Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматизації, комп'ютерно інтегрованих технологій та систем
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення підсистеми автоматизованого доступу до виробничих приміщень та обліку робочого часу персоналу
(тема)

Виконав:
здобувач 4 року навчання
групи АКТСІ-22-1
Ігор ГАРАСИМЧУК
(власне ім'я, прізвище)

Спеціальність 151 Автоматизація та комп'ютерно інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(повна назва освітньої програми)

Керівник ст. викл. Дмитро ГУРІН
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри КІТАРБІ _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2026 р.

Я, Гарасимчук Ігор Ярославович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

“06” липня 2026 р.  Ігор ГАРАСИМЧУК

Харківський національний університет радіоелектроніки

Факультет Автоматизації, комп'ютерно-інтегрованих технологій та систем

Кафедра Комп'ютерно-інтегровані технології, автоматизації, робототехніки та безпекової інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 151 – Автоматизація та комп'ютерно інтегровані технології
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Системна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

« » _____ 20 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Гарасимчук Ігорю Ярославовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення підсистеми автоматизованого доступу до виробничих приміщень та обліку робочого часу персоналу
затверджена наказом університету від "04" липня 2026 р. № 473 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 16.07.2026 р.
3. Вихідні дані до роботи 3.1 Мікроконтролер ESP32-S3;
3.2 Периферія OV2640 та PN532;
3.3 Енергоспоживання 234 мА;
3.5 Середовище розробки: ESP-IDF v6.0.0.
4. Перелік питань, що потрібно опрацювати в роботі 4.1 Дослідження;
існуючих рішень автоматизації та контролю доступом;
4.2 Розробка макету підсистеми;
4.3 Розробка програмного забезпечення підсистеми;
4.4 Тестування системи в різних умовах;
4.5 Розглянути питання з охорони праці;
4.6 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій. (п.5 включається до завдання за рішенням випускної кафедри) Графічний матеріал у вигляді презентації – 15 арк. ф. А4

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз технічного завдання та існуючих рішень	27.04.26-04.05.26	виконано
2	Вибір та обґрунтування компонентів	05.05.26-12.05.26	виконано
3	Збірка макету	13.05.26-31.05.26	виконано
4	Розробка програмного забезпечення	01.06.26-12.06.26	виконано
5	Експериментальні дослідження	13.06.26-19.06.26	виконано
6	Охорона праці	20-06.26-26.06.26	виконано
7	Подання роботи на перевірку Інтернет сервісом StrikePlagiarism	06.07.26-07.07.26	виконано
8	Оформлення пояснювальної записки	08.07.26-08.07.26	виконано
9	Подання роботи на рецензію	09.07.26-10.07.26	виконано
10	Подання роботи на підпис зав. кафедри	14.07.26-15.07.26	виконано
11	Подання кваліфікаційної роботи в ЕК	15.07.26-16.07.26	виконано

Дата видачі завдання 21.04.2026 р.

Здобувач _____ Ігор ГАРАСИМЧУК
(підпис) (власне ім'я, прізвище)

Керівник роботи _____ ст. викл. Дмитро ГУРІН
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 70 с., 14 табл., 59 рис., 39 джерел.

АВТОМАТИЗОВАНИЙ ДОСТУП, ОБЛІК ЧАСУ, ESP32-S3, PN532, OV2640, PYTHON, MQTT, HTTP, ЕЛЕКТРОМАГНІТНИЙ ЗАМОК

Об'єкт розробки – процес автоматизованого контролю доступу та моніторингу робочого часу на підприємстві.

Предмет розробки – апаратно-програмна реалізація підсистеми доступу та обліку часу на підприємстві.

Мета розробки – підвищення ефективності контролю пропускного режиму та обліку робочого часу працівників шляхом розроблення автоматизованої системи контролю доступу.

В першому розділі проведений аналіз принципів будови СКУД та існуючих методів ідентифікації.

В другому розділі була підібрана компонента база та розроблений макет.

В третьому розділі була розроблена програмна частина мікроконтролера та серверу.

В четвертому розділі підсистема була протестована в різних умовах та проведена перевірка лабораторії відповідно до умов охорони праці.

Також отримані результати роботи можна віднести до Цілі сталого розвитку 9 “Промисловість, інновації та інфраструктура”, а семе п 9.1 “Розвивати якісну, надійну, сталу та доступну інфраструктуру, яка базується на використанні інноваційних технологій, у т. ч. екологічно чистих видів транспорту”.

ABSTRACT

Explanatory Note: 70 pages, 14 tables, 59 figures, 39 references.

AUTOMATED ACCESS CONTROL, TIME TRACKING, ESP32-S3, PN532, OV2640, PYTHON, MQTT, HTTP, ELECTROMAGNETIC LOCK

The scope of this project is the process of automated access control and work time monitoring at the enterprise.

The focus of this project is the hardware and software implementation of an access control and time tracking subsystem at the enterprise.

The goal of this project is to improve the efficiency of access control and employee time tracking by developing an automated access control system.

The first chapter analyzes the principles of access control system architecture and existing identification methods.

In the second chapter, the component base was selected and a prototype was developed.

In the third chapter, the software for the microcontroller and server was developed.

In the fourth chapter, the subsystem was tested under various conditions, and a laboratory inspection was conducted in accordance with occupational safety requirements.

Furthermore, the results of this work align with Sustainable Development Goal 9, “Industry, Innovation, and Infrastructure,” specifically Target 9.1: “Develop high-quality, reliable, sustainable, and accessible infrastructure based on the use of innovative technologies, including environmentally friendly modes of transport.”

ЗМІСТ

Перелік скорочень 10

Вступ 11

1 Аналіз сучасних систем контролю доступу та обліку робочого часу 12

1.1 Аналіз принципів побудови та класифікація СКУД 12

1.2 Аналіз методів та засобів ідентифікації персоналу 13

1.3 Аналіз готових комерційних рішень 16

1.4 Висновки до розділу 18

2 Розробка підсистеми автоматизованого доступу та обліку робочого часу 19

2.1 Розробка загальної структурної схеми підсистеми 19

2.2 Вибір та обґрунтування апаратних модулів 20

2.2.1 Центральний мікроконтролер 21

2.2.2 Модуль RFID-ідентифікації 27

2.2.3 Модуль камери 28

2.2.4 Виконавчий пристрій 30

2.2.5 Блок індикації 32

2.2.6 Внутрішня пам'ять 33

2.2.7 Живлення системи 34

2.3 Математичне моделювання та аналіз стійкості підсистеми живлення 35

2.4 Розробка схеми підключення компонентів 38

2.6 Збірка та тестування макету підсистеми 41

2.7 Висновки до розділу 42

3 Розробка програмного забезпечення 43
3.1 Розробка прошивки для мікроконтролера ESP32 43
3.1.1 Розробка алгоритму функціонування підсистеми 43
3.1.2 Визначення та підключення необхідних бібліотек 45
3.1.3 Визначення структурної схеми 47
3.1.4 Розробка алгоритму роботи 48
3.1.5 Розробка програмного коду 50
3.2 Розробка програми для серверу 61
3.2.1 Визначення та підключення необхідних бібліотек 61
3.2.2 Розробка алгоритму 61
3.2.3 Налаштування середовища та розробка програмного коду 62
3.3 Висновки до розділу 68
4 Експерементальні дослідження 69
4.1 Експеримент з енергоефективністю 69
4.2 Експеримент з освітленням 70
4.3 Експеримент з пропускної здатності 75
4.4 Охорона праці 75
4.4.1 Характеристика робочого середовища 76
4.4.2 Мікроклімат і освітлення 76
4.4.3 Ідентифікація шкідливих виробничих факторів 76
4.4.4 Електробезпека 77
4.4.5 Пожежна безпека 77
4.5 Висновки до розділу 78
Висновки 80

Перелік джерел посилань 81

Додаток А Апробація результатів 86

Додаток Б Головна частина коду мікроконтролера 87

Додаток В Головна частина коду серверу 98

Додаток Г Демонстраційний матеріал 101

ПЕРЕЛІК СКОРОЧЕНЬ

ДСТУ – Державний стандарт України;

КІТАРБІ – кафедра комп’ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії;

ПК – персональний комп’ютер;

САР – система автоматичного регулювання;

СКУД – система контролю та управління доступом;

СУБД – система управління базами даних;

ТАУ – теорія автоматичного управління;

ХНУРЕ – Харківський національний університет радіоелектроніки;

ШІМ – широтно-імпульсна модуляція;

I2C – Inter-Integrated Circuit (апаратний протокол послідовного зв’язку);

PSRAM – Pseudo-Static RAM (тип оперативної пам’яті);

RFID – Radio Frequency Identification (радіо частотна ідентифікація);

SDK – Software Development Kit (набір інструментів, бібліотек, документації та прикладів коду).

ВСТУП

У наш час автоматизовані системи важливі для ефективності та безпеки на підприємстві.

Автоматизовані підсистеми доступу дозволяють мінімізувати вплив людського фактору та дають можливість збирання даних в реальному часі. Аналіз працездатності таких підсистем в реальних умовах виявляє їх сильні та слабкі сторони.

Об'єкт розробки – процес автоматизованого контролю доступу та моніторингу робочого часу на підприємстві.

Предмет розробки – апаратно-програмні засоби реалізації підсистеми доступу до приміщень та облік робочого часу.

Мета розробки – підвищення ефективності контролю пропускного режиму та обліку робочого часу працівників шляхом розроблення автоматизованої системи контролю доступу.

Для досягнення поставленої мети необхідно виконати такі завдання:

– провести аналітичний огляд існуючих систем контролю доступу та методів ідентифікації;

– вибрати компоненти для створення системи керування доступом;

– розробити схеми підключення та алгоритму роботи системи;

– створити прототип підсистеми для проведення експериментальних досліджень;

– створити керуючу програму;

– провести тестування та здійснити аналіз розробленої системи.

Пояснювальну записку кваліфікаційної роботи оформлено згідно з вимогами ДСТУ 3008 – 15, а також на основі методичних вказівок [2] та навчального посібника з дипломного проектування [1].

1 АНАЛІЗ СУЧАСНИХ СИСТЕМ КОНТРОЛЮ ДОСТУПУ ТА ОБЛІКУ РОБОЧОГО ЧАСУ

1.1 Аналіз принципів побудови та класифікація СКУД

Система контролю та управління доступом (СКУД) – це комплекс програмних та технічних засобів які використовується для контролю та моніторингу на точках проходу різних об’єктів (людей, транспортних засобів, роботів) [3].

Будь-яка сучасна система має такі компоненти:

– зчитувач, це пристрій що отримує дані з ідентифікатора і передає їх на контролер (мікрофон, камера, зчитувач карток);

– контролер, це головний пристрій системи, саме тут приймається рішення про допуск користувача та запис телеметрії на основі інформації зі зчитувача (мікроконтролер або повноцінний комп’ютер);

– виконавчий пристрій – цей механізм фізично обмежує доступ (турнікет, шлагбаум, електромагнітний замок).

Сучасні системи розрізняють на три види автономності:

– автономні системи. Є найпоширенішим класом рішень, орієнтованих на виконання базового набору функцій і позиціонованих як альтернатива механічним замкам. Зазвичай використовуються в простих цілях, таких як ключ карта в готелі. Перевагами є низька вартість, надійність та автономність. А недоліками є відсутність дистанційного керування та низький функціонал;

– мережеві (централізовані) системи. Є більш складнішими рішеннями які можуть виконувати широкий функціонал але є залежними до мережевої інфраструктури. Одним з прикладів використання є моніторинг руху транспорту на складській території. До плюсів можна віднести можливість збору даних в

реальному часі, гнучке налаштування графіків та правил доступу. До недоліків відносять низька автономність, висока складність впровадження та обслуговування та необхідність в зовнішній інфраструктурі;

– комбіновані системи. Зазвичай мають переваги двох попередніх класів завдяки підтримки двох режимів роботи та буферу для даних. Це дозволяє використовувати в об’єктах критичної інфраструктури. Наприклад в системах енергетики або банках. До плюсів відносять високий рівень надійності та можливість працювати в автономному режимі. З недоліків є надвисока вартість впровадження та обслуговування.

Також СКУД розрізняють за класами захисту. Від простих систем з ідентифікацією по RFID-картці. До над захищених систем які використовують унікальні відбитки кожної людини, наприклад відбиток радужки ока.

Вибір компонентів підсистеми має базуватися на балансі між комфортом використання, ціною та безпекою.

1.2 Аналіз методів та засобів ідентифікації персоналу

Методи ідентифікації які використовуються в СКУД поділяють на дві групи:

– матеріальні. Зазвичай використовується спеціальний ідентифікатор яким може бути картка, пін-код або ключ.

– біометричні. Базуються на характеристиках людини які майже не змінюються з часом. До цієї категорій відносять: відбиток пальця, форма лиця, райдужка ока;

Найпопулярнішими в наш час є такі методи ідентифікації:

– радіочастотна ідентифікація. Найпопулярніший метод який можна побачити в дуже багатьох системах. Принцип: Зчитувач генерує електромагнітне поле, якщо мітка знаходиться близько, вона починає обмінюватись інформацією.

Переваги: мінімальна вартість, висока швидкість та зручність використання. Недоліки: картку можна загубити, вкрасти, скопіювати, або передати;

– біометрична ідентифікація за відбитком пальця. Даний метод є розповсюдженим в бюджетних системах. Принцип: Базується на скануванні унікальних капілярних ліній на пальці та порівнянні з шаблоном в базі даних. Переваги: майже повне виключення ризику втрати або забування, високий рівень точності за нормальних умов та відносно низька собівартість впровадження. Недоліки: суттєве зростання відсотка хибних відмов в умовах промислового виробництва через забруднення рук а також при наявності пошкоджень шкіри;

– розпізнавання обличчя. Сучасний метод ідентифікації, може бути як бюджетним через камеру. Так і дорогим через складні датчики які працюють на базі створення 3D карти глибини обличчя. Принцип: зчитувач фіксує обличчя, виділяє ключові антропометричні точки та перетворює їх на цифровий код який записує в базу даних. При необхідності ідентифікації алгоритм знову сканує лице та порівнює його. Переваги: безконтактність, висока точність та зручність. Недоліки: залежність від інтенсивності освітлення, а також високі вимоги до обчислювальних потужностей системи, необхідність знімання аксесуарів які закривають обличчя;

– сканування райдужної оболонки ока. Метод є дуже надійний але є найскладнішим так як потребує використання складного обладнання. Принцип: зчитувач сканує унікальний відбиток райдужки ока, який унікальний навіть на різних очах однієї людини. Після цього алгоритм порівнює його з відбитком с бази даних. Переваги: надвисока точність, ідентифікатор не змінюється з віком. Недоліки: велика дистанція зчитування потребує дуже дорогої оптики та високої обчислювальної потужності, є дорогим для впровадження та складним у використанні;

– голосова ідентифікація. Найпростіший метод ідентифікації, не рекомендується використовувати як єдиний фактор. Принцип: аналіз

характеристик голосу та тембру з подальшим створенням цифрового голосового відбитку. Переваги: можливість віддаленої ідентифікації через мікрофон та відсутність потреби у спеціальному складному обладнанні. Недоліки: Голос може змінюватись із-за хвороби, шум навколо може заважати ідентифікації, та низька безпека методу [4].

Далі було створено порівняльну таблицю 1.1

Таблиця 1.1 – Порівняльний аналіз методів ідентифікації персоналу

Метод	Вартість	Рівень безпеки	Стабільність зчитування	Зручність для персоналу	Надійність
RFID-картки	Низька	Низький	Висока	Висока	Висока
Відбиток пальця	Середня	Високий	Низька	Середня	Низька
Обличчя	Середня	Високий	Середня	Висока	Середня
Райдужка ока	Висока	Високий	Висока	Низька	Висока
Голос	Низька	Низький	Середня	Висока	Висока

Аналіз даних, наведених у таблиці 1.1, демонструє що використання тільки одного з методів не є ефективним у межах виробництва. Також важливим є фактор взаємодії з пристроєм. Використання контактних методів менш безпечне з точки розповсюдженню хвороби та менш комфортне. А дуже технологічні методи є не доцільними в таких система із-за вартості та складності впровадження.

Для досягнення мети, потрібно використовувати двох факторний підхід. Це потребує розробки більш складного програмного забезпечення але дозволяє зробити систему більш комфортною та безпечною для людини та підприємства.

Тому була обрана двох факторна система. На першому етапі буде використовуватися RFID-картка а на другому сканування обличчя. Це дозволить економити ресурси при перевірці, завдяки зв'язку картки та обличчя а також підвищить рівень безпеки.

1.3 Аналіз готових комерційних рішень

Перед розробкою власної системи потрібно проаналізувати існуючи рішення. Зрозуміти їх сильні та слабкі сторони та визначити чи є доцільною розробка нового рішення. В даний момент популярними системами є:

– ZKTeco (серія F18). Компаніє є одним із лідерів в своїх сфері. Вони розроблять досить сучасні системи контролю доступу. Їх системи можуть працювати в багатьох режимах. Пристрої серії F18 поєднують RFID-зчитувач, сканер відбитку пальця та клавіатуру для введення PIN-коду. Термінали мають вбудований контролер та підтримують підключення до ПК через інтерфейси TCP/IP, USB або RS-485. Також термінал має систему відстежування стану дверей. Якщо двері були відкриті без дозволу, він увімкне тривогу. Програмне забезпечення ZKTime дозволяє формувати табелі та експортувати дані в формати Excel та 1C. Основним недоліком є закрита архітектура, це робить розширення функцій складними без придбання додаткових ліцензій та API-доступу. Також це робить термінал менш безпечним для використання в банківському та воєнних організаціях. Вигляд цього терміналу зображений на рисунку 1.1 [5];



Рисунок 1.1 – Біометричний термінал контролю доступом ZKTeco F18

– Suprema (BioEntry W2). Також є лідером ринку але в преміум сегменті. Теж мають методи ідентифікації RFID, відбиток пальця та обличчя. Пристрої відповідають стандарту захисту IP65, що дозволяє використовувати їх у складних промислових умовах. Висока ціна рішення (від 45 000 грн. за термінал) обмежує його застосування на підприємствах малого та середнього бізнесу. Також сильним недоліком є закрита архітектура системи, що також робить його менш безпечним в системах які це потребують. Вигляд терміналу зображений на рисунку 1.2 [6].



