Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20-3СК
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Буза В.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Бузі Віталію Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 8-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Коваленко А.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Буза В.В.
(прізвище та ініціали)

АНОТАЦІЯ

Буза В.В. Програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи дослідження протоколів IP-телефонії хмарної банківської мережі.

Метою розробки є програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі.

Результат роботи – програмна реалізація системи дослідження протоколів IP-телефонії хмарної банківської мережі.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, IP-телефонія, хмарна банківська мережа

ABSTRACT

Buza V.V. Software of the research system of IR-telephony protocols of the cloud banking network. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software was developed, which is intended for the research system of IP telephony protocols of the cloud banking network.

The goal of the development is the software of the system of research of IR-telephony protocols of the cloud banking network.

The result of the work is the software implementation of the system for researching IR-telephony protocols of the cloud banking network.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the RAD Studio Delphi 10.4 environment.

Keywords: computer engineering, IP telephony, cloud banking network

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АТМ	–	автоматична телефонна мережа
БД	–	база даних
БЗ	–	база знань
КОМ	–	банківські обчислювальні мережі
ЛОМ	–	локальні обчислювальні мережі
МА	–	мережне адміністрування
ПЗ	–	програмне забезпечення
СМ	–	система моніторингу
СУБД	–	система управління базою даних
ТфОП	–	телефонний оператор
CA	–	Call Agent, контролер сигналізацій
IP	–	Internet Protokol
MIB	–	база даних інформації керування
MG	–	транспортний шлюз
MGC	–	контролером шлюзу
MGCP	–	протокол керування шлюзами
RSVP	–	Resource Reservation Protocol
RTP	–	транспортний протокол реального часу
RTSP	–	протокол передачі потокової інформації в реальному часі
SDP	–	протокол опису параметрів зв'язку
SIP	–	протокол ініціювання сеансів
SNMP	–	простий протокол керування мережею
SS	–	програмний контролер
WINS	–	служба зіставлення netbios-імен комп'ютерів з IP-адресами вузлів
VoIP	–	Voice over IP

ВСТУП

Актуальність теми. IP-телефонія (телефонія Інтернет-протоколу) – це загальний термін для технологій, продуктів і послуг, які використовують з'єднання з комутацією пакетів Інтернет-протоколу для підтримки голосових викликів, голосової пошти, відеодзвінків, відеоконференцій, факсів і миттєвих повідомлень (IM).

Традиційно цей зв'язок здійснювався через виділені з'єднання з комутацією каналів комутованої телефонної мережі загального користування (PSTN). Під час використання Інтернету дзвінки передаються як пакети даних по спільним лініям, уникаючи плати за телефонну мережу загального користування.

IP-телефонія працює шляхом перетворення голосових дзвінків, факсів та іншої інформації в цифрові сигнали. Ці цифрові сигнали передаються через IP-мережі, такі як Інтернет, у вигляді пакетів даних за допомогою IP-з'єднань з комутацією пакетів. Голосові функції в технології IP, такі як голосові дзвінки та голосова пошта, називаються голосом через IP (VoIP).

IP-телефонія була представлена в 1991 році зі створенням першої програми VoIP Speak Freely. Наступного року InSoft випустила Communique для настільних відеоконференцій. У 1995 році Intel, Microsoft і Radvision почали стандартизувати системи VoIP.

Окрім голосових викликів, IP-телефонія також включає інші функції, наприклад:

– відеодзвінок;

– відеоконференція;

– IM;

– текстове повідомлення;

– факс.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем дослідження протоколів IP-телефонії хмарної банківської мережі.

– Дослідження системи дослідження протоколів IP-телефонії хмарної банківської мережі.

– Програмна реалізація системи дослідження протоколів IP-телефонії хмарної банківської мережі.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі дослідження протоколів IP-телефонії хмарної банківської мережі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Термін *IP-телефонія* часто використовується як синонім *VoIP*. Ці два терміни збігаються, і не вважається неправильним використовувати їх як синоніми, оскільки вони обидва використовують локальну мережу для підключення до Інтернету через маршрутизатор. Проте, визначено в найбільш формальних і конкретних термінах:

– VoIP є більш конкретним і описує стандартизовані технології, які забезпечують голосові функції, такі як дзвінки та голосова пошта, через мережі на основі IP; і

– З іншого боку, IP-телефонія є більш загальним терміном, який охоплює повний набір технологій, функцій і продуктів, включаючи голосові функції, а також не голосові функції, такі як миттєві повідомлення.

Які переваги бізнесу IP-телефонії?

Послуги IP-телефонії можуть запропонувати такі переваги організаціям будь-якого розміру:

– **Не потрібно підтримувати дві мережі.** До IP-телефонії компаніям потрібно було розгортати та підтримувати окремі мережі для Інтернету та телефону.

– **Менші витрати.** Використання IP-телефонії значно економічніше, ніж використання дротових телефонних послуг у аналогічному масштабі. Міжміські та міжнародні дзвінки також дешевші за допомогою IP-телефонії.

– **Налагоджена інфраструктура.** Системи IP-телефонії можуть інтегруватися в існуючу інфраструктуру, дозволяючи комунікації працювати через бізнес-додатки. Наприклад, факси можна надсилати й отримувати електронною поштою.

– **Масштабованість.** Розгортання нових бізнес-телефонних ліній IP, яке можна здійснити через онлайн-інтерфейс, набагато простіше та легше, ніж додавання нових проводових телефонних ліній.

– **Мобільність.** Доступ до IP-телефонів можна отримати за межами фізичного офісу, з портативних комп'ютерів і мобільних пристроїв.

Які проблеми IP-телефонії?

Незважаючи на переваги, пов'язані з використанням IP-телефонії, її використання може нести певні проблеми:

– **Проблеми з викликами 911.** Незважаючи на вимогу Федеральної комісії зі зв'язку, щоб послуги VoIP пропонували номер 911 як стандартну функцію, користувачі, що дзвонять, повинні зареєструвати свою фізичну адресу у свого провайдера. Якщо вони нехтують цим, дзвінки VoIP 911 можуть не надати належне місцезнаходження та номер телефону абонентів.

– **Затримка сервера.** Послуги потребують високошвидкісного підключення до Інтернету. Перебої в потужності інтернет-з'єднання можуть призвести до затримки сервера.

– **Вимоги до живлення**. Системи IP-телефонії, підключені до офісних блоків живлення, не працюватимуть під час відключення електроенергії.

Хто керує IP-телефонією?

IP-телефонією керують постачальники послуг VoIP та уніфікованих комунікацій (UC). Серед відомих постачальників послуг IP-телефонії:

– Avaya.
– Microsoft.
– RingCentral.
– Skype.
– Zoom.

1.2 Область застосування

Областю застосування розроблювальної системи є хмарна банківська мережа. Хмарні обчислення виходять на передній план як у центрі уваги IT-лідерів, керівників і членів правління.

Використання хмари для створення нових меж бізнесу

Банк 2030 року виглядатиме зовсім інакше, ніж сьогодні. Зіткнувшись зі зміною очікувань споживачів, новими технологіями та альтернативними бізнес-моделями, банки повинні почати впроваджувати стратегії вже зараз, щоб допомогти їм підготуватися до цього майбутнього. Важливий показник мінливого ландшафту? Хмарні обчислення виходять на передній план як у центрі уваги головного інформаційного директора, керівників і членів правління.

Лідери банківської сфери та ринків капіталу все більше визнають, що хмара – це більше, ніж технологія; це місце для банків та інших компаній, що надають фінансові послуги, де вони можуть зберігати дані та програми, а також отримувати доступ до передових програмних програм через Інтернет.

Провідні публічні хмарні постачальники пропонують низку інноваційних продуктів як послуги, до яких можна отримати доступ на їхніх платформах, і допомагають банкам впроваджувати бізнес-моделі та операційні моделі для покращення отримання прибутку, покращення аналізу клієнтів, стримування витрат, швидкої доставки продуктів, релевантних ринку. і ефективно, а також допоможе монетизувати корпоративні дані. Хмара також пропонує величезні можливості для синхронізації підприємства; щоб розбити операційні та дані, що стосуються ризиків, фінансів, регулювання, підтримки клієнтів тощо. Після того, як масивні набори даних об’єднано в одному місці, організація може застосувати розширену аналітику для інтегрованої інформації.

Після багатьох років зосередження уваги на цінності технології як дешевшої, швидшої та більш «гнучкої» альтернативи локальному сховищу даних, керівники банків розглядають, як вони можуть використовувати хмару в трьох

сферах «над лінією», щоб створити нові межі бізнесу і в трьох сферах «під лінією» для оптимізації організації. Застосування хмарних технологій у цих шести сферах може допомогти банкам підвищити продуктивність бізнесу та прибутки акціонерів.

Критичні джерела цінності, які створені завдяки хмарній трансформації:

«Над лінією»

Нові межі бізнесу.

Синхронізувати підприємство

– Краща інтеграція бізнес-підрозділів завдяки обміну даними, ухваленню комплексних рішень і швидшому вирішенню проблем клієнтів.

– Створення загальних пов'язаних наборів даних; надання глибшої, складнішої інформації та аналітики; покращення співпраці за допомогою нових спільних платформ та інструментів, а також збільшення швидкості прийняття рішень.

Стимулюйте бізнес-інновації

– Допомога впроваджувати інновації та розвивати стратегію для створення нового досвіду для клієнтів, створювати та продавати пропозиції, оптимізувати операції та керувати талантами за допомогою таких інструментів, як машинне навчання, платформи Інтернету речей, доповнена та віртуальна реальність, розпізнавання зображень, обробка природної мови тощо. Використання нових інструментів і можливостей для збільшення доходу, скорочення витрат, підвищення узгодженості операцій і ефективнішого утримання персоналу.

Розкрийте нові таланти та нові способи роботи

– Поєднання технологій із бізнес-підрозділом має сприяти функціям, які потребують нових талантів і нових способів роботи.

– Технологічні можливості та рішення залучають нових працівників і надають доступ до екосистем із новими наборами навичок – DevOps, гнучкість, досвід користувача тощо.

– Impact дозволяє вдосконалювати процеси, такі як автоматизація або допомога персоналу, щоб підвищити продуктивність і створити міцну інтеграцію, що забезпечує гнучкість, зв'язок і прозорість.

"Нижче лінії"

Оптимізація організації.

Створіть стійкі операції

– Підвищення загальної стійкості компаній, щоб швидше реагувати на фізичні збої, збої тощо.

– Перехід із центру обробки даних компанії, але отримання можливості тиражувати дані та служби додатків у кількох центрах обробки даних чи регіоні.

Підвищення безпеки IT

– Хмарні провайдери дотримуються високих стандартів безпеки та мають досвід роботи. Середовища можуть бути настільки ж безпечними або більш безпечними, ніж локальні, але лише за умови правильного впровадження та за допомогою кваліфікованих і навчених спеціалістів безпеки.

За потреби масштабуйте витрати на обчислення

– Допомога організаціям у тому, як вони платять за технології – від великих авансових капітальних витрат до операційної бази.

– Компанії можуть швидше реагувати на зміни на ринку або зміни у фінансових пріоритетах.

– Зафіксуйте економічну ефективність у динамічному хмарному ціноутворенні, збільшуючи або зменшуючи обчислювальну потужність за потреби та полегшуючи детальний контроль витрат.

Рішення корпоративного рівня

Керівники бізнес-підрозділів та IT-спеціалістів, які звикли до локального центру обробки даних, можуть вважати перспективу модернізації або заміни застарілих систем хмарним рішенням корпоративного рівня досить лякаючою. На щастя, банки можуть підходити до цієї трансформації поступово. Вони можуть поєднувати гібридні та мультихмарні рішення залежно від своїх організаційних

потреб, зрілості та готовності; більшість організацій обирають багатохмарний підхід. Незалежно від моделі розгортання, дані, що зберігаються в хмарі, можуть бути такими ж (або навіть більш) безпечними, ніж моделі локального зберігання.

Моделі розгортання хмари

Компанії можуть працювати ва-банк на хмарі, не будучи на 100% хмарою; вони можуть поєднуватися та поєднуватися залежно від потреб. У кожному варіанті дані можуть бути такими ж (або більш) безпечними, ніж у локальних варіантах.

Планування та впровадження хмарних рішень

Розробка бізнес-кейсу

Хмара не тільки допомагає впроваджувати інноваційну IT-стратегію, але й стає рушієм для швидкого створення нових можливостей і послуг для задоволення бізнес-імперативів. Багато трансформаційних рішень (наприклад, управління взаємовідносинами з клієнтами, фінанси, управління корпоративними ресурсами) уже базуються на хмарі – вони просто не передаються як такі. У хмарному бізнес-кейсі слід наголошувати на тому, як банк може рентабельно використовувати хмарні рішення, щоб залучати клієнтів до інформації, досвіду та пропозицій; зростати дохід; менші витрати; знайти та залучити кращий талант; і забезпечити більш послідовні корпоративні операційні платформи. Він також має включати базову модель оцінки цінності хмари для відображення економіки змін ринкових сил, ціноутворення та бізнес-припущень, а також допомоги у плануванні сценаріїв. нарешті, бізнес-кейс має стосуватися питань управління змінами: хмарні технології можуть кардинально змінити певні ролі співробітників; які кроки можуть знадобитися, щоб допомогти адаптувати культуру та мислення організації?

Проектування та виконання рішення

Вартість і зусилля, пов’язані з перенесенням робочих навантажень у хмару, можуть викликати серйозне занепокоєння для фінансових установ, які планують реалізувати хмарні стратегії. Вартість і час виходу на ринок є

ключовими факторами, коли компанії прагнуть використовувати такі технології для побудови бізнесу, як передова аналітика даних і машинне навчання. Зовнішні хмарні постачальники пропонують ці та інші можливості, які можуть скоротити час розробки порівняно зі створенням можливостей власними силами.

Управління постачальником

Протягом наступних років банківська галузь переходитиме як на гібридні, так і на багатохмарні середовища. Протягом цього тривалого періоду постачальники, ймовірно, регулярно пропонуватимуть нові хмарні послуги та можливості. Організаціям, які надають фінансові послуги, слід уникати прив'язки до постачальників, щоб вони могли адаптуватися до змін на ринку без необхідності змінювати платформу під час переходу від одного постачальника до іншого. Крім того, у міру зрілості постачальники можуть запропонувати кращу гнучкість ціноутворення, використовуючи різні хмарні платформи, які дозволяють організації переміщувати робочі навантаження з однієї хмари в іншу для задоволення потреб бізнесу, а також застосовувати найкращі практики, створені на одній хмарній платформі, до відділів, які використовують інші хмарні постачальники.. Прийняття стратегії з кількома постачальниками/багатьма хмарами може бути складним і викликом;

Безпека

Питання про безпеку даних є головною проблемою керівників банків. Важливою частиною розуміння хмари є розгляд того, як поточна інфраструктура та можливості підприємства можуть обмежувати його здатність виявляти та усунути нові ризики та вразливості, а також як хмарні технології можуть допомогти. Безпека в хмарі відрізняється через інструменти, які є рідними для середовища кожного постачальника хмари, і той факт, що постачальники хмари зазвичай відповідають за безпеку рівнів інфраструктури нижчого рівня. Спільна відповідальність за безпеку між хмарними провайдерами та клієнтами, які вони розміщують, змінює те, як організації повинні передбачати ризики безпеці та готуватися до них.

Відповідність нормативним вимогам

Хмарні обчислення можуть допомогти банкам і компаніям, які надають фінансові послуги, відповідати постійно змінюваним нормативним вимогам до звітності (наприклад, Комплексний аналіз і огляд капіталу, Платоспроможність II) у багатьох операційних юрисдикціях – критично важлива можливість у галузі, де транскордонні операції є нормою. Хмарні рішення також можуть допомогти банкам проводити внутрішньоденні розрахунки ліквідності та ризику, а також отримувати дані спостереження за торгівлею, щоб виявляти проблеми з відмиванням грошей та інші проблеми шахрайства. Хмарна платформа надає можливості розміщення даних-посередників на основі критичності даних і сертифікації Certified Safety Professional.

Нові випадки використання в FSI

З 2016 по 2018 рік спостерігалося трикратне збільшення кількості організацій, які використовують хмару для просування інновацій. Як і слід було очікувати, головний інформаційний директор/головний технічний директор є основним рушієм хмарної трансформації, за яким йдуть головний виконавчий директор і бізнес-лідерство. Компанії у всій світовій індустрії фінансових послуг перебувають на шляху до публічної хмари протягом останніх трьох-п’яти років, з величезним прискоренням за останні 12–18 місяців.

– Глобальна фінансова компанія та один із найбільших банків Сполучених Штатів почали використовувати приватну хмару платформи як послуги п’ять років тому. У 2016 році вона оцінила постачальників публічних хмарних технологій і наразі має два додатки для оптової торгівлі в публічній хмарі.

– Провідний інвестиційний банк використовує постачальника загальнодоступної хмари для рішень для регулятивної звітності. Він проводить пілотні проекти з двома постачальниками для хмарного рішення інфраструктури як послуги.

– Великий банк у Північній Америці наразі працює у приватній хмарі та починає працювати на публічній хмарі, переважно використовуючи програмне

забезпечення як послугу та інфраструктуру як послугу. Банк очікує, що протягом наступних чотирьох років стане мультихмарним.

Рівень витрат на технології та прогнози зростання підтверджують, що хмарні обчислення є найважливішою силою, яка формує ринок технологічних послуг. У світовій індустрії фінансових послуг організації використовують приватні, державні та гібридні хмарні рішення для створення інноваційних продуктів і послуг, сприяють трансформації підприємства та переосмислюють «мистецтво можливого».

Банки та IP-телефонія

У все більш конкурентному банківському середовищі клієнти бажають лише найкращого. Банки повинні робити все можливе, щоб надавати найкращі послуги. Оскільки обслуговування клієнтів може здійснюватися в різних формах, банки вирішили зосередитися на телефонних системах.

Ось де з'являються переваги IP-телефонії з усіма перевагами комунікаційних послуг на основі Інтернету. Фінансові установи мають кращі шанси покращити свій публічний зв'язок за допомогою IP-телефонії , ніж ті, які вони могли б мати за допомогою з'єднань телефонної мережі на основі каналів.

Банки та IP-телефонія – як вони можуть отримати вигоду

Бізнес розвивається швидко, коли компанії вирішують перенести свої комунікації в хмарні системи. Фірми, що надають фінансові послуги, також користуються перевагами інноваційних хмарних комунікацій, щоб покращити свої загальні послуги та процес спілкування.

Незважаючи на те, що IP-телефонія може отримати вигоду від IP-телефонії багатьма способами, ось деякі з найпоширеніших способів впливу цієї технології на банківську діяльність:

1. Жодних зборів за виклористання і MAC

Однією з поширених проблем, з якою стикаються багато банків навіть сьогодні, є додавання, зміна або видалення існуючої телефонної лінії (MAC), оскільки в кожному разі потрібно зв'язуватися з постачальником послуг.

Телефонні компанії стягують плату на основі кожного MAC і змушують банк чекати, оскільки вони працюють за своїм графіком.

Цю проблему можна усунути за допомогою IP-телефонії, оскільки всю мережу можна налаштувати в хмарі та масштабувати за вимогою. Немає необхідності звертатися до постачальника послуг. Не чекайте, доки вони виконають належну перевірку, щоб системи зв'язку знову почали працювати.

2. Децентралізований підхід

Оскільки IP-телефонія функціонує через використання Інтернету, банки більше не обмежуються покладатися на централізовану систему WAN для зв'язку. Тепер фінансові установи можуть скористатися перевагами децентралізованої програмної АТС, яка завжди працює. Це суперечить централізованим АТС, які були встановлені в різних місцях.

3. Відчуття атс у кожному відділенні

Банки та IP-телефонія дуже добре поєднуються, щоб представляти та надавати досвід, подібний до наявності АТС у кожному відділенні. Окрім цього, за допомогою IP-телефонії, Інтернет-протоколу Voiceover або VoIP усім можна керувати з централізованого місця. Завдяки цьому банки можуть зменшити плату за дзвінки з відділення до відділення. Це дає співробітникам багато корисних функцій, які раніше були недоступні.

Користувачі вибирають між прослуховуванням голосових повідомлень на телефоні чи комп'ютері, оскільки електронна та голосова пошта інтегровані. Завдяки розширеному графічному інтерфейсу телефонні системи можна швидко переналаштувати. Це означає, що це повністю усуває необхідність звертатися до телефонної компанії для будь-яких змін у MAC.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Види voip пристроїв, переваги-недоліки

Існує 7 видів VoIP пристроїв:

1) IP-телефони з провідною трубкою та провідним підключенням до мережі

(Grandstream GXP, Panasonic KX-HDTV, Fanvil X, Yealink SIP-T та ін.)

Рисунок 2.1 – IP-телефони з провідною трубкою та провідним підключенням до мережі

Переваги:

1. Є найнадійнішими з усіх видів VoIP обладнання.

2. Рідко виходять з ладу, не потрібно проміжного обладнання для підключення до sip (крім маршрутизатора, звичайно ж), підключаються до мережі дротом, живлення від електрики (отже не вимагають батарейок). Завдяки цим факторам дані апарати стабільно тримають sip реєстрацію, мають гарну якість зв'язку та тривалий термін служби.

3. Прості у первинному налаштуванні.

4. Прості у підключенні.

5. Ідеальні для колл-центру та технічної підтримки, оскільки є можливість підключити професійну гарнітуру (залежно від моделі).

6. Недорогі у покупці (залежно від функціоналу).

Недоліки:

1. Мобільність (оскільки трубка та гарнітура провідні, то немає можливості переміщатися під час розмови).

2. Вимагають додаткового простору на робочому столі.

2. IP-телефони з дротовою трубкою, що підключаються до мережі Wi-Fi

(Htek UC924E, Yealink SIP-T27G, Yealink SIP-T41S та ін.)

Рисунок 2.2 – IP-телефони з дротовою трубкою, що підключаються до мережі Wi-Fi

Загалом переваги і недоліки у даного виду обладнання практично все ті ж, що й у попереднього представника.

Додається тільки ще 1 плюс та 2 недоліки.

Додаткові переваги:

1. Зручні у підключенні. Немає необхідності тягнути дроти до кожного апарату, що скорочує рівень витрат на комутацію.

Додаткові недоліки:

1. Якість зв'язку та стабільність підключення до sip безпосередньо залежить від якості Wi-Fi сигналу. Необхідно встановлювати близько до роутера або використовувати репітери.

2. Висока ціна.

3. IP-телефони з радіотрубкою, прив'язаною до основної бази, яка підключається дротом до мережі

(Panasonic KX-TGP600, Gigaset A540, Yealink W52P, Grandstream DP750 та ін.)

Рисунок 2.3 – IP-телефони з радіотрубкою, прив'язаною до основної бази, яка підключається дротом до мережі

Переваги:

1. Достатньо надійне обладнання.
2. Мобільні у використанні. Дуже зручно, що можна вільно переміщатися з трубкою під час розмови.
3. Заощаджують робочий простір.
4. Заощаджують ресурси підключення. Провідом до мережі підключається лише базовий блок.
5. До одного базового блоку можна підключити до 8 трубок (залежно від моделі).

