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
“___” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-23М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Подкопаєв Д.М.
«____» _____ 2024 р.

Керівник проекту
доктор технічних наук, професор
_____ Мелешко Є.В.
«____» _____ 2024 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
«_» _____ 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Подкопаєву Дмитру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows*

2. Керівник роботи *Мелешко Єлизавета Владиславівна, доктор техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №19-13від 07.08.2024 року

3. Строк подання роботи до захисту *02.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Дослідження та програмна реалізація системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows*

4. 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. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Дореньска А.О.	05.10.2024	14.11.2024
Охорона праці	Марченко К.М.,к.т.н.,доцент	06.10.2024	16.11.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024 р.	
9.	Оформлення ПЗ	17.11.2024 р.	
10.	Попередній захист роботи	02.12.2024 р.	

Дата видачі завдання
«_»_____2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«_»_____ 2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Подкопаєв Д.М. Дослідження та програмна реалізація системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення для використання Android-планшета як графічного планшета для комп'ютера з ОС Windows

Метою розробки є дослідження та програмна реалізація системи, що дозволяє використовувати Android-планшет як графічного планшета для комп'ютера з ОС Windows.

Об'єктом дослідження є процес взаємодії між андроїд планшетом так комп'ютером з ОС Windows для забезпечення функціональності графічного планшета.

Предметом дослідження є методи передачі даних між операційними системами, методи обробки введення стилуса, методи емуляції графічного планшета в Windows.

Методи дослідження базуються на методах розробки кросплатформного програмного забезпечення, методах мережевої взаємодії, методах обробки введення користувача, методах оптимізації передачі даних.

Результат роботи – програмна реалізація системи, що дозволяє використовувати Android-планшет як графічний планшет для комп'ютера з ОС Windows.

Розроблено зручний інтерфейс для обох платформ. Наведені інструкції по встановленню та використанню програмних засобів.

Програма може використовуватися на Android-планшетах з Android 8.0 і вище та комп'ютерах з ОС Windows 10/11.

Програму розроблено на мовах програмування Kotlin та C#.

Ключові слова: комп'ютерна інженерія, графічний планшет, Android, Windows , мережева взаємодія.

ABSTRACT

Podkopaiev D.M. Research and software implementation of a system for using an Android tablet as a graphic tablet for a Windows computer. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the second (master's) level of higher education, software for using an Android tablet as a graphics tablet for a Windows computer was developed.

The purpose of the development is to study and implement a system that allows the use of an Android tablet as a graphic tablet for a Windows computer.

The object of research is the process of interaction between an Android tablet and a Windows computer to ensure the functionality of a graphic tablet.

The subject of the study is the methods of data transfer between operating systems, methods of processing stylus input, methods of emulating a graphic tablet in Windows.

Research methods are based on methods of cross-platform software development, methods of network interaction, methods of user input processing, methods of data transfer optimization.

The result of the work is a software implementation of the system that allows using an Android tablet as a graphic tablet for a Windows computer.

A user-friendly interface for both platforms has been developed. Instructions for installing and using the software are provided.

The software can be used on Android tablets with Android 8.0 and higher and computers with Windows 10/11.

The software is developed in Kotlin and C# programming languages.

Keywords: computer engineering, graphic tablet, Android, Windows, networking.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 4
ВСТУП 5
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 8
1.1 Призначення системи 8
1.2 Область застосування 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 10
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 10
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 17
2.3 Розгорнута постановка завдання 21
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 23
3.1 Опис функціонування системи 23
3.2 Розробка структурної схеми 24
3.3 Розробка функціональної схеми 26
3.4 Розробка діаграми процесів 28
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 30
4.1 Блок-схеми та опис алгоритмів функціонування системи 30
4.2 Захист розробленого програмного забезпечення 38
5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 40
6 НАУКОВА НОВИЗНА 43

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ ... 44
7.1 Визначення цільової аудиторії кінцевого готового продукту. ... 44
7.2 Оцінка привабливості шляхом застосування методів експертних оцінок . 45
7.3 Вибір методу оцінки вартості ПЗ ... 46
7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості ... 47
7.5 Пропозиція алгоритму просування проєкту розробки ПЗ ... 48
7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ ... 49
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ ... 52
8.1 Вступ ... 52
8.2 Аналіз санітарно-гігієнічних умов праці на рабочому місці програміста . 53
8.3 Розробка заходів з поліпшення стану охорони праці ... 56
8.4 Техніка безпеки та протипожежна профілактика ... 58
8.5 Розрахункова частина ... 60
8.6 Висновок ... 64
9 ОСНОВНІ ВИСНОВКИ ... 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ... 66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

UDP – протокол датаграм користувача;

TCP – протокол управління передачею;

ОС – операційна система;

ПЗ – програмне забезпечення;

PC – персональний комп’ютер;

API – прикладний програмний інтерфейс;

USB – універсальна послідовна шина.

ВСТУП

Актуальність теми. У сучасному світі цифрових технологій постійно зростає потреба в ефектних та доступних інструментах для творчості та роботи з графікою. Графічні планшети стали невід'ємною частиною робочого процесу дизайнерів, художників та інших креативних професіоналів. Однак, спеціалізовані графічні планшети часто мають високу вартість, що може бути перешкодою для багатьох користувачів, особливо початківців або студентів.

Використання Android-планшета як графічного планшета для комп'ютера з ОС Windows представляє інноваційне так економічно ефективне рішення цієї проблеми.

Це дозволяє використовувати вже наявний пристрій для розширення можливостей роботи з графіко на комп'ютері, що особливо актуально в умовах зростаючої популярності мобільних пристроїв та їх інтеграції з настільними рішеннями.

Дослідження в цій області є актуальним, оскільки воно спрямоване на подолання технічних викликів, пов'язаних з інтеграцією різних приладів з різними операційними системами та оптимізацією їх продуктивності. Крім того, розробка такої системи відкриває нові можливості для користувачів, які прагнуть розширити функціональність своїх Android пристроїв та підвищити ефективність роботи з графічними програмами на Windows.

Практична цінність отриманих результатів полягає в створенні доступного рішення для перетворення Android-планшета на повноцінний графічний інструмент, що може значно розширити можливості користувачів у сфері цифрового мистецтва, дизайну та інших галузях, де потрібна точно робота з графікою. Це дозволить багатьом користувачам отримати доступ до функціональності професійних графічних планшетів без значних фінансових витрат, що сприятиме розви-

тку креативних індустрій та підвищенню доступності цифрових інструментів для великого кола користувачів.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows.

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

– Дослідження існуючих систем для використання мобільних як графічних планшетів, а також існуючих методів передачі даних між різними операційними системами;

– Розробка методів та алгоритмів взаємодії Android-планшетом та комп'ютером з ОС Windows;

– Програмна реалізація системи для використання Android-планшета як графічного планшета в Windows.

Об'єктом дослідження є процес взаємодії між Android-планшетом та комп'ютером з ОС Windows для забезпечення функціональності графічного планшета.

Предметом дослідження є методи передачі даних між операційними системами, методи обробки введення стилуса та методи емуляції графічного планшета в Windows, які базуються на методах розробки кросплатформного програмного забезпечення, методах мережевої взаємодії, методах обробки введення користувача та методах оптимізації передачі даних.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1) Запропоновано метод використання Android-планшета як графічного планшета з оптимізованою передачею даних та мінімальною затримкою.

2) Розроблено вітчизняний програмний продукт, який має більш широкі можливості налаштування та оптимізації, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в створенні доступного рішення для перетворення Android-планшета на повноцінний графічний інструмент, що може значно розширити можливості користувачів у сфері цифрового мистецтва та дизайну без значних фінансових витрат.

Достовірність наукових результатів підтверджена теоретичними викладками, результатами тестування розробленого програмного забезпечення в різних умовах використання, а також відповідністю отриманих результатів окремим результатам, наведеним в у науковій літературі.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначення системи полягає в у створенні програмного мосту між Android-планшетом та комп'ютером з Windows, що забезпечує точну передачу всіх параметрів введення стилуса. Система здійснює захоплення даних про положення стилуса, силу натиску та кут нахилу на Android пристрої, після чого передає ці дані на комп'ютер через вибраний канал зв'язку. На стороні Windows ці дані інтерпретуються таким чином, щоб операційна система сприймала Android-планшет як повноцінний графічний планшет.\

Важливою особливістю системи є її здатність природньої передачі досвіду малювання завдяки мінімізованій затримці між введенням на планшеті та відображенням результату на комп'ютері. Система також надає користувачу можливість тонкого налаштування параметрів роботи стилуса, включаючи чутливість до натиску, криві відгуку, та налаштування областей введення, що дозволяє адаптувати її під індивідуальні потреби та стиль роботи користувача.

1.2 Область застосування

Розроблена система знаходить широке застосування к різноманітних сферах професійної діяльності та творчості. В галузі цифрового мистецтва та дизайну система надає художникам та ілюстраторам можливість створювати деталізовані цифрові малюнки малюнки з природним відчуттям традиційних художніх інструментів.

У сфері освіти система відриває нові можливості для дистанційного навчання, дозволяючи викладачам створювати інтерактивні навчальні матеріали та проводити онлайн заняття з можливістю живого малювання та письма. Це особ-

ливо актуально для викладання математики, фізики та інших предметів де важлива можливість швидко створювати візуальні пояснення. схеми та діаграми.

Архітектори та інженери також можуть використовувати систему для створення ескізів, технічних креслень та швидкого прототипування ідей. Точність введення стилуса дозволяє створювати деталізовані креслення, а можливість швидкого переключення між інструментами прискорить робочий процес.

В індустрії анімації, 3D моделювання та motion-дизайну система надає зручний інструмент для розширення робочої поверхні для створення розкадровок та анімаційних ескізів. Можливість природнього малювання допомагає швидко візуалізувати рук та об’єм і відкриває більшу можливість, щоб експериментувати з різними варіантами створення свої проектів.

Особливу цінність система представляє для фрілансерів та віддалених працівників креативної індустрії, надаючи їм можливість створювати професійне робоче місце без значних фінансових інвестицій у спеціалізоване обладнання. Це робить систему привабливою для широкого кола користувачів, від початківців-аматорів до професіоналів, які шукають гнучкі так економічно ефективне рішення для роботи з цифровою графікою.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми магістерської роботи

Графічні планшети є невід'ємною частиною сучасної індустрії дизайну, ілюстрації та цифрового мистецтва. Вони дозволяють художникам та дизайнгерам створювати високоякісні цифрові зображення, використовуючі звичні та зручні інструменти – стилус та планшет.

Графічні планшети складаються з двох частин: пера та планшетної поверхні. Планшетна поверхня містить сітку з дротів або інших сенсорів, які здатні визначити положення стилуса, силу натиску та нахил. Коли користувач торкається пером планшета або наближає стилус до його поверхні, сенсори фіксують ці дані та передають їх на комп'ютер. Комп'ютер, у свою чергу, інтерпретує ці дані як рухи курсора або пензля в графічному додатку, що дозволяє користувачеві малювати, пискати або редагувати зображення за допомогою планшета.

Операційна система Windows має підтримку графічних планшетів. Графічні планшети підключаються до комп'ютера з Windows через USB або Bluetooth і функціонують як пристрої введення, подібно до миші або трекпада. Windows забезпечує підтримку графічних планшетів через драйвери Windows Ink. Більшість графічних додатків для Windows, таких як Adobe Photoshop, Corel painter, Autodesk SketchBook, мають спеціальні функції та налаштування, оптимізовані для роботи з графічними планшетами. Це дозволяє користувачам повною мірою використовувати можливості графічних планшетів та досягти високої точності та виразності при створенні цифрових зображень.

Сучасні Android-планшети оснащені високоякісними дисплеями , чутливими до тиску стилусами та потужними процесорами, що дозволяє використовувати їх для малювання та дизайну. Проєкти, подібні до даної роботи, дозволяють

використовувати Android-планшети як графічні планшети для Windows, розширюючи можливості користувачів пропонуючи біль гнучкі та доступні рішення. Такий підхід може зробити професійні інструменти для створення цифрової графіки більш доступними для ширшого кола користувачів.

Розглянемо приклади існуючих рішень:

1) Duet Display – це система, що дозволяє використовувати iPad або iPhone як додатковий екран для комп'ютерів Mac або PC. Вона підтримує роботу з стилусом Apple Pencil, забезпечуючи низьку затримку та високу якість зображення. Серед переваг Duet Display - просте налаштування та повна інтеграція з екосистемою Apple. Недоліками є обмежена сумісність (тільки iOS/Mac), необхідність встановлення додаткового програмного забезпечення та платна модель розповсюдження.

Рисунок 2.1 – Duet Display

2) Astropad - це рішення, що перетворює iPad на високоточний графічний планшет для комп'ютерів Mac. Воно підтримує підключення через USB та Wi-Fi, має розширені налаштування пера та чутливості до тиску. Перевагами Astropad є висока точність введення та зручність налаштування, а також підтримка більшості графічних додатків на Mac. Недоліками є обмеження сумісності (тільки Mac), необхідність встановлення додаткового ПЗ та відносно висока ціна.

Рисунок 2.2 – Astropad

3) VirtualTablet - це застосунок для Android, який перетворює пристрій на бездротовий графічний планшет для Windows. Він використовує Wi-Fi для зв'язку з ПК і дозволяє налаштовувати кнопки та чутливість пера. Серед переваг VirtualTablet - сумісність з широким спектром Android-пристроїв та безкоштовне використання. Недоліками є залежність від якості Wi-Fi з'єднання та обмежені можливості налаштування у порівнянні з деякими платними рішеннями.

Рисунок 2.3 – VirtualTablet

4) Splashtop Wired XDisplay - це застосунок дозволяє використовувати планшет як додатковий екран для ПК через USB. Пропонує низьку затримку передачі. Серед переваг Splashtop Wired XDisplay висока якість зображення, стабільна робота через дротове з'єднання. Недоліками є вимагає дротового підключення, платна версія для розширених функцій.

Рисунок 2.4 - Splashtop Wired XDisplay

5) iDisplay – це застосунок, що підтримує iOS, Android та Windows. Дає можливість створити додатковий екран для ПК через Wi-Fi або USB. З переваг даного програмного рішення можна виділити універсальність платформи та просте налаштування. Недоліками є можлива висока затримка при Wi-Fi з'єднанні.

Рисунок 2.5 iDisplay

6) EasyCanvas - це рішення, що перетворює iPad на високоточний графічний планшет для комп'ютерів Mac. Має базові функції, такі як чутливість до натискання пера, чутливість до нахилу та відхилення долоні, а також зручні функції, включаючи настроюванні комбінації клавіш і жести пальцями, надаються для того, щоб користуватися тими ж функціями, що і на звичайному планшеті для малювання. З переваг можна виділити підтримку стилусів з функціями нахилу та чутливості до натиску. Недоліком є недоступність безкоштовної версії.

Рисунок 2.6 - EasyCanvas

Дослідивши існуючі системи бачимо що в більшості вони виконують лише частину функціональності від можливих. Основними недоліками розглянутих рішень є їх зосередження на трансляції зображення та центричність на системах Apple як Mac та IOS, що унеможливлює їх використання на інших платформах.

У моєму проекті основною ідеєю є розширення функцій обох систем Windows та Android. Це дозволяє охопити великий ринок приладів, та ще більшу кількість користувачів у яких вже є апаратне забезпечення, але відсутня програмна можливість використовувати Android-планшет як графічний планшет для ОС Windows.

Розглянемо технології та протоколи передачі даних:

1) WebSocket - це протокол двонаправленого обміну даними поверх TCP-з'єднання. Він забезпечує низьку затримку та підходить для передачі координат стилуса в реальному часі. Перевагами WebSocket є повнодуплексний зв'язок, низький оверхед та сумісність з більшістю веб-браузерів і мов програмування. Недоліками є необхідність постійного TCP-з'єднання та можливість блокування деякими файрволами.

2) UDP - це протокол без встановлення з'єднання, який дозволяє швидко передавати невеликі пакети даних. Він може використовуватися для передачі координат стилуса, але без гарантії доставки пакетів. Перевагами UDP є низька затримка, мінімальний оверхед та швидкість передачі даних. Недоліками є відсутність вбудованого механізму підтвердження доставки та обмежений розмір пакетів.

