Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Ком'ютерної інженерії та управління_____
(повна назва)
Кафедра_____Комп'ютерних інтелектуальних технологій та систем _____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____
Інтелектуальна система виявлення дефектів пакування на базі мікроконтролера STM32 з використанням бібліотеки методів комп'ютерного зору OpenCV
(тема)

Виконав:
здобувач ___2025___ року навчання,
групи _____КІУКІ-21-10_____
_____Євген ПАВЛЕНКО_____
(власне ім'я, прізвище)

Спеціальність 123 Комп'ютерна інженерія
_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
Освітня програма _____
_____Комп'ютерна інженерія_____
(повна назва освітньої програми)

Керівник ___доц. каф. КІТС Олег ІЛЮНІН___
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ ___Олег РУДЕНКО__
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Комп'ютерних інтелектуальних технологій та систем

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2025 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Павленку Євгену Максимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система виявлення дефектів пакування на базі мікроконтролера STM32 з використанням бібліотеки методів комп'ютерного зору OpenCV

затверджена наказом по університету від " 21 " травня 2025 р. № 399 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2025

3. Вхідні дані до роботи

1. Документація та програмні засоби для мікроконтролера STM32F746NGH6.
2. Документація та програмні засоби для роботи з камерою та Wi-Fi модулем.
3. Програмні засоби та документація для розробки серверної частини.
4. Програмні засоби та документація для розробки клієнтської частини.
5. Програмні засоби та документація для моделі глибокого навчання.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз предметної області та технологій.
2. Формування вимог до інтелектуальної системи.
3. Розробка апаратно-програмного комплексу.
4. Інструкція користувача.
5. Висновки.

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

6. Консультанти розділів роботи (заповнюється 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	Затвердження дипломного проєкту	26.05.2025	Виконано
2	Аналіз предметної області та існуючих рішень	27.05.2025-28.05.2025	Виконано
3	Вибір технологій та інструментальних засобів	29.05.2025-30.05.2025	Виконано
4	Планування та проектування архітектури	30.05.2025-31.05.2025	Виконано
5	Розробка моделей машинного навчання	31.05.2025-04.06.2025	Виконано
6	Розробка серверної та клієнтських частин	04.06.2025-07.06.2025	Виконано
7	Інтеграція компонентів та тестування системи	08.06.2025-10.06.2025	Виконано
8	Написання пояснювальної записки до проекту	10.06.2025-17.06.2025	Виконано
9	Підготовка презентації	18.06.2025	Виконано
10	Захист кваліфікаційної роботи	25.06.2025	Виконано

Дата видачі завдання 26 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____доц. Ілюнін О.О._____
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 120 с., 16 рис., 9 табл., 1 дод., 29 джерел.

ІІНТЕЛЕКТУАЛЬНА СИСТЕМА, ВИЯВЛЕННЯ ДЕФЕКТІВ, КОНТРОЛЬ ЯКОСТІ ПАКУВАННЯ, МІКРОКОНТРОЛЕР STM32, КОМП'ЮТЕРНИЙ ЗІР, OPENCV, МОДЕЛЬ ГЛИБОКОГО НАВЧАННЯ, EFFICIENTNET-B3, ВЕБ-СЕРВЕР, SPRING FRAMEWORK, WI-FI, ESP8266, РОЗПОДІЛЕНА АРХІТЕКТУРА, C, JAVA, PYTHON.

Метою атестаційної роботи є розробка та дослідження інтелектуальної системи виявлення дефектів пакування, що базується на мікроконтролері STM32F746 для захоплення та первинної обробки зображень, серверному компоненті на Java та бібліотеки OpenCV, а також моделі глибокого навчання для аналізу зображень, та веб-інтерфейс для моніторингу.

У ході виконання атестаціційної роботи було проведено аналіз предметної області контролю якості пакування, розглянуто існуючі системи та технології, що застосовуються для виявлення дефектів, та обґрунтовано актуальність обраного напрямку дослідження. Сформульовано детальні вимоги до розроблюваної системи. Розроблено розподілену архітектуру апаратно-програмного комплексу. Для реалізації мікроконтролерного вузла розроблено програмне забезпечення мовою C з використанням бібліотек HAL та ОСРЧ FreeRTOS, що забезпечує управління камерою, захоплення та передачу JPEG-зображень. Для Сервера Аналізу розроблено додаток на Java, що приймає зображення, виконує їх попередню обробку засобами OpenCV та класифікацію. Для Сервера Даних та Моніторингу створено інтерактивний веб-додаток, що архівує результати в базі даних MySQL та надає їх за запитом.

ABSTRACT

The explanatory note of qualification work 120 pages, 16 figures, 9 tables, 1 appendix, 29 sources.

INTELLIGENT SYSTEM, DEFECT DETECTION, PACKAGING QUALITY CONTROL, STM32 MICROCONTROLLER, COMPUTER VISION, OPENCV, DEEP LEARNING MODEL, EFFICIENTNET-B3, WEB SERVER, SPRING FRAMEWORK, WI-FI, ESP8266, DISTRIBUTED ARCHITECTURE, C, JAVA, PYTHON.

The purpose of the certification work is to develop an intelligent system for detecting packaging defects. This system is based on an STM32F746 microcontroller for image capture and primary processing, a Java server component, OpenCV libraries and a deep learning model for image analysis, as well as a web interface for monitoring.

During the certification work, an analysis of the packaging quality control subject area was carried out and existing systems and technologies used for defect detection were considered. The relevance of the chosen research direction was also justified. Detailed requirements for the system under development were formulated. A distributed architecture for the hardware and software complex was developed.

To implement the microcontroller node, software was developed in C using the HAL and FreeRTOS libraries to provide camera control and the capture and transmission of JPEG images. For the Analysis Server, a Java application was developed to receive images and perform pre-processing using OpenCV tools and classification. A Java application was developed for the Data and Monitoring Server to receive the analysis results, store them in a MySQL database and provide data for an interactive web dashboard.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – База даних

ОСРЧ – Операційна система реального часу

API (Application Programming Interface) – Прикладний програмний інтерфейс

ARM (Advanced RISC Machine) – Архітектура процесорів

AT (Attention) – Набір команд модулів зв'язку

CMOS (Complementary Metal-Oxide-Semiconductor) – Технологія виготовлення сенсорів зображення

CNN (Convolutional Neural Network) – Згорткова Нейронна Мережа

DCMI (Digital Camera Interface) – Цифровий інтерфейс камери

DMA (Direct Memory Access) – Прямий доступ до пам'яті

DMA2D (DMA for 2D data transfer) – DMA для 2D передачі даних, графічний прискорювач

GPIO (General Purpose Input/Output) – Порт вводу-виводу загального призначення

I2C (Inter-Integrated Circuit) – Послідовна шина даних

LTDC (LCD-TFT Display Controller) – Контролер LCD-TFT дисплея

SDRAM (Synchronous Dynamic Random Access Memory) – Синхронна динамічна оперативна пам'ять

UART (Universal Asynchronous Receiver/Transmitter) – Універсальний асинхронний приймач/передавач

ВСТУП

В умовах сучасної глобалізованої економіки та інтенсивної конкуренції на ринках товарів народного споживання, забезпечення високої якості продукції стає не просто конкурентною перевагою, а критично важливою умовою сталого розвитку та успішності виробничих підприємств. Невід'ємною складовою цього процесу є контроль якості пакування, яке виконує не лише захисну та логістичну функції, але й є важливим елементом маркетингової стратегії та формування іміджу бренду. Наявність дефектів пакування, таких як пошкодження цілісності, невірне або нечітке маркування, естетичні недоліки, може призвести до значних економічних збитків через повернення товарів, псування продукції, а також до серйозних репутаційних втрат. У зв'язку з цим, актуальність теми даної кваліфікаційної роботи зумовлена нагальною потребою сучасних виробництв у ефективних, надійних та економічно обґрунтованих методах контролю якості пакувальної продукції. Традиційні підходи, що базуються на ручному візуальному огляді, характеризуються високою суб'єктивністю, низькою продуктивністю та значними трудовими витратами, що робить їх малоефективними для високошвидкісних виробничих ліній.

Автоматизація процесів контролю якості, зокрема з використанням технологій комп'ютерного зору, є ключовим напрямком вирішення зазначених проблем. Системи комп'ютерного зору дозволяють здійснювати об'єктивний, швидкий та безперервний моніторинг продукції, ідентифікуючи дефекти з високою точністю та надійністю. Водночас, сучасні мікроконтролерні системи, такі як платформи на базі архітектури ARM Cortex-M, зокрема STM32, надають потужну та гнучку апаратну основу для створення компактних, енергоефективних та відносно недорогих вбудованих систем збору та первинної обробки візуальної інформації. Як показав аналіз існуючих комерційних рішень, попри їх високу функціональність, значна вартість та

обмежена гнучкість часто стають перешкодою для їх широкого впровадження, особливо на малих та середніх підприємствах. Наукові дослідження, в свою чергу, активно розвивають нові алгоритмічні підходи, проте не завжди пропонують завершені, інтегровані апаратно-програмні комплекси, готові до практичного застосування. Таким чином, розробка доступних, адаптивних та ефективних автоматизованих систем контролю якості пакування, що поєднують переваги сучасних мікроконтролерних технологій, методів комп'ютерного зору та веб-технологій для моніторингу, залишається актуальним науково-практичним завданням.

Метою даної кваліфікаційної роботи є розробка та дослідження автоматизованої системи виявлення дефектів пакування, що базується на мікроконтролері для захоплення та первинної обробки зображень, бібліотеці комп'ютерного зору для аналізу зображень на серверному компоненті, та веб-інтерфейсі для моніторингу та візуалізації результатів.

Для досягнення поставленої мети в роботі вирішувалися наступні завдання:

– провести комплексний аналіз предметної області контролю якості пакування, розглянути існуючі автоматизовані системи та технології, що застосовуються для виявлення дефектів, а також обґрунтувати актуальність обраного напрямку дослідження;

– сформулювати постановку задачі: формалізувати детальні функціональні та нефункціональні вимоги до розроблюваної автоматизованої системи виявлення дефектів пакування;

– розробити архітектуру апаратно-програмного комплексу, включаючи загальну структурну схему системи та схему взаємодії її основних компонентів, а також обґрунтувати вибір апаратних та програмних засобів;

– реалізувати програмне забезпечення для мікроконтролерного вузла для захоплення, відображення та передачі зображень, а також програмне забезпечення серверної частини для аналізу зображень, збереження даних та надання веб-інтерфейсу.

Об’єктом дослідження є процес автоматизованого контролю якості пакувальної продукції на конвеєрній лінії (або її лабораторній імітації).

Предметом дослідження є апаратно-програмний комплекс для виявлення дефектів пакування, що інтегрує мікроконтролерну платформу, методи комп’ютерного зору, технології бездротової передачі даних та веб-технології для реалізації розподіленої системи моніторингу та аналізу.

Для вирішення поставлених завдань у роботі використано наступні методи дослідження: системний аналіз для декомпозиції проблеми та визначення структури системи; методи комп’ютерного зору, включаючи алгоритми попередньої обробки зображень, сегментації та аналізу (реалізовані за допомогою OpenCV), а також принципи роботи моделей глибокого навчання (EfficientNet-B3); методи програмування вбудованих систем для розробки програмного забезпечення мікроконтролера; методи веб-програмування та розробки серверних додатків для створення серверної частини та користувацького інтерфейсу; експериментальні методи для перевірки працездатності та оцінки характеристик розробленої системи.

Значення отриманих результатів полягає у створенні функціонального прототипу автоматизованої системи контролю якості пакування, який може бути використаний практично – як основа для подальшого вдосконалення та впровадження на виробничих підприємствах з метою підвищення якості продукції, зниження рівня браку та зменшення операційних витрат, пов’язаних з ручним контролем. Розроблені програмні модулі та архітектурні рішення також можуть знайти застосування в навчальному процесі при підготовці фахівців з комп’ютерної інженерії та автоматизації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЙ

1.1 Аналіз процесів контролю якості пакування на виробництві

У сучасних умовах глобалізації та підвищеної конкуренції, забезпечення високої якості пакувальної продукції стає не просто бажаною, а критично необхідною умовою успішного функціонування будь-якого виробничого підприємства. Контроль якості пакування безпосередньо впливає на збереження властивостей продукту, його безпеку для споживача, логістичну ефективність та на формування іміджу бренду [12; 22]. Сучасні дослідження підкреслюють, що процеси контролю якості пакування мають на меті своєчасне виявлення та усунення дефектів, які можуть виникати на різних етапах виробничого циклу – від контролю вхідних пакувальних матеріалів до інспекції фінального запакованого продукту. Наслідки неадекватного контролю можуть варіюватися від економічних збитків через повернення товарів та псування продукції до серйозних репутаційних втрат та правових наслідків, особливо у таких галузях, як фармацевтика та харчова промисловість [3; 12].

Різноманітність пакувальних матеріалів та готової продукції зумовлює широкий спектр можливих дефектів. Науковці класифікують їх за кількома основними напрямками. По-перше, це механічні пошкодження, що порушують цілісність упаковки, такі як розриви, проколи, негерметичність зварних швів, тріщини або деформації контейнерів. Особлива увага приділяється контролю герметичності, наприклад, у харчовій промисловості для термоформованих лотків [6] або для промислових мішків, де застосовуються підходи на основі глибокого навчання для виявлення дефектів запаювання [22]. По-друге, значну групу становлять дефекти маркування, включаючи неправильне розташування етикеток, нечіткий або неповний друк, відсутність ключової інформації (терміни придатності, склад,

попереджувальні знаки) та помилки у штрих-кодах або QR-кодах, що є критичним для автоматизованої ідентифікації та логістики [1]. По-третє, естетичні дефекти, такі як подряпини, плями, нерівномірність кольору або зміщення елементів дизайну, хоча й не завжди впливають на функціональні властивості продукту, можуть суттєво знизити його привабливість для споживача. Важливість точної класифікації таких дефектів підкреслюється, наприклад, у дослідженнях контролю якості сигаретних пачок за допомогою згорткових нейронних мереж (CNN) [11].

Традиційні підходи до контролю якості пакування, що базуються на ручному візуальному огляді операторами, все частіше демонструють свою неспроможність відповідати вимогам сучасних високошвидкісних виробничих ліній. Основними недоліками ручного контролю є суб’єктивність оцінки, залежність від людського фактора (втома, неуважність), низька пропускна здатність та високі операційні витрати [22; 24]. Хоча в деяких випадках людська інспекція все ще застосовується, особливо для складних або нетипових дефектів, загальний тренд спрямований на автоматизацію цих процесів [4; 6].

У відповідь на ці виклики, в останні роки спостерігається стрімкий розвиток та впровадження автоматизованих систем контролю якості пакування, які переважно базуються на технологіях машинного (комп’ютерного) зору. Такі системи використовують промислові камери, спеціалізоване освітлення та потужні алгоритми обробки зображень для об’єктивного, швидкого та послідовного виявлення дефектів [20; 25]. Сучасні дослідження активно фокусуються на застосуванні методів глибокого навчання (Deep Learning), зокрема згорткових нейронних мереж (CNN), які демонструють високу точність та ефективність у класифікації дефектів пакування для різноманітних продуктів [12; 16; 27]. Інтеграція таких систем у виробничі лінії дозволяє не тільки ідентифікувати дефектні одиниці продукції в режимі реального часу, але й збирати дані для аналізу причин виникнення браку, сприяючи оптимізації технологічних процесів [5; 24]. При цьому,

бібліотеки з відкритим кодом, такі як OpenCV, надають широкий набір інструментів для розробки та впровадження алгоритмів комп’ютерного зору в промислових застосунках, включаючи контроль якості [12; 25]. Розвиток апаратної бази, зокрема продуктивних мікроконтролерів, таких як STM32, відкриває можливості для створення компактних та енергоефективних систем контролю безпосередньо на виробничих ділянках [8; 13; 17].

Отже, аналіз сучасних наукових публікацій підтверджує актуальність розробки та вдосконалення автоматизованих систем виявлення дефектів пакування, спрямованих на підвищення якості продукції та ефективності виробництва.

1.2 Аналіз проектів у галузі систем виявлення дефектів

Наукові дослідження в галузі автоматизованого виявлення дефектів пакування спрямовані на розробку нових методів обробки зображень та впровадження передових алгоритмів машинного навчання. Аналіз сучасних публікацій свідчить про активне використання як традиційних методів комп’ютерного зору, так і більш новітніх підходів. Наприклад, Лауцка та ін. [4] описують застосування алгоритмів обробки зображень для інспекції ПЕТ-преформ. Метью та ін. проводять огляд технік комп’ютерного зору для виявлення дефектів у скляних пляшках [21].

Значна увага в наукових колах приділяється методам машинного навчання. Так, Ву та Лу [28] повідомляють про використання методу опорних векторів (SVM) для детекції дефектів пакувальних коробок з точністю 98.03%. Особливо активно досліджуються можливості згорткових нейронних мереж (CNN). Наприклад, Са та ін. [19] використовують ResNet-архітектуру CNN з попередньою обробкою зображень за допомогою OpenCV для детекції дефектів у загальному пакуванні. Ву та ін. [28] демонструють застосування алгоритму YOLO для виявлення дефектів пакування в режимі реального часу. Дослідження також фокусуються на гібридних системах. Третола та ін.

комбінують комп’ютерний зір зі стереомікроскопією для виявлення залишків пакування [12].

Апаратна реалізація наукових проектів варіюється: одні дослідження детально описують використання смарт-камер, ПЗЗ-камер високої роздільної здатності, інфрачервоних сенсорів та спеціалізованого обладнання [17; 21; 28], тоді як інші можуть не надавати детальної інформації про апаратну частину. Використання бібліотеки OpenCV для попередньої обробки зображень або як основи для реалізації алгоритмів комп’ютерного зору є поширеною практикою в багатьох наукових роботах. Описані дослідження демонструють широкий діапазон методологій та апаратних реалізацій. Актуальними напрямками досліджень залишаються підвищення швидкості обробки на вбудованих платформах, мініатюризація систем та покращення їх адаптивності.

Однак, багато наукових проектів залишаються на рівні лабораторних прототипів або фокусуються на окремих аспектах системи (наприклад, лише на алгоритмічній частині), не завжди пропонуючи завершене, інтегроване апаратно-програмне рішення, готове до практичного впровадження або легкої адаптації. Крім того, хоча є дослідження, що використовують мікроконтролери, специфічна інтеграція мікроконтролерного вузла для захоплення та локального відображення зображення з подальшою передачею через Wi-Fi на сервер з OpenCV та Spring Framework для аналізу, є конфігурацією, що потребує детального дослідження та практичної реалізації. Саме така розподілена архітектура, де мікроконтролер виконує функції збору та первинної візуалізації даних, а сервер – ресурсомісткий аналіз, може запропонувати оптимальне поєднання вартості, гнучкості та продуктивності.

Отже, розробка автоматизованої системи виявлення дефектів пакування з використанням бібліотеки методів комп’ютерного зору та веб-сервера є актуальним завданням, спрямованим на створення функціонального, відносно недорогого та гнучкого рішення, що відповідає сучасним тенденціям розвитку промислової автоматизації та комп’ютерного зору.

1.3 Аналіз методів комп’ютерного зору для виявлення дефектів

Комп’ютерний зір (КЗ) є науковою та технологічною галуззю, що займається розробкою теорії та методів для створення систем, здатних «бачити» та інтерпретувати візуальну інформацію з навколишнього світу, аналогічно до людського зору. В контексті промислових застосувань, зокрема для виявлення дефектів пакування, методи КЗ надають потужні інструменти для автоматизації процесів контролю якості, забезпечуючи високу швидкість, об’єктивність та точність [13; 20]. Ефективність систем КЗ значною мірою залежить від коректного вибору та реалізації послідовності етапів обробки зображень, а також від застосування відповідних алгоритмів.

Вибір інструментарію для реалізації функцій комп’ютерного зору є фундаментальним етапом проектування автоматизованої системи виявлення дефектів пакування. Обрана бібліотека або програмний комплекс безпосередньо визначає не тільки функціональні можливості системи, але й її продуктивність, гнучкість в адаптації до специфічних завдань, швидкість розробки та потенціал для подальшого вдосконалення.

Для забезпечення обґрунтованого вибору проведемо порівняльний аналіз доступних на ринку рішень, що включають як потужні комерційні продукти, так і гнучкі інструменти з відкритим вихідним кодом.

Комерційні бібліотеки та програмні комплекси, такі як HALCON від MVTec, VisionPro від Cognex, або Matrox Imaging Library (MIL), позиціонуються як індустріально орієнтовані рішення високого класу. Вони зазвичай пропонують надзвичайно широкий спектр високо оптимізованих алгоритмів, призначених для вирішення складних завдань промислового машинного зору, включаючи прецизійні вимірювання, високошвидкісне читання кодів, 3D-інспекцію та багатоаспектний контроль якості в реальному часі. Практичними перевагами таких систем є їхня висока надійність, гарантована виробником, професійна технічна підтримка та, що важливо для промислових застосувань, часто глибока апаратна інтеграція з промисловим

обладнанням та наявність готових, відпрацьованих рішень для типових індустріальних задач. Однак, суттєвим аспектом є модель їх розповсюдження, що зазвичай передбачає ліцензійні відрахування, а також певна «закритість» платформ, яка може обмежувати можливості глибокої кастомізації або інтеграції з унікальними, нестандартними компонентами системи без залучення розробника.

У сегменті бібліотек комп’ютерного зору з відкритим вихідним кодом існує декілька поширених альтернатив. Scikit-image, бібліотека для мови Python, надає широкий набір алгоритмів для обробки зображень і чудово інтегрується з науковою екосистемою Python, зокрема з NumPy та SciPy, що робить її ефективним інструментом для наукових досліджень та швидкого прототипування алгоритмічних ідей. SimpleCV, також розроблена для Python, має на меті спрощення завдань комп’ютерного зору, роблячи їх більш доступними для користувачів з меншим досвідом у цій галузі. Спеціалізовані фреймворки, такі як TensorFlow Lite, PyTorch Mobile або ONNX Runtime, сфокусовані на ефективному розгортанні попередньо навчених моделей машинного навчання, зокрема глибоких нейронних мереж, на пристроях з обмеженими обчислювальними ресурсами. Ці інструменти відіграють ключову роль у реалізації концепції «edge AI», але їх основне призначення – виконання моделей, а не надання повного набору класичних алгоритмів обробки та аналізу зображень.