Недоліки:

1. Погіршення якості зв'язку в залежності від відстані між трубкою та базовим блоком, а також в залежності від товщини та матеріалу стін.
2. Потрібен додатковий елемент живлення – батарейки, відповідно і час на підзарядку.
3. Можливе погіршення стабільності роботи у разі великого навантаження на один базовий блок.
4. Досить складні в налаштуванні (залежно від моделі).
5. Є можливість відв'язування трубки від бази, що не очевидно звичайному користувачеві.
6. Висока вартість.

Незважаючи на 6 мінусів, апарати цього виду мають довгий термін служби і користуються великою популярністю. Необхідно просто дотримуватися правильного режиму експлуатації.

Рекомендації при експлуатації:

– підключайте на один базовий блок кількість трубок на 1-2 менше, ніж заявлено виробником.

– не встановлюйте базовий блок на дальню відстань від трубок, краще, щоб база знаходилася в одному кабінеті з трубками та на одному поверсі.

– своєчасно змінюйте акумуляторні батареї в трубках.

4. Аналогові телефони з проводовою трубкою, що підключаються через голосовий VoIP-шлюз

(телефони Panasonic KX-TS, Gigaset DA та ін., VoIP шлюзи Cisco SPA, Grandstream HT, AudioCodes MP та ін.)

Рисунок 2.4 – Аналогові телефони з проводовою трубкою, що підключаються через голосовий VoIP-шлюз

Переваги:

1. Можливість використання в IP-телефонії апаратів, що залишилися від старої аналогової телефонії.
2. Низька вартість аналогових апаратів.
3. Простота підключення до VoIP шлюзу.
4. Не потрібне підключення апаратів до електрики.
5. Довговічність.

Недоліки:

1. Безпосередньо залежать від VoIP шлюзу і якості патч-корду, що підключається.

2. Імовірність перемикання в імпульсний режим, що є неочевидним для звичайного користувача.

3. Імовірність проблем зі зв'язком аналогового характеру – писк, клацання, скрегіт, гул.

4. Налаштування VoIP шлюзу часто викликає труднощі навіть у системних адміністраторів.

5. Займає додатковий простір на робочому місці.

6. Мають нестабільне підключення до sip (іноді вимагають перезавантаження VoIP шлюзу).

5. Аналогові телефони з радіотрубкою, що підключаються через голосовий VoIP-шлюз

(телефони Panasonic KX-TG, Philips D та ін., VoIP шлюзи Cisco SPA, Grandstream HT, AudioCodes MP та ін.)

Рисунок 2.5 – Аналогові телефони з радіотрубкою, що підключаються через голосовий VoIP-шлюз

Переваги та недоліки практично все ті ж, що й у попереднього виду аналогових апаратів.

Додається 2 переваги та 2 недоліки.

Додаткові переваги:

1. Мобільність. Можна переміщатися з слухавкою під час розмови.
2. Не займають додаткового простору робочому місці.

Додаткові недоліки:

1. Можливе погіршення якості зв'язку в залежності від відстані до базового блоку або наявність перешкод радіоканалу.

2. Необхідний додатковий елемент живлення – акумуляторні батареї, відповідно і час на підзарядку.

6. Програмні IP-телефони (софтфони) для дзвінків з ПК та ноутбуків (MicroSIP, Jitsi, ZoiPer, X-Lite та ін.)

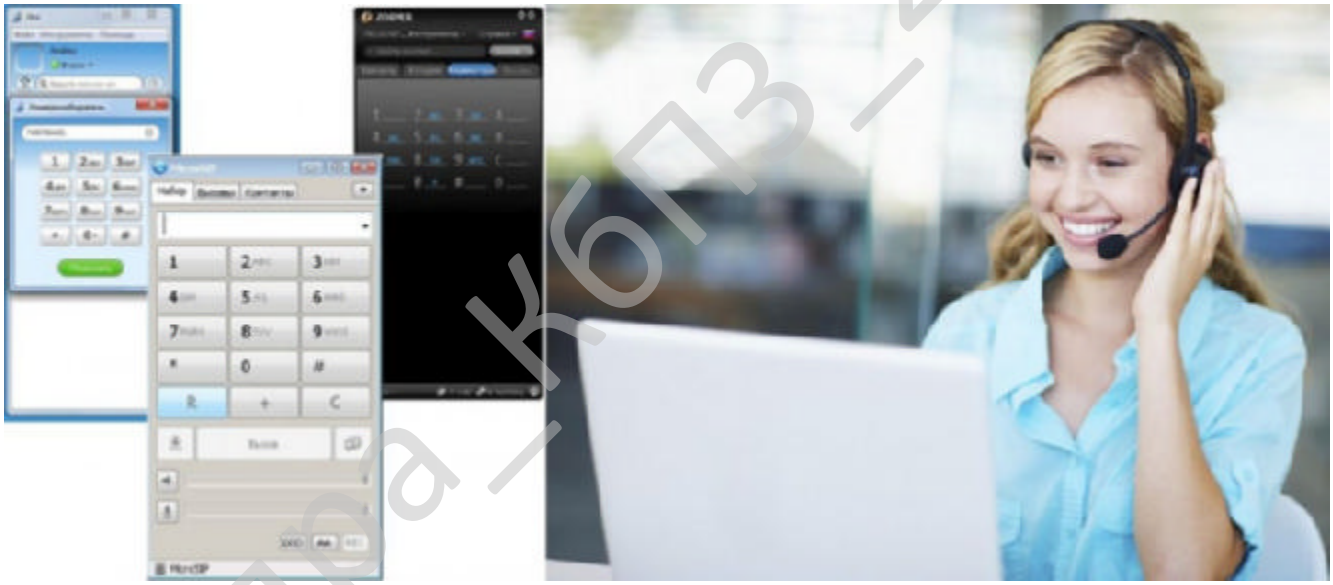

Рисунок 2.6 – Програмні IP-телефони (софтфони) для дзвінків з ПК та ноутбуків

Софтфони – це такі ж IP-телефони, тільки без фізичного корпусу з платами. Просто встановлюєш на ПК, підключаєш гарнітуру, налаштовуєш sip підключення та користуєшся.

Переваги:

1. Безкоштовні (за винятком додаткових платних функцій).
2. Не вимагають додаткової комутації мережі для підключення.
3. Встановлюються на будь-який ПК та основні ОС.
4. Легко налаштовуються (залежно від моделі).

Недоліки:

1. Безпосередньо залежить від швидкодії працездатності ПК.
2. Мають властивість "глючити" (потрібно перезапуск програми, або всього ПК).
3. Для хорошої якості зв'язку потрібне провідне підключення ПК до роутера.
4. Необхідний додатковий елемент – гарнітура.

Незважаючи на недоліки, цей вид VoIP пристроїв здобули велику популярність у використанні. Основні причини це – **бюджетність**.

7. Програмні IP-телефони (софтфони) для дзвінків зі Смартфона

(Grandstream Wave, Linphone, CSipSimple, ZoiPer та ін.)

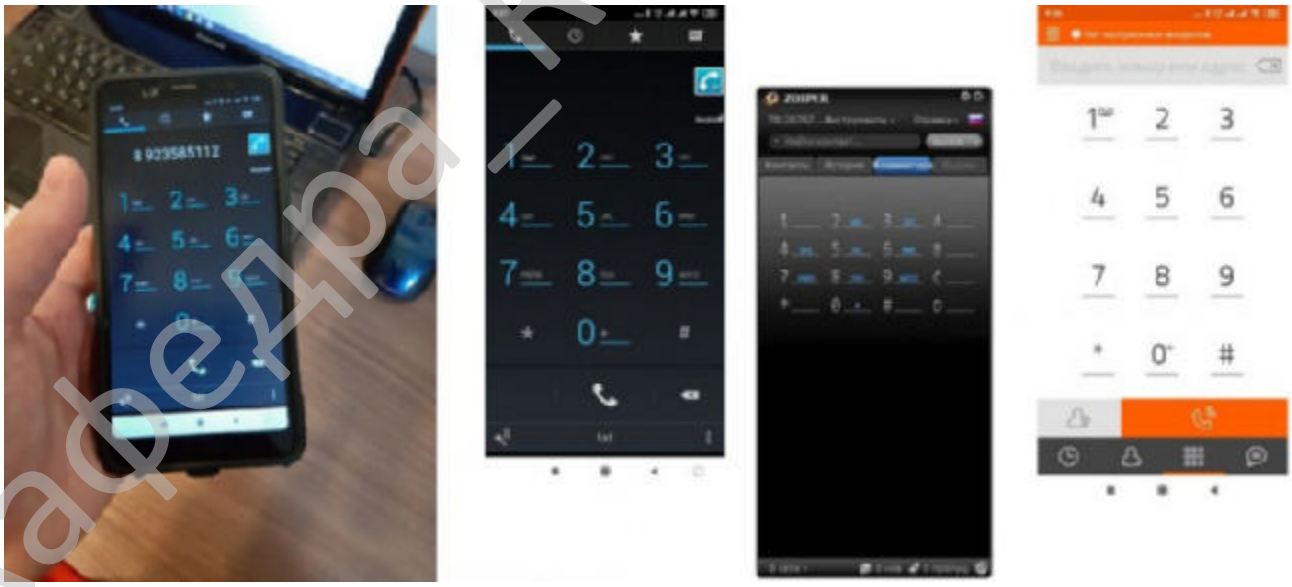

Рисунок 2.7 – Програмні IP-телефони (софтфони) для дзвінків зі Смартфона

Переваги та недоліки практично все ті ж, що й у попереднього виду софтфонів.

Але додається ще 2 переваги та 2 недоліки.

Додаткові переваги:

1. Мобільність.
2. Зручність.

Додаткові недоліки:

1. Потрібно постійно тримати програму запущеною (через це швидко сідає батарея телефону).
2. У 80% випадків погана якість зв'язку під час розмови (оскільки немає можливості підключити пристрій до інтернету дротом).

2. Таблиця "Оцінки надійності" за видами VoIP пристроїв та фактори, що впливають на надійність

Шкала оцінки:

10 балів – найвищий рівень надійності, далі за спаданням

Таблиця 2.1 – Види VoIP пристроїв, ступінь їх надійності та фактори, що впливають на стабільність роботи

10 балів

Дротовий IP-телефон

(Grandstream GXP, Panasonic KX-HDTV, Fanvil X, Yealink SIP-T та ін.)

Провід (вита пара)

1) патч-корд від апарата до світчу або роутера;

2) локальна мережа або маршрутизатори;

3) мережа Інтернет провайдера;

4) сам апарат.

9 балів

IP-телефон з базою та радіотрубками

(Panasonic KX-TGP600, Gigaset A540, Yealink W52P, Grandstream DP750 та ін.)

Провід (вита пара)

1) дальність прийому сигналу між базою і трубкою;

2) радіоперешкоди.

8 балів

Дротовий IP-телефон (Wi-Fi)

(Htek UC924E, Yealink SIP-T27G, Yealink SIP-T41S та ін.)

Wi-Fi

1) дальність прийому сигналу між роутером та апаратом;

2) радіоперешкоди Wi-Fi сигналу;

3) локальна мережа або маршрутизатори;

4) мережа Інтернет провайдера;

5) сам апарат.

7 балів

Аналоговий телефон та VoIP шлюз

(телефони Panasonic KX-TG, Philips D та ін., VoIP шлюзи Cisco SPA, Grandstream HT, AudioCodes MP та ін.)

Провід (вита пара)

1) патч-корд від VoIP шлюзу до освітлення або роутера;

2) патч-корд від VoIP шлюзу до апарату;

3) локальна мережа або маршрутизатори;

4) мережа Інтернет провайдера;

5) сам апарат;

6) сам VoIP шлюз.

6 балів

Софтфон на ПК

(MicroSIP, Jitsi, ZoiPer, X-Lite та ін.)

Провід (вита пара)

1) патч-корд від ПК до світчу або роутера;

2) локальна мережа або маршрутизатори;

3) мережа Інтернет провайдера;

4) сам ПК;

5) сам Софтфон;

6) гарнітура.

3 бали

Софтфон на ноутбуці

(MicroSIP, Jitsi, ZoiPer, X-Lite та ін.)

Wi-Fi

1) дальність прийому сигналу між роутером та ноутбуком;

2) радіоперешкоди Wi-Fi сигналу;

3) локальна мережа або маршрутизатори;

4) мережа Інтернет провайдера;

5) сам Ноутбук;

6) сам Софтфон;

7) гарнітура.

2 бали

Софтфон на смартфоні

(Grandstream Wave, Linphone, CSipSimple, ZoiPer та ін.)

Wi-Fi, 2G/3G/4G

1) дальність прийому сигналу між роутером і смартфоном;

2) радіоперешкоди Wi-Fi сигналу;

3) локальна мережа або маршрутизуючі;

4) пристрої;

5) мережа Інтернет провайдера

6) якість Мобільного інтернету;

7) сам Смартфон;

8) сам Софтфон;

9) постійна підтримка софтфону в активному стані.

Які із цього можна зробити висновки?

Вибір VoIP пристрою залежить від поставлених завдань і конкретних потреб.

Наприклад, якщо вам необхідно організувати **бюджетний кол-центр** з комп'ютерами і хорошим зв'язком, то найкращим варіантом буде використовувати софтфон на ПК, підключений до локальної мережі проводом, і професійну гарнітуру в межах 1,5-3 тисяч грн.

Якщо бюджет не дуже обмежений і в пріоритеті стоїть стабільність і якість зв'язку, тоді найкращим варіантом буде придбання IP-телефонів з дротовими трубками та підключенням до локальної мережі, а також придбання професійних гарнітур до даних апаратів.

Для організації зв'язку **секретарям або адміністраторам** переважно вибирають IP-телефони з радіотрубками або аналогові радіотрубки з VoIP шлюзом.

Менеджерам в основному встановлюють софтфони на ПК з гарнітурою або IP-телефони з радіотрубками. Іноді використовують софтфони на мобільних, але це дуже рідко через значні недоліки якості та стабільності зв'язку.

Для вирішення проблеми мобільності та роботи через IP-телефонію на смартфонах зараз існують такі рішення як **FMC або програми-агенти**, які дозволяють також здійснювати прокидок дзвінка, здійсненого з мобільного пристрою, та запису розмови до ВАТС та CRM-систем.

Огляд деяких моделей VoIP пристроїв

У цьому розділі поділюся досвідом інженерів при роботі з клієнтами, які користуються IP-телефонією та різними пристроями VoIP.

Я не став розписувати порівняння з іншими моделями. Скажу лише, що в ході роботи нам доводиться мати справу з великою кількістю різних моделей VoIP пристроїв, і серед них я вирішив виділити лише ті, які, на нашу думку, найдостойніші.

Прохання звернути увагу, що рекомендації мають виключно суб'єктивний характер, заснований на особистому досвіді інженерів у ході роботи з клієнтами, які активно використовують IP-телефонію.

1. IP-телефони з дротовою трубкою та дротовим підключенням до мережі

З цієї категорії зарекомендували себе апарати **Grandstream GXP**.

Дані моделі є в варіаціях GXP1400, GXP1405, GXP1610, GXP1620 і так далі. Залежно від кількості облікових записів, що настроюються, і додаткових функцій.

Чому Grandstream GXP?

Даний апарат має високий термін служби (від 10 років), простий у налаштуванні, sIP-реєстрацію тримає стабільно, має всі необхідні опції для тонкого налаштування, є можливість наскрізного підключення до мережі, не використовуючи додатковий порт у світчі або роутері, є можливість підключення професійної гарнітури має дуже інформативне цифрове табло.

Скільки пам'ятаю за свою роботу, на даний апарат практично ніхто не скаржиться, за винятком лише кількох моментів, коли виходили з ладу проводки від трубки до апарата. І було також кілька моментів, що телефон сам автоматично скидався до заводських налаштувань. Але це одиниці випадків.

Тому цей апарат ми завжди рекомендуємо нашим клієнтам.

2. IP-телефони з дротовою трубкою, що підключаються до мережі Wi-Fi

У своїй практиці лише один раз мав справу з IP-телефоном, підключеним до мережі по Wi-Fi. По-моєму, це був Escene, модель точно не пам'ятаю. Підключався віддалено, допомагав адміну з налаштуваннями.

Нічого не можу сказати по цьому апарату, тому що маю мало досвіду роботи з цією категорією пристроїв VoIP.

3. IP-телефони з радіотрубкою, прив'язаною до основної бази, яка підключається дротом до мережі

У цій категорії зарекомендували себе апарати **Panasonic KX-TGP500/TGP600**. TGP500 це більш стара модель, а TGP600 оновлена та актуальна на випуск.

Бонус у TGP500 у тому, що до цієї бази можна прив'язати навіть аналогові трубки, і вони нормально функціонуватимуть. Це дуже хороша особливість, тому що аналогові трубки з можливістю прив'язки до бази мають ціну в 3-4 рази меншу, ніж оригінальні трубки TPA50.

Але в оновленій моделі, TGP600 таку можливість відсікли.

Чому Panasonic TGP500/600?

Даний апарат має можливість підключення до 8 трубок до однієї бази та 8 ліній одночасних розмов, при цьому почувається нормально. Апарат досить простий у первинному налаштуванні sip акаунтів та прив'язки трубок. Зазвичай справляються самостійно багато адмінів і навіть деякі користувачі. Sip реєстрацію тримає стабільно, у разі залипання вирішується перезавантаженням бази. Є можливість тонкого налаштування ліній та інших функцій.

Іноді, звичайно, трапляються проблеми, наприклад, трубка перестає бачити базу або розподіл ліній некоректно працює, але такі несправності виникають вкрай рідко.

Тому рекомендую Panasonic KX-TGP500/TGP600.

4. Аналогові телефони, що підключаються через голосовий VoIP шлюз

Про самі апарати говорити немає сенсу, оскільки вони є суто приймачем сигналу і з ними, як правило проблеми виникають рідко. Це може бути лише проблеми апаратного характеру. Переведення в імпульсний режим, шипіння, тріск або тихий звук та ін. Основну функцію виконує шлюз VoIP. Він перетворює сигнал із цифрового на аналоговий і назад. На ньому налаштовується sip реєстрація та інші налаштування VoIP.

З VoIP шлюзів себе зарекомендували **Grandstream HT**. З цих моделей є варіації HT812, HT814, HT818. Залежно кількості FXS портів.

Пристрій стабільно тримає sip реєстрацію, має безліч корисних налаштувань, 2 профілю, при налаштуванні не викликає труднощів, рідко виходить з ладу.

Також часто зустрічаються VoIP-шлюзи AudioCodes MP-202. Цей шлюз теж досить непоганий. Єдина його болячка, вона сильно гріється і іноді залипає. Проблема вирішується перезавантаженням пристрою, а в гіршому випадку скидання на заводські налаштування. Після переналаштування шлюз працює у штатному режимі.

5. Програмні IP-телефони (софтфони) для дзвінків з ПК та ноутбуків

З усіх софтфонів найбільш стійко зарекомендував себе Opensource проект **MicroSIP**, написаний на C#.

Так, цей софтфон має досить дерев'яний інтерфейс. Але він стабільно тримає реєстрацію sip, має всі необхідні для IP-телефонії налаштування і простий у використанні. Немає ніяких крутих функцій, не навантажує операційну систему, практично не "глючить" і має високу швидкість роботи.

А також софтфон має досить просте налаштування, з яким справляється навіть рядовий користувач. Ми зробили спеціальну покрокову інструкцію, яка ні в кого не викликає запитань.

Є один мінус, софтфон **MicroSIP ще не написаний для MacOS**.

Для MacOS зазвичай добре підходять **Jitsi або ZoiPer**.

6. Програмні IP-телефони (софтфони) для дзвінків зі смартфона

Для використання IP-телефонії на смартфоні, на мою думку, найкращий варіант це **CSipSimple**. Проблема тільки в тому, що проектом вже давно ніхто не займається, тому софтфону немає Play Market. Це приносить труднощі в установці для рядових користувачів.

Але ми знайшли альтернативу.

Найкращим чином зарекомендував себе софтфон **Grandstream Wave**. Є версія як для **Android**, так і для **iOS**.

Цей софтфон простий у налаштуванні, справляється навіть рядовий користувач.

Grandstream Wave відрізняється стабільною роботою, приємним дизайном, широким функціоналом та можливістю тонкого налаштування. Не надто навантажує систему смартфона.

Є також альтернатива, це **LinPhone**. Але за ним були помічені прояви "глюків" іноді.

У будь-якому випадку, я б краще рекомендував Opensource **CSipSimple**. Завантажити інсталятор можна в інтернеті. Налаштовується легко, чудово працює.

У чому перевага Mitel banks та IP-телефонії?

Millennia та наші телефони Mitel допомагають розвивати концепцію IP-телефонних систем. Надаючи нашим клієнтам універсальний інформаційний центр для електронної пошти, телефону та голосових повідомлень. Завдяки унікальній розподіленій архітектурі розгортання систем голосового зв'язку та керування ними стало набагато простіше. Мало того, що весь процес пришвидшується, сучасні технології також дають шанси для кращого масштабування.

Представники служби отримують миттєвий доступ до різноманітної банківської інформації, включаючи історію рахунків, транзакції та інше завдяки розширеним додаткам CRM та ERP. Це може значно прискорити процес і підвищити ефективність персоналу.

Узгодженість між банками та IP-телефонією, яку підтримує Millennia & Mitel, також супроводжується продажами та обслуговуванням клієнтів найвищого рівня. Це може допомогти значно скоротити витрати, пов’язані з обслуговуванням, спільним використанням голосової лінії, подорожами тощо.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services.

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey.

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне

забезпечення, яке призначено для системи дослідження протоколів IP-телефонії хмарної банківської мережі.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис алгоритму IP-телефонії

IP-телефонія – це технологія, що дозволяє використовувати Інтернет або будь-яку іншу IP-мережу для ведення телефонних розмов і передачі факсів у режимі реального часу. Особливо актуально, з економічної точки зору, використання даної технології для здійснення міжнародних і міжміських телефонних розмов або для створення розподілених банківських телефонних мереж.

Устаткування для IP-телефонії

Для організації телефонного зв'язку по IP-мережах використовується спеціальне устаткування – шлюзи IP-телефонії. Загальний принцип дії телефонних шлюзів IP-телефонії такий: з однієї сторони шлюз підключається до телефонних ліній – і може з'єднатися з будь-яким телефоном миру. З іншої сторони шлюз підключений до IP-мережі – і може зв'язатися з будь-яким комп'ютером у світі. Шлюз приймає телефонний сигнал, оцифровує його (якщо він вихідно не цифровий), значно стискає, розбиває на пакети й відправляє через IP-мережу по призначенню з використанням протоколу IP. Для пакетів, що приходять із IP-мережі на шлюз і, що направляються в телефонну лінію, операція відбувається у зворотному порядку. Обидві складові процесу зв'язку (вхід сигналу в телефонну мережу і його вихід з телефонної мережі) відбуваються практично одночасно, що дозволяє забезпечити повнодуплексну розмову. На основі цих базових операцій можна побудувати багато різних конфігурацій.

У цей час, все більшу популярність здобувають IP-ATM, які крім функцій шлюзу IP-телефонії виконують також традиційні функції звичайних офісних АТМ. Таким чином, при організації телефонного зв'язку через IP-мережі з

використанням IP-ATM можна цілком обійтися без офісної ATM, тобто заощадити на додатковому встаткуванні.

