МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
д-р техн. наук, проф.

_____ І. М. Журавська

« __ » _____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

БАГАТОКАНАЛЬНА СИСТЕМА КЕРУВАННЯ ДОСТУПОМ НА БАЗІ RASPBERRY PI ТА OPEN CV

Галузь знань 12 Інформаційні технології
Спеціальність 123 Комп'ютерна інженерія
123 – КМР.1 – 605.21710504

Студент

_____ В. В. Бойченко
підпис

«__»_____2023 р.

Керівник доцент, канд. фіз.-мат. наук _____ С. В. Пузирьов
підпис

«__»_____2023 р.

Миколаїв – 2023

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АПЗ – Апаратно-Програмне Забезпечення
ДІ – Довірчий Інтервал
ІТ – Інфомаційні Технології
СМО – Система Масового Обслуговування
ШІ – Штучний Інтелект

API – Acronym for Programming Interface
CUDA – Compute Unified Device Architecture
DB – Database
EAC – Eurasian Conformity
FTP – File Transfer Protocol
FIFO – First Input First Output
FS – Fingerprint sensor
HDMI – High-Definition Multimedia Interface
HTPC – Home Theater PC
LIFO – Last Input First Output
MMX – Multimedia Extensions
NAS – Network Data Storage
OpenCV – Open-Source Computer Vision
OS – Operating System
PACS – Physical Access Control System
PGP – Pretty Good Privacy
SATA – Serial Advanced Technology Attachment
SBC – Session Border Controller
SSH – Secure Shell
SSE – Streaming SIMD Extensions
TTL – Time-To-Live
UART – Universal Asynchronous Receiver/Transmitter

ВСТУП

Керування доступом до різних об'єктів є актуальною проблемою, особливо в умовах сьогодення. З розвитком технологій та досвіду людства у боротьбі з системами безпеки, з’являється необхідність для покращення або оновлення систем отримання доступу до контролю фізичних приладів. Майже усі банківські компанії мають дуже сильні та сучасні системи кібербезпеки, але навіть цього не завжди достатньо, щоб уникнути злому. Щодо безпеки пристроїв фізичного збереження даних або коштів, є також багато способів, які і до сьогодні мають успіх. Основною проблемою є недостатньо безпечна система отримання доступу через отримання сигналу, так як зазвичай це може бути сканування сітківки ока, або відбиток пальця довіреної особи. Щоб уникнути таких випадків і комплексного вирішення цих питань використовуються різноманітні апаратно-програмні рішення. В даній роботі представлено одне з них, а саме система багатоканального керування доступом на базі одноплатного комп’ютеру Raspberry Pi та технології OpenCV.

Існує багато способів отримання доступу до певних об’єктів, наприклад розпізнавання голосу, обличчя, вібрації, сканування відбитків пальців або сітківки/райдужки ока, введення секретного кода, і тд. Найнадійнішим можна вважати відбиток пальця, сканування сітківки ока або розпізнавання обличчя через камеру.

Це саме ті технології, які будуть складовими для проєктування багатоканальної системи щоб контролю доступу.

Актуальність роботи: Проєктування масштабованої системи контролю доступу з різною кількістю каналів ідентифікації та моделювання роботи такої системи на основі теорії систем масового обслуговування

Мета: створення системи, яка надає різний набір каналів ідентифікації для доступу до певних ресурсів

Об’єкт: процес надання доступу при розпізнаванні об’єктів у відеопотоці OpenCV і додаткових методів ідентифікації на базі одноплатного комп’ютера Raspberry Pi 4

Предмет: Інтелектуальна багатоканальна система керування доступом.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– розкриття об’єкту і предмету дослідження;

– опис і аналіз структурних і функціональних особливостей об’єкта дослідження;

– огляд і аналіз сучасного стану інформаційних технологій у даній предметній області;

– вибір математичних методів та засобів комп’ютерної інженерії;

– огляд і аналіз принципів роботи технології OpenCV;

– моделювання і проєктування принципу роботи багатоканальної системи та вибір компонентів;

– розробка апаратно-програмного забезпечення системи;

– розробка питання з охорони праці та безпеки життєдіяльності.

Робота пройшла апробацію під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

Публікації. Основні положення та результати магістерської роботи опубліковані у збірнику матеріалів XXV Всеукраїнської науково-практичної конференції «Могилянські читання–2022» [1].

РОЗДІЛ 1 РОЗКРИТТЯ ПРИНЦИПІВ РОБОТИ СИСТЕМИ ТА АНАЛІЗ СУЧАСНИХ РІШЕНЬ

Ефективна безпека починається з розуміння відповідних принципів. Простого застосування певного набору процедур із пам’яті недостатньо в світі, де сьогоднішні «найкращі практики» — це завтрашні збої в системі безпеки. IT-безпека — це сфера, яка швидко розвивається, і знання того, як виконувати дії, необхідні для загальноприйнятих практик, недостатньо для забезпечення найкращої безпеки ваших систем.

Серед основних концепцій безпеки є контроль доступу. Це настільки фундаментально, що стосується безпеки будь-якого типу, а не лише IT-безпеки. Усе, починаючи від посадки у ваш автомобіль і закінчуючи запуском ядерних ракет, захищено, принаймні теоретично, певною формою контролю доступу. Через свою універсальну застосовність до безпеки, контроль доступу є однією з найважливіших концепцій безпеки для розуміння.

Існує три типи контрою доступу:

– Ідентифікація: щоб контроль доступу був ефективним, він повинен забезпечувати певний спосіб ідентифікації особи. Найслабші можливості ідентифікації просто ідентифікують когось як частину розпливчастої, погано визначеної групи користувачів, які повинні мати доступ до системи. Ім’я користувача, підпис електронної пошти PGP або ключ забезпечують певну форму ідентифікації;

– Аутентифікація — це процес забезпечення автентичності використовуваного ідентифікатора — того, що його використовує потрібна особа. У найпоширенішій формі IT-безпеки автентифікація передбачає підтвердження пароля, пов’язаного з іменем користувача. Існують також інші форми автентифікації, такі як відбитки пальців, смарт-карти та ключі шифрування;

– Авторизація: набір дій, дозволених для певної особи, становить основу авторизації. На комп’ютері авторизація зазвичай має форму дозволів на читання, запис і виконання, прив’язаних до імені користувача.

Ці три елементи контролю доступу поєднуються, щоб забезпечити необхідний захист. Наприклад, простого доступу до базових системних утиліт на робочій станції або сервері ідентифікація необхідна для обліку (тобто відстеження поведінки користувача) і надання чогось для автентифікації. Автентифікація необхідна, щоб переконатися, що ідентифікаційні дані не використовує не та особа, а авторизація обмежує ідентифікованого автентифікованого користувача від участі в заборонених діях[2].

Залежно від типу безпеки, який потрібен, різні рівні захисту можуть бути більш або менш важливими в конкретному випадку. Щоб отримати доступ до кімнати для переговорів, може знадобитися лише ключ, який зберігається в скриньці, яку легко зламати в зоні адміністратора, але доступ до серверів, ймовірно, потребує трохи більшої обережності.

Останнім часом багатофакторній автентифікації приділяється велика увага. На жаль, така ж обізнаність про безпеку не поширюється на основну частину кінцевих користувачів, які часто не вважають це чимось корисним, або витратою часу.

Однак, багато IT-відділів не настільки усвідомлюють важливість контролю доступу, не дивлячись на необхідність. Звичайно, можна використовувати двофакторний захист для захисту своїх девайсів, поєднуючи стандартну автентифікацію паролем зі сканером відбитків пальців. Але якщо все, що потрібно, щоб фізично дістатися до серверів, — це ключ, сканер відбитків пальців на ноутбуці не матиме великого значення. Існують способи обійти сканери відбитків пальців, включаючи можливість завантаження з операційної системи LiveCD або навіть фізичного видалення жорсткого диска та доступу до нього з системи, яка не забезпечує біометричний контроль доступу. За останні місяці деякі корпорації та державні установи важко засвоїли уроки керування ноутбуками.

Треба мати на увазі той факт, що при роботі з високотехнологічними системами, не виключає необхідності захисту від низькотехнологічних хакерів.

Тому треба розуміти основи контролю доступу та застосовувати їх до кожного аспекту процедур безпеки.

1.1 Контроль доступу фізичної безпеки з використанням біометрії

Паролі є загальним засобом перевірки ідентичності користувача перед наданням доступу до інформаційних систем.

Сьогодні біометрія використовується як для ідентифікації, так і для верифікації – іноді навіть для обох одночасно. Якщо необхідно більш високий рівень безпеки, багатофакторна перевірка є одним із шляхів. А використання біометричної перевірки як додаткового кроку в процесі безпеки, наприклад, після пред’явлення бейджа, є зручним способом підвищити рівень безпеки[3]. Це також безпечніше, ніж, наприклад, використання пін-коду для підтвердження, оскільки його можна передати іншим людям (рис.1.1).

Біометричне сканування зазвичай спирається на вбудовану технологію сенсорів мобільних та інших пристроїв, оскільки вона майже витіснила програмні, сторонні біометричні алгоритми. Деякі рішення для сканування відбитків пальців побудовані за децентралізованою моделлю, такою як FIDO, яка гарантує, що шаблон відбитків пальців користувача зберігається на його пристрої. У цьому випадку сканований користувачем відбиток пальця перевіряється локально, токен надсилається постачальнику послуг, і доступ надається. Біометрична автентифікація відбувається локально, а самі біометричні дані не зберігаються у постачальника послуг (справжній секрет).

Рисунок 1.1 – Контроль доступу фізичної безпеки використовуючи додаткову біометрію

Під час використання біометричних даних існують певні проблеми та навіть потенційні ризики. Користувачі повинні бути фізично присутні, що менш зручно для користувачів. При призначенні картки доступу або PIN-коду присутність необов’язкова – це можна зробити заздалегідь[4].

1.2 Контроль доступу за допомогою PIN-коду

Одним із методів фізичної безпеки є зчитування кода доступу з клавіатури. Зчитувачі клавіатури дозволяють використовувати персональні ідентифікаційні номери (PIN) як облікові дані доступу. Сьогодні використання PIN-кодів, які широко використовуються скрізь, від панелей сигналізації до банківських послуг і телефонів, стає все більш поширеним у програмах EAC[5]. Загрозу клонування картки можна звести нанівець, використовуючи секретні PIN-коди, які відомі лише користувачу. Насправді, хоча PIN-коди можуть значно підвищити безпеку програм EAC, вони настільки безпечні, наскільки пильність користувача (рис.1.2).

Рисунок 1.2 – Контроль доступу за допомогою PIN-коду

Інтегрований зчитувач клавіатури — пристрій, який приймає як PIN-код, так і облікові дані — може використовувати PIN-код трьома способами: лише PIN-код, картка чи PIN-код і картка плюс PIN-код. У випадках, коли контроль доступу реалізується лише для зручності, як альтернатива фізичним ключам, метод лише з PIN-кодом може бути достатнім, оскільки загрози безпеці можуть бути незначними. Це також ефективно та економно, оскільки усуває необхідність купувати та керувати фізичними картками чи тегами. Метод картки чи PIN-коду часто використовується, коли може знадобитися доступ кільком типам відвідувачів[12].

Для програм контролю доступу, які вимагають підвищеної безпеки, метод картки плюс PIN-код — форма двофакторної автентифікації — є найкращим варіантом. Коли PIN-код використовується в поєднанні з іншими обліковими даними, такими як картка або тег, безпека підвищується завдяки додаванню другого рівня ідентифікації до транзакції доступу. У цьому типі програми безпека базується на тому, що є у користувача(картці чи мітка), а також на тому, що користувач знає (PIN-код).

У сценаріях двофакторної автентифікації ані фізичні облікові дані, ані лише PIN-код не нададуть доступ; натомість їх слід використовувати разом. Це

забезпечує більш безпечне рішення контролю доступу. Незалежно від того, як використовуються зчитувачі клавіатури, правильне керування та продумане застосування PIN-кодів має вирішальне значення для підтримки цілісності системи контролю доступу.

1.3 Контроль доступу з використанням механічної системи безпеки

Механічні пристрої контролю доступу – механічні пристрої за своєю природою покладаються на рух на відміну від електронних пристроїв, які вмикають лише зміну стану. Механічні пристрої також мають долати інші бар’єри, такі як тертя, маса, імпульс, натяг пружини, крутний момент та інші тиски (рис.1.3).

Рисунок 1.3 – Контроль доступу за допомогою кодового механічного замка

Щоб відкрити такий замок, необхідно набрати правильну кодову комбінацію, найчастіше за допомогою багаторазового повороту спеціальної ручки. Кодові механічні замки невибагливі до навколишніх умов експлуатації[13]. Основним недоліком є негнучкість системи, так як при втраті коду доступу, буде дуже складно, і деяких випадках, неможливо змінити код доступу.

Більш складний пристрій замка обумовлює і вищу його вартість. Крім того, механічні кодові замки забезпечують найтриваліший час доступу до вмісту.

Всі кодові замки є програмованими або не є такими. Більшість кодових механічних замків є якраз не програмованими, тобто з незмінним кодом. І якщо власник забуде заводську кодову комбінацію, то відновити її буде можливо лише за запитом у виробника, а це довга і досконала процедура.

1.4 Контроль доступу з використанням сітківки ока

Сканування сітківки ока – це технологія біометричної перевірки, яка використовує зображення малюнка кровоносних судин сітківки ока людини як унікальну ідентифікаційну функцію для доступу до захищених об’єктів.

Технології біометричної верифікації засновані на способах унікальної ідентифікації людини за допомогою однієї або кількох біологічних ознак (Рис 1.4).

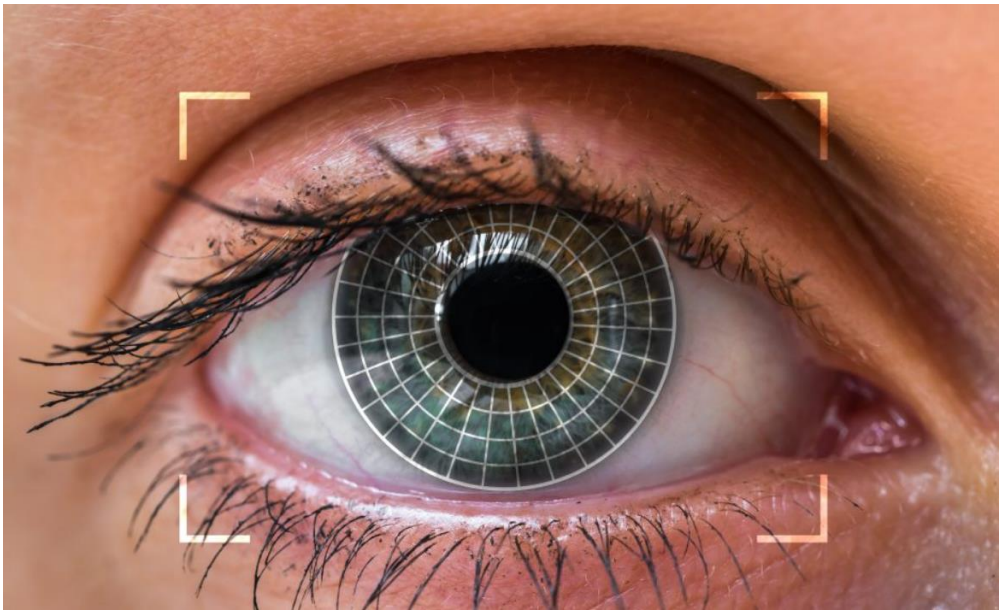

Рисунок 1.4 – Контроль доступу за допомогою сканування сітківки ока

Сканування сітківки вперше було запропоновано в 1935 році докторами. Карлтон Саймон і Айседор Голдштейн. Комерційне використання почалося в

1984 році з Eyedentity, яка випустила перші пристрої, які використовували технологію сканування сітківки.

Існує два типи сканування ока:

1. Сітківка – це внутрішній шар клітин, що вистилає задню стінку ока. Сітківка містить кілька шарів чутливої тканини і мільйони фоторецепторів. Цей шар тканини виявляє світло і створює електричні імпульси, які через зоровий нерв надсилаються до мозку, що дозволяє людині бачити. Іншими словами, сітківка, по суті, діє як датчик зображення в фотоапараті.Стовпчаста сітківка, що складається з декількох шарів нейронів, з'єднана серією синапсів, які підтримуються зовнішнім шаром пігментованих епітеліальних клітин. Основні світлочутливі клітини сітківки складаються з двох специфічних типів фоторецепторних клітин: паличок і колбочок. Палички зазвичай функціонують в умовах тьмяного освітлення і відповідають за чорно-білий зір, тоді як колбочки побудовані для більш яскравого освітлення і відповідають за розпізнавання кольорів;

2. Райдужка – це пігментована кільцеподібна м'язова завіса біля передньої частини ока, яка контролює розмір і діаметр зіниці. Райдужка регулює кількість світла, яке потрапляє в око, відкриваючи і закриваючи зіницю, що має вирішальне значення для підтримання правильної лінії зору. Райдужка є визначальним фактором кольору очей людини. Райдужка складається з двох різних шарів: строми, яка є пігментованим волокнисто-судинним шаром спереду, і пігментованих епітеліальних клітин, які знаходяться під стромою[23].

Сканування сітківки ока та розпізнавання райдужної оболонки ока є двома найбільш точними та надійними біометричними методами, які використовуються для підтвердження особи та автентифікації.

Одне дослідження біометричної автентифікації показало, що рівень помилкової відмови при скануванні сітківки становить 1,8%. Це комплексне дослідження чітко показало, що сканування сітківки є найточнішим з усіх інших біометричних методів.

Розпізнавання райдужної оболонки ока є одним із найшвидших методів біометричної перевірки особистості та автентифікації на ринку. Подібно до сканування сітківки ока, розпізнавання райдужної оболонки ока також може похвалитися одним із найнижчих показників помилкового прийняття та помилкового відхилення. Насправді рівень хибного відхилення та хибного прийняття для розпізнавання райдужної оболонки ока значно нижчий, ніж для розпізнавання відбитків пальців.

Сканування сітківки та райдужної оболонки є біометричними технологіями, які базуються на використанні очей. Однак існують чіткі відмінності, які відрізняють два методи, які підходять для конкретних випадків використання (Рис. 1.5).

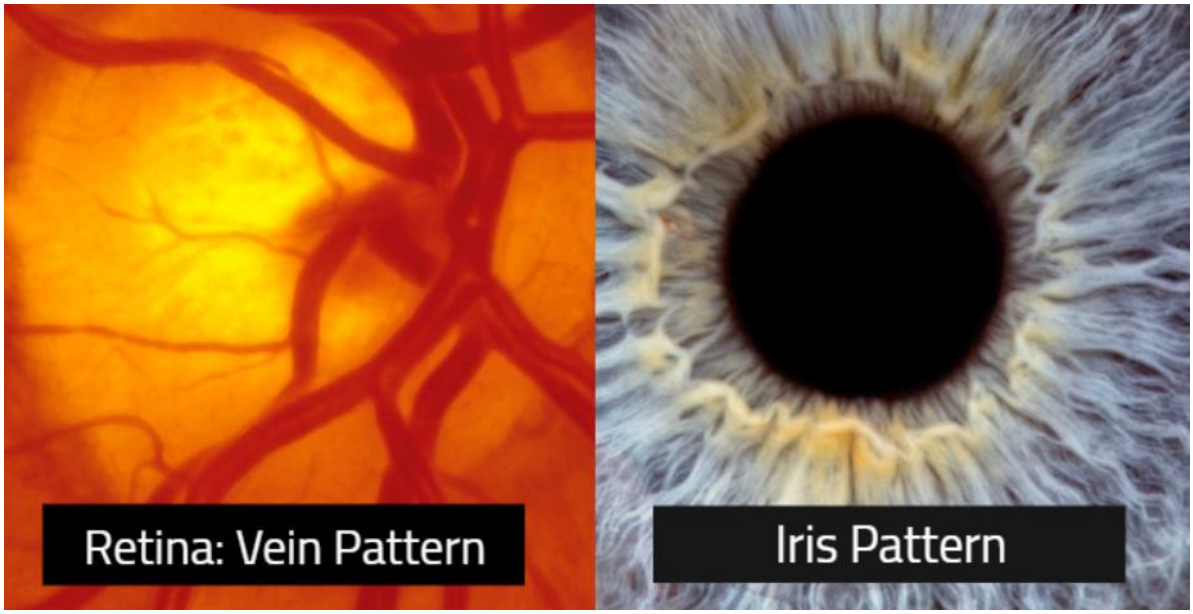

Рисунок 1.5 – Відмінності методів сканування сітківки та райдужки ока

Хоча структура райдужної оболонки залишається стабільною протягом тривалого часу, на точність сканування сітківки можуть впливати захворювання. На відміну від райдужної оболонки, сітківка може зазнавати змін, якщо людина страждає на захворювання або сліпоту. Сканування сітківки також потребує близькості до сканера (як під час перегляду в мікроскоп), тоді як розпізнавання райдужної оболонки, будучи звичайним фотографічним процесом, може бути виконане навіть на відстані.

Виходячи з унікальних фізіологічних характеристик ока для перевірки обличчя, як сканування сітківки, так і розпізнавання райдужної оболонки ока мають власний список переваг і недоліків.

Розпізнавання райдужної оболонки ока ширше використовується в комерційних цілях. Схвалення широкого загалу сканування сітківки надзвичайно низьке через його незручність та інвазивність. Розпізнавання райдужної оболонки, з іншого боку, вважається більш зручним для користувача, оскільки воно використовує безконтактну цифрову фотографію для точної перевірки особистості людини.

Розпізнавання на сітківці майже виключно використовується в програмах, які вимагають надзвичайно високого рівня безпеки. Технологія сканування сітківки в основному використовується на військових об’єктах і базах, ядерних установках і високотехнологічних лабораторіях.

