по формулах (2.13) і (2.18) знаходимо:

$$V(t)=pk_1A(t)+b. \tag{2.22}$$

Підставляючи отриманий результат в і-*е* з рівностей (2.21):

$$\int_{\theta_{i-1}}^{\theta_i} \frac{pk_1 A(t) + b}{g} dt = 1. \tag{2.23}$$

$$\int_{\theta_{i-1}}^{\theta_i} \frac{A(t) + \alpha_0}{\varepsilon} dt = 1. \tag{2.24}$$

Дорівнюючи коефіцієнти при $A(t)$ і вільні члени в підінтегральних вираженнях рівностей (2.23) і (2.24), знаходимо шукані формули для p и b:

$$p = g/k_1\varepsilon, \tag{2.25}$$

$$b = \alpha_0 g/\varepsilon. \tag{2.26}$$

Далі, виведемо формули для визначення параметрів настроювання q і c блоку 4. Із цією метою по формулах (2.15), (2.19) і (2.20) знаходимо вираження для сигналу $B_\omega(t)$:

$$B_\omega(t) = k_2 q g \sum_{i=1}^{m} \delta(t - \theta_i) - k_2 c. \tag{2.27}$$

З іншого боку, згідно (2.9) маємо:

$$B_\omega(t) = \varepsilon \sum_{i=1}^{m} \delta(t - \theta_i) - \alpha_0. \tag{2.28}$$

Дорівнюючи коефіцієнти при сумі й вільні члени в правих частинах рівностей (2.27) і (2.28), знаходимо шукані формули для q і p:

$$q = \varepsilon/k_2 g, \tag{2.29}$$

$$c=\alpha_0/k_2. \quad (2.30)$$

Параметр *r* повинен регулюватися таким чином, щоб звучання в телефоні збіглося по гучності з безпосереднім сприйняттям звуку *A*(*t*) поблизу мікрофона. Значення параметрів *b* і *p* повинні забезпечити, по-перше, щоб коливання сигналу *V*(*t*) укладалося в межі робочого діапазону граничного елемента й, по-друге, досягалося бажане значення мінімальної частоти ω.

По формулах (2.22), (2.25) і (2.26):

$$V(t)=\frac{g}{\varepsilon}(A(t)+\alpha_0). \quad (2.31)$$

Мінімальне значення *VM* сигналу *V*(*t*) досягається у випадку, коли *A*(*t*)=-α. У такий спосіб

$$V_M=\frac{g}{\varepsilon}(\alpha_0-\alpha). \quad (2.32)$$

Порівнюючи формули (2.6) і (2.32:

$$\omega=\frac{1}{g}V_M. \quad (2.33)$$

Таким чином, параметр ω можна регулювати до бажаної величини відповідно до формули (2.33), змінюючи напругу *VM*. При цьому величина параметра згідно з формулою (2.32), повинна бути прийнята рівної

Посилення *q* повинне забезпечувати рівність гучності сигналів $B_\omega(t)$ і *A*(*t*) при почерговому їхньому прослуховуванні. Величина зсуву *c* фактично не має потреби в регулюванні, оскільки її зміна викликає лише статичний прогин мембрани пристрою відтворення, що не впливає на якість звуку. Вона повинна

вибиратися з таким розрахунком, щоб статичний прогин мембрани телефону перебував у припустимих межах і якість звуку не погіршувалася.

В експериментах необхідно виміряти найбільший нижній рівень напруги $V_{кр}$ для звуку $A(t)$, при якім відчуття звуків $B_\omega(t)$ і $A(t)$ ще різняться.

Цей рівень можна визначити за допомогою електронного осцилографа, на екрані якого візуально спостерігається коливання в часі напруги $V(t)$. Критичну частоту можна визначити відповідно до формули (2.33):

$$\omega_{\text{ф р}} = \frac{1}{g} V_{\text{ф р}}. \qquad (2.34)$$

Для забезпечення точності експериментів визначення величини $\omega_{кр}$ необхідно повторювати багаторазово для кожного звуку $A(t)$ і кожного аудитора, після чого обчислити середні значення $\omega_{кр}$ і оцінити розкид значень $\omega_{кр.}$

Критична частота $\omega_{кр}$ повинна практично не залежати від виду звуку $A(t)$, і незначно залежати від вибору аудитора. Приблизно критична частота майже в точності повинна збігатися з максимальною частотою синусоїдального сигналу, чутного вухом, тобто перебувати в межах 16-20 кГц. Критична частота в різних випробуваних строго корелюється з верхньою межею чутної частоти. Можливо наступне пояснення цього факту: як тільки частота проходження імпульсів у сигналі $B_\omega(t)$ знижується на стільки, що попадає в чутний діапазон, вона починає сприйматися вухом. Підтвердженням цьому поясненню може служити наступне спостереження. Коли мінімальна частота проходження імпульсу в сигналі $B_\omega(t)$ хоча б на невеликих інтервалах часу знижується до величини $\omega_{кр}$ і звуки $A(t)$ і $B_\omega(t)$ стають різними на слух, та ця відмінність проявляється лише в тому, що у відчутті звуку $B_\omega(t)$ на тлі звуку $A(t)$ з'являється високочастотні завади.

3 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

3.1 Алгоритм порівняння звукової інформації

На виході розробленої системи одержано озвучені вихідні й сумарний звукові діаграми (у форматі .wav), що забезпечується блоком відтворення.

Програмний модуль створення й редагування wav-файлів Wav_Edit може створювати довільні звукові діаграми, однак для роботи системи досить синусоїдальних сигналів із заданими частотою й амплітудою. Формат створюваних файлів: 16ббіт, тактова частота 44100 Гц (див. рис. 3.1).

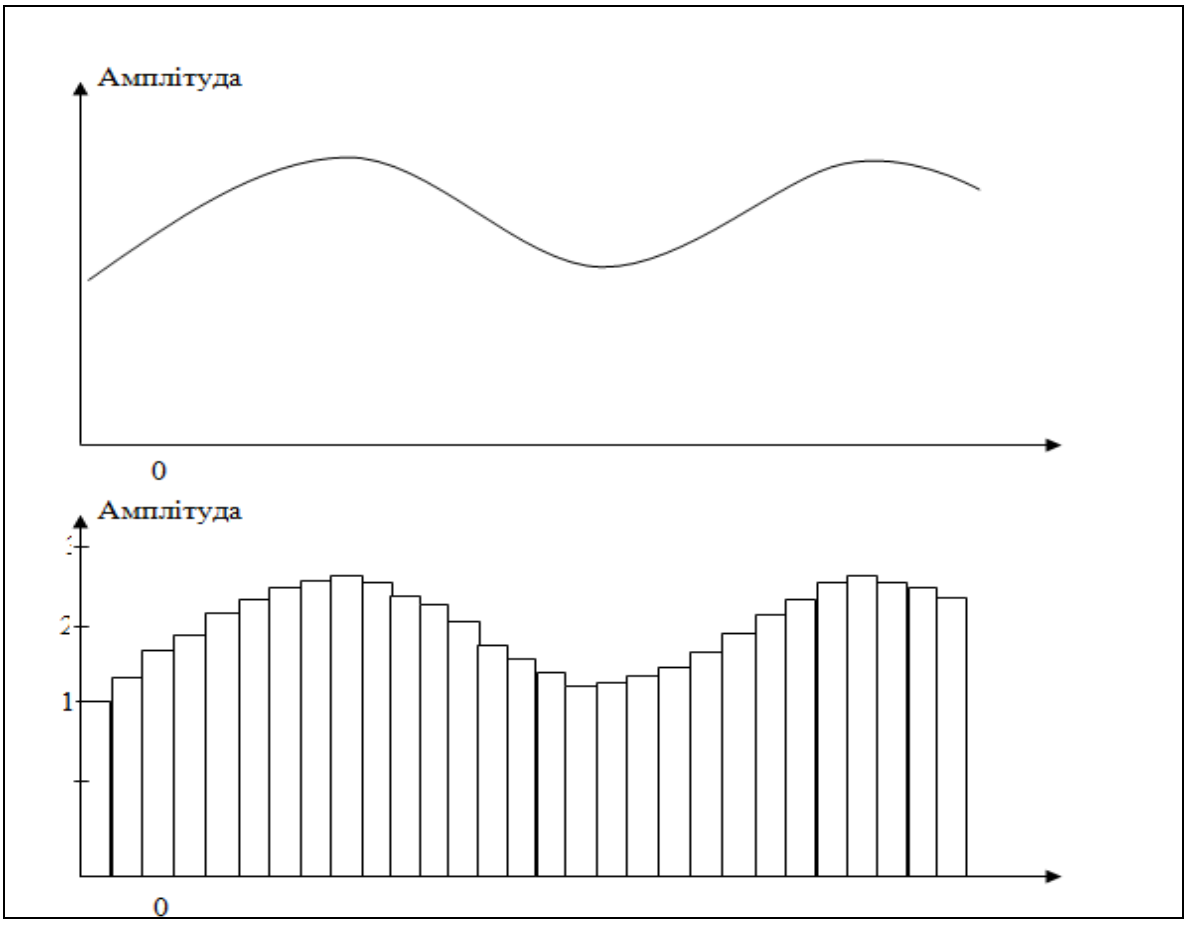

Рисунок 3.1 – Аналоговий синусоїдальний сигнал і його дискретна звукова діаграма після АЦП

Розглядається wav-файл (Windows PCM), він являє собою дві, що чітко діляться, області. Одна з них – заголовок файлу, інша – область даних.