Однією з найбільш функціонально насичених та широко використовуваних бібліотек з відкритим вихідним кодом є OpenCV (Open Source Computer Vision Library). Розроблена з акцентом на застосунки реального часу, вона надає надзвичайно великий набір функцій, що охоплюють практично всі аспекти сучасного комп’ютерного зору: від фундаментальних операцій обробки зображень (фільтрація, морфологічні перетворення, робота з кольоровими просторами) до складних алгоритмів (детектування та розпізнавання об’єктів, їх відстеження, калібрування камер, методи тривимірної реконструкції, аналіз руху) та інтегрованих модулів для

реалізації алгоритмів машинного навчання і запуску моделей глибокого навчання через модуль DNN [16].

Для більш детального порівняльного аналізу основних альтернатив, їх ключові характеристики, що впливають на практичне застосування, представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз бібліотек комп'ютерного зору

Критерій / Бібліотека	OpenCV	HALCON	Scikit-image	TensorFlow Lite
Основний фокус застосування	Широкий спектр КЗ	Промислова автоматизація	Наукові дослідження	Створення ML моделей
Функціональна повнота	Дуже висока	Дуже висока	Середня	Низька
Підтримка машинного/ глибокого навчання	Висока	Висока	Середня	Дуже висока
Продуктивність (оптимізація)	Висока	Дуже висока	Середня	Дуже висока
Кросплатформеність	Висока	Висока	Висока	Висока
Підтримка мов програмування	C++, Python, інші	Власні скриптові мови,	Python	C++, Python, інші
Спільнота та документація	Дуже велика	Обмежена	Велика	Велика
Гнучкість та відкритість	Висока	Низька	Висока	Висока
Рівень промислового застосування	Високий	Дуже високий	Низький	Середній

Отже, проведений аналіз свідчить, що для реалізації завдань дослідження, яке передбачає розробку функціональної, адаптивної та науково обґрунтованої системи виявлення дефектів пакування з реалізацією алгоритмів комп'ютерного зору на серверній частині, бібліотека OpenCV є найбільш технічно доцільним та функціонально обґрунтованим вибором.

Це рішення підкріплюється такими практичними перевагами. Виняткова функціональна повнота OpenCV забезпечує розробника всіма необхідними інструментами для реалізації повного циклу обробки зображень – від попередньої обробки та сегментації до виділення ознак та класифікації, а також інтеграції з моделями машинного навчання. Це дозволяє гнучко підходити до вирішення задачі виявлення різноманітних типів дефектів без необхідності комбінування декількох спеціалізованих бібліотек. Висока продуктивність та оптимізація C++ ядра OpenCV, а також наявність офіційних та добре підтримуваних Java-біндінгів, є критично важливими для ефективної реалізації серверної частини системи на базі Spring Framework. Це дозволяє обробляти зображення з мінімальними затримками, що важливо для системи, яка працює в режимі, близькому до реального часу, та забезпечує плавну інтеграцію без значних накладних витрат на взаємодію між компонентами. Кросплатформеність та широка підтримка мов програмування надають гнучкість у виборі середовища розробки та розгортання. Відкритість вихідного коду OpenCV надає унікальну можливість для глибокого вивчення реалізації алгоритмів, їх модифікації та адаптації під специфічні потреби проекту, що є надзвичайно цінним для дійсно кастомізованого рішення. У той час як комерційні продукти можуть пропонувати високу надійність «з коробки», вони часто поступаються у гнучкості та можливості адаптації. Інші відкриті бібліотеки або не мають такої широкої функціональності в контексті класичного комп'ютерного зору (як Scikit-image), або сфокусовані на більш вузьких завданнях, таких як виконання нейронних мереж (як TensorFlow Lite), і не можуть повністю закрити всі потреби проекту без залучення додаткових інструментів.

Отже, комплексний аналіз технічних характеристик, функціональних можливостей, продуктивності, доступності ресурсів підтримки та потенціалу для адаптації дозволяє зробити висновок, що бібліотека OpenCV є оптимальним вибором для реалізації завдань комп’ютерного зору в рамках даного проекту, забезпечуючи необхідний баланс між потужністю, гнучкістю та доступністю.

1.4 Аналіз апаратної бази для систем комп’ютерного зору

Вибір відповідної апаратної бази є критичним фактором для успішної реалізації систем комп’ютерного зору, особливо тих, що передбачають роботу на мікроконтролерах та передачу даних для подальшої обробки.

1.4.1 Обґрунтування вибору мікроконтролерної платформи

Вибір мікроконтролерної платформи є фундаментальним рішенням при проектуванні вбудованої частини автоматизованої системи контролю якості, оскільки саме мікроконтролер виконує функції управління периферійними пристроями, такими як камера та дисплей, здійснює первинний збір та підготовку даних, а також забезпечує взаємодію з іншими компонентами системи, зокрема з модулем бездротового зв’язку. Для реалізації завдань дослідження, що передбачають захоплення зображень, їх відображення на локальному дисплеї та подальшу передачу на серверний компонент, необхідно провести ретельний аналіз та порівняння доступних на ринку мікроконтролерів та відлагоджувальних платформ.

Серед потенційних кандидатів для реалізації таких завдань можна розглядати мікроконтролери різних виробників та архітектур. До поширених альтернатив належать сімейства мікроконтролерів від STMicroelectronics, наприклад, популярні, але менш продуктивні серії STM32F4 або, навпаки, більш потужні та спеціалізовані серії STM32H7. Також на ринку представлені

мікроконтролери від інших провідних виробників, таких як NXP з їх серіями LPC (на базі ARM Cortex-M), Microchip з родинами SAM (ARM Cortex-M) та PIC32 (MIPS), Texas Instruments з лінійками Tiva та MSP432 (ARM Cortex-M). Крім того, для певних задач можуть розглядатися одноплатні комп'ютери (ОПК) початкового рівня або потужні мікроконтролери з розширеними можливостями, наприклад, Raspberry Pi Pico (на базі RP2040) або навіть більш продуктивні рішення, такі як Raspberry Pi Zero, які функціонують під управлінням операційних систем типу Linux.

Кожна з цих альтернатив має свої специфічні переваги та недоліки в контексті поставлених завдань. Мікроконтролери серії STM32F4 від STMicroelectronics, хоча й є досить продуктивними та мають розвинену периферію, включаючи DCMI на деяких моделях, можуть зіткнутися з обмеженнями обчислювальної потужності при одночасній інтенсивній роботі з камерою, графічним дисплеєм високої роздільної здатності та швидкою передачею даних. Серія STM32H7 пропонує значно вищу продуктивність, що може бути надлишковим для даного проекту, де основний аналіз зображень винесений на сервер, і може призвести до невиправданого збільшення вартості та складності розробки.

Мікроконтролери від NXP, Microchip, Texas Instruments також пропонують конкурентоспроможні рішення на базі ядер ARM Cortex-M. Однак, часто екосистема STM32, включаючи програмні інструменти (STM32CubeIDE, STM32CubeMX), набір бібліотек HAL (Hardware Abstraction Layer) та LL (Low-Layer), а також обсяг доступної документації та підтримки з боку спільноти розробників, є більш розвиненою та зручною для швидкого старту та вирішення проблем. Наявність готових графічних бібліотек, таких як TouchGFX від ST, також є важливою перевагою при роботі з дисплеями.

Для більш структурованого порівняння ключових аспектів різних типів платформ, їх узагальнені характеристики, переваги та недоліки наведені в таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз альтернативних мікроконтролерних платформ

Платформа / Характеристика	STM32F7	STM32F4	Інші Cortex-M (NXP, Microchip, TI)	Raspberry Pi Pico (RP2040)	Raspberry Pi Zero (Linux SBC)
Продуктивність ядра	Висока (Cortex-M7, >200МГц, FPU, DSP)	Середня (Cortex-M4, ~180МГц, FPU, DSP)	Різна (Cortex-M0 до M7), часто < STM32F7	Середня (Dual Cortex-M0+, ~133МГц)	Висока (ARM11/Cortex-A, ~1ГГц)
Екосистема та інструменти	Дуже розвинена	Дуже розвинена	Залежить від виробника, часто менш об’ємна	У розвитку: Raspberry Pi Foundation_	Дуже розвинена (Linux спільнота)
Складність роботи з периферією реального часу	Низька (bare-metal/ RTOS)	Низька	Низька	Середня	Висока (через ОС)
Складність розробки ПЗ	Середня	Середня	Середня	Низька	Висока (Linux, драйвери) для ембед. задач

Використання Raspberry Pi Pico (RP2040) є привабливим з точки зору вартості та підтримки Python (через MicroPython). Проте, RP2040 має обмежену периферію порівняно з потужнішими мікроконтролерами STM32, і пряме управління паралельною камерою високої роздільної здатності та TFT-дисплеєм з апаратним прискоренням графіки може бути складним завданням. Raspberry Pi Zero або аналогічні ОПК, що працюють під Linux, надають значно більші обчислювальні можливості та легкість роботи з мережевими протоколами та мовами високого рівня, такими як Python. Однак, вони представляють інший клас пристроїв. Розробка програмного забезпечення реального часу та низькорівневе управління периферією на таких системах є значно складнішим завданням порівняно з програмуванням «bare-metal» або використанням ОСРЧ (наприклад, FreeRTOS) на мікроконтролерах.

Аналіз представлених альтернатив та їх характеристик дозволяє зробити висновок, що для реалізації завдань дослідження відлагоджувальна плата STM32F746G-DISCO з мікроконтролером STM32F746NGH6 є найбільш збалансованим та технічно обґрунтованим вибором.

Це рішення зумовлене кількома ключовими факторами. Висока обчислювальна потужність ядра ARM Cortex-M7 (до 216 МГц) з наявністю FPU та DSP-інструкцій забезпечує ефективне управління складними периферійними пристроями, такими як камера та TFT-дисплей, а також швидку підготовку даних для передачі, що є критичним для забезпечення роботи системи в режимі, близькому до реального часу. Хоча для завдань, де основний аналіз зображень винесений на сервер, надвисока продуктивність, як у STM32H7, може бути надлишковою, потужності STM32F7 достатньо для забезпечення плавного локального відображення та ефективної комунікації.

Вирішальним фактором є високий ступінь інтеграції плати STM32F746G-DISCO та наявність на ній ключових для проекту апаратних компонентів. Інтегрований 4.3-дюймовий кольоровий TFT LCD дисплей з ємнісним сенсорним екраном значно спрощує розробку, усуваючи необхідність підбору, підключення та програмування окремого дисплейного

модуля. Вбудований в мікроконтролер STM32F746NGH6 контролер LTDC (LCD-TFT Display Controller) та графічний прискорювач Chrom-ART Accelerator™ (DMA2D) забезпечують ефективну апаратну підтримку роботи з дисплеєм, звільняючи центральний процесор від значної частини графічних операцій. Це є суттєвою перевагою порівняно з платформами, де для роботи з дисплеєм довелося б використовувати менш продуктивний SPI-інтерфейс або реалізовувати складну логіку управління паралельним дисплеєм програмно.

Наявність на платі роз'єму для підключення камери, що безпосередньо взаємодіє з інтерфейсом DCMI (Digital Camera Interface) мікроконтролера, також значно полегшує інтеграцію камерного модуля. Ефективне захоплення відеопотоку через DCMI відбувається з мінімальним завантаженням ЦП, що дозволяє паралельно виконувати інші завдання. Багато альтернативних мікроконтролерних платформ не мають настільки зручного та апаратно підтримуваного інтерфейсу для паралельних камер високої роздільної здатності.

Додатково, мікроконтролер STM32F746NGH6 володіє багатим набором іншої необхідної периферії, такої як інтерфейси SPI та I2C (SCCB сумісний) для конфігурування камери OV5640, інтерфейси UART або SPI для підключення та обміну даними з Wi-Fi модулем ESP8266 (доступні через роз'єми на платі). Наявність Ethernet MAC, USB OTG FS/HS, слота для карт microSD та великої кількості виводів GPIO розширює потенційні можливості системи для подальшого розвитку або налагодження, хоча й не всі ці модулі будуть задіяні в базовій конфігурації даного проекту.

Нарешті, надзвичайно важливим аспектом є розвинена екосистема підтримки для мікроконтролерів STM32. Це включає потужні інтегровані середовища розробки, такі як STM32CubeIDE, інструмент графічного конфігурування STM32CubeMX, велику кількість програмних бібліотек HAL та LL, а також активну спільноту розробників, що надає доступ до численних прикладів коду, навчальних матеріалів та форумів для вирішення технічних проблем. Така інфраструктура значно прискорює процес розробки.

Отже, враховуючи поєднання високої продуктивності, інтегрованих апаратних компонентів (дисплей, інтерфейс камери, графічний прискорювач), багатої периферії та потужної екосистеми підтримки, відлагоджувальна плата STM32F746G-DISCO з мікроконтролером STM32F746NGH6 є оптимальним вибором для реалізації мікроконтролерного вузла автоматизованої системи виявлення дефектів пакування в рамках дослідження. Вона надає потужну, інтегровану та добре документовану платформу для ефективного захоплення, локального відображення зображень та їх подальшої передачі на сервер. Варто зазначити, що використання універсальної відлагоджувальної плати є доцільним на етапі дослідження та прототипування. Для подальшого розвитку системи та її потенційного впровадження як завершеного комерційного або спеціалізованого промислового пристрою, доцільно розглянути розробку спеціалізованої друкованої плати (PCB). Така плата буде містити мікроконтролер STM32F746NGH6 (або інший сумісний з урахуванням оптимізації вартості та доступності) та лише ту периферію, яка безпосередньо необхідна для функціонування системи (інтерфейси камери, дисплея, Wi-Fi модуля, живлення). Це дозволить оптимізувати розміри пристрою, його вартість, енергоспоживання та підвищити надійність за рахунок усунення зайвих компонентів та з’єднань, властивих відлагоджувальним платам.

1.4.2 Обґрунтування вибору модуля камери

Вибір сенсора зображення та відповідного камерного модуля є критично важливим етапом у проектуванні будь-якої системи комп’ютерного зору, оскільки якість, роздільна здатність та характеристики вхідного візуального потоку безпосередньо визначають можливості системи щодо надійного виявлення та класифікації дефектів. Для забезпечення ефективного функціонування розроблюваної системи контролю якості пакування необхідно провести ретельний аналіз доступних на ринку камерних модулів, їх технічних параметрів, інтерфейсів підключення та сумісності з обраною

мікроконтролерною платформою.

Ринок пропонує широкий спектр камерних модулів, призначених для використання у вбудованих системах. До основних альтернатив, що розглядалися, належать різні моделі КМОН-сенсорів (CMOS image sensors) від провідних виробників. Компанія OmniVision пропонує цілу лінійку популярних сенсорів, включаючи моделі з нижчою роздільною здатністю, такі як OV7670 (0.3 Мп) або OV2640 (2 Мп), які часто використовуються в бюджетних проектах або там, де не потрібна висока деталізація. Існують також більш нові та досконалі сенсори від цього ж виробника. Інші виробники, такі як Aptina (тепер частина ON Semiconductor) або Sony, також пропонують високоякісні КМОН-сенсори, що використовуються в промислових та споживчих застосунках. Окремою категорією є веб-камери з USB-інтерфейсом, які є легкодоступними та відносно недорогими. Однак, їх пряма інтеграція з мікроконтролерними платформами, такими як STM32, які не мають повноцінного USB Host контролера з відповідними драйверами для класу UVC (USB Video Class), є значно складнішим інженерним завданням і часто невиправданим для систем, де потрібен низькорівневий контроль над процесом захоплення зображення.

Сенсори з нижчою роздільною здатністю, такі як OV7670 або OV2640, можуть бути привабливими з точки зору меншого обсягу даних, що генеруються, та, відповідно, менших вимог до пропускної здатності інтерфейсів та обсягу буферної пам’яті. Однак, для задачі виявлення дрібних дефектів пакування, їх роздільна здатність може виявитися недостатньою, що призведе до пропуску невідповідностей або неможливості їх точної класифікації. Відсутність вбудованих апаратних функцій, таких як стиснення JPEG, у багатьох бюджетних сенсорах також означає необхідність передачі нестиснутих даних (наприклад, у форматах YUV або RGB), що значно збільшує навантаження на канал зв’язку та час передачі, або вимагає реалізації програмного стиснення на мікроконтролері, що є ресурсомісткою операцією для ядер класу Cortex-M.

Камерні модулі на базі сенсора OV5640 від OmniVision є популярним вибором для багатьох вбудованих систем завдяки збалансованому поєднанню характеристик. Це 5-мегапіксельний КМОН-сенсор, що забезпечує можливість отримання зображень з високою нативною роздільною здатністю (до 2592x1944 пікселів). Така роздільна здатність надає значну гнучкість: для завдань, де потрібна максимальна деталізація, можна використовувати повну роздільну здатність, а для зменшення обсягу даних та прискорення обробки – програмно або апаратно зменшувати її до стандартних значень, таких як HD (1280x720) або VGA (640x480), без значної втрати інформативності порівняно з сенсорами, що мають нижчу нативну роздільну здатність.

Для більш структурованого порівняння основних розглянутих типів модулів камери, їх ключові характеристики наведені в таблиці 1.3.

Таблиця 1.3 – Порівняльний аналіз альтернативних модулів камери

Тип модуля / Характеристика	OV5640 (5 Мп)	OV2640 (2Мп) / OV7670 (0.3 Мп)	Сенсори Aptina/Sony (різні)	USB Веб-камери
Макс. Роздільна здатність	Висока (5 Мп)	Середня (2 Мп) / Низька (0.3 Мп)	Різна	Різна
Інтерфейс даних	DVP (паралельний)	DVP (паралельний)	DVP, MIPI тощо	USB
Апаратне стиснення JPEG	Так	Зазвичай ні	Залежить від моделі	Так
Доступність та документація	Висока	Висока	Середня-Висока	Дуже висока
Гнучкість налаштувань сенсора	Висока (через SCCB/I2C)	Висока (через SCCB/I2C)	Висока	Обмежена (UVC команди)

Проведений аналіз технічних характеристик та особливостей інтеграції дозволяє зробити висновок, що для реалізації завдань даної бакалаврської роботи, яка передбачає захоплення якісних зображень пакування з можливістю їх ефективної передачі на сервер, камерний модуль на базі сенсора OV5640 від OmniVision є найбільш оптимальним та технічно обґрунтованим вибором.

Цей вибір підкріплюється кількома ключовими перевагами. Достатньо висока нативна роздільна здатність (5 мегапікселів) надає необхідну гнучкість у виборі робочої роздільної здатності, забезпечуючи можливість отримання деталізованих зображень, що важливо для виявлення дрібних або слабоконтрастних дефектів. На відміну від сенсорів з нижчою роздільною здатністю, таких як OV7670 або OV2640, які могли б обмежити можливості системи, OV5640 дозволяє адаптуватися до різних вимог щодо деталізації. Особливо цінною є наявність вбудованого апаратного JPEG-кодера. Можливість отримувати стиснуті зображення безпосередньо з камери суттєво зменшує обсяг даних, що передаються через Wi-Fi модуль. Це, в свою чергу, прискорює процес передачі, зменшує навантаження на бездротову мережу та знижує вимоги до пропускної здатності інтерфейсу між STM32 та Wi-Fi модулем. Альтернативні сенсори, що не мають такої функції, вимагали б або реалізації програмного JPEG-стиснення на мікроконтролері STM32F746NGH6, що є досить ресурсомісткою операцією і могла б негативно вплинути на загальну продуктивність системи та швидкість оновлення локального дисплея, або передачі значно більших обсягів нестиснутих даних.

Апаратна сумісність інтерфейсів OV5640 з мікроконтролером STM32F746NGH6 є ще одним важливим фактором. Паралельний цифровий відеопорт (DVP) камери ідеально підходить для прямого підключення до інтерфейсу DCMI (Digital Camera Interface) мікроконтролера, що забезпечує ефективне та швидке захоплення відеопотоку з мінімальним залученням центрального процесора. Керування параметрами камери здійснюється через стандартний інтерфейс SCCB (Serial Camera Control Bus), який є сумісним з

I2C-інтерфейсом, наявним на STM32. Це значно спрощує апаратну та програмну інтеграцію порівняно з USB-камерами, які потребували б реалізації складного USB Host стека та драйверів UVC на мікроконтролері.

Нарешті, камерні модулі на базі OV5640 є широко розповсюдженими, відносно доступними та мають велику кількість документації та прикладів інтеграції з різними мікроконтролерними платформами, включаючи STM32. Багато відлагоджувальних плат, зокрема STM32F746G-DISCO (через відповідні роз'єми або дочірні плати), передбачають можливість їх підключення, що спрощує процес розробки та налагодження на етапі прототипування.

Отже, враховуючи оптимальне поєднання роздільної здатності, підтримки апаратного JPEG-стиснення, апаратної сумісності з обраною мікроконтролерною платформою STM32F746G-DISCO, а також доступності та підтримки спільноти, камерний модуль OV5640 є найбільш відповідним рішенням для забезпечення якісного захоплення зображень пакування в рамках даного проекту.

1.4.3 Обґрунтування вибору модуля бездротового зв'язку

Забезпечення надійного та ефективного каналу бездротової передачі даних від мікроконтролерного вузла до серверної інфраструктури є ключовою вимогою для реалізації функціональності розподіленої системи виявлення дефектів пакування. Вибір технології та конкретного модуля бездротового зв'язку повинен базуватися на аналізі таких факторів, як необхідна пропускна здатність, дальність дії, енергоспоживання, складність інтеграції з обраною мікроконтролерною платформою, а також доступність та вартість.

Для реалізації бездротової комунікації в рамках даного проекту розглядалося декілька альтернативних технологій та відповідних апаратних модулів. До основних варіантів належать модулі, що працюють за технологією Wi-Fi (IEEE 802.11), такі як популярні чіпсети ESP8266/ESP8285 та їх більш

сучасний наступник ESP32 від Espressif Systems, а також Wi-Fi модулі від інших виробників (наприклад, Microchip, Texas Instruments, Realtek). Іншою поширеною технологією є Bluetooth, що включає як класичний Bluetooth (наприклад, модулі HC-05/06), так і Bluetooth Low Energy (BLE) (наприклад, модулі на базі чіпів Nordic nRF5x, Texas Instruments CC26xx). Для специфічних завдань, що вимагають передачі даних на значні відстані за відсутності локальної інфраструктури, можуть розглядатися модулі для стільникового зв'язку (GPRS, 3G, 4G LTE, NB-IoT), однак вони зазвичай мають вищу вартість, складнішу інтеграцію та тарифні плани.

