Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Проєктування системи цифрової обробки звукових сигналів»

Студента 2 курсу, 2м групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Мілевського Дмитра Володимировича

Науковий керівник кандидат економічних наук, доцент _____ підпис керівника — Палагута Катерина Олексіївна

Гарант освітньої програми доктор економічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис гаранта — Токар Володимир Володимирович

КИЇВ - 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«10» листопада 2020 р.

Завдання
на випускний кваліфікаційний проєкт студентові
Мілевському Дмитру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Проєктування системи цифрової обробки звукових сигналів»

Затверджена наказом ректора від «30» листопада 2020 р. № 3224

2. Строк здачі студентом закінченого проєкту 25 листопада 2021

3. Цільова установка та вихідні дані до проєкту

Мета проєкту: поліпшення однорангової міжмашинної комунікації шляхом використання акустичного каналу передачі даних

Об'єкт дослідження: програмне забезпечення для обміну даними за допомогою звукових хвиль.

Предмет дослідження: використання звукових пристроїв (динаміки, мікрофони) для обміну.

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ОСНОВНІ ВІДОМОСТІ

1.1. Потенціал технології обміну даними за допомогою звукових хвиль

1.2. Принцип роботи технології

1.2.1. Частотна маніпуляція

1.2.2. Перетворення Фур'є

1.3. Аналіз наявних систем обміну даних за допомогою звукових сигналів

1.3.1. LISNR

1.3.2. Sonarax

1.3.3. CopSonic

1.4. Висновки до розділу 1

РОЗДІЛ 2. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Вимоги до програмного забезпечення

2.1.1. Вимоги до системи в цілому

2.1.2. Вимога до показників призначення

2.1.3. Вимоги до захисту від несанкціонованого доступу

2.1.4. Вимоги до системних характеристик і функцій (завдань)

2.2. Вимоги до функцій (завдань)

2.2.1. Вимоги до функціональної структури системи (підсистеми і взаємодія)

2.2.2. Вимоги до окремих функціональних підсистем

2.3. Вимоги до видів забезпечення

2.3.1. Вимоги до інформаційного забезпечення

2.3.2. Вимога до сумісності із суміжними системами

2.3.3. Вимога до системи кодування і класифікації об'єктів і процесів

2.4. Вибір архітектури

2.5. Опис архітектури програмного забезпечення

6. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2020*	*21.09.2020*
2.	*Розробка та затвердження завдання на проєкт магістра (стац/заоч)*	*06.11.2020*	*22.12.2020*
3.	*Вступ та перелік літературних джерел*	*27.02.2021*	*27.02.2021*
4.	*Розробка технічного завдання*	*20.03.2021*	*20.03.2021*
5.	*Розділ 1.* (*Назва розділу 1*)	*16.04.2021*	*16.04.2021*
6.	*Розділ 2.* (*Назва розділу 2*)	*24.05.2021*	*24.05.2021*
7.	*Розділ 3.* (*Назва розділу 3*)	*21.06.2021*	*21.06.2021*
8.	*Розділ 4.* (*Назва розділу 4, при необхідності, можливе об'єднання розділів 3 ти 4*)	*20.09.2021*	*20.09.2021*
9.	*Розробка програми та методики тестування*	*18.10.2021*	*18.10.2021*
10.	*Написання наукової статті*	*22.05.2021*	*22.05.2021*
11.	*Керівництво користувача*	*21.10.2021*	*21.10.2021*
12.	*Висновки та пропозиції*	*01.11.2021*	*01.11.2021*
13.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*03.11.2021*	*03.11.2021*
14.	*Підготовка автореферату та презентації доповіді*	*03.11.2021*	*03.11.2021*
15.	*Попередній захист випускного кваліфікаційного проєкту*	*22.11.2021 – 25.11.2021*	*22.11.2021*
16.	*Здача зброшурованої випускного кваліфікаційного проєкту*	*25.11.2021*	*25.11.2021*
17.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*26.11.2021*	*26.11.2021*
18.	*Підготовка до публічного захисту випускного кваліфікаційного проєкту*	*за розкладом роботи ЕК*	

7. Дата видачі завдання «10» листопада 2020 р.

8. Науковий керівник випускного кваліфікаційного проєкту Палагута К.О

(прізвище, ініціали, підпис)

9. Гарант освітньої програми Токар В.В.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Мілевський Д.В.

(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента _____Мілевського Д.В.___

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____ Токар В.В.

(прізвище, ініціали, підпис)

Завідувач кафедри _____ Криворучко О. В.

(підпис, прізвище, ініціали)

«_____»_____20 ___р.

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена розробці програмного забезпечення для обміну даними за допомогою звукових хвиль.

В результаті порівняльного аналізу аналогічних рішень описано три існуючі рішення для обміну даними за допомогою звукових хвиль – платформи LISNR, CopSonic та технологію Sonarax. Досвід розробників цих рішень свідчить про те, що технологія перспективна та затребувана на ринку.

Розробка виконана у середовищі Microsoft Visual Studio

Ключові слова: ОБМІН ДАНИМИ, ПЕРЕДАЧА ДАНИХ, ОТРИМАННЯ ДАНИХ, КРИПТОГРАФІЯ, ІНТЕРНЕТ РЕЧЕЙ

ABSTRACT

In accordance with the purpose of the study, the work is devoted to the development of software for data exchange by sound waves.

As a result of a comparative analysis of similar solutions, three existing solutions for data exchange are described using sound waves - LISNR platform, CopSonic and Sonarax technology. The experience of developers of these solutions indicates that technology is promising and in demand in the market.

Development is made in Microsoft Visual Studio

Keywords: DATA EXCHANGE, DATA TRANSFER, DATA RECEIVING, CRYPTOGRAPHY, INTERNET OF THINGS

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IoT – інтернет речей

ЧМ – частотна маніпуляція

ШПФ – швидке перетворення Фур'є

ПК – програмний комплекс

ПЗ – програмне забезпечення

ОС – операційна система

ОЗП – оперативний запам'ятовуючий пристрій

ЦП – центральний процесор

ООП – об'єктно-орієнтоване програмування

					КНТЕУ 121 02-11.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		29.09.20	Проєктування системи цифрової обробки звукових сигналів	*Стадія*	*Арку*	*Аркушів*
Керівник		Палагута К.О.		29.09.20		*ПС*	*2*	*40*
Гарант		Токар В.В.		29.09.20		*Факультет інформаційних технологій 2 курс, 2м група*		
Розробив		Мілевський Д.В.		29.09.20	*Перелік умовних скорочень*			

ЗМІСТ

					КНТЕУ 121 02-11.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		22.12.20	Проєктування системи цифрової обробки звукових сигналів	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Палагута К.О.		22.12.20		*В*	*3*	*40*
Гарант		Токар В.В..		22.12.20		Факультет інформаційних технологій 2 курс, 2м група		
Розробив		Мілевський Д.В.		22.12.20	*Вступ*			

ВСТУП

Від смартфонів і планшетів до промислового обладнання, тисячі розумних пристроїв наразі потребують різних форм підключення. Неминучий попит на краще підключення збільшив очікування для навіть найбільш базових розумних пристроїв мати можливість обробляти звук у реальному часі та здійснювати інтелектуальну цифрову обробку сигналу на точках доступу до мережі.

Це, зрештою, відкрило шлях для безлічі інноваційних учасників ринку, які прагнуть одночасно кинути виклик і працювати в парі з традиційними рішеннями для забезпечення якісних можливостей передачі даних.

Обмін даними за допомогою звукових хвиль – це одна з технологій, яка швидко розповсюджується як захоплюючий варіант підключення для інженерів та розробників, які прагнуть досягти неперервної взаємодії між постійно зростаючою кількістю підключених пристроїв

Мета дослідження – поліпшення однорангової міжмашинної комунікації шляхом використання акустичного каналу передачі даних.

Об’єкт дослідження – програмне забезпечення для обміну даними за допомогою звукових хвиль.

Предмет дослідження – використання звукових пристроїв (динаміки, мікрофони) для обміну.

Методи дослідження – вибір мови програмування, дослідження технологій створення програмного забезпечення, вибір архітектури

РОЗДІЛ 1
ОСНОВНІ ВІДОМОСТІ

1.1 Потенціал технології обміну даними за допомогою звукових хвиль

Існує дуже чітка та практична цінність обміну даними за допомогою звукових хвиль. Починаючи з безперервної інтеграції в існуюче обладнання та з'єднань, до здатності працювати в автономному режимі навіть у найекстремальніших середовищах, є безліч можливих випадків використання технології обміну даними за допомогою звукових хвиль. Технологія має безліч додаткових можливостей у порівнянні з іншими технологіями підключення. Позиція технології як середовища один-до-багатьох означає, що вона полегшує деякі точки налаштування, часто пов'язані з традиційними альтернативами, такими як Bluetooth та Wi-Fi, представляючи привабливе та універсальне рішення для передачі даних. Загалом, порівняно з іншими технологіями, передачу даних за допомогою звукових хвиль також можна використовувати в дуже широких областях застосування, використовуючи переваги існуючого обладнання та без попереднього налаштування або конфігурації. Це робить можливим взаємозв'язок мільйонів пристроїв безперешкодним, масштабованим та економічно вигідним способом, щоб покращити взаємодію з кінцевими користувачами та збільшити вартість, не збільшуючи витрат на обладнання. Ця здатність вдосконалювати існуючу інфраструктуру неминуче привертала інтерес компаній, зацікавлених у додаванні

функціональних можливостей бездротового підключення, не додаючи до свого кошторису додаткових матеріалів.

Потенціал цієї технології, як форми зв'язку з низьким енергоспоживанням, яка застосовується у різних областях IoT, звуку, голосу виходить на передній план у зростаючій кількості варіантів використання. Настільки, що він все більше застосовується як невід'ємний інструмент для безперебійної передачі ультразвукових даних як в промислових, так і в любительських середовищах.

Майбутнє підключених пристроїв полягає у використанні широкого спектру рішень для підключення, включаючи передачу даних за допомогою звуку, де кожна технологія доповнює інші.

1.2Принцип роботи технології

Передача даних здійснюється за допомогою звукових хвиль між будь-якими пристроями, які мають динамік та мікрофон. Ця технологія забезпечує міжмашинний тип комунікації, передаючи дані по акустичному каналу. На практиці дані модулюються в акустичний сигнал – серію чутних (20-20000 Гц) або нечутних (ультразвукових) висот та тонів звуку, утворюючи своєрідний «звуковий штрих-код».

Рис. 1.1 Спектрограма акустичного сигналу

Потім акустичний сигнал відтворюється у просторі (як правило, у повітрі, але також це може бути цифровий канал передачі звукового сигналу), приймається та демодулюється пристроєм-слухачем або групою пристроїв-слухачів. Використовуючи діапазон звукових частот, програмісти можуть

вмістити більше інформації в меншу кількість звуку. Щоб відфільтрувати шум, розробники ретельно підбирають частоти та налаштовують програмне забезпечення, щоб пристрої могли комунікувати між собою для виявлення та декодування сигналів даних у галасливих умовах. Вдалим рішенням для кодування та декодування звукових даних є частотна маніпуляція.

1.3Частотна маніпуляція

Частотна маніпуляція (Frequency Shift Keying, FSK) – один з видів цифрової модуляції сигналу, у якій цифрова інформація передається через дискретні частотні зміни несучого сигналу. У випадку передачі даних через звуковий канал дані представлені змінами у висоті (частоті) звукового тону.

