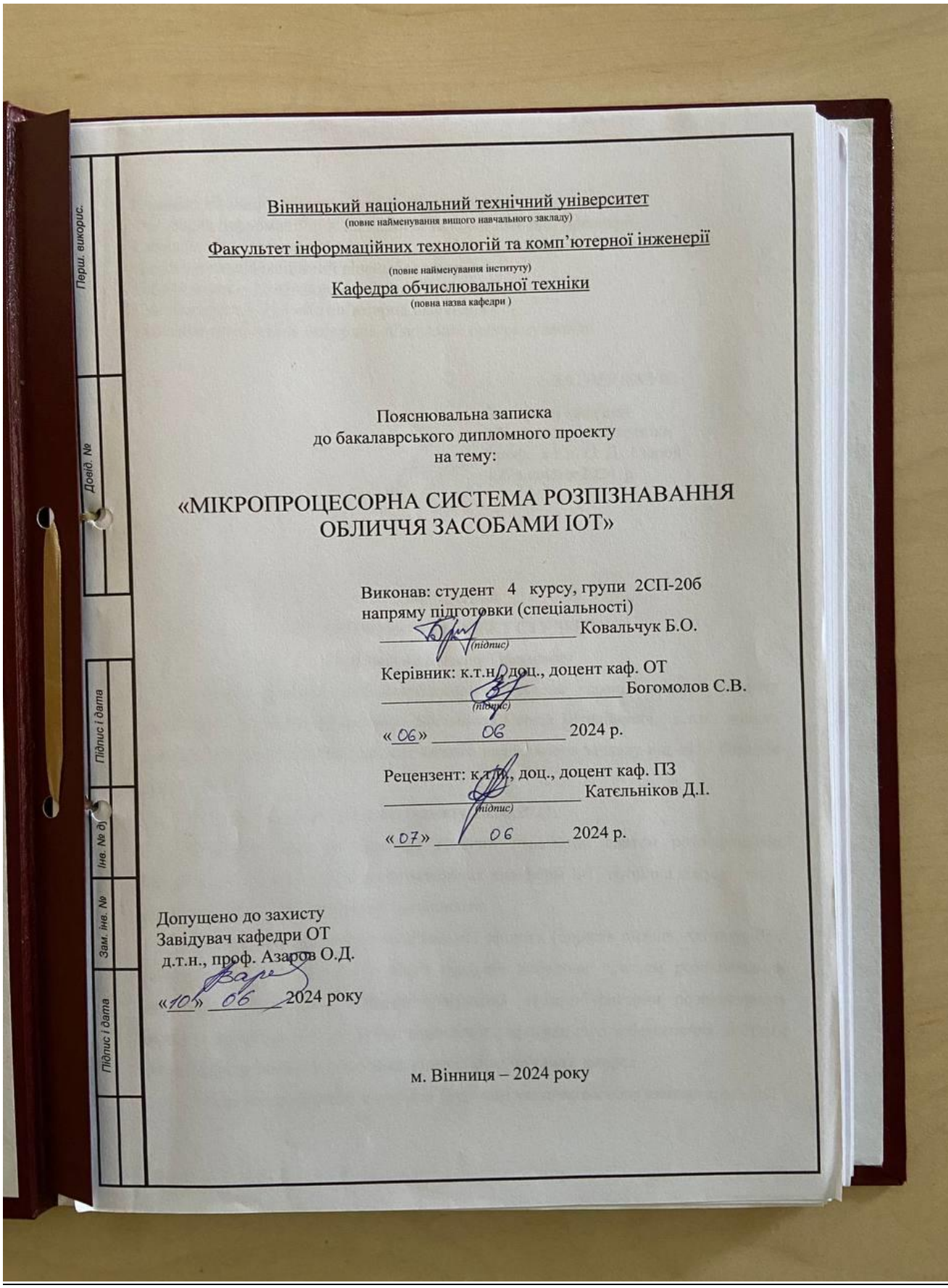

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту)

Кафедра обчислювальної техніки
(повна назва кафедри)

Пояснювальна записка
до бакалаврського дипломного проекту
на тему:

«МІКРОПРОЦЕСОРНА СИСТЕМА РОЗПІЗНАВАННЯ ОБЛИЧЧЯ ЗАСОБАМИ ІОТ»

Виконав: студент 4 курсу, групи 2СП-20б
напряму підготовки (спеціальності)
_____ Ковальчук Б.О.
(підпис)

Керівник: к.т.н., доц., доцент каф. ОТ
_____ Богомолов С.В.
(підпис)

«06» 06 2024 р.

Рецензент: к.т.н., доц., доцент каф. ПЗ
_____ Катєльніков Д.І.
(підпис)

«07» 06 2024 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.

«10» 06 2024 року

м. Вінниця – 2024 року

Перш. використ.
Довід. №
Підпис і дата
Інв. № дубл.
Зам. інв. №
Підпис і дата

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 123 «Комп'ютерна інженерія»
Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ

Завідувач кафедри
обчислювальної техніки
проф., д.т.н. О. Д. Азаров
«6» лютого 2024 р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Ковальчуку Богдану Олеговичу

1 Тема проекту «Мікропроцесорна система розпізнавання обличчя засобами IoT», керівник проекту Богомолов Сергій Віталійович, к.т.н., доцент кафедри ОТ, затверджена наказом вищого навчального закладу від «11» березня 2024 р. №80.

2 Строк подання студентом проекту:10.06.2024.

3 Вихідні дані до проекту: технічні параметри систем розпізнавання обличчя, технічний опис мікропроцесорних платформ IoT, технічна документація на модуль ESP32 на електронні компоненти.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, огляд і аналіз підходів побудови систем розпізнавання обличчя, проектування апаратно-програмної частини системи розпізнавання обличчя засобами IoT, розробка віддаленого програмного забезпечення системи розпізнавання обличчя, висновки, список використаних джерел.

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

6 Консультанти розділів проєкту наведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання отримав	завдання прийняв
1,2,3	к.т.н., доц. каф. ОТ Богомолов С.В.		

7 Дата видачі завдання: 08.03.2024 р.

8 Календарний план виконання роботи подано в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів виконання бакалаврського проекту	Строк виконання	Примітка
1	Постановка задач проекту	08.03.24	виконав
2	Огляд інформаційних джерел	11.03-24.03.24	виконав
3	Огляд і аналіз підходів побудови систем розпізнавання обличчя	25.03-07.04.24	виконав
4	Проектування апаратно-програмної частини системи розпізнавання обличчя засобами IoT	08.04-21.04.24	виконав
5	Розробка віддаленого програмного забезпечення системи розпізнавання обличчя	22.04-05.05.24	виконав
6	Оформлення пояснювальної записки та ілюстративного матеріалу	06.05-12.05.24	виконав
7	Аналіз виконання проекту. Висновки. Додатки	13.05-19.06.24	виконав
8	Перевірка якості виконання бакалаврського проекту та усунення недоліків	20.05.24	виконав

Студент _____ Б.О. Ковальчук
(підпис) (прізвище та ініціали)

Керівник роботи _____ С.В. Богомолов
(підпис) (прізвище та ініціали)

Анотація

УДК 681.004

Ковальчук Б.О. Мікропроцесорна система розпізнавання обличчя засобами IoT. Бакалаврський кваліфікаційний проєкт зі спеціальності 123 «Комп'ютерна інженерія», освітня програма «Системне програмування». Вінниця: ВНТУ, 2024. 99 с. На укр. мові. Бібліогр.: 17 назв; рис.: 47.

У бакалаврському дипломному проєкті розроблено мікропроцесорну систему розпізнавання обличчя засобами IoT. Цей апаратно-програмний комплекс базується на використанні технологій IoT, зокрема ESP32-CAM, та застосовує метод Віоли-Джонса і локальні бінарні шаблони.

Під час роботи були виконані наступні завдання: проаналізовано існуючі підходи та алгоритми розпізнавання обличчя, що дозволило визначити найефективніші методи та їх сильні і слабкі сторони; проведено аналіз ринкових систем розпізнавання для виявлення їх переваг і недоліків, що допомогло обрати оптимальні рішення; вибрано найбільш підходящі інструментальні засоби для розробки, що забезпечило створення ефективної системи; спроектовано та реалізовано систему розпізнавання облич на основі IoT з використанням методу Віоли-Джонса та локальних бінарних шаблонів; проведено тестування, що підтвердило працездатність апаратно-програмної частини.

Розроблена мікропроцесорна система розпізнавання обличчя засобами IoT може бути використана для різних завдань відеоаналітики, зокрема для систем контролю доступу та ідентифікації особистості. Крім того, під час роботи було здійснено огляд сучасних систем розпізнавання, виявлено їхні недоліки та проблеми, які впливають на ефективність роботи, вивчено методи обробки зображень та проведено аналіз сучасних алгоритмів розпізнавання.

Ключові слова: мікропроцесорна система, IoT, ESP32-CAM, розпізнавання, біометрична ідентифікація, метод Віоли-Джонса, локальні бінарні шаблони.

Annotation

Kovalchuk B.O. Microprocessor face recognition system by means of IoT. Bachelor qualification project in specialty 123 "Computer engineering", educational program "System programming". Vinnytsia: VNTU, 2024. 99 p. In Ukrainian speech Bibliography: 17 titles; Fig.: 47.

In the bachelor's diploma project, a microprocessor-based face recognition system was developed by means of IoT. This hardware and software complex is based on the use of IoT technologies, in particular ESP32-CAM, and applies the Viola-Jones method and local binary templates.

During the work, the following tasks were performed: the existing approaches and algorithms of face recognition were analyzed, which made it possible to determine the most effective methods and their strengths and weaknesses; an analysis of market recognition systems was carried out to identify their advantages and disadvantages, which helped to choose optimal solutions; the most suitable tools for development were selected, which ensured the creation of an effective system; designed and implemented an IoT-based face recognition system using the Viola-Jones method and local binary patterns; testing was carried out, which confirmed the functionality of the hardware and software part.

The developed microprocessor-based face recognition system by means of IoT can be used for various tasks of video analytics, in particular, for access control and personal identification systems. In addition, during the work, an overview of modern recognition systems was carried out, their shortcomings and problems affecting work efficiency were identified, image processing methods were studied and analysis of modern recognition algorithms was carried out.

Keywords: microprocessor system, IoT, ESP32-CAM, recognition, biometric identification, Viola-Jones method, local binary patterns.

ЗМІСТ

Вступ

У теперішній час більшість підприємств уже почали використовувати у своїх охоронних системах – СКУД. Сучасні біометричні системи ідентифікації значно підвищують рівень безпеки для підприємств та їхніх працівників. Раніше на входах підприємств встановлювали електронні турнікети зі зчитувачами карток або сканерами відбитків пальців. Однак, з розвитком біометричних технологій, компанії все частіше переходять на нові методи ідентифікації, які є більш точними та зручними. Наприклад, розпізнавання облич за допомогою відеопотоку в реальному часі стає дедалі **актуальнішим**.

З огляду на стрімке зростання потреб підприємств у біометричних системах, аналітики TrendForce прогнозують подальше збільшення інтересу до технологій розпізнавання облич у найближчі роки. Основними сферами застосування залишатимуться системи контролю доступу та безпеки, а також моніторингові системи, але їх використання буде поступово розширюватися на інші галузі. Наприклад, біометричні технології можуть знайти застосування у фінансовій сфері для підтвердження особи при проведенні транзакцій, в охороні здоров'я для ідентифікації пацієнтів, а також у роздрібній торгівлі для поліпшення обслуговування клієнтів.

Окрім підвищення рівня безпеки, біометричні системи пропонують додаткові переваги, такі як зручність використання та зменшення потреби у фізичних носіях інформації (наприклад, картках чи ключах), що може призвести до зниження витрат на їхнє виробництво та обслуговування. Такі системи також можуть інтегруватися з іншими технологіями, як-от штучний інтелект і великі дані, для створення більш комплексних рішень з управління безпекою та ефективністю роботи підприємств. [1].

Інтернет речей (IoT) значно підвищує ефективність та безпеку систем розпізнавання облич, дозволяючи моніторинг в реальному часі та автоматизацію процесів. Це забезпечує інтеграцію з іншими безпековими системами, персоналізацію послуг у роздрібній торгівлі та охороні здоров'я, а також дозволяє

аналізувати поведінку користувачів. Використання IoT в цих системах піднімає важливі питання захисту персональних даних, які потребують надійних рішень.

Метою проєкту є розробка мікропроцесорної системи розпізнавання обличчя засобами IoT. Дана система є апаратно-програмним комплексом, що базується на засобах технологій IoT таких як ESP32-CAM та використанням методу Віоли-Джонса та локальних бінарних шаблонів.

Щоб досягнути поставлену мету потрібно вирішити такі **задачі**:

– аналіз та огляд сучасних підходів для розпізнавання обличчя;

– визначити типи алгоритмів, які використовуються для розпізнавання;

– провести огляд та аналіз існуючих систем розпізнавання на ринку, зазначити їхні сильні та слабкі сторони;

– дослідити основні інструменти для розробки цих систем та вибрати найбільш підходящі;

– спроектувати та програмно реалізувати систему розпізнавання обличчя засобами IoT.

Об’єктом дослідження є процеси, що протікають у системах розпізнавання обличчя у відеопотоках.

Предметом дослідження є методи та засоби розпізнавання осіб у відеопотоках.

Практична цінність полягає у можливості підвищення точності розпізнавання розпізнавати особи у відеопотоках у режимі реального часу з використанням апаратно-програмного засобу IoT ESP32-CAM та методу Віоли-Джонса і локальних бінарних шаблонів.

Наукова новизна полягає у вдосконалені системи розпізнавання обличчя шляхом використання засобів IoT у поєднанні з системи машинного навчання виявляння об'єктів.

Апробацію результатів наукової роботи було проведено на науковій конференції «Молодь в науці: дослідження, проблеми, перспективи (МН–2024)» у кількості 2-х публікацій [2, 3].

1 Огляд і аналіз підходів побудови систем розпізнавання обличчя

Процес розпізнавання обличчя складається з кількох завдань, спрямованих на ідентифікацію людини за цифровим зображенням або відеофрагментом. Спочатку система отримує зображення з камери і за допомогою алгоритмів визначає межі обличчя (етап виявлення). Наступний етап – розпізнавання, де обличчя піддається різним трансформаціям (зміна яскравості, вирівнювання, масштабування тощо) для приведення його до стандартного вигляду. Потім відбувається обчислення характеристик і порівняння їх з еталонами, збереженими в базі даних. Цей завершальний етап може бути ідентифікацією або верифікацією, залежно від системи.

Верифікація передбачає порівняння за схемою "1:1". Система порівнює біометричний зразок з одним шаблоном, що зберігається в базі даних, і відповідає на питання: "Чи є ця людина тією, чий шаблон використовується для порівняння?". Ідентифікація здійснюється за схемою "1:N". Система порівнює біометричний зразок з усіма шаблонами, що зберігаються в базі даних, і відповідає на питання: "Хто це?".

На рисунку 1.1 зображено узагальнений алгоритм розпізнавання обличчя із зображення.

Рисунок 1.1 – Узагальнений алгоритм розпізнавання обличчя

1.1 Огляд методів виявлення обличчя

Після передачі зображення у формі цифрових даних з камери на комп'ютер, воно піддається обробці за допомогою спеціального алгоритму. Цей алгоритм визначає розташування області обличчя за його основними рисами. Наразі існує багато методів виявлення осіб, більшість з яких є комбінацією інших методів. Проте всі вони можуть бути розділені на дві категорії: методи, що базуються на знаннях і використовують досвід людини, і методи виявлення особи за зовнішніми ознаками. Останні вимагають етапу навчання системи, який полягає в обробці тестових зображень. Класифікація цих методів виявлення наведено рисунку 2.1.

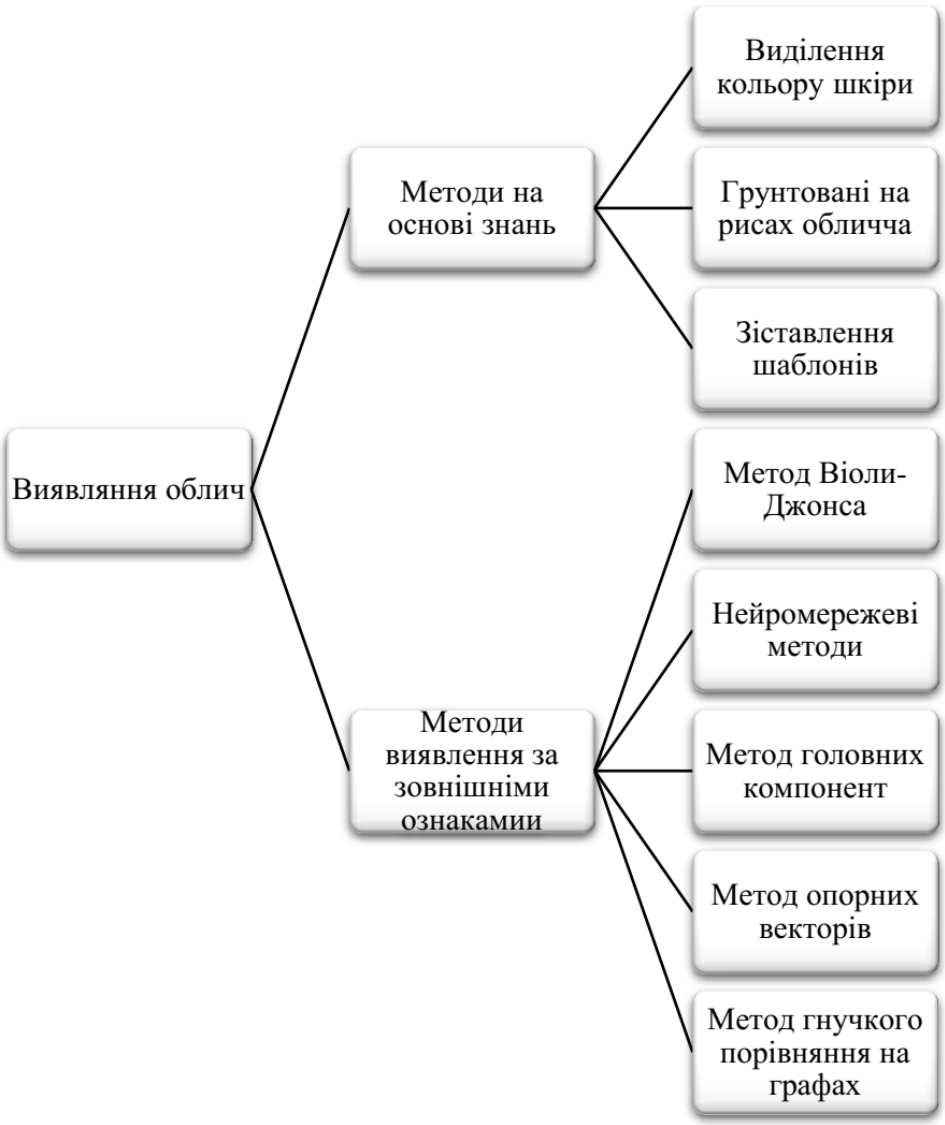

Рисунок 1.2 – Систематизація методів виявлення осіб за їх обличчями

Методи, що базуються на знаннях. Використовують інформацію про особу, її риси, форму, текстуру. В цих підходах відзначається певний набір правил, який включає в себе характеристики та особливості конкретної особи., яким повинен відповідати фрагмент кадру, щоб вважатися людським обличчям. Визначити такий набір правил досить просто. Всі ці правила – це формалізовані знання, якими керується людина при визначенні, чи особа перед ним чи не особа.

Наприклад, одним з основних правил є те, що області очей, носа та рота відрізняються за яскравістю в порівнянні з іншими частинами обличчя; очі на обличчі завжди розташовуються симетрично між собою. Засновуючись на цих та інших схожих властивостях, розробляються алгоритми, які під час виконання перевіряють наявність правил на зображенні.

Ця категорія методів включає більш загальний підхід – метод порівняння з шаблоном. В цьому методі використовується опис властивостей окремих частин обличчя та їх заданого взаємного розташування визначається стандарт особи (шаблон), з яким надалі порівнюють вихідне зображення.

Підходи, що базуються на знаннях, стали досить поширеними і відзначаються хорошими результатами, проте вони показують хороші результати лише на зображеннях з гарним розширенням, без шумів та з нескладним тлом. На відеокадрах з камер або відеопотоку, розміщених у громадських місцях, можуть спостерігатися різні ракурси та повороти осіб, змінне освітлення та багато об'єктів на задньому плані. Ці умови створюють складнощі для точного виявлення облич, і можуть призвести до помилок. Методи виявлення осіб за їх зовнішніми ознаками працюють у напрямку вирішення цих проблем. Вони не намагаються ретельно відтворити процеси, які відбуваються в людському мозку, а замість цього спираються на виявлення закономірностей та властивостей зображення обличчя за допомогою методів математичної статистики та машинного навчання. Ці підходи, вільні від багатьох недоліків, які притаманні іншим методам, стають все більш популярними у системах відеоспостереження. Для виявлення осіб у таких методах використовується аналіз всіх прямокутних

фрагментів зображення з метою класифікації їх на відповідні класи: до класу, що містять особу або до класу зображень без обличчя.

За рахунок такого великого обсягу роботи методи мають надмірність і велику обчислювальну складність. Для оптимізації процесу виявлення облич та прискорення його, дослідники використовують різноманітні додаткові стратегії скорочення кількості аналізованих фрагментів.

Декілька найбільш актуальних і заслуговують на увагу методів виявлення осіб розглянуті нижче:

Метод Віоли-Джоса(Viola-Jones object detection). Запропонований у 2001 році Паулом Віолою та Майклом Джонсом, цей метод став першим, що продемонстрував високу ефективність обробки зображень у реальному часі. Існують різноманітні реалізації цього методу, включаючи використання його у складі бібліотеки комп'ютерного зору OpenCV, де він реалізований як функція cvHaarDetectObjects(). [2]. Докладно цей метод розглянуто у наступному розділі.

Переваги цього методу:

– висока швидкість роботи завдяки використанню каскадного класифікатора;

– висока точність виявлення облич, повернутих на кут до 30 градусів (при більшому куті ефективність методу значно знижується).

Недоліки:

– тривалий час навчання, оскільки алгоритму потрібно проаналізувати велику кількість тестових зображень.

– існують обмеження при виявленні облич у певних положеннях.

Метод гнучкого порівняння на графах (Elastic Graph Matching) належить до 2D моделювання. Його суть полягає в зіставленні графів, які описують обличчя, де обличчя представляється у вигляді сітки з індивідуальним розташуванням вершин та ребер.

Процедура розпізнавання відбувається наступним чином: еталонний граф, що характеризує основний параметр розпізнавання, залишається незмінним, тоді

як інші графи деформуються під впливом структури обличчя, орієнтуючись на основні антропометричні точки, такі як відстань між очима, вухами, лінія носа, ширина губ тощо. Чим більше цих точок використовується, тим точніше буде система розпізнавання, а й суттєво збільшиться час на обробку одного об'єкта.

Недоліки методу:

– висока складність алгоритму розпізнавання вимагає значних обчислювальних ресурсів;

– процедура додавання нових шаблонів у базу даних є доволі складною;

– швидкодія аналітичної системи обернено пропорційно розмірам баз даних;

Приховані марківські моделі (СММ).Метод базується на статистичному порівнянні об'єкта з базою шаблонів. Приховані Марківські моделі використовують статистичні властивості сигналів та враховують їх просторові характеристики. Елементи моделі включають початкову ймовірність станів, множину спостережуваних станів, множину прихованих станів та матрицю перехідних ймовірностей. Кожному з цих елементів відповідає своя Марківська модель. Під час процесу розпізнавання людини перевіряються всі згенеровані Марківські моделі, і визначається та, яка має найбільшу ймовірність., що послідовність спостережень для об'єкта згенерована відповідною моделлю.

Недоліки:

– низька швидкість спрацьовування;

– низька здатність розрізняти і не оптимальний алгоритм навчання;

– нистема може оптимізувати лише час обробки даних та відгуку на власну модель, однак це не дозволяє зменшити час на перебір інших моделей.

Метод головних компонент (PCA). Основною метою PCA є зменшення простору ознак без значної втрати інформації, щоб він максимально точно описував "типові" образи, що належать різним людям. У задачі розпізнавання облич PCA використовується переважно для представлення обличчя як вектора малої розмірності, який потім порівнюється з еталонними векторами з бази

даних.

Набір власних векторів, отриманих один раз на навчальній вибірці, застосовується для кодування інших зображень облич, які можна представити як зважену комбінацію цих власних векторів. Використовуючи обмежену кількість власних векторів, можна отримати стислу апроксимацію вхідного зображення обличчя, яку потім можна зберігати в базі даних як вектор коефіцієнтів, що одночасно слугує ключем пошуку в базі даних.

PCA чудово відрекомендував себе у додатках. Однак тоді, коли на зображенні обличчя є значні зміни у виразі обличчя або освітленості, ефективність методу значно зменшується. Це пов'язано з тим, що метод головних компонентів спрямований на вибір підпростору з метою максимальної апроксимації вхідного набору даних, а не на розділення між різними класами осіб.

Щодо методу опорних векторів (SVM), це набір схожих алгоритмів навчання з учителем, які використовуються для класифікації та регресійного аналізу [4]. Суть методу опорних векторів полягає у знаходженні гіперплощини в ознаковому просторі, що відділяє клас зображень осіб від зображень "не-осіб". Для цього обирається гіперплощина, яка найкраще розділяє ці два класи, і відстань до якої від кожного класу є максимальною серед всіх можливих гіперплощин.

Серед переваг цього методу можна відзначити:

– високу стійкість до перенавчання;

– високу швидкість роботи, порівняно з нейронними мережами;

– можливість зниження чутливості до шуму за рахунок зниження точності.

Проте, метод має й свої недоліки:

– менша точність порівняно з іншими методами.

Нейромережеві методи. Методи глибокого навчання є досить поширеними і включають різноманітні алгоритми. Їхньою особливістю є навчання на вже

наявних прикладах, які заздалегідь внесені до бази даних. Під час процесу навчання нейронна мережа самостійно визначає важливі ознаки та встановлює взаємозв'язки між ними. Після завершення навчання нейронна мережа може використовувати цей досвід для розпізнавання раніше невідомих об'єктів. Методи глибокого навчання демонструють одні з найкращих результатів в багатьох завданнях у сфері розпізнавання, але вважаються найскладнішими для реалізації.

Перевагоє є те, що при належному налаштуванні параметрів мережі досягається висока точність виявлення.

Недоліки:

– складність процедури занесення змін (внесення будь-якої зміни потребує перенавчання мережі);

– важко формалізувати архітектуру мережі (кількість нейронів, верств, характер зв'язків);

– висока обчислювальна складність.

1.2 Початкове опрацювання відображення

Для підвищення ефективності систем розпізнавання та покращення якості виділення облич у кадрах використовується передобробка вхідних зображень. Після виявлення обличчя на кадрі та визначення його параметрів, таких як розмір, положення, поза та ключові риси, зображення зазвичай піддається нормалізації. Цей процес включає кодування, масштабування та перетворення до горизонтального положення лінії, що з'єднує центри очей. Крім того, при передобробці зображень людських облич застосовуються різні фільтри для зниження рівня шуму, такі як медіанні та гаусівські фільтри.

Методи передобробки є досить різноманітними і залежать від конкретних завдань дослідження. Один з найпоширеніших підходів - це використання нелінійних фільтрів, призначених для видалення імпульсного шуму на

зображенні. Ці фільтри працюють, визначаючи позиції кожного імпульсу та замінюючи їх значеннями фіксованої або випадкової величини.

Іншим поширеним методом є фільтр Гауса, який використовується для розмиття зображення, особливо у випадку, коли на зображенні присутні дрібні деталі, які не потребують виділення від фону.

Медіанні фільтри також широко використовуються для підсилення контурів та приглушення імпульсних шумів. Вони ефективно працюють у випадках, коли щільність шуму невелика. Ці методи передобробки дозволяють забезпечити належну якість обробки зображень перед подальшим їх аналізом та розпізнаванням.

1.3 Методи розпізнавання обличчя

Одним з ключових етапів у системах автоматичного розпізнавання облич є застосування відповідних алгоритмів для самого процесу розпізнавання. На сьогоднішній день існує багато різних алгоритмів, кожен з яких має свої особливості, швидкість та надійність.

Алгоритми розпізнавання можна розділити на дві основні категорії в залежності від технології розпізнавання, а саме на двовимірні та тривимірні. Двовимірні системи (2D-технології) працюють з плоскими, двомірними зображеннями та аналізують обличчя, зосереджуючись на його текстурі та контрастних ділянках. Однак при змінах освітлення або положенні обличчя, ці системи можуть виявити значні труднощі у розпізнаванні.

З іншого боку, тривимірні системи (3D-технології) виявляються більш стійкими до таких змін, оскільки при їхньому застосуванні враховується будова черепа. Однак вони поки не набули широкого поширення через обмежену можливість обслуговування великої кількості користувачів у режимі ідентифікації та невисоку швидкість. До того ж, для роботи 3D-технологій необхідні значні обчислювальні ресурси, а вартість відповідного устаткування набагато вища порівняно з двовимірними системами.

У сучасних системах відеоспостереження часто використовується декілька ефективних алгоритмів розпізнавання, які базуються або на значеннях пікселів, або на характерних точках.Нижче наведено короткий опис кожної з цих підгруп.

Методи, що ґрунтуються на значеннях пікселів, використовують для розпізнавання виявлених осіб колір або яскравість. Найпростішими подібними методами є:

Eigenfaces – алгоритм, запропонований у 1991 році Метью Терком та Алексом Пентландом. Алгоритм Eigenfaces став першим успішним методом розпізнавання облич, завдяки своїй унікальній ідейній концепції.

Його основна ідея полягає в тому, щоб перетворити характеристики обличчя на двовимірні зображення у градаціях сірого кольору. Під час ідентифікації особи, алгоритм порівнює ці зображення зі збереженими шаблонами та обчислює коефіцієнт відмінності, що вказує на ступінь схожості.

У добре освітлених приміщеннях та умовах, коли можливе точне сканування обличчя у фас, алгоритм Eigenfaces демонструє високу ефективність. Однак, його продуктивність різко знижується при змінах умов, таких як недостатнє освітлення.

Fisherfaces – подальшим розвитком методу Eigenfaces, призначеним для підвищення точності розпізнавання в умовах зміни освітлення або виразу обличчя. Відмінність полягає у тому, що в основі Fisherfaces лежить лінійний дискримінантний аналіз (LDA), який спрямований на максимізацію відокремленості між класами зображень.

Алгоритм Local Binary Patterns (LBP) є ще одним ефективним методом, який відображає текстурні характеристики обличчя. Його суть полягає в утворенні локальних бінарних шаблонів на основі інтенсивності пікселів. Він є ефективний метод розпізнавання осіб. При розпізнаванні методом LBP кожному пікселю зображення за допомогою функції присвоюється значення яскравості, що описує його околиця. Отримане зображення поділяється на області, кожної з яких розраховується гістограма. Далі гістограми конкатенуються та шляхом методів

машинного навчання винонується їх порівняння. У стандартному варіанті використається метод найближчого сусіда.

На відміну від EigenFaces, алгоритм є стійким до постійних однакових змін підсвітки. Ця характеристика створює умови для використання його у системах обробки в реальному часі для розпізнавання осіб.

Методи, що базуються на характерних точках відносно попередніх, не оцінюють яскравості пікселів, а використовують координати характерних точок на зображенні. Такі точки можуть включати центри очей, позицію носа, лінію брів, а також рот і т. д. Серед цього роду методів можна виділити активні моделі зовнішнього вигляду (AAM) та активні моделі форми (ASM).

AAM є статистичними моделями, які можуть бути адаптовані до реальних зображень за допомогою різних видів деформацій. Цей підхід був запропонований Тімом Кутсом та Крісом Тейлором у 1998 році. Модель містить два типи параметрів: параметри форми, які описують геометричні особливості, і параметри зовнішнього вигляду, що відображають текстурні особливості. Перед застосуванням модель має бути навчена на великій кількості апріорно розмічених зображень.

