Форма № Н-9.02

Міністерство освіти і науки України

Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»

(повне найменування вищого навчального закладу)

Відділення інформаційних технологій, менеджменту, туризму і підготовки іноземних громадян

(назва відділення)

Циклова комісія комп'ютерної інженерії

(повна назва циклової комісії)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

фахового молодшого бакалавра

(освітньо-професійного ступеня)

на тему: **Розробка системи комп'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації**

Виконав: студент IV курсу, групи КІ-405

Спеціальності 123 Комп'ютерна інженерія

(шифр і назва спеціальності)

_____ Андрій ШПИРНА

(ім'я та прізвище)

Керівник Андрій НЕДОШИТКО

(ім'я та прізвище)

Рецензент _____

(ім'я та прізвище)

Тернопіль – 2024

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
імені ІВАНА ПУЛЮЯ»

Відділення **інформаційних технологій, менеджменту, туризму та підготовки іноземних громадян**
Циклова комісія **комп'ютерної інженерії**
Освітньо-професійний ступінь **фаховий молодший бакалавр**
Освітньо-професійна програма: **Обслуговування комп'ютерних систем і мереж**
Спеціальність: **123 Комп'ютерна інженерія**
Галузь знань: **12 Інформаційні технології**

ЗАТВЕРДЖУЮ
Голова циклової комісії
комп'ютерної інженерії
_____ Андрій ЮЗЬКІВ
"30" березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шпирна Андрій Ігорович
(прізвище, ім'я, по батькові)

Тема кваліфікаційної роботи: **Розробка системи комп'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації**

керівник роботи Недошитко Андрій Григорович
(прізвище, ім'я, по батькові)

затверджені наказом Відокремленого структурного підрозділу «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя» від 27.03.2026р № 4/9-167.

2. Строк подання студентом роботи: 15 червня 2026 року.

3. Вихідні дані до роботи: плани приміщень, завдання на проєктування, стандарти ANSI/EIA/TIA 568 - "Commercial Building Telecommunications Wiring Standart" і ANSI/EIA/TIA 569 - "Commercial Building Standart for Telecommunications Pathwais and Spaces

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Загальний розділ. Розробка технічного та робочого проєкту. Спеціальний розділ. Економічний розділ. Охорона праці та безпека життєдіяльності.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
- структурна схема системи класифікації об'єктів утилізації;
- функціональна схема системи класифікації об'єктів утилізації;
- архітектура нейронної мережі на основі MobileNet V2;
- блок-схема алгоритму роботи системи;
- таблиця техніко-економічних показників.

6. Консультанти розділів роботи

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Богдана МАРТИНЮК викладач		
Охорона праці та безпека життєдіяльності	Володимир ШТОКАЛО викладач		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання	01.04	
2	Збір і узагальнення інформації	08.05	
3	Написання першого розділу	15.05	
4	Розробка технічного та робочого проекту	22.05	
5	Написання спеціального розділу	28.05	
6	Розрахунок економічної частини	1.06	
7	Написання розділу охорони праці	3.06	
8	Виконання графічної частини	8.06	
9	Оформлення проєкту	10.06	
10	Погодження нормоконтролю	11.06	
11	Попередній захист роботи	12.06	
12	Захист кваліфікаційної роботи	**23.06**	

7. Дата видачі завдання: 31 березня 2026 року

Студент _____ Андрій ШПИРНА
(підпис) (ім'я та прізвище)

Керівник роботи _____ Андірй НЕДОШИТКО
(підпис) ім'я та прізвище)

АНОТАЦІЯ

Шпирна А.І. Розробка системи комп'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації: кваліфікаційна робота на здобуття освітньо-кваліфікаційного ступеня молодшого бакалавра, спеціальність 123 Комп'ютерна інженерія. Тернопіль: ВСП «ТФК ТНТУ», 2026. – 85 с.

Метою роботи є розробка системи комп'ютерного зору для автоматичної класифікації відходів на базі ESP32-CAM для інтеграції у домашній смітник з автоматичним сортуванням.

Проведено аналіз існуючих рішень, обрано модуль ESP32-CAM з камерою OV2640 як апаратну платформу. Для навчання використовується Python, TensorFlow та архітектура MobileNetV2 з технологією перенесення навчання, для обробки зображень — OpenCV.

Розроблено модуль навчання нейронної мережі та модуль розпізнавання у реальному часі. Система класифікує відходи на чотири категорії: пластик, скло, метал, органіка. Результати відображаються через веб-інтерфейс ESP32 у локальній мережі Wi-Fi.

Досягнуто точність класифікації 85% при часі обробки 1,8 с. Підготовлено інструкцію з експлуатації, методику перевірки та проведено економічне обґрунтування.

Кваліфікаційна робота складається з 58 аркушів.

ANNOTATION

Shpyrna A.I. Development of a computer vision system based on machine learning algorithms for waste object classification: qualification paper for the educational-professional degree of professional junior bachelor, specialty 123 Computer Engineering. Ternopil: VSP “TFK TNTU”, 2026. – 85 p.

The aim of this work is to develop a computer vision system for automatic classification of recyclable objects based on the ESP32-CAM module for integration into a home waste bin with automatic sorting.

An analysis of existing solutions was conducted; the ESP32-CAM module with OV2640 camera was selected as the hardware platform. Python, TensorFlow, and MobileNetV2 with transfer learning are used for neural network training; OpenCV is used for image processing.

A neural network training module and a real-time recognition module were developed. The system classifies waste into four categories: plastic, glass, metal, and organic. Results are displayed via the ESP32 web interface over a local Wi-Fi network.