Найпростішою формою ЧМ є бінарна ЧМ, яка використовує пару дискретних частот для передачі бінарної (нулі та одиниці) інформації. Бінарна ЧМ не підходить для швидкісних комунікацій, оскільки має низьку пропускну здатність порівняно з іншими формами модуляцій, але в той же час більш стійка до спотворень сигналу, що зменшує кількість помилок в отриманих даних.

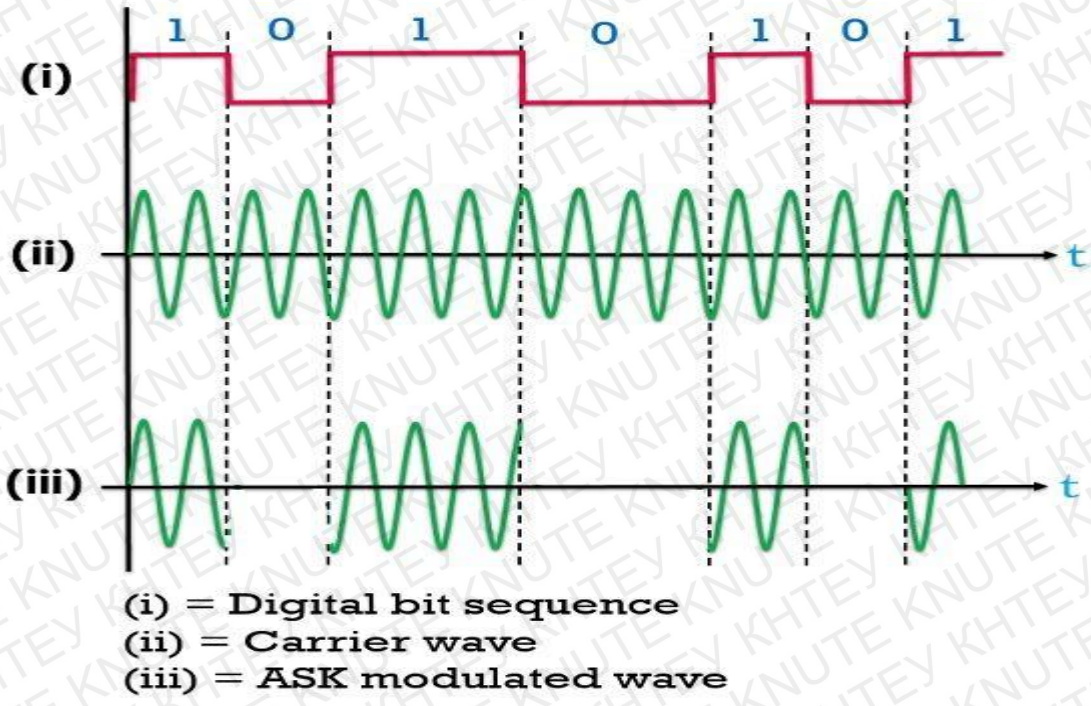

Рис. 1.2. Бінарна ЧМ

Іншою формою ЧМ є мультичастотна маніпуляція, яка розширює бінарну ЧМ вводячи більшу кількість дискретних частот, що використовуються для модуляції. Більша кількість частот збільшує пропускну здатність каналу зменшуючи стійкість сигналу до спотворень.

Враховуючи вищезазначене, потрібно ретельно підбирати форму ЧМ для оптимального застосування її у технологіях передачі даних. Нині ЧМ використовується у різних сферах, зокрема у системах екстреного оповіщення США, стандарті GSM тощо.

1.3.1. Перетворення Фур'є

Для демодуляції звукового сигналу використовується алгоритм швидкого перетворення Фур'є (Fast Fourier Transform, FFT). Алгоритм дозволяє перетворити сигнал у окремі спектральні компоненти тим самим забезпечуючи інформацію про частоту сигналу. ШПФ широко використовуються для аналізу несправностей, контролю якості та контролю стану машин або систем. ШПФ – це оптимізований алгоритм для реалізації дискретного перетворення Фур'є (ДФП). Сигнал дискретизується протягом певного періоду і ділиться на його частотні складові. Ці компоненти є одиничними синусоїдальними коливаннями на різних частотах, кожна з яких має власну амплітуду та фазу. Виявлені частоти декодуються у двійкові дані тим самим чином, яким вони були закодовані.

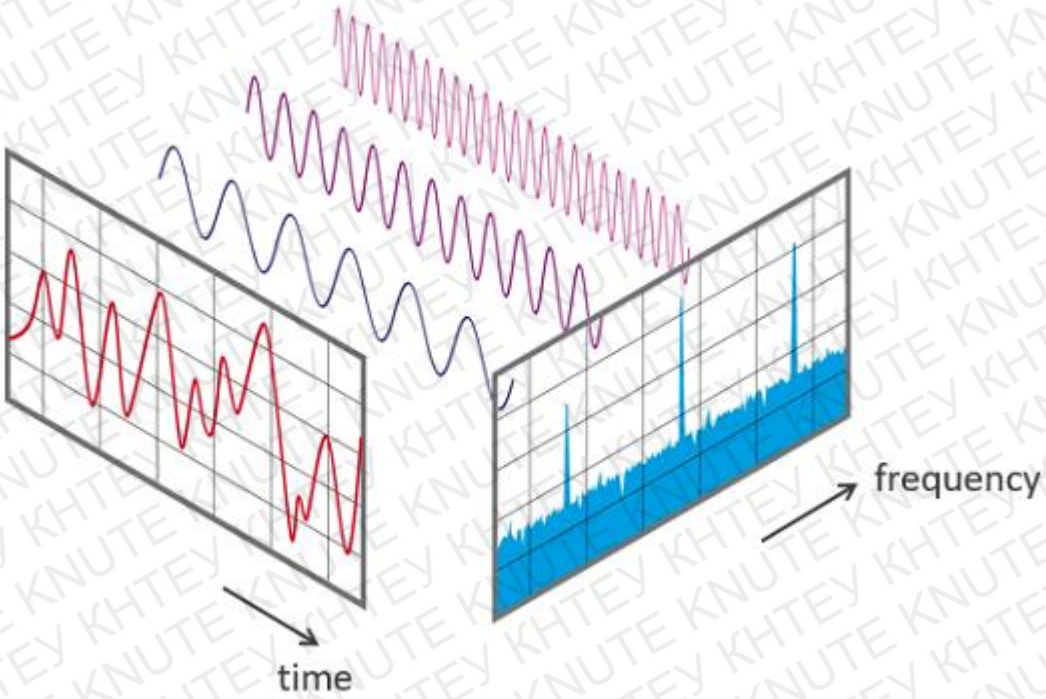

Рис. 1.3. Процес ШПФ

1.4Аналіз наявних систем обміну даних за допомогою звукових сигналів

1.4.1 LISNR

LISNR - це пропрієтарний, близький до ультразвукового протокол, який надсилає дані за допомогою аудіо.

Протоколи LISNR надсилають дані через аудіо з пропускною здатністю (12-19 кГц), які чутні або нечутні, залежно від способу використання та розподілу, для забезпечення оптимальної продуктивності. Додаткові покращення продуктивності включають різні типи звукових тонів, частотні канали та профілі, оптимізовані для різного використання високоякісних характеристик фактичного використання. Технологія побудована на базі C-бібліотеки, що дозволяє зручно адаптуватися до пов'язаної світової екосистеми. Ця бібліотека забезпечує взаємодію всіх програмних додатків

з операційними системами пристроїв та вбудованими системами на рівні чіпу.

1.4.2 Sonarax

Sonarax – запатентована ультразвукова технологія передачі даних за допомогою звукових хвиль, яка забезпечує безпечний одноранговий зв'язок та бездротові підключення через будь-який пристрій з мікрофоном та динаміком. Протокол Sonarax використовує звукові хвилі, які є єдиним носієм зв' язку, що може міститися в певному місці, оскільки звукові хвилі не проходять крізь стіни, стелю, підлогу чи перегородки, що надає можливість виявити точне місце розташування звукового сигналу та перевіряє його точну фізичну присутність. Платформа Sonarax здатна розраховувати відстань з неймовірною точністю, використовуючи незначну модуляцію ефекту доплера в звукових хвилях. Ця модуляція, спільно із запатентованим алгоритмом дозволяє застосунку вимірювати відстань між користувачами в режимі реального часу.

1.4.3 CopSonic

CopSonic - це технологія, яка дозволяє спілкуватися та взаємодіяти між двома пристроями за допомогою звукових хвиль. Безпечна передача досягається за допомогою мікрофонів і динаміків. Зараз і в майбутньому мобільні телефони, планшети та розумні пристрої зможуть обмінюватися інформацією без проблем із сумісністю, на відміну від Bluetooth LE, NFC.

Аутентифікація та керування системою відбуваються на хмарному VOIP-сервері CopSonic. Мобільний пристрій зв'язується з CopSonic Cloud через мережу GSM. Відстань обміну коливається від одного сантиметра до декількох метрів.

У режимі польоту на смарт-пристроях (смартфоні, планшеті або смарт-годинниках) інформація обробляється в режимі реального часу на першому пристрої, щоб надсилатися через його динамік на мікрофон другого пристрою.

1.5Висновки до розділу 1

У першому розділі дипломного проєкту було описано потенціал технології обміну даними за допомогою звукових хвиль та обґрунтовано її актуальність. Описано принцип роботи даної технології та її аспектів.

Також описано три існуючі рішення для обміну даними за допомогою звукових хвиль – платформи LISNR, CopSonic та технологію Sonarax. Досвід розробників цих рішень свідчить про те, що технологія перспективна та затребувана на ринку.

РОЗДІЛ 2
ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вимоги до програмного забезпечення

2.1.1 Вимоги до системи в цілому

Системні вимоги наведені у таблиці 2.1.

Таблиця 2.1.

	Android	Windows
Версія ОС	8.0 або вище	10 v1909 або вище
ОЗП	1 Гб або вище	
Архітектура ЦП	Arm64	X64
ЦП	2-ядерний з тактовою частотою 1 ГГц або вище	
Вільне місце на накопичувачі	500 Мб або більше	
Роздільна здатність екрану	1280x720 або вище	
Додатково	Наявність звукових пристроїв введення та виведення	

2.1.2 Вимога до показників призначення

- ПК не повинен використовувати інші канали передачі даних, окрім звукового;
- час відгуку ПК для операцій навігації по графічному інтерфейсі не повинен перевищувати 5 сек.

2.1.3 Вимоги до захисту від несанкціонованого доступу

- забезпечення можливості шифрування даних, що надсилаються за допомогою алгоритму RSA для захисту від несанкціонованого втручання у процес обміну даними;
- засоби захисту не повинні істотно погіршувати основні функціональні характеристики ПК.

2.1.4 Вимоги до системних характеристик і функцій (завдань)

Забезпечення надсилання та отримання даних (текстових повідомлень, файлів тощо) за допомогою звукових хвиль.

2.2Вимоги до функцій (завдань)

2.2.1 Вимоги до функціональної структури системи (підсистеми і взаємодія)

Кожен з програмних засобів (як для ОС Windows, так і Android) ПК повинен мати три логічні рівні: рівень представлення, рівень бізнес-логіки та рівень обміну даними.