Хоча деякі програми для смартфонів стверджують, що базуються на скануванні сітківки ока, зазвичай вони базуються на скануванні райдужної оболонки ока, методі ідентифікації людини за унікальними візерунками в кільцеподібній області навколо зіниці ока. Сканування сітківки приблизно в 70 разів точніше, ніж сканування райдужної оболонки ока, і в 20 000 разів точніше, ніж методи на основі відбитків пальців. Однак сканування сітківки вимагає, щоб об’єкт зосередився на одній точці протягом усіх 15 секунд.

Враховуючи складність біометрії сітківки, не дивно, що технологія сканування сітківки має низький рівень комерційного визнання. Проте розпізнавання райдужної оболонки широко визнано комерційно життєздатним методом порівняно зі скануванням сітківки. Оскільки очікується, що вартість впровадження технології розпізнавання райдужної оболонки ока зменшиться, технологія сканування райдужної оболонки ока домінуватиме на світовому ринку біометричної перевірки та автентифікації особи.

1.5 Інновації у сфері систем фізичного контролю доступу

Завдяки останнім досягненням технологій безпеки тепер доступні системи контролю фізичного доступу з багатьма розширеними функціями та опціями.

Для забезпечення найкращої безпеки та цінності, облікові дані для мобільних пристроїв дають користувачам зручність використання мобільних телефонів для авторизації і надання доступу вбудованою багатофакторною біометричною автентифікацією. Також зникає ймовірність втратити, залишити смартфон вдома чи передати в офісі, як у випадку з чіп-картою.

Іншим фактором контролю фізичного доступу, є обслуговування та керування системою. Багато застарілих систем контролю доступу використовують громіздкі зчитувачі та локальні сервери, які потребують особистого керування та обслуговування. Затримки в оновленні системи можуть підвищити ризик зламу. Крім того, якщо облікові дані потрібно перепризначити або створити заново – з’являється необхідність адміністратора. Якщо є необхідність у віддаленому доступі до системи або у отриманні оновлень безпеки в реальному часі, слід розглянути системи, які використовують сучасніше програмне забезпечення. Вибираючи PACS для моделювання пристрою, буде отримано додаткові переваги від використання системи контролю фізичного доступу, яка працює на хмарній платформі [6].

За допомогою цього типу PACS обладнання для контролю доступу (зчитувачі, концентратори та плати керування) комунікує з програмним забезпеченням через хмару для більш гнучкого, масштабованого рішення безпеки. Ось деякі переваги використання хмарного PACS:

– Можливість дистанційного керування апаратним забезпеченням з будь-якого місця, включаючи активацію розблокувань і блокувань;

– Миттєва перевірка облікових даних для нових користувачів, а також можливість негайного скасування доступу;

– Можливість отримувати сповіщення в реальному часі про події доступу та потенційні загрози;

– Технічне обслуговування та усунення несправностей можна виконувати віддалено в хмарі без необхідності перебування на місці;

– Оновлення системи автоматично завантажуються з хмари, що пришвидшує та спрощує отримання найновіших засобів безпеки;

– Журнали аудиту в реальному часі для всіх дій доступу, які зберігаються в хмарі, тому до них можна отримати доступ з будь-якого авторизованого пристрою;

– Можливість інтегрувати систему контролю фізичного доступу з іншим програмним забезпеченням безпеки, таким як відеоспостереження, керування відвідувачами, засоби зв’язку та рішення для управління простором.

Хоча системи контролю фізичного доступу є чудовим способом обмежити доступ, вони не пропонують повної картини безпеки. Особливо для організацій із гнучкими або ступінчастими робочими змінами, розподіленими командами та змінним списком підрядників або тимчасових працівників, менеджери нерухомості повинні стежити за тим, що відбувається в просторі в будь-який момент: ось чому відеоспостереження є важливою частиною сучасного PACS[18].

На додаток до встановлення комерційних камер безпеки та інтеграції хмарної системи керування відео, необхідні зчитувачі з підтримкою відео. Перевага відеозчитувача полягає в тому, що він дає командам безпеки «очі» на кожну подію доступу, де вона відбувається. Завдяки дистанційному хмарному керуванню пристроями для зчитування відео PACS можна візуально перевірити, чи ідентифікаційні дані збігаються з обліковими даними, помітити спроби неавторизованого входу, пом’якшити переслідування та отримати доступ до камери будь-де.

Це не означає, що також зникає необхідність у камерах безпеки. Відеозчитувач необхідний, щоб побачити, що відбувається у певному просторі. Разом повністю інтегровані системи контролю фізичного доступу з підтримкою

відео та VMS покращують моніторинг безпеки, забезпечують швидший час відгуку та мінімізують ризики для всієї організації.

1.5.1 Мультиспектральна біометрія обличчя у контролі доступу

В даний час розробляються різні інтелектуальні технології контролю доступу, наприклад прикордонний контроль. Вони включають прийняття рішень і взаємодію людини з машиною, перевірку особи та біометрію, оцінку ризику біометрії поведінки, проїзні документи, системи перевірки документів і виявлення шахрайства.

Ця підтримка може бути досягнута шляхом ефективного використання топології автоматизованого контролю доступу, передових біометричних технологій і методів взаємодії людини і машини. Останній використовує досвід розробки відомих діалогових систем, таких як, зокрема, SmartKom, який має спеціальну обробку вхідних даних, специфічний для модальності аналіз і злиття, а також управління взаємодією[9].

Конкретна структура системи підтримки скринінгу та інтерв’ю показана на рис. 1.6. Система складається з камер у RGB-Depth (RGB-D) та інфрачервоному діапазонах, процесорів попередньої інформації та онлайн-даних (таких як списки спостереження), і підтримку прийняття рішень, яка включає перетворення результатів у семантичну форму, таку як протокол, доступний для персоналу. Ці семантичні дані базуються на попередній інформації, зібраній під час спостереження за перевіреною особою у видимому та інфрачервоному діапазонах, а також інформації, отриманій із спостереження, розмови та додаткових джерел. Процес генерування питань ініціюється інформацією, отриманою біометричними пристроями. Стратегія анкетування може зменшити деякі помилки, а також ненадійність біометричних даних[14].

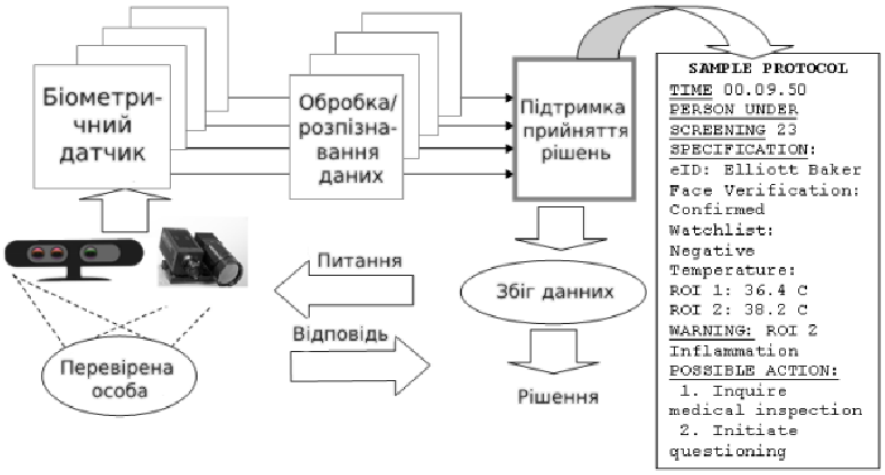

Рисунок 1.6 – Система контролю доступу з використанням біометричних даних обличчя

Іншою перевагою до такої системи є обробка штучних матеріалів (фальшиві вуса, макіяж, перуку тощо) та хірургічні зміни (технології пластичної хірургії). Дані інфрачервоного спектру надають корисну інформацію для виявлення замаскованих ознак.

Окрім автоматичної допомоги, система може аналізувати отримані необроблені зображення у видимому та інфрачервоному спектрах. Дані особи (RGB) перевіряються за даними в eID (наприклад, електронний паспорт), а також порівнюються з глобальними базами даних (такими як списки спостереження). Зауважте, що дані про особу не завжди можуть бути доступні в базі даних – це найгірший сценарій, і розумна підтримка в цьому випадку життєво важлива

Також для моделювання системи було розглянуто використання швидкого алгоритму оцінки пози голови, який виконує обробку зображення в реальному часі за допомогою камери RGB-D[16].

Зокрема, будь-який із поточних датчиків Kinect, Carmine або Asus надає як дані про глибину, так і відео RGB, забезпечуючи роздільну здатність 640 × 480. Замість використання даних глибини для отримання 3D-даних, їх можна використовувати для виконання попередньої обробки та виявлення. Після цього можна продовжити розпізнавання 2D.

Крім того, датчик глибини може допомогти у перевірці живості об’єкта, щоб запобігти фальсифікаціям, показуючи датчику фотографію або зображення людини зі смартфона.

Алгоритм оцінки пози голови, де дані про глибину збираються для оцінки орієнтації голови за допомогою реалізації лісу випадкової регресії. Цей підхід використовується для пошуку найбільш оптимального положення обличчя.

На рисунку 1.7 зображено 3D-модель об’єкта з використанням інформації про глибину з камери Kinect. Зелений циліндр представляє вектор орієнтації людини і малюється шляхом з’єднання центру голови з носом. І центр голови, і ніс розраховуються шляхом підгонки 3D-маски до моделі глибини; маска створюється на основі алгоритму лісу випадкової регресії.

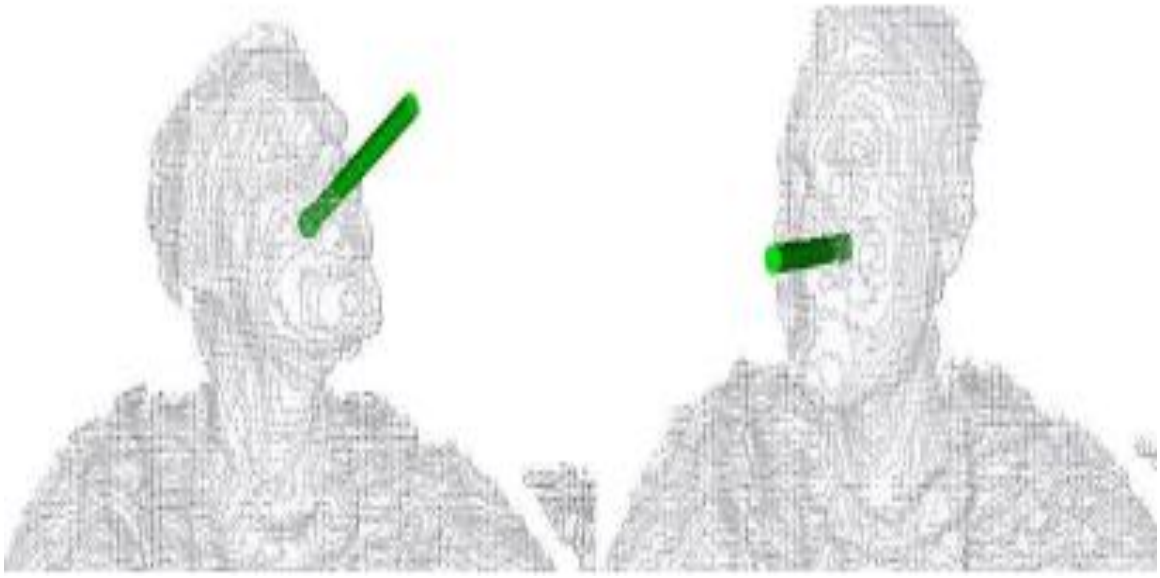

Рисунок 1.7 – Вектор орієнтації голови

Після обчислення вектора нахилу голови, останній порівнюється з віссю z за допомогою скалярного добутку. Це порівняння дає кут зсуву між векторами. Використовуючи кут зсуву, можна фільтрувати зображення переднього виду у відео. Отримання зображень фронтального вигляду має вирішальне значення для точності алгоритму розпізнавання обличчя. Крім того, вибравши перший вид спереду з відео, можна виконати розпізнавання обличчя швидше в

порівнянні з методами, які безперервно виконують розпізнавання протягом відеопослідовності, щоб забезпечити хороші показники розпізнавання[7].

Найпопулярнішим підходом до виявлення обличчя на зображенні є виявлення ознак Хаара, реалізоване в OpenCV. Він передбачає пошук області інтересу (ROI), а потім використання алгоритму ковзного вікна, розмір якого повільно зростає до заданого порогу. Для кожної ROI виявлення функції виконується через каскадний класифікатор.

У багатоканальній системі керування доступом, розпізнавання обличчя реалізовано за допомогою класу FaceRecognizer у OpenCV.

Було оцінено три основні алгоритми розпізнавання:

– EigenFace
– FisherFace
– Локальні бінарні гістограми (LBPH)

Іншою формою виявлення обличчя є використання алгоритму оцінки положення голови. Алгоритм оцінки пози визначає вектор орієнтації голови, а також розташування голови. На основі цих даних обличчя можна обрізати та використовувати для розпізнавання обличчя. Це також допомагає в обробці інфрачервоного зображення обличчя з метою виявлення цікавої області та подальшої оцінки температури обличчя. У даній системі розпізнавання ми замінюємо алгоритм виявлення на оцінку положення голови. Рисунок 1.8 ілюструє весь алгоритм[8].

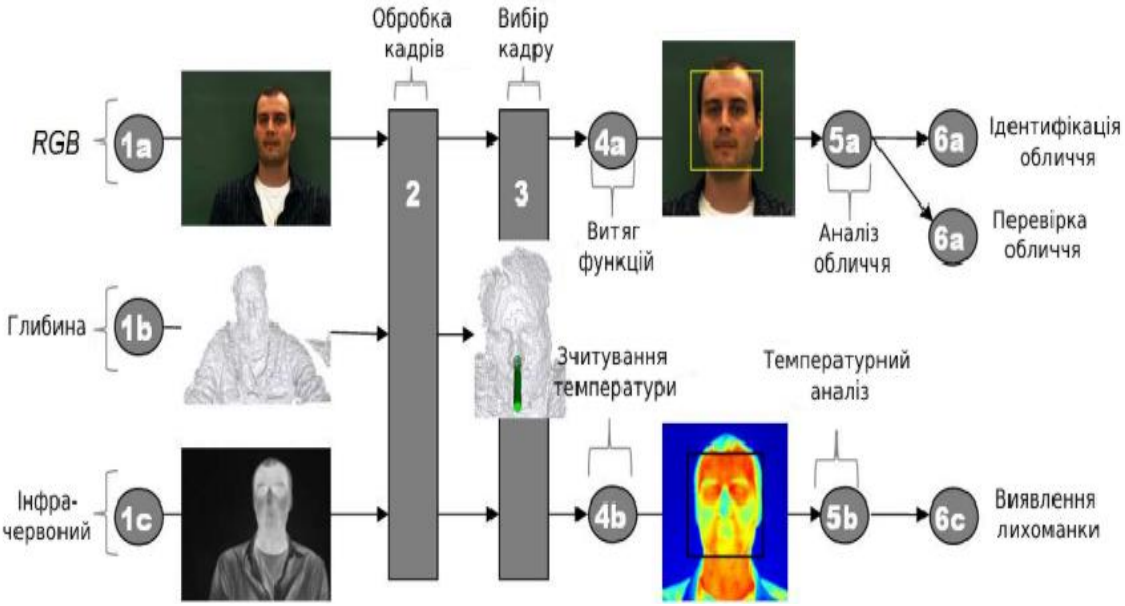

Рисунок 1.8 – Мультиспектральний аналіз біометрії обличчя

Біометрична технологія, яка використовує інфрачервону термографію, походить від медичних застосувань, а саме діагностичних методів, які надають інформацію про нормальне та ненормальне функціонування сенсорної та симпатичної нервової системи, судинну дисфункцію, міофасціальну травму та місцеві запальні процеси[17].

Аналіз інфрачервоного зображення включає запис інфрачервоного відеозображення, обробку інфрачервоного зображення та оцінку, зокрема, температури та швидкості кровотоку. Коливання температури в різних ділянках обличчя в першу чергу пов'язане зі зміною швидкості кровотоку[15].

У контексті систем контролю доступу, які виконують багатоспектральний аналіз біометричних даних людини, температура є важливим джерелом інформації. Наприклад, системи ситуаційної обізнаності та моніторингу подорожуючих у прикордонному контролі мають мати неінвазивне дистанційне вимірювання температури за допомогою ІЧ-камер для виявлення лихоманки.

1.6 Порівняння різних систем ідентифікації за рівнем відмов

Беручи за основу документ “Порівняльне дослідження методів розпізнавання облич”[20] за 2019 рік, було проаналізовано та порівняно точність розпізнавання обличчя та час передбачення. Дані, використані в результатах, взяті з семи різних моделей. Чотири моделі були створені з бібліотеки scikit-learn, дві - з бібліотеки OpenCV і одна - з FaceNet.

Вимір точності розпізнавання обличчя часто є єдиним представленим вимірюванням в роботах в області розпізнавання облич, що може давати викривлені результати. Причина викривленості результатів, полягає в тому, що вона обчислюється шляхом взяття всіх правильних прогнозів і діленням цього значення на загальну кількість. Результати порівняння наведено в Таблиці 1.1

Таблиця 1.1 – Порівняльна таблиця точності розпізнавання обличчя

	FN	CV Eigen	CV Fisher	S Eigen_K	S Eigen_S	S Fisher_K	S Fisher_S
FN							
CV_Eigen			0.696				
CV_Fisher		0.696					
S_Eigen_K							
S_Eigen_S						0.909	1
S_Fisher_K					0.909		0.919
S_Fisher_S					1	0.919	

Односторонній тест ANOVA показує, що існує значуща різниця між методиками що означає, що нульова гіпотеза, h0, відхиляється, оскільки p нижче рівня значущості (F(6,63) = 309, p = .0001). Ці результати лише показують, що існує значна різниця між методиками.

Зелений колір з діагональними лініями показує, що методика в рядку є значно кращою (pvalue менше 0,05), ніж методика в стовпчику. Червоний колір показує, що техніка в рядку значно гірша за техніку у стовпчику. Жовтим кольором позначено техніки які не мають значущої різниці.

Висновки до розділу 1

У першому розділі було оглянуто і проаналізовано існуючі методи контролю доступу, такі як: використання біометрії, PIN-код, механічний замок, сканування сітківки та райдужки ока. Було розглянуто інновації у сфері у сфері систем фізичного контролю доступу, та зроблено висновок, що інтегровані системи контролю фізичного доступу з підтримкою відео та VMS покращують моніторинг безпеки, забезпечують швидший час відгуку та мінімізують ризики злому.

Було розглянуто методи мультиспектральної біометрії обличчя у контролі доступу та проаналізовано алгоритми оцінки пози голови, де дані про глибину збираються для оцінки орієнтації голови за допомогою реалізації лісу випадкової регресії. Та розглянуто підходи, які використовуються для пошуку найбільш оптимального положення обличчя.

Було проаналізовано підходи до виявлення обличчя на зображенні. Найбільш результативним і оптимальним визначено метод виявлення ознак Хаара, реалізоване в OpenCV.

Також у розділі було порівняно різні системи ідентифікації за рівнем відмов у методах обчислення точності розпізнавання обличчя.

РОЗДІЛ 2 ВИБІР МАТЕМАТИЧНИХ МЕТОДІВ ТА ЗАСОБІВ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ДЛЯ МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Основи математичної статистики та теорії ймовірності

Теорія ймовірності в абстрактній формі відображає закономірності, властиві випадковим подіям (явищам) масового характеру. Уявлення про випадкові події базується на життєвому досвіді. Таким чином, на відміну від достовірної події, подія називається випадковою, якщо її настання не можна гарантувати заздалегідь, тобто подія характеризується лише тим, що вона можлива. Предметом теорії ймовірностей є дослідження ймовірнісних закономірностей масових випадкових подій.

Основними поняттями теорії ймовірностей є ймовірнісний експеримент і подія. Імовірнісний експеримент означає забезпечення певної сукупності умов, які або створюються штучно, або виникають незалежно від волі експериментатора, в результаті чого відбувається одна подія з багатьох можливих.

Наприклад, продукт, який виготовляється хвилями по n штук кожна. Перевірка якості одного виробу призводить до його знищення, тому для перевірки якості партії відбирають m виробів (m n). Експеримент складається з відбору предметів із партії та їх тестування. Результатом експерименту (події) є кількість бракованих виробів [21].

Події в теорії ймовірностей зазвичай позначають великими літерами латинського алфавіту: A, B, C і т. д. Події класифікують за рядом ознак: сумісні й несумісні, протилежні, рівноможливі, елементарні. Над подіями введено дві основні операції: додавання і множення.

2.1.1 Базова інформація про систему випадкових процесів

Однолінійна система масового обслуговування працює наступним чином. Сервісний запит, отриманий системою, коли сервісний пристрій вільний, приймається до обслуговування. Інші запити, отримані системою, коли

службовий пристрій зайнятий, ставляться в чергу. Коли черга закінчується, сервісний пристрій вибирає запит із черги та починає його обслуговувати. Якщо запитів немає, сервісний пристрій – неактивний та очікує наступного запиту. Іншими словами, кількість запитів у черзі та в системі постійно змінюється в СМО як під час надходження запитів, так і після закінчення обслуговування запитів.

Як відомо, процес – це послідовна зміна стану об'єкта в часі. Таким чином, в системах масового обслуговування відбуваються різні процеси. На рис. 2.1 показано часові діаграми основних процесів в однолінійній СМО.