IP-телефонія опирається на дві основних операції: перетворення двунаправленої аналогової мови в цифрову форму усередині що кодує/декодувального пристрою (кодека) і впакування в пакети для передачі по IP-мережі. В IP-телефонії використовується особлива система передачі пакетів зі звуковою інформацією, що обумовлено специфікою передачі даних по IP-мережах.

У традиційних телефонних лініях між абонентами під час розмови створюється електричне коло, і цим забезпечується фіксована пропускна здатність для передачі сигналу. У той час як IP-мережа являє собою систему, що реалізує принцип комутації й маршрутизації пакетів, і не надає гарантованого шляху між точками зв'язку. Вся інформація, передана через IP (голос, текст, зображення, і т.п.) розділяється на пакети даних, що мають у своєму составі адреси точок призначення (прийому й передачі) і порядковий номер. Вузли IP направляють ці пакети по мережі до закінчення маршруту доставки.

Після прибуття пакетів до точки призначення, для відновлення вихідного обсягу впорядкованих даних використовуються порядкові номери пакетів. Для додатків, де не важливі порядок і інтервал приходу пакетів, таких як e-mail, час затримок між окремими пакетами не має вирішального значення. IP-телефонія є однією з областей передачі даних, де важлива динаміка передачі сигналу, що забезпечується сучасними методами кодування й передачі інформації. Для забезпечення стабільного телефонного зв'язку по IP-мережах введені спеціальні протоколи передачі даних, наприклад, H.323 і SIP.

Основні протоколи IP-телефонії

H.323 – основний стандарт, прийнятий ITU-T, де описується, яким образом чутливий до затримки трафік, зокрема голос і відео, одержує пріоритет у локальних і глобальних мережах. Він складається з ряду рекомендацій (стека протоколів) по суміжних технічних питаннях, таким, як якість мови, стандарти

кодування звуковий і відеоінформації та ін. Протокол SIP (Session Initiation Protocol) прийнятий у березні 2000 року організацією IETF як стандарт RFC 2543. SIP більшою мірою відповідає ідеології TCP/IP, ніж стек протоколів H.323. Про підтримку цього протоколу заявили такі виробники як 3Com, Cisco, Ericsson, Siemens і ін. Однозначність стандарту SIP дозволяє із упевненістю говорити про сумісність IP-шлюзів різних виробників.

При передачі в режимі реального часу до 30% пакетів можуть бути загублені або отримані із запізненням (що в режимі реального часу те саме). Гарний додаток IP-телефонії повинен відшкодувати недостачу пакетів, відновивши загублені дані. Сам алгоритм кодування мови також впливає на відновлення даних.

H.323

H.323, містить описи термінальних пристроїв, устаткування й мережних служб, призначених для здійснення мультимедійного зв'язку в мережах з комутацією пакетів (наприклад, Intranet або Інтернет). Термінальні пристрої й мережне встаткування стандарту H.323 можуть передавати дані, мову й відеоінформацію в масштабі реального часу. У рекомендації H.323 не визначені: мережний інтерфейс, фізичне середовище передачі інформації й транспортний протокол, використовуваний у мережі. Мережа, через яку здійснюється зв'язок між терміналами H.323, може являти собою сегмент або безліч сегментів зі складною топологією. Термінали H.323 можуть бути інтегровані в персональні комп'ютери або реалізовані як автономні пристрої. Але підтримка мовного обміну – обов'язкова функція для будь-якого пристрою стандарту H.323.

Рекомендації H.323 передбачають:

– Керування смугою пропускання. Передача аудіо– і відеоінформації досить інтенсивно навантажує канали зв'язку, і, якщо не стежити за ростом цього навантаження, працездатність критично важливих мережних сервісів може бути порушена. Тому рекомендації H.323 передбачають керування смугою пропущення. Можна обмежити як число одночасних з'єднань, так і сумарну смугу пропущення

для всіх додатків H.323. Ці обмеження допомагають зберегти необхідні ресурси для роботи інших мережних додатків. Кожний термінал H.323 може управляти своєю смугою пропущення в конкретній сесії конференції.

– Міжмережні конференції. Рекомендації H.323 пропонують засоби з'єднання учасників відеоконференції в різнорідних мережах (наприклад, IP і ISDN, IP і PSTN).

– Платформна незалежність. H.323 "не прив'язаний" до яких-небудь технологічних рішень, пов'язаних з устаткуванням або програмним забезпеченням. Взаємодіючі між собою додатки можуть створюватися на основі різних платформ, з різними операційними системами.

– Підтримка багатоточкових конференцій. Рекомендації H.323 дозволяють організовувати конференцію із трьома або більше учасниками. Багатоточкові конференції можуть проводитися як з використанням центрального контролера – MCU (пристрою багатоточкової конференції), так і без нього.

– Підтримка багатоадресної передачі. H.323 підтримує багатоадресну передачу в багатоточковій конференції, якщо мережа підтримує протокол керування груповою адресацією. При багатоадресній передачі один пакет інформації відправляється всім необхідним адресатам без зайвого дублювання. багатоадресна передача використовує смугу пропущення набагато більш ефективно, оскільки всім адресатам – учасникам списку розсилання відправляється рівно один потік.

– Стандарти для кодеків. H.323 установлює стандарти для кодування й декодування аудіо– і відеопотоків з метою забезпечення сумісності устаткування різних виробників. Разом з тим стандарт досить гнучкий. Сформульовано вимоги, виконання яких обов'язково, і існують опціональні можливості, у випадку використання яких також необхідно строго додержуватися стандарту. Крім цього, виробник може включати в мультимедійні продукти й додатки додаткові можливості, якщо вони не суперечать обов'язковим і опціональним вимогам стандарту.

– Сумісність. Можливі випадки, коли учасники конференції хочуть спілкуватися один з одним, не піклуючись про питання сумісності між собою. Рекомендації H.323 підтримують з'ясування загальних можливостей устаткування кінцевих користувачів і встановлюють найкращі із загальних для учасників конференції протоколів кодування, виклику й керування.

– Гнучкість. H.323 конференція може включати учасників, кінцеве встаткування яких має різні можливості. Наприклад, один з учасників може використовувати термінал тільки з аудіо можливостями, у той час як інші учасники конференції можуть мати можливості передачі/прийому також відео й даних.

Архітектура стандарту H.323

У рекомендації H.323 установлюється чотири основних компоненти VoIP–з'єднання.

– термінал;

– контролер зони;

– шлюз (gateway);

– пристрій керування багатоточковою конференцією (MCU).

Термінал (Terminal) – окінцевий мультимедійне (голос, відео, дані) пристрій, призначений для участі в конференції. Під терміналом стандарт розуміє встаткування кінцевих точок мережі, що дозволяє користувачам спілкуватися один з одним у реальному часі. H.323-термінал повинен забезпечувати підтримку наступних протоколів:

1. H.245 для встановлення можливостей терміналів і створення каналу обміну аудіоінформацією.

2. H.225 для сигналізації виклику й установки параметрів зв'язку.

3. RAS для реєстрації термінала користувача й установки додаткових параметрів керування контролером зони.

4. RTP/RTCP для упорядкування звукових і відеопакетів.

H.323-термінал повинен також підтримувати звуковий кодер-декодер відповідно до G.711.

Протоколи H.225 і RAS використовуються між H.323-окінцевими точками (терміналами й шлюзами) і контролером зони для забезпечення:

– виявлення контролера зони (GRQ);

– реєстрації окінцевої точки;

– визначення розташування окінцевої точки;

– керування автентифікацією;

– завдання маркера доступу.

RAS-повідомлення передаються через ненадійні RAS-канали, тому при обміні повідомленнями можливі втрати, затримки й повторні передачі.

Принципи побудови протоколу SIP

Протокол ініціювання сеансів (Session Initiation Protocol – SIP) є протоколом прикладного рівня й призначається для організації, модифікації й завершення сеансів зв'язку (наприклад, мультимедійних конференцій, телефонних з'єднань). Користувачі можуть брати участь в існуючих сеансах зв'язку, запрошувати інших користувачів і бути запрошеними ними до нового сеансу зв'язку.

Протокол SIP розроблений групою MMUSIC комітету IETF, а специфікації протоколу представлені в документі RFC 2543. В основу протоколу закладені наступні принципи:

1. Персональна мобільність користувачів. Користувачі можуть переміщатися без обмежень у межах мережі. Користувачеві привласнюється унікальний ідентифікатор, а мережа надає йому послуги зв'язку поза залежністю від того, де він перебуває.

2. Масштабованість мережі. Вона характеризується, у першу чергу, можливістю збільшення кількості елементів мережі при її розширенні. Серверна структура мережі, побудована на базі протоколу SIP, відповідає цій вимозі.

3. Розширюваність протоколу. Вона характеризується можливістю доповнення протоколу новими функціями при введенні нових послуг і його адаптації до роботи з різними додатками.

Розширення функцій протоколу SIP може бути зроблене за рахунок введення нових заголовків повідомлень, які повинні бути зареєстровані в організації IANA. При цьому якщо SIP-сервер приймає повідомлення з невідомими йому атрибутами, то він просто ігнорує їх.

Для розширення можливостей протоколу SIP можуть бути також додані й нові типи повідомлень.

4. Інтеграція в стек існуючих протоколів Інтернету, розроблених IETF. Протокол SIP є частиною глобальної архітектури мультимедіа, розробленої IETF. Ця архітектура містить у собі також і інші протоколи: резервування ресурсів (Resource Reservation Protocol – RSVP, RFC 2205), транспортний протокол реального часу (Real-Time Transport Protocol – RTP, RFC 1889), протокол передачі потокової інформації в реальному часі (Real-Time Streaming Protocol – RTSP, RFC 2326), протокол опису параметрів зв'язку (SDP, RFC 2327). Однак функції самого протоколу SIP не залежать від жодного із цих протоколів.

5. Взаємодія з іншими протоколами сигналізації. Протокол SIP може бути використаний разом із протоколом H.323.

Протокол керування шлюзами MGCP

Протокол запропонований робочою групою MEGACO (Media Gateway Control Protocol) комітету IETF.

Основна ідея MGCP дуже проста. Вона полягає в тому, що керування сигналізацією (Call Control) зосереджено на центральному керуючому пристрої, називаному контролером сигналізацій (Call Agent, CA), і повністю відділено від медіа-потоків. Ці потоки обробляються шлюзами або абонентськими терміналами, які здатні виконувати лише обмежений набір команд, що виходять від керуючого пристрою. В архітектурі протоколу MGCP-мережі можна виділити два основних функціональних компоненти. Перший може бути представлений транспортним шлюзом (Media Gateway, MG) або IP-телефоном, а другий – пристроєм керування викликами, що може називатися контролером сигналізацій (CA), контролером шлюзу (Media Gateway Controller, MGC) або програмним

контролером (Softswitch, SS). Іноді контролер сигналізацій представляють у вигляді двох компонентів – власно контролера (Call Agent), що виконує функції керування шлюзами, і шлюзу сигналізації (Signaling Gateway), що забезпечує обмін сигнальною інформацією й узгодження між традиційною телефонною мережею й мережею IP.

При розробці протоколу керування шлюзами робоча група MEGACO опиралася на принцип декомпозиції, відповідно до якого шлюз розбивається на окремі функціональні блоки:

– транспортний шлюз – Media Gateway, що виконує функції перетворення мовної інформації, що надходить із боку ТфОП з постійною швидкістю, у вид, придатний для передачі по мережах з маршрутизацією пакетів IP: кодування й упакування мовної інформації в пакети RTP/UDP/IP, а також зворотне перетворення;

– пристрій керування викликами – Call Agent, що виконує функції керування шлюзом;

– шлюз сигналізації – Signaling Gateway, що забезпечує доставку сигнальної інформації, що надходить із боку ТфОП, до пристрою керування шлюзом і перенос сигнальної інформації у зворотному напрямку.

Контролери обмінюються зі шлюзами (або IP-телефонами) аудіоданними в простому текстовому форматі, а функціональне призначення кожного шлюзу визначається набором команд, які він "розуміє". Маніпулюючи наборами команд, можна одержувати спеціалізовані шлюзи: транкові (Trunking gateways, TGW), абонентські (Residential gateways, RGW), шлюзи доступу (Access gateways, AGW) і ін.

Контролер сигналізацій CA сприймає мережа як набір двох логічних елементів – пристроїв (endpoints) і з'єднань (connections) між ними. Пристрої можуть бути фізичними (наприклад, IP-телефони або лінії на шлюзах) або віртуальними (наприклад, лінії до серверів голосових повідомлень). З'єднання можуть бути орієнтовані на передачу голосу, факс-повідомлень або даних.

Керування цими елементами, тобто організація з'єднань між пристроями, відбувається шляхом посилки команд у вигляді текстових (ASCII) повідомлень по протоколі UDP – при цьому може використовуватися протокол SDP.

Найпростіший сценарій з'єднання буде виглядати в такий спосіб: користувач телефону, підключеного до MGCP-шлюзу, знімає трубку, після чого шлюз повідомляє контролер про цю подію, а СА дає команду шлюзу включити в телефонну лінію сигнал готовності (dial-ton). Тепер користувач чує в трубці безперервний гудок. Далі треба набір телефонного номера – теж послідовність подій для контролера. Аналізуючи ці події, СА може встановити з'єднання з іншим абонентом в IP-мережі або в телефонній мережі.

Класифікація шлюзів по області застосування

Застосовується наступна класифікація транспортних шлюзів (Media Gateways):

– Trunking Gateway – шлюз між ТфОП і мережею з маршрутизацією пакетів IP, орієнтований на підключення до телефонної мережі;

– Voice over ATM Gateway – шлюз між ТфОП і ATM-мережею, що також підключається до телефонної мережі за допомогою великої кількості цифрових трактів;

– Residential Gateway – шлюз, що підключає до IP-мережі аналогові, кабельні модеми, лінії xDSL і широкополосні пристрої бездротового доступу;

– Access Gateway – шлюз для підключення до мережі IP-телефонії невеликої АТМ для установ через аналоговий або цифровий інтерфейс;

– Business Gateway – шлюз із цифровим інтерфейсом для підключення до мережі з маршрутизацією IP-пакетів АТМ установи;

– Network Access Server – сервер доступу до IP-мережі для передачі даних;

– Circuit switch або packet switch – комутаційні пристрої з інтерфейсом для керування від зовнішнього пристрою.

Основні алгоритми стиску звуку, використовувані в IP-телефонії

Для кодування звукової інформації звичайно використовуються наступні кодеки: G.711, G.722, GSM0610, G.723, G.723.1, G.728, і G.729. Для кодека G.711 потрібна ширина смуги частот в 64 Кбіт/с, тому він прийнятний не у всіх IP-мережах (наприклад, в Інтернет), тому що більшість користувачів Інтернету має канал свідомо меншої ширини. Кодеки з низькою шириною смуги частот – G.729 в 8 Кбіт/с і G.723.1 в 5.3/6.3 Кбіт/с – цілком підходять для використання в Інтернет. Зокрема, G.723.1 є одним з декількох "стандартних" кодеків для IP-телефонії, особливо після того, як Intel, Microsoft і Netscape оголосили про підтримку цього стандарту звукового кодування.

Взаємодія протоколів voip

При використанні протоколів, які безпосередньо мають справа з VoIP, важливо правильне розуміння специфікації, внесеної цими протоколами. На рисунку 3.1 показаний стек протоколів VoIP. Тут відсутній верхній рівень, що має на увазі в собі будь-яку розмовну мову. Даний Рисунок характеризує винятково передачу голосових даних.

6. Рівень подань G.729/G.711
5. Рівень сеансу H.323, шлюз SIP/SDP
4. Транспортний рівень, протоколи RTP/UDP/RSVP
3. Мережний рівень IP/LLQ
2. Канальний рівень MLPPP/FR/ATM AAL5
1. Фізичний рівень

Рисунок 3.1 – Стек протоколів VoIP

Технологія VoIP може працювати в будь-якому фізичному середовищі, що може використовуватися звичайним протоколом IP. Такі середовища можуть бути представлені у вигляді кабелю крученої пари (використовуваної в

традиційному Ethernet), телефонних проводів, бездротових з'єднань (протокол IEEE 802.11) і ін.

Другий рівень цієї моделі – канальний рівень – указує, що протокол IP для створення фреймів може використовувати різні формати. Як показано на рисунку 3.1, він включає багатоканальний PPP (Multilink PPP), Frame Relay (FR) і ATM. При проектуванні мережі можливі й інші варіанти, оскільки передавати голос можуть також Ethernet, Wi-Fi і інші технології локальних мереж.

На третьому, мережному рівні використовується протокол IP як спосіб передачі голосу, однак звичайний IP повинен бути доповнений спеціальними засобами. Оскільки існують проблеми із затримкою, протоколу IP потрібно використовувати який-небудь спосіб установлення черговості для того, щоб голосовим даним не довелося очікувати передачі в умовах конкуренції зі звичайними даними. На маршрутизаторах повинна бути використана черговість із малою затримкою (Low-Latency queuing – LLQ) або яка-небудь інша сучасна схема установки черговості, щоб голосові дані відправлялися раніше звичайних даних. Крім того, повинні використовуватися схеми маркування (marking) із завданням пріоритетів (coloring), називані IP-пріоритетами, для забезпечення того, щоб голосові дані розглядалися системою як більше важливі для першочергової передачі, чим звичайні дані.

Наступним рівнем є транспортний. Оскільки для передачі голосу використовується протокол UDP, системі не вистачає механізму установки черговості пакетів, щоб пакети доставлялися в необхідній послідовності. Транспортний протокол реального часу (Real-Time Transport Protocol – RTP) для виконання цієї вимоги додає номер пакета в послідовності передачі й механізм розміщення тимчасових міток. Також може використовуватися протокол резервування (Resource Reservation Protocol – RSVP) для резервування смуги пропущення уздовж шляхи проходження голосу по IP-мережі. Даний протокол виключає застосування пакетами звичайних даних зарезервованої смуги пропущення.

П'ятий рівень моделі – сеансовий. На сьогоднішній день мережі VoIP переходять зі стандарту ITU-T H.323 на інший протокол ініціювання сеансу (Session Initiation Protocol – SIP) і протокол опису сеансу (Session Description Protocol – SDP).

Шостим рівнем моделі є рівень подань. Як визначено в моделі OSI, рівень подань аналізує й інтерпретує формати даних. У термінах передачі голосу рівень подань забезпечує методи кодування й стиску, використовувані для передачі голоси.

Всі рівні стека протоколів спільно застосовуються для того, щоб вирішити проблеми мінімізації затримки й забезпечити необхідний порядок проходження пакетів.

Види з'єднань, взаємодія з комп'ютерною мережею

Можна виділити три найбільше часто використовуваних сценарії IP-телефонії:

– комп'ютер-комп'ютер;

– телефон-комп'ютер;

– телефон-телефон.

Перші сценарій " комп'ютер-комп'ютер" реалізується на базі стандартних комп'ютерів, оснащених засобами мультимедіа й підключених до мережі Інтернет.

Компоненти сценарію " комп'ютер-комп'ютер"

У цьому сценарії аналогові мовні сигнали від мікрофона абонента А перетворяться в цифрову форму за допомогою аналого-цифрового перетворювача (АЦП). Відрахунки мовних даних у цифровій формі потім стискуються пристроєм, що кодує, для скорочення потрібної для їхньої передачі смуги у відношенні 4:1, 8:1 або 10:1. Вихідні дані після стиску формуються в пакети, до яких додаються заголовки протоколів, і потім пакети передаються через IP-мережу в систему IP-телефонії, що обслуговує абонента Б. Коли пакети приймаються системою абонента Б, заголовки протоколу віддаляються, а стислі

мовні дані надходять у пристрій, що розгортає їх у первісну форму, після чого мовні дані знову перетворяться в аналогову форму за допомогою цифро-аналогового перетворювача (ЦАП) і попадають у динамік телефону абонента Б. Для звичайного з'єднання між двома абонентами системи IP-телефонії на кожному кінці одночасно реалізують як функції передачі, так і функції прийому. Під IP-мережею, мається на увазі або глобальній мережі Інтернет, або банківська мережа підприємства Intranet.

Для підтримки сценарію "комп'ютер-комп'ютер" постачальникові послуг Інтернет необхідно мати окремий сервер (GateKeeper), що перетворить імена користувачів у динамічні адреси IP. Сам сценарій орієнтований на користувача, якому мережа потрібна в основному для передачі даних, а програмне забезпечення IP-телефонії потрібно лише іноді для розмов з колегами. Ефективне використання телефонного зв'язку по сценарії " комп'ютер-комп'ютер" звичайно пов'язане з підвищенням продуктивності роботи великих компаній, наприклад, при організації віртуальної презентації в банківській мережі з можливістю не тільки бачити документи на веб-сервері, але й обговорювати їхній зміст за допомогою IP-телефону.

Для проведення телефонних розмов один з одним абоненти А и Б повинні мати доступ до Інтернету або до іншої мережі із протоколом IP. Розберемо можливий алгоритм організації зв'язку між цими абонентами на прикладі протоколу H.323.

1. Абонент А запускає свій додаток IP-телефонії, що підтримує протокол H.323.

2. Абонент Б також заздалегідь запустив свій додаток IP-телефонії, що підтримує протокол H.323.

3. Абонент А знає доменне ім'я абонента Б – Domain Name System (DNS), уводить це ім'я в розділ "кому подзвонити" у своєму додатку IP-телефонії й натискає кнопку Return.

4. Додаток IP-телефонії звертається до DNS-сервера (який у даному прикладі реалізований безпосередньо в персональному комп'ютері абонента А) для того, щоб перетворити доменне ім'я абонента Б у IP-Адреса.

5. Сервер DNS повертає IP-адресу абонента Б.

6. Додаток IP-телефонії абонента А одержує IP-Адресу абонента Б і відправляє по цій адресі сигнальне повідомлення H.225 Setup.

7. При одержанні повідомлення H.225 Setup додаток Б сигналізує абонентові Б о вхідному виклику.

8. Абонент Б приймає виклик і додаток IP-телефонії відправляє відповідне повідомлення H.225 Connect.

9. Додаток IP-телефонії в абонента А починає взаємодія з додатком в абонента Б у відповідності з рекомендацією H.245.

10. Після закінчення взаємодії по протоколі H.245 і відкриття логічних каналів абоненти А и Б можуть розмовляти один з одним через IP-мережу.

При цьому блок "Керування й сигналізація" управляє пакетизацією і депакетизацією переданих фрагментів, а також здійснює контроль при їхній передачі.

У цьому прикладі не показані деякі службові деталі, які необхідні постачальникові послуг для розгортання мережі IP-телефонії.

Сценарій "телефон-комп'ютер"

За назвою "комп'ютер" у всіх сценаріях ми будемо розуміти термінал користувача, включений в IP-мережу, а за назвою "телефон" – термінал користувача, включений у мережу комутації каналів будь-якого типу: ТфОП, ISDN або GSM.