Рисунок 1.2 – Біометричний термінал контролю доступом Suprema (BioEntry) W2

Аналіз демонструє що використання комерційних рішень є досить дорогим для малих та середніх підприємств. Також це є менш безпечним в критичних системах які це потребують. Це доводить що розробка такої системи є актуальною на відритих компонентах з відкритим вихідним кодом.

1.4 Висновки до розділу

Проведений аналіз визначив, що розробка своєї системи є доцільною та актуальною. Також можна сформувати такі висновки:

По-перше, для реалізації системи ідентифікації, потрібно використовувати двох факторний підхід. На першому етапі необхідно використовувати RFID-карту яка зв’язана в базі даних з людиною. А на другому етапі систему розпізнавання обличчя, це дозволить захистити систему від втрати картки або їх передачі другій особі.

По-друге, для цілі автоматизації доступу та контролю часу необхідно використовувати архітектуру мережевих систем.

2 РОЗРОБКА ПІДСИСТЕМИ АВТОМАТИЗОВАНОГО ДОСТУПУ ТА ОБЛІКУ РОБОЧОГО ЧАСУ

2.1 Розробка загальної структурної схеми підсистеми

Структурна схема є важливим етапом розробки. Вона описує архітектуру рішення а також список блоків та зв’язки між ними.

Підсистема працює за способом двухфакторної ідентифікації. На першому етапі людина прикладає RFID-карту. Після цього контролер зчитує її унікальний ідентифікатор та відправляє на хмарний сервер через Wi-Fi. Після цього сервер шукає цей ідентифікатор в базі даних. Якщо ідентифікатор знайдений, сервер у відповідь надсилає відбиток обличчя. Далі контролер робить знімок з камери та порівнює данні. Якщо особу підтверджено, контролер відмикає замок та відправляє на сервер дані що ідентифікація успішна. Сервер записує цю інформацію для подальшого аналізу.

Підсистема складається з таких функціональних блоків:

– центральний мікроконтролер. Є головним локальним обчислювальним вузлом. Він отримує данні з інших модулів, аналізує їх приймає рішення та взаємодіє з сервером;

– модуль RFID-ідентифікації. Необхідний для першої стадії ідентифікації. Зчитує RFID-карту та передає інформацію з неї на контролер;

– модуль відеокамери. Забезпечує другу стадію ідентифікації. Камера робить знімок обличчя та передає його на головний контролер;

– виконавчий пристрій (електромагнітний замок). Необхідний для фізичного обмеження людини. Отримує команди від мікроконтролера;

– блок індикації. Необхідний для інформування про стан системи. Використовується як світлова індикація, так і звукова;

– модуль живлення. Забезпечує стабілізоване живлення всіх компонентів підсистеми. Мікроконтролера та периферійних модулів;

– хмарний сервер. Є верхнім рівнем архітектури підсистеми. Виконує функції централізованого сховища бази даних персоналу, журналу подій доступу з часовими мітками, а також веб-інтерфейсу для адміністратора системи.

Узагальнений алгоритм взаємодії структурних блоків підсистеми виглядає так, у режимі очікування мікроконтролер постійно опитує модуль зчитування карток на наявність в зоні зчитування, при цьому зелений світлодіод перебуває у активному стані. При виявленні картки зчитується її UID та надсилається запит до хмарної бази даних для перевірки. У разі підтвердження реєстрації картки система переходить до другого етапу - активується камера та виконується захоплення зображення обличчя. Результат розпізнавання обличчя порівнюється з еталонним шаблоном, прив'язаним до даного UID. За умови збігу обох факторів мікроконтролер формує керуючий сигнал на реле замка, активує зелений світлодіод та зуммер, а подія фіксується у хмарному журналі з відміткою часу входу. У разі невідповідності хоча б одного з факторів активується червоний світлодіод та тривожний сигнал, подія також фіксується у журналі як невдала спроба доступу.

Саме такий алгоритм забезпечує автоматизовану ідентифікацію та контроль робочого часу, що відповідає вимогам завдання.

2.2 Вибір та обґрунтування апаратних модулів

Вибір апаратних модулів є важливим етапом проєктування підсистеми, від характеристик кожного компонента залежить надійність, точність та швидкодія всієї системи. У цьому підрозділі наводиться детальний опис та технічне обґрунтування вибору кожного апаратного модуля підсистеми.

2.2.1 Центральний мікроконтролер

Вибір мікроконтролерної платформи є важливим рішенням. Від цього залежить безпека системи, складність розробки, ціна та можливі методи ідентифікації. В наш час є багато сучасних платформ. Нижче ми порівнюємо найпопулярніші з них.

– Arduino Uno / Arduino Mega. Платформа Arduino на базі мікроконтролерів AVR є найбільш поширеною серед прототипів навчального та аматорського рівня. Головною перевагою є легкість розробки та висока кількість різних бібліотек і навчального матеріалу. Але вона має слабкий процесор та мало пам’яті для задач ідентифікації та невелику кількість портів. Також проблемою є відсутність модулів зв’язку (Wi-Fi, Bluetooth), це потребує використання сторонніх модулів. Це робить плату не актуальною для сучасних задач ідентифікації. Зовнішній вигляд мікроконтролерів представлений на рисунку 2.1 [7];

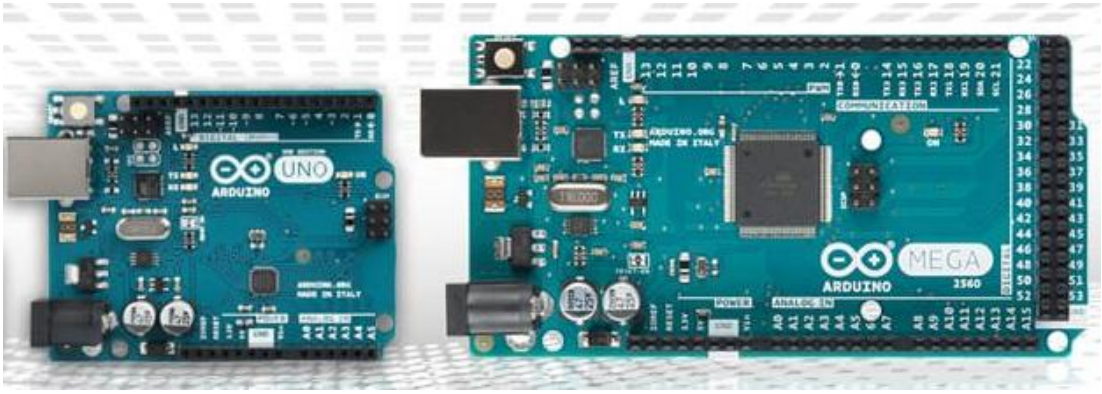


Рисунок 2.1 – Мікроконтролери Arduino Uno та Arduino Mega

– ESP32 (Espressif Systems). Мікроконтролер ESP32 є більш потужнішим. Він має більш потужний процесор та немалу кількість пам’яті. Наявність модулів зв’язку (Wi-Fi, Bluetooth) робить його більш актуальним для IoT пристроїв. Також є різні модифікації, які є ще потужніші і підходять для задач пов’язаними з нейро-мережами. Також плюсом є низька складність розробки. Це робить цей

контролер ідеальним для невеликих підприємств. Плата мікроконтролера зображена на рисунку 2.2 [8];



Рисунок 2.2 – Мікроконтролер ESP32

– STM32 (STMicroelectronics). Мікроконтролери серії STM32 на базі ядра ARM Cortex-M є стандартом у промислових системах. Цей модуль має непоганий процесор але недостатнім для використання в задачах с нейро-мережами. Великим плюсом є наявність гарної документації та високої надійності яка перевірена часом. Плата мікроконтролера зображена на рисунку 2.3 [9];

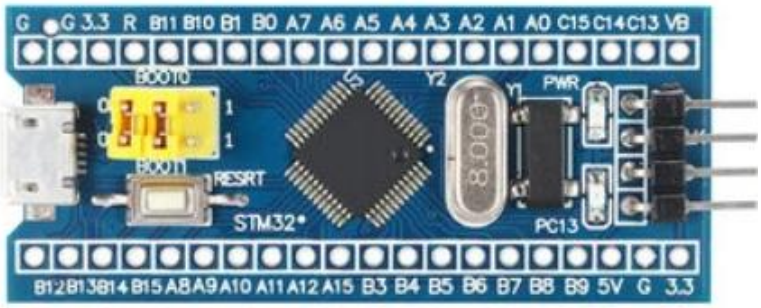


Рисунок 2.3 – Мікроконтролер STM32 F405

– Raspberry Pi (Sony UK Technology Centre). Дуже потужний, і вже не є мікроконтролером. Працює на системі Linux і частіше використовується як головний сервер. Але це робить його над складним для використання в системах автоматизації контролю доступу. Також висока ціна робить його не вигідним для цих цілей. Вигляд плати зображений на рисунку 2.4 [10].

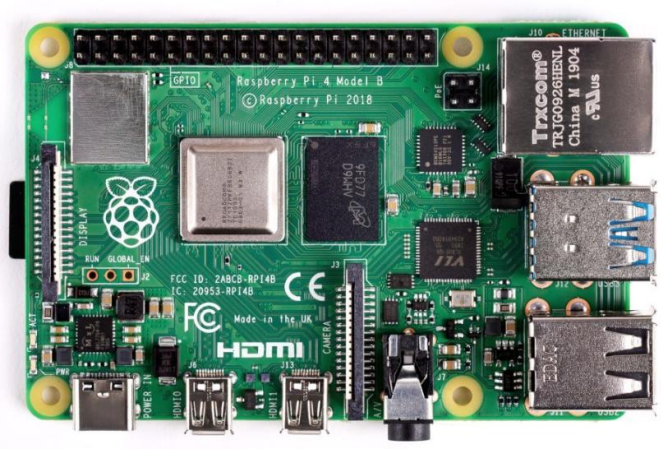


Рисунок 2.4 – Одноплатний комп’ютер Raspberry Pi 4

У таблиці 2.1 наведено порівняльний аналіз розглянутих апаратних платформ.

Таблиця 2.1 – Порівняльний аналіз апаратних платформ

Параметр	Arduino UNO	ESP32	STM32F4	Raspberry Pi 4
Тактова частота	16 МГц	240 МГц	180 МГц	1800 МГц
ОЗП	2 кБ	520 кБ	192 кБ	4096 МБ
Вбудований Bluetooth	Відсутній	Є	Відсутній	Є
Вбудований Wi-Fi	Відсутній	Є	Відсутній	Є

Продовження таблиці 2.1

Параметр	Arduino UNO	ESP32	STM32F4	Raspberry Pi 4
Підтримка OS	Немає	FreeRTOS	FreeRTOS	Linux
Складність розробки	Низька	Середня	Висока	Середня
Придатність для СКУД	Низька	Висока	Середня	Висок
Вартість модуля	250 грн.	350 грн.	140 грн.	7500 грн.

За результатами проведеного аналізу визначено що більш доцільним є використання мікроконтролера ESP32. Він має достатню потужність, підтримку різних методів зв’язку без сторонніх модулів. Також він є досить компактним та низьку ціну. Тому вибір цього контролера є правильним.

Але є проблема, звичайний ESP32 не має потрібних інструкцій для роботи з нейро мережами які необхідні для розпізнавання обличчя, тому більш доцільним є використання більш потужної версії плати, а саме ESP32-S3 WROOM. Порівняння характеристик наведено в таблиці 2.2.

Таблиця 2.2 – Технічні характеристики які відрізняють класичну ESP32 від більш потужної ESP32-S3

Параметр	ESP32	ESP32-S3
Архітектура процесора	Xtensa Dual-Core 32-bit LX6	Xtensa Dual-Core 32-bit LX7
Векторні інструкції (AI)	Немає	Є (прискорення для нейромереж, DSP)
Додаткова SRAM	8 КБ (RTC)	16 КБ (RTC) + 8 КБ (TCM)

Продовження таблиці 2.2

Параметр	ESP32	ESP32-S3
Кількість виводів GPIO	34	45
Інтерфейс камери / LCD	Програмний (I2S паралельний)	Апаратний (LCD 8-16 біт, DCMI для камер)

ESP32-S3 оснащений вбудованим модулем Wi-Fi стандарту 802.11 b/g/n (2,4 ГГц) та Bluetooth 5.0 (LE + Classic), що забезпечує бездротову передачу даних на хмарний сервер без використання додаткових модулів. Апаратно підтримуються інтерфейси SPI, I2C, I2S, UART, USB OTG та DVP-інтерфейс для підключення камери. Наявність до 45 програмованих GPIO-портів забезпечує достатню кількість ліній для підключення всіх периферійних модулів підсистеми.

Важливою особливістю ESP32-S3, що відрізняє його від попередніх версій, є наявність апаратного векторного прискорювача для операцій машинного навчання. Це дозволяє виконувати операції згорткових нейронних мереж (CNN), що використовуються при розпізнаванні обличчя, швидше програмної реалізації, різниця більш ніж десятикратна. Додатково в наявності апаратний модуль шифрування AES-128/256, SHA та RSA, що забезпечує захист даних при передачі на хмарний сервер та при необхідності на локальному сховищі.

Також плата Freenove на базі ESP32-S3 має ряд конструктивних переваг: вбудований USB-to-UART конвертер для зручного програмування, роз'єм для підключення камери який мінімізує проблеми з контактом, вбудований світлодіод та кнопки скидання й завантаження. Наявність зручного pinout з підписаними портами суттєво спрощує процес макетування та налагодження схеми. Тому вибір потужної версії є більш коректним для розроблення такої системи. Порівняння характеристик зі звичайною версією знаходиться в таблиці 2.3, а зображення плати на рисунку 2.5 [11].

Таблиця 2.3 – Технічні характеристики ESP32-S3

Параметр	Значення
Процесорне ядро	Xtensa LX7, двоядерний
Тактова частота	До 240 МГц
SRAM	512 кБ
Flash-пам'ять	8 МБ (на платі Freenove)
Wi-Fi	802.11 b/g/n, 2,4 ГГц
Bluetooth	5.0 BLE + Classic
Кількість GPIO	До 45
Інтерфейси	SPI, I²C, UART, USB OTG, DVP
Напруга живлення	3,0 – 3,6 В
Апаратне шифрування	AES, SHA, RSA

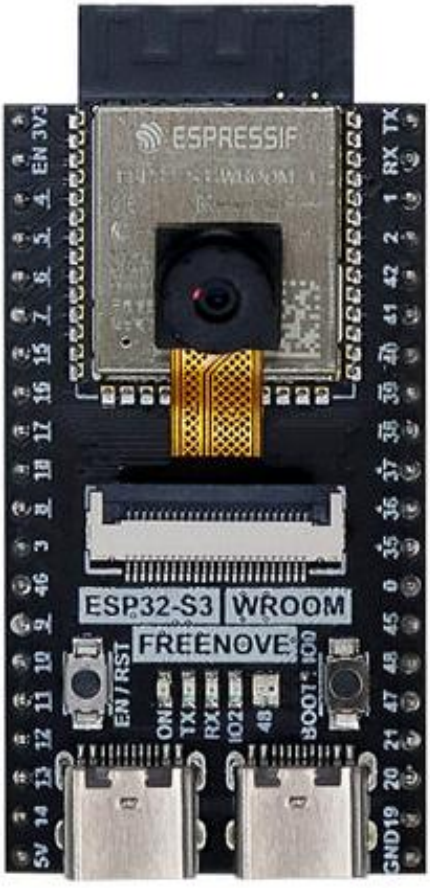


Рисунок 2.5 – Мікроконтролер ESP32-S3 Freenove

2.2.2 Модуль RFID-ідентифікації

Для реалізації першого етапу ідентифікацій є декілька варіантів модулів зчитування картки. Найпоширеніші з них представлені в таблиці 2.4.

Таблиця 2.4 – Порівняльна таблиця модулів зчитування карток [12-14]

Характеристика	MFRC522 (RC522)	PN532	PN5180
Основний фокус	Бюджетний RFID-зчитувач	Повноцінний NFC-контролер	Промисловий зчитувач великої дальності
Підтримувані протоколи	ISO/IEC 14443A, MIFARE	ISO/IEC 14443A/B, FeliCa, ISO/IEC 18092, MIFARE	ISO/IEC 14443A/B, ISO/IEC 15693, FeliCa, ISO 18000-3
Інтерфейси підключення	SPI, I2C, UART	SPI, I2C, UART	SPI (швидкісний + піни BUSY/IRQ)
Дальність зчитування	~2–3 см	~3–5 см	~8-10 см
Ціна	70 грн.	250 грн.	600 грн.

Для реалізації першого етапу ідентифікації обрано модуль PN532 – спеціалізований контролер NFC/RFID виробництва компанії NXP Semiconductors, який є промисловим стандартом у системах контролю доступу.

