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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“ Програмне забезпечення системи збирання та обробки електрофізіологічних сигналів ”

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

Керівник проекту
доктор технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"*Інформаційні технології*"*_____
Спеціальність *123 "*Комп'ютерна інженерія*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

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

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

Селіванову Тарасу В'ячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи збирання та обробки електрофізіологічних сигналів*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи збирання та обробки електрофізіологічних сигналів*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Селіванов Т. В.
(прізвище та ініціали)

АНОТАЦІЯ

Селіванов Т.В. Програмне забезпечення системи збирання та обробки електрофізіологічних сигналів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи збирання та обробки електрофізіологічних сигналів.

Метою розробки є програмне забезпечення системи збирання та обробки електрофізіологічних сигналів.

Результат роботи – програмна реалізація системи збирання та обробки електрофізіологічних сигналів.

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

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

Програма може використовуватися на на платформах з IOS та Android.

Програму розроблено в середовищі C# із використанням SQLite, та інших додаткових бібліотек.

Ключові слова: комп'ютерна інженерія, сканування, збирання електрофізіологічних сигналів, обробка електрофізіологічних сигналів.

ABSTRACT

Selivanov T.V. Software of the system of collection and processing of electrophysiological signals. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software was developed, which is intended for the system of collecting and processing electrophysiological signals.

The purpose of the development is the software system for collecting and processing electrophysiological signals.

The result of the work is the software implementation of the system for collecting and processing electrophysiological signals.

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

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

The program can be used on IOS and Android platforms.

The program was developed in the C# environment using SQLite and other additional libraries.

Keywords: computer engineering, scanning, collection of electrophysiological signals, processing of electrophysiological signals.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕФС	–	електрофізіологічний сигнал
ЛОМ	–	локальна обчислювальна мережа
ПК	–	персональний комп’ютер
ПЗ	–	програмне забезпечення
DNS	–	Domain Name System
http	–	HyperText Transfer Protocol
IP	–	Internet Protocol
FCL	–	Framework Class Library
XML	–	Extensible Markup Language
SDK	–	Software development kit

ВСТУП

Актуальність теми. Системи збирання та обробки електрофізіологічних сигналів (ЕФС) широко використовуються в різних сферах життя людини, таких як медицина, наука, спорт та розваги.

У медицині ЕФС використовуються для діагностики та моніторингу різних станів тіла. Наприклад, електрокардіографія (ЕКГ) використовується для вимірювання електричної активності серця та діагностики різних серцевих захворювань. У науці ЕФС використовуються для дослідження різних фізіологічних процесів в тілі. Наприклад, ЕЕГ використовується для дослідження різних процесів.

У спорті ЕФС використовуються для моніторингу стану спортсменів та покращення їх тренувань. Наприклад, ЕЕГ може використовуватися для діагностики стану спортсменів під час тренувань та конкурсів та для покращення їх стратегій та техніки.

У розвагах ЕФС використовуються для створення інтерактивних ігор та додатків. Наприклад, деякі відеоігри використовують ЕЕГ-сигнали для контролювання дій гравців.

Застосування систем збирання та обробки ЕФС може бути корисним для людей в повсякденному житті, допомагаючи їм продовжувати моніторувати свій стан здоров'я та фізіологічні процеси в їх тілі.

Наприклад, на ринку існує декілька портативних пристроїв, що використовують технологію ЕЕГ для моніторингу рівня стресу та зосередженості в повсякденному житті.

Ці пристрої можуть надавати корисну інформацію для тих, хто працює в стресових умовах або хоче підвищити свою продуктивність.

Проте одним з найактуальніших електрофізіологічних сигналів є біопараметричні дані користувача.

У роботі розглядається використання відбитків пальця авторизації та автентифікації на основі операційної системи Android.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи збирання та обробки електрофізіологічних сигналів.

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

– Огляд існуючих систем збирання та обробки електрофізіологічних сигналів.

– Дослідження системи керування даними електрофізіологічних сигналів.

– Програмна реалізація системи збирання та обробки електрофізіологічних сигналів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі проходження авторизації за допомогою електрофізіологічних сигналів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи збирання та обробки електрофізіологічних сигналів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Системи збирання та обробки електрофізіологічних сигналів (ЕФС) є дуже актуальною темою та вони призначені у багатьох галузях, таких як медицина, наука про спорт, біологія, психологія тощо.

У медицині, ЕФС використовуються для діагностики різних захворювань, таких як епілепсія, хвороба Паркінсона, аритмія серця, інсульт тощо. Збір та аналіз ЕФС може допомогти встановити діагноз та надати ефективну терапію.

У науці про спорт, ЕФС використовуються для вивчення різних фізіологічних параметрів під час фізичної активності, таких як пульс, кисневий дефіцит, кількість спожитої енергії тощо. Це дозволяє вченим більш детально вивчати реакції організму на фізичну навантаження та вдосконалювати методи тренувань.

У біології, ЕФС допомагають вивчати фізіологічні процеси, такі як робота м'язів та нервової системи, що може бути корисним для розуміння різних фізіологічних механізмів.

У психології, ЕФС використовуються для вивчення емоцій та психічних станів людини. Наприклад, вивчення ЕФС може допомогти визначити рівень стресу або депресії у людини.

Отже, розробка систем збирання та обробки ЕФС є дуже важливою та актуальною темою в багатьох галузях, оскільки дозволяє отримувати цінну інформацію про фізіологічні процеси та стани людини.

Окрім того, зростання популярності досліджень, пов'язаних з ЕФС, призвело до розвитку технологій збору, обробки та аналізу ЕФС. Наприклад, розробка датчиків ЕФС, які можуть бути вбудовані в пристрої для носіння, такі

як годинники або фітнес-трекери, дозволяє отримувати дані про фізіологічні процеси в режимі реального часу та збирати їх на протязі тривалого періоду часу.

Розробка програмного забезпечення для збирання та обробки ЕФС дозволяє вченим та медичним працівникам легко збирати та аналізувати дані, що дозволяє встановлювати діагнози та надавати ефективну терапію.

1.2 Область застосування

Система збирання та обробки електрофізіологічних сигналів, включаючи сканер відбитків пальця, має широкий спектр застосувань Такі системи можна використовувати так:

– Безпека та аутентифікація: Сканери відбитків пальця використовуються для ідентифікації особи на основі унікальних рис папіллярних ліній на пальці. Це може бути застосовано для розблокування смартфонів, комп'ютерів, дверей, автомобілів та інших систем, що потребують персоналізованого доступу.

– Медична діагностика: Електрофізіологічні сигнали, такі як електрокардіограма (ЕКГ) та електроенцефалограма (ЕЕГ), можуть бути використані для діагностики різних захворювань серця та мозку. Системи збирання та обробки цих сигналів допомагають лікарям аналізувати патологічні зміни та приймати правильні медичні рішення.

– Дослідження мозку: Для дослідження мозку використовуються системи електроенцефалографії (ЕЕГ), які збирають та аналізують електричні сигнали, що генеруються мозком. Це дозволяє вивчати когнітивні функції, сприйняття, емоції та інші аспекти мозкової діяльності.

– Контроль пристроїв за допомогою мозку: В інтерфейсах людина-машини можна використовувати електроенцефалографічні сигнали для керування різними пристроями, наприклад, прокрутка сторінок на електронних пристроях, керування протезами, віртуальною реальністю та іграми.

– Дослідження сну: Системи збирання електрофізіологічних сигналів можуть бути використані для моніторингу сну та аналізу різних фаз сну. Це допомагає вивчати розлади сну, такі як безсоння, апное у сні та інші проблеми зі сном.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи збирання та обробки електрофізіологічних сигналів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Електрофізіологічні процеси - це процеси, які відбуваються в живих організмах і пов'язані з генерацією, передачею та реагуванням на електричні сигнали.

Одним з прикладів електрофізіологічних процесів є акційний потенціал - електричний сигнал, який виникає в нервових та м'язових клітинах під час їх збудження.

Акційний потенціал генерується за допомогою іонних каналів, які регулюють рух іонів через клітинну мембрану. Зараз існує багато різних систем, технологій, архітектур та програмних рішень для систем збирання та обробки електрофізіологічних сигналів. Ось деякі з них:

– EEG (електроенцефалограма) - це технологія, що дозволяє записувати електричну активність мозку за допомогою електродів, що розміщуються на голові. Ця технологія використовується для діагностики різних патологій мозку, а також для досліджень в галузі нейронауки та психології.

– ECG (електрокардіограма) - це технологія, що дозволяє записувати електричну активність серця за допомогою електродів, що розміщуються на грудях. Ця технологія використовується для діагностики різних серцевих патологій.

– EMG (електроміографія) - це технологія, що дозволяє записувати електричну активність м'язів за допомогою електродів, що розміщуються на поверхні шкіри біля м'язів. Ця технологія використовується для діагностики

різних м'язових патологій, а також для досліджень в галузі кінезіології та спортивної науки.

В першу чергу коли йде мова про електрофізіологічні процеси звертають увагу на модулі MATLAB. Перед проведенням порівняльного аналізу розглянемо основну систему та її можливості.

MATLAB - це інтерактивне середовище для наукових обчислень та програмування, яке спеціально розроблене для обробки матричних операцій. MATLAB використовується в багатьох галузях, включаючи науку, інженерію, фінанси та біомедичну інженерію.

MATLAB має велику кількість вбудованих функцій та бібліотек, що дозволяє легко та швидко виконувати обчислення, включаючи обробку електрофізіологічних сигналів. Наприклад, MATLAB має вбудовані функції для обробки сигналів, такі як фільтрація, декомпозиція, аналіз спектра, регресія та багато інших.

Окрім цього, MATLAB також має інструментарій для роботи з базами даних, статистичного аналізу, оптимізації, розподілених обчислень та моделювання. MATLAB також підтримує різні формати даних, включаючи сигнали, зображення та відео.

Для обробки електрофізіологічних даних, таких як ЕЕГ, ЕКГ та ЕМГ, MATLAB має спеціальний інструментарій, який називається Signal Processing Toolbox. Signal Processing Toolbox містить набір функцій для обробки сигналів, таких як фільтрація, декомпозиція на базисні функції, аналіз спектра та багато інших

У MATLAB також є інтерактивне середовище для розробки, що дозволяє користувачам візуально створювати, відлагоджувати та виконувати свої програми. Це середовище відображає дані та графіки у вигляді матриць та графіків, що спрощує аналіз та інтерпретацію результатів.

MATLAB було створено в 1970-х роках Джеком Літлом, але насправді його розробка почалася ще в 1960-х роках, коли Літл був аспірантом в Массачусетському технологічному інституті (MIT).

Початково MATLAB був розроблений як інтерактивне середовище для обчислень матриць, але згодом було додано багато інших функцій та інструментів, таких як обробка сигналів, статистичний аналіз, оптимізація та моделювання систем.

Перші версії MATLAB були написані на мові Фортран, але згодом було розроблено власну мову програмування MATLAB, яка мала більш високорівневий синтаксис та була більш зручною для роботи з матрицями.

MATLAB став популярним серед науковців та інженерів, які використовували його для розв'язування складних математичних задач, а також для моделювання та симуляції різних систем. Сьогодні MATLAB залишається одним з найпопулярніших інструментів для наукових обчислень та інженерної розробки, і його використовують в багатьох галузях, від автомобілебудування до фінансів.

Базові можливості

MATLAB має дуже багато функцій для наукових обчислень та числового аналізу. Ось кілька базових можливостей MATLAB:

1. Математичні операції: MATLAB має вбудовані функції для виконання математичних операцій, таких як додавання, віднімання, множення та ділення чисел. Ви також можете виконувати складніші операції, такі як обчислення матриць, диференціювання та інтегрування функцій.

2. Візуалізація даних: MATLAB надає потужні інструменти для візуалізації даних. Ви можете побудувати графіки, діаграми, контурні та поверхневі графіки, а також анімації. Це допомагає вам аналізувати дані та візуалізувати результати вашого дослідження.

3. Робота з масивами та матрицями: MATLAB є мовою програмування, спеціально розробленою для роботи з масивами та матрицями. Ви можете

створювати, індексувати, змінювати розміри та виконувати операції на масивах та матрицях. Це робить MATLAB потужним інструментом для числового аналізу та обробки даних.

4. Робота з функціями: MATLAB дозволяє створювати власні функції та сценарії. Ви можете написати функцію для виконання певних операцій та повторно використовувати її у своїх програмах. MATLAB також має велику кількість вбудованих функцій для виконання різних завдань.

5. Робота з файлами: MATLAB дозволяє зчитувати та записувати дані з файлів різних форматів,Інтеграція даних

6. Дані інтегруються з графічною моделлю за допомогою атрибутів (Attributes). Це пари (ім'я, значення), які зіставляють імена вузлів чи ребер певним значенням даних. Значення атрибута можуть приймати будь-який тип (наприклад, текстові рядки, дискретні чи безперервні числа, URL-адреси або списки) або завантажуються зі сховища даних, або генеруються динамічно в сеансі. Графічні браузери дозволяють користувачеві переглядати всі атрибути вибраних вузлів та ребер.

Модулі MATLAB

Модулі, що підключаються, являють собою потужні засоби розширення можливостей для реалізації нових алгоритмів, додаткового мережевого аналізу та біологічної семантики. Ось деякі модулі які я можу відмітити:

1. Core MATLAB: Це основний модуль, який містить основні функції та інструменти для числових обчислень, матричних операцій, векторизації, графіків та роботи з масивами.

2. MATLAB Graphics: Цей модуль містить функції для візуалізації даних, побудови графіків, діаграм, контурних та поверхневих графіків, анімації та інших візуальних ефектів.

3. Symbolic Math Toolbox: Цей модуль надає функції для символьних обчислень, включаючи символьне диференціювання, інтегрування, розв'язання алгебраїчних рівнянь, маніпуляцію символьними виразами та інші операції.

