Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра _____Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____
_____Система біометричної автентифікації з оптимізованою реалізацією методів Віоли-Джонса на основі мікрокомп'ютеру Raspberry Pi 4B_____
_____
(тема)

Виконав:
студент _2_ курсу, групи ___*СКСм-20-1*_____
_____Ткаченко Д. О._____
(прізвище, ініціали)
Спеціальність _____
_____123 Комп'ютерна інженерія_____
(код і повна назва спеціальності)
Тип програми ___освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _____Спеціалізовані_____
___комп'ютерні системи_____
(повна назва освітньої програми)
Керівник _____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва)

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

Освітня програма Спеціалізовані комп'ютерні системи
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)

«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Ткаченку Даниїлу Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Система біометричної автентифікації з оптимізованою реалізацією методів Віоли-Джонса на основі мікрокомп'ютеру Raspberry Pi 4B
затверджена наказом по університету від "04" 11 2021 р. № 1635 Ст.
2. Термін подання студентом роботи (проекту) 15.12.2021
3. Вихідні дані до роботи (проекту)
Мова програмування C++,

Мікрокомп'ютер Raspberry Pi 4B

IDE QTCreator
4. Зміст пояснювальної записки (перелік питань, що потрібно розробити)
Аналіз предметної галузі та постановка задачі проектування.

Розробка структури стенду

Реалізація демонстраційного програмного забезпечення

Тестування програмного забезпечення

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

_____ 15 слайдів_____

_____

_____

_____

_____

_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	25.08.2021 -01.09.2021	
2	Аналіз предметної області	01.09.2021 -10.05.2021	
3	Аналіз джерел з проблемної галузі	10.09.2021 -17.05.2021	
4	Розробка схеми макету	18.09.2021 -25.05.2021	
5	Написання демонтсраційного програмного	25.09.2021 -30.05.2021	
6	Оформлення пояснювальної записки	31.09.2021 -05.10.2021	
7	Оформлення графічного матеріалу	05.10.2021 -07.10.2021	

Дата видачі завдання _____25.08.2021_____

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи містить 62с., 18 рис., 3 табл., 3 дод. ,15 джерел.

АУТЕНТИФІКАЦІЯ, ІДЕНТИФІКАЦІЯ, МЕТОД ВІОЛИ-ДЖОНСА, RASPBERRY PI 4B

Задачею кваліфікаційної роботи є аналіз існуючих реалізацій методів Віоли-Джонса та створення максимально оптимізованих програмних реалізацій цих методів для використовування їх на базі мікрокомп’ютера по типу Raspberry Pi 4B

Метою проектування є створення програмного забезпечення для біометричної автентифікації з використанням оптимізованих методів Віоли-Джонса, використовуючи та розширяючи систему багатофакторної біометричної автентифікації, на основі платформи Raspberry Pi 4B.

Результатом кваліфікаційної роботи є модульна система для багатофакторної біометричної автентифікації яка використовує оптимізовані методи Віоли-Джонса, результати вимірювань часової різниці між методами які були реалізовані у контексті цієї роботи та реалізацій опублікованих в відкритій бібліотеці OpenCV.

ABSTRACT

Explanatory note: 62 pages, 18 figures, 3 appendices ,15 sources.

AUTHENTICATION, IDENTIFICATION, IRIS, RASPBERRY PI 4B

The task of the finale project is to analyze the existing implementations of Viola-Jones algorithm and create the most optimized software implementations of these methods for use with microcomputers such as Raspberry Pi 4B

The aim of this project is to create software for biometric authentication using custom Viola-Jones method implementation, using and extending the multifactor biometric authentication system, based on the Raspberry Pi 4B platform.

The result of the this project is a modular system for multifactor biometric authentication that uses custom Viola-Jones algorithm implementation, the results of time difference measurements between methods that have been implemented in the context of this project and implementations published in the open library OpenCV.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ББІ – Багатофакторна біометрична ідентифікація

БХЛ – біологічних характеристиках людини

FRR – - False Rejection Rate

Скріншот – зображення, отримане комп'ютером, що відображає те, що бачить користувач на екрані монітора.

FAR – False Acceptance Rate

MITM – “Man in the middle” attack

Оклюзія – ситуація, в якій два об’єкти розташовані приблизно на одній лінії і один об’єкт, розташований ближче до віртуальної камери або вікна перегляду, частково або повністю закриває видимість іншого об'єкта.

OpenCV – бібліотека комп’ютерного зору з відкритим кодом.

Tracking–відстеження об’єкта на відео послідовності.

ROI – Region of interest

ВСТУП

Біометрична автентифікація безупинно набирає обороти і розвивається на ринку систем захисту персональної інформації та домівок. Деякими прикладами можуть служити системи захисту у банківських установах, приватних компаніях, а також в аеропортах, вокзалах, метро, та других систем, робота яких заснована на обробці біометричних даних для прийняття різних рішень (рисунок 1). У системах контролю та управління доступом, які використовуються для отримання доступу до заповідної території, використання біометричних даних може підвищити рівень безпеки заповідних територій, а також зручність отримання доступу до заповідної території. У системах обліку робочого часу, що використовуються для підтвердження часу роботи персоналу на місці, використання біометричних даних дозволяє підвищити достовірність обліку робочого часу на підприємстві, а також підвищити дисципліну. У системах відеоспостереження з функцією розпізнавання, які використовуються для ідентифікації людей у громадських місцях за певною базою даних, використання біометричних даних дозволить підрахувати кількість людей у громадських місцях, а також надасть можливість виявлення злочинних елементів, зменшить ризик заворушень і терористичних актів.

Рисунок 1.1 – Області застосування біометричних технологій

Описані системи використовують методи біометричної верифікації та ідентифікації людини на основі унікальних біологічних характеристик людини (BHL), які можна розділити на дві великі підгрупи - статичні (відбитки пальців, геометрія обличчя, геометрія долоні, малюнок вен руки, сітківка, ДНК), динамічні (підпис, голос, хода, швидкість і сила набору тексту на клавіатурі, почерк, пульс) (рисунок 1.1)

Описані системи використовують методи біометричної верифікації та ідентифікації людини, засновані на унікальних біологічних характеристиках людини (БХЛ), які можна виділити в дві великі підгрупи - статичні (відбитки пальців, геометрія особи, геометрія долоні, малюнок вен руки, сітківка ока, ДНК), динамічні (підпис, голос, хода, швидкість і сила набору на клавіатурі, почерк, серцевий ритм) (рис. 1.2).

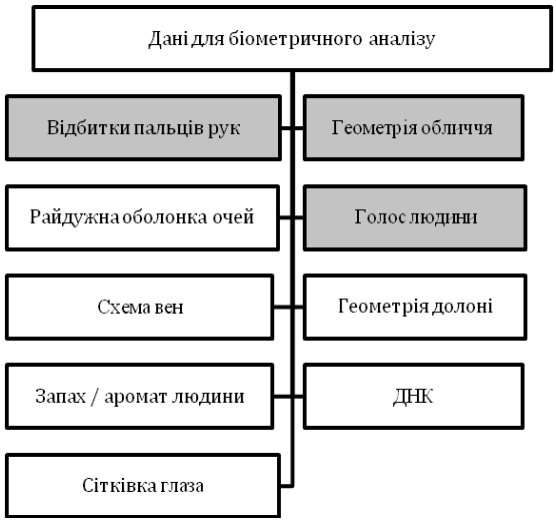

Рисунок 1.2 – Використовувані БХЧ

Основним вимогами до систем ББІ є:

– надійність (узагальнює показники, які показують, наскільки легко обдурити біометричний ідентифікатор (стійкість до підробки), наскільки стабільна система в різних зовнішніх умовах, наприклад, зміни освітлення або кімнатної температури (стійкість до навколишнього середовища);

– швидкість отримання результату;

– ненасильницький метод отримання даних для аналізу (наскільки складно скористатися біометричним датчиком).

До додаткових вимог можна віднести:

– відносна дешевизна;

– універсальність системи;

– компактність / портативність.

Для оцінки методів ідентифікації, заснованих на описаному ВСН, поряд з вимогою простоти отримання ідентифікаційних даних використовується вимога надійності біометричної системи, що виражається в помилках першого роду. (FRR – False Rejection Rate – пропуск цілі – показує ймовірність відмови доступу людині, що має допуск – система не пропускає легального користувача) і помилки другого роду (FAR – False Acceptance Rate – помилкова тривога – показує ймовірність помилкового збігу біометричних характеристик двох людей – система помиляється і приймає одну людину за ін.) (таблиця 1.1).

Таблиця 1.1 – Оцінки методів ідентифікації, заснованих на наведених БХЛ

Біометрична ознака	Точність		Простота використання
	FRR	FAR	
Геометрія обличчя	близько 6%	близько 0,1%	Висока
Відбиток пальця	близько 1%	близько 0,00002%	Середня
Голос	близько 3%	близько 0,1%	Висока
Райдужна оболонка ока	близько 0,2%	близько 0,0001%	Середня
Схема вен	близько 0,01%	близько 0,00008%	Середня

До суб'єктивних характеристик вибору факторів створення біометричної системи ідентифікації можна віднести наступне. Розглянемо їх більш детально.

Для методу ідентифікації відбитків пальців – низька вартість пристроїв, які сканують зображення відбитка пальця; проста процедура сканування відбитків пальців. Недоліки – лінії папілярного малюнка легко пошкоджуються від дрібних подряпин, порізів, впливу хімічних реагентів.

Для методу ідентифікації по геометрії особи – можливість проводити приховану зйомку осіб в людних місцях; можливе використання недорогого обладнання (проте, можливість розпізнавання на значних відстанях від камери забезпечують лише дорогі аналоги).

Недоліки – високі вимоги, що пред'являються до висвітлення; вплив зовнішніх перешкод – наявність окулярів, бороди, зміни зачіски, поворот

голови, часта зміна міміки – зменшують точність ідентифікації; низька достовірність роботи при класифікації близнюків.

Для методу ідентифікації по голосу – простота отримання ідентифікаційних даних, а також низька вартість датчиків.

Недоліки – мінливість ідентифікатора (голосу) у часі (наприклад, зміни голосу при простудних захворюваннях); нездатність ідентифікувати тупих людей. У роботі [7] показано, що системи, які використовують лише біометричну ознаку (наприклад, геометрію обличчя) для ідентифікації особи, не забезпечують високої ймовірності ідентифікації.

Там, де потрібна висока надійність, використовується комбінація кількох біометричних факторів – багатофакторна біометрична ідентифікація (MBI) людини. Прикладом використання таких систем є аналіз поведінки людини [8].

Метою даної роботи є розробка системи багатофакторної біометричної аутентифікації за допомогою геометрії обличчя, відбитків пальців та запису голосу з використанням мікрокомп’ютера Raspberry Pi 4B та бібліотеки комп’ютерного зору OpenCV.

1 ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ РОБОТИ

Надійність ідентифікації людини є найважливішою вимогою до систем ідентифікації, яка не може бути досягнута шляхом однофакторної перевірки, як показано в [9]. Тимчасове пошкодження або недоступність зчитування деяких біометричних ідентифікаторів також є підставою для переходу на міжбіометричні системи або змішані типи аутентифікації в системах ІВІ, які дозволяють генерувати ідентифікатор та перевіряти особу авторизованого користувача шляхом об’єднання результатів ідентифікації. актуальність запропонованого в роботі методу МБІ на основі аналізу відповідності статичного набору таких факторів, які представляє користувач, як відбитки пальців, геометрія обличчя, голос ідентифікованої особи для визначення його автентичності. Фактори, що використовуються в запропонованій системі, є найбільш доступним джерелом біометричних ідентифікаційних даних, а також забезпечують відповідність таким вимогам до систем ідентифікації та аутентифікації, як ненасильницький збір даних для аналізу, відносна дешевизна, компактність/переносність.