У заголовку wav-файлу зберігається інформація про:

– розмірі файлу;

– кількості каналів;

– частоті дискретизації;

– кількості біт у семплі (цю величину ще називають глибиною звучання).

Для більшого розуміння змісту величин у заголовку розглянемо більш докладно область даних і оцифровку звуку. Звук складається з коливань, які при оцифровці здобувають східчастий вид. Цей вид обумовлений тим, що комп'ютер може відтворювати в будь-який короткий проміжок часу звук певної амплітуди (гучності) і цей короткий момент далеко не нескінченно короткий. Тривалість цього проміжку й визначає частота дискретизації. Наприклад, у нас файл із частотою дискретизації 44.1 кГц, це значить, що той короткий проміжок часу рівний 1/44100 секунди (випливає з розмірності величини Гц = 1/с). Сучасні звукові карти підтримують частоту дискретизації до 192 кГц.

Тепер, що стосується амплітуди (гучності звуку в короткому проміжку часу). Амплітуда виражається числом, займаним у пам'яті (файлі) 8, 16, 24, 32 біт (теоретично можна й більше). Якась одна амплітуда в якийсь короткий проміжок часу в пам'яті (файлі) може займати 1, 2, 3, 4 байта відповідно. Таким чином, чим більше число займає місця в пам'яті (файлі), тим більше діапазон значень для цього числа, а значить і для амплітуди.

1 байт – 0..255;

2 байта – 0..65535;

3 байта – 0..16777216;

4 байта – 0..4294967296.

При цьому необхідно враховувати, що діаграма звуку ділиться на позитивну й негативну частині. Звідси випливає, що модуль значення амплітуди для діаграми, наприклад, 16обітного формату, не може перевищувати значення 65535/2=32676.

У моно варіанті значення амплітуди розташовані послідовно. У стерео ж, наприклад, спочатку йде значення амплітуди для лівого каналу, потім для правого, потім знову для лівого і так далі.

Сукупність амплітуди й короткого проміжку часу є семпл.

Формат заголовка wavйфайлів має вигляд:

– RIFF – формат звукового файлу;

– розмір даних фрагмента RIFF (4 байта);

– Wavefmt – ідентифікатор формату даних;

– розмір формату структури WAVEFORMATEX;

– ідентифікатор типу звукового формату;

– кількість каналів (1 – моно, 2 – стерео);

– частота дискретизації, Гц;

– кількість байт у секунду;

– розмір хвилини одиниці вимірів, у байтах;

– кількість біт в одній вибірці сигналу (8 або 16);

– data – заголовок фрагмента даних;

– розмір даних;

– безпосередньо звукові дані.

Програма ТестЗвук може, крім того, сама генерувати також шумові сигнали – білий шум і суміш синусоїдальних сигналів з випадковими частотними параметрами.

У блоці створення вихідних звукових діаграм (програма ТестЗвук) дискретні звукові діаграми перетворяться для наступного підсумовування в блоці підсумовування звукових діаграм із заданими тактовою частотою й величиною шпаруватості. Після завдання параметрів відтворення звукові діаграми подаються на лінійний вихід комп'ютера для озвучування акустичною системою.

3.2 Вибір інструментарію програмування

Останнім часом С і С + + є найбільш використовуваними мовами для розробки комерційних і бізнес додатків. Ці мови влаштовують багатьох розробників, але насправді не забезпечують належної продуктивності розробки. Наприклад, процес написання програми на С + + часто займає значно більше часу, ніж розробка еквівалентного додатки, скажімо, на Visual Basic. Зараз існують мови, що збільшують продуктивність розробки за рахунок втрати в гнучкості, яка так звична і необхідна програмістам на С / С + +. Подібні рішення є досить незручними для розробників і часто пропонують значно менші можливості. Ці мови також не орієнтовані на взаємодію з з'являються сьогодні системами і дуже часто вони не відповідають існуючій практиці програмування для Web. Багато розробники хотіли б використовувати сучасну мову, який дозволяв би писати, читати і супроводжувати програми з простотою Visual Basic і в той же час давав міць і гнучкість С + +, забезпечував доступ до всіх функціональних можливостей системи, взаємодіяв б з існуючими програмами і легко працював з виникаючими Web стандартами.

Враховуючи всі подібні побажання, Microsoft розробила нову мову - С #. У нього входить багато корисних особливостей - простота, об'єктна орієнтованість, типова захищеність, "збірка сміття", підтримка сумісності версій і багато іншого. Дані можливості дозволяють швидко і легко розробляти програми, особливо COM + додатки і Web сервіси. При створенні С #, його автори враховували досягнення багатьох інших мов програмування: С + +, С, Java, SmallTalk, Delphi, Visual Basic і т.д. Треба зауважити що з причини того, що С # розроблявся з чистого листа, у його авторів була можливість (якої вони явно скористалися), залишити в минулому всі незручні і неприємні особливості (існуючі, як правило, для зворотної сумісності), будь-якого з попередніх йому мов . В результаті вийшов дійсно простий, зручний і сучасний мову, за потужністю не поступається С + +, але істотно підвищує продуктивність розробок.

Дуже часто можна простежити таку зв'язок - чим більше мова захищена і стійкий до помилок, тим менше продуктивність програм, написаних на ньому. Наприклад розглянемо дві крайнощі - очевидно це Assembler і Java. У першому випадку ви можете добитися фантастичної швидкості своєї програми, але вам доведеться дуже довго примушувати її працювати правильно не так на вашому комп'ютері. У випадку ж з Java - ви отримуєте захищеність, незалежність від платформи, але, на жаль, швидкість вашої програми навряд чи сумісна зі сформованим поданням про швидкість, наприклад, якого окремого клієнтського застосування (звичайно існують застереження - JIT компіляція та інше). Розглянемо C + + з цієї точки зору - на мій погляд співвідношення в швидкості і захищеності близько до бажаного результату, але на основі власного досвіду програмування я можу з упевненістю сказати, що практично завжди краще понести незначну втрату в продуктивності програми і придбати таку зручну особливість, як " збірка сміття ", яка не тільки звільняє вас від стомлюючої обов'язки керувати пам'яттю вручну, але і допомагає уникнути вам багатьох потенційних помилок у вашому додатку. Насправді скоро "збірка сміття", та й будь-які інші кроки до усунення потенційних помилок стану відмінними рисами сучасної мови. У C #, як в безсумнівно сучасній мові, також існують характерні особливості для обходу можливих помилок. Наприклад, крім згаданої вище "збірки сміття", там всі змінні автоматично инициализируются середовищем і володіють типовою захищеністю, що дозволяє уникнути невизначених ситуацій у разі, якщо програміст забуде ініціалізувати змінну в об'єкті або спробує справити неприпустиме перетворення типів. Також в C # були зроблені заходи для виключення помилок при оновленні програмного забезпечення. Зміна коду, в такій ситуації, може непередбачувано змінити суть самої програми. Щоб допомогти розробникам боротися з цією проблемою C # включає підтримку сумісності версій (vesioning). Зокрема, на відміну від C + + і Java, якщо метод класу був змінений, це повинно бути спеціально обумовлено. Це дозволяє обійти помилки в коді і забезпечити гнучку сумісність версій. Також новою особливістю

є native підтримка інтерфейсів і спадкоємства інтерфейсів. Дані можливості дозволяють розробляти складні системи і розвивати їх з часом.

У C # була уніфікована система типів, тепер ви можете розглядати кожен тип як об'єкт. Незважаючи на те, використовуєте ви клас, структуру, масив або вбудований тип, ви можете звертатися до нього як до об'єкта. Об'єкти зібрані в простори імен (namespaces), які дозволяють програмно звертатися до чого-небудь. Це означає що замість списку включення файл заголовків у своїй програмі ви повинні написати які простори імен, для доступу до об'єктів і класам всередині них, ви хочете використовувати. У C # вираз using дозволяє вам не писати кожного разу назву простору імен, коли ви використовуєте клас з нього. Наприклад, простір імен System містить декілька класів, в тому числі і Console. І ви можете писати або назва простору імен перед кожним зверненням до класу, або використовувати using як це було показано в прикладі вище.

Важливою і відмітною від C + + особливістю C # є його простота. Приміром, чи завжди ви пам'ятаєте, коли пишіть на C + +, де потрібно використовувати "->", де "::", а де "."? Навіть якщо немає, то компілятор завжди поправляє вас у разі помилки. Це говорить лише про те, що насправді можна обійтися тільки одним оператором, а компілятор сам розпізнаватиме його значення. Так в C #, оператор "->" використовується дуже обмежена (в unsafe блоках, про які йтиметься нижче), оператор "::" взагалі не існує. Практично завжди ви використовуєте тільки оператор "." і вам більше не потрібно стояти перед вибором.