Модулі Bluetooth, особливо BLE, характеризуються низьким енергоспоживанням, що робить їх привабливими для автономних пристроїв з батарейним живленням. Однак, пропускна здатність класичного Bluetooth та особливо BLE є суттєво нижчою порівняно з Wi-Fi, що може стати значним обмеженням при передачі зображень, навіть стиснутих. Дальність дії Bluetooth також обмежена, зазвичай декількома десятками метрів, що може бути недостатньо для зв'язку з сервером через стандартну точку доступу Wi-Fi. Таким чином, для завдань, що передбачають передачу візуальних даних на сервер через локальну мережу, Bluetooth є менш пріоритетним варіантом.

Модулі стільникового зв'язку забезпечують найширше покриття та незалежність від локальних мереж Wi-Fi. Однак, їх вартість, як самих модулів, так і послуг мобільних операторів, значно вища. Крім того, інтеграція таких модулів є більш складною, а енергоспоживання – вищим, що робить їх надлишковими та економічно невигідними для проекту, де передбачається робота в зоні покриття Wi-Fi.

У контексті Wi-Fi модулів, чіпсети ESP8266/ESP8285 (наприклад, у вигляді модулів ESP-01, ESP-07, ESP-12, ESP-15 на базі ESP8285) здобули надзвичайну популярність завдяки своїй доступності та функціональності. Вони містять мікроконтролер Tensilica L106 та повний стек протоколів TCP/IP, що дозволяє їм самостійно обробляти всі аспекти Wi-Fi з'єднання та мережевих протоколів, значно розвантажуючи основний мікроконтролер

системи (в даному випадку, STM32F746NGH6). Інтеграція з хост-мікроконтролером найчастіше здійснюється через простий послідовний інтерфейс UART з використанням набору AT-команд, що є відносно легким для реалізації. Модулі ESP32 є більш сучасними та потужними, пропонуючи двоядерний процесор, вищу продуктивність, підтримку як Wi-Fi, так і Bluetooth (включаючи BLE), а також більшу кількість периферійних інтерфейсів та вищу швидкість SPI. Однак, для задачі простої передачі зображень з STM32 на сервер через Wi-Fi, додаткові можливості ESP32 можуть бути надлишковими, а його вартість зазвичай дещо вища. Інші Wi-Fi модулі від провідних виробників напівпровідників можуть пропонувати вищу надійність або специфічні функції, але часто поступаються рішенням від Espressif за співвідношенням ціна/можливості та активністю спільноти розробників.

На основі проведеного аналізу технічних характеристик, функціональних можливостей та економічних аспектів, для реалізації бездротової передачі даних у системі виявлення дефектів пакування було обрано модуль на базі чіпсету ESP8266/ESP8285 (ESP-15).

Такий вибір обґрунтовується наступними практичними перевагами. Надзвичайно низька вартість модуля при наявності достатнього функціоналу для реалізації Wi-Fi клієнта є важливим фактором для проекту, не знижуючи при цьому надійності базового функціоналу передачі даних. Наявність вбудованого мікроконтролера (Tensilica L106) та повного стека протоколів TCP/IP дозволяє модулю ESP8266/ESP8285 самостійно обробляти всі аспекти встановлення Wi-Fi з’єднання, аутентифікації та передачі даних за протоколами TCP або UDP. Це суттєво розвантажує основний мікроконтролер STM32F746NGH6, який може зосередитися на виконанні своїх основних завдань – захопленні зображень, управлінні дисплеєм та взаємодії з іншою периферією.

Для більш детального порівняння ключових характеристик розглянутих типів модулів бездротового зв’язку, їх узагальнені дані наведені в таблиці 1.4.

Таблиця 1.4 – Порівняльний аналіз альтернативних модулів бездротового зв’язку

Характеристика / Тип модуля	ESP8266/ESP8285	ESP32	Bluetooth (Класичний /BLE)	Стільниковий зв’язок
Основна технологія	Wi-Fi 802.11 b/g/n	Wi-Fi 802.11 b/g/n	Bluetooth Classic / BLE	GSM/GPRS/EDGE/UMTS/LTE/NB-IoT
Пропускна здатність	Середня (достатня для JPEG)	Висока	Низька (BLE)	Різна (залежить від технології)
Дальність дії (типова)	Десятки-сотні метрів (з AP)	Десятки-сотні метрів (з AP)	Метри-десятки метрів	Кілометри (покриття оператора)
Складність інтеграції	Низька-Середня	Середня	Низька-Середня	Висока
Енергоспоживання (активне)	Середнє	Середнє-Високе	Дуже низьке	Високе
Вартість модуля	Дуже низька	Низька-Середня	Низька-Середня	Висока

Хоча модуль ESP32 пропонує вищу продуктивність та додатковий функціонал (наприклад, Bluetooth), для основної задачі даного проекту – передачі зображень через Wi-Fi на сервер – можливостей ESP8266/ESP8285 є цілком достатньо. Додаткова потужність та функції ESP32 в даному контексті можуть бути надлишковими, що не виправдовує його дещо вищу вартість та потенційно складнішу інтеграцію (якщо використовуються всі його можливості).

Таким чином, зважаючи на оптимальне поєднання вартості, функціональності, простоти інтеграції, наявності вбудованого стека TCP/IP та потужної підтримки спільноти, модуль на базі чіпсету ESP8266/ESP8285 є найбільш доцільним та обґрунтованим вибором для реалізації функції бездротової передачі даних в автоматизованій системі виявлення дефектів пакування в рамках даної бакалаврської роботи.

1.5 Огляд програмних технологій

Вибір відповідних програмних технологій є не менш важливим, ніж вибір апаратної бази, для успішної реалізації автоматизованої системи виявлення дефектів пакування. Програмне забезпечення охоплює розробку для вбудованої системи на базі мікроконтролера STM32F7, розробку серверної частини для аналізу зображень та, можливо, допоміжні інструменти для прототипування та тестування.

1.5.1 Мови програмування

Для програмування мікроконтролера STM32F7 основною мовою обрано C. Це традиційний та ефективний вибір для вбудованих систем, що забезпечує низькорівневий доступ до апаратних ресурсів, високу продуктивність та контроль над використанням пам’яті. Враховуючи задачі захоплення зображення, управління периферією DCMI, LTDC, DMA2D, взаємодії з

дисплеєм та модулем ESP8266, а також реалізації логіки роботи в режимі реального часу під керуванням FreeRTOS, мова C дозволяє досягти необхідної швидкодії та компактності коду. Хоча C++ також широко використовується для STM32, для даного проекту, де основна обчислювально складна частина винесена на сервер, C є достатнім та оптимальним вибором.

Для розробки веб-сервера, що буде отримувати зображення від мікроконтролерного вузла, аналізувати їх за допомогою алгоритмів комп'ютерного зору (OpenCV) та надавати інтерфейс для моніторингу, обрано мову Java у поєднанні з фреймворком Spring. Java є потужною, об'єктно-орієнтованою мовою програмування з великою екосистемою бібліотек та інструментів, що добре підходить для створення надійних та масштабованих серверних додатків. Використання Spring Framework спрощує розробку веб-сервісів, управління залежностями, безпеку та взаємодію з базами даних, якщо це буде потрібно. Бібліотека OpenCV має офіційні прив'язки (bindings) для Java, що дозволяє інтегрувати алгоритми обробки зображень безпосередньо в серверний додаток.

Мова Python розглядається для допоміжних задач, таких як швидке прототипування алгоритмів комп'ютерного зору з використанням OpenCV перед їх можливою імплементацією на сервері (Java) або адаптацією для мікроконтролера (C, якщо б така задача стояла). Python з його простим синтаксисом та великою кількістю наукових бібліотек (NumPy, SciPy, Matplotlib, OpenCV) є ідеальним інструментом для експериментів, візуалізації даних та тестування окремих модулів. Також Python може використовуватися для написання скриптів автоматизації (наприклад, для генерації тестових даних, аналізу логів) або для розробки простих інструментів для взаємодії з системою на етапі розробки та налагодження.

1.5.2 Обґрунтування вибору технологій

Для забезпечення зв'язку між мікроконтролерним вузлом (STM32 + ESP8266) та веб-сервером використовується стек протоколів TCP/IP - стандартний та надійний набір протоколів для передачі даних у комп'ютерних мережах. TCP забезпечує гарантовану доставку даних та контроль цілісності, що є критично важливим при передачі зображень.

Для програмування мікроконтролера STM32F746NGH6 використовуються бібліотеки HAL (Hardware Abstraction Layer) від STMicroelectronics. HAL забезпечує високорівневий, платформонезалежний API для доступу до периферійних модулів мікроконтролера. Використання HAL спрощує процес розробки, зменшує ймовірність помилок, пов'язаних з низькорівневим налаштуванням регістрів, та полегшує портування коду. Хоча існують альтернативи, для задач даного проекту функціоналу HAL достатньо.

Для організації багатозадачності та управління ресурсами використовується FreeRTOS - популярна, безкоштовна та компактна ОСРЧ для вбудованих систем. FreeRTOS дозволяє структурувати програмне забезпечення, виділивши окремі завдання для захоплення зображень з камери, оновлення TFT-дисплея, взаємодії з сенсорним екраном та обміну даними з ESP8266. Використання ОСРЧ підвищує надійність та відгукливість системи, дозволяє ефективно керувати пріоритетами завдань через механізми синхронізації (семафори, м'ютекси, черги).

Для розробки веб-сервера обрано Spring Framework - потужний фреймворк для створення корпоративних Java-додатків. Використовуються Spring Boot для швидкого налаштування та розгортання, Spring MVC для створення RESTful веб-сервісів, а також Spring Data для взаємодії з базою даних. Spring забезпечує модульність, тестовність та підтримку сучасних практик розробки. Інтеграція OpenCV з Java дозволить реалізувати логіку аналізу дефектів безпосередньо на сервері, використовуючи його обчислювальні ресурси.

2 ФОРМУВАННЯ ВИМОГ ДО ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

На основі проведеного в попередньому розділі всебічного аналізу предметної області, існуючих комерційних та наукових рішень у сфері автоматизованого контролю якості, а також доступних апаратних та програмних технологій, даний розділ присвячений формуванню та формалізації детальних вимог до розроблюваної автоматизованої системи виявлення дефектів пакування. Формулювання чітких та вичерпних вимог є критично важливим етапом життєвого циклу будь-якої інформаційної системи, оскільки саме вони визначають очікувану функціональність, експлуатаційні характеристики та критерії успішності проекту, слугуючи основою для подальшого проектування, розробки, тестування та валідації системи [2].

2.1 Загальні вимоги до системи

Загальні вимоги окреслюють фундаментальне призначення створюваної системи, визначають стратегічні цілі її розробки, включаючи аспекти подальшого розвитку та адаптації, а також характеризують передбачуване операційне середовище, в якому система повинна ефективно функціонувати.

2.1.1 Призначення, та потенціал масштабованості системи

Розроблювана автоматизована система призначена для здійснення інтелектуальної інспекції візуальних дефектів пакувальних одиниць, що переміщуються на ділянці конвеєрної лінії або подаються на стаціонарну позицію контролю. Система повинна функціонувати в режимі, що забезпечує своєчасне виявлення невідповідностей, максимально наближеному до реального часу виробничого процесу. Основними функціональними

компонентами системи є модуль автоматизованого захоплення цифрових зображень об’єктів контролю, підсистема бездротової передачі отриманих візуальних даних на віддалений серверний компонент для поглибленого аналізу з використанням алгоритмів комп’ютерного зору, а також інтерфейсний модуль, що забезпечує відображення поточного відеопотоку з камери на локальному дисплеї мікроконтролерного вузла та надання авторизованому персоналу доступу до результатів інспекції та агрегованих статистичних даних через спеціалізований веб-інтерфейс.

Стратегічними цілями створення системи є не лише вирішення конкретної задачі виявлення дефектів, але й створення основи для подальшого розвитку та адаптації. По-перше, фундаментальною ціллю є суттєве підвищення ефективності та об’єктивності процедур контролю якості пакування. Це досягається шляхом автоматизації процесу дефектоскопії, що дозволяє мінімізувати вплив суб’єктивного людського фактора, характерного для ручних методів інспекції, та одночасно підвищити швидкість і достовірність контрольних операцій [12; 24]. По-друге, система спрямована на зниження частки бракованої продукції в загальному обсязі виробництва завдяки оперативному виявленню пакувальних одиниць, що не відповідають встановленим критеріям якості. Це створює передумови для їх своєчасного відбракування або для ініціювання коригувальних дій на попередніх етапах технологічного процесу. По-третє, важливою ціллю є забезпечення прозорого та ефективного моніторингу процесу контролю якості. Реалізація веб-інтерфейсу надає авторизованому персоналу засоби для віддаленого спостереження за роботою системи в реальному часі, доступу до архіву зображень дефектної продукції та аналізу агрегованих статистичних даних, що сприяє покращенню керованості виробничими процесами. По-четверте, в рамках освітньо-дослідницького аспекту, проект має на меті розробку, реалізацію комплексного апаратно-програмного рішення, яке демонструє ефективну інтеграцію сучасних мікроконтролерних платформ (зокрема, STM32F746NGH6) [8; 26], методів комп’ютерного зору на базі бібліотеки

OpenCV, реалізованих на серверній стороні [20], технологій бездротової передачі даних за допомогою модулів ESP8266, та сучасних серверних веб-технологій на основі Spring Framework. Нарешті, п’ятою ціллю є дослідження та демонстрація практичного потенціалу розподілених архітектур у системах комп’ютерного зору, де мікроконтролерні вузли виконують функції збору, первинної обробки (в даному випадку, відображення) та передачі візуальної інформації, тоді як ресурсомісткий аналіз делегується потужнішим серверним обчислювальним ресурсам.

Особлива увага при формулюванні цілей приділяється забезпеченню масштабованості та адаптивності системи. Під масштабованістю розуміється здатність системи ефективно функціонувати при збільшенні обсягу оброблюваних даних (наприклад, при зростанні швидкості конвеєра або збільшенні кількості контрольованих параметрів) або при розширенні функціоналу (наприклад, додавання нових типів дефектів для виявлення, інтеграція з іншими виробничими системами). Архітектура системи, що передбачає розподіл завдань між мікроконтролерним вузлом та сервером, закладає основи для горизонтального масштабування (додавання нових мікроконтролерних вузлів для контролю на декількох лініях) та вертикального масштабування (збільшення потужності сервера). Адаптивність системи полягає у її здатності до легкого переналагодження для роботи з різними типами пакування та виявлення широкого спектру дефектів. Це досягається за рахунок використання гнучких алгоритмів комп’ютерного зору, можливості навчання системи на нових зразках продукції та дефектів (особливо при використанні методів машинного навчання на сервері), а також модульної структури програмного забезпечення, що дозволяє легко модифікувати окремі компоненти без суттєвого впливу на інші частини системи. Система повинна бути спроектована таким чином, щоб її встановлення, налаштування та подальше обслуговування були максимально спрощені та не вимагали надмірної кваліфікації від експлуатуючого персоналу.

2.1.2 Умови експлуатації

Ефективність та надійність функціонування автоматизованої системи виявлення дефектів пакування значною мірою залежать від умов її експлуатації. Система проектується з урахуванням можливості її розгортання як у лабораторних умовах для тестування та досліджень, так і в реальних промислових середовищах на виробничих лініях, хоча в межах поточного дослідження пріоритет надається симуляції промислових умов на лабораторному стенді.

Передбачається, що мікроконтролерний вузол, який безпосередньо здійснює захоплення зображень, буде розміщений у безпосередній близькості до об’єкта контролю, наприклад, стаціонарно змонтований над ділянкою конвеєрної стрічки. В умовах реального промислового виробництва необхідно передбачити захист цього вузла від таких несприятливих факторів, як підвищені вібрації, запиленість, вплив вологи та потенційно агресивних хімічних речовин, якщо такі присутні в робочій зоні. У рамках даного проекту, однак, передбачається експлуатація в контрольованих лабораторних умовах, що мінімізують екстремальні впливи.

Функціонування системи повинно бути узгоджене з динамікою конвеєрної лінії або іншого механізму подачі об’єктів. Швидкість руху пакувальних одиниць є критичним параметром, який визначає вимоги до часу експозиції камери, швидкості захоплення кадрів, а також до алгоритмів синхронізації системи з рухом об’єктів. Можливі сценарії включають як безперервний рух об’єктів, так і їх короткочасну зупинку в зоні візуального контролю.

Забезпечення адекватного та стабільного освітлення зони контролю є фундаментальною передумовою для отримання високоякісних цифрових зображень, придатних для подальшого аналізу методами комп’ютерного зору. Для мінімізації впливу флуктуацій природного освітлення або нерівномірності загального освітлення виробничого приміщення, рекомендується

використання спеціалізованих зовнішніх джерел штучного світла, таких як світлодіодні панелі, кільцеві освітлювачі або структуроване освітлення. Правильно підібрана схема освітлення повинна забезпечувати достатню контрастність зображення, мінімізувати тіні та відблиски від пакувальних матеріалів, а також робити візуально помітними характерні ознаки дефектів.

Температурний режим експлуатації мікроконтролерного вузла повинен відповідати стандартним специфікаціям для промислової електроніки, як правило, в діапазоні від +10°C до +40°C, а відносна вологість повітря не повинна призводити до утворення конденсату на електронних компонентах. Серверний компонент системи експлуатується в умовах, типових для розміщення серверного та комп'ютерного обладнання. Важливою умовою є забезпечення стабільного та якісного електроживлення для всіх апаратних компонентів системи, включаючи мікроконтролерний вузол, камеру, освітлювальні прилади та сервер.

Для реалізації бездротової передачі даних зображень від мікроконтролерного вузла (через модуль ESP8266) на сервер необхідна наявність стійкої бездротової мережі Wi-Fi у зоні функціонування системи, що забезпечує надійне з'єднання та достатню пропускну здатність для передачі відеоданих.

Характеристики об'єктів контролю, тобто пакувальних одиниць, суттєво впливають на вимоги до системи. Розміри, форма, матеріал (картон, пластик, скло, метал), колір, текстура поверхні, наявність глянцевих, матових або прозорих елементів – усі ці фактори визначають оптимальні параметри камери (роздільна здатність, тип об'єктива, налаштування експозиції), вибір схеми освітлення та специфіку алгоритмів обробки зображень, що будуть застосовуватися на сервері. Система повинна мати можливість адаптації до різних типів пакування.

Враховуючи вищезазначені аспекти, система повинна бути розроблена з урахуванням принципів легкості встановлення, налаштування та подальшого технічного обслуговування. Це передбачає модульну конструкцію апаратних

компонентів, інтуїтивно зрозумілий інтерфейс для конфігурування параметрів системи (наприклад, через веб-інтерфейс для налаштування типів дефектів або параметрів обробки на сервері), а також можливість діагностики та оновлення програмного забезпечення без надмірних зусиль з боку обслуговуючого персоналу.

2.2 Вимоги до апаратної частини

Функціональність, продуктивність та надійність розроблюваної автоматизованої системи виявлення дефектів пакування нерозривно пов’язані з адекватним вибором та характеристиками її апаратних компонентів. Вимоги до апаратної частини формулюються на основі аналізу завдань, покладених на кожен з елементів системи, та з урахуванням необхідності забезпечення ефективної взаємодії між ними, як це було окреслено при аналізі апаратної бази в підрозділі 1.4. Даний підрозділ деталізує специфічні вимоги до мікроконтролерного вузла, системи захоплення зображень та модуля бездротової комунікації.

2.2.1 Вимоги до мікроконтролера

Мікроконтролер є центральним елементом вбудованого вузла системи, відповідальним за управління периферійними пристроями, захоплення даних з камери, їх підготовку та передачу, а також за відображення інформації на локальному дисплеї. Враховуючи обґрунтування вибору платформи STM32F746G-DISCO (див. підрозділ 1.4.1), вимоги до мікроконтролера конкретизуються наступним чином.

Обчислювальна потужність мікроконтролера повинна бути достатньою для одночасного виконання декількох ключових завдань: ініціалізація та управління камерою через інтерфейс DCMI та I2C; прийом потоку даних зображення від камери; обробка даних для відображення на TFT-дисплеї з

роздільною здатністю 480x272 пікселів, включаючи можливе масштабування або перетворення форматів за допомогою LTDC та DMA2D; управління взаємодією з сенсорним екраном (якщо ця функціональність буде реалізована); формування пакетів даних зображення та їх передача через інтерфейс UART на модуль Wi-Fi; а також виконання логіки операційної системи реального часу (FreeRTOS) для управління задачами. Хоча основний ресурсомісткий аналіз зображень виконується на сервері, мікроконтролер повинен забезпечити мінімальні затримки при передачі даних та оновленні локального дисплея. Ядро ARM Cortex-M7 з тактовою частотою не менше 200 МГц вважається мінімально необхідним.

Обсяг пам’яті є критичним параметром. Мікроконтролер повинен мати достатній обсяг вбудованої Flash-пам’яті для зберігання програмного коду, включаючи прошивку, драйвери периферії, код ОСРЧ та графічні ресурси для інтерфейсу користувача. Мінімально необхідний обсяг Flash-пам’яті оцінюється в 1 Мбайт. Обсяг вбудованої статичної оперативної пам’яті (SRAM) повинен бути достатнім для розміщення буферів кадрів зображення (як мінімум, одного повного кадру в обраному форматі та роздільній здатності для захоплення та одного буфера для відображення), стека ОСРЧ, глобальних змінних та буферів для передачі даних. Враховуючи роздільну здатність дисплея та можливі формати зображень з камери (наприклад, RGB565 для дисплея, JPEG з камери), мінімальний загальний обсяг SRAM оцінюється в 320 Кбайт.

Набір периферійних модулів повинен включати специфічні інтерфейси, необхідні для реалізації функціоналу системи. Обов’язковою є наявність паралельного інтерфейсу для підключення камери (DCMI). Також необхідний контролер для управління TFT-дисплеєм (LTDC) та бажано наявність графічного прискорювача (DMA2D) для розвантаження ЦП. Для конфігурації камери потрібен інтерфейс. Для зв’язку з модулем Wi-Fi необхідний принаймні один високошвидкісний інтерфейс UART. Наявність таймерів загального призначення для генерації сигналів синхронізації або управління

часовими інтервалами також є важливою. Інші периферійні модулі, такі як ADC, DAC, CAN, можуть бути корисними для потенційного розширення функціоналу, але не є критичними для базової конфігурації. Конкретні мінімальні вимоги до мікроконтролера наведені в таблиці 2.1.