Оптимізація алгоритму Віоли-Джонса має під собою те що ці алгоритми достатньо вимогливі у плані продуктивності апарату який виконує обчислення що є підставою для того щоб за можливістю максимально оптимізувати обчислення при роботі з, наприклад, такими мікрокомп’ютерами як Raspberry Pi

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

2.1 Постановка задачі

Метою роботи є розробка системи багатофакторної біометричної ідентифікації з застосуванням оптимізованої реалізації методів Віоли-Джонса. Для досягнення поставленої мети повинні бути вирішені наступні завдання:

- аналіз основних методів для побудови системи біометричної ідентифікації,
- порівняльний аналіз існуючих систем біометричної ідентифікації,
- розробка системи біометричної ідентифікації людини.

2.2 Аналіз існуючих рішень

Існуючі рішення на сьогоднішній день використовують біометричні ознаки окремо та мають за основний тип автентифікації пароль або ключ-карту, такі апарати не використовують у місцях з високим рівнем захисту, так як біометричні ознаки є лише вторинним засобом автентифікації, ще таки апарати є доволі дорогі що не дає можливості використовувати їх повсюдно

Рисунок 2.1 – Приклад біометричної системи ZKTeco ProFace X

Такі системи як зображено на рисунку 2 мають дуже просунуті методи організації автентифікації, наприклад показаний девайс має 2 камери для отримання 3Д зображення що не дає можливості обманути сканер фотографією працівника.

Зважаючи на увесь функціонал цього девайсу він є доволі дорогим, ціна може досягати двох тисяч американських доларів, також таку систему не можна встановити у місцях підвищеної безпеки, так як основний модуль знаходиться зі сторони двері тому є можливість силового втручання, так як замки в таких системах використовують нормально відкриті реле, які можуть бути відкриті за допомогою з’єднання окремих контактів.

Запропоноване рішення цього проекту використовує декілька біометричних факторів для створення рішення що надає достатній рівень захисту для використовування у захищених місцях, наприклад банках або воєнних об’єктів, також за дизайном, ця система буде знаходитися у захищеному приміщені а назовні будуть виведені тільки апарати для зчитування біометричних ознак, що не дає можливість фізичного втручання у систему розпізнавання.

2.3 Загальні відомості про використану ЭОМ

Основою проекту є одноплатний комп’ютер Raspberry Pi 4B.

Рисунок 2.2 – Одно-платний комп’ютер Raspberry Pi 4B

Raspberry Pi (читається як Ра́збері па́й) — одноплатний комп'ютер, розроблений британським фондом Raspberry Pi. Його основна мета – стимулювати вивчення в школах основ інформатики.

Raspberry Pi використовує у своїй базі кристал Broadcom BCM2835, який включає процесор ARM за тактовою частотою у 700 МГц, графічний процесор VideoCore 5 та 512 або 256 мегабайт оперативної пам’яті. Жорсткого диска немає, замість нього використовується SD-карта. За допомогою цього обладнання ви можете відтворювати відео H.264 з роздільною здатністю 1080p і запускати комп’ютерні ігри, як-от Quake III Arena.

Організатором проекту Raspberry Pi є фонд Raspberry Pi Foundation. Комп’ютер був розроблений як пристрій для програмування для дітей, але він набув популярності і в інших сферах – зокрема, він використовується для створення домашніх медіацентрів. Найдешевший Raspberry Pi поставляється без футляра і виглядає як картка розміром з кредитну картку. Вага плати становить коло 45 грамів, використана система на кристалі використовує архиітектуру ARM; На платі розташовані роз’єм для навушників та слот для карти памяті. Існує ді варіації плат Raspberry Pi, молодша(A) та старша (B), вони відрізняються кількістю портів юсб, наявністю Ethernet порту, також відрізняються і кількість оперативної пам’яті. Крім того, старша модель має підключення 10/100 Ethernet, а молодша споживає на третину менше енергії.

Ранні конструкції плат Raspberry Pi Model A і B включали лише 256 МБ оперативної пам’яті (RAM). З цього, ранні бета-версії плат Model B за замовчуванням виділяли 128 МБ для графічного процесора, залишаючи лише 128 МБ для ЦП. На перших 256-мегабайтних випусках моделей A і B були можливі три різні розділення. Розподіл за замовчуванням склав 192 МБ для ЦП, що повинно бути достатньо для автономного декодування відео 1080p або для простої обробки 3D. 224 МБ було лише для обробки Linux, лише з буфером кадру 1080p, і, ймовірно, не вдалось для будь-якого відео або 3D. 128 МБ було

призначено для важкої обробки 3D, можливо, також з декодуванням відео. Для порівняння, Nokia 701 використовує 128 МБ для Broadcom VideoCore IV.

Пізніша модель B з 512 МБ ОЗУ була випущена 15 жовтня 2012 р. і спочатку була випущена з новими стандартними файлами розділеної пам'яті з 256 МБ, 384 МБ і 496 МБ CPU. , і з 256 МБ, 128 МБ та 16 МБ відеопам’яті відповідно. Але приблизно через тиждень фонд випустив нову версію start.elf, яка могла читати новий запис у config.txt і могла динамічно призначати обсяг оперативної пам’яті (від 16 до 256 МБ за кроком 8 МБ). до графічного процесора, застарівши старий метод розділення пам'яті, і один start.elf працював так само для 256 МБ і 512 МБ Raspberry Pis.

Raspberry Pi 2 має 1 ГБ оперативної пам’яті.

Raspberry Pi 3 має 1 Гб оперативної пам’яті в моделях B і B+ і 512 МБ оперативної пам’яті в моделі A+. Raspberry Pi Zero і Zero W мають 512 МБ оперативної пам’яті.

Raspberry Pi 4 доступний з 2, 4 або 8 ГБ оперативної пам’яті. Модель 1 ГБ спочатку була доступна при запуску в червні 2019 року, але була припинена в березні 2020 року, а модель з 8 ГБ була представлена в травні 2020 року.

Raspberry Pi 4 Model B був випущений в червні 2019 року з 1,5 ГГц 64-розрядним чотириядерним процесором ARM Cortex-A72, вбудованим 802.11ac Wi-Fi, Bluetooth 5, повним гігабітним Ethernet (пропускна здатність не обмежена), двома Порти USB 2.0, два порти USB 3.0, 2-8 ГБ оперативної пам’яті та підтримка двох моніторів через пару портів micro HDMI (HDMI Type D) для роздільної здатності до 4K. Від версії з 1 ГБ оперативної пам’яті відмовилися, а ціни на версію з 2 ГБ знижені. Версія на 8 ГБ має перероблену друковану плату. Pi 4 також живиться через порт USB-C, що дає можливість додаткового живлення периферійним пристроям, що знаходяться на нижньому напрямку, при використанні з відповідним блоком живлення. Але Pi може працювати лише від 5 вольт, а не від 9 чи 12 вольт, як інші міні-комп’ютери цього класу. Початкова плата Raspberry Pi 4 має недолік дизайну, коли сторонні

USB-кабелі з електронною позначкою, такі як ті, що використовуються на Apple MacBook, неправильно ідентифікують її та відмовляються забезпечити живлення. У середині 2021 року з'явилися моделі Pi 4 B з покращеним Broadcom BCM2711C0. Зараз виробник використовує цей чіп для Pi 4 B і Pi 400. Однак частота прихватки Pi 4 B не була збільшена на заводі.

Зручною особливістю мікрокомп'ютера Raspberry Pi є можливість роботи повноцінної операційної системи на базі Linux на пристрої з великою кількістю GPIO.

Linux має стандартизований і досить простий інтерфейс для взаємодії з пристроями за допомогою стандартних шин, таких як перераховані вище. Такі пристрої представлені у вигляді файлів пристроїв у каталозі /dev/ і використовуються з використанням бібліотек Linux, які є стандартними і мають велику кількість документації та прикладів використання.

Як камера для збору вхідних даних була використана веб-камера Logitech з характеристиками:

- роздільна здатність зображення 720p;
- роздільна здатність матриці 15mp;
- кількість кадрів в сек. 30 fps.
- напруга 5V.

2.4 Огляд архітектури ARM

Одною з цілей цього проекту було створити максимально дешеву, при цьому максимально продуктивну систему, для досягнення цієї цілі було обрано мікрокомп'ютер який у своїй базі використовує процесор на архітектурі ARM

Архітектура ARM (від англ. Advanced RISC Machine – покращена машина RISC; іноді – Acorn RISC Machine) – це система інструкцій і сімейство описів і готових топологій 32-розрядних і 64-розрядних ядер мікропроцесора/ мікроконтролера, розроблених ARM Limited.

На конкретно обраному мікрокомп’ютері використовується ARM CortexA71

ARM Cortex-A7 MPCore – це ядро процесора, розроблене ARM Holdings і реалізує архітектуру ARM v7. Він був анонсований

у жовтні 2011 року на ARM TechCon і має кодову назву Cortex-A7 «Kingfisher».

Ядро мало дві основні цілі дизайну: перш за все, стати швидшим, більш енергоефективним і меншим за розміром замінником Cortex A8.

Друге завдання — використовувати в рішеннях архітектуру big.LITTLE, яка об’єднує одне або кілька ядер Cortex A7 з одним або кількома ядрами Cortex A15 в гетерогенній обчислювальній системі. Для цього ядро було розроблено, щоб бути повністю архітектурно сумісним із Cortex A15.

Ключові особливості:

- частково суперскалярний двосторонній (для цілочисельних операцій), регулярний 8-ступінчастий обчислювальний конвеєр з динамічним прогнозуванням розгалужень;
- інтегрований SIMD NEON (з 64-розрядним ALU);
- блок з плаваючою точкою VFPv4;
- набір інструкцій Thumb-2;
- Jazelle RCT;
- апаратна віртуалізація;
- великі розширення адреси сторінки (LPAE);
- інтегрований кеш-пам'ять L2 (0-1 МБ);
- 1,9 DMIPS / МГц [2];
- підтримує контролер переривань GIC-400 і когерентну шину кешу CCI-400.

2.5 Опис обраного датчика відбитків пальцю

Обираючи датчик відбитку пальця можуть бути декілька варіантів обробки вхідних даних:

- перетворення відбитка на цифровий код за допомогою оптичного сенсора;
- перетворення відбитка за допомогою лінійного теплового датчика;
- перетворення відбитка за допомогою ємнісного датчика;
- перетворення відбитка за допомогою датчика тиску;
- перетворення відбитка за допомогою генератора ультразвуку.

Датчики з оптичним сенсором мають багато реалізацій такі як:

- FTIR-сканери;
- оптоволоконні сканери;
- електрооптичні сканери;
- оптичні протяжні сканери;
- роликові сканери;
- безконтактні сканери.

Рисунок 2.3 – Оптичний сканер відбитку пальця

Ємнісні сканери (capacitive scanners) — найпоширеніший тип напівпровідникових сканерів.

Ці скаенри використовують ефект зміни ємності пн-переходу при зіткненні поверхні пальця з елементом матриці, також існують модифікації, у яких кожен напівпровідниковий елемент у матриці сканера виступає у ролі однієї пластини конденсатора, а палець – у ролі інший. При додатку пальця до сенсора між кожним чутливим елементом та виступом-впадиною папілярного візерунка утворюється певна ємність, величина якої визначається відстанню між поверхнею пальця та елементом. Матриця цих ємностей перетворюється на зображення відбитка пальця.