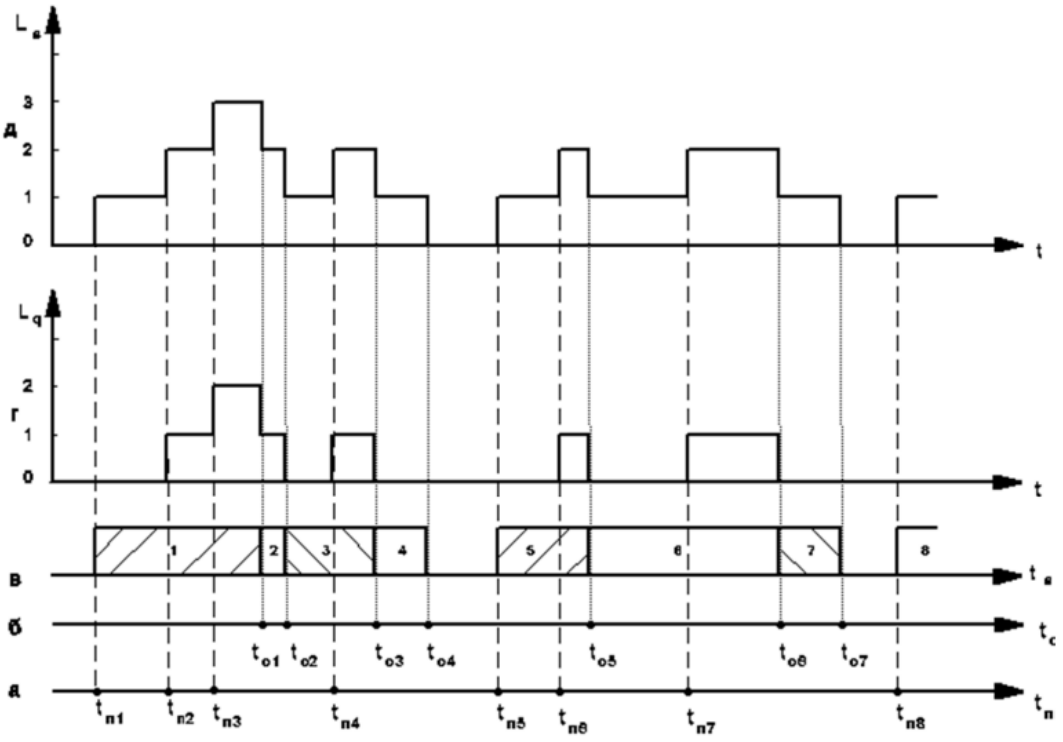

Рисунок 2.1 – Розподіл процесів в однолінійній СМО

Моменти прибуття позначені tпі. З кожним вхідним запитом кількість запитів у системі збільшується, а якщо службовий пристрій зайнятий, то збільшується і довжина черги. З рисунку 2.1 випливає, що стани СМО змінюються у випадкові моменти часу, відповідно всі процеси є імовірнісними і для їх дослідження необхідно використовувати теорію випадкових процесів.

2.1.2 Потоки подій та їх характеристики

Процес надходження запитів на обслуговування в СМО є імовірнісним і формує потік однорідних або різнорідних подій, що відбуваються через випадкові проміжки часу (рис. 2.2, де ti моменти надходження запитів, які утворюють потік випадкових подій; ξi – інтервали часу між сусідніми запитами).

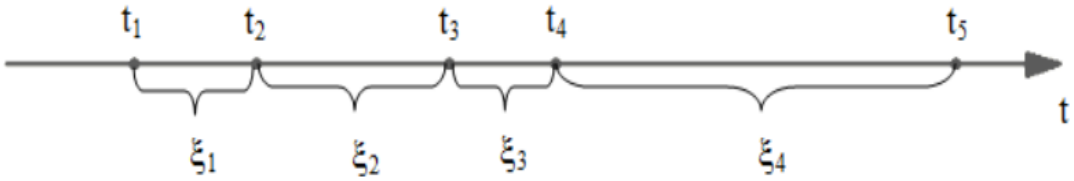

Рисунок 2.2 – Процес надходження запитів для обслуговування

Базовим параметром потоку є інтенсивність потоку λ(t) – математичне сподівання кількості подій за одиницю часу:

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{m_k(t, t+\Delta t)}{\Delta t} \quad ,$$

тобто це межа відношення математичного сподівання групи подій $m_k(t, t+\Delta t)$ на інтервалі (t, t + Δt) до довжини цього інтервалу, яка наближається до нуля.

Якщо значення ξi незалежні та рівномірно розподілені, то такий потік називають потоком з обмеженою послідовністю (потік Пальма). Таким чином, основним описом потоку є функція розподілу A(t) = P{ξ<t}. Такі потоки є фундаментальними в теорії масового обслуговування. Кожен запит на послугу надходить до обслуговуючого пристрою протягом випадкового часу, який описує функцію розподілу B(t) = P{η<t}. Якщо припустити, що на вході обслуговуючого пристрою постійно є додатки, то на виході обслуговуючого пристрою буде формуватися потік обслуговуваних додатків з функцією розподілу B(t). Таким чином, обслуговування виконується так, ніби потік обслуговування спрямоване на запит, отриманий для обслуговування. Така постановка задачі методологічно дуже зручна, оскільки дозволяє узагальнено

розглянути процеси надходження та обслуговування. Таким чином, у системі масового обслуговування розрізняють два типи потоків – надходження та обслуговування, які характеризуються загальними функціями розподілу.

Випадкові проміжки часу між появою подій у потоці можуть характеризуватися різними законами розподілу. Однак, якщо є потреба в роботах з теорії масового обслуговування, особливо прикладних, розглядається пуассонівський потік, у якому ймовірність настання подій в інтервалі часу t задається формулою Пуассона:

$$p_k(t)\frac{(\lambda t)^k}{k!}e^{-\lambda t},$$

Де $\lambda > 0$ – параметр потоку.

Найпростіший потік характеризується трьома основними властивостями: стаціонарністю, відсутністю післядії та ординарністю. Випадковий потік називається стаціонарним, якщо ймовірність отримання певної кількості запитів за певний проміжок часу залежить від його розміру і не залежить від його початкового відліку на осі часу. Це означає, що якщо відкласти часовий інтервал τ на осі неперекриваючих часових рівнів (рис. 2.3), то ймовірність певної кількості заявок на цей потік залежить від значення τ і не залежить від інтервалу розташування на осі часу.

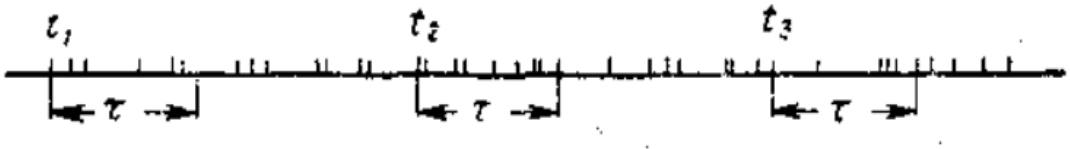

Рисунок 2.3 – Стаціонарний потік однорідних подій

Відсутність наслідків означає, що ймовірність отримання певної кількості запитів протягом інтервалу часу τ не залежить від того, скільки запитів вже надійшло в систему, тобто не залежить від передумови досліджуваного явища. Відсутність післядії передбачає взаємну незалежність процесу протягом непересічних інтервалів часу.

Порядковість потоку вимог означає, що практично неможливо, щоб дві або більше вимог з'явилися одночасно.

Таким чином, найпростіший потік – це стаціонарна, звичайна течія без післядії.

Найпростіший потік в теорії масового обслуговування відіграє ту ж роль, що й нормальний закон розподілу випадкових величин у теорії ймовірностей. У разі підсумовування великої кількості звичайних стаціонарних потоків з практично будь-яким розподілом утворюється потік, близький до найпростішого.

Найпростіший потік відіграє важливу роль у теорії моделювання. По-перше, найпростіші та близькі до найпростіших потоки подій дуже поширені на практиці. По-друге, навіть з потоком подій, який відрізняється від найпростішого, зазвичай можна знайти задовільну точність, замінивши потік будь-якої структури найпростішим потоком з тією ж середньою інтенсивністю.

У практиці моделювання, крім експоненціального розподілу, для опису реальних потоків використовують такі розподіли:

Ерлангівський потік із функцією розподілу:

$$\mathrm{A}(t) = 1 - \sum_{i=1}^{k} \frac{(\lambda \mathrm{t})^{i-1}}{i-1!} e^{-\lambda \mathrm{t}}$$

Розподіл Ерланга формується шляхом підсумовування k інтервалів, кожен з яких розподілений за k-специфічним законом з параметром λ. Таким чином, потік подій, описаний розподілом Ерланга k-го порядку, формується з розподілу Пуассона шляхом видалення всіх подій, які відбуваються між l-ю та (l + k)-ю подіями. Коли k = 1, розподіл Ерланга стає експоненціальним.

Гіперекспоненціальний потік з функцією розподілу:

$$\mathrm{A}\,(\mathrm{t}) = 1 - \sum_{i=1}^{k} a_i e^{-\lambda_i t}, \sum_{i=1}^{k} a_i = 1$$

Потік подій, що описується гіперекспоненціальним розподілом, формується, якщо з імовірністю $a_i, \sum a_1 = 1$, наступний проміжок часу між появою

суміжних подій вибрано шляхом розподілу за експоненціальним законом з параметром λ_1. При k = 1 гіперекспоненціальний розподіл перетворюється на експоненціальний. За допомогою гіперекспоненціального розподілу можна досить точно описати потоки подій, які мають коефіцієнт варіації більше одиниці.

2.2 Теорія систем масового обслуговування

Теорія масового обслуговування – це математичне дослідження формування, функціонування та перевантаження ліній очікування, або черг.

Хтось або щось, що запитує послугу – зазвичай називається клієнтом, завданням або запитом.

Хтось або щось, що виконує або надає послуги – зазвичай це сервер.

Наприклад, якщо розглядати ситуацію з чергою в банку, то клієнтами є люди, які хочуть покласти або зняти гроші, а серверами – банківські касири. Якщо розглядати ситуацію з чергою на принтері, то клієнтами є запити, які були надіслані на принтер, а сервером – принтер (рис 2.4).

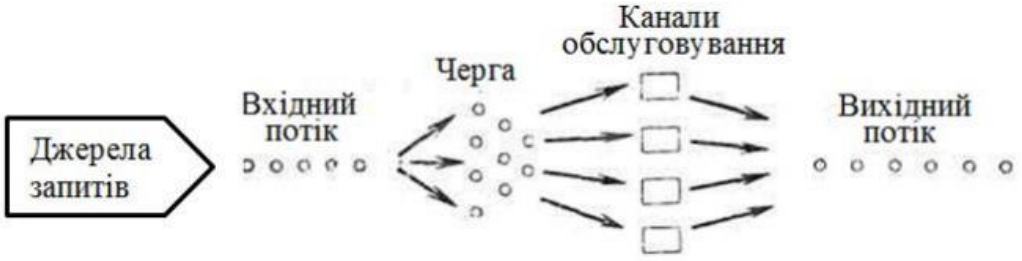

Рисунок 2.4 – Структура СМО

Системи масового обслуговування – це спрощені математичні моделі, що пояснюють затори. У широкому розумінні, система черг виникає щоразу, коли "клієнти" вимагають "послуги" від якогось об'єкта; зазвичай, як прибуття клієнтів, так і час обслуговування вважаються випадковими. Якщо всі "сервери" зайняті, коли прибувають нові клієнти, вони, як правило, чекають у черзі до наступного вільного сервера. Прості системи масового обслуговування визначаються наступним чином:

- схема прибуття клієнтів;

- механізм обслуговування;
- дисципліна черги.

З імовірнісної точки зору, властивості черг зазвичай виводяться з властивостей стохастичних процесів, пов'язаних з ними. Однак у всіх чергах, окрім найпростіших, визначення ймовірностей станів надзвичайно складне.

Часто, однак, можна визначити їх велику часову межу, так званий рівноважний або стаціонарний розподіл. Цей розподіл не залежить від початкових умов системи і є стаціонарним. Ергодичні умови дають обмеження на параметри, за яких система врешті-решт досягне рівноваги.

Здебільшого теорія масового обслуговування займається обчисленням стаціонарних ймовірностей та їх використанням для обчислення інших (стаціонарних) мір ефективності роботи черги. Коли потрібні лише очікувані значення, надзвичайно корисною формулою для систем, що перебувають у рівновазі, є закон Літтла. Більшість зусиль в теорії масового обслуговування було присвячено імовірнісній розробці моделей масового обслуговування та вивченню їх математичних властивостей; тобто параметри, що керують моделями, здебільшого вважаються заданими.

Статистичних досліджень, в яких вводиться невизначеність, порівняно дуже мало. Висновки в системах масового обслуговування не є простими: розробка необхідних вибіркових розподілів може бути дуже складною, і часто аналіз обмежується асимптотичними результатами. Статистичний аналіз є простішим, якщо підходити до нього з байєсівської точки зору. Оскільки байєсівський аналіз нечутливий до (неінформативних) правил зупинки, все, що потрібно від даних – це функція правдоподібності, яка в поєднанні з попереднім розподілом параметрів дає апостеріорний розподіл, на основі якого робляться висновки. Це є важливим спрощенням в аналізі черг, де існує безліч можливих способів спостереження за системою, багато з яких забезпечують пропорційні функції ймовірності, але дуже різні розподіли вибірки. Попередній розподіл кількісно описує все, що відомо про систему до моменту збору даних. Зазвичай про чергу апріорі відомо багато інформації, особливо якщо

припустити, що вона перебуває в рівновазі. Однак можна також зберегти паралелізм з аналізом ймовірностей і уникнути включення додаткових суб'єктивних даних, провівши байєсівський аналіз, який зазвичай називають "об'єктивним", тому що попередній розподіл, який використовується, є "неінформативним" або "об'єктивним" типом. На основі апостеріорного розподілу одразу обчислюються оцінки та стандартні похибки. Також можна обчислити ймовірності, які нас безпосередньо цікавлять (наприклад, ймовірність того, що виконується ергодична умова). Найважливіше те, що обмеження в просторі параметрів, що накладаються припущенням про рівновагу, легко включаються в аналіз. Прогнозування показників перевантаженості системи (кількість клієнтів у черзі, час очікування в черзі, кількість зайнятих серверів і т.д.) здійснюється на основі відповідних прогнозних розподілів, які також дуже корисні для проектування і втручання в систему [19].

Характерною особливістю СМО є імовірнісний характер процесів і можливість формування черги заявок на обслуговування.

Основні поняття системи масового обслуговування:

1. Джерело запиту. Джерело запитів визначається як зовнішня по відношенню до СМО система, з якої надходять запити на обслуговування. Джерело називається нескінченним або скінченним залежно від того, містить воно нескінченну чи скінченну кількість запитів. Якщо джерело містить кінцеву, але досить велику кількість запитів, його зазвичай вважають нескінченним.

2. Вхідний потік. Запити, що надходять з нескінченного джерела, надходять в канал обслуговування в момент часу $t_0 < t_1 < \cdots < t_k$ ... Інтервали часу $\xi_k = t_k - t_{k-1} (k > 1)$ між послідовними моментами надходження запитів є випадковими величинами. Припускається, що ξ_k утворюють послідовність незалежних і однаково розподілених випадкових величин із функцією розподілу A(t) = P$\{\xi_k < t\}$. У математичній теорії масового

обслуговування обмежений набір законів розподілу використовується для опису вхідних потоків. Основними з них є Пуассона.

3. Процес обслуговування. Наприклад, буде взято η_k — час обслуговування для k-го запиту. Вважається, що η_k (k = 1, 2, ...) є незалежними однаково розподіленими випадковими величинами з функцією розподілу $B(t) = p\{\eta_k < t\}$. Функція B(t) називається розподілом тривалості обслуговування. Припускаємо, що існує щільність ймовірності b(t) = B(t). Подібно до опису потоку вхідних запитів, теорія масового обслуговування використовує обмежений набір законів розподілу для опису часу обслуговування. Найбільш поширеним є експоненціальний закон розподілу з функцією розподілу $B(t) = 1 - \exp(-\mu t)$. Параметр μ інтерпретується як інтенсивність обслуговування. Середній час обслуговування дорівнює 1/μ. Крім експоненціального розподілу використовують розподіл Ерланга і гіперекспоненціальний розподіл, які отримують перетворенням експоненціального закону.

4. Сервісні канали. СМО може мати один або декілька каналів обслуговування (ліній, пристроїв). СМО з одним каналом називаються одноканальними або однолінійними, тоді як системи обслуговування з більшою кількістю каналів обслуговування називаються багатоканальними або багатоканальними. Сервісні пристрої можуть бути однорідними і різнорідними. У однорідній системі обслуговування всі пристрої обслуговують запити однаково. У СМО з різнорідними пристроями пристрої відрізняються один від одного за деякими параметрами, наприклад інтенсивністю обслуговування.

5. Службова дисципліна. Правилом, за яким відбираються заявки на службу, є службова дисципліна. Умовно всі службові дисципліни поділяють на дві групи за перевагами надання послуг: непріоритетні та пріоритетні. Кожна з цих груп поділяється на ряд підгруп.

Непріоритетні дисципліни обслуговування поділяються на дисципліни обслуговування першим прийшов, першим обслужено, зворотне, випадковий вибір із черги та циклічні дисципліни обслуговування.

Дисципліна першим прийшов перший обслужений (FIFO) є найпоширенішою, для цієї дисципліни проводиться більшість досліджень з теорії масового обслуговування. Він широко використовується в операційних системах.

Дисципліна обслуговування в зворотному порядку (LIFO) використовується в операційних системах при обробці переривань і організації стеків. При обробці інформації в режимі поділу часу використовуються циклічні дисципліни обслуговування. При пріоритетних дисциплінах обслуговування першими з черги обслуговування вибираються програми з вищим пріоритетом. Вони поділяються на дисципліни з фіксованими та динамічними пріоритетами.

При відносному пріоритетному обслуговуванні не допускається переривання обслуговування запитів на каналі. Якщо система з абсолютним пріоритетом обслуговування отримує запит із вищим пріоритетом, ніж той, що обслуговується, вона припинить обслуговувати цей запит і перейде до служби.

Системи з абсолютним пріоритетом розрізняють за кількістю рівнів пріоритету, а також за алгоритмами обслуговування перерваних запитів.

При динамічних пріоритетних дисциплінах обслуговування, пріоритет конкретних запитів змінюється в залежності від зміни певних значень, наприклад, часу очікування в черзі.

2.2.1 Ефективність СМО у технічних і економічних аспектах

Показники ефективності систем масового обслуговування поділяються на технічні показники, що характеризують якість і умови роботи транзитної системи, та економічні показники, які відображають економічні особливості системи.

До технічних показників відносяться наступні:

− Імовірність збою обслуговування. Імовірність того, що запит що увійшовши в систему, відмовиться стати в чергу і буде втрачено системою є P_{vidm}. Цей показник для системи масового обслуговування з відмовами

дорівнює ймовірності того, що система містить стільки ж запитів скільки і каналів обслуговування;

- середня кількість запитів, що очікують на обслуговування;
- відносна і абсолютна пропускна здатність системи;
- середня кількість каналів, зайнятих послугою для систем;
- загальна кількість вимог, що містяться в системі Ls;
- середній час очікування запитів на початок обслуговування.

Показники, що характеризують економічні особливості СМО, зазвичай формуються відповідно до певного типу системи та її призначення.

Одним із загальних показників є економічна ефективність системи:

$$G = \lambda P_{obsl} C_s T - E \ ,$$

де C_s– середній економічний ефект, отриманий при обслуговуванні однієї потреби; T – розглянутий інтервал часу; E – величина втрат в системі.

2.2.2 Багатоканальна система масового обслуговування з обмеженою чергою

У кваліфікаційній роботі будемо розглядати та брати за основу пуассонівську систему масового обслуговування з **n** каналами обслуговування та кількістю місць очікування **m**. Позначимо через {i}, i = 1, 2, ...,(n+m-1), (n+m) стан системи, коли є **i** запитів. Якщо i ≥ **n**, то буде обслужено **n** запитів, а решта стоять у черзі. Імовірність стану {i} позначається p_i. Графік станів і переходів для цієї системи продемонстровано на Рисунку 2.5.

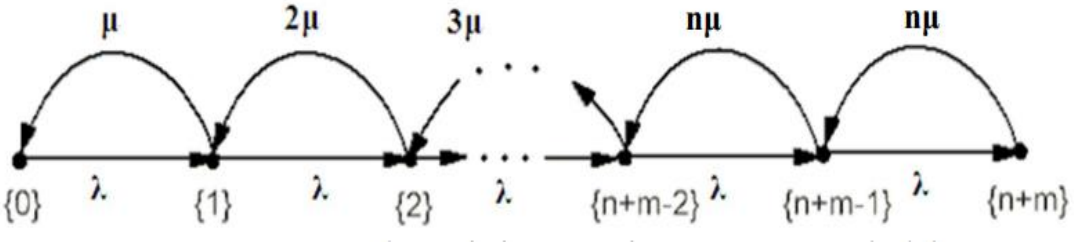

Рисунок 2.5 – Граф станів і переходів для системи M/M/n/m

Першим кроком було складено систему лінійних рівнянь для ймовірностей p_i, $i = 1,2, \dots, n+m$:

$$-\lambda p_0 + \mu p_1 = 0$$

$$-(\lambda + i\mu)p_i + \lambda p_{i-1} + (i+1)\mu p_i = 0, \qquad 0{,}1 \le i < n$$

$$-(\lambda + n\mu)p_i + \lambda p_{i-1} + n\mu p_i = 0, \qquad n \le i < n+m$$

$$-n\mu p_{n+m} + \lambda p_{n+m-1} = 0$$

Система доповнюється умовою нормування:

$$p_0 + p_1 + p_2 + \cdots + p_{n+m-1} + p_{n+m} = 1$$

Наступний крок — розділити кожне рівняння на nμ і ввести позначення ρ = λ / nμ — робоче навантаження всієї системи. Система буде переписана наступним чином:

$$-\rho p_0 + \mu p_1/n = 0,$$

$$-\left(\rho + \frac{i}{n}\right)p_i + \rho p_{i-1} + (i+1)n * p_i = 0, \qquad 0{,}1 \le i < n$$

$$-(\rho + 1)p_i + \rho p_{i-1} + p_i = 0, \qquad n \le i < n+m$$

$$-p_{n+m} + \rho p_{n+m-1} = 0$$

Перше рівняння системи показує, що: $p_1 =$ nρp_0. Підставимо цей вираз у друге рівняння системи і отримаємо $p_2 = \left(\frac{n^i}{i!}\right)\rho^i p_0$.