Таблиця 2.1 – Мінімальні вимоги до мікроконтролера

Параметр	Вимога
Архітектура ядра	ARM Cortex-M7 або еквівалент за продуктивністю
Тактова частота	≥ 200 МГц
Flash-пам’ять	≥ 1 Мбайт
SRAM	≥ 320 Кбайт
Інтерфейс камери	DCMI (Digital Camera Interface)
Контролер дисплея	LTDC або аналогічний для прямого управління TFT
Інтерфейс конфігурації камери	I2C
Інтерфейси зв’язку з Wi-Fi	UART
Наявність ОСРЧ підтримки	Так (для FreeRTOS)

2.2.2 Вимоги до камери та оптики

Система захоплення зображень, що складається з камери та відповідної оптики, є первинним джерелом даних для аналізу. Якість отриманих зображень безпосередньо впливає на точність та надійність виявлення дефектів. Вимоги до камери формулюються з урахуванням специфіки об’єктів контролю та умов освітлення, описаних в підрозділі 2.1.2, а також на основі характеристик обраного модуля OV5640 (див. підрозділ 1.4.2).

Роздільна здатність сенсора камери повинна бути достатньою для розрізнення найдрібніших потенційних дефектів на поверхні пакування. Враховуючи, що аналіз виконується на сервері, бажано мати можливість

захоплення зображень з роздільною здатністю не нижче VGA (640x480 пікселів) для забезпечення прийнятної деталізації. Використання сенсора з вищою нативною роздільною здатністю (наприклад, 5 Мп як у OV5640) надає гнучкість у виборі робочої роздільної здатності та можливість цифрового масштабування без значної втрати якості.

Камера повинна підтримувати різні формати вихідних даних зображення. Для відображення на локальному TFT-дисплеї може бути зручним формат RGB565. Для передачі на сервер з метою зменшення обсягу даних перевага надається форматам YUV (з можливістю передачі лише компоненти яскравості для монохромного аналізу) або, що ще краще, наявності вбудованого апаратного стиснення у формат JPEG. Можливість отримання JPEG-потоку безпосередньо з камери значно спрощує задачу передачі даних через Wi-Fi модуль з обмеженою пропускною здатністю.

Чутливість сенсора та динамічний діапазон повинні бути достатніми для роботи в умовах контрольованого, але потенційно не ідеального освітлення. Камера повинна мати можливість автоматичного або програмного налаштування параметрів експозиції (AEC), балансу білого (AWB) та коефіцієнта підсилення (AGC) для адаптації до змін умов освітлення та характеристик об’єктів контролю.

Тип об’єктива (оптики) та його характеристики (фокусна відстань, кут огляду, глибина різкості, світлосила) повинні бути підібрані таким чином, щоб забезпечити чітке зображення всієї контрольованої області пакування з необхідної робочої відстані. Об’єктив повинен мати можливість ручного або автоматичного фокусування. Важливо мінімізувати геометричні спотворення, що вносяться оптикою, або мати можливість їх програмної корекції. Для забезпечення стабільності та повторюваності результатів оптика повинна бути жорстко зафіксована.

Інтерфейс підключення камери до мікроконтролера повинен бути паралельним цифровим (DVP), сумісним з інтерфейсом DCMI мікроконтролера STM32. Керуючий інтерфейс повинен бути стандартним,

наприклад, I2C-сумісний).

Вимоги до камери та оптики підсумовані в таблиці 2.2.

Таблиця 2.2 – Вимоги до камери та оптики

Параметр	Вимога
Тип сенсора	CMOS
Мінімальна робоча роздільна здатність	VGA (640x480 пікселів)
Підтримувані формати виводу	RAW; обов’язкова підтримка JPEG (апаратне стиснення)
Частота кадрів (при VGA)	≥ 15 fps
Функції автоналаштування	AEC, AWB, AGC
Оптика	Фіксована фокусна відстань або з можливістю налаштування; достатня глибина різкості; мінімальні спотворення
Інтерфейс даних	Паралельний (DVP), сумісний з DCMI
Інтерфейс керування	I2C-сумісний

2.2.3 Вимоги до модуля Wi-Fi

Модуль бездротового зв’язку Wi-Fi є ключовим елементом для передачі зображень з мікроконтролерного вузла на сервер для подальшого аналізу. Вимоги до цього модуля визначаються необхідністю забезпечення надійного та достатньо швидкого каналу зв’язку, а також простотою його інтеграції з мікроконтролером STM32, як було зазначено при аналізі модуля ESP8266 (див. підрозділ 1.4.3).

Модуль повинен підтримувати стандарт IEEE 802.11b/g/n для забезпечення сумісності з сучасними Wi-Fi мережами та достатньої

пропускної здатності. Наявність вбудованого стека TCP/IP є обов'язковою, оскільки це дозволяє модулю самостійно обробляти мережеві протоколи, розвантажуючи основний мікроконтролер. Модуль повинен мати можливість функціонувати в режимі Wi-Fi клієнта (Station mode) для підключення до існуючої точки доступу.

Інтерфейс зв'язку з мікроконтролером STM32 повинен бути стандартним та простим в реалізації, перевага надається UART з підтримкою AT-команд. Пропускна здатність цього інтерфейсу повинна бути достатньою для передачі даних зображення (особливо стиснутих JPEG-файлів) без створення «вузького місця».

Модуль повинен мати вбудовану антену для забезпечення стабільного Wi-Fi з'єднання на прийнятній відстані від точки доступу, враховуючи можливі перешкоди в промисловому середовищі (2.1.2). Енергоспоживання модуля повинно бути помірним, хоча для стаціонарної системи це не є настільки критичним, як для автономних пристроїв з батарейним живленням. Важливою є також простота конфігурування параметрів Wi-Fi мережі (SSID, пароль) та налаштувань TCP/IP з'єднання. Основні вимоги до модуля Wi-Fi представлені в таблиці 2.3.

Таблиця 2.3 – Вимоги до модуля Wi-Fi

Параметр	Вимога
Підтримувані стандарти Wi-Fi	IEEE 802.11b/g/n
Вбудований стек TCP/IP	Так
Режими роботи	Wi-Fi клієнт (Station mode)
Інтерфейс з хост-МК	UART (з AT-командами)
Антенне виконання	Вбудована антена
Безпека Wi-Fi	Підтримка WPA/WPA2 PSK

2.3 Вимоги до програмного забезпечення

Програмне забезпечення є невід’ємною та критично важливою складовою розроблюваної автоматизованої системи виявлення дефектів пакування, забезпечуючи логіку функціонування всіх апаратних компонентів та їх взаємодію. Вимоги до програмного забезпечення формулюються окремо для вбудованої підсистеми, що функціонує на мікроконтролері STM32F746NGH6, та для серверної підсистеми, відповідальної за аналіз зображень та надання веб-інтерфейсу. Ці вимоги базуються на загальних цілях системи (підрозділ 2.1.1), функціональних потребах та обраному технологічному стеку (підрозділ 1.5).

2.3.1 Вимоги до вбудованого програмного забезпечення

Вбудоване програмне забезпечення, що виконується на мікроконтролері STM32F746NGH6, повинно забезпечувати ініціалізацію та управління всією периферією мікроконтролера, захоплення зображень з камери OV5640, їх відображення на TFT-дисплеї, підготовку та передачу даних на сервер через Wi-Fi модуль ESP8266, а також, за необхідності, обробку взаємодії з користувачем через сенсорний екран. Враховуючи багатозадачний характер цих операцій та необхідність забезпечення відгукливості системи, ключовою вимогою є використання операційної системи реального часу (ОСРЧ).

ПЗ мікроконтролера повинно бути розроблене з використанням мови програмування C, що забезпечує високу продуктивність та низькорівневий доступ до апаратних ресурсів. Для взаємодії з периферією мікроконтролера STM32 повинні використовуватися бібліотеки HAL (Hardware Abstraction Layer), що надаються STMicroelectronics. Це забезпечує стандартизований та відносно високорівневий доступ до апаратних модулів, таких як DCMI, LTDC, DMA2D, I2C, UART, SPI, спрощуючи процес розробки та підвищуючи портативність коду в межах сімейства STM32. Також повинна бути

забезпечена сумісність та можливість використання стандартних бібліотек CMSIS (Cortex Microcontroller Software Interface Standard), які надають доступ до функцій ядра ARM Cortex-M7 та стандартизовані інтерфейси до периферії.

Функціонування вбудованого ПЗ повинно базуватися на операційній системі реального часу FreeRTOS. Використання FreeRTOS є необхідним для ефективного управління паралельними процесами, такими як одночасне захоплення відеопотоку з камери, оновлення зображення на дисплеї, обмін даними з Wi-Fi модулем та обробка команд користувача. ОСРЧ повинна забезпечувати механізми планування задач з різними пріоритетами, синхронізації (семафори, м’ютекси) та обміну даними між задачами (черги). Програмне забезпечення повинно бути структуроване у вигляді окремих, незалежних задач FreeRTOS для підвищення модульності та надійності.

Драйвери периферійних пристроїв повинні забезпечувати коректну ініціалізацію та функціонування камери OV5640 (через DCMI та SCCB/I2C), TFT-дисплея (через LTDC та, можливо, SPI для сенсорного екрану), графічного прискорювача DMA2D та Wi-Fi модуля ESP8266 (через UART або SPI). Драйвер камери повинен підтримувати конфігурування роздільної здатності, формату вихідних даних (включаючи JPEG) та параметрів експозиції. Драйвер дисплея повинен забезпечувати виведення зображень з буфера кадру та, можливо, елементів графічного інтерфейсу користувача. Драйвер Wi-Fi модуля повинен реалізовувати протокол обміну (наприклад, AT-команди) для встановлення з’єднання та передачі даних.

2.3.2 Вимоги до серверного програмного забезпечення

Серверне програмне забезпечення (ПЗ) відіграє центральну роль в аналізі отриманих зображень, зберіганні даних та наданні користувацького інтерфейсу для моніторингу системи. Воно повинно бути реалізоване як веб-додаток, що забезпечує гнучкість доступу та взаємодії.

Серверне ПЗ повинно бути розроблене з використанням мови

програмування Java та фреймворку Spring (зокрема, Spring Boot, Spring MVC/WebFlux, Spring Data). Вибір Spring Framework зумовлений його потужними можливостями для створення надійних, масштабованих та безпечних веб-додатків, а також широкою підтримкою спільноти та великою кількістю готових модулів (див. підрозділ 1.5.2).

Ключовою функціональною вимогою до серверного ПЗ є прийом зображень від мікроконтролерного вузла через мережевий протокол, HTTP POST запити з тілом, що містить дані зображення. Сервер повинен мати реалізований API для цієї взаємодії.

Серверне ПЗ повинно забезпечувати зберігання даних, як мінімум, інформації про виявлені дефекти (час, тип дефекту, можливо, посилання на зображення). Для цього передбачається використання реляційної системи управління базами даних (СКБД) MySQL. Взаємодія з базою даних повинна здійснюватися через відповідні компоненти Spring Data (наприклад, Spring Data JPA). Структура бази даних повинна бути спроектована для ефективного зберігання та вибірки необхідної інформації.

Сервер повинен надавати веб-інтерфейс користувача для моніторингу роботи системи. Цей інтерфейс повинен відображати статистику виявлених дефектів, можливо, журнал подій, а також надавати доступ до зображень з виявленими дефектами. Веб-інтерфейс має бути реалізований як односторінковий додаток (SPA) з використанням JavaScript-фреймворків, що взаємодіє з REST API сервера.

3 РОЗРОБКА АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

На основі сформульованих у попередньому розділі вимог до автоматизованої системи виявлення дефектів пакування, даний розділ присвячений детальному опису процесу розробки її апаратно-програмного комплексу. Ключовим етапом, що передує безпосередній реалізації, є проектування архітектури системи, яка визначає її основні компоненти, їх функції, взаємозв’язки та принципи взаємодії. Чітко визначена архітектура слугує дорожньою картою для подальшої розробки, забезпечує модульність, полегшує інтеграцію компонентів та сприяє створенню надійної та ефективної системи.

3.1 Розробка архітектури системи

Архітектура розроблюваної автоматизованої системи виявлення дефектів пакування проектувалася з урахуванням вимог до функціональності, продуктивності, умов експлуатації та обраного технологічного стеку. Вона базується на розподіленому підході, де завдання збору та первинної візуалізації даних покладаються на вбудований мікроконтролерний вузол, тоді як ресурсомісткий аналіз зображень та управління системою здійснюються на серверному компоненті. Такий підхід дозволяє поєднати переваги локальної обробки в реальному часі та потужних обчислювальних можливостей сервера.

3.1.1 Загальна структурна схема системи

Загальна структурна схема системи, представлена на рисунку 3.1, наочно ілюструє основні функціональні блоки системи та їх взаємозв’язки на високому рівні.

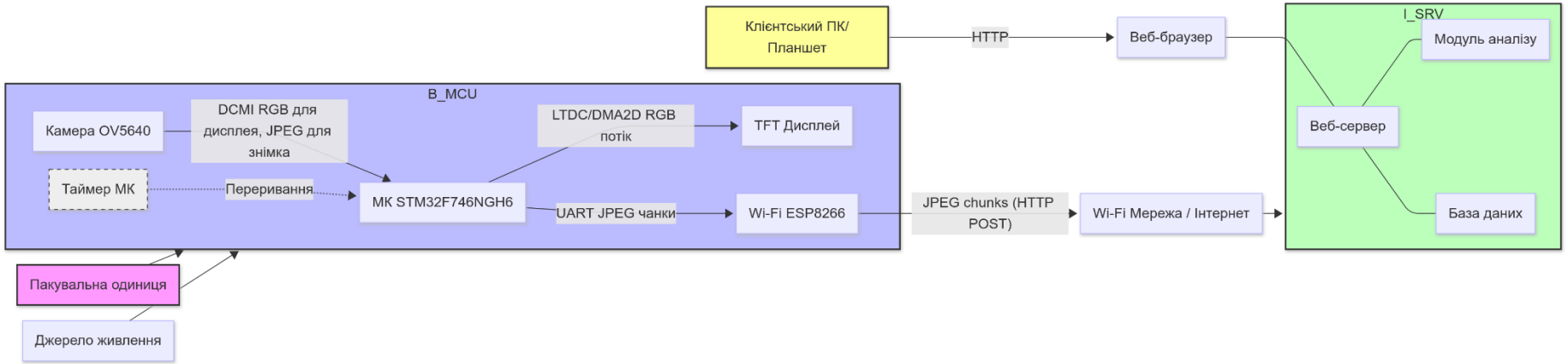

Рисунок 3.1 – Загальна структурна схема системи

Система концептуально розділена на декілька ключових взаємодіючих частин. Вхідним елементом є «Пакувальна одиниця», зображення якої підлягає аналізу. Основним елементом збору даних є Мікроконтролерний вузол STM32F746G. Всередині мікроконтролерного вузла знаходиться камера OV5640, що підключена до центрального МК STM32F746NGH6. Камера передає на мікроконтролер два типи відеопотоків: потік даних у форматі RGB для безперервного відображення на локальному дисплеї (через LTDC/DMA2D потік RGB), та окремі кадри у форматі JPEG для подальшого аналізу. Захоплення JPEG-знімків ініціюється сигналами переривання від таймера МК. Отримані JPEG-зображення передаються частинами через послідовний інтерфейс (UART) на модуль Wi-Fi ESP8266, який, у свою чергу, надсилає їх за допомогою HTTP POST запитів через Wi-Fi Мережу/Інтернет на серверний вузол.

Серверний вузол складається з веб-сервера реалізованого на Spring Application, модуля аналізу, що використовує OpenCV та бази даних MySQL. Ці компоненти взаємодіють між собою для обробки отриманих зображень та зберігання результатів.

Для взаємодії з оператором передбачена клієнтська частина, представлена, на якій через веб-браузер користувач отримує доступ до дашборду, що відображає дані, отримані від веб-сервера по протоколу HTTP.

Така архітектура чітко розмежовує завдання: збір та первинна візуалізація даних на мікроконтролерному рівні, передача даних через бездротовий канал та глибокий аналіз і зберігання на серверному рівні, з наданням користувацького інтерфейсу через веб-технології.

3.1.2 Схема взаємодії компонентів

Для глибшого розуміння динамічних аспектів функціонування системи, послідовності обміну даними та керуючими сигналами між її ключовими програмними та апаратними компонентами, була розроблена діаграма потоків

даних та керуючих сигналів, представлена на рисунку 3.2. Ця діаграма візуалізує основний робочий цикл системи, починаючи від моменту ініціалізації захоплення зображення і закінчуючи обробкою даних на сервері та наданням результатів оператору.

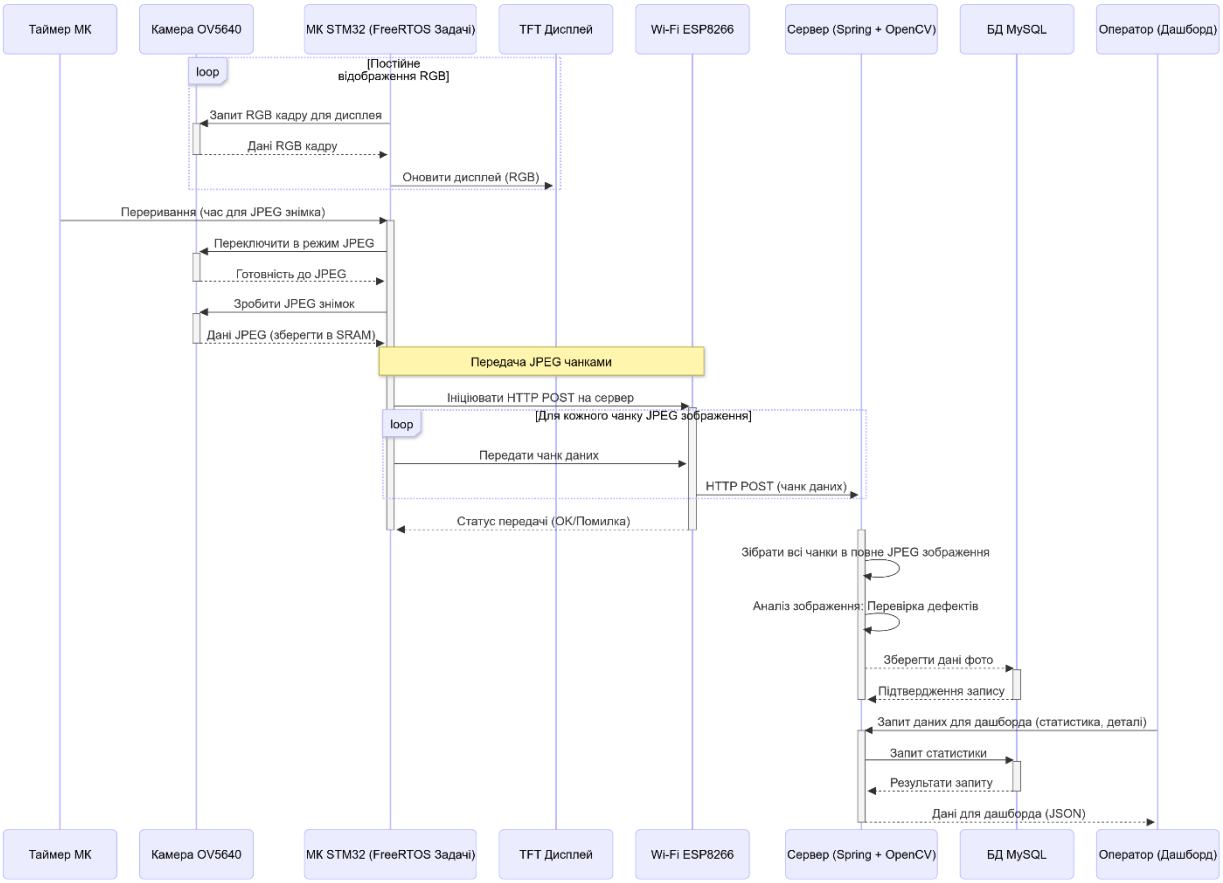

Рисунок 3.2 – Діаграма потоків даних та керуючих сигналів

Як видно зі схеми на рисунку. 3.2, робота системи характеризується паралельним виконанням процесів. МК STM32 (FreeRTOS Задачі) в безперервному режимі забезпечує отримання RGB-відеопотоку від камери OV5640 та його виведення на локальний дисплей, що дозволяє оператору здійснювати постійний візуальний моніторинг зони контролю. Захоплення ж зображення для аналізу ініціюється асинхронно, за сигналом переривання від таймера МК. У відповідь на це переривання, мікроконтролер дає команду

камері на переключення в режим формування JPEG-знімка, здійснює його захоплення та збереження у внутрішній пам’яті.

Наступним етапом є передача отриманого JPEG-зображення на сервер. Цей процес детально ілюструє взаємодію між МК STM32, Wi-Fi ESP8266 модулем та сервером (Spring + OpenCV). Мікроконтролер ініціює HTTP POST запит та передає зображення по частинах на модуль ESP8266, який транслює ці чанки на сервер.

На серверній стороні, після отримання та збору всіх чанків в єдине зображення, запускається процес його аналізу для виявлення дефектів. Результати цього аналізу, разом із супутньою інформацією про знімок, зберігаються в базу даних. Діаграма також відображає взаємодію оператора дашборду із системою: користувач може ініціювати запит до сервера для отримання актуальної статистики або деталізованої інформації, яку сервер формує на основі даних з бази.

Таким чином, представлена на рисунку 3.2 діаграма послідовності розкриває логіку взаємодії компонентів системи в часі, демонструючи ключові етапи обробки інформації – від її первинного збору на мікроконтролерному рівні до комплексного аналізу та представлення результатів на серверному та клієнтському рівнях. Вона підкреслює розподілений характер системи та асинхронність виконання основних операцій, що є важливим для забезпечення її ефективності та відгукливості.

3.2 Розробка апаратної частини

Реалізація апаратної частини автоматизованої системи виявлення дефектів пакування передбачає фізичне з’єднання та конфігурацію ключових компонентів: мікроконтролерної плати STM32F746G-DISCO, камерного модуля OV5640 та Wi-Fi модуля ESP8266. Детальний опис апаратних компонентів та обґрунтування їх вибору було наведено у розділах 1 та 2. Цей підрозділ фокусується на конкретних аспектах їх апаратної інтеграції,

описуючи принципові схеми підключення та конфігурації інтерфейсів.

3.2.1 Принципова схема підключення камери OV5640 до STM32F746G-DISCO (інтерфейс DCMI та I2C)

Підключення камерного модуля OV5640 до мікроконтролерної плати STM32F746G-DISCO здійснюється з використанням спеціалізованого роз'єму для камери, передбаченого на платі, та інтерфейсу DCMI (Digital Camera Interface) мікроконтролера STM32F746NGH6 для передачі даних зображення, а також інтерфейсу I2C (сумісного з SCCB камери) для конфігурування та управління камерою.