PN532 підтримує широкий перелік стандартів безконтактних карток, а також NFC-теги. Це забезпечує сумісність з переважною більшістю корпоративних карток доступу, що використовуються на виробничих підприємствах України.

Модуль підтримує три інтерфейси підключення до мікроконтролера: UART, SPI та I2C, це забезпечує гнучкість при проєктуванні. Для підключення до ESP32-S3 найбільш коректним є використання інтерфейсу I2C, оскільки він потребує лише двох сигнальних ліній і дозволяє звільнити інтерфейс SPI для камери. Вибір даного протоколу також спрощує паралельне підключення інших периферійних пристроїв до спільної шини без використання додаткових пінів мікроконтролера. Окрім, апаратна підтримка I2C у мікроконтролері дозволяє стабільно працювати на стандартних частотах передачі даних, що забезпечує високу швидкість відгуку системи зчитування. Модуль зображений на рисунку 2.6.

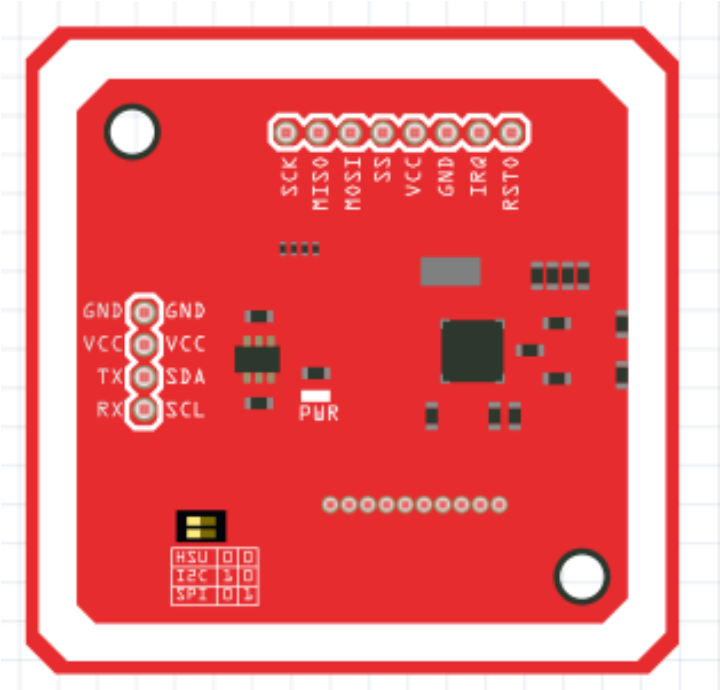


Рисунок 2.6 – Модуль зчитування карток PN532

2.2.3 Модуль камери

Для реалізації другого етапу ідентифікації необхідно обрати сумісний модуль камери. Найпоширеніші модулі представлено в порівняльній таблиці 2.5 [15-17].

Таблиця 2.5 – Порівняльна таблиця камер які підгримуються ESP-32

Характеристика	OV2640	OV3660	OV5640
Максимальна роздільна здатність	2 Мп (1600 x 1200)	3 Мп (2048 x 1536)	5 Мп (2592 x 1944)
Інтерфейс передачі	DCMI (8-бітний паралельний)	DCMI (8-бітний паралельний)	DCMI (8-біт) + підтримка MIPI (в ESP32 не використовується)
Частота кадрів (FPS)	800x600 30 к/с, 1600x1200 15 к/с	1080p 15 к/с, 720p 30 к/с	720p 30 к/с, 1080p 30 к/с, 720p 60 к/с
Ціна	350 грн.	200 грн.	500 грн.

Для реалізації другого етапу ідентифікації обрано камерний модуль OV2640, який є стандартним модулем для плат серії ESP32-CAM та сумісний із роз'ємом плати Freenove ESP32-S3.

Плата Freenove ESP32-S3 має спеціалізований 24-контактний роз'єм для підключення модуля OV2640 через “шлейф”, що спрощує монтаж та забезпечує надійне з'єднання. Зовнішній вигляд камери зображений на рисунку 2.7.



Рисунок 2.7 – Камера OV2640

2.2.4 Виконавчий пристрій

Для фізичного обмеження доступу необхідно обрати електромагнітний замок, найпопулярніші варіанти наведені в таблиці 2.6 [18-20].

Таблиця 2.6 – Порівняльна таблиця електромагнітних замків

Характеристика	Електромагнітний YLI YM-280	Електромеханічний Atis Lock SS	Електроригельний YLI YB-100
Принцип роботи	Магнітне утримання плити (потрібна постійна напруга)	Механічний заводний штифт (відкриття коротким імпульсом)	Висувний металевий штифт за допомогою соленоїда (відкриття коротким імпульсом)
Сила утримання	Висока (до 500 кг)	Середня (залежить від міцності ригеля, ~400 кг)	Дуже висока (до 800-1000 кг на зріз)
Стан при вимкненні живлення	Відкритий	Закритий	Є оба варіанти
Струм споживання	Постійно: ~500 мА	Лише в момент імпульсу (0.5 сек): ~1.5А – 2.5А	В момент старту: ~900 мА В режимі очікування: ~150 мА

Продовження таблиці 2.6

Характеристика	Електромагнітний YLI YM-280	Електромеханічний Atis Lock SS	Електроригельний YLI YB-100
Ресурс / Зносостійкість	Максимальний	Середній	Високий
Ціна	1500 грн.	1000 грн.	2000 грн.

Для фізичного обмеження доступу обрано електромагнітний замок YLI YM-280. Принцип дії електромагнітного замка базується на утриманні феромагнітної пластини (якоря), вмонтованої у дверне полотно, за рахунок електромагнітної притягальної сили, що виникає при проходженні струму через обмотку електромагніта. Замок зображений на рисунку 2.8.

Електромагнітні замки живляться напругою 12 В постійного струму та споживають струм у діапазоні 400–500 мА в активному стані.

Оскільки ESP32-S3 працює з рівнями напруги 3,3 В і не здатний безпосередньо керувати замком з напругою 12 В, для комутації використовується проміжний каскад на основі транзисторного ключа або електромеханічного реле. Керуючий сигнал від GPIO-порту мікроконтролера через обмежувальний резистор надходить на затвор MOSFET-транзистора, що забезпечує гальванічну розв'язку логічної та силової частин схеми.

При проєктуванні схеми керування замком необхідно визначити режим роботи при відмові живлення. Режим Fail-Safe (замок залишається відкритим при відключенні живлення) є більш безпечним з точки безпеки евакуації. Тому вибір саме цього режиму є доцільним.

У межах розробки макету підсистеми електромагнітний замок імітується синім світлодіодом, що дозволяє візуально підтвердити факт спрацювання виконавчого пристрою без підключення силового кола живлення 12 В



Рисунок 2.8 - Електромагнітний замок YLI YM-280

2.2.5 Блок індикації

Для забезпечення зворотного зв'язку з користувачем, підсистема оснащена двома світлодіодними індикаторами та активним п’єзоелектричним зуммером. Використання комбінованої системи дозволить краще інформувати користувача про стан системи.

У системі застосовуються світлодіоди двох кольорів, кожен з них підключається окремо до свого піна мікроконтролера через обмежуючий резистор. Це необхідно щоб не світлодіоди та піни мікроконтролера не руйнувалися через високу напругу. Зображення світлодіоду знаходиться на рисунку 2.9.

П’єзоелектричний зуммер, який виконує роль звукового індикатора, підключається через захисний NPN транзистор, що також запобігає пошкодженню компонентів. На відміну від статичної індикації світлодіодів, звуковий супровід реалізовано через програмне керування за допомогою широтно-імпульсної модуляції (ШІМ). Зображення зумеру знаходиться на рисунку 2.10.



Рисунок 2.9 – Світлодіоди



Рисунок 2.10 – П’єзоелектричний зуммер

2.2.6 Внутрішня пам'ять

Для додаткової пам'яті використовується SD-картка на 1 гігабайт. Яка підключена через спеціальний роз’єм на платі мікроконтролера. Вона використовується для конфігурування системи. Вигляд карти пам’яті на рисунку 2.11.



Рисунок 2.11 – SD-картка

2.2.7 Живлення системи

Живлення системи відбувається через роз’єм DC Barrel Jack від зовнішнього блоку живлення 12 В. Але для живлення мікроконтролера необхідно 5В. Для цих цілей необхідно обрати перетворювач напруги. Порівняння характеристик представлено в таблиці 2.7 [21-23].

Таблиця 2.7 – Порівняльна таблиця перетворювачів

Характеристика	Mini-360 (MP2307)	MP1584EN	LM2596S
Тип перетворювача	Імпульсний	Імпульсний	Імпульсний
Макс. струм (тривало)	1.8 А (до 1.2 А без радіатора)	3.0 А	2.0 А (до 3 А з радіатором)
ККД	92%	93%	85%
Ціна	55 грн.	70 грн.	90 грн.

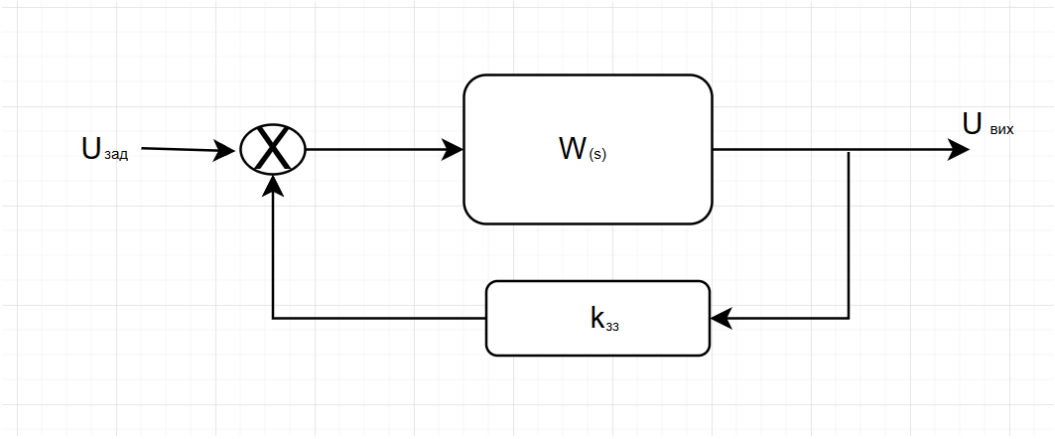
Судячи з порівняльної таблиці, доцільним є використання саме Mini-360 (MP2307). Він має високий ККД та низьку ціну. Максимальний струм в 1.2 А цілком достатній для системи. Цей перетворювач зображений на рисунку 2.12.



Рисунок 2.12 – Перетворювач напруги Mini 360

2.3 Математичне моделювання та аналіз стійкості підсистеми живлення

Перетворювач напруги Mini-360 побудований на базі мікросхеми MP2307 та реалізує топологію знижувального імпульсного перетворювача напруги з внутрішньою петлею зворотного зв'язку за вихідною напругою. З точки зору теорії автоматичного управління (ТАУ), такий перетворювач є замкненою системою автоматичного управління (САР) вихідної напруги [24]. Спрощена структурна схема системи наведена на рисунку 2.13.



$U_{зад}$ – задане значення напруги; W(s) – передатна функція об'єкта управління; $k_{зз}$ – коефіцієнт зворотного зв'язку(дільник напруги); $U_{вих}$ – вихідна напруга.

Рисунок 2.13 – Структурна схема САР підсистеми живлення

Вихідний фільтр перетворювача апроксимується аперіодичною ланкою першого порядку з передатною функцією (2.1).

$$W_{(s)} = \frac{k}{T_s+1}, \tag{2.1}$$

де k = 1 – коефіцієнт передачі;

T – стала часу ланки.

Стала часу ланки визначається з параметрів вихідного кола перетворювача за формулою (2.2).

$$T = R \cdot C = 10 \cdot 100 \cdot 10^{-6} = 10^{-3} c, \quad (2.2)$$

де C = 100 мкФ – ємність вихідного конденсатора;

R = 10 Ом – опір навантаження (при $U_{вих}$ 5В, $I_{вих}$ = 0,5 А).

Передатна функція об’єкта управління (2.3).

$$W_{(s)} = \frac{1}{0{,}001s + 1}. \quad (2.3)$$

Коефіцієнт зворотного зв’язку визначається із співвідношення внутрішньої опорної напруги мікросхеми MP2307 до номінальної вихідної напруги (2.4).

$$k_{зз} = \frac{U_{оп}}{U_{вих}} = \frac{0{,}925}{5} = 0{,}185, \quad (2.4)$$

де $U_{оп}$ = 0,925 В – внутрішня опорна напруга мікросхеми MP2307;

$U_{вих}$ = 5 В – номінальна вихідна напруга.

Передатна функція замкненої системи з від’ємним одиночним зворотнім зв’язком (2.5).

$$W_{зам}(s) = \frac{W(s)}{1 + k_{зз} \cdot W(s)} = \frac{1}{0{,}001s + 1 + k_{зз}}. \quad (2.5)$$

Характеристичне рівняння замкненої системи (2.6).

$$0{,}001s + 1 + k_{зз} = 0. \quad (2.6)$$

Підставляємо числове значення $k_{зз}$ (2.7, 2.8).

$$0{,}001s + 1 + 0{,}185 = 0, \tag{2.7}$$

$$0{,}001s + 1{,}185 = 0. \tag{2.8}$$

Характеристичне рівняння є рівнянням першого порядку з коефіцієнтами: $a_0 = 0{,}001$; $a_1 = 1{,}185$

Перевіримо необхідно умову стійкості: всі коефіцієнти характеристичного рівняння повинні бути одного знаку (2.9).

$$a_0 > 0, a_1 > 0. \tag{2.9}$$

Необхідна умова виконується, оскільки система першого порядку, необхідна умова є водночас достатньою.

Єдиний корінь характеристичного рівняння (2.10).

$$\lambda_1 = -\frac{a_1}{a_0} = -\frac{1{,}185}{0{,}001} = -1185. \tag{2.10}$$

Корінь є від'ємним дійсним числом, тобто розташований у лівій напівплощині комплексної площини, що є необхідною і достатньою умовою стійкості лінійної системи.

Проведений аналіз підтвердив, що підсистема живлення на базі перетворювача Mini-360 є стійкою системою автоматичного регулювання. Всі коефіцієнти характеристичного рівняння мають позитивне значення, а єдиний корінь $\lambda_1 = -1185$ має від'ємну дійсну частину. Це забезпечує стійке регулювання вихідної напруги 5 В для живлення мікроконтролера ESP32 та периферійних пристроїв системи від зовнішнього блоку живлення 12 В.

Постійна часу замкненої системи становить (2.11).

$$t = \frac{1}{|\lambda^1|} = \frac{1}{|-1185|} = 0{,}000844\text{с.} \quad (2.11)$$

Це свідчить про високу швидкодію контуру регулювання порівняно з характерними часами роботі мікроконтролера ESP32.

2.4 Розробка схеми підключення компонентів

Перелік апаратних модулів компонентів, їх призначення та напруга наведені в таблиці 2.8.

Таблиця 2.8 – Зведена таблиця апаратних модулів підсистеми

Модуль	Призначення	Інтерфейс	Напруга
ESP32-S3 Freenove	Центральний контролер	-	3,3 В
PN532	RFID-зчитувач	I2C	3,3 В
OV2640	Камера розпізнавання обличчя	DVP	3,3 В
Електромагнітний замок	Виконавчий пристрій	MOSFET	12 В
Перетворювач напруги	Перетворення напруги	DC Barrel Jack	12 В - 5 В
Світлодіоди (×2)	Візуальна індикація	GPIO	3,3 В

Продовження таблиці 2.8

Модуль	Призначення	Інтерфейс	Напруга
П'єзозуммер	Звукова індикація	GPIO (ШІМ)	3,3 В
SD-картка	Конфігурація	I2C	3.3 В

Після перевірки встановлено, усі обрані апаратні модулі є технічно сумісними між собою, доступними на ринку та забезпечують реалізацію всіх функціональних вимог, сформульованих завданні на розробку підсистеми. Сукупність обраних компонентів створює достатню апаратну базу для переходу до розробки схеми підключення.

Схема підключення визначає усі електричні з’єднання в системі. Схема має бути надійною та безпечною. Тому необхідно правильно підібрати допоміжні компоненти, такі як резистори, транзистори та мосфети.

В макеті живлення здійснюється від USB-порту комп'ютера через вбудований стабілізатор плати Freenove ESP32-S3. Вхідна напруга USB становить 5 В, після цього вбудований стабілізатор формує з них 3.3В, які необхідні для роботи самого мікроконтролера та модулів ідентифікації. Оскільки у макеті замість електромагнітного замка використовується світлодіод-імітатор, необхідність у окремому силовому колі живлення 12 В відсутня, що суттєво спрощує схему прототипу.

Модуль PN532 підключається до мікроконтролера ESP32-S3 через інтерфейс I2C. Перед підключенням необхідно встановити перемички на модулі які переводять його в саме такий режим.

Лінія тактування SCL підключається до піна GPIO1 мікроконтролера, лінія даних SDA - до піна GPIO2. Живлення модуля здійснюється напругою 3,3 В від відповідного виводу плати. Підтягуючі резистори номіналом 4,7 кОм встановлюються між лініями SDA/SCL та шиною живлення 3,3 В для забезпечення коректних рівнів сигналів шини I2C. На більшості готових модулів

PN532 ці резистори вже розпаяні на платі, що не потребує їх додаткового встановлення на макеті.