2.2.2 Вимоги до окремих функціональних підсистем

- на рівні представлення забезпечується взаємодія користувача з програмним засобом;
- на рівні бізнес-логіки забезпечується кодування даних у акустичний сигнал та декодування акустичного сигналу в дані за допомогою методу ЧМ. Кодова база бізнес-логіки має бути спільною для обох застосунків;
- на рівні обміну даними забезпечується введення та виведення акустичного сигналу за допомогою звукових пристроїв введення/виведення.

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до інформаційного забезпечення

Структура зберігання даних в ПК повинна складатися з наступних областей:

- область тимчасового зберігання даних;
- область вітрини даних.

2.3.2 Вимога до сумісності із суміжними системами

ПК не повинен бути закритим для суміжних систем і повинен підтримувати можливість експорту даних в суміжні системи.

2.3.3 Вимога до системи кодування і класифікації об'єктів і процесів

- забезпечення кодування даних у акустичний сигнал та декодування акустичного сигналу в дані за допомогою методу ЧМ;
- перелік назв класифікаторів об’єктів та процесів повинен бути у окремому файлі та бути спільним для обох програмних засобів ПК.

2.3Вибір архітектури

Архітектура програмного забезпечення системи відображає організацію або структуру системи та надає пояснення того, як вона працює. Система представляє набір компонентів, які виконують певну функцію або набір функцій. Серія архітектурних рішень впливає на якість, продуктивність, ремонтопридатність та загальний успіх системи. Неврахування загальних проблем та довгострокових наслідків може поставити систему під загрозу. Існує безліч моделей та принципів архітектури високого рівня, які зазвичай використовуються в сучасних системах. Їх часто називають

архітектурними стилями. Архітектура програмної системи рідко обмежується одним архітектурним стилем. Натомість поєднання стилів часто становить повну систему.

Для програмного комплексу для обміну даними за допомогою звукових хвиль найоптимальнішим є шаблон (патерн) MVU (Model-View-Update, ще відомий як Elm архітектура), який є шаблоном для проектування інтерактивних програм.

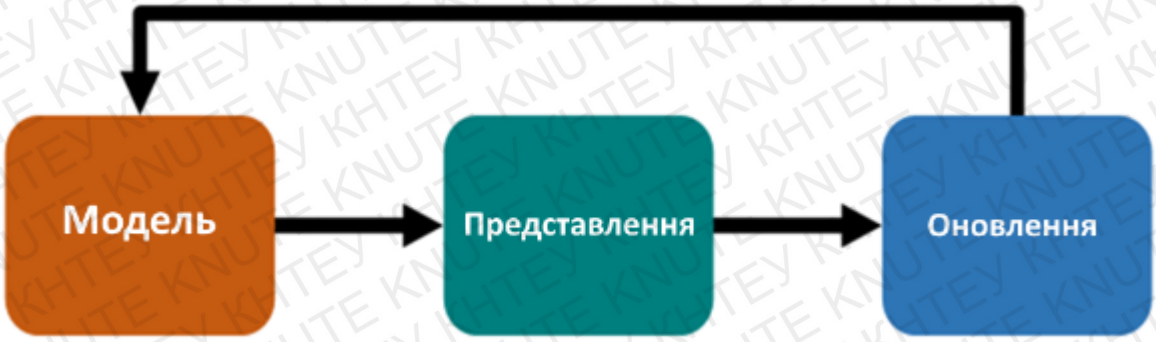

Рис. 2.1. Патерн MVU

Патерн MVU складається з трьох компонентів: Model (модель), View (представлення) та Update (оновлення):

- Модель: на відміну від інших патернів, таких як MVVM, модель не описує набір функціональних можливостей програми, а її стан та дані (стан моделі формують її дані);
- Представлення: є набором методів, які забезпечують побудову інтерфейсу користувача на основі тих даних, які зберігаються в моделі;
- Оновлення: єдине місце, де відбувається оновлення стану моделі.

Головною перевагою патерну MVU є його простота: він складається всього з трьох частин та описує, як вони взаємодіють між собою. В MVU існує лише один єдиний визначений спосіб обробки взаємодій та управління станом – і він забезпечує хорошу основу для модульності, повторного використання коду та тестування за замовчуванням.

Патерн MVVM дозволяє відокремити логіку програмного забезпечення від візуальної частини (представлення). Даний патерн є архітектурним. Патерн був представлений як модифікація патерну Presentation Model та був спрямований на розробку додатків в WPF. І хоча зараз патерн вийшов за межі WPF та використовується в багатьох різних технологіях, в тому числі розробці під Android та iOS, WPF являється доволі показовою технологією, що розкриває можливості даного патерну.

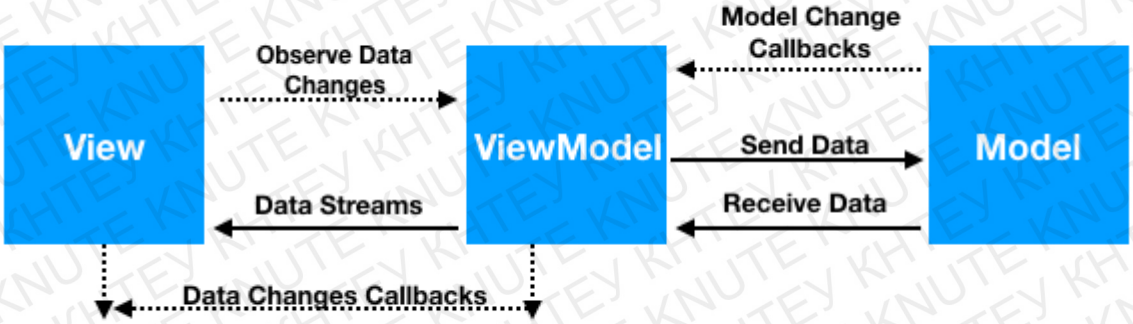

Рис. 2.2. Патерн MVVM

Патерн MVVM складається з трьох компонентів: Модель (Model), поделі представлення (ViewModel) та представлення (View):

- Модель: модель описує дані, що використовуються в додатку. Моделі можуть містити логіку, безпосередньо пов'язану з цими даними, наприклад логіку валідації властивостей моделі. Втім модель не повинна містити ніяку логіку, пов'язану з відображенням даних та взаємодією з візуальними елементами керування;
- Представлення: представлення визначає візуальний інтерфейс, за допомогою якого користувач взаємодіє з додатком.
- Модель представлення: пов'язує модель та представлення через механізм прив'язки даних. Модель представлення також містить логіку по отриманню даних з моделі, котрі потім передаються в представлення.

І також модель представлення визначає логіку по оновленню даних в моделі.

Патерн MVP (Minimal Viable Product) – тестова варсія товару, послуги чи сервісу з мінімальним набором функцій (іноді навіть лише однією), що несе цінність для кінцевого користувача.

MVP створюють для тестування гіпотез та перевірки життєпридатності продукту, наскільки він буде цінним на ринку.

Результати тестування мінімально життєпридатного продукту та зворотний зв'язок від цільової аудиторії допомагають зрозуміти, чи варто розвивати проект далі, які зміни слід внести в стратегію, а що залишити в початковому вигляді. Існують різні види MVP.

MVP Флінстоуна. Даний підхід передбачає імітацію наявності функціоналу, хоча насправді технічно його ніяк не реалізовано. MVP націлено на перевірку гіпотези, доказ життєпридатності обраної моделі розвитку.

Консьєрж MVP. Даний вид більше підходить для онлайн-сервісів, мета яких – автоматизувати рішення проблем кінцевої аудиторії. На початкових етапах реалізації продукту послуга надається вручну.

Розрізнений MVP. Метод використовують, коли ідею можна перевірити і реалізувати без розробки унікального програмного забезпечення. Замість цього збирають готові інструменти, поєднують в одну систему та представляють в єдиному інтерфейсі.

Продукт з одним параметром. Цю різновидність використовують найчастіше, коли є готовий продукт з мінімальним набором функцій (зазвичай однієї). Випуск продукту з однією функцією (параметром) дозволяє звузити цільову аудиторію, отримати зворотній зв'язок та проаналізувати його, після чого перейти до тестування.

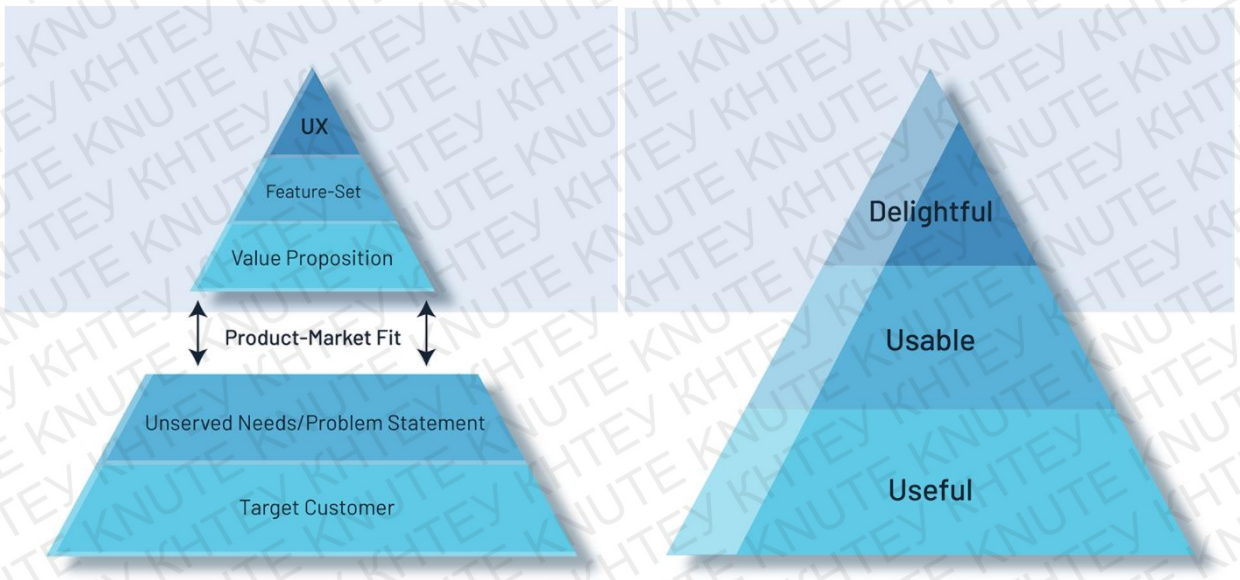

Рис. 2.3. Патерн MVP

Патерн модель-представлення-контролер (Model-view-controller, MVC) ділить дані ПЗ та керуючої логіки на три окремі компоненти. Модифікація кожного компоненту може проводитись незалежно.

Модель надає дані та реагує на команди контролеру, змінюючи свій стан. Модель не залежить від представлення (не може візуалізувати дані) і контролера (не має точок взаємодії з користувачем), просто надаючи доступ до даних та їх керуванням.

Представлення відповідає за відображення даних моделі користувачу, реагуючи на зміни моделі. Представлення не обробляє введені дані користувача.

Контролер інтерпретує дії користувача, сповіщуючи модель про необхідність змін. Він є «зв'язком» між користувачем та системою, контролює та направляє дані до системи і назад. Використовує модель і представлення для реалізації необхідної дії.