На платі STM32F746G-DISCO для підключення камери передбачений роз'єм P1 (Camera module connector) який представлено на рисунку 3.3. Цей роз'єм виводить необхідні сигнали DCMI та I2C основні сигнали які було використано представлено у таблиці 3.1. Сигнали даних DCMI (DCMI_D0-DCMI_D7), сигнали синхронізації (DCMI_HSYNC, DCMI_VSYNC, DCMI_PIXCK), сигнал живлення камери (DCMI_PWR_EN) та тактовий сигнал для камери (Camera_CLK) підключаються до відповідних пінів мікроконтролера STM32F746NGH6. Наприклад, сигнали даних DCMI_D0-DCMI_D7 підключені до пінів PH9-PH12, PH14, PE5, PE6, PI11.

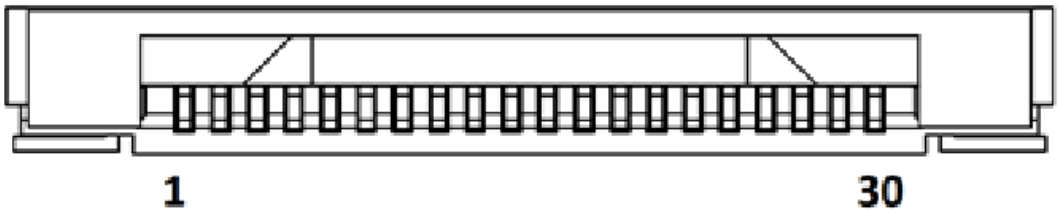

Рисунок 3.3 – Роз'єм P1 (Camera module connector)

Для конфігурування камери OV5640 використовується її інтерфейс SCCB (Serial Camera Control Bus), який є I2C-сумісним. На платі

STM32F746G-DISCO для цього зазвичай використовується один з доступних I2C-інтерфейсів мікроконтролера, виведених на роз'єм P1. На роз'ємі P1 присутні сигнали DCMI_SDA (пін 27) та DCMI_SCL (пін 28). Вони відповідають I2C_SDA та I2C_SCL лініям, підключеним до відповідних пінів мікроконтролера.

Таблиця 3.1 – Основні сигнали підключення камери OV5640 до STM32F746G-DISCO (через роз'єм P1)

Сигнал OV5640 / Роз'єму P1	Призначення	Інтерфейс
PCLK	Піксельний тактовий сигнал	DCMI
HSYNC	Горизонтальна синхронізація	DCMI
VSYNC	Вертикальна синхронізація	DCMI
D0 - D7	8-бітна паралельна шина даних зображення	DCMI
XCLK	Тактовий сигнал для камери	–
SCL	Лінія тактування I2C для керування	I2C/SCCB
SDA	Лінія даних I2C для керування	I2C/SCCB
PWDN	Управління живленням камери	GPIO
RESET	Сигнал скидання камери	GPIO
3V3	Живлення камери	Живлення
GND	Земля	Живлення

3.2.2 Принципова схема підключення модуля Wi-Fi ESP8266 до STM32F746G-DISCO (інтерфейс UART)

Для забезпечення бездротової передачі даних зображень на сервер використовується Wi-Fi модуль на базі чіпсету ESP8266. Як було обґрунтовано в підрозділі 1.4.3, одним з найпростіших та найпоширеніших способів інтеграції ESP8266 з хост-мікроконтролером є використання

інтерфейсу UART для обміну даними та керуючими AT-командами. У даному проекті для зв’язку між STM32F746NGH6 та ESP8266 буде використаний інтерфейс USART6, з пінами PC6 (USART6_TX) та PC7 (USART6_RX) мікроконтролера STM32F746NGH6.

Ці піни доступні на роз’ємах розширення плати STM32F746G-DISCO. Зокрема, сигнали PC6 (USART6_TX) та PC7 (USART6_RX) виведені на Arduino™ конектор CN4 який представлено на рисунку 3.4.

-	PI3	D7	8	CN4 digital
TIM12_CH1	PH6	D6	7	
TIM5_CH4,SPI 2_NSS	PI0	D5	6	
-	PG7	D4	5	
TIM3_CH1	PB4	D3	4	
-	PG6	D2	3	
USART6_TX	PC6	D1	2	
USART6_RX	PC7	D0	1	

Рисунок 3.4 – Роз’єм CN4 з необхідними пінами USART6

Підключення модуля ESP8266, що має виведені UART піни TXD та RXD)до STM32F746NGH6 буде здійснюватися наступним чином:

– пін TXD модуля ESP8266 підключається до піна PC7 (USART6_RX) мікроконтролера STM32F746NGH6;

– пін RXD модуля ESP8266 підключається до піна PC6 (USART6_TX) мікроконтролера STM32F746NGH6;

– також необхідно забезпечити спільну землю (GND) між платою та модулем ESP8266, а також подати живлення 3.3В на модуль ESP8266 яке

можна взяти з плати, враховуючи споживаний струм модуля ESP8266, особливо під час передачі даних.

Піни, що використовуються для підключення наведено у таблиці 3.2.

Таблиця 3.2 – Підключення модуля ESP8266 до STM32F7 (USART6)

Пін ESP8266	Пін/Сигнал МК	Призначення
TXD	PC7 (USART6_RX)	Передача даних від ESP8266 до STM32
RXD	PC6 (USART6_TX)	Прийом даних ESP8266 від STM32
VCC	3.3V (з плати STM32)	Живлення модуля ESP8266
GND	GND (з плати STM32)	Спільна земля
CH_PD (EN)	3.3V (з плати STM32)	Увімкнення модуля (Enable)

Розробка апаратної частини на основі цих схем та специфікацій дозволить забезпечити коректну взаємодію між усіма компонентами мікроконтролерного вузла та його здатність виконувати поставлені завдання щодо захоплення, відображення та передачі зображень.

3.3 Розробка програмного забезпечення для мікроконтролера

Програмне забезпечення для мікроконтролера STM32F746NGH6 є ключовим компонентом вбудованого вузла, що відповідає за координацію роботи всіх апаратних модулів, реалізацію логіки захоплення та обробки зображень, а також за взаємодію з серверною частиною системи. Розробка ПЗ здійснювалася мовою C з використанням бібліотек HAL (Hardware Abstraction Layer) та операційної системи реального часу FreeRTOS для забезпечення багатозадачності та ефективного управління ресурсами. Далі детально розглянуто основні етапи розробки вбудованого ПЗ.

3.3.1 Ініціалізація периферії

Коректна ініціалізація периферійних модулів мікроконтролера STM32F746NGH6 є фундаментальним етапом, що забезпечує їх правильну роботу та взаємодію. На відміну від підходу з використанням інструменту графічного конфігурування STM32CubeMX, у даному проекті ініціалізація периферії реалізована програмно шляхом написання спеціалізованих функцій для кожного модуля.

Ініціалізація GPIO портів. Для кожного задіяного периферійного модуля (DCMI, LTDC, I2C, USART6) насамперед виконується конфігурування відповідних виводів GPIO. Це включає увімкнення тактування відповідних портів GPIO, налаштування режиму роботи пінів, вибір підтягуючих резисторів, швидкості та конкретної альтернативної функції. Наприклад DCMI для портів PA, PD, PE, PG, PH необхідно налаштувати GPIO_MODE_AF_PP, GPIO_PULLUP, GPIO_SPEED_HIGH та GPIO_AF13_DCMI. Аналогічна детальна конфігурація GPIO портів здійснюється для LTDC, задіюючи порти GPIOE, GPIOG, GPIOI, GPIOJ, GPIOK для виведення сигналів керування та даних дисплея. Для I2C1, що використовується для керування камерою, ініціалізація GPIO (PB8, PB9).

Ініціалізація DCMI та DMA для камери. Модуль DCMI (Digital Camera Interface) ініціалізується для прийому даних з камери OV5640. У функції dcmiInit відбувається налаштування параметрів DCMI, таких як режим синхронізації, полярність сигналів, частота захоплення, розширений режим даних та, що важливо для даного проекту, увімкнення режиму JPEG. Для передачі даних з DCMI в пам’ять без участі ЦП налаштовується контролер прямого доступу до пам’яті (DMA). Ініціалізується DMA2_Stream1 (канал 1) для роботи з DCMI. Налаштовується напрямок передачі (периферія-в-пам’ять), інкрементація адреси пам’яті, вирівнювання даних, режим роботи, пріоритет та параметри FIFO. Створений хендлер DMA пов’язується з хендлером DCMI. Також налаштовуються та вмикаються переривання від

DCMI та відповідного потоку DMA для обробки подій завершення кадру або лінії.

Ініціалізація LTDC та DMA2D для дисплея. Модуль LTDC (LCD-TFT Display Controller) відповідає за формування сигналів для керування TFT-дисплеєм. Ініціалізація LTDC відбувається у функції lcdInit(). Тут налаштовуються часові параметри синхронізації (HSYNC, VSYNC, HBP, VBP, ActiveH, ActiveW, TotalH, TotalW), що відповідають характеристикам дисплея RK043FN48H. Також конфігурується тактування для LTDC та параметри шару відображення (розміри вікна, формат пікселів, адреса буфера кадру в SDRAM, параметри альфа-змішування). Вмикається живлення дисплея та підсвічування. Модуль DMA2D використовується для прискорення операцій копіювання даних з буфера камери в буфер кадру дисплея. Його ініціалізація включає налаштування режим роботи (M2M – пам’ять-в-пам’ять), вихідний формат кольору (DMA2D_OUTPUT_RGB565) та параметри шару.

Ініціалізація I2C для камери. Для управління камерою OV5640 (запис та читання її регістрів) використовується інтерфейс I2C1. Ініціалізація I2C1 відбувається у функції initI2C(). Тут налаштовуються параметри таймінгу I2C, режим адресації та інші специфічні налаштування. Важливо, що функція I2Cx_MspInit() яка викликається з initI2C() відповідає за низькорівневу ініціалізацію GPIO та тактування для I2C1, а також за налаштування переривань. У наданому коді також присутній цикл для сканування I2C шини з метою виявлення підключених пристроїв.

Ініціалізація USART6 для Wi-Fi модуля. Інтерфейс USART6 використовується для зв’язку з Wi-Fi модулем ESP8266. Його ініціалізація описана у функції uart6_Init(). Тут встановлюється швидкість передачі, довжина слова, кількість стоп-бітів, відсутність контролю парності, режим роботи (прийом та передача), відсутність апаратного контролю потоку та параметри надлишкової вибірки. Також налаштовується та вмикається глобальне переривання для USART6 (HAL_NVIC_SetPriority(USART6_IRQn, 6, 0); HAL_NVIC_EnableIRQ(USART6_IRQn);), що є важливим для

асинхронної обробки даних, особливо при використанні HAL_UARTEx_ReceiveToIdle_IT у драйвері ESP8266.

Ініціалізація SDRAM. Зовнішня SDRAM пам’ять, наявна на платі STM32F746G, використовується для зберігання буфера кадру дисплея та, можливо, для буферизації зображень з камери. Її ініціалізація відбувається у функції BSP_SDRAM_Init().

3.3.2 Інтеграція та налаштування FreeRTOS: формалізація задач

Для забезпечення ефективного управління паралельними обчислювальними процесами та раціонального розподілу ресурсів мікроконтролера STM32F746NGH6, в архітектурі вбудованого програмного забезпечення застосовано операційну систему реального часу (ОСРЧ) FreeRTOS. Такий підхід дозволяє структурувати програмний код у вигляді набору логічно ізольованих, незалежно виконуваних потоків управління, або задач, кожна з яких відповідає за реалізацію певного функціонального блоку системи.

Програмна архітектура мікроконтролерного вузла включає декілька ключових задач FreeRTOS. Задача, реалізована у функції vTaskCAMERA, несе відповідальність за комплексне управління камерним модулем OV5640. Її функціонал включає ініціалізацію камери, динамічне перемикання між режимами виводу даних (RGB для локального відображення та JPEG для захоплення знімків), безпосереднє ініціювання процесу захоплення JPEG-зображень за запитом від інших задач, а також оновлення глобальних змінних, що містять інформацію про адресу та розмір останнього захопленого JPEG-кадру (shared_jpeg_image_addr, shared_jpeg_image_size). Для цієї задачі встановлено відносно високий пріоритет (tskIDLE_PRIORITY + 3) та виділено стек розміром орієнтовно configMINIMAL_STACK_SIZE * 4, що є достатнім для виконання її функцій. Взаємодія цієї задачі з іншими компонентами системи здійснюється через семафори capture_request_sem (для отримання

запиту на захоплення) та image_ready_sem (для сигналізації про готовність зображення).

Іншою важливою задачею є vTaskWIFI, яка забезпечує управління Wi-Fi модулем ESP8266. Її основні функції – це ініціалізація модуля, встановлення з’єднання з Wi-Fi мережею та точкою доступу, а також передача захоплених JPEG-зображень на серверний компонент. Передача ініціюється за сигналом від таймера, що обробляється через семафор. Ця задача має доступ до глобальних змінних з даними зображення та взаємодіє з низькорівневим драйвером ESP8266. Їй призначено пріоритет, tskIDLE_PRIORITY + 3, та виділено значний обсяг стека, орієнтовно configMINIMAL_STACK_SIZE * 8, враховуючи потенційну потребу в буферизації даних для AT-команд та мережевих операцій.

Задача vTaskLCD відповідає за ініціалізацію TFT-дисплея. Безперервне оновлення RGB-зображення з камери на дисплеї реалізовано більш ефективним апаратним шляхом, а саме через використання модуля DMA2D, який копіює дані безпосередньо з буфера DCMI в буфер кадру SDRAM під управлінням обробників переривань DCMI. Таким чином, задача vTaskLCD виконує лише початкове налаштування, а її пріоритет (tskIDLE_PRIORITY + 2) та розмір стека (configMINIMAL_STACK_SIZE * 2) є відносно низькими.

Також у системі присутня задача vTaskSDRAM, єдиним призначенням якої є ініціалізація зовнішньої SDRAM-пам’яті. Аналогічно до задачі дисплея, після виконання своєї функції вона призупиняється. Її пріоритет та розмір стека також є мінімальними.

Формалізація взаємодії між цими задачами –є критично важливою для злагодженої роботи системи. Основний сценарій захоплення та передачі JPEG-зображення ілюструє цю взаємодію. Задача vTaskWIFI, отримавши сигнал про необхідність передачі, ініціює процес, сигналізуючи задачі vTaskCAMERA про необхідність захоплення зображення. Це відбувається шляхом «віддачі» семафора capture_request_sem. Задача vTaskCAMERA, яка очікує на цей семафор (приклад очікування наведено на рисунку 3.5), після його отримання

виконує процедуру захоплення JPEG-кадру, зберігає інформацію про його адресу та розмір у глобальних змінних shared_jpeg_image_addr та shared_jpeg_image_size. Доступ до цих спільних ресурсів при читанні з vTaskWIFI синхронізується за допомогою критичних секцій (taskENTER_CRITICAL() / taskEXIT_CRITICAL()), як це видно у фрагменті коду на рисунку 3.6. Після успішного захоплення, vTaskCAMERA сигналізує про готовність зображення шляхом «віддачі» семафора image_ready_sem. Задача vTaskWIFI, в свою чергу, очікує на цей семафор і, отримавши його, приступає до процедури передачі даних на сервер.

```
      while (1) {
              // Очікування команди на захоплення зображення
              if (xSemaphoreTake(capture_request_sem, portMAX_DELAY) == pdTRUE)
{
                  // ... (логіка захоплення JPEG) ...
                  if (jpeg_found_successfully && jpeg_actual_file_size > 0) {
                      shared_jpeg_image_addr  =  JPEG_FRAME_BUFFER_ADDRESS  +
jpeg_offset_in_buffer;
                      shared_jpeg_image_size = jpeg_actual_file_size;
                  } else {
                      shared_jpeg_image_addr = 0;
                      shared_jpeg_image_size = 0;
                  }
                  // ... (повернення камери в режим RGB) ...

                  // Повідомити wifiTask, що зображення готове (або сталася
помилка)
                  xSemaphoreGive(image_ready_sem);
              }
          }
```

Рисунок 3.5 – Фрагмент очікування запиту на захоплення та сигналізації

```
      if (xSemaphoreGive(capture_request_sem) != pdPASS) { // Сигнал задачі
камери
              return;
          }
          if (xSemaphoreTake(image_ready_sem, pdMS_TO_TICKS(30000)) == pdPASS){
              taskENTER_CRITICAL();
              image_address = shared_jpeg_image_addr;
              image_size = shared_jpeg_image_size;
              taskEXIT_CRITICAL(); // ... (подальша передача зображення)
          }
```

Рисунок 3.6 – Фрагмент ініціювання захоплення та очікування зображення

Щодо безперервного відображення RGB-потоку на дисплеї, цей процес реалізований асинхронно від основних задач, використовуючи можливості DMA2D. Модуль DMA2D копіює прийняті від DCMI лінії зображення безпосередньо в буфер кадру, розташований в SDRAM. Таким чином, активна участь окремої задачі FreeRTOS у постійному опитуванні та копіюванні даних для дисплея не потрібна, що є ефективним з точки зору використання ресурсів ЦП.

Взаємодія програмного коду з апаратними обробниками переривань та FreeRTOS API також є важливим аспектом. У файлі stm32f7xx_it.c визначені обробники системних переривань. Принципово, що в тих обробниках, які взаємодіють з об’єктами синхронізації FreeRTOS (наприклад, EXTI15_10_IRQHandler, який викликає xSemaphoreGiveFromISR, як показано у лістингу 3.1), використовуються спеціалізовані версії API FreeRTOS, призначені для виклику з контексту переривань (функції з суфіксом FromISR). Також, за необхідності, використовується функція portYIELD_FROM_ISR для негайного запиту перемикання контексту на задачу з вищим пріоритетом, яка могла бути розблокована внаслідок події переривання.

Такий підхід до структурування програмного забезпечення з використанням FreeRTOS забезпечує необхідну багатозадачність, дозволяє ефективно розподіляти обчислювальні ресурси мікроконтролера та реалізувати чутливу систему з мінімальним часом відгуку, здатну одночасно виконувати декілька критичних операцій.

3.3.3 Розробка драйвера для роботи з камерою OV5640

Функціонування системи захоплення зображень неможливе без спеціалізованого програмного драйвера, що забезпечує низькорівневе управління камерним модулем OV5640. Драйвер, реалізований та інкапсулює логіку ініціалізації камери, конфігурування її численних параметрів, перемикання режимів роботи (RGB та JPEG) та управління процесом

захоплення даних через інтерфейси DCMI та SCCB/I2C.

Основним етапом роботи драйвера є ініціалізація камери, яка виконується функцією OV5640_Init(). Ця функція, відповідає за подачу живлення на камеру, перевірку ідентифікатора чіпа камери шляхом читання регістрів для верифікації її працездатності та відповідності очікуваній моделі. У випадку успішної ідентифікації, відбувається запис послідовності початкових конфігураційних значень у внутрішні регістри сенсора. Ці значення зберігаються у статичній таблиці ov5640_init_reg_tbl і забезпечують базове налаштування камери для подальшої роботи. Взаємодія з регістрами камери для їх читання та запису реалізована через функції OV5640_RD_Reg(uint16_t reg) та OV5640_WR_Reg(uint16_t reg, uint8_t data), які, у свою чергу, використовують I2C функції для комунікації з камерою за її I2C-адресою. Драйвер також містить функції для більш тонкого налаштування камери, такі як ініціалізація та управління системою автофокусування, регулювання яскравості, контрастності та налаштування різкості, що дозволяє адаптувати камеру до різних умов зйомки та характеристик об’єкта контролю.

Ключовою функціональною можливістю розробленого драйвера є динамічне перемикання між режимами виводу даних камери: формат RGB565, призначений для безперервного відображення на локальному TFT-дисплеї, та формат JPEG, що використовується для захоплення окремих статичних кадрів з метою їх подальшої передачі на сервер для аналізу. Реалізація цього функціоналу покладена на функцію OV5640_changeCameraMode(OV5640_mode_t mode). Залежно від значення вхідного параметра mode (RGB або JPEG), ця функція викликає відповідні процедури для реконфігурації регістрів камери OV5640. Функція OV5640_rgb565Mode() завантажує специфічні налаштування з таблиці ov5640_rgb565_reg_tbl та встановлює необхідний вихідний розмір зображення (визначений макросами XSIZE, YSIZE), оптимізований для локального дисплея. Натомість, функція OV5640_jpegMode(uint8_t jpg_size) завантажує

конфігурацію для режиму JPEG з таблиці OV5640_jpeg_reg_tbl, встановлює бажаний розмір JPEG-зображення (наприклад, CAM_1280x720, визначений у масиві jpeg_size_tbl) та оновлює значення максимального очікуваного розміру буфера для JPEG-даних (jpeg_buf_size).

Процес захоплення даних з камери через інтерфейс DCMI та їх передачі в пам’ять за допомогою DMA ініціюється функцією readDataFromCamera(), приклад якої наведено на рисунку 3.7.

```
      void readDataFromCamera(void)
      {
      uint8_t  *memAddress  =  (uint8_t*)JPEG_FRAME_BUFFER_ADDRESS;  //  Адреса
буфера JPEG в cameraTask.c
      // uint32_t dma_transfer_size; // Розмір передачі DMA

      if(ov5640.mode == JPEG) {
              //  dma_transfer_size  =  jpeg_buf_size;  //  Використовується  в
HAL_DCMI_Start_DMA
            while(HAL_DCMI_GetState(&hdcmi) != HAL_DCMI_STATE_READY);
                HAL_DCMI_Start_DMA(&hdcmi,   DCMI_MODE_CONTINUOUS,   //   Або
DCMI_MODE_SNAPSHOT для JPEG
                                   (uint32_t*)memAddress, jpeg_buf_size );
      } else { // RGB mode
              // dma_transfer_size = XSIZE/2; // Для однієї лінії в словах
(XSIZE - ширина в пікселях)
            while(HAL_DCMI_GetState(&hdcmi) != HAL_DCMI_STATE_READY);
            HAL_DCMI_Start_DMA(&hdcmi,                  DCMI_MODE_CONTINUOUS,
(uint32_t)camLine, XSIZE/2);
      }
            __HAL_DCMI_ENABLE_IT(&hdcmi,DCMI_IT_FRAME);      //      Увімкнення
переривання кінця кадру

      }
```

Рисунок 3.7 – Запуск захоплення даних з камери

Реалізація цієї функції залежить від поточного активного режиму камери, який зберігається у змінній ov5640.mode. У випадку режиму RGB, функція HAL_DCMI_Start_DMA(&hdcmi, DCMI_MODE_CONTINUOUS, (uint32_t)camLine, XSIZE/2) запускає безперервне захоплення даних DCMI. Дані для кожної лінії зображення приймаються в тимчасовий буфер camLine. Потім, в обробнику переривання HAL_DCMI_LineEventCallback() ці дані копіюються за допомогою модуля DMA2D у відповідну позицію основного буфера кадру, розташованого в SDRAM. Цей механізм забезпечує ефективне

формування повного кадру для безперервного відображення на TFT-дисплеї. У випадку режиму JPEG, функція HAL_DCMI_Start_DMA(&hdcmi, DCMI_MODE_CONTINUOUS, (uint32_t*)memAddress, jpeg_buf_size), де memAddress вказує на спеціальний буфер для JPEG-даних в SDRAM, наприклад, JPEG_FRAME_BUFFER_ADDRESS ініціює захоплення JPEG-потоку. Важливо, що DCMI для цього режиму було попередньо налаштовано з увімкненим DCMI_JPEG_ENABLE. Після отримання даних, що відповідають одному JPEG-кадру, очікується, що обробник переривання DCMI зупинить поточну DMA-транзакцію та DCMI.