4. Control System Toolbox: Цей модуль містить функції для аналізу та проектування систем керування, включаючи моделювання динамічних систем, аналіз стійкості, проектування регуляторів, оптимальне керування та інші.

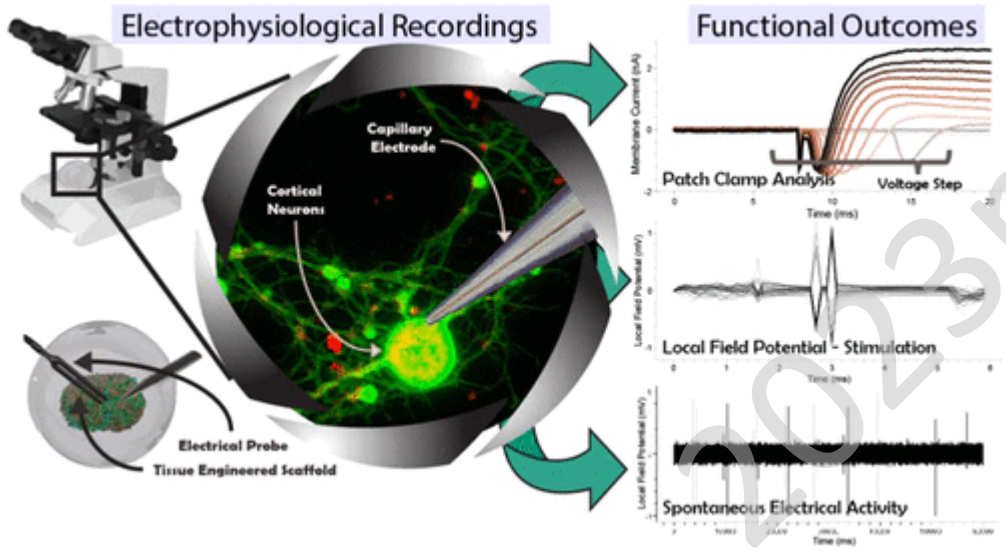

Рисунок 2.1 – Приклад використання модулів MATLAB для отримання електрофізіологічних сигналів

Тепер розглянемо практичні приклади, у роботі в першу чергу приділяється увага таким сигналам як розпізнавання відбитків пальців.

Програми розпізнавання відбитків пальців стають все більш популярними для автентифікації та безпеки.

Хоча існує велика кількість доступних програм для розпізнавання відбитків пальців, важливо зазначити, що визначення того, яка з них краща, може залежати від конкретних випадків використання, вимог і контексту, у якому вони реалізуються та обраної операційної системи.

Розглянемо список загальновідомих програм розпізнавання відбитків пальців.

VeriFinger

Розроблена Neurotechnology, VeriFinger є загальновизнаною технологією ідентифікації відбитків пальців, відомою своєю точністю та швидкістю. Він

підтримує різні сканери відбитків пальців і може використовуватися в різних програмах.

Рисунок 2.2 – Вікно отримання розпізнавання відбитків пальців ПЗ VeriFinger

MegaMatcher

MegaMatcher, також розроблений Neurotechnology, є комплексним біометричним SDK, який включає розпізнавання відбитків пальців серед своїх функцій. Він пропонує високу точність і масштабованість, що робить його придатним для великомасштабних програм.

Рисунок 2.3 – Отримання відбитків пальців ПЗ MegaMatcher

Fingerprint Recognition System (FRS)

FRS це популярна програма розпізнавання відбитків пальців, розроблена компанією Innovatrics. Він забезпечує швидке та точне зіставлення відбитків пальців і може бути інтегрований у різні системи, зокрема контролю доступу та обліку робочого часу.

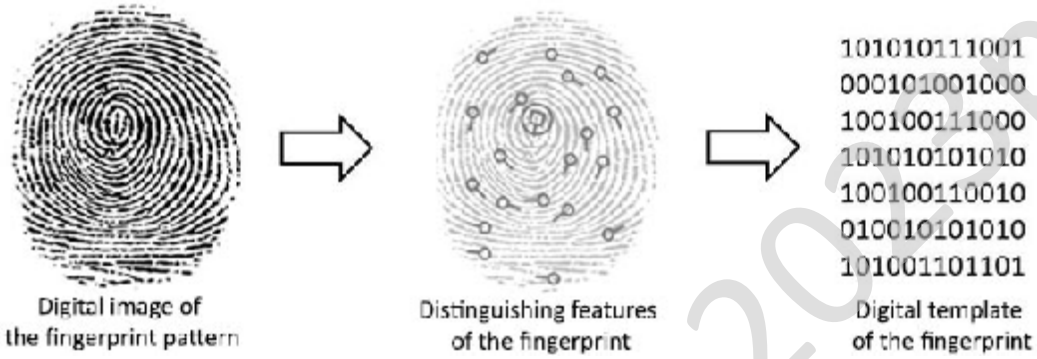

Рисунок 2.4 – Алгоритм роботи ПЗ FRS

BioID

BioID - це хмарна платформа біометричної автентифікації, яка включає розпізнавання відбитків пальців як один із методів автентифікації. Він забезпечує надійний захист і може використовуватися на різних пристроях і платформах.

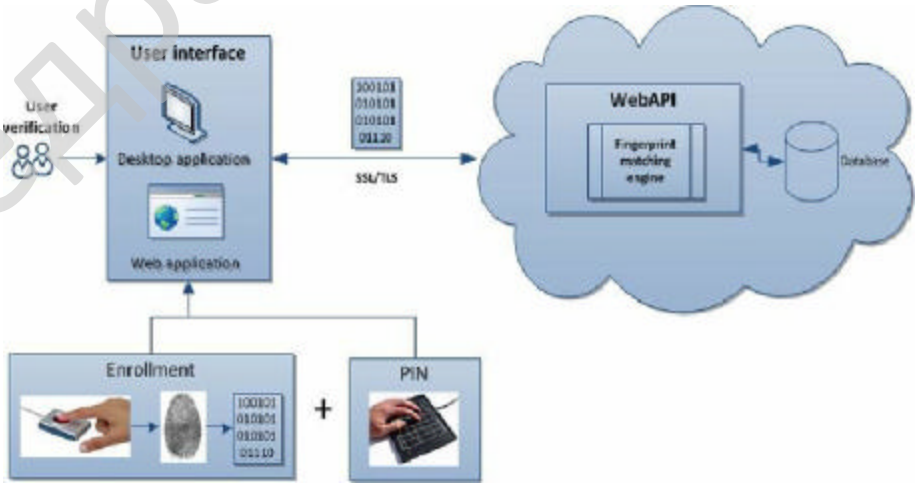

Рисунок 2.5 – Алгоритм роботи системи BioID

Touch N Go

Touch N Go – це програмне забезпечення для розпізнавання відбитків пальців, надане Bayometric. Він підтримує різні сканери відбитків пальців і пропонує зручний інтерфейс для легкої інтеграції в різні програми.

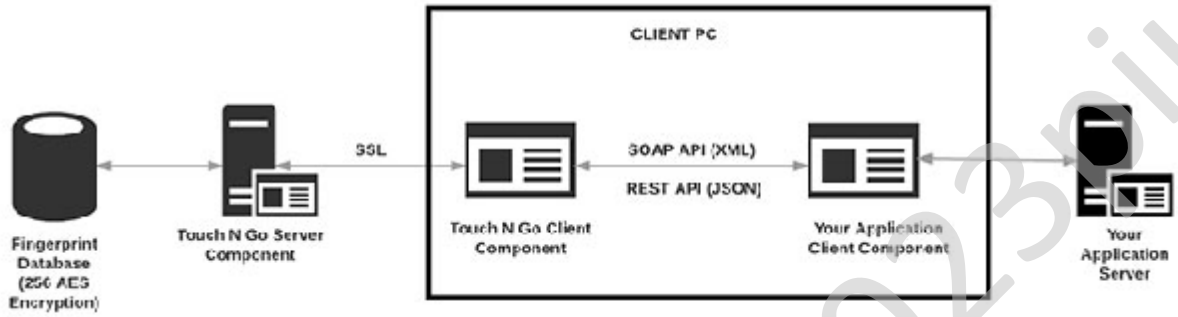

Рисунок 2.6 – Алгоритм роботи системи Touch N Go

DigitalPersona

DigitalPersona – це програмне та апаратне рішення для розпізнавання відбитків пальців, яке пропонує Crossmatch. Він забезпечує безпечну та надійну автентифікацію за допомогою біометрії відбитків пальців і може бути інтегрований у широкий спектр програм.

Рисунок 2.7 – Алгоритм роботи системи DigitalPersona

Висновок

Порівняння «найкращої» чи прямий порівняльний аналіз програми розпізнавання відбитків пальців залежить від таких факторів, як точність, швидкість, масштабованість, легкість інтеграції, сумісність з платформами, вартість і особливі вимоги. Важливо оцінити ці фактори на основі конкретного випадку використання користувачами, щоб визначити, яка програма найбільше підійде для відповідних потреб.

При цьому незважаючи на те, що всі програми, представлені в огляді, виконують ту самі дії, навряд чи можна сказати однозначно, яка краще, а яка – гірше.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою C#. Ця мова обрана виходячи з наступних міркувань. C# – строго типізована об’єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних додатків, виконуваних у середовищі .NET Framework. Мовою C# можна розробляти звичайні клієнтські додатки Windows, веб-служби XML, розподілені компоненти, додатки типу “ сервер-клієнт”, додатки баз даних і багато яких інших. В Visual C# 2008 є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку додатків мовою C# версії 3.0 і .NET Framework версії 3.5.

Синтаксис C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C# або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови C#. Розроблювачі, що знають кожну із цих мов, як правило, зможуть домогтися ефективної роботи з мовою C# за дуже короткий час. Синтаксис C# робить простіше те, що було складно в C#, і забезпечує потужні можливості, такі як типи значень Nullable, перерахування, делегати,

лямбда-вираження й прямий доступ до пам'яті, чого немає в Java. C# підтримує універсальні методи й типи, забезпечуючи більше високий рівень безпеки й продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власне поводження ітерації, що може легко використовуватися в клієнтському коді. В C# 3.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об’єктно-орієнтована мова, C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод Main – крапку входу додатка – інкапсуюється у визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення. У мові C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримуюче спадкування.

На додаток до основних описаних об’єктно-орієнтованих принципів, мова C# спрощує розробку компонентів програмного забезпечення завдяки декільком інноваційним конструкціям мови, у число яких входять наступні:

– Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.

– Властивості, що виступають у ролі методів доступу для закритих змінних-членів.

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

– LINQ (Language-Integrated Query), що пропонує вбудовані можливості запитів у різних джерелах даних.

Якщо буде потрібно забезпечити взаємодію з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL Win32, у мові C# можна використовувати процес, що називається "Interop". Процес

Interop дозволяє програмам на C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C#. Мова C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови C# у порівнянні з C і C# простий і є більше гнучким, чим в Java. Немає окремих файлів заголовка, а методи й типи не потрібно повідомляти в певному порядку. У вихідному файлі C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою C# виконується в середовищі .NET Framework – інтегрованому компоненті Windows, що містить віртуальну систему виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR являє собою комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, що є міжнародним стандартом, який лежить в основі створення середовищ виконання й розробки, у яких забезпечується тісна взаємодія між мовами й бібліотеками.

Вихідний код, написаний мовою C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску у файлі, що виконується, називаному складанням, з розширенням EXE або DLL у більшості випадків. Складання містить маніфест із відомостями про типи складання, версії, мови й регіональні параметри та вимоги безпеки.

При виконанні програми на C# складання завантажується в середовище CLR залежно від відомостей у маніфесті. Далі, якщо вимоги безпеки дотримані, середовище CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середовище CLR також надає інші служби, що відносяться до автоматичного збору сміття, обробки виключень і керуванню ресурсами. Код, виконуваний середовищем CLR, іноді називають "керованим кодом" у протиставлення "некерованому коду", що компілюється в машинний код,

призначень для певної системи. Далі показані відносини під час компіляції й час виконання між файлами з вихідним кодом C#, бібліотеками класів .NET Framework, складаннями й середовищем CLR.

Взаємодія між мовами є ключовою особливістю .NET Framework. Оскільки код IL, створюваний компілятором C# відповідає специфікації CTS, код IL на основі C# може взаємодіяти з кодом, створюваним версіями мов Visual Basic, Visual C#, Visual J# платформи .NET Framework і ще більш ніж 20 CTS-сумісних мов. В одному складанні може бути кілька модулів, написаних на різних мовах платформи .NET Framework, і типи можуть посилатися один на одного, як якби вони були написані на одній мові.

Крім служб часу виконання, в .NET Framework також є велика бібліотека, що складається з більш ніж 4000 класів, організованих по просторах імен, які забезпечують різноманітні корисні функції для будь-яких дій, починаючи від введення й виведення файлів для керуванням рядками для розбивки XML, і закінчуючи елементами керування Windows Forms. У звичайному додатку мовою C# бібліотека класів .NET Framework інтенсивно використовується для "устрою" коду.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи збирання та обробки електрофізіологічних сигналів.

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

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

б) вибрати та обґрунтувати методику побудови системи контролю роботи

технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розглянемо обрані програмні рішення

Щоб розробити програму розпізнавання відбитків пальців для операційної системи Android, потрібно буде використовувати комбінацію бібліотек, модулів і фреймворків. Ось детальний огляд основних компонентів:

1. Android SDK (набір програмного забезпечення)

Android SDK надає необхідні інструменти, бібліотеки та API для розробки програм Android. Він включає Android Debug Bridge (ADB) для зв'язку з пристроєм і налаштування.

2. Fingerprint API

Android Fingerprint API дозволяє розробникам інтегрувати розпізнавання відбитків пальців у свої програми.

Він забезпечує стандартизований інтерфейс для доступу до апаратного забезпечення відбитків пальців на підтримуваних пристроях Android.