Продовжуючи цю операцію, отримуємо:

$$p_i = \left(\frac{n^i}{i!}\right)\rho^i p_0, 1 \le i \le n,$$

$$p_i = \left(\frac{n^n}{n!}\right)\rho^i p_0, n < i \le n+m.$$

Підставляючи p_i в умови нормування і знаходимо вираз для ймовірності простою системи:

$$p_0[1 + \sum_{i=1}^{n-1}\frac{n^i\rho^i}{i!} + \sum_{i=n}^{n+m}\frac{n^i\rho^i}{n!}]^{-1} = [\sum_{i=0}^{n-1}\frac{n^i\rho^i}{i!} + \frac{n^n}{n!}\frac{\rho^n(1-\rho^{m+1})}{1-\rho}]^{-1}.$$

Таким чином, визначено всі ймовірності стану. Визначимо основні показники цієї системи. Ймовірність відмови системи становить:

$$P_{vidm} = p_{m+n} = \left(\frac{n^n}{n!}\right)\rho^{n+m}p_0$$

Цей вираз означає, що запит буде відхилено в обслуговуванні, якщо всі лінії та зони очікування зайняті. Кількість запитів, яким буде відмовлено в обслуговуванні протягом часу T – N_{vidm} , можна обчислити за виразом:

$$N_{vidm} = \lambda p_{m+n} T = \lambda(n^n / n!)\rho^{n+m} p T_0$$

Імовірність того, що запит буде обслужено системою, –

$$P_{obsl} = 1 - P_{vidm} = 1 - (n^n/n!)\rho^{n+m}\rho_0$$

Крім того, це відносна пропускна здатність системи, яка дорівнює середній частці запитів, отриманих системою і обслужених нею.

Абсолютна пропускна здатність дорівнює середньому числу запитів, які може обробити SMO за одиницю часу, тобто це частка вхідних запитів, які надходять у систему та обслуговуються нею:

$$A = \lambda P_{obsl} = \lambda 1 - (n^n/n!)\rho^{n+m}\rho_0$$

Наступним кроком було визначено середню кількість запитів, що очікують на обслуговування L_q. Якщо СМО знаходиться в стані $\{ n+1 \}$, то в черзі буде один запит, імовірність цього дорівнює p_{n+1}. Якщо СМО знаходиться в стані $\{ n+m \}$, то в черзі буде m запитів, імовірність цього дорівнює p_{n+m}. Середня кількість заявок у черзі на обслуговування дорівнює математичному сподіванню кількості заявок у черзі і може бути розрахована за формулою:

$$L_q \sum_{i=1}^{m} i * p_{n+i} + \sum_{i=1}^{m} i * -(n^n/n!)\rho^{n+m}\rho_0 = (n^n/n!)\rho^{n+1}(1 + 2\rho + \cdots + \mathrm{m}\rho^{m-1})\rho_0$$

Якщо $\rho = 1$, то вираз у дужках перетворюється на арифметичну прогресію $1 + 2 + 3 + \cdots + m$. Його сума дорівнює $m(m+1)/2$. Тоді:

$$L_q = 0{,}5(n^n/n!)m(m+1)\rho_0$$

Припустимо що $\rho \neq$. Далі було спрощено вираз у дужках за виразом:

$$1 + 2\rho + 3\rho^2 + \cdots + m\rho^{m-1} = \frac{\mathrm{d}}{\mathrm{d}\rho}(\rho + \rho^2 + \rho^3 + \cdots + \rho^m)$$

Вираз у дужках є скінченною геометричною прогресією зі знаменником ρ. Сума його доданків:

$$\rho + \rho^2 + \rho^3 + \cdots + \rho^m = \frac{\rho(1-\rho^m)}{1-\rho}$$

Далі було обчислено похідну:

$$\frac{\mathrm{d}}{\mathrm{d}\rho}\left(\frac{\rho(1-\rho^m)}{1-\rho}\right) = \frac{1-\rho^m(1+m-m\rho)}{(1-\rho)^2}$$

$$L_q = \begin{cases} \dfrac{n^n}{n!}\dfrac{1-\rho^m(1+m-m\rho)}{(1-\rho)^2}\rho^{n+1}\rho_0, & \rho \neq 1 \\ 0{,}5\left(\dfrac{n^n}{n!}\right)m(m+1)\rho_0, & \rho = 1 \end{cases}$$

На рисунку 2.6 наведено графік залежності середньої кількості заявок, що очікують на обслуговування, від завантаження всієї СМО ρ при різних значеннях кількості каналів n і однаковій кількості місць очікування m=7.

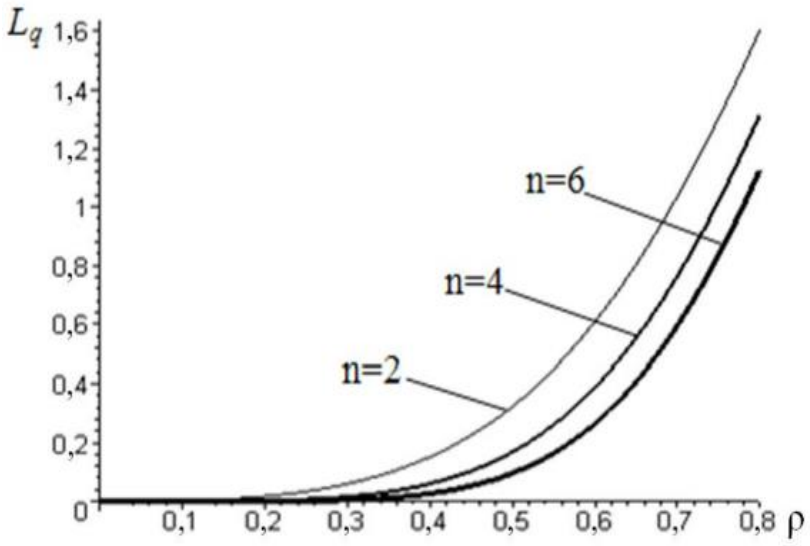

Рисунок 2.6 – Залежність L_q від завантаженості системи ρ при різних n

Зрозуміло, що при значені – 0,5, довжина черги різко зростає; при збільшені n і однаковому значенні ρ L_q також різко зростає, а потім зростання сповільнюється.

На рис. 2.7 наведено графік залежностей середньої кількості запитів, що очікують на обслуговування, від завантаження всієї СМО ρ при різних

значеннях кількості місць очікування каналу n і однаковій кількості каналів n = 5.

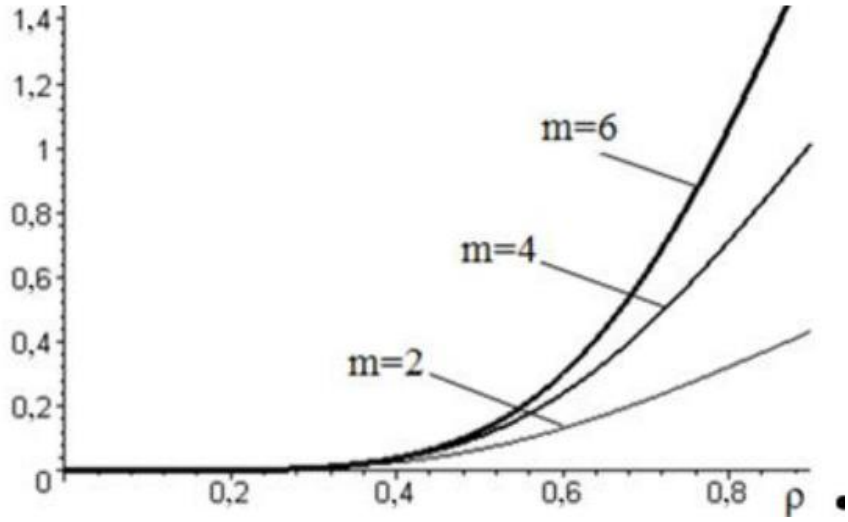

Рисунок 2.7 – Залежність L_q від завантаження системи ρ при різних m

Оптимізацію можна проводити або за кількістю каналів обслуговування n, або за інтенсивністю обслуговування μ, тільки якщо цей параметр можна змінити[22].

2.3 Технологія OpenCV

У сфері штучного інтелекту комп’ютерний зір є одним із найцікавіших і найскладніших завдань. Комп’ютерне бачення діє як міст між комп’ютерним програмним забезпеченням і візуалізаціями. Комп’ютерне бачення дозволяє комп’ютерному програмному забезпеченню розуміти та вивчати візуалізацію навколишнього середовища.

OpenCV (Open Source Computer Vision) — це популярна бібліотека комп’ютерного бачення, створена компанією Intel у 1999 році. Ця крос-платформна бібліотека зосереджена на обробці зображень у реальному часі та включає безпатентні реалізації найновіших алгоритмів комп’ютерного бачення. OpenCV підтримується різними мовами програмування, такими як R, Python тощо. Можливо, він працює на більшості платформ, таких як Windows, Linux і macOS[10].

OpenCV 2.4 тепер постачається з новим класом FaceRecognizer для розпізнавання обличчя, тому з’являється можливість експериментувати з розпізнаванням обличчя відразу.

Наразі доступні такі алгоритми:

– Власні обличчя;

– Fishherfaces;

– Гістограми локальних бінарних шаблонів.

Існує багато програм, які розв’язуються за допомогою OpenCV, деякі з них наведено нижче:

– Розпізнавання обличчя;

– автоматизований огляд і спостереження;

– кількість людей – підрахунок (пішохідний рух у торговому центрі тощо);

– підрахунок транспортних засобів на магістралях разом із їх швидкістю;

– інтерактивні арт-інсталяції;

– виявлення аномалій (дефектів) у виробничому процесі (дивні браковані вироби);

– зшивання зображень перегляду вулиць;

– пошук і пошук відео/зображень;

– навігація та керування автомобілем без роботи та водія;

– розпізнавання об'єктів;

– аналіз медичного зображення;

– фільми – 3D структура з руху;

– розпізнавання реклами телеканалів.

Функціональність OpenCV:

– Введення/виведення зображення/відео, обробка, відображення (ядро, imgproc, highgui);

– виявлення об’єктів/функцій (objdetect, features2d, nonfree);

– монокулярний або стереокомп’ютерний зір на основі геометрії (calib3d, зшивання, videostab);

– комп'ютерна фотографія (фото, відео, суперзйомка);

– машинне навчання та кластеризація (ml, flann);

– прискорення CUDA (gpu)класифікацію.

OpenCV використовується великими підприємствами та державними установами, наприклад Google, Toyota, IBM, Microsoft, SONY, Siemens і Facebook. Крім того, відомі стартапи комп’ютерного бачення використовують OpenCV для створення потужних продуктів комп’ютерного бачення та рішень ШІ, зокрема viso.ai. Багато дослідницьких центрів використовують OpenCV, наприклад Stanford, MIT, INRIA, Cambridge та CMU. Сфери використання комп’ютерного зору величезні. Хоча більшість знають про досить популярні варіанти використання в системах безпеки та відеоспостереження чи в безпілотних автомобілях, менше людей можуть побачити варіанти використання в конкретних галузях, таких як промислове виробництво чи роздрібна аналітика. Швидкий розвиток комп’ютерного зору за останні роки дав змогу компаніям у різних галузях розробляти спеціальні програми комп’ютерного зору, які вирішують вузькоспеціальні проблеми (виявляють проблеми з продуктом, підраховують об’єкти, аналізують поведінку тощо)[11].

Він має інтерфейси C++, Python, Java і MATLAB і підтримує Windows, Linux, Android і Mac OS. OpenCV здебільшого орієнтується на програми бачення в реальному часі та використовує переваги інструкцій MMX і SSE, якщо вони доступні. Зараз активно розробляються повнофункціональні інтерфейси CUDA і OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які створюють або підтримують ці алгоритми. OpenCV написаний на C++ і має шаблонний інтерфейс, який бездоганно працює з контейнерами STL.

Комп’ютерне бачення є нетривіальним, і досягнення високої продуктивності за прийнятною ціною є основою масштабованого комп’ютерного бачення. Під час роботи з камерами дані зображення часто

спотворюються шумом і спотвореннями, що виникають через зміни у фізичному світі (освітлення, відбиття, рух, недоліки об’єктива (викривлення, поле зору), сенсора або механічних налаштувань (кут, положення) , висота). Щоб подолати ці складні проблеми, розробникам потрібно створити складні конвеєри комп’ютерного зору, які моделюють шлях потоків даних. Ця логіка програми поєднує різні завдання, від отримання кадрів до їх попередньої обробки (усунення шумів, фільтрація, усунення викривлень тощо).) і об’єднання їх в один або кілька алгоритмів бачення. OpenCV надає стандартний набір інструментів для розробників для вирішення проблем комп’ютерного зору. У деяких випадках функціональних можливостей високого рівня в бібліотеці буде достатньо для вирішення складніших проблем бачення ШІ.

Однак, написання звичайного коду може швидко стати складним, його важко зрозуміти, підтримувати чи оновлювати, коли змінюються бізнес-вимоги чи правила. Під час розробки платформа Viso Suite використовує технологію без коду для використання можливостей OpenCV. Це дозволяє розробникам візуально будувати конвеєри комп’ютерного зору за допомогою модульних будівельних блоків. Редактор без коду та автоматизоване керування моделлю штучного інтелекту полегшують співпрацю та значно пришвидшують створення та підтримку конвеєрів комп’ютерного зору. Використання OpenCV без коду приносить користь як експертам із бачення, так і розробникам із базовими знаннями.

OpenCV є найбільш універсальним інструментом комп’ютерного зору, який використовується в широкому спектрі задач комп’ютерного зору, починаючи від розпізнавання зображень, 2D або 3D аналізу до відстеження руху, розпізнавання обличчя тощо.

Основні можливості технології:

1. Технологія виявлення об’єктів використовується для застосування розпізнавання зображень і визначення місцезнаходження певних об’єктів у відеоданих або зображеннях, таких як автомобілі, люди, тварини та окремі

частини або обладнання в промисловому виробництві. Виявлення кількох об’єктів на основі ML із OpenCV – створено за допомогою Viso Suite

2. Сегментація зображення застосовує алгоритми обробки зображення для поділу зображення на різні сегменти. Сегментація зазвичай застосовується для спрощення, зміни або покращення зображення, часто в поєднанні з подальшими завданнями комп’ютерного зору. Прикладом є автономне водіння, де для визначення дороги використовується сегментація зображення.

3. Розпізнавання поз і жестів людини використовуються для інтерпретації та розуміння жестів людей за допомогою аналізу відео. Рухи тіла, рук або обличчя можна розпізнавати та класифікувати, щоб призначити попередньо визначену категорію. Аналіз рухів часто є частиною оцінки пози для аналізу рухів тіла з опорними ключовими точками (суглоби, кінцівки). Розрахунок пози об'єкта дозволяє зрозуміти, як об'єкт розташований у 3D просторі, наприклад, як він обертається. Аналіз руху з оцінкою пози за допомогою ключових точок – створено на Viso Suite

4. Автоматичне розпізнавання облич використовується для ідентифікації людей шляхом виявлення людського обличчя та зіставлення його з базою даних на основі виявлених рис обличчя. FaceRecognizer OpenCV надає набір популярних алгоритмів розпізнавання обличчя для використання в реальних програмах. Розпізнавання обличчя за допомогою Computer Vision

5. Доповнена реальність Доповнена реальність (AR) дозволяє взаємодіяти в реальному часі між реальним світом і віртуальним світом. Таким чином, доповнена реальність має на меті доповнити фізичний світ навколо нас перцептивною інформацією, створеною комп’ютером.

Дослідивши вище описану інформацію, можна зробити висновок, що використання технології OpenCV, буде найбільш успішним вибором для створення багатоканальної системи керування доступом на базі Raspberry Pi.

Розпізнавання обличчя має вирішальне значення у повсякденному житті для того, щоб ідентифікувати родину, друзів чи когось із кого ми знайомі. Людський інтелект дозволяє нам отримувати інформацію та інтерпретувати

інформацію в процесі розпізнавання. Ми отримуємо інформацію через зображення, яке проектується в наші очі, зокрема через сітківку у вигляді світла. Світло – це форма електромагнітних хвиль, які випромінюються від джерела на об’єкт і проектуються на людський зір. Робінсон-Ріглер, Г. та Робінсон-Ріглер, Б. (2008) згадували, що після візуальної обробки, проведеної зоровою системою людини, ми фактично класифікуємо форму, розмір, контур та текстуру об'єкта для аналізу інформації. Проаналізовану інформацію буде порівняно з іншими зображеннями предметів або обличчя, які існують у нашій пам’яті для розпізнавання. Насправді скласти автоматизовану систему, яка має таку ж здатність, що і людина, розпізнавати обличчя, важко. Однак нам потрібна велика пам’ять, щоб розпізнавати різні обличчя, наприклад, в університетах є багато студентів з різною расою та статтю, неможливо запам’ятати кожне обличчя людини без помилок. Для подолання людських обмежень у системах розпізнавання обличь використовуються комп’ютери з майже безмежною пам’яттю, високою швидкістю обробки та потужністю.

Людське обличчя – це унікальне зображення індивідуальної ідентичності. Таким чином, розпізнавання обличчя визначається як біометричний метод, при якому ідентифікація особи здійснюється шляхом порівняння зображень у реальному часі із захопленими зображеннями, що зберігаються в базі даних людини.

У наш час система розпізнавання обличь є поширеною завдяки своїй простоті та надзвичайній продуктивності. Наприклад, системи захисту аеропортів та ФБР використовують розпізнавання обличчя для розслідування кримінальних справ шляхом відстеження підозрюваних, зниклих дітей. Окрім цього, Facebook, який є популярним веб-сайтом соціальних мереж, реалізує розпізнавання облич, щоб дозволити користувачам позначити своїх друзів на фотографії. Крім того, компанія Intel дозволяє користувачам використовувати розпізнавання обличь, щоб отримати доступ до свого онлайн-рахунку. Apple дозволяє користувачам розблокувати свій мобільний телефон iPhone X за допомогою розпізнавання обличчя.

Робота з розпізнавання обличчя розпочалася в 1960 році. Вуді Бледсо, Хелен Чан Вольф та Чарльз Біссон запровадили систему, яка могла виявляти очі, вуха, ніс та рот на зображеннях. Потім обчислюється та порівнюється відстань та співвідношення між розташованими об’єктами та загальними орієнтирами. Дослідження вдосконалено Гольдштейном, Гармоном та Леском у 1970 р., Використовуючи інші функції, такі як колір волосся та товщина губ для автоматизації розпізнавання. У 1988 році Кірбі та Сірович вперше запропонували аналіз принципових компонентів (PCA) для вирішення проблеми розпізнавання обличчя.

Висновки до розділу 2

При виборі математичних методів для створення багатоканальної системи було розглянуто основи математичної статистики та теорії ймовірності. Детальніше було розглянуто базову інформацію про систему випадкових процесів та потоки подій на їх характеристики. Роблячи висновки з розглянутої інформації перевагу було надано теорії систем масового обслуговування. Теорія масового обслуговування – це математичне дослідження формування, функціонування та перевантаження ліній очікування, або черг.

Так як, багатоканальна система керування доступом може виконувати декілька дій одночасно, тому необхідно використовувати методи, які б справлялися з чергами. Було розглянуто характерні особливості СМО та характер процесів і можливість формування черги заявок на обслуговування.

Для більш детального ознайомлення з теорією було також розглянуто ефективність СМО у технічних і економічних сферах та також використання системи для створення багатоканальної системи масового обслуговування.

Також, було наведено детальну інформацію про технологію OpenCV, її функціональність і характеристики. Дослідивши вище описану інформацію, можна зробити висновок, що використання технології OpenCV, буде найбільш успішним вибором для моделювання багатоканальної системи керування доступом на базі Raspberry Pi.

РОЗДІЛ 3 МОДЕЛЮВАННЯ ПРОЦЕСІВ ТА ОГЛЯД АЛЬТЕРНАТИВНИХ АПАРАТНИХ РІШЕНЬ

3.1 Моделювання підключень компонентів і блок-схема процесів багатоканальної системи

Багатоканальна система надання доступу може отримувати дані з двох або більше джерел вхідних даних. ЇЇ структурна схема зображена на Рисунку 3.1.

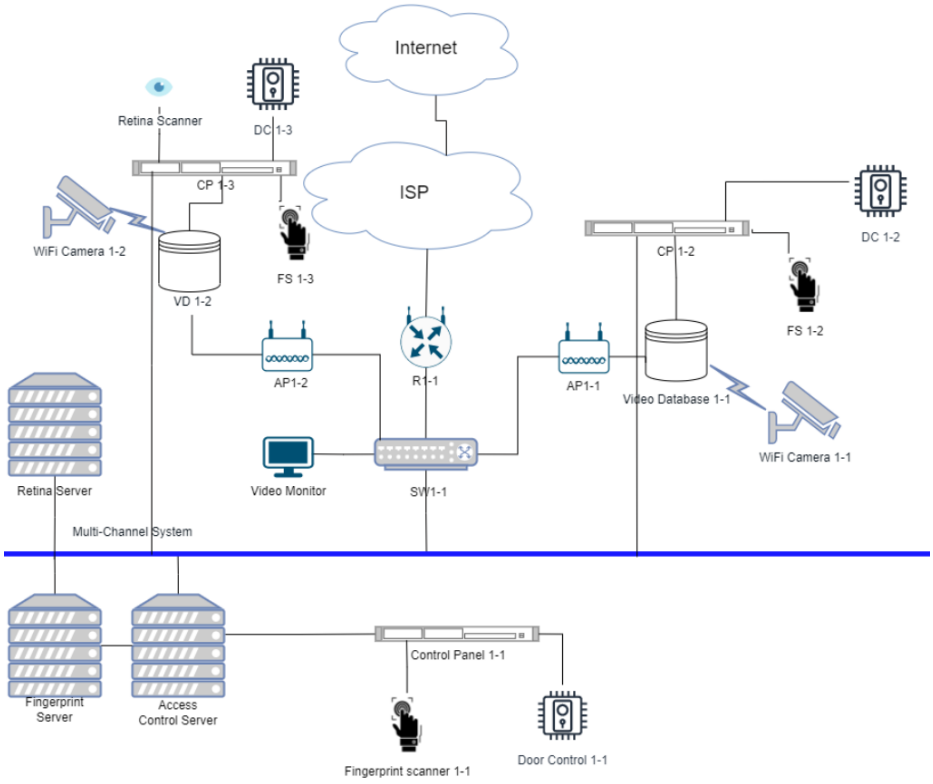

Рисунок 3.1 – Структурна схема багатоканальної системи контролю доступу

Змодельована схема демонструє принципи роботи багатоканальної системи та її функції. Система має три рівні безпеки. Перший рівень спрямований для осіб першої групи, для отримання доступу яким, необхідно лише виконати перевірку сканування відбитку пальця. Якщо данні збігаються з тим, що були задані у сервері, то буде надано доступ автоматично через панель керування.