Розроблений таким чином програмний драйвер для камери OV5640 забезпечує необхідний рівень абстракції для взаємодії з апаратним забезпеченням камери, дозволяє гнучко конфігурувати її параметри та режими роботи, а також ефективно використовувати апаратні можливості мікроконтролера STM32F746NGH6 для захоплення та первинної обробки візуальних даних, що є основою для подальшого функціонування всієї системи виявлення дефектів.

3.4 Програмне управління модулем Wi-Fi ESP8266 з боку STM32

Реалізація бездротового каналу зв’язку в розроблюваній системі покладається на Wi-Fi модуль на базі чіпсету ESP8266. Цей модуль функціонує як мережевий інтерфейс для мікроконтролера STM32F746NGH6, забезпечуючи передачу захоплених зображень на серверний компонент. У даній архітектурі модуль ESP8266 використовується зі стандартною прошивкою, що інтерпретує AT-команди. Відповідно, вся логіка управління модулем, включаючи процедури налаштування Wi-Fi з’єднання, встановлення TCP-сесій та безпосередню передачу даних, інкапсульована в спеціалізованому програмному драйвері, який виконується на мікроконтролері STM32. Цей драйвер, представлений файлами esp8266_driver.c та esp8266_driver.h, дозволяє централізувати управління

мережевою взаємодією, використовуючи обчислювальні ресурси STM32 для формування та послідовної відправки AT-команд.

3.4.1 Ініціалізація та конфігурування Wi-Fi з’єднання

Процес встановлення з’єднання з Wi-Fi мережею ініціалізується та повністю контролюється програмним забезпеченням мікроконтролера STM32 за допомогою функції esp_wifi_init_and_connect(esp_driver_handle_t* esp_handle). Ця функція реалізує послідовний алгоритм для перевірки працездатності модуля ESP8266, його коректної конфігурації та подальшого підключення до цільової точки доступу Wi-Fi.

На початковому етапі відбувається перевірка адекватності зв’язку з модулем шляхом відправки тестової команди AT та очікування стандартної відповіді. У випадку відсутності відповіді, драйвер робить спробу апаратного перезавантаження модуля командою AT+RST, після чого очікує на сигнал готовності. Цей механізм забезпечує початкову синхронізацію та перевірку працездатності модуля. Після успішного встановлення зв’язку, модуль переводиться в режим Wi-Fi станції (клієнта) командою AT+CWMODE=1, а також вимикається exo-режим (локальне повторення команд) командою ATE0 для спрощення обробки відповідей.

Ключовим кроком є безпосереднє підключення до Wi-Fi мережі. Це здійснюється шляхом динамічного формування та відправки команди AT+CWJAP="SSID","PASSWORD", де параметри SSID та PASSWORD відповідають ідентифікатору та паролю цільової мережі. Для підвищення надійності підключення реалізовано механізм повторних спроб: у випадку невдачі, драйвер може спробувати від’єднатися від будь-якої попередньої мережі командою AT+CWQAP і потім повторити спробу підключення. Успішне завершення цієї процедури позначається встановленням внутрішнього прапора esp_handle->is_module_connected_to_wifi.

3.4.2 Передача даних зображення на веб-сервер

Після успішного встановлення Wi-Fi з’єднання, мікроконтролерний вузол готовий до передачі захоплених зображень на серверний компонент для їх подальшого аналізу. У даному проекті для передачі даних використовується протокол HTTP, зокрема метод POST, а самі дані зображення передаються по частинах. Ця логіка інкапсульована у функції esp_http_post_data_chunked(), яка керує процесом розбиття зображення на чанки та послідовної відправки кожного чанку.

Для передачі кожного окремого фрагменту (чанку) даних внутрішня функція _internal_send_http_post_chunk() виконує такі кроки. Спочатку встановлюється TCP-з’єднання з цільовим сервером, потім формуються HTTP POST заголовки, які включають шлях до серверного ендпоінта, інформацію про хост, а також спеціальні заголовки, необхідні для коректної обробки чанкованої передачі на сервері: X-File-ID (унікальний ідентифікатор файлу, що передається), X-Chunk-Index (порядковий номер поточного чанку) та X-Total-Chunks (загальна кількість чанків). Заголовок Content-Type встановлюється як application/octet-stream, а Content-Length вказує на розмір тіла поточного чанку. Приклад формування таких заголовків наведено нижче.

```
      http_headers_length                =                snprintf(esp_handle-
>http_headers_internal_buffer_ptr,
                                    esp_handle-
>http_headers_internal_buffer_actual_size,
                                    "POST %s HTTP/1.0\r\nHost: %s\r\n"
                                    "X-File-ID:           %s\r\nX-Chunk-Index:
%lu\r\nX-Total-Chunks: %lu\r\n"
                                    "Content-Type:          application/octet-
stream\r\nContent-Length: %lu\r\n"
                                    "Connection: close\r\n\r\n",
                                    target_server_path,            HTTP_HOST,
unique_file_identifier,
                                    (unsigned         long)current_chunk_num,
(unsigned long)total_number_of_chunks,
                                    (unsigned
long)current_chunk_payload_size);
```

Рисунок 3.8 – Фрагмент формування HTTP POST заголовків для чанку

Після формування заголовків, відправляється команда AT+CIPSEND=length, де length – це сумарний розмір HTTP заголовків та даних поточного чанку. Отримавши від модуля ESP8266 сигнал готовності (символ >), драйвер передає спочатку самі HTTP заголовки, а потім – безпосередньо байти даних поточного чанку. Після завершення передачі чанку, драйвер очікує відповідь від сервера, яка включає підтвердження відправки від модуля ("SEND OK") та закриття TCP-з'єднання ("CLOSED"). Ця послідовність кроків повторюється для кожного чанку, доки все зображення не буде повністю передано на сервер. Управління прийомом асинхронних відповідей від ESP8266 через UART реалізовано з використанням механізму переривань та функцій зворотного виклику esp_uart_rx_event_callback() та esp_uart_error_callback(), які обробляють отримані дані та можливі помилки на каналі зв'язку.

Такий підхід до управління модулем Wi-Fi, реалізований на стороні STM32, дозволяє гнучко контролювати процес мережевої взаємодії та адаптувати його до специфічних вимог системи передачі зображень.

3.5 Розробка веб-сервера на Spring (мова Java)

Серверна інфраструктура автоматизованої системи виявлення дефектів пакування є ключовим компонентом, що відповідає за обробку візуальних даних, їх інтелектуальний аналіз, персистентне зберігання результатів та надання інтерфейсу для моніторингу та взаємодії з користувачем. Зважаючи на різнорідність завдань – від прийому та обробки потоку зображень до аналізу даних та формування статистики – серверна частина була реалізована у вигляді двокомпонентної архітектури. Обидва компоненти розроблені на платформі Java з використанням фреймворку Spring, що забезпечує високу продуктивність, масштабованість та гнучкість.

Перший серверний компонент (надалі – «Сервер Аналізу») спеціалізується на прийомі зображень від мікроконтролерного вузла, їх

попередній обробці засобами бібліотеки OpenCV та безпосередній класифікації на предмет наявності дефектів за допомогою попередньо навченої моделі глибокого навчання. Другий серверний компонент (надалі – «Сервер Даних та Моніторингу») відповідає за прийом результатів аналізу від Сервера Аналізу, їх збереження в реляційній базі даних MySQL, а також за агрегацію даних та надання користувацького веб-інтерфейсу (дашборда) для моніторингу. Така архітектура дозволяє логічно розділити навантаження та спеціалізувати кожен сервер на виконанні певного кола завдань.

3.5.1 Проектування API

Ефективна взаємодія між мікроконтролерним вузлом, Сервером Аналізу, Сервером Даних та Моніторингу, а також клієнтською частиною (веб-дашбордом) забезпечується за допомогою ретельно спроектованих прикладних програмних інтерфейсів (API), реалізованих на базі RESTful веб-сервісів.

Основним завданням Сервера Аналізу є прийом зображень від мікроконтролерного вузла ESP8266, який, як було зазначено раніше, передає дані по частинах (чанками). Для цього в класі ImageUploadController (файл ImageUploadController.txt) реалізовано ендпоінт /uploadChunk, що обробляє HTTP POST запити з типом контенту application/octet-stream. Кожен такий запит містить частину (чанк) файлу зображення, а також спеціальні HTTP заголовки: X-File-ID (унікальний ідентифікатор файлу), X-Chunk-Index (порядковий номер поточного чанку) та X-Total-Chunks (загальна кількість чанків). Логіка контролера передбачає акумуляцію отриманих чанків у тимчасовому сховищі на сервері (використовуючи клас UploadSession для управління сесіями завантаження) під унікальним X-File-ID. Фрагмент реалізації цього ендпоінта наведено нижче.

```
      @RestController
      public class ImageUploadController {
          // ... (поля та конструктор) ...
          private static class UploadSession { /* ... (внутрішня структура) ...
*/ }
          private final Map<String, UploadSession> activeUploads = new
ConcurrentHashMap<>();

          @PostMapping(value = "/uploadChunk", consumes = "application/octet-
stream")
          public ResponseEntity<?> handleChunkUpload(
                  HttpServletRequest httpRequest,
                  @RequestHeader("X-File-ID") String fileId,
                  @RequestHeader("X-Chunk-Index") int chunkIndex,
                  @RequestHeader("X-Total-Chunks") int totalChunks) {
              // ... (отримання або створення UploadSession для fileId) ...
              try (InputStream inputStream = httpRequest.getInputStream()) {
                  // ... (запис даних чанку в тимчасовий файл сесії) ...
              }
              // ... (перевірка, чи всі чанки отримано) ...
              synchronized (session) {
                  session.receivedChunks++;
                  if (session.receivedChunks == totalChunks) {
                      //  ...  (закриття  тимчасового  файлу,  переміщення  у
фінальне місце) ...
                      Path completedFilePath = /* ... (шлях до зібраного файлу)
... */;
                      String  clientIp  =  getClientIpAddress(httpRequest);  //
Отримання IP для передачі
                      // Виклик сервісу аналізу та відправки звіту
                      analysisResult                                          =
imageAnalysisService.analyzeImageFromPathAndReport(completedFilePath,
clientIp);
                      // ... (обробка результату, видалення сесії) ...
                      return
ResponseEntity.status(HttpStatus.CREATED).body(analysisResult);
                  }
              }
              return  ResponseEntity.accepted().build();  //  Повідомлення  про
успішний прийом чанку
          }
          // ... (допоміжні методи) ...
      }
```

Рисунок 3.9 – Фрагмент ендпоінта для прийому чанків на Сервері Аналізу

Після отримання всіх чанків та успішної збірки файлу, ImageUploadController викликає метод analyzeImageFromPathAndReport сервісу ImageAnalysisService. Цей сервіс, як детально описано в підрозділі 3.6.1, виконує попередню обробку зображення засобами OpenCV, його класифікацію за допомогою ONNX-моделі EfficientNet-B3 та, що ключове для взаємодії серверів, формує та відправляє HTTP POST запит з результатами аналізу (об’єкт AnalysisResponse) на Сервер Даних та Моніторингу ("diplom").

Для цього ImageAnalysisService (не надано, але припускаємо його логіку) використовує HTTP-клієнт (наприклад, RestTemplate або WebClient зі Spring) для звернення до відповідного ендпоінта на сервері "diplom". Об’єкт AnalysisResponse (файл AnalysisResponse.txt) інкапсулює прогнозований клас ("Брак"/"Норма"), впевненість моделі, оригінальне ім’я файлу та можливі повідомлення про помилки.

Сервер Даних та Моніторингу відповідає за прийом оброблених даних від Сервера Аналізу та за взаємодію з клієнтським веб-дашбордом. Для прийому результатів аналізу від Сервера Аналізу, в класі PhotoController (файл PhotoController.txt) реалізовано ендпоінт, що обробляє HTTP POST запити, наприклад, на шляху /api/v1/photos (якщо метод createPhotoMetadata використовується для цього). Цей ендпоінт приймає дані, що відповідають структурі, яку надсилає Сервер Аналізу (ймовірно, адаптовані до PhotoCreateRequestDTO з файлу PhotoController.txt). Приклад цього ендпоінта наведено нижче.

```
@RestController
@RequestMapping("/api/v1/photos")
@RequiredArgsConstructor
public class PhotoController {

    private final PhotoService photoService;

    @PostMapping // Цей ендпоінт приймає дані від Сервера Аналізу
    public ResponseEntity<PhotoResponseDTO> createPhotoMetadata(
            @Valid          @RequestBody          PhotoCreateRequestDTO
photoCreateRequest) {
        PhotoResponseDTO                createdPhoto                =
photoService.createPhotoMetadata(photoCreateRequest);
        return new ResponseEntity<>(createdPhoto, HttpStatus.CREATED);
    }
    // ... (інші ендпоінти для GET запитів від дашборда) ...
}
```

Рисунок 3.10 – Ендпоінт для прийому результатів аналізу на Сервері Даних та Моніторингу

Поле photoUrl у PhotoCreateRequestDTO містить посилання на зображення, яке може зберігатися Сервером Аналізу.

Для взаємодії з клієнтською частиною (веб-дашбордом)

використовується ендпоінт /api/v1/dashboard-data/summary в DashboardDataController, який вже був детально описаний. Він повертає агреговані дані DashboardDataDTO для візуалізації на фронтенді.

3.5.2 Розробка моделі даних та взаємодія з базою даних MySQL

Персистентне зберігання даних про результати інспекції, конфігураційну інформацію про камери, конвеєри та робочі зміни здійснюється за допомогою реляційної системи управління базами даних MySQL. Програмна модель даних на сервері реалізована за допомогою Java-класів сутностей (Entities), які точно відображають структуру таблиць бази даних, визначену на етапі проектування (детальний опис таблиць наведено у розділі 2). Для спрощення та стандартизації взаємодії з базою даних використовується технологія JPA (Java Persistence API) та її реалізація в рамках Spring Data JPA.

Основними класами-сутностями в системі є Photo, Camera, Conveyor та Shift. Кожен з цих класів анотований як @Entity і містить поля, що відповідають стовпцям відповідних таблиць у базі даних. Анотації JPA, такі як @Id та @GeneratedValue, використовуються для визначення первинних ключів та стратегій їх генерації. Зв’язки між сутностями (наприклад, «один-до-багатьох» або «багато-до-одного») реалізовані за допомогою відповідних анотацій, таких як @ManyToOne у класі Photo для встановлення зв’язку з сутностями Camera та Conveyor, що забезпечує цілісність даних та можливість навігації по зв’язаних об’єктах.

Для виконання операцій доступу до даних та більш складних запитів для кожної сутності створено відповідний інтерфейс-репозиторій, який наслідується від JpaRepository з Spring Data JPA. Ці інтерфейси автоматично надають набір стандартних методів для роботи з даними, таких як save(), findById(), findAll(), deleteById(). Крім того, Spring Data JPA дозволяє визначати кастомні методи запитів безпосередньо в інтерфейсах репозиторіїв.

Це може бути зроблено за допомогою анотації @Query, яка дозволяє вказати запит на JPQL (Java Persistence Query Language) або нативному SQL. У PhotoRepository (рисунок 3.11) продемонстровано активне використання анотації @Query для реалізації низки складних запитів, необхідних для вибірки та агрегації даних для дашборду. Такий підхід з використанням Spring Data JPA та репозиторіїв значно спрощує та структурує логіку доступу до даних у сервісному шарі додатку, зменшуючи кількість шаблонного коду.

```
      // --- Для OverallStats та Trend ---
          @Query("SELECT COALESCE(SUM(CASE WHEN p.status = 'DEFECT' THEN 1.0
ELSE 0.0 END) * 100.0 / COUNT(p.photoId), 0.0) " + // <-- Виправлено p.id ->
p.photoId
                  "FROM Photo p WHERE p.creationTime >= :startTime AND
p.creationTime < :endTime")
          Double calculateDefectRateBetween(@Param("startTime") LocalDateTime
startTime, @Param("endTime") LocalDateTime endTime);

          // --- Для DefectsByPeriod - Week ---
          @Query(value = "SELECT CONCAT('Тиж ', WEEK(p.creation_time, 1)) as
weekName, " +
                  "MIN(DATE(p.creation_time)) as startDateOfWeek, " +
                  "MAX(DATE(p.creation_time)) as endDateOfWeek, " +
                  "COALESCE(SUM(CASE WHEN p.status = 'DEFECT' THEN 1.0 ELSE 0.0
END) * 100.0 / COUNT(p.photo_id), 0.0) as defectRate " + // <-- Виправлено p.id
-> p.photo_id
                  "FROM photos p WHERE p.creation_time >= :startDate AND
p.creation_time < :endDate " +
                  "GROUP BY weekName ORDER BY MIN(p.creation_time) ASC",
nativeQuery = true)
          List<Object[]>       findDefectRateByWeekInRange(@Param("startDate")
LocalDateTime startDate, @Param("endDate") LocalDateTime endDate);


          // --- Для DefectsByConveyor ---
          @Query("SELECT cv.conveyorId, cv.name, COUNT(CASE WHEN p.status =
'DEFECT' THEN 1 ELSE NULL END) as defectCount " +
                  "FROM Photo p JOIN p.conveyor cv " +
                  "WHERE p.creationTime >= :startTime AND p.creationTime <
:endTime " +
                  "GROUP BY cv.conveyorId, cv.name ORDER BY defectCount DESC")
          List<Object[]>             findDefectsByConveyor(@Param("startTime")
LocalDateTime startTime, @Param("endTime") LocalDateTime endTime);
```

Рисунок 3.11 – Фрагмент використання анотації @Query у PhotoRepository

3.5.3 Реалізація бізнес-логіки сервера

Бізнес-логіка, що визначає функціональну поведінку автоматизованої системи виявлення дефектів, розподілена між двома серверними компонентами: Сервером Аналізу, відповідальним за обробку зображень, та Сервером Даних та Моніторингу, що забезпечує зберігання результатів та взаємодію з користувачем. Реалізація цієї логіки здійснюється у відповідних сервісних класах Java, анотованих @Service в рамках Spring Framework, що забезпечує управління залежностями та транзакціями.

На Сервері Аналізу основне завдання сервісного шару, представленого класом ImageAnalysisService, полягає у послідовній обробці файлу зображення, зібраного з отриманих від мікроконтролера чанків. Після того, як ImageUploadController успішно формує повний файл зображення, він передається до ImageAnalysisService. Тут відбувається багатоетапний процес, що починається із завантаження та попередньої обробки зображення засобами бібліотеки OpenCV. Наступним кроком є класифікація підготовленого зображення. Для цього воно перетворюється у відповідний тензорний формат та подається на вхід попередньо навченої ONNX-моделі EfficientNet-B3, інференс якоїздійснюється за допомогою спеціалізованого сервісу, наприклад, OnnxInferenceService. Результатом роботи моделі є вектор ймовірностей або логітів, на основі яких сервіс інтерпретує вихідні дані для визначення прогнозованого класу ("Брак" або "Норма") та відповідного рівня впевненості прогнозу. Завершальним етапом роботи ImageAnalysisService є формування об’єкта відповіді, такого як AnalysisResponse, який інкапсулює результати класифікації, включаючи прогнозований клас, впевненість, оригінальне ім’я файлу та можливі повідомлення про помилки. Сформований об’єкт AnalysisResponse потім відправляється за допомогою HTTP POST запиту на Сервер Даних та Моніторингу. Цей запит також містить необхідну метаінформацію, таку як IP-адреса камери (отримана з HttpServletRequest у ImageAnalysisController), ідентифікатори камери та конвеєра, що дозволяє

однозначно асоціювати результат аналізу з джерелом даних.

На Сервері Даних та Моніторингу бізнес-логіка реалізована у наборі спеціалізованих сервісних класів. Ключову роль відіграє PhotoService, який відповідає за обробку даних, отриманих від Сервера Аналізу. Метод createPhotoMetadata, викликаний відповідним контролером (наприклад, PhotoController), приймає дані про проаналізоване. На основі отриманих ідентифікаторів, сервіс витягує з бази даних сутності Camera та Conveyor через відповідні репозиторії, виконуючи при цьому необхідну валідацію, наприклад, перевірку належності камери вказаному конвеєру. Далі створюється нова сутність Photo, посилання на пов’язані сутності камери та конвеєра, статус браку («Брак»/«Норма») та код помилки, що надійшли від Сервера Аналізу. Також формується та зберігається URL-адреса, за якою можна отримати доступ до файлу зображення (який може фізично зберігатися на Сервері Аналізу або іншому файловому сховищі). Сформована сутність Photo потім зберігається в базу даних MySQL за допомогою методу save() відповідного репозиторію (PhotoRepository).

Інший важливий сервісний клас, DashboardService, відповідає за формування комплексної аналітичної інформації для веб-дашборду. Його основний метод getDashboardSummary() агрегує дані з бази, активно використовуючи кастомні запити, визначені в PhotoRepository. Ці запити дозволяють розраховувати різноманітні статистичні показники, такі як середній відсоток браку за різні часові періоди (день, тиждень), визначати тренди його зміни, аналізувати розподіл дефектів за окремими конвеєрами та будувати часові ряди динаміки рівня браку протягом доби. DashboardService також може взаємодіяти з ShiftRepository для аналізу даних у розрізі робочих змін, хоча деталізація цієї логіки залежить від конкретних вимог до дашборду. Всі зібрані та агреговані дані структурують у об’єкт DashboardDataDTO, який потім серіалізується у формат JSON та передається на клієнтську частину для візуалізації. Додаткові сервіси, такі як CameraService, ConveyorService та ShiftService, надають функціонал для управління відповідними

конфігураційними сутностями (створення, читання, оновлення, видалення), що може використовуватися як для адміністративних потреб системи, так і для надання допоміжної інформації на дашборді (наприклад, списку доступних конвеєрів або камер).