3. BiometricPrompt API.

API BiometricPrompt, представлений в Android 9 (рівень API 28) і вище, є вдосконаленою версією API відбитків пальців. Він пропонує більш безпечний і послідовний спосіб автентифікації користувачів за допомогою біометричних факторів, таких як відбитки пальців.

BiometricPrompt також підтримує додаткові біометричні методи, такі як розпізнавання обличчя або сканування райдужної оболонки ока, якщо вони доступні на пристрої.

4. AndroidX Libraries

AndroidX – це набір бібліотек і інструментів, наданих Google для підтримки розробки сучасних програм Android. Бібліотека androidx.biometric, яка

є частиною AndroidX, пропонує зворотну сумісність для API BiometricPrompt на старіших пристроях.

5. OpenCV

OpenCV (Бібліотека комп'ютерного бачення з відкритим кодом) - це потужна бібліотека комп'ютерного зору та обробки зображень із відкритим кодом. Він надає різноманітні алгоритми та функції, які можна використовувати для обробки зображень відбитків пальців, виділення ознак і зіставлення.

6. Third-Party Fingerprint SDKs

Для всебічного використання різного АПІ необхідно розглянути можливість використання сторонніх SDK для відбитків пальців, які надають додаткові функції та легку інтеграцію. Ці SDK часто містять готові компоненти, API та утиліти для реєстрації відбитків пальців, зіставлення та перевірки.

7. Android Studio чи Visual Studio

Android Studio – це офіційне інтегроване середовище розробки (IDE) для розробки програм для Android. Він забезпечує зручний інтерфейс для кодування, налагодження та тестування програм Android. Я використовував Visual Studio тому що було використано мову C#/

8. Gradle Build System

Gradle – це система збірки, яка використовується в Android Studio для керування залежностями, компіляції коду та створення файлу APK (пакет Android). Необхідно налаштувати файл збірки Gradle, щоб включити необхідні залежності та бібліотеки.

9. Додаткові інструменти та бібліотеки.

Залежно від конкретних вимог можуть знадобитися додаткові інструменти та бібліотеки для обробки зображень, виділення функцій і алгоритмів зіставлення.

Приклади включають бібліотеки покращення зображення, інфраструктури машинного навчання (наприклад, TensorFlow або PyTorch) і бібліотеки статистичного аналізу.

Важливо зазначити, що конкретні бібліотеки та інструменти, які було обрано, можуть відрізнятися залежно від вимог сборки проекту, рівня контролю, який потрібен на процесом розпізнавання відбитків пальців, а також цільових версій Android і пристроїв. Крім того було використано офіційну документацією Android, IT форуми та посібники.

При створенні програми сканера відбитку пальця в середовищі C#, я враховував кілька проектних рішень, щоб забезпечити ефективність, точність та безпеку процесу сканування. Ось опис і обґрунтування кількох ключових проектних рішень:

Вибір бібліотеки або фреймворку для сканування відбитку пальця

Для реалізації сканування відбитку пальця в C# можна використовувати різні бібліотеки або фреймворки, такі як Neurotechnology, Futronic, Android V7 app compact. При виборі бібліотеки слід враховувати такі критерії:

– Функціональні можливості: Перевірте, чи пропонує бібліотека необхідні функції для сканування, обробки та порівняння відбитків пальців.

– Сумісність: Переконайтеся, що бібліотека підтримує роботу з C# та вашим обладнанням сканування відбитку пальця.

– Надійність та підтримка: Оцініть, наскільки надійна та підтримувана бібліотека, звертаючи увагу на відгуки користувачів та наявність документації.

Інтерфейс користувача

Розробка зручного та простого інтерфейсу користувача є важливим аспектом проекту. Для створення інтерфейсу користувача в середовищі C# я використовував Windows Forms або WPF (Windows Presentation Foundation) яка забезпечує більшу гнучкість і можливості стилізації, але вимагає деякого навчання. Windows Forms є простішим у використанні, але має менше функціональних можливостей.

Зберігання та обробка відбитків пальця

Під час сканування відбитків пальця необхідно зберігати та обробляти отримані дані. Один з варіантів - зберігання відбитків у вигляді зображень або

бітових даних у базі даних або на файловій системі. Для зберігання в базі даних я використовував SQL-сервер, такий як SQLite.

Алгоритми обробки відбитків пальця

При обробці відбитків пальця необхідно я використовував спеціалізовані алгоритми для виявлення особливостей, екстракції характеристик та порівняння відбитків. Наприклад, я використовував алгоритми, такі як алгоритм кореляції, алгоритм видалення шуму, алгоритм оцінки якості відбитку тощо.

Заходи безпеки

У розробленій програмі сканування відбитку пальця я враховував заходи безпеки для захисту від несанкціонованого доступу до відбитків та конфіденційності даних користувачів. Використовував алгоритм шифрування для збереження та передачі відбитків пальця, а також використовувати механізми аутентифікації та авторизації для контролю доступу до функцій програми.

Головне вікно складається із таких блоків як:

– Кнопка виклику сканера;

– Кнопка виходу з программи;

– Панель сканування містить компоненти:

– Вікно виклику сканера;

– Перегляд та збереженння відсканованого відбитку;

– Вихід на головне меню.

Етап виведення інтерфейсу користувача є важливим кроком у розробці системи імітаційного моделювання поширення інформаційних вірусів у соціальних мережах, оскільки він визначає спосіб візуалізації програми та створення головного вікна, яке слугує основним інтерфейсом взаємодії з користувачем.

На цьому етапі розробник створює головне вікно програми, яке містить елементи інтерфейсу, такі як кнопки, панелі, меню та інші елементи управління.

Головне вікно дозволяє користувачеві взаємодіяти з системою, налаштовувати параметри симуляції, переглядати результати та керувати процесом моделювання.

При проектуванні головного вікна важливо враховувати принципи зручності, ефективності та естетичного оформлення. Головне вікно повинно бути інтуїтивно зрозумілим для користувача, забезпечувати зручний доступ до основних функцій системи та мати привабливий зовнішній вигляд.

Після створення головного вікна, розробник може продовжувати роботу над іншими компонентами інтерфейсу, додавати додаткові функції та покращувати зовнішній вигляд програми. Етап виведення інтерфейсу користувача є початковою точкою для подальшої роботи з інтерфейсом, його налаштування та удосконалення з метою забезпечення зручності та задоволення користувачів під час взаємодії з системою імітаційного моделювання.

При натисканні на кнопку виклику з’являється вікно сканування, при прикладанні пальця до сканера відбувається сканування відбитку, та його подальше збереження в системі.

При завершенні сканування програма сповіщаєш що сканування та збереження пройшло успішно, та показує меню з відсканованими відбитками.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема розробленого програмного забезпечення. Розглянемо її структурні блоки.

1. Модуль інтерфейсу користувача:

– Взаємодія з користувачем для отримання вхідних даних і відображення результатів.

– Представлення інформації про процес сканування та результати ідентифікації або аутентифікації.

– Графічний інтерфейс.

2. Модуль ідентифікації і аутентифікації:

– Зв'язок з модулем захоплення відбитку пальця для отримання вхідного зображення.

– Процес аутентифікації.

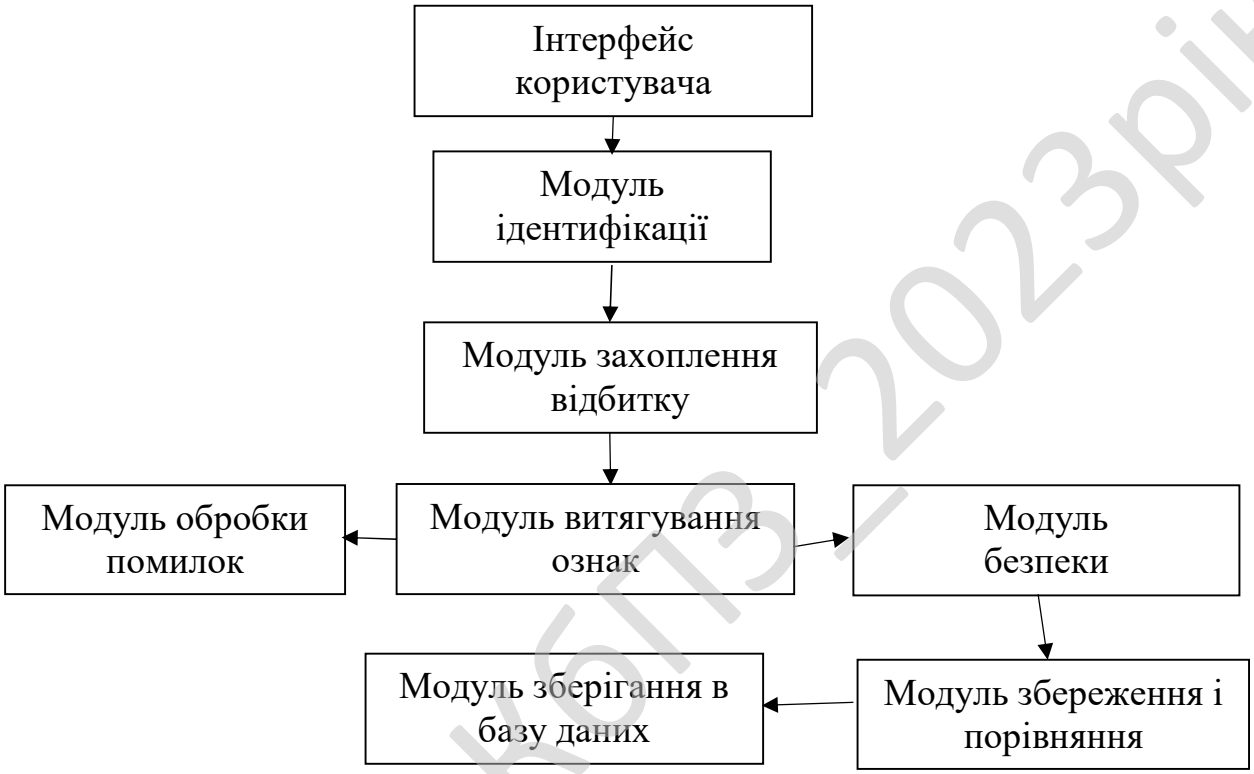

Рисунок 3.1 – Структурна схема розробленої системи

3. Модуль захоплення відбитку пальця:

– Зчитувач або сенсор для захоплення відбитку пальця.

– Алгоритми обробки і попередньої обробки зображень пальця.

– Фільтрація шуму і покращення якості зображення.

– Калібрування.

– Розміщення пальця.

4. Модуль витягування ознак:

– Виділення основних ознак.

– Нормалізації ознак.

5. Модуль обробки помилок:

– Виявлення помилок.

– Обробка помилок.

– Відновлення та обробка винятків.

6. Модуль безпеки:

– Захист даних.

– Керування доступом.

7. Модуль збереження та порівняння

– Алгоритм порівняння.

– Управління збереження.

8. Модуль збереження в базу даних:

– База даних.

– Управління базою даних.

Модуль інтерфейсу користувача відповідає за взаємодію з користувачем та надання зручного та зрозумілого інтерфейсу для введення даних, відображення результатів і керування програмою сканера відбитку пальця. Основні компоненти модуля інтерфейсу користувача можуть включати:

1. Взаємодія з користувачем:

– Надання інструкцій та довідкової інформації щодо користування сканером відбитку пальця.

– Обробка введених команд, які виконуються програмою, наприклад, пошук або оновлення записів у базі даних.

2. Відображення результатів:

– Представлення результатів сканування відбитку пальця, включаючи зображення самого відбитку та візуалізацію витягнутих ознак.

– Відображення інформації про процес сканування.

3. Графічний інтерфейс:

– Розробка інтуїтивно зрозумілого та зручного графічного інтерфейсу (GUI) з використанням кнопок, меню, полів введення, списків.

Модуль ідентифікації і аутентифікації відповідає за використання результатів порівняння відбитків пальців для ідентифікації особи або підтвердження її автентичності.

Основні компоненти модуля ідентифікації і аутентифікації можуть включати:

1. Зв'язок з модулем захоплення відбитку пальця для отримання вхідного зображення: У цьому процесі вхідний відбиток пальця порівнюється зі всіма збереженими в базі даних відбитками для визначення, чи є збіг. Якщо відбиток пальця збігається з одним зі збережених пальців, то особа ідентифікується.

2. Процес аутентифікації: У цьому процесі вхідний відбиток пальця порівнюється з одним конкретним збереженим в базі даних пальцем для підтвердження автентичності особи. Якщо відбиток пальця збігається зі збереженим пальцем, то особа підтверджує свою ідентичність. Наприклад, він може використовуватися у системі фізичної безпеки для контролю доступу до приміщень або у системі ідентифікації.

Модуль захоплення відбитку пальця відповідає за фізичний процес отримання зображення пальця за допомогою сенсора або зчитувача.

Цей модуль може включати такі компоненти:

1. Сенсор або зчитувач: Це пристрій, що використовується для безконтактного або контактного сканування відбитків пальців. Сенсори можуть бути оптичними, ємнісними, ультразвуковими або термальними залежно від застосованої технології.

2. Алгоритми обробки та попередньої обробки зображень: Ці алгоритми використовуються для поліпшення якості отриманих зображень пальця. Вони можуть включати функції, такі як шумозаглушення, видалення артефактів,

регулювання контрастності та яскравості зображення для забезпечення чіткості та однорідності пальцевого зразка.

3. Фільтрація шуму: Шум може бути присутнім у зображеннях пальців через різні джерела, такі як освітлення, дотики інших об'єктів та фізіологічні особливості шкіри. Модуль захоплення відбитку пальця може включати алгоритми фільтрації шуму для покращення якості зображення та виділення основних ознак пальця.

4. Калібрування: Цей процес дозволяє налаштувати сенсор. Калібрування: Цей процес дозволяє налаштувати сенсор або зчитувач для оптимального отримання відбитків пальців. Він може включати налаштування параметрів, таких як розширення, чутливість, швидкість сканування та інші, щоб забезпечити високу якість зображень пальців.