Ще один приклад. При написанні програм на C / C + + вам доводилося думати не тільки про типи даних, а й про їх розмір в конкретній реалізації. У C # все спрощено - тепер символ Unicode називається просто char (а не wchar_t, як в C + +) і 64-бітове ціле тепер - long (а не __ int64). Також в C # немає знакових і беззнакових символьних типів.

У C #, також як і в Visual Basic після кожного виразу case в блоці switch мається на увазі break. І більше не відбуватиметься дивних речей якщо ви забули поставити цей break. Однак якщо ви дійсно хочете щоб після одного виразу case

програма перейшла до наступного ви можете переписати свою програму з використанням, наприклад, оператора goto.

Багатьом програмістам (на той момент, напевно, майбутнім програмістам) було не так легко під час вивчення C + + повністю освоїтися з механізмом посилань і покажчиків. У C # (хтось зараз згадає про Java) немає покажчиків. Насправді нетривіальність покажчиків відповідала їх корисності. Наприклад, часом, важко собі уявити програмування без покажчиків на функції. Відповідно до цього в C # присутні Delegates - як прямий аналог покажчика на функцію, але їх відрізняє типова захищеність, безпека і повна відповідність концепціям об'єктно-орієнтованого програмування.

Хотілося б підкреслити сучасну зручність C #. Коли ви почнете роботу з C #, а, сподіваюся, це відбудеться якомога швидше, ви побачите, що досить велике значення в ньому мають простору імен. Вже зараз, на основі першого прикладу, ви можете судити про це - адже всі файли заголовків замінені саме простором імен. Так в C #, крім просто вираження using, надається ще одна дуже зручна можливість - використання додаткового імені (alias) простору імен або класу.

Сучасність C # виявляється і в нових кроках до полегшення процесу налагодження програми. Традиційні засоби для налагодження програм на стадії розробки в C + + є маркування великих частин коду директивами # ifdef і т.д. У C #, використовуючи атрибути, орієнтовані на умовні слова, ви можете куди швидше писати налагоджувати код.

У наш час, коли посилюється зв'язок між світом комерції і світом розробки програмного забезпечення, і корпорації витрачають багато зусиль на планування бізнесу, відчувається необхідність відповідно абстрактних бізнес процесів їх програмним реалізаціям. На жаль, більшість мов реально не мають прямого шляху для зв'язку бізнес логіки та коду. Наприклад, сьогодні багато програмісти коментують свої програми для пояснення того, які класи реалізують якийсь абстрактний бізнес об'єкт. C # дозволяє використовувати типізовані, розгортаються метадані, які можуть бути прикріплені до об'єкта. Архітектурою проекту можуть визначатися локальні атрибути, які будуть пов'язані з будь-якими

елементами мови - класами, інтерфейсами і т.д. Розробник може програмно перевірити атрибути якого елемента. Це істотно спрощує роботу, наприклад, замість того щоб писати автоматизований інструмент, який буде перевіряти кожен клас або інтерфейс, на те, чи є він дійсно частиною абстрактного бізнес об'єкта, можна просто скористатися повідомленнями заснованими на визначених у об'єкті локальних атрибутах.

C #, будучи останнім з широко поширених мов програмування, повинен увібрати в себе весь наявний досвід і увібрати кращі сторони існуючих мов програмування, при цьому будучи спеціально створеним для роботи в. NET. Сама архітектура. NET продиктувала йому (як і багатьом іншим мовам, на яких можна писати під. NET) об'єктно-орієнтовану спрямованість. Звичайно, це не є правилом, можливе створення компіляторів навіть функціональних мов по. NET, на цю тему існують спеціальні роботи.

Свій синтаксис C # в чому успадкував від C + + і Java. Розробники, які мають досвід написання додатків на цих мовах, знайдуть в C # багато знайомих рис. Але разом з тим він є багато в чому новаторським - аттрібути, делегати та події, прекрасно вписані в загальну ідеологію мови, міцно зайняли місце в серцях. NET - розробників. Їх введення дозволило застосовувати принципово нові прийоми програмування.

Звичайно, улюбленим об'єктом для порівняння з C # у світової ком'юніті є Java. Також розроблений для роботи в віртуальному середовищі виконання, що має об'єктно-орієнтовану архітектуру і збирач сміття, основанний на механізмі посилань. При порівнянні з цією мовою відразу вироблена такі особливості, як можливість оголошувати декілька класів в одному файлі, з чого випливає синтаксична підтримка ієрархічної системи просторів імен. З реалізації ООП-концепцій подібність у механізмі наслідування і реалізації (і в Java і в C # можливо одиничне успадкування, але множинна реалізація інтерфейсів, на відміну від C + +). Але в Java відсутні властивості і індексатори (а також делегати та події, але вони відсутні ще багато де). Також є можливість перерахування контейнерів.

З речей, включених в специфікацію мови, але які не є чисто "програмістські" необхідно відзначити можливість використання коментарів у форматі XML. Якщо коментарі відповідають спеціально описаної структурі, компілятор по них може згенерувати єдиний XML-файл документації.

Але C # вніс і свої унікальні риси, які вже були згадані - це події, індексатори, атрибути і делегати. Всі ці елементи будуть обговорені в наступних частинах, зараз лише зазначу, що вони надають собою дуже корисні можливості, які не залишаться незатребуваними.

3.3 Обґрунтування архітектури розробки

Схема архітектури (рис. 3.1) Model-View-Controller (MVC) розділяє додаток на три основних компоненти: модель, уявлення і контролер. Платформа ASP.NET MVC являє собою альтернативу схемі веб-форм ASP.NET при створенні веб-додатків. Платформа ASP.NET MVC є легковагій платформою відображення з широкими можливостями тестування і, подібно програмам на основі веб-форм, інтегрована з існуючими функціями ASP.NET, наприклад з головними сторінками і перевіркою автентичності на основі членства. Платформа MVC визначається в збірці System.Web.Mvc.

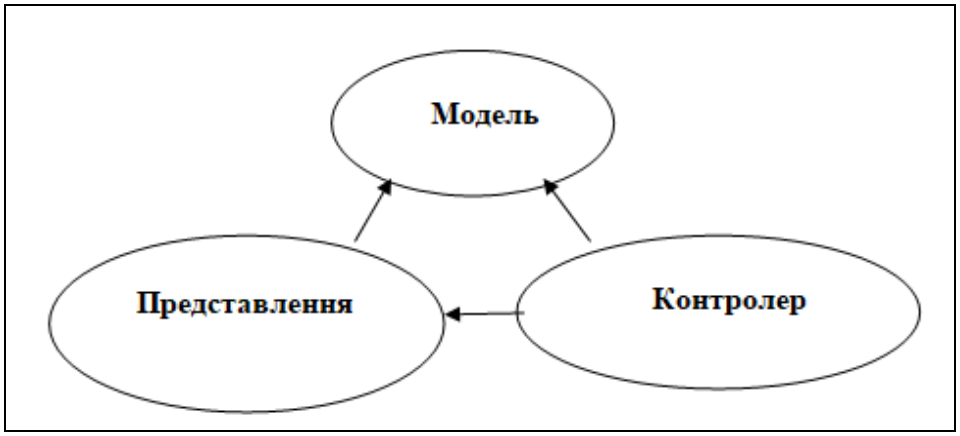

Рисунок 3.1 – Схема архітектури Model-View-Controller

MVC являє собою стандартний шаблон розробки, знайомий багатьом фахівцям. Деякі типи веб-додатків мають переваги при створенні на платформі MVC. Для інших може бути доцільно використання традиційної схеми програми ASP.NET, заснованої на веб-формах і зворотної передачі. У деяких випадках можливе поєднання двох підходів: застосування однієї схеми не виключає використання іншого.

До складу платформи MVC входять наступні компоненти.

– моделі – об'єкти моделей є частинами програми, що реалізують логіку для домену даних програми. Об'єкти моделей часто отримують і зберігають стан моделі в базі даних. Наприклад, об'єкт Product може отримувати інформацію з бази даних, працювати з нею, а потім записувати оновлені дані в таблицю Products бази даних SQL Server. У невеликих додатках ця модель має на увазі концептуальне, а не фізичний поділ. Наприклад, якщо додаток тільки зчитує набір даних і відправляє його в уявлення, то фізичний шар моделі і пов'язаних класів відсутня. У цьому випадку набір даних приймає роль об'єкта моделі;

– представлення служать для відображення інтерфейсу додатку. Інтерфейс користувача зазвичай створюється на основі даних моделі. Прикладом може служити уявлення для редагування таблиці Products, яке містить текстові поля, списки, що розкриваються і прапорці, значення яких засновані на поточний стан об'єкта Product;

– контролери здійснюють взаємодію з користувачем, роботу з моделлю, а також вибір подання, що відображає користувальницький інтерфейс. У додатку MVC подання лише відображають дані, а контролер обробляє дані, що вводяться і відповідає на дії користувача. Наприклад, контролер може обробляти строкові значення запиту і передавати їх в модель, яка може використовувати ці значення для відправлення запиту в базу даних.

Шаблон MVC дозволяє створювати додатки, різні аспекти яких (логіка введення, бізнес-логіка і логіка інтерфейсу) розділені, але досить тісно взаємодіють один з одним. Ця схема вказує розташування кожного виду логіки у додатку. Інтерфейс користувача розташовується в поданні. Логіка введення

розташовується в контролері. Бізнес-логіка знаходиться в моделі. Це розділення дозволяє працювати зі складними структурами при створенні програми, так як забезпечує одночасну реалізацію тільки одного аспекту. Наприклад, розробник може сконцентруватися на створенні подання окремо від бізнес-логіки.