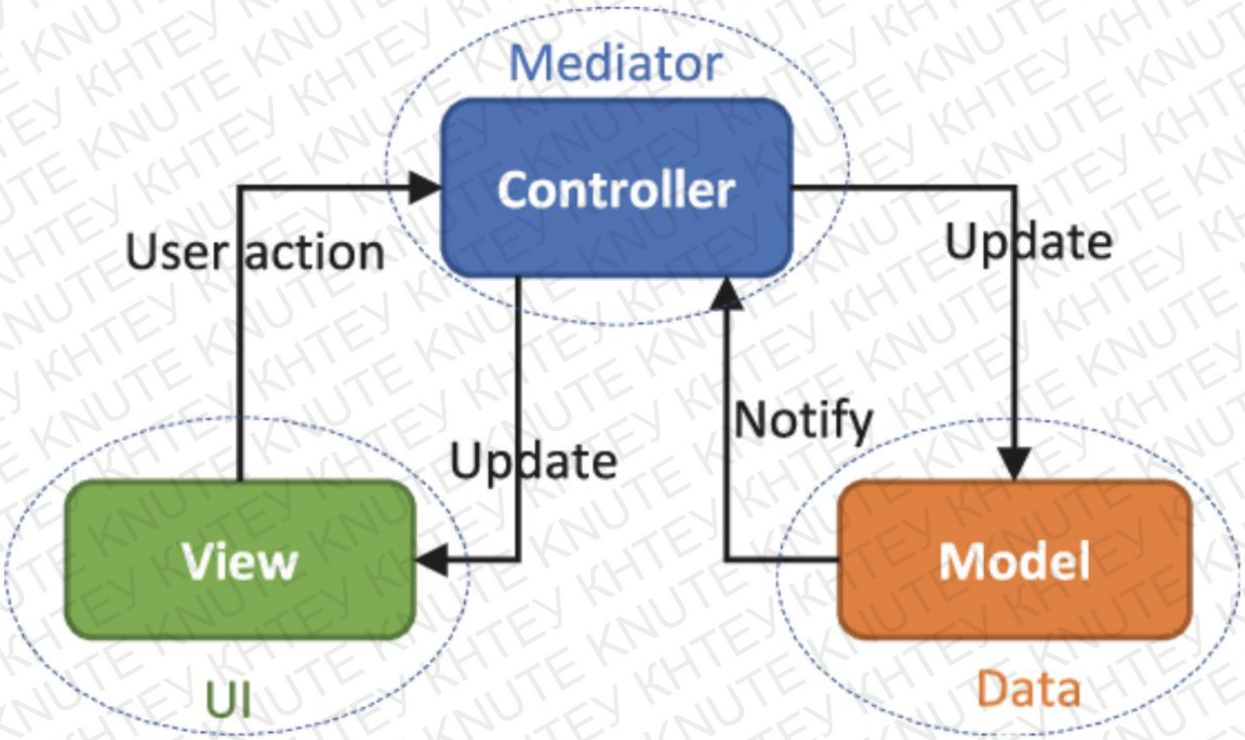

Рис. 2.4. Патерн MVC

2.5 Опис архітектури програмного забезпечення

ПК призначений для обміну даними за допомогою звукових хвиль – даними можуть бути різні текстові повідомлення, файли тощо, які можна як надсилати, так і одержувати. Дані кодуються в акустичний сигнал та відтворюються динаміком пристрою, або ж акустичний сигнал захоплюється мікрофоном та декодується у дані, в залежності від того, відбувається надсилання чи одержання даних відповідно. Якщо користувач вказав, що необхідно шифрувати дані, то перед кодуванням дані шифруються за допомогою алгоритму RSA, а після декодування – дешифруються. Алгоритм RSA є асиметричним криптографічним алгоритмом, що означає, що використовується дві пари ключів – приватний та публічний. Публічний ключ є відкритим для всіх та використовується для шифрування даних. У випадку використання шифрування користувачі ПК мають обмінятися публічними ключами, щоб мати змогу шифрувати та дешифрувати дані.

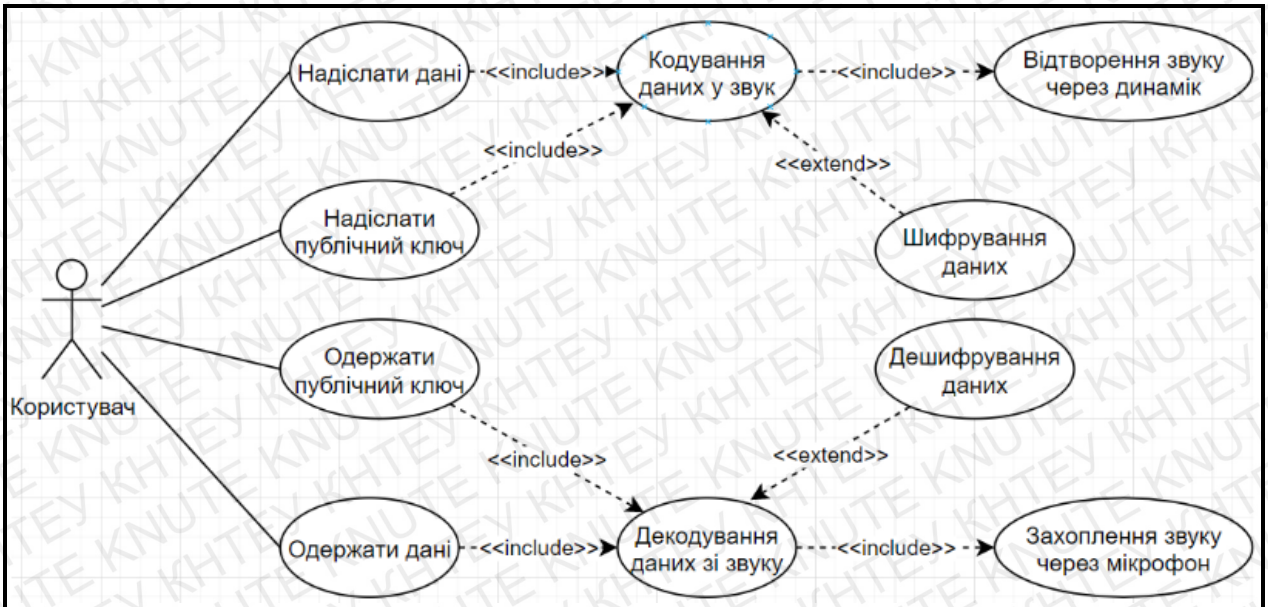

Рис. 2.2. Діаграма варіантів використання

Одразу після запуску ПЗ генерується публічний ключ, який згодом можна відправити іншому користувачеві, щоб той зміг шифрувати повідомлення для поточного користувача.

Процес роботи з ПЗ передбачає захоплення акустичного сигналу в пасивному режимі за допомогою мікрофона пристрою, поки очікується команда користувача. Отриманий акустичний сигнал декодується в дані, це може бути публічний ключ, який буде збережено для подальшої можливості шифрування повідомлень, або ж файл чи текстове повідомлення, які, до того ж, можуть бути зашифрованими, що потребуватиме дешифрування цих даних, одержані файл чи текстове повідомлення буде виведено користувачеві.

Користувач може надіслати публічний ключ або повідомлення, яке може бути файлом або текстом. Після введення файлу чи тексту користувач може зашифрувати ці дані, які буде закодовано в акустичний сигнал та відтворено за допомогою динаміка пристрою.

Описаний процес подано на блок-схемі алгоритму (рисунок 3.3)

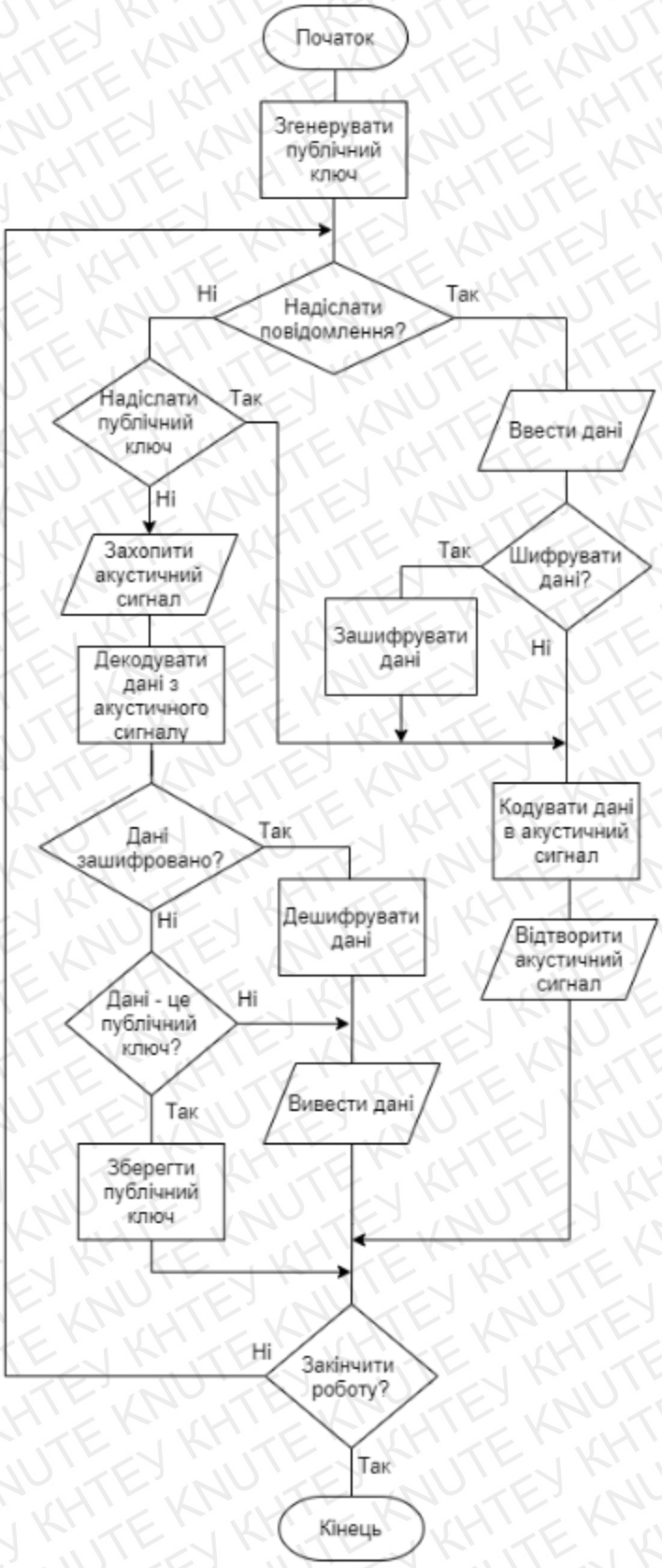

Рис. 2.3. Блок-схема алгоритму

Логічна архітектура ПК для обміну даними за допомогою звукових хвиль є трирівневою (рисунок 2.4).

Рис. 2.4. Логічна архітектура програмного забезпечення

Структура логічної архітектури програмного забезпечення:

- графічний інтерфейс користувача (різний для мобільних і настільних платформ);
- бізнес-логіка, а саме кодування даних у акустичний сигнал та декодування акустичного сигналу у дані. Найоптимальніша реалізація у вигляді бібліотеки, це забезпечить єдину кодову базу, що відповідає принципу повторного використання коду;
- рівень обміну даними: Відтворення акустичного сигналу через динаміки пристрою та захоплення звуку через мікрофон для перетворення його в акустичний сигнал забезпечується через системні API відповідних операційних систем.

Розгортання ПК можливе на мобільних (під керуванням ОС Android) та настільних (під керуванням ОС Windows) платформах. Вони можуть вільно обмінюватися даними за допомогою звукових хвиль незалежно від платформи відправника та отримувача.