Камера OV2640 підключається до плати мікроконтролера через спеціалізований 24-контактний роз'єм CAM, розташований зверху. Підключення є стандартизованим і не потребує додаткових зусиль. Модуль камери вставляється у роз'єм шлейфом, що повністю визначає розподіл сигналів DVP-інтерфейсу. Живлення сенсора (3,3 В та 1,8 В для ядра) також формується схемою плати автоматично.

Зелений світлодіод підключається анодом через обмежувальний резистор номіналом 330 Ом до піна GPIO46, а катодом - до шини GND. Червоний світлодіод аналогічно підключається через резистор 330 Ом до піна GPIO21. Номінал обмежувального резистора розраховано виходячи з напруги живлення 3,3 В, прямого падіння напруги на світлодіоді близько 2,0–2,2 В та робочого струму 10 мА.

П'єзоелектричний зуммер підключається через NPN транзистор яким керує пін GPIO47 мікроконтролера, другим виводом – до шини +5V. Це необхідно для захищення піну ESP32 від перевантаження.

У макеті підсистеми електромагнітний замок імітується синім світлодіодом, підключеним також через NPN транзистор до піна GPIO42. Це дозволяє візуально підтвердити факт спрацювання виконавчого пристрою та коректність логіки без підключення реального силового кола. У промисловому варіанті підсистеми транзистор керує затвором MOSFET-транзистора, що комутує обмотку електромагнітного замка з живленням 12 В.

Також до бази транзисторів під’єднані стягуючи резистори на землю. Вони необхідні щоб мінімізувати наводки від других частин системи та швидко закривати транзистори.

Розроблена схема підключення є технічно коректною, відповідає електричним характеристикам усіх компонентів та забезпечує необхідну функціональність макету підсистеми. Схема зображена на рисунку 2.13.

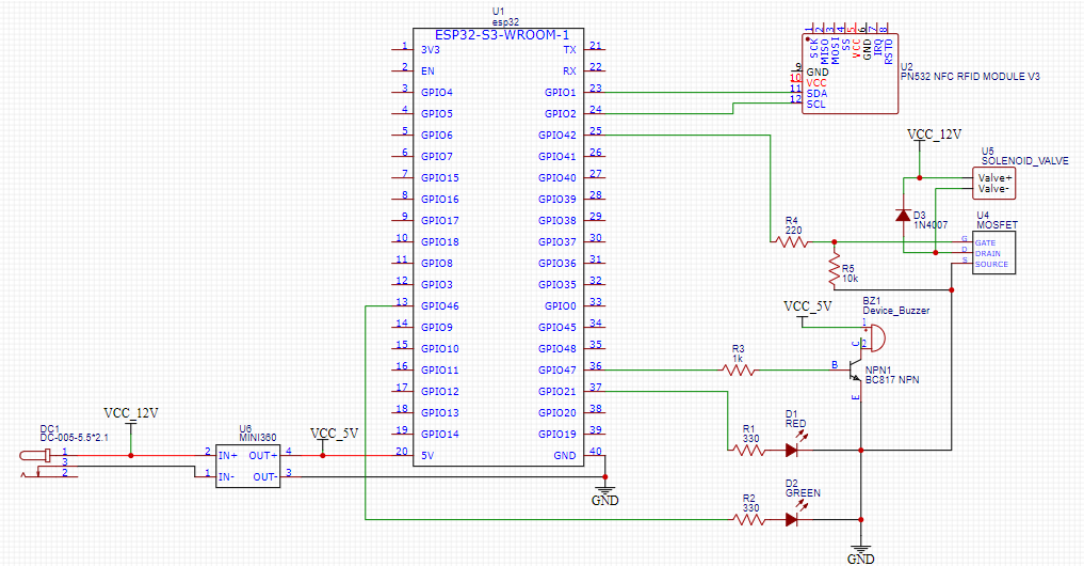


Рисунок 2.13 – Схема підключення підсистеми

2.6 Збірка та тестування макету підсистеми

Наступним етапом було виконано збірку макету підсистеми на безпаяній макетній платі. Такий підхід дозволяє легко вносити зміни до схеми в процесі налагодження без необхідності паяння компонентів, що є стандартною практикою при розробці прототипів вбудованих систем. Для спрощення схеми електромагнітний замок був замінений на синій світлодіод. Система отримує живлення через внутрішній стабілізатор який живиться від 5 вольт USB. Таке спрощення прискорює розробку, зменшує вартість прототипу та підвищує рівень безпеки під час налагодження схеми.

Після завершення збірки було проведено тестування всіх модулів для перевірки коректності підключення та справність модулів. Зібраний макет зображений на рисунку 2.14.

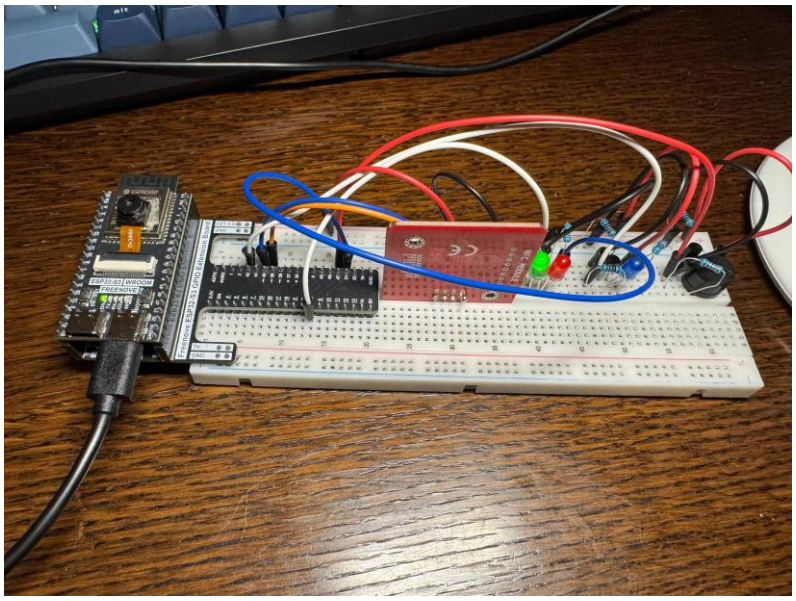


Рисунок 2.14 – Зібраний макет підсистеми автоматизованого доступу

2.7 Висновки до розділу

У другому розділі було виконано повний цикл розробки макету підсистеми. Від вибору та порівняння різних мікроконтролерів та пристроїв периферії, до розробки схеми підключення та збірки макету на безпаяній макетній платі.

Після збірки було подано живлення на схему та тестування пристроїв через вбудовані шаблони програм. Всі пристрої пройшли етап ініціалізація та видають необхідні данні. Також перевірено, що після п’яти хвилин роботи все працює коректно, температура компонентів в нормі, що свідчить про правильне підключення.

Після цього можна приступати до фінального етапу, а саме розробки програмного забезпечення.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка прошивки для мікроконтролера ESP32

3.1.1 Розробка алгоритму функціонування підсистеми

Алгоритм функціонування описує послідовність дій мікроконтролера від моменту увімкнення живлення до завершення циклу ідентифікації та фіксації події у хмарному журналі. Алгоритм реалізує логіку мультифакторної автентифікації та є основою для розробки програмного забезпечення.

Він складається з чотирьох послідовних етапів: ініціалізація системи, режим очікування, двофакторна ідентифікація та виконання рішення з фіксацією події. Кожен етап детально описаний нижче.

1. Ініціалізація системи. Після подачі живлення мікроконтролер ESP32-S3 виконує ініціалізацію всіх периферійних модулів та підучується до мережі через Wi-Fi. Якщо помилок не виявлено, система переходить в режим очікування.

2. Режим очікування. У режимі очікування мікроконтролер циклічно з інтервалом 50 мс опитує модуль PN532 на наявність RFID-картки або брелка в зоні зчитування. Паралельно в фоновому режимі підтримується активне з'єднання з MQTT-сервером шляхом надсилання службових повідомлень. У разі втрати з'єднання з хмарним сервером система автоматично спробує відновити підключення, перериваючи процес опитування зчитувача. Якщо модуль знаходить карту в радіусі зчитування, система переходить до першого етапу ідентифікації.

3. Перший фактор ідентифікації. При виявленні картки в зоні зчитується унікальний ідентифікатор та передається на мікроконтролер. Мікроконтролер формує запит до хмарного сервера через MQTT, передаючи отриманий UID для

перевірки у базі даних зареєстрованого персоналу. Сервер порівнює отриманий UID з базою даних та повертає відповідь. У разі отримання відповіді не зареєстровано система переходить до стану відмови та повертається у режим очікування. У разі отримання відповіді зареєстровано сервер також повертає дані необхідні для наступного етапу після система переходить до другого етапу ідентифікації.

4. Другий фактор ідентифікації. Після підтвердження RFID-картки мікроконтролер активує камеру та виконує захоплення кадру. Далі формується масив векторних біометричних ознак та порівнюється з інформацією отриманою від сервера на попередньому етапі. Якщо обличчя на кадрі не виявлено або ступінь схожості з еталонним шаблоном є нижчим за встановлений пороговий рівень, система переходить до стану відмови з відповідною індикацією та фіксацією події у журналі з міткою “невідповідність обличчя” і додатково відправляє знімок з камери.

5. Виконання рішення та фіксація події. У разі успішного проходження обох факторів ідентифікації мікроконтролер відкриває доступ до дверей, фіксує вхід та переходить знову в режим очікування.

Для коректного обліку робочого часу необхідно розрізняти події входу та виходу працівника. Це реалізується на рівні хмарного сервера: при отриманні події від конкретного працівника сервер перевіряє останній запис у журналі для цього UID. Якщо останньою зафіксованою подією був вихід або записів немає - поточна подія класифікується як вхід. Якщо останньою подією був вхід - поточна подія класифікується як вихід, після чого автоматично розраховується тривалість перебування на робочому місці.

Розроблений алгоритм забезпечує двофакторну ідентифікацію персоналу, коректний облік робочого часу та стійку роботу підсистеми в умовах нестабільного з'єднання. Алгоритм є основою для розробки програмного забезпечення мікроконтролера, що розглядається у наступному розділі.

3.1.2 Визначення та підключення необхідних бібліотек

Для розробки такої складної системи с функцією розпізнавання обличчя необхідно вибрати бібліотеки та фреймворки. Основним фреймворком обраний ESP-IDF версії 6.0.0. Він є офіційним для розробки програмного забезпечення для мікроконтролерів сімейства ESP32 від Expressif Systems.

Він базується на операційній системі FreeRTOS, це дозволяє створювати ефективні багато потокові пристрої які можуть працювати з мережкою на зручному рівні абстракції. Також значним плюсом є повний контроль чіпа на низькому рівні на відміну від альтернативного середовища Arduino IDE [25].

Після цього необхідно визначити стек бібліотек для роботи з периферією, даними та мережею. Для такої цілі обраний такий список компонентів:

– IDF – це базовий фреймворк, він надає основні інструменти для роботи з пам’яттю, драйвери периферії, Wi-Fi та Bluetooth та доступ до основник компонентів FreeRTOS;

– ESP32-camera – офіційний компоненти для роботи з камерою для чіпів сімейства ESP32. Він відповідає за ініціалізацію пристрою, його конфігурацію та захопленням зображень через інтерфейс DVP [26];

– Human face detect – цей компонент реалізує алгоритм штучного інтелекту для знаходження обличчя людини на зображенні яке ми отримуємо з камери. Використовується для другого етапу ідентифікації [27];

– Human face recognition – а цей компонент відповідає саме за розпізнавання обличчя. Це можливо завдяки унікальним біометричним характеристикам людини. Також використовуємо для другого етапу але уже саме для ідентифікації [28];

– Esp IDF PN532 – цей компонент який необхідний для роботи з нашим RF зчитувачем. Використовуватися для першого етапу ідентифікації [29];

– MQTT – цей компонент реалізує функцію клієнта для зв’язку через протокол MQTT. Він буде використовуватися для зв’язку с сервером [30];

– Arduino JSON – це ефективний та популярний компонент для читання та створення даних у форматі JSON. Потрібна для формування складних повідомлень та відповідей для сервера [31].

Ці бібліотеки необхідно вписати в файл (idf_component.yml) який буде використовуватись для збірки прошивки. При збірці вони будуть автоматично завантажені. Зміст цього файлу наведено на рис 3.1.

```yaml
dependencies:
  idf:
    version: '*'
  espressif/esp32-camera: '*'
  espressif/human_face_detect: '*'
  espressif/human_face_recognition: '*'
  garag/esp-idf-pn532: '*'
  espressif/mqtt: '*'
  bblanchon/arduinojson: '*'
```

Рисунок 3.1 – Файл для збірки idf_component.yml

Також необхідно добавити ці компоненти в файл конфігурації саме для проєкту CMakeLists.txt. Це робиться через команду idf_component_register. Також одразу включимо компоненти необхідні для роботи з SD карткою, мережею, та додатковою пам’яттю. Зміст файлу наведено на рисунку 3.2.

```
idf_component_register(
        SRCS ${SOURCES}
        INCLUDE_DIRS "."
        REQUIRES
        esp32-camera
        human_face_detect
        human_face_recognition
        esp_psram
        esp_wifi
        esp_event
        esp_netif
        nvs_flash
        mqtt
        vfs
        fatfs
        sdmmc
        esp_driver_sdmmc
)
```

Рисунок 3.2 – Підключення всіх необхідних компонентів до проєкту ESP32

3.1.3 Визначення структурної схеми

Після підключення усіх необхідних компонентів необхідно визначити архітектуру. Вибір архітектури є одним з ключових етапів проєктування, оскільки він впливає на швидкість розробки, зручність та можливість подальшого масштабування системи. Тому була обрана компонентна архітектура. Це дозволить легко додавати нові функції, швидко тестувати систему та знаходити помилки. Для необхідного рівня абстракції систему поділено на такі модулі:

– App – це головний модуль системи. Він відповідає за ініціалізацію та взаємодію модулів, реалізує основну логіку ідентифікації а також забезпечує спілкування з сервером;

– Buzzer – це модуль який відповідає за звукові сповіщення за допомогою п'єзоелектричного випромінювача;

– Indicators – цей допоміжний модуль керує станом світлодіодів для візуального сповіщення. Він може лише керувати станом світлодіоди;

– DoorRelay – цей допоміжний модуль відповідає за стан виконуючого механізму (електромагнітного замку);

– StateVisualizer – цей модуль використовується для відображення візуальної анімації в залежності від стану системи за допомогою компоненту Indicators;

– RFScanner – цей компонент відповідає за ініціалізацію шини I2C та модуля PN532. Також він займається постійним пошуком та зчитуванням картки с записом у внутрішній буфер;

– Camera – цей компонент відповідає за ініціалізацію та конфігурацію камери та отриманням зображення з неї;

– Recognitor – цей модуль ініціалізує компоненти для роботи з штучним інтелектом, також відповідає за обробку зображення з компоненту Camera. Використовується для аналізу обличчя;

– Wifi – цей допоміжний модуль ініціалізує компоненти для роботи з мережею та встановлює зв'язок з Wi-Fi точкою;

– Mqtt – цей модуль відповідає за підключення до сервера та передачу даних до нього за допомогою компоненту Wifi;

– AppConfig – цей модуль займається зчитуванням та збереженням конфігураційних даних з SD картки.

Отже є головний компонент App в якому виконується основна логіка. В собі він містить другорядні модулі, які вже відповідають за взаємодію з периферійними пристроями. Така архітектура дозволить легко змінювати модулі, наприклад камеру та вносити зміни лише в відповідний модуль. Також таку систему легко тестувати та модернізувати. Схема взаємодії компонентів представлена на рисунку 3.3.

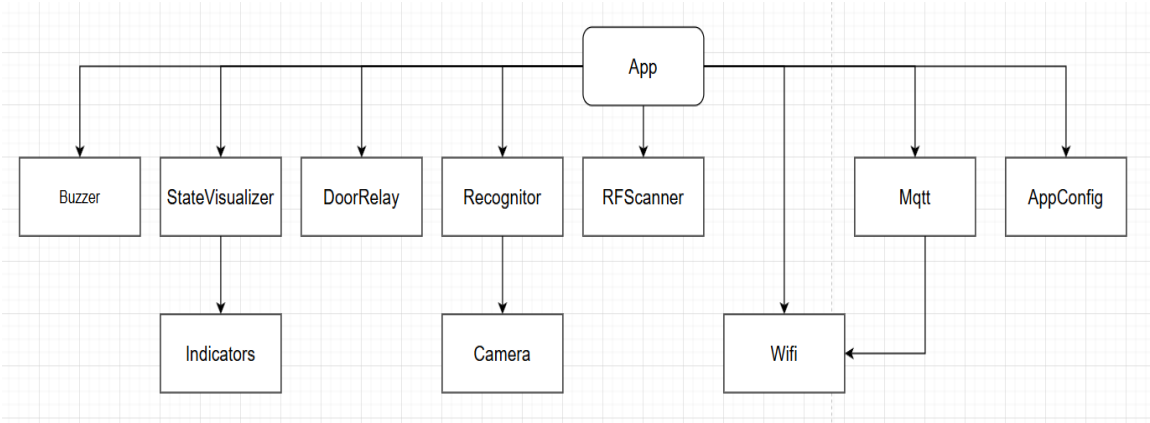


Рисунок 3.3 – Структурна схема взаємодії компонентів для прошивки ESP32

3.1.4 Розробка алгоритму роботи

Перед початком розробки програмного коду, необхідно створити алгоритм який описує логіку мікроконтролера на високому рівні абстракції. З урахуванням вимоги розробки двох факторної підсистеми ідентифікації, був розроблений алгоритм зображений на рисунку 3.4.