Зв'язок між основними компонентами додатку MVC також полегшує паралельну розробку. Наприклад, один розробник може створювати уявлення, інший - логіку контролера, а третій - бізнес-логіку моделі.

На додаток до спрощення складних структур схема MVC також полегшує тестування додатків в порівнянні з веб-додатками ASP.NET на основі веб-форм. Наприклад, у веб-додатку ASP.NET на основі веб-форм один клас використовується для відображення виводу і для відповіді на введення користувача. Створення автоматичних тестів для додатків ASP.NET на основі веб-форм може представляти складності, так як для тестування окремої сторінки слід створити екземпляр класу сторінки, всіх дочірніх елементів управління та інших залежних класів додатки. Велике число примірників класів, необхідний для запуску сторінки, ускладнює створення тестів для окремих частин програми. Через це тестування додатків ASP.NET на основі веб-форм може бути складніше тестування додатка MVC. Більше того, для тестування програми ASP.NET необхідний веб-сервер. Платформа MVC розділяє компоненти і активно використовує інтерфейси, що дозволяє тестувати окремі елементи поза решті структури.

Слід уважно продумати питання про створення веб-додатка на основі платформи ASP.NET MVC або на основі моделі веб-форм ASP.NET. Платформа MVC не замінює собою модель веб-форм. Обидві моделі можна використовувати для веб-додатків. (за наявності існуючих додатків на основі веб-форм вони будуть продовжувати роботу в нормальному режимі).

Перед використанням платформи MVC або моделі веб-форм для певного веб-сайту слід зважити всі переваги кожного з підходів

Платформа ASP.NET MVC має такі переваги:

– полегшує управління складними структурами шляхом поділу додатки на модель, уявлення і контролер;

– не використовує стан перегляду і серверні форми. Це робить платформу MVC ідеальною для розробників, яким необхідний повний контроль над поведінкою програми;

– використовує схему основного контролера, при якій запити веб-додатка обробляються через один контролер. Це дозволяє створювати додатки, що підтримують розширену інфраструктуру маршрутизації. Додаткові відомості див Front Controller;

– забезпечує розширену підтримку розробки на основі тестування;

– добре підходить для веб-додатків, підтримуваних великими колективами розробників, а також веб-розробникам, яким необхідний високий рівень контролю над поведінкою програми.

Платформа ASP.NET MVC надає наступні можливості:

– поділ завдань додатки (логіка введення, бізнес-логіка і логіка користувача інтерфейсу), широке можливості тестування і розробки на основі тестування. Всі основні контракти платформи MVC засновані на інтерфейсі і підлягають тестуванню за допомогою макетів об'єкта, які імітують поведінку реальних об'єктів програми. Додаток можна піддавати модульним тестування без запуску контролерів у процесі ASP.NET, що прискорює тестування і робить його більш гнучким. Для тестування можливе використання будь-якої платформи модульного тестування, сумісної з. NET Framework;

– розширює і доповнює платформа. Компоненти платформи ASP.NET MVC можна легко замінити або налаштувати. Розробник може підключати власний механізм уявлень, політику маршрутизації URL-адрес, сериализацию параметрів методів дій і інші компоненти. Платформа ASP.NET MVC також підтримує використання моделей контейнера впровадження залежності (DI) і інверсії елемента управління (IOC). Модель впровадження залежності дозволяє впроваджувати об'єкти в клас, а не очікувати створення об'єкта самим класом.

Модель інверсії елемента управління вказує на те, що якщо один об'єкт вимагає інший об'єкт, то перші об'єкти повинні отримати другий об'єкт із зовнішнього джерела (наприклад, з файлу конфігурації). Це полегшує тестування;

– розширена підтримка маршрутизації ASP.NET. Цей потужний компонент зіставлення URL-адрес дозволяє створювати додатки з зрозумілими URL-адресами, які можна використовувати в пошуку. URL-адреси не повинні містити розширення імен файлів і призначені для підтримки шаблонів іменування URL-адрес, що забезпечують адресацію, оптимізовану для пошукових систем (SEO) і для передачі репрезентативного стану (REST);

– підтримка використання розмітки в існуючих файлах сторінок ASP.NET (ASPX), елементів управління (ASCX) і головних сторінок (MASTER) як шаблонів уявлень. Разом з платформою ASP.NET MVC можна використовувати існуючі функції ASP.NET, наприклад вкладені головні сторінки, вбудовані вирази (<% =%>), декларативні серверні елементи управління, шаблони, прив'язку даних, локалізацію і т. д.

– підтримка існуючих функцій ASP.NET. ASP.NET MVC дозволяє використовувати такі функції, як перевірка справжності за допомогою форм і Windows, перевірка достовірності за URL-адресою, членство та ролі, кешування виводу і даних, управління станом сеансу і профілю, спостереження за працездатністю, система конфігурації та архітектура постачальника.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Опис об'єктної моделі розробленої системи

На вхід основної програми подається пара звукових wav-файлів, створених у програмному модулі, а також параметри кодування звукових діаграм. Ці дані йдуть у блок створення вихідних звукових діаграм, потім у блок підсумовування звукових діаграм.

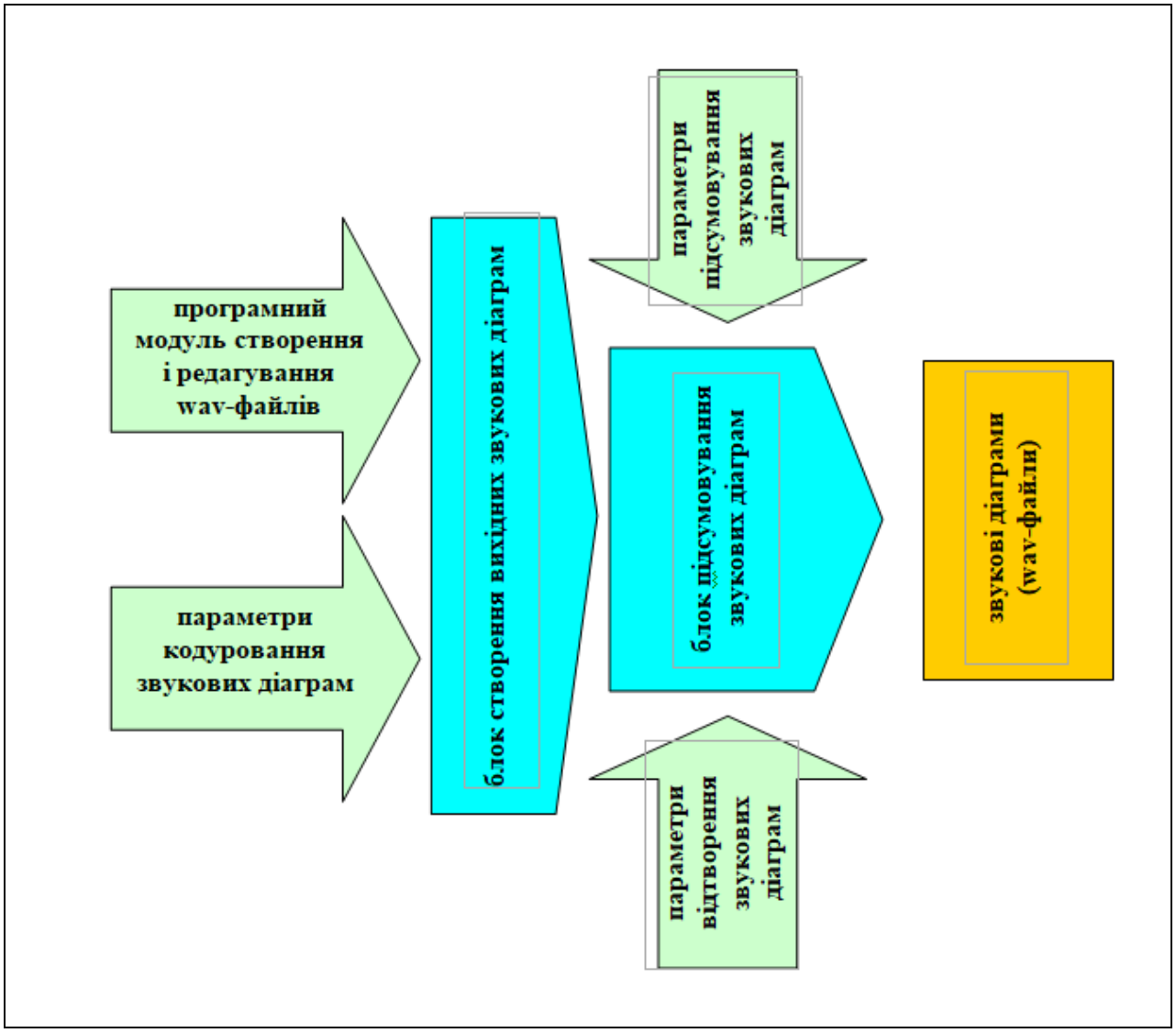

Рисунок 4.1 – Об'єктна модель розробленої системи

На роботу блоку підсумовування звукових діаграм впливають параметри підсумовування звукових діаграм і параметри відтворення звукових діаграм.