3) TCP - це протокол зі встановленням з'єднання, який гарантує доставку пакетів у правильному порядку. Він може використовуватися для передачі зображень між пристроями. Перевагами TCP є надійність, автоматичне повторне надсилання втрачених пакетів та контроль потоку даних. Недоліками є вища затримка у порівнянні з UDP та більший оверхед через механізми контролю.

4) Bluetooth - це бездротовий протокол передачі даних на коротких відстанях. Він дозволяє встановити пряме з'єднання між Android-пристроєм та комп'ю-

тером з Windows для передачі даних стилуса. Перевагами Bluetooth є низьке енергоспоживання, вбудовані механізми шифрування та автентифікації, а також відсутність потреби в додатковому обладнанні. Недоліками є обмежена пропускна здатність, відносно короткий радіус дії та можливі перешкоди від інших пристроїв.

5) Wi-Fi Direct - це стандарт Wi-Fi для бездротових з'єднань, який дозволяє двом пристроям встановлювати пряме з'єднання Wi-Fi без проміжної бездротової точки доступу, маршрутизатора або підключення до Інтернету. Перевагами Wi-Fi Direct відсутність необхідності використання точки доступу або підключення до мережі інтернет. Це робить його схожим з технологією Bluetooth, але пропонує більший радіус дії. Недоліком є лише необхідність, що прилади підтримували технологію Wi-Fi Direct.

6) Usbip-win - це порт реалізації USB/IP (USB over IP) для операційної системи Windows. Цей проєкт дозволяє передавати дані від USB-пристроїв через мережу, дозволяючи використовувати віддалені USB-пристрої так, ніби вони підключені безпосередньо до локальної машини. З переваг такого рішення є використання сценаріїв SMTP і SNTP для моніторингу стану пристроїв та USB-пристрої доступні необмеженій кількості користувачів (з можливістю створення групових політик і рівнів доступу) без необхідності фізичного перемикання з будь-якої точки світу. Недоліком є, що не всі USB-пристрої можуть нормально працювати по мережі через збільшений час відгуку та висока вартість апаратних рішень.

Розглянемо програмну реалізацію:

1) На стороні Windows може бути створено віртуальний пристрій (планшет), який емулює реальний графічний планшет. Дані зі стилуса Android-пристрою інтерпретуються цим драйвером. Перевагами такого підходу є сумісність з більшістю графічних додатків на Windows та прозорість для користувача. Недоліками є потенційно нижча продуктивність у порівнянні з прямою інтеграцією та залежність від якості драйвера.

2) Інтеграція через API графічних додатків. Деякі графічні програми мають власні API (Application Programming Interface), через які можна напряму передава-

ти дані про натискання, координати пера та інші параметри. Це дозволяє реалізувати більш тісну інтеграцію між Android-застосунком та графічним додатком на Windows. Перевагами такого підходу є потенційно вища продуктивність та точність, а також розширені можливості налаштування. Недоліками є обмежена сумісність (тільки з додатками, що мають відповідний API) та потреба в розробці окремих модулів для кожного додатка.

3) Android надає власний API (наприклад, MotionEvent) для отримання даних про натискання, координати та нахил стилуса. Ці дані можуть бути захоплені Android-застосунком та передані на обробку у Windows. Перевагами такого підходу є доступ до "сирих" даних введення та можливість реалізації власних алгоритмів обробки та фільтрації. Недоліками є потреба у розробці власного протоколу передачі даних та потенційно вища складність реалізації.

При виборі конкретних рішень слід врахувати велику низку вимог та обмежень проєкту, такі як необхідна продуктивність, сумісність з існуючими системами, зручність використання та доступні ресурси для розробки

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для створення такого амбіційного проєкту будуть необхідні комплексні середовища розробки:

Android Studio

Android Studio – це офіційне інтегроване середовище розробки (IDE) для платформи Android. Воно базується на IntelliJ IDEA та розроблене спеціально для створення Android-додатків. Android Studio пропонує потужні інструменти для редагування, налагодження, тестування та профілювання коду.

IDE має зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє ефективно працювати над проектами. Вона підтримує розробку додатків для різних

пристроїв Android, включаючи смартфони, планшети, годинники, TV-приставки та інші.

Android Studio включає візуальний редактор макетів, який дозволяє швидко створювати користувацькі інтерфейси методом перетягування. Також доступний потужний редактор коду з підсвічуванням синтаксису, автодоповненням, рефакторингом та іншими корисними функціями.

Вбудований емулятор Android дозволяє тестувати та налагоджувати додатки на різних віртуальних пристроях без необхідності використовувати фізичні девайси. Android Studio також має інтеграцію з системами контролю версій, такими як Git, що полегшує командну роботу над проектами.

Мови що підтримуються:

1) Kotlin;

2) Java;

3) C++.

Android Studio регулярно оновлюється, отримуючи нові функції та покращення. Вона має активну спільноту розробників, які створюють плагіни та розширення для розширення можливостей середовища розробки.

Загалом, Android Studio є потужним та універсальним інструментом для розробки високоякісних Android-додатків, що поєднує в собі зручність використання, багатий функціонал та можливості кастомізації.

Visual Studio.

Visual Studio для розробки Windows частини.

Visual Studio - це інтегроване середовище розробки (IDE) від компанії Microsoft. Воно призначене для створення різноманітних типів додатків, включаючи десктопні програми, веб-сайти, веб-сервіси, мобільні додатки та ігри.

Мови що підтримуються:

1) C#;

2) Visual Basic;

3) F#;

4) C++;

5) Python;

6) JavaScript;

7) TypeScript.

Visual Studio надає потужні інструменти для редагування, налагодження та профілювання коду, що спрощує процес розробки.

Середовище має зручний та настроюваний інтерфейс, який дозволяє розробникам персоналізувати робочий простір відповідно до своїх потреб. Вбудований редактор коду пропонує підсвічування синтаксису, автодоповнення, рефакторинг та інші корисні функції.

Visual Studio інтегрується з різними фреймворками та технологіями, такими як .NET, ASP.NET, Unity, Xamarin та іншими. Це дозволяє розробникам створювати додатки для різних платформ, включаючи Windows, macOS, iOS, Android та веб.

IDE також включає інструменти для роботи з базами даних, тестування, налагодження та розгортання додатків. Вона підтримує системи контролю версій, такі як Git та Team Foundation Server, що полегшує командну роботу над проектами.

Мови програмування:

Kotlin

Вибір Kotlin як основної мови програмування для Android-компонента базується на кількох ключових перевагах цієї мови. Kotlin пропонує сучасний синтаксис та функціональні можливості, які роблять код більш надійним та легшим для підтримки.

Kotlin має чіткий та лаконічний синтаксис, що робить код більш читабельним та зрозумілим. Мова пропонує багато функцій, які дозволяють писати менше шаблонного коду та зосередитися на бізнес-логіці.

Однією з ключових переваг Kotlin є її безпечність. Мова має вбудовану підтримку null-безпеки, що допомагає уникнути поширених помилок, пов'язаних з

нульовими посиланнями (null pointer exceptions). Kotlin також пропонує розширені можливості обробки виключень та забезпечує безпечну роботу з типами даних.

Kotlin є повністю інтероперабельним з Java, що означає, що код на Kotlin може легко взаємодіяти з існуючим Java-кодом. Це дозволяє поступово впроваджувати Kotlin у існуючі Java-проекти без необхідності повного переписування.

Мова підтримує функціональне програмування та надає потужні інструменти для роботи з колекціями, такі як функції вищого порядку, лямбда-вирази та потоки даних. Kotlin також має зручний синтаксис для створення класів даних (data classes) та роботи зі структурами даних.

Особливо важливою для нашого проекту є вбудована підтримка корутинів у Kotlin. Корутини дозволяють елегантно обробляти асинхронні операції, такі як отримання даних від сенсорів стилуса та мережева передача, без створення складних колбек-структур. Це робить код більш читабельним та менш схильним до помилок.

Система типів Kotlin з підтримкою null-безпеки допомагає запобігти поширеним помилкам часу виконання, пов'язаним з null-посиланнями. Розширення функцій дозволяють додавати нову функціональність до існуючих класів Android SDK без успадкування, що робить код більш модульним.

C#

C# був обраний для розробки Windows-компонента завдяки його потужним можливостям для системного програмування та відмінній інтеграції з Windows API. Мова надає зручний доступ до низькорівневих функцій операційної системи через P/Invoke, що необхідно для емуляції пристроїв введення.

C# має чіткий та виразний синтаксис, який подібний до інших мов програмування, таких як C++ та Java. Це полегшує вивчення мови для розробників, які вже мають досвід роботи з цими мовами.

Однією з ключових особливостей C# є його потужна система типів. Мова підтримує як статичну, так і динамічну типізацію, що дозволяє писати безпечний

та надійний код. C# також має вбудовану підтримку обробки винятків, що спрощує обробку помилок та покращує стабільність програм.

C# тісно інтегрується з платформою .NET, яка надає багатий набір бібліотек та фреймворків для розробки різноманітних типів додатків. Це включає розробку десктопних програм з використанням Windows Forms або WPF, веб-додатків за допомогою ASP.NET, мобільних додатків з Xamarin та ігор за допомогою Unity.

Мова підтримує сучасні концепції програмування, такі як лямбда-вирази, LINQ (мова інтегрованих запитів), асинхронне програмування за допомогою async/await та інші. Ці функції дозволяють писати більш стислий та виразний код.

C# постійно розвивається, і кожна нова версія мови додає нові можливості та вдосконалення. Наприклад, останні версії C# представили такі функції, як null-умовні оператори, шаблони зіставлення та локальні функції.

Для забезпечення комунікацій між компонентами системи було обрано протокол WebSocket. Вибір обумовлений необхідністю забезпечення двонаправленого обміну даними в реальному часі та при мінімальній затримці. Також вбудована підтримка WebSocket у більшості сучасних платформ розробки спрощує реалізацію та підтримку коду.

Та для обробки графіки на стороні Android було обрано OpenGL ES, оскільки ця технологія забезпечує апаратне прискорення графічних операцій, що критично важливо для плавного відображення введення стилуса. OpenGL ES також надає низькорівневий контроль над графічними операціями, що дозволить оптимізувати продуктивність і зменшити затримку системи.

2.3 Розгорнута постановка завдання

Основним завданням розробки є створення програмного рішення, який дозволить використовувати Android-планшет як повноцінну заміну професійному графічному планшету при роботі з комп’ютером під управлінням Windows. Система повинна забезпечувати максимально простий досвід малювання та відпові-

дати наступним технічним вимогам.

У частині обробки введення система повинна точно фіксувати всі параметри взаємодії стилуса з екраном планшета. Включно з координатами дотику, силою натиску, кутом нахилу та обертанням стилуса. Особлива увага необхідна з частотою опитаування сенсорів для забезпечення плавного відстеження руху стилуса. Система повинна коректно обробляти різні жести та багатодотикове введення.

Необхідно реалізувати надійний та швидкий механізм комунікації між Android-планшетом та Windows комп'ютером. Затримка між введенням та відображенням не повинна перевищувати 20 мс для забезпечення плавної роботи системи. Система повинна підтримувати різні способи підключення: Wi-Fi, USB. Bluetooh та автоматично обирати оптимальний канал зв'язку.

На стороні Windows система повинна забезпечити емуляцію графічного планшета, що дозволить використовувати Android-планшет з будь-якими програмами, які підтримують роботу з графічними планшетами. Це включає реалізацію відповідних драйверів та інтеграцію з Windows Ink API.

Важливою частиною також є розробка зручного та зрозумілого користувацького інтерфейсу, який дозволить налаштувати параметри роботи системи без необхідності глибокого розуміння технічних деталей. Користувач повинен вільно калібрувати планшет, чутливість стилусу, визначати активну область введення та зберігати різні профілі налаштувань.

Система повинна бути стабільною та відмово стійкою протягом довго часу, ефективно використовувати ресурси пристроїв та мати механізми відновлення після збою. Необхідно передбачити можливість оновлення програмного забезпечення та розширень функцій.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розроблена система представляє собою перетворення Android-планшета в повноцінний графічний планшет для роботи з комп'ютером під управлінням Windows. Система складається з двох основних взаємодіючих компонентів: Android-додатку та Windows-програми.

Android-додаток відповідає за захоплення та обробку введення стилуса. Використовуючи спеціалізовані API операційної системи, програма отримує точні дані про положення стилуса, силу натиску та кут нахилу. Ці дані обробляються в реальному часі із застосуванням алгоритмів згладжування. Важливою особливістю є висока частота опитування сенсорів, що забезпечує плавне відстеження руху стилуса без помітних затримок.

Однією з ключових особливостей системи є функціонал передачі зображення з Windows на екран Android-планшета в реальному часі. Ця функція дозволяє користувачу бачити результат своєї роботи безпосередньо на екрані планшета, що значно покращує точність та зручність малювання. Система використовує ефективні алгоритми стиснення зображення та оптимізовану передачу даних, щоб забезпечити мінімальну затримку між діями користувача та відображенням результату. При цьому якість переданого зображення адаптивно регулюється в залежності від доступної пропускної здатності мережі, забезпечуючи оптимальний баланс між якістю та швидкодією.

Передача даних між пристроями здійснюється через оптимізований протокол на базі WebSocket, який забезпечує стабільний двонаправлений зв'язок з мінімальною затримкою. Система використовує адаптивне стиснення даних для ефективного використання пропускної здатності мережі без втрати якості передачі. Реалізовано механізм буферизації, який допомагає зберігати плавність роботи

навіть при тимчасових проблемах зі з'єднанням.

3.2 Розробка структурної схеми

Система складається з двох основних компонентів: Android-додатку та Windows-програми, які взаємодіють між собою для забезпечення функціональності графічного планшета.

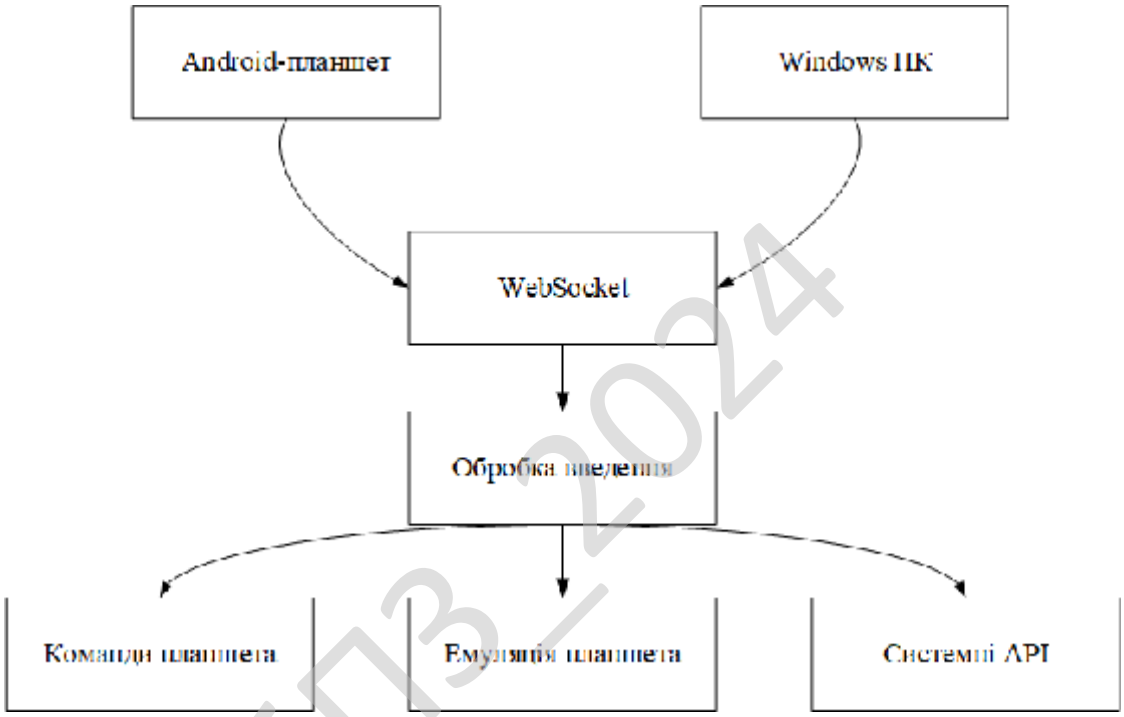

Рисунок 3.1 - Структурна схема системи

Вона побудована покладаючись на взаємозв'язок усіх процесів додатку. За допомогою розробки функціональних частин можна окреслити зв’язки між ними.