5. Розміщення пальця: Цей компонент визначає правильне розташування пальця на сенсорі або в зоні сканування. Він може включати вказівки або маркери для користувача, як тримати або позиціонувати палець для оптимального захоплення відбитку пальця.

Модуль витягування ознак відповідає за аналіз отриманих зображень пальців і виділення унікальних ознак, які можуть бути використані для подальшого порівняння і ідентифікації.

Основні компоненти модуля витягування ознак можуть включати:

1. Виділення основних ознак: У цьому кроці використовуються алгоритми для виділення основних структурних ознак з сегментованих регіонів пальця. Це може включати виявлення і вимірювання довжин, кутів, відстаней між лініями та інших параметрів, які характеризують унікальну конфігурацію пальця.

2. Нормалізація: У цьому кроці основні ознаки пальця нормалізуються або перетворюються на стандартизований формат, щоб забезпечити незалежність від масштабування, розташування та орієнтації.

Модуль обробки помилок і винятків відповідає за виявлення, обробку і коректне управління помилками та винятками, що можуть виникнути під час виконання програми. Його головні компоненти можуть включати:

1. Виявлення помилок: Механізми для виявлення помилок, такі як перевірка вхідних даних, перевірка стану системи та перевірка результатів операцій. Застосування валідації даних, перевірки діапазону, перевірки цілісності та інших методів для виявлення помилок.

2. Обробка помилок: Забезпечення обробки помилок шляхом встановлення механізмів обробки винятків або використання спеціальних функцій для обробки помилок. Виконання відповідних дій після виникнення помилки, таких як відновлення системи, повідомлення про помилку або відміна поточної операції.

3. Відновлення та обробка винятків: Виконання відповідних дій для відновлення системи після виникнення винятку. Здійснення обробки винятків шляхом забезпечення альтернативного шляху виконання, виправлення проблеми або повідомлення користувачу про виняток.

Модуль безпеки відіграє важливу роль у забезпеченні захисту приватності та конфіденційності пальцевих даних, а також у попередженні несанкціонованого доступу до системи. Основні компоненти модуля безпеки можуть включати:

1. Захист даних:

– Криптографічне захищення пальцевих даних під час передачі та зберігання. Це може включати шифрування даних для запобігання несанкціонованому доступу.

– Застосування механізмів хешування для збереження і перевірки цілісності пальцевих даних. Це дозволяє переконатися, що дані не були змінені під час передачі або зберігання.

2. Керування доступом:

– Установлення механізмів контролю доступу до системи сканера відбитку пальця. Це включає ідентифікацію та автентифікацію користувачів, а також надання рівнів доступу до різних функцій та функціональностей системи.

– Встановлення політик безпеки, таких як складність паролів, періодичність їх зміни, блокування неуспішних спроб автентифікації тощо.

Модуль збереження і порівняння відповідає за збереження стандартизованих представлень пальців у базі даних і виконання порівняння між вхідними відбитками і збереженими пальцями. Основні компоненти модуля збереження і порівняння можуть включати:

1. Алгоритми порівняння: У цьому кроці використовуються алгоритми порівняння для визначення ступеня співставлення між вхідним відбитком і збереженими в базі даних. Ці алгоритми можуть використовувати різні методи, такі як відстань, кореляційні коефіцієнти або нейронні мережі, щоб визначити подібність між відбитками.

2. Алгоритм збереження: У цьому році вібувається формування відбитку в бінарний код для подальшого збереження у відповідній графі

Модуль збереження в базу даних відповідає за обробку і коректне додавання відбитку в базу даних з подальшим його використанням у системі, для різних цілей та має такі компоненти:

1. База даних: Цей компонент відповідає за збереження стандартизованих представлень пальців, що отримані з модулю витягування ознак. База даних може бути структурованою системою зберігання даних, яка дозволяє ефективний доступ до пальцевих даних для порівняння.

2. Управління базою даних: Цей компонент забезпечує функціональність для додавання, видалення і оновлення записів пальців у базі даних. Він також може забезпечувати пошук, індексацію та інші операції для ефективного управління пальцевими даними.

Виходячи з розгляду перерахованих вище можливостей, перейдемо до побудови функціональної схеми та діаграми розробленого, у результаті виконання бакалаврського проекту, програмного забезпечення.

3.3 Розробка функціональної схеми

Функціональна схема зображена на рисунку 3.2. Та повністю відповідає представленій темі – функціональні особливості проекту.

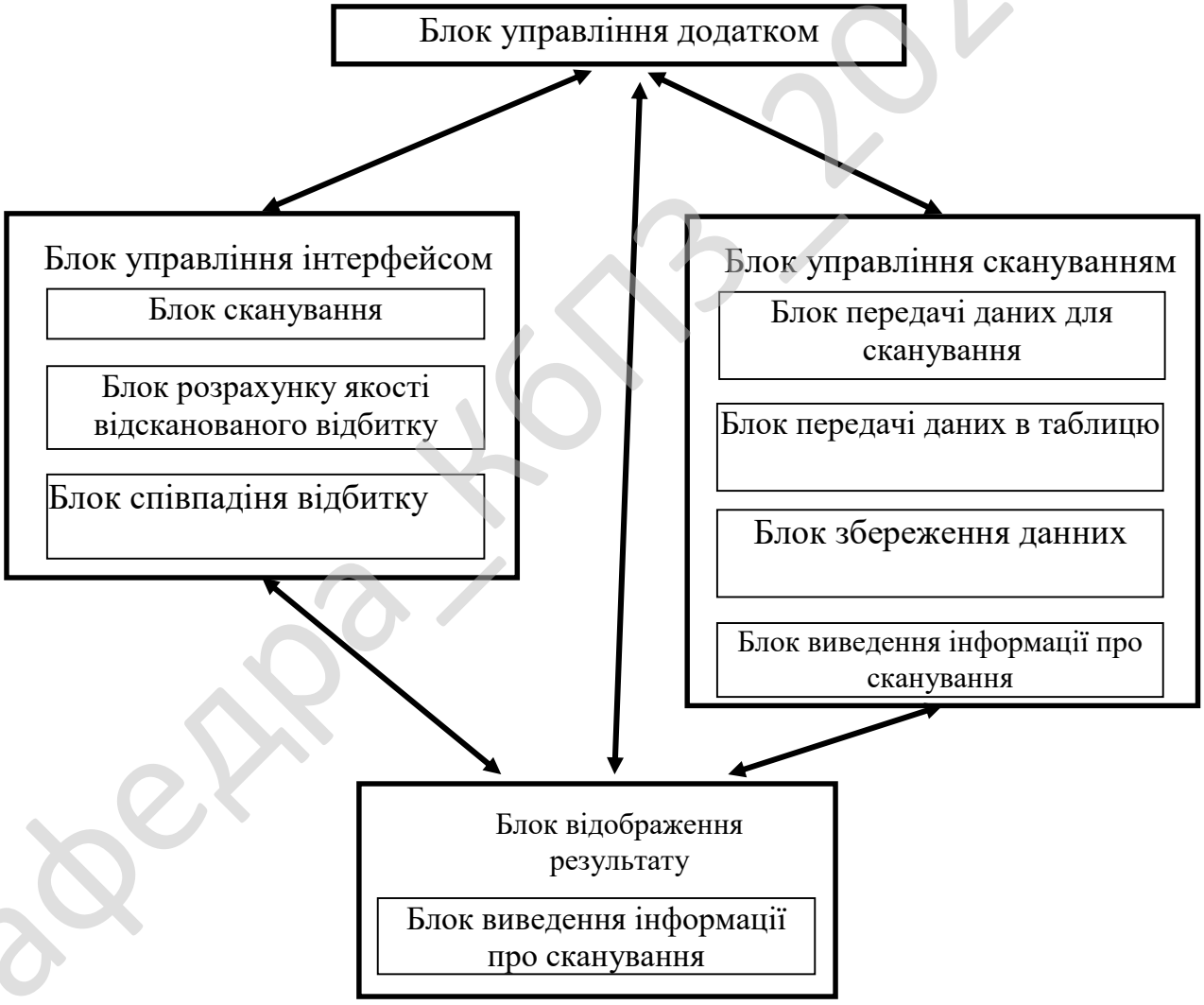

Рисунок 3.2 – Функціональна схема розробленої системи

У контексті програми розпізнавання відбитків пальців для операційної системи Android схеми функціонального програмного забезпечення стосуються різних підходів і методів, які використовуються для реалізації функціональних можливостей програми.

Ці схеми окреслюють ключові етапи процесу розпізнавання відбитків пальців і допомагають розробникам відповідно проектувати та структурувати програмне забезпечення.

Розглянемо схематичні покрокові дії програмного забезпечення, які зазвичай використовуються в програмах розпізнавання відбитків пальців для Android.

1. Схема реєстрації відбитків пальців.

Ця схема зосереджена на процесі захоплення та реєстрації відбитка пальця користувача в системі. Він передбачає використання Android Fingerprint API або BiometricPrompt API для отримання високоякісних зображень відбитків пальців. Зібрані дані відбитків пальців обробляються, функції витягуються та створюється унікальне представлення (шаблон) відбитка пальця.

Шаблон безпечно зберігається в базі даних або на пристрої для подальшого порівняння під час автентифікації.

2. Схема автентифікації за відбитками пальців.

Ця схема описує процес автентифікації відбитка пальця користувача за зареєстрованими шаблонами відбитків пальців. Користувач прикладає свій палець до датчика відбитків пальців пристрою Android. Датчик відбитків пальців фіксує нове зображення відбитка пальця та обробляє його.

Функції витягуються із зробленого зображення та порівнюються із зареєстрованими шаблонами за допомогою відповідних алгоритмів.

Результат відповідності визначає, чи збігається представлений відбиток з будь-яким із зареєстрованих відбитків пальців, таким чином надаючи або забороняючи доступ.

3. Схема управління шаблоном.

Ця схема зосереджена на управлінні шаблонами відбитків пальців у системі. Він включає такі операції, як додавання нових шаблонів під час реєстрації, оновлення шаблонів для змін відбитків пальців і видалення шаблонів, коли вони більше не потрібні. Схема також вирішує проблеми, пов'язані зі зберіганням шаблонів, пошуком і безпечною обробкою для захисту даних користувачів.

4. Схема обробки помилок.

Ця схема обробляє можливі помилки та винятки, які можуть виникнути під час процесу розпізнавання відбитків пальців. Він містить механізми виявлення помилок для виявлення таких проблем, як низька якість зображення, часткове захоплення відбитків пальців або збій датчика.

Методи обробки помилок можуть включати надання інформативного зворотного зв'язку користувачеві, спонукання його змінити положення пальця або відображення повідомлень про помилки, коли це необхідно.

5. Схема інтеграції.

Ця схема спрямована на інтеграцію функції розпізнавання відбитків пальців у загальну програму Android. Це передбачає використання відповідних бібліотек, API та фреймворків (як зазначено в попередній відповіді) для безпроблемного впровадження можливостей розпізнавання відбитків пальців. Схема інтеграції гарантує, що програма розпізнавання відбитків пальців гармонійно працює з іншими функціями програми, інтерфейсом користувача та заходами безпеки.

Ці функціональні кроки функціональної схеми програмного забезпечення забезпечують системний підхід до розробки та реалізації програми розпізнавання відбитків пальців для Android.

Вони допомагають організувати різні етапи процесу розпізнавання відбитків пальців, забезпечуючи надійну та зручну роботу. Розробники можуть адаптувати та налаштовувати ці схеми на основі конкретних вимог, випадків використання та можливостей пристрою Android і API, що використовуються.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

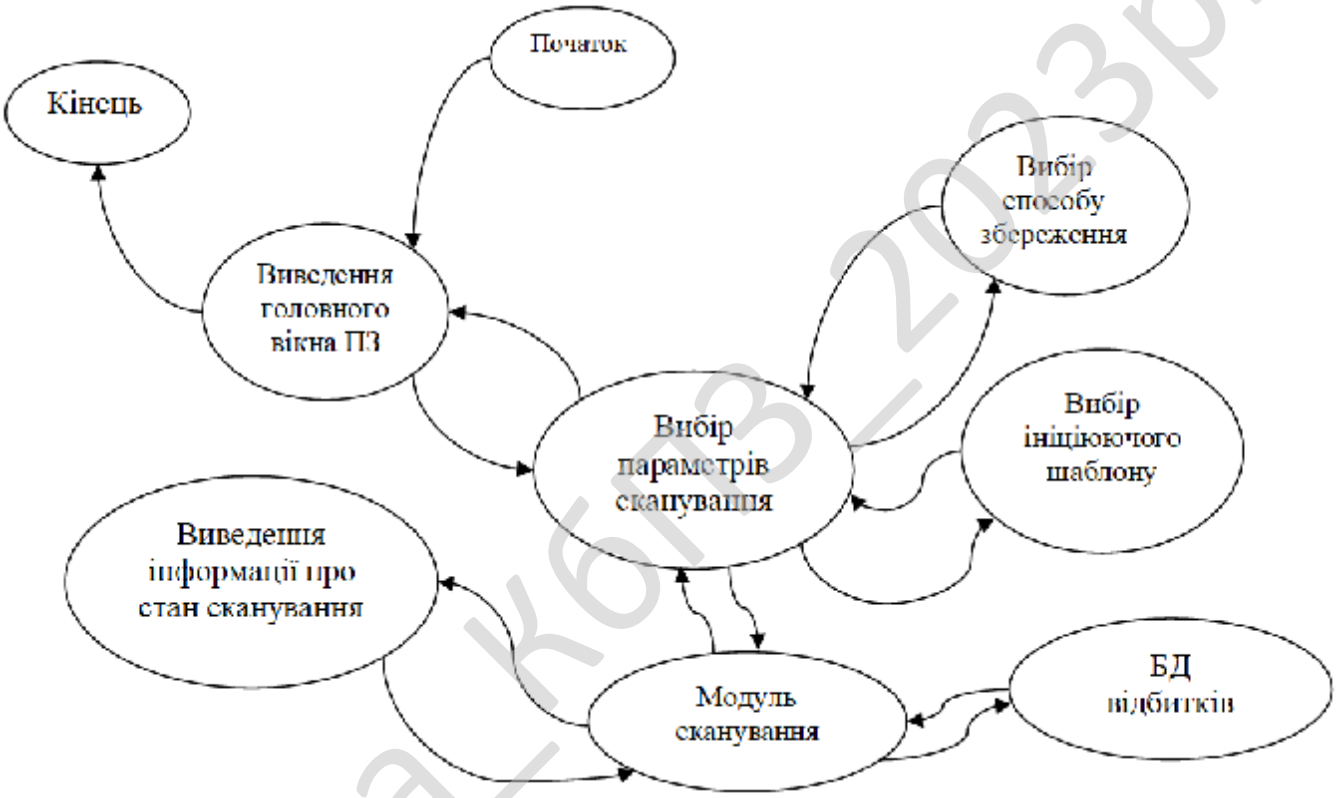

Рисунок 3.3 – Діаграма процесів системи

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розроблені під час виконання роботи блок-схеми програмного забезпечення представлено на рисунках 4.1 та 4.2. З блок-схеми основної програми видно, що спочатку відбувається виведення головного вікна програми.