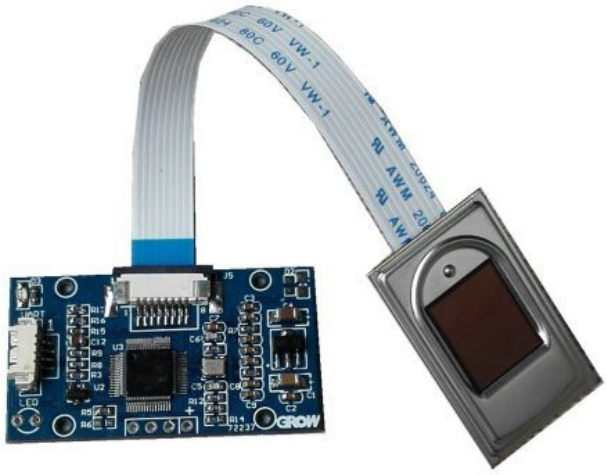

Рисунок 2.4 – Ємнісний сканер відбитку пальця

Чутливі до тиску сканери (pressure scanners) – такі сканери використовують сенсори які складаються з матриці п’єзоелементів, які у свою чергу, при прикладанні палцю до матриці, зчитують тиск гребенця папілярного візерунка та отримують зображення вібдитка, тільки деяка підмножина елементів доторкається поверхні, відповідно западини ніякого тиску не надають. Матриця отриманих з п'єзоелементів напруг перетворюється на зображення поверхні пальця.

Термо-сканери - це сканери які використовують сенсори які побудовані з використанням піроелектричних елементів які фіксують різницю температури на матриці та перетворюють її на напругу. При прикладанні пальця до сканеру, гребенець торкається піроелектричних елементів і так як під виступами темпрература відрізняється то виступи гребенця формують картину відбитка

Ультразвукові сканери – це сканери, які сканують поверхню пальця ультразвуковими хвилями і за допомогою відбитого ними відлуння вимірюють відстань між джерелом хвилі та западинами і виступами на поверхні пальця. Якість зображення, отриманого таким чином, в десятки разів краще, ніж за допомогою будь-якого іншого методу, доступного на біометричному ринку. Крім того, слід зазначити, що цей метод практично повністю захищений від манекенів, оскільки дає можливість отримати деякі додаткові характеристики про їх стан, крім відбитка пальця. (наприклад, пульс всередині пальця).

Для проекту було обрано сканер оптичного FPM10A типу як найкращій у ціні та характеристикам. Даний сканер використовує протокол UART для з’єднання з управляючою частиною(мікрокомп’ютером).

2.6 Опис протоколу UART

Універсальний асинхронний приймач-передавач (UART) — це комп’ютерний апаратний пристрій для асинхронного послідовного зв’язку, в якому можна налаштувати формат даних і швидкість передачі. Він посилає біти даних один за одним, від найменшого значущого до найбільш значущого, обрамленого початковим і кінцевим бітами, щоб канал зв’язку обробляв точний час. Рівні електричної сигналізації обробляються схемою драйвера, зовнішньою по відношенню до UART. Два поширених рівня сигналу: RS-232, 12-вольтова система, і RS-485, 5-вольтова система. Ранні телетайпи використовували струмові петлі.

UART зазвичай містить такі компоненти:

- генератор тактової частоти, зазвичай кратний бітрейту, щоб дозволити вибірку в середині періоду розряду;
- вхідні та вихідні регістри зсуву;
- контроль передачі/приймання;
- логіка управління читанням/записом;
- автоматичне вимірювання швидкості (необов’язково);
- буфери передачі/приймання (необов'язково);
- буфер системної шини даних (необов'язково);
- буферна пам'ять "перший прийшов, перший вийшов" (FIFO) (необов'язково);
- сигнали, необхідні сторонньому контролеру DMA (необов'язково);
- інтегрований контролер DMA для керування шиною (опціонально).

Універсальний асинхронний приймач-передавач (UART) приймає байти даних і передає окремі біти послідовно. У пункті призначення другий UART повторно збирає біти в повні байти. Кожен UART містить регістр зсуву, який є основним методом перетворення між послідовними і паралельними формами. Послідовна передача цифрової інформації (бітів) через один провід або інший носій є менш витратною, ніж паралельна передача через кілька проводів.

UART зазвичай безпосередньо не генерує і не приймає зовнішні сигнали, які використовуються між різними елементами обладнання. Окремі інтерфейсні пристрої використовуються для перетворення сигналів логічного рівня UART в і з зовнішніх рівнів сигналізації, якими можуть бути стандартизовані рівні напруги, рівні струму або інші сигнали.

Зв'язок може бути симплексним (тільки в одному напрямку, без можливості приймального пристрою надсилати інформацію назад до пристрою, що передає), повним дуплексом (обидва пристрої надсилають і отримують одночасно) або напівдуплексним (пристрої по черзі передають та отримують).

2.7 Вимоги до проекту

Вимоги до системи багатофакторної багатофакторної біометричної автентифікації з оптимізованою реалізацією алгоритму Віоли-Джонса:

- використання мікрокомп’ютера Raspberry Pi 4B;
- використання щонайменше трьох біометричних факторів;
- наявність графічного інтерфейсу користувача для комфортної роботи з програмним забезпеченням;
- своя реалізація алгоритму Віоли-Джонса, яка у пріоритеті повинна бути дещо швидшою за реалізацію OpenCV.

2.8 Задача проекту

Задачі, які необхідно вирішити, для реалізації системи багатофакторної біометричної автентифікації з оптимізованою реалізацією алгоритму Віоли-Джонса:

- Порівняння та вибір біометричних алгоритмів для роботи системи
- Порівняння та вибір девайсів для зчитування біометричних ознак
- Вибір сканеру відбитка пальця
- Вибір камери для зчитування фото обличчя
- Вибір мікрофону для зчитування голосу
- Зборка тестового стенду з усіма обраними модулями для зчитування біометричних ознак
- Створення програмної бази для демонстрації функціоналу
- Створення бібліотеки для роботи з UART інтерфейсом для роботи зі сканером відбитків пальцю
- Створення бібліотеки на основі бібліотеки OpenCV для роботи з камерою

- Створення бібліотеки для розпізнавання голосу на основі доступних бібліотек та рішень
- Створення алгоритму для отримання сумісного рішення від обробки декількох біометричних ознак
- Імплементація алгоритму Віоли-Джонса для розпізнавання обличь
- Оптимізація імплементованого алгоритму за допомогою багато-поточності та SIMD-інструкцій
- Створення тестів для перевірки правильної роботи програми
- Мануальне тестування програми

3 РОЗРОБКА КОМП’ЮТЕРНОЇ СИСТЕМИ

Основним завданням цього проекту є дослідження методів Віоли-Джонса для визначення осіб та інших об'єктів на зображенні, далі буде більш детально розглянуто алгоритм Віоли-Джонса і методи, які він використовує.

3.1 Огляд використаних алгоритмів

3.1.1 Алгоритм розпізнавання обличчя

Серед існуючих методів порівняння особин розглядалися такі: LBPH, Fisherfaces та Eigenfaces. Метод LBPH виявився найбільш гнучким (алгоритм адаптується до різних розмірів зображень без додаткових етапів попередньої обробки), найменш ресурсомістким і найшвидшим. Так само цей метод є досить гнучким, наприклад, для навчання на основі цього методу не потрібно вирізати обличчя за ROI, але він використовується, по-перше, для зменшення розміру файлів для зберігання в базі даних, а також для отримання більш однорідної гістограми. Приклад обрізання фотографії за ROI показано на (рис. 3.1).

Локальні двійкові шаблони (LBP) — це простий оператор, який використовується для класифікації текстур у комп’ютерному баченні. Вперше описано в 1996 році [1] LBS є описом піксельного околиці зображення у двійковій формі. Оператор LBS, який застосовується до пікселя зображення, використовує вісім пікселів сусідства, приймаючи центральний піксель як поріг. Пікселі, значення яких більше центрального пікселя (або рівне йому), приймають значення «1», ті, які менші за центр, приймають значення «0». Таким чином, виходить 8-розрядний двійковий код, який описує околиці пікселя, що описано на (рис. 3.1).

LBPH — це метод отримання інформації про зображення шляхом застосування операції LBP (логічні двійкові шаблони) до матриці на основі отриманого зображення. Після застосування цієї операції виходить нове зображення, що повністю складається з двійкових даних. Далі отримане

зображення розбивається на сегменти і будуються гістограми, які є його характеристиками.

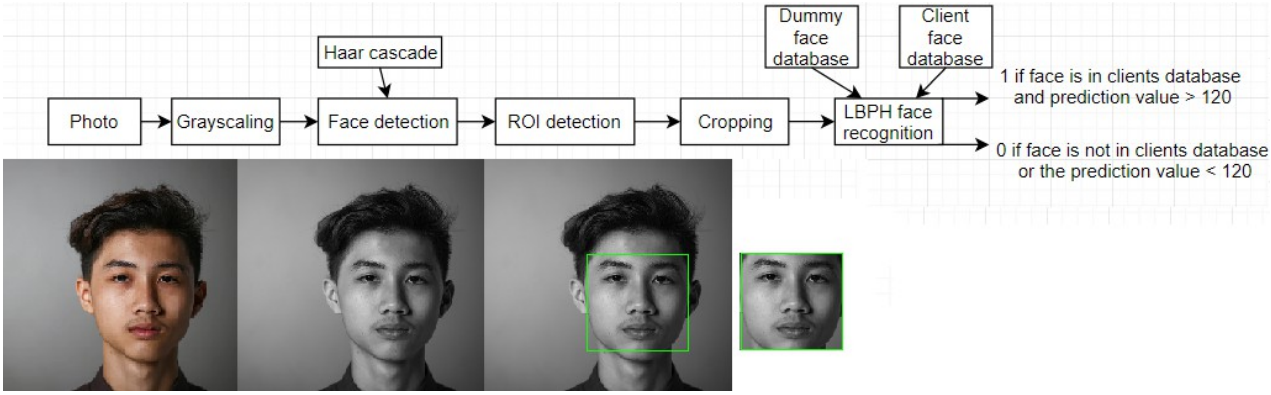

Рисунок 3.1 – Алгоритм, застосовуваний для ідентифікації за фотозображенням особи

Операція LPB полягає в отриманні двійкової матриці з десяти елементів. Для початку вибирається опорна комірка матриці. Далі значення цієї комірки порівнюється з іншими навколо неї. Радіус і кількість сусідніх комірок змінні і вибираються користувачем. У цьому прикладі таких комірок 8 нове значення для комірки отримується після операції. Способи його отримання різні, але результати не будуть критично відрізнятися. У цьому прикладі (рис. 3.2) після операції LBP з матриці виходить значення «1000 1101» = 141. Більш детальний опис цього алгоритму можна побачити в [4].

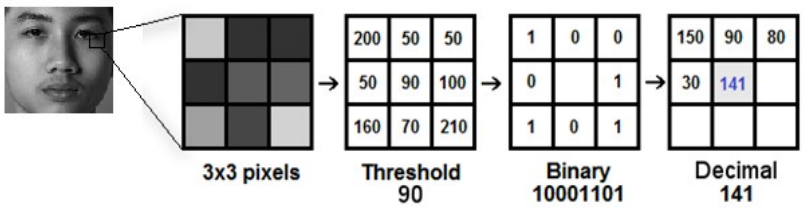

Рисунок 3.2 – Застосування LBP операції до зображення

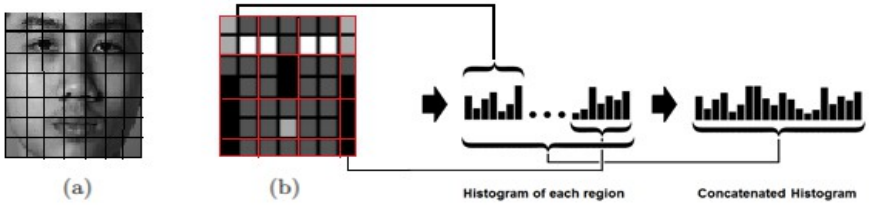

Рисунок 3.3 – Наслідок застосування LBP операції до зображення і отримання гістограм

3.1.2 Алгоритм розпізнавання відбитку пальця

Прийняття рішення на двовимірному напівтоновому зображенні папілярних ліній пальця, отриманому на вхід системи MBI, виконується за алгоритмом, наведеним на (рис. 3.4).