Android-планшет виступає як пристрій введення, що захоплює всі дані зі стилуса, включаючи його позицію, силу натиску та кут нахилу. Він обробляє ці дані та передає їх через мережу на комп’ютер. Додатково, урижимі з передачею зображення, планшет також відображає екран комп’ютера, що дозволяє бачити результати малювання безпосередньо на його екрані.

ОС Windows приймає дані від планшета та емулює роботу стандартного графічного планшета для системи. У режимі з передачею зображення комп'ютер

також захоплює вміст екрану або його частини та передає це зображення назад на планшет. Комп'ютер забезпечує взаємодію з різними графічними програмами через системні API.

Емуляція планшета створює в системі Windows віртуальний графічний планшет, який поводиться так само, як і фізичний пристрій. Цей компонент забезпечує повну сумісність з різними графічними програмами, емулюючи всі необхідні функції стандартного графічного планшета.

WebSocket забезпечує надійний двосторонній зв'язок між планшетом та комп'ютером. Цей протокол дозволяє передавати дані в реальному часі з мінімальною затримкою, що критично важливо для природного відчуття малювання. Через WebSocket передаються як дані стилуса, так і зображення екрану.

Обробка введення є центральним компонентом системи, який приймає дані від WebSocket з'єднання та розподіляє їх між різними модулями системи. Цей компонент відповідає за аналіз отриманих даних, їх валідацію та синхронізацію між різними частинами системи.

Системні API забезпечують взаємодію з операційною системою Windows, дозволяючи емулювати введення, керувати графічними пристроями та отримувати доступ до системних функцій. Через ці API здійснюється інтеграція віртуального планшета з системою та графічними програмами.

Всі ці компоненти працюють разом як єдина система, забезпечуючи точне введення стилусом, мінімальну затримку при роботі, стабільне з'єднання між пристроями та повну сумісність з різними програмами для роботи з графікою. Система оптимізована для максимальної ефективності та зручності використання, дозволяючи перетворити звичайний Android-планшет на повноцінний графічний планшет для роботи з Windows.

3.3 Розробка функціональної схеми

Ця функціональна схема детально ілюструє послідовність обробки даних та потік інформації в системі використання Android-планшета як графічного планшета для Windows.

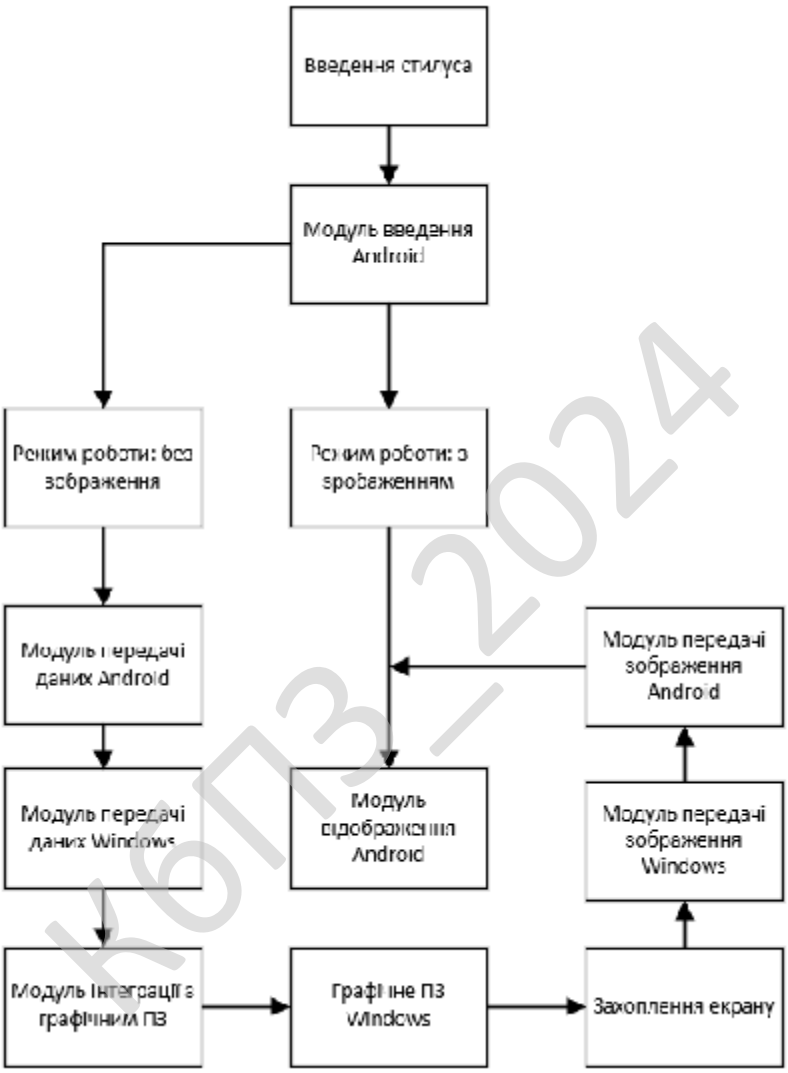

Рисунок 3.2 - Функціональна схема системи

Процес починається з введення стилуса користувачем на Android-планшеті. Ці дані спочатку обробляються модулем введення Android, який відповідає за розпізнавання та первинну обробку дотиків стилуса, включаючи позицію, тиск та кут нахилу.

В режимі роботи без зображення дані стилуса відразу передаються через модуль передачі даних Android до відповідного модуля Windows. Цей режим забезпечує мінімальну затримку та оптимальне використання ресурсів системи, оскільки користувач дивиться безпосередньо на екран комп'ютера під час роботи.

В режимі роботи з зображенням система додатково забезпечує відображення екрану комп'ютера на планшеті. При цьому дані спочатку проходять через модуль відображення Android, який синхронізує введення стилуса з отриманим зображенням екрану комп'ютера.

Модуль інтеграції з графічним ПЗ забезпечує взаємодію з програмним забезпеченням Windows, емулюючи роботу стандартного графічного планшета. Після обробки даних графічним ПЗ Windows відбувається захоплення екрану, яке через модуль передачі зображення Windows передається назад на Android-планшет, замикаючи цикл візуального зворотного зв’язку.

Модуль передачі зображення Android отримує захоплений екран та передає його модулю відображення, який оновлює картинку на екрані планшета. Така архітектура забезпечує плавну роботу системи в обох режимах, дозволяючи користувачу вибрати найбільш зручний варіант використання планшета як графічного пристрою введення.

Вся система працює як єдиний механізм, де кожен модуль виконує свою специфічну функцію, забезпечуючи ефективну передачу даних між пристроями та коректну емуляцію графічного планшета в системі Windows.

Така архітектура забезпечує два важливі сценарії використання:

- Легкий режим без передачі зображення, який споживає менше ресурсів та підходить для досвідчених користувачів.
- Повний режим з передачею зображення, який забезпечує візуальний зворотний зв'язок безпосередньо на екрані планшета.

Схема чітко показує, як система забезпечує безперервний потік даних між пристроями, підтримуючи природний процес малювання з мінімальною затрим-

кою. Кожен модуль має чітко визначену роль в загальному процесі, що спрощує розробку, тестування та подальшу підтримку системи.

Особливо важливо відзначити, що схема передбачає оптимізацію навантаження завдяки наявності двох режимів роботи, дозволяючи користувачам вибирати найбільш підходящий варіант відповідно до їхніх потреб та можливостей обладнання.

3.4 Розробка діаграми процесів

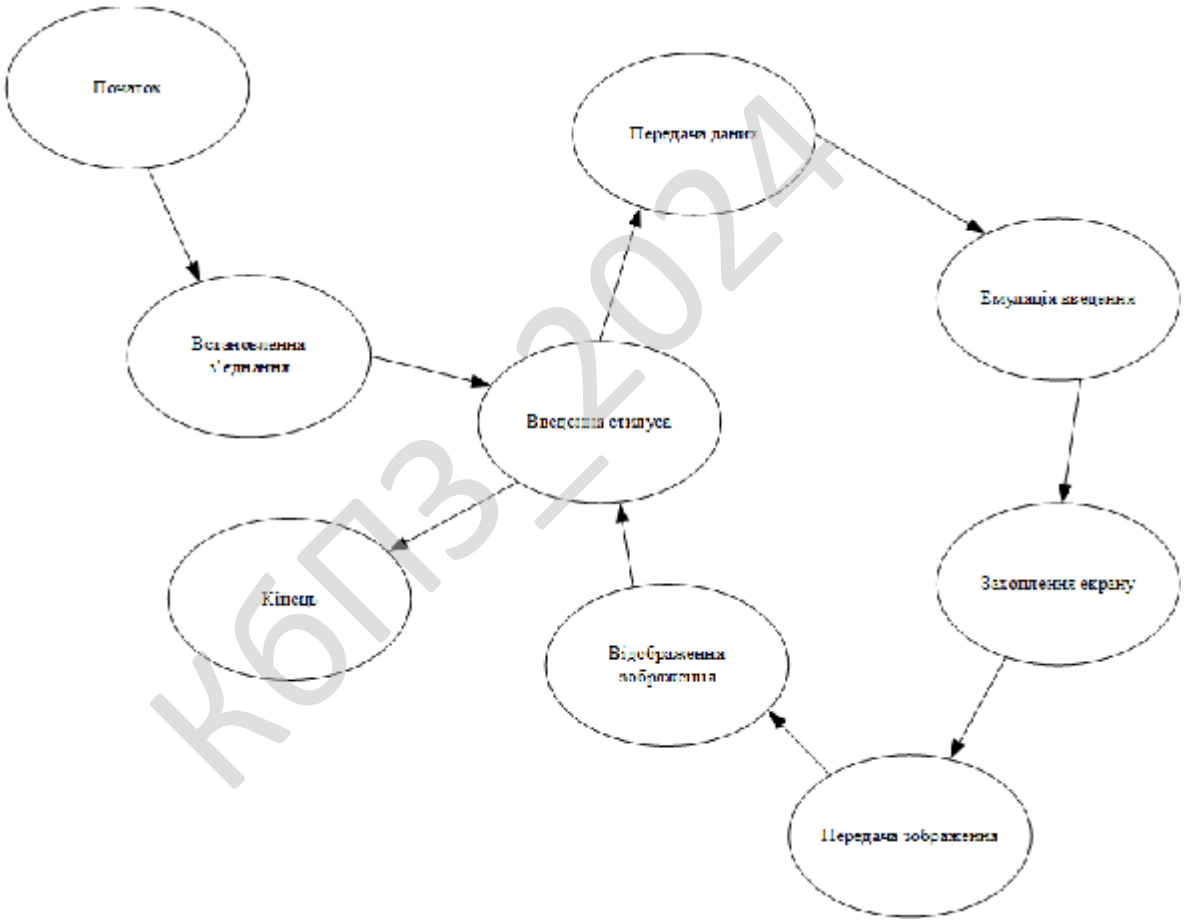

Рисунок 3.3 - Діаграма процесів

Діаграма процесів ілюструє повний цикл роботи системи використання Android-планшета як графічного планшета для Windows. Давайте розглянемо як відбувається взаємодія між різними процесами системи.

Процес роботи системи починається з встановлення з'єднання між

Android-планшетом та комп'ютером з Windows. Після успішного підключення система переходить до основного етапу - введення стилуса.

Процес введення стилуса є центральним елементом системи, який взаємодіє з кількома іншими процесами. При отриманні даних від стилуса вони передаються на обробку. Передані дані використовуються для емуляції введення на комп'ютері, що дозволяє системі Windows сприймати Android-планшет як стандартний графічний планшет.

Після емуляції введення система переходить до захоплення екрану, що необхідно для забезпечення візуального зворотного зв'язку. Захоплений екран передається назад на планшет що забезпечує доставку візуальних даних на Android-планшет.

Отримане зображення відображається на екрані планшета, який синхронізується з введенням стилуса для забезпечення природної взаємодії. Цей цикл постійно повторюється під час роботи системи, забезпечуючи плавне та відповідне введення.

Процес може бути завершений у будь-який момент, коли користувач закінчує роботу з системою. Всі процеси взаємопов'язані та працюють синхронно, забезпечуючи ефективну роботу системи як єдиного цілого.

Діаграма процесів демонструє циклічну природу роботи системи, де кожен процес є важливою ланкою в забезпеченні функціональності графічного планшета, а взаємозв'язки між процесами забезпечують безперервну та плавну роботу всієї системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Виходячи з розроблених функціональних схем та структурних рішень, розглянемо алгоритми, за якими працює система використання Android-планшета як графічного планшета для комп'ютера з ОС Windows.

На початку роботи система ініціює запуск Android-додатку, який відповідає за обробку введення стилуса. Цей компонент містить важливу інформацію про параметри стилуса, включаючи його координати, силу натиску та кут нахилу. Після запуску додатку система проводить перевірку наявності з'єднання з Windows-комп'ютером, щоб визначити можливість прямої передачі даних.

Можливість миттєвого підключення реалізується у випадку, якщо пристрої вже були з'єднані раніше і мають збережені налаштування. Якщо з'єднання активне, система автоматично починає передачу даних, що значно зменшує час налаштування та покращує користувацький досвід.

У випадку відсутності активного з'єднання система виконує пошук доступних пристроїв та намагається встановити нове підключення. Цей процес включає перевірку мережевих параметрів та встановлення WebSocket з'єднання. Якщо система успішно знаходить Windows-комп'ютер, вона переходить до етапу калібрування стилуса.

Модуль калібрування є одним із найважливіших компонентів системи, оскільки тут відбувається налаштування всіх параметрів введення, які надходять від стилуса. Цей модуль також відповідає за коректну емуляцію графічного планшета в системі Windows, забезпечуючи точне відтворення рухів стилуса. Під

час калібрування система аналізує відповідність координат на планшеті та екрані комп'ютера, а також налаштовує чутливість до натиску.

Якщо всі перевірки та налаштування проходять успішно, система активує режим роботи графічного планшета, реєструючи всі параметри поточної конфігурації. Ця інформація зберігається для подальшого використання та швидкого відновлення налаштувань. Завдяки такій архітектурі система забезпечує високу точність введення та надійність роботи, а також оптимізує процес взаємодії між планшетом та комп'ютером.

В цілому, описаний процес підкреслює важливість злагодженої роботи всіх компонентів системи, що забезпечує ефективне перетворення Android-планшета у повноцінний графічний планшет для роботи з Windows-програмами, максимальну точність введення та комфортне використання системи.

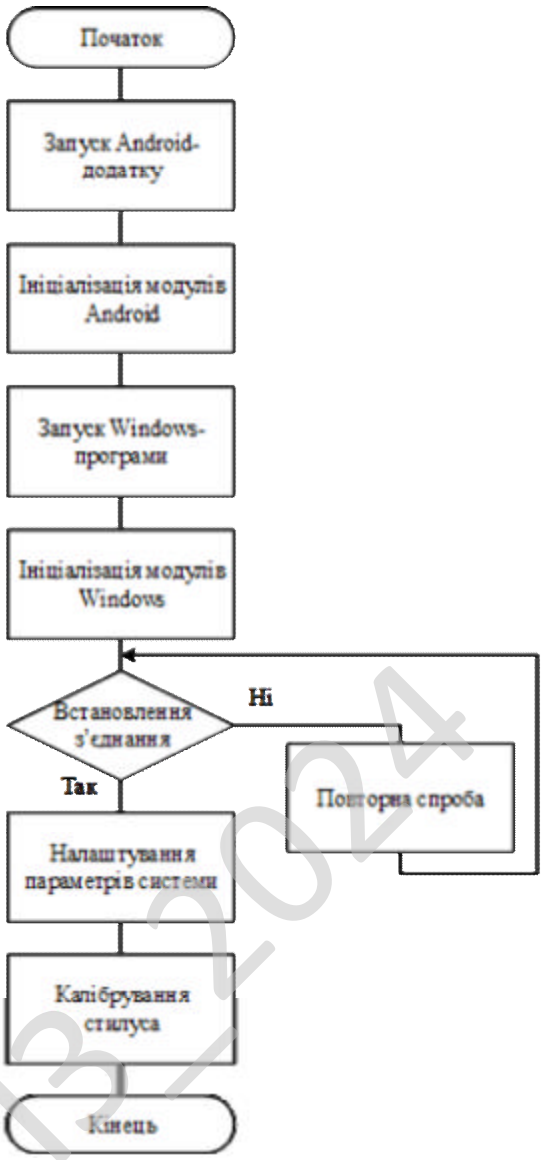

Рисунок 4.1 - Блок-схема алгоритму ініціалізації системи

Загальний принцип роботи алгоритму полягає у обробці даних, отриманих від стилуса Android-планшета, їх передачі та емуляції на Windows-комп'ютері. За допомогою цих даних виконується точне відтворення введення та синхронізація відображення. Кожен блок працює автоматично та має зв'язки з іншими модулями системи.