Наступний сценарій "телефон-комп'ютер" знаходить застосування в різного роду довідково-інформаційних службах Інтернету, у службах збуту товарів або в службах технічної підтримки. Користувач, що підключився до серверу WWW якої-небудь компанії, має можливість звернутися до оператора

довідкової служби. Це цілком відповідає стилю життя сучасних споживачів, пов'язаному з потребою в додаткових зручностях і економії часу.

У другому сценарії "телефон-комп'ютер" з'єднання встановлюється між користувачем ТфОП і користувачем IP-мережі. Передбачається, що встановлення з'єднання ініціює користувач мережі комутації каналів.

Шлюз для взаємодії мереж ТфОП і IP може бути реалізований як окремим пристроєм, так і інтегрованим в існуюче встаткування ТфОП або IP-мережі.

Можливий й інший різновид другого сценарію, коли з'єднання встановлюється між користувачем IP-мережі й абонентом ТфОП, але ініціює його створення абонент ТфОП.

Розглянемо трохи докладніше архітектуру системи IP-телефонії по сценарії "телефон-комп'ютер". При спробі викликати довідково-інформаційну службу, використовуючи послуги пакетної телефонії й звичайний телефон, на початковій фазі абонент А викликає прилеглий шлюз IP-телефонії для мінімізації витрат на послуги зв'язку. Від шлюзу до абонента А надходить запит вводу номеру, до якого повинен бути спрямований виклик (наприклад, номер служби), і особистий ідентифікаційний номер (PIN) для автентифікації й наступного нарахування плати, якщо ця служба платна. Ґрунтуючись на викликуваному номері, шлюз визначає найбільш доступний шлях до даної служби. Крім того, шлюз активізує свої функції. Роз'єднання з будь-якої сторони передається протилежній стороні по протоколу сигналізації й викликає завершення встановлених з'єднань і звільнення ресурсів шлюзу для обслуговування наступного виклику.

Ефективність об'єднання послуг передачі мови й даних є основним стимулом використання IP-телефонії по сценаріях "комп'ютер-комп'ютер" і "телефон-комп'ютер", не наносячи при цьому збитку інтересам операторів традиційних телефонних мереж.

Сценарій "телефон-телефон"

Третій сценарій " телефон-телефон" у значній мірі відрізняється від перших двох сценаріїв IP-телефонії своєю соціальною значимістю, оскільки метою його застосування є надання звичайним абонентам ТфОП альтернативної можливості міжміського й міжнародного телефонного зв'язку.

Як правило, обслуговування викликів по такому сценарії IP-телефонії виглядає в такий спосіб. Постачальник послуг IP-телефонії підключає свій шлюз до комутаційного вузла або станції ТфОП по мережі Інтернет або по виділеному каналі до аналогічного шлюзу, що перебуває в іншому місті або іншій країні.

Типова послуга IP-телефонії по сценарії "телефон-телефон" використовує стандартний IP-телефон, а замість міжміського компонента ТфОП задіє або частну IP-мережу, або мережу Інтернет. Завдяки маршрутизації телефонного трафіка по IP-мережі стало можливим обходити мережі загального користування й, відповідно, не платити за міжміський/міжнародний зв'язок операторам цих мереж.

Постачальники послуг IP-телефонії надають послуги "телефон-телефон" шляхом установки шлюзів IP-телефонії на вході й виході IP-мереж. Абоненти підключаються до шлюзу постачальника послуг IP-телефонії через ТфОП, набираючи спеціальний номер доступу. Абонент одержує доступ до шлюзу, використовуючи персональний ідентифікаційний номер (PIN) або послугу ідентифікації номера зухвалого абонента (Calling Line Identification). Після цього шлюз просить ввести телефонний номер викликуваного абонента, аналізує цей номер і визначає, який шлюз має кращий доступ до потрібного телефону. Як тільки між вхідним і вихідним шлюзами встановлюється контакт, подальше встановлення з'єднання до викликуваного абонента виконується вихідним шлюзом через його місцеву телефонну мережу.

Повна вартість такого зв'язку буде складатися для користувача з розцінок ТфОП на зв'язок із вхідним шлюзом, розцінок інтернет-провайдеру на

транспортування даних і розцінок віддаленої ТфОП на зв'язок вихідного шлюзу з викликаним абонентом.

Одним з алгоритмів організації зв'язку по сценарії "телефон-телефон" є випуск постачальником послуги своїх телефонних карт. Маючи таку карту, користувач, що бажає подзвонити в інше місто, набирає номер постачальника даної послуги, потім у режимі донабору вводить свій ідентифікаційний номер і PIN-код, зазначений на карті. Після процедури автентифікації він набирає телефонний номер адресата.

3.2 Розробка структурної схеми

На рисунку 3.2 зображена структурна схема розробленого програмного забезпечення IP-зв’язку розгалуженої банківської мережі.

Розглянемо основні структурні елементи, які застосовані у розробленій схемі IP-телефонії розгалуженої банківської мережі.

GateKeeper – сервер, що перетворить імена користувачів у динамічні адреси IP.

ТфОП – телефонний оператор, який надає послуги традиційного телефонного зв’язку від шлюзу до телефонної станції.

IP-phone – спеціальний телефон, який можливо під’єднати до IP-мережі напряму. Повнофункціональний IP-телефон дає можливість користувачеві ініціювати й приймати дзвінки, взаємодіючи з банківською VoIP-системою або інфраструктурою постачальника послуг IP-телефонії

Soft phone – програмне забезпечення, яке встановлене на ЕОМ та дозволяє спілкуватися використовуючи мікрофон та WEB-камеру, під’єднані до ЕОМ.

Analog phone – звичайний телефон.

Gateway – комп'ютери з'єднання, “перекладачі”. Вони використовуються для того, щоб утворити з'єднання зі звичайною телефонною мережею. Ці Gateways з'єднані як з комп'ютерною мережею, так і з нормальною телефонною

мережею й передають запити в обоє напрямків. При цьому IP-пакети з однієї сторони перетворяться в цифровий потік даних.

PBX – внутрибанківська телефонна станція.

VoIP provider – провайдер, який надає послуги VoIP-зв'язку.

VoIP gateway – це gateway, який з'єднує аналогові телефони з VoIP-маршрутизатором.

VoIP router – маршрутизатор VoIP-трафіку. Він направляє IP-пакети до їхньої кінцевої точки, почасти різними шляхами.

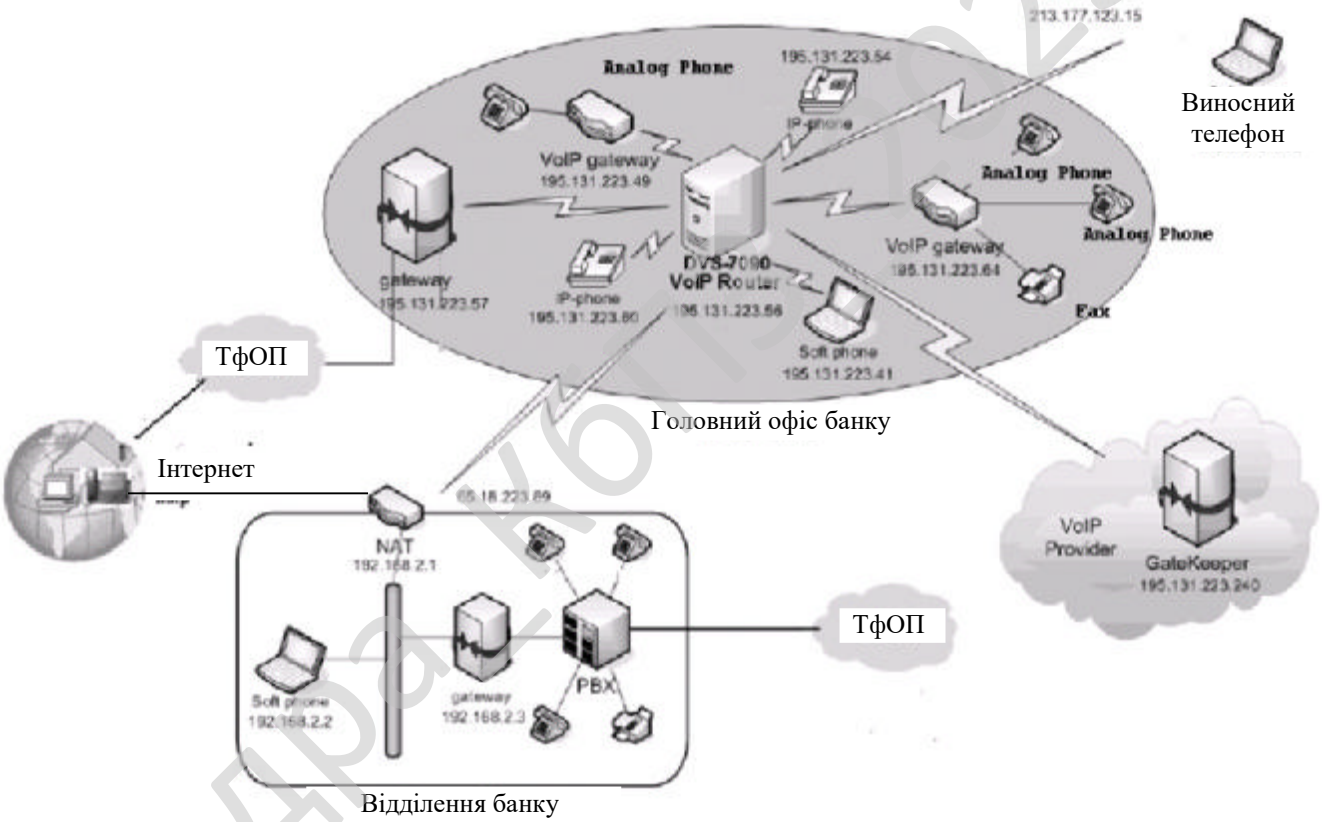

Рисунок 3.2 – Структурна схема розробленої системи

Варіант телефонної системи – телефонна мережа, що охоплює абонентів, що перебувають в одному будинку або групі близько розташованих будинків. У цьому випадку мережна інфраструктура, як правило, представляє із себе локальну/кампусну мережу на базі технологій Ethernet/Fast Ethernet/Gigabit

Ethernet/ Wi-Fi, у рамках якої можуть бути розміщені як абонентські телефони й персональні комп'ютери, так і керуючого пристрою телефонної мережі.

Для організації IP-телефонії з філіями банку, розташованими на значній відстані від центрального офісу, в інших містах або інших країнах, використовується мережа інтернет, у якій, у стеці протоколів TCP/IP, по протоколу IP, телефонія реалізована за допомогою використання протоколу H.323.

Основним пристроєм для керування мережею IP-телефонії є сервер керування дзвінками. Один сервер може підтримувати велику кількість IP-телефонів. Їхнє число залежить від моделі сервера, на якому встановлене програмне забезпечення. З метою масштабування системи й для забезпечення відказостійкості сервери можуть бути об'єднані, що збільшує число телефонів, що приєднуються одночасно. Користувальницькі IP-телефони, так само як і сервери, підключаються до комутаторів локальної/кампусної мережі; при цьому можливо використання технологій, наприклад, Power over Ethernet, що забезпечують подачу електроживлення для IP-телефонів від комутаторів Ethernet. Багато моделей телефонів оснащені убудованим двупортовим комутатором Ethernet, що дозволяє підключити персональний комп'ютер абонента до банківської мережі.

Також у межах локальної мережі розміщаються сервери додатків, таких як система голосової пошти або система інтерактивної мовної взаємодії, які забезпечують додаткові сервіси для абонентів системи.

Для підключення системи IP-телефонії до телефонної мережі загального користування використовуються голосові шлюзи на базі маршрутизаторів, при цьому вибір конкретної моделі шлюзу залежить від типу й кількості інтерфейсів, застосовуваних для стикування із ТфОП (можливе використання як аналогових, так і цифрових інтерфейсів). Шлюзи також потрібні якщо буде потреба підключення системи IP-телефонії до встановленого раніше офісної АТС.

Основні характеристики моделі побудови мережі IP-телефонії для одного будинку або кампуса (декількох будинків, об'єднаних високошвидкісною локальною мережею):

– для організації системи IP-телефонії може використовуватися D-Link DVX-7090, що виконує функції голосового маршрутизатора й рекомендується для застосування в мережах середніх і великих офісів;

– для подальшого масштабування мережі можливе використання декількох подібних пристроїв;

– для підключення до телефонної мережі загального користування (ТфОП) аналогових телефонів і факсових апаратів, стикування з існуючими УАТМ застосовуються голосові шлюзи;

– ресурси голосових сервісних модулів використовуються для організації аудіоконференцій;

– для забезпечення якісної роботи різних додатків рекомендується застосування комутаторів, що підтримують необхідні засоби забезпечення якості сервісу (QoS).

3.3 Розробка функціональної схеми

На рисунку 3.3 зображена функціональна схема розробленого програмного забезпечення.

З функціональної схеми ми бачимо, що основний функціональний модуль включає в себе наступні функціональні підблоки:

– IP-телефонія.

– Селективний зв'язок.

– Відеоконференція.

– Формування вхідної інформації.

– База даних абонентів банківської мережі.

– Біллінг.

– Виведення статусу ПЗ.

– Довідник.

Данні, які поступають з основного функціонального модуля кодуються перешкодостійким алгоритмом кодування Хеммінга, та перетворюються згідно протоколу Н.323, після чого передаються до IP-мережі.

Робота з програмою починається з введення інформаційного вікна й активізації системи меню. Робота програми здійснюється по діалоговому і по-дійному режиму, при цьому по діалогом розуміється надання користувачу декількох альтернатив і обробка його вибору. У діалогову систему входять головне меню з відповідними спливаючими підменю а також діалогові вікна. Під подіями розуміються процеси, що активізуються користувачем (наприклад – натискання функціональних клавіш), а також програмні події – одержання з'єднання з абонентом або закінчення з'єднання з абонентом. На підставі даних подій активізуються процедури контролю допустимості даних.

Головне вікно програми призначений для запуску основних процедур програми і завершення роботи з програмою.

Модуль роботи з довідниками містить у собі два довідники:

– Довідник – Довідкова система.

– Довідник – Інструкція користувача.

Модуль роботи з базою даних абонентів банківської мережі включає в себе наступні підмодулі:

– Редагування легенди IP-номерів.

– Редагування бази даних IP-адрес.

– Моніторинг бази даних IP-адрес та легенд.

Призначення даного модуля є пошук і перегляд інформації з телефонних даних адрес абонентів корпорації, а також їх легенд.

Інформаційною базою даного модуля є таблиці: Значення IP-номерів та Легенда IP-номерів. Дані в інформаційну базу заносяться за допомогою спеціальних форм, що викликаються з головного меню програми.

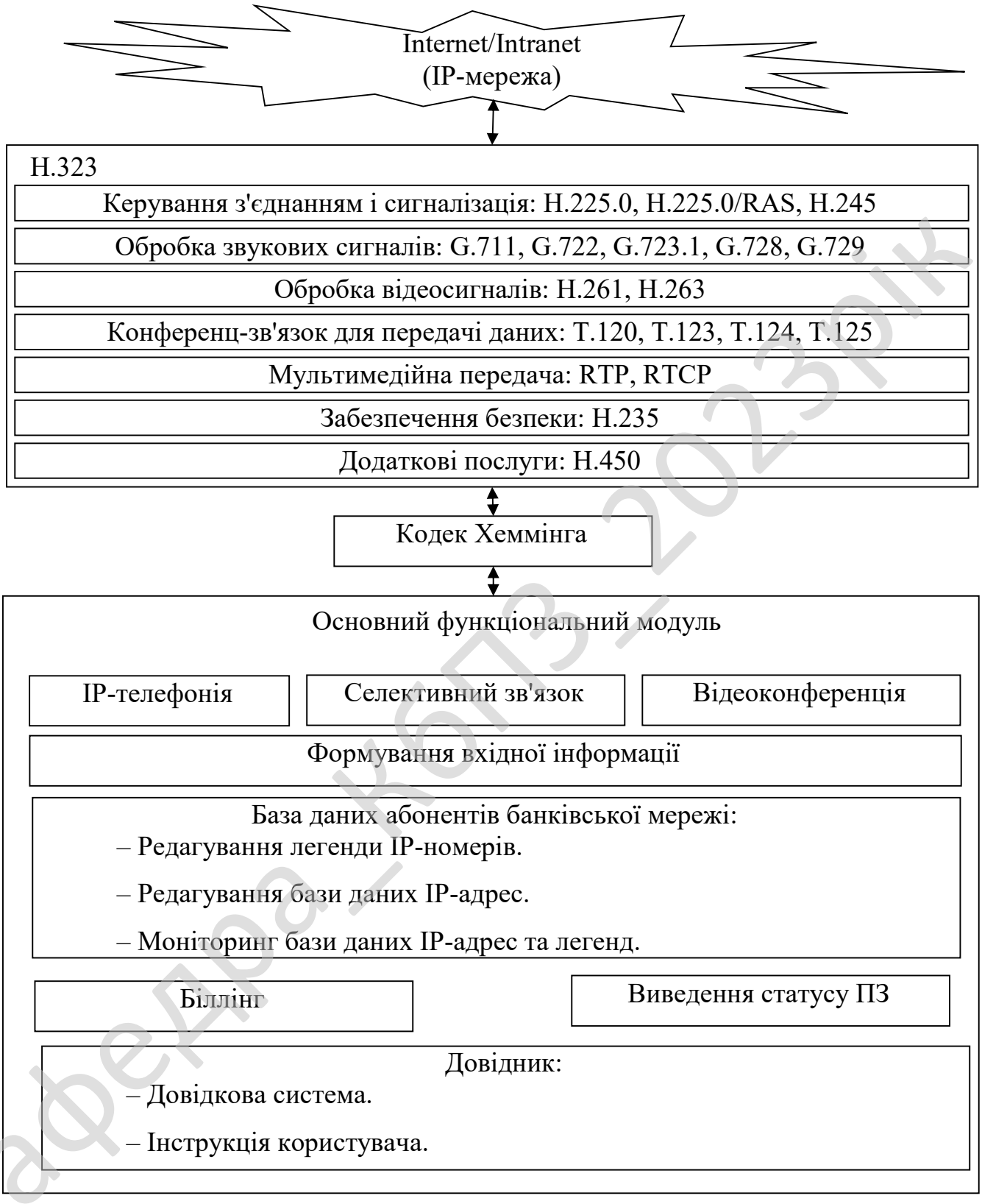

Рисунок 3.3 – Функціональна схема розробленої системи

Модуль «Формування вхідної інформації» призначений для введення первинних даних і перегляду раніше занесених. Даний модуль реалізує задачі обліку телефонних номерів, забезпечуючи введення номерів та їх легенд, ранжування за важливістю та їх знищення.

У комп'ютерних системах користувачі для введення, перегляду та редагування інформації бази даних (IP-адрес та відповідних легенд) можуть застосовувати форми. Основні переваги використання форм наступні:

– При введенні даних у поля-форми, додаток може зчитувати словник даних сервера й автоматично перевірити допустимість даних відповідно до правил цілісності.

– Поле введення у формі може представляти список допустимих значень, з яких користувачі можуть легко вибрати потрібне.

– Область форми може виводити шаблон, що відповідає поточної виведеної у формі запису.

– Командні кнопки у формі можуть виконувати дії, зв'язані з виведеної у формі поточною записом.

Біллінгова система – автоматизована система розрахунків з абонентами за надані послуги. Керування й статистика може бути доступна з будь-якої точки мережі через Веб-інтерфейс.

Опис алгоритму Хеммінга

Побудова кодів Хеммінга базується на принципі перевірки на парність ваги W (числа одиничних символів) в інформаційній групі кодового блоку.

Пояснимо ідею перевірки на парність на прикладі найпростішого коригувального коду, що так і називається кодом з перевіркою на парність або кодом з перевіркою за паритетом (рівності).

У такому коді до кодових комбінацій безнадлишкового первинного двійкового k – розрядного коду додається один додатковий розряд (символ перевірки на парність, називаний перевірочним, або контрольним). Якщо число символів "1" вихідної кодової комбінації парне, то в додатковому розряді формують контрольний символ 0, а якщо число символів "1" непарне, то в

додатковому розряді формують символ 1. У результаті загальне число символів "1" у будь-якій переданій кодовій комбінації завжди буде парним.

Таким чином, правило формування перевірочного символу зводиться до наступного: $r_1 = i_1 \oplus i_2 \oplus ... \oplus i_k$, де i – відповідний інформаційний символ (0 або 1), k – загальне їхнє число а під операцією "⊕" тут і далі розуміється додавання по mod2. Очевидно, що додавання додаткового розряду збільшує загальне число можливих комбінацій удвічі в порівнянні із числом комбінацій вихідного первинного коду, а умова парності розділяє всі комбінації на дозволені й недозволені. Код з перевіркою на парність дозволяє виявляти одиночну помилку при прийманні кодової комбінації, тому що така помилка порушує умову парності, переводячи дозволену комбінацію в заборонену.

Критерієм правильності прийнятої комбінації є рівність нулю результату S підсумовування по mod 2 всіх n символів коду, включаючи перевірочний символ r_1. При наявності одиночної помилки S приймає значення 1:

$S = r_1 \oplus i_1 \oplus i_2 \oplus ... \oplus i_k = 0$ – помилки немає або $= 1$ – однократна помилка.

Цей код є (k+1, k) – кодом, або (n, n-1) – кодом. Мінімальна відстань коду дорівнює двом ($d_{min} = 2$), і, отже, ніякі помилки не можуть бути виправлені. Простий код з перевіркою на парність може використовуватися тільки для виявлення (але не виправлення) однократних помилок.

Збільшуючи число додаткових перевірочних розрядів і формуючи за певними правилами перевірочні символи r, рівні 0 або 1, можна підсилити коригувальні властивості коду так, щоб він дозволяв не тільки виявляти, але й виправляти помилки. На цьому й заснована побудова кодів Хеммінга.

Розглянемо ці коди, що дозволяють виправляти одиночну помилку, за допомогою безпосереднього опису. Для кожного числа перевірочних символів r = 3,4,5... існує класичний код Хеммінга з маркуванням $(n,k) = (2^r-1, 2^r-1-r)$, тобто – (7,4), (15,11), (31,26)..

При інших значеннях числа інформаційних символів k виходять так звані усічені (укорочені) коди Хеммінга. Так, для міжнародного телеграфного коду МТК-2, що має 5 інформаційних символів, буде потрібно використання

коригувального коду (9,5), що є усіченим від класичного коду Хеммінга (15,11), тому що число символів у цьому коді зменшується (коротшає) на 6. Для приклада розглянемо класичний код Хеммінга (7,4), якому можна сформувати й описати за допомогою кодера. У найпростішому варіанті при заданих чотирьох (k=4) інформаційних символах i_2, i_3, i_4) будемо думати, що вони згруповані на початку кодового слова, хоча це й не обов'язково. Доповнимо ці інформаційні символи трьома перевірочними символами (r = 3), задаючи їхніми наступними рівностями перевірки на парність, які визначаються відповідними алгоритмами [3,5]: $r_1 = i_1 \oplus i_2 \oplus i_3$; $r_2 = i_2 \oplus i_3 \oplus i_4$; $r_3 = i_1 \oplus i_2 \oplus i_4$, де знак $\oplus$ означає додавання за модулем 2.