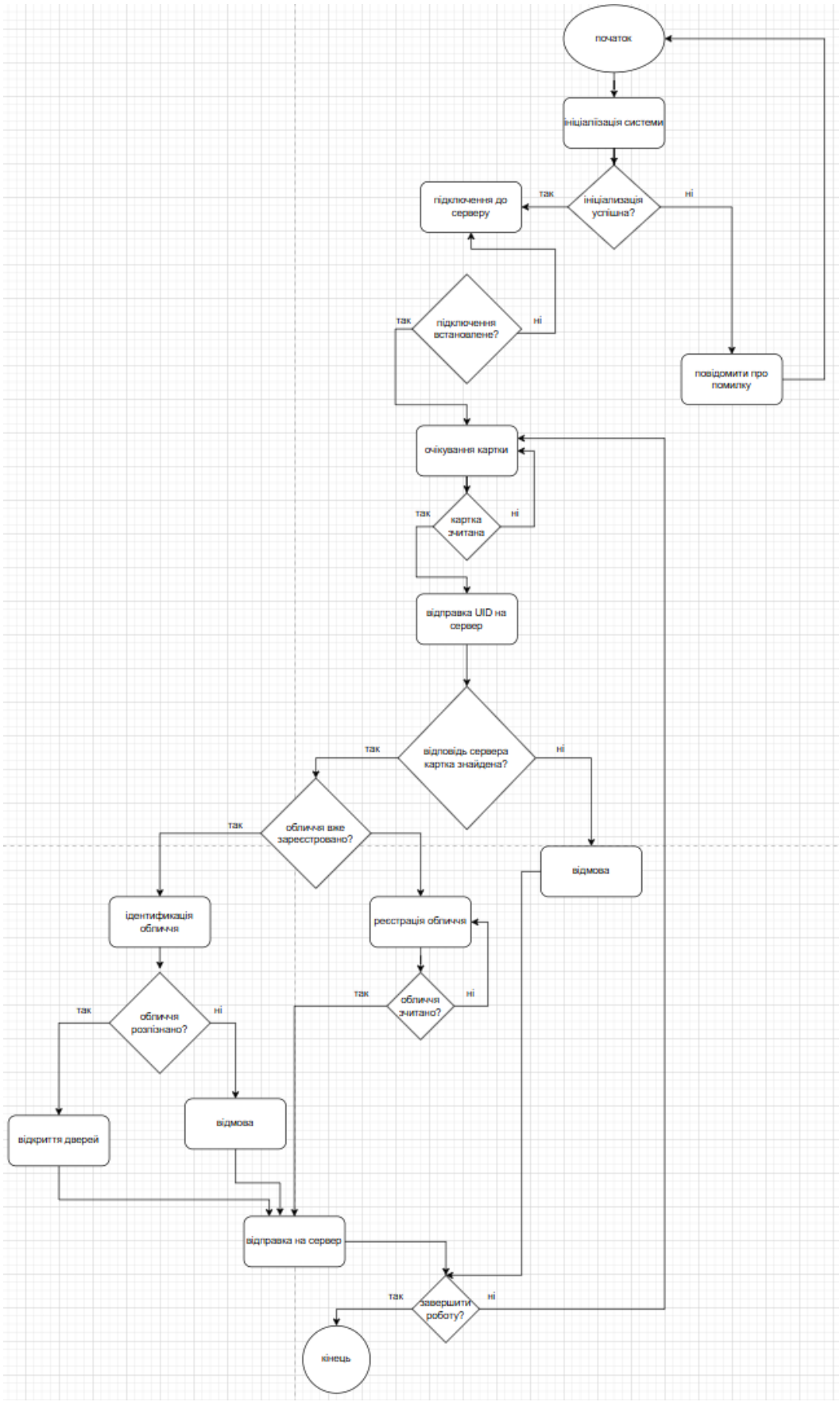


Рисунок 3.4 – Алгоритм програми ESP32

3.1.5 Розробка програмного коду

Після визначення структури та алгоритму можемо приступити до розробки компонентів.

Головним компонентом програми є “App”. Він містить усі другорядні компоненти, саме в ньому головна логіка. Зміст цього файлу зображено на рисунку 3.5.

```
20  class App {
21  public:
22      App();
23      ~App();
24      void run();
25
26  private:
27      void updateState(AppState new_state);
28      AppState getState();
29      SemaphoreHandle_t _state_mutex;
30
31      void handleState();
32      void init();
33      void onReadCard(const RFScanner::UidResult& uid);
34      void handleMqttMessage(const MqttMessage& msg);
35
36      void setupNetwork();
37
38      std::vector<std::vector<float>> startLearn();
39
40      AppState _current_state;
41      uint32_t _current_state_time;
42      RFScanner::UidResult _current_card;
43      std::vector<std::vector<float>> _current_recognition_data;
44
45      Buzzer _buzzer;
46      Indicators _indicators;
47      DoorRelay _door_relay;
48      StateVisualizer _state_visualizer;
49      RFScanner _rf_scanner;
50      Camera _camera;
51      Recognitor _recognitor;
52      Wifi _wifi;
53      Mqtt _mqtt;
54      AppConfig _config;
55  };
```

Рисунок 3.5 – Вигляд головного компоненту App

В момент старту система передає управління в метод “run”. В цьому методі виконується ініціалізація та перевірка всіх компонентів через допоміжний метод “init”, його зміст представлено на рисунку 3.6.

```
313  void App::init() {
314      esp_err_t ret = nvs_flash_init();
315      if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
316          ESP_ERROR_CHECK(nvs_flash_erase());
317          ESP_ERROR_CHECK(nvs_flash_init());
318      }
319
320      _buzzer.init();
321      _indicators.init();
322      _door_relay.init();
323
324      _indicators.red(true);
325
326      if (_rf_scanner.init() != ESP_OK) {
327          ESP_LOGE(TAG, "RF Scanner init failed, restarting...");
328          vTaskDelay(pdMS_TO_TICKS(2000));
329          esp_restart();
330      }
331
332      if (_camera.init() != ESP_OK) {
333          ESP_LOGE(TAG, "Camera init failed, restarting...");
334          vTaskDelay(pdMS_TO_TICKS(2000));
335          esp_restart();
336      }
337
338      _config.init();
339      _recognitor.init();
340      _config.load();
341      _buzzer.play(MelodyID::STARTUP);
342  }
```

Рисунок 3.6 – Вигляд методу “init”

Якщо ініціалізація пройшла успішно, створюється три додаткових задачі необхідних для задач які виконуються незалежно від основного циклу програми. Код запуску задач представлено на рисунку 3.7.

– buzzer_task – відповідає за звукову індикацію;
– state_visualizer_task – відповідає за візуальну індикацію;
– rf_scanner_task – відповідає за зчитування картки.

```
xTaskCreate(Buzzer::task_wrapper, "buzzer_task", 2048, &_buzzer, 3, NULL);
xTaskCreate(StateVisualizer::task_wrapper, "state_visualizer_task", 2048, &_state_visualizer, 3, NULL);
xTaskCreate(RFScanner::task_wrapper, "rf_scanner_task", 4096, &_rf_scanner, 4, NULL);
```

Рисунок 3.7 – Запуск допоміжних потоків

Після цього виконується спроба підключення до серверу. У разі успішно підключення, система переходить у стан очікування карти для зчитування та циклічно виконує функцію для обробки стану “handleState”. Головний цикл представлено на рисунку 3.8.

```
149     _wifi.connect();
150
151     while (true) {
152         if (!_mqtt.isConnected()) updateState(AppState::NO_CONNECTION);
153         _state_visualizer.setState(_current_state);
154         handleState();
155         vTaskDelay(pdMS_TO_TICKS(10));
156     }
```

Рисунок 3.8 – Підключення до сервера та переходу в стан очікування картки

Основною функцією системи є “handleState” яка виконується в постійному циклі, вона містить розгалуження в залежності від стану системи.

Визначені такі стани системи:

– очікування картки – в цьому стані система постійно зчитує буфер компоненту “RFScanner” на наявність картки, код наведено на рисунку 3.9:

```
case AppState::WAITING_FOR_CARD: {
    RFScanner::UidResult uid = _rf_scanner.getUid();
    if (uid.length > 0) {
        _buzzer.play(MelodyID::CARD_READED);
        onReadCard(uid);
    }
    break;
}
```

Рисунок 3.9 – Обробка стану очікування картки

Якщо картка зчитана, виконується метод “onReadCard”. В ньому UID картки передається на сервер, а система переходить в стан очікування відповіді від сервера “WAITING_SERVER_RESPONSE”, код методу наведено на рисунку 3.10.

```
27  void App::onReadCard(const RFScanner::UidResult &uid) {
28      char json_buffer[128];
29
30      std::string hex_str = Utils::bytesToHexString(uid.data.data(), uid.length);
31
32      ESP_LOGI(TAG, "Read card, HEX value: %s", hex_str.c_str());
33
34      JsonDocument read_card_data;
35      read_card_data["uid"] = hex_str;
36
37      serializeJson(read_card_data, json_buffer, sizeof(json_buffer));
38      _mqtt.publish("/esp32/data", json_buffer);
39
40      _current_card = uid;
41      updateState(AppState::WAITING_SERVER_RESPONSE);
42  }
```

Рисунок 3.10 – Метод обробки зчитаної картки

– очікування відповіді від сервера – цей стан не має функціоналу. Лише містить обмеження по часу, якщо відповідь від сервера не надійшла протягом 30 секунд, система переводиться в стан помилки “ERROR”, його код представлено на рисунку 3.11;

```
297             case AppState::WAITING_SERVER_RESPONSE: {
298                 if (delta_time_state > 30000) updateState(AppState::ERROR);
299                 break;
300             }
```

Рисунок 3.11 – Код очікування відповіді від сервера

– стан помилки – в цьому стані система намагається постійно підключитися до сервера, якщо зв'язок відновлений, система переходить у стан очікування картки. Код стану представлено на рисунку 3.12;

```
301             case AppState::ERROR: {
302                 if (_mqtt.isConnected()) updateState(AppState::WAITING_FOR_CARD);
303                 else vTaskDelay(pdMS_TO_TICKS(500));
304                 break;
305             }
```

Рисунок 3.12 – Код стану помилки

– стан ідентифікації – в цьому стані система намагається знайти та ідентифікувати обличчя, також в ньому передбачено запобіжний ліміт за часом. Якщо обличчя не виявлено або воно не співпадає за відведений проміжок часу, система переходить в стан відмови, робить фото та надсилає його на сервер для можливості подальшого аналізу зі сторони адміністратора. Код обробки цього стану представлено на рисунку 3.13;

```
202         case AppState::FACE_RECOGNITION: {
203             ESP_LOGI(TAG, "Start recognition");
204             if (_current_recognition_data.size() > 0) {
205                 bool success = _recognitor.find(_current_recognition_data, 10000);
206 
207                 if (success) {
208                     char json_buffer[128];
209                     JsonDocument event_data;
210                     event_data["uid"] = Utils::bytesToHexString(_current_card.data.data(), _current_card.length);
211                     event_data["event"] = "enter";
212                     serializeJson(event_data, json_buffer, sizeof(json_buffer));
213                     _mqtt.publish("/esp32/data", json_buffer);
214 
215                     _buzzer.play(MelodyID::GOOD);
216                     updateState(AppState::OPEN_DOOR);
217                 } else {
218                     _buzzer.play(MelodyID::DENIED);
219 
220                     std::string photo_b64 = _camera.captureJpegBase64();
221                     std::string uid_str = Utils::bytesToHexString(_current_card.data.data(), _current_card.length);
222 
223                     const size_t BUF_SIZE = 100 * 1024; // 100 KB
224                     char* denied_buf = new (std::nothrow) char[BUF_SIZE];
225                     if (denied_buf) {
226                         JsonDocument denied_data;
227                         denied_data["uid"]    = uid_str;
228                         denied_data["denied"] = true;
229                         if (!photo_b64.empty()) {
230                             denied_data["photo"] = photo_b64;
231                         }
232 
233                         size_t written = serializeJson(denied_data, denied_buf, BUF_SIZE);
234                         if (written >= BUF_SIZE - 1) {
235                             ESP_LOGE(TAG, "Denied JSON buffer overflow! Photo truncated.");
236                         }
237                         _mqtt.publish("/esp32/data", denied_buf);
238                         ESP_LOGI(TAG, "Denied event sent for %s (photo: %s)",
239                                  uid_str.c_str(), photo_b64.empty() ? "no" : "yes");
240                         delete[] denied_buf;
241                     } else {
242                         ESP_LOGE(TAG, "Failed to allocate denied_buf, sending without photo");
243                         char fallback[128];
244                         JsonDocument denied_data;
245                         denied_data["uid"]    = uid_str;
246                         denied_data["denied"] = true;
247                         serializeJson(denied_data, fallback, sizeof(fallback));
248                         _mqtt.publish("/esp32/data", fallback);
249                     }
250 
251                     updateState(AppState::ACCESS_DENIED);
252                 }
253                 _current_recognition_data.clear();
254             }
255             break;
256         }
```

Рисунок 3.13 – Код стану ідентифікації

– стан навчання – під час цього система намагається зробити 5 відбитків обличчя які пізніше будуть використовуватися для ідентифікації. Якщо навчання успішне, данні відправляються на сервер а система переходить в стан очікування картки. Код стану представлено на рисунку 3.14;

```
257         case AppState::FACE_LEARNING: {
258             char* json_buffer = new (std::nothrow) char[8192];
259             if (!json_buffer) {
260                 ESP_LOGE(TAG, "Failed to allocate memory for JSON buffer");
261                 updateState(AppState::WAITING_FOR_CARD);
262                 break;
263             }
264 
265             auto embeddings = startLearn();
266 
267             if (embeddings.size() == 0) {
268                 _buzzer.play(MelodyID::DENIED);
269                 delete[] json_buffer;
270                 updateState(AppState::WAITING_FOR_CARD);
271                 break;
272             }
273 
274             _buzzer.play(MelodyID::GOOD);
275 
276             JsonDocument learn_data;
277 
278             std::string hex_str = Utils::bytesToHexString(_current_card.data.data(), _current_card.length);
279             learn_data["uid"] = hex_str;
280 
281             JsonArray json_embeddings = learn_data["embeddings"].to<JsonArray>();
282             for (const auto& embedding : embeddings) {
283                 JsonArray json_vector = json_embeddings.add<JsonArray>();
284                 for (float val : embedding) {
285                     json_vector.add(val);
286                 }
287             }
288 
289             size_t written = serializeJson(learn_data, json_buffer, 8192);
290             if (written >= 8191) ESP_LOGE(TAG, "JSON buffer overflow!");
291             _mqtt.publish("/esp32/data", json_buffer);
292 
293             delete[] json_buffer;
294             updateState(AppState::WAITING_FOR_CARD);
295             break;
296         };
```

Рисунок 3.14 – Код стану сканування нового обличчя

– відмова або відкриття дверей – цей стан є перехідним, він необхідний для затримки переходу і після повертається в стан очікування картки. Код представлено на рисунку 3.15.

```
189         case AppState::ACCESS_DENIED: {
190             if (delta_time_state > 3000) updateState(AppState::WAITING_FOR_CARD);
191             break;
192         }
193         case AppState::OPEN_DOOR: {
194             if (delta_time_state < _config.open_relay_timeout_delay) {
195                 _door_relay.set(true);
196             } else {
197                 _door_relay.set(false);
198                 updateState(AppState::WAITING_FOR_CARD);
199             }
200             break;
201         }
```

Рисунок 3.15 – Код відмови або відкриття дверей

Розглянемо код головних модулів більш детальніше. Перший головний модуль це RFScanner. Він використовується для сканування RF карток. Пошук та зчитування відбувається в окремому потоці “rf_scanner_task” який виконує метод “task”. В цьому методі відбувається постійне опитування PN532 з інтервалом 50 мс. Якщо картка знайдена, вона зчитується та її унікальний номер та записується до внутрішнього буфера компонента. Код який виконується потоком представлено на рисунку 3.16.

```
44  void RFScanner::task() {
45      while (true) {
46          uint8_t uid[MAX_LEN_UID] = {0};
47          uint8_t uid_length = 0;
48
49          esp_err_t err = pn532_read_passive_target_id(&_pn532_io, PN532_BRTY_ISO14443A_106KBPS, uid, &uid_length, 50);
50
51          if (err == ESP_OK) {
52              if (_uid_mutex && xSemaphoreTake(_uid_mutex, pdMS_TO_TICKS(100)) == pdTRUE) {
53                  _uid_buffer.fill(0);
54                  std::copy(uid, uid + uid_length, _uid_buffer.begin());
55                  _uid_length = uid_length;
56                  xSemaphoreGive(_uid_mutex);
57              }
58              vTaskDelay(pdMS_TO_TICKS(100));
59          }
60
61          vTaskDelay(pdMS_TO_TICKS(50));
62      }
63  }
```

Рисунок 3.16 – Код методу зчитування з RF картки

Наступна важлива функція знаходиться в модулі “Recognitor”. Це “get_embedding”, це допоміжна функція яка вираховує вектор обличчя без додаткових перевірок на відстань чи якість. Код цього методу представлено на рисунку 3.17.

```
119 std::vector<float> Recognitor::get_embedding(camera_fb_t* frame) {
120     dl::image::img_t img = {
121         .data = frame->buf,
122         .width = (uint16_t)frame->width,
123         .height = (uint16_t)frame->height,
124         .pix_type = dl::image::DL_IMAGE_PIX_TYPE_RGB565BE,
125     };
126
127     auto& detect_results = _face_detector.run(img);
128     if (detect_results.empty()) return {};
129
130     dl::TensorBase* feat = _feat_model.run(img, detect_results.front().keypoint);
131     if (!feat || !feat->data) return {};
132
133     float* data = (float*)feat->data;
134     int len = feat->get_size() / sizeof(float);
135     return std::vector<float>(data, data + len);
136 }
```