Другий рівень являє собою подвійний рівень безпеки, який можуть мати користувачі, які потребують доступ до більш захищених приміщень, наприклад системні адміністратори або охорона. У такому випадку для отримання доступу необхідно зробити скан відбитку пальця, та отримання збігу з базою данних через WiFi камеру. Після цього, доступ буде отримано.

Наступний рівень, підходить для директора компанії для отримання доступу до кабінету або сейфу, також може підходити до серверної, та відділу фінансів. Цей рівень потребує перевірки через відбиток пальця, збігу обличчя через технологію OpenCV, та скану райдужки ока. Алгоритм роботи багатоканальної системи продемонстровано на Рисунку 3.2.

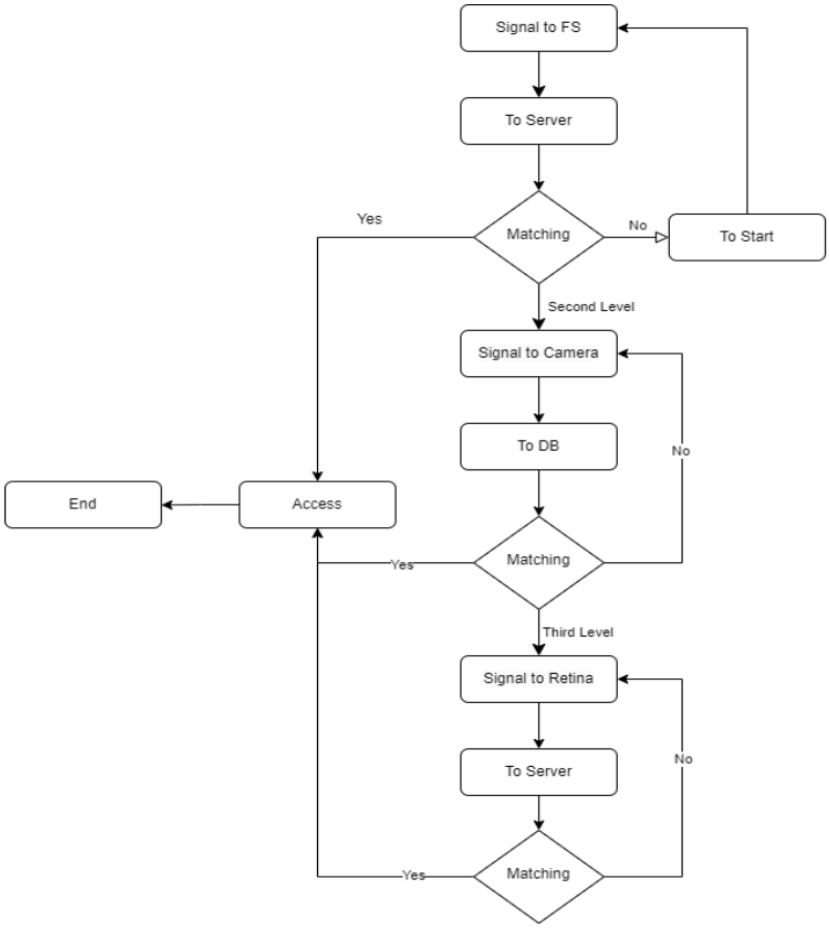

Рисунок 3.2 – Блок-схема алгоритму роботи системи.

Блок-схема була створення у веб додатку Drawio [24] та демонструє основний принцип роботи системи. Перший рівень являє собою отримання сигналу відбитку пальця (FS), який проходить перевірку у базі данних на

сервері. Якщо сигнал збігається з заданим, при першому рівні безпеки, то доступ буде надано. При другому або третьому рівні, необхідна буде перевірка обличчя за допомогою OpenCV. Якщо сигнал співпадає з одним із заданих у базі данних (DB) то наступним, то доступ буде отримано, або буде перехід на наступний рівень. При незадовільному результаті, буде наступна спроба. На третьому рівні, потрібен скан райдужки ока, при позитивному результаті, буде отримано доступ. При негативному – повторна спроба.

Система є гнучкою та дозволяє змінювати компоненти згідно з потреб, наприклад, відбиток пальця на чіп-карту, або ключ-код. Також може бути замінено скан райдужки ока на сканування сітківки, що значно покращить систему, але це збільшить витрати.

Це надає повну безпеку фізичних даних, від зловмисників. Так як, системи безпеки і виклику охорони працюють дуже швидко, то при пограбуванні фізичний спосіб отримання доступу буде складним і довготривалим. Тому, у випадку з сейфом, багатоканальна система безпеки може бути складним завданням, так як при ситуації отримання якихось з ідентифікаторів, сейф все одно буде зачинений. Це може бути чинником розгублення, і наведення на думку, що система заблокована і немає можливості її відкриття за допомогою ідентифікаторів, а тільки фізично. Це дає змогу виграти додаткову частку часу для приїзду охорони, або поліції.

3.2 Огляд альтернативних апаратних рішень

При виборі одноплатного міні-ПК, було враховано альтернативи інших виробників. У цьому розділі детально представлені та детально враховано альтернативи одноплатні плати різних виробників.

Вибираючи найкращий одноплатний комп'ютер для моделювання системи, було враховувано пріоритети. Було приділено увагу вартості: найдешевші одноплатні комп'ютери коштують близько 5 доларів США, а дорогі, але потужні плати для розробників – кілька сотень доларів.

Для базових завдань, які не потребують великої обчислювальної потужності, підійде майже будь-яка плата. Але для багатозадачності або розширеної обробки даних потрібен більш потужний пристрій.

Не менш важливою є сумісність базової архітектури та операційної системи (ОС). Було приділено увагу підтримці операційних систем, наприклад, Window. Тож, основними критеріями вибору одноплатного комп'ютера є:

- вартість;
- передбачуване використання;
- розмір;
- базова архітектура – ARM;
- бажана ОС.

3.2.1 Raspberry Pi 4

Raspberry Pi 4, є найкращим одноплатним комп'ютером для більшості користувачів. Хоча існують більш потужні SBC, співвідношення ціни та продуктивності, Raspberry Pi 4 вважається найкращим варіантом. З кількома різними моделями, що пропонують на вибір 2 ГБ, 4 ГБ або 8 ГБ оперативної пам'яті, і низькою стартовою ціною всього в 35 доларів, Pi 4 – це багатофункціональний пристрій розміром з кредитну картку. Ця плата є доступна для виробників усіх рівнів кваліфікації завдяки своїй доступності та універсальності. Raspberry Pi Foundation пропонує безліч офіційних ресурсів від форумів до операційної системи на базі Linux в Raspberry Pi OS. Крім того, є наявна підтримка спільноти з безліччю сторонніх ресурсів, починаючи від книг, апаратних аксесуарів та операційних систем і закінчуючи форумами, блогами та субредакціями (Рис. 3.3).

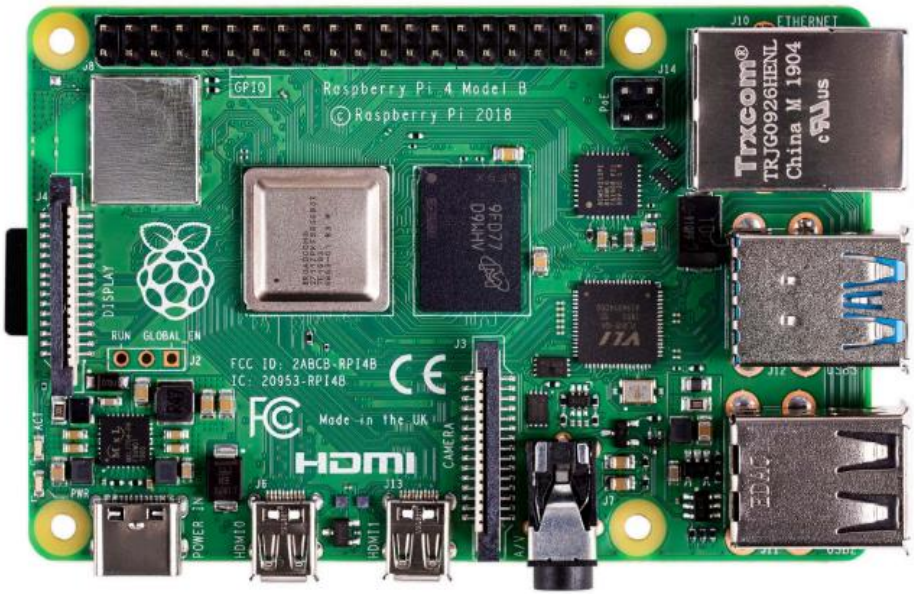

Рисунок 3.3 – Raspberry Pi 4 Model B

Крім того, Raspberry Pi 4 надзвичайно добре працює для безлічі різних застосувань. Він однаково добре підходить як заміна настільного комп'ютера для написання, редагування, редагування зображень і подкастингу, так і для емуляції. На її основі можна створити медіа-сервер, який насправді є досить компетентним, або дешевий файловий сервер, альтернативний до готового мережевого пристрою зберігання даних (NAS). Розумні домашні концентратори, сервери керування 3D-друком і домашні кінотеатри (HTPC) з такими програмами, як Kodi, працюють чудово. Raspberry Pi 4, незважаючи на свої можливості, не здатний виконувати деякі завдання. Одноплатний комп’ютер добре задокументований, і Pi здатен виконувати багато завдань. Наприклад, можна запустити Linux, гру з RetroPie, зробити робот-автомобіль або створення проєктів зі штучним інтелектом.

Плюси:

- універсальний;
- доступна ціна;
- кілька різних варіантів оперативної пам'яті;
- простий у використанні;
- відмінне співвідношення ціни та продуктивності;

- невеликий розмір;
- низьке енергоспоживання.

3.2.2 Одноплатний комп’ютер UDOO Bolt

UDOO Bolt виділяється як найпотужніший одноплатний комп'ютер у 2021 році. Він займає більше місця, ніж Raspberry Pi та інші невеликі SBC. UDOO Bolt – це SBC на базі x86. Таким чином, це одноплатний комп'ютер підтримує Windows 10 і дистрибутиви Linux, такі як Ubuntu та Debian (Рис. 3.4).

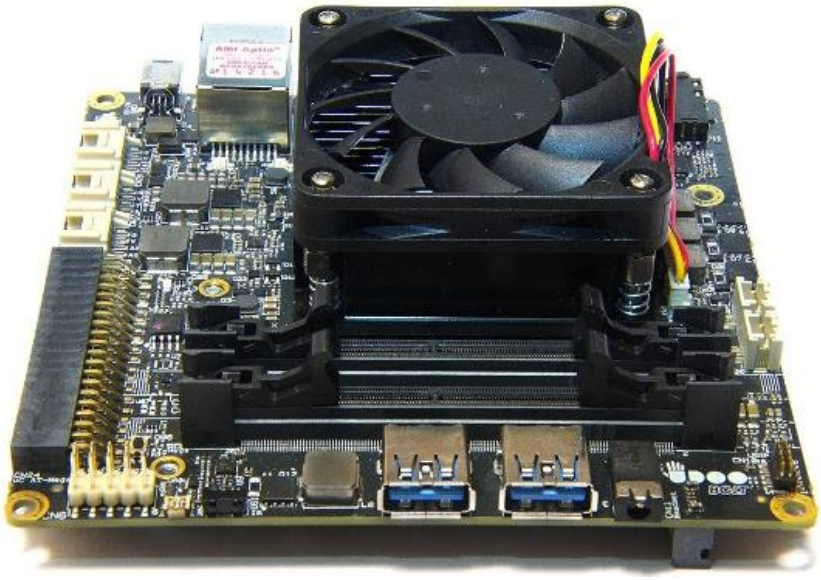

Рисунок 3.4 – UDOO BOLT V8

Існує два різних варіанти Udoo Bolt: V3 і V8. Завдяки вбудованій технології AMD Ryzen, UDOO Bolt V3 оснащений двоядерним чотирьохпотоковим процесором Ryzen V1202B з тактовою частотою 2,3 ГГц з можливістю підвищення до 3,2 ГГц та графікою AMD Radeon Vega 3. UDOO Bolt V8 має вбудований чотирьохядерний восьмиядерний процесор Ryzen V1605B з тактовою частотою 2 ГГц з можливістю підвищення до 3,6 ГГц і графікою Radeon Vega 8. Форм-фактор більший, ніж у Pi 4. Такі зручності, як SO-DIMM 64-бітної оперативної пам'яті DDR4 до 32 ГБ, SATA і 32 ГБ пам'яті eMMC, роблять Bolt дуже потужним. Це справжня заміна настільному комп'ютеру, здатна відтворювати сучасні AAA ігри та емулювати такі консолі,

як PlayStation 3 і Wii U. UDOO Bolt V3 коштує понад 300 доларів, а V8 – понад 400 доларів [27]. Це недешево, але з продуктивністю майже вдвічі більшою, ніж у Apple MacBook Pro 13", це дуже потужний маленький комп'ютер. Для надійного одноплатного комп'ютера з архітектурою x86, який може працювати під управлінням Windows, Linux та високопродуктивних додатків, цей SBC з SATA є фактичною заміною настільного комп'ютера.

3.2.3 Міні-ПК Nvidia Jetson Xavier NX

Одноплатний комп’ютер Nvidia Jetson Xavier NX може впоратися з багатьма завданнями. Завдяки потужному графічному процесору Nvidia Volta, Nvidia Jetson Xavier NX – це плата для розробки зі штучним інтелектом. Працюючи на 384-ядерному графічному процесорі Nvidia Volta, Jetson Xavier NX має шестиядерний процесор ARMv8.2. Комп’ютер має 8 ГБ 128-бітної оперативної пам'яті LPDDR4, модуль eMMC на 16 ГБ і два двигуни Nvidia Deep Learning Accelerator. Він здатний відтворювати відео у форматі 4K за допомогою двох виходів DisplayPort/HDMI і має чудові входи/виходи (I/O) від USB 3.1 до GPIO (Рис 3.5).

Рисунок 3.5 – Nvidia Jetson Xavier NX

Завдяки 48 тензорним ядрам і високим показникам комп'ютерної потужності, Nvidia Jetson Nano є найкращим ARM SBC для різноманітних

цілей. На ньому можна запускати програми штучного інтелекту, такі як машинне навчання, з такою ж витонченістю, як і робототехніка, настільні програми та навіть ретро-ігри. Jetson Xavier NX можна вважати кращим варіантом у питані потужності і гнучкості [28]. В якості альтернативи, трохи менш потужний, але все ще продуктивний Nvidia Jetson Nano оснащений чотирьохядерним процесором ARM A57, 128-ядерним графічним процесором Nvidia Maxwell і 4 ГБ оперативної пам'яті DDR4.

Плюси:

– підтримує штучний інтелект – обробляє машинне навчання, обробку природної мови тощо;
– підходить для робототехніки;
– підходить для настільних комп'ютерів;
– можна використовувати для емуляції ретро-ігор високого класу;
– ввід/вивід – Wi-Fi, Bluetooth, USB 3.1, GPIO, HDMI/DisplayPort;
– до 21 ТВт комп'ютерної потужності;
– відеовихід 4K.

3.2.4 Огляд датчиків камери для багатоканальної системи

Для моделювання багатоканальної системи з функцією розпізнаванні об’єктів у відеопотоці необхідно висока якість відео та можливість додаткових функцій, наприклад, функція нічного бачення. Це основні принципи, які повинна забезпечувати система, задля уникнення зломів і коректної ідентифікації.

Для цього було розглянуто такі камери:

1. Модуль камери Raspberry Pi 2 – 8-мегапіксельний сенсор Sony IMX219 (порівняно з 5-мегапіксельним сенсором OmniVision OV5647 оригінальної камери). Модуль камери 2 можна використовувати для зйомки відео високої чіткості, а також фотографій. Він простий у використанні для початківців, але має багато чого запропонувати досвідченим користувачам. Модуль зробив великий крок вперед у якості зображення, точності передачі

кольору та продуктивності в умовах низької освітленості. Камера підтримує режими відео 1080p30, 720p60 і VGA90, а також фотозйомку. Камера підключається за допомогою 15-сантиметрового стрічкового кабелю до порту CSI на Raspberry Pi.

Камера працює з усіма моделями Raspberry Pi 1, 2, 3 і 4. Доступ до неї можна отримати через інтерфейси MMAL і V4L API, і для неї створено безліч сторонніх бібліотек, зокрема бібліотеку Picamera Python. Модуль камери дуже популярний у системах домашньої безпеки та в пастках для спостереження за дикими тваринами [29].

2. ZeroCam Noir – це модуль камери для Raspberry Pi Zero або Raspberry Pi Zero W. Ця камера не має інфрачервоного фільтра на об'єктиві, тому вона ідеально підходить для зйомки в умовах низької освітленості. Особливостями камери є 5 Мп сенсор, 2592×1944 пікселів, відео 1080p зі швидкістю 30 кадрів в секунду (або 60 кадрів в секунду при 720p, 90 кадрів в секунду при 480p), об'єктив 2.9, фокусна відстань 3.60 мм, 53,50 градусів по горизонталі, 41,41 градусів по вертикалі [35].

3. Fisheye Camera – це камера, сумісна з Raspberry Pi. Вона має ширококутний об'єктив "риб'яче око" з кутом огляду 175º. Ось деякі з функцій камери: 5-Мп модуль камери Omnivision 5647 Роздільна здатність фотографій: 2592 x 1944 Кут огляду: 175 градусів.

4. OpenMV Cam H7 R2 - це невелика, малопотужна мікроконтролерна плата, яка дозволяє легко впроваджувати машинний зір у реальному світі. Найкраще в OpenMV те, що вбудована камера призначена не тільки для захоплення зображень, але також може бути використана для розпізнавання облич, відстеження кольорів, зчитування QR-кодів та багато іншого. Камера OpenMV H7 R2 має процесор STM32H743VI ARM Cortex M7, який працює на частоті 480 МГц, 1 МБ оперативної пам'яті та 2 МБ флеш-пам'яті. Камера оснащена сенсором зображення MT9M114, який може знімати зображення з роздільною здатністю 640x480. Більшість простих алгоритмів можуть

працювати в діапазоні 40-80 кадрів в секунду при роздільній здатності QVGA (320 x 240) і нижче (Рис 3.6).

Рисунок 3.6 – Камера OpenMV Cam H7

Модуль OpenMV Cam можна программувати за допомогою високорівневих скриптів Python (завдяки операційній системі MicroPython). Він дає можливість легко запускати зйомку фотографій і відео за зовнішніми подіями або виконувати алгоритми машинного зору [30].

Висновки до розділу 3

В рамках третього розділу було змодельовано структурну схему багатоканальної системи контролю доступу на якій продемонстровано підключення компонентів для коректного функціонування багатоканальної системи доступу.

Також, було спроектовано блок-схему алгоритму роботи системи на якій продемонстровано основний принцип роботи системи з трьома рівнями доступу. Перший рівень являє собою отримання сигналу відбитку пальця, для другого рівня безпеки необхідна буде перевірка обличчя за допомогою OpenCV та на третьому рівні, потрібен скан райдужки ока, при позитивному результаті, буде отримано доступ. При негативному – повторна спроба.

Система є гнучкою та дозволяє змінювати компоненти згідно з потреб, наприклад, відбиток пальця на чіп-карту, або ключ-код. Також може бути замінено скан райдужки ока на сканування сітківки, що значно покращить систему, але це збільшить витрати.

Також було розглянуто альтернативні апаратні рішення, їх функціонал, швидкість відтворення, підтримку компонентів та ціни. Було оглянуто три варіанта одноплатних комп’ютерів та 4 датчика камери. Підраховуючи усі переваги, для створення багатоканальної системи було обрано плату Raspberry Pi 4 та датчик камери Raspberry Pi 2.

РОЗДІЛ 4 РОРОБКА ПРОГРАМНОЇ ЧАСТИНИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ БАГАТОКАНАЛЬНОЇ СИСТЕМИ

4.1 Вибір компонентів та мови програмування

Для моделювання системи було підібрано компоненти, які надають повний спектр функціональних якостей для чіткої роботи. За основу було обрано системну плату Raspberry Pi 4, сканер відбитків пальця Basic Fingerprint Sensor With Socket Header Cable та камеру Raspberry Pi Camera Module v2, яка має чітку якість зображення та сумісна з технологією OpenCV. Мовою програмування було обрано Python.

4.1.1 Raspberry Pi 4

Raspberry Pi 4 Model B — останній продукт у популярній лінійці комп’ютерів Raspberry Pi. Він забезпечує революційне збільшення швидкості процесора, мультимедійної продуктивності, пам’яті та можливостей підключення порівняно з попереднім поколінням Raspberry Pi 3 Model B+, зберігаючи при цьому зворотну сумісність і аналогічне енергоспоживання. Для кінцевого користувача Raspberry Pi 4 Model B забезпечує продуктивність настільного комп’ютера, порівнянну з ПК x86 початкового рівня [25].

Основні характеристики цього продукту включають високопродуктивний 64-розрядний чотирьохядерний процесор, підтримку подвійного дисплея з роздільністю до 4K через пару портів micro-HDMI, апаратне декодування відео з роздільністю до 4Kp60, до 4 ГБ оперативної пам’яті, подвійний -діапазон бездротової локальної мережі 2,4/5,0 ГГц, Bluetooth 5.0, Gigabit Ethernet, USB 3.0 і можливість PoE (через окрему надбудову PoE HAT).