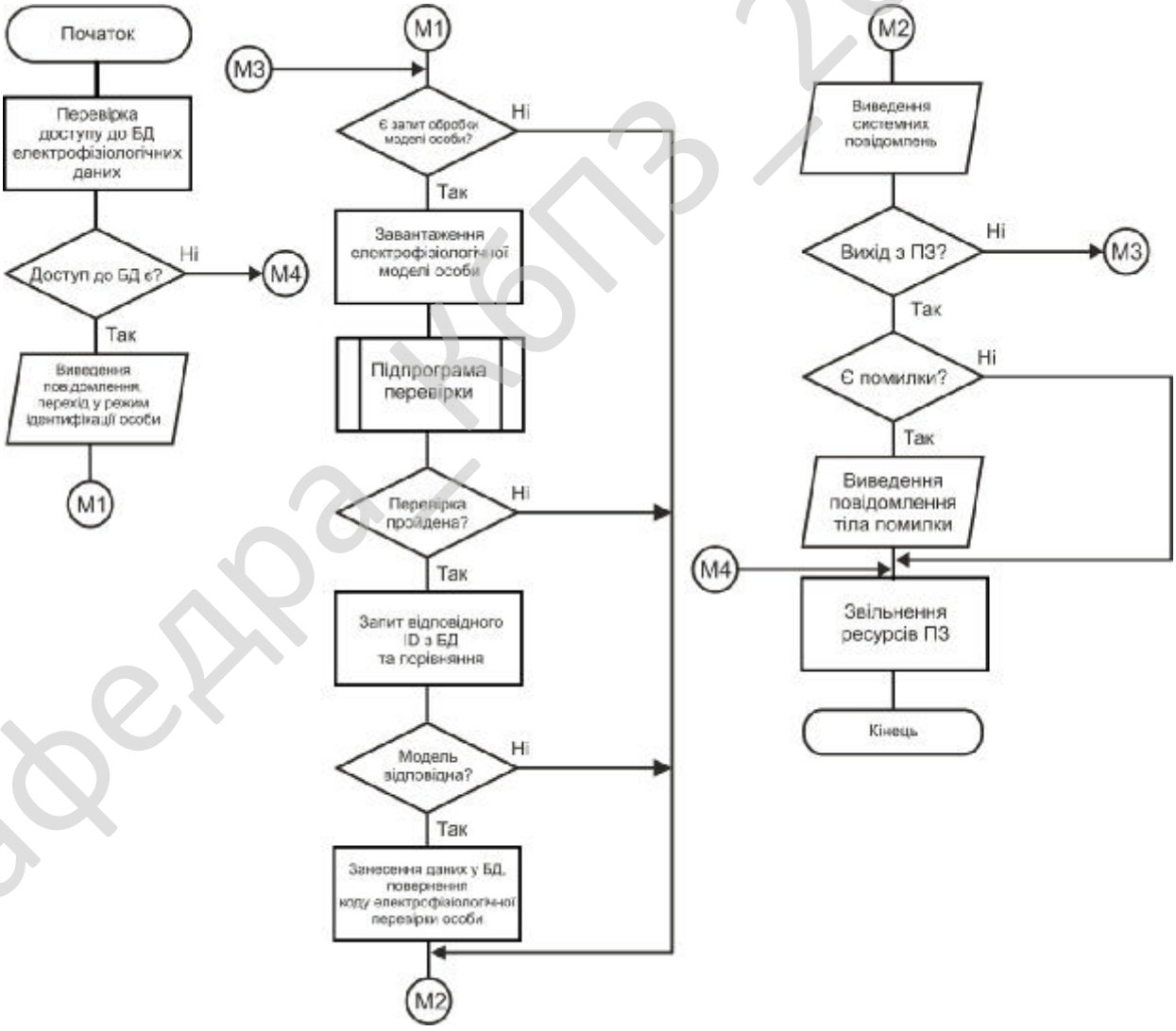

Рисунок 4.1 – Блок-схема роботи основної програми

Потім користувачу запропоновано запустити засіб сканування вібитку. Користувач прикладає палець і запускається процес зчитування відбитку. Цей параметр відповідає за считування форми та конуру вібитку. Користувач може спостерігати за станом сканування та його статусом. Користувач може спостерігати за створенням шаблону пальцю.

Як показано на блок-схемі – далі відбувається зберігання відбитку в спеціальну графу, створену в базі даних.

Після того, як вібиток згенерувався в бінарний код та зберігся в базі данних відбувається звірення відбитку на перевірка.

Коли відбиток вже перевірено, користувачу доступна інформація про стан сканування та про додавання вібитку.

Далі користувач може або додати ще один відбиток або ж просто завершити роботу.

Розглянемо алгоритм дій що передбачає що використовується Xamarin Android у Visual Studio або еквівалентному середовищі розробки.

Кожен крок коментується, щоб пояснити його призначення, від перевірки апаратної підтримки до обробки результатів автентифікації та помилок.

Крок 1: Імпорт необхідних просторів імен

```
using Android.App;
using Android.Content;
using Android.OS;
using Android.Widget;
using Android.Support.V4.Hardware.Fingerprint;
using Android;
using Android.Content.PM;
using Android.Support.V4.App;

namespace FingerprintRecognitionApp
{
[Activity(Label = "FingerprintRecognitionApp", MainLauncher = true)]
public class MainActivity : Activity
{
```

Крок 2: Визначення необхідних змінних

```
private static readonly int RequestFingerprintPermissionId = 1001;
```

```
        private FingerprintManagerCompat fingerprintManager;
        private FingerprintManagerCompat.AuthenticationCallback
authenticationCallback;
```

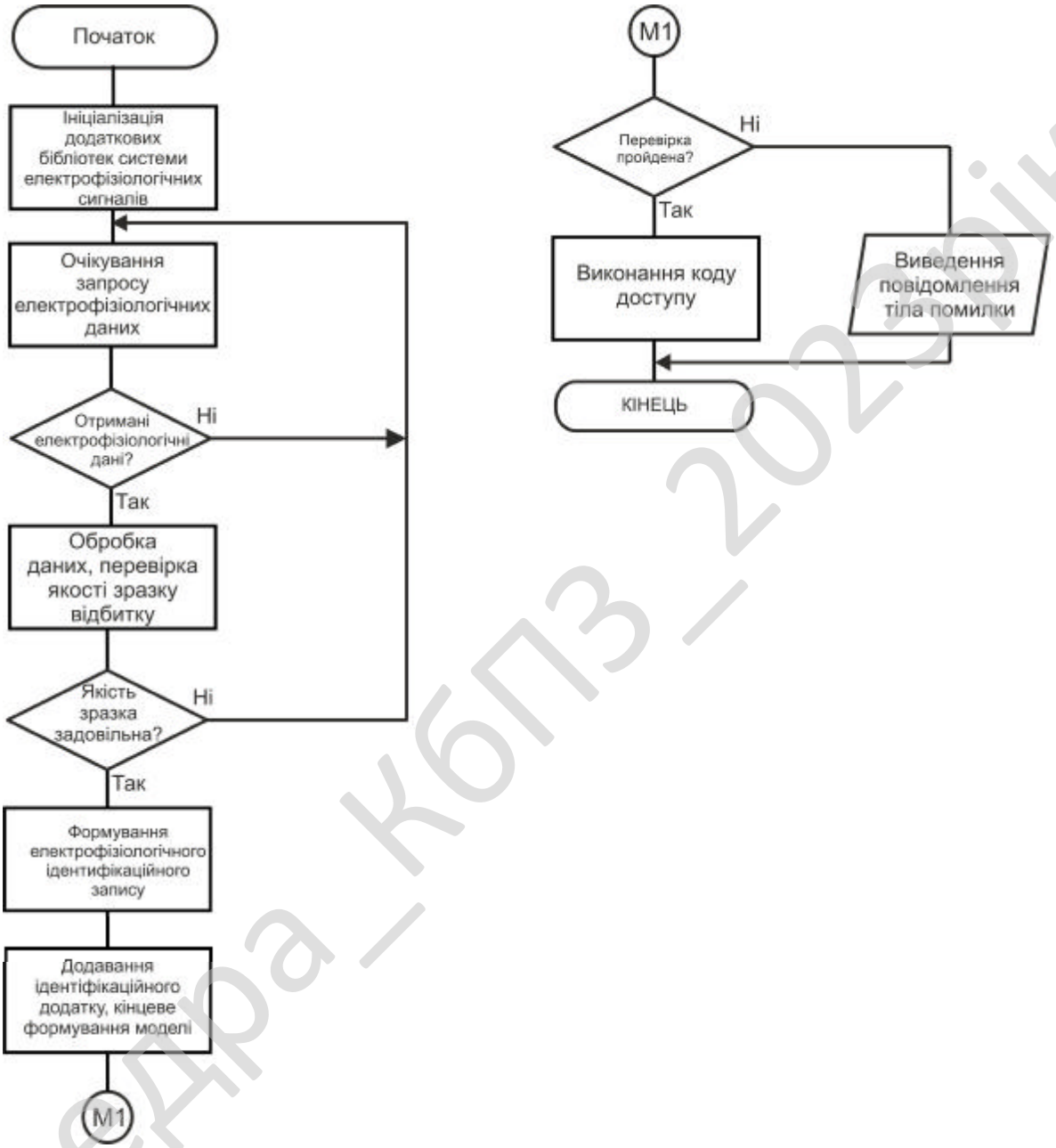

Рисунок 4.2 – Блок-схема роботи підпрограми

```
        protected override void OnCreate(Bundle savedInstanceState)
        {
         base.OnCreate(savedInstanceState);
         SetContentView(Resource.Layout.Main);
```

Крок 3: Перевірка, чи пристрій підтримує автентифікацію за відбитками пальців

```
        fingerprintManager = FingerprintManagerCompat.FromContext(this);
        if (!fingerprintManager.IsHardwareDetected)
        {
            Toast.MakeText(this, "Fingerprint authentication is not supported on this device.", ToastLength.Long).Show();
            return;
        }
```

Крок 4: Перевірка наявності необхідних дозволів

```
if (ActivityCompat.CheckSelfPermission(this, Manifest.Permission.UseFingerprint) != (int)Permission.Granted)
        {
            ActivityCompat.RequestPermissions(this, new string[] { Manifest.Permission.UseFingerprint }, RequestFingerprintPermissionId);
            return;
        }
```

Крок 5: Налаштування зворотнього виклику автентифікації

```
        authenticationCallback = new FingerprintAuthenticationCallback();
        var cryptoObject = new FingerprintManagerCompat.CryptoObject(null);
        fingerprintManager.Authenticate(cryptoObject, 0, null, authenticationCallback, null);
    }
```

Крок 6: Обробка результату запиту дозволу

```
    public override void OnRequestPermissionsResult(int requestCode, string[] permissions, Permission[] grantResults)
    {
        if (requestCode == RequestFingerprintPermissionId)
        {
            if (grantResults[0] == Permission.Granted)
            {
                // Permission granted, start fingerprint authentication
            var cryptoObject = new FingerprintManagerCompat.CryptoObject(null);
            fingerprintManager.Authenticate(cryptoObject, 0, null, authenticationCallback, null);
            }
            else
            {
                Toast.MakeText(this, "Fingerprint permission denied.", ToastLength.Long).Show();
```

```
            }
        }
    }
```

Крок 7: Створення вкладеня класу для зворотного виклику автентифікації

```
    private class FingerprintAuthenticationCallback :
FingerprintManagerCompat.AuthenticationCallback
    {
     public override void
OnAuthenticationSucceeded(FingerprintManagerCompat.AuthenticationResult result)
     {
```

Крок 8: Виконання успішної автентифікації за відбитками пальців

```
    Toast.MakeText(Application.Context, "Fingerprint authentication
succeeded.", ToastLength.Short).Show();
     }

     public override void OnAuthenticationFailed()
     {
```

Крок 9: Обробка невдалої автентифікації за відбитком пальця

```
         Toast.MakeText(Application.Context, "Fingerprint authentication
failed.", ToastLength.Short).Show();
     }

     public override void OnAuthenticationError(int errMsgId, ICharSequence
errString)
     {
```

Крок 10: Виправка помилки автентифікації відбитків пальців

```
Toast.MakeText(Application.Context, $"Fingerprint authentication error:
{errString}", ToastLength.Long).Show();
    }}}}
```

Для створення додатку використовувалась бібліотека Android v7 AppCompat та мова програмування C#.