Рисунок 3.17 – Код допоміжного методу “get_embedding”

Тепер переходимо до головних методів. Один з таких “find_embedding”, він необхідний для якісного сканування обличчя під час його реєстрації. В ньому є перевірка відстані до обличчя за допомогою вимірювання відстані між очей, яка має бути не меншою за 35 пікселів (визначено експериментальним шляхом). Код методу представлений на рисунку 3.18.

```
61  std::vector<float> Recognitor::find_embedding(int timeout) {
62      uint32_t start_time = esp_timer_get_time() / 1000;
63
64      while ((esp_timer_get_time() / 1000) - start_time < timeout) {
65          camera_fb_t* frame = _camera.capture();
66          if (!frame) { vTaskDelay(pdMS_TO_TICKS(100)); continue; }
67
68          dl::image::img_t img = {
69              .data = frame->buf,
70              .width = (uint16_t)frame->width,
71              .height = (uint16_t)frame->height,
72              .pix_type = dl::image::DL_IMAGE_PIX_TYPE_RGB565BE,
73          };
74
75          auto& detect_results = _face_detector.run(img);
76          if (detect_results.empty()) {
77              _camera.release_frame(frame);
78              vTaskDelay(pdMS_TO_TICKS(50));
79              continue;
80          }
81
82          auto& face = detect_results.front();
83
84          if (face.score < 0.75f) {
85              ESP_LOGD(TAG, "Bad face visible (score: %.2f)", face.score);
86              _camera.release_frame(frame);
87              vTaskDelay(pdMS_TO_TICKS(50));
88              continue;
89          }
90
91          float dx = face.keypoint[1] - face.keypoint[0];
92          float dy = face.keypoint[3] - face.keypoint[2];
93          float eye_dist = std::sqrt(dx*dx + dy*dy);
94
95          if (eye_dist < 35.0f) {
96              ESP_LOGD(TAG, "Face is too far");
97              _camera.release_frame(frame);
98              vTaskDelay(pdMS_TO_TICKS(50));
99              continue;
100         }
101
102         dl::TensorBase* feat = _feat_model.run(img, face.keypoint);
103         _camera.release_frame(frame);
104
105         if (!feat || !feat->data) {
106             vTaskDelay(pdMS_TO_TICKS(50));
107             continue;
108         }
109
110         float* data = (float*)feat->data;
111         int len = feat->get_size() / sizeof(float);
112
113         return std::vector<float>(data, data + len);
114     }
115
116     return {};
117 }
```

Рисунок 3.18 – Код методу “find_embedding”

Остання важлива функція це “find”. Вона відповідає за ідентифікацію обличчя через вектори які були збережені з етапу навчання. Код представлено на рисунку 3.19.

Порівняння точок відбувається через функцію обчислення конусної подібності “cosine_similarity”. Код функції представлено на рисунку 3.20.

```
bool Recognitor::find(const std::vector<std::vector<float>>& embedding_base, int timeout) {
    if (embedding_base.empty()) {
        ESP_LOGI(TAG, "Base empty(");
        return false;
    }
    uint32_t start_time = esp_timer_get_time() / 1000;

    while ((esp_timer_get_time() / 1000) - start_time < timeout) {
        camera_fb_t* frame = _camera.capture();
        if (!frame) { vTaskDelay(pdMS_TO_TICKS(100)); continue; }

        auto frame_embedding = get_embedding(frame);
        _camera.release_frame(frame);

        if (frame_embedding.empty()) { vTaskDelay(pdMS_TO_TICKS(5)); continue; }

        float max_score = 0.0f;
        for (const auto& target_emb : embedding_base) {
            if (target_emb.size() != frame_embedding.size()) continue;

            float score = Utils::cosine_similarity(frame_embedding.data(), target_emb.data(), frame_embedding.size());
            if (score > max_score) max_score = score;
        }

        ESP_LOGI(TAG, "Best score: %.4f", max_score);

        if (max_score > _min_face_score) {
            return true;
        }

        vTaskDelay(pdMS_TO_TICKS(10));
    }
    return false;
}
```

Рисунок 3.19 – Код методу “find”

```
float Utils::cosine_similarity(const float* a, const float* b, int len) {
    float dot = 0, norm_a = 0, norm_b = 0;
    for (int i = 0; i < len; i++) {
        dot    += a[i] * b[i];
        norm_a += a[i] * a[i];
        norm_b += b[i] * b[i];
    }
    if (norm_a == 0.0f || norm_b == 0.0f) return 0.0f;
    return dot / (sqrtf(norm_a) * sqrtf(norm_b));
}
```

Рисунок 3.20 – Код функції “cosine_similarity”

Функція мережевої взаємодії також є простою, коли система знаходиться в статі очікування відповіді від сервера, вона починає слухати команди керування, такі як “access_denied”, “start_recognition”, “start_learn”.

Та виконує дії в залежності від цих команд. Код на рисунку 3.21.



```
void App::handleMqttMessage(const MqttMessage &msg) {
    ESP_LOGI(TAG, "[%s] %s", msg.topic.c_str(), msg.data.c_str());

    if (msg.topic == "/esp32/cmd") {
        auto buffer_ptr = std::make_unique<JsonDocument>();
        JsonDocument& buffer = *buffer_ptr;

        DeserializationError error = deserializeJson(buffer, msg.data.c_str());

        if (error) {
            ESP_LOGE(TAG, "Error JSON: %s", error.c_str());
            return;
        }

        if (buffer["cmd"].is<const char*>()) {
            const char* cmd = buffer["cmd"];

            if (getState() == AppState::WAITING_SERVER_RESPONSE) {
                if (strcmp(cmd, "access_denied") == 0) {
                    _buzzer.play(MelodyID::DENIED);
                    updateState(AppState::ACCESS_DENIED);
                }

                if (strcmp(cmd, "start_recognition") == 0) {
                    _current_recognition_data.clear();

                    if (buffer["embeddings"].is<JsonArray>()) {
                        JsonArray json_embeddings = buffer["embeddings"].as<JsonArray>();
                        _current_recognition_data.reserve(json_embeddings.size());

                        for (JsonArray json_vector : json_embeddings) {
                            std::vector<float> embedding;
                            embedding.reserve(json_vector.size());

                            for (float val : json_vector) {
                                embedding.push_back(val);
                            }

                            _current_recognition_data.emplace_back(std::move(embedding));
                        }

                        ESP_LOGI(TAG, "Successfully unpacked %d embeddings from heap memory",
                                 _current_recognition_data.size());
                        updateState(AppState::FACE_RECOGNITION);
                    } else {
                        ESP_LOGW(TAG, "Command 'start_recognition' received, but 'embeddings' is missing!");
                        updateState(AppState::WAITING_FOR_CARD);
                    }
                }

                if (strcmp(cmd, "start_learn") == 0) {
                    updateState(AppState::FACE_LEARNING);
                }
            }
        }
    }
}
```

Рисунок 3.21 – Код обробки відповіді від серверу

Завдяки такій модульній архітектурі можливо легко тестувати та впроваджувати новий функціонал в систему.

Для збірки прошивки необхідно визначити конфігураційний файл розмітки. В ньому необхідно вказати додаткові розділи для моделей нейромереж. Це робиться в файлі “partitions.csv”. Зміст файлу на рисунку 3.22.

```
# Name,          Type, SubType,  Offset,   Size,    Flags
nvs,             data, nvs,      0x9000,   0x6000,
phy_init,        data, phy,      0xf000,   0x1000,
factory,         app,  factory,  0x10000,  0x300000,
human_face_det,  data, 0x40,     0x310000, 0x080000,
human_face_feat, data, 0x41,     0x390000, 0x160000,
storage,         data, spiffs,   0x4F0000, 0x100000,
```

Рисунок 3.22 – Вигляд файлу розмітки “partitions.csv”

Для завантаження моделей на контролер створимо окремий файл “flash_models.sh”. Моделі необхідно завантажити тільки один раз, якщо не змінювати файл розмітки. Зміст файлу на рисунку 3.23.

```
echo "Flashing face detect model..."
esptool.py --chip esp32s3 --port $PORT \
  write_flash 0x310000 \
  managed_components/espressif__human_face_detect/models/s3/human_face_detect_msr_s8_v1.espdl
esptool.py --chip esp32s3 --port $PORT \
  write_flash 0x340000 \
  managed_components/espressif__human_face_detect/models/s3/human_face_detect_mnp_s8_v1.espdl

echo "Flashing face feat model..."
esptool.py --chip esp32s3 --port $PORT \
  write_flash 0x390000 \
  managed_components/espressif__human_face_recognition/models/s3/human_face_feat_mfn_s8_v1.espdl

echo "Done!"
```

Рисунок 3.23 – Вигляд допоміжного файлу для завантаження моделей

Також необхідно включити параметр, який буде використовувати нашу розмітку та визначити місцезнаходження моделей через конфігураційне меню ESP-IDF. Для відкриття меню необхідно ввести команду “idf.py menuconfig”. Зовнішній вигляд меню зображено на рисунку 3.24.

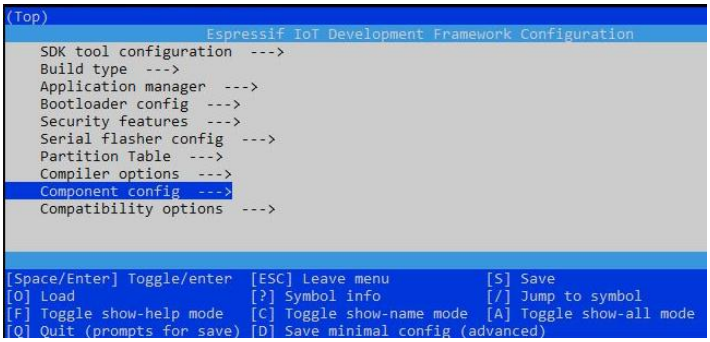


Рисунок 3.24 – Меню налаштування фреймворку IDF

3.2 Розробка програми для серверу

Архітектура сервера повинна включати в собі веб-інтерфейс для адміністратора та API інтерфейс для мікроконтролера. Також архітектура повинна легко розділятися на контейнери. Це зробить всю систему більш надійною та легкою в масштабуванні. Для цієї цілі обрана мова програмування Python яка часто використовується в промислових рішеннях такого рівня.

3.2.1 Визначення та підключення необхідних бібліотек

Для розробки коду серверу необхідно визначити бібліотеки та фреймворки. Для задач з управління доступом та менеджменту персоналу буде достатньо такого стеку:

– Docker – це платформа яка дозволяє ізолювати застосунки в стандартизований блок. Це робить проєкт більш стійким та його легко налагоджувати на інших пристроях [32];

– Eclipse Mosquitto – це брокер повідомлень який реалізує протокол MQTT. Він діє як центральний хаб, отримуючи повідомлення від різних пристроїв та пересилає їх [33];

– Flask – є популярним фреймворком для створення простих веб додатків. Буде використовуватися для панелі керування базою даних [34];

– gmqtt – це асинхронна бібліотека для протоколу MQTT. Вона буде використовуватися для обробки повідомлень від клієнту (ESP32) [35];

– Openpyxl – це бібліотека для роботи з таблицями в форматі Excel. Буде використовуватися для формування таблиць з панелі адміністратора [36].

3.2.2 Розробка алгоритму

Перед розробкою програмного коду був розроблений основний алгоритм представлений на рисунку 3.25.

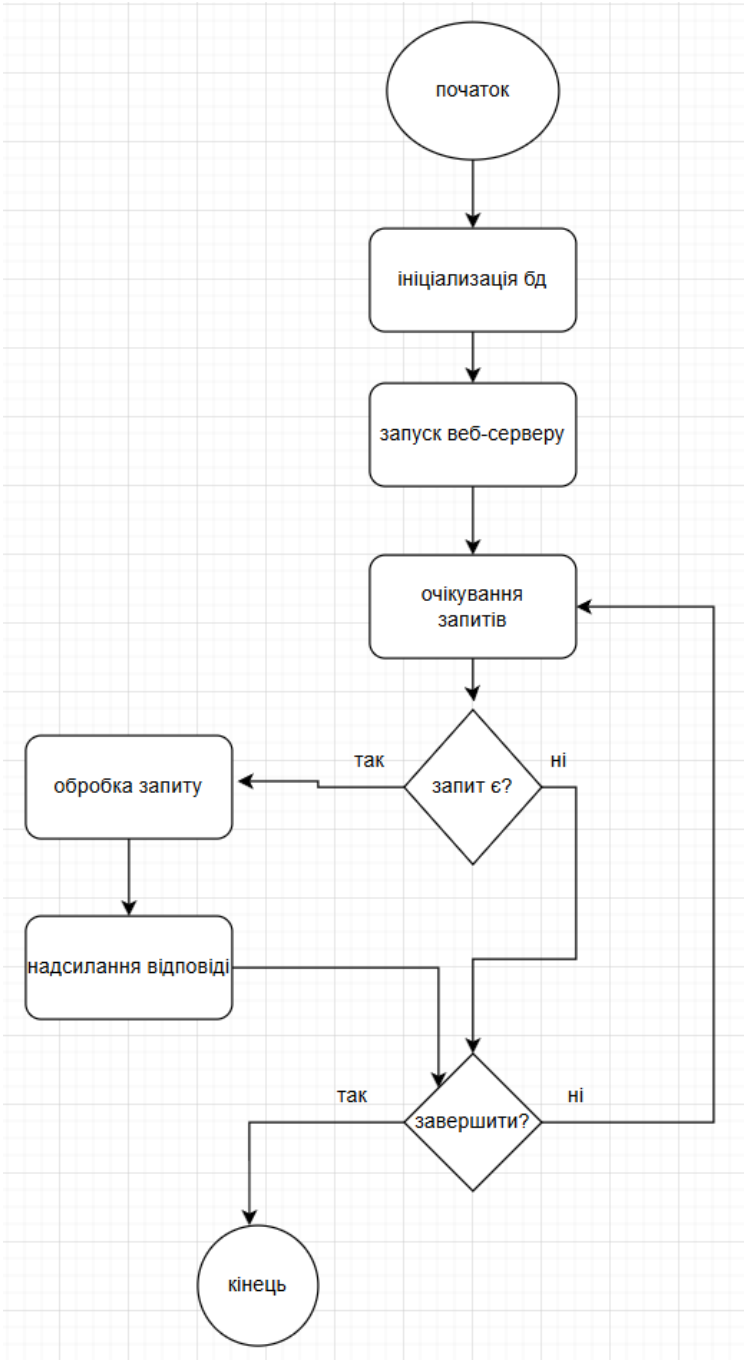


Рисунок 3.25 – Алгоритм для серверу

3.2.3 Налаштування середовища та розробка програмного коду

Спочатку необхідно налаштувати Docker. В файлі конфігурації необхідно описати взаємодію контейнерів, необхідні порти, та файли бази даних. Після внесення всіх параметрів збережемо в файл “docker-compose.yml”: Зміст цього файлу представлено на рисунку 3.26.

```yaml
version: '3.8'

services:
  mqtt-broker:
    image: eclipse-mosquitto:latest
    container_name: mqtt_broker
    ports:
      - "1883:1883"
    command: mosquitto -c /mosquitto-no-auth.conf

  backend:
    build: ./backend
    container_name: python_backend
    depends_on:
      - mqtt-broker
    environment:
      - MQTT_BROKER_HOST=mqtt-broker
      - MQTT_BROKER_PORT=1883
      - FLASK_DEBUG=1
    ports:
      - "5000:5000"
    volumes:
      - db_data:/data
      - ./backend:/app
    restart: always

volumes:
  db_data:

```

Рисунок 3.26 – Файл налаштувань Docker

Далі необхідно заповнити файл для підключення бібліотек requirements.txt, його фінальний вигляд представлено на рисунку 3.27.

```
gmqtt==0.7.0
flask==3.1.3
openpyxl==3.1.5
```

Рисунок 3.27 – Файл для підключення Python бібліотек

Після цього можливо приступити до написання програмного коду для сервера.