ASM враховує статистичні зв'язки між антропометричними точками на обличчі. Експерт розмічає розташування цих точок на кожному зображенні. Далі виконується узагальнений прокрустовий аналіз, щоб привести всі точки до єдиної системи координат. Потім обчислюються середня форма і матриця коваріації. На основі цих даних визначаються власні вектори, які використовуються для локалізації моделі на новому зображенні.

1.4 Аналіз продуктивності систем розпізнавання

Між спеціалізованими та аматорськими системами розпізнавання існує велика різниця. Аматорські – виглядають красивішими, дорожчими, і до них пред'являються нижчі вимоги. Також варто зазначити, що через те, що внаслідок можливих помилок в цих системах відбувається незначне порушення, їх вартість

в кінцевому підсумку виявляється значно нижчою, ніж вартість більш професійних систем, які піддаються більш суворим вимогам. Для наочного прикладу можна вказати технологію розпізнавання осіб на фотографіях. Такі системи, хоча і можуть допускати помилки, але їхні наслідки незначні, що зменшує вартість їх використання і обслуговування.

Прикладом може бути технологія розпізнавання обличчя на фотографіях у соціальних мережах, та професійну систему, призначену для пошуку людей, які розшукуються. Перша система оперує фотографіями, на яких люди, зазвичай, спрямовують свій погляд у камеру. Друга система враховує зображення людей, взятих у громадському місці, де вони можуть не спостерігати за камерою відеоспостереження або навіть намагатися приховатися від неї. Важливо також відзначити, що у разі помилки першої системи наслідки можуть бути непомітні і нічого жахливого не станеться, а от якщо система пошуку розшукуваних людей припускатиме багато помилок, то це серйозний недолік, оскільки будь-яка пропущена особа може бути злодієм чи терористом.

Обираючи систему важливою є правильно відзначити не лише яка мета, а й критерії з метою оцінки ефективності цієї системи. А це означає визначити пристосованість системи до роботи в певних умовах та різні пороги припустимих помилок.

В сучасний час для аналізу результативності систем розпізнавання ідентифікуються та використовуються два ключові показники. У біометричній сфері вони відомі як Коефіцієнт помилкового доступу (FAR - False Acceptance Rate) та Коефіцієнт помилкового відмови (FRR - False Rejection Rate).

Коефіцієнт помилкового відмови вказує на ймовірність того, наскільки часто система помилково відхиляє правильні відповіді, або, іншими словами, коли вона не здатна впізнати особу в базі даних. Чим менше цей коефіцієнт, тим більша точність розпізнавання.

Коефіцієнт помилкового допуску, навпаки, показує ймовірність того, наскільки часто система неправильно визнає дійсні відповіді.

Також існує ще один показник, який дозволяє порівнювати біометричні системи – коефіцієнт EER (рівень помилок при рівній ймовірності помилок). Коефіцієнт EER враховує обидві помилки (помилку прийому та помилку відхилення) у рівній мірі. Чим нижче значення коефіцієнта EER, тим вища точність біометричної системи.

На рисунку 1.3 показані взаємозв'язки характеристик FAR, FRR та EER.

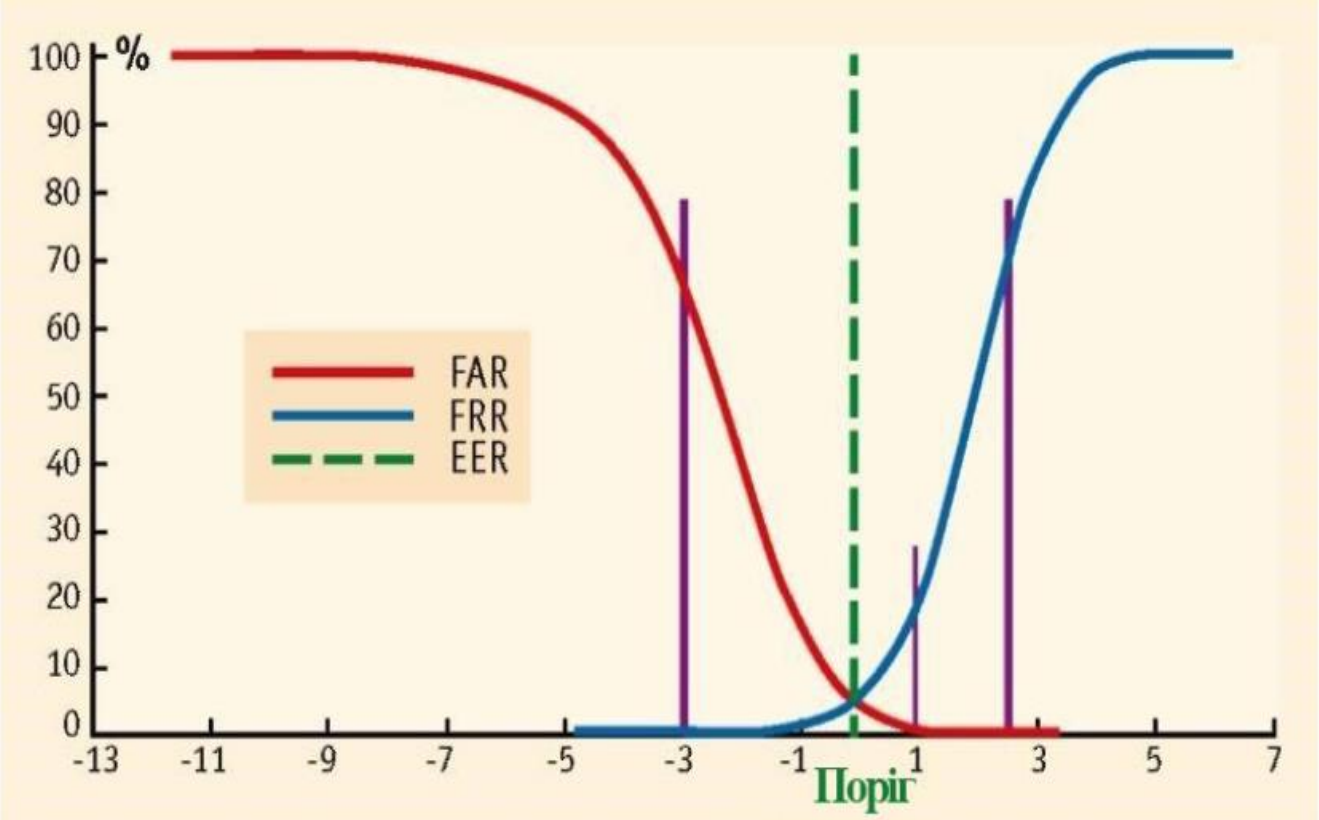

Рисунок 1.3 – Взаємозв'язок характеристик FAR, FRR та EER

Для порівняння різних систем та візуалізації залежності між помилками застосовують графік ROC (Receiver Operating Characteristic). Крива ROC ілюструє, наскільки сильно зразок повинен відрізнятися від шаблону, щоб система вважала його відповідним. При налаштуванні порогів помилок важливо пам'ятати, що значення FAR і FRR взаємопов'язані: зменшення одного призводить до збільшення іншого. Таким чином, зниження порогу може зменшити кількість помилкових відкидань, але збільшити кількість помилкових прийомів.

1.5 Огляд і аналіз існуючих систем

З кожним роком зростає зацікавленість у технологіях розпізнавання осіб. На сьогоднішній день існує значна кількість систем розпізнавання, які застосовуються у різних сферах людського життя. У цьому напрямку лідерами вважаються:

- FaceVACS-VideoScan компанії Cognitec Systems GMBH (Німеччина);
- NEC's Face Recognition компанії NEC (Японія);
- VeriLook SDK компанії Neurotechnology (Литва).

Програмне забезпечення «FaceVACS-VideoScan», що розробляється компанією «Cognitec Systems», представляє собою простий у використанні інструмент розпізнавання облич у реальному часі за допомогою відеопотоку.

Ця система складається з ряду компонентів, включаючи відеосервер для керування потоками відео, сервер відеосканування для координації всіх процесів та здійснення основних біометричних операцій, обчислювальний вузол для розподілу навантаження, інтерфейс користувача, диспетчер сигналів для отримання повідомлень про події, а також операційну базу даних та комплект інтеграції.

На сьогоднішній день, для реалізації технології FaceVACS використовується алгоритм розпізнавання облич B10T9. Цей алгоритм відзначається стійкістю до змін міміки, обертання обличчя (на ±15º), часткового приховування, використання окулярів та змін освітлення [7].

Крім того, система FaceVACS має такі особливості:

- можливість одночасного відстеження кількох осіб;
- порівняння осіб відбувається у режимі реального часу;
- можливість відображення та надсилання даних щодо потоків;
- можливість інтерактивної реєстрації з мобільного пристрою;
- використання C++ API та Web Services API.

Розпізнавання облич NeoFace від NEC – передова система впізнавання облич, розроблена японською компанією NEC, що відзначається своєю

надзвичайною здатністю ідентифікувати осіб на кадрах з тривалою історією, навіть у випадках, коли вони носять окуляри або активно мімікуються. Усі визнані обличчя зберігаються у спеціальній базі даних, що дозволяє користувачам отримати повну історію відеозаписів з часовими мітками для будь-якого збереженого зображення у випадку необхідності.

Технологія NEC виділяється серед численних інших систем розпізнавання завдяки своїй вищій точності та надзвичайній швидкості. Вона проявляє високу ефективність у різних сценаріях, включаючи роботу з відео низької якості та сильно стислими зображеннями. NEC аналізує індивідуальні риси облич (розмір, форму зіниць, лінії носа та рота), а також їх просторові відношення для відповідності їх відповідним записам у базі даних.

Система складається з кількох модулів, які реалізують наступні алгоритми:

1. Використання методу загальної відповідності облич (GMFD), що забезпечує надзвичайно швидкий пошук та точне розпізнавання обличчя. GMFD, заснований на нейронних мережах, проводить попередній пошук пари очей та інші етапи аналізу.

2. Алгоритм методу простору збурень (PSM) дозволяє ефективно вирішувати складнощі, пов'язані з орієнтацією облич у кадрах, такими як нахилена або кутова позиція обличчя. Він використовує комплексні математичні моделі для точного визначення положення та орієнтації обличчя на зображенні.

3. Метод адаптивного регіонального злиття відповідностей (ARBM) призначений для зменшення впливу незначних змін облич (наприклад, міміка обличчя, носіння окулярів, головники) на точність розпізнавання. Цей метод використовує складні алгоритми аналізу текстури та структури обличчя для точного визначення особливостей та їх впливу на результати розпізнавання.

Система розпізнавання облич NeoFace має також низку додаткових особливостей, включаючи:

- спостерігання та контроль у режимі реального часу;
- ідентифікація на основі індивідуальних рис особи;

– множинне розпізнавання;

– можливість пошуку подій базою даних;

– ведення журналу зображень осіб;

– висока стійкість до зміни положення обличчя, здатність працювати ефективно при повороті обличчя на ±15º та нахилі голови до 45º у будь-якому напрямку від фронтального положення.

– інтуїтивно зрозуміле управління методом "Drag and Drop", що забезпечує зручність та простоту взаємодії з системою.

– можливість масштабування та безлічевості розміру Бази Даних, що забезпечує гнучкість та масштабованість в зберіганні даних.

– незалежне розпізнавання напряму погляду та особливостей обличчя, таких як носіння окулярів, наявність бороди чи вираз обличчя, що робить систему більш адаптивною та універсальною.

Технологія ідентифікації осіб «VeriLook SDK», розроблена компанією «Neurotechnology», є передовою системою виявлення та розпізнавання облич. Її високу ефективність підтверджує можливість одночасного множинного розпізнавання осіб у кадрі, здатність знаходити до 100 000 осіб за секунду. При цьому, SDK «VeriLook» відмінно працює на різних операційних системах, таких як Windows, Linux, Mac OS X, iOS та Android [10].

Основним принципом роботи алгоритму «VeriLook» є локалізація обличчя, здійснювана за допомогою передових алгоритмів обробки цифрових зображень, заснованих на глибоких нейронних мережах.

Серед ключових переваг системи «VeriLook» варто відзначити:

– можливість одночасної обробки кількох осіб на кадрі;

– гендерну класифікацію, яка дозволяє визначати стать кожної особи на зображенні;

– функцію «живого» розпізнавання обличчя, яка дозволяє відрізняти живі особи від фотографій;

– аналіз емоцій, що дозволяє виявляти шість основних емоцій на обличчі;

– виявлення різних атрибутів особи, таких як посмішка, відкритий рот, закриті очі, окуляри, борода або вуса;

– оцінку якості зображення обличчя, що забезпечує збереження тільки найкращих шаблонів в базі даних;

– можливість використання кількох зразків однієї особи для підвищення точності розпізнавання;

– ідентифікаційну здатність, що дозволяє використовувати функції «VeriLook» для порівняння 1-до-1 або 1-до багатьох;

– невеликий розмір шаблону обличчя, який сприяє ефективності роботи системи;

– особливості режиму узагальнення, що генерує колекцію узагальнених функцій особи з різних зображень.

Алгоритм «VeriLook» також може працювати з межами, захопленими в інфрачервоному спектрі, що розширює його можливості в умовах низького освітлення.

2 Проектування апаратно-програмної частини системи розпізнавання обличчя засобами IoT

2.1 Вибір та налаштування засобу IoT

За основу апаратно-програмної частини системи розпізнавання обличчя засобами IoT обрано модуль ESP32-CAM, що включає в себе, власне, модуль ESP32, а також камеру OV2640 . Модуль стає досить популярним рішенням тому розглянемо нюанси та питання, пов'язані з його використанням. Старт із цією платою можна вважати повноцінним за умови задіяння основних функції – а саме Wi-Fi підключення та отримання відео-потоку з камери.

Рисунок 2.1 – Зовнішній вигляд модуля ESP32-CAM

На платі присутні:

– бездротовий модуль ESP32-S з інтегрованим Wi-Fi та Bluetooth контролерами;

– камера OV2640 йде окремо, але на платі передбачений роз'єм для її підключення;

– гніздо для карт пам'яті micro-SD;

– світлодіод, який, якщо дотримуватися логічних міркувань, мабуть покликаний відігравати роль спалаху.

І оскільки ESP32 не просто реалізує зв'язок по бездротових інтерфейсах, але і є повноцінним контролером сам по собі, то на платі його сигнали виведені на штирьові роз'єми для подальшого використання. ESP32 має:

– процесор з архітектурою 32-біт з можливістю роботи на тактовій частоті 160 або 240 МГц.

– підтримка Wi-Fi зі стандартами 802.11 b/g/n;

– модуль Bluetooth з версією v4.2, який підтримує BR/EDR і BLE.

Велика кількість периферійних модулів, у тому числі:

– SPI x 4;

– АЦП;

– ЦАП x 2;

– UART x 3;

– CAN;

– I2C x 2;

– 520 КБ SRAM.

І це ще далеко не повний перелік. Прошивається ESP32 UART, тому взято для цих цілей USB-UART перехідник на базі PL2303. У цього перехідника відповідні необхідним рівні - 3.3 В.

Для підключення використовуються виводи:

– вивід GPIO1, що відповідає за передачу даних UART (U0TXD), повинен бути підключений до виводу Rx UART;

– вивід GPIO3, призначений для отримання даних UART (U0RXD), має бути підключений до виводу Tx UART.

Додатково, для програмування модуля необхідно з'єднати вивід GPIO0 з нульовим потенціалом. Відповідно, після закінчення прошивки підтяжку треба

прибрати. І ось так у результаті виглядає повне підключення, як показано на рисунку 2.3.

Рисунок 2.2 – Зовнішній вигляд USB-UART перехідника на базі PL2303

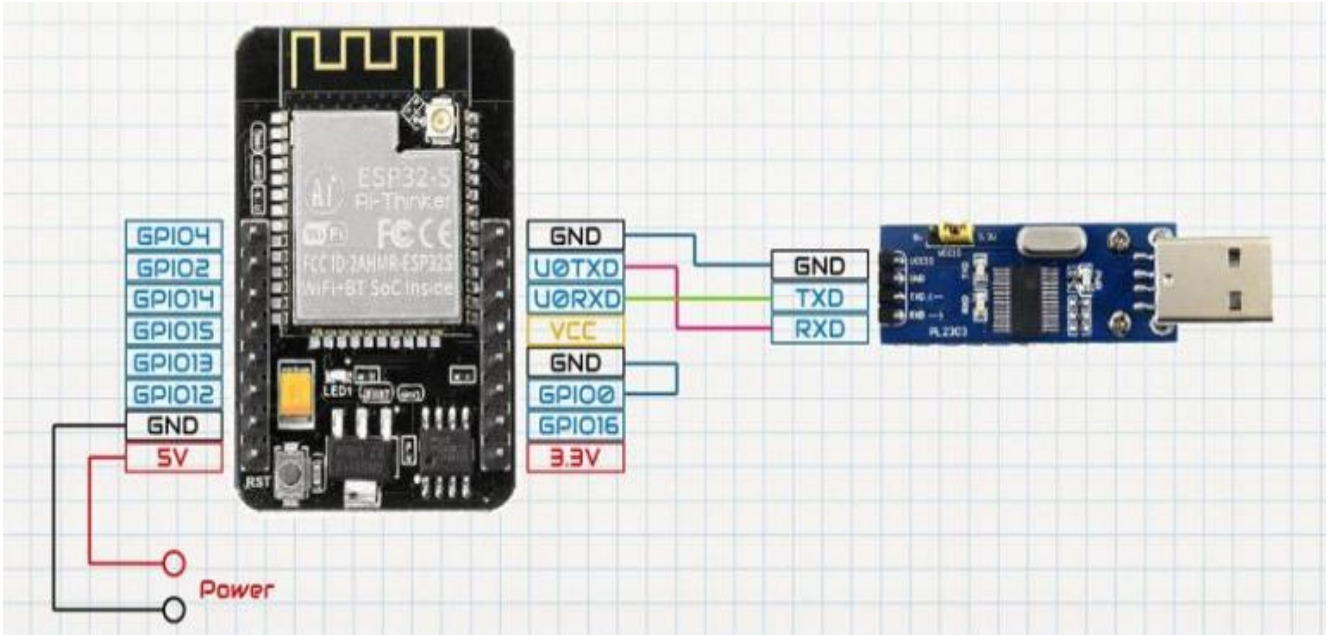

Рисунок 2.3 – Підключення USB-UART перехідника до модуля ESP32-CAM

У випадку підключення живлення до входу 3.3 В спостерігалося багато різноманітних проблем – прошивалося через раз, працювати модуль відмовлявся в принципі і т. д. Щоб усунути проблему разом необхідно здійснити подачу 5 В живлення через відповідний вхід (як на схемі вище).

Але при цьому напруга логічних рівнів на входах – 3.3 В! Сигнали 5 можуть пошкодити контролер.

На цьому із фізичним підключенням закінчуємо.

Отже, для швидкого запуску ESP32-CAM використовуємо існуючі приклади та бібліотеки для середовища Arduino IDE (див. рис. 2.4) . Щоб почати користувантись Arduino IDE, то переходимо до Кроку 0.

Крок 0

Установка Arduino IDE. З цим все просто – завантажуємо з офіційного сайту , встановлюємо та запускаємо.

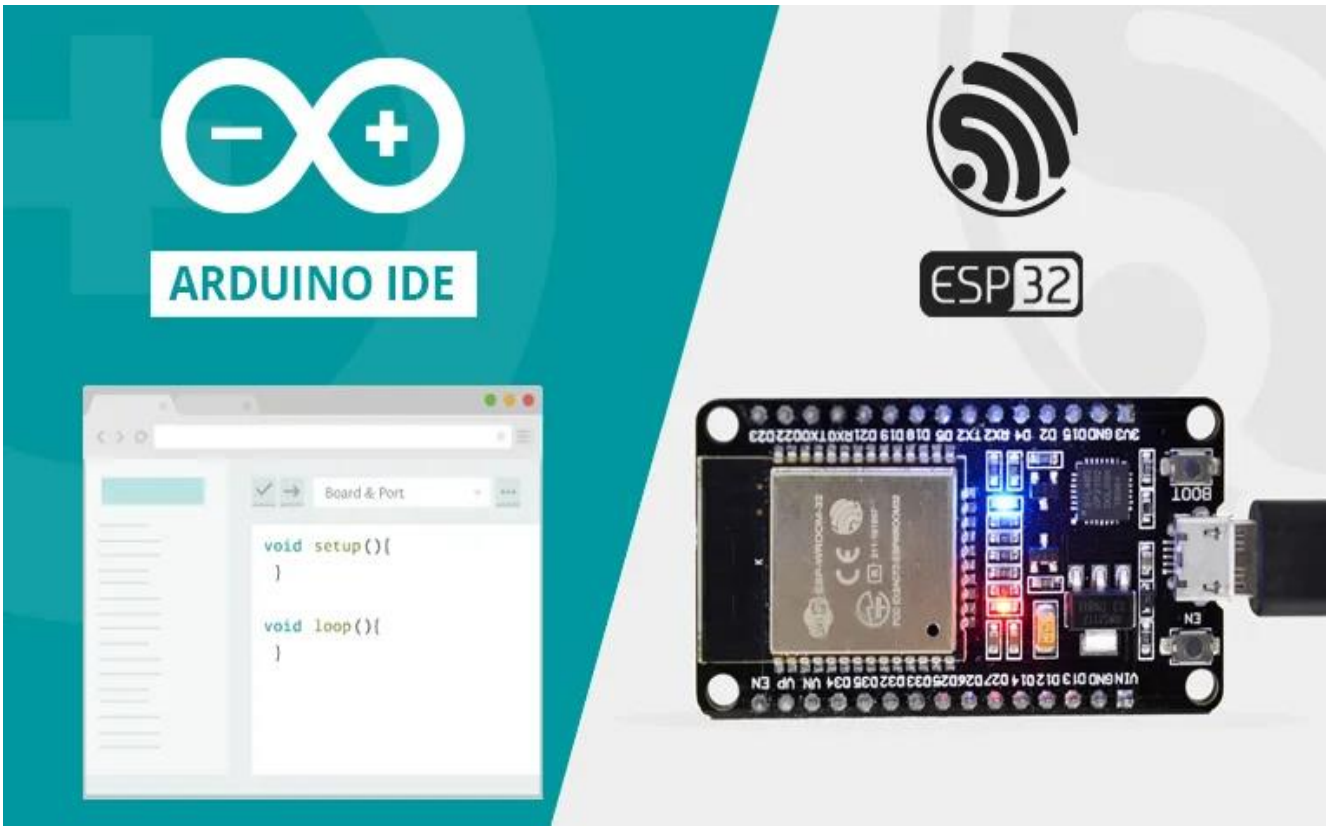

Рисунок 2.4 – Середовище Arduino IDE

Крок 1

Тепер у нову IDE необхідно інтегрувати підтримку нашого модуля ESP32. Йдемо в налаштування (File - Preferences) і в нижній частині вікна налаштувань Additional Boards Manager URLs додамо стрічку:

`https: //dl.espressif.com/dl/package_esp32_index.json`

На цьому закриваємо вікно налаштувань та переходимо до Кроку 2.

Крок 2

Тепер перейдіть до Менеджера плат: Інструменти - Плата - Менеджер плат, як показано на зображенні 2.5.

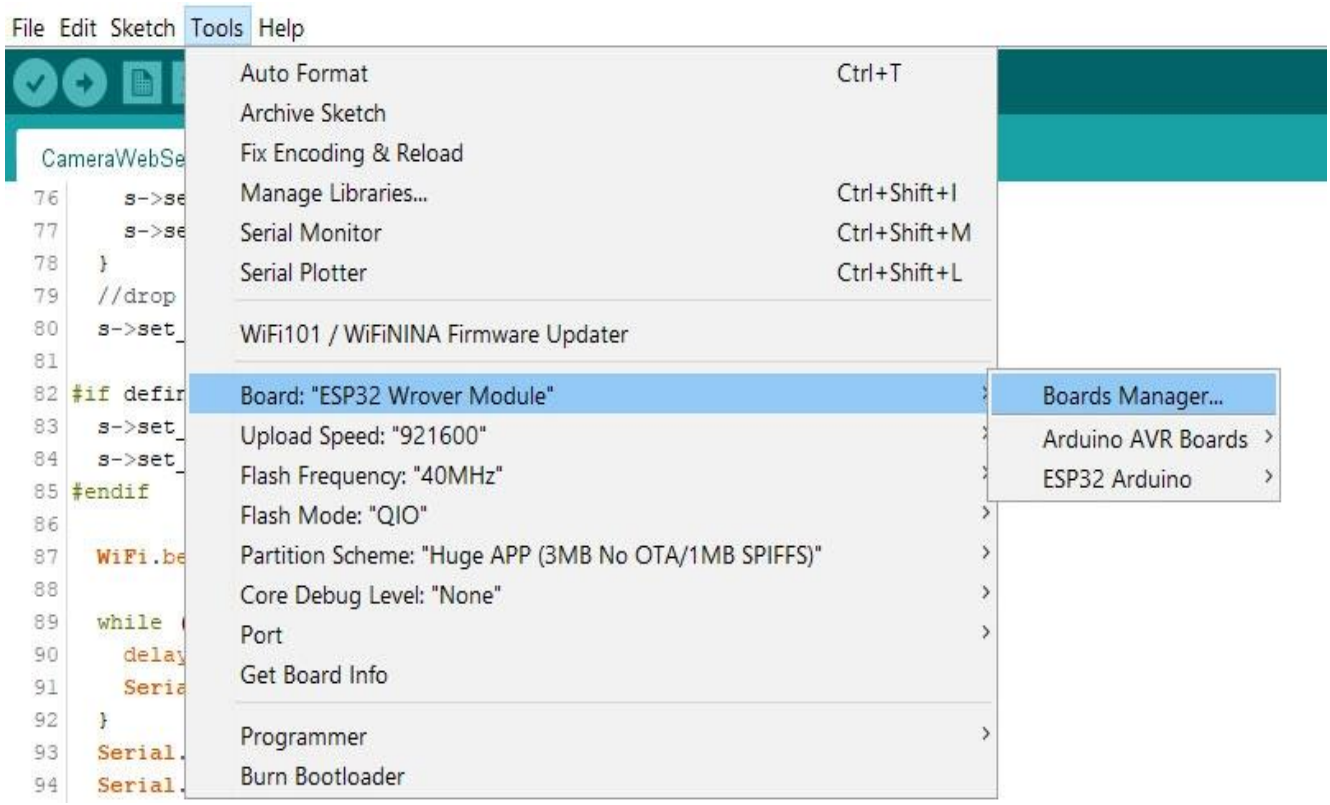

Рисунок 2.5 – Запуск Boards Manager

У вікні, що відкрилося, в пошуку вводимо «esp32», отримуємо один варіант і натискаємо на Install (див. рис. 2.6).

Очікуємо завершення операції та переходимо до наступного етапу. Тільки попередньо в Tools-Board-ESP32 Arduino вибираємо плату, що використовується. Розробники IDE періодично перетасовують і змінюють список доступних плат, підсумкова задача проте проста - потрібно знайти у списку свою і вибрати її.

Крок 3

Відкриваємо у Arduino IDE приклад для роботи з камерою. Для цього слідуємо в File-Examples-ESP32-Camera-CameraWebServer. Отримуємо готовий проект з ініціалізацією функції setup() і основним циклом програми функції loop().

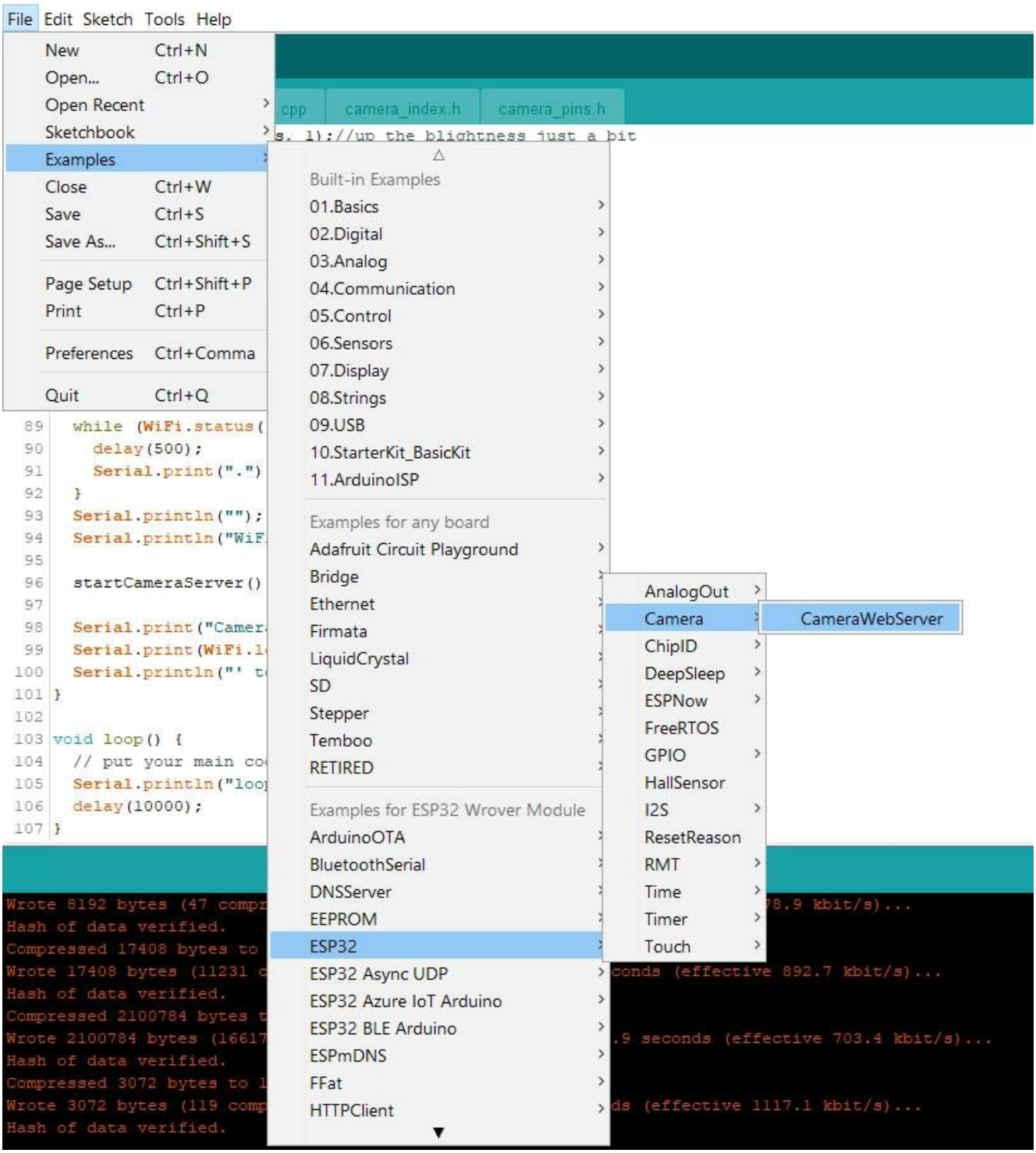

Рисунок 2.6 – Вибір CameraWebServer

Крок 4

Тепер необхідно трохи змінити код для того, щоб плата та камера успішно запрацювала. Насамперед вибираємо модель камери. Здійснюється це коментуванням непотрібного та розкоментуванням потрібного(див. рис.2.7).

```
// Виберіть модель камери
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER
```

Рисунок 2.7 – Частина лістингу із моделями камер

Друга зміна полягає в установці імені мережі та пароля для підключення Wi-Fi. Наприклад:

```
const char * ssid = "Домашня сторінка" ;
const char * password = "pass" ;
```

На цьому з кодом закінчуємо та переходимо до прошивки.

Крок 5.

Підключення вже здійснили, тому на цьому етапі суто програмні моменти. У меню Tools нам потрібно задати актуальні параметри для програмування плати, зокрема номер COM-порту (плату вже вибрали на Кроку 2). Список налаштувань для плати показано на рисунку 2.8.