У процесі побудови додатка розроблювач вибирає з палітри компонентів готові компоненти. Ще до компіляції він бачить результати своєї роботи – після підключення до джерела даних їх можна бачити відображеними на формі, можна переміщатися за даними, представляти їх у тому або іншому виді. У цьому змісті проектування в середовищах проектування основних мов програмування мало в чому відрізняється від проектування в інтерпретуючім середовищі, однак після виконання компіляції ми одержуємо код, який виконується в 10-20 раз швидше, чим те ж саме, зроблене за допомогою інтерпретатора.

Об'єктно-орієнтована модель програмних компонентів. Основний упор цієї моделі робиться на максимальному ревикористанні коду. Це дозволяє розроблювачам будувати додатка досить швидко із заздалегідь підготовлених об'єктів, а також дає їм можливість створювати свої власні об'єкти для середовища програмування. Ніяких обмежень по типах об'єктів, які можуть створювати розроблювачі, не існує. Дійсно, усе в *C* написане на ньому ж, тому розроблювачі мають доступ до тим же об'єктам і інструментам, які використовувалися для створення середовища розробки.

4.2 Структура й основні функції програмного модуля Wave_Edit

Основною програмою проекту є Digsound, однак для її роботи необхідні спеціальні звукові файли у форматі .wav. Для створення таких файлів був розроблений програмний модуль Wave_Edit (рис. 4.2).

У відповідності зі стандартом основні характеристики звукового файлу описуються типом `waveformatex`:

```
Type
  wformattag: Integer;
  nchannels: Integer;
  nsamplespersec: Long;
  navgbytespersec: Long
  nblockalign: Integer
```

```
  wbitspersample: Integer
End;
```

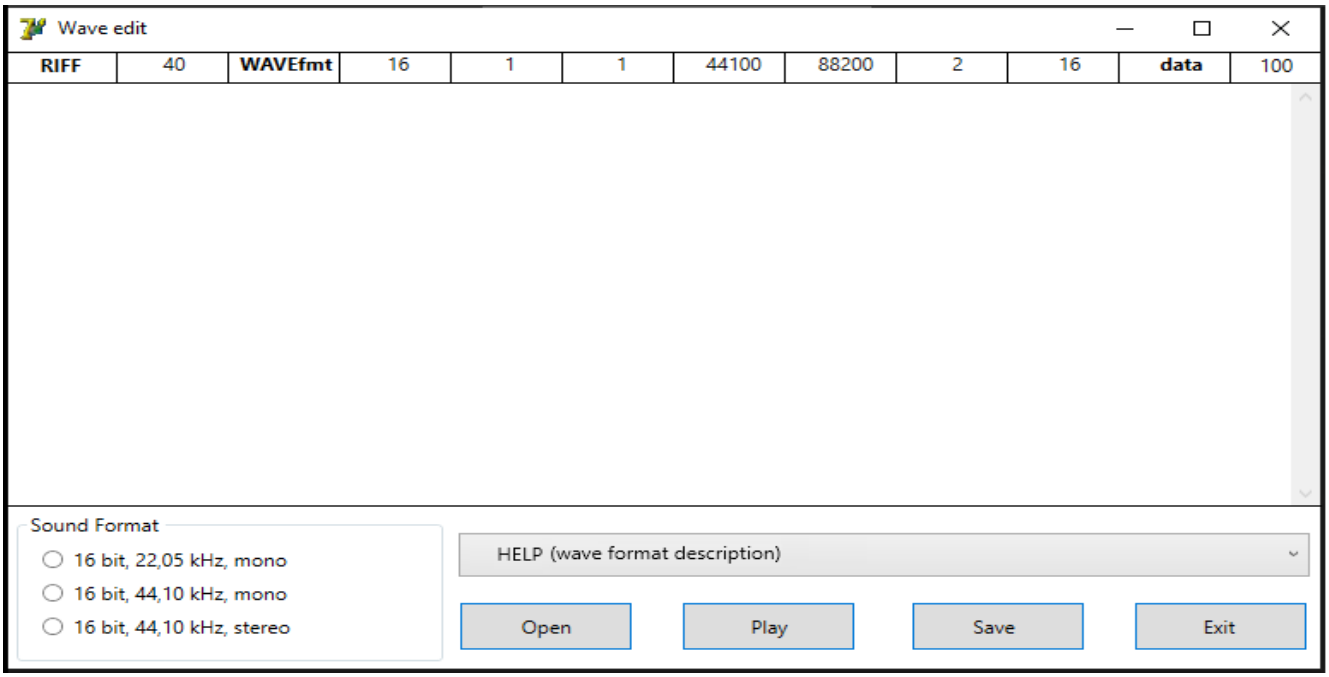

Рисунок 4.2 – Програмний модуль Wave_Edit

У програмному модулі Wave_Edit використовується 16-бітне кодування при тактовій частоті 44100 Гц:

```
fmt_44_16.wformattag := 1
fmt_44_16.nchannels := 1
fmt_44_16.wbitspersample := 16;
fmt_44_16.nblockalign := 1;
fmt_44_16.nsamplespersec := 44100;
fmt_44_16.navgbytespersec := 44100;
```

У верхньому рядку текстових полів вікна модуля задається повний опис формату даних звукового файлу:

```
Namefmt
Size_RIFF
Typefmt
Size_wfx
fmt_44_16.wformattag
fmt_44_16.nchannels
fmt_44_16.nsamplespersec
fmt_44_16.navgbytespersec
fmt_44_16.nblockalign
```

```
fmt_44_16.wbitspersample
Datafmt
Size_data
```

Інтерфейс програмного модуля Wave_Edit дозволяє безпосередньо задавати як заголовок формату даних звукового файлу, так і самі дані в десятковому форматі. Докладний опис, що таке RIFF-формат wav-файлу.

У файлах Riff-формату (Resource Interchange File Format формат файлу для обміну ресурсами) зберігаються, що мають відношення до мультимедіа дані (звук, відео й ін.). Як файли, що містять звук wav-файли, так і avi-файли, що містять відеоінформацію, мають формат RIFF. Файл формату RIFF містить вкладені фрагменти (chunk's); зовнішній фрагмент складається із заголовка й області даних.

Перше подвійне слово заголовка містить чотирибуквений код FOURCC, що ідентифікує дані, що зберігаються у фрагменті. Друге подвійне слово заголовка являє собою розмір області даних у байтах (без обліку розміру самого заголовка).

Область даних має змінну довжину, однак вона повинна бути вирівняна на границю слова (при необхідності доповнюється наприкінці нульовим байтом до цілого числа слів).

Формат RIFF не описує конкретний формат даних; практично файл в RiffpФорматі може містити будь-які мультимедіа-дані, причому формат конкретних даних залежить від типу цих даних (RIFF є скоріше стандартом опису контейнера даних).

Позначена як «Дані» область може містити усередині себе інші фрагменти. Для утримуючого звукові дані файлу (wav-файл) ця область містить ідентифікатор даних ‘WAVE’, фрагмент формату звукових даних ‘fmt’ (три символи ‘fmt’ і пробіл наприкінці), а також фрагмент звукових даних.

Файл може додатково містити фрагменти інших типів, тому не слід припускати, що заголовок wav-файлу має фіксований формат. Наприклад, у форматі можуть бути присутнім фрагменти ‘LIST’ або ‘ABOUT’, що містять інформацію про права копіювання й опис самого мультимедиа-файла.

Область «Формат даних» описує звукові дані. Формат цієї області для файлів PCM (записаних з використанням імпульсно-кодової модуляції) відповідають структурі PCMWAVEFORMAT, певної у файлі mmsystem.h у такий спосіб

```
typedef struct pcmwaveformat_tag
{WAVEFORMAT wf;
WORD wbitspersample;
} PCMWAVEFORMAT;
typedef PCMWAVEFORMAT *PPCMWAVEFORMAT;
typedef PCMWAVEFORMAT NEAR *NPPCMWAVEFORMAT;
typedef PCMWAVEFORMAT FAR *LPPCMWAVEFORMAT;
```

Структура WAVEFORMAT описана у файлі mmsystem.h у такий спосіб

```
typedef struct waveformat_tag
{WORD wformat Tag; // тип формату
WORD nchannels; // кількість каналів (моно або стерео)
DWORD nsanplespersec; // частота дискретизації
DWORD navgbytespersec; // швидкість потоку даних
WORD nblockalign; // вирівнювання блоку даних } WAVEFORMAT;
typedef WAVEFORMAT *PWAVEFORMAT;
typedef WAVEFORMAT NEAR *NPWAVEFORMAT;
typedef WAVEFORMAT FAR *LPWAVEFORMAT;
```

Поле wformattag описує тип формату звукових даних (для підтримуваного стандартною бібліотекою mmsystem.dll методу імпульсно-кодової модуляції PCM у цьому полі повинне перебувати зарезервоване у файлі mmsystem.h значення WAVE_FORMAT_PCM) `#define WAVE_FORMAT_PCM 1`

Поле nchannels містить кількість каналів, у ньому можуть перебувати значення 1 (моно) або 2 (стерео).