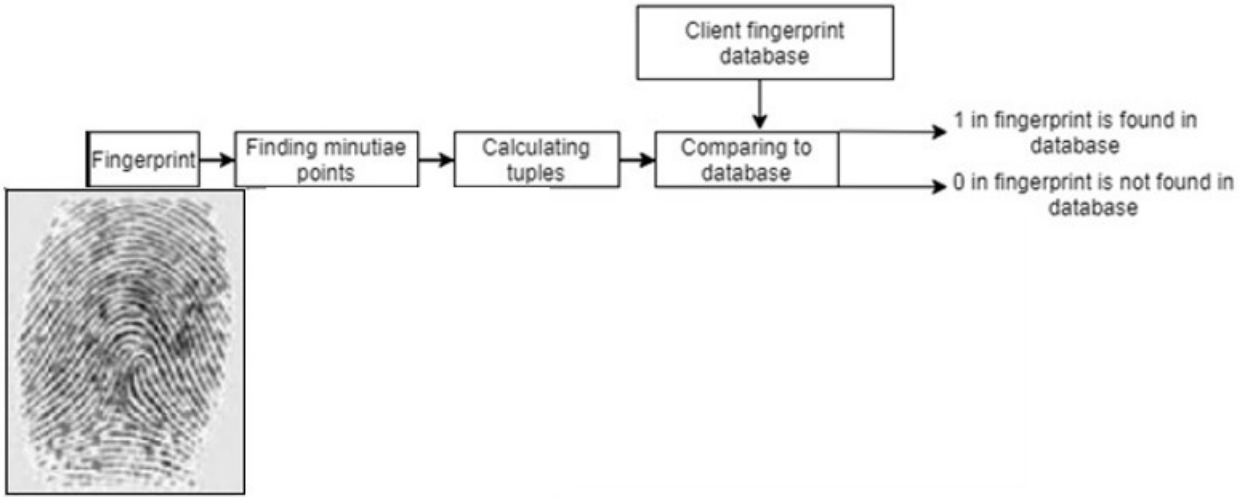

Рисунок 3.4 – Алгоритм, що застосовуються для ідентифікації по фотозображення папілярних ліній

У роботі використовується алгоритм порівняння точок деталізації (minutiae point), який полягає у знаходженні унікальних точок на зображенні, таких як скелі, окремі короткі лінії або гілки. Цей алгоритм має низьку складність і простоту реалізації, описані в [13].

3.1.3 Алгоритм розпізнавання голосу

Ухвалення рішення по голосовому відбитку, що надійшов на вхід системи МБІ, виконується відповідно до алгоритму, наведеному на рисунку3.5.

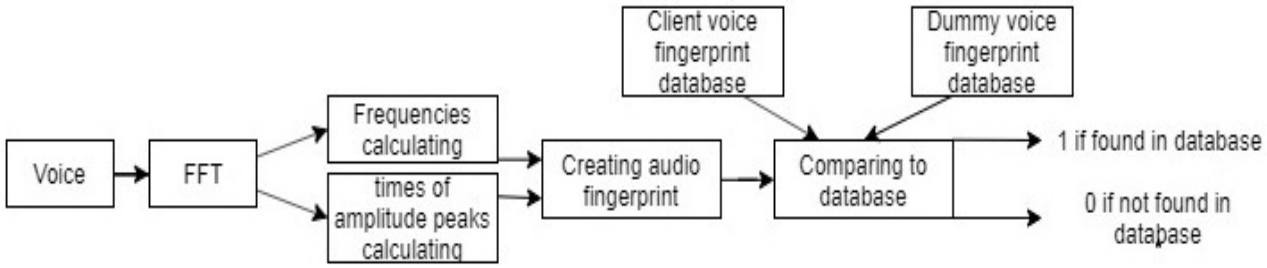

Рисунок 3.5 – Алгоритм, застосовуваний для ідентифікації по голосовому відбитку

Для голосової аутентифікації використовувався алгоритм, описаний у [14].

Алгоритм полягає в наступній послідовності кроків:

1. визначення активності голосу (VAD);
2. алгоритм довготривалої спектральної розбіжності (LTSD) для визначення реальних проміжків проголошення ключових слів.

Для створення вільної та вільно розповсюджуваної системи використовувався загальнодоступний інтерфейс CMU Sphinx, він забезпечує розпізнавання ключових слів, які вимовляє користувач, а також усуває необхідність підключення до мережі, що робить атаку MITM практично неможливою.

3.2 Огляд алгоритму Віоли-Джонса

Система виявлення об’єктів Віоли – Джонса - це система виявлення об’єктів, запропонована в 2001 році Полом Віолою та Майклом Джонсом. Незважаючи на те, що його можна навчити виявляти різноманітні класи об’єктів, це мотивувалось насамперед проблемою виявлення обличчя.

Проблема, яку потрібно вирішити, - виявлення граней на зображенні. Людина може зробити це легко, але комп’ютеру потрібні точні інструкції та обмеження. Щоб зробити завдання більш керованим, алгоритм Віоли – Джонса вимагає повного огляду фронтальних вертикальних граней. Таким чином, щоб

бути виявленим, все обличчя повинно бути спрямоване до камери і не повинно бути нахилене в будь-яку сторону. Хоча, здається, ці обмеження можуть дещо зменшити корисність алгоритму, оскільки за кроком виявлення найчастіше слідує крок розпізнавання, на практиці ці обмеження на позі цілком прийнятні.

Характеристиками алгоритму Віоли – Джонса, які роблять його хорошим алгоритмом виявлення, є:

- Надійна - дуже висока частота виявлення (істинно-позитивний показник) і дуже низька частота помилково-позитивних.
- У реальному часі - Для практичних застосувань потрібно обробляти щонайменше 2 кадри в секунду.
- Лише виявлення обличчя(не розпізнавання) - мета полягає в тому, щоб відрізнити обличчя від не обличь (виявлення - це перший крок у процесі розпізнавання).
- Алгоритм має чотири етапи:
- Вибір наборів ознак Хаара
- Створення цілісного образу
- Навчання Adaptive boost
- Каскадні класифікатори

Характеристики, які шукає система виявлення, загалом включають суми пікселів зображення в прямокутних областях. Як такі, вони мають деяку схожість з базовими функціями Хаара, які раніше використовувались у сфері виявлення об'єктів на основі зображень. Однак, оскільки всі функції, які використовують Віола та Джонс, залежать від кількох прямокутних областей, вони, як правило, є більш складними. Рисунок 5 ілюструє чотири різні типи функцій, що використовуються в рамках. Значення будь-якої даної ознаки - це сума пікселів у чітких прямокутниках, віднятих із суми пікселів у затінених прямокутниках. Прямокутні ознаки такого роду примітивні у порівнянні з такими альтернативами, як керовані фільтри. Незважаючи на те, що вони

чутливі до вертикальних та горизонтальних характеристик, їх відгуки значно грубші.

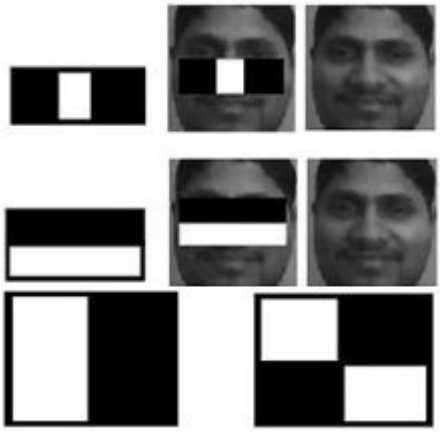

Рисунок 3.6 – Візуалізація ознак Хаара

3.3 Ознаки Хаара

Виявлення об’єктів з використанням каскадних класифікаторів на основі функцій Хаара є ефективним методом виявлення об’єктів, запропонованим Полом Віолою та Майклом Джонсом у їхній статті «Швидке виявлення об’єктів з використанням посиленого каскаду простих функцій»[7] у 2001 році. Це підхід, заснований на машинному навчанні, де каскадна функція тренується з великої кількості позитивних і негативних зображень. Потім він використовується для виявлення об’єктів на інших зображеннях.

Працюючи з розпізнаванням обличчя, алгоритм потребує багато позитивних зображень (зображень облич) і негативних зображень (зображень без облич) для навчання класифікатора. Потім нам потрібно витягти з нього особливості. Для цього використовуються функції Haar, показані на зображенні нижче. Вони подібні до згорткового ядра. Кожна функція — це одне значення, отримане шляхом віднімання суми пікселів під білим прямокутником із суми пікселів під чорним прямокутником.

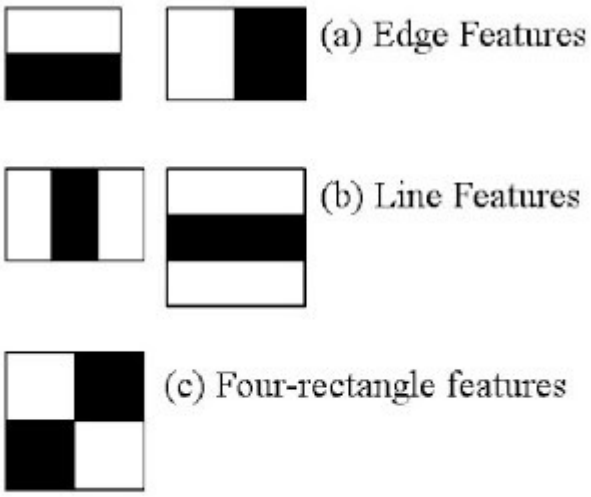

Рисунок 3.7 – Візуалізація ознак Хаара

Для всіх можливих розмірів та розташувань кожного ядра використовуються для обчислення багатьох функцій. Для кожного розрахунку функції потрібно знайти суму пікселів під білим і чорним прямокутниками. Щоб вирішити це, вони ввели цілісне зображення. Яким би великим не було зображення, воно зводить обчислення для даного пікселя до операції, яка включає лише чотири пікселі.

Але серед усіх розрахованих особливостей більшість з них не мають значення. Для прикладу розглянемо (рисунок 5). Верхній ряд показує дві хороші характеристики. Перша вибрана характеристика, здається, зосереджена на властивості, що область очей часто темніше, ніж область носа і щік. Друга обрана характеристика спирається на властивість, що очі темніше, ніж перенісся. Але ті ж вікна, нанесені на щоки або будь-яке інше місце, не мають значення.

Для цього застосовується кожна функція до всіх навчальних зображень. Для кожної характеристики алгоритм знаходить найкращий поріг, який буде класифікувати обличчя на позитивні та негативні. Очевидно, будуть помилки або неправильна класифікація. Вибираються об’єкти з мінімальною частотою помилок, що означає, що саме вони найточніше класифікують зображення обличчя та зображення без обличь.

Остаточний класифікатор є зваженою сумою цих слабких класифікаторів. Його називають слабким, оскільки він сам по собі не може класифікувати зображення, але разом з іншими утворює сильний класифікатор. У статті говориться, що навіть 200 функцій забезпечують виявлення з точністю 95%. Їх остаточна установка мала близько 6000 функцій.

3.4 Огляд задачі

Завдання цього проекту є реалізація алгоритму Віоли-Джонса і порівнювання цієї реалізації з реалізацією відкритої бібліотеки OpenCV, наприклад:

- Порівняння тимчасових на виконання алгоритму
- Порівнювання відмовостійкості на зіпсованих даних (наприклад шум зображення)

Для отримання найкращих результатів буде використані такі методи реалізації як використання SIMD інструкцій процесора що значно збільшує швидкодію, а також реалізація багато-поточної реалізації для оптимального використання усіх ресурсів процесору.