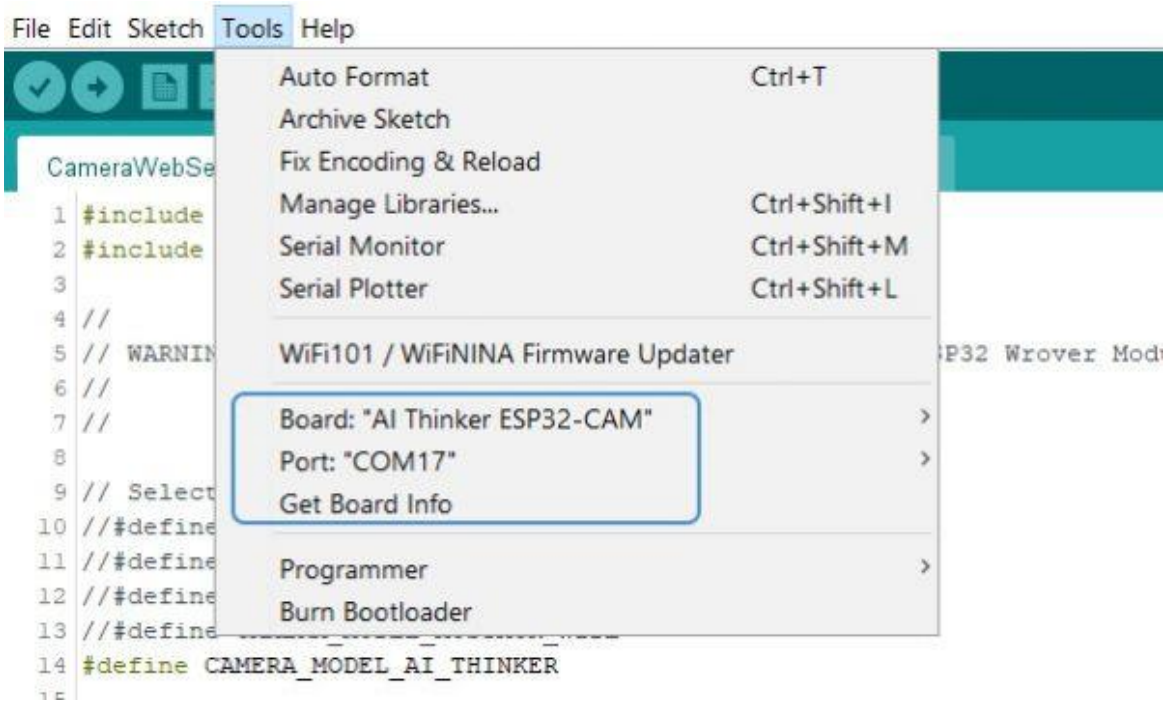

Рисунок 2.8 – Список налаштувань для плати ESP32-CAM

Крок 6

Для завантаження програмного забезпечення на плату натисніть кнопку «Завантажити» (upload), але перед цим не забудьте підтягнути вивід GPIO0 до землі. (див. рис. 2.9). Лістинг основного коду наведено в додатку Б.Лог успішної прошивки показано на рисунку 2.10.

Крок 7

Відключаємо підтяжку GPIO0 і запускаємо Serial Monitor (Tools-Serial monitor) (див. рис. 2.11).

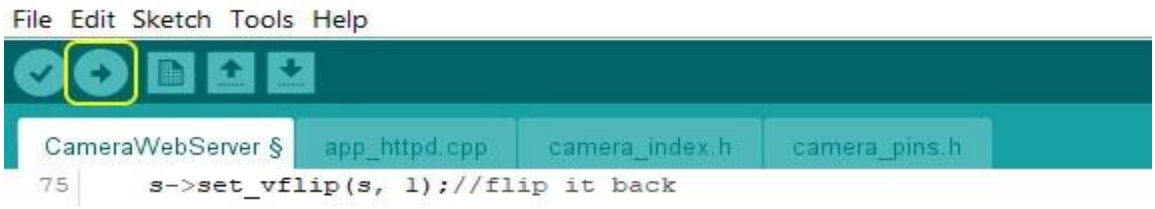

Рисунок 2.9 – Прошивка плати ESP32-CAM

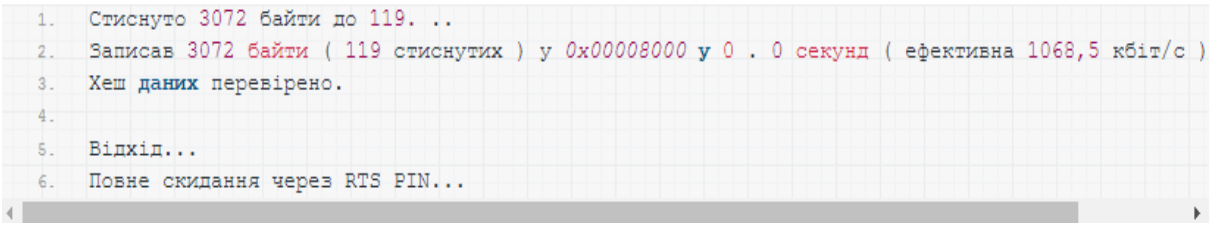

Рисунок 2.10 – Завершальна частина успішного лога

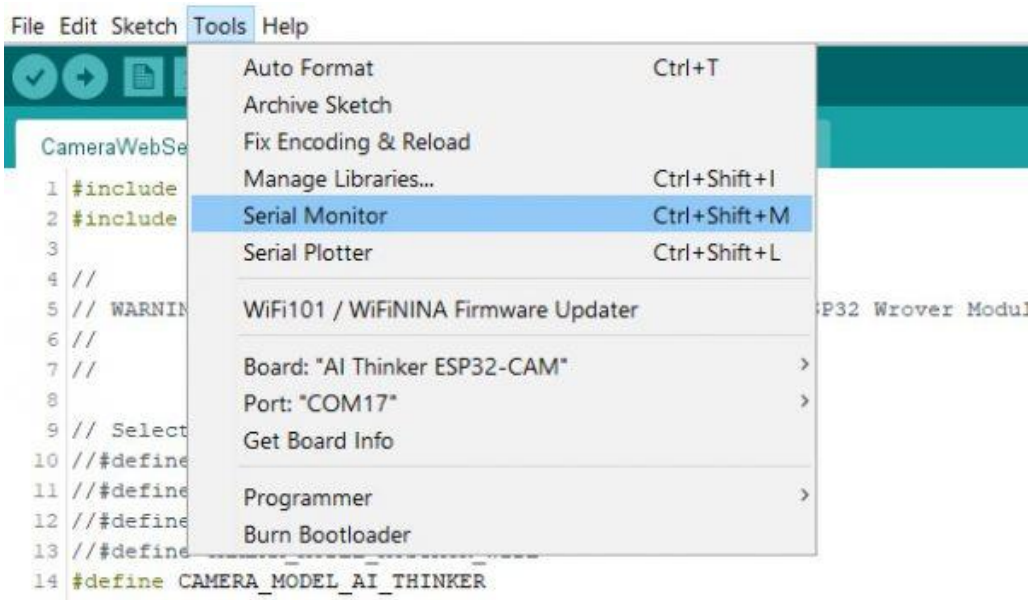

Рисунок 2.11 – Прошивка плати ESP32-CAM

Тепер можемо перезапустити плату кнопкою Reset і на екрані з'являться рядки, що сигналізують про успішне підключення до мережі Wi-Fi:

WiFi підключено

Запуск веб-сервера на порту: '80'

Запуск потокового сервера на порту: '81'

Камера готова! Для підключення використовуйте «http://192.168.0.104».

Крок 8

Власне, залишається тільки перейти в браузері за вказаною IP-адресою, щоб побачити результат роботи програми. Далі знаходимо внизу кнопку Start Stream і отримуємо зображення з OV2640 (див. рис. 2.12-2.13).

Можна потестувати наявні параметри камери, зокрема цікаві режими розпізнавання обличчя.

Включаємо Face Detection та Face Recognition і далі кнопкою Enroll Face можна зареєструвати людину. І згодом камера визначить, чи знаходиться в кадрі зареєстрована особа чи якась інша.

Рисунок 2.12 – Вікно успішного підключення до OV2640

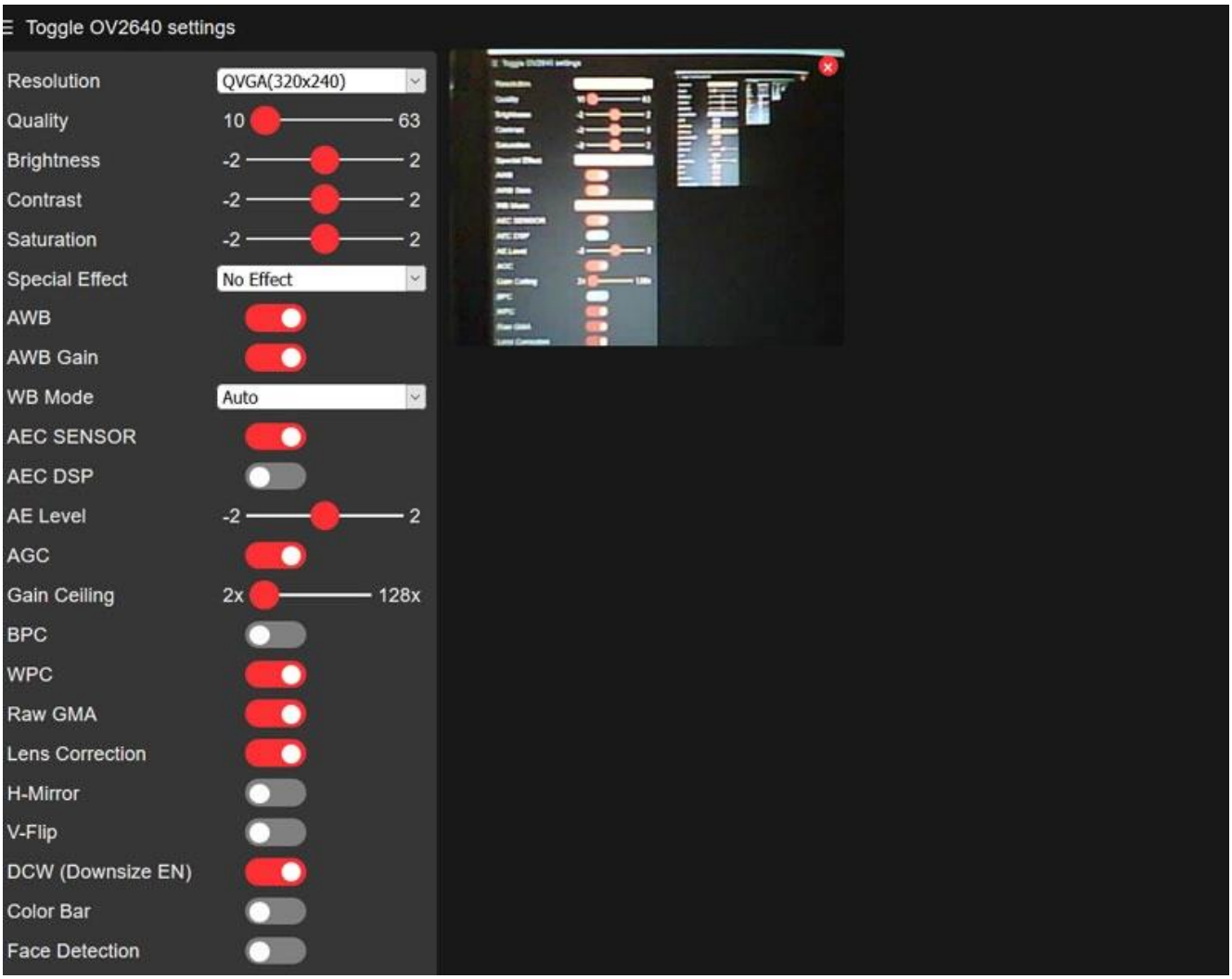

Рисунок 2.13 – Вікно налаштувань та зображення з OV2640

2.2 Вибір додаткових електронних компонентів

Головним компонентом системи є модуль ESP32-CAM зкрихітною платою з камерою, слотом для карт microSD і мікроконтролером ESP32. Також вона має можливості використання технологій Wi-Fi та Bluetooth, що робить її відмінним вибором для багатьох проектів тематики інтернету речей (IoT). Але робота з модулем ESP32-CAM трохи ускладнена тим, щоб завантажити в нього програми необхідно використовувати зовнішню плату для програмування (FTDI плату). Ці питання розглянуто в попередньому підрозділі.

Щоб спроектувати системи в цілому на основі модуля ESP32-CAM необхідні додаткові компоненти. Також для живлення нашої схеми розробимо

інтегроване джерело живлення з напругою 5 і 3.3 В, яке буде запитуватись від акумулятора 18650. Також конструкція нашого проекту міститиме роз'єми для підключення плати FTDI з можливістю перемикання напруги між 5 і 3.3 В. буде містити і безліч інших опцій, які можуть стати вам у нагоді для розробки проектів інтернету речей, що запитуються від батарейок/акумуляторів.

Отже, конструкція нашого проекту забезпечуватиме:

- доступ до контактів модуля ESP32-CAM за допомогою конекторів типу «тато» та «мама»;
- кілька контактів для напруг 5 і 3.3, а також землі (Ground);
- світлодіоди для індикації подачі живлення та підключення до мережі Інтернет;
- живлення від акумулятора 18650 з можливістю його заряджання та схеми перетворювача напруги;
- можливість закріплення конструкції проекту на стіну;
- можливість використання у різних проектах для розпізнавання обличчя та виявлення руху;
- джампер IO0 для режиму програмування.

У системі контролю доступу на основі модуля ESP32-CAM для розпізнавання облич ми будемо використовувати код CameraWebServer. При успішному розпізнаванні особи його ідентифікатор (face ID) з ім'ям користувача, асоційованим з ним, передаватиметься до документів Google (Google Sheets).

Необхідні компоненти наступні:

1. Власне модуль ESP32-CAM .
2. Регулятор напруги 3.3V AMS1117.
3. Мікросхема заряду Li-Ion батарей TP4056.
4. Мікросхема конвертера постійного струму (Boost Converter IC) FP6291.
5. 5-контактний роз'єм Micro USB 2.0 типу B.
6. Резистори 100 Ом, 1 кОм (2 шт.), 1,2 кОм, 6 кОм, 48 кОм, 51 кОм.
7. Конденсатори 0,1 мкФ (2 шт.), 10 мкФ (4 шт.), 20 мкФ (2 шт.).

8. Світлодіод – 4 шт.
9. Котушка індуктивності 4.7µH.
10. Діод 1N5388BRLG
11. Акумулятор літієвий 18650.

2.3 Розробка електричних принципових схеми

2.3.1 Основна схема

Схема мікропроцесорної системи розпізнавання обличчя засобами IoT на основі модуля ESP32-CAM представлена на рисунку 2.14.

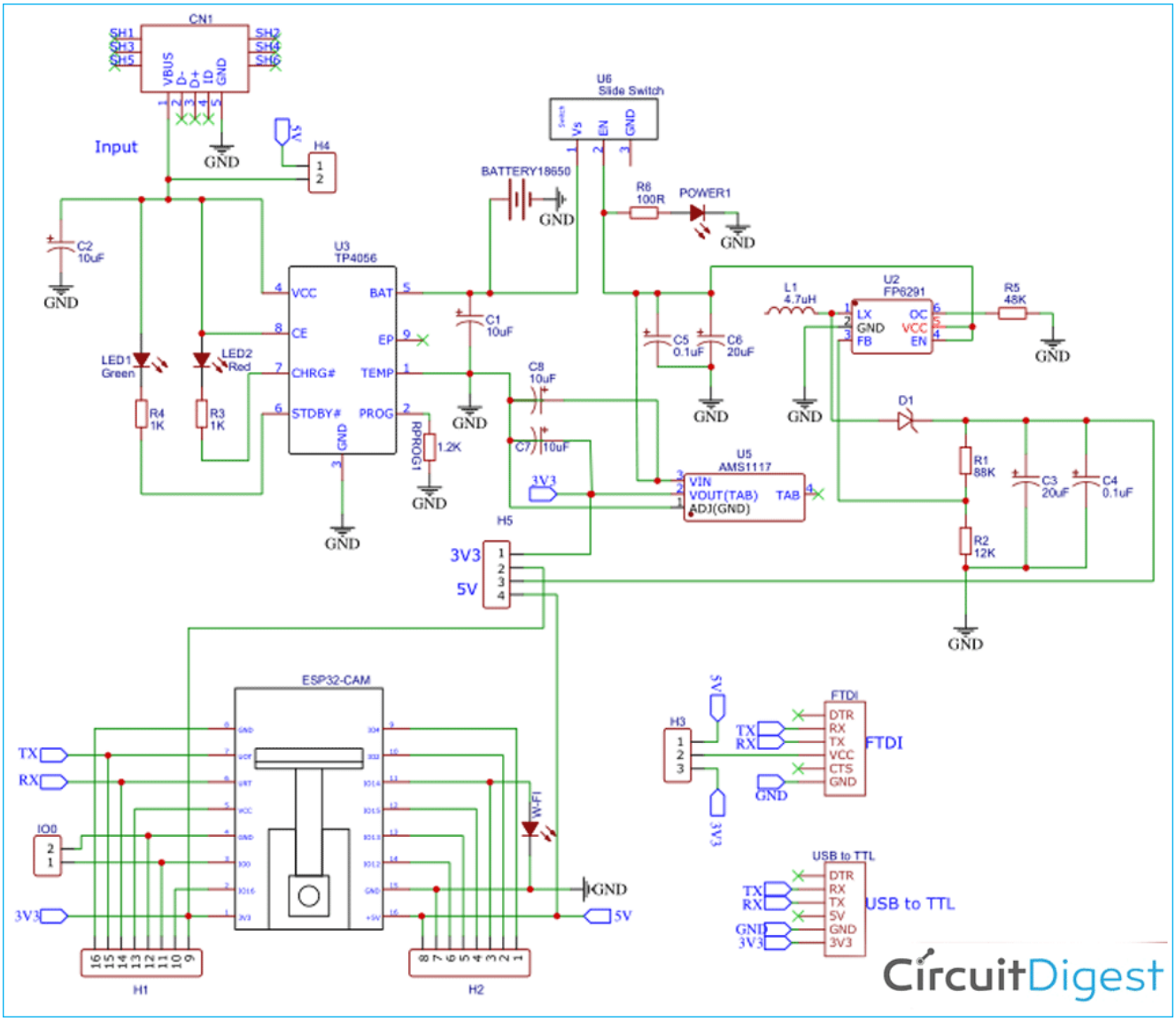

Рисунок 2.14 – Основна схема мікропроцесорної системи розпізнавання обличчя

Подана схема містить 3 основні частини:

- схема заряду батареї;
- схема конвертера постійного струму (DC to DC boost converter circuit);
- схема програмування модуля ESP32-CAM.

Схема конвертера постійного струму використовується перетворення напруги з 3.7 В до 4.5-6 В. На зарядному боці цієї схеми розташований 5-піновий конектор Micro USB 2.0 B типу.

2.3.2 Схема заряду батареї

Схема заряду батареї показано на рисунку 2.15.

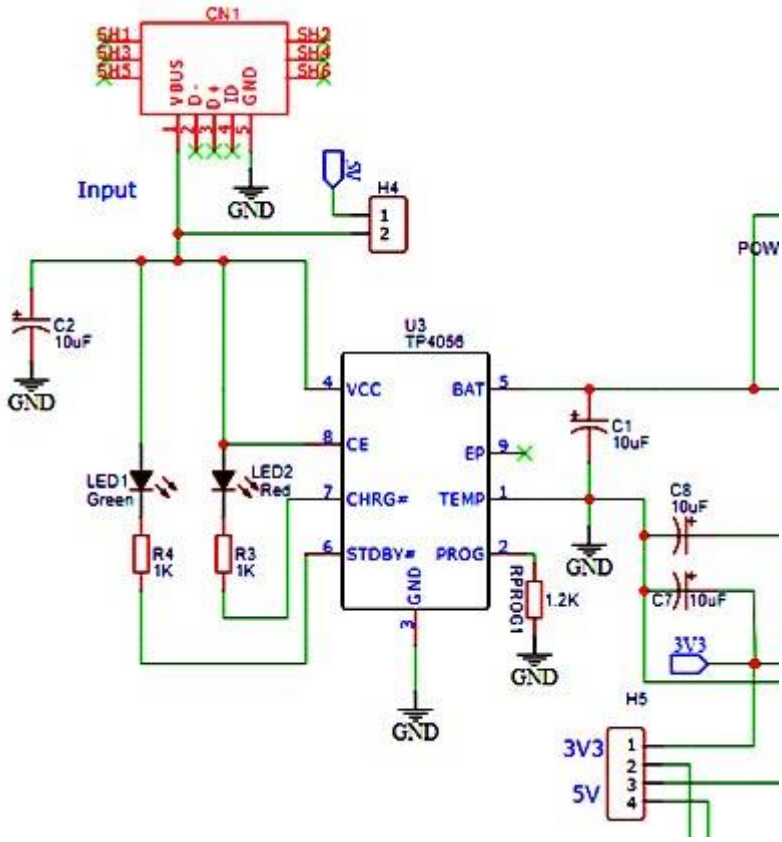

Рисунок 2.15 – Схема заряду батареї

Основою цієї схеми є мікросхема заряду літій-іонних батарей TP4056. З її допомогою можна заряджати поодинокі літій-іонні батареї в режимі постійного струму/постійної напруги. Оскільки вона упакована в SOP корпус і вимагає малої кількості зовнішніх компонентів для своєї роботи, вона є відмінним рішенням для

використання в пристроях, що носяться. Мікросхеми виконує операції заряду, обробляючи напругу постійного струму 5 В, що отримується через роз'єм Micro USB. Світлодіоди у цій схемі показують статус заряду.

2.3.3 Схема конвертера постійного струму

Схема конвертера постійного струму показано на рисунку 2.16.

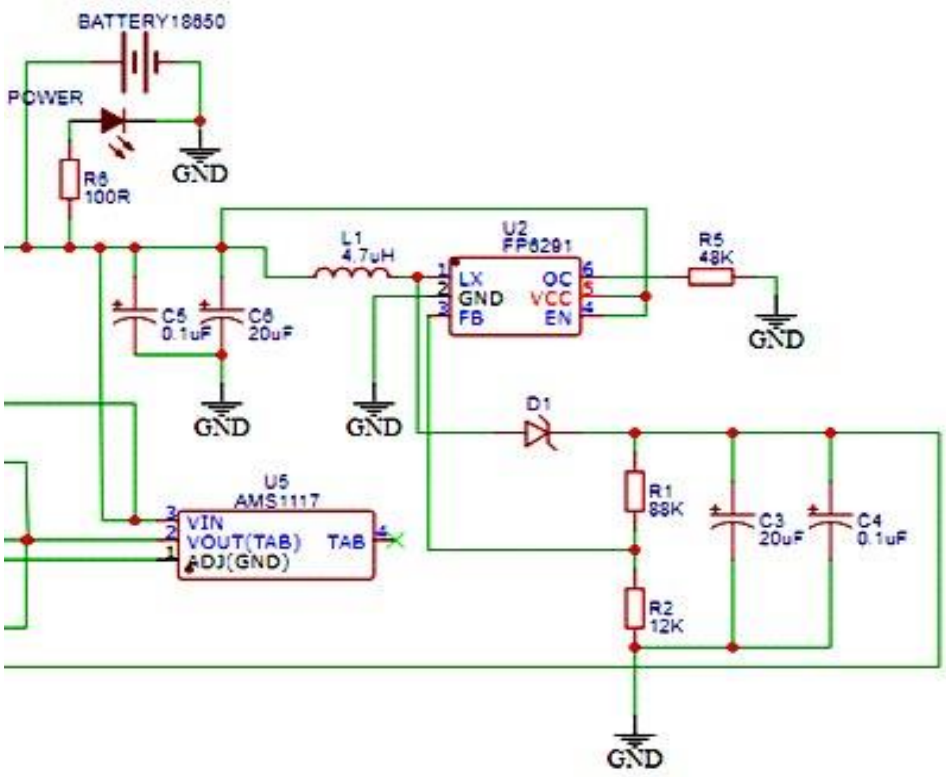

Рисунок 2.16 – Схема конвертера постійного струму

Основою даної схеми є мікросхема FP6291, яка працює на частоті 1 МГц і використовується в багатьох конвертерах подібного типу, наприклад, для отримання стабільної 5 В напруги з напруги 3 В, що отримується з батареї. У нашому випадку на вхід цієї схеми надходить напруга з контактів батареї (+ та -), яка обробляється мікросхемою FP6291, на виході якої ми отримуємо стабільні 5 В постійного струму через стандартний роз'єм USB.

Дана схема складається з модуля ESP32-CAM з одним слотом для плати FTDI та іншим слотом для конвертера USB TTL.

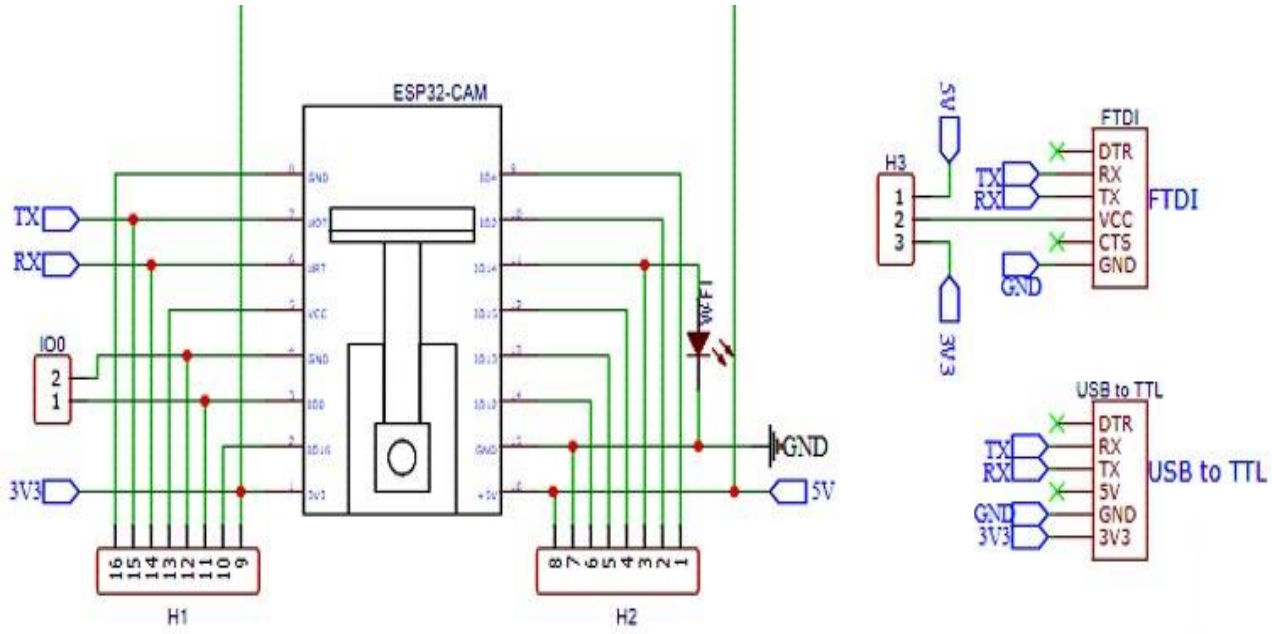

Рисунок 2.17 – Схема програмування модуля ESP32-CAM

2.3.4 Друкована плата

Розроблену 3D модель друкованої плати для нашого проекту показано на наступному рисунку 2.18.

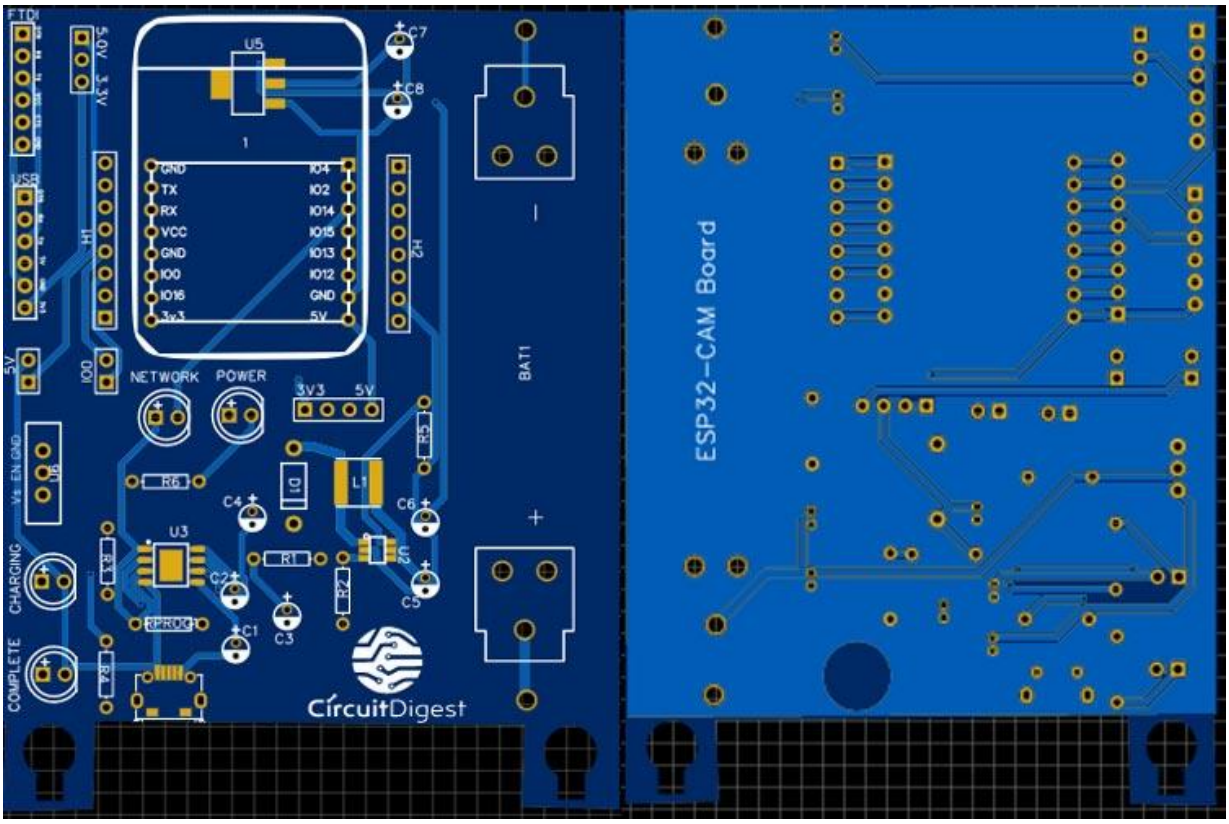

Рисунок 2.18 – Друкована плата

2.4 Конфігурація сервісу IFTTT для передачі даних у Google Sheets

Дані, які приймаються з модуля ESP32-CAM, будемо передавати в документи/аркуші Google (Google Sheets). Для цього використовуватимемо аплети (applets) IFTTT. Ці аплети будуть отримувати дані з модуля ESP32-CAM за допомогою веб-хуків (webhooks) та потім передавати їх до документів Google.

Спочатку потрібно зареєструйтесь, якщо у вас там ще немає облікового запису. Потім увійти до свого облікового запису на сервісі IFTTT і виконати пошук 'Webhooks'.