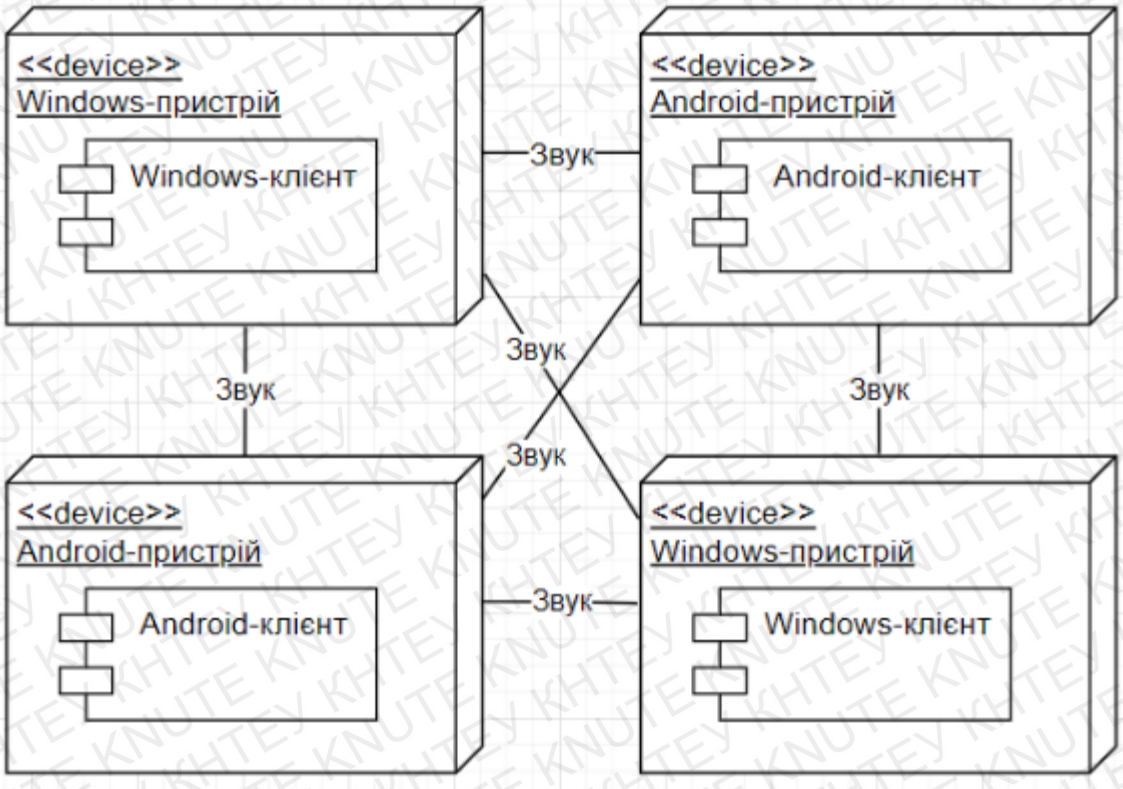

Рис. 2.5. Розгортання програмного забезпечення

2.6 Висновки до розділу 2

У другому розділі дипломного проекту описано функціональні та нефункціональні вимоги до ПЗ. Згідно з вимогами ПЗ повинен складатися з двох програмних засобів, один з яких буде працювати на ОС Windows, а інший – на ОС Android, при цьому рівень бізнес-логіки має бути спільним для обох платформ. Також зазначено, що має бути можливість використання шифрування даних за допомогою алгоритму RSA, а також – експорту даних в суміжні системи (для прикладу – в Microsoft Excel).

Також було обрано та описано архітектуру ПЗ. Визначено, що ПЗ матиме трирівневу архітектуру, яка складатиметься з рівня представлення, бізнес-логіки та рівня обміну даними. Для побудови ПЗ використовуватиметься патерн MVU. Даний розділ містить діаграми варіантів використання, розгортання, блок-схему алгоритму та схему логічної архітектури. Для розробки ПЗ обрано мову C#, яка є частиною платформи .NET.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Вибір мови програмування

Мова C# – це багатопарадигмова об'єктно-орієнтована та компонентно-орієнтована мова програмування зі строгою типизацією, розроблена для платформи .NET. Використання мови визначене стандартами ECMA-334 та ISO/IEC 23270:2006. Розроблена командою Microsoft Research під керівництвом Андерса Гейлсберга. Синтаксис мови близький до мов C++ та Java. Деякі риси (наприклад, строга статична типизація), наближують її структуру до Delphi (Object Pascal).

Згідно стандарту ECMA-334, цілі, поставлені при розробці мови C#, були такими:

- C# має бути простою, сучасною, об'єктно-орієнтованою мовою програмування;
- мова має підтримувати безпечні принципи програмування, такі як строга перевірка типів, перевірка меж масиву, виявлення спроб використання неініціалізованих змінних, і автоматичне прибирання сміття;
- можливість розробки програмних компонентів для розподілених систем;
- мова має підтримувати переносимість коду;
- підтримка національних мовних та інших особливостей має максимально бути простою.

Рис. 3.1. Платформа .NET 5

Базовий синтаксис C# схожий до інших C-подібних мов, таких як C, C++ та Java, зокрема:

- крапку з комою використовують для позначення кінця інструкції;
- фігурні дужки використовують для формування програмних блоків;
- значення змінним присвоюють за допомогою знаку "=", а для їх порівняння використовують "==";
- квадратні дужки використовують для індексації масивів.

Проте, C# має суттєві відмінності від розглянутих мов-попередників:

- краща переносимість між різними платформами, пов'язана з тим, що мова відповідає специфікації CLI;
- строга типизація робить мову безпечнішою. C# також підтримує логічний тип boolean;
- мова забезпечує використання властивостей у класах, роблячи роботу з об'єктами зручнішою та безпечнішою;

- програма на C# краще структурована завдяки групуванню коду в простори імен;
- обмежене використання вказівників робить безпечнішою роботу з пам'яттю.

3.2. Опис структури програмного забезпечення

Рішення програмного комплексу складається з шести проектів: DSL, BLL, SharedModel, PL_Windows, PL_Android та PL_Android.Android. Ієрархію цих проектів зображено на рисунку 3.1.

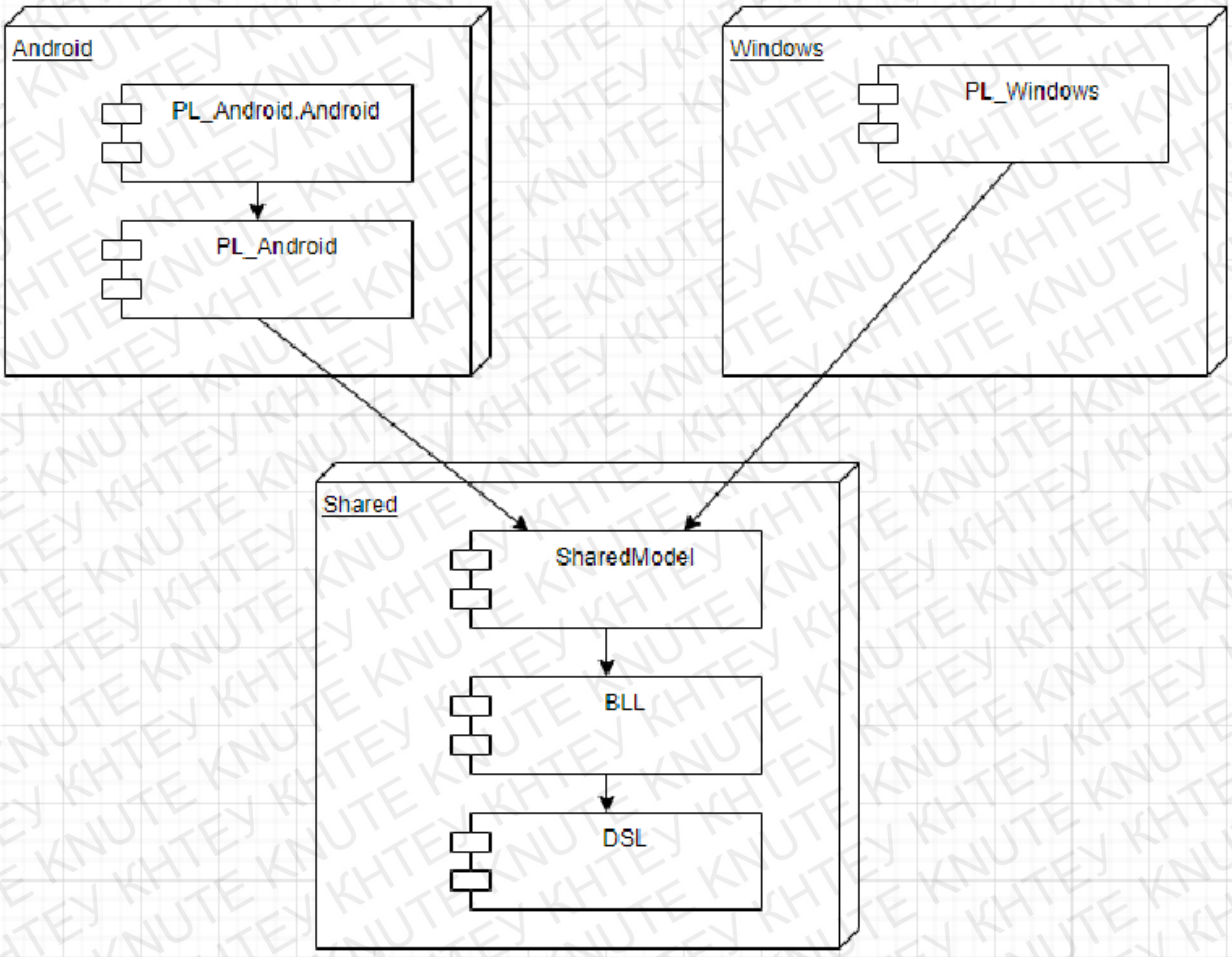

Рис. 3.2. Діаграма компонентів

У проекті DSL реалізовано рівень доступу до даних. Роботу зі звуковими пристроями захоплення та відтворення забезпечено за допомогою бібліотеки SDL2. Це вільно поширювана кросплатформна мультимедійна бібліотека, яка надає інтерфейс до графіки, звуку та пристроїв вводу. Даний проект типу Dynamic-Link Library (DLL) написано на мові C++.

У проекті BLL реалізовано бізнес-логіку. На цьому рівні дані кодуються в акустичний сигнал та декодуються з нього. За перетворення даних відповідає вільно поширювана бібліотека ggwave, у якій реалізовано алгоритми шістнадцятичастотної модуляції та ШПФ. Даний проект типу Class Library написано на мові C# (.Net Standard 2.1).

Рівень представлення даних складається з кількох проектів, а саме: SharedModel, PL_Windows, PL_Android та PL_Android.Android.

У проекті SharedModel реалізовано компонент «модель» (Model) патерну MVU. Модель зберігає поточні налаштування програмного засобу, забезпечує первинну обробку даних перед передачею їх рівню бізнес-логіки. Оскільки модель однакова як у програмному засобі для ОС Windows, так і для ОС Android, її було винесено як окремий проект для забезпечення принципу повторного використання ООП. Даний проект типу Class Library написано на мові C# (.Net Standard 2.1).

У проекті PL_Windows реалізовано інтерфейс програмного засобу з графічною підсистемою WPF на ОС Windows. На цьому рівні прописано відображення (View) та оновлення (Update) патерну MVU. Даний проект типу WPF Windows Application написано на мові C# (.NET 5).