SIMD (англ. Single instruction, multiple data - одиночний потік команд, множинний потік даних) - принцип комп'ютерних обчислень, що дозволяє забезпечити паралелізм на рівні даних. Один з класів обчислювальних систем в класифікації Флінна.

SIMD-комп'ютери складаються з одного командного процесора (керуючого модуля), званого контролером, і декількох модулів обробки даних, званих процесорними елементами. Керуючий модуль приймає, аналізує і виконує команди. Якщо в команді зустрічаються дані, контролер розсилає на всі процесорні елементи команду, і ця команда виконується на декількох або на всіх процесорних елементах. Кожен процесорний елемент має свою власну пам'ять для зберігання даних.

Одним з переваг даної архітектури вважається те, що в цьому випадку більш ефективно реалізована логіка обчислень. До половини логічних

інструкцій звичайного процесора пов'язано з управлінням виконанням машинних команд, а решта їх частина відноситься до роботи з внутрішньою пам'яттю процесора і виконання арифметичних операцій. У SIMD-комп'ютері управління виконується контролером, а «арифметика» віддана процесорним елементам.

3.5 Огляд програмних компонентів розробки

Інтегроване середовище розробки набору компонентів, інструментів для розроблення програмного забезпечення. Типове середовище розробки містить:

редактор програмного коду;

компілятор та/або інтерпретатор;

засоби автоматизації збірки продукту;

відладчик.

Для розробки демонстраційного програмного забезпечення для плати було обрано інтегроване середовище розробки QTCreator (рис. 3.6).

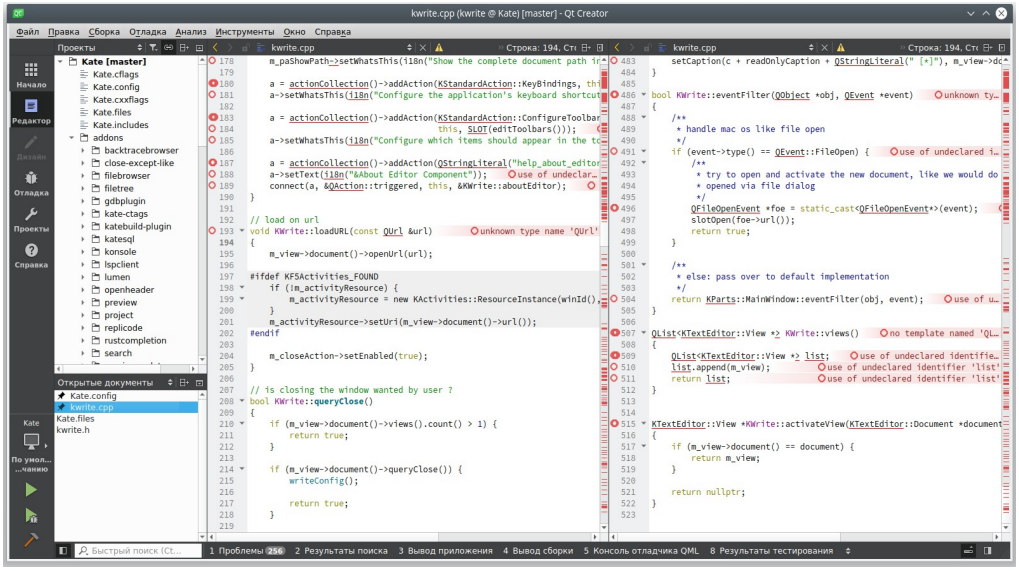

Рисунок 3.8 – Інтегроване середовище розробки QTCreator

QTCreator є інтегрованим середовищем розробки з підтримкою системи контроля версій Git. Підтримка мов програмування, засобів налагодження, додаткових інструментів профілювання здійснюється за допомогою розширень, що можуть бути встановлені з магазину додатків Microsoft. При розробці вбудованого програмного забезпечення на базі процесорів Cortex-M використовуються наступні розширення:

- C/C++ plugin;
- CMake;
- CMake Tools.

Розглянемо кожен плагін окремо. Плагін C / C ++, який додає підтримку мов програмування C і C ++. Підтримка включає автозаповнення програмного коду, засоби налагодження, засоби IntelliSense. CMake додає підтримку мови CMake, яка використовується для міжплатформної компіляції проекту. CMake Tools - додаткові інструменти CMake, які дозволяють налаштувати проект, завантажити прошивку на пристрій, запустити компіляцію проекту. Як основну мову програмування для реалізації проекту було обрано останній доступний стандарт мови програмування C++, а саме C++17.

C++ - це мова програмування загального призначення, яку можна використовувати для розробки додатків з обмеженими ресурсами або додатків з високими вимогами до продуктивності. Він був розроблений Бйорном Страуструпом у 1979 році як розширення мови програмування С С ++, об’єктно-орієнтованої мови програмування з низькою вартістю абстракцій мови програмування високого рівня. Таким чином, для розробки прошивки інтерфейс мов С і С++ є найбільш привабливим варіантом як за вартістю, так і з точки зору швидкості та гнучкості розробки.

3.5 Огляд інструменту CMake

У розробці програмного забезпечення CMake - це безкоштовне міжплатформне програмне забезпечення з відкритим кодом для автоматизації

складання, тестування, пакування та встановлення програмного забезпечення за допомогою незалежних від компілятора методів. CMake не є системою збірки, а генерує файли збірки іншої системи. Він підтримує ієрархії каталогів і програми, які залежать від кількох бібліотек. Він використовується в поєднанні з рідними середовищами збірки, такими як Make, Qt Creator, Ninja, Android Studio, Apple Xcode і Microsoft Visual Studio. Він має мінімальні залежності, вимагає лише компілятор C++ на власній системі збірки

CMake може генерувати файли проекту для кількох популярних середовищ IDE, таких як Microsoft Visual Studio, Xcode та Eclipse CDT. Він також може створювати сценарії збірки для MSBuild або NMake в Windows; Unix Make на Unix-подібних платформах, таких як Linux, macOS та Cygwin; і Ninja на Windows і Unix-подібних платформах.

Складання програми або бібліотеки за допомогою CMake є двоетапним процесом. По-перше, стандартні файли збірки створюються (генеруються) з файлів конфігурації (CMakeLists.txt), які написані мовою CMake. Потім власні інструменти збірки платформи (нативний ланцюг інструментів) використовуються для фактичного створення програм.

Файли збірки налаштовуються залежно від використовуваного генератора (наприклад, Unix Makefiles для make). Досвідчені користувачі також можуть створювати та включати додаткові генератори make-файлів для підтримки своїх конкретних потреб компілятора та ОС. Згенеровані файли зазвичай розміщуються (за допомогою прапорця cmake) у папку поза вихідною (поза вихідною збіркою), наприклад, build/.

Кожен проект збірки, у свою чергу, містить файл CMakeCache.txt і каталог CMakeFiles у кожному (під-)каталозі проектів (які раніше були включені командою add_subdirectory(...)), що допомагає уникнути або прискорити етап регенерації після його запуску. знову.

Після створення Makefile (або альтернативи) поведінку збірки можна точно налаштувати за допомогою цільових властивостей (починаючи з версії

3.1) або за допомогою глобальних змінних із префіксом CMAKE_.... Останнє не рекомендується для конфігурацій лише для цільових сторін, оскільки змінні також використовуються для налаштування самого CMake та для встановлення початкових значень за замовчуванням.

3.6 Огляд бібліотеки OpenCV

OpenCV (Open Source Computer Vision Library) — це бібліотека функцій програмування, в основному спрямованих на комп'ютерний зір у реальному часі. Спочатку розроблений Intel, пізніше він був підтриманий Willow Garage, а потім Itseez (який пізніше був придбаний Intel). Бібліотека є кросплатформною та безкоштовною для використання за ліцензією Apache 2 з відкритим кодом. Починаючи з 2011 року, OpenCV має прискорення графічного процесора для операцій у реальному часі.

Бібліотека надає інструменти для обробки та аналізу збереженого вмісту, включаючи розпізнавання об'єктів на фотографіях (наприклад, обличчя та фігури людей, текст тощо), відстеження руху об'єктів, перетворення зображень, застосування методів машинного навчання та визначення загальних елементів на різних зображення. Бібліотека розроблено Intel и ніні підтримується Willow Garage [en] та Itseez. Сирцевої код бібліотеки написаний мовою C ++ и поширюється під ліцензією BSD. Біндінгі підготовлені для різніх мов програмування, таких як Python, Java, Ruby, Matlab, Lua та других. Може вільно використовуватися в академічних та комерційних цілях.

Офіційно запущений у 1999 році проект OpenCV спочатку був ініціативою Intel Research для розвитку додатків із інтенсивним процесором, частиною серії проектів, включаючи трасування променів у реальному часі та 3D-дисплеї.[4] Основними учасниками проекту були ряд експертів з оптимізації в Intel в Росії, а також команда Intel Performance Library. На початку OpenCV цілі проекту були описані як:

- Розвивайте дослідження бачення, надаючи не тільки відкритий, але й оптимізований код для базової інфраструктури бачення. Більше не потрібно винаходити велосипед.
- Розповсюджуйте знання про бачення, надаючи спільну інфраструктуру, на якій розробники могли б розвивати, щоб код був легше читаним і переданим.
- Розвивайте комерційні додатки на основі бачення, роблячи портативний, оптимізований за продуктивністю код доступним безкоштовно – з ліцензією, яка не вимагає, щоб код був відкритим або вільним.Бібліотека містить понад 2500 оптимізованих алгоритмів, серед яких повний набір як класичних так і практичних алгоритмів машинного навчання і комп'ютерного зору.

Області застосування OpenCV включають:

- Набори інструментів 2D та 3D
- Оцінка егоемоцій
- Система розпізнавання облич
- Розпізнавання жестів
- Взаємодія людини з комп’ютером (HCI)
- Мобільна робототехніка
- Розуміння руху
- Виявлення об'єктів
- Сегментація та розпізнавання
- Стереопсисний зір: сприйняття глибини з 2 камер
- Структура від руху (SFM)
- Відстеження руху

- Доповнена реальність

3.7 Опис використаних компіляторів

Для реалізації проекту був використаний компілятор G++ з колекції компіляторів GCC

Колекція компіляторів GNU (GCC) — це оптимізуючий компілятор, створений проектом GNU, який підтримує різні мови програмування, апаратні архітектури та операційні системи. Фонд вільного програмного забезпечення (FSF) розповсюджує GCC як вільне програмне забезпечення під загальною суспільною ліцензією GNU (GNU GPL). GCC є ключовим компонентом набору інструментів GNU і стандартним компілятором для більшості проектів, пов’язаних з GNU та ядром Linux. Маючи приблизно 15 мільйонів рядків коду в 2019 році, GCC є однією з найбільших безкоштовних програм, які існують.[5] Вона відіграла важливу роль у розвитку вільного програмного забезпечення як інструмент і приклад.

Коли він був вперше випущений в 1987 році Річардом Столманом, GCC 1.0 був названий компілятором GNU C, оскільки він обробляв лише мову програмування C.[1] Його було розширено для компіляції C++ у грудні того ж року. Пізніше передні частини були розроблені для Objective-C, Objective-C++, Fortran, Ada, D і Go, серед інших. Специфікації OpenMP і OpenACC також підтримуються в компіляторах C і C++.

GCC був портований на більшу кількість платформ і архітектур набору інструкцій, ніж будь-який інший компілятор, і широко використовується як інструмент у розробці як безкоштовного, так і запатентованого програмного забезпечення. GCC також доступний для багатьох вбудованих систем, включаючи чіпи на основі ARM і Power ISA.