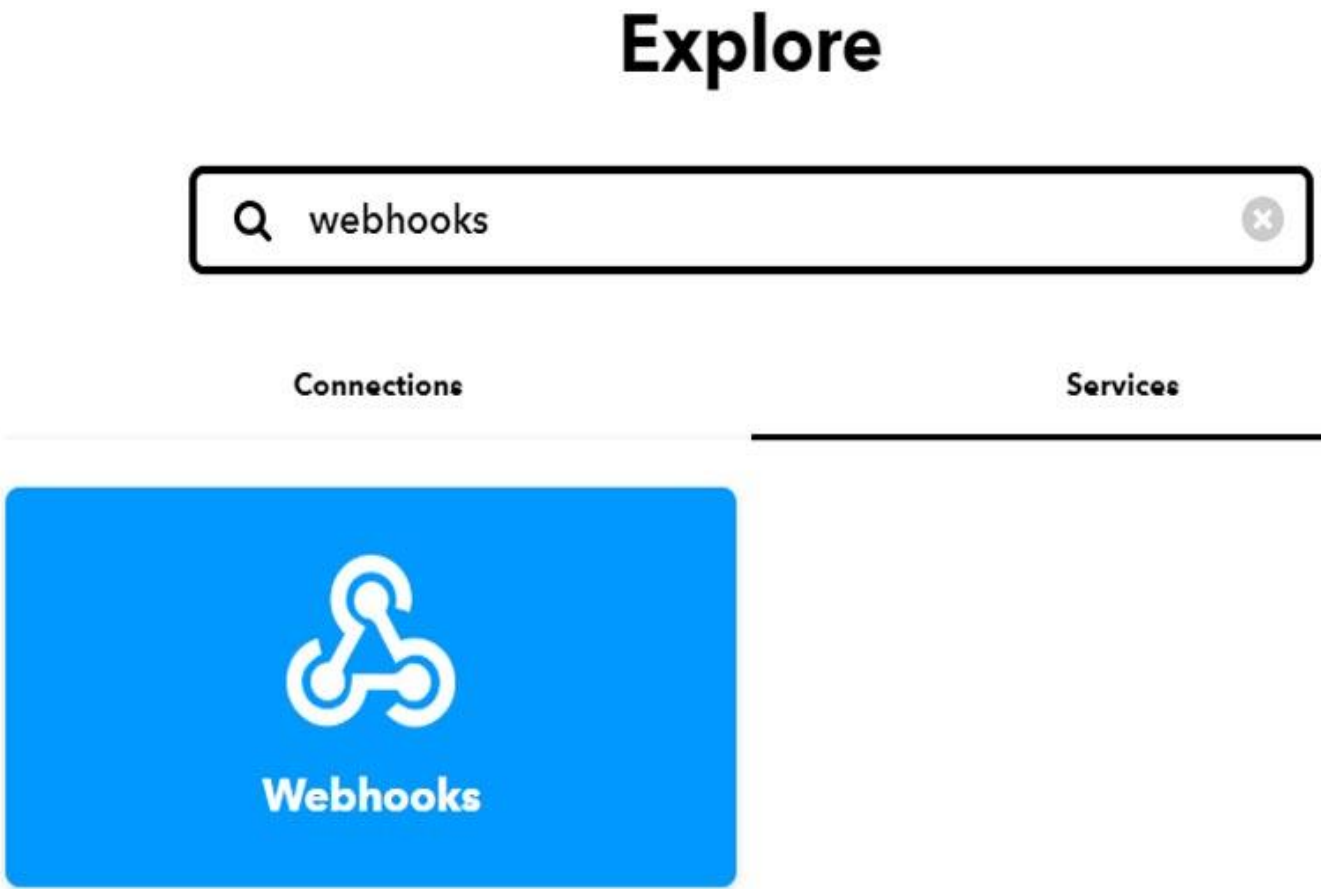

Рисунок 2.19 – Пошук 'Webhooks'на сервісі IFTTT

Далі, щоб отримати приватний ключ, натисніть 'Documentation'. Скопіюйте цей ключ кудись у надійне місце, надалі він нам знадобиться в коді програми.

Рисунок 2.20 – Отримання приватного ключа

Після отримання приватного ключа можемо створити аплет, використовуючи веб-хуки (Webhooks) та документи Google. Щоб створити аплет, натисніть кнопку 'Create' у верхньому правому куті екрана.

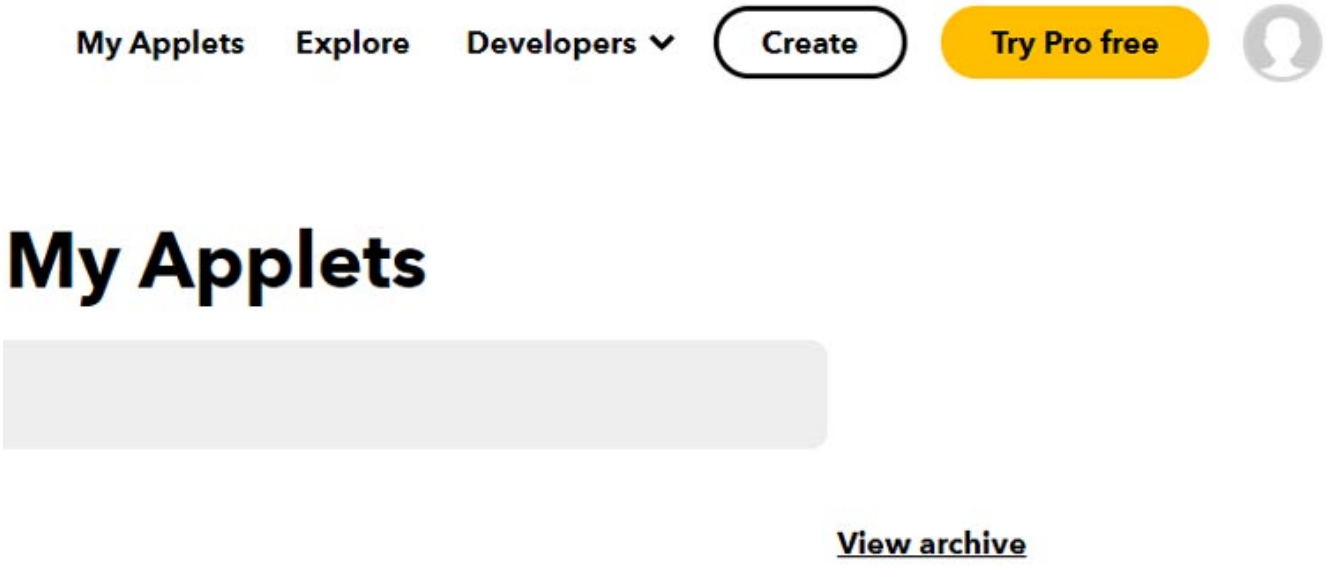

Рисунок 2.21 – Створення атлета

У наступному вікні виберіть "If This Then That".

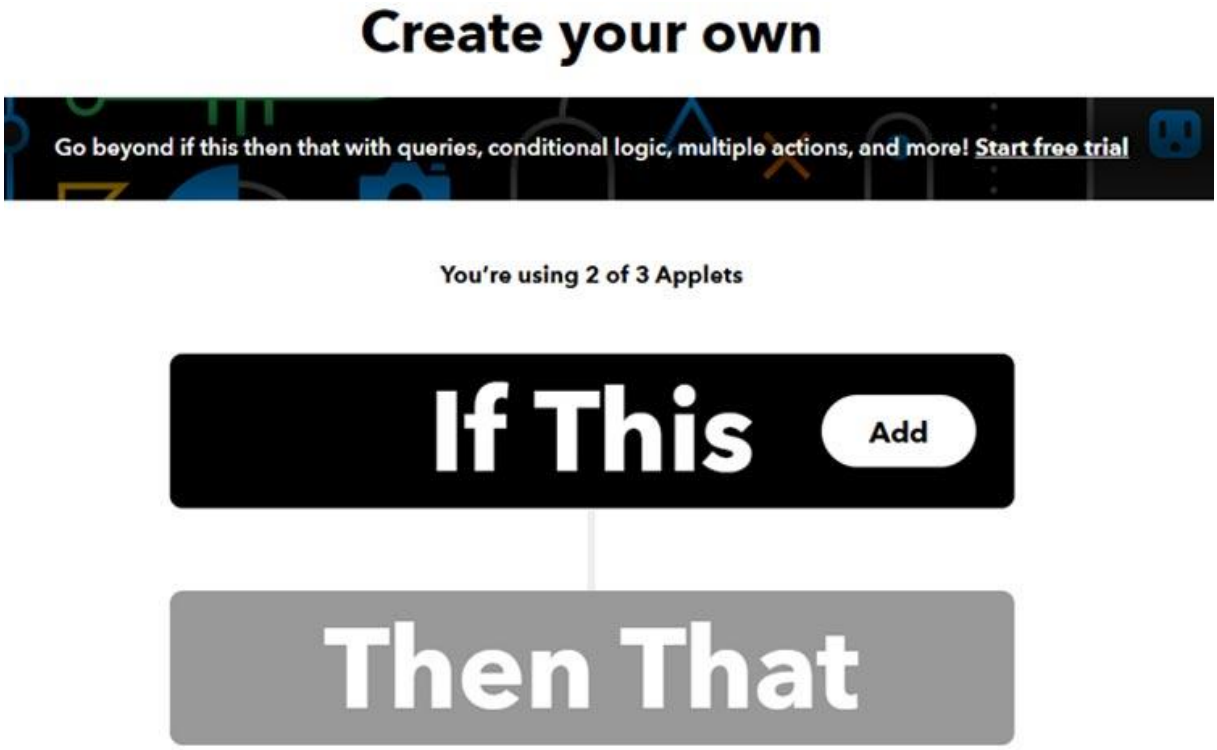

Рисунок 2.22 –Створення свого «If This Then That»

У полі 'This' будемо використовувати веб-хуки для отримання веб-запитів від модуля ESP32-CAM.

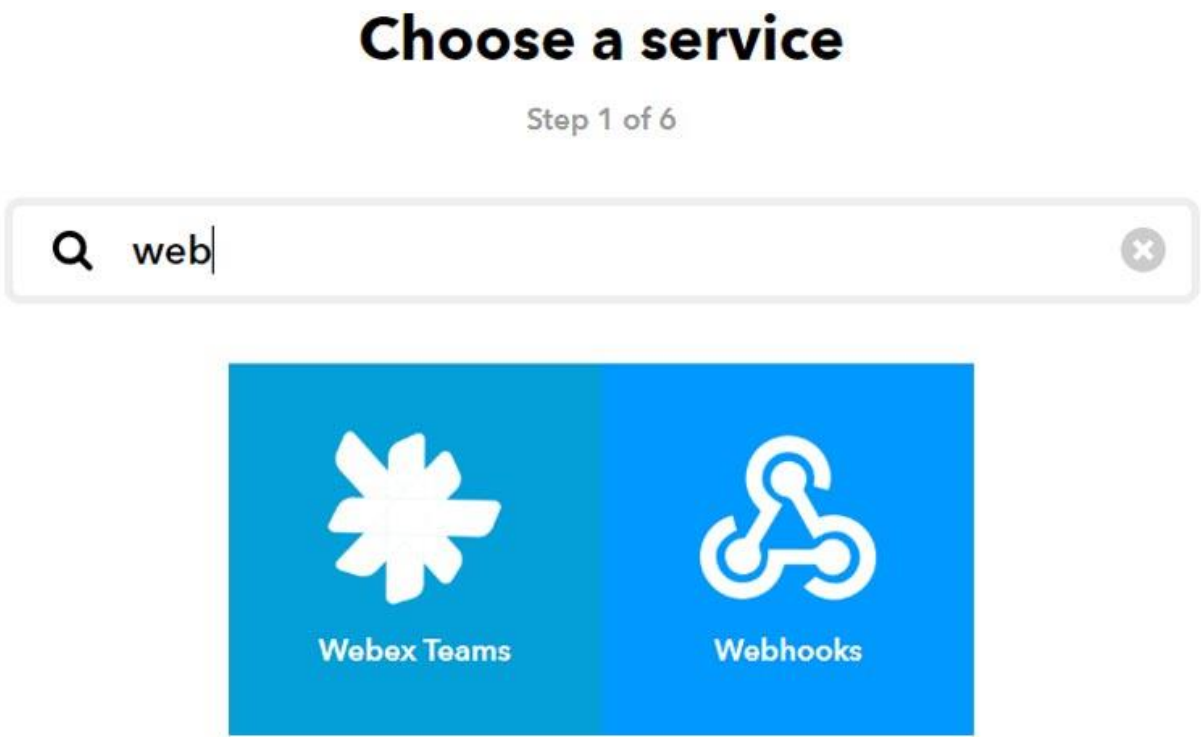

Рисунок 2.23 – Вибір вебхуків

Виберіть тригер 'Receive a Web Request', потім введіть як ім'я події "attendance", а потім натисніть на 'Create Trigger.' (Створити тригер).

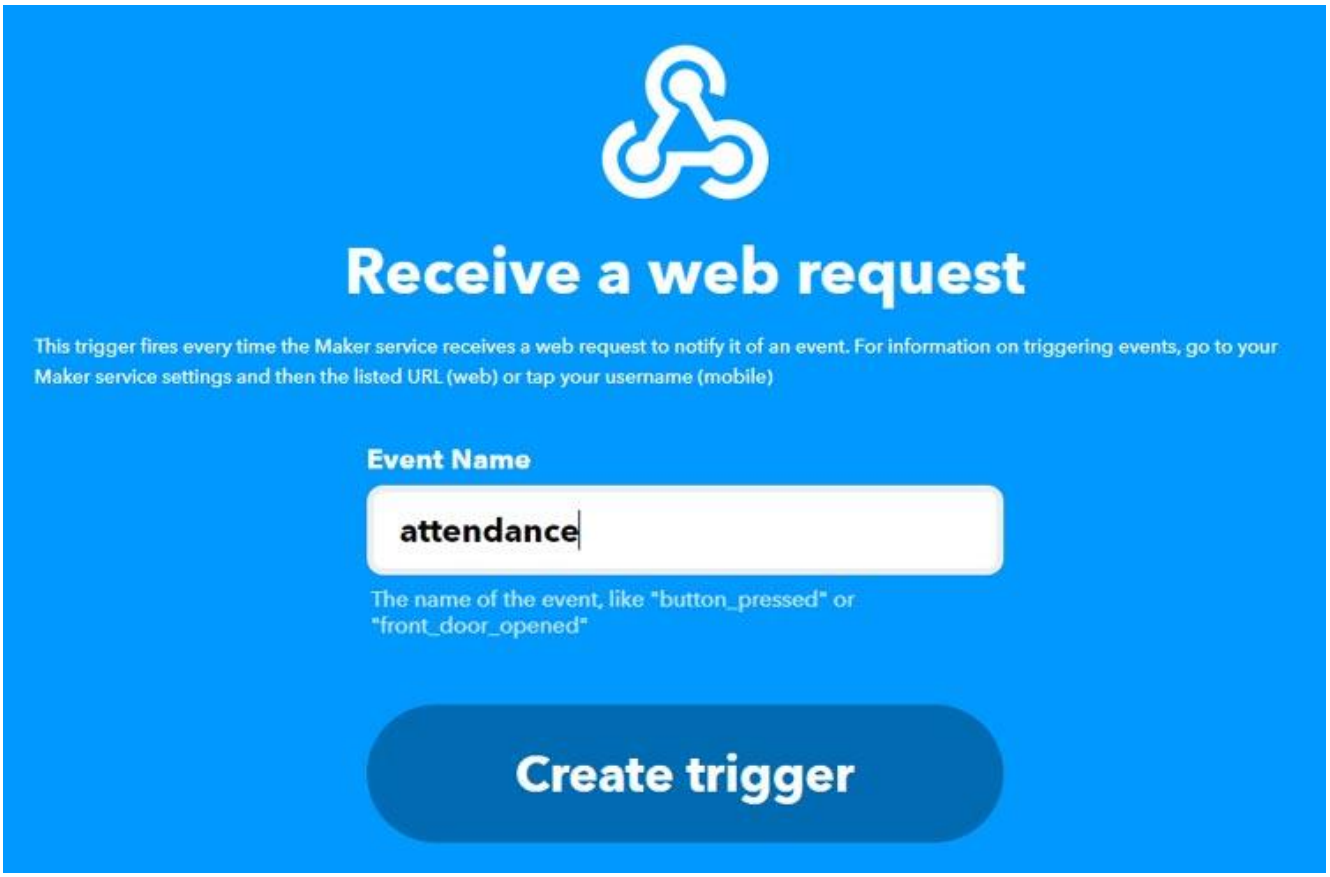

Рисунок 2.24 – Створення тригера

Після цього натисніть на 'Then That', виконайте пошук 'Google Sheets' та виберіть піктограму документа Google.

На наступному кроці виберіть дію “Add a row to spreadsheet” (додати рядок на аркуш).

Після цього заповніть поля створюваної дії. Дайте ім'я аркушу, вкажіть шлях до каталогу Google Drive та залиште у полі "Formatted row" значення за промовчанням. Якщо залишите поле шляху до каталогу Google Drive порожнім, то сервіс IFTTT зберігатиме ваш лист у каталог з ім'ям "IFTTT" у вашому каталозі Google Drive. І, нарешті, виберіть “Create action” (створити дію) з меню, що випадає.

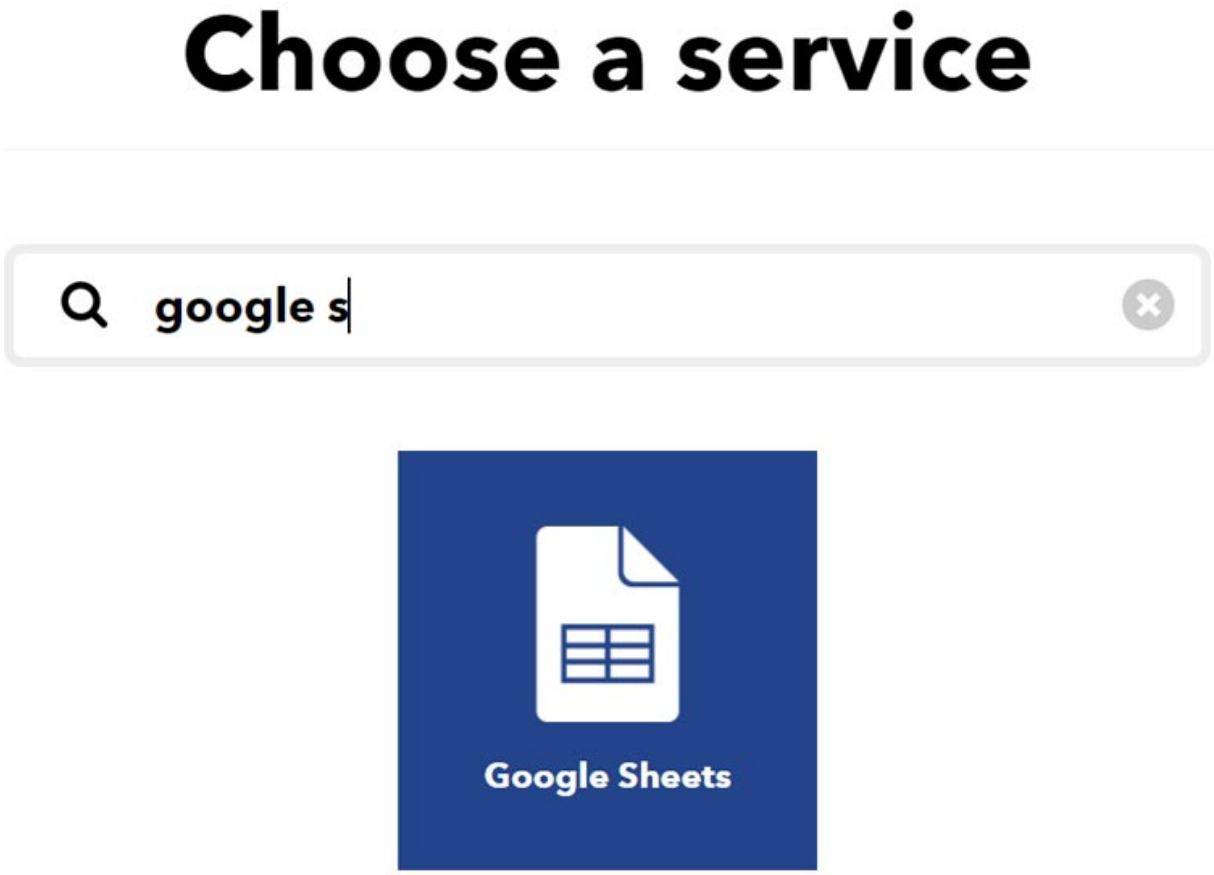

Рисунок 2.25 – Пошук 'Google Sheets'

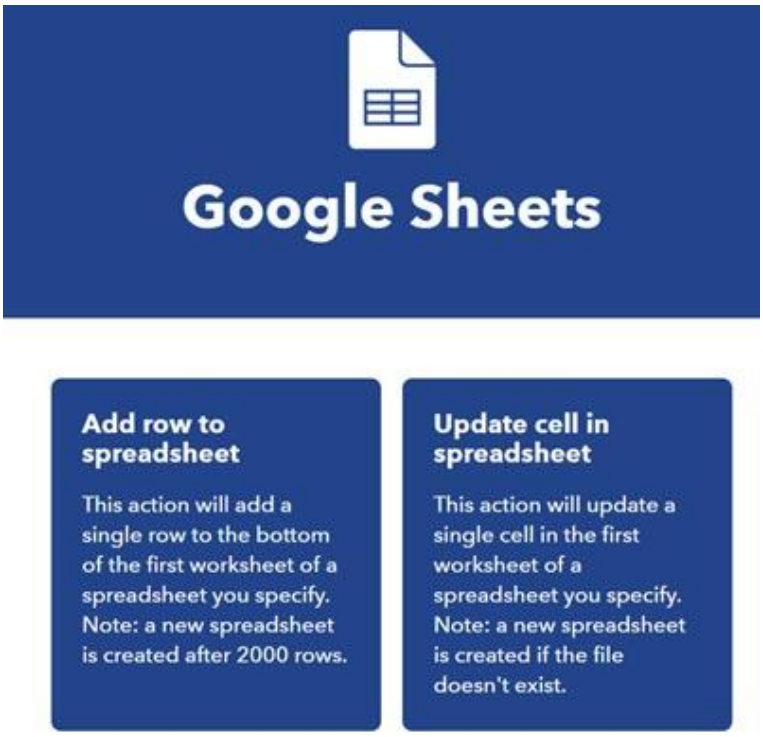

Рисунок 2.26 – Додавання рядка на аркуш

Рисунок 2.27 – Вибір “Create action”

2.5 Тестування роботи системи розпізнаванням обличчя

Насамперед у програмі підключимо використовувані бібліотеки. У нашому випадку бібліотека "esp_camera.h" використовується для ініціалізації камери з метою подальшого розпізнавання облич, а бібліотека "WiFi.h" використовується для підключення модуля ESP32-CAM до мережі Wi-Fi.

```
#include "esp_camera.h"
#include <WiFi.h>
```

Далі вкажемо параметри для доступу до мережі WiFi – її ім'я (ідентифікатор) та пароль.

```
const char* ssid = "*********";
const char* password = "*********";
```

Потім нам необхідно вказати унікальну URL-адресу з сервісу IFTTT. Для цього увійдіть у свій обліковий запис на сервісі IFTTT, у розділ "webhooks documentation" і скопіюйте посилання, яке наведене в розділі "triggering an event". Після цього вставте це посилання в код програми.

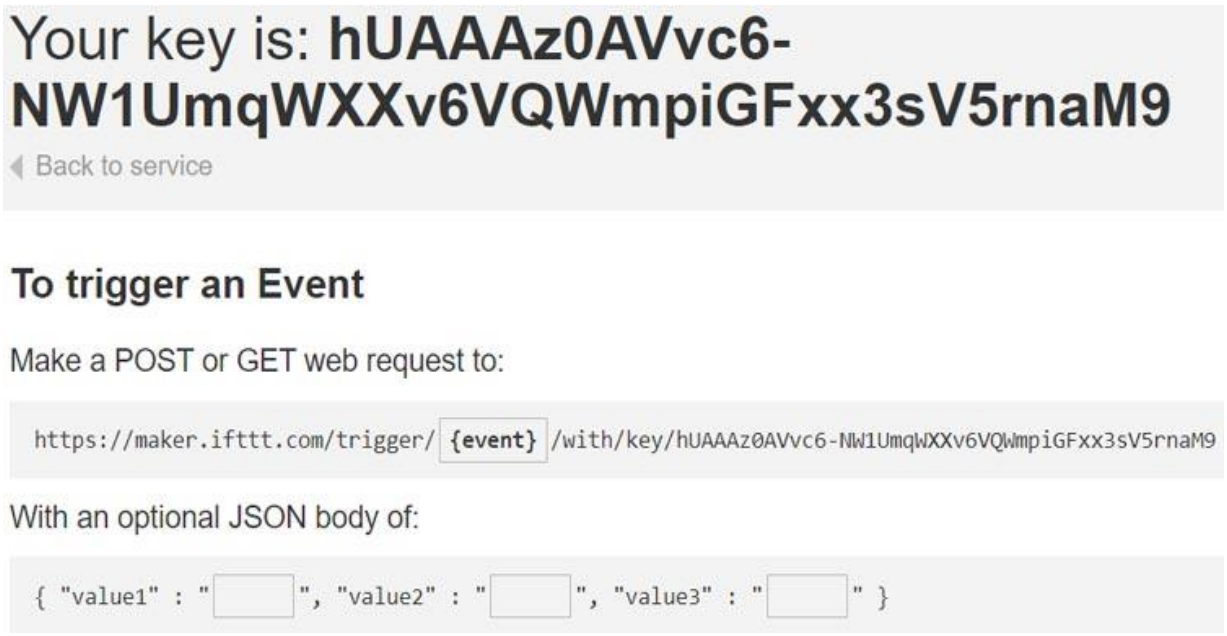

Рисунок 2.29 – Внесення даних в сервіс IFTTT

Не будемо вносити жодних змін до функції void setup(), тому залишимо її за замовчуванням. На наступному кроці будемо отримувати ідентифікатори обличчя (face ids) з веб-сторінки та оновлювати аркуш доступу відповідно до отриманих даних. Призначатимемо різні імена відповідно до отриманих ідентифікаторів (IDs) і передаватимемо ці імена до документів google (google sheets).

```
Serial.print(FACE_ID);
  if (FACE_ID == 0){
     student = "Name 1";
     makeIFTTTRequest();
     delay(10000);
  }
  if (FACE_ID == 1){
     student = "Name 2";
     makeIFTTTRequest();
     delay(10000);
  }
……..
}
```

Щоб завантажити код програми в модуль ESP32-CAM, підключіть до нього FDTI/USB to TTL і в налаштуваннях Arduino IDE виберіть 'ESP32 Wrover Module' як тип вашої плати. Змініть ряд інших налаштувань, як показано на наступному рисунку.

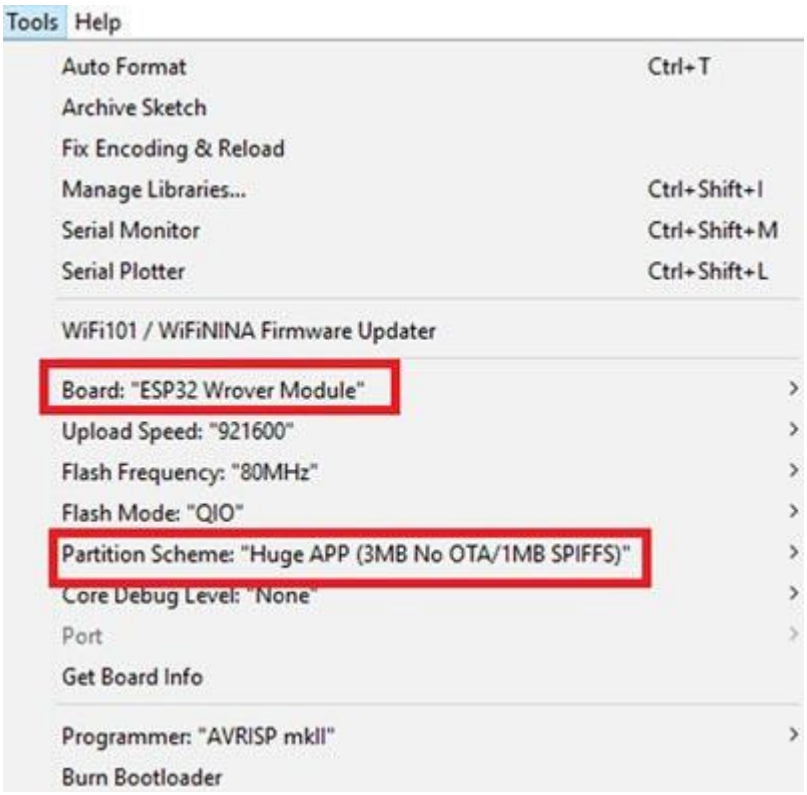

Рисунок 2.30 – Налаштування Arduino IDE

Не забудьте замкнути контакт IO0 модуля на землю (GND) перед завантаженням програмного коду. Також натисніть на модулі ESP32 кнопку скидання і після цього натискайте кнопку завантаження програми Arduino IDE.

Після завантаження коду програми вимкніть контакт IO0 модуля від землі (GND). Потім відкрийте вікно монітора послідовного зв'язку (serial monitor) і встановіть у ньому швидкість 115200. Після цього натисніть кнопку скидання на модулі ESP32, внаслідок чого у вікно монітора послідовного зв'язку буде виведено IP адресу модуля та номер порту, до якого він підключений.

```
rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:9720
ho 0 tail 12 room 4
load:0x40080400,len:6352
entry 0x400806b8

.
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.43.226' to connect
```

Рисунок 2.31 – Вікно монітора послідовного зв'язку

Після цього введіть певну IP-адресу модуля в адресному рядку браузера. Внаслідок цього відкриється вікно з відео трансляцією з модуля. Щоб почати відео трансляцію з модуля, натисніть кнопку 'Start Stream' внизу даної сторінки.

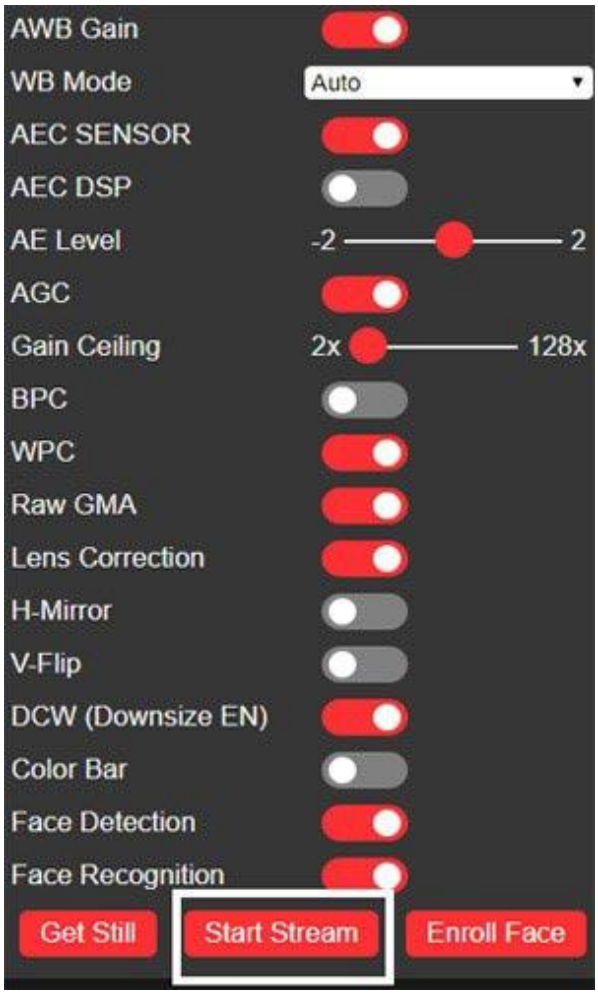

Рисунок 2.32 – Вікно з відео трансляцією з модуля

Щоб розпізнавати обличчя за допомогою модуля ESP32-CAM спочатку повинні записати їх у пам'ять модуля (enroll the faces). Для цього перейдіть на вкладку виявлення та розпізнавання осіб (Face recognition and detection features) і в ній натисніть кнопку Enroll Face. Відбудеться кілька спроб збереження особи. Після збереження особи йому надається ім'я subject 0, де нуль позначатиме ідентифікатор особи (face ID). Подібним чином ви можете зберегти пам'ять модуля та інші особи.