Розглянемо кожен блок окремо:

1) **Введеня стилуса.** На цьому етапі здійснюється обробка даних стилуса:

- Отримання координат, тиску та нахилу стилуса.
- Перевірка коректності вхідних даних.

- Первинна обробка та фільтрація даних.
- У разі коректних даних система переходить до наступного блоку.

2) **Обробка даних.** Цей блок відповідає за підготовку даних до передачі:

- Нормалізація координат та параметрів стилуса.
- Застосування калібрувальних коефіцієнтів.
- Буферизація даних для плавної передачі.
- При помилках обробки ініціюється повторна спроба.

3) **Передача даних.** При виникненні проблем з передачею:

- Система повторює спробу передачі.
- Перевіряє стан з'єднання.
- У разі стабільних проблем переходить до відновлення з'єднання.

4) **Емуляція введення.** На цьому етапі система емулює роботу графічного планшета:

- Перетворення отриманих даних у системні події.
- Емуляція натиску та руху стилуса.
- Синхронізація з графічними програмами.

5) **Захоплення екрану.** Цей блок відповідає за візуальний зворотній зв'язок:

- Захоплення області екрану Windows.
- Оптимізація зображення для передачі.
- Відправка даних на Android-планшет.

6) **Відображення.** Відображення забезпечує візуальний зворотній зв'язок:

- Оновлення екрану планшета.
- Синхронізація з введенням стилуса.
- Забезпечення мінімальної затримки відображення.

7) **Моніторинг з'єднання**. Після встановлення з'єднання виконується постійний контроль:

- Перевірка стабільності з'єднання.
- Вимірювання затримки передачі.

- Оптимізація параметрів передачі даних.

8) **Управління режимами.** Відбувається контроль режимів роботи:

- Перемикання між режимами роботи.
- Налаштування параметрів відображення.
- Оптимізація використання ресурсів.

9) **Калібрування.** Система забезпечує точність введення:

- Збір калібрувальних даних.
- Розрахунок коефіцієнтів корекції.
- Застосування калібрування до вхідних даних.

10) **Збереження налаштувань.** Фінальний етап, на якому зберігаються всі параметри:

- Збереження калібрувальних даних.
- Запис параметрів з'єднання.
- Збереження користувацьких налаштувань.

Цей процес є неперервним та автоматизованим, що забезпечує стабільну роботу та високу точність емуляції графічного планшета. Система постійно оптимізує свою роботу для забезпечення мінімальної затримки та максимальної точності введення.

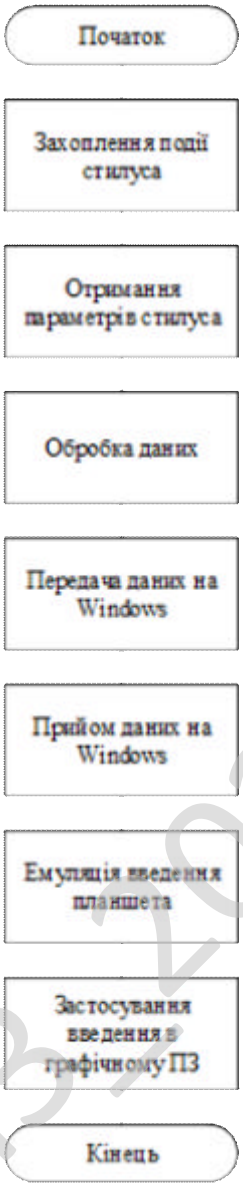

Рисунок 4.2 - Блок-схема алгоритму обробки введення стилуса

Блок-схема ілюструє як збираються та передаються параметри основного інструменту для роботи системи, а саме стилуса. Алгоритм починається з етапу захоплення подій стилуса на Android-планшеті. На цьому етапі система активує всі необхідні обробники подій та готується до прийому даних від стилуса. Відбувається ініціалізація системних компонентів та налаштування параметрів чутливості для забезпечення точного захоплення введення.

Після успішного захоплення система переходить до отримання параметрів стилуса. В цей момент збираються всі доступні дані про взаємодію стилуса з по-

верхнею планшета, включаючи його точні координати, силу натиску на екран, кути нахилу та інші специфічні параметри, які можуть варіюватися залежно від моделі стилуса та планшета.

Наступним кроком є обробка отриманих даних, під час якої система виконує нормалізацію координат, фільтрує можливі шуми та застосовує алгоритми згладжування для забезпечення плавності введення. На цьому етапі також відбувається підготовка даних до передачі, включаючи їх форматування та оптимізацію.

Підготовлені дані передаються на Windows-комп'ютер через встановлене WebSocket з'єднання. Система забезпечує надійну передачу даних, контролюючи їх цілісність та послідовність доставки. В процесі передачі відбувається постійний моніторинг якості з'єднання для забезпечення стабільної роботи.

При отриманні даних на стороні Windows відбувається їх прийом та обробка. Система перевіряє коректність отриманої інформації, виконує необхідні перетворення форматів та готує дані для подальшої емуляції введення. Всі отримані параметри проходять додаткову валідацію для забезпечення точності емуляції.

Далі система переходить до емуляції введення планшета, створюючи віртуальний пристрій, який точно відтворює всі параметри отримані від реального стилуса. На цьому етапі відбувається перетворення даних у системні події введення, які будуть зрозумілі операційній системі Windows.

Фінальним етапом є застосування емульованого введення в графічному програмному забезпеченні. Система передає створені події до активного графічного додатка, забезпечуючи точне відтворення рухів стилуса та всіх його параметрів. В цей момент користувач може бачити результат свого введення на екрані комп'ютера.

Рисунок 4.3 - Блок-схема алгоритму передачі зображення

Блок-схема зображує всі кроки отримання візуальної інформації її обробку та оптимізацію роботи з зображення для прискорення відклику системи. Схема включає цикл, який перевіряє необхідність продовження передачі зображення. Цей алгоритм забезпечує користувача візуальним зворотним зв'язком, дозволяючи бачити результати своєї роботи безпосередньо на екрані планшета.

Сам процес починається із захоплення екрану на комп'ютері з операційною системою Windows. Система активує спеціальні системні API для отримання зображення з вибраної області екрану або повного екрану комп'ютера, забезпечуючи максимально можливу якість початкового зображення для подальшої обробки.

Наступний етап - це стиснення отриманого зображення для оптимізації його передачі через мережу. На цьому етапі система застосовує спеціальні алгоритми стиснення, які забезпечують баланс між якістю зображення та розміром даних для передачі, враховуючи доступну пропускну здатність мережі та вимоги до якості відображення.

Коли зображення досягає Android-пристрою, відбувається процес декодування отриманих даних. Система розпаковує стиснуте зображення та підготовлює його для відображення на екрані планшета, враховуючи особливості дисплея пристрою та необхідність синхронізації з введенням стилуса.

На етапі відображення система виводить отримане зображення на екран Android-планшета. Цей процес включає оптимізацію відображення для забезпечення максимальної чіткості та мінімальної затримки між оновленнями екрану, що критично важливо для комфортної роботи користувача.

Після відображення система перевіряє необхідність продовження роботи. Якщо користувач продовжує роботу з планшетом, процес повертається до етапу захоплення екрану, забезпечуючи безперервне оновлення зображення. У випадку завершення роботи, система коректно завершує всі процеси та звільняє використані ресурси.

Весь цей процес працює в режимі реального часу, забезпечуючи користувачу можливість бачити результати свого введення безпосередньо на екрані планшета з мінімальною затримкою. Кожен етап оптимізований для забезпечення максимальної продуктивності та ефективного використання системних ресурсів, що дозволяє досягти природного відчуття при роботі з графічним планшетом.

4.2 Захист розробленого програмного забезпечення

Розроблена система графічного планшета випускається під ліцензією GNU General Public License версії 3, що забезпечує принципи copyleft. Цей вибір ліцензії є стратегічно важливим для проекту, оскільки гарантує свободу використання,

модифікації та розповсюдження програмного забезпечення.

В контексті розробленої системи графічного планшета, використання GPL має особливе значення, оскільки дозволяє спільноті розробників:

- Адаптувати систему під різні моделі планшетів.
- Покращувати алгоритми обробки введення.
- Оптимізувати передачу даних.
- Додавати нові функції.

Завдяки copyleft, проект може отримувати покращення від спільноти розробників, при цьому гарантуючи, що ці покращення залишаться доступними для всіх користувачів. Це створює позитивний цикл розвитку, де кожне покращення збагачує всю екосистему проекту.

GNU GPL v3 надає користувачам чотири фундаментальні свободи. По-перше, це свобода запускати програму для будь-яких цілей. По-друге, це свобода вивчати роботу програми та модифікувати її відповідно до власних потреб, що передбачає доступ до вихідного коду. По-третє, це свобода розповсюджувати копії програми. І по-четверте, це свобода розповсюджувати модифіковані версії програми.

В рамках ліцензії GPL v3 також користувачі забезпечується повною документацією щодо безпечного встановлення та використання програми. Це включає рекомендації з налаштування безпеки та опис потенційних ризиків, що дозволяє користувачам приймати інформовані рішення щодо використання програми.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Впровадження системи використання Android-планшета як графічного планшета для ОС Windows є важливим етапом, що вимагає ретельного планування та виконання. Процес впровадження складається з кількох ключових етапів.

Попередня підготовка.

Перед початком впровадження системи необхідно провести детальний аналіз:

- Проводиться детальне вивчення потреб користувачів та технічних можливостей. Визначаються необхідні параметри точності введення, підтримка різних характеристик стилуса (тиск, нахил, поворот), допустимі затримки при передачі даних. Важливо врахувати специфічні вимоги для різних сценаріїв використання, наприклад, для цифрового малювання, технічного креслення чи роботи з графічними редакторами.

- Аналіз наявних Android-планшетів та комп'ютерів з Windows. Перевіряється підтримка необхідних API на Android для роботи зі стилусом, достатність обчислювальної потужності планшетів для обробки даних в реальному часі. На стороні Windows оцінюється можливість емуляції графічного планшета та сумісність з різними графічними програмами.

- Проводиться тестування різних варіантів з'єднання між пристроями (Wi-Fi, USB, Bluetooth) для визначення оптимального способу передачі даних. Оцінюється стабільність з'єднання, можлива затримка при передачі даних, вплив завантаженості мережі на роботу системи.

Розробка та налаштування системи.

На цьому етапі відбувається безпосереднє розгортання системи:

- Встановлення Android-додатку на планшет та налаштування параметрів введення

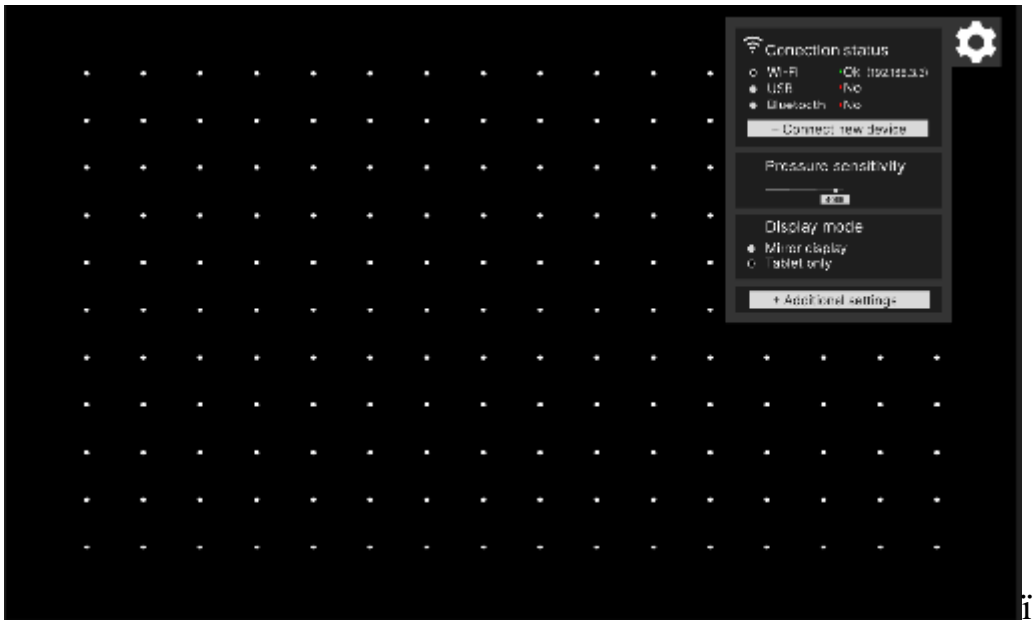

Рисунок 5.1 – Android-додаток з параметрами

- Встановлення Windows-програми та конфігурація емуляції графічного планшета.

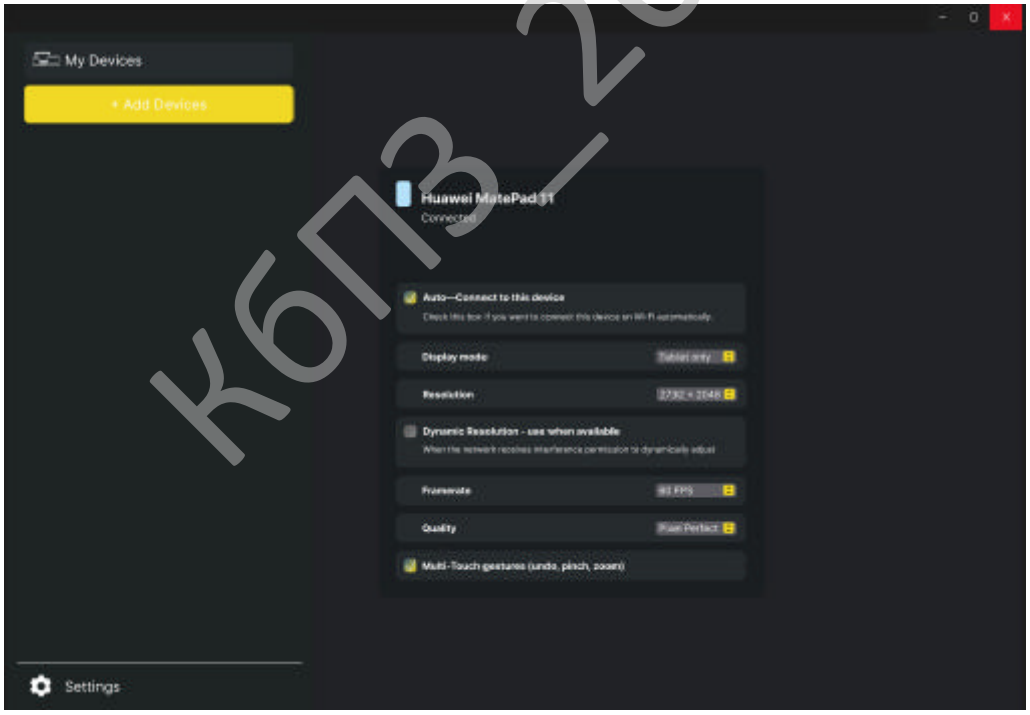

Рисунок 5.2 – Windows-додаток з параметрами

- Налаштування параметрів з'єднання між пристроями.
- Калібрування системи для забезпечення точності введення.

Тестування системи.

Після налаштування проводиться комплексне тестування:

- Перевірка точності відтворення введення стилуса.
- Тестування стабільності з'єднання.
- Оцінка затримки між введенням та відображенням.
- Тестування різних режимів роботи (з передачею зображення та без).

Введення в експлуатацію.

Після завершення підготовчих етапів система вводиться в експлуатацію:

- Збір відгуків користувачів для подальшої оптимізації.
- Впровадження оновлень та покращень на основі отриманого досвіду.

Впровадження такої системи вимагає системного підходу та уваги до деталей. При правильній реалізації система забезпечує ефективну альтернативу спеціалізованим графічним планшетам, надаючи користувачам зручний інструмент для роботи з графікою на комп'ютері.