Перш за все потрібно розробити структури даних `Home` та `FingerPrintHandlres`.

Структура даних `Home` має наступний вигляд:

```
using Android.App;
using Android.OS;
using Android.Support.V7.App;
```

```
namespace FingerAuth
{
    [Activity(Label = "HomeActivity" , Theme ="@style/AppTheme")]
    public class HomeActivity : AppCompatActivity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            // Create your application here
            SetContentView(Resource.Layout.Home);
        }
    }
}
```

Структура даних `FingerPrintHandlres` має наступний вигляд:

```
using System;
using Android;
using Android.Content;
using Android.Hardware.Fingerprints;
using Android.OS;
using Android.Support.V4.App;
using Android.Widget;
namespace FingerAuth
{
internal class FingerprintHandler:FingerprintManager.AuthenticationCallback
{
     private Context mainActivity;
     public FingerprintHandler(Context mainActivity)
        {
            this.mainActivity = mainActivity;
        }
internal void StartAuthentication(FingerprintManager fingerprintManager,
                FingerprintManager.CryptoObject cryptoObject)
{
CancellationSignal cancellationSignal = new CancellationSignal();
if (ActivityCompat.CheckSelfPermission(mainActivity,
     Manifest.Permission.UseFingerprint)
     != (int)Android.Content.PM.Permission.Granted)
     return;
fingerprintManager.Authenticate(cryptoObject, cancellationSignal, 0,
                                this, null);
        }
```

```
        public override void OnAuthenticationFailed()
        {
Toast.MakeText(mainActivity, "Fingerprint Authentication
               failed!",ToastLength.Long).Show();
        }
public override void
OnAuthenticationSucceeded(FingerprintManager.AuthenticationResult result)
        {
mainActivity.StartActivity(new Intent(mainActivity, typeof(HomeActivity)));
        }
    }
}
```

Далі слід сгенерувати MainActivity який демонструє наступне.

```
using Android.App;
using Android.Widget;
using Android.OS;
using Android.Support.V7.App;
using Java.Security;
using Javax.Crypto;
using Android.Hardware.Fingerprints;
using Android.Support.V4.App;
using Android;
using System;
using Android.Security.Keystore;
namespace FingerAuth
{
[Activity(Label = "FingerAuth", MainLauncher = true,
          Theme ="@style/AppTheme")]
public class MainActivity : AppCompatActivity
    {
        private KeyStore keyStore;
        private Cipher cipher;
        private string KEY_NAME = "Ahsan";
        protected override void OnCreate(Bundle savedInstanceState)
        {
base.OnCreate(savedInstanceState);
// Set our view from the "main" layout resource
SetContentView(Resource.Layout.Main);
KeyguardManager keyguardManager =
                (KeyguardManager)GetSystemService(KeyguardService);
FingerprintManager fingerprintManager =
```

```
                    (FingerprintManager)GetSystemService(FingerprintService);
if (ActivityCompat.CheckSelfPermission(this,
    Manifest.Permission.UseFingerprint)
    != (int)Android.Content.PM.Permission.Granted)
    return;
if (!fingerprintManager.IsHardwareDetected)
    Toast.MakeText(this, "FingerPrint authentication permission not
     enable", ToastLength.Short).Show();
else
{
if(!fingerprintManager.HasEnrolledFingerprints)
Toast.MakeText(this, "Register at least one fingerprint in Settings",
               ToastLength.Short).Show();
else
{
if (!keyguardManager.IsKeyguardSecure)
Toast.MakeText(this, "Lock screen security not enable in Settings",
               ToastLength.Short).Show();
else
GenKey();
if (CipherInit())
{
FingerprintManager.CryptoObject cryptoObject = new
       FingerprintManager.CryptoObject(cipher);
       FingerprintHandler handler = new FingerprintHandler(this);
       handler.StartAuthentication(fingerprintManager, cryptoObject);
}
}
}
}
private bool CipherInit()
{
try
{
cipher = Cipher.GetInstance(KeyProperties.KeyAlgorithmAes
+ "/" + KeyProperties.BlockModeCbc  + "/"
+ KeyProperties.EncryptionPaddingPkcs7);
keyStore.Load(null);
IKey key = (IKey)keyStore.GetKey(KEY_NAME, null);
cipher.Init(CipherMode.EncryptMode, key);
return true;
}
```

```
        catch(Exception ex) { return false; }
        }
        private void GenKey()
        {
        keyStore = KeyStore.GetInstance("AndroidKeyStore");
        KeyGenerator keyGenerator = null;
        keyGenerator = KeyGenerator.GetInstance(KeyProperties.KeyAlgorithmAes,
"AndroidKeyStore");
        keyStore.Load(null);
        keyGenerator.Init(new KeyGenParameterSpec.Builder(KEY_NAME,
KeyStorePurpose.Encrypt | KeyStorePurpose.Decrypt)
        .SetBlockModes(KeyProperties.BlockModeCbc)
        .SetUserAuthenticationRequired(true)
        .SetEncryptionPaddings(KeyProperties
        .EncryptionPaddingPkcs7).Build());
        keyGenerator.GenerateKey();
        }
        }
        }
```

Також після того як ми створили головні етапи програми, потрібно додати візульні елементи, які показані в Styles, Main, та в Home

Styles:

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
  <item name="colorPrimary">#263237</item>
  <item name="colorPrimaryDark">#263237</item>
  <item name="colorAccent">#1e282d</item>
</style>
</resources>
```

Main:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#263237"
        android:padding="16dp">
        <ImageView
            android:id="@+id/fingerImage"
```

```
                android:src="@drawable/fingerprint"
                android:layout_centerInParent="true"
                android:layout_width="150dp"
                android:layout_height="150dp" />
            <TextView
                android:layout_below="@+id/fingerImage"
                android:textColor="#F5F5F5"
                android:textSize="20sp"
                android:text="Please place your fingertip on the scanner to
verify your identity"
                android:layout_centerHorizontal="true"
                android:textAlignment="center"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
        </RelativeLayout>
```

Home:

```
        <?xml version="1.0" encoding="utf-8"?>
        <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
            android:padding="16dp"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <TextView
        android:textColor="#95aab4"
        android:textSize="20sp"
        android:text="You have successfully logged in with Fingerprint
                    Authentication"
                android:layout_centerHorizontal="true"
                android:textAlignment="center"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
        </RelativeLayout>
```

4.2 Захист розробленого програмного забезпечення

Я розробив відповідий захист програмного забезпечення, який зберігає у відповідні графи таблиць інформацію в унікальному бінарному коді, та дає доступ для перегляду інформації, лише обмеженому колу осіб.

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм MMB, в основі якого лежить змішування операцій різних алгебраїчних груп. MMB – ітеративний алгоритм, що складається з лінійних дій (XOR і використання ключа) і паралельного застосування чотирьох великих оборотних нелінійних підстановок. Ці підстановки визначаються за допомогою множення по модулю 2^{32}-1 з постійними множниками. У підсумку з'являється алгоритм, що використовує 128-бітовий ключ і 128-бітовий блок.

Алгоритм MMB оперує 32-бітовими підблоками тексту (x_0, x_1, x_2, x_3) і 32-бітовими підблоками ключу (k_0, k_1, k_2, k_3). Це спрощує реалізацію алгоритму на сучасних 64-бітових процесорах. Чергуючись із операцією XOR, шість разів використовується нелінійна функція *f*. Запишемо операції алгоритму (всі операції з індексами виконуються по модулю 4):

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

Функція *f* виконується в три кроки:

1. $x_i = c_i * x_i$ для $i = 0..3$ (Якщо на вході множення одні одиниці, то на виході – теж одні одиниці).

2. Якщо молодший значущий біт $x_0 = 1$, то $x_0 = x_0 \oplus C$. Якщо молодший значущий байт $x_3 = 0$, то $x_3 = x_3 \oplus C$.

3. $x_i = x_{i-1} \oplus x_i \oplus x_{i+1}$ для $i = 0..3$.

Всі операції з індексами виконуються по модулю 4. Операція множення на кроці 1 виконується по модулі 2^{32}-1. Спеціальний випадок для даного алгоритму: якщо другий операнд дорівнює 2^{32}-1, результат теж дорівнює 2^{32}-1. В алгоритмі використовуються наступні константи:

$$C = 2aaaaaaa,\ c_0 = 025f1cdb,\ c_1 = 2*c_0,\ c_2 = 2^3 * c_0,\ c_3 = 2^7 * c_0.$$

Константа C – «найпростіша» константа без кругової симетрії, високою трійковою вагою й нульовим молодшим значущим бітом. У константи c_0 є інші особливі характеристики. Константи c_1, c_2 і c_3 – зрушені версії c_0, і служать для запобігання атак, заснованих на симетрії.

Розшифрування виконується у зворотному порядку, Етапи 2 і 3 інверсні їм самим. На етапі 1 замість c_i використовується c_i^{-1}. Значення c_0^{-1} = 0dad4694.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс програми складається з вікна запиту відбитку пальця користувача, всі налаштування приховані від користувача в цілях підвищення безпеки даних.

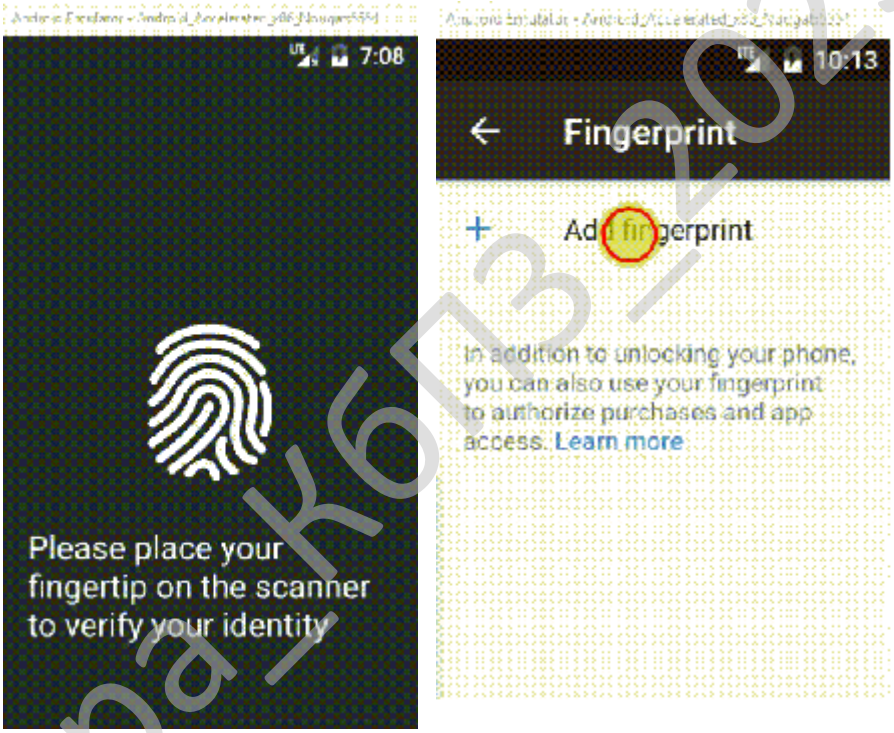

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Android без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і

працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

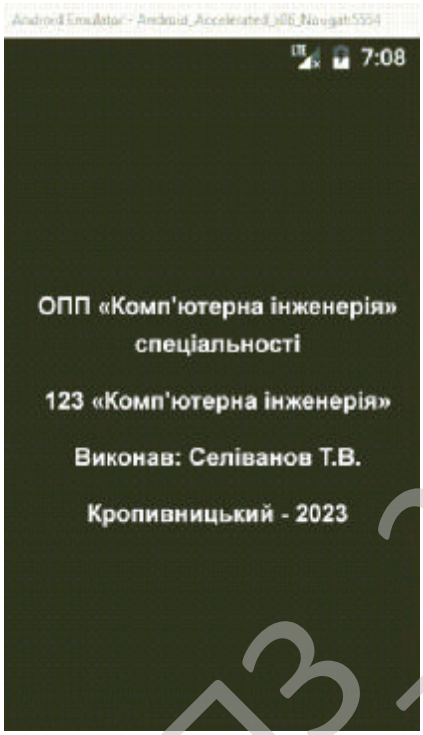

Рисунок 5.2 – Довідка

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних

вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи збирання та обробки електрофізіологічних сигналів.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем збирання та обробки електрофізіологічних сигналів.

– Досліджена система збирання та обробки електрофізіологічних сигналів.

– На основі отриманих результатів досліджень створена програмна реалізація системи збирання та обробки електрофізіологічних сигналів.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання збирання та обробки електрофізіологічних сигналів.

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

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

Програма реалізована на мові високого рівня C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи збирання та обробки електрофізіологічних сигналів. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку.

Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Android.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм MMB.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

2. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.: ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

3. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС, 2012. – Т. 1., Вип. 3(101). – С. 139-142.

4. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. – Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Российский университет кооперации". – 2012. – Том.1. – С. 152-160.

5. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

6. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

7. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и

сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

8. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

9. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

10. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

11. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

12. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

13. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам

Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

14. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

15. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

16. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

17. Смирнов С. А. Комплекс gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. - практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

18. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

19. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

20. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы /

В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

21. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

22. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

23. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

24. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

25. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

26. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару

«Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

27. Смирнов С. А. технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

28. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

29. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

30. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

31. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

32. Смирнов С. А. Разработка комплекса gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук.-практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

33. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

34. Смирнов С. А. gert-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

35. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

36. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

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

38. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.
39. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.
40. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.
41. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.
42. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.
43. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.
44. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.
45. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення

поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

46. Смірнов О.А., Смірнова Т.В., Солових Є.К., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.
47. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.
48. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.
49. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.
50. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-123.23.0025.00.00.ТЗ	
Вим.	Арк.	№ документа	Підпис	Дата		
Розробив		Селіванов Т. В.			Програмне забезпечення системи збирання та обробки електрофізіологічних сигналів	Літ. Б / Аркуш 1 / Аркушів 6
Перевірив		Коваленко О.В.				ЦНТУ КІ-20-3СК
Н. Контр.		Гермак В.С.				
Затв.		Смірнов О.А.				