Програмний засіб на ОС Android складається з двох підпроектів – PL_Android та PL_Android.Android з огляду на специфіку побудови застосунків для цієї ОС на мові C# у середовищі Visual Studio.

У проекті PL_Android реалізовано інтерфейс програмного засобу для ОС Android. На цьому рівні прописано відображення (View) та оновлення (Update) патерну MVU. Даний проект типу Xamarin.Forms написано на мові C# (.Net Standard 2.1).

У проекті PL_Android.Android містяться автоматично згенеровані компоненти, які необхідні для побудови програмного засобу для ОС Android. Даний проект типу Xamarin.Native написано на мові C# (.Net Standard 2.1).

Основні класи ПЗ зображено на рисунку 3.3.

Рис. 3.3. Діаграма класів

3.3 Інтерфейс програмного забезпечення

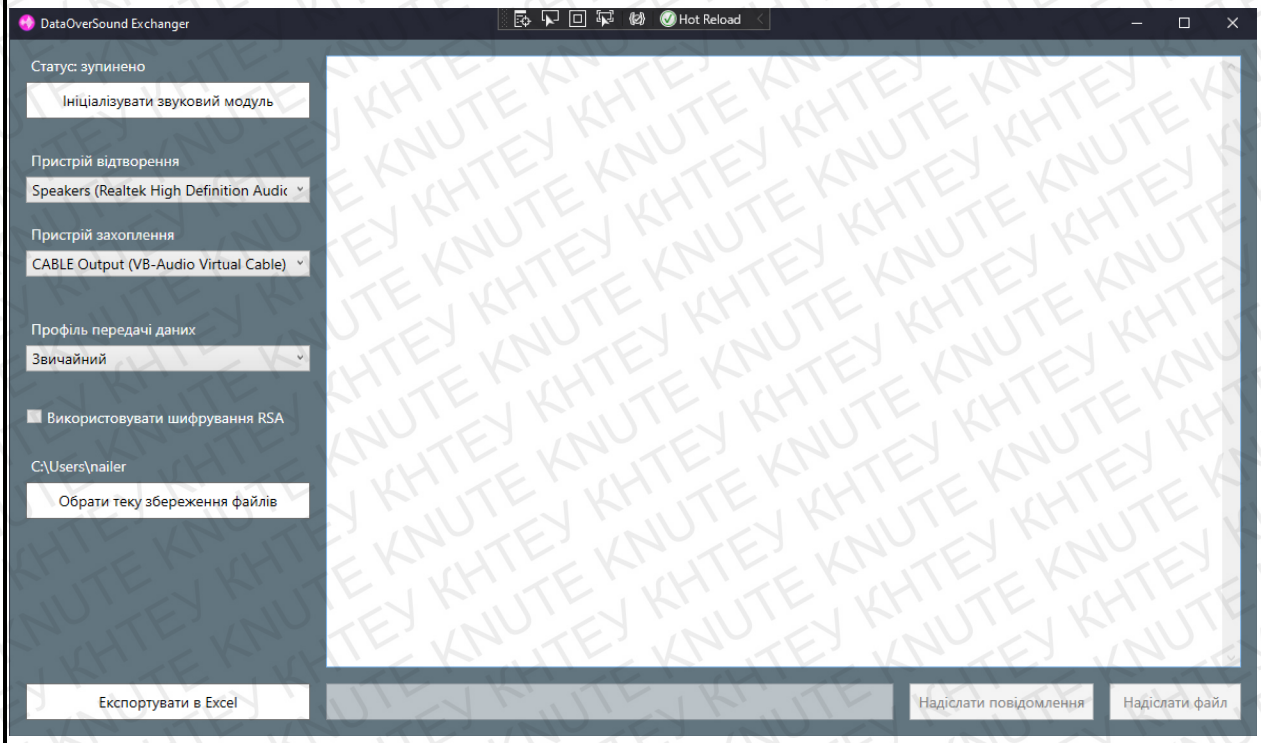

Рис. 3.4. Програмний засіб на ОС Windows

У лівій частині вікна розташовані елементи інтерфейсу, за допомогою яких здійснюється налаштування програмного засобу та показуються сервісні повідомлення.

У верхньому лівому куті вікна розташоване текстове поле «Статус» відображує поточний режим роботи програмного засобу: очікування (очікування вхідного повідомлення), одержання (одержання вхідного повідомлення), зупинено (програмний засіб не використовує звукові пристрої).

Одразу під тестовим полем розташована кнопка ініціалізації/деініціалізації звукового модуля з рівня обміну даними. Якщо звуковий модуль ініціалізовано, то стає можливим отримання і надсилання даних.

Нижче розташовані випадні списки для вибору пристроїв відтворення та захоплення звуку. Якщо в системі використовується кілька звукових

пристроїв, то користувач зможе обрати потрібний. Звуковий пристрій можна обрати лише коли звуковий модуль деініціалізовано («Статус: зупинено»).

Далі розташовано випадний список вибору профілю передачі даних. Всього їх три: звичайний (пропускна здатність близько 11 байт/с), швидкий (близько 16 байт/с), найшвидший (близько 33 байт/с). Профіль передачі даних теж можна обирати лише коли звуковий модуль зупинено.

Під випадним списком розташовано чек-бокс «Використовувати шифрування RSA», якщо його активовано, то надіслані дані й файли будуть шифруватися алгоритмом RSA. Одразу після активації цього чек-боксу програмний засіб надішле свій публічний ключ.

Нижче розташовано текстове поле, яке показує шлях збереження вхідних файлів, за цим шляхом також буде збережено експортовані дані. Шлях можна змінити за допомогою кнопки «Обрати теку для збереження файлів», натискання на котру відкриє діалогове вікно вибору директорії збереження.

У правій верхній частині вікна знаходиться текстовий блок, у який виводиться лог. Лог містить сповіщення про надіслані й отримані тестові повідомлення, файли, публічні ключі.

Під блоком логу в правій частині вікна розташовано кнопку «Надіслати файл», натискання на яку відкриє діалогове вікно вибору файлу. Після обрання користувачем файлу його буде надіслано.

З лівого боку під блоком логу розташовано текстове поле, в якому користувач може ввести текстове повідомлення. Після натискання кнопки «Надіслати повідомлення» введене повідомлення буде надіслано.

У лівому нижньому куті вікна розташовано кнопку «Експортувати дані в Excel», натискання на цю кнопку збереже лог у csv-файл, який можна відкрити у Microsoft Excel або іншому табличному процесорі (рисунок 3.5)

	A	B
1	Column1	Column2
2	20:46:28	Одержано текстове повідомлення: Use RSA please
3	20:46:41	Одержано публічний ключ
4	20:46:47	Надіслано публічний ключ
5	20:46:54	Надіслано зашифроване текстове повідомлення: okay
6	20:47:18	Надіслано зашифрований файл: report.txt

Рис. 3.5. Експортовані дані в Excel

Інтерфейс програмного засобу на ОС Windows адаптивний, тобто розмір вікна можна змінювати для комфортнішої взаємодії з програмним засобом.

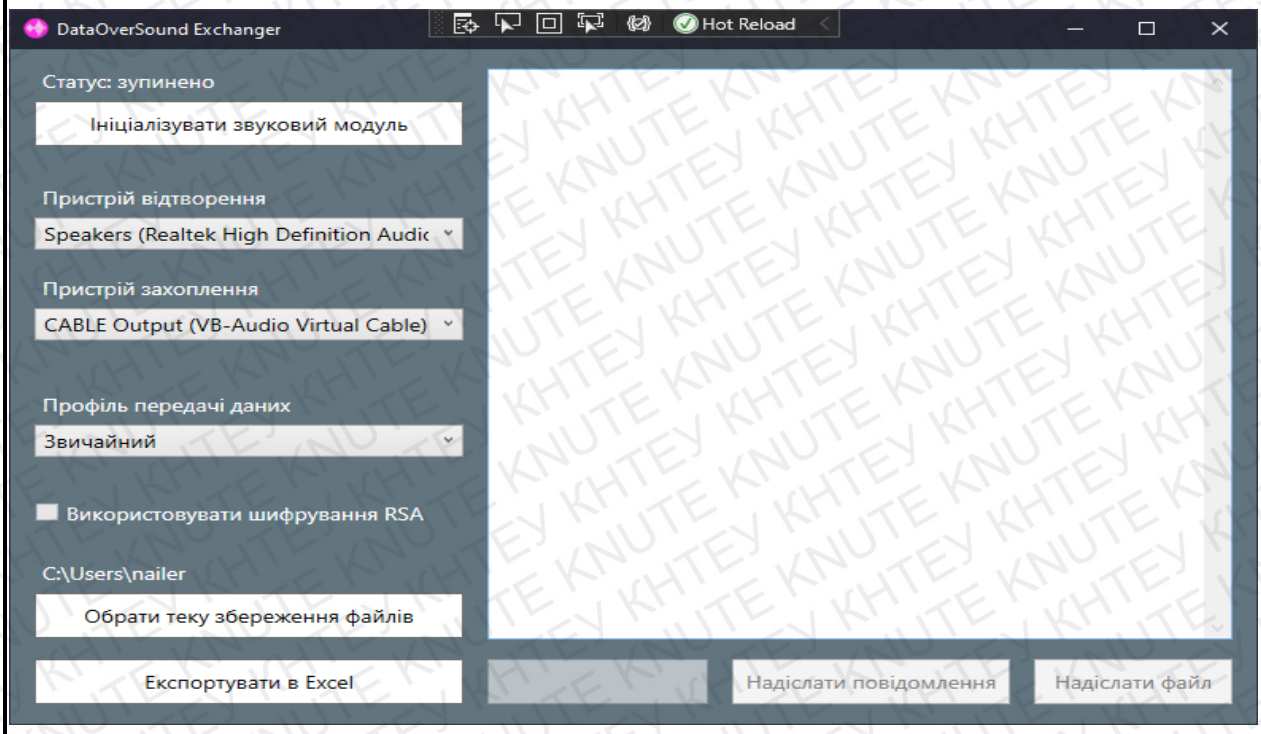

Рис. 3.6. Вікно зміненого розміру

Інтерфейс програмного засобу на ОС Android має аналогічні елементи інтерфейсу за винятком того, що випадні списки вибору звукових пристроїв відсутні з огляду на відсутність потреби обирати звукові пристрої на даній платформі (рисунок 3.7).