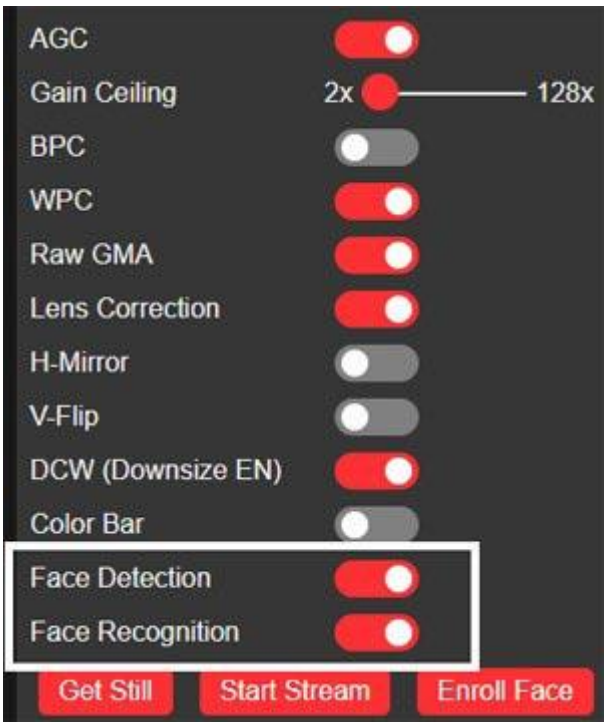

Рисунок 2.33 – Вікно збереження обличчя

Після цього, якщо обличчя розпізнаватиметься у відео потоці, модуль ESP32-CAM оновлюватиме документ google (google sheet) ім'ям, асоційованим з ідентифікатором розпізнаної особи.

3 Розробка віддаленого програмного забезпечення системи розпізнавання обличчя

3.1 Етапи опрацювання відеокадрів

Як було зазначено раніше, обробка відеокадрів складається із двох основних етапів. У першому етапі відбувається виявлення обличчя на кадрі, але в другому – саме розпізнавання виявленого обличчя.

При розробці системи використали метод виявлення обличчя Віоли-Джонса, а розпізнавання – метод найближчого сусіда з використанням гістограм центрально-симетричних локальних бінарних шаблонів.

В системі розпізнавання осіб використовується метод знаходження мінімальної відстані між гістограмою вхідного зображення обличчя та гістограм, що зберігаються на базі.

Також варто відразу відзначити, що продуктивність вибраних алгоритмів суттєво залежить від різних факторів, наприклад: освітлення, положення обличчя, заднього тла тощо. Отже, для забезпечення оптимальної ефективності системи важливо створити сприятливі умови для її використання:

- фронтальне, чи близьке щодо нього становище особи;
- особа, яка не перекривається іншими об'єктами;
- нейтральний вираз обличчя;
- тестова вибірка повинна зніматися за однакових умов освітлення.

Крім основних етапів виявлення і розпізнавання, в системі, що розробляється, існують проміжні етапи обробки знайдених облич: Після виявлення обличчя використовується фільтр Гаусса для зменшення впливу шумів, а також маска значущих областей, яка видаляє вплив кутових ділянок зображення, де зазвичай міститься задній план. Це призводить до створення узагальненого алгоритму обробки кадрів, що містить наступні етапи: виявлення обличчя, обробка знайдених облич за допомогою фільтра гауса, LBP трансформація знайдених облич з наступним застосуванням маски значних областей, розрахунок гістограм знайдених облич, класифікація облич методом

найближчого сусіда з гістограм. Загальний алгоритм системи, що розробляється, представлений на рисунку 3.1.

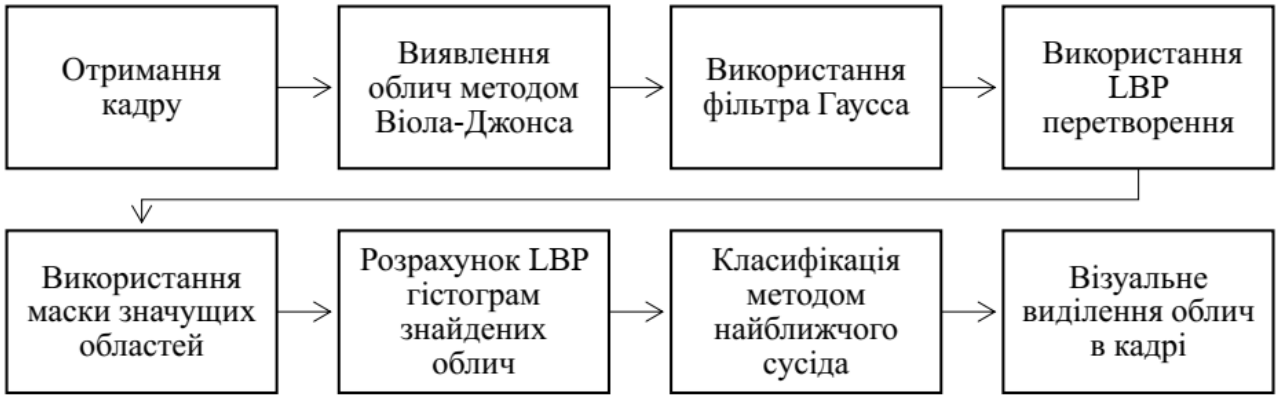

Рисунок 3.1 – Алгоритм, що використовується для обробки послідовних кадрів у відеопотоці

Крім того, згідно створеного узагальненого алгоритму обробки відеопотоку, було складено необхідний функціонал системи, що розробляється:

– передача живого відеопотоку зі зовнішньої веб-камери або можливість відтворення відео з файлу;

– відображення даних про належність розпізнаного обличчя до конкретного класу.графічне відображення гистограми і LBP подання особи, що відстежується;

– можливість навчання та додавання класів облич з використанням камеричерез інтерфейс програми;

– можливість налаштування параметрів роботи алгоритмів.

Для розробки системи розпізнавання був обраний метод виявлення облич Віоли-Джонса, який завдяки своїй високій швидкості, а також вкрай низькій ймовірності помилкового виявлення обличчя досі є одним із основоположних методів пошуку об'єкту на зображеннях.

Головними принципи, що є основою даного методу:

– фіксація зображень в інтегральному вигляді;

– знаходження облич з допомогою ознак Хаара;

– каскадна класифікація;

– навчання системи впізнавання об'єктів за допомогою методу «AdaBoost».

Для підвищення продуктивності обробки даних у методі Віоли-Джонса використовується метод інтегрального уявлення. Цей підхід дозволяє швидко обчислити суму яскравості пікселів у будь-якому прямокутному регіоні на зображенні. У методі інтегрального уявлення зображення розглядається як матриця, розміри якої збігаються з розмірами вихідного зображення. Кожен елемент матриці містить суму інтенсивностей всіх пікселів, що знаходяться вище та ліворуч від розглянутого пікселя, плюс вага цього пікселя.

У вдосконаленому методі Віоли-Джонса, що використовується у бібліотеці комп'ютерного зору OpenCV і в нашій розробці, застосовуються додаткові ознаки Хаара. Каскади Хаара представляють собою прямокутні області, складені з кількох сусідніх прямокутних зон, що відрізняються світлотою або темрявою. Приклади таких додаткових ознак показані на рисунку 3.2.

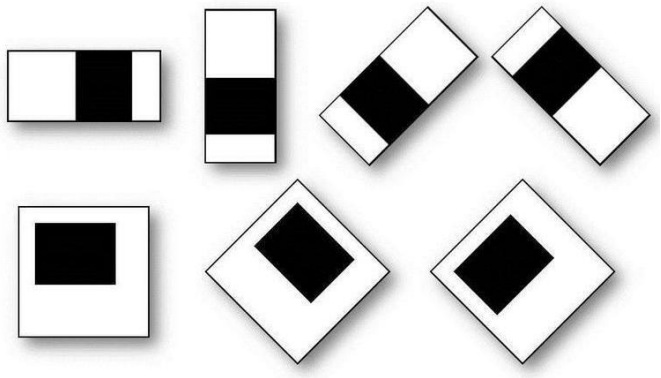

Рисунок 3.2 – Додаткові ознаки Хаара

Обчислюваним значенням таких додаткових ознак є різниця сум значень яскравостей точок, що закриваються світлою частиною ознаки і точок, що закриваються темною частиною ознаки.

Пошук обличчя відбувається за допомогою так званого вікна, що сканує, алгоритм сканування якого виглядає так:

– є досліджуване зображення, вибрано вікно сканування, вибрані ознаки, що використовуються:

– обирається досліджуване зображення, вибирається вікно сканування та визначаються необхідні ознаки;

– вікно сканування рухається послідовно за зображенням з кроком в одну комірку вікна, при цьому розмір віконного вікна становить 24 на 24 комірки.

– під час сканування в кожному вікні обчислюється близько 200 000 варіантів розташування ознак, за допомогою зміни масштабу та їх положення в межах вікна сканування;

– сканування виконується послідовно для різних масштабів;

– масштабується саме вікно, що сканується, тобто змінюється розмір комірки;

– всі знайдені ознаки передаються класифікатору, який, аналізуючи їх значення, визначає, чи належить область зображення, що відповідає вікну, обличчю чи ні.

Для докладнішого опису об'єкта потрібно використовувати більше характеристик Хаара, вони дуже підходять для класифікації чи навчання. При цьому,для пришвидшення процесів виявлення, у методі Віоли-Джонса використовується каскадний класифікатор, який дозволяє прискорити виявлення облич, фокусуючи роботу на найцікавіших областях зображення [11,12].

Отже, використовуючи обмежені обчислювальні ресурси, можна на початкових етапах процесу розпізнавання відсіяти зображення, які імовірно не містять шуканого об'єкта (у цьому випадку людину). Кожен рівень каскаду навчається за допомогою алгоритму AdaBoost.

З метою усунення шумів на зображеннях облич було застосовано фільтр Гаусса. Фільтр Гауса - це метод розмиття зображення, що використовує нормальний розподіл, також відомий як Гаусовий розподіл, для обчислення перетворення, яке застосовується до кожного пікселя зображення.

Розмиття по Гауссу дозволяє позбутися небажаних шумів на зображеннях, і зводить до мінімуму їх вплив при класифікації облич.

LBP оператор вперше був запропонований в 1996 для класифікації текстур [13]. Проте пізніше знайшов застосування й у розпізнавання обличчя. Суть роботи оператора полягає в тому, що для кожного пікселя зображення застосовується порігове перетворення, в якому значення яскравості пікселя порівнюється з значеннями яскравостей його сусідніх пікселів. Результат порівняння для кожного пікселя визначає, якщо він буде оброблений, конкатенується в двійкове число.

Після використання оператора LBP зображення розділяється на прямокутні області, для кожної з яких обчислюються гістограми. Ці гістограми вказують на те, як часто в цій області зустрічаються пікселі з різними значеннями яскравості.

Отримані гістограми нормалізуються, конкатенуються і використовуються надалі як ознаки класифікації. Приклад розбиття зображення прямокутні області і формування гістограм можна побачити рисунок 3.3.

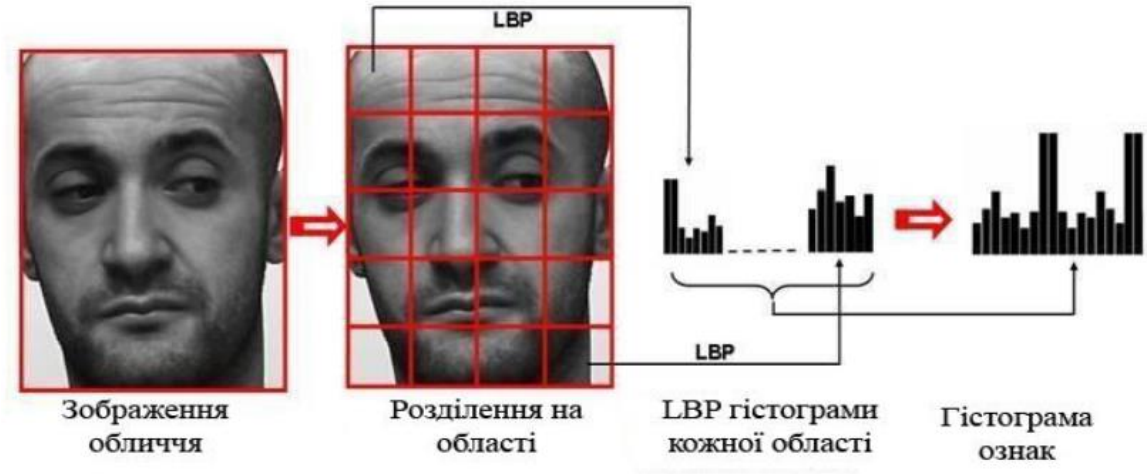

Рисунок 3.3 – Поділ зображення на прямокутні сегменти та створення гістограми

У результаті отримуємо опис обличчя на трьох рівнях локалізації. У цьому такий опис залежить від монотонних змін освітлення.

Зображення обличчя, які отримують після процедури виявлення, мають квадратну форму, проте саме обличчя не заповнює увесь простір такого

зображення. Тому було б логічно виключити вплив класифікатора на області зображення, де відсутнє обличчя.

Найпростішим способом вирішення цієї проблеми є застосування маски значних областей зображення. Така маска є зображенням з одним розміром, які і зображення, що обробляєтьсяПікселі з ненульовою яскравістю в масці відповідають значним областям.

При розв'язанні задачі класифікації за допомогою локальних бінарних шаблонів використовується маска значущих областей доцільно застосувати після виконання LBP перетворення перед розрахунком гістограм. Таким чином, на гістограмі незначні пікселі зображення будуть об'єднані в одне значення. Зразок застосування маски значущих областей до зображення, обробленого оператором LBP, наведено на рисунку 3.4.

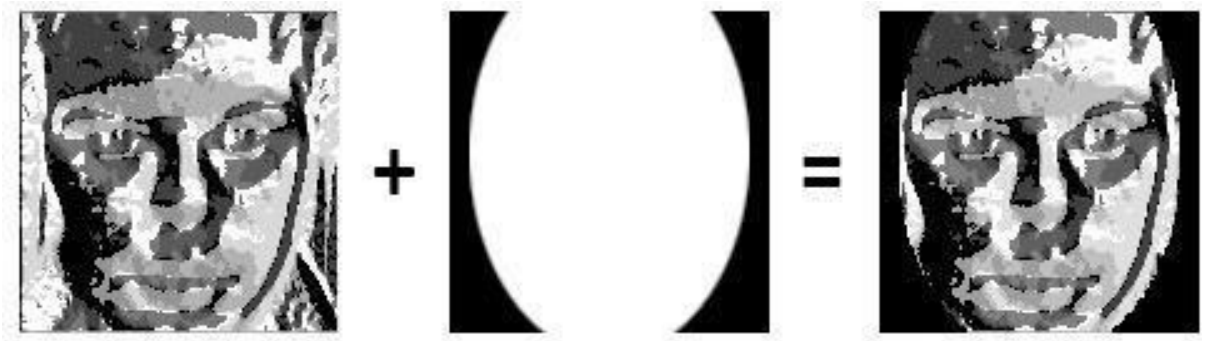

Рисунок 3.4 – Застосування маски значних областей

Як видно з рисунка, в результаті застосування цієї операції пікселі зображення, які не повинні впливати на результат класифікації, набувають нульового значення яскравості.

Метод найближчого сусіда представляє собою простий алгоритм класифікації, що ґрунтується на тому, що об'єкт відноситься до того класу, елемента якого він найближче перебуває. Наприклад, на рисунку 3.5 зелене коло відповідно до даного алгоритму має бути класифіковано як червоний трикутник.

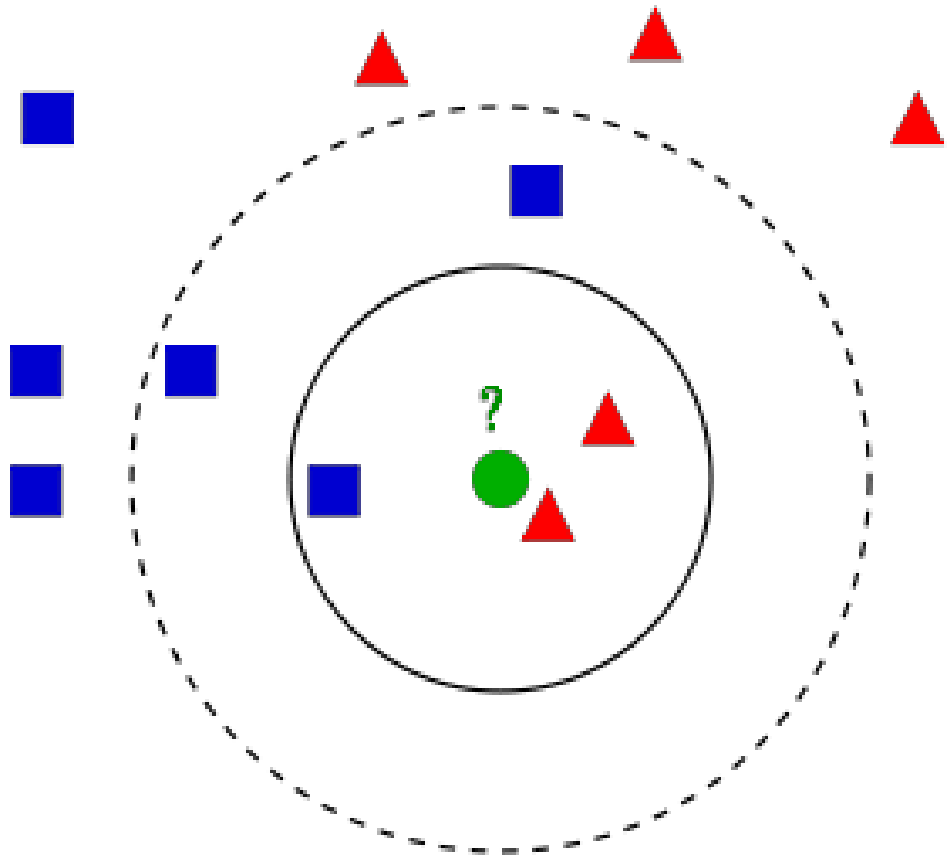

Рисунок 3.5 – Метод найближчого сусіда

Для підвищення ефективності також використовується метод, де об'єкт класифікують, враховуючи його приналежність до конкретного класу, до якого належить більшість його сусідів в околиці заданого розміру. Проте за розв'язання завдання класифікації облич такий підхід негативно впливає роботу класифікатора.

Цей метод застосовують у разі, коли ціна помилки неправильної класифікації є величезною, а ваги помилок даних маленькі. Головним недоліком методу найближчого сусіда є його чутливість до відповідей окремих (іноді неточних) даних. Незважаючи на це, метод проявляє високу ефективність у широкому спектрі завдань класифікації.

Також важливим є вибір метрики, що визначає відстань між гістограмами. Для досягнення максимальної точності класифікації важливо обрати ту метрику, яка найкраще відображатиме відмінності між гістограмами зображень різних класів.

3.2 Засоби розробки

Розробка системи велася на мові програмування C#, яка є об'єктно-орієнтована. Розробка велась у у середовищі Microsoft Visual Studio 2017.

Microsoft Visual Studio - це набір програмних продуктів від компанії Microsoft, який включає інтегроване середовище розробки програмного забезпечення, а також різноманітні інструменти та ресурси для розробки програм.

В якості основної мови було обрано C#, оскільки вона володіє необхідними функціями для розробки, включаючи вбудовану підтримку узагальнень, делегатів і подій, що спрощує процес реалізації.

Мова програмування C# належить до сімейства мов з синтаксисом, схожим на мову C, і відрізняється такими особливостями, як статична типізація, підтримка поліморфізму, можливість перевантаження операторів, включаючи явне та неявне приведення типів, а також функції, такі як делегати, атрибути, події, властивості, узагальнені типи та методи, ітератори, анонімні функції зі змиканнями, LINQ, обробка винятків та коментарі у форматі XML. Мій вибір цієї мови програмування був зумовлений моїм досвідом розробок, отриманим під час навчання.

Додатково було встановлено для використання як одну з основних – бібліотеку OpenCV. Ця бібліотека розроблена на C/C++ і має обгортку для .NET мов – EmguCV, яка була використана в роботі. EmguCV містить алгоритми для обробки, реконструкції та обробка зображень, розпізнавання візуальних образів, запис відео, відстеження об'єктів, налаштування камери та інші функції, які були включені до системи [16,17].

3.3 Опис основних класів, функцій та методів

У цьому підрозділі наведено опис основних класів, функцій та методів розробленої програми.

Головний клас програми MainForm містить список класів облич типу FaceClass та список розпізнаних облич типу RecognizedFace. Також він використовує у своїй роботі детектор облич FaceDetector і LBP перетворювач зображень LBPTransformer. Кожна розпізнана особа RecognizedFace при цьому посилається на відповідний клас облич FaceClass.

Клас головної форми програми містить елементи інтерфейсу програми, а також список розпізнаних та список нерозпізнаних облич, якими оперуватиме система при обробках кадрів. В додаток до цього, у вказаному класі реалізовані функції для обчислення гістограм та відображення їх результатів у графічному вигляді. Подробний опис елементів інтерфейсу та механізмів їх взаємодії буде представлений у розділі, присвяченому розробці інтерфейсу системи.

Назва класу: RecognizedFace.

Клас RecognizedFace описує обличчя розпізнане класифікатором. Елементи даного класу зберігаються в списку розпізнаних облич, що оновлюється. Інформація, яка відображається на формі, формується на основі даних, які надходять від_faceClass.

Атрибути класу:

private FaceClass_faceClass – клас облич, якого належить розпізнане обличчя.

private Rectangle_rect – прямокутна область, яка відповідає положенню обличчя у кадрі відеопотоку.

private Mat_histogram – LBP гістограми особличчя.

private Image<Gray, Byte>_face – Оброблене в даний момент зображення обличчя, взяте з кадрів відеоданих.

private Image<Gray, Byte>_lbp –_face представлене у форматі LBP (Local Binary Patterns)

private double_distance – Визначається відстань між гістограмою LBP обличчя та гістограмою LBP найближчого елемента його класу.

Методи класу:

Public RecognizedFace(FaceClass fc, Rectangle r, Image<Gray, Byte> f, Image<Gray, Byte> l, Mat h, double d) – Конструктор класу. Як параметри якого передаються значення, що присвоюються атрибутам створюваного об'єкта.

Клас LBPTransformer описує CS-LBP перетворювач. Цей клас є статичним класом, не має атрибутів і містить лише один метод, який виконує цетрально-симетричне LBP перетворення зображення.

Методи класу:

public static Image<Gray, Byte> CSTransform(Image<Gray, Byte> input) – забезпечує LBP перетворення вхідних зображень (input) і повертає перетворені зображення.

FaceDetector

Клас FaceDetector описує детектор облич, який використовує за базу метод Віоли-Джонса у своїй роботі.

Атрибути класу:

private CascadeClassifier_haar – Каскадний класифікатор, що дозволяє проводити ідентифікацію облич.

Методи класу, який застосовуються:

public FaceDetector() – Конструктор класу. Бере у вигляді xml файлу в атрибут_haar навчений каскад для розпізнавання облич.

public Rectangle[] GetFacesRect(Image<Bgr, Byte> frame, double scaleFactor, int minNeighbors, int sz) – Приймає на вході зображення (frame), у яких проводимо пошук облич та приймає дані для налаштування параметрів виявлення, такі як фактор збільшення вікна (scaleFactor), мінімальна кількість сусідів (minNeighbors) і мінімальний розмір обличчя (sz). Результатом є список прямокутників, які відображають положення облич на зображенні (frame).

Важливо відзначити, що мінімальна кількість сусідів впливає на точність виявлення облич. Даний параметр задає необхідну для визнання області зображення обличчям кількість спрацьовувань детектора при різних масштабах скануючого вікна, що змінює свої розміри відповідно до фактора збільшення шкалифактора, в конкретній частині зображення.

Алгоритм виявлення обличчя здійснюється використовуючи функцію CascadeClassifier.DetectMultiScale бібліотеки Emgu CV, що викликається для класифікатора каскадного_haar. Ця функція є реалізацію каскадної класифікації з методу Віоли-Джонса. Їй і передаються, описані вище параметри.

FaceClass

Даний клас списує клас (категорію) обличчявідповідає конкретній людині, що розпізнається.

Атрибутами класу є:

private Image<Gray, Byte>_img – зображення особи, яка використовується разом із назвою класу для зручності ідентифікації та розрізнення класів.

private Mat[]_histogram – дані LBP гістограми для відмінних зображень особличчя, яка відповідає класу. Використання не однієї, а кількох гістограм продиктовано особливостями методу найближчого сусіда, оскільки при ідентифікації програма відшукує найподібнішого представника класу з усіх присутніх.

MainWindow

Клас MainWindow описує головне вікно програми.

Атрибути класу:

private FaceDetector_detector – Детектор обличчя.

private List<FaceClass>_faces список класів обличчя, які зберігаються в пам'яті програми в даний момент.

privateList<RecognizedFace>_recognizedFaces – Список обличчя, розпізнаних у поточних кадрах відеопотоків.

private List<Rectangle>_notRecognized – перелік областей ключових кадрів відеопотоку, що містять особи, які класифікатор не може віднести до будь-якої особи з наявних класів.

private Image<Gray, Byte>_mask – Зображення масок значущих ділянок.

private Image<Gray, Byte>[]_currentFaces – Користувач знімає зображення обличчя для створення нового класу облич.

private int_currentFacesCount – Кількість поточних зображень облич для створення нового класу.

private Rectangle[]_facesRect – Перелік регіонів на поточному кадрі відеопотоку, що містять особи.

private bool_capturing – прапор, що показує чи зараз захоплення відеопотоку.

privateCapture_capture – Об'єкт, що забезпечує фіксацію відео.

private String_videoPath – Шлях до відеофайлів.

private Mat_frame – поточний кадр відео.

private Image<Bgr, Byte>_frameImg – поточний кадр відео з яким необхідно виконати ряд операцій, які неможливо виконати з об'єктом типу Mat.

Методи класу:

public MainForm() – Конструктор класу.

Public Mat CalcHistogram(Image<Gray, Byte> image) – Повертає гістограми зображень image.

private void RecognizeFace(Rectangle rect, Image<Gray, Byte> face, Image<Gray, Byte> lbp, Mat histogram, decimal threshold) – метод здійснює розпізнавання обличчя та формування списків розпізнаних та нерозпізнаних облич. Метод приймає наступні параметри: зображення обличчя (face), відповідну область кадру (rect) та LBP-зображення обличчя (lbp) та його гістограма histogram, а також поріг розпізнавання threshold. На основі цих даних та списку класів обличчя виконується класифікація особи методом найближчого сусіда.

Після визначення класу особи відбувається порівняння відстані до класу з пороговим значенням, внаслідок чого особа або поміщається до списку розпізнаних облич або до списку нерозпізнаних облич.

private void ProcessFaces() – даний метод здійснює обробку виявлених у кадрі відео обличчя у відповідності з алгоритмом, блок-схему показано на рисунку 3.6.

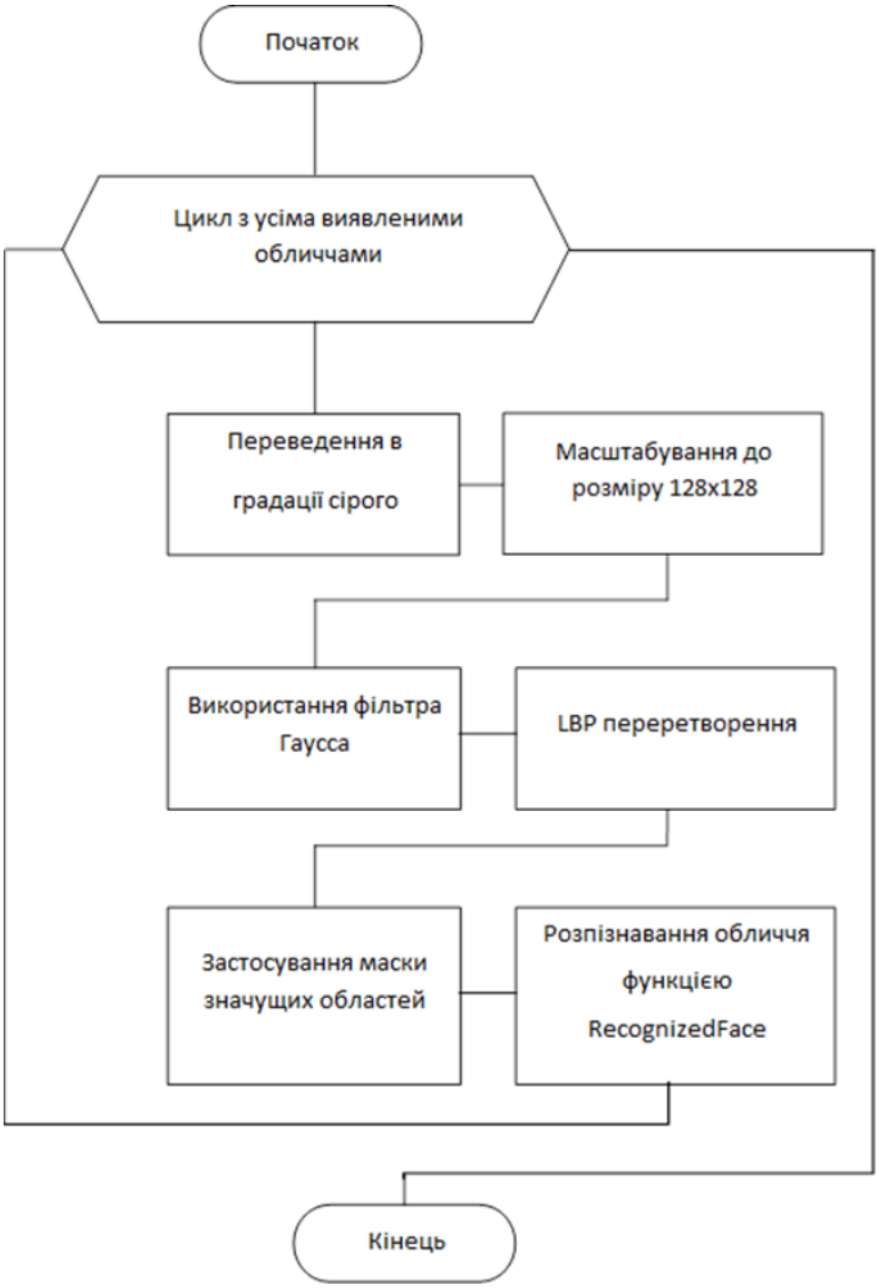

Рисунок 3.6 – Алгоритм обробки зафіксованого обличчя

private void ProcessFrame(object sender, EventArgs arg) – функції опрацювання кадрів відео.

private void DrawDetected() – ця функція забезпечує візуальні відзначення обличчя у кадрах відео та виведення інформації про обраного користувача особі на форму.

private void ClearData() – метод знищує у формі дані про обрану особу. Використовується щоразу при оновленні кадрів відео під час завантаження нових даних про обличчя з кадру.

private void DrawHistogram(Mat histogram) – метод формує спеціальні елементи у формі гістограм histogram.

private void openFaceClass(object sender, CancelEventArgs e) – метод завантажує в додатки класів ообличчя із зовнішніх файлів.

private void addFaceButton_Click(object sender, EventArgs e) – метод доповнює наявні зараз у кадрі обличчя у список зображень, що використовується формування нових класів облич.

private void addFaceButton_Click(object sender, EventArgs e) – метод додає наявне зараз у кадрі обличчя у список зображень, що використовується формування нового класу обличчя.

private void addClassButton_Click(object sender, EventArgs e) метод, який доповнює до списків класу обличчя нові класи.

private void faceClassesListBox_SelectedIndexChanged(object sender, EventArgs e) – метод, який автоматично відображає зображення нового обличчя на формі після його класифікації.

private void removeFaceButton_Click(object sender, EventArgs e) – метод, що знищує останні зображення зі списку обличчя, які використовуються для створення нового класу.

private void videoButton_Click(object sender, EventArgs e) – метод, завдяки якому стартує зчитування кадрів із відео.

private void saveFaceClass(object sender, CancelEventArgs e) – метод, який зберігає обраний клас обличчя у файлі.

private void saveClassButton_Click(object sender, EventArgs e) – метод, що запускається при натисканні кнопки. Відкриває діалогове вікно збереження файла.

private void openClassButton_Click(object sender, EventArgs e) – метод , який запускає та відображає діалогове вікно для відкриття файла.

private void removeClassButton_Click(object sender, EventArgs e) – метод, що знижщує обраний клас обличчя в списках.

private void openVideoButton_Click(object sender, EventArgs e)– метод, що опрацьовує натискання кнопок. Відкриває діалогове вікно обрання відеофайлу.

private void openFileDialog2_FileOk(object sender, CancelEventArgs e) – зберігає шлях до вибраного відеофайлу до атрибуту_videoPath.