Така дворівнева серверна архітектура, де Сервер Аналізу спеціалізується на обчислювально інтенсивній обробці зображень та класифікації, а Сервер Даних та Моніторингу – на зберіганні, агрегації та представленні даних, дозволяє ефективно розподілити навантаження та забезпечити гнучкість і масштабованість системи. Використання Spring Framework на обох серверах забезпечує надійну та структуровану реалізацію всієї необхідної бізнес-логіки.

3.5.4 Розробка інтерфейсу користувача

Інтерфейс користувача автоматизованої системи виявлення дефектів пакування реалізований у вигляді інтерактивної веб-сторінки, що функціонує як інформаційна панель або дашборд. Призначення даного компонента полягає у наданні оператору або іншому уповноваженому персоналу засобів для візуального моніторингу ключових показників ефективності процесу контролю якості, а також для доступу до агрегованої статистичної інформації про виявлені дефекти. Розробка клієнтської частини системи базується на стандартних веб-технологіях: мова гіпертекстової розмітки HTML використовується для визначення семантичної структури та контенту сторінки; каскадні таблиці стилів (CSS), зокрема з використанням можливостей фреймворку Bootstrap 5.3, застосовуються для візуального оформлення та забезпечення адаптивності інтерфейсу; мова програмування JavaScript, з залученням бібліотек jQuery для спрощення маніпуляцій з DOM (Document Object Model) та виконання AJAX-запитів, а також бібліотеки Chart.js для динамічної побудови та візуалізації графічних даних, відповідає за реалізацію інтерактивної логіки та взаємодію з серверною частиною.

Структурно-семантична основа веб-сторінки дашборда визначена у

файлі index.html. Сторінка містить інформаційний заголовок «Моніторинг браку на виробництві», а також елементи управління, такі як кнопка для примусового оновлення даних та індикатор часу останнього успішного оновлення. Візуальне представлення даних організовано за допомогою логічних блоків, стилізованих як картки (card у термінології Bootstrap) з ефектом тіні для покращення візуальної ієрархії. Компонування цих блоків на сторінці здійснюється з використанням адаптивної сітки Bootstrap, що забезпечує коректне відображення інтерфейсу на пристроях з різними діагоналями та роздільними здатностями екранів.

Ключові інформаційні модулі, представлені на дашборді, включають блок вибору часового періоду для аналізу даних, що дозволяє користувачу перемикатися між поданням статистики за день або тиждень. Поруч з цим селектором розташовані картки, що відображають узагальнені показники: поточний середній відсоток браку для обраного періоду (елемент з ідентифікатором average-defect-value) та динаміку зміни цього показника порівняно з попереднім періодом (елемент trend-value з відповідною іконкою trend-icon).

Значну частину інтерфейсу займають графічні візуалізації даних, реалізовані за допомогою бібліотеки Chart.js. До них належить діаграма «Відсоток браку за [період]», яка динамічно перебудовується при зміні обраного користувачем часового періоду, відображаючи рівень браку у вигляді стовпчастої діаграми. Горизонтальна стовпчаста діаграма «Конвеєри з найбільшою кількістю браку (за добу)» надає інформацію про розподіл дефектів за виробничими лініями. Динаміка зміни рівня браку протягом доби візуалізується за допомогою лінійного графіка «Часові піки браку». Додатково, на дашборді представлені картки, що акцентують увагу на критичних показниках: час та значення пікового відсотка браку за обраний період, а також конвеєр з максимальною кількістю дефектів за останню добу. Окремий блок присвячений порівнянню поточного середнього рівня браку з попередньо встановленим цільовим показником (2.5%), що супроводжується

графічним індикатором у вигляді прогрес-бару.

Клієнтська логіка, що забезпечує інтерактивність дашборда, отримання даних з серверної частини та їх динамічне відображення, реалізована мовою JavaScript у файлі dashboard.js. При завантаженні сторінки, а також при ініціюванні користувачем примусового оновлення, виконується асинхронний HTTP GET-запит (AJAX) до серверного ендпоінта /api/v1/dashboard-data/summary. Цей ендпоінт, як детально описано в підрозділі 3.5.1, повертає об’єкт DashboardDataDTO, що містить всю необхідну інформацію для заповнення дашборда. Функція fetchDashboardData(), приклад якої наведено на рисунку 3.12, інкапсулює логіку цього запиту, включаючи обробку успішного отримання даних та можливих помилок при взаємодії з сервером.

```
      function fetchDashboardData() {
          //   ...   (блокування   кнопки   оновлення,   встановлення   статусу
"Завантаження...") ...
          $.ajax({
              url:    API_URL,    //    http://localhost:8080/api/v1/dashboard-
data/summary
              method: 'GET',
              dataType: 'json',
              success: function(data) {
                  dashboardData  =  data;  //  Збереження  отриманих  даних  у
глобальній змінній
                  updateDashboardUI(); // Виклик функції оновлення інтерфейсу
              },
              // ... (обробка помилок та complete callback) ...
          });
```

Рисунок 3.12 – Фрагмент функції отримання даних з сервера

Після успішного завантаження даних з сервера викликається функція updateDashboardUI(), яка відповідає за оновлення всіх візуальних елементів дашборда. Ця функція делегує оновлення специфічних блоків окремим підфункціям: updateOverallStats(overallStats) оновлює картки з загальною статистикою, розраховуючи середній відсоток браку для поточного обраного часового інтервалу; renderCharts() ініціює перемальовування всіх графіків на основі нових даних; updateHighlights(highlights) оновлює інформаційні картки з виділеними показниками. Також відбувається оновлення текстової

інформації про час останнього завантаження даних з сервера, причому дата та час форматуються відповідно до української локалі.

Інтерактивність дашборда, зокрема вибір часового періоду для аналізу, реалізована через обробники подій для кнопок у селекторі timeframe-selector. При виборі нового періоду («День» або «Тиждень»), оновлюється значення глобальної змінної current_timeframe, після чого викликаються функції renderDefectsByPeriodChart та updateOverallStats для негайного перерахунку та перемальовування відповідних елементів інтерфейсу з урахуванням нового часового фільтра.

Застосування фреймворку Bootstrap забезпечує адаптивність та сучасний вигляд інтерфейсу, бібліотека Chart.js надає потужні засоби для інтерактивної візуалізації даних, а jQuery спрощує DOM-маніпуляції та реалізацію AJAX-запитів. У комплексі ці технології дозволяють створити функціональний, інформативний та зручний у використанні дашборд для оперативного моніторингу процесу контролю якості пакування та аналізу ефективності виробничих процесів.

3.6 Реалізація конвеєра обробки зображень

Сервер Аналізу, реалізований на платформі Java з використанням Spring Framework (проект "demo"), виконує центральну функцію в автоматизованій системі – перетворення необроблених візуальних даних, отриманих від мікроконтролерного вузла, на структуровану інформацію про наявність або відсутність дефектів пакувальної продукції. Цей складний процес організований у вигляді конвеєра обробки. Він включає ретельну підготовку вхідних зображень засобами бібліотеки OpenCV та їх подальшу класифікацію за допомогою попередньо навченої та відповідним чином адаптованої моделі глибокого навчання. Важливим підготовчим етапом, що передував інтеграції моделі в серверний компонент, було її спеціалізоване навчання та конвертація у формат, оптимізований для розгортання.

3.6.1 Підготовка, моделі глибокого навчання

Для вирішення задачі автоматичної класифікації зображень пакувань на предмет наявності дефектів (визначення класів "damaged" – пошкоджено, або "intact" – ціле) була обрана згорткова нейронна мережа архітектури EfficientNet-B3. Ця модель відома своєю високою ефективністю, яка досягається завдяки застосуванню методу композитного масштабування (compound scaling), що оптимально балансує глибину, ширину та роздільну здатність мережі для досягнення високої точності при відносно невеликій кількості параметрів та обчислювальних витрат. Процес створення та підготовки моделі для використання в системі контролю якості був виконаний окремо, в середовищі програмування Python з використанням бібліотеки для машинного навчання PyTorch.

Формування навчального датасету передбачало збір та розмітку зображень пакувальних одиниць. Для ефективного навчання моделі та підвищення її здатності до узагальнення на нових, раніше не бачених даних, активно застосовувалися техніки аугментації зображень. Для цього використовувалася бібліотека Albumentations, яка дозволяє застосовувати широкий спектр випадкових перетворень до навчальних зразків. Конвеєр аугментації, визначений у data_transforms_albu_final[‘train’], включав різноманітні геометричні перетворення, такі як горизонтальні та вертикальні віддзеркалення, обертання на довільний кут, випадкові зсуви, зміни масштабу та перспективні спотворення. Також застосовувалися колірні трансформації, що змінюють яскравість, контрастність, насиченість та відтінок зображень, імітуючи можливі варіації освітлення. Додатково використовувалися техніки додавання різних типів шумів (гаусівський, ISO-шум) та ефекти часткової оклюзії (CoarseDropout), що моделюють можливі перешкоди або часткове закриття об’єкта на зображенні.

Навчання моделі EfficientNet-B3 проводилося з використанням підходу трансферного навчання (transfer learning). За основу була взята модель,

попередньо навчена на великому загальному наборі даних ImageNet, що дозволило використовувати вже вивчені нею низькорівневі ознаки зображень.

Для адаптації до специфічного завдання класифікації дефектів пакування, вихідний класифікаційний шар моделі EfficientNet-B3 був замінений на новий. Цей новий шар складається з шару Dropout, призначеного для запобігання перенавчанню шляхом випадкового "вимкнення" частини нейронів під час тренування (з ймовірністю DROPOUT_P = 0.4), та повнозв’язного шару (nn.Linear), кількість виходів якого відповідає числу класів у задачі (два класи: "damaged" та "intact"). Процес тренування був розділений на два основні етапи. На першому етапі відбувалося навчання лише ваг новоствореного класифікаційного шару, тоді як ваги згорткових шарів базової моделі EfficientNet-B3 залишалися "замороженими" (не оновлювалися). На другому етапі проводилося тонке налаштування (fine-tuning), під час якого розморожувалися та донавчалися ваги декількох останніх згорткових блоків ознак моделі разом з класифікатором, що дозволило краще адаптувати всю модель до специфіки цільового датасету пакувань. В якості функції втрат (loss function), що мінімізувалася в процесі навчання, використовувалася перехресна ентропія (nn.CrossEntropyLoss) з додаванням техніки згладжування міток (label smoothing) для покращення узагальнюючої здатності моделі. Оптимізація ваг моделі здійснювалася за допомогою алгоритму AdamW. Швидкість навчання (learning rate) під час тренування ретельно контролювалася та адаптувалася за допомогою спеціалізованих планувальників, таких як lr_scheduler.CosineAnnealingLR та lr_scheduler.ReduceLROnPlateau, які дозволяють поступово зменшувати швидкість навчання для досягнення кращої збіжності. Після завершення процесу тренування, версія моделі, що продемонструвала найкращі показники точності на окремій валідаційній вибірці даних, була збережена у форматі файлу .pth, що містить її натреновані ваги.

Для забезпечення можливості використання натренованої PyTorch-моделі в Java-середовищі Сервера Аналізу, було проведено її конвертацію у

формат ONNX (Open Neural Network Exchange). Цей процес був реалізований за допомогою окремого Python-скрипта. Скрипт завантажував архітектуру моделі EfficientNet-B3, її натреновані ваги з .pth файлу, переводив модель у режим оцінки (model.eval()) для відключення шарів Dropout та інших специфічних для тренування механізмів. Потім створювався фіктивний вхідний тензор необхідного розміру (1, 3, 300, 300 – що відповідає одному зображенню, трьом колірним каналам та роздільній здатності 300x300 пікселів). Після цього виконувалася експортування моделі у файл формату .onnx. Важливим параметром при експорті є opset_version (в даному випадку встановлено на 12), який визначає набір операторів ONNX, що використовуються для представлення операцій моделі. Також вказувалися імена вхідного та вихідного тензорів та можливість використання динамічних осей для розміру пакету (batch size). Формат ONNX є відкритим стандартом для представлення моделей машинного навчання, що дозволяє розгортати моделі, навчені в одному фреймворку (наприклад, PyTorch), в інших середовищах та на різних апаратних платформах, зокрема з використанням бібліотеки ONNX Runtime в Java.

3.6.2 Попередня обробка зображень засобами OpenCV

Моделі глибокого навчання, зокрема архітектури родини EfficientNet, пред’являють строгі вимоги до формату, розміру, колірного простору та статистичних характеристик вхідних зображень. Відхилення від цих вимог може суттєво знизити точність класифікації або навіть унеможливити коректну роботу моделі. Зображення, отримані від камерного модуля OV5640 у форматі JPEG, проходять на сервері багатоетапний конвеєр обробки за допомогою функцій OpenCV, перш ніж будуть перетворені на тензорний формат для подачі на вхід нейронної мережі.

Первинним кроком є декодування JPEG-зображення у внутрішній формат представлення даних OpenCV – багатовимірний масив Mat. Ця

операція виконується за допомогою функції org.opencv.imgcodecs.Imgcodecs.imdecode(), яка перетворює стиснений байтовий потік у піксельне представлення. Отриманий об’єкт Mat зазвичай містить зображення у колірному просторі BGR (Blue-Green-Red), який є стандартним для OpenCV при роботі з кольоровими зображеннями. Оскільки моделі EfficientNet, як правило, навчаються на зображеннях у колірному просторі RGB, необхідна конвертація колірного простору. Це здійснюється за допомогою функції org.opencv.imgproc.Imgproc.cvtColor(sourceMat, destinationMat, Imgproc.COLOR_BGR2RGB), яка виконує перестановку колірних каналів.

Наступним важливим етапом є приведення зображення до фіксованого вхідного розміру, який очікує модель EfficientNet-B3. Стандартний вхідний розмір для цієї архітектури становить 300x300 пікселів. Оскільки роздільна здатність зображень, отриманих з камери OV5640, відрізняється від необхідної, застосовується операція масштабування (resizing). Функція org.opencv.imgproc.Imgproc.resize(sourceMat, destinationMat, newSize, fx, fy, interpolationMethod) бібліотеки OpenCV дозволяє виконати це перетворення. Вибір interpolationMethod є важливим, оскільки різні методи інтерполяції мають різний вплив на якість зображення та обчислювальну складність. Наприклад, білінійна інтерполяція (Imgproc.INTER_LINEAR), що є поширеним вибором для масштабування, обчислює значення нового пікселя як зважене середнє чотирьох найближчих сусідніх пікселів вихідного зображення. Якщо (x, y) – координати пікселя на масштабованому зображенні, а відповідні координати на вихідному зображенні є (x', y'), то значення яскравості $P(x, y)$ (формула 3.1) обчислюється на основі значень $Q_{11}, Q_{12}, Q_{21}, Q_{22}$ в цілочисельних координатах, що оточують (x', y') , та дробових частин $dx = x' - \lfloor x' \rfloor$ і $dy = y' - \lfloor y' \rfloor$:

$$P(x,y) \approx (1-dx)(1-dy)Q_{\lfloor x' \rfloor,\lfloor y' \rfloor} + dx(1-dy)Q_{\lceil x' \rceil,\lfloor y' \rfloor} + (1-dx)dyQ_{\lfloor x' \rfloor,\lceil y' \rceil} + dxdyQ_{\lceil x' \rceil,\lceil y' \rceil}. \quad (3.1)$$

Для зменшення розміру зображення часто рекомендується метод Imgproc.INTER_AREA, який забезпечує краще збереження деталей та уникнення муару шляхом усереднення пікселів відповідної області.

Після приведення до необхідного розміру, дані зображення підлягають нормалізації. Цей процес стандартизує діапазон значень пікселів, що є критично важливим для стабільної роботи та збіжності нейронних мереж. Зазвичай моделі, попередньо навчені на великих наборах даних, таких як ImageNet (на якому часто навчаються базові ваги для EfficientNet), вимагають нормалізації вхідних даних за середнім значенням (mean) та стандартним відхиленням (std) для кожного колірного каналу, розрахованими на цьому навчальному наборі. Типові значення для ImageNet: mean = [0.485, 0.456, 0.406] та std = [0.229, 0.224, 0.225] для каналів R, G, B відповідно (після масштабування значень пікселів до діапазону). Нормалізація кожного каналу виконується за формулою 3.2:

$$X_{norm}^{(c)} = \frac{(X^c/255.0)-\mu^c}{\sigma^{(c)}}, \quad (3.2)$$

де $X(c)$ – вихідне значення пікселя для каналу сс (в діапазоні 0-255),

$\mu(c)$ – середнє значення для каналу с ,

$\sigma(c)$ – стандартне відхилення для каналу с.

В OpenCV ця операція може бути реалізована за допомогою функцій для попіксельних арифметичних операцій з масивами, таких як org.opencv.core.Core.divide() (для ділення на 255.0), org.opencv.core.Core.subtract() (для віднімання середнього) та знову org.opencv.core.Core.divide() (для ділення на стандартне відхилення). Ці операції застосовуються до кожного колірного каналу окремо.

Додатково, на етапі підготовки даних, може виникати необхідність у застосуванні технік покращення зображення (Image Enhancement), якщо якість вхідних зображень є недостатньою. Наприклад, для підвищення контрастності на ділянках з низьким динамічним діапазоном може бути корисним адаптивне гістограмне вирівнювання CLAHE (Contrast Limited Adaptive Histogram Equalization). На відміну від глобального гістограмного вирівнювання, CLAHE обчислює перетворення гістограми для локальних контекстних областей зображення, що дозволяє уникнути надмірного підсилення шумів у відносно однорідних областях. Математично, для кожної такої області обчислюється кумулятивна функція розподілу (CDF) яскравостей F(i), і значення нового пікселя g визначається як

$$g = round(\frac{CDF(f)-CDF_{min}}{M\times N-CDF_{min}} \times (L-1)), \quad (3.3)$$

де f – вихідне значення яскравості,

L – кількість рівнів яскравості,

$M\times N$ – кількість пікселів в області,

CDF_{min} – мінімальне значення CDF.

Обмеження контрасту в CLAHE досягається шляхом кліпування гістограми перед обчисленням CDF. В OpenCV CLAHE реалізовано через клас org.opencv.imgproc.CLAHE, який створюється за допомогою Imgproc.createCLAHE() та застосовується методом apply().

Після виконання всіх необхідних перетворень засобами OpenCV, отриманий масив даних (об’єкт Mat), що представляє нормалізоване та підготовлене зображення, конвертується у формат тензора, сумісний з бібліотекою PyTorch. Цей процес зазвичай включає зміну порядку осей (наприклад, з HWC – висота, ширина, канали – на CHW – канали, висота, ширина), оскільки PyTorch переважно працює з таким форматом для згорткових шарів, та приведення типу даних до float.

Таким чином, конвеєр обробки зображень на сервері, реалізований з

активним використанням функціоналу бібліотеки OpenCV, забезпечує критично важливу адаптацію та нормалізацію вхідних візуальних даних, створюючи оптимальні умови для подальшого аналізу моделлю глибокого навчання EfficientNet-B3 та, як наслідок, для досягнення високої точності виявлення дефектів пакування.

3.6.3 Класифікація зображень за допомогою ONNX Runtime

Після завершення всіх етапів попередньої обробки засобами OpenCV, підготовлене зображення передається до сервісу OnnxInferenceService (для безпосередньої класифікації за допомогою розгорнутої ONNX-моделі EfficientNet-B3.

Клас OnnxInferenceService інкапсулює всю логіку взаємодії з бібліотекою ONNX Runtime. При ініціалізації цього сервісу (наприклад, при старті Spring-додатку, оскільки він, ймовірно, конфігурується як бін, а його метод init() викликається після ін’єкції залежностей або через @PostConstruct, якщо modelBytes передаються в конструктор) відбувається завантаження .onnx файлу моделі та створення активної сесії інференсу ONNX Runtime (ai.onnxruntime.OrtSession). При цьому визначається ім’я вхідного тензора моделі (inputName).

Основний метод класифікації predict(BufferedImage processedImageReadyForModel) спочатку виконує фінальне перетворення вхідного BufferedImage у формат, придатний для ONNX-моделі. Це включає конвертацію зображення у java.nio.FloatBuffer з одночасною нормалізацією значень пікселів (як описано вище, з діленням на 255.0, відніманням середніх значень MEAN та діленням на стандартні відхилення STD для кожного каналу) та зміною порядку осей пікселів на CHW (канали, висота, ширина), оскільки саме такий формат даних очікує більшість згорткових нейронних мереж, включаючи EfficientNet. Потім створюється вхідний тензор ai.onnxruntime.OnnxTensor з отриманого FloatBuffer та відповідними

розмірами (1, 3, 300, 300, де 1 – розмір пакету). Після цього виконується безпосередній інференс моделі шляхом виклику методу session.run(), куди передається словник (Map) з ім’ям вхідного тензора та самим тензором. Результатом роботи цього методу є об’єкт OrtSession.Result, з якого витягується вихідний тензор моделі.

Отриманий масив float[] з вихідними значеннями моделі далі обробляється в ImageAnalysisService. Тут відбувається інтерпретація цих значень: знаходиться індекс класу з максимальною ймовірністю (або значенням логіта), і на основі цього індексу визначається прогнозований клас («damaged» або «intact» зі списку CLASS_NAMES). Також фіксується саме значення максимальної ймовірності, яке може слугувати мірою впевненості моделі у своєму прогнозі. На основі цих даних, а також оригінального імені файлу та можливих повідомлень про помилки, що могли виникнути на етапах обробки, формується об’єкт AnalysisResponse (визначений у файлі AnalysisResponse.txt). Цей об’єкт, що інкапсулює фінальний результат класифікації, потім серіалізується у формат JSON та відправляється HTTP POST запитом на Сервер Даних та Моніторингу для подальшого збереження в базі даних та використання у формуванні статистики для веб-дашборду. Процес відправки результатів реалізований у методі sendPhotoRecordToReportingServer класу ImageAnalysisService.