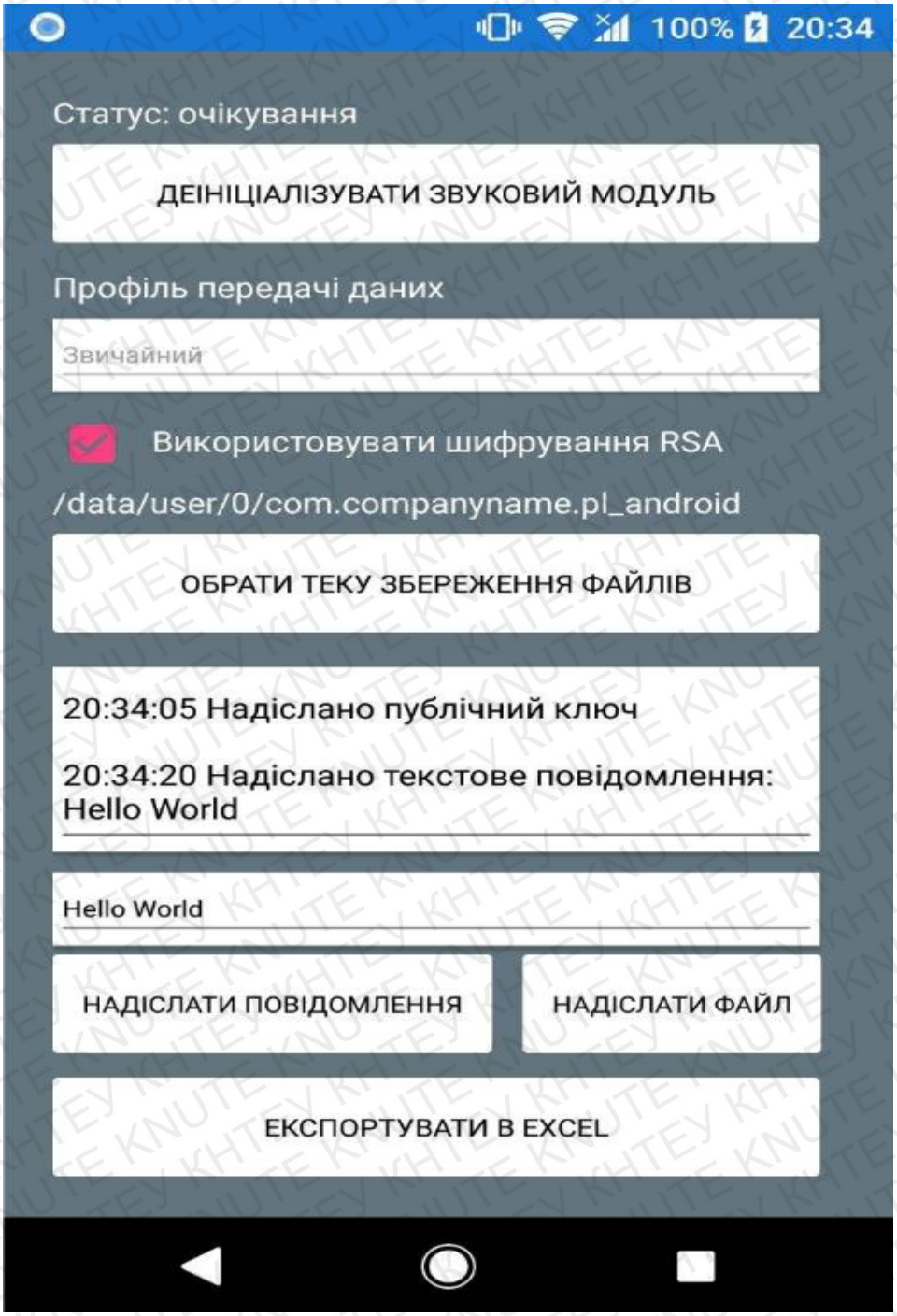

Рис. 3.7. Програмний засіб на ОС Android

3.4. Висновок до розділу 3

У третьому розділі дипломного проекту реалізовано прототип ПК для обміну даними за допомогою звукових хвиль. Описано деталі реалізації

кожного з проектів, їхню ієрархію відображено на діаграмі компонентів. Також наведено скріншоти інтерфейсу кожного з програмних засобів та описано усі компоненти інтерфейсу.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Даний дипломний проект має мету поліпшення однорангової міжмашинної комунікації шляхом використання акустичного каналу передачі даних. Міжмашинна комунікація здійснюється за допомогою звукових хвиль, які відтворюються звуковими пристроями виведення та захоплюються звуковими пристроями введення комп'ютерів та смартфонів.

Для створення програмного комплексу було проведено аналіз предметної області, проаналізовано існуючі системи обміну даними за допомогою звукових хвиль. Існуючі системи мають закритий код та поширюються по корпоративній ліцензії, але досвід розробників та користувачів цих систем показав – сфер застосування подібних систем безліч. Це, наприклад, авторизація та автентифікація користувачів, платіжні операції, звукові QR-коди тощо.

Інтерфейс створених програмних засобів програмного комплексу є простим та зрозумілим, обрана архітектура дозволяє легко та швидко супроводжувати програмний комплекс без необхідності працювати над логікою окремих програмних засобів.

Розроблений програмний комплекс відповідає всім описаним вимогам, стандартам та нормам у проектуванні програм.

Впровадження програмного комплексу дозволить створити захищений локальний канал обмін даними між різними пристроями на вже існуючому обладнанні без додаткових витрат.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sending Data Over Sound: How and Why? [Електронний ресурс] // lectronic Design. – 2019. – Режим доступу до ресурсу: https://www.electronicdesign.com/industrial-utomation/article/21808186/sending-data-over-sound-how-and-why.
2. Binary Frequency Shift Keying [Електронний ресурс] // Northern Illinois University – Режим доступу до ресурсу: https://www.niu.edu/remote-lab/resources/graphical-user-interfaces/frequency-shift.shtml.
3. Multiple Frequency-Shift Keying (MFSK) [Електронний ресурс] // Technopedia – Режим доступу до ресурсу: https://www.techopedia.com/definition/14828/multiple-frequency-shift-keying-mfsk.
4. Fast Fourier Transformation FFT - Basics [Електронний ресурс] // NTi Audio – Режим доступу до ресурсу: https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft.
5. LISNR [Електронний ресурс] – Режим доступу до ресурсу: https://lisnr.com/.
6. Sonarax [Електронний ресурс] – Режим доступу до ресурсу: https://www.sonarax.com/
7. Software Architecture & Software Security Design [Електронний ресурс] // Synopsys – Режим доступу до ресурсу: https://www.synopsys.com/glossary/what-is-software-architecture.html.

8. The Elm Architecture [Електронний ресурс] // dennisreimann – Режим доступу до ресурсу: https://dennisreimann.de/articles/elm-architecture-overview.html.
9. GeeksForGeeks [Електронний ресурс] // RSA Algorithm in Cryptography – Режим доступу до ресурсу: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/.
10. Коноваленко І.В. Програмування мовою C# 7.0 : навчальний посібник / Коноваленко І.В., Марущак П.О., Савків В.Б. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя 2017 – 300 с.
11. What's new in .NET [Електронний ресурс] // Microsoft. – 2020. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/core/dotnet-five.
12. Gerganov G. ggwave [Електронний ресурс] / Georgi Gerganov // GitHub – Режим доступу до ресурсу: https://github.com/ggerganov/ggwave.

ДОДАТКИ

ДОДАТОК А

Код класу Model.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;

namespace SharedModel
{
    /// <summary>
    /// Модель патерну MVU
    /// </summary>
    public class Model : IDisposable
    {
        public delegate void UpdateHandler();
        //подія оновлення
        public event UpdateHandler OnUpdate;

        public delegate void NotificationHandler(string text);
        //подія сповіщення
        public event NotificationHandler OnNotification;

        //список пристроїв відтворення та захоплення
        public List<string> PlaybackDevices { get; }
        public List<string> CaptureDevices { get; }

        //шлях збереження
        public string SavePath { set; get; }

        //ідентифікатори пристроїв захоплення та відтворення
        public int CaptureId { set; private get; }
        public int PlaybackId { set; private get; }

        //протокол передачі
        public int TxProtocolId { set; private get; }

        //стан отримання
        public bool IsReceiving { get; private set; } = false;

        private bool useRSA = false;
        //використання шифрування
        public bool UseRSA
        {
            set
            {
                useRSA = value;
                if (useRSA)
                    SendKey();
            }
        }

        //стан моделі
        public bool IsWorking { get; private set; } = false;

        private string publicRSAkey; //публічний ключ моделі
        private string privateRSAkey; //приватний ключ моделі

        private string remoteRSAkey = null; //віддалений публічний ключ
```

```
        const int rsaSize = 384; //довжина RSA-ключа

        private BLL.BLL bll;

        /// <summary>
        /// Конструктор моделі
        /// </summary>
        public Model()
        {
            //отримання стандартної теки збереження
            SavePath = Environment.GetFolderPath(Environment.SpecialFolder.Personal);

            //збереження згенерованих ключів
            using (var rsa = new RSACryptoServiceProvider(rsaSize))
            {
                publicRSAkey = rsa.ToXmlString(false);
                privateRSAkey = rsa.ToXmlString(true);
            }

            if (System.Environment.OSVersion.Platform == PlatformID.Win32NT)
                bll = new BLL.BLL();
            else
                bll = null;

            if (bll != null)
            {
                //отримання пристроїв
                CaptureDevices = bll.CaptureDevices.ToList();
                PlaybackDevices = bll.PlaybackDevices.ToList();

                //подія початку отримання даних
                bll.DataReceiving += () =>
                {
                    IsReceiving = true;
                    OnUpdate?.Invoke();
                };

                //подія закінчення отримання даних
                bll.DataReceived += (inData) =>
                {
                    IsReceiving = false;

                    try
                    {
                        var message = Message.GetMessage(inData); //конвертування json в
повідомлення

                        UnicodeEncoding byteConverter = new UnicodeEncoding();

                        string data;

                        //якщо використовується шифрування
                        if (message.E)
                        {
                            using (var rsa = new RSACryptoServiceProvider(rsaSize))
                            {
                                rsa.FromXmlString(privateRSAkey);

                                data =
byteConverter.GetString(rsa.Decrypt(byteConverter.GetBytes(message.D), false));
                            }
                        }
                        else
                        {
```

```
                        data = message.D;
                    }

                    //якщо тип - текст
                    if (message.T == "Text")
                    {
                        //сповіщення про отримання тексту
                        OnNotification?.Invoke($"{DateTime.Now.ToString("HH:mm:ss")}
Одержано {(message.E ? "зашифроване " : "")}текстове повідомлення: {data}");
                    }
                    //якщо тип - ключ
                    else if (message.T == "Key")
                    {
                        //підставлення потрібних тегів
                        remoteRSAkey = $"<RSAKeyValue><Modulus>{data.Replace("/ME",
"</Modulus><Exponent>")}</Exponent></RSAKeyValue>";

                        //сповіщення про отримання ключа
                        OnNotification?.Invoke($"{DateTime.Now.ToString("HH:mm:ss")}
Одержано публічний ключ");
                    }
                    //інакше тип - файл
                    else
                    {
                        //отримання байтів
                        var bytes = byteConverter.GetBytes(data);

                        //запис байтів у файл
                        File.WriteAllBytes(Path.Combine(SavePath, message.T), bytes);

                        //сповіщення про отримання файлу
                        OnNotification?.Invoke($"{DateTime.Now.ToString("HH:mm:ss")}
Одержано {(message.E ? "зашифрований " : "")}файл: {message.T}");
                    }
                }
                catch { }
            };
        }
    }

    /// <summary>
    /// Надіслати публічний ключ
    /// </summary>
    public void SendKey()
    {
        var m = new Message
        {
            E = false,
            T = "Key",
            //заміна стандартних тегів для скорочення повідомлення
            D = publicRSAkey
            .Replace("<RSAKeyValue><Modulus>", "")
            .Replace("</Exponent></RSAKeyValue>", "")
            .Replace("</Modulus><Exponent>", "/ME")
        };

        bll?.SendData(m.ToString(), (BLL.TxProtocolId)TxProtocolId); //надсилання
ключа

        //сповіщення про надсилання
        OnNotification?.Invoke($"{DateTime.Now.ToString("HH:mm:ss")} Надіслано
публічний ключ");
    }