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи збирання та обробки електрофізіологічних сигналів.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи збирання та обробки електрофізіологічних сигналів.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему збирання та обробки електрофізіологічних сигналів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на Android та з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 61 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 12.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Коваленко О.В.

Програмне забезпечення системи збирання та обробки електрофізіологічних сигналів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 19

Літера: РП

Кропивницький – 2023 року

Файл FingerprintHandler.cs

```
using System;
using Android;
using Android.Content;
using Android.Hardware.Fingerprints;
using Android.OS;
using Android.Support.V4.App;
using Android.Widget;
namespace FingerAuth
{
    internal class FingerprintHandler :
FingerprintManager.AuthenticationCallback
    {
        private Context mainActivity;
        public FingerprintHandler(Context mainActivity)
        {
            this.mainActivity = mainActivity;
        }
        internal void StartAuthentication(FingerprintManager fingerprintManager,
FingerprintManager.CryptoObject cryptoObject)
        {
            CancellationSignal cancellationSignal = new CancellationSignal();
            if (ActivityCompat.CheckSelfPermission(mainActivity,
Manifest.Permission.UseFingerprint)
                != (int)Android.Content.PM.Permission.Granted)
                return;
            fingerprintManager.Authenticate(cryptoObject, cancellationSignal, 0,
this, null);
        }
        public override void OnAuthenticationFailed()
        {
            Toast.MakeText(mainActivity, "Fingerprint Authentication
failed!",ToastLength.Long).Show();
        }
        public override void
OnAuthenticationSucceeded(FingerprintManager.AuthenticationResult result)
        {
            mainActivity.StartActivity(new Intent(mainActivity,
typeof(HomeActivity)));
        }
    }
}
```

Файл Home.cs

```
using Android.App;
using Android.OS;
using Android.Support.V7.App;
namespace FingerAuth
{
    [Activity(Label = "HomeActivity" , Theme ="@style/AppTheme")]
    public class HomeActivity : AppCompatActivity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            // Create your application here
            SetContentView(Resource.Layout.Home);
        }
    }
}
```

Файл MainActivity.cs

```
using Android.App;
using Android.Widget;
using Android.OS;
using Android.Support.V7.App;
using Java.Security;
using Javax.Crypto;
using Android.Hardware.Fingerprints;
using Android.Support.V4.App;
using Android;
using System;
using Android.Security.Keystore;
namespace FingerAuth
{
    [Activity(Label = "FingerAuth", MainLauncher = true, Theme
="@style/AppTheme")]
    public class MainActivity : AppCompatActivity
    {
        private KeyStore keyStore;
        private Cipher cipher;
        private string KEY_NAME = "77hsan";
        protected override void OnCreate(Bundle savedInstanceState)
        {
base.OnCreate(savedInstanceState);
// Set our view from the "main" layout resource
SetContentView(Resource.Layout.Main);
KeyguardManager keyguardManager =
(KeyguardManager)GetSystemService(KeyguardService);
FingerprintManager fingerprintManager =
(FingerprintManager)GetSystemService(FingerprintService);
if (ActivityCompat.CheckSelfPermission(this, Manifest.Permission.UseFingerprint)
!= (int)Android.Content.PM.Permission.Granted)
return;

if (!fingerprintManager.IsHardwareDetected)
Toast.MakeText(this, "FingerPrint authentication permission not enable",
ToastLength.Short).Show();
            else
            {
if(!fingerprintManager.HasEnrolledFingerprints)
Toast.MakeText(this, "Register at least one fingerprint in Settings",
ToastLength.Short).Show();
                else
                {
if (!keyguardManager.IsKeyguardSecure)
Toast.MakeText(this, "Lock screen security not enable in Settings",
ToastLength.Short).Show();
                    else
                        GenKey();
                    if (CipherInit())
                    {
```

```
FingerprintManager.CryptoObject cryptoObject = new
FingerprintManager.CryptoObject(cipher);
FingerprintHandler handler = new FingerprintHandler(this);
handler.StartAuthentication(fingerprintManager, cryptoObject);
                    }
                }
            }
        }
        private bool CipherInit()
        {
            try
            {
                cipher = Cipher.GetInstance(KeyProperties.KeyAlgorithmAes
                    + "/"
                    + KeyProperties.BlockModeCbc
                    + "/"
                    + KeyProperties.EncryptionPaddingPkcs7);
                keyStore.Load(null);
                IKey key = (IKey)keyStore.GetKey(KEY_NAME, null);
                cipher.Init(CipherMode.EncryptMode, key);
                return true;
            }
            catch(Exception ex) { return false; }
        }
        private void GenKey()
        {
keyStore = KeyStore.GetInstance("AndroidKeyStore");
KeyGenerator keyGenerator = null;
keyGenerator = KeyGenerator.GetInstance(KeyProperties.KeyAlgorithmAes,
"AndroidKeyStore");
keyStore.Load(null);
keyGenerator.Init(new KeyGenParameterSpec.Builder(KEY_NAME,
KeyStorePurpose.Encrypt | KeyStorePurpose.Decrypt)
                .SetBlockModes(KeyProperties.BlockModeCbc)
                .SetUserAuthenticationRequired(true)
                .SetEncryptionPaddings(KeyProperties
                .EncryptionPaddingPkcs7).Build());
            keyGenerator.GenerateKey();
        }
    }
}
```

Файл Styles.xml, Home.axml, Main.axml

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
  <item name="colorPrimary">#263237</item>
  <item name="colorPrimaryDark">#263237</item>
  <item name="colorAccent">#1e282d</item>
</style>
</resources>
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:padding="16dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:textColor="#95aab4"
        android:textSize="20sp"
        android:text="You have successfully logged in with Fingerprint
Authentication"
        android:layout_centerHorizontal="true"
        android:textAlignment="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout mlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#263237"
    android:padding="16dp">
<ImageView
        android:id="@+id/fingerImage"
        android:src="@drawable/fingerprint"
        android:layout_centerInParent="true"
        android:layout_width="150dp"
        android:layout_height="150dp" />
<TextView
        android:layout_below="@+id/fingerImage"
        android:textColor="#F5F5F5"
        android:textSize="20sp"
        android:text="Please place your fingertip on the scanner to verify your
identity"
        android:layout_centerHorizontal="true"
        android:textAlignment="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

Файл BiometricAuthService.cs

```
using System;
using System.Threading.Tasks;
using Android;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Security.Keystore;
using Android.Support.V4.App;
using Android.Support.V4.Hardware.Fingerprint;
using Android.Util;
using Android.Widget;
using BiometricAuthentication.Droid;
using Java.Security;
using Javax.Crypto;

[assembly: Xamarin.Forms.Dependency(typeof(BiometricAuthService))]
namespace BiometricAuthentication.Droid
{
public class BiometricAuthService : IBiometricAuthenticateService
{
Context context = Android.App.Application.Context;
private KeyStore keyStore;
private Cipher cipher;
private string KEY_NAME = "XamarinLife";
public static bool IsAutSucess;

public string GetAuthenticationType()
{
return "";
}

public Task<bool> AuthenticateUserIDWithTouchID()
{
var tcs = new TaskCompletionSource<bool>();//used to wait the mainUI to get the
response of the touchId

KeyguardManager keyguardManager =
(KeyguardManager)context.GetSystemService(Context.KeyguardService);
FingerprintManagerCompat fingerprintManager =
FingerprintManagerCompat.From(context);

if (Build.VERSION.SdkInt >= BuildVersionCodes.P)
{
if (ActivityCompat.CheckSelfPermission(context,
Manifest.Permission.UseBiometric)
!= (int)Android.Content.PM.Permission.Granted)
return tcs.Task;
if (!fingerprintManager.IsHardwareDetected)
Toast.MakeText(context, "FingerPrint authentication permission not enable",
ToastLength.Short).Show();
else
```

```
{
if (!fingerprintManager.HasEnrolledFingerprints)
Toast.MakeText(context, "Register at least one fingerprint in Settings",
ToastLength.Short).Show();
else
{
if (!keyguardManager.IsKeyguardSecure)
 Toast.MakeText(context, "Lock screen security not enable in Settings",
ToastLength.Short).Show();
else
 GenKey();
if (CipherInit())
{
 FingerprintManagerCompat.CryptoObject cryptoObject = new
FingerprintManagerCompat.CryptoObject(cipher);
 BiometricHandler handler = new BiometricHandler(context);
 handler.StartAuthentication(fingerprintManager, cryptoObject);
}
}
}
}
else if (Build.VERSION.SdkInt >= BuildVersionCodes.M)
{
if (ActivityCompat.CheckSelfPermission(context,
Manifest.Permission.UseFingerprint)
!= (int)Android.Content.PM.Permission.Granted)
return tcs.Task;
if (!fingerprintManager.IsHardwareDetected)
Toast.MakeText(context, "FingerPrint authentication permission not enable",
ToastLength.Short).Show();
else
{
if (!fingerprintManager.HasEnrolledFingerprints)
Toast.MakeText(context, "Register at least one fingerprint in Settings",
ToastLength.Short).Show();
else
{
if (!keyguardManager.IsKeyguardSecure)
 Toast.MakeText(context, "Lock screen security not enable in Settings",
ToastLength.Short).Show();
else
 GenKey();
if (CipherInit())
{
 FingerprintManagerCompat.CryptoObject cryptoObject = new
FingerprintManagerCompat.CryptoObject(cipher);
 BiometricHandler handler = new BiometricHandler(context);
 handler.StartAuthentication(fingerprintManager, cryptoObject);
}
}
}
}
else
```

```
{
return tcs.Task;
}
tcs.SetResult(IsAutSucess);
return tcs.Task;
}

private bool CipherInit()
{
try
{
cipher = Cipher.GetInstance(KeyProperties.KeyAlgorithmAes
+ "/"
+ KeyProperties.BlockModeCbc
+ "/"
+ KeyProperties.EncryptionPaddingPkcs7);
keyStore.Load(null);
IKey key = (IKey)keyStore.GetKey(KEY_NAME, null);
cipher.Init(CipherMode.EncryptMode, key);
return true;
}
catch (Exception ex) { return false; }
}
private void GenKey()
{
keyStore = KeyStore.GetInstance("AndroidKeyStore");
KeyGenerator keyGenerator = null;
keyGenerator = KeyGenerator.GetInstance(KeyProperties.KeyAlgorithmAes,
"AndroidKeyStore");
keyStore.Load(null);
keyGenerator.Init(new KeyGenParameterSpec.Builder(KEY_NAME,
KeyStorePurpose.Encrypt | KeyStorePurpose.Decrypt)
.SetBlockModes(KeyProperties.BlockModeCbc)
.SetUserAuthenticationRequired(true)
.SetEncryptionPaddings(KeyProperties
.EncryptionPaddingPkcs7).Build());
keyGenerator.GenerateKey();
}

public bool fingerprintEnabled()
{
Activity activity = MainActivity.FormsContext;
KeyguardManager keyguardManager =
(KeyguardManager)context.GetSystemService(Context.KeyguardService);
FingerprintManagerCompat fingerprintManager =
FingerprintManagerCompat.From(context);

/*
*Condition I : Check if the andoid version is device is greater than
*Pie, since Biometrics is supported by greater devices
*no fingerprint manager from Android >9.0
*/
```

```
if (Build.VERSION.SdkInt >= BuildVersionCodes.P)
{
if (ActivityCompat.CheckSelfPermission(context,
Manifest.Permission.UseBiometric) == Android.Content.PM.Permission.Granted)
{
if (fingerprintManager != null && fingerprintManager.IsHardwareDetected)
{
if (keyguardManager.IsKeyguardSecure)
{
 if (fingerprintManager.HasEnrolledFingerprints)
 {
     //user has enrolled one or more fingerprints to authenticate
     //ShowBiometricPrompt();
     return true;
 }
 else
 {
     Log.Error("P Biometric Error", "User not enrolled any fingerprints to
authenticate");
     return false;
 }
}
else
{
 Log.Error("P Biometric Error", "Keyguard is not secure");
 return false;
}
}
else
{
Log.Error("P Biometric Error","Device don't support Biometrics authentication");
 return false;
}
}
else
{
Log.Error("P Biometric Error", "User not given permission to access
Biometrics");
ActivityCompat.RequestPermissions(activity, new string[] {
Manifest.Permission.UseBiometric }, 200);
return false;
}
}

/*
*Condition II: check if the android device version is greater than *Marshmallow
*/

else if (Build.VERSION.SdkInt >= BuildVersionCodes.M)
{
//FingerprintManager fingerprintMan =
(FingerprintManager)context.GetSystemService(Context.FingerprintService);//.From
(context);
```

```
if (ActivityCompat.CheckSelfPermission(context,
Manifest.Permission.UseFingerprint) == Android.Content.PM.Permission.Granted)
{
if (fingerprintManager != null && fingerprintManager.IsHardwareDetected)
{
if (keyguardManager.IsKeyguardSecure)
{
 if (fingerprintManager.HasEnrolledFingerprints)
 {
     //user has enrolled one or more fingerprints to authenticate
     return true;
 }
 else
 {
     Log.Error("M Biometric Error", "User not enrolled any fingerprints to
authenticate");
     return false;
 }
}
else
{
 Log.Error("M Biometric Error", "Keyguard is not secure");
 return false;
}
}
else
{
Log.Error("M Biometric Error", "Device don't support Biometrics
authentication");
 return false;
}
}
else
{
Log.Error("P Biometric Error", "User not given permission to access
Biometrics");
ActivityCompat.RequestPermissions(activity, new string[] {
Manifest.Permission.UseFingerprint }, 200);
return false;
}
}
/*
*Lower version don't support for biometric authentication
*/
else
{
Log.Equals("Biometric Error"," Device don't support Fingerprint");
return false;
}
}
}
}
```