Основним компонентом для взаємодії сервера та веб інтерфейсу адміністратора є Flask. Вся взаємодія відбувається через “роути”. Це дозволяє

спростити код та дозволить його в майбутньому ділити на різні потоки. Приклад функції відображення головної сторінки з файлу “web.py” зображений на рисунку 3.28.

```python
@app.route("/")
def index():
    users = db.get_all_users()
    for u in users:
        u["pass_count"] = db.get_pass_count(u["card_uid"])
        last = db.get_last_pass(u["card_uid"])
        u["last_pass"] = last["ts"] if last else "—"
        u["trained"] = db.has_embeddings(u["card_uid"])
    return render_template("index.html", users=users)
```

Рисунок 3.28 – Функція формування головної HTML сторінки

Веб архітектура будується завдяки шаблонним сторінкам без даних, завдяки яким Flask генерує заповнену необхідними даними HTML сторінку. Визначено 5 таких шаблонів:

– base.html – допоміжний шаблон, містить елементи які мають бути однаковими для усіх сторінок, а саме меню, футер, і загальні стилі для всієї панелі. Код представлено на рисунку 3.29;

backend > templates > base.html > html

```html
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Панель | {% block title %}{% endblock %}</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/base.css') }}">
    {% block styles %}{% endblock %}
</head>
<body>

<header>
    <h1 class="header-logo">Дипломна робота</h1>
    <nav class="main-menu">
        <a href="/" class="main-menu-item">Головна</a>
        <a href="/log" class="main-menu-item">Логи</a>
        <a href="/time" class="main-menu-item">Час</a>
    </nav>
</header>

<main class="main-block">
    {% block content %}
    {% endblock %}
</main>

<footer>
    <p>Підсистема автоматизованого доступу до виробничих приміщень та обліку робочого часу персоналу</p>
    <p>Гарасимчук Ігор АКТСІ-22-1 2026</p>
</footer>
{% block scripts %}{% endblock %}
</body>
</html>
```

Рисунок 3.29 – Файл допоміжного шаблону

– index.html – містить стартову сторінку, має функціонал для додавання нової картки а також список всіх карток які є в базі даних. Вигляд сторінки зображений на рисунку 3.30;

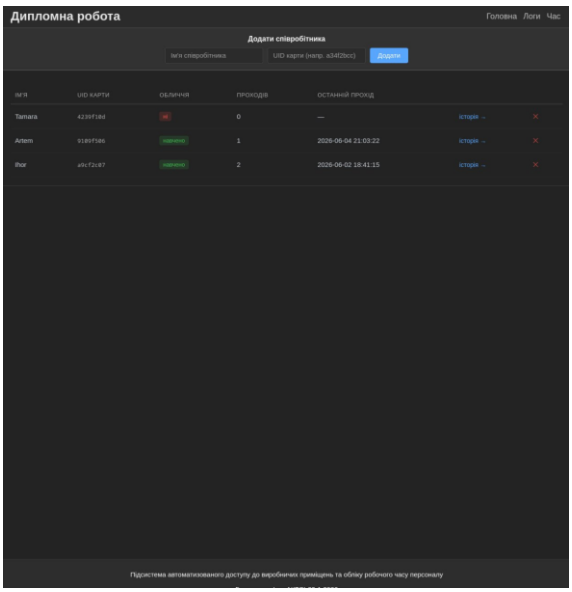


Рисунок 3.30 – Стартова сторінка

– log.html – містить інформацію про всі останні дії з точки проходу, наприклад зчитування невідомої картки. Вигляд сторінки зображений на рисунку 3.31;

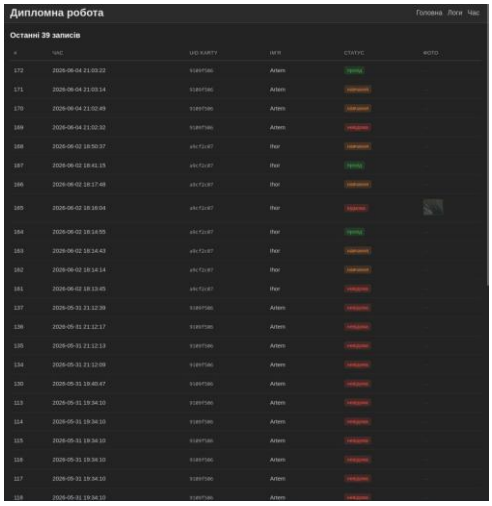


Рисунок 3.31 – Сторінка з загальними записами

– time.html – містить список всіх карток, статус робітників а також кількість годин які були проведені на роботі. Також має фільтри за часом та дозволяє сформувати Excel таблицю з усіма даними. Вигляд сторінки зображено на рисунку 3.32 та готову таблицю на рисунку 3.33;

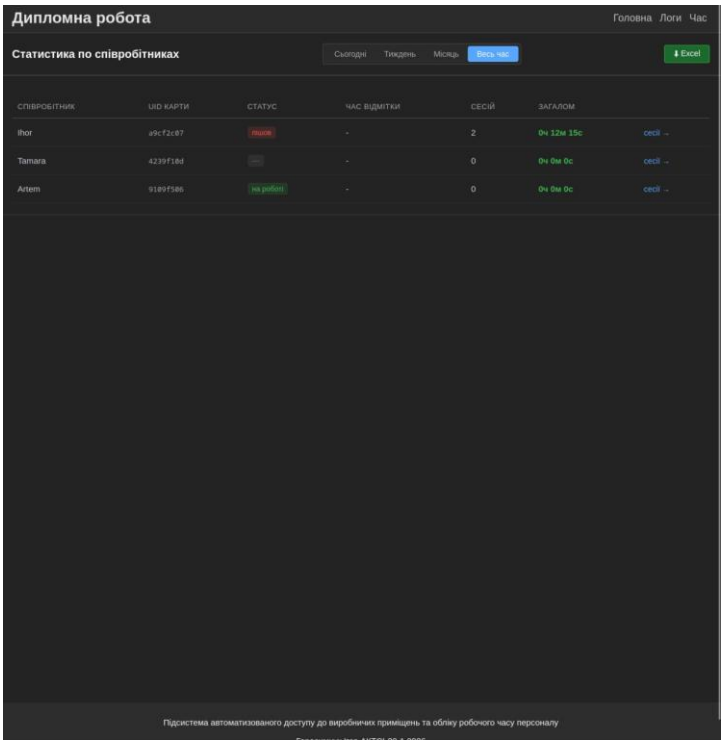


Рисунок 3.32 – Сторінка з записом часу кожного робітника

	A	B	C	D	E	F	G
1	Статистика робочого часу — Весь час						
2							
3	№	Співробітник	UID карти	Статус	Сесій	Загальний час	
4	1	Ihor	a9cf2c07	Пішов	2	0ч 12м 15с	
5	2	Tamara	4239f10d	—	0	0ч 0м 0с	
6	3	Artem	9109f506	На роботі	0	0ч 0м 0с	
7							

Рисунок 3.33 – Сформована таблиця на основі даних з системи

– user.html – містить схожу на попередню таблицю, але вже по конкретній картці, також має фільтри та можливість сформувати Excel таблицю. Вигляд сторінки зображено на рисунку 3.34;

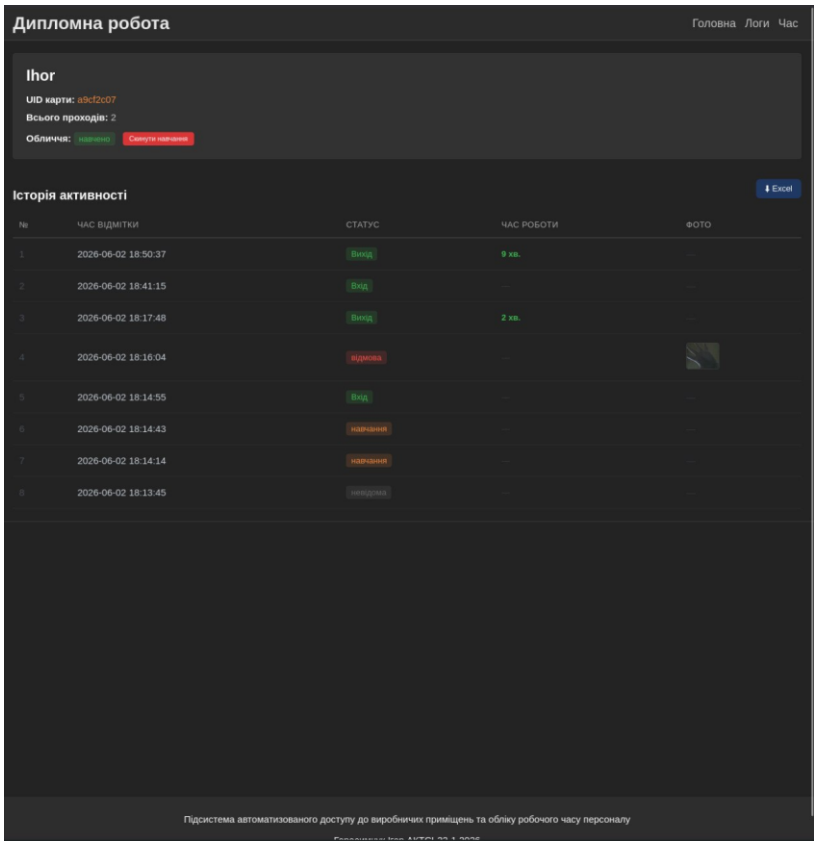


Рисунок 3.34 – Сторінка з інформацією про конкретного робітника

За взаємодію з ESP32 відповідає файл “server.py”. Тут відбувається обробка команд від мікроконтролера та формування відповіді. Приклад повідомлення для перевірки зв’язку представлено на рисунку 3.35.

```
19  async def on_message(client, topic, payload, qos, properties):
20      try:
21          message = payload.decode()
22      except Exception:
23          return
24
25      if topic == "/esp32/status" and message == "online":
26          publish(client, {"cmd": "ping"})
27          return
```

Рисунок 3.35 – Обробка повідомлення для перевірки зв’язку

Для роботи з базою даних створений модуль “database.py”, в ньому є усі необхідні функції для додавання, модифікації та формування даних [37].

3.3 Висновки до розділу

В третьому було розроблене програмне забезпечення для мікроконтролера та серверу яке виконує поставлене технічне завдання та дозволяє розпочати етап тестування. Прошивка мікроконтролера дозволяє швидко ідентифікувати особистість та реагувати на втрату зв’язку з сервером.

Серверна частина включає як API для мікроконтролера, так і веб-панель для адміністратору. Адміністратор має можливість додавати, видаляти картки, отримувати дані про спроби компрометації та формувати звіти в форматі Excel таблиць, що дозволяє легко передавати ці дані в інші системи.

Розроблене програмне забезпечення повністю виконує поставлені вимоги для систем СКУД та побудоване на архітектурі яку легко модифікувати та тестувати.

4 ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Експеримент з енергоефективністю

Перший експеримент визначає енергоефективність системи в різних станах. Вимірювання проводилося за допомогою мультиметра підключеного послідовно в ланцюг живлення системи. Вигляд процесу тестування зображено на рисунку 4.1, а результати записані в таблицю 4.1.

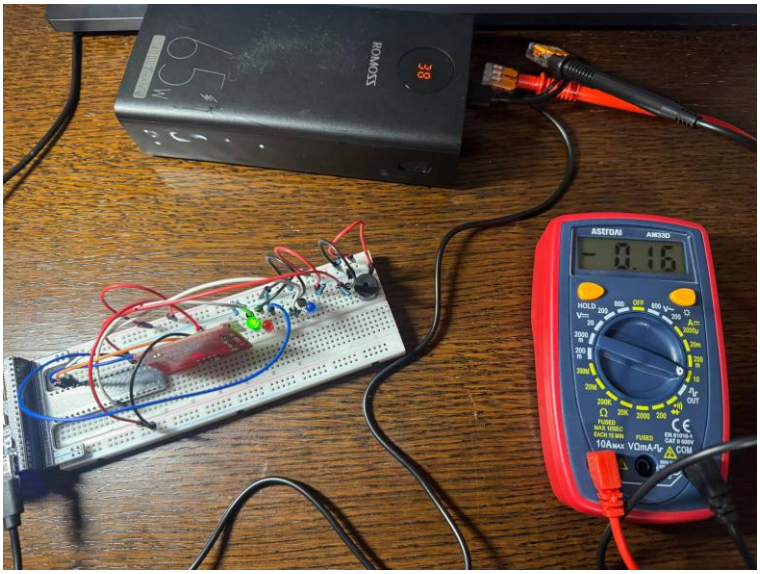


Рисунок 4.1 – Процес вимірювання енергоспоживання

Таблиця 4.1 – Вимірювання енергоспоживання в різних станах

Режим роботи	Споживання (мА)
Запуск системи	300
Очікування картки	160
Сканування картки	250
Розпізнавання обличчя	200
Спрацювання замка	260

Найменше споживання спостерігається в режимі очікування, що є позитивним показником, оскільки система проводить більшість часу в саме в цьому стані. Найбільше споживання фіксується під час запуску системи та імітації спрацювання електромагнітного замка, що пояснюється короткочасним пусковим струмом відповідних компонентів.

За результатами вимірювань середнє споживання струму під час роботи становить близько 234 мА без електромагнітного замку. За умови використання повербанку ємкістю 10 000 мАч, розрахунковий час автономної роботи приблизно 34 години з урахуванням ККД перетворювача напруги 80%.

Також це доводить, що вибір перетворювача напруги Mini 360 є коректним, оскільки його потужність становить 1,2 А, залишається близько 900 мА запасу.

4.2 Експеримент з освітленням

Другий експеримент визначає яка мінімальна освітленість необхідна для ідентифікації особистості а також залежність освітленості до швидкості розпізнавання.

Всі вимірювання проводились в однакових умовах, відстань від обличчя до камери 40 см. Швидкість обміну даних з сервером не враховувалась, тестувалась саме швидкість локальної ідентифікації. Для підвищення рівня достовірності результатів проводилось 5 вимірювань та розраховувалось середнє арифметичне на кожному рівні освітлення.

В ході дослідження визначено, що мінімальним освітленням за якого система може виконати ідентифікацію є 78 люкс. При цьому тривалість ідентифікації склала 4.4 секунди. Також за таких умов суттєво зростає вірогідність помилкового розпізнавання або відмови в доступі, що робить цей

режим критичним для практичного використання. Дані вимірювань наведені в таблиці 4.2 а фото зроблене при мінімальному освітленні на рисунку 4.2.

Таблиця 4.2 – Вимірювання швидкості ідентифікації при різних рівнях освітлення.

Освітлення (люкс)	Швидкість ідентифікації (секунд)
78	4,4
143	3,6
350	2,9
400	2,7
550	2,4
650	2,2



Рисунок 4.2 – Зображення з камери при мінімальному освітленні 78 люкс

Після цього тесту стало зрозуміло що необхідно додати можливість вмикати додаткове освітлення. Для цього схему було модифіковано, а саме були доданий ще один MOSFET-транзистор та клеми для підключення підсвітки.

Живлення підсвітки відбувається від 12 вольт та рекомендованим максимальним током є половина потужності від зовнішнього блока живлення Також був модифікований код, тепер при розпізнаванні вмикається підсвітка. Розташування підсвітки заплановане на корпусі пристрою та залежить від вимог підприємства. Нова схема з можливістю підключення підсвітки зображена на рисунку 4.3.

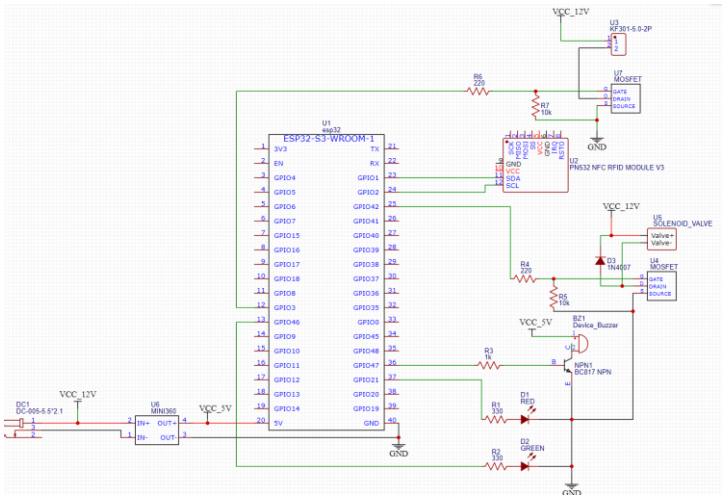


Рисунок 4.3 – Покращена схема підсистеми

Для тесту ефективності підсвітки був підключений модуль HK-LEDPCB-12V-X. Характеристики модуля представлені в таблиці 4.3 а його зображення на рисунку 4.4.

Таблиця 4.3 – Характеристики підсвітки HK-LEDPCB-12V-X [38]

Параметр	Значення
Робоча напруга	12-24 В
Діаметр плати	30 мм
Колір	Білий (5500 – 6000К)
Кут випромінювання	140 градусів
Світловий потік	200 люменів

Продовження таблиці 4.3

Параметр	Значення
Потужність	3 Вт
Розрахунковий струм при 12В	250 мА
Ціна	200 грн.



Рисунок 4.4 – Зображення модуля HK-LEDPCB-12V-X

Після додавання модуля було проведене повторне дослідження, умови не змінилися. Результати наведені в таблиці 4.4 та графіку на рисунку 4.5.