    /// <summary>
```

```
        /// Надіслати текст
        /// </summary>
        /// <param name="text">Текст, що надсилається</param>
        public void SendText(string text)
        {
            //чи не порожній текст
            if (string.IsNullOrWhiteSpace(text))
                return;

            Message message;

            if (useRSA && remoteRSAkey != null) //якщо використовується шифрування і є публічний ключ
            {
                UnicodeEncoding byteConverter = new UnicodeEncoding();

                var data = byteConverter.GetBytes(text); //отримання байтів з тексту

                using (var rsa = new RSACryptoServiceProvider(rsaSize))
                {
                    rsa.FromXmlString(remoteRSAkey); //використання публічного ключа

                    var encryptedData = rsa.Encrypt(data, false); //шифрування даних

                    var sdata = byteConverter.GetString(encryptedData); //конвертування байтів у рядок

                    //ініціалізація повідомлення
                    message = new Message
                    {
                        E = true,
                        T = "Text",
                        D = text
                    };
                }
            }
            else
            {
                //ініціалізація повідомлення
                message = new Message
                {
                    E = false,
                    T = "Text",
                    D = text
                };
            }

            bll?.SendData(message.ToString(), (BLL.TxProtocolId)TxProtocolId); //надсилання повідомлення

            //сповіщення про надсилання
            OnNotification?.Invoke($"{DateTime.Now.ToString("HH:mm:ss")} Надіслано {(message.E ? "зашифроване " : "")}текстове повідомлення: {text}");
        }

        /// <summary>
        /// Надіслати файл
        /// </summary>
        /// <param name="path">Шлях до файлу</param>
        public void SendFile(string path)
        {
            var bytes = File.ReadAllBytes(path); //прочитати байти файлу

            UnicodeEncoding byteConverter = new UnicodeEncoding();
```

```
            Message message;

            if (useRSA && remoteRSAkey != null) //кщо використовується шифрування і є публічний ключ
            {
                using (var rsa = new RSACryptoServiceProvider(rsaSize))
                {
                    rsa.FromXmlString(remoteRSAkey); //використання публічного ключа

                    var encryptedData = rsa.Encrypt(bytes, false); //шифрування даних

                    var sdata = byteConverter.GetString(encryptedData); //конвертування байтів у рядок

                    //ініціалізація повідомлення
                    message = new Message
                    {
                        E = true,
                        T = Path.GetFileName(path),
                        D = byteConverter.GetString(bytes)
                    };
                }
            }
            else
            {
                //ініціалізація повідомлення
                message = new Message
                {
                    E = false,
                    T = Path.GetFileName(path),
                    D = byteConverter.GetString(bytes)
                };
            }

            bll?.SendData(message.ToString(), (BLL.TxProtocolId)TxProtocolId); //надсилання повідомлення

            //сповіщення про надсилання
            OnNotification?.Invoke($"{DateTime.Now.ToString("HH:mm:ss")} Надіслано {(message.E ? "зашифрований " : "")}файл: {message.T}");
        }

        /// <summary>
        /// Ініціалізація моделі
        /// </summary>
        public void Init()
        {
            IsWorking = !IsWorking;

            if (IsWorking)
                bll?.Start(PlaybackId, CaptureId); //якщо не працює - запустити
            else
                bll?.Stop(); //якщо працює - зупинити
        }

        //Метод знищення моделі
        public void Dispose()
        {
            bll?.Dispose();
        }
    }
}
```

Код класу Message.cs

```
using Newtonsoft.Json;

namespace SharedModel
{
    /// <summary>
    /// Повідомлення, яке передається
    /// </summary>
    class Message
    {
        /// <summary>
        /// Позначає, чи зашифровано повідомлення
        /// </summary>
        public bool E { get; set; }

        /// <summary>
        /// Тип повідомлення
        /// </summary>
        public string T { get; set; }

        /// <summary>
        /// Дані, які необхідно передати
        /// </summary>
        public string D { get; set; }

        /// <summary>
        /// Серіалізація
        /// </summary>
        /// <returns>JSON-рядок</returns>
        public override string ToString()
        {
            return JsonConvert.SerializeObject(this);
        }

        /// <summary>
        /// Десеріалізація
        /// </summary>
        /// <param name="text">JSON-рядок</param>
        /// <returns>Повідомлення</returns>
        public static Message GetMessage(string text)
        {
            return JsonConvert.DeserializeObject<Message>(text);
        }
    }
}
```

Код класу Update.cs

```
using System.Windows;

namespace PL_Windows
{
    public partial class View : Window
    {
        /// <summary>
        /// Метод Оновлення патерну MVU
        /// </summary>
        public void Update()
        {
            this.Dispatcher.Invoke(() =>
            {
                //шлях до теки збереження файлів
                SavePathLabel.Content = System.IO.Path.GetDirectoryName(model.SavePath);

                //збереження обраного пристрою захоплення, відтворення, чи протоколу
                model.CaptureId = CaptureDeviceCB.SelectedIndex;
                model.PlaybackId = PlaybackDeviceCB.SelectedIndex;
                model.TxProtocolId = TransmissionProtocolCB.SelectedIndex;

                //стандартне значення пристрою захоплення
                CaptureDeviceCB.ItemsSource = model.CaptureDevices;

                //стандартне значення пристрою відтворення
                PlaybackDeviceCB.ItemsSource = model.PlaybackDevices;

                //чи працює модель
                if (model.IsWorking)
                {
                    //можна використовувати шифрування, надсилати повідомлення і файли
                    MessageTB.IsEnabled = true;
                    SendBtn.IsEnabled = true;
                    SendFileBtn.IsEnabled = true;
                    RsaCB.IsEnabled = true;

                    StatusLabel.Content = $"Статус: { (model.IsReceiving ? "одержання" :
"очікування") }";

                    //не можна змінити профіль передачі чи пристрій
                    TransmissionProtocolCB.IsEnabled = false;
                    CaptureDeviceCB.IsEnabled = false;
                    PlaybackDeviceCB.IsEnabled = false;

                    PlayPauseBtn.Content = "Деініціалізувати звуковий модуль";
                }
                else
                {
                    //не можна використовувати шифрування, надсилати повідомлення і файли
                    MessageTB.IsEnabled = false;
                    SendBtn.IsEnabled = false;
                    SendFileBtn.IsEnabled = false;
                    RsaCB.IsEnabled = false;

                    StatusLabel.Content = $"Статус: зупинено";

                    //можна змінити профіль передачі чи пристрій
                    TransmissionProtocolCB.IsEnabled = true;
                    CaptureDeviceCB.IsEnabled = true;
                    PlaybackDeviceCB.IsEnabled = true;

                    PlayPauseBtn.Content = "Ініціалізувати звуковий модуль";
```

```
                }
            });
        }
    }
}
```

Код класу BLL.cs

```
using System;
using System.Runtime.InteropServices;
using System.Threading;

namespace BLL
{
    static class p
    {
        public const string lib = "DSL.dll"; //Назва бібліотеки
    }

    /// <summary>
    /// Бізнес-логіка
    /// </summary>
    public class BLL : IDisposable
    {
        public delegate void DataHandler(string data);
        /// <summary>
        /// Подія "Дані отримано"
        /// </summary>
        public event DataHandler DataReceived;

        public delegate void ReceivingHandler();
        /// <summary>
        /// Подія отримання даних
        /// </summary>
        public event ReceivingHandler DataReceiving;

        private static string[] captureDevices;
        /// <summary>
        /// Список пристроїв захоплення
        /// </summary>
        public string[] CaptureDevices { get { return captureDevices; } }

        private static string[] playbackDevices;
        /// <summary>
        /// Список пристроїв відтворення
        /// </summary>
        public string[] PlaybackDevices { get { return playbackDevices; } }

        /// <summary>
        /// Статичний конструктор ініціалізації бізнес-логіки
        /// </summary>
        static BLL()
        {
            captureDevices = null;
            playbackDevices = null;

            var devices = GGWaveBindings.GetAvailableDevices(); //отримання списку
пристроїв

            if (devices.CaptureDevicesNum > 0)
            {
                captureDevices = new string[devices.CaptureDevicesNum];

                for (int i = 0; i < devices.CaptureDevicesNum; i++)
                {
                    var ptr = GGWaveBindings.GetCaptureDevice(i); //отримання деталей про
пристрій

                    captureDevices[i] = Marshal.PtrToStringAnsi(ptr); //копіювання даних
з С-вказівника
```

```
            }
        }

        if (devices.PlaybackDevicesNum > 0)
        {
            playbackDevices = new string[devices.PlaybackDevicesNum];

            for (int i = 0; i < devices.PlaybackDevicesNum; i++)
            {
                var ptr = GGWaveBindings.GetPlaybackDevice(i);  //отримання деталей
про пристрій

                playbackDevices[i] = Marshal.PtrToStringAnsi(ptr); //копіювання даних
з С-вказівника
            }
        }
    }

    static object locker = new object();

    /// <summary>
    /// Надіслати дані
    /// </summary>
    /// <param name="data">Дані, що надсилаються</param>
    /// <param name="txProtocol">Протокол передачі даних</param>
    public void SendData(string data, TxProtocolId txProtocol)
    {
        var ptr = Marshal.StringToHGlobalAnsi(data); //створення динамічного
вказівника

        lock (locker)
        {
            GGWaveBindings.SendData(data.Length, ptr, (int)txProtocol, 10);
//надсилання даних
        }

        Marshal.FreeHGlobal(ptr); //очищення вказівника
    }

    static bool flag = false;
    private Thread thread;
    /// <summary>
    /// Основний цикл
    /// </summary>
    /// <param name="playbackId">Пристрій відтворення</param>
    /// <param name="captureId">Пристрій захоплення</param>
    /// <returns>Стан</returns>
    public bool Start(int playbackId, int captureId)
    {
        var status = GGWaveBindings.GGWave_Init(playbackId, captureId);
//ініціалізація бібліотеки

        if (!status) return false;

        flag = true;

        thread = new Thread(() => //основний потік звукового пристрою
        {
            bool isReceiving = false;

            while (flag)
            {
                lock (locker)
                {
                    GGWaveBindings.GGWave_MainLoop(); //цикл бібліотеки
```

```
                    if (GGWaveBindings.IsReceiving() && !isReceiving) //перевірка чи
почали захоплюватися дані
                    {
                        isReceiving = true;

                        DataReceiving?.Invoke(); //виклик події
                    }
                    else if (!GGWaveBindings.IsReceiving() && isReceiving)
//перевірка чи захопилися дані
                    {
                        isReceiving = false;

                        IntPtr ptr = Marshal.AllocHGlobal(2048); //ініціалізація
вказівника

                        GGWaveBindings.GetData(ptr); //отримання даних

                        var data = Marshal.PtrToStringAnsi(ptr); //отримання даних з
вказівника

                        Marshal.FreeHGlobal(ptr); //очищення вказівника

                        DataReceived?.Invoke(data); //подія що дані захопилися
                    }
                }

                try
                {
                    Thread.Sleep(1);
                }
                catch { }
            }
        });
        thread.Start();

        return true;
    }

    /// <summary>
    /// Призупинити цикл
    /// </summary>
    /// <returns>Стан</returns>
    public bool Stop()
    {
        flag = false;

        thread.Interrupt();

        return GGWaveBindings.GGWave_DeInit();
    }

    /// <summary>
    /// Знищити об'єкт
    /// </summary>
    public void Dispose()
    {
        GGWaveBindings.GGWave_Destroy(); //знищення об'єкта бібліотеки
    }
}
}
```