Крім того, що GCC є офіційним компілятором операційної системи GNU, він був прийнятий як стандартний компілятор багатьма іншими сучасними

операційними системами, подібними до Unix, включаючи більшість дистрибутивів Linux. Більшість операційних систем сімейства BSD також перейшли на GCC невдовзі після його випуску, хоча з тих пір FreeBSD, OpenBSD і Apple macOS перейшли на компілятор Clang, переважно через ліцензування. GCC також може компілювати код для Windows, Android, iOS, Solaris, HP-UX, AIX і DOS.

Також для крос-компіляції був використаний компілятор Clang який використовувався для компіляції під платформу Raspberry Pi.

Clang - це інтерфейс компілятора для мов програмування C, C++, Objective-C і Objective-C++, а також фреймворків OpenMP, OpenCL, RenderScript, CUDA і HIP[8]. . Він діє як додаткова заміна для колекції компіляторів GNU (GCC), підтримуючи більшість прапорів компіляції та неофіційних мовних розширень. Він включає в себе статичний аналізатор і кілька інструментів аналізу коду.

Clang працює в парі з серверною частиною компілятора LLVM і є підпроектом LLVM 2.6 і новіших версій.[12] Як і LLVM, це безкоштовне програмне забезпечення з відкритим кодом під ліцензією Apache License 2.0. Його учасниками є Apple, Microsoft, Google, ARM, Sony, Intel і AMD.

Clang 12, остання основна версія Clang станом на квітень 2021 року, має повну підтримку всіх опублікованих стандартів C++ аж до C++17, реалізує більшість функцій C++20 і додає початкову підтримку майбутнього стандарту C++23. Починаючи з версії 6.0.0, Clang за замовчуванням компілює C++, використовуючи діалект GNU++14, який включає функції стандарту C++14 і відповідні розширення GNU.

3.8 Структура програмної частини

Запропонована в роботі система БІ заснована на визначенні відповідності пред'явленого користувачем статичного набору трьох біометричних факторів (відбитки пальців рук, геометрія особи, голос ідентифікованого людини) наявного набору в базі даних системи, для визначення його автентичності. Узагальнене уявлення системи представлено на рисунку 7.

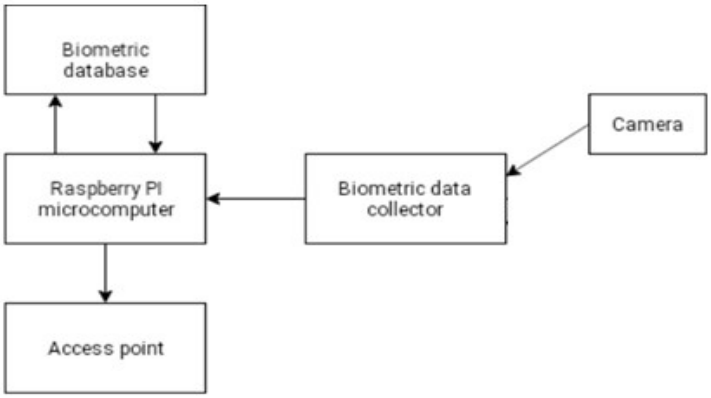

Рисунок 3.9 - Узагальнене уявлення системи МБІ

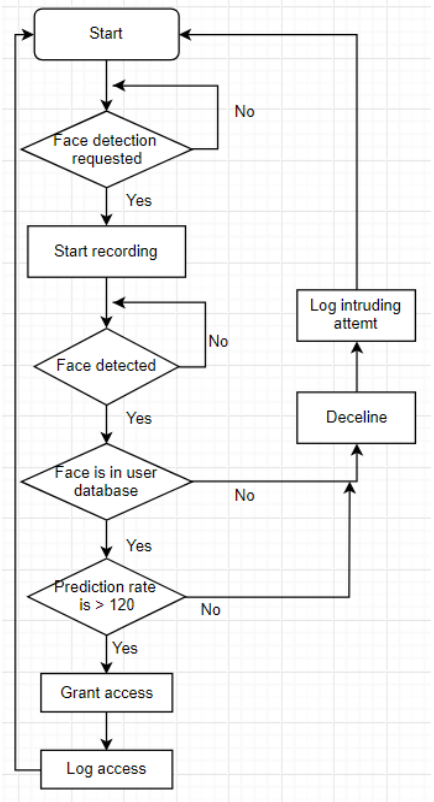

Рисунок 3.10 - Граф-схема алгоритму роботи

На вхід системи видаються біометричні дані - зображення обличчя.

Оскільки дані, отримані з датчиків, мають різну природу - двовимірне півтонування особи і папілярного візерунка пальця, а також голосовий відбиток, то алгоритми, що застосовуються на етапі попередньої і безпосередньої обробки даних, будуть відрізнятися.

Ухвалення рішення по двовимірному напівтонового зображення особи, що надійшов на вхід системи БІ, виконується в відповідності з алгоритмом, наведеним на рисунку 3.10.

Як навчальної вибірки для розпізнавання осіб була використана загальнодоступна база даних AT & T складається з 40 персон, по 10 фотографій на кожного

В якості тестової вибірки також були взяті фото двох людей з бази осіб AT & T (по 10 фото на людину), а також фото трьох випадкових людей, які погодилися на проведення тестування (по 10 фото на людину).

4 ПРОГРАМНА ТА АПАРАТНА РЕАЛІЗАЦІЯ

У відповідності до структури, описаної у розділі 2.4 даної записки, було розроблено тестову модель системи БІ та програмне забезпечення для здійснення автентифікації :

До системи входять:

- Мікрокомп’ютер Raspberry Pi 4B
- Блок живлення
- USB веб камера

4.1 Реалізація демонстраційного програмного забезпечення

У ході написання програми були використанні такі бібліотеки:

Бібліотека QT для створення та управління UI, також деякі модулі QT для роботи з Serial портом, для роботи з файловою системою, для роботи зі строками та управління потоками.

Бібліотека OpenCV для реалізацій розпізнавання та автентифікації обличчя, також до цієї бібліотеки входять допоміжні бібліотеки для роботи з медіа ресурсами, вони також були використанні у цієї реалізації.

Бібліотека Adafruit_Fingerprint для роботи зі сканером відбитків пальця, яка була модернізована для роботи на платформі мікрокомп’ютера Raspberry Pi

Основний функціонал програми знаходиться у класі “applicationcontroller”, далі будуть пояснені основні функції цього модуля:

Лістинг 4.1 – Конструктор класу ApplicationController

```
ApplicationController::ApplicationController(QLabel* videoOutput, QLabel* logLabel) :
userProfcontroller(new UserProfileController())
{
    vidOut = videoOutput;
    logOut = logLabel;
```

```
    model = cv::face::LBPHFaceRecognizer::create();
    TrainFaceRecogniserOnDummy();
    UpdateFaceRecogniser(fn_csv_usr.data());
}
```

Тут можна побачити процес створення об’єкту класу ApplicationController та призначення його ресурсів, таких як :

ресурс вікна для виводу зображення;

ресурс log для показу службової інформації;

ресурс model який являє собою об’єкт класу для автентифікації.

Лістинг 4.2 – Функція перенавчання модуля автентифікації

```
void ApplicationController::retrainRecogniser()
{
    model->clear();
    TrainFaceRecogniserOnDummy();
    UpdateFaceRecogniser(fn_csv_usr.data());
}
```

Лістинг 4.3 – Основна функція роботи з відео-потоком

```
Void ApplicationController::ProcessVideoOutput(EVidOutMode mode,
std::vector<std::string>& paths, int userID, bool& auth)
{
    if(is_displaying)
        return;

        auto deviceId = 0;

        cv::VideoCapture cap(deviceId);
        cap.set(cv::CAP_PROP_FOURCC, cv::VideoWriter::fourcc('M', 'J', 'P', 'G'));

        if(!cap.isOpened())
```

```
            std::cerr << "Capture Device ID " << deviceId << "cannot be opened." <<
std::endl;

        haarCascade.load(fn_haar.data());

        cv::Mat frame;


        Timer timer;
        int labell=-1;
        double accur=0.0;
        int countRecognised = 0;

        timer.start();
        is_displaying = true;
        while(is_displaying)
        {
            cap.read(frame);
            // Clone the cursolutionrent frame:
            cv::Mat original = frame.clone();
            // Convert the current frame to grayscale:
            cv::Mat gray;
            cvtColor(original, gray, cv::COLOR_BGRA2GRAY);
            // Find the faces in the frame:
            std::vector< cv::Rect_<int> > faces;
            haarCascade.detectMultiScale(gray, faces);

            for(size_t i = 0; i < faces.size(); i++)
            {
                cv::Rect face_i = faces[i];
                cv::Mat face = gray(face_i);
                cv::Mat face_resized;
                cv::resize(face, face_resized, cv::Size(100, 100), 1.0, 1.0,
cv::INTER_CUBIC);
                rectangle(original, face_i, CV_RGB(0, 255,0), 1);

                if(mode == EVidOutMode::REGISTER_USER && toMakePhoto)
                    paths.push_back(savePhoto(face_resized, userID));

                if(mode == EVidOutMode::RECOGNISE_USER)
                {
```

```
                    int pos_x = std::max(face_i.tl().x - 10, 0);
                    int pos_y = std::max(face_i.tl().y - 10, 0);
                    std::string box_text = cv::format("Prediction = %d Confidence=%f",
labell, accur);
                    putText(original, box_text, cv::Point(pos_x, pos_y),
cv::FONT_HERSHEY_PLAIN, 1.0, CV_RGB(255,0,0), 2.0);
                    model->predict(face_resized, labell, accur);

                    if(labell == userID && accur > 60)
                        countRecognised++;

                    if(countRecognised > 20)
                    {
                        auth = true;
                        is_displaying = false;
                        break;
                    }
                    else if(timer.elapsedSeconds() > 20)
                    {
                        auth = false;
                        is_displaying = false;
                        break;
                    }
                }
            }

            if(is_displaying)
            {
                cvtColor(original, original, cv::COLOR_BGR2RGB);
                QImage image1= QImage((uchar*) original.data, original.cols,
original.rows, original.step, QImage::Format_RGB888);

                vidOut->setPixmap(QPixmap::fromImage(image1));
                vidOut->show();
            }
        }
}

void ApplicationController::StopDisplayingImage()
{
    if(is_displaying)
```

```
    {
        is_displaying = false;
        controllerThread->join();
    }
}

void ApplicationController::UpdateFaceRecogniser(const std::string& path)
{
    std::vector<cv::Mat> images;
    std::vector<int> labels;
    try {
        readCsv(path, images, labels);
    } catch (cv::Exception& e) {
        std::cerr << "Error opening file \"" << path << "\". Reason: " << e.msg <<
std::endl;
        exit(1);
    }

    if(images.size() == 0)
        return;

    imWidth = images[0].cols;
    imHeight = images[0].rows;

    model->update(images, labels);
}

void ApplicationController::TrainFaceRecogniserOnDummy()
{
    std::vector<cv::Mat> images;
    std::vector<int> labels;
    try {
        readCsv(fn_csv.data(), images, labels);
    } catch (cv::Exception& e) {
        std::cerr << "Error opening file \"" << fn_csv << "\". Reason: " << e.msg <<
std::endl;
        exit(1);
    }
    imWidth = images[0].cols;
    imHeight = images[0].rows;
```

```
    model->train(images, labels);
}
```

Ця функція відповідає за показ відео-зображення, використання модулів розпізнавання обличчя, автентифікації та створення нових біометричних записів.

Функція зчитування даних о аккаунтах, місцезнаходження фото, персональний ID наведена у лістигну 4.4.