Відповідно до цього алгоритму визначення значень перевірочних символів r нижче виписані всі можливі 16 кодових слів (7,4) – коду Хеммінга.

Кодові слова (7,4) – коду Хеммінга.

к =4				r =3		
i_1	i_2	i_3	i_4	r_1	r_2	r_3
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1

Число можливих синдромів визначається вираженням $S = 2^r$.

При числі перевірочних символів r = 3 є вісім можливих синдромів (2^3 = 8). Нульовий синдром (000) указує на те, що помилки при прийманні відсутні або не виявлені. Усякому ненульовому синдрому відповідає певна конфігурація помилок, що і виправляється. Класичні коди Хеммінга мають число синдромів, точно рівне їхньому необхідному числу, дозволяють виправити всі однократні помилки в будь-якому інформативному й перевірочному символах і включають один нульовий синдром. Такі коди називаються плотновпакованими.

Усічені коди є не щільно впакованими, тому що число синдромів у них перевищує необхідне. Так, у коді (9,5) при чотирьох перевірочних символах число синдромів буде дорівнює 2^4 =16, у той час як необхідно всього 10. Зайві 6 синдромів свідчать про неповне впакування коду (9,5).

Для розглянутого коду (7,4) представлені ненульові синдроми й відповідні конфігурації помилок. Таким чином, код (7,4) дозволяє виправити всі одиночні помилки. Проста перевірка показує, що кожна з помилок має свій єдиний синдром. При цьому можливо створення такого цифрового коректора помилок (дешифратора синдрому), що по відповідному синдромі виправляє відповідний символ у прийнятій кодовій групі. Після внесення виправлення перевірочні символи r_i можна на вихід декодера не виводити. Дві або більше помилки перевищують можливості коригувального коду Хеммінга, і декодер буде помилятися. Це означає, що він буде вносити неправильні виправлення й видавати перекручені інформаційні символи.

Ідея побудови подібного коригувального коду, природно, не міняється при перестановці позицій символів у кодових словах. Всі такі варіанти також називаються (7,4) – кодами Хеммінга.

Стек протоколів H.323

Стандарт H.323 визначає широкі вимоги для багатьох різних протоколів, які становлять повний стек протоколів H.323.

Стек H.323 складають 7 груп протоколів:

– керування й сигналізація;

– обробка звукових сигналів;

– обробка відеосигналів;

– конференц-зв'язок;

– передача мультимедійної інформації;

– забезпечення інформаційної безпеки;

– додаткові послуги;

1. Керування з'єднанням і сигналізація:.

– H.225.0: протоколи сигналізації й пакетування мультимедійного потоку (використовує підмножину протоколу сигналізації Q.931).

– H.225.0/RAS: процедури реєстрації, допуску й стану.

– H.245: протокол керування для мультимедіа.

2. Обробка звукових сигналів:.

– G.711: імпульсно-кодова модуляція тональних частот.

– G.722: кодування звукового сигналу 7 кГц в 64 кбіт/с.

– G.723.1: мовні кодери на дві швидкості передачі для організації мультимедійного зв'язку зі швидкістю передачі 5.3 і 6.3 кбіт/с.

– G.728: кодування мовних сигналів 16 кбіт/с за допомогою лінійного пророкування з кодуванням сигналу порушення з малою затримкою.

– G.729: кодування мовних сигналів 8 кбіт/с за допомогою лінійного пророкування з алгебраїчним кодуванням сигналу порушення сполученої структури.

3. Обробка відеосигналів:.

– H.261: відеокодеки для аудіовізуальних послуг зі швидкістю 64 кбіт/с.

– H.263: кодування відеосигналу для передачі з малою швидкістю.

4. Конференц-зв'язок для передачі даних:.

– T.120: це стек протоколів (який включає T.123, T.124, T.125) для передачі даних між окінцевими пунктами. Він може використовуватися для різних додатків в області спільної роботи (Collaboration Work), такий як колективне редагування растрових зображень, спільне використання додатків і

спільна організація документів. В Т.120 застосовується багаторівнева архітектура, подібна моделі OSI.

5. Мультимедійна передача:.

– RTP: транспортний протокол реального часу.

– RTCP: протокол керування передачею в реальному часі.

6. Забезпечення безпеки:.

– H.235: забезпечення безпеки й шифрування для мультимедійних терміналів мережі H.323.

7. Додаткові послуги:.

– H.450.1: узагальнені функції для керування додатковими послугами в H.323.

– H.450.2: переклад з'єднання на телефонний номер третього абонента.

– H.450.3: переадресація виклику.

– H.450.4: утримання виклику.

– H.450.5: паркування виклику (park) і відповідь на виклик (pick up).

– H.450.6: повідомлення про виклик, що надійшов, у стані розмови.

– H.450.7: індикація повідомлення, що очікує.

– H.450.8: служба ідентифікації імен.

– H.450.9: служба завершення з'єднання для мереж H.323.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.4.

З діаграми взаємодії процесів ми бачимо, що у системі відбуваються наступні процеси.

Робота системи починається з запуску процесу початку/кінця роботи, який взаємодіє з процесом підключення до мережі. Цей процес перевіряє наявність мережі, та відбуває підключення до цієї мережі, якщо мережі немає, то процес видає помилку.

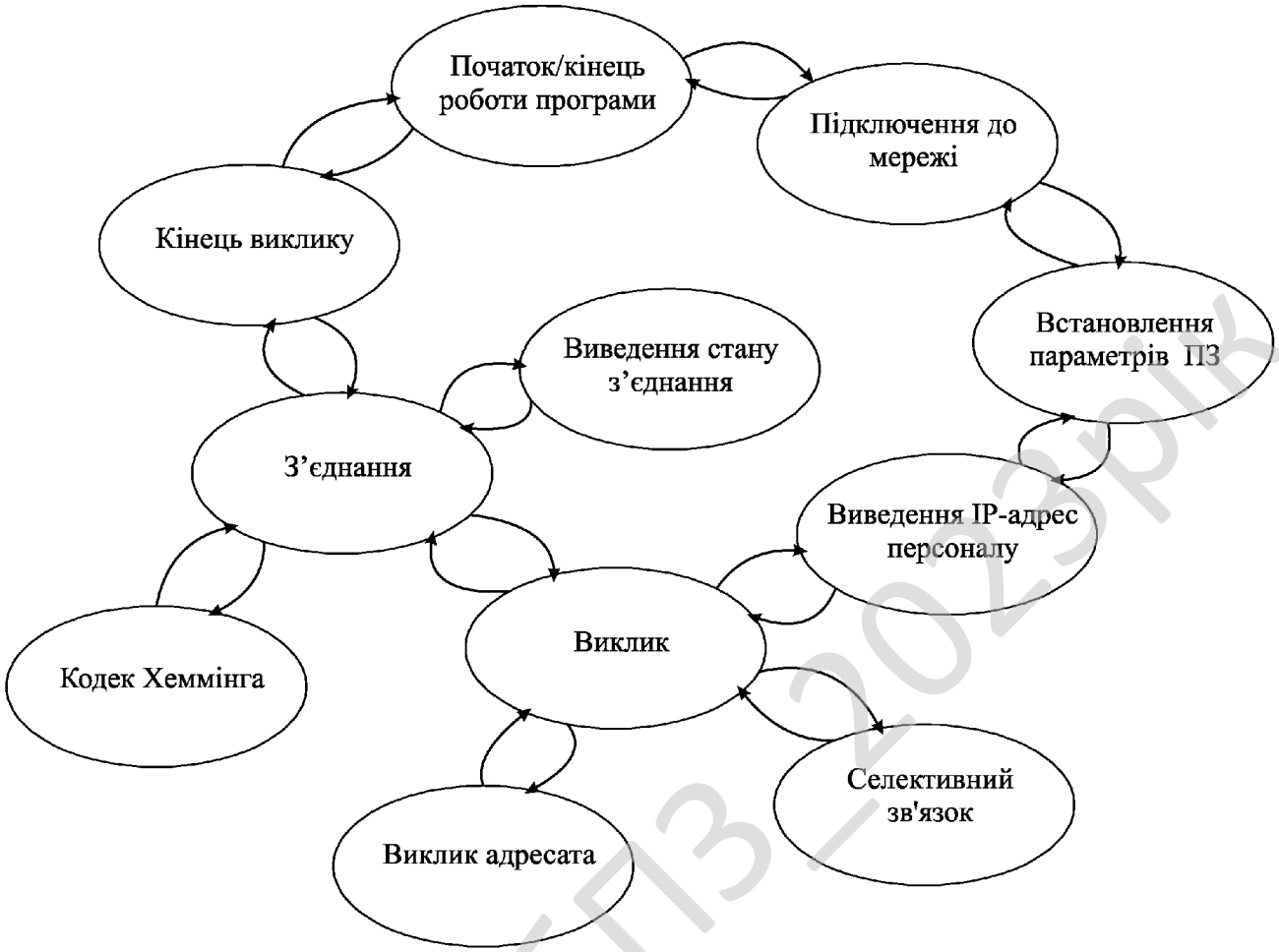

Рисунок 3.4 – Діаграма процесів системи

Процес підключення до мережі у свою чергу взаємодіє з процесом встановлення параметрів програмного забезпечення. До цих параметрі відносяться наступні:

– визначення параметрів користувача програмою;

– визначення прав доступу користувача;

– визначення кодексів, які потрібні для роботи програми.

Процес встановлення параметрів програмного забезпечення взаємодіє з процесом виведення IP-адрес персоналу банківської установи. Цей процес виводить список IP-адрес усіх користувачів, з якими має право спілкуватися конкретний банківський співробітник.

Процес виведення IP-адрес персоналу банківської установи взаємодіє з процесом виклику. При запуску цього процесу відбувається виклик або конкретного користувача, або ряду користувачів, у цьому випадку реалізується селективний зв'язок.

Процес виклику взаємодіє з процесами селективний зв'язок, виклик адресата, з’єднання.

Процес з’єднання взаємодіє з процесами виведення стану з’єднання, обробка інформації за допомогою кодека Хеммінга та процесом кінця виклику.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

У цьому розділі приводяться блок-схеми основної програми, підпрограм, та опис алгоритму їх функціонування.

На рисунку 4.1 зображена блок-схема основної програми. Як ми бачимо з алгоритму програма виконує наступні дії. Спершу відбувається ініціалізація початкових значень програмного забезпечення. Після цього відбувається підключення додаткових файлів. За підключенням додаткових файлів відбувається пошук та читання файлів програмного забезпечення. Якщо файли програмного забезпечення не знайдені то відбувається виведення на екран вікна заставки й програма завершує свою роботу. У іншому випадку, тобто, якщо файли програмного забезпечення знайдені то запускається блок налагодження програмного забезпечення. Після запуску цього блоку виконується налагодження інтерфейсних властивостей програмного забезпечення.

Після проведення усіх підготовчих робіт для роботи програмного забезпечення, відбувається виведення на екран вікна заставки, на якому вказується, хто, де й коли розробив програмний продукт.

Після виведення вікна заставки, запускається підпрограма підключення форми програмного забезпечення.

Коли підпрограма підключення форми програмного забезпечення закінчує свою роботу то завантажується модуль перевірки надійності.

Після завантаження модулю перевірки надійності, на екрані відбувається приховання заставки та виводиться основне вікно програми, тобто на екрані формується основний інтерфейс користувача.

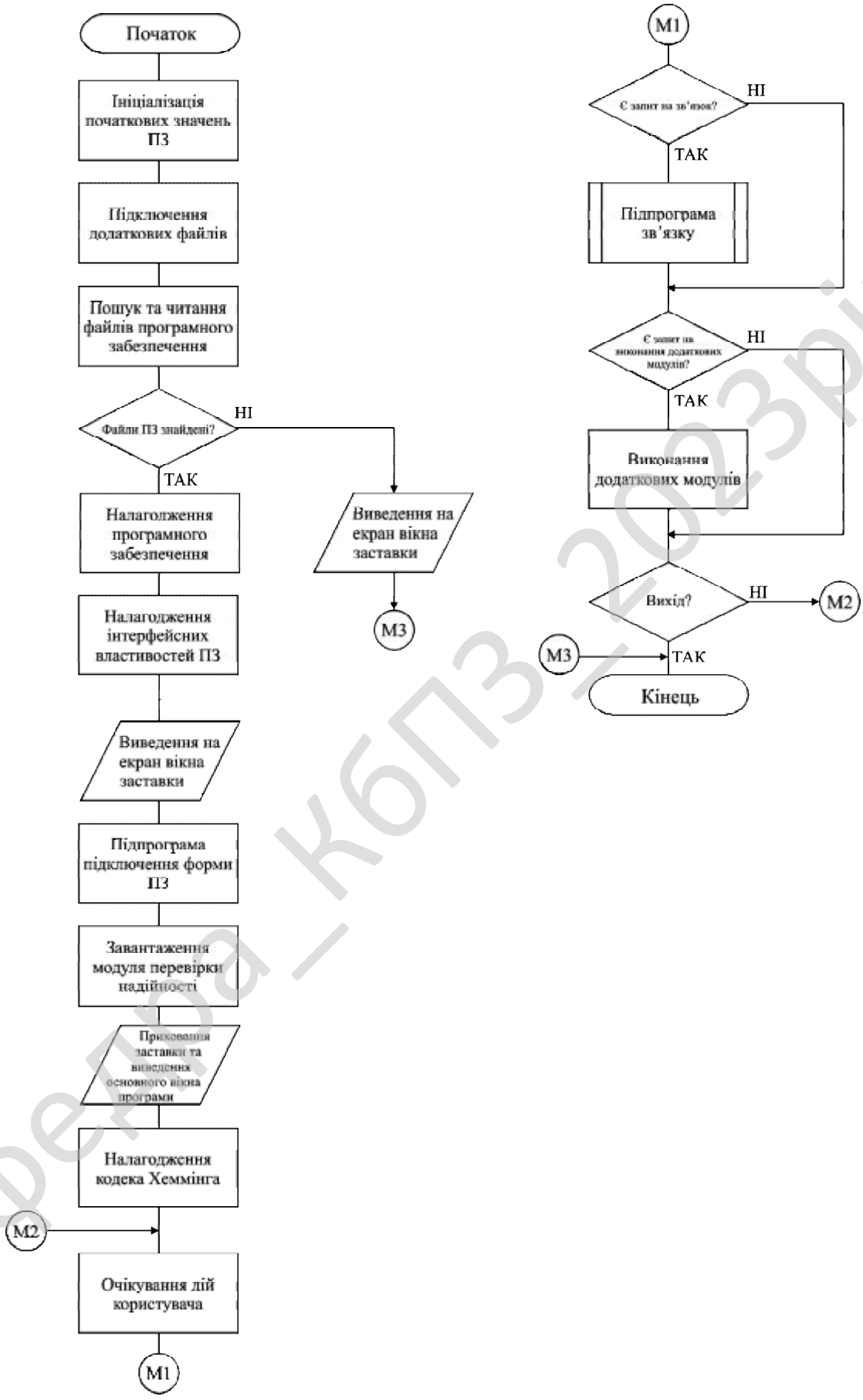

Рисунок 4.1 – Блок схема основної програми

Після виведення основного вікна програми відбувається налагодження кодеку Хеммінга.

Після цього програма переходить у стан очікування дій користувача.

Якщо відбувається запит на зв'язок, то запускається підпрограма встановлення зв’язку.

Якщо ж цього запиту не відбувається, то перевіряється чи є запит на виконання додаткових модулів.

Якщо є запит на виконання додаткових модулів, то відбувається виконання додаткових модулів.

У іншому випадку перевіряється бажання користувача вийти з програми.

Якщо він бажає вийти з програми, то програма завершує свою роботу.

У іншому випадку, відбувається перехід до очікування дій користувача.

На рисунку 4.2 зображена блок-схема підпрограми зв’язку.

Вона виконується наступним чином.

Спершу відбувається спроба доступу до звукової карти.

Якщо ця спроба не є успішною, то виводиться повідомлення про помилку, й програма завершує свою роботу.

Якщо ж спроба доступу до звукової карти успішна, то відбувається спроба доступу до мережного з'єднання.

Якщо ця спроба не є успішною, то виводиться повідомлення про помилку, й програма завершує свою роботу.

У іншому випадку, якщо спроба доступу до мережного з'єднання є успішною, то відбувається вибір одного адресату зв’язку.

Якщо цей вибір не є успішним, то виводиться повідомлення про помилку, й програма завершує свою роботу.

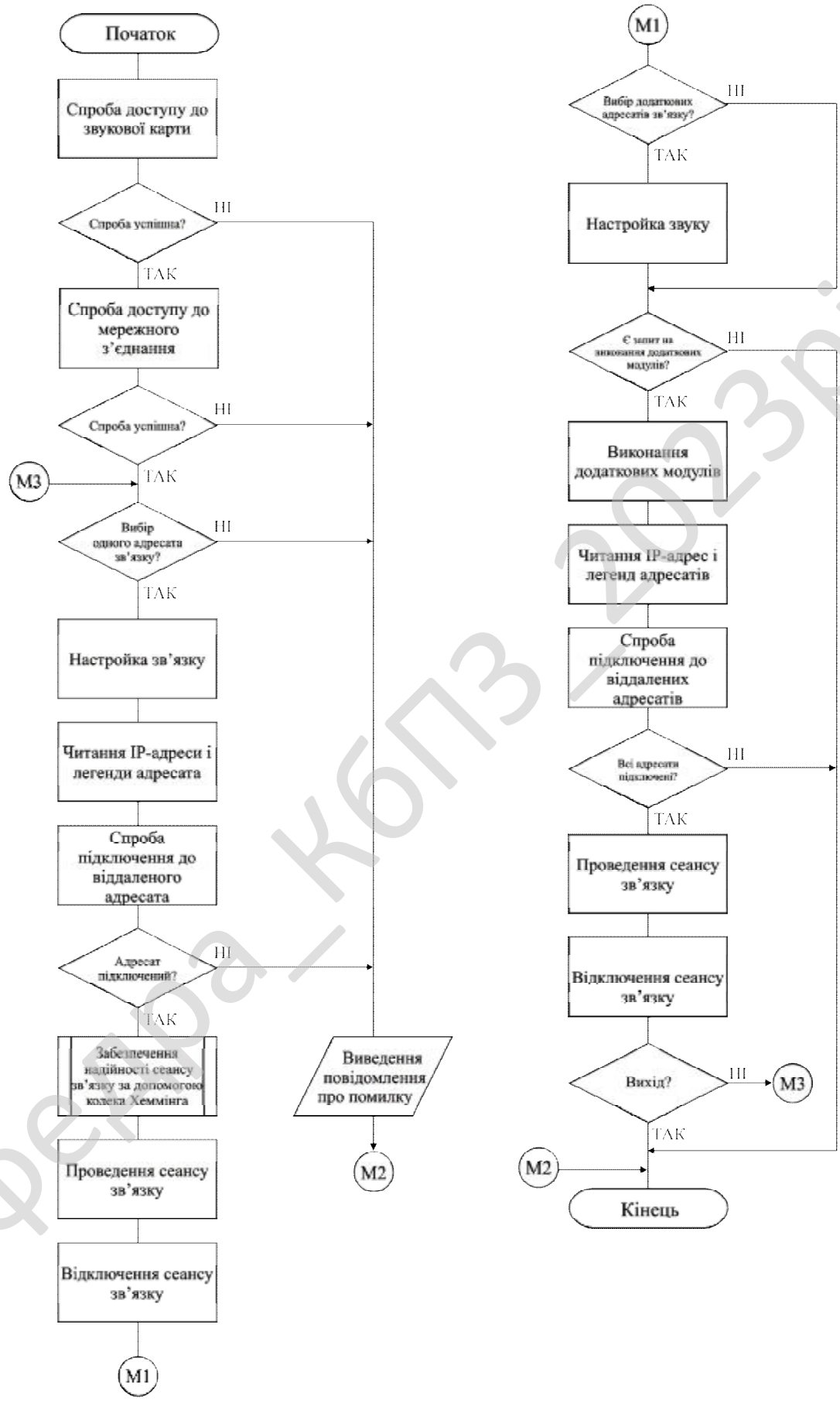

Рисунок 4.2 – Блок-схема підпрограми зв'язку

Інакше, якщо вибір одного адресату зв’язку є успішним, то відбуваються наступні дії:

– Налаштування зв’язку.

– Читання IP-адреси й легенди адресата.

– Спроба підключення до віддаленого адресата.

Проводиться перевірка на те, підключений, або ні адресат.

Якщо він не підключений, то виводиться повідомлення про помилку, й програма завершує свою роботу.

У іншому випадку, тобто, якщо адресат підключений, то відбувається виклик підпрограми забезпечення надійності сеансу зв’язку за допомогою кодека Хеммінга.

Після цього відбувається проведення сеансу зв’язку.

Коли сеанс зв’язку завершується, то відбувається відключення сеансу зв’язку.

Після цього відбувається перевірка на наявність додаткових адресатів зв’язку.

Якщо вони є, то відбувається налаштування звуку.

У іншому випадку, відбувається перевірка запиту на виконання додаткових модулів.

Якщо цей запит відбувається то виконуються наступні дії:

– Виконання додаткових модулів.

– Читання IP-адрес і легенд адресатів.

– Спроба підключення до віддалених адресатів.

У іншому випадку відбувається завершення програми.

Після перевірки запиту на виконання додаткових модулів, відбувається перевірка запиту, на те чи всі адресати підключені.

Якщо так, то виконуються наступні дії:

– Проведення сеансу зв’язку.

– Відключення сеансу зв’язку.

У іншому випадку відбувається завершення програми.

Після виконання усіх вище перерахованих дій, відбувається перевірка на те, хоче, або ні користувач вийти з програми.

Якщо так, то програма завершує свою роботу.

У іншому випадку, відбувається перехід до вибору одного адресату зв’язку.

На рисунку 4.3 зображена блок-схема підпрограми реалізації кодування/декодування Хеммінга.

Вона працює наступним чином.

Спершу обирається варіант:

– Кодувати.

– Декодувати.

Якщо обрано кодувати, то відбувається формування перевірочних символів.

Ці символи додаються до інформаційних.

Сформований пакет передається у мережу.

Якщо ж обрано функцію декодувати, то спершу відбувається обчислення синдрому помилки.

Якщо синдром помилки дорівнює нулю, то видаляються перевірочні біти, й вважається, що повідомлення прийняте правильно.

У іншому випадку відбуваються наступні дії:

– Визначається місцезнаходження помилки.

– Відбувається інвертування біту, у якому знайдено помилку.

Таким чином повідомлення перетворюється на правильне.

Нижче приведемо частини коду, які відповідають за виконання деяких функцій розробленої програми.