Повністю код програми представлено у додатку В.

3.4 Опис інтерфейсу системи

Під час розробки інтерфейсу було створено одне основне вікно, яке містить у собі активні елементи для налаштування параметрів роботи програми, та області виведення даних про обличчя, що розпізнаються.

Інтерфейс програми містить кілька областей: область виведення відеопотоку з камери або з файлу, область налаштування параметрів роботи програми, область роботи з класами обличчя та область виведення інформації про обрану особу. Загальний вигляд головної форми додатка представлений рисунку 3.7.

Розглянемо докладніше різні сфери інтерфейсу розробленої системи.

У зоні відображення відеопотоку відображаються оброблені кадри, і також розміщені кнопки для налаштування джерела відеопотоку, виклику діалогового вікна відкриття відеофайлу та управління початком та зупинкою захоплення відеопотоку.

Як джерело відеопотоку може використовуватися ESP32-CAM з відповідним налаштуванням. Для вибору відеофайлу-джерела потрібно натиснути кнопку Open File. Після того, як користувач вибере файл у діалоговому вікні і натисне кнопку "OK", шлях до вибраного файлу відобразиться на формі. Запуск/зупинка обробки та виведення кадрів із вибраного джерела здійснюється натисканням кнопки Start/Stop.

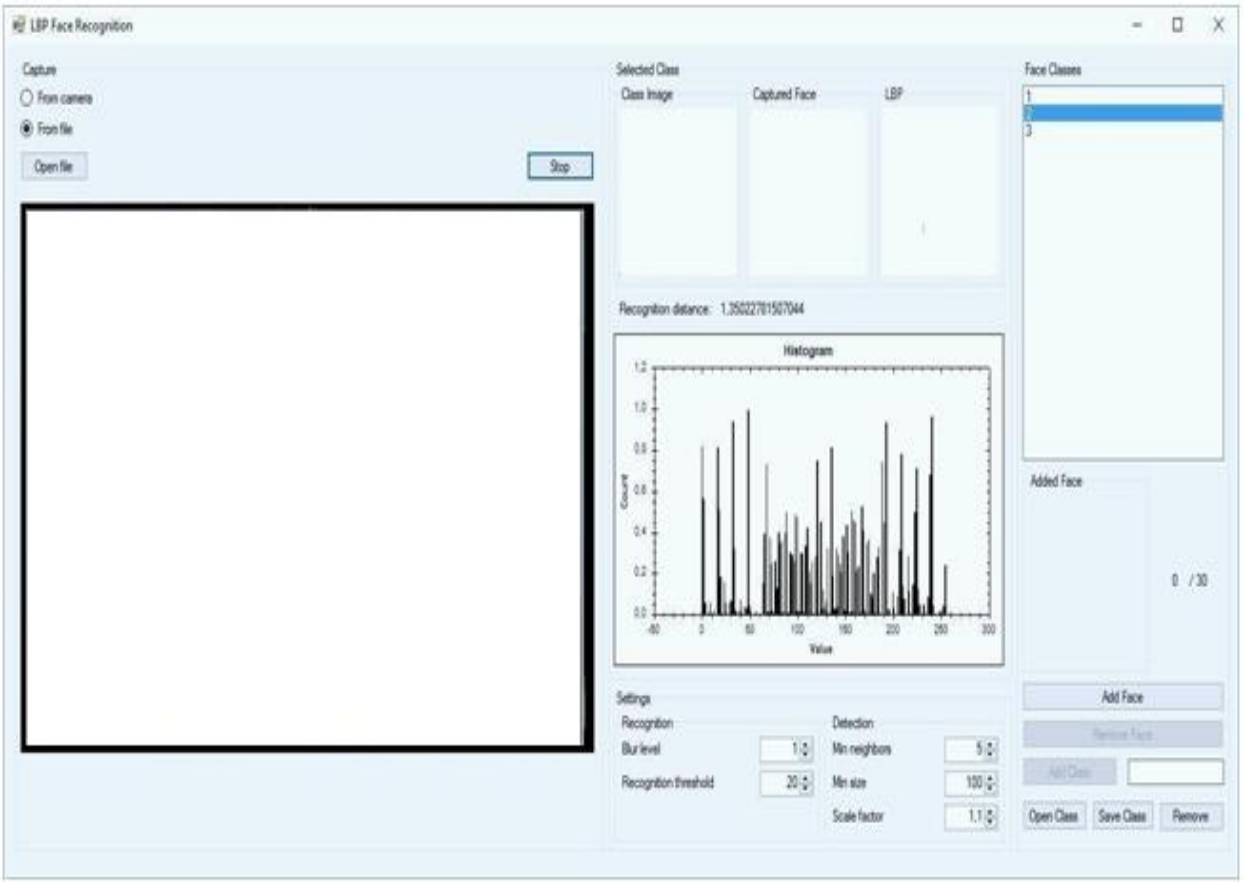

Рисунок 3.7 – Інтерфейс програми

Також можна обрати з використанням сервісу IFTTT для передачі даних у Google Sheets, що збирає дані через ESP32-CAM. Область виведення відеопотоку представлена рисунку 3.8.

Область виведення інформації про обрану особу призначена для відображення даних облич, що розпізнаються додатком. Джерелом даних для відображення є обраний користувачем у списку Face Classes клас обличчя та розпізнана особа, яка відповідає цьому класу.

У цій галузі форми виводяться зображення обраного класу, прямокутна область поточного кадру, що відповідає особі в цьому класі та LBP перетворенню відображення облич. Також, відобразиться дійсна відстань між гістограмою розпізнаної особи та гістограмою її класу та графічне представлення гістограми LBP зображення обличчя.

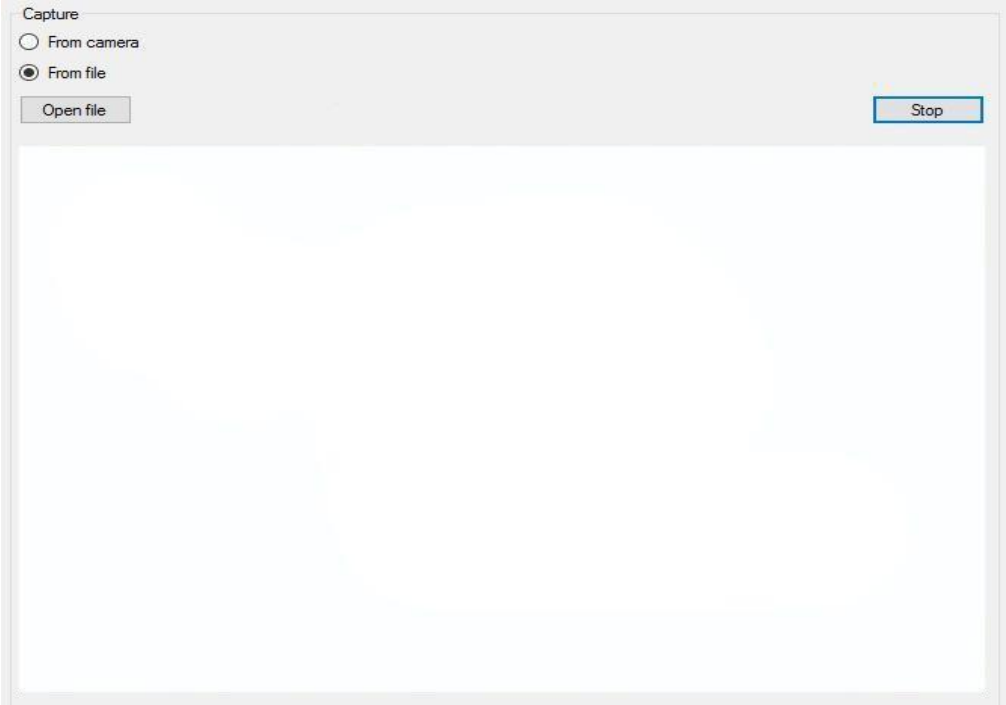

Рисунок 3.8 – Область виведення відеопотоку

Область виведення інформації про обрану користувачем особу представлена на рисунку 3.9.

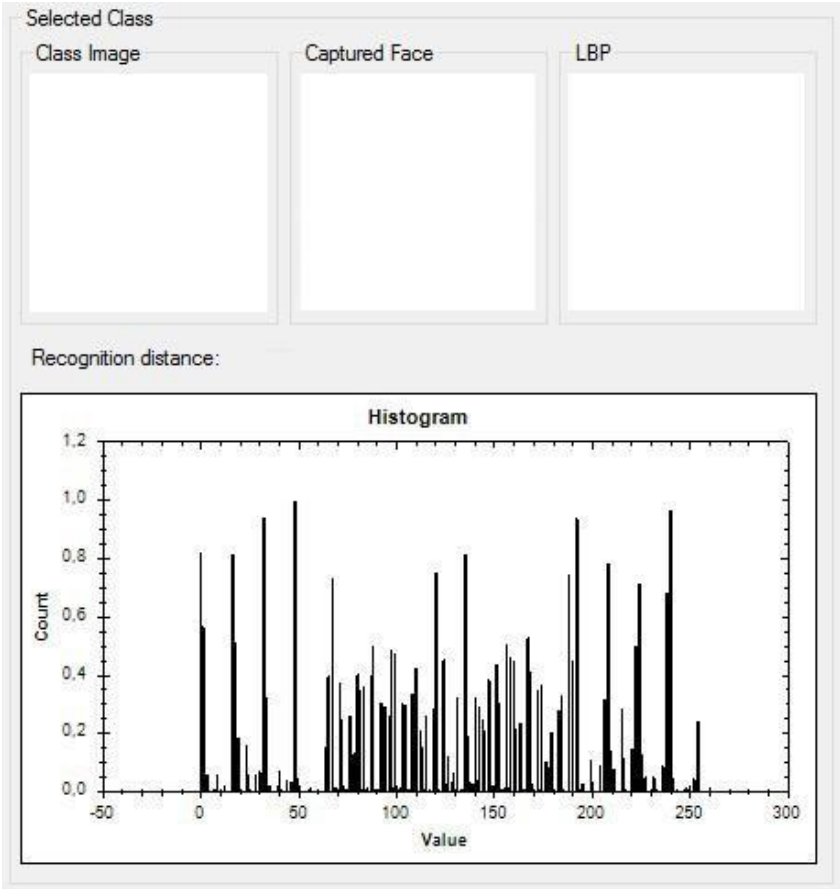

Рисунок 3.9 – Область виведення інформації про обрану особу

Область налаштувань роботи програми включає налаштування двох груп. Перша категорія включає в себе налаштування для роботи з системою розпізнавання облич. До них відноситься рівень розмиття по Гауссу і поріг розпізнавання. Поріг розпізнавання – максимально допустима для ухвалення рішення про належність особи класу відстань між LBP гістограмою особи та гістограмою класу.

В другій групі налаштувань цієї області знаходяться параметри для налаштування детектора облич. Значення даних установок передаються методу GetFacesRect. Область налаштувань роботи програми представлена рисунку 3.10.

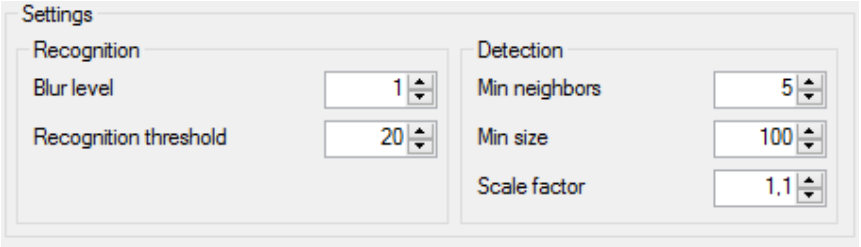

Рисунок 3.10 – Область налаштувань роботи програми

Робоча зона, що взаємодіє з класами облич, створена для роботи з переліком класів облич та формування нових класів. У верхній частині області знаходиться список завантажених зараз у програмі класів облич. Користувач може обирати необхідний клас, натискаючи ліву кнопку миші, і отримувати інформацію про вибраний клас та обличчя, які йому відповідають, буде відображатися в області виведення інформації.

Нижче розташовані елементи інтерфейсу на формування нового класу обличчя. Після натискання кнопки "Додати обличчя" в набір зображень для формування класу буде додано зображення обличчя, яке зараз присутнє на кадрі відеопотоку. За допомогою кнопки "Видалити обличчя" можна видалити останнє зображення із набору. Як тільки необхідний набір зображень (до 30 включно) буде сформовано, необхідно ввести ім'я класу у текстове поле та,щоб створити новий клас облич, користувач повинен натиснути кнопку "Додати клас". Новий клас буде створений на основі набору зображень, які були додані користувачем. Створений клас відразу ж з'явиться у списку класів і

використовуватиметься при розпізнаванні обличчя. В даній області також доступні кнопки для відкриття існуючого класу обличчя з файлу, збереження обраного класу у файл та видалення обраного класу обличчя представлена рисунку 3.11.

Рисунок 3.11 – Область роботи з класами обличчя

Також варто відзначити, що для запобігання помилкам різні елементи інтерфейсу системи включаються і вимикаються в залежності від активності відеопотоку і наявності в ньому зображень обличчя.

Висновки

У ході виконання бакалаврського дипломного проєкту було розроблено мікропроцесорну систему розпізнавання обличчя засобами IoT. Цей апаратно-програмний комплекс базується на технологіях IoT, зокрема на використанні ESP32-CAM, а також застосовує метод Віоли-Джонса та локальні бінарні шаблони.

Під час роботи були виконані наступні завдання:

Проаналізовано існуючі підходи для розпізнавання обличчя, що дозволило визначити найефективніші методи.

Визначено класифікацію алгоритмів розпізнавання, що дозволило зрозуміти їх сильні та слабкі сторони.

Проведено аналіз існуючих на ринку систем розпізнавання, визначено їх переваги та недоліки, що дозволило обрати оптимальні рішення для розробки.

Проаналізовано основні інструментальні засоби для розробки та вибрано оптимальні з них, що дозволило створити ефективну систему.

Спроектовано та програмно реалізовано систему розпізнавання облич засобами IoT, що дозволило побудувати їх на існуючих компонентах.

Впроваджено метод Віоли-Джонса для детекції обличчя та локальні бінарні шаблони для їх розпізнавання, що дозволило підвищити точність розпізнавання.

Проведено тестування як апарано-програмної частини, так і окремого програмного забезпечення, що дозволило підтвердити їх працездатність.

Розроблена мікропроцесорна система розпізнавання обличчя засобами IoT може бути використана для різних завдань відеоаналітики, зокрема для систем контролю доступу та ідентифікації особистості. Крім того, в процесі роботи було здійснено огляд сучасних систем розпізнавання, виявлено їхні недоліки та проблеми, які впливають на ефективність роботи, вивчено методи обробки зображень та проведено аналіз сучасних алгоритмів розпізнавання.

Оскільки поставлені задачі виконано, мету бакалаврського дипломного проєкту досягнуто.

Список використаних джерел

1. Analysis market recognition technology [Електронний ресурс]. // TrendForce. Режим доступу:https://www.trendforce.com/ (Дата звернення: 2.05.2024).

2. Мікропроцесорна система розпізнавання сигналів чутного діапазону О.Д. Азаров, С.В. Богомолов, Д.Т. Гріша, Б.О. Ковальчук, конференції ВНТУ електронні наукові видання, Молодь в науці: дослідження, проблеми, перспективи (МН-2024). Електронний ресурс] – Режим доступу URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21327/17709

3. Мікропроцесорна система розпізнавання обличчя за допомогою IoT О.Д. Азаров, С.В. Богомолов, Б.О. Ковальчук, Д.Т. Гріша, конференції ВНТУ електронні наукові видання, Молодь в науці: дослідження, проблеми, перспективи (МН-2024). Електронний ресурс] – Режим доступу URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/21398/17744

4. Метод Віоли - Джонса [Електронний ресурс]. // Wikipedia. - Режим доступу:https://ua.wikipedia.org/wiki/Метод_Віоли_—_Джонса (Дата звернення: 5.05.2024).

5. Спосіб опорних векторів [Електронний ресурс]. // Wikipedia. – Режим доступу: https://ua.wikipedia.org/wiki/Метод_опорних_векторів (дата звернення: 15.05.2024).

6. Системи технічної безпеки та охорони «MTI». FaceVACS – VideoScan [Електронний ресурс].//Режим доступу: http://www.security.mti.ua/products/sistemy-videonabludeniya/Soft-raspoznavanie-liz/Cognitec/152-facevacsvideoscan_/ (Дата звернення: 12.05.2024).

7. NEC. Facial Recognition [Електронний ресурс]. // Режим доступу:https://ua.nec.com/solutions/security/technologies/face_recognition.html (дата звернення: 12.05.2024).

8. VisionLabs. LUNA SDK [Електронний ресурс]. // Режим доступу:https://visionlabs.ai/ru/luna-platform-info.html (Дата звернення: 13.05.2024).

9. Neurotechnology. VeriLook SDK [Електронний ресурс]. // доступу:http://www.neurotechnology.com/ (Дата звернення: 13.05.2024).

10. Локальні бінарні шаблони [Електронний ресурс]. // Режим доступу:https://ua.wikipedia.org/wiki/Локальні_бінарні_шаблони (дата звернення: 13.05.2024).

11. Татаренков Д. А. Аналіз методів виявлення осіб на зображенні [Електронний ресурс]. // Молодий вчений. - 2015. - №4. - С. 270-276. - Режим доступу: https://moluch.com.ua/archive/84/15524/ (дата звернення: 11.05.2024).

12. Броневич А. Н. Лекції з методів машинного навчання[Електронний ресурс].// Режим доступу: http://window.edu.ua/resource/files/lect_Lepskiy_Bronevich.pdf (Дата звернення: 05.05.2024).

13. Методи найближчого сусіда та k-найближчих сусідів [Електронний ресурс]. // Режим доступу: http://studbooks.net/2429081/informatika

14. /metody_blizhayshego_soseda_blizhayshih_sosedey (Дата звернення: 13.05.2024).

15. About OpenCV [Електронний ресурс]. // Режим доступу:http://opencv.org/about.html(Дата звернення: 08.05.2024).

16. EmguCV [Електронний ресурс]. // Режим доступу:http://www.emgu.com/wiki/index.php/Main_Page(Дата звернення: 18.05.2024).

17. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія») / Уклад. О. Д. Азаров, С. В. Богомолов, С. І. Швець, — Вінниця : ВНТУ, 2022. – 106 с.

Додаток А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

_____О. Д. Азаров

“06” травня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврського дипломного проекту

«Мікропроцесорна система розпізнавання обличчя засобами IoT»

08-54.БДП.023.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Богомолов С.В.

Студент групи 2КІ-19б

_____ Ковальчук Б.О.

Вінниця 2024

1 Найменування та область застосування

Система контролю та управління доступом на базі ESP32-CAM з можливістю автоматичного розпізнавання обличчя ідентифікованого персоналу та інтеграцією з базами даних.

2 Основи для розробки

Необхідність реалізації системи контролю та управління доступом до захищених об’єктів з підвищеним рівнем безпеки, що використовує біометричні методи ідентифікації.

3 Мета та призначення розробки

Створення системи контролю та управління доступом, яка допоможе автоматично розпізнавати обличчя персоналу, обробляти, накопичувати та передавати дані через мережі у відповідні бази даних, підвищуючи рівень безпеки об’єктів.

4 Етапи БДП та очікувані результати

Робота виконується в п’ять етапів, що наведені в таблиці 4.1.

Таблиця 4.1 – Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	11.03.20	24.04.24	Вступ
2	Огляд і аналіз підходів побудови систем розпізнавання обличчя	25.03.24	07.04.24	Розділ 1
3	Проектування апаратно-програмної частини системи розпізнавання обличчя засобами IoT	08.04.24	21.04.24	Розділ 2
4	Розробка віддаленого програмного забезпечення системи розпізнавання обличчя	22.04.24	05.05.24	Розділ 3
5	Оформлення пояснювальної записки	06.04.24	19.05.24	ПЗ, презентація

5 Матеріали, що подаються до захисту БДП

Пояснювальна записка БДП, графічні і ілюстративні матеріали, протокол попереднього захисту БДП на кафедрі, відзив наукового керівника, рецензія опонента, протоколи складання державних екзаменів, анотації до БДП українською та іноземною мовами, довідка про відповідність оформлення БДП діючим вимогам.

6 Порядок контролю виконання та захисту БДП

Виконання етапів графічної та розрахункової документації БДП контролюється науковим керівником згідно зі встановленими термінами. Захист БДП відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

7 Вимоги до оформлення БДП

методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Системне програмування»). Кафедра обчислювальної техніки ВНТУ 2022, ДСТУ 3008-95, ДСТУ 3974-2000 «Правила виконання дослідно-конструкторських робіт. Загальні положення» та діючого ГОСТ 2.114-95 ЄСКД.

8 Вимоги щодо технічного захисту інформації в БДП з обмеженим доступом

Відсутні.

Технічне завдання до виконання отримав_____ Ковальчук Б.О.

ДОДАТОК Б

Лістинг програми модуля ESP32

```
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "fd_forward.h"
#include "dl_lib.h"
#include "fr_forward.h"
#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7
#define FACE_COLOR_WHITE  0x00FFFFFF
#define FACE_COLOR_BLACK  0x00000000
#define FACE_COLOR_RED    0x000000FF
#define FACE_COLOR_GREEN  0x0000FF00
#define FACE_COLOR_BLUE   0x00FF0000
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN   (FACE_COLOR_BLUE | FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE | FACE_COLOR_RED)
//int FACE_ID;
extern int FACE_ID;
typedef struct {
      size_t size; //number of values used for filtering
      size_t index; //current value index
      size_t count; //value count
      int sum;
      int * values; //array to be filled with values
} ra_filter_t;
typedef struct {
      httpd_req_t *req;
      size_t len;
} jpg_chunking_t;
#define PART_BOUNDARY "123456789000000000000987654321"
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";
static ra_filter_t ra_filter;
httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;
static mtmn_config_t mtmn_config = {0};
```

```
static int8_t detection_enabled = 0;
static int8_t recognition_enabled = 0;
static int8_t is_enrolling = 0;
static face_id_list id_list = {0};
static ra_filter_t * ra_filter_init(ra_filter_t * filter, size_t sample_size){
    memset(filter, 0, sizeof(ra_filter_t));
    filter->values = (int *)malloc(sample_size * sizeof(int));
    if(!filter->values){
        return NULL;
    }
    memset(filter->values, 0, sample_size * sizeof(int));
    filter->size = sample_size;
    return filter;
}
static int ra_filter_run(ra_filter_t * filter, int value){
    if(!filter->values){
        return value;
    }
    filter->sum -= filter->values[filter->index];
    filter->values[filter->index] = value;
    filter->sum += filter->values[filter->index];
    filter->index++;
    filter->index = filter->index % filter->size;
    if (filter->count < filter->size) {
        filter->count++;
    }
    return filter->sum / filter->count;
}
static void rgb_print(dl_matrix3du_t *image_matrix, uint32_t color, const char * str){
    fb_data_t fb;
    fb.width = image_matrix->w;
    fb.height = image_matrix->h;
    fb.data = image_matrix->item;
    fb.bytes_per_pixel = 3;
    fb.format = FB_BGR888;
    fb_gfx_print(&fb, (fb.width - (strlen(str) * 14)) / 2, 10, color, str);
}
static int rgb_printf(dl_matrix3du_t *image_matrix, uint32_t color, const char *format, ...){
    char loc_buf[64];
    char * temp = loc_buf;
    int len;
    va_list arg;
    va_list copy;
    va_start(arg, format);
    va_copy(copy, arg);
    len = vsnprintf(loc_buf, sizeof(loc_buf), format, arg);
    va_end(copy);
    if(len >= sizeof(loc_buf)){
```

```
            temp = (char*)malloc(len+1);
            if(temp == NULL) {
                return 0;
            }
        }
        vsnprintf(temp, len+1, format, arg);
        va_end(arg);
        rgb_print(image_matrix, color, temp);
        if(len > 64){
            free(temp);
        }
        return len;
    }
    static void draw_face_boxes(dl_matrix3du_t *image_matrix, box_array_t *boxes, int
face_id){
        int x, y, w, h, i;
        uint32_t color = FACE_COLOR_YELLOW;
        if(face_id < 0){
            color = FACE_COLOR_RED;
        } else if(face_id > 0){
            color = FACE_COLOR_GREEN;
        }
        fb_data_t fb;
        fb.width = image_matrix->w;
        fb.height = image_matrix->h;
        fb.data = image_matrix->item;
        fb.bytes_per_pixel = 3;
        fb.format = FB_BGR888;
        for (i = 0; i < boxes->len; i++){
            // rectangle box
            x = (int)boxes->box[i].box_p[0];
            y = (int)boxes->box[i].box_p[1];
            w = (int)boxes->box[i].box_p[2] - x + 1;
            h = (int)boxes->box[i].box_p[3] - y + 1;
            fb_gfx_drawFastHLine(&fb, x, y, w, color);
            fb_gfx_drawFastHLine(&fb, x, y+h-1, w, color);
            fb_gfx_drawFastVLine(&fb, x, y, h, color);
            fb_gfx_drawFastVLine(&fb, x+w-1, y, h, color);
    #if 0
            // landmark
            int x0, y0, j;
            for (j = 0; j < 10; j+=2) {
                x0 = (int)boxes->landmark[i].landmark_p[j];
                y0 = (int)boxes->landmark[i].landmark_p[j+1];
                fb_gfx_fillRect(&fb, x0, y0, 3, 3, color);
            }
    #endif
        }
```

```
}
static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t *net_boxes){
  dl_matrix3du_t *aligned_face = NULL;
  int matched_id = 0;
  aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
  if(!aligned_face){
    Serial.println("Could not allocate face recognition buffer");
    return matched_id;
  }
  if (align_face(net_boxes, image_matrix, aligned_face) == ESP_OK){
    if (is_enrolling == 1){
      int8_t left_sample_face = enroll_face(&id_list, aligned_face);

      if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
        Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
      }
      Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail,
ENROLL_CONFIRM_TIMES - left_sample_face);
      rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]", id_list.tail,
ENROLL_CONFIRM_TIMES - left_sample_face);
      if (left_sample_face == 0){
        is_enrolling = 0;
        Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
      }
    } else {
      matched_id = recognize_face(&id_list, aligned_face);
      if (matched_id >= 0) {
        Serial.printf("Match Face ID: %u\n", matched_id);
        rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u",
matched_id);
        FACE_ID = matched_id;
      } else {
        Serial.println("No Match Found");
        rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
        matched_id = -1;
      }
    }
  } else {
    Serial.println("Face Not Aligned");
    //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
  }
  dl_matrix3du_free(aligned_face);
  return matched_id;
}
static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t len){
  jpg_chunking_t *j = (jpg_chunking_t *)arg;
  if(!index){
    j->len = 0;
```

```
        }
        if(httpd_resp_send_chunk(j->req, (const char *)data, len) != ESP_OK){
            return 0;
        }
        j->len += len;
        return len;
    }
    static esp_err_t capture_handler(httpd_req_t *req){
        camera_fb_t * fb = NULL;
        esp_err_t res = ESP_OK;
        int64_t fr_start = esp_timer_get_time();
        fb = esp_camera_fb_get();
        if (!fb) {
            Serial.println("Camera capture failed");
            httpd_resp_send_500(req);
            return ESP_FAIL;
        }
        httpd_resp_set_type(req, "image/jpeg");
        httpd_resp_set_hdr(req, "Content-Disposition", "inline; filename=capture.jpg");
        size_t out_len, out_width, out_height;
        uint8_t * out_buf;
        bool s;
        bool detected = false;
        int face_id = 0;
        if(!detection_enabled || fb->width > 400){
            size_t fb_len = 0;
            if(fb->format == PIXFORMAT_JPEG){
                fb_len = fb->len;
                res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
            } else {
                jpg_chunking_t jchunk = {req, 0};
                res = frame2jpg_cb(fb, 80, jpg_encode_stream, &jchunk)?ESP_OK:ESP_FAIL;
                httpd_resp_send_chunk(req, NULL, 0);
                fb_len = jchunk.len;
            }
            esp_camera_fb_return(fb);
            int64_t fr_end = esp_timer_get_time();
            Serial.printf("JPG: %uB %ums\n", (uint32_t)(fb_len), (uint32_t)((fr_end -
fr_start)/1000));
            return res;
        }
        dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);
        if (!image_matrix) {
            esp_camera_fb_return(fb);
            Serial.println("dl_matrix3du_alloc failed");
            httpd_resp_send_500(req);
            return ESP_FAIL;
        }
```

```
            out_buf = image_matrix->item;
            out_len = fb->width * fb->height * 3;
            out_width = fb->width;
            out_height = fb->height;
            s = fmt2rgb888(fb->buf, fb->len, fb->format, out_buf);
            esp_camera_fb_return(fb);
            if(!s){
               dl_matrix3du_free(image_matrix);
               Serial.println("to rgb888 failed");
               httpd_resp_send_500(req);
               return ESP_FAIL;
            }
            box_array_t *net_boxes = face_detect(image_matrix, &mtmn_config);
            if (net_boxes){
               detected = true;
               if(recognition_enabled){
                  face_id = run_face_recognition(image_matrix, net_boxes);
               }
               draw_face_boxes(image_matrix, net_boxes, face_id);
               free(net_boxes->box);
               free(net_boxes->landmark);
               free(net_boxes);
            }
            jpg_chunking_t jchunk = {req, 0};
            s = fmt2jpg_cb(out_buf, out_len, out_width, out_height, PIXFORMAT_RGB888, 90,
jpg_encode_stream, &jchunk);
            dl_matrix3du_free(image_matrix);
            if(!s){
               Serial.println("JPEG compression failed");
               return ESP_FAIL;
            }
            int64_t fr_end = esp_timer_get_time();
            Serial.printf("FACE: %uB %ums %s%d\n", (uint32_t)(jchunk.len), (uint32_t)((fr_end -
fr_start)/1000), detected?"DETECTED ":"", face_id);
            return res;
         }
         static esp_err_t stream_handler(httpd_req_t *req){
            camera_fb_t * fb = NULL;
            esp_err_t res = ESP_OK;
            size_t _jpg_buf_len = 0;
            uint8_t * _jpg_buf = NULL;
            char * part_buf[64];
            dl_matrix3du_t *image_matrix = NULL;
            bool detected = false;
            int face_id = 0;
            int64_t fr_start = 0;
            int64_t fr_ready = 0;
            int64_t fr_face = 0;
```

```
int64_t fr_recognize = 0;
int64_t fr_encode = 0;
static int64_t last_frame = 0;
if(!last_frame) {
    last_frame = esp_timer_get_time();
}
res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
if(res != ESP_OK){
    return res;
}
while(true){
    detected = false;
    face_id = 0;
    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.println("Camera capture failed");
        res = ESP_FAIL;
    } else {
        fr_start = esp_timer_get_time();
        fr_ready = fr_start;
        fr_face = fr_start;
        fr_encode = fr_start;
        fr_recognize = fr_start;
        if(!detection_enabled || fb->width > 400){
            if(fb->format != PIXFORMAT_JPEG){
                bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
                esp_camera_fb_return(fb);
                fb = NULL;
                if(!jpeg_converted){
                    Serial.println("JPEG compression failed");
                    res = ESP_FAIL;
                }
            } else {
                _jpg_buf_len = fb->len;
                _jpg_buf = fb->buf;
            }
        } else {
            image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);
            if (!image_matrix) {
                Serial.println("dl_matrix3du_alloc failed");
                res = ESP_FAIL;
            } else {
                if(!fmt2rgb888(fb->buf, fb->len, fb->format, image_matrix->item)){
                    Serial.println("fmt2rgb888 failed");
                    res = ESP_FAIL;
                } else {
                    fr_ready = esp_timer_get_time();
                    box_array_t *net_boxes = NULL;
```

```
                    if(detection_enabled){
                      net_boxes = face_detect(image_matrix, &mtmn_config);
                    }
                    fr_face = esp_timer_get_time();
                    fr_recognize = fr_face;
                    if (net_boxes || fb->format != PIXFORMAT_JPEG){
                      if(net_boxes){
                        detected = true;
                        if(recognition_enabled){
                          face_id = run_face_recognition(image_matrix, net_boxes);
                        }
                        fr_recognize = esp_timer_get_time();
                        draw_face_boxes(image_matrix, net_boxes, face_id);
                        free(net_boxes->box);
                        free(net_boxes->landmark);
                        free(net_boxes);
                      }
                      if(!fmt2jpg(image_matrix->item, fb->width*fb->height*3, fb->width, fb->height, PIXFORMAT_RGB888, 90, &_jpg_buf, &_jpg_buf_len)){
                        Serial.println("fmt2jpg failed");
                        res = ESP_FAIL;
                      }
                      esp_camera_fb_return(fb);
                      fb = NULL;
                    } else {
                      _jpg_buf = fb->buf;
                      _jpg_buf_len = fb->len;
                    }
                    fr_encode = esp_timer_get_time();
                  }
                  dl_matrix3du_free(image_matrix);
                }
              }
            }
            if(res == ESP_OK){
              size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
              res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
            }
            if(res == ESP_OK){
              res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
            }
            if(res == ESP_OK){
              res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
            }
            if(fb){
              esp_camera_fb_return(fb);
              fb = NULL;
```

```
            _jpg_buf = NULL;
          } else if(_jpg_buf){
            free(_jpg_buf);
            _jpg_buf = NULL;
          }
          if(res != ESP_OK){
            break;
          }
          int64_t fr_end = esp_timer_get_time();
          int64_t ready_time = (fr_ready - fr_start)/1000;
          int64_t face_time = (fr_face - fr_ready)/1000;
          int64_t recognize_time = (fr_recognize - fr_face)/1000;
          int64_t encode_time = (fr_encode - fr_recognize)/1000;
          int64_t process_time = (fr_encode - fr_start)/1000;
          int64_t frame_time = fr_end - last_frame;
          last_frame = fr_end;
          frame_time /= 1000;
          uint32_t avg_frame_time = ra_filter_run(&ra_filter, frame_time);
          Serial.printf("MJPG: %uB %ums (%.1ffps), AVG: %ums (%.1ffps),
%u+%u+%u+%u=%u %s%d\n",
            (uint32_t)(_jpg_buf_len),
            (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,
            avg_frame_time, 1000.0 / avg_frame_time,
            (uint32_t)ready_time, (uint32_t)face_time, (uint32_t)recognize_time,
(uint32_t)encode_time, (uint32_t)process_time,
            (detected)?"DETECTED ":"", face_id
          );
        }
        last_frame = 0;
        return res;
      }
      static esp_err_t cmd_handler(httpd_req_t *req){
        char*  buf;
        size_t buf_len;
        char variable[32] = {0,};
        char value[32] = {0,};
        buf_len = httpd_req_get_url_query_len(req) + 1;
        if (buf_len > 1) {
          buf = (char*)malloc(buf_len);
          if(!buf){
            httpd_resp_send_500(req);
            return ESP_FAIL;
          }
          if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
            if (httpd_query_key_value(buf, "var", variable, sizeof(variable)) == ESP_OK &&
              httpd_query_key_value(buf, "val", value, sizeof(value)) == ESP_OK) {
            } else {
              free(buf);
```

```
            httpd_resp_send_404(req);
            return ESP_FAIL;
        }
    } else {
        free(buf);
        httpd_resp_send_404(req);
        return ESP_FAIL;
    }
    free(buf);
} else {
    httpd_resp_send_404(req);
    return ESP_FAIL;
}
int val = atoi(value);
sensor_t * s = esp_camera_sensor_get();
int res = 0;
if(!strcmp(variable, "framesize")) {
    if(s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s, (framesize_t)val);
}
else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);
else if(!strcmp(variable, "contrast")) res = s->set_contrast(s, val);
else if(!strcmp(variable, "brightness")) res = s->set_brightness(s, val);
else if(!strcmp(variable, "saturation")) res = s->set_saturation(s, val);
else if(!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s, (gainceiling_t)val);
else if(!strcmp(variable, "colorbar")) res = s->set_colorbar(s, val);
else if(!strcmp(variable, "awb")) res = s->set_whitebal(s, val);
else if(!strcmp(variable, "agc")) res = s->set_gain_ctrl(s, val);
else if(!strcmp(variable, "aec")) res = s->set_exposure_ctrl(s, val);
else if(!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);
else if(!strcmp(variable, "vflip")) res = s->set_vflip(s, val);
else if(!strcmp(variable, "awb_gain")) res = s->set_awb_gain(s, val);
else if(!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);
else if(!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);
else if(!strcmp(variable, "aec2")) res = s->set_aec2(s, val);
else if(!strcmp(variable, "dcw")) res = s->set_dcw(s, val);
else if(!strcmp(variable, "bpc")) res = s->set_bpc(s, val);
else if(!strcmp(variable, "wpc")) res = s->set_wpc(s, val);
else if(!strcmp(variable, "raw_gma")) res = s->set_raw_gma(s, val);
else if(!strcmp(variable, "lenc")) res = s->set_lenc(s, val);
else if(!strcmp(variable, "special_effect")) res = s->set_special_effect(s, val);
else if(!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);
else if(!strcmp(variable, "ae_level")) res = s->set_ae_level(s, val);
else if(!strcmp(variable, "face_detect")) {
    detection_enabled = val;
    if(!detection_enabled) {
        recognition_enabled = 0;
    }
}
```

```
    else if(!strcmp(variable, "face_enroll")) is_enrolling = val;
    else if(!strcmp(variable, "face_recognize")) {
        recognition_enabled = val;
        if(recognition_enabled){
            detection_enabled = val;
        }
    }
    else {
        res = -1;
    }
    if(res){
        return httpd_resp_send_500(req);
    }
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, NULL, 0);
}
static esp_err_t status_handler(httpd_req_t *req){
    static char json_response[1024];
    sensor_t * s = esp_camera_sensor_get();
    char * p = json_response;
    *p++ = '{';
    p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);
    p+=sprintf(p, "\"quality\":%u,", s->status.quality);
    p+=sprintf(p, "\"brightness\":%d,", s->status.brightness);
    p+=sprintf(p, "\"contrast\":%d,", s->status.contrast);
    p+=sprintf(p, "\"saturation\":%d,", s->status.saturation);
    p+=sprintf(p, "\"special_effect\":%u,", s->status.special_effect);
    p+=sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
    p+=sprintf(p, "\"awb\":%u,", s->status.awb);
    p+=sprintf(p, "\"awb_gain\":%u,", s->status.awb_gain);
    p+=sprintf(p, "\"aec\":%u,", s->status.aec);
    p+=sprintf(p, "\"aec2\":%u,", s->status.aec2);
    p+=sprintf(p, "\"ae_level\":%d,", s->status.ae_level);
    p+=sprintf(p, "\"aec_value\":%u,", s->status.aec_value);
    p+=sprintf(p, "\"agc\":%u,", s->status.agc);
    p+=sprintf(p, "\"agc_gain\":%u,", s->status.agc_gain);
    p+=sprintf(p, "\"gainceiling\":%u,", s->status.gainceiling);
    p+=sprintf(p, "\"bpc\":%u,", s->status.bpc);
    p+=sprintf(p, "\"wpc\":%u,", s->status.wpc);
    p+=sprintf(p, "\"raw_gma\":%u,", s->status.raw_gma);
    p+=sprintf(p, "\"lenc\":%u,", s->status.lenc);
    p+=sprintf(p, "\"vflip\":%u,", s->status.vflip);
    p+=sprintf(p, "\"hmirror\":%u,", s->status.hmirror);
    p+=sprintf(p, "\"dcw\":%u,", s->status.dcw);
    p+=sprintf(p, "\"colorbar\":%u,", s->status.colorbar);
    p+=sprintf(p, "\"face_detect\":%u,", detection_enabled);
    p+=sprintf(p, "\"face_enroll\":%u,", is_enrolling);
    p+=sprintf(p, "\"face_recognize\":%u", recognition_enabled);
```

```
    *p++ = '}';
    *p++ = 0;
    httpd_resp_set_type(req, "application/json");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, json_response, strlen(json_response));
}
static esp_err_t index_handler(httpd_req_t *req){
    httpd_resp_set_type(req, "text/html");
    httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
    return httpd_resp_send(req, (const char *)index_html_gz, index_html_gz_len);
}
void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();
    httpd_uri_t index_uri = {
        .uri       = "/",
        .method    = HTTP_GET,
        .handler   = index_handler,
        .user_ctx  = NULL
    };
    httpd_uri_t status_uri = {
        .uri       = "/status",
        .method    = HTTP_GET,
        .handler   = status_handler,
        .user_ctx  = NULL
    };
    httpd_uri_t cmd_uri = {
        .uri       = "/control",
        .method    = HTTP_GET,
        .handler   = cmd_handler,
        .user_ctx  = NULL
    };
    httpd_uri_t capture_uri = {
        .uri       = "/capture",
        .method    = HTTP_GET,
        .handler   = capture_handler,
        .user_ctx  = NULL
    };
   httpd_uri_t stream_uri = {
        .uri       = "/stream",
        .method    = HTTP_GET,
        .handler   = stream_handler,
        .user_ctx  = NULL
    };
    ra_filter_init(&ra_filter, 20);
    mtmn_config.min_face = 80;
    mtmn_config.pyramid = 0.7;
    mtmn_config.p_threshold.score = 0.6;
    mtmn_config.p_threshold.nms = 0.7;
```

```
    mtmn_config.r_threshold.score = 0.7;
    mtmn_config.r_threshold.nms = 0.7;
    mtmn_config.r_threshold.candidate_number = 4;
    mtmn_config.o_threshold.score = 0.7;
    mtmn_config.o_threshold.nms = 0.4;
    mtmn_config.o_threshold.candidate_number = 1;
    mtmn_config = mtmn_init_config();
    face_id_init(&id_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);
    Serial.printf("Starting web server on port: '%d'\n", config.server_port);
    if (httpd_start(&camera_httpd, &config) == ESP_OK) {
      httpd_register_uri_handler(camera_httpd, &index_uri);
      httpd_register_uri_handler(camera_httpd, &cmd_uri);
      httpd_register_uri_handler(camera_httpd, &status_uri);
      httpd_register_uri_handler(camera_httpd, &capture_uri);
    }
    config.server_port += 1;
    config.ctrl_port += 1;
    Serial.printf("Starting stream server on port: '%d'\n", config.server_port);
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
      httpd_register_uri_handler(stream_httpd, &stream_uri);
    }
}
```