У поле nsamplespersec записується частота дискретизації (кількість вибірок сигналу в секунду). У цьому полі можуть перебувати стандартні (11025 кГц, 22050 кГц або 44100 кГц) або нестандартні значення (такі як 5000 кГц або 4400 кГц), однак не всі драйвери звукових адаптерів можуть коректно працювати з нестандартними частотами дискретизації.

Поле navgbytepersec містить середню швидкість потоку даних (кількість байт у секунду, переданих драйверу обладнання або одержуваних від нього). Ця інформація може використовуватися додатком для оцінки розміру

необхідного для розміщення звукових даних буфери (наприклад, для монофонічного сигналу з дискретизацією 8 біт значення швидкості чисельно збігається зі значенням частоти дискретизації, для стереофонічного сигналу з дискретністю 8 біт вона вдвічі вище). Точне значення величини navgbytepersec розраховується по формулі

navgbytepersec=(nchannel * nsamples * wbitsperample) / 8.

У поле nblockalign перебуває інформація про вирівнювання блоку в байтах, причому

nblockalign=(nchannels * wbitspersample) / 8.

Поле `wbitpersample` визначає дискретність сигналу (кількість біт, використовуване для однієї вибірки сигналу); звичайно використовуються значення 8 або 16.

Формат самих звукових даних залежить від кількості каналів і від дискретності.

Для монофонічного сигналу з дискретністю 8 біт звукові дані являють собою масив однобайтовых значень, кожне з яких є вибіркою сигналу.

Для стереофонічного сигналу з дискретністю 8 біт звукові дані мають формат масиву двухбайтовых слів, причому молодший байт слова відповідає лівому каналу, а старший – правому.

Формат звукових даних з дискретністю 16 біт виглядає аналогічно (для монофонічного сигналу дані зберігаються в масиві 16вбітових слів; для стереофонічного використовується масив подвійних слів, причому молодшому слову відповідає лівий канал, а старшому – правий).

Діапазон зміни значень вибірок визначається дискретизацією. Для 8ыбітових даних діапазон становить від 0 до 255 (0xff), причому відсутності сигналу (повна тиша) відповідає значення 128 (0x80); для 16вбітових значень

діапазон зміни становить від –32768 (-0x8000) до 32767 (0x7fff), відсутності сигналу відповідає значення 0.

Розглянемо тепер, які функції реалізовані в програмному модулі Wave_Edit. По кнопці «Open» можна відкрити існуючий звуковий файл, при цьому у верхньому рядку текстових полів вікна модуля відображається формат цього звукового файлу, а нижче цього рядка в багаторядковім текстовім полі відображається сам код звуку в десятковому форматі. При необхідності код звуку можна відредагувати або замінити іншим. Формат цього звукового файлу також можна відредагувати.

Кнопка «Save» призначена для збереження нового або відредагованого звукового файлу.

Кнопка «Play» призначена для відтворення звукового файлу. Відтворення забезпечується використанням функції Windows sndplaysound:

```
Private Declare Function sndplaysound Lib "winmm.dll" Alias "sndplaysounda" (Byval lpszsoundname: String, Byval uflags: Long): Long;
```

У програмному модулі Wave_Edit передбачено 3 варіанта 16-бітного формату звукового файлу:

– моно, частота дискретизації 22050 Гц;

– моно, частота дискретизації 44100 Гц;

– стерео, частота дискретизації 44100 Гц.

4.3 Структура й основні функції програми Testsound

Програма Testsound призначена для тестування методу частотно-імпульсного кодування звуку. Вона забезпечує керування всіма необхідними параметрами звукових файлів (рис. 4.3).

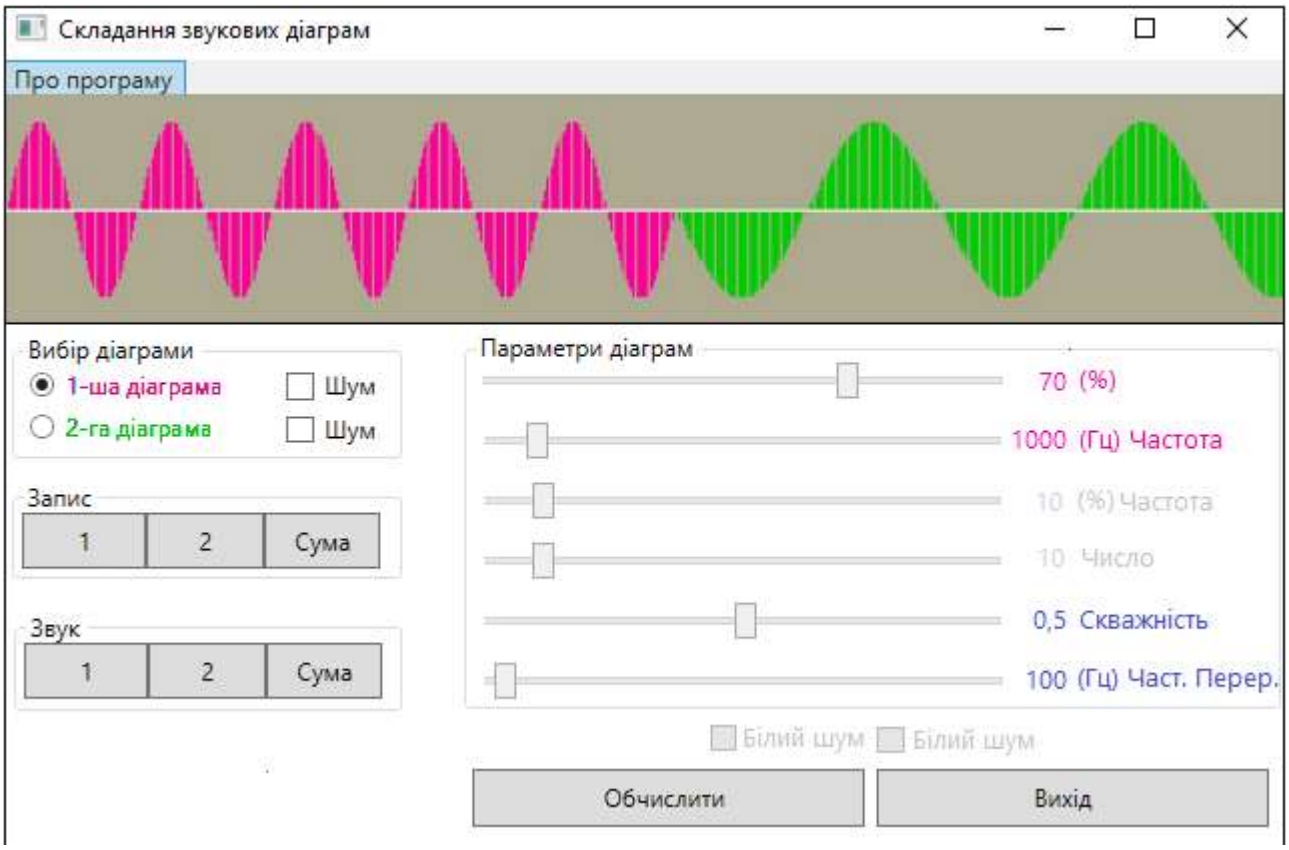

Рисунок 4.3 – Основне вікно програми Testsound

Суть роботи програми полягає в додаванні двох звукових діаграм із заданими параметрами. У якості звукових діаграм використовуються синусоїдальні й шумові сигнали. При цьому шум може бути синусоїдальним – з випадково мінливою частотою, і білим. Для кожної звукової діаграми задається її амплітуда, частота, шпаруватість і частота переривання. Для шумових сигналів задається також тип шуму (синусоїдальний або білий), величина припустимого відхилення частоти й число можливих частот.

Кожну звукову діаграму можна створити із заданими параметрами, записати й прослухати окремо. Але кінцевою метою програми є озвучування суми обох звукових діаграм.

Докладний опис параметрів звукових діаграм. Амплітуда задається у відсотках від максимального припустимого значення, яким є 16-річне число FFFFFF. Частота задається в герцах і визначає для синусоїдальних діаграм число пара півхвиль у секунду. Для шумових синусоїдальних сигналів задається також

припустимий розкид частот у відсотках від основної частоти.

Два параметри, що залишилися ставляться до сумарної звукової діаграми. Сумарна діаграма складається з ділянок вихідних діаграм, вирізаних з них відповідно до параметрів шпаруватості й частоти переривання. Шпаруватість визначає співвідношення ширини ділянок, що вирізьблюються з вихідних діаграм, у частках від одиниці. Тобто, якщо шпаруватість першої вихідної діаграми 0,3, то шпаруватість другої вихідної діаграми буде дорівнювати 0,7. Таким чином, сума пари ділянок вихідних діаграм буде рівна 1. Тривалість цієї одиничної суми задається другим параметром сумарної звукової діаграми – частотою переривання. Наприклад, при частоті переривання 100 Гц і шпаруватості першої звукової діаграми 0,4 тривалість ділянок, що вирізьблюються з першої звукової діаграми, буде рівнятися 4 мс, а із другої звукової діаграми – 6 мс. При цьому повна тривалість усіх трьох звукових діаграм буде однакової й рівнятися 5 с.