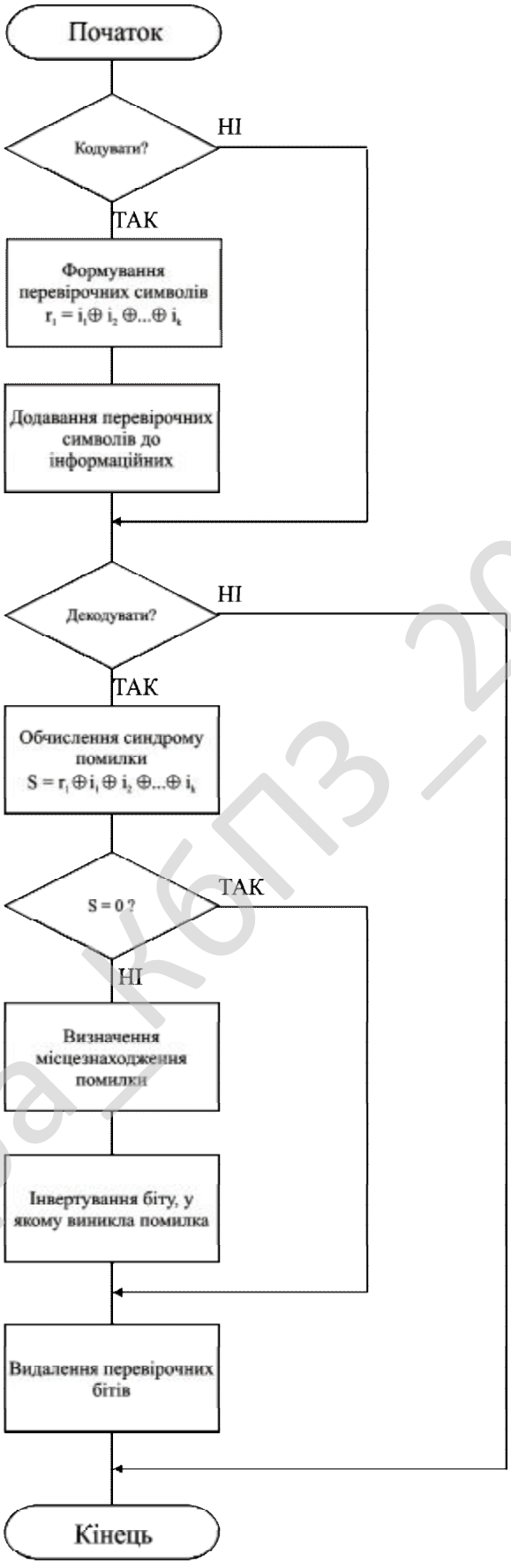

Рисунок 4.3 – Блок-схема підпрограми реалізації кодування/декодування Хеммінга

Почнемо з підпрограми організації зв’язку.

```
{Підпрограма додавання IP-адреси користувача – працівника банку}
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  InputString: string;
begin
  InputString:=  InputBox('Додавання  ip  адреси',  'Прошу',  'Введення  IP  не
здійснено');
  if (InputString<>'Введення IP не здійснено') then
  begin
  Listbox1.Items.add(InputString);
  end;
end;
{Підпрограма додавання легенди IP-адреси користувача – працівника банку}
procedure TForm1.BitBtn3Click(Sender: TObject);
var
  InputString: string;
begin
 InputString:= InputBox('Додавання легенди ip адреси', 'Прошу', 'Введення легенди
не здійснено');
 if (InputString<>'Введення легенди не здійснено') then
 begin
 Listbox2.Items.add(InputString);
 end;
end;
```

Наведемо програмний код блоку визначення повноважень та прав доступу працівника банківської установи до системи IP-зв’язку.

```
if Crypt(InputBox('Увага!','Введіть старий пароль',''),KEY,false)=S then
    begin
        H:=Crypt(InputBox('Увага!','Введіть новий пароль',''),KEY,false);
        if H<>'' then
        begin
         MessageDlg('Дякую пароль змінено',mtInformation,[mbOK],0);
         AssignFile(F, 'main.dat');
          Rewrite(F);
          Writeln(F,H);
          CloseFile(F);
        end else  MessageDlg('Введіть значення!',mtInformation,[mbOK],0);
    end
    else
     begin
```

```
        MessageDlg('Файл main.dat не знайдено чи пароль невірний! завершення програми',mtInformation,[mbOK],0);
        Application.Terminate;
      end;
```

І на завершення наведемо програмний код видачі інформації про розробника програмного продукту.

```
procedure TForm5.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('БАКАЛАВРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Коваленко А.С.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Буза В.В.');
Memo1.Lines.Add('                    гр. КІ-20-3СК');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SEED – у криптографії симетричний блоковий криптоалгоритм на основі Мережі Фейстеля, розроблений Корейським агентством інформаційної безпеки (Korean Information Security Agency, KISA) в 1998 році. В алгоритмі використовується 128-бітний блок і ключ довжиною 128 біт. Алгоритм одержав широке поширення й використовується фінансовими й банківськими структурами, виробничими підприємствами й бюджетними

установами Південної Кореї, оскільки 40-бітний SSL не забезпечує на даний момент мінімально необхідного рівня безпеки. Агентством по захисту інформації специфіковане використання шифру SEED у протоколах TLS і S/MIME. У той же час, алгоритм SEED не реалізований у більшості сучасних браузерів і інтернет-додатків, що утрудняє його використання в даній сфері поза межами Південної Кореї.

SEED являє собою Мережа Фейстеля з 16 раундами, 128-бітовими блоками й 128-бітовим ключем. Алгоритм використовує дві 8×8 таблиці підстановки, які, як такі з Safer, виведені з дискретного зведення в ступінь (у цьому випадку, x^{247} і x^{251} – плюс деякі «несумісні операції»). Це є деякою подібністю с MISTY1 у рекурсивності його структури: 128-бітовий повний шифр – мережа Фейстеля з F-функцією, що впливає на 64-бітові половини, у той час як сама F-функція – Мережа Фейстеля, складена з G-функції, що впливає на 32-розрядні половини. Однак рекурсія не простягнеться далі, тому що G-функція – не Мережа Фейстеля. В G-функції 32-розрядне слово розглядають як чотири 8-бітових байта, кожний з яких проходить через одну або іншу таблицю підстановки, потім поєднується в помірковано комплексному наборі булевих функцій таким чином, що кожний біт виводу залежить від 3 з 4 вхідних байтів.

SEED має складний ключовий розклад, генеруючи тридцять два 32-розрядних додаткових символу, використовуючи G-функції на серіях обертань вихідного неопрацьованого ключа, комбінованого зі спеціальними раундовими константами (як в TEA) від «Золотого співвідношення» (англ. Golden ratio).

Згідно з дослідженнями KISA, алгоритм SEED «надійно протистоїть відомим атакам».

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

У цьому розділі наведемо методику впровадження розробленого програмного продукту у промислову експлуатацію.

Програмний продукт призначений для організації IP-зв'язку з підвищеною надійністю у банківській мережі.

У якості засобу підвищення надійності використовується кодек Хеммінга.

При запуску програми з'являється заставка наведена на рисунку 5.1.

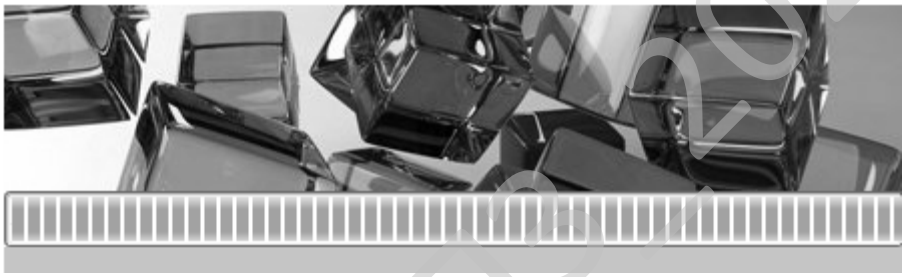

Рисунок 5.1 – Заставка

Після зникнення заставки, під час виведення якої відбувається завантаження кодексів, та підключаємих модулів, програма виводить головне вікно роботи користувача, зображене на рисунку 5.2, у якому існують наступні поля:

– IP-адреси.

– Легенда IP-номерів.

– IP-виклик користувача.

– Вікно, у якому вказується, з ким спілкується користувач.

Крім того у головному вікні наведені кнопки, які дозволяють виконувати наступні дії:

– Виклик.

– Кінець виклику.

– Селективний зв'язок.

– Додати або видалити IP-адреси й ленеди IP-номерів.

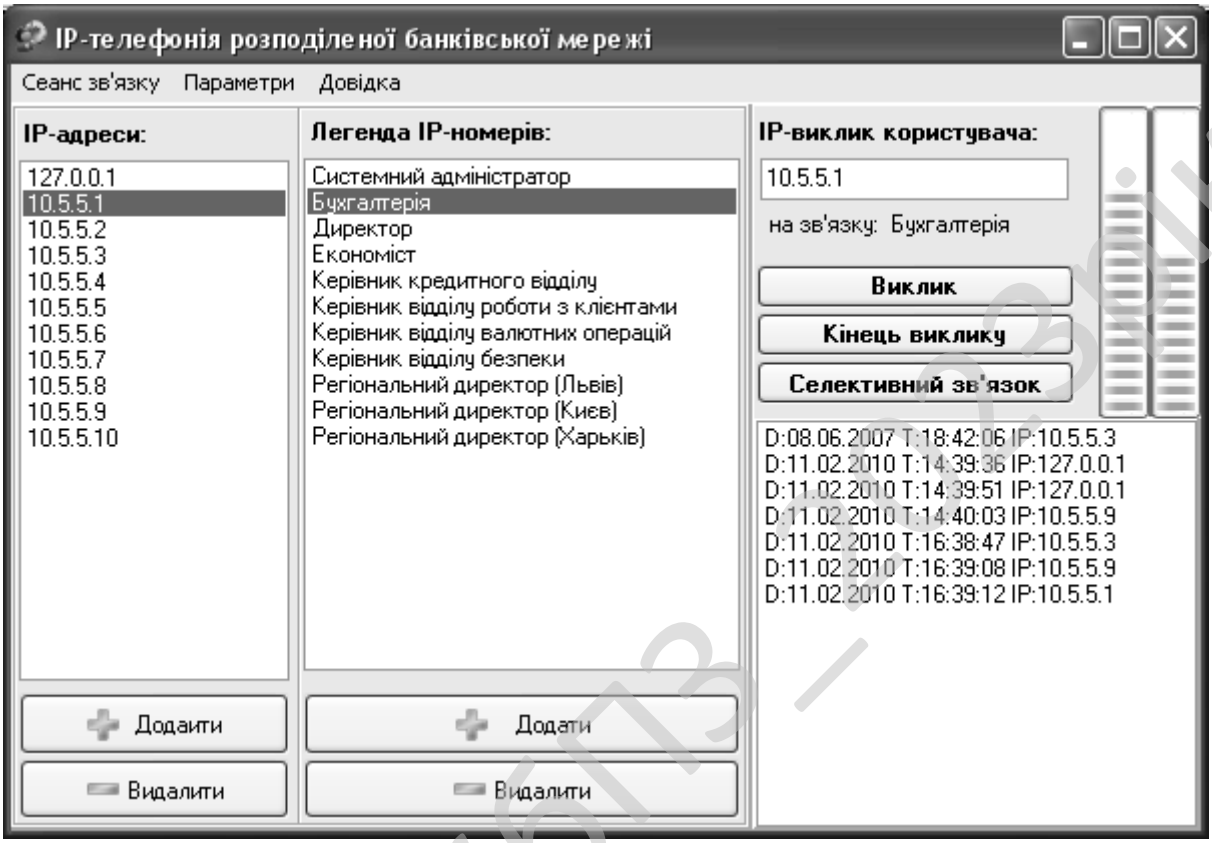

Рисунок 5.2 – Головне вікно програми

На рисунку 5.3 зображено вікно селективного зв’язку. З нього ми бачимо, що існує можливість встановити зв'язок, закінчити виклик, та приховати вікно.

Крім того існує поле, у якому відображаються усі IP-адреси учасників селективного зв’язку.

Інформація про розробника програмного продукту наведена на рисунку 5.4.

Рисунок 5.3 – Селективний зв’язок

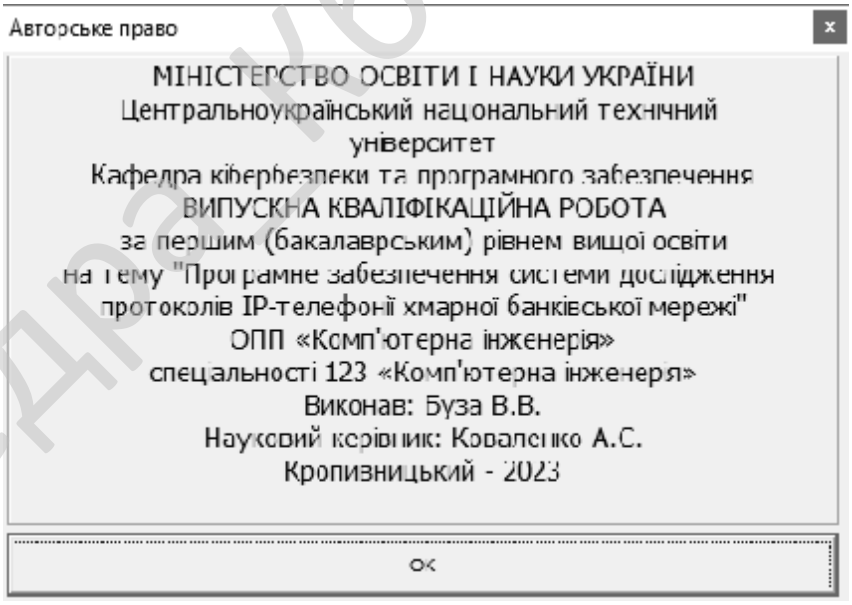

Рисунок 5.4 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи дослідження протоколів IP-телефонії хмарної банківської мережі.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем дослідження протоколів IP-телефонії хмарної банківської мережі.

– Досліджена система дослідження протоколів IP-телефонії хмарної банківської мережі.

– На основі отриманих результатів досліджень створена програмна реалізація системи дослідження протоколів IP-телефонії хмарної банківської мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання дослідження протоколів IP-телефонії хмарної банківської мережі.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи дослідження протоколів IP-телефонії хмарної банківської мережі. Це

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SEED.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

2. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

3. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

4. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

5. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science,* Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w **(Scopus).**

6. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T),* 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

7. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

8. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

9. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256. **(Scopus)**.

10. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings.* Volume 2740, 2020, Pages 102-114. **(Scopus)**.

11. Smirnov O.A., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346. **(Scopus)**.

12. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

13. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14. **(Scopus)**.

14. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

15. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. **Springer**, Cham. 2021. pp 557-587. **(Scopus)**.

16. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136. **(Scopus)**.

17. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379. **(Scopus)**.

18. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43. **(Scopus)**.

19. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645. **(Scopus)**.

20. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660., **(Scopus)**.

21. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407. (**Scopus)**.

22. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

23. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019. **(Scopus)**.

24. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629. **(Scopus)**.

25. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures.* ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

27. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures**. Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

28. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. **Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

29. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

30. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.

31. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

32. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

33. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98. 2022.

34. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

35. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

36. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного

процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

37. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

38. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

39. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.

40. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

41. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

42. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

43. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

44. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

45. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

46. Смірнов О.А., Смірнова Т.В., Солових Є.К., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

47. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.

48. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

49. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

50. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-123.23.0010.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Буза В.В.			Програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Коваленко А.С.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20-3СК	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи дослідження протоколів IP-телефонії хмарної банківської мережі.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 8-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи дослідження протоколів IP-телефонії хмарної банківської мережі;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 89 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 8.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Коваленко А.С.

Програмне забезпечення системи дослідження протоколів IP-телефонії хмарної банківської мережі

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 51

Літера: РП

Кропивницький – 2023 року

iptel.pas – Програма організації зв`язку

```
program iptel;
uses
  Forms,
  Dialogs,
  Windows,
  Sysutils,
  Messages,
  usimple in 'usimple.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2},
  Unit3 in 'Unit3.pas' {Form3},
  Unit1 in 'Unit1.pas' {Form0},
  Unit4 in 'Unit4.pas' {Form4};
{$R simple.RES}
var
  i:integer;
{        function Crypt(Text,Key: String; Encode: boolean): String;
        var
          i, KeyLength: integer;
          Sign: ShortInt;
        begin
         KeyLength:=Length(Key);
         if Encode then Sign :=-1 else Sign:=1;
         for i:=1 to Length(Text) do
          Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
          Result:=Text;
        end;}
begin
Application.Initialize;
{
if FileExists('main.dat')=false then
 begin
    MessageDlg('Файл       main.dat       не       знайдено!       завершення
програми',mtInformation,[mbOK],0);
    Application.Terminate;
  end else
  begin
    AssignFile(F, 'main.dat');
    Reset(F);
    Readln(F, S);
    CloseFile(F);
       if Crypt(InputBox('Увага!','Введіть пароль',Y),KEY,false)=S then
       begin
            MessageDlg('Дякую, пароль вірний!', mtInformation,[mbOK],0);}
             Try
```

```
                Form2:=TForm2.Create(Application);
                Form2.Show;
                Form2.Update;
                for i:=1 to 10 do
                begin
                  sleep(200);
                  Form2.ProgressBar1.Position:=i*10;
                  Form2.Update;
                end;
              Application.Title := 'IpTel';
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm0, Form0);
  Application.CreateForm(TForm4, Form4);
  Finally
                Form2.Free;
   End;
     Application.Run;
{      end
      else
      begin
      MessageDlg('Невірний пароль!',mtInformation,[mbOK],0);
      Application.Terminate;
      end;
  end;}
end.
```

U_SPLASH.pas – Підпрограма завантаження заставки

```
unit U_SPLASH;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, jpeg, ExtCtrls, StdCtrls, Gauges;

type
  TForm_SPLASH = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Gauge1: TGauge;
    Timer1: TTimer;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_SPLASH: TForm_SPLASH;

implementation

{$R *.dfm}

end.
```

usimple.pas – Головне вікно програми

```
unit usimple;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  telefoon, StdCtrls, ComCtrls, ExtCtrls, jpeg, Menus, Buttons;

type
  TForm1 = class(TForm)
    Telefoon1: TTelefoon;
    StatusBar1: TStatusBar;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    Panel1: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel2: TPanel;
    Label1: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Button2: TButton;
    Button1: TButton;
    Panel5: TPanel;
    TabSheet2: TTabSheet;
    Panel6: TPanel;
    Label2: TLabel;
    Panel7: TPanel;
    MainMenu1: TMainMenu;
    Panel8: TPanel;
    ListBox1: TListBox;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    ListBox2: TListBox;
    Panel9: TPanel;
    Label3: TLabel;
    Panel10: TPanel;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    N1: TMenuItem;
    IP1: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
```

```
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    BitBtn5: TBitBtn;
    Memo1: TMemo;
    TabSheet3: TTabSheet;
    Panel11: TPanel;
    Image3: TImage;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    N11: TMenuItem;
    N12: TMenuItem;
    ListBox3: TListBox;
    Label9: TLabel;
    Label10: TLabel;
    N2: TMenuItem;
    PB1: TProgressBar;
    PB2: TProgressBar;
    T1: TTimer;
    Image1: TImage;
    Timer1: TTimer;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure ListBox2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure N10Click(Sender: TObject);
    procedure BitBtn5Click(Sender: TObject);
    procedure N12Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure N3Click(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure T1Timer(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit3, Unit2, Unit4, Unit1;
var
  KEY:string='2%#$T%&(*qwrdas@@@#45';
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
begin
 Telefoon1.placecall(Edit1.text);
 ListBox3.items.add('D:'+DateToStr(now)+'                T:'+TimeToStr(now)+'
IP:'+Edit1.text);
  T1.Enabled:=true;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
 Telefoon1.calling:=false;
 T1.Enabled:=false;
 PB1.Position:=0;
 PB2.Position:=0;
end;
procedure TForm1.ListBox1Click(Sender: TObject);
begin
 Edit1.text:=ListBox1.Items.Strings[ListBox1.ItemIndex];
end;
procedure TForm1.ListBox2Click(Sender: TObject);
begin
 Label4.Caption:=ListBox2.Items.Strings[ListBox2.ItemIndex];
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  InputString: string;
begin
  InputString:= InputBox('Додавання ip адреси', 'Прошу', 'Введення IP не
здійснено');
  if (InputString<>'Введення IP не здійснено') then
  begin
  Listbox1.Items.add(InputString);
  end;
end;
procedure TForm1.BitBtn3Click(Sender: TObject);
```

```
var
  InputString: string;
begin
 InputString:= InputBox('Додавання легенди ip адреси', 'Прошу', 'Введення
легенди не здійснено');
 if (InputString<>'Введення легенди не здійснено') then
 begin
 Listbox2.Items.add(InputString);
 end;
end;
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
 ListBox1.Items.Strings[ListBox1.ItemIndex]:='';
 ListBox1.Items.Delete(ListBox1.ItemIndex);
end;
procedure TForm1.BitBtn4Click(Sender: TObject);
begin
 ListBox2.Items.Strings[ListBox1.ItemIndex]:='';
 ListBox2.Items.Delete(ListBox1.ItemIndex);
end;
procedure TForm1.N10Click(Sender: TObject);
begin
 Application.Terminate;
end;
procedure TForm1.BitBtn5Click(Sender: TObject);
var
 i:integer;
 G:boolean;
begin
 Form3.ListBox1.Clear;
 g:=false;
 for i:=0 to (ListBox1.Items.Count - 1) do
 begin
  if ListBox1.Selected[i] then
    begin
        Form3.ListBox1.Items.add(ListBox1.Items.Strings[i]);
        g:=true;
    end;
    if g then Form3.show;
 end;
end;
procedure TForm1.N12Click(Sender: TObject);
var
  F: TextFile;
  H,S:string;
  OLD:string;
```

```
        function Crypt(Text,Key: String; Encode: boolean): String;
        var
          i, KeyLength: integer;
          Sign: ShortInt;
        begin
         KeyLength:=Length(Key);
         if Encode then Sign :=-1 else Sign:=1;
         for i:=1 to Length(Text) do
          Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
          Result:=Text;
        end;
begin
if FileExists('main.dat') then
begin
    AssignFile(F, 'main.dat');
    Reset(F);
    Readln(F, S);
     CloseFile(F);
  if Crypt(InputBox('Увага!','Введіть старий пароль',''),KEY,false)=S then
    begin
        H:=Crypt(InputBox('Увага!','Введіть новий пароль',''),KEY,false);
        if H<>'' then
        begin
          MessageDlg('Дякую пароль змінено',mtInformation,[mbOK],0);
          AssignFile(F, 'main.dat');
          Rewrite(F);
          Writeln(F,H);
          CloseFile(F);
        end else  MessageDlg('Введіть значення!',mtInformation,[mbOK],0);
    end
    else
     begin
        MessageDlg('Файл main.dat не знайдено чи пароль невірний! завершення
програми',mtInformation,[mbOK],0);
         Application.Terminate;
      end;
end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
randomize;
PB1.Position:=0;
PB2.Position:=0;
T1.Enabled:=false;
Timer1.Enabled:=true;
if FileExists('MainLegend.dat')=false then
```