Таблиця 4.4 – Порівняльна таблиця після додавання підсвітки

Освітлення (люкс)	Швидкість без підсвітки (с)	Швидкість з підсвіткою (с)	Покращення (%)
78	4,4	3,1	29,5
143	3,6	2,7	25,0

Продовження таблиці 4.4

Освітлення (люкс)	Швидкість без підсвітки (с)	Швидкість з підсвіткою (с)	Покращення (%)
350	2,9	2,4	17,2
400	2,7	2,3	14,8
550	2,4	2,2	8,3
650	2,2	2,1	4,5

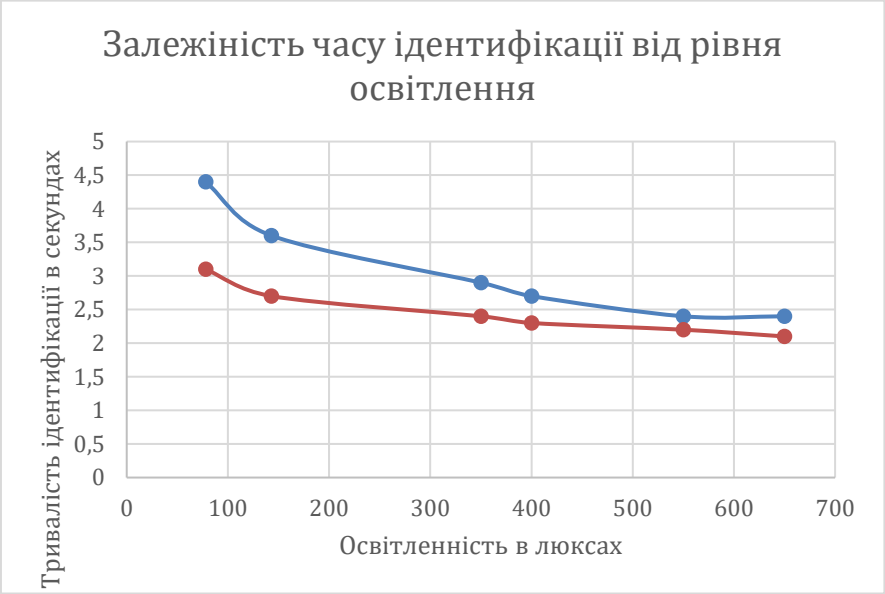


Рисунок 4.5 – Графік швидкості розпізнавання в залежності від освітлення (синя лінія – без підсвітки, оранжева – з підсвіткою)

Дослідження демонструє, що мінімальним рівнем освітлення для ідентифікації є 78 люкс, але при цьому зростає тривалість та вірогідність помилки. Також, немає сенсу підвищувати рівень освітлення після 550 люкс, оскільки система досягає свого максимуму. Тому рекомендованим рівнем встановлення без підсвітки є 350 люкс.

Додавання підсвітки підвищило швидкість розпізнавання на всіх рівнях освітленості. Але найбільший ефект спостерігається на низькому та середньому рівнях (17,2%), в високих рівнях підсвітка менш ефективна(4,5%).

4.3 Експеримент з пропускної здатності

Третій експеримент визначає скільки людей система може пропустити за одну хвилину. Алгоритм дії кожної людини на кожній ітерації: людина підходить, прикладає картку, дивиться в камеру, відкриття замка, людина проходить, замок закривається, підходить наступна людина. Експеримент проводився з високим рівнем освітлення. Для отримання статистично достовірних даних було виконано серію з 20 безперервних ітерацій проходу та визначено середнє арифметичне.

За результатами вимірювань встановлено що повний цикл проходу однієї людини в середньому триває 11,2 секунди. Отже розрахункова пропускна здатність системи складає 5,3 особи за хвилину або приблизно 320 осіб на годину.

Отримані дані дозволяють рекомендувати розроблену систему на об’єктах із низькою та середньою інтенсивністю руху. Для об’єктів із потоком понад 5 осіб на хвилину рекомендується розглянути встановлення кількох паралельних точок контролю доступу.

4.4 Охорона праці

Однією з найважливіших етапів професійної діяльності, є забезпечення належного рівня охорони праці. Організація належного рівня безпеки є обов’язковою в сучасному світі. Це запобігає зниженню шкідливого впливу виробничих факторів на здоров’я працівників, підвищує ефективність праці та уникнення аварійних ситуацій [39].

Цей розділ містить характеристику середовища де виконувались роботи, також фактори можливої небезпеки та засоби їх усунення.

4.4.1 Характеристика робочого середовища

Розробка підсистеми здійснювалась у лабораторному яке відповідає необхідним вимогам. Площа приміщення становила 30 м^2 а об’єм 100 м^3. Також в приміщенні розміщувались комп’ютери та необхідне обладнання для монтажу мікроелектроніки. Одночасно в приміщенні перебувало не більше п’яти осіб що відповідає нормі – не менше 6 м^2 та 20 м^3 на одного працівника.

Роботи включали пайку компонентів, їх підключення та тестування на макетній платі. Також створення програмного коду та фінальне тестування системи в різних умовах.

4.4.2 Мікроклімат і освітлення

Температурний режим у приміщенні підтримувався в межах 22-25 °C, відносна вологість становила 45-60%, швидкість руху повітря не перевищувала 0.1 м/с, що відповідає нормам для категорії легкових робіт. Освітлення поєднувало природне (вікна площею 9м^2) та штучне (лампи білого світла), рівень освітленості на робочих поверхням був не менш ніж 300 люкс, чого повністю достатньо для таких робіт.

4.4.3 Ідентифікація шкідливих виробничих факторів

У ході виконання роботи виявлені шкідливі фактори які є специфічними для апаратно-програмної розробки:

– статична електрика – при монтажі електроніки є ризик її пошкодження розрядом статики яка накопичується на тілі людини. Цей ризик виключено використанням антистатичного браслету та заземленого інструменту;

– термічні фактори – основним ризиком є паяльник, який нагрівається до високої температури, також можливе перегрівання компонентів при тривалій роботі. Використовується спеціальна термостійкий килим та підставка для паяльника, та монтажний пінцет для утримання компонентів;

– зорове навантаження – тривала робота з дрібними електронними компонентами та з текстом коду на моніторі сильно напружують очі. Дотримано перерву кожні 45 хвилин;

– хімічний фактор – на деяких електронних компонентах відсутні роз’єми для підключення, тому їх необхідно паяти. Для пайки використовується флюс який при нагріванні виділяє шкідливі хімічні продукти небезпечні для дихання. Необхідне регулярне провітрювання приміщення.

4.4.4 Електробезпека

Макет підсистеми живиться від USB-порту ПК через вбудований стабілізатор плати. Оскільки в макеті плати силова лінія 12 В та стабілізатор напруги замінено імітуючим світлодіодом, риск враження током суттєво зменшується.

У промисловому варіанті підсистеми, де передбачено живлення замка та підсвітки від 12 В через MOSFET-транзистор, усі з’єднання з мережею 220 В виконуються через підібраний сертифікований блок живлення з захистом від короткого замикання та перевантаження. Підключенням компонентів здійснюється через спеціальний безпечний роз’єм при повністю знеструмленій системі.

4.4.5 Пожежна безпека

Основним джерелом займання в лабораторії є можливий перегрів електронних компонентів при помилковому підключенні.

В приміщенні встановлено порошкові вогнегасники у доступних місцях. Проведено інструктаж щодо правил пожежної безпеки при роботі з електронним обладнанням. Шляхи евакуації позначені та вільні.

4.5 Висновки до розділу

В четвертому розділі виконано комплексне тестування ключових параметрів розробленої підсистеми та виконаний аналіз лабораторного приміщення де виконувались роботи.

У першому експерименті було проведено тестування з енергоспоживанням системи та розрахована тривалість роботи від акумулятору. Середній рівень споживання становить 234 мА.

У другому експерименті визначений мінімальний та рекомендований рівень освітленості для ідентифікації особи а також встановлена залежність між освітленням та швидкістю розпізнавання. Також схему було модифіковано, додана можливість вмикання додаткової підсвітки. Після цього експеримент було проведено повторно. За результатами визначено, мінімальний рівень ідентифікації є 78 люкс, а рекомендований 350 люкс. Після рівня 550 люкс тривалість не зменшується, оскільки система досягає технічного максимуму. Також встановлено, що використання підсвітки є ефективним при мінімальному або середньому рівні освітленості, оскільки при підвищенні рівня освітленості приміщення, різниця зменшується (з 17,2% до 4,5%).

Третій експеримент дозволив визначити пропускну здатність однієї точки при повному циклі проходу особи. Експеримент продемонстрував що одна точка проходу може обслуговувати до 5 людей за хвилину при гарному рівні освітлення. Якщо необхідна більша пропускна можливість, необхідно встановлювати декілька точок проходу, що підтримується архітектурою підсистеми.

Далі був проведений аналіз умов охорони праці. За результатами встановлено, що всі параметри в межах встановлених нормативних вимог та усі проблеми вирішені відповідними технічними засобами. Результати наведено в таблиці 4.5.

Таблиця 4.5 – Ключові параметри робочого середовища

Параметр	Значення / Захід
Площа приміщення	30м^2
Об’єм приміщення	100м^3
Кількість працівників	До 5 осіб одночасно
Температура повітря	22-25 ℃
Вологість повітря	45-60%
Швидкість повітря	≤ 0,1 м/с
Природне освітлення	Вікна площею 9 м^2
Штучне освітлення	Освітлювальні прилади, > 300 лк
Фізичні фактори	Фактор опіку, статична електрика
Психофізіологічні фактори	Статика, зорове навантаження, стрес
Хімічні/Біологічні фактори	Пари флюсу та каніфолі при паянні.
Вентиляція	Природна + примусова
Електробезпека	Заземлення, стабілізатори, автоматичні вимикачі
Пожежна безпека	Порошкові вогнегасники, інструктаж, евакуаційний план

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи була мета реалізувати підсистему автоматизованого доступу робітничого персоналу та обліку часу за допомогою сучасних методів ідентифікації. Система призначена для автоматизованого використання в пунктах пропуску персоналу.

У першому розділі проаналізовані існуючи рішення СКУД. Визначено що використання двох-факторного підходу забезпечує високу швидкість, зручність та безпеку та встановлено що необхідно використовувати саме мережеву архітектуру.

У другому розділі була спроєктована структурна схема, був проведений аналіз існуючих компонентів та обрані які найкраще підходять для підсистеми. Після була створена схема підключення та зібраний макет.

У третьому розділі було створене програмне забезпечення для мікроконтролера та сервера яке містить весь необхідний функціонал.

У четвертому розділі було проведено три експерименти. В першому була перевірена енергоефективність в різних режимах роботи. В другому були проведені експерименти в різних умовах освітленості та перевірена необхідність додавання підсвітки. В третьому була протестована пропускна здатність системи. В кінці був виконаний аналіз рівня охорони праці, знайдені основні ризики та методи захисту від них.

Результатом роботи є сучасна, ефективна та доступна підсистема контролю доступу та обліку робочого часу яка може бути використана на підприємствах де необхідний середній рівень безпеки.

Вимоги поставлені в вступі виконані, система може бути встановлена для тестування на реальних пропускних пунктах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти денної і заочної форми навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І.Ш. Невлюдов, О.І. Филипенко, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2023. 64 с.

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ: «УкрНДНЦ», 2016. 26 с.

3. Що таке СКУД і як це працює? [Електронний ресурс] – Режим доступу: https://u-prox.systems/yak-praczyuye-sistema-kontrolyu-dostupu-skd-abo-skud/ (дата звернення 27.04.2026).

4. Системи контролю і управління доступом від А до Я Bolt [Електронний ресурс] / – Режим доступу: https://deps.ua/ua/knowegable-base/reference-information/7824.html (дата звернення 27.04.2026).

5. ZKTeco F18 Product Page [Електронний ресурс] / – Режим доступу: https://zkteco.com/en/FingerprintAeH/F18 (дата звернення 27.04.2026).

6. Suprema BioEntry W2 [Електронний ресурс] / – Режим доступу: https://www.supremainc.com/en/hardware/compact-outdoor-fingerprint-device-bioentry-w2.asp (дата звернення 27.04.2026).

7. Arduino UNO – популярна плата розробки [Електронний ресурс] / – Режим доступу: https://itmaster.biz.ua/directory/kits-nabory/arduino-uno.html (дата звернення 27.04.2026).

8. Детальний гід по ESP32: архітектура, характеристика, застосування [Електронний ресурс] / – Режим доступу: https://masterlexon.com/esp32-introduction-architecture-applications-ua (дата звернення 27.04.2026).

9. Datasheet STM32F405xx STM32F407xx [Електронний ресурс] / – Режим доступу: https://www.st.com/resource/en/datasheet/dm00037051.pdf (дата звернення 28.04.2026).

10. Raspberry Pi 4 Model B 8GB [Електронний ресурс] / – Режим доступу: https://arduino.ua/prod4005-raspberry-4-8gb (дата звернення 28.04.2026).

11. Freenove. Freenove ESP32-S3 WROOM Board User Manual. 2023 [Електронний ресурс] / – Режим доступу: https://github.com/Freenove/Freenove_ESP32_S3_WROOM_Board (дата звернення 28.04.2026).

12. NXP Semiconductors. PN532 User Manual. Rev. 02. 2007 [Електронний ресурс] / – Режим доступу: https://www.nxp.com/docs/en/user-guide/141520.pdf (дата звернення: 06.05.2026).

13. RC522 – Easy Start with RFID [Електронний ресурс] / – Режим доступу: https://easyelecmodule.com/rfid-mfrc522/ (дата звернення 06.05.2026).

14. PN5180 Explained: The Ultimate NFC Module with Full Protocol Compatibility [Електронний ресурс] / – Режим доступу: https://easyelecmodule.com/pn5180-explained-the-ultimate-nfc-module-with-full-protocol-compatibility/ (дата звернення 06.05.2026).

15. OmniVision Technologies. OV2640 Camera Module Software Application Notes. 2006 [Електронний ресурс] / – Режим доступу: https://www.uctronics.com/download/cam_module/OV2640DS.pdf (дата звернення 08.05.2026).

16. Модуль камери OV3660 ESP32 [Електронний ресурс] / – Режим доступу: https://ua.cameramodule.com/fpc-camera-module/mipi-camera-module/ov3660-esp32-camera-module.html (дата звернення 08.05.2026).

17. Waveshare OV5640 5MP Camera Board [Електронний ресурс] / – Режим доступу: https://evo.net.ua/ua/waveshare-ov5640-5mp-camera-board/ (дата звернення 08.05.2026).

18. Single door magnetic lock [Електронний ресурс] / – Режим доступу: https://paradox.ee/wp-content/uploads/2024/02/YM-280W-S_andmeleht.pdf (дата звернення 10.05.2026).

19. Електромеханічний замок ATIS Lock SS з нержавіючої сталі для контролю доступу [Електронний ресурс] / – Режим доступу: https://viatec.ua/ua/product/ATIS-Lock-SS (дата звернення 10.05.2026).

20. Fail Safe Electric Drop Bolt [Електронний ресурс] / – Режим доступу https://www.bezpeka-shop.com/download/no-wm-manuals/yli-electronic-yb-100.pdf (дата звернення 10.05.2026).

21. MP1584EN Buck converter [Електронний ресурс] / – Режим доступу: https://www.matts-electronics.com/wp-content/uploads/2018/06/MP1584EN.pdf (дата звернення 11.05.2026).

22. LM2596/LM2596HV [Електронний ресурс] / – Режим доступу: https://images.tuyacn.com/smart/A_TUYA/cropper/LM2596S.pdf (дата звернення 11.05.2026).

23. Mini-360 Datasheet [Електронний ресурс] / – Режим доступу: https://www.matts-electronics.com/wp-content/uploads/2018/06/MINI-360.pdf (дата звернення 11.05.2026).

24. Теорія автоматичного управління (Збірник задач) 2020. І. Ш. Невлюдов. О. В. Токарєва 238 с.

25. ESPRESSIF Documentation [Електронний ресурс] / – Режим доступу: https://documentation.espressif.com/en/home (дата звернення 01.06.2026).

26. Інтеграція камери ESP32-CAM OV2640 з мобільними месенджерами [Електронний ресурс] / – Режим доступу: https://ena.lpnu.ua/items/2630ee9c-f214-40b1-9924-151d0bf0674e (дата звернення: 02.06.2026).

27. ESP32-CAM Video Streaming and Face Recognition with Arduino IDE [Електронний ресурс] / – Режим доступу: https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/ (дата звернення: 02.06.2026).

28. Face Recognition with AI Thinker ESP32-CAM and ESP-WHO Machine Learning library [Електронний ресурс] / – Режим доступу: https://github.com/klumw/esp32_cam_face_recognition (дата звернення: 03.06.2026).

29. PN532 ESP-IDF Component [Електронний ресурс] / – Режим доступу: https://github.com/jef-sure/esp32-component-pn532/blob/main/README.md (дата звернення 04.06.2026).

30. MQTT on ESP32: A Beginner's Guide [Електронний ресурс] / – Режим доступу: https://www.emqx.com/en/blog/esp32-connects-to-the-free-public-mqtt-broker (дата звернення: 05.06.2026).

31. ArduinoJSON API Reference [Електронний ресурс] / – Режим доступу: https://arduinojson.org/v7/api/ (дата звернення 06.06.2026).

32. Docker Documentation [Електронний ресурс] / – Режим доступу: https://docs.docker.com/guides/ (дата звернення 07.06.2026).

33. Mosquitto Documentation [Електронний ресурс] / – Режим доступу: https://mosquitto.org/documentation/ (дата звернення 07.06.2026).

34. Flask Documentation [Електронний ресурс] / – Режим доступу: https://flask.palletsprojects.com/en/stable/ (дата звернення 07.06.2026).

35. gmqtt: Python async MQTT client implementation [Електронний ресурс] / – Режим доступу: https://github.com/wialon/gmqtt/blob/master/README.md (дата звернення 07.06.2026).

36. OpenPyxl Documentation [Електронний ресурс] / – Режим доступу: https://openpyxl.readthedocs.io/en/stable/ (дата звернення 07.06.2026).

37. PostgreSQL 16.0 Documentation. The PostgreSQL Global Development Group. 2023 [Електронний ресурс] / – Режим доступу: https://www.postgresql.org/docs/16/index.html (дата звернення 08.06.2026).

38. HK-LEDPCB-12V-X Komponente [Електронний ресурс] / Режим доступу: https://www.ledlightsworld.com/de-de/products/1w-3w-dc12v-dc24v-led-pcb-component-30mm-wide-round-board?variant=17882992476250
(дата звернення 08.06.2026).

39. МВ до лаб. робіт з дисципліни «Основи охорони праці» для студентів усіх напрямів та форм навчання. / Упоряд.: Т.Є. Стиценко, В.А. Айвазов, О.В. Мамонтов. – Харків: ХНУРЕ, 2018.– 120 с.