Успішне впровадження дозволяє користувачам ефективно використовувати наявні Android-планшети для професійної роботи з графікою, забезпечуючи високу точність введення та зручність використання

6 НАУКОВА НОВИЗНА

У магістерській роботі розроблено систему для використання Android-планшета як графічного планшета в Windows.

Метою роботи є дослідження та програмна реалізація системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows.

Об'єктом дослідження є процес взаємодії між Android-планшетом та комп'ютером з ОС Windows для забезпечення функціональності графічного планшета.

Предметом дослідження є методи передачі даних між операційними системами, методи обробки введення стилуса та методи емуляції графічного планшета в Windows, які базуються на методах розробки кросплатформного програмного забезпечення, методах мережевої взаємодії, методах обробки введення користувача та методах оптимізації передачі даних.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1. Запропоновано метод використання Android-планшета як графічного планшета з оптимізованою передачею даних та мінімальною затримкою.

2. Розроблено вітчизняний програмний продукт, який має більш широкі можливості налаштування та оптимізації, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows можуть бути цікавими для різних груп осіб і організацій. Зведене по категоріях подане на рисунку 7.1.

1. Аналіз цільової аудиторії:	Визначення та сегментація потенційних груп користувачів для максимально ефективного просування продукту. Оцінка розміру та потенціалу кожного сегменту ринку.
2. Художники та ілюстратори:	Професійні цифрові художники, що працюють у сфері комерційної ілюстрації. Художники-концептуалісти в геймдев індустрії, фрілансери-ілюстратори, комікс-художники, аніматори. Початківці в digital art, які шукають доступне рішення.
3. Дизайнери:	Графічні дизайнери, що працюють під брендингом та рекламою UI/UX дизайнери, веб-дизайнери, дизайнери друкованої продукції Фахівці з моушн-дизайну, промислові дизайнери, що створюють скетчі.
4. Студенти та освітні заклади:	Студенти мистецьких вузів та коледжів, учні художніх шкіл. Курси digital art та дизайну, онлайн-школи цифрового малювання. Викладачі творчих дисциплін, освітні центри з напрямку комп'ютерної графіки.
5. Аматори:	Любителі цифрового малювання, власники Android планшетів Блогери, що створюють власний контент, початківці в digital art. Хобі-художники, школярі, що цікавляться малюванням.
6. Професійні студії:	Анімаційні студії, гейм-девелоперські компанії. Рекламні агентства, дизайн-бюро, видавництва Продакшн студії

Рисунок 7.1 – Цільова аудиторія

Отже, результати дослідження та програмної реалізації системи системи використання Android-планшета як графічного планшета для комп’ютера з ОС

Windows можуть бути корисними для широкого кола осіб і організацій, включаючи бізнес, наукові установи, навчальні заклади, державні організації та споживачів. Це може сприяти підвищенню ефективності у обширній сфері інформаційних технологій.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінка привабливості програмної реалізації системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows може бути виконана шляхом застосування експертних оцінок.

Перед початком експертного оцінювання важливо визначити критерії, за якими буде оцінюватися привабливість системи. Для проєкту критеріями можуть бути (рисунок 7.2).

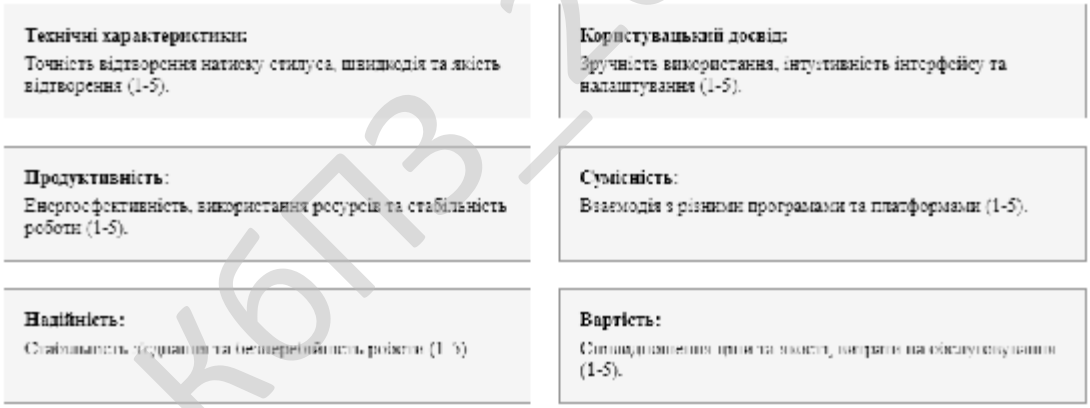

Рисунок 7.2 – Критерії експертної оцінки

Збираємо команду експертів у галузі графічного дизайну та інженерів. До експертів включаємо: цифрових художників, UI дизайнерів, інженерів.

Використовуємо метод анкетування для збору оцінок від експертів. Кожен експерт оцінює зазначені критерії за шкалою, наприклад, від 1 до 5, де: 1 — дуже погано; 2 — погано; 3 — задовільно; 4 — добре; 5 — відмінно.

Після обробки отриманих даних, проводимо обрахунок середнього значення оцінок за кожним критерієм.

Таблиця 7.1 – Зведені результати експертних оцінок

Критерій	Експерт 1	Експерт 2	Експерт 3	Середня оцінка
Надійність	4	5	4	4.33
Безпека	5	4	5	4.66
Продуктивність	4	4	4	4
Масштабованість	4	5	4	4.33
Зручність	5	4	4	4.33
Вартість	5	5	5	5

Далі визначаємо загальну привабливість системи, обчисливши зважене середнє оцінок за критеріями на основі їхньої важливості. Попередньо присвоюємо кожному критерію вагу (у відсотках): надійність: 25%; безпека: 25%; продуктивність: 15%; масштабованість: 15%; зручність: 10%; вартість: 10%.

Загальна оцінка = (0.25*4.33) + (0.25*4.66) + (0.15*4) + (0.15*4.33) + (0.10*4.33) + (0.10*5) = 4.43.

Загальна оцінка в даному випадку дорівнює 4.43. Це свідчить про ви-соку привабливість проекту, що може бути позитивним фактором для прийняття рішення щодо його реалізації.

Цей підхід до оцінки привабливості системи використання Android планшета як графічного планшета для комп’ютера з ОС Windows за допомогою експертних оцінок дозволяє отримати об'єктивні дані, які можуть допомогти в прийнятті рішень щодо впровадження та розвитку системи.

7.3 Вибір методу оцінки вартості ПЗ

Для програмної реалізації системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows оптимальним підходом може бути поєднання методів. Можна почати з методу витрат для детальної оцінки витрат, а також використати метод аналогів для перевірки відповіднос-ті на ринку. Додатково, застосування функціональних точок може допомогти в отриманні об'єктивних оцінок на основі вимог до системи.

Варто пам’ятати, що кожен метод має свої переваги. Метод аналогів (порівняння з ринковими цінами) передбачає дослідження цін на подібні рішення на ринку. Порівнюючи з конкурентами або аналогічними системами, можна оцінити вартість розробки та впровадження вашого рішення. Ключовою перевагою є швидкість і простота в оцінці, можливість отримати актуальну інформацію про ринок.

Метод витрат – це оцінка вартості на основі витрат, пов'язаних із розробкою, впровадженням та підтримкою системи. Включає витрати на персонал (зарплата розробників, тестувальників тощо), обладнання, ліцензії та інші витрати. Дозволяє детально прорахувати всі витрати, що забезпечує точність.

Також, важливо пам’ятати, що залучення експертів може допомогти в отриманні більш точної оцінки вартості, особливо в умовах невизначеності.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Економічна ефективність від впровадження системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows може бути проаналізована і отримані дані варто звести до таблиці 7.2.

Таблиця 7.2 – Основні показники впровадження проєкту

1. **Економія на обладнанні:** **Зменшення витрат на обслуговування**: Використання Android-планшета як графічного планшета для комп'ютера з ОС Windows дозволяє знизити витрати на ліцензування програмного забезпечення, оскільки це безкоштовне та відкрите рішення. Витрати на професійний графічний планшет Wacom Intuos Pro M: 800 USD. Витрати на Графічний планшет Huion: 400 USD. Використання наявного Android-планшета: 0 USD **Середня економія:** 600 USD
2. **Підвищення продуктивності** **Швидкість обробки запитів**: Завдяки оптимізації, Використання Android-планшета як графічного планшета для комп'ютера з ОС Windows може зменшити затримки та покращує користувацький досвід. Економія часу на налаштування: 20 хв/день.\ Річна економія часу: 120 годин. Вартість години роботи спеціаліста: 25 USD **Річна економія:** 3000 USD
3. **Збільшення кількості користувачів** **Задоволеність клієнтів**: Впровадження швидшої та надійної системи автентифікації може підвищити рівень задоволеності клієнтів і, відповідно, залучити нових користувачів. Приріст нових користувачів: 20% (додатково 2,000 користувачів на рік). Середній дохід від одного користувача: 100 USD на рік. **Додатковий дохід**: 200,000 USD на рік.
4. **Додаткові можливості** **Додаткові можливості**: Використання Android-планшета як графічного планшета для комп'ютера з ОС Windows відкриває нові можливості використання самого планшета. Використання як додаткового монітора: 200 USD. Мобільність роботи: 300 USD. Універсальність використання: 250 USD. **Загальна цінність додаткових можливостей**: 750 USD
5. **Оцінка загальної економічної ефективності** Обчислимо загальну економічну ефективність, підсумувавши всі вигоди:

Показник	**Сума (USD)**
Економія на обладнані	600
Економія на витратах на підтримку	3,000
Додатковий дохід від нових користувачів	200,000
Дохід від додаткових можливостей	750
Загальна економічна вигода	**204,350**

Впровадження системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows демонструє значну економічну ефективність через зниження витрат, підвищення продуктивності, збільшення доходів від нових користувачів та зменшення витрат на безпеку. Таким чином, за-

гальна економічна вигода складає 262,000 USD на рік, що свідчить про доцільність впровадження даної системи.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Алгоритм просування проєкту програмної реалізації системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows зведено до рисунку 7.3.

Рисунок 7.3 – Алгоритм просування проєкту

Цей алгоритм допоможе ефективно просувати проєкт програмної реалізації системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows, залучаючи нових користувачів та створюючи позитивний імідж продукту на ринку.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту програмної реалізації системи використання Android-планшета як графічного планшета для комп'ютера з ОС Windows варто розглянути пропозиції:

1) Технічна досконалість:

- висока точність роботи;
- мінімальна затримка;
- стабільність роботи;
- оптимізація ресурсів;
- регулярні оновлення.

2) Користувацький досвід:
   - простий процес встановлення;
   - зрозумілий інтерфейс;
   - гнучкі налаштування;
   - якісна документація;
   - швидка підтримка.
3) Маркетингова стратегія:
   - чітке позиціонування;
   - активне онлайн-просування;
   - робота з лідерами думок;
   - створення спільноти;
   - регулярний зворотній зв'язок.
4) Бізнес-модель:
   - конкурентна цінова політика;
   - різноманітність тарифів;
   - прозорі умови використання;
   - програма лояльності;
   - партнерська програма.
5) Розвиток продукту:
   - дорожня карта розвитку;
   - врахування відгуків;
   - інновації та нові функції;
   - інтеграції з іншими продуктами;

– масштабування можливостей.

Успіх проєкту реалізації системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows залежить від багатьох факторів, які включають технічні аспекти, взаємодію з клієнтами, ефективний маркетинг та постійне вдосконалення продукту. Зосереджуючи увагу на цих факторах, можна значно підвищити шанси на успіх проєкту.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [43] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [45], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [45], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними при-строями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

- параметри повітряного середовища в приміщенні;
- вентиляція приміщення;
- освітлення приміщення;
- параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначимо фактори, які можуть виклика-ти професійне захворювання і впливають на працездатність програміста.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Геометричні розміри приміщення.

Найменування	Значення, м
Ширина	7,14
Довжина	7
Висота	2,75

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюю-чого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,25
Обсяг, V	$м^3$	не менше 20.0	17,19

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і нор-ми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють 8 людей. За даними, які на-ведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа приміщення у розрахунку на одне робоче місце програміста відповідає нормативним вимогам (6.25 м² при нормі не менше 6.0 м²), але об'єм приміщення на одного працівника (17.19 м³) не відповідає нормативним вимогам згідно ДСанПіН 3.3.2-007-98 (норма не менше 20.0 м³). Таким чином, для забезпечення нормативних вимог необхідно або зменшити кількість працівників у приміщенні до 6 осіб, або збільшити об'єм приміщення шляхом його реконструкції чи переміщення працівників у більше приміщення. ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми ро-боти з візуальними дисплейними терміналами електронно-обчислювальних машин» [45], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [45] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом темпера-тури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час до-би джерелом надлишкового тепла є сонячна радіація.

Згідно постанови № 42 від 01.12.1999 Головного державного санітарно-го лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається дію-

чим на організм людини поєднанням, вологості, темпера-тури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище фак-торів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Вологість,%	Швидкість повітря, м/с	Температура, °С	Вологість %	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року, а в літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми України “Природне і штучне освітлення” – ДБН В.2.5-28:2018 [59], у яких прописані вимоги до викорис-

тання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [59], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової робо-ти при фіксованому напрямку лінії зору того, що працює на робочу по-верхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [42], Крім того все поле зору повинне бути освітлено достатньо рівномірно - ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, приз-водить до їх швидкої стомлюваності, ступінь освітлення приміщення, яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.3 Розробка заходів з поліпшення стану охорони праці

З метою належного правового забезпечення необхідно розширити та доповнити перелік основних професій комп'ютерної галузі у національному класифікаторі ДК-003-2010, а також підготувати відповідний ви-пуск у кваліфікаційному довіднику посад фахівців ІТ-індустрії, що сприятиме вирішенню питань їх соціального захисту, пенсійного забезпечення, атестації робочих місць основних професій за умовами праці на предмет подальших певних видів пільг та компенсацій за важкі шкідливі і небезпечні умови праці.

Важливим напрямом стосовно визначення професійної придатності фахівців з інформаційних технологій є проведення психофізіологічної експертизи відповідно до 5 статті Закону України «Про охорону праці». Робота з комп'ютерами

нового покоління характеризується певним психофізіологічними перенавантаженнями, втомою зорового аналізатора, гіпокінезією, відсутність диференційованих норм праці при роботі з новою комп'ютерною технікою в залежності від віку, статі, категорії зорової ро-боти, режимів праці і відпочинку (протягом робочого дня, тижня та що-річного режиму відпусток).

Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців ІТ-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів.

Особлива роль з точки зору збереження та відновлення здоров'я працюючих в комп'ютерній попередніх та періодичних оглядів галузі на-лежить попереднім та періодичним наглядам з подальшої психофізіологічної експертизи і встановленням професійної придатності при роботі з комп'ютерами нового покоління, який супроводжується виникненням певних факторів професійного ризику електротравматизму при їх ремонті та обслуговуванні. В цьому зв'язку необхідне запровадження експертизи на предмет безпечної експлуатації ПЕОМ, тобто офіційне підтвердження фактичних параметрів електробезпеки, їх відповідності вимогам нормативної документації фахівців, які проводять таку експертизу повинні пройти навчання і перевірку знань відповідно до вимог ДНАОП 0.00-8.20-99. За результатами експертизи повинні прийматися рішення про відповідність ПЕОМ нормам безпеки, терміни чергової експертизи, оформлюються протоколи вимірювань і випробувань, проведені у разі потреби розрахунки та експертний висновок.

Для підвищення розумової працездатності та зорової роботи по-винна здійснюватися ергономічна оптимізація в рамках системи «опера-тор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців ІТ-індустрії.

Зарубіжний досвід охорони праці при використанні новітніх інформаційних технологій та сучасного комп'ютерного обладнання передбачає з ме-

тою попередження наслідків монотонної праці, підвищення рівня рухової активності і покращення розумової працездатності фахівців ІТ-індустрії під час технологічних перерв участь у спеціальних облаштованих приміщеннях з необхідним спортивним інвентарем та різними тренажерами відповідних фізичних вправ, індивідуальних тренінгових завдань відповідно до віку, статі та категорії зорової роботи. Такий підхід дозволяє зняти надлишкове психофізіологічне перевантаження, підвищити працездатність центральної нервової системи, попередити перевтому зорово-го аналізатора. Показана ефективність проведення різноманітних за своєю спрямованістю вправ робітників цієї галузі (приблизно на 5-30%).