Файл Fingerprint577.cs

```
using System;
using Android.Content;
using Android.Support.V4.Hardware.Fingerprint;
using Android.Hardware.Fingerprints;
using CancellationSignal = Android.Support.V4.OS.CancellationSignal;

namespace Xamarin.Android.Fingerprint
{
/// <summary>
/// Fingerprint authenticator.
/// </summary>
public sealed partial class FingerprintAuthenticator
{
private FingerprintManagerCompat _fingerprintManager;
private IFingerprintAuthenticatorCallbacks _authenticationCallback;
private CancellationSignal mCancellationSignal;

/// <summary>
/// Initializes a new instance of the <see
cref="T:Xamarin.Android.Fingerprint.FingerprintAuthenticator"/> class.
/// </summary>
/// <param name="context">Context.</param>
/// <param name="callbacks">Callbacks.</param>
public FingerprintAuthenticator(Context context,
IFingerprintAuthenticatorCallbacks callbacks)
{
if (context == null)
{
throw new ArgumentNullException(nameof(context));
}
if (callbacks == null)
{
throw new ArgumentNullException(nameof(callbacks));
}

_authenticationCallback = callbacks;
_fingerprintManager = FingerprintManagerCompat.From(context);
}

/// <summary>
/// Check if fingerprint is available.
/// </summary>
/// <returns><c>true</c>, if fingerprint available, <c>false</c>
otherwise.</returns>
/// <param name="context">Context.</param>
public static bool IsFingerprintAvailable(Context context)
{
var fingerprintManager = FingerprintManagerCompat.From(context);
return fingerprintManager.IsHardwareDetected &&
fingerprintManager.HasEnrolledFingerprints;
}
```

```
/// <summary>
/// Has enrolled fingerprints.
/// </summary>
/// <returns><c>true</c>, if has enrolled fingerprints, <c>false</c>
otherwise.</returns>
/// <param name="context">Context.</param>
public static bool HasEnrolledFingerprints(Context context) =>
FingerprintManagerCompat.From(context).HasEnrolledFingerprints;

/// <summary>
/// Has fingerprint hardware.
/// </summary>
/// <returns><c>true</c>, if has fingerprint hardware, <c>false</c>
otherwise.</returns>
/// <param name="context">Context.</param>
public static bool IsHardwareDetected(Context context) =>
FingerprintManagerCompat.From(context).IsHardwareDetected;

/// <summary>
/// Check the fingerprint is available.
/// </summary>
/// <returns><c>true</c>, if fingerprint is available, <c>false</c>
otherwise.</returns>
private bool InternalIsFingerprintAvailable()
{
// As said in documentation
// If we are bellow android M this returns true.
if (!_fingerprintManager.IsHardwareDetected)
{
_authenticationCallback.FingerprintNotSupported();
return false;
}
if (!_fingerprintManager.HasEnrolledFingerprints)
{
_authenticationCallback.FingerprintsNotEnrolled();
return false;
}
return true;
}

/// <summary>
/// Starts the authentication.
/// </summary>
public void StartAuthentication()
{
// Stop any previous scan.
StopAuthentication();

if (!InternalIsFingerprintAvailable())
{
return;
}
```

```
try
{
mCancellationSignal = new CancellationSignal();
_fingerprintManager.Authenticate(CryptoObjectBuilder.Build(),
                                 (int)FingerprintAuthenticationFlags.None,
                                 mCancellationSignal,
                                 new AuthenticationCallbacks(this),
                                 null);
}
catch (Exception ex)
{
_authenticationCallback.AuthenticationError(AuthenticationErrorCodes.Unexpected,
ex.Message);
}
}

internal class AuthenticationCallbacks :
FingerprintManagerCompat.AuthenticationCallback
{
private FingerprintAuthenticator _fingerprintAuthenticator;

public AuthenticationCallbacks(FingerprintAuthenticator
fingerprintAuthenticator)
{
_fingerprintAuthenticator = fingerprintAuthenticator;
}

public override void OnAuthenticationError(int errMsgId, Java.Lang.ICharSequence
errString)
{
base.OnAuthenticationError(errMsgId, errString);

var error = (AuthenticationErrorCodes)errMsgId;
if (error != AuthenticationErrorCodes.ErrorCanceled)
{
   _fingerprintAuthenticator._authenticationCallback.AuthenticationError(error,
errString.ToString());
}
}

public override void OnAuthenticationHelp(int helpMsgId, Java.Lang.ICharSequence
helpString)
{
base.OnAuthenticationHelp(helpMsgId, helpString);
_fingerprintAuthenticator._authenticationCallback.AuthenticationHelp((Authentica
tionHelpCodes)helpMsgId, helpString.ToString());
}

public override void OnAuthenticationFailed()
{
base.OnAuthenticationFailed();
_fingerprintAuthenticator._authenticationCallback.FingerprintNotRecognized();
```

```
}

public override void
OnAuthenticationSucceeded(FingerprintManagerCompat.AuthenticationResult result)
{
try
{
   // Calling DoFinal on the Cipher ensures that the encryption worked.
   result.CryptoObject.Cipher.DoFinal(new byte[] { 1, 2, 3 });
   _fingerprintAuthenticator._authenticationCallback.AuthenticationSucceeded();
}
catch (Exception ex)
{

fingerprintAuthenticator._authenticationCallback.AuthenticationError(Authenticat
ionErrorCodes.Unexpected, ex.Message);
}
}
}
/// <summary>
/// Stops the authentication.
/// </summary>
public void StopAuthentication()
{
if (mCancellationSignal != null)
{
mCancellationSignal.Cancel();
mCancellationSignal = null;
}
}
}
}
```

Файл WorkAdditions.cs

```
using Android.Runtime;
using Java.Util.Concurrent;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AndroidX.Work
{
public partial class OneTimeWorkRequest
{
public static OneTimeWorkRequest From<TWorker>() where TWorker :
AndroidX.Work.Worker
=> From(typeof(TWorker));

public static OneTimeWorkRequest From(Type type)
=> From(Java.Lang.Class.FromType(type));

public partial class Builder
{
public Builder(Type type)
: this(Java.Lang.Class.FromType(type))
{
}

public static Builder From<TWorker>() where TWorker : AndroidX.Work.Worker
=> new Builder(Java.Lang.Class.FromType(typeof(TWorker)));

// The base type returns a WorkRequest.Builder and we want it to return
OneTimeWorkRequest.Builder instead
#region Base Builder New Implementations
public new OneTimeWorkRequest Build()
=> base.Build().JavaCast<OneTimeWorkRequest>();

public new Builder AddTag(string tag)
=> base.AddTag(tag).JavaCast<Builder>();

public new Builder KeepResultsForAtLeast(long duration, TimeUnit timeUnit)
=> base.KeepResultsForAtLeast(duration, timeUnit).JavaCast<Builder>();

public Builder KeepResultsForAtLeast(TimeSpan duration)
=> base.KeepResultsForAtLeast((long)duration.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();


public new Builder SetBackoffCriteria(BackoffPolicy policy, long backoffDelay,
TimeUnit timeUnit)
=> base.SetBackoffCriteria(policy, backoffDelay, timeUnit).JavaCast<Builder>();

public Builder SetBackoffCriteria(BackoffPolicy policy, TimeSpan backoffDelay)
```

```
=> base.SetBackoffCriteria(policy, (long)backoffDelay.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();

public new Builder SetConstraints(Constraints constraints)
=> base.SetConstraints(constraints).JavaCast<Builder>();

public new Builder SetInitialRunAttemptCount(int runAttemptCount)
=> base.SetInitialRunAttemptCount(runAttemptCount).JavaCast<Builder>();

public new Builder SetInitialState(WorkInfo.State state)
=> base.SetInitialState(state).JavaCast<Builder>();

public new Builder SetInputData(Data data)
=> base.SetInputData(data).JavaCast<Builder>();

public new Builder SetPeriodStartTime(long periodStartTime, TimeUnit timeUnit)
=> base.SetPeriodStartTime(periodStartTime, timeUnit).JavaCast<Builder>();

public Builder SetPeriodStartTime(TimeSpan periodStartTime)
=> base.SetPeriodStartTime((long)periodStartTime.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();

public new Builder SetScheduleRequestedAt(long scheduleRequestedAt, TimeUnit
timeUnit)
=> base.SetPeriodStartTime(scheduleRequestedAt, timeUnit).JavaCast<Builder>();

public Builder SetScheduleRequestedAt(TimeSpan scheduleRequestedAt)
=> base.SetPeriodStartTime((long)scheduleRequestedAt.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();
#endregion
}
}

public partial class PeriodicWorkRequest
{
public partial class Builder
{
public Builder(Type type, long repeatInterval, TimeUnit repeatIntervalTimeUnit)
: this(Java.Lang.Class.FromType(type), repeatInterval, repeatIntervalTimeUnit)
{ }

public Builder(Type type, long repeatInterval, TimeUnit repeatIntervalTimeUnit,
long flexInterval, TimeUnit flexIntervalTimeUnit)
: this(Java.Lang.Class.FromType(type), repeatInterval, repeatIntervalTimeUnit,
flexInterval, flexIntervalTimeUnit)
{ }

public Builder(Type type, TimeSpan repeatInterval)
: this(Java.Lang.Class.FromType(type), (long)repeatInterval.TotalMilliseconds,
TimeUnit.Milliseconds)
{ }

public Builder(Type type, TimeSpan repeatInterval, TimeSpan flexInterval)
```

```
: this(Java.Lang.Class.FromType(type), (long)repeatInterval.TotalMilliseconds,
TimeUnit.Milliseconds, (long)flexInterval.TotalMilliseconds,
TimeUnit.Milliseconds)
{ }

public static Builder From<TWorker>(long repeatInterval, TimeUnit
repeatIntervalTimeUnit) where TWorker : AndroidX.Work.Worker
=> new Builder(typeof(TWorker), repeatInterval, repeatIntervalTimeUnit);

public static Builder From<TWorker>(TimeSpan repeatInterval) where TWorker :
AndroidX.Work.Worker
=> new Builder(typeof(TWorker), (long)repeatInterval.TotalMilliseconds,
TimeUnit.Milliseconds);

public static Builder From<TWorker>(long repeatInterval, TimeUnit
repeatIntervalTimeUnit, long flexInterval, TimeUnit flexIntervalTimeUnit) where
TWorker : AndroidX.Work.Worker
=> new Builder(typeof(TWorker), repeatInterval, repeatIntervalTimeUnit,
flexInterval, flexIntervalTimeUnit);

public static Builder From<TWorker>(TimeSpan repeatInterval, TimeSpan
flexInterval) where TWorker : AndroidX.Work.Worker
=> new Builder(typeof(TWorker), (long)repeatInterval.TotalMilliseconds,
TimeUnit.Milliseconds, (long)flexInterval.TotalMilliseconds,
TimeUnit.Milliseconds);

// The base type returns a WorkRequest.Builder and we want it to return
PeriodicWorkRequest.Builder instead
#region Base Builder New Implementations
public new PeriodicWorkRequest Build()
=> base.Build().JavaCast<PeriodicWorkRequest>();

public new Builder AddTag(string tag)
=> base.AddTag(tag).JavaCast<Builder>();

public new Builder KeepResultsForAtLeast(long duration, TimeUnit timeUnit)
=> base.KeepResultsForAtLeast(duration, timeUnit).JavaCast<Builder>();

public Builder KeepResultsForAtLeast(TimeSpan duration)
=> base.KeepResultsForAtLeast((long)duration.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();


public new Builder SetBackoffCriteria(BackoffPolicy policy, long backoffDelay,
TimeUnit timeUnit)
=> base.SetBackoffCriteria(policy, backoffDelay, timeUnit).JavaCast<Builder>();

public Builder SetBackoffCriteria(BackoffPolicy policy, TimeSpan backoffDelay)
=> base.SetBackoffCriteria(policy, (long)backoffDelay.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();

public new Builder SetConstraints(Constraints constraints)
=> base.SetConstraints(constraints).JavaCast<Builder>();
```

```
public new Builder SetInitialRunAttemptCount(int runAttemptCount)
=> base.SetInitialRunAttemptCount(runAttemptCount).JavaCast<Builder>();

public new Builder SetInitialState(WorkInfo.State state)
=> base.SetInitialState(state).JavaCast<Builder>();

public new Builder SetInputData(Data data)
=> base.SetInputData(data).JavaCast<Builder>();

public new Builder SetPeriodStartTime(long periodStartTime, TimeUnit timeUnit)
=> base.SetPeriodStartTime(periodStartTime, timeUnit).JavaCast<Builder>();

public Builder SetPeriodStartTime(TimeSpan periodStartTime)
=> base.SetPeriodStartTime((long)periodStartTime.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();

public new Builder SetScheduleRequestedAt(long scheduleRequestedAt, TimeUnit
timeUnit)
=> base.SetPeriodStartTime(scheduleRequestedAt, timeUnit).JavaCast<Builder>();

public Builder SetScheduleRequestedAt(TimeSpan scheduleRequestedAt)
=> base.SetPeriodStartTime((long)scheduleRequestedAt.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();
#endregion

}
}

public partial class Constraints
{
public partial class Builder
{
public Builder AddContentUriTrigger(System.Uri uri, bool triggerForDescendants)
=> this.AddContentUriTrigger(Android.Net.Uri.Parse(uri.OriginalString),
triggerForDescendants).JavaCast<Builder>();

public Builder SetTriggerContentMaxDelay(TimeSpan duration)
=> this.SetTriggerContentMaxDelay((long)duration.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();
public Builder SetTriggerContentUpdateDelay(TimeSpan duration)
=> this.SetTriggerContentUpdateDelay((long)duration.TotalMilliseconds,
TimeUnit.Milliseconds).JavaCast<Builder>();
}
}
}
```