Лістинг 4.4 – Функція зчитування даних

```
void ApplicationController::readCsv(const std::string& filename
                                    , std::vector<cv::Mat>& images
                                    , std::vector<int>& labels
                                    , char separator)
{
    std::ifstream file(filename.c_str(), std::ifstream::in);

    if (!file)
    {
          std::string error_message = "No valid input file was given, please check the
given filename.";
        std::cerr << error_message << std::endl;
        return;
    }

    std::string line, path, classlabel;
    while (getline(file, line))
    {
        std::stringstream liness(line);
        getline(liness, path, separator);
        getline(liness, classlabel);
        if(!path.empty() && !classlabel.empty())
        {
            const auto& im = cv::imread(path, 0);
            cv::Mat resized;
            if(!im.empty())
            {
                resize(im, resized, cv::Size(92, 112), 1.0, 1.0, cv::INTER_CUBIC);
```

```
                images.push_back(im);
                labels.push_back(atoi(classlabel.c_str()));
            }
        }
    }
}
```

Ця функція відповідає за додавання нових користувачів у базу. Створюється персональний ID та зчитується біометрична інформація користувача. Ця функція виконується у віддільному потоці для того щоб основний потік не був заблокований.

Лістинг 4.5 – Функція додавання нових користувачів

```
void ApplicationController::addUser()
{

    controllerThread = std::make_unique<std::thread>([this]()
    {
        auto userID = userProfcontroller->getUserDBSize() + dummyFaceCount + 1;

                logOut->setText(logOut->text() + QString("Adding user under ID %d
\n").arg(userID));

        logOut->setText(logOut->text() + QString("Adding fingerprint\n"));

                            finger.reset(new    Adafruit_Fingerprint{FingerprintSerial,
FingerPrinttBaudrate});
        finger->begin(FingerPrinttBaudrate);
        EnrollExample::runEnroll(*finger.get(), userID);
        finger->close();

        std::vector<std::string> paths;

        bool dummy;
        ProcessVideoOutput(EVidOutMode::REGISTER_USER, paths, userID, dummy);

        UserProfile prof (userID, paths);
```

```
        userProfcontroller->addUserProfile(prof);
        vidOut->hide();
        retrainRecogniser();
    });

}
```

Функція автентифікації, функціонально схожа на функцію додавання нових користувачів, але знята інформація тут порівнюється з базою даних.

Лістинг 4.6 – Функція автентифікації

```
bool ApplicationController::authenticate()
{

    bool faceRecognised;

    controllerThread = std::make_unique<std::thread>([this, &faceRecognised]()
    {
                            finger.reset(new    Adafruit_Fingerprint{FingerprintSerial,
FingerPrinttBaudrate});
        finger->begin(FingerPrinttBaudrate);
        int userID = FingerprintExample::runFind(*finger.get());
        finger->close();

        std::vector<std::string> dummy;
        ProcessVideoOutput(EVidOutMode::RECOGNISE_USER, dummy, userID, faceRecognised);
    });

    while(!controllerThread->joinable()){}

    controllerThread->join();
    vidOut->hide();

    return faceRecognised;
}

void ApplicationController::makePhoto()
{
    if(is_displaying)
```

```
        toMakePhoto = true;
}

std::string ApplicationController::savePhoto(cv::Mat& pic, int userID)
{
    assert(is_displaying);
    assert(toMakePhoto);

    toMakePhoto = false;

    auto saveDirPath = "/home/pi/facesQT/faces/g" + QVariant(userID).toString();

    if(!QDir(saveDirPath).exists())
        QDir().mkdir(saveDirPath);

    int fileIndex = 0;

      QString saveFilePath = saveDirPath + "/p" + QVariant(fileIndex).toString() +
".bmp";

    while(QFile(saveFilePath).exists())
    {
        fileIndex++;
        saveFilePath = saveDirPath + "/p" + QVariant(fileIndex).toString() + ".bmp";
    }

    std::cout << saveFilePath.toStdString() << std::endl;

    cv::imwrite(saveFilePath.toStdString(), pic);

    return saveFilePath.toStdString();

}
```

Реалізація адаптеру протоколу UART для роботи з сканером відбитків пальців:

Лістинг 4.7 – Конструктор адаптеру:

```
FingerprintAdapter:: FingerprintAdapter (
            const QString& _portName
```

```
          ,     const QString& _baudrate
          ,     uint32_t password
      )
      :    m_hComPort{nullptr}
  {
    thePassword = password;
    theAddress = 0xFFFFFFFF;

    if( m_hComPort != nullptr )
        return;

    m_hComPort = new QSerialPort();

    m_hComPort->setPortName( _portName );
    m_hComPort->setBaudRate( _baudrate.toInt() );
    m_hComPort->setDataBits( QSerialPort::Data8 );
    m_hComPort->setParity( QSerialPort::NoParity );
    m_hComPort->setStopBits( QSerialPort::OneStop );
    m_hComPort->setFlowControl( QSerialPort::NoFlowControl );
  }
```

Функія початку роботи, яка відповідає за початкові налаштування, такі як швидкість передачі даних:

Лістинг 4.8 – З’єднання адаптеру з датчиком

```
  void FingerprintAdapter::begin(const QString& _baudrate) {
    std::this_thread::sleep_for(std::chrono::milliseconds(1000)); // one second delay
to let the sensor 'boot up'
      //connectReadEvent();
      m_hComPort->setBaudRate( _baudrate.toInt() );
      m_hComPort->open( QIODevice::ReadWrite );
  }
```

Функція яка відповідає за налаштування першого діалогу зі сканером відбитків пальцю шляхом впровадження паролю:

Лістинг 4.9 – Впровадження паролю для датчика

```
uint8_t Adafruit_Fingerprint::checkPassword(void) {
  auto packetReceived = GET_CMD_PACKET(
                FINGERPRINT_VERIFYPASSWORD
```

```
              , static_cast<uint8_t>(thePassword >> 24)
              , static_cast<uint8_t>(thePassword >> 16)
              , static_cast<uint8_t>(thePassword >> 8)
              ,     static_cast<uint8_t>(thePassword & 0xFF));
  if( packetReceived.has_value() )
  {
      auto packetResult = packetReceived.value();
      if (packetResult.data[0] == FINGERPRINT_OK)
        return FINGERPRINT_OK;

      return FINGERPRINT_PACKETRECIEVEERR;
  }
  else
    return FINGERPRINT_PACKETRECIEVEERR;
}
```

4.2 Реалізація алгоритму Віоли-Джонса

Функція генерації ознак

Лістинг 4.10 – Генерування ознак Хаара для використання у визначенні облич

```
std::vector<FeatureValue*> generateFeatureValues(int resolution) {
 std::vector<FeatureValue*> featureValueSet;
 FeatureValue *f;
 int minimalWidth, minimalHeight, width, height, x, y;
 int i = 0;

 for (int type = 0; type<4; type++) {
  if (type != 2) {
   minimalWidth = 4;
   width = 4;
  }
  else {
   width = 3;
   minimalWidth = 3;
  }
  if (type != 3) height = 4;
  else height = 3;
```

```
        x = 0;
        y = 0;
        while (height <= resolution)
        {
         while (width <= resolution)
         {
          while (y + height <= resolution)
          {
           while (x  +width <= resolution)
           {
            f = new FeatureValue(type, x, y, width, height);
            featureValueSet.push_back(f);
            x++;
           }
           x = 0;
           y++;
          }
          y = 0;
          if (type == 0) width+=2;
          else if (type == 2) width+=3;
          else width++;
         }
         width = minimalWidth;
         if (type == 1) height+=2;
         else if (type == 3) height+=3;
         else height++;
        }
       }
       return featureValueSet;
      }
```

Реалізація функції обучення алгоритму AdaBoost яка використовується для скорочення часу обчислення

Лістинг 4.11 – Алгоритм AdaBoost

```
ClassifierStrongVer1* AdaBoostLearning(CascadeClassifier *cc
                                       , std::vector<FeatureValue*>
&featureValueSet
                                       , std::vector<float*>
&SetOfPositiveFeatureValues
                                       , std::vector<float*> &negative_set
                                       , std::vector<float*> &validation_set
```

```
                                        , float minimalFpr
                                        , float maximalFnr
                                        , int resolution
                                        , bool verbose) {
        ClassifierWeakVer1 *bestWeakClassifier, *wc;
        ClassifierStrongVer1 *falseClassify = new ClassifierStrongVer1();
        float werticalSum, ErrorMinimal, werCError, betat, cfpr;
        int i, j;
        std::vector<FeatureValue*>::iterator it;

        float *weights = new float[SetOfPositiveFeatureValues.size()
+negative_set.size()];

        for (i=0; i<SetOfPositiveFeatureValues.size(); i++)
              weights[i] = 1/float(2*SetOfPositiveFeatureValues.size());

        for (i=0; i<negative_set.size(); i++)
              weights[SetOfPositiveFeatureValues.size()+i] =
1/float(2*negative_set.size());

        cfpr = 1.0;
        float *fValues = new float[SetOfPositiveFeatureValues.size()
+negative_set.size()];
        while (cfpr > minfimalFr) {
          if (falseClassify->fpr(negative_set, resolution) == 0)
          {
            break;
          }
```

Нормалізація ваги:

```
          werticalSum = 0;
          for (i=0; i<(SetOfPositiveFeatureValues.size()+negative_set.size()); i++)
               werticalSum += weights[i];

          for (i=0; i<(SetOfPositiveFeatureValues.size()+negative_set.size()); i++)
              weights[i] = weights[i]/werticalSum;
```

Вибір кращщого класифікатора:

```
      ErrorMinimal = 1;
          bestWeakClassifier = NULL;
          for (it = featureValueSet.begin(); it != featureValueSet.end(); ++it)
          {
```

```
            wc = new WeakClassifier(*it);
            for (i=0; i<SetOfPositiveFeatureValues.size(); i++)
              fValues[i] = (*it)->getValue(SetOfPositiveFeatureValues[i], resolution,
0, 0);

            for (i=0; i<negative_set.size(); i++)
              fValues[SetOfPositiveFeatureValues.size()+i] = (*it)-
>getValue(negative_set[i], resolution, 0, 0);
            werCError = wc->find_optimum_threshold(fValues,
SetOfPositiveFeatureValues.size(), negative_set.size(), weights);

            if (werCError < ErrorMinimal) {
              delete bestWeakClassifier;
              bestWeakClassifier = wc;
              ErrorMinimal = werCError;
            }
            else delete wc;
          }
```

Оновлення ваг:

```
          weight = ErrorMinimal/(1-ErrorMinimal);
          for (i=0; i<SetOfPositiveFeatureValues.size(); i++)
              if (bestWeakClassifier->classify(SetOfPositiveFeatureValues[i],
resolution, 0, 0, 0, 1) == 1)
                   weights[i] = weights[i]*weight;

          for (i=0; i<negative_set.size(); i++)
               if (bestWeakClassifier->classify(negative_set[i], resolution, 0, 0, 0,
1) == -1)
                        weights[SetOfPositiveFeatureValues.size()+i] =
weights[SetOfPositiveFeatureValues.size()+i]*weight;
```

Оновлення хибнопозитивного співвідношення:

```
          falseClassify->add(bestWeakClassifier, log(1/weight));
          falseClassify->optimiseThreshold(SetOfPositiveFeatureValues, resolution,
maximalFnr);
          if (validation_set.size() > 0) {
            cc->push_back(falseClassify);
            cfpr = cc->fpr(validation_set);
            cc->pop_back();
          }
          else cfpr = falseClassify->fpr(negative_set, resolution);
```

```
          if (verbose) {
            if (validation_set.size() > 0) cout << "    -> Added new Haar featureValue
(Validation set FPR=" << cfpr << ")" << endl;
            else cout << "    -> Added new Haar featureValue (FPR=" << cfpr << ")" <<
endl;
          }
        }
        delete[] weights;
        delete[] fValues;

        return falseClassify;
      }
```