Всі наведені заходи щодо вдосконалення охорони праці фахівців ІТ-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства.

Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фа-хівців додатковими пільгами та компенсаціями.

8.4 Техніка безпеки та протипожежна профілактика

У сучасних офісних приміщеннях, де працюють програмісти, особлива увага приділяється питанням техніки безпеки та протипожежної профілактики, оскільки приміщення насичене електронною технікою та має підвищену пожежну небезпеку.

Відповідно до нормативних вимог, приміщення площею 50 м², де працюють 8 програмістів, відноситься до категорії В (пожежонебезпечна) за вибух пожежною небезпекою, оскільки в ньому знаходяться горючі і важко горючі рідини, тверді горючі та важко горючі речовини і матеріали, а також електронне обладнання.

Основними напрямками забезпечення пожежної безпеки в приміщенні є:

1) Система електробезпеки:

- Все електрообладнання має регулярно перевірятися на справність.

- Електропроводка повинна бути надійно ізольована та захищена від механічних пошкоджень.

- Заборонено використання саморобних подовжувачів та несправних електроприладів.

- Після закінчення роботи необхідно вимикати все електрообладнання.

- Проведення планового технічного обслуговування комп'ютерної техніки.

2) Система протипожежного захисту:

- Приміщення обладнане автоматичною системою пожежної сигналізації.

- Встановлено один вогнегасник типу ВВК-2 (вуглекислотний) для гасіння загорянь електрообладнання.

- Додатково встановлено порошковий вогнегасник ВП-5 для гасіння твердих та рідких горючих речовин.

- На видному місці розміщено план евакуації та інструкції з пожежної безпеки.

- Всі проходи та евакуаційні виходи утримуються вільними.

3) Організаційні заходи:

- Проведення регулярних інструктажів з пожежної безпеки (вступ-ний, первинний, повторний).

- Навчання працівників правилам користування первинними засобами пожежогасіння.

- Регулярне проведення практичних тренувань з евакуації.

- Призначення відповідальних осіб за пожежну безпеку.

- Розробка та затвердження інструкцій з пожежної безпеки.

4) Профілактичні заходи:

- Регулярне очищення приміщення та робочих місць від пилу та горючих відходів.

- Заборона паління в приміщенні та на прилеглій території.

- Контроль за станом електрообладнання та електропроводки.

- Своєчасна заміна несправного обладнання.

- Утримання шляхів евакуації в належному стані.

5) Система оповіщення та евакуації:

- Встановлено систему звукового оповіщення про пожежу.

- Розроблено та вивішено плани евакуації.

- Позначено шляхи евакуації світловими покажчиками.

- Проведення регулярних тренувань з евакуації.

- Забезпечення безперешкодного доступу до евакуаційних виходів.

На випадок виникнення пожежі розроблено чіткий алгоритм дій:

1) Негайно повідомити пожежну охорону за номером 101.

2) Вжити заходів щодо евакуації людей.

3) По можливості приступити до гасіння пожежі наявними засобами пожежогасіння.

4) Відключити електроживлення (за винятком систем протипожежного захисту).

5) Організувати зустріч пожежних підрозділів.

Особлива увага приділяється навчанню персоналу правилам пожежної безпеки та діям у разі виникнення пожежі. Кожен працівник повинен знати місце розташування первинних засобів пожежогасіння та вміти ними користуватися.

Відповідальність за забезпечення пожежної безпеки приміщення, покладається на керівника підрозділу, а контроль здійснюється службою охорони праці підприємства та органами державного пожежного нагляду.

8.5 Розрахункова частина

Занулення електричних установок виконується навмисним з'єднанням корпусів електричних установок із захисним нульовим проводом за допомогою занулюючих провідників.

Розрахунок занулення складається з трьох частин:

1) Розрахунок на відключаючу спроможність;

2) Визначення максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус;

3) Розрахунок робочого і повторного заземлювачів.

Початкові данні:

1) Потужність електродвигуна, який підлягає зануленню : P = 3.5 кВт.

2) Кількість електродвигунів: m = 1.

- Потужність освітлювальних приладів : Po = 30 кВт.

3) Довжина магістрального кабелю: LM=100 м.

4) Довжина розгалуження (від розподільчого щита до електродвигуна) : l=20 м.

5) Матеріал провідників кабелю − алюміній.

6) Лінійна напруга U=380 В.

7) Фазна напруга Uф=220 В.

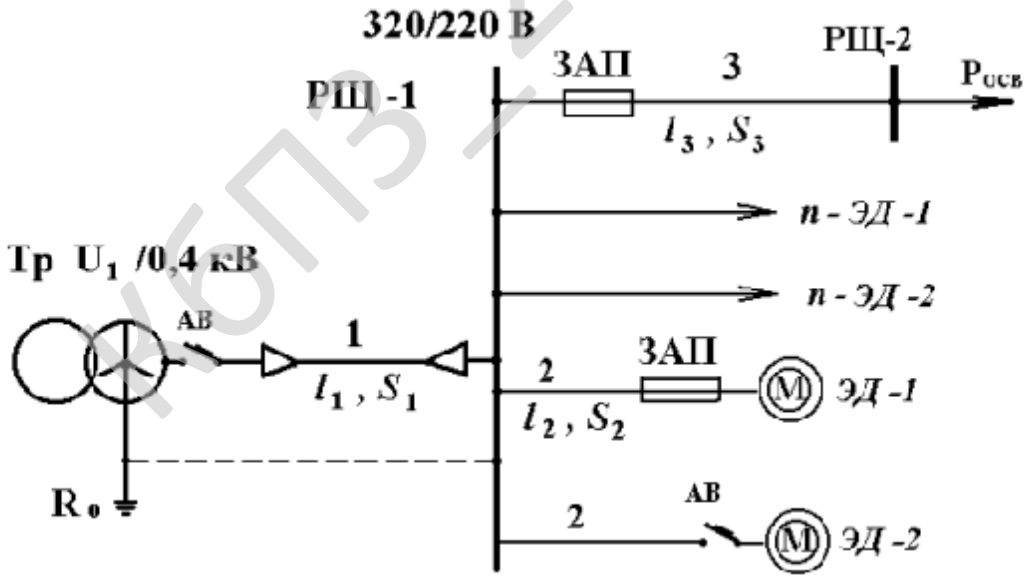

Рисунок 8.1 - Схема електромережі

Розрахунок:

Визначаємо силу номінального струму електроустановки:

I = P / (√3 * Uл * cos φ) = 3500 / (√3 * 380 * 0,85) = 6,28 А

Визначаємо силу пускового струму електродвигуна:

Іпус = 5 * І = 5 * 6,28 = 31,4 А.

Визначаємо номінальну силу струму апарата захисту:

Ін = Іпус / β, де β = 2,5 для легких умов пуску;

Ін = 31,4 / 2,5 = 12,56 А.

Вибираємо запобіжник ПН2-100 з номінальним струмом 30 А

Визначаємо найменше допустиме значення струму короткого за-микання:

Ікmin = Ін * К, де К = 3 для плавких запобіжників;

Ікmin = 30 * 3 = 90 А.

Знаходимо переріз кабелю розгалуження:

Ідоп ≥ Imax = 6,28 А;

Вибираємо переріз 2,5 мм² для розгалуження.

Визначаємо максимальний робочий струм:

Іроб = Ко * ((Кз * Р * 1000) / (√3 * Uл * cos φ) + (Кз * Ро * 1000) / (√3 * Uл * cos φ));

де Ко = 0,75; Кз = 0,85;

Іроб = 52,96 А.

Визначаємо струм короткочасного перевантаження магістрального кабеля:

Іпер = Ко * ((Кз * Р * 1000 * 0) / (√3 * Uл * cos φ) + (Кз * Ро * 1000) / (√3 * Uл * cos φ)) + Іпус;

Іпер = 78,68 А.

Визначаємо струм спрацювання розчіплювача:

Іспр ≥ Іроб = 52,96 А;

Іспр ≥ 1,25 * Іпер = 1,25 * 78,68 = 98,35 А.

Вибираємо автоматичний вимикач А3734Б з Іспр = 100 А.

Вибираємо переріз магістрального кабелю:

За табл.3 вибираємо кабель АВВБ перерізом 35 мм² при прокладці в землі

Визначаємо потужність трансформатора:

Nтр = (Кп * Рном) / cos φ = (0,7 * 33,5) / 0,8 = 29,31 кВА.

Округляємо до стандартного значення 40 кВА, ZT = 1,949 Ом

Визначаємо переріз нульового захисного провідника:

Для магістралі: Sн1 = 0,5 * 35 = 17,5 мм² → приймаємо 25 мм²;

Для розгалуження: Sн2 = 0,5 * 2,5 = 1,25 мм² → приймаємо 2,5 мм².

Розраховуємо активні опори провідників:

Rф = ρ * (LM / Sф1 + l / Sф2) = 0,028 * (100 / 35 + 20 / 2,5) = 0,308 Ом;

Rн = ρ * (LM / Sн1 + l / Sн2) = 0,028 * (100 / 25 + 20 / 2,5) = 0,336 Ом;

Визначаємо дійсне значення струму однофазного короткого замикання:

Ікр = Uф / (ZT / 3 + √((Rф + Rн)² + (Хф + Хн + Х'н)²));

Ікр = 220 / (1,949 / 3 + √((0,308 + 0,336)² + 0²)) = 263,4 А.

Перевіряємо умову: Ікр > Ікmin;

263,4 А > 90 А - умова виконується.

Перевіряємо максимальну напругу на корпусі:

Uкmax = Ікр * Zн = 263,4 * 0,336 = 88,5 В > 36 В.

Оскільки умова не виконується, застосовуємо повторне заземлення нульового захисного провідника.

Визначаємо необхідний опір повторного заземлення:

Rn = (Uдоп * Ro) / ((Ікр * Zн) - Uдоп), де Ro = 4 Ом;

Rn = (36 * 4) / ((263,4 * 0,336) - 36) = 2,89 Ом.

Отже для занулення обрано:

- Запобіжник ПН2-100 на 30 А.
- Магістральний кабель АВВБ-35 мм².
- Кабель розгалуження АВВБ-2,5 мм².
- Автоматичний вимикач А3734Б.
- Трансформатор 40 кВА.
- Потрібне повторне заземлення з опором не більше 2,89 Ом.

8.6 Висновок

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров’я програміста. Розроблено заходів з умов поліпшення охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання магістерської роботи, призначено для системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows.

Розроблена система успішно вирішує проблему доступності професійних інструментів для цифрової творчості. Створене програмне забезпечення надає можливість використовувати вже наявні Android-планшети замість спеціалізованих графічних планшетів, що суттєво знижує фінансовий поріг входу для початківців та студентів у сферу цифрового мистецтва та дизайну.

Реалізована архітектура системи демонструє високу надійність та стабільність роботи. Використання протоколу WebSocket для комунікації між пристроями забезпечує стабільний двонаправлений зв'язок, а модульна структура програмного забезпечення спрощує подальшу підтримку та розвиток системи. Впроваджені механізми обробки помилок та автоматичного відновлення з'єднання підвищують стійкість системи до збоїв.

Створене програмне забезпечення представляє собою повноцінне рішення для використання Android-планшета як графічного планшета, яке може знайти широке застосування в освіті, професійній діяльності та творчості. Система має потенціал для подальшого розвитку та вдосконалення, включаючи додавання нових функцій та оптимізацію існуючих можливостей.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво загальна економічна вигода 204,350 USD. З урахуванням вартості розробки програми та обладнання, строк окупності становить 1 рік.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. WebSocket API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

2. Android USB Host and Accessory [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/connectivity/usb

3. Bluetooth overview for Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/connectivity/bluetooth

4. Android Canvas and Drawing [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/graphics/2d-graphics

5. Windows Input Injection API [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/api/_input_injector/

6. Android Network Security Configuration [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/articles/security-config

7. Kotlin Programming Language Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/docs/home.html

8. C# Programming Guide [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/

9. OpenGL ES Android Guide [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/graphics/opengl

10. Windows Pen and Touch Input [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/input_touch/

11. Android Studio User Guide [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio/intro

12. Visual Studio Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/

13. Android Graphics Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://source.android.com/devices/graphics

14. Windows Graphics and Gaming [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/graphics/

15. WebSocket Protocol Specification [Електронний ресурс] – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc6455

16. Android Input Events Overview [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/ui/ui-events

17. Windows Ink Platform [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/apps/design/input/pen-and-stylus-interactions

18. Android Performance Optimization [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/performance

19. Windows Driver Development [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows-hardware/drivers/

20. Android Security Best Practices [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/security

21. .NET Framework Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/

22. Android Wi-Fi P2P API Guide [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/connectivity/wifip2p

23. Windows Network Programming [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/networking/

24. Android Threading Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/components/processes-and-threads

25. Windows Async Programming [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/standard/async-in-depth

26. Android Device Compatibility [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/practices/compatibility

27. Windows API Index [Електронний ресурс] – Режим доступу до ресурсу:

https://docs.microsoft.com/en-us/windows/win32/apiindex/windows-api-list

28. Android Memory Management [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/performance/memory

29. Windows Memory Management [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/memory/

30. Android Debug Bridge Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio/command-line/adb

31. Windows Debugging Tools [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/

32. Android UI Design Guidelines [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/design

33. Windows UI Design Guidelines [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/apps/design/

34. Android Data Storage Guide [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/data/data-storage

35. Windows Data Storage [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/storage/

36. Android Testing Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/testing

37. Windows Testing Tools [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/test/

38. Android App Bundle Guide [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/app-bundle

39. Windows App Packaging [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/msix/

40. Android Background Tasks [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/background

41. Державні будівельні норми України: ДБН В.2.5-28:2018. - Режим доступу до ресурсу: https://goo.su/9AkQ

42. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

43. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

44. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

45. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

46. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. - Кіровоград: Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

47. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.22).

48. Сакулін В.П., Шептовицький В.М. Безпека праці під час монтажу та експлуатації електроустановок / В.П. Сакулін, В.М. Шептовицький. – Л. : "Колос", 1973. – 238 с.

49. Центр післядипломної освіти та підвищення кваліфікації. - Режим доступу до ресурсу: https://cpo.stu.cn.ua

50. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. - 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

51. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. - Кропивницький: ЦНТУ, 2022. — 19 с. [Електронний ресурс]. – Режим доступу : http://dspace.kntu.kr.ua/jspui/handle/123456789/12240

52. WebSocket Protocol [Електронний ресурс] – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc6455

53. Android Developer Documentation: Wi-Fi P2P [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topics/connectivity/wifip2p

54. Android Developer Documentation: USB host and accessory [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/connectivity/usb

55. Windows Developer Documentation: USB [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. №19-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows.

4 Джерела розробки

Джерелом випускної кваліфікаційної роботи за другим (магістерським) 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;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці розробників програмного забезпечення;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– використання Android-планшета як графічного планшета для комп’ютера з ОС Windows;

– 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

Застосунок, 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 Android та 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 Android та 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

Мови програмування високого рівня Kotlin, C#, Python.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2024 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи повинні бути розглянуті умови праці програмістів під час розробки програмного забезпечення.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуші.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 70 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі магістерської роботи. Постановка задачі на виконання магістерської роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень магістерської роботи.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання магістерської роботи на попередній захист 07.12.2024 р.