Додаток В

Код файлу camera_index.h

```
//File: index.html.gz, Size: 3663
#define index_html_gz_len 3663
const uint8_t index_html_gz[] = {
 0x1F, 0x8B, 0x08, 0x08, 0x60, 0x15, 0x36, 0x5C, 0x00, 0x03, 0x69, 0x6E, 0x64, 0x65, 0x78, 0x2E,
 0x68, 0x74, 0x6D, 0x6C, 0x00, 0xDD, 0x5C, 0xEB, 0x72, 0x9B, 0xC8, 0x12, 0xFE, 0x7F, 0x9E, 0x02,
 0x93, 0xDD, 0x08, 0x6A, 0x91, 0x2C, 0xC9, 0x8A, 0xE3, 0x20, 0x0B, 0x1F, 0x5B, 0x76, 0x92, 0xAD,
 0xCA, 0x6D, 0xE3, 0x3D, 0xBB, 0x5B, 0xB5, 0xB5, 0x95, 0x8C, 0x60, 0x90, 0x26, 0x46, 0x8C, 0x02,
 0x83, 0x2E, 0xD1, 0xF2, 0x1C, 0xE7, 0x81, 0xCE, 0x8B, 0x9D, 0x9E, 0x19, 0x40, 0xA0, 0x8B, 0x65,
 0x49, 0x89, 0x94, 0x5A, 0xBB, 0xCA, 0x1A, 0xA0, 0xBB, 0xA7, 0xBB, 0xBF, 0xBE, 0x0C, 0x88, 0xF1,
 0xF9, 0x91, 0x43, 0x6D, 0x36, 0x19, 0x60, 0xA5, 0xC7, 0xFA, 0x9E, 0xF5, 0xAF, 0x73, 0xF9, 0xA1,
 0xC0, 0xCF, 0x79, 0x0F, 0x23, 0x47, 0x0E, 0xC5, 0x61, 0x1F, 0x33, 0xA4, 0xD8, 0x3D, 0x14, 0x84,
 0x98, 0xB5, 0xD4, 0x88, 0xB9, 0xE5, 0x33, 0x75, 0xFE, 0xB2, 0x8F, 0xFA, 0xB8, 0xA5, 0x0E, 0x09,
 0x1E, 0x0D, 0x68, 0xC0, 0x54, 0xC5, 0xA6, 0x3E, 0xC3, 0x3E, 0x90, 0x8F, 0x88, 0xC3, 0x7A, 0x2D,
 0x07, 0x0F, 0x89, 0x8D, 0xCB, 0xE2, 0xC0, 0x20, 0x3E, 0x61, 0x04, 0x79, 0xE5, 0xD0, 0x46, 0x1E,
 0x6E, 0xD5, 0xF2, 0xB2, 0x18, 0x61, 0x1E, 0xB6, 0x6E, 0x6E, 0xDF, 0x9D, 0xD4, 0x95, 0xB7, 0xBF,
 0xD5, 0x1B, 0xA7, 0xD5, 0xF3, 0x63, 0x79, 0x6E, 0x46, 0x13, 0xB2, 0x09, 0x3F, 0xEE, 0x50, 0x67,
 0x32, 0x75, 0x61, 0x9A, 0xB2, 0x8B, 0xFA, 0xC4, 0x9B, 0x98, 0x97, 0x01, 0x08, 0x35, 0x5E, 0x62,
 0x6F, 0x88, 0x19, 0xB1, 0x91, 0x11, 0x22, 0x3F, 0x2C, 0x87, 0x38, 0x20, 0x6E, 0xB3, 0x83, 0xEC,
 0xBB, 0x6E, 0x40, 0x23, 0xDF, 0x31, 0x1F, 0xD5, 0xCE, 0xF8, 0x6F, 0xD3, 0xA6, 0x1E, 0x0D, 0xCC,
 0x47, 0x37, 0xCF, 0xF9, 0x6F, 0x53, 0xC8, 0x09, 0xC9, 0x17, 0x6C, 0xD6, 0x4E, 0x07, 0xE3, 0xB8,
 0x57, 0x9F, 0xE6, 0xCE, 0x9C, 0xC1, 0x99, 0x10, 0xDB, 0x8C, 0x50, 0xBF, 0xD2, 0x47, 0xC4, 0x9F,
 0x3A, 0x24, 0x1C, 0x78, 0x68, 0x62, 0xBA, 0x1E, 0x1E, 0xC7, 0x8F, 0xFA, 0xD8, 0x8F, 0x8C, 0xC2,
 0x75, 0x7E, 0xBE, 0xEC, 0x90, 0x40, 0x9E, 0x33, 0x61, 0xAA, 0xA8, 0xEF, 0x4B, 0xC2, 0x8C, 0xD7,
 0xA7, 0x3E, 0x6E, 0x0A, 0xC2, 0x51, 0x80, 0x06, 0x70, 0xC8, 0x3F, 0x9A, 0x7D, 0xE2, 0x4B, 0x27,
 0x99, 0x27, 0x8D, 0xEA, 0x60, 0x5C, 0x50, 0xFC, 0xE4, 0x94, 0xFF, 0x36, 0x07, 0xC8, 0x71, 0x88,
 0xDF, 0x35, 0xCF, 0xF8, 0x65, 0x1A, 0x38, 0x38, 0x28, 0x07, 0xC8, 0x21, 0x51, 0x68, 0x36, 0xE0,
 0x4C, 0x1F, 0x05, 0x5D, 0x90, 0xC1, 0xE8, 0xC0, 0x2C, 0xD7, 0xAA, 0xB3, 0x13, 0x01, 0xE9, 0xF6,
 0x98, 0xC9, 0xCF, 0xC4, 0x8F, 0x12, 0x6C, 0x0A, 0x66, 0xE4, 0x54, 0x11, 0x8A, 0x20, 0x8F, 0x74,
 0xFD, 0x32, 0x61, 0xB8, 0x1F, 0x9A, 0x21, 0x0B, 0x30, 0xB3, 0x7B, 0xB1, 0x4B, 0xBA, 0x51, 0x80,
 0xA7, 0xA9, 0x02, 0xD5, 0x44, 0x36, 0x0C, 0xCA, 0x23, 0xDC, 0xB9, 0x23, 0xAC, 0x9C, 0x4C, 0xD6,
 0xC1, 0x2E, 0x0D, 0x70, 0x46, 0x50, 0xEE, 0x78, 0xD4, 0xBE, 0x2B, 0x87, 0x0C, 0x05, 0x6C, 0x91,
 0x18, 0xB9, 0x0C, 0x07, 0xF3, 0xB4, 0x18, 0x0C, 0x5E, 0xA0, 0x4C, 0x05, 0x24, 0x87, 0xC4, 0xF7,
 0x88, 0x8F, 0x57, 0x89, 0x95, 0x12, 0x8A, 0xA4, 0xE2, 0x5C, 0x62, 0x86, 0x42, 0xFA, 0xDD, 0xCC,
 0x03, 0x62, 0xD2, 0xA6, 0x74, 0x7C, 0xAD, 0x5A, 0xFD, 0xB1, 0xD9, 0xC3, 0xC2, 0x5F, 0x28, 0x62,
 0xF4, 0x7E, 0x27, 0xF3, 0xD8, 0xF8, 0x77, 0x1F, 0x3B, 0x04, 0x29, 0xDA, 0x0C, 0x3C, 0xE5, 0xAC,
 0x0A, 0x9E, 0xD6, 0x15, 0xE4, 0x3B, 0x8A, 0x46, 0x03, 0x02, 0xDE, 0x46, 0x22, 0x14, 0x3C, 0x38,
 0x03, 0x61, 0x3F, 0xC0, 0xFA, 0x74, 0x1D, 0x0C, 0x49, 0x44, 0xAC, 0x06, 0x62, 0x89, 0x05, 0x7D,
 0x34, 0x2E, 0xE7, 0xAC, 0xE0, 0x87, 0x89, 0x25, 0x90, 0x6A, 0xB6, 0x06, 0x27, 0x87, 0x3D, 0xA5,
 0xAC, 0xF0, 0xD0, 0xD2, 0x13, 0x73, 0x85, 0x89, 0x39, 0x73, 0xFF, 0x29, 0x28, 0xA7, 0x19, 0xFB,
 0xA8, 0x13, 0x31, 0x46, 0xFD, 0x70, 0x8D, 0x9B, 0x3F, 0x45, 0x21, 0x23, 0xEE, 0xA4, 0x9C, 0x80,
 0x62, 0x86, 0x03, 0x04, 0xF5, 0xAA, 0x83, 0xD9, 0x08, 0x63, 0x48, 0x5D, 0x1F, 0x0D, 0x01, 0xEE,
 0x6E, 0xD7, 0xC3, 0x53, 0x3B, 0x0A, 0x42, 0xA8, 0x1C, 0x03, 0x4A, 0x80, 0x32, 0x68, 0x16, 0x00,
 0xC8, 0x13, 0x96, 0xED, 0xCE, 0x94, 0x46, 0x8C, 0xAB, 0x04, 0x2A, 0x52, 0x90, 0x47, 0xD8, 0x04,
 0x46, 0xD2, 0xED, 0xD5, 0xD4, 0xE7, 0xD5, 0x39, 0x1E, 0xD3, 0xEE, 0x61, 0xFB, 0x0E, 0x3B, 0x3F,
 0x15, 0xCB, 0x85, 0x28, 0x35, 0x15, 0xE2, 0x0F, 0x22, 0x56, 0xE6, 0x05, 0x61, 0xB0, 0xC6, 0x1E,
 0xE1, 0x89, 0x64, 0x8A, 0x7A, 0x3D, 0x8B, 0x59, 0xF3, 0xC9, 0x60, 0xAC, 0x54, 0x0B, 0x82, 0x2C,
 0x0F, 0x75, 0xB0, 0x97, 0x89, 0x4B, 0x9C, 0x28, 0xE3, 0x29, 0x09, 0x82, 0x5C, 0xF5, 0xC8, 0x55,
 0xA8, 0xC6, 0xD3, 0x1F, 0x0B, 0x82, 0x14, 0x31, 0x36, 0x0A, 0xA7, 0x42, 0xEC, 0x01, 0x0C, 0xB2,
```

0x20, 0xC2, 0x99, 0x91, 0x59, 0x8B, 0x2B, 0x01, 0xF2, 0xBB, 0x18, 0x00, 0x1C, 0x1B, 0xE9, 0x30, 0x57, 0x52, 0x97, 0x4D, 0x6F, 0x56, 0x15, 0x50, 0x3B, 0x96, 0x40, 0x2E, 0x44, 0x7C, 0x6A, 0x56, 0x8E, 0xBA, 0x56, 0xCF, 0x6A, 0x23, 0x38, 0xBA, 0xE0, 0x0A, 0x5E, 0x35, 0xE7, 0x10, 0x4C, 0x3A, 0x81, 0xEB, 0x16, 0xFB, 0x84, 0xEB, 0x9E, 0x54, 0x4F, 0x1A, 0x73, 0xD9, 0xCF, 0xE7, 0x29, 0xF6, 0x8A, 0x66, 0x86, 0x71, 0xA2, 0xA0, 0xD9, 0xA3, 0x43, 0x1C, 0x4C, 0x8B, 0xA2, 0x1A, 0xCF, 0x1A, 0x4E, 0x7A, 0x1D, 0x41, 0x5C, 0x0E, 0x71, 0x91, 0xA0, 0x5E, 0xB3, 0xEB, 0xB5, 0x84, 0xA0, 0x02, 0x16, 0xA2, 0x8E, 0x87, 0x9D, 0x34, 0xD4, 0x1C, 0xEC, 0xA2, 0xC8, 0x63, 0x05, 0xED, 0x50, 0x95, 0xFF, 0xC6, 0xC2, 0xD7, 0x7F, 0xF2, 0x36, 0xDE, 0x12, 0xBE, 0xFC, 0x6B, 0x9A, 0x26, 0x08, 0x1A, 0x0C, 0x30, 0x82, 0x73, 0x36, 0x96, 0xAD, 0x66, 0xB1, 0xB8, 0x89, 0xB0, 0x58, 0xD2, 0x60, 0xE6, 0xDC, 0x93, 0xA6, 0xFF, 0xE2, 0x5C, 0xA6, 0x4B, 0xED, 0x28, 0x9C, 0x05, 0xF9, 0x12, 0x0A, 0x33, 0x55, 0x27, 0xF4, 0x88, 0x70, 0x63, 0xE4, 0xFB, 0xDC, 0xB6, 0x32, 0x0B, 0x60, 0xE2, 0xE9, 0x12, 0xA5, 0x16, 0xF1, 0xC9, 0xAB, 0x98, 0xB4, 0xEB, 0x22, 0x28, 0xD5, 0x0C, 0x6B, 0x25, 0xA4, 0x30, 0x8F, 0x92, 0x90, 0x3D, 0x40, 0x1F, 0xD6, 0x8B, 0xFA, 0x9D, 0x69, 0xC2, 0x5E, 0x83, 0xDC, 0x90, 0x02, 0x82, 0x6E, 0x07, 0x69, 0x55, 0xA3, 0x6A, 0x9C, 0xC0, 0x1F, 0xBD, 0xE0, 0x30, 0xA9, 0x72, 0xBD, 0xBE, 0xD0, 0x7D, 0x9F, 0xCC, 0xF7, 0xEB, 0x24, 0x80, 0xE6, 0xAC, 0x59, 0x85, 0x4F, 0xA1, 0x71, 0xD7, 0x2A, 0x3C, 0xE0, 0x57, 0x38, 0x7C, 0x9D, 0x53, 0x17, 0xFD, 0xB5, 0xD4, 0x11, 0x7D, 0xFA, 0xA5, 0x2C, 0xF3, 0xEF, 0x60, 0x58, 0xE4, 0x54, 0xD8, 0x37, 0x0E, 0xCB, 0xF5, 0x09, 0xB7, 0xF4, 0x45, 0x55, 0x49, 0xED, 0x2E, 0xCB, 0x6A, 0x02, 0x62, 0x7C, 0x68, 0x21, 0x01, 0xB4, 0x92, 0xE6, 0xC2, 0x99, 0x55, 0x73, 0xBB, 0xC4, 0xF3, 0xCA, 0x1E, 0x1D, 0xCD, 0x55, 0x8F, 0x82, 0x9F, 0xE7, 0xFD, 0x3A, 0xEF, 0xFE, 0x7B, 0x65, 0x47, 0x10, 0x73, 0xDF, 0x40, 0xF6, 0xFE, 0x93, 0x68, 0x06, 0xCA, 0x3D, 0x49, 0xB2, 0xCE, 0xA3, 0x0F, 0x60, 0x5D, 0x74, 0x98, 0xAC, 0x91, 0x71, 0x25, 0x1C, 0x11, 0x58, 0x89, 0xCD, 0x35, 0xA3, 0x01, 0x0D, 0x89, 0x58, 0xE6, 0x05, 0xD8, 0x43, 0xBC, 0xC8, 0x2F, 0xB6, 0xE1, 0xB9, 0xE6, 0x91, 0xBB, 0x94, 0xCA, 0x94, 0x6D, 0xF4, 0x61, 0x4B, 0x87, 0x8A, 0xAC, 0x00, 0x49, 0xBC, 0x0A, 0xE7, 0x15, 0x8A, 0x7B, 0xC1, 0xB7, 0xF5, 0x7B, 0x63, 0x38, 0x09, 0xDC, 0x6E, 0x80, 0x27, 0xA9, 0x58, 0x23, 0xF9, 0x34, 0xE5, 0x4A, 0x6F, 0x79, 0x8F, 0x16, 0x71, 0x2D, 0xAD, 0xAE, 0x34, 0xC2, 0x78, 0x8E, 0x65, 0xD1, 0x23, 0xE9, 0x02, 0x4B, 0x55, 0x17, 0xA0, 0xCF, 0x92, 0x4D, 0xB8, 0x26, 0xC9, 0x41, 0x3E, 0xF4, 0xB0, 0xCB, 0xC4, 0xC2, 0x9B, 0x57, 0xC7, 0x93, 0x42, 0x84, 0x94, 0x67, 0xDD, 0x5B, 0xE2, 0x99, 0xAD, 0x9F, 0x52, 0xDF, 0x2C, 0xA3, 0xE5, 0x31, 0xB5, 0x9C, 0x3C, 0x55, 0x3C, 0x2D, 0xB1, 0xC2, 0x3C, 0x38, 0xD3, 0x97, 0x09, 0x0C, 0x46, 0xE0, 0x3F, 0xB4, 0xFA, 0x29, 0x5F, 0x3F, 0xAF, 0xBE, 0x14, 0x27, 0xCB, 0x9E, 0x85, 0x94, 0x48, 0x5B, 0x6C, 0x2E, 0x0A, 0x1A, 0x73, 0x98, 0xCD, 0x70, 0x5F, 0x58, 0x79, 0xC0, 0x6A, 0xAB, 0x8F, 0xA0, 0x58, 0x72, 0x17, 0xC2, 0x6D, 0x26, 0xD8, 0xB6, 0xE8, 0xDE, 0xD9, 0xF2, 0xAC, 0x76, 0xCA, 0x6F, 0xF6, 0x2A, 0xB6, 0x47, 0xC3, 0x1C, 0x0E, 0xA8, 0x03, 0x9A, 0x44, 0x0C, 0x37, 0xE5, 0x92, 0xEE, 0x49, 0xE2, 0xD4, 0x27, 0xCB, 0xD3, 0x2E, 0x87, 0x41, 0x1E, 0x9A, 0xA2, 0x66, 0x35, 0x7E, 0xAF, 0x93, 0x5F, 0x45, 0x31, 0x3C, 0x86, 0xFE, 0xC6, 0xEF, 0x5B, 0x4C, 0x1B, 0x8B, 0x30, 0xCB, 0xA7, 0x41, 0x6D, 0x71, 0x09, 0x16, 0x57, 0x7A, 0xC4, 0x71, 0xB0, 0x5F, 0xB8, 0x39, 0x8E, 0x67, 0x77, 0xFC, 0xC7, 0xC9, 0x2D, 0xBF, 0x3C, 0x98, 0x3D, 0x9D, 0x38, 0xE7, 0xCF, 0x00, 0xF2, 0x4F, 0x06, 0xE4, 0x92, 0x5F, 0xB1, 0x3D, 0x14, 0x86, 0x2D, 0x95, 0xDF, 0x8B, 0xE7, 0x1E, 0x2E, 0x08, 0x12, 0x87, 0x0C, 0x15, 0xE2, 0xB4, 0x54, 0x8F, 0x76, 0xE9, 0xDC, 0x35, 0x71, 0x5D, 0x2C, 0x86, 0x15, 0x40, 0xB5, 0xA5, 0x16, 0x96, 0xE5, 0xAA, 0xE0, 0x9A, 0x9D, 0x52, 0xAD, 0xC7, 0x8F, 0x9E, 0x3D, 0x7D, 0x7A, 0xDA, 0x7C, 0xEC, 0x77, 0xC2, 0x41, 0xF2, 0xF7, 0x57, 0x71, 0x09, 0x16, 0xBD, 0x8C, 0xC1, 0x42, 0x34, 0x3C, 0x3F, 0x16, 0xD2, 0xE6, 0x34, 0x38, 0x06, 0x15, 0x56, 0x28, 0x95, 0xE4, 0xC6, 0x32, 0xBD, 0x52, 0x92, 0x10, 0x82, 0xB4, 0x83, 0x82, 0x25, 0x24, 0x82, 0x4C, 0xC4, 0xB4, 0x22, 0x4A, 0x9A, 0x2A, 0x22, 0xBB, 0x43, 0xC7, 0xF3, 0xAA, 0x0B, 0x6B, 0x92, 0xB0, 0x4F, 0xA8, 0xB0, 0xB3, 0x4A, 0x20, 0xB0, 0x09, 0x76, 0x7E, 0x33, 0xB2, 0x82, 0x26, 0xD3, 0x2F, 0x71, 0x7B, 0x6E, 0xFD, 0x2F, 0xA7, 0x76, 0x03, 0xD4, 0xC7, 0x3C, 0xDA, 0x93, 0x93, 0xAB, 0xC5, 0xCC, 0x43, 0x90, 0x71, 0xAA, 0xD6, 0x7B, 0x2C, 0x02, 0x17, 0xE0, 0x5D, 0xEA, 0xD6, 0x05, 0x29, 0x32, 0x05, 0x8B, 0xF3, 0xAB, 0xA9, 0x8A, 0xC9, 0x8A, 0xBA, 0x8C, 0x44, 0xBC, 0xAC, 0x51, 0x48, 0x88, 0xA3, 0x03, 0x11, 0x59, 0x43, 0xE4, 0x45, 0xE0, 0xDA, 0x5A, 0x55, 0xB5, 0xFE, 0xF3, 0xC7, 0x8B, 0x4B, 0x0D, 0x92, 0xAC, 0x3A, 0xAE, 0xD5, 0xAB, 0x55, 0xFD, 0xFC, 0x58, 0x92, 0x6C, 0x2C, 0xEB, 0x99, 0x6A, 0xDD, 0x0A, 0x51, 0xF5, 0x33, 0x10, 0x55, 0xAD, 0x37, 0xB6, 0x17, 0x75, 0xA6, 0x5A, 0x42, 0x12, 0x08, 0x19, 0x3F, 0x3D, 0x3D, 0xDB, 0x5E, 0xD0, 0x53, 0xD0, 0xE9, 0x37, 0x90,