Таким чином, розроблений на Сервері Аналізу конвеєр обробки зображень є комплексним рішенням, яке ефективно поєднує гнучкі можливості бібліотеки OpenCV для ретельної попередньої обробки та підготовки візуальних даних з високою точністю та продуктивністю класифікації, що забезпечується сучасною моделлю глибокого навчання EfficientNet-B3, розгорнутою за допомогою ONNX Runtime. Це дозволяє системі автоматично аналізувати зображення пакувальних одиниць та генерувати структуровані дані про наявність дефектів для їх подальшого використання в загальній системі контролю якості.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

Даний розділ містить інструкції для користувача щодо експлуатації розробленої автоматизованої системи виявлення дефектів пакування. Описано основні кроки взаємодії з апаратною частиною системи та її програмним інтерфейсом, призначеним для моніторингу процесу контролю якості. Передбачається, що система вже належним чином налаштована та готова до роботи, а користувач має відповідні повноваження для доступу до її компонентів.

4.1 Апаратна конфігурація системи та її запуск

Апаратна частина системи, відповідальна за безпосереднє захоплення зображень, складається з мікроконтролерного вузла, що включає плату STM32F746G-DISCO, камерний модуль OV5640 та Wi-Fi модуль ESP8266. Фізичне компонування цих елементів для лабораторного стенду представлено на рисунку 4.1. На рисунку видно камерний модуль OV5640, зорієнтований для захоплення зображення об’єкта контролю, плату STM32F746G-DISCO з активним TFT-дисплеєм, що відображає поточний відеопотік, та підключений до неї Wi-Fi модуль ESP8266 для передачі даних.

Для початку роботи системи необхідно забезпечити живлення мікроконтролерного вузла STM32F746G-DISCO, наприклад, через USB-порт, як це показано. Також слід переконатися, що Wi-Fi модуль ESP8266 коректно підключений до мікроконтролера та має доступ до Wi-Fi мережі, через яку здійснюватиметься зв’язок із серверним компонентом.

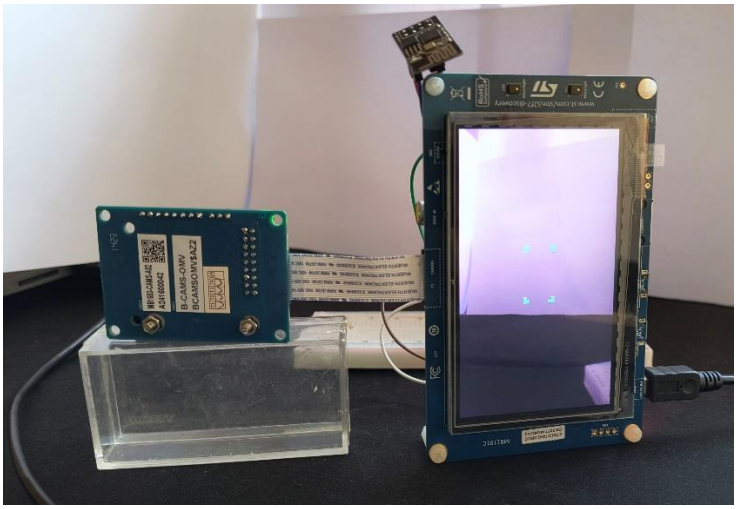

Рисунок 4.1 – Фізичне компонування апаратної частина системи

Серверний компонент системи (Сервер Аналізу та Сервер Даних та Моніторингу) повинен бути запущений та доступний у мережі. Після подачі живлення на мікроконтролерний вузол, він автоматично починає процес ініціалізації периферії, включаючи камеру та дисплей. На TFT-дисплеї плати STM32F746G-DISCO має з’явитися зображення з камери, що свідчить про коректну роботу модуля захоплення та локального відображення.

4.2 Процес автоматичного контролю та аналізу зображень

Після успішної ініціалізації мікроконтролерний вузол переходить у режим автоматичного захоплення зображень пакувальних одиниць. Захоплення ініціюється сигналами від внутрішнього таймера мікроконтролера, що забезпечує періодичне фотографування об’єктів, які потрапляють у поле зору камери. Камера OV5640 налаштовується на формування зображень у форматі JPEG для їх подальшої передачі на Сервер Аналізу. Приклад вихідного зображення, захопленого камерою, та результат його попередньої обробки (кадрування та, можливо, покращення контрасту) на Сервері Аналізу засобами OpenCV, що ілюструє виділення об’єкта пакування з фону, представлено на рисунку. 4.2 та 4.3. На цих рисунках

ліворуч показано вихідне зображення пакувальної одиниці на текстурованому фоні, а праворуч – результат роботи алгоритмів попередньої обробки, де об’єкт пакування виділено та підготовлено для подальшого аналізу нейронною мережею.

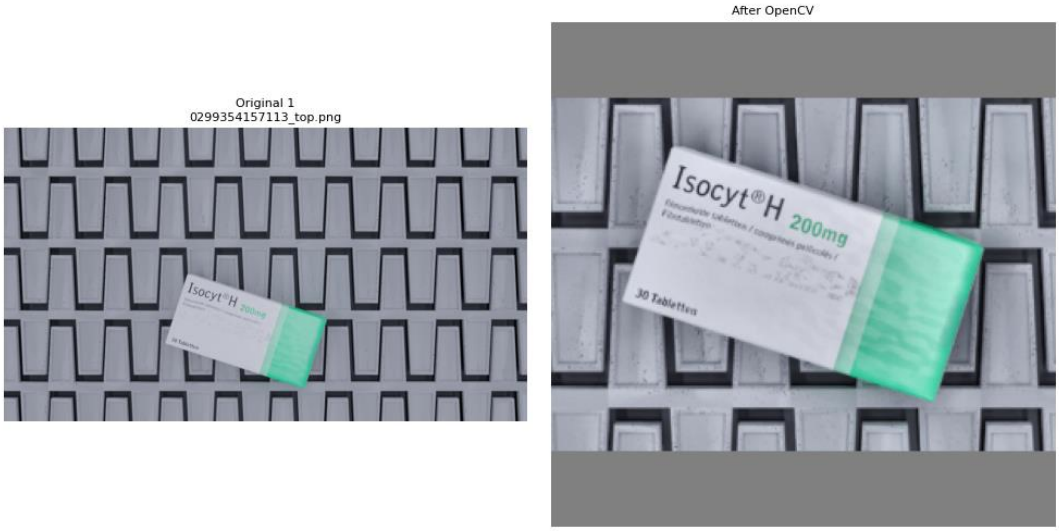

Рисунок 4.3 – Зображення нормального пакування після обробки засобами OpenCV

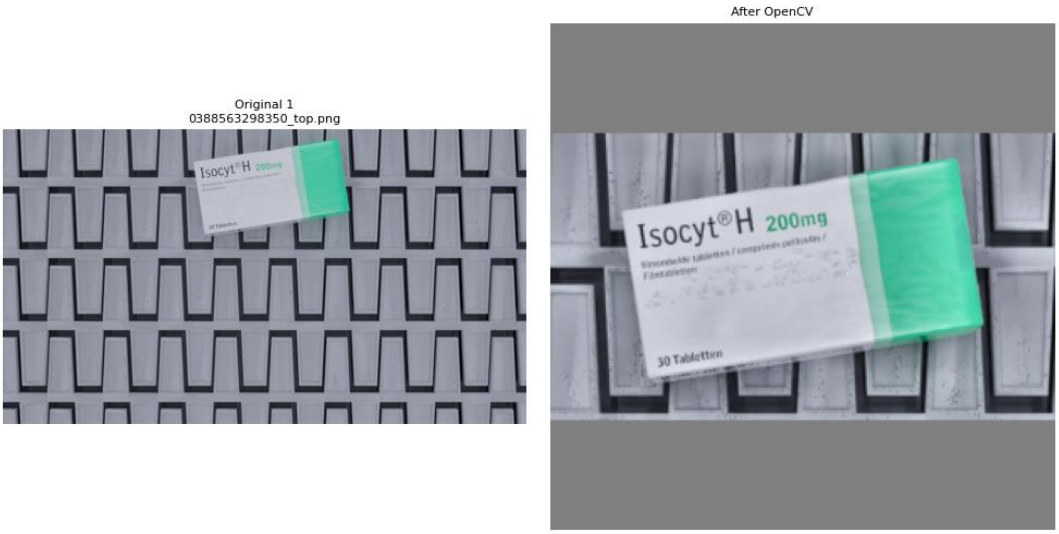

Рисунок 4.3 – Зображення пошкодженого пакування після обробки засобами OpenCV

Захоплені та попередньо оброблені JPEG-зображення передаються по частинах (чанками) через Wi-Fi модуль ESP8266 на Сервер Аналізу. На сервері зображення збирається, проходить етапи покращення якості та нормалізації засобами OpenCV, після чого подається на вхід моделі глибокого навчання EfficientNet-B3 для класифікації на предмет наявності дефектів. Результати аналізу (статус «Брак»/«Норма») разом із метаданими та посиланням на зображення передаються на Сервер Даних та Моніторингу для збереження в базі даних MySQL.

4.3 Взаємодія з веб-інтерфейсомдля

Для оперативного контролю за процесом виявлення дефектів та аналізу статистичних даних користувачеві надається веб-інтерфейс у вигляді дашборда. Доступ до дашборда здійснюється через стандартний веб-браузер шляхом переходу за відповідною URL-адресою серверного додатку. Загальний вигляд веб-інтерфейсу системи моніторингу представлений на рисунку 4.4.

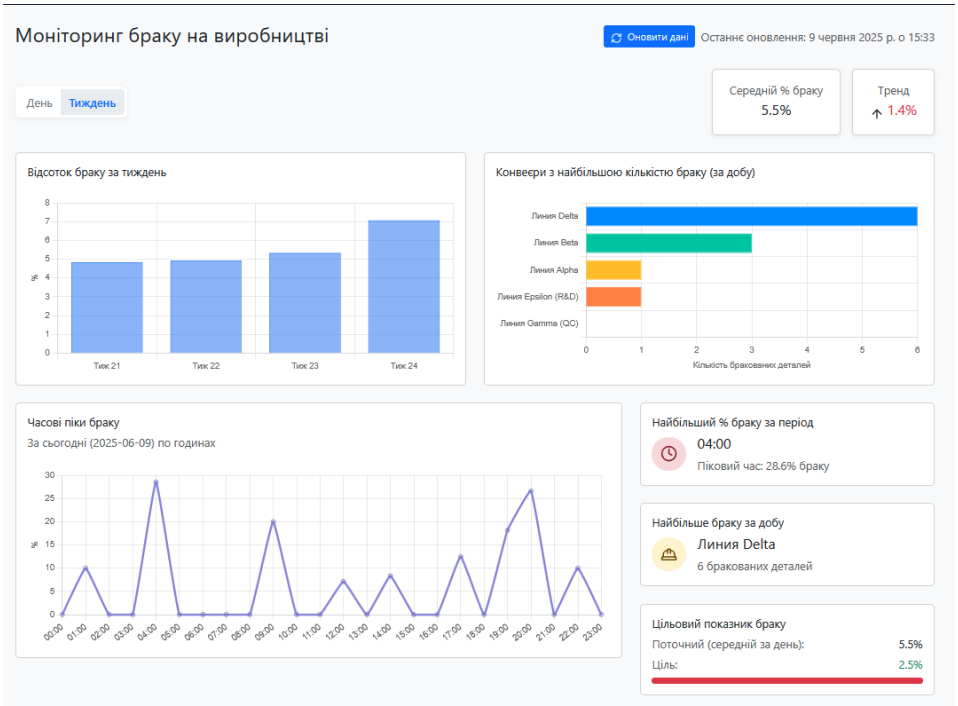

Рисунок 4.4 – Вигляд веб-інтерфейсу системи моніторингу

На дашборді (рисунок 4.4) користувач може бачити актуальну інформацію про процес контролю якості. У верхній частині відображається час останнього оновлення даних та кнопка для їх примусового оновлення. Нижче розташований селектор часового періоду («День», «Тиждень. Основні показники, такі як середній відсоток браку для обраного періоду та тренд його зміни, представлені у вигляді інформаційних карток.

Центральну частину дашборда займають графічні візуалізації. Діаграма «Відсоток браку за тиждень» (або за день, залежно від вибору) наочно демонструє динаміку рівня браку за обраний часовий інтервал. Горизонтальна стовпчаста діаграма «Конвеєри з найбільшою кількістю браку (за добу)» дозволяє ідентифікувати найбільш проблемні виробничі лінії. Лінійний графік "Часові піки браку" показує розподіл виявлених дефектів по годинах протягом поточної доби (або іншого обраного періоду), що може допомогти виявити закономірності у виникненні браку.

Додатково, на дашборді представлені картки з ключовими моментами, такі як «Найбільший % браку за період», що вказує на час та значення максимального зафіксованого рівня браку, та «Найбільше браку за добу», що ідентифікує конвеєр з найбільшою абсолютною кількістю дефектних одиниць. Також відображається порівняння поточного середнього рівня браку з цільовим показником, що супроводжується прогрес-баром для наочної оцінки відповідності встановленим нормам якості.

Взаємодія з дашбордом є інтуїтивно зрозумілою. Користувач може обирати часові періоди для аналізу та оновлювати дані за потреби. Вся інформація, що відображається, динамічно завантажується з Сервера Даних та Моніторингу, який, у свою чергу, агрегує дані, отримані в результаті аналізу зображень Сервером Аналізу.

Таким чином, розроблена система надає користувачеві як можливість локального візуального контролю через дисплей мікроконтролерного вузла, так і потужний інструмент для віддаленого моніторингу та аналізу процесу контролю якості пакування через інтерактивний веб-дашборд.

ВИСНОВКИ

Створена автоматизована система виявлення дефектів пакування ефективно інтегрує мікроконтролерний вузол захоплення та первинної обробки зображень з серверним компонентом, розширеним бібліотекою комп’ютерного зору для їх аналізу та веб-інтерфейсом для моніторингу й візуалізації результатів.

Впроцесі виконання роботи послідовно вирішено низку ключових завдань.

Проведено аналіз предметної області контролю якості пакування, що дозволило виявити недоліки існуючих автоматизованих систем та технології, а саме – відсутність повного комплексу функціоналу, складне масштабування, а також всебічно обґрунтувати актуальність та практичну значущість обраного напрямку дослідження. На основі цього аналізу сформульовано деталізовані функціональні та нефункціональні вимоги до розроблюваної системи, що забезпечило чітке розуміння архітектури очікуваного функціоналу та стало підґрунтям для подальшого проектування.

Розроблено архітектуру апаратно-програмного комплексу. Було обґрунтовано використання розподіленої архітектури, що включає мікроконтролерний вузол та серверний компонент, а також розроблено загальну структурну схему та інформаційну модель взаємодії ключових елементів системи. Здійснено аргументований вибір апаратних (мікроконтролер STM32, камера OV5640, модуль ESP8266) та програмних (мова C, FreeRTOS, HAL для мікроконтролера; Java, Spring Framework, OpenCV, MySQL для сервера) засобів.

Реалізовано програмне забезпечення для обох основних частин системи. Для мікроконтролерного вузла створено прошивку, що забезпечує захоплення зображень, їх відображення на локальному дисплеї та передачу на сервер. Для веб-сервера розроблено додаток, що виконує аналіз отриманих зображень,

зберігає результати в базі даних та надає користувацький веб-інтерфейс для моніторингу.

Аналіз процесу автоматизованого контролю якості пакувальної продукції, що був об’єктом даної роботи, показав ефективність запропонованого апаратно-програмного комплексу. Створений прототип системи демонструє успішне поєднання вбудованих технологій, комп’ютерного зору та серверних рішень для вирішення завдання виявлення дефектів.

Розроблена система має практичне значення як функціональний прототип, що може бути використаний для подальшого вдосконалення та потенційного впровадження з метою підвищення якості продукції та оптимізації виробничих процесів. Результати роботи також можуть знайти застосування в освітній та науково-дослідній діяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A deep learning-based approach for quality control and defect detection for industrial bagging systems / M. Juncker et al. *2020 IEEE 19th international conference on cognitive informatics & cognitive computing (ICCI*CC)*, Beijing, China, 26–28 September 2020. 2020. DOI: https://doi.org/10.1109/iccicc50026.2020.9450251.

2. Application and practice of intelligent control system in microcontroller experiment courses. *Curriculum and teaching methodology*. 2024. Vol. 7, no. 6. DOI: https://doi.org/10.23977/curtm.2024.070620.

3. Automated robotic inspection cells for quality control. *International journal of mechatronics and applied mechanics*. 2021. Vol. 1, no. 10. DOI: https://doi.org/10.17683/ijomam/issue10/v2.1.

4. Computer vision system for defects detection in PET preform / A. Laucka et al. *2016 21st international conference on methods and models in automation and robotics (MMAR)*, Miedzyzdroje, Poland, 29 August – 1 September 2016. 2016. DOI: https://doi.org/10.1109/mmar.2016.7575323.

5. Deep learning and computer vision techniques for enhanced quality control in manufacturing processes / M. R. Islam et al. *IEEE access*. 2024. P. 1. DOI: https://doi.org/10.1109/access.2024.3453664.

6. Dep learning for the quality control of thermoforming food packages / N. Banús et al. *Scientific reports*. 2021. Vol. 11, no. 1. DOI: https://doi.org/10.1038/s41598-021-01254-x.

7. Design of air quality detector based on STM32 / F. Yanjiao et al. *2019 international conference on meteorology observations (ICMO)*, Chengdu, China, 28–31 December 2019. 2019. DOI: https://doi.org/10.1109/icmo49322.2019.9025906.

8. Developing a smart water quality monitoring system using iot technologies / H. Phuoc Vo Luong et al. *IEEJ transactions on sensors and micromachines*. 2024. Vol. 144, no. 10. P. 311–315. DOI:

https://doi.org/10.1541/ieejsmas.144.311.

9. Diaz J. Z., Farooq M. A., Corcoran P. Automatic inspection of seal integrity in sterile barrier packaging: a deep learning approach. *IEEE access*. 2024. P. 1. DOI: https://doi.org/10.1109/access.2023.3348779.

10. Efficient packaging defect detection: leveraging pre-trained vision models through transfer learning / W. Prastiwinarti et al. *Indonesian journal of electrical engineering and computer science*. 2024. Vol. 34, no. 3. P. 2096. DOI: https://doi.org/10.11591/ijeecs.v34.i3.pp2096-2106.

11. Enhancing integrated circuit quality control: a cnn-based approach for defect detection in scanning acoustic tomography images / Y.-T. Jou et al. *Processes*. 2025. Vol. 13, no. 3. P. 683. DOI: https://doi.org/10.3390/pr13030683.

12. Former food products safety evaluation: computer vision as an innovative approach for the packaging remnants detection / M. Tretola et al. *Journal of food quality*. 2017. Vol. 2017. P. 1–6. DOI: https://doi.org/10.1155/2017/1064580.

13. Gao X. Research on automated defect detection system based on computer vision. *Applied and computational engineering*. 2024. Vol. 101, no. 1. P. 192–197. DOI: https://doi.org/10.54254/2755-2721/101/20241010.

14. Huang Z. Research progress of welding defect detection based on machine vision. *Science and technology of engineering, chemistry and environmental protection*. 2024. Vol. 1, no. 9. DOI: https://doi.org/10.61173/14k2c164.

15. IC packaging material identification via a hybrid deep learning framework with cnn–transformer bidirectional interaction / C. Zhang et al. *Micromachines*. 2024. Vol. 15, no. 3. P. 418. DOI: https://doi.org/10.3390/mi15030418.

16. Implementing circularity measurements in industry 4.0-based manufacturing metrology using MQTT protocol and Open CV: a case study / Y. Saif et al. *Plos one*. 2023. Vol. 18, no. 10. P. e0292814. DOI: https://doi.org/10.1371/journal.pone.0292814.

17. Mac T. T., Hung N. T. Automated pill quality inspection using deep learning. *International journal of modern physics B*. 2021. Vol. 35, no. 14n16. P. 2140050. DOI: https://doi.org/10.1142/s0217979221400506.

18. Machine vision-based defect detection using deep learning algorithm / D.-H. Kim et al. *Journal of the korean society for nondestructive testing*. 2020. Vol. 40, no. 1. P. 47–52. DOI: https://doi.org/10.7779/jksnt.2020.40.1.47.

19. Packaging defect detection system based on machine vision and deep learning / J. Sa et al. 2020 5th international conference on computer and communication systems (ICCCS), Shanghai, China, 15–18 May 2020. 2020. DOI: https://doi.org/10.1109/icccs49078.2020.9118413.

20. Qamar R., Zardari B. A. Application of computer vision in manufacturing. *Machine vision and industrial robotics in manufacturing*. Boca Raton, 2024. P. 36–56. DOI: https://doi.org/10.1201/9781003438137-3.

21. Quality analysis of food products through computer aided visual inspection: a review / G. Mathew et al. *2013 international conference on current trends in engineering and technology (ICCTET)*, COIMBATORE, India, 3 July 2013. 2013. DOI: https://doi.org/10.1109/icctet.2013.6675921.

22. Quality control of cigarettes packaging using convolutional neural network / A. Nazar et al. *IOP conference series: materials science and engineering*. 2019. Vol. 462. P. 012002. DOI: https://doi.org/10.1088/1757-899x/462/1/012002.

23. Stm32 based water quality monitoring system / S. K et al. *Interantional journal of scientific research in engineering and management*. 2023. Vol. 07, no. 10. P. 1–11. DOI: https://doi.org/10.55041/ijsrem26015.

24. Stout K. The need for quality control. *Quality control in automation*. Boston, MA, 1985. P. 11–13. DOI: https://doi.org/10.1007/978-1-4684-7499-2_1.

25. Surface defect detection methods for industrial products: a review / Y. Chen et al. *Applied sciences*. 2021. Vol. 11, no. 16. P. 7657. DOI: https://doi.org/10.3390/app11167657.

26. Tie Y., Chen P. Environmental monitoring system design based on STM32 platform. *Theoretical and natural science*. 2024. Vol. 53, no. 1. P. 1–9. DOI: https://doi.org/10.54254/2753-8818/53/20240656.

27. Vu T.-T.-H., Pham D.-L., Chang T.-W. A yolo-based real-time packaging defect detection system. *Procedia computer science*. 2023. Vol. 217. P. 886–894.

DOI: https://doi.org/10.1016/j.procs.2022.12.285.

28. Wu Y., Lu Y. An intelligent machine vision system for detecting surface defects on packing boxes based on support vector machine. *Measurement and control*. 2019. Vol. 52, no. 7-8. P. 1102–1110. DOI: https://doi.org/10.1177/0020294019858175.

29. Package classification with STM32 / Ujin Pavlenko. – URL: https://gitlab.com/ujin.pavlenko/package_classification_with_stm