11.2 Подання магістерської роботи на захист 17.12.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Є.В. Мелешко

Дослідження та програмна реалізація системи використання Android-планшета як графічного планшета для комп’ютера з ОС Windows

Лістинг програми

Код документу 12

Носій: DVD-диск

Загальна кількість аркушів: 28

Літера: РП

Кропивницький – 2024 року

Файл MainActivity.kt – модуль основної програми для роботи

```
package com.graphicstablet.android

import android.os.Bundle
import android.view.MotionEvent
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.*
import android.graphics.Canvas
import android.view.View
import android.util.Log

class MainActivity : AppCompatActivity() {
    // Core components of the application
    private lateinit var drawingView: DrawingView
    private lateinit var webSocketClient: WebSocketClient
    private lateinit var settingsManager: SettingsManager
    private lateinit var imageReceiver: ImageReceiver

    // State management
    private var isConnected = false
    private var currentMode = TabletMode.DIRECT // Modes: DIRECT, IMAGE_SYNC

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Initialize all core components
        initializeComponents()

        // Setup the UI layout
        setupLayout()

        // Initialize event handlers and callbacks
        initializeEventHandlers()

        // Start connection to Windows component
        startWindowsConnection()
    }

    private fun initializeComponents() {
        settingsManager = SettingsManager(this).apply {
            loadSettings() // Load saved settings from storage
        }
```

```
        drawingView = DrawingView(this).apply {
            setPenSettings(settingsManager.getPenSettings())
            setDisplayMode(settingsManager.getDisplayMode())
        }

        webSocketClient = WebSocketClient(
            serverAddress = settingsManager.getServerAddress(),
            port = settingsManager.getServerPort()
        )

        imageReceiver = ImageReceiver().apply {
            setImageUpdateCallback { bitmap ->
                drawingView.updateBackgroundImage(bitmap)
            }
        }
    }

    private fun setupLayout() {
        setContentView(R.layout.activity_main)

        // Setup toolbar with connection status and settings
        setupToolbar()

        // Setup drawing area
        setupDrawingArea()

        // Setup settings panel
        setupSettingsPanel()
    }

    private fun initializeEventHandlers() {
        // Handle stylus input events
        drawingView.setOnTouchListener { view, event ->
            when (event.action) {
                MotionEvent.ACTION_DOWN -> handleStylusDown(event)
                MotionEvent.ACTION_MOVE -> handleStylusMove(event)
                MotionEvent.ACTION_UP -> handleStylusUp(event)
                else -> false
            }
        }

        // Handle connection status changes
        webSocketClient.setConnectionCallback(object : ConnectionCallback {
            override fun onConnected() {
                handleConnection()
            }
```

```
            override fun onDisconnected() {
                handleDisconnection()
            }

            override fun onError(error: String) {
                handleConnectionError(error)
            }
        })
    }

    private fun handleStylusDown(event: MotionEvent): Boolean {
        if (!isConnected) return false

        val inputData = InputData(
            x = normalizeX(event.x),
            y = normalizeY(event.y),
            pressure = event.pressure,
            tiltX = event.getAxisValue(MotionEvent.AXIS_TILT_X),
            tiltY = event.getAxisValue(MotionEvent.AXIS_TILT_Y),
            timestamp = System.currentTimeMillis()
        )

        // Send input data to Windows component
        webSocketClient.sendInputData(inputData)

        // Update local visualization if in direct mode
        if (currentMode == TabletMode.DIRECT) {
            drawingView.startStroke(event.x, event.y, event.pressure)
        }

        return true
    }

    private fun handleStylusMove(event: MotionEvent): Boolean {
        if (!isConnected) return false

        // Send movement data with pressure and tilt information
        val inputData = InputData(
            x = normalizeX(event.x),
            y = normalizeY(event.y),
            pressure = event.pressure,
            tiltX = event.getAxisValue(MotionEvent.AXIS_TILT_X),
            tiltY = event.getAxisValue(MotionEvent.AXIS_TILT_Y),
            timestamp = System.currentTimeMillis()
        )
```

```
            webSocketClient.sendInputData(inputData)

            // Update local visualization
            if (currentMode == TabletMode.DIRECT) {
                drawingView.updateStroke(event.x, event.y, event.pressure)
            }

            return true
        }

        // ... More Android implementation continues ...
    }

    /*
     * WINDOWS COMPONENT
     * This section contains the complete implementation of the Windows application
     * that receives tablet input and emulates a graphics tablet
     */

    // Program.cs - Main entry point for the Windows application

    using System;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace GraphicsTabletEmulator {
        public class Program {
            private static WebSocketServer webSocketServer;
            private static TabletEmulator tabletEmulator;
            private static SettingsManager settingsManager;
            private static ImageTransmitter imageTransmitter;

            [STAThread]
            static void Main() {
                Application.EnableVisualStyles();
                Application.SetCompatibleTextRenderingDefault(false);

                // Initialize core components
                InitializeComponents();

                // Start services
                StartServices();

                // Run the application
                Application.Run(new MainForm());
```

```
}

private static void InitializeComponents() {
    // Initialize settings manager
    settingsManager = new SettingsManager();
    settingsManager.LoadSettings();

    // Initialize tablet emulator
    tabletEmulator = new TabletEmulator(new TabletConfig {
        Width = settingsManager.TabletWidth,
        Height = settingsManager.TabletHeight,
        PressureLevels = settingsManager.PressureLevels,
        TiltSupport = settingsManager.TiltSupport
    });

    // Initialize WebSocket server
    webSocketServer = new WebSocketServer(
        port: settingsManager.ServerPort,
        maxConnections: settingsManager.MaxConnections
    );

    // Initialize image transmitter if needed
    if (settingsManager.EnableImageTransfer) {
        imageTransmitter = new ImageTransmitter(
            captureInterval: settingsManager.CaptureInterval,
            quality: settingsManager.ImageQuality
        );
    }
}

private static async Task StartServices() {
    try {
        // Start WebSocket server
        await webSocketServer.Start();

        // Start image transmission if enabled
        if (settingsManager.EnableImageTransfer) {
            await imageTransmitter.Start();
        }

        // Initialize tablet emulation
        await tabletEmulator.Initialize();

    } catch (Exception ex) {
        MessageBox.Show(
            $"Error starting services: {ex.Message}",
```

```
                    "Error",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error
                );
                Application.Exit();
            }
        }
    }
}
```

DrawingView.kt

```
package com.graphicstablet.android.views

import android.content.Context
import android.graphics.*
import android.view.View
import androidx.core.graphics.scale
import kotlin.math.max
import kotlin.math.min

class DrawingView(context: Context) : View(context) {
    // Drawing tools and state management
    private val paint = Paint().apply {
        isAntiAlias = true
        strokeJoin = Paint.Join.ROUND
        strokeCap = Paint.Cap.ROUND
        style = Paint.Style.STROKE
    }

    private val path = Path()
    private var currentStroke: StrokeData? = null
    private var backgroundBitmap: Bitmap? = null
    private val strokeHistory = mutableListOf<StrokeData>()

    // Configuration parameters
    private var minStrokeWidth = 1f
    private var maxStrokeWidth = 10f
    private var pressureSensitivity = 1f

    // Cache for performance optimization
    private val drawingCache = HashMap<Long, Path>()

    init {
        // Set up initial paint properties
        paint.color = Color.BLACK
        paint.strokeWidth = minStrokeWidth
    }
```

```
override fun onSizeChanged(w: Int, h: Int, oldw: Int, oldh: Int) {
    super.onSizeChanged(w, h, oldw, oldh)
    // Adjust background bitmap if needed
    backgroundBitmap?.let { bitmap ->
        backgroundBitmap = bitmap.scale(w, h, true)
    }
}

override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)

    // Draw background image if available
    backgroundBitmap?.let { bitmap ->
        canvas.drawBitmap(bitmap, 0f, 0f, null)
    }

    // Draw all completed strokes
    strokeHistory.forEach { stroke ->
        drawStroke(canvas, stroke)
    }

    // Draw current stroke if active
    currentStroke?.let { stroke ->
        drawStroke(canvas, stroke)
    }
}

private fun drawStroke(canvas: Canvas, stroke: StrokeData) {
    // Get cached path or create new one
    val path = drawingCache.getOrPut(stroke.id) {
        createPathFromStroke(stroke)
    }

    paint.strokeWidth = calculateStrokeWidth(stroke.pressure)
    canvas.drawPath(path, paint)
}

private fun createPathFromStroke(stroke: StrokeData): Path {
    return Path().apply {
        moveTo(stroke.points.first().x, stroke.points.first().y)

        // Use quadratic bezier curves for smooth lines
        for (i in 1 until stroke.points.size - 1) {
            val p1 = stroke.points[i]
            val p2 = stroke.points[i + 1]
```

```
                val cx = (p1.x + p2.x) / 2
                val cy = (p1.y + p2.y) / 2

                quadTo(p1.x, p1.y, cx, cy)
            }

            // Add final point
            if (stroke.points.size > 1) {
                val lastPoint = stroke.points.last()
                lineTo(lastPoint.x, lastPoint.y)
            }
        }
    }

    private fun calculateStrokeWidth(pressure: Float): Float {
        return minStrokeWidth + (maxStrokeWidth - minStrokeWidth) *
               pressure.pow(pressureSensitivity)
    }
}

// WebSocketClient.kt - Handles real-time communication with Windows component

package com.graphicstablet.android.network

import com.google.gson.Gson
import okhttp3.*
import java.util.concurrent.TimeUnit
import kotlinx.coroutines.channels.Channel
import kotlinx.coroutines.flow.MutableStateFlow

class WebSocketClient(
    private val serverAddress: String,
    private val port: Int,
    private val connectionTimeout: Long = 10,
    private val readTimeout: Long = 30,
    private val writeTimeout: Long = 30
) {
    private var webSocket: WebSocket? = null
    private val gson = Gson()
    private val messageChannel = Channel<String>(Channel.BUFFERED)
    private val connectionState =
MutableStateFlow<ConnectionState>(ConnectionState.Disconnected)

    private val client = OkHttpClient.Builder()
        .connectTimeout(connectionTimeout, TimeUnit.SECONDS)
```

```
            .readTimeout(readTimeout, TimeUnit.SECONDS)
            .writeTimeout(writeTimeout, TimeUnit.SECONDS)
            .build()

    init {
        setupWebSocket()
    }

    private fun setupWebSocket() {
        val request = Request.Builder()
            .url("ws://$serverAddress:$port")
            .build()

        webSocket = client.newWebSocket(request, createWebSocketListener())
    }

    private fun createWebSocketListener(): WebSocketListener {
        return object : WebSocketListener() {
            override fun onOpen(webSocket: WebSocket, response: Response) {
                connectionState.value = ConnectionState.Connected
                // Send initial configuration
                sendConfiguration()
            }

            override fun onMessage(webSocket: WebSocket, text: String) {
                handleServerMessage(text)
            }

            override fun onClosing(webSocket: WebSocket, code: Int, reason:
String) {
                webSocket.close(1000, null)
                connectionState.value = ConnectionState.Disconnected
            }

            override fun onFailure(webSocket: WebSocket, t: Throwable, response:
Response?) {
                connectionState.value = ConnectionState.Error(t.message ?:
"Unknown error")
                // Attempt to reconnect
                scheduleReconnect()
            }
        }
    }

    fun sendInputData(inputData: InputData) {
        val json = gson.toJson(InputMessage(type = "input", data = inputData))
```

```
        webSocket?.send(json)
    }

    private fun handleServerMessage(message: String) {
        try {
            val serverMessage = gson.fromJson(message,
ServerMessage::class.java)
            when (serverMessage.type) {
                "config" -> handleConfigMessage(serverMessage.data)
                "image" -> handleImageMessage(serverMessage.data)
                "error" -> handleErrorMessage(serverMessage.data)
                else -> Log.w(TAG, "Unknown message type:
${serverMessage.type}")
            }
        } catch (e: Exception) {
            Log.e(TAG, "Error parsing server message", e)
        }
    }
}
```

TabletEmulator.cs

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

namespace GraphicsTabletEmulator {
    public class TabletEmulator : IDisposable {
        private readonly ITabletDriver driver;
        private bool isInitialized;
        private readonly TabletConfig config;

        // Native Windows API imports for tablet device emulation
        [DllImport("user32.dll")]
        private static extern void mouse_event(uint dwFlags, uint dx, uint dy,
uint dwData, int dwExtraInfo);

        public TabletEmulator(TabletConfig config) {
            this.config = config;
            this.driver = new WindowsInkDriver();
        }

        public async Task Initialize() {
            if (isInitialized) return;

            try {
                await driver.Initialize(config);
                isInitialized = true;
```

```
            } catch (Exception ex) {
                throw new TabletEmulatorException("Failed to initialize tablet
driver", ex);
            }
        }

        public void ProcessInput(InputData input) {
            if (!isInitialized) throw new TabletEmulatorException("Tablet
emulator not initialized");

            // Convert input coordinates to tablet coordinates
            var (x, y) = ConvertCoordinates(input.X, input.Y);

            // Create tablet report
            var report = new TabletReport {
                X = x,
                Y = y,
                Pressure = ConvertPressure(input.Pressure),
                TiltX = input.TiltX,
                TiltY = input.TiltY,
                Timestamp = input.Timestamp
            };

            // Send report to the system
            driver.SendReport(report);
        }

        private (int x, int y) ConvertCoordinates(float normalizedX, float
normalizedY) {
            return (
                (int)(normalizedX * config.Width),
                (int)(normalizedY * config.Height)
            );
        }

        private int ConvertPressure(float normalizedPressure) {
            return (int)(normalizedPressure * config.PressureLevels);
        }

        public void Dispose() {
            driver?.Dispose();
            isInitialized = false;
        }
    }
}
```

```
// ImageTransmitter.cs - Screen capture and transmission

using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Threading.Tasks;

public class ImageTransmitter : IDisposable {
    private readonly ScreenCapture screenCapture;
    private readonly ImageCompressor imageCompressor;
    private readonly IWebSocketServer webSocketServer;
    private bool isRunning;
    private readonly ImageTransmitterConfig config;

    public ImageTransmitter(
        IWebSocketServer webSocketServer,
        ImageTransmitterConfig config
    ) {
        this.webSocketServer = webSocketServer;
        this.config = config;
        this.screenCapture = new ScreenCapture();
        this.imageCompressor = new ImageCompressor(config.Quality);
    }

    public async Task Start() {
        if (isRunning) return;

        isRunning = true;
        await Task.Run(TransmissionLoop);
    }

    private async Task TransmissionLoop() {
        while (isRunning) {
            try {
                // Capture screen
                using var screenshot = screenCapture.CaptureScreen();

                // Compress image
                var compressedImage = imageCompressor.CompressImage(screenshot);

                // Send to connected clients
                await webSocketServer.BroadcastImage(compressedImage);

                // Wait for next capture
                await Task.Delay(config.CaptureInterval);
```

```
            } catch (Exception ex) {
                OnError?.Invoke(this, new ErrorEventArgs(ex));
            }
        }
    }

    private class ScreenCapture {
        public Bitmap CaptureScreen() {
            var screen = Screen.PrimaryScreen;
            var bounds = screen.Bounds;

            var bitmap = new Bitmap(bounds.Width, bounds.Height);
            using (var graphics = Graphics.FromImage(bitmap)) {
                graphics.CopyFromScreen(
                    bounds.X, bounds.Y,
                    0, 0,
                    bounds.Size
                );
            }

            return bitmap;
        }
    }

    public void Dispose() {
        isRunning = false;
        screenCapture?.Dispose();
        imageCompressor?.Dispose();
    }
}
activity_main.xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
```

```
            android:background="?attr/colorPrimary">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="horizontal"
                android:gravity="center_vertical">

                <TextView
                    android:id="@+id/connection_status"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textColor="@android:color/white"/>

                <Space
                    android:layout_width="0dp"
                    android:layout_height="match_parent"
                    android:layout_weight="1"/>

                <ImageButton
                    android:id="@+id/settings_button"
                    android:layout_width="48dp"
                    android:layout_height="48dp"
                    android:src="@drawable/ic_settings"

android:background="?attr/selectableItemBackgroundBorderless"
                    android:contentDescription="@string/settings"/>
            </LinearLayout>

        </androidx.appcompat.widget.Toolbar>

    </com.google.android.material.appbar.AppBarLayout>

    <com.graphicstablet.android.views.DrawingView
        android:id="@+id/drawing_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"/>

    <com.google.android.material.bottomsheet.BottomSheetBehavior
        android:id="@+id/bottom_sheet"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

app:layout_behavior="com.google.android.material.bottomsheet.BottomSheetBehavior
">
```

```
        <!-- Settings panel content -->
        <include layout="@layout/settings_panel"/>

    </com.google.android.material.bottomsheet.BottomSheetBehavior>

</androidx.coordinatorlayout.widget.CoordinatorLayout>

// SettingsActivity.kt - Settings management UI for Android

class SettingsActivity : AppCompatActivity() {
    private lateinit var settingsManager: SettingsManager
    private lateinit var binding: ActivitySettingsBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivitySettingsBinding.inflate(layoutInflater)
        setContentView(binding.root)

        settingsManager = SettingsManager(this)
        setupSettingsUI()
    }

    private fun setupSettingsUI() {
        // Connection settings
        binding.serverAddress.setText(settingsManager.getServerAddress())
        binding.serverPort.setText(settingsManager.getServerPort().toString())