0x74, 0x06, 0xD6, 0x9D, 0xEE, 0x62, 0xDC, 0xA9, 0x6A, 0x71, 0x39, 0xA7, 0x8D, 0xEA, 0xB8, 0x71,
0xB6, 0x83, 0x9C, 0x27, 0x6A, 0x72, 0x2B, 0xC9, 0x43, 0x36, 0x1D, 0xA9, 0x56, 0xFB, 0xE7, 0xE7,
0x5A, 0x03, 0x74, 0xAC, 0x3F, 0x3B, 0xDD, 0x5E, 0x76, 0x43, 0xB5, 0x7E, 0xE1, 0x4A, 0x9E, 0xD4,
0x41, 0x50, 0x63, 0x07, 0x25, 0x4F, 0x54, 0xEB, 0xA5, 0x90, 0x04, 0x52, 0xC6, 0xB5, 0xA7, 0x3B,
0xA8, 0x04, 0xE1, 0xF5, 0x8B, 0x90, 0x04, 0xF1, 0xC5, 0xC3, 0xEB, 0x81, 0x92, 0xA0, 0x50, 0x0A,
0xD7, 0xDC, 0x93, 0xA7, 0x8B, 0xD5, 0xA7, 0x70, 0xF9, 0xBE, 0x34, 0xFE, 0x1C, 0x41, 0x4D, 0x67,
0x93, 0x8D, 0x93, 0x38, 0xE1, 0x03, 0x93, 0xE4, 0xE0, 0x61, 0xF9, 0x9B, 0xD3, 0x24, 0x7B, 0x4A,
0xA0, 0x5A, 0xB5, 0xEA, 0x1A, 0x0B, 0x04, 0x6F, 0xBE, 0x0A, 0x0A, 0xE6, 0x82, 0x01, 0xAA, 0x02,
0xA2, 0x44, 0x0E, 0x2B, 0x7D, 0x34, 0x86, 0x18, 0x3D, 0x51, 0x73, 0x79, 0xBD, 0x55, 0x89, 0x58,
0xA2, 0x2D, 0x1A, 0xAB, 0xD6, 0xE9, 0xC9, 0x3A, 0x7F, 0xEF, 0x00, 0x47, 0x47, 0x74, 0x70, 0x1F,
0x87, 0xE1, 0xC6, 0x88, 0xCC, 0x58, 0x55, 0xEB, 0x2A, 0x1B, 0xEF, 0x82, 0x4B, 0xB9, 0xBE, 0x03,
0x2E, 0x39, 0x75, 0x24, 0x34, 0xE5, 0x7A, 0x02, 0x4D, 0x5D, 0x9D, 0x65, 0xC4, 0xD7, 0x04, 0x66,
0x9D, 0xB6, 0xBB, 0xE0, 0xC2, 0x9B, 0x78, 0x80, 0x42, 0xB6, 0x31, 0x2A, 0x29, 0x23, 0x94, 0xB5,
0x64, 0x74, 0x30, 0x44, 0x32, 0x55, 0xFE, 0x01, 0x78, 0x84, 0x88, 0x45, 0x81, 0x78, 0xFA, 0xBE,
0x31, 0x22, 0x33, 0x56, 0xE8, 0x87, 0xD9, 0xF8, 0x60, 0xA8, 0xE4, 0xD4, 0xF9, 0x27, 0xE0, 0x32,
0xC0, 0x36, 0x41, 0xDE, 0x07, 0xEC, 0xBA, 0xD0, 0xB2, 0x36, 0xC7, 0xA6, 0xC0, 0x0E, 0xF8, 0xC8,
0x63, 0xE5, 0x46, 0x1C, 0x6F, 0xBC, 0x46, 0x9C, 0x13, 0xF7, 0xB5, 0x16, 0x8A, 0xD5, 0xE5, 0xEB,
0x96, 0x37, 0x34, 0xD3, 0x73, 0xCB, 0x15, 0x42, 0x0D, 0x84, 0xE0, 0xAE, 0xB8, 0xE7, 0xDB, 0x5A,
0x46, 0x5D, 0xB5, 0x5E, 0x04, 0x68, 0x22, 0xBE, 0x86, 0xDD, 0x65, 0xD1, 0xF3, 0x1E, 0x3B, 0xCA,
0xAF, 0x70, 0x23, 0xB7, 0xCB, 0x0A, 0xEC, 0x45, 0x80, 0xB1, 0xBF, 0x9B, 0x94, 0x27, 0xD0, 0xCC,
0x60, 0xB0, 0x9B, 0x10, 0x58, 0xB0, 0xDE, 0xE2, 0x01, 0x41, 0xDF, 0xC3, 0x82, 0x0B, 0x8D, 0x3A,
0x1B, 0xA7, 0x05, 0xF0, 0xA8, 0xD6, 0xE5, 0xEF, 0x57, 0x1B, 0x17, 0x29, 0xF9, 0xF0, 0xE9, 0x21,
0x11, 0x2E, 0xAB, 0x53, 0xA2, 0xA0, 0xBA, 0x70, 0xB3, 0xB9, 0x3C, 0x73, 0x1E, 0x7A, 0xC3, 0xB9,
0xC4, 0xAE, 0x54, 0x41, 0xF1, 0x7C, 0x46, 0xCD, 0x99, 0xF9, 0x30, 0x1B, 0xBF, 0x5D, 0x05, 0x03,
0x25, 0x3E, 0x74, 0x11, 0xD9, 0xBC, 0xAF, 0xA4, 0x8C, 0x02, 0x29, 0xE5, 0x05, 0x8C, 0xF6, 0x05,
0x97, 0x9C, 0xF6, 0x60, 0x98, 0x25, 0x56, 0x1F, 0x1A, 0x38, 0x50, 0xA4, 0x4F, 0x9D, 0xCD, 0x1F,
0x47, 0x24, 0x7C, 0xAA, 0x05, 0xA8, 0xBD, 0x86, 0xC1, 0xC6, 0x5D, 0x26, 0x15, 0xF0, 0x8D, 0xDB,
0xCB, 0x65, 0xC4, 0xE8, 0x2E, 0x9D, 0xE5, 0x36, 0xF2, 0xFD, 0xC9, 0x2E, 0x6D, 0xA5, 0xED, 0xD1,
0xC8, 0xD9, 0x5E, 0x02, 0xF4, 0x94, 0xB7, 0xAE, 0x4B, 0xEC, 0xED, 0xBB, 0x12, 0x74, 0x94, 0x97,
0xB4, 0xFF, 0x40, 0xFE, 0x6F, 0x5C, 0xC5, 0xB1, 0xBD, 0x79, 0x81, 0xC0, 0x36, 0xA0, 0x78, 0xD3,
0x56, 0x6E, 0x6F, 0xDE, 0xDC, 0xBE, 0x7D, 0xBF, 0x9F, 0xEA, 0x00, 0x73, 0x1E, 0xA8, 0x30, 0x70,
0x6B, 0x0F, 0x5D, 0x13, 0x40, 0x89, 0xFA, 0x36, 0x38, 0xD5, 0x25, 0x50, 0xD7, 0xB7, 0xEF, 0xF6,
0x85, 0x52, 0xFD, 0x70, 0x30, 0xD5, 0xBF, 0x07, 0x9C, 0x3E, 0x78, 0x78, 0x88, 0xBD, 0x2D, 0xB0,
0x92, 0x8C, 0x1C, 0x2F, 0xE5, 0x15, 0x1F, 0x1D, 0xEC, 0x46, 0x2E, 0x53, 0xE5, 0x1F, 0x70, 0x1B,
0x07, 0x51, 0xF1, 0x41, 0x28, 0xBD, 0x4D, 0xF2, 0x48, 0x4E, 0xD5, 0xBA, 0x19, 0x0F, 0x68, 0x18,
0x05, 0x0F, 0x6C, 0xA8, 0xCB, 0x11, 0xD9, 0xE5, 0xC9, 0xE0, 0x4C, 0x15, 0x89, 0x48, 0xFA, 0x68,
0x90, 0x3F, 0xD9, 0xCF, 0x30, 0xA9, 0x57, 0x1B, 0x5F, 0x15, 0x15, 0x2E, 0xFC, 0x5B, 0x02, 0xD3,
0xDD, 0xA2, 0xEF, 0x74, 0x79, 0xDF, 0x79, 0xD1, 0xDE, 0x4F, 0x29, 0xEB, 0x1E, 0xAC, 0xE1, 0x74,
0x0F, 0xDA, 0x70, 0x14, 0xF9, 0x6D, 0x67, 0x06, 0xD3, 0x96, 0x37, 0x11, 0x09, 0x23, 0xDC, 0x3B,
0x6F, 0x73, 0x03, 0x91, 0x7F, 0xA8, 0x3E, 0xDE, 0x25, 0x75, 0x52, 0x35, 0x8A, 0x99, 0x73, 0x32,
0xCB, 0x9B, 0x27, 0x5F, 0x35, 0x6B, 0x4E, 0xD6, 0x6A, 0xBB, 0x4B, 0xD2, 0x70, 0x4B, 0x6C, 0x4C,
0x3C, 0xFE, 0xD2, 0xE3, 0xA6, 0x80, 0xE4, 0x78, 0x25, 0x26, 0x4A, 0x5B, 0x1E, 0xED, 0x82, 0x4D,
0x7D, 0x17, 0x6C, 0xF2, 0x1A, 0x15, 0xE1, 0x39, 0xFD, 0x46, 0x9D, 0xA6, 0x56, 0x3F, 0xFB, 0x96,
0xF0, 0x74, 0x06, 0x9B, 0xD7, 0x34, 0xE0, 0x51, 0xAD, 0xAB, 0x77, 0xFB, 0xA9, 0x69, 0x7C, 0xB2,
0x07, 0xD6, 0xB4, 0x9D, 0x2A, 0x98, 0x30, 0xEA, 0xD0, 0x4B, 0xB1, 0xD1, 0x16, 0x68, 0x8C, 0xB8,
0xE2, 0xBF, 0xEF, 0x09, 0x8D, 0xD1, 0xC3, 0xD1, 0xF8, 0xCA, 0x1D, 0x66, 0xF4, 0x3D, 0xE0, 0x13,
0xA0, 0xD1, 0x87, 0x6E, 0x1F, 0x6D, 0x8C, 0x51, 0xC2, 0xA7, 0x5A, 0xEF, 0xD1, 0x48, 0x79, 0xF1,
0xFA, 0x72, 0x2F, 0x58, 0xA5, 0x93, 0x1E, 0x06, 0xAF, 0xCC, 0xE4, 0x43, 0x63, 0xE6, 0x61, 0x7F,
0xF3, 0xA4, 0xE2, 0x4C, 0xAA, 0xF5, 0x0A, 0xFB, 0xA1, 0xD2, 0xA6, 0x41, 0xB2, 0xED, 0x68, 0x2F,
0xA8, 0x89, 0x99, 0x0F, 0x03, 0x99, 0x34, 0xFA, 0xD0, 0x78, 0xF5, 0xFA, 0x24, 0x08, 0x68, 0xB0,
0x31, 0x64, 0x09, 0x9F, 0x6A, 0xBD, 0x2C, 0xBF, 0x16, 0xA3, 0xBD, 0xC0, 0x95, 0xCE, 0x7A, 0x18,
0xC4, 0x32, 0x9B, 0x0F, 0x0D, 0xDA, 0xD0, 0xF5, 0xC8, 0x60, 0x63, 0xC8, 0x04, 0x97, 0x6A, 0xFD,

0x56, 0x7E, 0x0E, 0x9F, 0x7B, 0x81, 0x4B, 0xCE, 0x78, 0x18, 0xB0, 0x12, 0x6B, 0x0F, 0x0D, 0x95, 0x63, 0x8F, 0x36, 0x06, 0x0A, 0x78, 0x54, 0xEB, 0xBA, 0xFD, 0xBB, 0xA2, 0x5D, 0xD3, 0x91, 0xCF, 0x5F, 0xFC, 0x53, 0x6E, 0xDE, 0xE8, 0x7B, 0x41, 0x8C, 0x4F, 0x7D, 0x18, 0xBC, 0x84, 0xD1, 0x87, 0x46, 0x4B, 0xBC, 0x04, 0xDC, 0x41, 0x9B, 0x97, 0xC3, 0x94, 0x91, 0xBF, 0xFB, 0x02, 0x23, 0xE5, 0x0A, 0xED, 0xA7, 0x20, 0x66, 0xF3, 0xEE, 0x63, 0xD1, 0x3E, 0x33, 0xF2, 0xD0, 0x38, 0xB9, 0xC8, 0xC6, 0x1F, 0x1C, 0xCC, 0xB6, 0x79, 0xF1, 0x22, 0xC7, 0xAB, 0x5A, 0xCF, 0xE1, 0x40, 0xB9, 0x16, 0x07, 0xFB, 0x5A, 0x72, 0xE4, 0xE7, 0xDF, 0x07, 0x6A, 0x05, 0x7B, 0xBF, 0x0B, 0xE0, 0x60, 0x81, 0x47, 0xBB, 0xFE, 0x56, 0xEF, 0x53, 0x17, 0xD8, 0x13, 0xF8, 0xDE, 0xCB, 0xE3, 0xFD, 0x02, 0x38, 0x53, 0x62, 0x6F, 0x18, 0xE6, 0xEC, 0xDE, 0x07, 0x8C, 0xE9, 0x66, 0x04, 0xF1, 0x58, 0x40, 0xEE, 0x41, 0x5E, 0x87, 0x94, 0x24, 0x93, 0x8F, 0x6E, 0x30, 0x2B, 0x87, 0x8C, 0x78, 0x9E, 0x6A, 0xBD, 0xC0, 0x4C, 0xB9, 0xE5, 0xC3, 0xF3, 0x63, 0x49, 0xF0, 0x70, 0x29, 0xC9, 0x0B, 0xFF, 0x7C, 0xDF, 0x38, 0xEA, 0xAB, 0xD6, 0x2D, 0xDF, 0x44, 0x0D, 0xB2, 0xF8, 0xD1, 0xE6, 0xC2, 0x84, 0x13, 0xB1, 0x1F, 0x50, 0x50, 0x2A, 0x03, 0x29, 0xD9, 0xAA, 0xAA, 0x2A, 0xE9, 0x28, 0x77, 0xCE, 0xBA, 0x11, 0xC4, 0x0A, 0x8F, 0xB2, 0xF5, 0xD3, 0xF1, 0x6F, 0x61, 0xED, 0xD5, 0x5F, 0xD6, 0x9E, 0x1F, 0xFB, 0x68, 0x89, 0xBB, 0x57, 0xA0, 0x70, 0x2E, 0x77, 0xB1, 0xAF, 0x10, 0x95, 0x6D, 0xA6, 0x10, 0x9E, 0x98, 0xED, 0xA7, 0xC9, 0xCC, 0x9A, 0xDB, 0x67, 0x93, 0x3E, 0xB0, 0x7D, 0x58, 0xD2, 0x8A, 0x1D, 0x37, 0x49, 0x3F, 0xE4, 0xC3, 0xCC, 0xFD, 0xFF, 0xFB, 0xEF, 0xBA, 0x98, 0x21, 0xFD, 0x6E, 0x4E, 0x31, 0x55, 0x09, 0x03, 0xBB, 0xA5, 0xAE, 0xDA, 0x9A, 0xB1, 0xC2, 0xF2, 0xE3, 0x65, 0xA6, 0xCF, 0x11, 0x2F, 0xF1, 0xF5, 0x79, 0x68, 0x07, 0x64, 0xC0, 0xAC, 0x7F, 0x39, 0xD4, 0x8E, 0xFA, 0xD8, 0x67, 0x15, 0xE4, 0x38, 0x37, 0x43, 0x18, 0xBC, 0x22, 0x21, 0xC3, 0xE0, 0x05, 0xAD, 0x74, 0xFD, 0xF6, 0x75, 0x5B, 0x6E, 0x51, 0x79, 0x45, 0x91, 0x83, 0x9D, 0x92, 0xE1, 0x46, 0xBE, 0x90, 0xA3, 0xE9, 0xD3, 0x74, 0xA8, 0x74, 0xB4, 0x2B, 0x7D, 0xEA, 0x41, 0xD0, 0xB6, 0x9B, 0xB2, 0x3C, 0x68, 0x57, 0x15, 0x9E, 0xE3, 0xFA, 0xD4, 0x46, 0x21, 0x2E, 0xA5, 0x89, 0x5E, 0x32, 0xDB, 0xAD, 0xAB, 0x4A, 0xB2, 0xF6, 0xB9, 0xA8, 0xF1, 0x0D, 0x4F, 0x60, 0xF4, 0x5D, 0x53, 0x10, 0x89, 0x47, 0x8A, 0x25, 0x53, 0x8C, 0xE5, 0x97, 0xF3, 0x65, 0xEA, 0x63, 0xC9, 0x22, 0x1E, 0x5C, 0xE6, 0x89, 0x65, 0x64, 0xA5, 0xD4, 0x51, 0xA7, 0x4F, 0x18, 0xA7, 0x2C, 0xD5, 0x4A, 0x09, 0x55, 0x52, 0x4A, 0xCC, 0x00, 0xB3, 0x28, 0xF0, 0x9B, 0x31, 0x00, 0x1B, 0x32, 0xE5, 0xBA, 0xF5, 0xF1, 0x87, 0xA9, 0x1D, 0x1F, 0x8B, 0x97, 0x5D, 0xA9, 0x77, 0x31, 0x44, 0x41, 0xEB, 0x87, 0xE9, 0x55, 0x85, 0x38, 0xF1, 0x63, 0x98, 0x03, 0xC6, 0xED, 0xF8, 0x63, 0xD3, 0xE5, 0xFF, 0x71, 0x41, 0xBB, 0xD6, 0x2B, 0xAC, 0x87, 0x7D, 0xED, 0xA6, 0x65, 0x4D, 0x39, 0x37, 0xF5, 0x70, 0xC5, 0xA3, 0x5D, 0xED, 0x63, 0x80, 0x3F, 0x47, 0x18, 0x84, 0x31, 0xAA, 0xFC, 0x30, 0xBD, 0x8E, 0x15, 0x97, 0xF8, 0x24, 0xEC, 0x61, 0xC7, 0x50, 0x42, 0x86, 0x58, 0x14, 0x9A, 0x70, 0xFA, 0xA6, 0x22, 0xC7, 0xF1, 0x47, 0x3D, 0xD6, 0x63, 0x98, 0x46, 0xB1, 0x5B, 0x99, 0x97, 0x3D, 0x6A, 0x8B, 0x57, 0x3A, 0x2B, 0x34, 0x20, 0x5D, 0xE2, 0x37, 0xA5, 0x6E, 0xB8, 0x75, 0x05, 0x33, 0x81, 0x7B, 0x78, 0x48, 0x71, 0x00, 0x38, 0x1A, 0x5A, 0x49, 0xC6, 0x61, 0x49, 0x8F, 0x0D, 0x77, 0x81, 0x20, 0xC0, 0x7D, 0x3A, 0xC4, 0x79, 0x9A, 0xEE, 0x72, 0x21, 0x69, 0x7E, 0x96, 0x74, 0xE3, 0x2A, 0xDB, 0x6B, 0xDE, 0x3A, 0xAA, 0xC6, 0x46, 0x6F, 0xA5, 0xD0, 0x15, 0x3C, 0xB5, 0xD8, 0x20, 0x2D, 0xED, 0xCA, 0x68, 0x1B, 0xD7, 0x3A, 0x70, 0x5E, 0xB7, 0x8E, 0x34, 0x3F, 0xF2, 0xBC, 0xA3, 0xD6, 0xB5, 0xFE, 0xF7, 0xDF, 0xD7, 0x4D, 0x1E, 0x04, 0x37, 0xCD, 0x19, 0xE2, 0xAD, 0x56, 0x4B, 0x86, 0xC2, 0x05, 0x38, 0x32, 0xC3, 0xDE, 0x68, 0xB7, 0x8E, 0x8E, 0xDA, 0x46, 0x76, 0xDC, 0x6A, 0xEB, 0xA6, 0xB8, 0x2E, 0x80, 0x36, 0x92, 0x4F, 0x38, 0x6B, 0x5C, 0x3F, 0x7E, 0x7C, 0x73, 0xD4, 0x6A, 0xB5, 0x2F, 0x78, 0x88, 0x99, 0x47, 0x70, 0xA8, 0x95, 0x10, 0xB6, 0xA5, 0x5C, 0xE2, 0x5C, 0xB4, 0x2F, 0xB0, 0x36, 0xD4, 0x4D, 0x97, 0xFF, 0x29, 0xA1, 0x6E, 0xFE, 0x82, 0xE6, 0x6A, 0x4C, 0x37, 0xB0, 0x16, 0xEA, 0x20, 0x1C, 0xF3, 0xB1, 0x2B, 0xC6, 0xA5, 0xF4, 0xAD, 0xA4, 0x1C, 0xAD, 0xAB, 0x8D, 0x75, 0x13, 0xF3, 0x3F, 0xA5, 0x62, 0xE3, 0x48, 0x69, 0x60, 0xDE, 0xF6, 0x45, 0x4F, 0xF3, 0x75, 0xB3, 0x0B, 0x7F, 0x74, 0x3D, 0x6E, 0x66, 0x70, 0x42, 0x34, 0x04, 0x93, 0x5B, 0x11, 0xB1, 0x34, 0xB8, 0xF4, 0x3C, 0xAD, 0x24, 0x77, 0xE0, 0x95, 0xF4, 0x0A, 0x74, 0xA2, 0x1B, 0xC4, 0xB3, 0x41, 0xF8, 0x98, 0xFA, 0xB6, 0x47, 0xEC, 0xBB, 0x96, 0xC6, 0x1D, 0x87, 0x21, 0x45, 0xE4, 0xDE, 0xE0, 0x37, 0xD4, 0xC1, 0x7A, 0x1C, 0x83, 0x7A, 0x22, 0xEE, 0x64, 0x84, 0xCA, 0xF0, 0xF9, 0x98, 0xC4, 0x60, 0x96, 0x73, 0x90, 0x66, 0x32, 0xA2, 0x95, 0xAB, 0xCA, 0xA7, 0x90, 0x27, 0x61, 0xBC, 0x84, 0xE4, 0x3E, 0xD5, 0x8A, 0x3D, 0x36, 0xA7, 0x63, 0x1B, 0x94, 0x22, 0x1A, 0x80, 0xF2, 0x67, 0x1B, 0xEC, 0xFD, 0xCB, 0x38, 0xAA, 0xF1, 0xD0, 0xD5, 0x93, 0xE8, 0xFC, 0x34, 0x0B, 0x5F, 0xE8, 0x53, 0x37, 0x1E, 0xE6, 0xC3, 0xAB, 0xC9,

```
0xCF, 0x10, 0x5C, 0xB2, 0x72, 0x41, 0x98, 0xDC, 0xAD, 0xA3, 0x99, 0x95, 0x57, 0xA0, 0xF6, 0x56,
0x53, 0x67, 0x9D, 0x10, 0xC8, 0xFA, 0xAB, 0xC9, 0x0A, 0xAD, 0x0E, 0x48, 0xFD, 0xD5, 0xA4, 0xB9,
0x46, 0x06, 0x84, 0x74, 0x35, 0x61, 0xBE, 0x7C, 0x03, 0xE5, 0x40, 0x82, 0x35, 0x22, 0xBE, 0x43,
0x47, 0x90, 0xD3, 0x74, 0xA0, 0x81, 0x4A, 0x15, 0xE2, 0x83, 0x0D, 0x2F, 0x7F, 0x7D, 0xFD, 0xAA,
0x55, 0xCA, 0x37, 0xD8, 0x52, 0x6C, 0x7C, 0x96, 0x0C, 0x9F, 0x2A, 0xBC, 0x8E, 0x73, 0x28, 0x7F,
0x2A, 0x99, 0x67, 0xB5, 0x12, 0x07, 0x94, 0x53, 0x7C, 0x84, 0x18, 0xBC, 0x5B, 0x90, 0x40, 0x07,
0x99, 0x80, 0xA6, 0x57, 0x0C, 0x13, 0x3E, 0xDF, 0x4C, 0x18, 0x54, 0x2E, 0x34, 0x00, 0xF8, 0xF1,
0xC5, 0x07, 0xBB, 0x03, 0xD5, 0xEA, 0x1A, 0x31, 0x5C, 0xF1, 0xE9, 0x08, 0xC2, 0x40, 0x4A, 0x8E,
0x0D, 0xBA, 0xC8, 0x8F, 0xC5, 0x85, 0x7E, 0xF1, 0x82, 0x84, 0xF5, 0xAA, 0x38, 0x3D, 0x04, 0x7B,
0x4E, 0xB5, 0xE6, 0xD5, 0x05, 0xB0, 0x9B, 0x9F, 0x41, 0xBA, 0xE1, 0x17, 0xB9, 0x3B, 0x90, 0x04,
0xB1, 0xB1, 0x55, 0x9C, 0x65, 0xB9, 0xD0, 0xE3, 0x05, 0x5F, 0x88, 0xE3, 0xB9, 0x9D, 0x45, 0x5A,
0xB0, 0x1A, 0x1C, 0x9E, 0xDF, 0xBA, 0x11, 0xDE, 0x4B, 0x90, 0xFB, 0x66, 0x15, 0x68, 0xD9, 0x3D,
0x41, 0x36, 0xFF, 0xBD, 0x5F, 0x49, 0x6F, 0x06, 0x45, 0xBD, 0xC0, 0xCC, 0x40, 0x37, 0x82, 0xAC,
0x63, 0xAD, 0xA8, 0x28, 0x71, 0xA2, 0x79, 0x74, 0x8F, 0x62, 0x98, 0x6B, 0x3E, 0xBC, 0x97, 0x20,
0xFF, 0x4E, 0x05, 0xE8, 0x12, 0x2D, 0xE8, 0x12, 0xE9, 0x46, 0x94, 0xE9, 0x92, 0x95, 0xBD, 0x74,
0xF6, 0xD1, 0x3D, 0xC2, 0xD3, 0x82, 0xA7, 0x1B, 0xE3, 0xD5, 0x54, 0x85, 0x57, 0x24, 0x41, 0x81,
0xD1, 0x82, 0x02, 0x23, 0xDD, 0x18, 0x65, 0x0A, 0x64, 0x25, 0x33, 0x55, 0x60, 0xB2, 0x26, 0xFD,
0xE4, 0x0D, 0x15, 0xE8, 0xF0, 0x65, 0x0D, 0xE1, 0xAC, 0xF8, 0xEA, 0xC6, 0xE5, 0x3D, 0xB4, 0xE9,
0x1E, 0x4F, 0xD0, 0xF5, 0x72, 0x41, 0xD7, 0x4B, 0xDD, 0x78, 0x72, 0x7E, 0x29, 0x1B, 0x09, 0x14,
0x6F, 0xA2, 0x4D, 0x78, 0x45, 0x33, 0x88, 0xF6, 0x85, 0x7F, 0x42, 0xF0, 0x4E, 0xE6, 0x58, 0x92,
0xBA, 0x9A, 0x31, 0x5D, 0x68, 0xC8, 0xC3, 0x01, 0xD3, 0x4A, 0xEF, 0x3C, 0x0C, 0xAB, 0x8C, 0xE4,
0xAD, 0x4B, 0xA5, 0xFD, 0xF3, 0x73, 0x85, 0x06, 0x8A, 0xF8, 0x0F, 0x03, 0x4A, 0x90, 0xED, 0x50,
0x55, 0xE4, 0x26, 0x72, 0x05, 0xF3, 0x7F, 0xCB, 0x01, 0x21, 0xA5, 0xB0, 0x1E, 0x09, 0x15, 0x17,
0xF3, 0xFD, 0x1B, 0xF8, 0x88, 0x63, 0x4F, 0x89, 0xA3, 0x24, 0x5A, 0xE8, 0x26, 0x3F, 0xD2, 0x3A,
0xDA, 0x44, 0x37, 0x8E, 0x26, 0xA9, 0x47, 0x41, 0x4B, 0xDE, 0x5B, 0x32, 0x15, 0x41, 0xC7, 0x2F,
0x07, 0xD1, 0xF1, 0x4B, 0x41, 0xC7, 0x2F, 0x00, 0xD8, 0x2C, 0x03, 0x7A, 0x52, 0x43, 0x30, 0xA3,
0xAA, 0x27, 0xBD, 0x10, 0x5A, 0x57, 0x33, 0xBF, 0xCC, 0x4C, 0x16, 0x95, 0xF2, 0x48, 0x6E, 0xD7,
0x3E, 0x3F, 0x16, 0xFF, 0x6A, 0xEE, 0xFF, 0x81, 0x09, 0x07, 0x8B, 0x81, 0x4E, 0x00, 0x00
};
```

Додаток Г

Код файлу CameraWebServer.ino

```
#include "esp_camera.h"
#include <WiFi.h>

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_M5STACK_PSRAM
#define CAMERA_MODEL_AI_THINKER
const int WiFi_led = 14;
const char* ssid = "jervisdesktop";
const char* password = "joydip98";
int FACE_ID;
String student, student1, student2;
const char *server = "maker.ifttt.com";
const char* resource = "/trigger/attendance/with/key/hUAAAz0AVvc6-
NW1UmqWXXv6VQWmpiGFxx3sV5rnaM9";
#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    21
#define SIOD_GPIO_NUM    26
#define SIOC_GPIO_NUM    27
#define Y9_GPIO_NUM      35
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      19
#define Y4_GPIO_NUM      18
#define Y3_GPIO_NUM       5
#define Y2_GPIO_NUM       4
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22
#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     25
#define SIOC_GPIO_NUM     23
#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
```

```
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       32
#define VSYNC_GPIO_NUM    22
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21
#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#else
#error "Camera model not selected"
#endif
void startCameraServer();
void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
```

```
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  //drop down frame size for higher initial frame rate
  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s, FRAMESIZE_QVGA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  while (WiFi.status() == WL_CONNECTED) {
    digitalWrite(WiFi_led, HIGH);
    Serial.print("LED High");
  }
  startCameraServer();
  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}
void loop() {
  // put your main code here, to run repeatedly:
  Serial.print("Face ID: ");
  Serial.print(FACE_ID);
  if (FACE_ID == 1){
      student = "Thor";
      makeIFTTTRequest();
      delay(10000);
  }
  if (FACE_ID == 2){
```

```
      student = "Ashish Mishra";
      makeIFTTTRequest();
      delay(10000);
    }
    if (FACE_ID == 3){
      student = "Joydip Dutta";
      makeIFTTTRequest();
      delay(10000);
    }
    if (FACE_ID == 4){
      student = "Ashish Choudhary";
      makeIFTTTRequest();
      delay(10000);
    }
  }
  void makeIFTTTRequest() {
    Serial.print("Connecting to ");
    Serial.print(server);
    WiFiClient client;
    int retries = 5;
    while(!!!client.connect(server, 80) && (retries-- > 0)) {
      Serial.print(".");
    }
    Serial.println();
    if(!!!client.connected()) {
      Serial.println("Failed to connect...");
    }
    Serial.print("Request resource: ");
    Serial.println(resource);
    // Temperature in Celsius
   // String jsonObject = String("{\"value1\":\"") + student + "\"}";
   String jsonObject = String("{\"value1\":\"") + student + "\",\"value2\":\"" + student1
                    + "\",\"value3\":\"" + student2 + "\"}";
    // Comment the previous line and uncomment the next line to publish temperature readings in
  Fahrenheit
    /*String jsonObject = String("{\"value1\":\"") + (1.8 * bme.readTemperature() + 32) +
  "\",\"value2\":\""
                    + (bme.readPressure()/100.0F) + "\",\"value3\":\"" + bme.readHumidity() + "\"}";*/
    client.println(String("POST ") + resource + " HTTP/1.1");
    client.println(String("Host: ") + server);
    client.println("Connection: close\r\nContent-Type: application/json");
    client.print("Content-Length: ");
    client.println(jsonObject.length());
    client.println();
    client.println(jsonObject);
    int timeout = 5 * 10; // 5 seconds
    while(!!!client.available() && (timeout-- > 0)){
      delay(100);
```

```
  }
  if(!!!client.available()) {
    Serial.println("No response...");
  }
  while(client.available()){
    Serial.write(client.read());
  }
  Serial.println("\nclosing connection");
  client.stop();
}
```

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва проекту: Мікропроцесорна система розпізнавання обличчя засобами IoT

Тип проекту: бакалаврський дипломний проект
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 94% Схожість 6%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор проекту _____ Ковальчук Д.О.
(підпис) (прізвище, ініціали)

Керівник проекту _____ Богомолов С.В.
(підпис) (прізвище, ініціали)