```
 begin
    MessageDlg('Файл MainLegend.dat не знайдено!',mtInformation,[mbOK],0);
 end else
 begin
    ListBox2.Items.LoadFromFile('MainLegend.dat');
  end;
if FileExists('MainIP.dat')=false then
 begin
    MessageDlg('Файл MainIP.dat не знайдено!',mtInformation,[mbOK],0);
 end else
 begin
    ListBox1.Items.LoadFromFile('MainIP.dat');
  end;
if  FileExists('MainHISTORY.dat')=false  then  MessageDlg('Файл  MainHISTORY.dat
не знайдено!',mtInformation,[mbOK],0)
else ListBox3.Items.LoadFromFile('MainHISTORY.dat');
if  FileExists('mainHelp.dat')=false  then  MessageDlg('Файл  mainHelp.dat  не
знайдено!',mtInformation,[mbOK],0)
else Memo1.Lines.LoadFromFile('mainHelp.dat');
end;
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
if FileExists('MainLegend.dat')=false then MessageDlg('Файл MainLegend.dat не
знайдено!',mtInformation,[mbOK],0)
else ListBox2.Items.SaveToFile('MainLegend.dat');
if   FileExists('MainIP.dat')=false   then   MessageDlg('Файл   MainIP.dat   не
знайдено!',mtInformation,[mbOK],0)
else ListBox1.Items.SaveToFile('MainIP.dat');
if  FileExists('MainHISTORY.dat')=false  then  MessageDlg('Файл  MainHISTORY.dat
не знайдено!',mtInformation,[mbOK],0)
else ListBox3.Items.SaveToFile('MainHISTORY.dat');
end;
procedure TForm1.N3Click(Sender: TObject);
begin
TabSheet2.Visible:=true;
end;
procedure TForm1.N6Click(Sender: TObject);
begin
 Telefoon1.calling:=false
end;
procedure TForm1.N2Click(Sender: TObject);
begin
Form0.show;
end;
procedure TForm1.FormShow(Sender: TObject);
begin
```

```
PB1.Position:=0;
PB2.Position:=0;
end;
procedure TForm1.T1Timer(Sender: TObject);
var
i1,i2:integer;
begin
PB1.Position:=random(100);
PB2.Position:=PB1.Position;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
Form4.show;
Form1.hide;
Timer1.Enabled:=false;
end;
end.
```

Unit1.pas – Підпрограма налагодження звукової карти AudioSettings

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ComCtrls, StdCtrls, AMixer, MMSystem;
type
  TForm0 = class(TForm)
    ComboBox1: TComboBox;
    ComboBox2: TComboBox;
    TrackBar: TTrackBar;
    CheckBox: TCheckBox;
    Label1: TLabel;
    Label2: TLabel;
    Mixer: TAudioMixer;
    Label3: TLabel;
    Label4: TLabel;
    ComboBox3: TComboBox;
    LabelStereo: TLabel;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure ComboBox2Change(Sender: TObject);
    procedure MixerControlChange(Sender: TObject; MixerH, ID: Integer);
    procedure TrackBarChange(Sender: TObject);
    procedure CheckBoxClick(Sender: TObject);
    procedure ComboBox3Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
    Setting:Boolean;
  public
    { Public declarations }
  end;
var
  Form0: TForm0;
implementation
uses usimple;
{$R *.DFM}
procedure TForm0.FormCreate(Sender: TObject);
var A:Integer;
begin
  For A := 0 to Mixer.MixerCount - 1 do
```

```
    ComboBox3.Items.Add ('Mixer '+IntToStr(A));
  If (ComboBox3.Items.Count > 0) then
    ComboBox3.ItemIndex := 0;
  ComboBox3Change (Sender);
end;
procedure TForm0.ComboBox1Change(Sender: TObject);
var A:Integer;
begin
  ComboBox2.Items.Clear;                                    ComboBox2.Items.Add
(Mixer.Destinations[ComboBox1.ItemIndex].Data.szName);
  For A:=0 to Mixer.Destinations[ComboBox1.ItemIndex].Connections.Count-1 do
ComboBox2.Items.Add(Mixer.Destinations[ComboBox1.ItemIndex].Connections[A].Dat
a.szName);
  If ComboBox2.Items.Count>0 then
  begin
    ComboBox2.ItemIndex:=0;     ComboBox2Change (Self);
  end;
end;
procedure TForm0.ComboBox2Change(Sender: TObject);
var L,R,M:Integer;    VD,MD:Boolean;    Stereo:Boolean;
    IsSelect:Boolean;
begin
  Mixer.GetVolume                    (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,L,R,M,Stereo,VD,MD,IsSelect);
  Setting:=True;
  TrackBar.Visible:=not VD;
  Label1.Visible:=not VD;
  Label3.Visible:=VD;
  If TrackBar.Visible then
    TrackBar.Position:=L;
  CheckBox.Visible:=not MD;
  Label2.Visible:=not MD;
  Label4.Visible:=MD;
  If CheckBox.Visible then
    CheckBox.Checked:=M<>0;
  If (Stereo) then
    LabelStereo.Caption := '- stereo -'
  else
    LabelStereo.Caption := '- mono -';
  Setting:=False;
end;
procedure TForm0.MixerControlChange(Sender: TObject; MixerH, ID: Integer);
begin
  ComboBox2Change (Self);
end;
procedure TForm0.TrackBarChange(Sender: TObject);
```

```
begin
  If (not Setting) then
  begin
    Setting:=True;
    Mixer.SetVolume                    (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,TrackBar.Position,TrackBar.Position,Integer(CheckBox.Checked));
    Setting:=False;
  end;
end;
procedure TForm0.CheckBoxClick(Sender: TObject);
begin
  If not Setting then
  begin
    Setting:=True;
    Mixer.SetVolume                    (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,TrackBar.Position,TrackBar.Position,Integer(CheckBox.Checked));
    Setting:=False;
  end;
end;
procedure TForm0.ComboBox3Change(Sender: TObject);
var A:Integer;
begin
  If (ComboBox3.ItemIndex >= 0) AND (ComboBox3.ItemIndex < Mixer.MixerCount)
then
    Mixer.MixerId := ComboBox3.ItemIndex;
  ComboBox1.Items.Clear;
  If Mixer.MixerCount>0 then
  begin
    For A:=0 to Mixer.Destinations.Count-1 do
      ComboBox1.Items.Add (Mixer.Destinations[A].Data.szName);
    If ComboBox1.Items.Count>0 then
    begin
      ComboBox1.ItemIndex:=0;    ComboBox1Change (Self);
    end;
  end
  else
  begin
    ComboBox1.OnChange:=nil;     ComboBox2.OnChange:=nil;
    TrackBar.OnChange:=nil;       CheckBox.OnClick:=nil;
    MessageDlg ('У системі немає звукового міксера!',mtError,[mbOK],0);
  end;
  Setting:=False;
end;
procedure TForm0.Button1Click(Sender: TObject);
begin
Form0.hide;  Form1.show;
```

```
end;
procedure TForm0.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form0.hide; Form1.show;
end;
end.
```

Unit2.pas – Підпрограма виклику вікна налагодження звукової карти AudioSettings

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  jpeg, ExtCtrls, StdCtrls, ComCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    RichEdit1: TRichEdit;
    ProgressBar1: TProgressBar;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.DFM}

end.
```

Unit3.pas – Підпрограма селективного зв`язку

```
unit Unit3;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, jpeg, ComCtrls;

type
  TForm3 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    ListBox1: TListBox;
    Animate1: TAnimate;
    Panel4: TPanel;
    BitBtn2: TBitBtn;
    BitBtn1: TBitBtn;
    BitBtn3: TBitBtn;
    Image1: TImage;
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

uses usimple;

{$R *.DFM}

procedure TForm3.BitBtn2Click(Sender: TObject);
begin
form3.hide;
end;
```

```
procedure TForm3.BitBtn1Click(Sender: TObject);
begin
Animate1.Active:=true;
form1.telefoon1.placecall(form1.edit1.text);
end;

procedure TForm3.BitBtn3Click(Sender: TObject);
begin
Form1.Telefoon1.calling:=false;
Animate1.Active:=false;
end;

end.
```

Unit4.pas – Підпрограма парольного захисту

```
unit Unit4;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, Mask, ExtCtrls;
type
  TForm4 = class(TForm)
    Panel1: TPanel;
    M: TMaskEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form4: TForm4;
implementation
uses usimple;
VAR
  KEY:string='2%#$T%&(*qwrdas@@@#45';
  DAT:string='VUW';
{$R *.DFM}
        function Crypt(Text,Key: String; Encode: boolean): String;
        var
          i, KeyLength: integer;
          Sign: ShortInt;
        begin
         KeyLength:=Length(Key);
         if Encode then Sign :=-1 else Sign:=1;
         for i:=1 to Length(Text) do
          Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
          Result:=Text;
        end;
procedure TForm4.BitBtn1Click(Sender: TObject);
var
  F: TextFile;
  i:integer;
  s:string;
  Y:string;
  UUU:string;
begin
```

```
if FileExists('main.dat')=TRUE then
 begin
   AssignFile(F, 'main.dat');
   Reset(F);
   Readln(F, S);
   CloseFile(F);
UUU:=Crypt(M.text,KEY,false);
if UUU=S then
      begin
           Form4.hide;
           Form1.show;
           MessageDlg('Дякую, пароль вірний!',mtInformation,[mbOK],0);
      end
      else MessageDlg('Введіть пароль!',mtInformation,[mbOK],0);
 end
      else       begin
    MessageDlg('Файл       main.dat       не       знайдено!       завершення
програми',mtInformation,[mbOK],0);
    Application.Terminate;
      end;
end;
end.
```

AMixer.pas – Розроблений компонент налагодженя звука

```
unit AMixer;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  MMSystem;

type
  TAudioMixer=class;

  TPListFreeItemNotify=procedure (Pntr:Pointer) of object;
  TMixerChange=procedure (Sender:TObject;MixerH:HMixer;ID:Integer) of object;
    {MixerH is handle of mixer, which sent this message.
     ID is ID of changed item (line or control).}

  TPointerList=class(TObject)
  private
    FOnFreeItem:TPListFreeItemNotify;
    Items:Tlist;
  protected
    function GetPointer (Ind:Integer):Pointer;
    function GetCount :integer;
  public
    constructor Create;
    destructor Destroy; override;
    procedure Clear;
    procedure Add (Pntr:Pointer);
    property Count:Integer read GetCount;
    property Pointer[Ind:Integer]:Pointer read GetPointer; default;
    property   OnFreeItem:TPListFreeItemNotify   read   FOnFreeItem   write
FOnFreeItem;
  end;

  TMixerControls=class(TObject)
  private
    heap:pointer;
    FControls:TPointerList;
  protected
    function GetControl (Ind:Integer):PMixerControl;
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
```

```
    destructor Destroy; override;
    property Control[Ind:Integer]:PMixerControl read GetControl; default;
    property Count:Integer read GetCount;
  end;

  TMixerConnection=class(TObject)
  private
    XMixer:TAudioMixer;
    FData:TMixerLine;
    FControls:TMixerControls;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property Controls:TMixerControls read FControls;
    property Data:TMixerLine read FData;
  end;

  TMixerConnections=class(TObject)
  private
    XMixer:TAudioMixer;
    FConnections:TPointerList;
  protected
    procedure DoFreeItem (Pntr:Pointer);
    function GetConnection (Ind:Integer):TMixerConnection;
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property  Connection[Ind:Integer]:TMixerConnection  read  GetConnection;
default;
    property Count:Integer read GetCount;
  end;

  TMixerDestination=class(TObject)
  private
    XMixer:TAudioMixer;
    FData:TMixerLine;
    FControls:TMixerControls;
    FConnections:TMixerConnections;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property Connections:TMixerConnections read FConnections;
    property Controls:TMixerControls read FControls;
    property Data:TMixerLine read FData;
  end;
```

```
  TMixerDestinations=class(TObject)
  private
    FDestinations:TPointerList;
  protected
    function GetDestination (Ind:Integer):TMixerDestination;
    procedure DoFreeItem (Pntr:Pointer);
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer);
    destructor Destroy; override;
    property Count:Integer read GetCount;
    property  Destination[Ind:Integer]:TMixerDestination  read  GetDestination;
default;
  end;

  TAudioMixer = class(TComponent)
  private
    XWndHandle:HWnd;

    FDestinations:TMixerDestinations;
    FMixersCount:Integer;
    FMixerHandle:HMixer;
    FMixerId:Integer;
    FMixerCaps:TMixerCaps;
    FDriverVersion: MMVERSION;
    FManufacturer: String;
    FProductId: Word;
    FNumberOfLine: Integer;
    FProductName: String;
    FOnLineChange:TMixerChange;
    FOnControlChange:TMixerChange;
  protected
    procedure SetMixerId (Value:Integer);
    procedure MixerCallBack (var Msg:TMessage);
    procedure CloseMixer;
  published
    constructor Create (AOwner:TComponent); override;
    destructor Destroy; override;
    property DriverVersion: MMVERSION read FDriverVersion;
    property ProductId: WORD read FProductId;
    property NumberOfLine: Integer read FNumberOfLine;
    property Manufacturer: string read FManufacturer;
    property ProductName: string read FProductName;
    property MixerId:Integer read FMixerId write SetMixerId;
      {Opened mixer - value must be in range 0..MixersCount-1
```

```
      If no mixer is opened this value is  -1}
    property OnLineChange:TMixerChange read FOnLineChange write FOnLineChange;
    property  OnControlChange:TMixerChange  read  FOnControlChange  write
FOnControlChange;
  public
    function  GetVolume  (ADestination,  AConnection:Integer;  var  LeftVol,
RightVol,    Mute:Integer;    var    Stereo,    VolDisabled,    MuteDisabled,
MuteIsSelect:Boolean):Boolean;
      {Ця функція повертає значення Destination and Connection.
       ADestination must be from range 0..Destinations.Count-1
       AConnection           must           be           in           range
0..Destinations[ADestination].Connections.Count-1
       If you want to read master volume of some Destination, you have to
         set AConnection to -1.
       If LeftVol, RightVol or Mute is not supported by queried connection,
         it's return value will be -1.

       LeftVol and RightVol are in range 0..65536

       If Mute is non-zero then the connection is silent (or vice-versa - see
MuteIsSelect parameter)
       If specified line is recording source then Mute specifies if programs
will
         record from this connection (it is copy of "Select" Checkbox in
         standard Windows Volume Control program)
       Stereo is true, then this control is stereo.
       VolDisabled or MuteDisabled is True when you cannot apply settings to
this
         control (but can read it).
       MuteIsSelect returns True is "mute" work here as select - opposite of
mute.

       Return value of the function is True if no error has occured,
         otherwise it returns False.}
    function SetVolume (ADestination, AConnection:Integer; LeftVol, RightVol,
Mute:Integer):Boolean;
      {This function sets volume.
       If you set RightVol to -1 and connection is stereo then LeftVol will be
         copied to RightVol.
       If LeftVol or Mute is -1 then this value will not be set.
       Note  that  "Mute"  can  be  "select"  (which  is  reversed  mute)  -  see
function
         GetVolume, parameter MuteIsSelect.

       Return  value  is  True  if  ADestination  and  AConnection  are  correct,
otherwise False.}
```

```
    function GetPeak(ADestination, AConnection:Integer; var LeftPeak,
RightPeak:Integer):Boolean;
    function GetMute(ADestination, AConnection:Integer; var
Mute:Boolean):Boolean;
    function SetMute(ADestination, AConnection:Integer; Mute:Boolean):Boolean;

    property Destinations:TMixerDestinations read FDestinations;
      {Ind must be in range 0..DestinationsCount-1}
    property MixerCaps:TMixerCaps read FMixerCaps;
    property MixerCount:Integer read FMixersCount;
      {Number of mixers present in system; mostly 1}
    property MixerHandle:HMixer read FMixerHandle;
      {Handle of opened mixer}
  end;

procedure Register;

implementation

{------------}
{TPointerList}
{------------}

constructor TPointerList.Create;
begin
  Items := TList.Create;
end;

destructor TPointerList.Destroy;
begin
  Clear;
  Items.Free;
end;

procedure TPointerList.Add (Pntr:Pointer);
begin
  Items.Add (Pntr);
end;

function TPointerList.GetPointer (Ind:Integer):Pointer;
begin
  Result := nil;
  If (Ind < Count) then
    Result := Items[Ind];
end;
```

```
procedure TPointerList.Clear;
var I:Integer;
begin
  for I := 0 to Items.Count-1 do begin
    If Assigned (FOnFreeItem) then
      FOnFreeItem (Items[I])
  end;
  Items.Clear;
end;

function TPointerList.GetCount:Integer;
begin
  Result := Items.Count;
end;

{--------------}
{TMixerControls}
{--------------}
constructor TMixerControls.Create (AMixer:TAudioMixer; AData:TMixerLine);
var MLC:TMixerLineControls;
    A,B:Integer;
    P:PMixerControl;
begin
  FControls := TPointerList.Create;
  GetMem (P, SizeOf(TMixerControl)*AData.cControls);
  heap := P;
  MLC.cbStruct := SizeOf(MLC);
  MLC.dwLineID := AData.dwLineID;
  MLC.cbmxctrl := SizeOf(TMixerControl);
  MLC.cControls := AData.cControls;
  MLC.pamxctrl := P;
  A          :=          MixerGetLineControls(AMixer.MixerHandle,          @MLC,
MIXER_GETLINECONTROLSF_ALL);
  If A = MMSYSERR_NOERROR then
  begin
   For B := 0 to AData.cControls-1 do
    begin
      FControls.Add (P);
      P := PMixerControl (DWORD(P) + sizeof (TMixerControl));
    end;
  end;
end;

destructor TMixerControls.Destroy;
begin
```

```
  FControls.free;
  freemem(heap);
  inherited;
end;

function TMixerControls.GetControl (Ind:Integer):PMixerControl;
begin
  Result := FControls.Pointer[Ind];
end;

function TMixerControls.GetCount:Integer;
begin
  Result := FControls.Count;
end;

{----------------}
{TMixerConnection}
{----------------}

constructor TMixerConnection.Create (AMixer:TAudioMixer; AData:TMixerLine);
begin
  FData := AData;
  XMixer := AMixer;
  FControls := TMixerControls.Create (AMixer, AData);
end;

destructor TMixerConnection.Destroy;
begin
  FControls.Free;
  inherited;
end;

{----------------}
{TMixerConnections}
{----------------}

constructor TMixerConnections.Create (AMixer:TAudioMixer; AData:TMixerLine);
var A,B:Integer;
    ML:TMixerLine;
begin
  XMixer := AMixer;
  FConnections := TPointerList.Create;
  FConnections.OnFreeItem := Dofreeitem;
  ML.cbStruct := SizeOf(TMixerLine);
  ML.dwDestination := AData.dwDestination;
  For A := 0 to AData.cConnections-1 do
```

```
  begin
    ML.dwSource := A;
    B        :=        MixerGetLineInfo        (AMixer.MixerHandle,        @ML,
MIXER_GETLINEINFOF_SOURCE);
    If B = MMSYSERR_NOERROR then
      FConnections.Add (Pointer(TMixerConnection.Create (XMixer, ML)));
  end;
end;

destructor TMixerConnections.Destroy;
begin
  FConnections.Free;
  inherited;
end;

procedure TMixerConnections.DoFreeItem (Pntr:Pointer);
begin
  TMixerConnection(Pntr).Free;
end;

function TMixerConnections.GetConnection (Ind:Integer):TMixerConnection;
begin
  Result := FConnections.Pointer[Ind];
end;

function TMixerConnections.GetCount:Integer;
begin
  Result := FConnections.Count;
end;

{-----------------}
{TMixerDestination}
{-----------------}

constructor TMixerDestination.Create (AMixer:TAudioMixer; AData:TMixerLine);
begin
  FData := AData;
  XMixer := AMixer;
  FConnections := TMixerConnections.Create (XMixer, FData);
  FControls := TMixerControls.Create (XMixer, AData);
end;

destructor TMixerDestination.Destroy;
begin
  Fcontrols.Free;
  FConnections.Free;
```

```
  inherited;
end;

{------------------}
{TMixerDestinations}
{------------------}

constructor TMixerDestinations.Create (AMixer:TAudioMixer);
var A,B:Integer;
    ML:TMixerLine;
begin
  FDestinations := TPointerList.Create;
  FDestinations.OnFreeItem := DoFreeItem;
  if (AMixer = nil) then
    Exit;
  For A := 0 to AMixer.MixerCaps.cDestinations-1 do
  begin
    ML.cbStruct := SizeOf(TMixerLine);
    ML.dwDestination := A;
    B        :=        MixerGetLineInfo        (AMixer.MixerHandle,        @ML,
MIXER_GETLINEINFOF_DESTINATION);
    If B = MMSYSERR_NOERROR then
      FDestinations.Add (Pointer(TMixerDestination.Create (AMixer, ML)));
  end;
end;

procedure TMixerDestinations.DoFreeItem (Pntr:Pointer);
begin
  TMixerDestination(Pntr).Free;
end;

destructor TMixerDestinations.Destroy;
begin
  FDestinations.Free;
  inherited;
end;

function TMixerDestinations.GetDestination (Ind:Integer):TMixerDestination;
begin
  Result := nil;
  If (Assigned (FDestinations)) then
    Result := FDestinations.Pointer[Ind];
end;

function TMixerDestinations.GetCount:Integer;
begin
```

```
  Result := FDestinations.Count;
end;

{-----------}
{TAudioMixer}
{-----------}

constructor TAudioMixer.Create (AOwner:TComponent);
begin
  inherited Create (AOwner);
  XWndHandle := AllocateHWnd (MixerCallBack);
  FMixersCount := mixerGetNumDevs;
  FMixerId := -1;
  if (FMixersCount = 0) then
    FDestinations := TMixerDestinations.Create (nil)
  else
  begin
    FDestinations := nil;
    SetMixerId (0);
  end;
end;

destructor TAudioMixer.Destroy;
begin
  CloseMixer;
  if XWndHandle <> 0 then
    DeAllocateHwnd (XWndHandle);
  inherited;
end;

procedure TAudioMixer.CloseMixer;
begin
  If FMixerId >= 0 then
  begin
    mixerClose (FMixerHandle);
    FMixerId := -1;
  end;
  FDestinations.Free;
  FDestinations := nil;
end;

procedure TAudioMixer.SetMixerId (Value:Integer);
label AllOK;
begin
  If (Value < 0) OR (Value >= FMixersCount) then
    Exit;
```

```
  CloseMixer;

  If mixerOpen (@FMixerHandle, Value, XWndHandle, 0, CALLBACK_WINDOW OR
MIXER_OBJECTF_MIXER) = MMSYSERR_NOERROR then
    goto AllOK;

  // we will go here very rarely, but sometimes it could help
  If mixerOpen (@FMixerHandle, Value, XWndHandle, 0, CALLBACK_WINDOW) =
MMSYSERR_NOERROR then
    goto AllOK;
  If mixerOpen (@FMixerHandle, Value, 0, 0, 0) = MMSYSERR_NOERROR then
    goto AllOK;

  // відбулась помилка
  FMixerId := -1;
  FDestinations := TMixerDestinations.Create (nil);

  Exit;

AllOK:
  FMixerId := Value;
  mixerGetDevCaps (MixerId, @FMixerCaps, SizeOf (TMixerCaps));

  if FMixerCaps.wMid = MM_MICROSOFT then
    FManufacturer := 'Microsoft'
  else
    FManufacturer := IntToStr(FMixerCaps.wMid) + ' = Unknown';
  FDriverVersion := FMixerCaps.vDriverVersion;
  FProductId := FMixerCaps.wPid;
  FProductName := StrPas(FMixerCaps.szPName);
  FNumberOfLine := FMixerCaps.cDestinations;