Дводіапазонна бездротова локальна мережа та Bluetooth мають модульну сертифікацію на відповідність, що дозволяє проектувати плату в кінцеві продукти зі значно скороченим тестуванням на відповідність, покращуючи як вартість, так і час виходу на ринок.

Таблиця 4.1 – Технічна інформація і відмінності Raspberry Pi 3 Model B+ та Raspberry Pi 4 Modell B

	Raspberry Pi 3 Modell B+	**Raspberry Pi 4 Modell B**
CPU	Broadcom BCM2837B0, Quad-core Cortex-A53 64-bit SoC @ 1,4 GHz	Broadcom 2711, Quad-core Cortex- A72 64-bit SoC @ 1,5 GHz
RAM	1 GB LPDDR2 SDRAM	2 GB, 4GB oder 8 GB LPDDR4 SDRAM
Connectivity	2,4GHz / 5.0 GHz IEEE 802.11.b/g/n/ac Wireless LAN Bluetooth 4.2, BLE 4x USB 2.0 Ports Gigabit Ethernet über USB2.0 (max. 300MPS)	2,4GHz / 5,0 GHz IEEE 802.11.b/g/n/ac wireless LAN Bluetooth 5.0, BLE 2x USB 3.0 / 2x USB 2.0 Ports Echtes Gigabit Ethernet
Accessibility	40-Pin GPIO Header	40-Pin GPIO Header
Video & Sound	1x Standard HDMI Typ A Buchse 1x 15 Pin MIPI DSI Display Port 1x 15 Pin MIPI CSI Kamera Port 1x 4 Pin Stereo Audio und Composite Video Jack	2x Micro HDMI Typ D Buchse 1x 15 Pin MIPI DSI Display Port 1x 15 Pin MIPI CSI Kamera Port 1x 4 Pin Stereo Audio und Composite Video Jack

Таблиця 4.1 – Технічна інформація і відмінності Raspberry Pi 3 Modell B+ та Raspberry Pi 4 Modell B

Max. Resolution Image output	HDMI: 1080p / 1920 x 1080p @ 60Hz Composite: 720x480@60i NTSC, 720x576@50i PAL	1x HDMI 4K / 3840 x 2160p @ 60Hz 2x HDMI 4K / 3840 x 2160p @ 30Hz Composite: 720x480@60i NTSC, 720x576@50i PAL
Multimedia	H.264, MPEG-4 decode (1080p30) H.264 encode (1080p30) OpenGL ES 1.1, 2.0 graphics	H.265 decode (4kp60) H.264 decode (1080p60) H.264 encode (1080p30) OpenGL ES 1.1, 2.0, 3.0 graphics
SD Card Support	Micro SD-Format	Micro SD-Format
Power Supply	5V/2.5A DC über micro Typ B USB 5V DC GPIO PoE via PoE HAT	5V/3A DC über USB Type C 5V DC GPIO PoE via PoE HAT

Плата має два порти micro HDMI, кожен з яких може виводити відео 4K зі швидкістю 30 кадрів в секунду. При використанні одного екрану, можна отримати 60 кадрів в секунду. Ця функція, а також версія з 4 ГБ оперативної пам’яті, забезпечують досить хороший комп’ютер для програмування, який може відповідати продуктивності старих ноутбуків.

4.1.2 Сканер відбитків пальців

Для першого рівня безпеки використано сканер відбитків пальця Basic Fingerprint Sensor With Socket Header Cable. Цей модуль оснащений FLASH-

пам'яттю для зберігання відбитків пальців і працює з будь-яким мікроконтролером або системою з TTL-послідовним інтерфейсом[26]. Ці модулі можуть бути додані до систем безпеки, дверних замків, систем обліку робочого часу та багато іншого (Рис. 4.1).

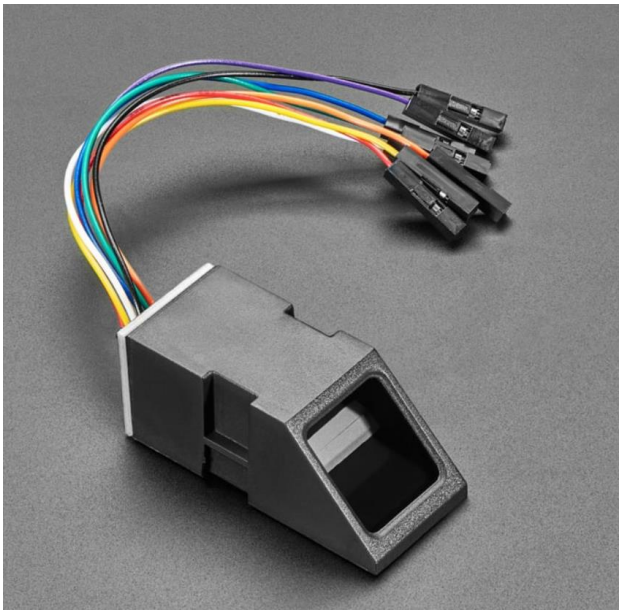

Рисунок 4.1 – Сканер відбитків пальців Basic Fingerprint Sensor

Технічні характеристики датчика відбитків пальців, який буде використано для моделювання багатоканальної системи:

- напруга живлення: Постійний струм від 3.6 до 6.0 В;
- струм живлення: <120 мА;
- колір підсвічування: зелений;
- інтерфейс: UART;
- погана швидкість: 9600;
- рівень безпеки: п'ять (від низького до високого: 1,2,3,4,5);
- частота помилкових прийомів (FAR): <0.001% (рівень безпеки 3);
- частота помилкових відмов (FRR): <1.0% (рівень безпеки 3);
- здатність зберігати 127 різних відбитків пальців.

Тож, цей датчик є ідеальним варіантом для проєкту, виходячи з вище перелічених функцій та є повністю сумісний з платою Raspberry Pi 4.

4.1.3 Модуль камери Raspberry Pi 2

Оновлений Raspberry Pi Camera Module v2 - це нова офіційна плата камери, випущена Raspberry Pi Foundation, яка підключається до будь-якого Raspberry Pi або обчислювального модуля. Raspberry Pi Camera Module v2 - це високоякісна 8-мегапіксельна камера на основі датчика зображення Sony IMX219, що дозволяє створювати відео у форматі HD та фотознімки. Це спеціально розроблена додаткова плата для Raspberry Pi з об'єктивом з фіксованим фокусом (Рис. 4.2).

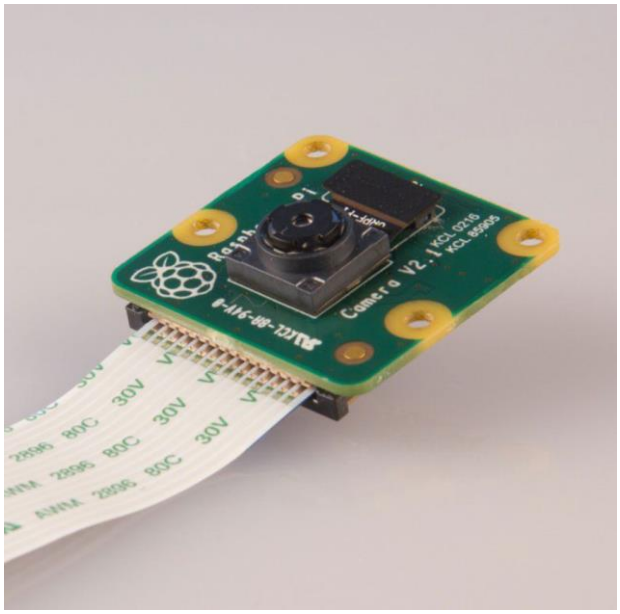

Рисунок 4.2 – Модуль камери Raspberry Pi Camera Module v2

Вона здатна створювати статичні зображення з роздільною здатністю 3280 x 2464 пікселів, а також підтримує відео 1080p30, 720p60 і 640x480p90. Вона приєднується до Pi за допомогою одного з невеликих гнізд на верхній поверхні плати і використовує спеціальний інтерфейс CSi, розроблений спеціально для взаємодії з камерами.

Основні переваги модуля:

- вбудований об'єктив з фіксованим фокусом;

- покращена роздільна здатність - 8-мегапіксельний датчик з власною роздільною здатністю, здатний створювати статичні зображення з роздільною здатністю 3280 x 2464 пікселів;
- підтримує відео 1080p30, 720p60 і 640x480p90;
- розмір 25мм x 23мм x 9мм;
- вага трохи більше 3 г;
- підключається до плати Raspberry Pi за допомогою короткого стрічкового кабелю (в комплекті);
- Camera v2 підтримується останньою версією Raspbian, улюбленої операційної системи Raspberry Pi;
- датчик використовує зображення Sony IMX219PQ - високошвидкісна відеозйомка та висока чутливість;
- 1,4 м X 1,4 м пікселів з технологією OmniBSI для високої продуктивності (висока чутливість, низький рівень перехресних перешкод, низький рівень шуму);
- оптичний розмір 1/4";
- працює з будь-якою версією Raspberry Pi або обчислювального модуля.

Завдяки мініатюрним розмірам (25 мм x 23 мм x 9 мм) і вазі трохи більше 3 г, він є ідеальним аксесуаром для мобільних або інших додатків, де розмір і вага є важливими. Модуль камери Raspberry Pi забезпечує зменшення забруднення зображення, наприклад, шуму та розмиття фіксованого зображення. Він також має функції автоматичного керування, такі як контроль експозиції, баланс білого та визначення яскравості.

4.1.4 Мова програмування

Мовою програмування для багатоканальної системи на основі Raspberry Pi 4 та OpenCV було обрано Python. Ця мова ідеально підходить для програмування на Raspberry Pi та крім того, за допомогою Python можна програмувати камеру Raspberry Pi Camera Module v2.

Python – це інтерпретована, об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою, розроблена Гвідо ван Россумом. Вона була вперше випущена в 1991 році (Рис. 4.3). Назва "Python", створена для того, щоб бути легкою та веселою, є відсиланням до британської комедійної групи Monty Python. Python має репутацію мови, зручної для початківців, замінивши Java як найпоширеніша вступна мова, оскільки вона вирішує більшу частину складних завдань для користувача, дозволяючи новачкам зосередитися на повному розумінні концепцій програмування, а не на дрібних деталях [31].

Рисунок 4.3 – Мова програмування Python

Python використовується для серверної веб-розробки, розробки програмного забезпечення, математики та системних сценаріїв, а також популярна для швидкої розробки додатків і як мова сценаріїв або клейка мова для зв'язування існуючих компонентів завдяки своїм високорівневим вбудованим структурам даних, динамічній типізації та динамічному зв'язуванню. Завдяки синтаксису, що легко засвоюється, та акценту на читабельності, витрати на підтримку програм зменшуються завдяки Python. Крім того, підтримка модулів і пакетів у Python полегшує створення модульних програм і повторне використання коду. Python – це мова з відкритим вихідним кодом, тому численні незалежні програмісти постійно створюють для неї бібліотеки та функціональні можливості.

Приклади використання Python:

- створення веб-додатків на сервері;
- побудова робочих процесів, які можна використовувати в поєднанні з програмним забезпеченням;
- підключення до систем баз даних;
- читання та модифікація файлів;
- виконання складних математичних обчислень;
- обробка великих даних;
- швидке створення прототипів;
- розробка готового до виробництва програмного забезпечення.

З професійної точки зору, Python чудово підходить для внутрішньої веб-розробки, аналізу даних, штучного інтелекту та наукових обчислень. Розробники також використовують Python для створення інструментів для підвищення продуктивності, ігор та десктопних додатків.

Можливості та переваги Python:

– сумісність з різними платформами, включаючи Windows, Mac, Linux, Raspberry Pi та інші;

– використовує простий синтаксис, порівнянний з англійською мовою, що дозволяє розробникам використовувати менше рядків, ніж інші мови програмування;

– працює на системі інтерпретаторів, що дозволяє негайно виконувати код, прискорюючи створення прототипів;

– може бути процедурною, об'єктно-орієнтованою або функціональною.

Python, мова з динамічною типізацією, є особливо гнучкою, усуваючи жорсткі правила побудови функцій і пропонуючи більшу гнучкість у вирішенні проблем за допомогою різноманітних методів. Вона також дозволяє компілювати та запускати програми аж до проблемної області, оскільки використовує перевірку типів під час виконання, а не під час компіляції.

З іншого боку, Python нелегко підтримувати. Одна команда може мати декілька значень в залежності від контексту, оскільки Python є динамічно-

типізованою мовою. Крім того, зі збільшенням розміру та складності додатку на Python стає дедалі важче підтримувати його, особливо знаходити та виправляти помилки. Користувачам знадобиться досвід для проектування коду або написання модульних тестів, які полегшують супровід.

Швидкість – ще одне слабке місце Python. Її гнучкість, оскільки вона динамічно типізується, вимагає значної кількості посилань, щоб приземлитися на правильне визначення, що уповільнює продуктивність. Це можна зменшити, використовуючи альтернативну реалізацію Python (наприклад, PyPy).

Дослідники штучного інтелекту – фанати Python. Google TensorFlow, а також інші бібліотеки (scikit-learn, Keras) створюють основу для розробки ШІ завдяки зручності та гнучкості, які вони пропонують користувачам Python. Ці бібліотеки та їхня доступність є критично важливими, оскільки вони дозволяють розробникам зосередитися на розвитку та створенні.

4.2 Встановлення операційної системи Raspberry Pi

При наявності всіх необхідних компонентів, було виконано наступні кроки, щоб створити завантажувальний диск, який знадобиться для налаштування Raspberry Pi.

Спершу було вставлено карту microSD комп'ютер. Далі завантажено та встановлено офіційну програму Raspberry Pi Imager. Доступний для Windows, macOS або Linux, цей додаток завантажить і встановить останню версію Raspberry Pi OS.

Наступним кроком було обрано операційну систему (Рис. 4.4) із представленого списку. Для дипломної роботи було обрано Raspberry Pi OS (32-bit).

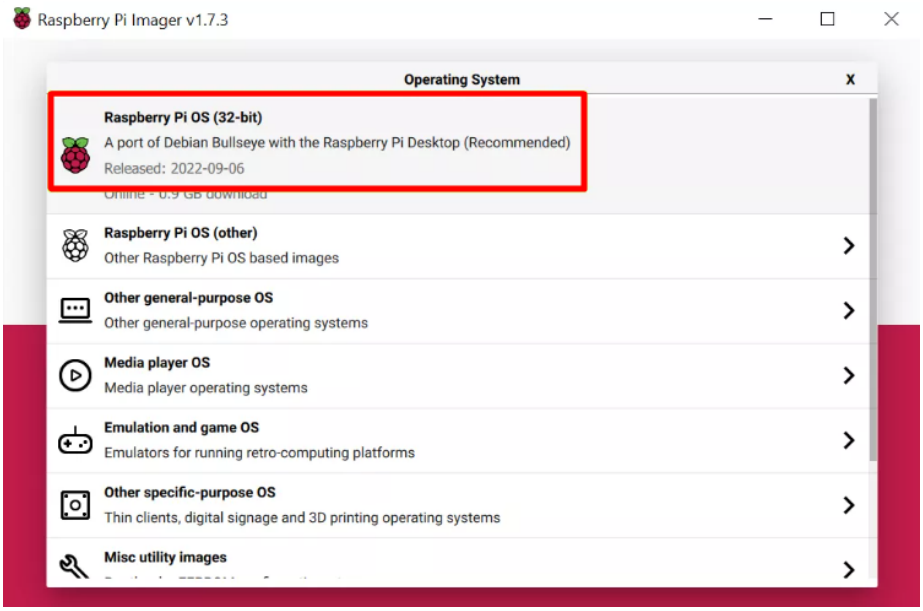

Рисунок 4.4 – Вибір ОС через Raspberry Pi Imager

Далі було вибрано вкладку сховище і обрано SD-карту. Наступним кроком треба було перейти до налаштувань натиснувши CTRL + SHIFT + X.

Наступник кроком було заповнено поля налаштувань, як показано на Рисунку 4.5, а потім збережено. Ці поля рекомендуємо заповнювати, щоб Raspberry Pi був налаштований і підключений до мережі, як тільки буде завантажений. Але ім'я користувача можна вибрати пізніше через майстер налаштування, який попросить створити їх при першому завантаженні.

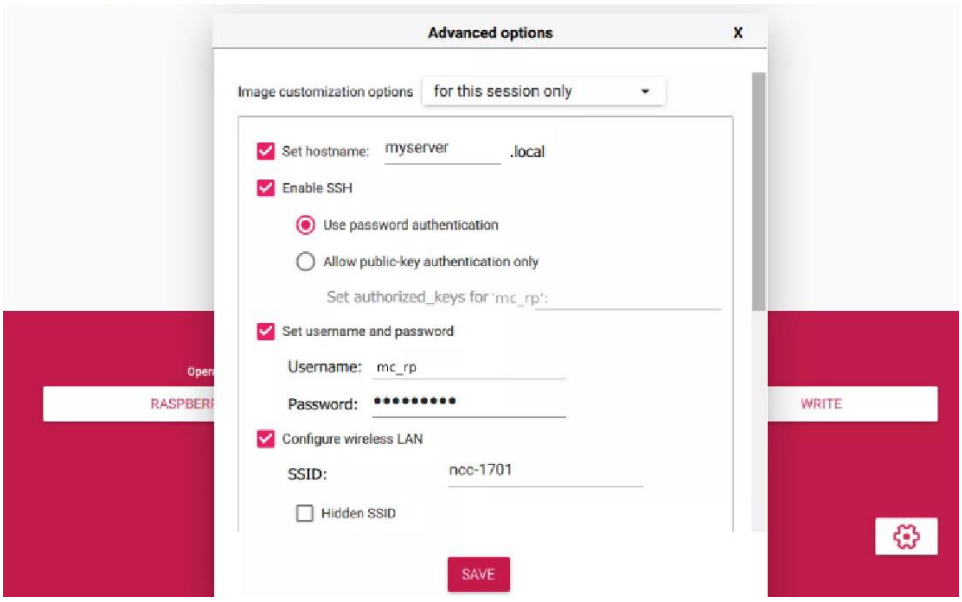

Рисунок 4.5 – Заповнення даних у вкладці налаштувань

Далі була натиснута кнопка "Записати". Тепер додатку знадобиться кілька хвилин, щоб завантажити операційну систему і записати дані на CD картку.

І було вставлено карту microSD в Raspberry Pi, підключено плату до монітора, клавіатури і миші. Наступним кроком підключено кабель Ethernet, для використання провідного інтернету, щоб підключити Pi до мережі для увімкнення.

4.3 Налаштування SSH та FTP серверів на Raspberry Pi

З’єднання через SSH потрібні, якщо пристрій не має графічного інтерфейсу користувача для конфігурації або коли потрібне точне налаштування. Щоб віддалено керувати Raspberry Pi без використання графічної оболонки, було вирішено підключитися до пристрою через SSH.

Щоб активувати сервер SSH у конфігурації операційної системи, потрібно виконати наступні дії:

1. Ввести команду sudo raspi-config у терміналі
2. Далі перейти на сторінку параметрів інтерфейсу
3. Знайти пункт SSH і активувати сервер SSH.
4. Натиснути Enter і перейти до Finish, щоб вийти з інструменту.

Для коректного передавання відео-даними між Raspberry Pi та комп’ютерами систем безпеки. Для цього було вибрано FTP протокол, який дозволить передавати, завантажувати і зберігати файли[32].

Для встановлення FTP серверу необхідно виконати ряд наступних дій:

1. Спочатку треба переконатись що ОС Raspberry Pi працює з останніми доступними пакетами за допомогою команд:

```
sudo apt update
sudo apt full-upgrade
```

2. Необхідно встановити програму vsftpd допомогою команди:

```
sudo apt-get install vsftpd
```

3. Перш ніж підключитися до нового FTP-сервера Raspberry Pi, потрібно змінити деякі налаштування за допомогою команди:

```
sudo nano /etc/vsftpd.conf
```

Та зробити такі налаштування:

```
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
chroot_local_user=YES
user_sub_token=$USER
local_root=/home/$USER/FTP
```

4. Далі потрібно зберегти та вийти за допомогою команд CTRL + X, а потім Y.

5. Наступним кроком було створення директорії FTP для передачі файлів за допомогою команди:

```
mkdir -p /home/<user>/FTP/files
```

6. Після додавання каталогу було змінено його дозволи, щоб видалити дозвіл на запис із каталогу FTP:

```
chmod a-w /home/<user>/FTP
```

7. Далі було перезавантажено для того щоб працювали нові налаштування

```
sudo service vsftpd restart
```

8. Далі було використано FTP-клієнт FileZilla, оскільки він підтримує як FTP, так і SFTP, а також працює з багатьма операційними системами. Перелік дій для налаштування Raspberry Pi для входу на FTP сервер через FileZilla:

- ввести IP-адресу вашого Raspberry Pi
- ввести ім'я користувача
- ввести пароль для цього користувача
- заповнити параметр порту. При використані FTP на Raspberry Pi було обрано порт 21 (Рис 4.6).

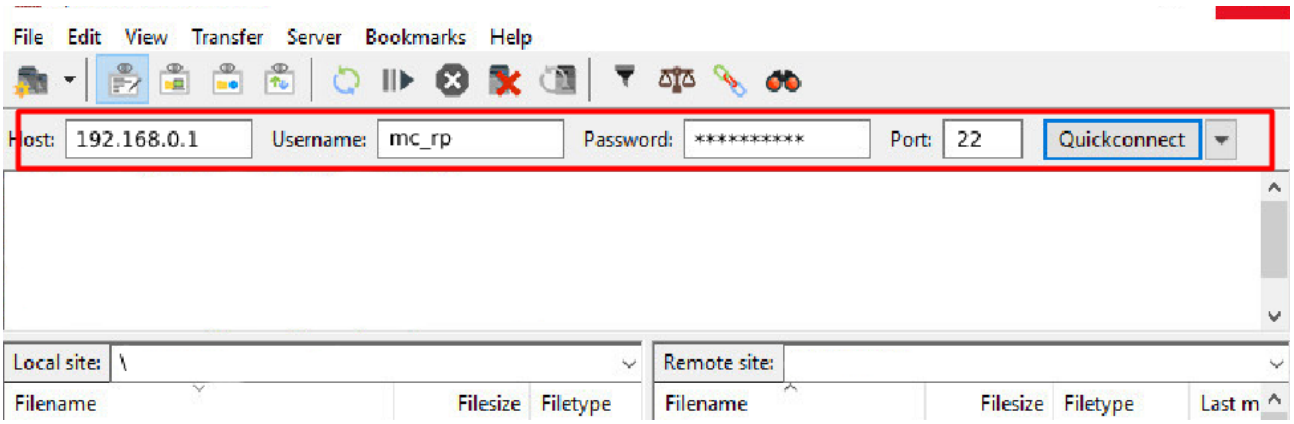

Рисунок 4.6 – Налаштування Raspberry Pi для входу на FTP сервер через FileZilla

Після чого було виконано команду Quickconnect та успішно підключено до FTP серверу.