        // Pressure sensitivity settings
        binding.pressureSensitivitySlider.value =
            settingsManager.getPressureSensitivity()
        binding.pressureSensitivitySlider.addOnChangeListener { _, value, _ ->
            settingsManager.setPressureSensitivity(value)
            updatePressureCurvePreview()
        }

        // Display mode settings
        binding.displayModeGroup.check(
            when (settingsManager.getDisplayMode()) {
                DisplayMode.DIRECT -> R.id.mode_direct
                DisplayMode.IMAGE_SYNC -> R.id.mode_image_sync
            }
        )

        // Additional settings setup...
    }
```

```
}

// Windows UI Implementation
// MainForm.cs

public partial class MainForm : Form {
    private readonly TabletEmulator tabletEmulator;
    private readonly SettingsManager settingsManager;
    private readonly IWebSocketServer webSocketServer;
    private readonly NotifyIcon trayIcon;

    public MainForm(
        TabletEmulator tabletEmulator,
        SettingsManager settingsManager,
        IWebSocketServer webSocketServer
    ) {
        InitializeComponent();

        this.tabletEmulator = tabletEmulator;
        this.settingsManager = settingsManager;
        this.webSocketServer = webSocketServer;

        // Initialize system tray icon
        trayIcon = new NotifyIcon {
            Icon = Resources.AppIcon,
            Visible = true,
            ContextMenuStrip = CreateTrayMenu()
        };

        // Setup UI event handlers
        SetupEventHandlers();

        // Initialize status display
        UpdateStatusDisplay();
    }

    private ContextMenuStrip CreateTrayMenu() {
        var menu = new ContextMenuStrip();
        menu.Items.Add("Show Window", null, ShowWindow_Click);
        menu.Items.Add("Settings", null, Settings_Click);
        menu.Items.Add("-"); // Separator
        menu.Items.Add("Exit", null, Exit_Click);
        return menu;
    }

    private void SetupEventHandlers() {
```

```
        webSocketServer.ClientConnected += (s, e) => {
            Invoke(new Action(() => {
                UpdateStatusDisplay();
                logListBox.Items.Add($"Client connected: {e.ClientAddress}");
            }));
        };

        webSocketServer.ClientDisconnected += (s, e) => {
            Invoke(new Action(() => {
                UpdateStatusDisplay();
                logListBox.Items.Add($"Client disconnected: {e.ClientAddress}");
            }));
        };

        // More event handlers...
    }

    private void UpdateStatusDisplay() {
        var status = webSocketServer.GetStatus();
        statusLabel.Text = $"Connected Clients: {status.ConnectedClients}";
        serverStatusLabel.Text = $"Server Status: {status.ServerStatus}";
    }
}

// Protocol Definitions and Data Structures
// Messages.cs - Shared message definitions

public class InputMessage {
    public string Type { get; set; } = "input";
    public InputData Data { get; set; }
}

public class InputData {
    public float X { get; set; }
    public float Y { get; set; }
    public float Pressure { get; set; }
    public float TiltX { get; set; }
    public float TiltY { get; set; }
    public long Timestamp { get; set; }
}

public class ConfigurationMessage {
    public string Type { get; set; } = "config";
    public TabletConfiguration Data { get; set; }
}
```

```
public class TabletConfiguration {
    public int Width { get; set; }
    public int Height { get; set; }
    public int PressureLevels { get; set; }
    public bool TiltSupport { get; set; }
    public PressureCurve PressureCurve { get; set; }
}
PressureCurveManager.cs
public class PressureCurveManager {
    private readonly List<Vector2> controlPoints;
    private readonly int resolution;
    private float[] lookupTable;

    public PressureCurveManager(int resolution = 1024) {
        this.resolution = resolution;
        this.controlPoints = new List<Vector2> {
            new Vector2(0, 0),    // Start point
            new Vector2(0.5f, 0.5f), // Middle control point
            new Vector2(1, 1)     // End point
        };

        // Initialize the pressure curve lookup table
        UpdateLookupTable();
    }

    private void UpdateLookupTable() {
        lookupTable = new float[resolution];

        for (int i = 0; i < resolution; i++) {
            float t = i / (float)(resolution - 1);
            lookupTable[i] = CalculateBezierPoint(t);
        }
    }

    private float CalculateBezierPoint(float t) {
        // Cubic Bezier curve calculation for smooth pressure response
        float u = 1 - t;
        float tt = t * t;
        float uu = u * u;
        float uuu = uu * u;
        float ttt = tt * t;

        // Calculate point on the curve using the control points
        float result = uuu * controlPoints[0].Y
                     + 3 * uu * t * controlPoints[1].Y
                     + 3 * u * tt * controlPoints[2].Y
```

```
                    + ttt * controlPoints[3].Y;

        return Math.Max(0, Math.Min(1, result)); // Clamp to [0,1]
    }

    public float TransformPressure(float rawPressure) {
        // Convert raw pressure to index in lookup table
        int index = (int)(rawPressure * (resolution - 1));
        index = Math.Max(0, Math.Min(resolution - 1, index));

        return lookupTable[index];
    }
}

// ScreenRegionSelector.cs - Screen region selection for tablet mapping

public class ScreenRegionSelector : Form {
    private Rectangle selectedRegion;
    private bool isSelecting;
    private Point selectionStart;

    public ScreenRegionSelector() {
        // Create a transparent, full-screen form
        this.FormBorderStyle = FormBorderStyle.None;
        this.WindowState = FormWindowState.Maximized;
        this.TopMost = true;
        this.TransparencyKey = this.BackColor;

        // Initialize drawing tools
        InitializeGraphics();

        // Set up event handlers for mouse interaction
        this.MouseDown += OnMouseDown;
        this.MouseMove += OnMouseMove;
        this.MouseUp += OnMouseUp;

        // Add keyboard handler for cancellation
        this.KeyDown += (s, e) => {
            if (e.KeyCode == Keys.Escape) this.DialogResult =
DialogResult.Cancel;
        };
    }

    protected override void OnPaint(PaintEventArgs e) {
        base.OnPaint(e);
```

```
        if (selectedRegion.Width > 0 && selectedRegion.Height > 0) {
            // Draw semi-transparent overlay
            using (var brush = new SolidBrush(Color.FromArgb(128, 0, 0, 0))) {
                // Draw darkened areas outside selection
                e.Graphics.FillRectangle(brush, 0, 0, this.Width,
selectedRegion.Top);
                e.Graphics.FillRectangle(brush, 0, selectedRegion.Bottom,
this.Width,
                    this.Height - selectedRegion.Bottom);
                e.Graphics.FillRectangle(brush, 0, selectedRegion.Top,
                    selectedRegion.Left, selectedRegion.Height);
                e.Graphics.FillRectangle(brush, selectedRegion.Right,
selectedRegion.Top,
                    this.Width - selectedRegion.Right, selectedRegion.Height);
            }

            // Draw selection rectangle
            using (var pen = new Pen(Color.White, 2)) {
                e.Graphics.DrawRectangle(pen, selectedRegion);
            }

            // Draw sizing handles
            DrawSelectionHandles(e.Graphics);
        }
    }
}

// CalibrationManager.cs - Tablet calibration system

public class CalibrationManager {
    private readonly List<CalibrationPoint> calibrationPoints;
    private Matrix transformationMatrix;
    private bool isCalibrated;

    public CalibrationManager() {
        calibrationPoints = new List<CalibrationPoint> {
            new CalibrationPoint { ScreenPosition = new PointF(0.1f, 0.1f) },
            new CalibrationPoint { ScreenPosition = new PointF(0.9f, 0.1f) },
            new CalibrationPoint { ScreenPosition = new PointF(0.9f, 0.9f) },
            new CalibrationPoint { ScreenPosition = new PointF(0.1f, 0.9f) }
        };
    }

    public void AddCalibrationPoint(int index, PointF tabletPoint) {
        if (index < 0 || index >= calibrationPoints.Count)
            throw new ArgumentOutOfRangeException(nameof(index));
```

```
        calibrationPoints[index].TabletPosition = tabletPoint;

        // Check if we have all points and can calculate transformation
        if (calibrationPoints.All(p => p.TabletPosition.HasValue)) {
            CalculateTransformationMatrix();
        }
    }

    private void CalculateTransformationMatrix() {
        // Calculate perspective transformation matrix using calibration points
        var sourcePoints = calibrationPoints.Select(p =>
p.TabletPosition.Value).ToArray();
        var targetPoints = calibrationPoints.Select(p =>
p.ScreenPosition).ToArray();

        transformationMatrix = CalculatePerspectiveTransform(sourcePoints,
targetPoints);
        isCalibrated = true;
    }

    public PointF TransformPoint(PointF tabletPoint) {
        if (!isCalibrated)
            throw new InvalidOperationException("Calibration not completed");

        // Apply transformation matrix to convert tablet coordinates to screen
coordinates
        var result = transformationMatrix.TransformPoint(tabletPoint);
        return result;
    }
}
DiagnosticsManager.cs
public class DiagnosticsManager {
    private readonly CircularBuffer<PerformanceMetric> performanceHistory;
    private readonly ILogger logger;
    private readonly PerformanceCounter latencyCounter;
    private readonly Timer diagnosticTimer;

    public DiagnosticsManager(ILogger logger) {
        this.logger = logger;
        this.performanceHistory = new CircularBuffer<PerformanceMetric>(1000);

        // Initialize performance counters
        latencyCounter = new PerformanceCounter {
            CategoryName = "GraphicsTablet",
            CounterName = "InputLatency",
```

```
            MachineName = ".",
            ReadOnly = false
        };

        // Start periodic diagnostics
        diagnosticTimer = new Timer(CollectDiagnostics, null, 0, 1000);
    }

    private void CollectDiagnostics(object state) {
        try {
            var metric = new PerformanceMetric {
                Timestamp = DateTime.UtcNow,
                CpuUsage = GetCpuUsage(),
                MemoryUsage = GetMemoryUsage(),
                InputLatency = latencyCounter.NextValue(),
                ActiveConnections = GetActiveConnections()
            };

            performanceHistory.Add(metric);

            // Check for performance issues
            AnalyzePerformance(metric);

        } catch (Exception ex) {
            logger.LogError("Error collecting diagnostics", ex);
        }
    }

    private void AnalyzePerformance(PerformanceMetric metric) {
        // Check for concerning latency
        if (metric.InputLatency > 20) { // More than 20ms latency
            logger.LogWarning($"High input latency detected:
{metric.InputLatency}ms");
        }

        // Check for memory issues
        if (metric.MemoryUsage > 0.9) { // More than 90% memory usage
            logger.LogWarning("High memory usage detected");
        }
    }
}
NetworkDiagnostics.cs
public class NetworkDiagnostics {
    private readonly NetworkMetricsCollector metricsCollector;
    private readonly ConnectionOptimizer connectionOptimizer;
    private readonly ILogger logger;
```

```
    public NetworkDiagnostics(ILogger logger) {
        this.logger = logger;
        this.metricsCollector = new NetworkMetricsCollector();
        this.connectionOptimizer = new ConnectionOptimizer();

        // Start continuous network monitoring
        StartMonitoring();
    }

    private void StartMonitoring() {
        Task.Run(async () => {
            while (true) {
                try {
                    var metrics = await metricsCollector.CollectMetrics();
                    AnalyzeNetworkPerformance(metrics);
                    await Task.Delay(1000); // Check every second
                } catch (Exception ex) {
                    logger.LogError("Network monitoring error", ex);
                }
            }
        });
    }

    private void AnalyzeNetworkPerformance(NetworkMetrics metrics) {
        // Check latency
        if (metrics.AverageLatency > 50) { // More than 50ms latency
            logger.LogWarning("High network latency detected");
            connectionOptimizer.OptimizeForLatency();
        }

        // Check packet loss
        if (metrics.PacketLossRate > 0.01) { // More than 1% packet loss
            logger.LogWarning("Significant packet loss detected");
            connectionOptimizer.EnablePacketLossCompensation();
        }

        // Bandwidth optimization
        if (metrics.BandwidthUtilization > 0.8) { // More than 80% bandwidth
usage
            logger.LogWarning("High bandwidth utilization");
            connectionOptimizer.OptimizeBandwidthUsage();
        }
    }
}
```

```
// AutomatedTesting.cs - Automated testing framework for system validation

public class AutomatedTesting {
    private readonly TestSuiteRunner testRunner;
    private readonly TestResultAnalyzer resultAnalyzer;
    private readonly ILogger logger;

    public AutomatedTesting(ILogger logger) {
        this.logger = logger;
        this.testRunner = new TestSuiteRunner();
        this.resultAnalyzer = new TestResultAnalyzer();

        // Register test cases
        RegisterTestCases();
    }

    private void RegisterTestCases() {
        // Input accuracy tests
        testRunner.RegisterTest(new InputAccuracyTest());
        testRunner.RegisterTest(new PressureSensitivityTest());
        testRunner.RegisterTest(new TiltSensitivityTest());

        // Network reliability tests
        testRunner.RegisterTest(new NetworkLatencyTest());
        testRunner.RegisterTest(new ConnectionStabilityTest());

        // Performance tests
        testRunner.RegisterTest(new ResourceUsageTest());
        testRunner.RegisterTest(new RenderingPerformanceTest());
    }

    public async Task<TestResults> RunSystemValidation() {
        logger.LogInfo("Starting system validation");

        var results = await testRunner.RunAllTests();
        var analysis = resultAnalyzer.AnalyzeResults(results);

        // Generate detailed report
        GenerateTestReport(analysis);

        return results;
    }
}

// UpdateManager.cs - System update and maintenance management
```

```
public class UpdateManager {
    private readonly VersionChecker versionChecker;
    private readonly UpdateDownloader downloader;
    private readonly UpdateInstaller installer;
    private readonly ILogger logger;

    public UpdateManager(ILogger logger) {
        this.logger = logger;
        this.versionChecker = new VersionChecker();
        this.downloader = new UpdateDownloader();
        this.installer = new UpdateInstaller();

        // Start periodic update checks
        StartUpdateChecks();
    }

    private async void StartUpdateChecks() {
        while (true) {
            try {
                var updateInfo = await versionChecker.CheckForUpdates();

                if (updateInfo.UpdateAvailable) {
                    await HandleUpdateAvailable(updateInfo);
                }

                await Task.Delay(TimeSpan.FromHours(24)); // Check daily

            } catch (Exception ex) {
                logger.LogError("Update check failed", ex);
            }
        }
    }

    private async Task HandleUpdateAvailable(UpdateInfo updateInfo) {
        // Notify user about update
        NotifyUpdateAvailable(updateInfo);

        if (updateInfo.IsAutoUpdateEnabled) {
            await PerformUpdate(updateInfo);
        }
    }

    private async Task PerformUpdate(UpdateInfo updateInfo) {
        try {
            // Download update
            var updatePackage = await downloader.DownloadUpdate(updateInfo);
```

```
            // Verify package integrity
            if (await VerifyUpdatePackage(updatePackage)) {
                // Install update
                await installer.InstallUpdate(updatePackage);

                logger.LogInfo("Update successfully installed");
            } else {
                logger.LogError("Update package verification failed");
            }

        } catch (Exception ex) {
            logger.LogError("Update installation failed", ex);
        }
    }
}

// CustomizationManager.cs - Advanced UI customization and personalization

public class CustomizationManager {
    private readonly UserPreferences preferences;
    private readonly ThemeManager themeManager;
    private readonly ShortcutManager shortcutManager;
    private readonly ILogger logger;

    public CustomizationManager(ILogger logger) {
        this.logger = logger;
        this.preferences = new UserPreferences();
        this.themeManager = new ThemeManager();
        this.shortcutManager = new ShortcutManager();

        LoadUserCustomizations();
    }

    private void LoadUserCustomizations() {
        // Load and apply user preferences
        var savedPreferences = preferences.Load();
        ApplyCustomizations(savedPreferences);
    }

    private void ApplyCustomizations(UserPreferences prefs) {
        // Apply theme
        themeManager.ApplyTheme(prefs.Theme);

        // Configure shortcuts
        shortcutManager.ConfigureShortcuts(prefs.Shortcuts);
```

```
            // Apply other customizations
            ApplyLayoutCustomizations(prefs.Layout);
            ApplyBehaviorCustomizations(prefs.Behavior);
        }
    }
```