  FDestinations := TMixerDestinations.Create (Self);
end;

procedure TAudioMixer.MixerCallBack (var Msg:TMessage);
begin
  case Msg.Msg of
    MM_MIXM_LINE_CHANGE:
        If Assigned (OnLineChange) then
          OnLineChange (Self, Msg.wParam, Msg.lParam);
    MM_MIXM_CONTROL_CHANGE:
        If Assigned (OnControlChange) then
          OnControlChange (Self, Msg.wParam, Msg.lParam);
    else
```

```
      Msg.Result   :=   DefWindowProc   (XWndHandle,   Msg.Msg,   Msg.WParam,
Msg.LParam);
  end;
end;

const MIXER_LONG_NAME_CHARS = 64;

type MIXERCONTROLDETAILS_LISTTEXT = record
       dwParam1:DWORD;
       dwParam2:DWORD;
       szName:Array [0..MIXER_LONG_NAME_CHARS-1] of Char;
     end;

type ListTextArray = array [0..1000] of MIXERCONTROLDETAILS_LISTTEXT;

function TAudioMixer.GetVolume (ADestination,AConnection:Integer;var LeftVol,
RightVol,     Mute:Integer;var     Stereo,     VolDisabled,     MuteDisabled,
MuteIsSelect:Boolean):Boolean;
var MD:TMixerDestination;
    MC:TMixerConnection;
    Cntrls:TMixerControls;
    MCD:TMixerControlDetails;
    Cntrl:PMixerControl;
    A,B:Integer;
    ML:TMixerLine;
    details:array [0..100] of Integer;
    ltext:^ListTextArray;
    MCDText:TMixerControlDetails;

begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
 Stereo := False;
  MuteDisabled := True;
  VolDisabled := True;
  LeftVol := -1;
  RightVol := -1;
  Mute := -1;
  MuteIsSelect := False;
  MD := Destinations[ADestination];
  MC := nil;
  If AConnection <> -1 then
    MC := MD.Connections[AConnection];

  If MD <> nil then
```

```
  begin

    Result := True;

//      If Mute = -1 then
    begin
      If AConnection <> -1 then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Mute = -1) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
//                If (Cntrl.dwControlType AND MIXERCONTROL_CONTROLTYPE_MIXER =
MIXERCONTROL_CONTROLTYPE_MIXER) OR
//                    (Cntrl.dwControlType AND MIXERCONTROL_CONTROLTYPE_MUX =
MIXERCONTROL_CONTROLTYPE_MUX) then

//                  If (Cntrl.dwControlType AND MIXERCONTROL_CT_CLASS_MASK =
MIXERCONTROL_CT_CLASS_LIST) then

            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
               // Mux is similar to mixer, but only one line can be selected
at a time
            begin
              MCD.cbStruct := SizeOf(TMixerControlDetails);
              MCD.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                MCD.cChannels := 1
              else
                MCD.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                MCD.cMultipleItems := Cntrl.cMultipleItems
              else
                MCD.cMultipleItems := 0;
              MCD.cbDetails := 4;
              MCD.paDetails := @Details;
              B                  :=                  mixerGetControlDetails
(FMixerHandle,@MCD,MIXER_GETCONTROLDETAILSF_VALUE);
              If B <> MMSYSERR_NOERROR then
              begin
                Inc (A);
```

```
            continue;
          end;

          MCDText.cbStruct := sizeof (MCDText);
          MCDText.dwControlID := Cntrl.dwControlID;
          If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
            MCDText.cChannels := 1
          else
            MCDText.cChannels := ML.cChannels;
          If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
            MCDText.cMultipleItems := Cntrl.cMultipleItems
          else
            MCDText.cMultipleItems := 0;
          GetMem  (ltext,  MCDText.cChannels  *  MCDText.cMultipleItems  *
sizeof (MIXERCONTROLDETAILS_LISTTEXT));
          MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
          MCDText.paDetails := ltext;
          B    :=    mixerGetControlDetails    (FMixerHandle,    @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
          If B <> MMSYSERR_NOERROR then
          begin
            FreeMem (ltext);
            Inc (A);
            continue;
          end;
          B := MCD.cChannels - 1;
          while (B < integer(MCD.cMultipleItems)) do
          begin
            if (ltext[B].dwParam1 = MC.Data.dwLineID) then
              break;
            Inc (B, MCD.cChannels);
          end;
          FreeMem (ltext);

          If (B < integer (MCD.cMultipleItems)) then
          begin
            Mute := Details[B];
            MuteDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
            MuteIsSelect := True;
            break;
          end;
        end;
        Inc (A);
      end;
```

```
        end;
      end;
    end;

    If AConnection = -1 then
    begin
      Cntrls := MD.Controls;
      ML := MD.Data;
    end
    else
    begin
      If MC <> nil then
      begin
        Cntrls := MC.Controls;
        ML := MC.Data;
      end
      else
        Cntrls := nil;
    end;
    If Cntrls <> nil then
    begin
      A := 0;
      while ((LeftVol = -1) OR (Mute = -1)) AND (A < Cntrls.Count) do
      begin
        Cntrl := Cntrls[A];
        If Cntrl <> nil then
        begin
          If ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) OR
              ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) AND (Mute
= -1))) AND
             (Cntrl.fdwControl    AND    MIXERCONTROL_CONTROLF_MULTIPLE    <>
MIXERCONTROL_CONTROLF_MULTIPLE)
              then
          begin
            if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) then
              MCD.cbStruct := SizeOf(TMixerControlDetails)
            else
              MCD.cbStruct := SizeOf(TMixerControlDetails);
            MCD.dwControlID := Cntrl.dwControlID;
            If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
              MCD.cChannels := 1
            else
              MCD.cChannels := ML.cChannels;
            MCD.cMultipleItems := 0;
            MCD.cbDetails := SizeOf(Integer);
            MCD.paDetails := @details;
```

```
            B     :=     mixerGetControlDetails     (FMixerHandle,     @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
            If B = MMSYSERR_NOERROR then
            begin
              If  (Cntrl.dwControlType  =  MIXERCONTROL_CONTROLTYPE_VOLUME)  AND
(LeftVol = -1) then
              begin
                VolDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                LeftVol := details[0];
                If MCD.cChannels > 1 then
                begin
                  RightVol := Details[1];
                  Stereo := True;
                end
                else
                  RightVol := LeftVol;
              end
              else  If  (Cntrl.dwControlType  =  MIXERCONTROL_CONTROLTYPE_MUTE)
AND (Mute = -1) then
              begin
                MuteDisabled         :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                If Details[0] <> 0 then
                  Mute := 1
                else
                  Mute := 0;
              end
// NEW ->
(*              else If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_ONOFF)
AND (Mute = -1) then
              begin
                MuteDisabled         :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                If Details[0] <> 0 then
                  Mute := 1
                else
                  Mute := 0;
                MuteIsSelect := True;
              end;*)
// <- NEW
            end;
          end;
        end;
        Inc (A);
      end;
```

```
{       If LeftVol = -1 then
          VolDisabled := True;
        If Mute = -1 then
          MuteDisabled := True;}
    end;
  end;
end;

function  TAudioMixer.SetVolume  (ADestination,  AConnection:Integer;  LeftVol,
RightVol, Mute:Integer):Boolean;
var MD:TMixerDestination;
    MC:TMixerConnection;
    Cntrls:TMixerControls;
    MCD:TMixerControlDetails;
    Cntrl:PMixerControl;
    A,B:Integer;
    ML:TMixerLine;
    details:array [0..100] of Integer;
    VolSet,MuteSet:Boolean;
    ltext:^ListTextArray;
    MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  MC := nil;
  MD := Destinations[ADestination];
  If MD <> nil then
  begin
    If AConnection <> -1 then
      MC := MD.Connections[AConnection];

    VolSet := LeftVol = -1;
    MuteSet := Mute = -1;
    Result := True;

    If not MuteSet then
    begin
      If AConnection <> -1 then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
```

```
          while not MuteSet AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
            If (*(Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR*)
               (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
            begin
              MCD.cbStruct := SizeOf(TMixerControlDetails);
              MCD.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                MCD.cChannels := 1
              else
                MCD.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                MCD.cMultipleItems := Cntrl.cMultipleItems
              else
                MCD.cMultipleItems := 0;
              MCD.cbDetails := 4;
              MCD.paDetails := @Details;
              MuteSet := True;
              mixerGetControlDetails          (FMixerHandle,          @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
              if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
                For B := 0 to Cntrl.cMultipleItems-1 do
                  Details[B] := 0;

              GetMem  (ltext,  MCD.cChannels  *  MCD.cMultipleItems  *  sizeof
(MIXERCONTROLDETAILS_LISTTEXT));
              MCDText.cbStruct := sizeof (MCDText);
              MCDText.dwControlID := Cntrl.dwControlID;
              MCDText.cChannels := MCD.cChannels;
              MCDText.cMultipleItems := MCD.cMultipleItems;
              MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
              MCDText.paDetails := ltext;
              mixerGetControlDetails          (FMixerHandle,          @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
              B := MCD.cChannels - 1;
              while (B < integer (MCD.cMultipleItems)) do
              begin
                if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                  break;
                Inc (B, MCD.cChannels);
              end;
              FreeMem (ltext);

              If (B < integer (MCD.cMultipleItems)) then
```

```
              begin
                Details[B] := Mute;
                mixerSetControlDetails          (FMixerHandle,          @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;
    end;



    If AConnection = -1 then
    begin
      Cntrls := MD.Controls;
      ML := MD.Data;
    end
    else
    begin
      If MC <> nil then
      begin
        Cntrls := MC.Controls;
        ML := MC.Data;
      end
      else
        Cntrls := nil;
    end;
    If Cntrls <> nil then
    begin

      A := 0;
      while (not VolSet OR not MuteSet) AND (A < Cntrls.Count) do
      begin
        Cntrl := Cntrls[A];
        If Cntrl <> nil then
        begin
          If (((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) AND not
VolSet) OR
              ((Cntrl.dwControlType  =  MIXERCONTROL_CONTROLTYPE_MUTE)  AND  not
MuteSet) (* NEW -> *) (*OR
              ((Cntrl.dwControlType  =  MIXERCONTROL_CONTROLTYPE_ONOFF)  AND  not
MuteSet)*) (* <- NEW *)) AND
```

```
            (Cntrl.fdwControl     AND     MIXERCONTROL_CONTROLF_MULTIPLE     <>
MIXERCONTROL_CONTROLF_MULTIPLE)
             then
          begin
            MCD.cbStruct := SizeOf(TMixerControlDetails);
            MCD.dwControlID := Cntrl.dwControlID;
            If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
              MCD.cChannels := 1
            else
              MCD.cChannels := ML.cChannels;
            MCD.cMultipleItems := 0;
            MCD.cbDetails := SizeOf(Integer);
            MCD.paDetails := @Details;
            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) then
            begin
              Details[0] := LeftVol;
              If RightVol = -1 then
                Details[1] := LeftVol
              else
                Details[1] := RightVol;
              VolSet := True;
            end
            else if ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) (*
NEW -> *) (* OR
                     (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_ONOFF) *)
(* <- NEW *)) then
            begin
              For B := 0 to MCD.cChannels - 1 do
                Details[B] := Mute;
              MuteSet := True;
            end;
            mixerSetControlDetails            (FMixerHandle,            @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
           end;
         end;
         Inc (A);
       end;

    end;
  end;
end;

function  TAudioMixer.GetMute(ADestination,  AConnection:  Integer;  var  Mute:
Boolean):Boolean;
var
  MD : TMixerDestination;
```

```
  MC : TMixerConnection;
  mlcMixerLineControlsMute : TMIXERLINECONTROLS;
  mcdMixerDataMute : TMIXERCONTROLDETAILS;
  pmcMixerControlMute : PMIXERCONTROL;
  pmcdsMixerDataUnsignedMute : PMIXERCONTROLDETAILSBOOLEAN;
  mlMixerLine : TMixerLine;
  Cntrl:PMixerControl;
  Cntrls:TMixerControls;
  ML:TMixerLine;
  A,B:Integer;
  details:array [0..100] of Integer;
  ltext:^ListTextArray;
  MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  MC := nil;
  Mute := False;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
      mlMixerLine := MD.Data
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
        mlMixerLine := MC.Data
      else
        Exit;
    end;

    GetMem(pmcMixerControlMute, SizeOf(TMIXERCONTROL));
    GetMem(pmcdsMixerDataUnsignedMute, SizeOf(TMIXERCONTROLDETAILSBOOLEAN));

    with mlcMixerLineControlsMute do
    begin
      cbStruct := SizeOf(TMIXERLINECONTROLS);
      dwLineID := mlMixerLine.dwLineID;
      dwControlType := MIXERCONTROL_CONTROLTYPE_MUTE;
      cControls := 1;
      cbmxctrl := SizeOf(TMIXERCONTROL);
      pamxctrl := pmcMixerControlMute;
    end;
```

```
    if      (mixerGetLineControls(FMixerHandle,      @mlcMixerLineControlsMute,
MIXER_GETLINECONTROLSF_ONEBYTYPE) = MMSYSERR_NOERROR) then
    begin
      with mcdMixerDataMute do
      begin
        cbStruct := SizeOf(TMIXERCONTROLDETAILS);
        dwControlID := pmcMixerControlMute^.dwControlID;
        cChannels := 1;
        cMultipleItems := 0;
        cbDetails := SizeOf(TMIXERCONTROLDETAILSBOOLEAN);
        paDetails := pmcdsMixerDataUnsignedMute;
      end;

      if        mixerGetControlDetails(FMixerHandle,       @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) = MMSYSERR_NOERROR then
      begin
        Mute := pmcdsMixerDataUnsignedMute^.fValue = 1;
        Result := True;
      end;
    end
    else
    begin
      If (AConnection <> -1) then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Result = False) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
           If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR
             (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
          begin
            mcdMixerDataMute.cbStruct := SizeOf(TMixerControlDetails);
            mcdMixerDataMute.dwControlID := Cntrl.dwControlID;
            If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
              mcdMixerDataMute.cChannels := 1
            else
              mcdMixerDataMute.cChannels := ML.cChannels;
            If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
              mcdMixerDataMute.cMultipleItems := Cntrl.cMultipleItems
            else
              mcdMixerDataMute.cMultipleItems := 0;
```

```
              mcdMixerDataMute.cbDetails := 4;
              mcdMixerDataMute.paDetails := @Details;
              mixerGetControlDetails     (FMixerHandle,     @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE);

              GetMem        (ltext,        mcdMixerDataMute.cChannels        *
mcdMixerDataMute.cMultipleItems * sizeof (MIXERCONTROLDETAILS_LISTTEXT));
              MCDText.cbStruct := sizeof (MCDText);
              MCDText.dwControlID := Cntrl.dwControlID;
              MCDText.cChannels := mcdMixerDataMute.cChannels;
              MCDText.cMultipleItems := mcdMixerDataMute.cMultipleItems;
              MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
              MCDText.paDetails := ltext;
              mixerGetControlDetails          (FMixerHandle,          @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
              B := mcdMixerDataMute.cChannels - 1;
              while (B < integer (mcdMixerDataMute.cMultipleItems)) do
              begin
                if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                  break;
                Inc (B, mcdMixerDataMute.cChannels);
              end;
              FreeMem (ltext);

              If (B < integer (mcdMixerDataMute.cMultipleItems)) then
              begin
                Result := True;
                Mute := Details[B] <> 0;
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;

      FreeMem(pmcdsMixerDataUnsignedMute);
      FreeMem(pmcMixerControlMute);
    end;
end;

function   TAudioMixer.SetMute(ADestination,   AConnection:   Integer;   Mute:
Boolean):Boolean;
var
  MD : TMixerDestination;
```

```
  MC : TMixerConnection;
  mlcMixerLineControlsMute : TMIXERLINECONTROLS;
  mcdMixerDataMute : TMIXERCONTROLDETAILS;
  pmcMixerControlMute : PMIXERCONTROL;
  pmcdsMixerDataUnsignedMute : PMIXERCONTROLDETAILSBOOLEAN;
  mlMixerLine : TMixerLine;
  Cntrl:PMixerControl;
  Cntrls:TMixerControls;
  ML:TMixerLine;
  A,B:Integer;
  details:array [0..100] of Integer;
  ltext:^ListTextArray;
  MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  MC := nil;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
      mlMixerLine := MD.Data
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
        mlMixerLine := MC.Data
      else
        Exit;
    end;

    GetMem(pmcMixerControlMute, SizeOf(TMIXERCONTROL));
    GetMem(pmcdsMixerDataUnsignedMute, SizeOf(TMIXERCONTROLDETAILSBOOLEAN));

    with mlcMixerLineControlsMute do
    begin
      cbStruct := SizeOf(TMIXERLINECONTROLS);
      dwLineID := mlMixerLine.dwLineID;
      dwControlType := MIXERCONTROL_CONTROLTYPE_MUTE;
      cControls := 0;
      cbmxctrl := SizeOf(TMIXERCONTROL);
      pamxctrl := pmcMixerControlMute;
    end;
```

```
    if      (mixerGetLineControls(FMixerHandle,      @mlcMixerLineControlsMute,
MIXER_GETLINECONTROLSF_ONEBYTYPE) = MMSYSERR_NOERROR) then
    begin
      with mcdMixerDataMute do
      begin
        cbStruct := SizeOf(TMixerControlDetails);
        dwControlID := pmcMixerControlMute^.dwControlID;
        cChannels := 1;
        cMultipleItems := 0;
        cbDetails := SizeOf(TMIXERCONTROLDETAILSBOOLEAN);
        paDetails := pmcdsMixerDataUnsignedMute;
      end;

      if Mute then
        pmcdsMixerDataUnsignedMute^.fValue := 1
      else
        pmcdsMixerDataUnsignedMute^.fValue := 0;

      if
(mixerSetControlDetails(FMixerHandle,@mcdMixerDataMute,MIXER_SETCONTROLDETAILS
F_VALUE) = MMSYSERR_NOERROR) then
        Result := True;
    end
    else
    begin
      If (AConnection <> -1) then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Result = False) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR
               (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
            begin
              mcdMixerDataMute.cbStruct := SizeOf(TMixerControlDetails);
              mcdMixerDataMute.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                mcdMixerDataMute.cChannels := 1
              else
                mcdMixerDataMute.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
```

```
                mcdMixerDataMute.cMultipleItems := Cntrl.cMultipleItems
              else
                mcdMixerDataMute.cMultipleItems := 0;
              mcdMixerDataMute.cbDetails := 4;
              mcdMixerDataMute.paDetails := @Details;
              if (mixerGetControlDetails (FMixerHandle, @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) <> MMSYSERR_NOERROR) then
              begin
                Inc (A);
                continue;
              end;
              if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
                For B := 0 to Cntrl.cMultipleItems-1 do
                  Details[B] := 0;
              If Mute then
              begin
                GetMem (ltext, mcdMixerDataMute.cChannels *
mcdMixerDataMute.cMultipleItems * sizeof (MIXERCONTROLDETAILS_LISTTEXT));
                MCDText.cbStruct := sizeof (MCDText);
                MCDText.dwControlID := Cntrl.dwControlID;
                MCDText.cChannels := mcdMixerDataMute.cChannels;
                MCDText.cMultipleItems := mcdMixerDataMute.cMultipleItems;
                MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
                MCDText.paDetails := ltext;
                mixerGetControlDetails (FMixerHandle, @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
                B := mcdMixerDataMute.cChannels - 1;
                while (B < integer (mcdMixerDataMute.cMultipleItems)) do
                begin
                  if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                    break;
                  Inc (B, mcdMixerDataMute.cChannels);
                end;
                FreeMem (ltext);

                If (B < integer (mcdMixerDataMute.cMultipleItems)) then
                  Details[B] := 1;
              end;
              if (mixerSetControlDetails (FMixerHandle, @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) = MMSYSERR_NOERROR) then
              begin
                Result := True;
                break;
              end;
            end;
            Inc (A);
```

```
          end;
        end;
      end;
    end;

    FreeMem(pmcdsMixerDataUnsignedMute);
    FreeMem(pmcMixerControlMute);
  end;
end;

function TAudioMixer.GetPeak (ADestination, AConnection:Integer; var LeftPeak,
RightPeak:Integer): Boolean;
var
  MD : TMixerDestination;
  MC : TMixerConnection;
  mcdMixerDataPeak : TMIXERCONTROLDETAILS;
  pmcMixerControlPeak : PMIXERCONTROL;
{  pmcdsMixerDataSignedPeak : PMIXERCONTROLDETAILSSIGNED;}
  mlMixerLine : TMixerLine;
  A:Integer;
  Cntrls:TMixerControls;
  Details:Array [1..100] of Integer;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  LeftPeak := 0;
  RightPeak := 0;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
    begin
      mlMixerLine := MD.Data;
      Cntrls := MD.Controls;
    end
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
      begin
        mlMixerLine := MC.Data;
        Cntrls := MC.Controls;
      end
      else
        Exit;
```

```
    end;
    GetMem(pmcMixerControlPeak, SizeOf(TMIXERCONTROL));

    A := 0;
    while (A < Cntrls.Count) do
    begin
      If    (Cntrls[A].dwControlType    AND    MIXERCONTROL_CT_CLASS_MASK)    =
MIXERCONTROL_CT_CLASS_METER then
        break;
      Inc (A);
    end;
    If A = Cntrls.Count then
    begin
      FreeMem(pmcMixerControlPeak);
      Exit;
    end;

    with mcdMixerDataPeak do
    begin
      cbStruct := SizeOf(TMIXERCONTROLDETAILS);
      dwControlID := Cntrls[A].dwControlID;
      cChannels := mlMixerLine.cChannels;
      If    (Cntrls[A].fdwControl    AND    MIXERCONTROL_CONTROLF_MULTIPLE)    =
MIXERCONTROL_CONTROLF_MULTIPLE then
        cMultipleItems:=Cntrls[A].cMultipleItems
      else
        cMultipleItems:=0;
      cbDetails := SizeOf(TMIXERCONTROLDETAILSSIGNED);
      paDetails := @Details;
    end;
    if
(mixerGetControlDetails(FMixerHandle,@mcdMixerDataPeak,MIXER_GETCONTROLDETAILS
F_VALUE) = MMSYSERR_NOERROR) then
    begin
      LeftPeak := Details[1];
      if mlMixerLine.cChannels = 2 then
        RightPeak := Details[2]
      else
        RightPeak := LeftPeak;
      Result := True;
    end;
    FreeMem(pmcMixerControlPeak);
   end;
end;
procedure Register;
begin
```

```
  RegisterComponents('Samples', [TAudioMixer]);
end;
end.
```

AMixer.pas– Довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TForm5 = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form5: TForm5;

implementation

{$R *.dfm}

procedure TForm5.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('БАКАЛАВРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Програмне забезпечення системи дослідження протоколів IP-
телефонії хмарної банківської мережі');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Коваленко А.С.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Буза В.В.');
Memo1.Lines.Add('                    гр. КІ-20-3СК');
Memo1.Lines.Add('');
```

```
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;

procedure TForm5.Button1Click(Sender: TObject);
begin
Form5.Close;
end;

end.
```