4.4 Підключення камери та налаштування OpenCV

Основою багатоканальної системи керування доступом є розпізнавання об'єктів у відеопотоці. Для цього було використано модуль камери Raspberry Pi 2. Для підключення і коректного налаштування було виконано наступні кроки.

1. Підключення модулю відеокамери

Спочатку потрібно вимкнути живлення Raspberry Pi, знайти на платі порт, підключений до камери, і обережно витягнути його виступом збоку. Верхня частина роз'єму підніметься приблизно на 2 мм і злегка переміститься вбік.

Далі потрібно вставити один кінець кабелю камери у відкритий роз'єм так, щоб синій виступ був збоку від аудіороз'єму, а контакти були звернені до роз'єму HDMI. Після підключення все має виглядати, як показано на рисунку 4.7:

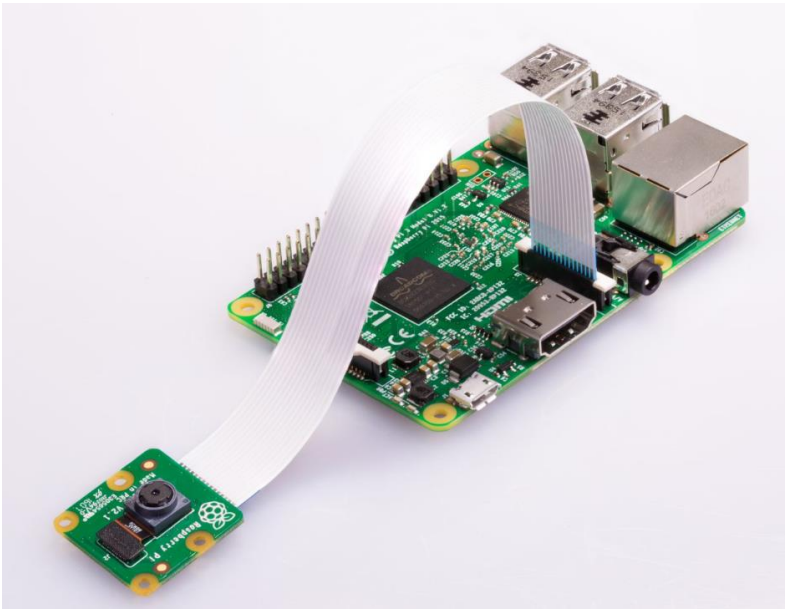

Рисунок 4.7 – Підключення модулю камери до Raspberry Pi

2. Далі було відкрите меню конфігурації Raspberry Pi для увімкнення опції "Камера" у вкладці "Інтерфейси". Після увімкнення потрібно перезавантажити Raspberry Pi.

3. Наступним кроком, необхідно було навчити систему розпізнавати обличчя. Для цього буде використовуватись технологія OpenCV. Це бібліотека комп’ютерного зору з відкритим вихідним кодом, яка включає функції та алгоритми комп’ютерного зору, а також алгоритми обробки зображень. Бібліотека має вбудовані засоби обробки та аналізу зображень, такі як розпізнавання об’єктів на фотографіях, людей, тварин, номерних знаків автомобілів, тексту, відстеження руху об’єктів, обробка та трансформація зображень, застосування методів машинного навчання [33].

У каталозі `home/pi directory/facial_recognition` було створено папку з назвою `headshots_picam.py`. Цей код на python дозволяє зробити кілька фотографій облич за допомогою офіційної камери Raspberry Pi.

Далі було відкрито цей скрипт Python і замінено виділений рядок коду власною назвою (Рис. 4.8). Потім потрібно зберегти скрипт.

```
import cv2
from picamera import PiCamera
from picamera.array import PiRGBArray

name = 'Volodymyr'

cam = PiCamera()
cam.resolution = (512, 304)
cam.framerate = 10
rawCapture = PiRGBArray(cam, size=(512, 304))
```

Рисунок 4.8 – Налаштування скрипту

Далі потрібно відкрити папку Photo і додати ще одну папку з власним ім'ям. Ця тека буде місцем, куди будуть завантажуватися фотофайли.

Потім, у редакторі Python, було відкрито файл `headshots_picam.py`, і запущено код. Після цього відкриється маленьке вікно і вікно терміналу, які було використано для збереження зображень обличчя. Щоб зробити знімок потрібно натиснути клавішу Space. Було зроблено близько 10 фото, для того щоб камера могла чітко визначати особу. Необхідно надати програмі різні ракурси обличчя, щоб вона могла краще визначити розмір особи.

Після закриття програми зображення обличчя зберігаються в папці власного імені (Рис 4.9). У такий самий спосіб до багатоканальної систему будуть додаватися наступні обличчя.

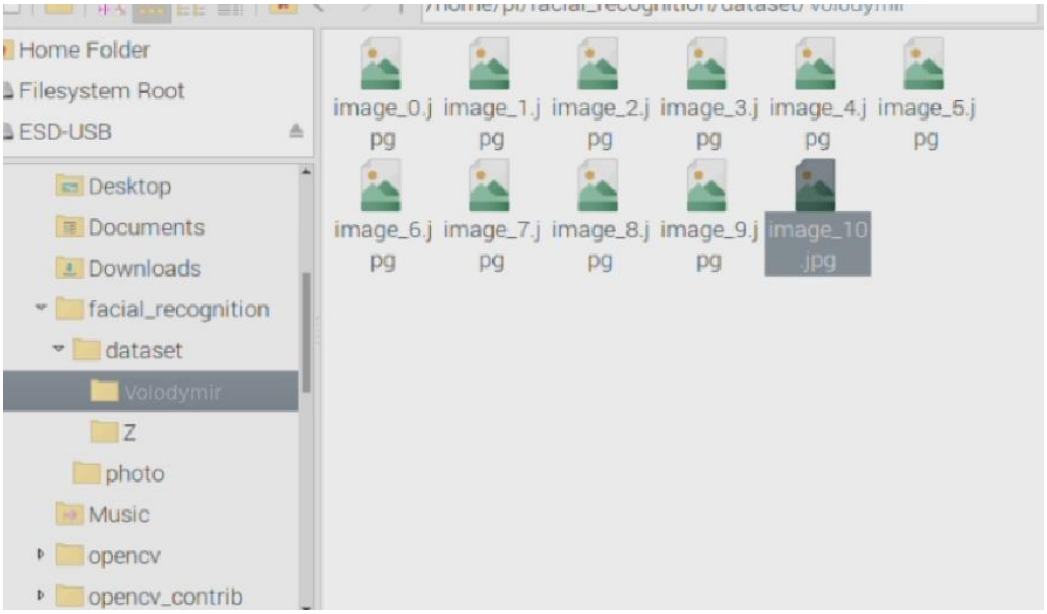

Рисунок 4.9 – Збережені фото для розпізнавання обличчя програмою

Далі зроблені фотографії будуть використовуватися кодом на Python у файлі `train_model.py`. Будь-які зображення в папці з набором даних будуть проаналізовані цим кодом, після запуску. У відкритому терміналі було введено наступний код, натискаючи enter після кожного рядка:

```
cd facial_recognition
python train_model.py
```

Це запустить процес навчання для кожного зображення зроблених у попередньому кроці. Після цього Raspberry Pi 4 Model B запам’ятає обличчя і буде його ідентифікувати (Рис. 4.9).

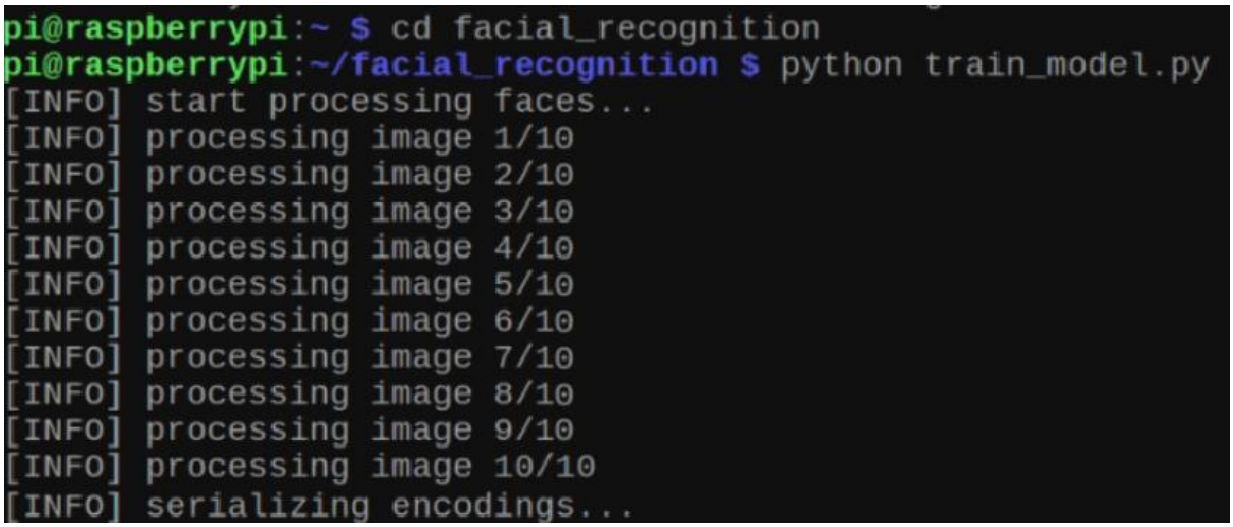

Рисунок 4.9 – Процес обробки зображень скриптом

Отже, після цього потрібно запустити код, який буде розпізнавати обличчя. При знаходженні доданого обличчя буде з’являтись надане ім'я поруч з ним. Для цього потрібно відкрити термінал та ввести наступні команди:

```
cd facial_recognition
python facial_req.py
```

Після цього з’явиться невелике вікно з прямою трансляцією з камери Raspberry Pi. При наведені камери на обличчя, з’явиться жовта рамка навколо голови і з назначеним ім'я (Рис. 4.10). Система також визначає, чи відоме обличчя або ні, то ж при розпізнавані невідомого обличчя, біля рамочки буде напис "Unknown".

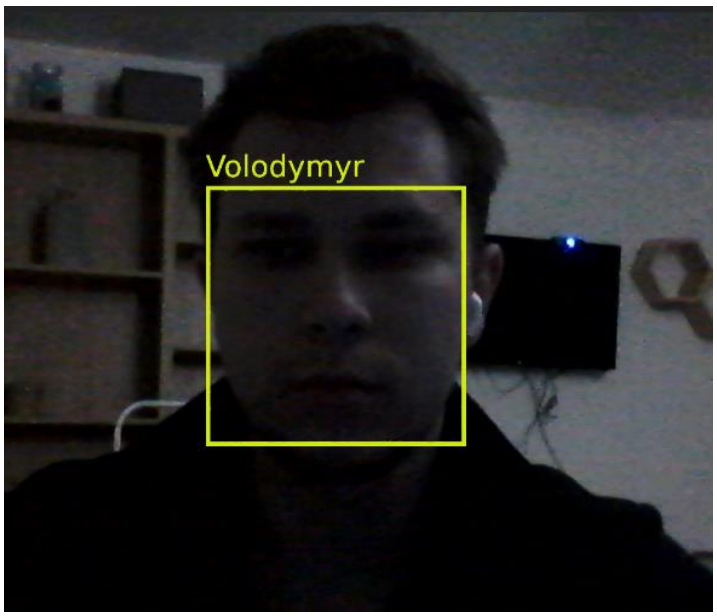

Рисунок 4.10 – Розпізнавання обличчя камерою

Також, було виявлено, що при нахилені голови на пару градусів, то розпізнавання обличчя повністю вимкнеться, закривання носу рукою, програма також не зможе його розпізнати.

4.5 Підключення та налаштування модулю відбитків пальців

Спершу, було підключено дроти з комплекту до сканера, ретельно. Потім під'єднати чотири дроти, що виходять зі сканера, до одноплатного комп'ютера Raspberry Pi (Рис. 4.11). Червоний дріт призначений для живлення 3,3 В і повинен бути підключений до контакту живлення 3,3 В на Raspberry Pi. Білий дріт підключається до TX GPIO 14 Pin на Raspberry Pi. Жовтий дріт підключається до 15-го контакту RX GPIO на Raspberry Pi. Чорний провід призначений для заземлення і повинен підключатися до контакту заземлення на Raspberry Pi. Цей сканер відбитків пальців зламається, якщо ви підключите його до 5 В, тому обов'язково перевірте, що ви підключили живлення до контакту 3,3 В перед тим, як увімкнути систему.

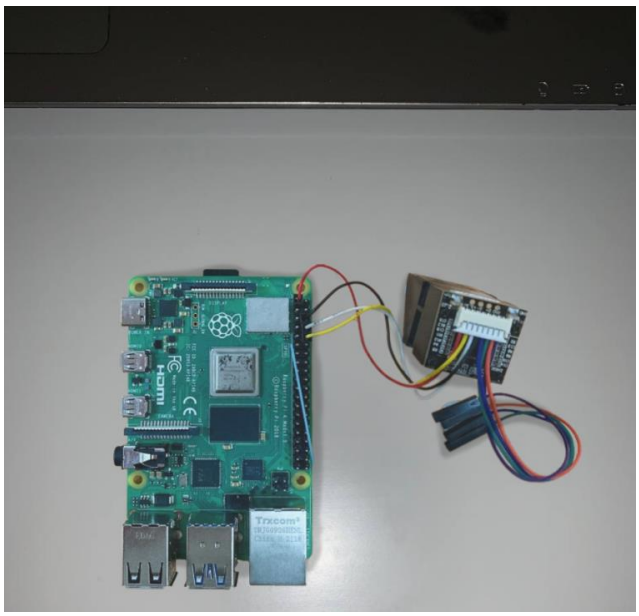

Рисунок 4.11 – Підключення модуля до Raspberry Pi

Потім потрібно ввімкнути систему Raspberry Pi, підключивши блок живлення USB-C. Для коректної роботи з модулем треба встановити необхідні бібліотеки. Це забезпечить правильний зв'язок між сканером відбитків пальців і Raspberry Pi.

Першим кроком буде увімкнення послідовного порту, оскільки це дозволить одноплатному комп'ютеру Raspberry Pi обмінюватися даними зі сканером відбитків пальців через послідовний зв'язок TTL UART у меню конфігурацій.

Потім було відкрито вікно терміналу для встановлення потрібних бібліотек. Їх було додано за допомогою наступних команд [34]:

```
sudo pip3 install adafruit-circuitpython-fingerprint pyfingerprint
git clone https://github.com/adafruit/Adafruit_CircuitPython_Fingerprint.git
git clone https://github.com/bastianraschke/pyfingerprint.git
```

Після цього, усі необхідні налаштування встановлено на одноплатний комп'ютер Raspberry Pi для роботи зі сканером відбитків пальців.

Наступним кроком потрібно відкрити скрипт `fingerprint_simpletest_rpi.py`. з каталогу скриптів

`/home/pi/Adafruit_CircuitPython_Fingerprint/examples`, за допомогою Thonny IDE.

Було закоментовано рядок 14, та розкоментовано рядок 17 та збережено скрипт після цього (Рис 4.12).

```
 9
10 # import board
11 # uart = busio.UART(board.TX, board.RX, baudrate=57600)
12
13 # If using with a computer such as Linux/RaspberryPi, Mac, Windows with USB/serial converter
14 #uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)
15
16 # If using with Linux/Raspberry Pi and hardware UART:
17 uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)
18
19 # If using with Linux/Raspberry Pi 3 with pi3-disable-bt
20 # uart = serial.Serial("/dev/ttyAMA0", baudrate=57600, timeout=1)
21
22 finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)
23
24 ##################################################
25
```

Рисунок 4.12 – Редагування скрипту для налаштування модуля

Далі запускаємо програму. На екрані з'являться кілька варіантів дій. Для зареєстрування пальця було вибрано пункт |e|, та вибрано варіант |1|, щоб ідентифікувати відбиток як перший. Кожному відбитку буде присвоєно індексний номер, і модуль може підтримувати до 300 індексованих відбитків пальців (Рис. 4.13).

Рисунок 4.13 – Реєстрація відбитку пальця

Після цього було прикладено палець до сканера, коли він засвітиться два рази. Для перевірки правильності реєстрації відбитку, було обрано пункт | f |.

Потім прикладено той самий палець до сканера. Після цього на екрані висвітився позитивний результат з показником довіри (Рис. 3.14).

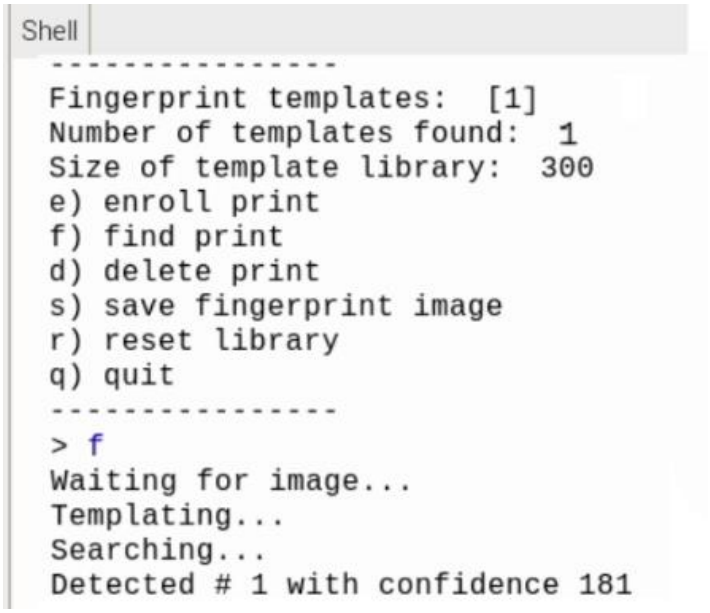

Рисунок 4.14 – Перевірка правильності реєстрації відбитка пальця

Отож, модуль було успішно підключено до одноплатного комп'ютера Raspberry Pi 4, разом з модулем камери з налаштованою бібліотекою OpenCV.

4.6 Реалізація програмної частини системи

Програмна частина системи написана на мові програмування Python.

Багатоканальну систему можна умовно на процесуальні частини, а саме рівні керування доступом:

1. Додавання критеріїв ідентифікації до бази даних.
2. Тестування модулів керування доступом.
3. Керування доступом за допомогою датчика відбитків пальця.
4. Керування доступом за допомогою камери та технології OpenCV.
5. Керування доступом використовуючи третій рівень безпеки (ключ карта, сканування сітківки/райдужки ока і тд.).
6. Модуль навчання системи згідно з отриманим датасетом.
7. Надання доступу.

4.7 Проведення експерименту і тестування

Для перевірки системи та її покращення, було проведено ряд експериментів:

1. Тестування модулю відбитків пальця.
2. Тестування модулю розпізнавання облич.
3. Швидкість процесу керування доступом.

Було проведено декілька тестувань системи, щоб виміряти приблизний час процесу керування доступом. Починаючи з першого кроку, а саме сканування відбитку пальця, був заміряний інтервал довіри та час сканування.

З п’яти спроб середній інтервал довіри був близько 283 із 500 можливих. Довірчий інтервал – це діапазон оцінок для невідомого параметра. Довірчий інтервал обчислюється на певному рівні довіри. Рівень довіри представляє довгострокову частку ДІ (при заданому рівні довіри), які теоретично містять істинне значення параметра. Фактори, що впливають на ширину ДІ, включають розмір вибірки, варіабельність вибірки та рівень довіри. За інших рівних умов, більша вибірка дає вужчий довірчий інтервал, більша варіабельність вибірки дає ширший довірчий інтервал, а вищий рівень довіри дає ширший довірчий інтервал.

Враховуючи усі фактори, можна зробити висновок що, це значення є припустимим, так як система використовує декілька рівнів безпеки, але його також можна покращити використовуючи більш точний датчик сканування відбитків пальця.

Щодо камери та зчитування обличчя, то середня швидкість реагування камери приблизно 2-3 секунди для зчитування обличчя. Також, було проведено тестування, під різним кутом нахилу обличчя. Найкращий результат по швидкості та точності системи зафіксовано при куті нахилу від 0°-45°.

У таблиці 4.2 продемонстровано результати тестування розпізнавання обличчя під різними кутами відносно камери.

Таблиця 4.2 – Порівняльна таблиця розпізнавання під різними кутами

Орієнтація обличчя	**Швидкість розпізнавання**	**Процент розпізнавання обличчя**
0° (Фронтальне зображення)	1 сек	100 %
30	2 сек	68 %
60°	3.5 сек	46 %
90°	Неможливо виявити	0 %

Тож, у середньому процес дворівневої ідентифікації займає приблизно до 7-9 секунд, якщо виконувати дії швидко і послідовно. Якщо уявляти процес середнього інтервалу часу на проходження усіх рівнів ідентифікації, то це б зайняло приблизно 20-25 секунд. У випадку використання сканеру райдужки ока, процес перевірки того, що райдужна оболонка ока жива, займає близько 3 секунд. Для порівняння, навіть системи розпізнавання райдужної оболонки ока, яким потрібен лише один знімок, зазвичай витрачають приблизно стільки ж часу на ідентифікацію. При використання датчику сканування сітківки, процес ідентифікації займе приблизно 5-10 секунд.

Тож, у середньому, 30 секунд вистачить для проходження 3-рівневої системи керування доступом.