По кнопці «Обчислити» обчислюється звукова діаграма із заданими параметрами. По кнопках «1» і «2», «Сума» на панелі «Запис» зберігаються у файлі відповідно перша, друга або сумарна звукова діаграма. По кнопках «1» і «2», «Сума» на панелі «Звук» відповідно відтворюються перша, друга або сумарна звукова діаграма.

Елементи керування у вікні програми, що ставляться тільки до першої звукової діаграми, виділені червоним цвітом, до другої звукової діаграми – зеленим цвітом. Елементи керування, що ставляться до параметрів синусоїдальних діаграм, виділені коричневим цвітом, синусоїдальних шумових діаграм – фіолетовим цвітом, сумарної звукової діаграми – синім кольором.

За допомогою програми Testsound були поставлені й проведені звукові експерименти. Розроблена програма дозволила поставити експерименти із частотно-імпульсним способом модуляції звукових сигналів, що дозволяють визначити параметри модуляції. При цьому в якості вихідних сигналів застосовувалися не аналогові сигнали, а дискретні коди зі стандартними параметрами: 44,1 кГц, 16 біт, моно.

Асинхронний частотно-імпульсний код звуку має ідеальне, тобто таке, що

не відрізняється на слух від оригіналу, звучання при досягненні критичної частоти $\omega_{кр}$. Однак зберегти асинхронний частотно-імпульсний код звуку на дискретному носії не можна, тому що інформаційні імпульси такого коду виробляються в довільні моменти часу.

Для рішення цього завдання, тобто запису на дискретному носії асинхронного частотно-імпульсного код звуку необхідно його синхронізувати з якоюсь тактовою частотою. Тоді можна буде, зміщаючи імпульси асинхронного коду звуку з найближчому наступному такту, одержати код, синхронізований з якоїсь тактовою частотою. Чим більше тактова частота, тем менше будуть зсуву імпульсів асинхронного коду до тактових імпульсів. При цьому для забезпечення чистого звучання помилка зсуву не повинна перевищувати 2%.

З іншого боку, щоб не були чутні окремі імпульси частотно-імпульсного код звуку, їх частота повинна бути не менш ніж 20 кГц – верхньої границі стандартного звукового діапазону, що чує людина та відтворює звукова апаратура.

Із цього випливає, що мінімальна тактова частота для синхронізації асинхронного частотно-імпульсного коду звуку повинна бути не менш, ніж

$$50*20 \text{ кГц} = 1 \text{ Мгц}. \qquad (4.1)$$

Оскільки програма Testsound використовує АЦП-код із частотою дискретизації 44,1 кГц, яка менше за 1 МГц, то довільні аналогові сигнали після їхнього частотно-імпульсного перетворення й наступної синхронізації не будуть точно збігатися по звучанню з їхнім дискретним варіантом. Однак експерименти, поставлені за допомогою розробленої програми, дозволили без апаратної реалізації визначити деякі важливі параметри синхронізованого частотно-імпульсного коду звуку. Зокрема, були знайдені параметри шпаруватості й частоти переривання при підсумовуванні двох звукових діаграм, проаналізований вплив шумових компонентів різної амплітуди на сумарний звуковий сигнал.

4.4 Програмна реалізація основних алгоритмів

Програма Digsound складається із двох типів процедур – процедур обробки подій натискання кнопок і процедур обробки подій на елементах керування параметрами звукових діаграм.

Для прикладу реалізацію кнопки «1» на панелі «Запис» процедурою Sav1_Click:

```
Procedure Tform1.Sav1_Click(Sender: Tobject);
  begin
Size_data := 100000;
//Установка формату записи файлу
 Sndfile := Freefile
 Open App.Path & "\sound1.wav" For Binary As Sndfile;
  Put Sndfile,, "RIFF";
  Put Sndfile,, Size_RIFF
  Put Sndfile,, "Wavefmt";
  Put Sndfile,, Size_wfx;
  Put Sndfile,, fmt_44_16;
  Put Sndfile,, "data";
  Put Sndfile,, Size_data;
//Запис 1-й діаграми
 Progressbar1.Value := 0;
 Progressbar1.Visible := True;
  For k := 0 To Size_data \ quant1 do begin
    Calc_1;
    For i := 1 To quant1 do begin
      Put Sndfile,, Sound_1(i);
    end;
    Progressbar1.Value := k * 100 \ (Size_data \ quant1);
  end;
Close Sndfile;
Progressbar1.Value := 0;
Progressbar1.Visible := False;
End; //Sav1_Click
```

У розглянутій процедурі викликається інша процедура – Calc_1:

```
Procedure Tform1.Calc_1(Sender: Tobject);
begin
  If (Noise1 = True) And (Wnoise1 = False) Then do begin
```

```
      Fricmin := Fricaver1 – Fricaver1 * (Fricrange1 / 200);
      For i := 1 To Numharm1 do begin
        Fric(i) := Fricmin + Rnd * Fricaver1 * (Fricrange1 / 100);
        Pd(i) := Diskret \ Fric(i);
      end;
      For takt := 1 To quant1 do begin
        Sumharm := 0;
        For i := 1 To Numharm1 do begin
          Sumharm := Sumharm + Int(Sin(Dg(i)) * amp1 * 320 /
Numharm1);
          Dg(i) := Dg(i) + 2 * Pi / Pd(i);
        end;
        Sound_1(takt) := Sumharm;
        Deg2 := Deg2 + 2 * Pi / prd2;
      end;
      end;
      If (Noise1 = True) And (Wnoise1 = True) Then do begin
        For takt := 1 To quant1;
          Sound_1(takt) := Int((Rnd * 2 – 1) * amp1 * 320);
          Deg2 := Deg2 + 2 * Pi / prd2;
        end;
      end;
      If Noise1 := False Then do begin
        For takt := 1 To quant1 do begin
          Sound_1(takt) := Int(Sin(Deg1) * amp1 * 320);
          Deg1 := Deg1 + 2 * Pi / prd1;
          Deg2 := Deg2 + 2 * Pi / prd2;
        end;
      end;
    end; //Calc_1
```

У цій процедурі обробляються три можливі варіанти звукових діаграм, відповідно – синусоїдальних шумових діаграм, діаграм білого шуму й синусоїдальних діаграм.

Реалізація слайдера, елемента керування параметром частоти переривання процедурою Slider6_Scroll:

```
    Procedure Tform1.Slider6_Scroll(Sender: Tobject);
    begin
       Deg1 := 0;
       Deg2 := 0;
       Frprer := Diskret \ (Diskret \ Slider6.Value);
       Lbl6.Caption = Frprer;
       quant1 = Round(Diskret * Sqv1 / Frprer);
       quant2 = Round(Diskret * Sqv2 / Frprer);
       Calc_1;
       Calc_2;
       Drowdiagr;
    End;
```

Виведення у вікні програми графічного представлення звукових діаграм здійснюється процедурою Drowdiagr:

```
Procedure Tform1.Drowdiagr(Sender: Tobject);
begin
  Pct1.Cls;
  For k := 0 To 460 Step quant1 + quant2 do begin
    For i := 1 + k To quant1 + k do begin
      Pct1.Line (i, 44)-(i, 44 - Sound_1(i - k) \ 650),
RGB(255, 0, 150)
    End;
    For i := quant1 + 1 + k To quant1 + quant2 + k do
      Pct1.Line (i, 44)-(i, 44-sound_2(i-k-quant1)\650), RGB(0,
200, 0);
    Calc_1;
    Calc_2;
  End;
Pct1.Line (0, 44)-(460, 44), RGB(255, 255, 255);
End; //Drowdiagr
```

Озвучування першої звукової діаграми здійснюється процедурою Snd1_Click:

```
Procedure Tform1.Snd1_Click(Sender: Tobject);
begin
  R := sndplaysound(App.Path & "\sound1.wav", SND_SYNC);
End;
```

Відтворення реалізоване функцією Windows sndplaysound.

5 ОПИС МОЖЛИВОСТІ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Система частотно-імпульсного кодування звуку ґрунтується на багаторічних дослідженнях (теоретичних і експериментальних) механізму слухового аналізатора людини. Був відкритий ефект згладжування слухових відчуттів і сформульовані основні властивості процесу такого згладжування. Істинність ефекту згладжування перевірялася за допомогою спеціальних психо-акустичних експериментів. Випробуваному в процесі цих експериментів мають бути пред'явлені звуки, що перериваються з високою частотою, а також звуки, що закодовані синхронізованої частотно-імпульсною модуляцією. Було встановлено, що при певних режимах переривання звуку й певних параметрах імпульсній-частотно-імпульсної модуляції звуку, слухове сприйняття кодів звуку не відрізняється від сприйняття вихідних звуків. Ці експерименти були покладені в основу розробки комп'ютерної системи частотно-імпульсного кодування й декодування звуку, а також в основу створення програми системою управління, кодування, декодування, зберігання і обробки частотно-імпульсних кодів звуків.