Лістинг 4.12 – Функція виявлення об’єктів

```
png::image<png::rgb_pixel> detectObjects(png::image<png::rgb_pixel> Immage,
CascadeClassifier *cc, float fscale, float fincrement) {
  png::rgb_pixel pixel;
  int fnotfound = 0;
  float mean, stdev;
  int* detection;

  float *gsImmage, *iImmage, *siImmage;
  int i, j, a, b, increment;
  size_t x, y;

  vector<int*> detections;
```

Конвертація зображення у сіре зображення:

```
  gsImmage = new float[Immage.get_width()*Immage.get_height()];
  for (y = 0; y < Immage.get_height(); ++y) {
    for (x = 0; x < Immage.get_width(); ++x) {
      p = Immage.get_pixel(x, y);
      gsImmage[(y*Immage.get_width())+x] = ((0.21*p.red)+(0.71*p.green)+(0.07*p.blue));
    }
  }
```

Обчислення цілісне зображення та квадратне ціле зображення:

```
  iImmage = integralImage(gsImmage, Immage.get_width(), Immage.get_height());
```

```
  siImmage = squaredIntegralImage(gsImmage, Immage.get_width(), Immage.get_height());
  delete[] gsImmage;
```

Виконування розпізнавання облич у кількох масштабах:

```
int resolution = cc->getBaseResolution();
  while (resolution <= Immage.get_width() && resolution <= Immage.get_height()) {
    increment = resolution*fincrement;
    if (increment < 1) increment = 1;

    for (i=0; (i+resolution)<=Immage.get_width(); i+=increment) {
      for (j=0; (j+resolution)<=Immage.get_height(); j+=increment) {
        mean=evaluateIntegralRectangle(iImmage, Immage.get_width(), i, j, resolution,
resolution)/pow(resolution, 2);
        stdev = sqrt((evaluateIntegralRectangle(siImmage, Immage.get_width(), i, j,
resolution, resolution)/pow(resolution, 2))-pow(mean, 2));

        if (cc->classify(iImmage, Immage.get_width(), i, j, mean, stdev) == true) {
          detection = new int[3];
          detection[0]=i; detection[1]=j; detection[2]=resolution;
          detections.push_back(detection);
        }
        else fnotfound++;
      }
    }

    cc->scale(fscale);
    resolution = cc->getBaseResolution();
  }

  merge_detections(detections);

  for (std::vector<int*>::iterator it = detections.begin(); it != detections.end(); +
+it) {
    draw_square(&Immage, (*it)[0], (*it)[1], (*it)[2]);
  }

  return Immage;
}
```

5 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

5.1 Аналіз розробленого програмного забезпечення

При розробці програмного забезпечення для вбудованих систем слід виділити такі критерії, як:

- можливість заміни частини компонентів тестовими зразками;
- тесто-придатність системи;
- можливість швидкого діагностування помилки;
- гнучкість спроектованої системи;

При розробці тестового програмного забезпечення було досягнуто наступні характеристики:

- архітектура програми дозволяє проводити швидкі зміни та додавати нові програмні компоненти;
- частина компонентів системи може бути замінена тестовими сервісами а/або частково тестовими даними;
- сервіс логування та асерцій дозволяє локалізовувати та діагностувати отримані помилки у ході роботи системи;

На етапі тестування і використання, запропонована система має вигляд, представлений на рисунку 9.

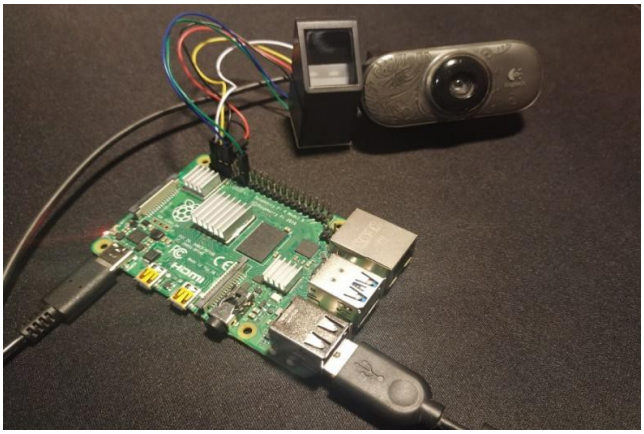

Рисунок 5.1 – Тестування запропонованої системи БІ

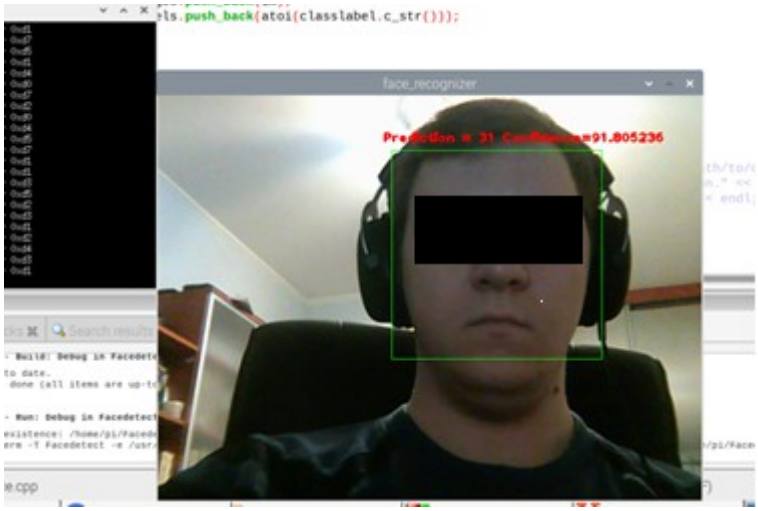

Рисунок 5.2 – тестування програмного забезпечення

5.2 Результати тестування

У процесі розробки були зняті такі часові втрати та порівнянні з іншими методами біометричної автентифікації:

Біометрична ознака	Точність		Час обробки	Простота використання
	FRR	FAR	мс	
Геометрія обличчя	**~ 6%**	**~ 0,1%**	**~750мс**	**Висока**
Відбитки пальців	~ 1%	~ 0,00002%	~400мс	Середня
Голос	~ 3%	~ 0,1%	~1500мс	Висока

Таблиця 5.1 – Результати роботи розробленої системи біометричної ідентифікації людини

Реалізація	Точність		Середній час обробки
	FRR	FAR	мс
Проектна реалізація	**~ 15%**	**~ 5%**	**~500мс**
OpenCV	~ 10%	~ 3%	~750мс

Таблиця 5.2 – Результати порівняння ефективності реалізацій алгоритму Віоли-Джонса

В процесі дослідження порівнялися реалізації алгоритмів Віоли-Джонса який був реалізований у контексті цього проекту та бібліотечної реалізації OpenCV.

Для дослідження були використані такі набори даних:

- Зображення обличь
- Зображення чужорідних об’єктів
- Зображення обличь з накладеним на зображення шумом
- Зображення на яких присутні як особи, так і чужорідні предмети
- Зображення на яких присутні як особи, так і чужорідні предмети з накладеним на зображення шумом

Усереднені результати цих досліджень наведені в таблиці 3.

ВИСНОВКИ

В процесі виконання атестаційної роботи була розроблена система біометричної автентифікації яка використовує обличчя людини як біометричний фактор.

Було реалізоване програмне забезпечення для цієї системи з використанням мови C++ та компілятора GCC.

Був реалізований алгоритм Віоли-Джонса з використанням багато-поточного програмування та використання SIMD інструкцій для забезпечення максимальної продуктивності, для використання його у процесі розпізнавання обличчя людини.

Було проведено дослідження у вигляді порівняння реалізованого алгоритму з бібліотечною реалізацією відкритої бібліотеки OpenCV та отримані результати які говорять про те що проектна реалізація в деяких аспектах ліпше бібліотечної реалізації OpenCV.

Дослідження показало перспективність використання SIMD оптимізації коду та багато-поточного програмування у контексті реалізації алгоритмів комп’ютерного зору, особливо у разі використання мікрокомп’ютерів з відносно непродуктивними процесорами, наприклад Raspberry Pi 4B.

Подальші дослідження в області включають в себе реалізації інших оптимізованих алгоритмів комп’ютерного зору, наприклад подальшого ланцюгу в розпізнавані обличь людини та подальшої автентифікації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кухарев Г.А. Биометрические системы: методы и средства идентификации личности человека / Г.А. Кухарев. – СПб. : Политехника, 2001. – 240 с, Li, P.; Zhang, R. The evolution of biometrics. In Proceedings of the IEEE International Conference on Anti-Counterfeiting Security and Identification in Communication, Chengdu, China, 18–20 July 2010; pp. 253–256
2. O.Barkovska, N. Axak, D. Rosinskiy, S. Liashenko Application of mydriasis identification methods in parental control systems. Proceeding of 9th International IEEE Conference “DEpendable Systems, SERvices and Technologies” (DESSERT’2018). – Kyiv, Ukraine. – May 24-27, 2018. – Pp. 459-463.
3. Буэно, Суарес, Эспиноса. Обработка изображений с помощью OpenCV = Learning Image Processing with OpenCV. — М.: ДМК-Пресс, 2016. — 210 с. — ISBN 978-5-97060-387-1.
4. DC. He and L. Wang (1990), "Texture Unit, Texture Spectrum, And Texture Analysis", Geoscience and Remote Sensing, IEEE Transactions on, vol. 28, pp. 509 - 512.
5. Wang and DC. He (1990), "Texture Classification Using Texture Spectrum", Pattern Recognition, Vol. 23, No. 8, pp. 905 - 910.
6. M. Heikkilä, M. Pietikäinen, "A texture-based method for modeling the background and detecting moving objects", IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(4):657-662, 2006.
7. Paul Viola and Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", ACCV AND PATTERN RECOGNITION 2001
8. A.K. Jain et al., 50 years of biometric research: Accomplishments, challenges, and opportunities, Pattern Recognition Letters (2016), http://dx.doi.org/10.1016/j.patrec.2015.12.013

9. Lanitis A., A survey of the effects of aging on biometric identity verification, Int. J. Biom. (2009), 2, 1, 34-52.
10. Agata Wojciechowska, Michał Choraś, Rafał Kozik THE OVERVIEW OF TRENDS AND CHALLENGES IN MOBILE BIOMETRICS Journal of Applied Mathematics and Computational Mechanics (2017), 16(2), 173-185 DOI: 10.17512/jamcm.2017.2.14
11. Ross A., Jain A.K. (2015) Biometrics, Overview. In: Li S.Z., Jain A.K. (eds) Encyclopedia of Biometrics. Springer, Boston, MA https://doi.org/10.1007/978-1-4899-7488-4_182
12. I.Ruban, V.Martovytskyi, A.Kovalenko, N.Lukova-Chuiko Identification in informative systems on the basis of users' behaviour. Proceeding of 8th International Conference on Advanced Optoelectronics and Lasers (CAOL*2019). – Sozopol, Bulgaria. – September 06-08, 2019. – Pp. 574-577.
13. Ross A., Jain A.K. (2015) Biometrics, Overview. In: Li S.Z., Jain A.K. (Eds) Encyclopedia of Biometrics. Springer, Boston, MA https://doi.org/10.1007/978-1-4899-7488-4_182
14. J. K. Hundal and S. T. Hamde, "Some feature extraction techniques for voice based authentication system," (2017) IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, 2017, pp. 419-421. doi: 10.1109/ICPCSI.2017.8392328
15. O Barkovska.,I Movsesian., N Yeromina., O Liashenko., D.Tkachenko «System of Individual Multidimensional Biometric Authentication», (2019) International Journal of Emerging Trends in Engineering Research(IJETER) https://doi.org/10.30534/ijeter/2019/147122019