Висновки до розділу 4

У четвертому розділі, було обрано необхідно компоненти для створення багатоканальної системи керування доступом. Ураховуючи усі переваги було обрано:

- Одноплатний комп’ютер Raspberry Pi 4;
- сканер відбитків пальця Basic Fingerprint Sensor With Socket Header Cable;
- датчик камери Raspberry Pi Camera Module v2.

Мовою програмування було обрано Python, так як ця мова ідеально підходить для програмування на Raspberry Pi та модулів для цієї плати.

Також було описано налаштування операційної системи для Raspberry Pi 4 та налаштування SSH сервера та FTP протоколу для відправлення отриманих даних на сервер.

Далі було продемонстровано процеси підключення камери і сканеру відбитків пальців до одноплатного комп’ютера. Було продемонстровано налаштування бібліотеки OpenCV та процеси навчання системи для розпізнавання обличчя та відбитку пальця.

У розділі також було описано план для реалізації програмної частини та було проведено експеримент та тестування багатоканальної системи керування доступом для розуміння коректності працездатності системи і визначення ділянки для подальших покращень.

ВИСНОВКИ

Під час виконання дипломної роботи було проаналізовано принципи роботи сучасних аналогових рішень, а саме різні системи контролю доступу, які є актуальними у теперішній час, розглянуто використання алгоритмів розпізнавання обличь для контролю доступу, а також порівняння різних систем ідентифікації за рівнем відмов.

Було розглянуто інноваційні методи та системи у сфері фізичного контролю доступу та розглянуто принцип роботи мультиспектральної біометрії обличчя у контролі доступу. Було сформовано переваги та недоліки таких систем, виходячи із розглянутої літератури, а також проаналізовано можливості створення та втручання багатоканальної системи у сфери діяльності для надання більш надійної системи безпеки, яка буде гнучкою та готовою для покриття потрібної безпеки у сучасних компаніях або банках.

Було розглянуто та проаналізовано теорію систему масового обслуговування для використання у моделюванні системи. Було приділено увагу багатоканальній системі масового обслуговування з обмеженою чергою та розглянуто технологію OpenCV, її функціональність та шляхи застосування.

Було змодельовано структурну схему багатоканальної системи контролю доступу та спроектовано блок-схему алгоритму роботи системи на якій продемонстровано основний принцип роботи системи з трьома рівнями доступу.

Система є гнучкою та дозволяє змінювати компоненти згідно з потреб, наприклад, відбиток пальця на чіп-карту, або ключ-код. Також може бути замінено скан райдужки ока на сканування сітківки, що значно покращить систему, але це збільшить витрати.

Також було розглянуто альтернативні апаратні рішення, їх функціонал, швидкість відтворення, підтримку компонентів та ціни.

Було обрано необхідно компоненти для створення багатоканальної системи керування доступом. Ураховуючи усі переваги було зроблено вибір:

- Одноплатний комп’ютер Raspberry Pi 4;
- сканер відбитків пальця Basic Fingerprint Sensor With Socket Header Cable;
- датчик камери Raspberry Pi Camera Module v2.

Мовою програмування було обрано Python, так як ця мова ідеально підходить для програмування на Raspberry Pi та модулів для цієї плати.

Далі було продемонстровано процеси підключення камери і сканеру відбитків пальців до одноплатного комп’ютера. Було продемонстровано налаштування бібліотеки OpenCV та процеси навчання системи для розпізнавання обличчя та відбитку пальця.

У розділі також було описано план для реалізації програмної частини та було проведено експеримент та тестування багатоканальної системи керування доступом для розуміння коректності працездатності системи і визначення ділянки для подальших покращень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бойченко В. В., Пузирьов С. В. Багатоканальна система керування доступом на базі Raspberry Pi та Open CV. Могилянські читання – 2022 : тези доп. XXV Всеукр. наук.-метод. конф. Миколаїв, 7–11 листоп. 2022 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2022. С. 111–113.

2. The three elements of access control. URL: https://www.techrepublic.com/article/the-three-elements-of-access-control/ (дата звернення 10.12.2022).

3. Біометричні системи доступу. URL: https://cctv.lviv.ua/biometrychni-systemy-kontrolyu-dostupu.php/ (дата звернення 10.12.2022).

4. Biometrics in physical security, where to begin? URL: https://www.nedapsecurity.com/insight/biometrics-in-physical-security-where-to-begin/ (дата звернення 10.12.2022).

5. PIN-Code Best Practices for Secure Access Control. URL: https://www.linkedin.com/pulse/pin-code-best-practices-secure-access-control-tom-piston (дата звернення 10.12.2022).

6. Guide to physical access control systems. URL: https://www.openpath.com/blog-post/physical-access-control (дата звернення 10.12.2022).

7. Learn How to Implement Face Recognition using OpenCV with Python! URL: https://www.analyticsvidhya.com/blog/2021/06/learn-how-to-implement-face-recognition-using-opencv-with-python/ (дата звернення 10.12.2022).

8. Face Recognition with OpenCV. URL: https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html (дата звернення 10.12.2022).

9. Access Control Topology. URL: https://protectorsecurity.net/protector-security-business-services/ontario-business-security-services-access-control-topology-2/ (дата звернення 10.12.2022).

10. OpenCV – Overview. URL: https://www.geeksforgeeks.org/opencv-overview/ (дата звернення 10.12.2022).

11. About OpenCV. URL: https://opencv.org/about/ (дата звернення 10.12.2022).

12. Zhicheng Zhong, Hongqin Li. Recognition and prediction of ground vibration signal based on machine learning algorithm, 2020. P. 1937–1947.

13. Сейфи, їх види і призначення. URL: http://www.apache-ssl.com/bankovskoe-oborudovanie/seiefi-ix-vidi-i-naznachenie.html. (дата звернення 10.12.2022).

14. Stephen Elliott. Signal Processing for Active Control. Academic Press. 2000. №20(3): 511.

15. Waldemar Rebizant , Janusz Szafran , Andrzej Wiszniewski. Digital Signal Processing in Power System Protection and Control. Springer, Berlin. 2011. №3:622.

16. John Wiley & Sons. Computational methods of signal recovery and recognition. 605 Third Ave. New York, NY. 1992.

17. Stephen Spiro, Ray Dixon. Sensory Mechanisms in Bacteria: Molecular Aspects of Signal Recognition. Caister Academic Press. 2010.

18. Thampi, S.M., Krishnan, S., Corchado, J.M., Das, S., Wozniak, M., Al-Jumeily, D. Advances in Signal Processing and Intelligent Recognition Systems. SIRS. 2017.

19. C. Armero, M.J. Bayarri, in International Encyclopedia of the Social & Behavioral Sciences, 2001

20. A Comparative Study of Facial Recognition Techniques. URL: https://www.diva-portal.org/smash/get/diva2:1327708/FULLTEXT01.pdf (дата звернення 24.01.2023).

21. Basic Probability. URL: https://seeing-theory.brown.edu/basic-probabil-ity/index.html#:~:text=Probability%20theory%20is%20the%20mathematical,impossibility%20and%201%20indicates%20certainty. (дата звернення 26.01.2023).

22. Багатоканальна СМО з відмовами в обслуговуванні. URL: http://ebib.pp.ua/bagatokanalna-smo-z-vidmovami-v-obslugovuvanni-ekonomiko-matematichni-metodi.html (дата звернення 26.01.2023).

23. Difference Between IRIS Recognition and Retina Scanning. URL: https://blog.mantratec.com/difference-between-iris-recognition-retina-scanning (дата звернення 18.01.2023).

24. Drawio. URL: https://app.diagrams.net/ (дата звернення 30.01.2023).

25. Raspberry Pi 4 Computer. URL: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf (дата звернення 31.01.2023).

26. Basic Fingerprint Sensor With Socket Header Cable. URL: https://www.adafruit.com/product/4690 (дата звернення 31.01.2023).

27. UDOO BOLT V8. URL: https://shop.udoo.org/en/udoo-bolt-v8.html (дата звернення 05.02.2023).

28. Jetson Xavier NX Series. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/ (дата звернення 06.02.2023).

29. Raspberry Pi Camera Module 2. URL: https://www.raspberrypi.com/products/camera-module-v2/ (дата звернення 24.01.2023).

30. OpenMV Cam H7. URL: https://openmv.io/products/openmv-cam-h7 (дата звернення 08.02.2023).

31. What is Python? Executive Summary. URL: https://www.python.org/doc/essays/blurb/. (дата звернення 09.02.2023).

32. How to Setup FTP on the Raspberry Pi. URL: https://pimylifeup.com/raspberry-pi-ftp/ (дата звернення 10.02.2023).

33. How to Train your Raspberry Pi for Facial Recognition. URL: https://www.tomshardware.com/how-to/raspberry-pi-facial-recognition (дата звернення 12.02.2023).

34. Adafruit_CircuitPython_Fingerprint. URL: https://github.com/adafruit/Adafruit_CircuitPython_Fingerprint (дата звернення 14.02.2023).

35. ZeroCam NOIR - Camera for Raspberry Pi Zero. URL: https://thepihut.com/products/zerocam-noir-camera-for-raspberry-pi-zero (дата звернення 08.02.2023).

ДОДАТОК А

Код програми

Файл Main.py

```
import RPi.GPIO as GPIO

import time

import cv2

import numpy as np

import os

import sys

import pyfingerprint

from pyfingerprint.pyfingerprint import PyFingerprint



# Set up GPIO pins for servo motor and LED

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.OUT)

GPIO.setup(23, GPIO.OUT)

# Connect to fingerprint scanner

try:

    f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

    if not f.verifyPassword():

        raise ValueError('The given fingerprint sensor password is wrong!')

except Exception as e:

    print('Exception message: ' + str(e))

    sys.exit(1)
```

```
    # Define function to capture video stream and detect faces
    def detect_faces():
        face_cascade = cv2.CascadeClassifier('/usr/share/OpenCV/haarcascades/haarcascade_frontalface_default.xml')
        cap = cv2.VideoCapture(0)
        cap.set(3, 640)
        cap.set(4, 480)
        while True:
            ret, frame = cap.read()
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, 1.3, 5)
            for (x,y,w,h) in faces:
                cv2.rectangle(frame, (x,y), (x+w,y+h), (255,0,0), 2)
            cv2.imshow('frame', frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        cap.release()
        cv2.destroyAllWindows()
    # Define function to scan fingerprint and unlock door
    def scan_fingerprint():
        try:
            print('Waiting for finger...')
            while not f.readImage():
                pass
```

```
            f.convertImage(0x01)

            result = f.searchTemplate()

            position = result[0]

            if position == -1:

                print('Fingerprint not found')

                return False

            else:

                print('Fingerprint found at position #' + str(position))

                GPIO.output(23, GPIO.HIGH)

                time.sleep(1)

                GPIO.output(23, GPIO.LOW)

                GPIO.output(18, GPIO.HIGH)

                time.sleep(2)

                GPIO.output(18, GPIO.LOW)

                return True

        except Exception as e:

            print('Exception message: ' + str(e))

            return False

    # Run main loop

    while True:

        choice = input('Enter choice (1=scan fingerprint, 2=detect faces): ')

        if choice == '1':

            if scan_fingerprint():

                print('Access granted')

            else:
```

```
                print('Access denied')

        elif choice == '2':

            detect_faces()

        else:

            print('Invalid choice')
```

Файл fingerprint_simpletest_rpi.py

```
import time
import serial

import adafruit_fingerprint

# import board
# uart = busio.UART(board.TX, board.RX, baudrate=57600)

# If using with a computer such as Linux/RaspberryPi, Mac, Windows with USB/serial converter:
#uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)

# If using with Linux/Raspberry Pi and hardware UART:
uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)

# If using with Linux/Raspberry Pi 3 with pi3-disable-bt
# uart = serial.Serial("/dev/ttyAMA0", baudrate=57600, timeout=1)

finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)
def get_fingerprint():
    """Get a finger print image, template it, and see if it matches!"""
    print("Waiting for image...")
    while finger.get_image() != adafruit_fingerprint.OK:
        pass
    print("Templating...")
    if finger.image_2_tz(1) != adafruit_fingerprint.OK:
        return False
    print("Searching...")
    if finger.finger_search() != adafruit_fingerprint.OK:
        return False
    return True
```

```
# pylint: disable=too-many-branches
def get_fingerprint_detail():
    """Get a finger print image, template it, and see if it matches!
    This time, print out each error instead of just returning on failure"""
    print("Getting image...", end="")
    i = finger.get_image()
    if i == adafruit_fingerprint.OK:
        print("Image taken")
    else:
        if i == adafruit_fingerprint.NOFINGER:
            print("No finger detected")
        elif i == adafruit_fingerprint.IMAGEFAIL:
            print("Imaging error")
        else:
            print("Other error")
        return False

    print("Templating...", end="")
    i = finger.image_2_tz(1)
    if i == adafruit_fingerprint.OK:
        print("Templated")
    else:
        if i == adafruit_fingerprint.IMAGEMESS:
            print("Image too messy")
        elif i == adafruit_fingerprint.FEATUREFAIL:
            print("Could not identify features")
        elif i == adafruit_fingerprint.INVALIDIMAGE:
            print("Image invalid")
        else:
            print("Other error")
        return False

    print("Searching...", end="")
    i = finger.finger_fast_search()
    # pylint: disable=no-else-return
    # This block needs to be refactored when it can be tested.
    if i == adafruit_fingerprint.OK:
        print("Found fingerprint!")
        return True
```

```
    else:
        if i == adafruit_fingerprint.NOTFOUND:
            print("No match found")
        else:
            print("Other error")
        return False


# pylint: disable=too-many-statements
def enroll_finger(location):
    """Take a 2 finger images and template it, then store in 'location'"""
    for fingerimg in range(1, 3):
        if fingerimg == 1:
            print("Place finger on sensor...", end="")
        else:
            print("Place same finger again...", end="")

        while True:
            i = finger.get_image()
            if i == adafruit_fingerprint.OK:
                print("Image taken")
                break
            if i == adafruit_fingerprint.NOFINGER:
                print(".", end="")
            elif i == adafruit_fingerprint.IMAGEFAIL:
                print("Imaging error")
                return False
            else:
                print("Other error")
                return False

        print("Templating...", end="")
        i = finger.image_2_tz(fingerimg)
        if i == adafruit_fingerprint.OK:
            print("Templated")
        else:
            if i == adafruit_fingerprint.IMAGEMESS:
                print("Image too messy")
            elif i == adafruit_fingerprint.FEATUREFAIL:
                print("Could not identify features")
            elif i == adafruit_fingerprint.INVALIDIMAGE:
```

```
                print("Image invalid")
            else:
                print("Other error")
            return False

        if fingerimg == 1:
            print("Remove finger")
            time.sleep(1)
            while i != adafruit_fingerprint.NOFINGER:
                i = finger.get_image()

    print("Creating model...", end="")
    i = finger.create_model()
    if i == adafruit_fingerprint.OK:
        print("Created")
    else:
        if i == adafruit_fingerprint.ENROLLMISMATCH:
            print("Prints did not match")
        else:
            print("Other error")
        return False

    print("Storing model #%d..." % location, end="")
    i = finger.store_model(location)
    if i == adafruit_fingerprint.OK:
        print("Stored")
    else:
        if i == adafruit_fingerprint.BADLOCATION:
            print("Bad storage location")
        elif i == adafruit_fingerprint.FLASHERR:
            print("Flash storage error")
        else:
            print("Other error")
        return False

    return True


def save_fingerprint_image(filename):
    """Scan fingerprint then save image to filename."""
    while finger.get_image():
```

```
            pass

        # let PIL take care of the image headers and file structure
        from PIL import Image  # pylint: disable=import-outside-toplevel

        img = Image.new("L", (256, 288), "white")
        pixeldata = img.load()
        mask = 0b00001111
        result = finger.get_fpdata(sensorbuffer="image")

        # this block "unpacks" the data received from the fingerprint
        #   module then copies the image data to the image placeholder "img"
        #   pixel by pixel.  please refer to section 4.2.1 of the manual for
        #   more details.  thanks to Bastian Raschke and Danylo Esterman.
        # pylint: disable=invalid-name
        x = 0
        # pylint: disable=invalid-name
        y = 0
        # pylint: disable=consider-using-enumerate
        for i in range(len(result)):
            pixeldata[x, y] = (int(result[i]) >> 4) * 17
            x += 1
            pixeldata[x, y] = (int(result[i]) & mask) * 17
            if x == 255:
                x = 0
                y += 1
            else:
                x += 1

        if not img.save(filename):
            return True
        return False

    def get_num(max_number):
        """Use input() to get a valid number from 0 to the maximum size
        of the library. Retry till success!"""
        i = -1
        while (i > max_number - 1) or (i < 0):
            try:
                i = int(input("Enter ID # from 0-{}: ".format(max_number - 1)))
            except ValueError:
```

```
                pass
        return i


    while True:
        print("----------------")
        if finger.read_templates() != adafruit_fingerprint.OK:
            raise RuntimeError("Failed to read templates")
        print("Fingerprint templates: ", finger.templates)
        if finger.count_templates() != adafruit_fingerprint.OK:
            raise RuntimeError("Failed to read templates")
        print("Number of templates found: ", finger.template_count)
        if finger.read_sysparam() != adafruit_fingerprint.OK:
            raise RuntimeError("Failed to get system parameters")
        print("Size of template library: ", finger.library_size)
        print("e) enroll print")
        print("f) find print")
        print("d) delete print")
        print("s) save fingerprint image")
        print("r) reset library")
        print("q) quit")
        print("----------------")
        c = input("> ")

        if c == "e":
            enroll_finger(get_num(finger.library_size))
        if c == "f":
            if get_fingerprint():
                print("Detected    #",    finger.finger_id,    "with    confidence",
finger.confidence)
            else:
                print("Finger not found")
        if c == "d":
            if          finger.delete_model(get_num(finger.library_size))          ==
adafruit_fingerprint.OK:
                print("Deleted!")
            else:
                print("Failed to delete")
        if c == "s":
            if save_fingerprint_image("fingerprint.png"):
                print("Fingerprint image saved")
```

```
            else:
                print("Failed to save fingerprint image")
        if c == "r":
            if finger.empty_library() == adafruit_fingerprint.OK:
                print("Library empty!")
            else:
                print("Failed to empty library")
        if c == "q":
            print("Exiting fingerprint example program")
            raise SystemExit
```

Файл LED-Fingerprint-Control.py

```
import hashlib
from pyfingerprint.pyfingerprint import PyFingerprint
from pyfingerprint.pyfingerprint import FINGERPRINT_CHARBUFFER1

#Control for LED, setting up the GPIO
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(21, GPIO.OUT)



## Search for a finger
##

## Tries to initialize the sensor
try:
    f = PyFingerprint('/dev/ttyS0', 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given fingerprint sensor password is wrong!')

except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

## Gets some sensor information
print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+
str(f.getStorageCapacity()))
```

```
    ## Tries to search the finger and calculate hash
    try:
        print('Waiting for finger...')

        ## Wait that finger is read
        while ( f.readImage() == False ):
            pass

        ## Converts read image to characteristics and stores it in charbuffer 1
        f.convertImage(FINGERPRINT_CHARBUFFER1)

        ## Searchs template
        result = f.searchTemplate()

        positionNumber = result[0]
        accuracyScore = result[1]

        if ( positionNumber == -1 ):
            print('No match found!')
            exit(0)
        else:
            print('Found template at position #' + str(positionNumber))
            print('The accuracy score is: ' + str(accuracyScore))
            GPIO.output(21, GPIO.HIGH)
            time.sleep(1)
            GPIO.output(21, GPIO.LOW)

        ## OPTIONAL stuff
        ##

        ## Loads the found template to charbuffer 1
        f.loadTemplate(positionNumber, FINGERPRINT_CHARBUFFER1)

        ## Downloads the characteristics of template loaded in charbuffer 1
        characterics                                                                    =
str(f.downloadCharacteristics(FINGERPRINT_CHARBUFFER1)).encode('utf-8')

        ## Hashes characteristics of template
        print('SHA-2 hash of template: ' + hashlib.sha256(characterics).hexdigest())
```

```
except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

Файл headshot_picam.py

```
import cv2
from picamera import PiCamera
from picamera.array import PiRGBArray

name = 'Volodymyr'

cam = PiCamera()
cam.resolution = (512, 304)
cam.framerate = 10
rawCapture = PiRGBArray(cam, size=(512, 304))

img_counter = 0

while True:
    for frame in cam.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
        image = frame.array
        cv2.imshow("Press Space to take a photo", image)
        rawCapture.truncate(0)

        k = cv2.waitKey(1)
        rawCapture.truncate(0)
        if k%256 == 27: # ESC pressed
            break
        elif k%256 == 32:
            # SPACE pressed
            img_name = "dataset/"+ name +"/image_{}.jpg".format(img_counter)
            cv2.imwrite(img_name, image)
            print("{} written!".format(img_name))
            img_counter += 1

    if k%256 == 27:
        print("Escape hit, closing...")
        break
```

```
cv2.destroyAllWindows()
```

Файл train_model.py

```
# import the necessary packages
from imutils import paths
import face_recognition
#import argparse
import pickle
import cv2
import os

# images are located in the dataset folder
print("[INFO] start processing faces...")
imagePaths = list(paths.list_images("dataset"))

# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
# extract the person name from the image path
print("[INFO] processing image {}/{}".format(i + 1,
       len(imagePaths)))
name = imagePath.split(os.path.sep)[-2]

# load the input image and convert it from RGB (OpenCV ordering)
# to dlib ordering (RGB)
image = cv2.imread(imagePath)
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# detect the (x, y)-coordinates of the bounding boxes
# corresponding to each face in the input image
boxes = face_recognition.face_locations(rgb,
       model="hog")

# compute the facial embedding for the face
encodings = face_recognition.face_encodings(rgb, boxes)

# loop over the encodings
```

```
for encoding in encodings:
	# add each encoding + name to our set of known names and
	# encodings
	knownEncodings.append(encoding)
	knownNames.append(name)

# dump the facial encodings + names to disk
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open("encodings.pickle", "wb")
f.write(pickle.dumps(data))
f.close()
```