Система частотно-імпульсного кодування звуку призначена для застосування при введенні інформації про звук в комп’ютерній техніці, для машинного зберігання, запам'ятовування й відтворення звуку. Вона призначає також для будь-який аудіо- і відеотехніки. Зокрема, застосуванням частотно-імпульсних кодів звуків можна досягти істотного вдосконалення якості цифрових магнітофонів, що виражається в поліпшенні якості звучання, істотнім збільшенні тривалості звучання при тих же дисках, спрощенні електронної схеми цифрового магнітофона.

Якісна відмінність отриманих результатів характеризується тим, що багаторозрядний код звуку перетворюється в однорозрядний двійковий сигнал (одиночний імпульс), а постійний інтервал, з яким випливають багаторозрядні коди звуку, замінюється змінним інтервалом. Уся інформація про код звуку тепер міститься в тривалості інтервалу між сусідніми імпульсами в коді звуку. В ході

програмних експериментів було встановлено, що натуральне звучання коду звуку досягається, якщо цей інтервал спотворюється не більше ніж на 2%. Ця норма покладена в основу частотно-імпульсних систем кодування й декодування звуку.

ВИСНОВКИ

Система частотно-імпульсного кодування звуку ґрунтується на багаторічних дослідженнях (теоретичних і експериментальних) механізму слухового аналізатора людини, які проводилися на кафедрі ПІ. Був відкритий ефект згладжування слухових відчуттів і сформульовані основні властивості процесу такого згладжування. Істинність ефекту згладжування перевірялася за допомогою спеціальних психо-акустичних експериментів. Випробуваному в процесі цих експериментів пред'являлися звуки, що перериваються з надзвуковою частотою, а також звуки, закодовані синхронізованої частотно-імпульсною модуляцією. Було встановлено, що при певних режимах переривання звуку й певних параметрах частотно-імпульсної модуляції звуку, слухове сприйняття кодів звуку неотличимо від сприйняття вихідних звуків. Ці експерименти були покладені в основу розробки комп'ютерної системи частотно-імпульсного кодування й декодування звуку, а також в основу створення програми системою, що управляє, кодування, декодування, зберігання й обробки частотно-імпульсних кодів звуків.

Є перспективи для подальшого вдосконалювання системи частотно-імпульсного кодування звуку. До них ставляться пошуки ще більш ощадливих способів згортання й розгортання кодів звуків, дослідження питання про оптимальне число градацій тривалості інтервалу між сусідніми імпульсами в коді звуку й про зв'язок цього числа з рівнем зашумленості кодуємого сигналу, а також з можливою шириною відтвореного динамічного діапазону звучання сигналу. Також є більші можливості для розробки різних режимів обробки частотно-імпульсних кодів звуків за допомогою цифрових диференціальних аналізаторів послідовної й паралельної дії, що особливо важливо для систем автоматичного розпізнавання злитої мови.

Були отримані наступні результати:

– сформульований посилений узагальнений закон Талбота, на основі якого введено поняття критичної частоти проходження імпульсів у звуковому сигналі;

– розроблений метод формування асинхронних частотно-імпульсних кодів довільних звуків, невідрізнимих на слух від вихідних звуків;

– виконані експерименти по визначенню критичної частоти проходження імпульсів в асинхронному коді довільного звуку;

– встановлено, що, якщо інтервали між сусідніми імпульсами не перевищують 50 мкс, то асинхронні коди будь-яких звуків неотличимы на слух від вихідних звуків;

– сформульований закон існування порога в слуху, за допомогою якого введено поняття критичної частоти синхронізації звукових сигналів;

– виконані програмні експерименти по визначенню критичної частоти синхронізації для довільних звукових сигналів;

– установлено, що критична частота синхронізації залежить тільки від величини мінімального інтервалу між імпульсами в асинхронному коді звуку, причому залежність ця лінійна.

Виходячи з перерахованих вище результатів, можна сказати, що розроблені алгоритми можуть бути використані при дискретизації акустичних коливань і пояснюють механізми роботи слухового аналізатора людини.

Є перспективи для подальшого вдосконалювання системи частотно-імпульсного кодування звуку. До них ставляться пошуки ще більш ощадливих способів згортання й розгортання кодів звуків, дослідження питання про оптимальне число градацій тривалості інтервалу між сусідніми імпульсами в коді звуку й про зв'язок цього числа з рівнем зашумленості кодуємого сигналу, а також з можливою шириною відтвореного динамічного діапазону звучання сигналу. Також є можливості для розробки різних режимів обробки частотно-імпульсних кодів звуків за допомогою цифрових диференціальних аналізаторів послідовної й паралельної дії, що особливо важливо для систем автоматичного розпізнавання злитої мови.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Магдануров Г., Юнев, В. ASP.NET MVC Framework; БХВ–Петербург – 2015. – 320 с.

2. Фримен, А. ASP.NET MVC 4 с примерами на C# для профессионалов; Вильямс –, 2013. – 688 с.

3. Чедвик, Дж. , Снайдер Т., Панда Х. ASP.NET MVC БХВ–Петербург –, 2015. – 320 с.

4. Freeman, A. Pro ASP.NET MVC 4; Apress – , 2013. – 756 с.

5. Evjen, B., Nagel, C. .NET 2.0 Wrox Box: Professional ASP.NET 4.5, Professional C# Professional .NET Generics and Professional .NET Framework; Wrox –, 2015. – 709 с.

6. Sutton, C. Programming ASP. NET MVC;, 2008. – 350 с.

7. Esposito, D. Building Web Solutions with ASP.Net and ADO .Net;, 2014. – 416 с.

8. Gefen, D., Govindarajulu, C. Advanced Visual Basic.NET: Programming Web and Desktop Applications in ADO.NET and ASP.NET; – 2012. – 595 с.

9. Emad, I. ASP.NET MVC Test Driven Development; – 2009. – 312 с.

10. McManus, J., Kinsman, C. Visual Basic(R) .NET Developer's Guide to ASP .NET, XML and ADO.NET; 2014. – 622 с.

11. McManus, J. P., Kinsman, C. C# Developer's Guide to ASP.NET, XML, and ADO.NET; Addison–Wesley Professional –, 2019. – 608 с.

12. Palermo, J., Scheirman, B. ASP.NET MVC in Action;, 2018. – 275 с.

13. Alexander, J., Developing Web Applications with Visual Basic.NET and ASP.NET - 2012. - 400 с.

14. Chikina V., Dudar Z., Shabanov-Kushnarenko S. The modeling of some intelligence functions // Radio electronics and computer science. - Kharkiv: KNURE, 2003. – № 3. – pp. 166-172.

15. The Methods of Adaptation in Computer-Based Training Systems / Shubin I., Umiarov K.// Proceeding of 2015 Information Technologies in Information Business Conference (ITIB) 7 – 9 October, 2015, IEEE Catalog Number CFP15D13-PRT pp. 64-67.

16. The Interaction Domain// World Wide Web Consortium, - November 2003. http://www.w3.org/Interaction/.

17. M.F. Bondarenko, Z.V. Dudar, N.T. Protsay, V.V. Cherkashyn, V.A. Chykyna, Y.P. Shabanov-Kushnarenko, “Algebra of predicates and predicate operations” Radio electronics and informatics, no. 1, 2004, pp. 5-22.

18. G.G. Chetverikov, I.D. Vechirska, S.S.Tanyanskiy, “The methods of algebra finite predicates in the intellectual system of complex calculations of telecommunication companies,” International Conference Proceedings Crimean Microwave and Telecommunication Technology (CriMiCo), 6959425, 2014, pp.

19. DTD for XML Schema // World Wide Web Consortium, - August 2002. http://www.w3.org/TR/xmlschema-1/#nonnormative-schemadtd

20. XSL Transformations (XSLT), W3C Recommendation // World Wide Web Consortium, - November 1999. http://www.w3.org/TR/1999/ Rec-xslt-19991116.

21. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation // World Wide Web Consortium, - February 1999. http://www.w3.org/TR/2019/Rec-rdf-syntax-19990222/.

22. Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation // World Wide Web Consortium, - March 2020. http://www.w3.org/TR/2020/Cr-rdf-schema-20000327/.

23. OWL Web Ontology Language // World Wide Web Consortium, - September 2004. http://www.w3.org/2014/OWL/.

24. OWL Web Ontology Language Reference, W3C Recommendation // World Wide Web Consortium, - February 2014. http://www.w3c.org /TR/owl-ref/.

25. Web Ontology // World Wide Web Consortium, - March 2019. http://www.w3.org/Help/siteindex.html#webont.

26. DAML+OIL Reference Description // World Wide Web Consortium, - March 2020. http://www.w3.org/TR/daml+oil-reference

27. SWRL: A Semantic Web Rule Language Combining OWL and Ruleml // World Wide Web Consortium, - November 2003. http://www.ruleml.org.

28. Integration of information Systems: Bridging Heterogeneous Databases. / Editing by Amar Gupta, Sloan School of Management, Massachusetts Institute of Technology.-1996.

29. G. Wiederhold. Mediators in the architecture of future information systems// IEEE Computer, -March 2016. - p. 38-49.

30. Gruber T. Towards principles for the design of ontologies used for knowledge sharing// International Journal of Human-Computer Studies, 43: 907-928,- 2014.

31. N. Guarino. Some ontological principles for designing upper level lexical resources. In Proceedings of the First International Conference on Language Resources and Evaluation, Granada, 2008. pp. 3–15