A classification accuracy of 85% was achieved with an average processing time of 1.8 seconds. An operation manual, testing methodology, and economic feasibility justification were prepared.

The qualification paper consists of 58 pages.

ЗМІСТ

ВСТУП

Проблема поводження з відходами є однією з найгостріших екологічних проблем сучасності. За даними Світового банку [28], щороку у світі утворюється понад 2 мільярди тонн твердих побутових відходів, і прогнозується, що до 2050 року цей показник зросте до 3,4 мільярда тонн. Значна частина цих відходів потрапляє на сміттєзвалища без попереднього сортування, що призводить до забруднення ґрунтів, водних ресурсів та атмосфери.

Одним із ключових факторів, що ускладнює процес утилізації, є неправильне сортування відходів на побутовому рівні. Люди часто не знають, до якої категорії належить той чи інший предмет, або ж просто не мають бажання витрачати на це час. Внаслідок цього значна кількість матеріалів, які могли б бути перероблені, потрапляє на звалища, а небезпечні відходи змішуються з побутовим сміттям.

В Україні проблема поводження з побутовими відходами стоїть особливо гостро. За даними Міністерства захисту довкілля та природних ресурсів, рівень переробки побутових відходів в Україні становить лише 5–7 відсотків, що значно нижче середньоєвропейського показника у 47 відсотків. Більшість побутових відходів потрапляє на полігони та несанкціоновані звалища, загальна площа яких перевищує 9 тисяч гектарів. Впровадження автоматизованих систем сортування на побутовому рівні може стати одним із ключових факторів покращення цієї ситуації.

Європейський Союз у рамках директиви 2018/851 встановив амбітну мету: до 2035 року рівень переробки побутових відходів повинен досягти 65 відсотків. Для України, яка прагне до євроінтеграції, виконання цих вимог є стратегічно важливим. Автоматизація процесу сортування на побутовому рівні може стати одним із ключових факторів досягнення цих показників.

Розвиток технологій комп'ютерного зору та машинного навчання відкриває нові можливості для автоматизації процесу класифікації відходів безпосередньо у домашніх умовах. Сучасні алгоритми глибокого навчання, зокрема згорткові нейронні мережі, здатні розпізнавати об'єкти на зображеннях з високою точністю навіть на мікроконтролерних платформах, таких як ESP32 [14], що робить їх перспективним інструментом для побудови компактних побутових пристроїв автоматичного сортування.

Метою даної кваліфікаційної роботи є розробка системи комп'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації. Система призначена для інтеграції у компактний домашній смітник з автоматичним сортуванням, який являє собою циліндричний контейнер з обертовою платформою. Об'єкт сміття розміщується на верхній платформі, камера ESP32 аналізує його та визначає один з чотирьох типів відходів: пластик, скло, метал або органіка. Після класифікації циліндр повертається у відповідне положення, а платформа нахиляється, скидаючи сміття у потрібний відсік контейнера.

У рамках даної кваліфікаційної роботи розробляється саме програмна частина системи, а саме: модуль навчання нейронної мережі для класифікації відходів та модуль розпізнавання, який аналізує зображення з камери та визначає тип сміття. Результат класифікації відображається через веб-інтерфейс ESP32, доступний у локальній мережі. Механічна частина пристрою, а саме обертовий циліндр із сервоприводом та платформа з нахилом, не є предметом даної роботи.

Практичне значення розробки полягає у тому, що створена система може бути використана як основа для побудови повністю автоматизованого домашнього смітника, який допоможе мешканцям квартир та приватних будинків правильно сортувати відходи без додаткових зусиль. Крім того, система може використовуватися в освітніх цілях для підвищення екологічної свідомості.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– провести аналіз існуючих рішень у сфері автоматичної класифікації відходів та обґрунтувати вибір технологій;
– обрати та обґрунтувати елементну базу та програмні засоби для реалізації системи;
– розробити архітектуру системи та функціональну схему взаємодії модулів;
– реалізувати модуль навчання нейронної мережі на основі MobileNetV2 [22] з використанням технології перенесення навчання;
– реалізувати модуль розпізнавання об’єктів у реальному часі з використанням OpenCV [11] та TensorFlow [7];
– розробити веб-інтерфейс для відображення результатів класифікації через ESP32;
– підготувати інструкцію з експлуатації та методику перевірки функціонування системи.

У першому розділі кваліфікаційної роботи обґрунтовано актуальність обраної теми та проведено аналітичний огляд існуючих рішень у сфері автоматичної класифікації відходів. У другому розділі описано процес розробки технічного та робочого проекту, включаючи аналіз технічного завдання, вибір програмних та апаратних засобів, розробку архітектури системи, алгоритму роботи та написання програмного коду. У третьому розділі наведено інструкцію з експлуатації та методику перевірки функціонування системи.

1 ЗАГАЛЬНА ЧАСТИНА

1.1 Обґрунтування актуальності теми кваліфікаційної роботи

Метою даної кваліфікаційної роботи є розробка системи комп'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації, призначеної для роботи у складі компактного домашнього смітника з автоматичним сортуванням.

Проблема управління побутовими відходами набуває дедалі більшої гостроти у контексті сталого розвитку. Швидке зростання обсягів споживання призводить до постійного збільшення кількості побутових відходів. За оцінками міжнародних екологічних організацій, лише близько 13 відсотків усіх побутових відходів у світі піддається переробці, тоді як решта потрапляє на звалища або спалюється.

Особливо гостро стоїть проблема сортування сміття на побутовому рівні. У більшості українських домогосподарств відсутня інфраструктура для роздільного збору відходів, а мешканці часто не мають достатніх знань або мотивації для правильного сортування. Згідно з Національною стратегією управління відходами в Україні до 2030 року, одним із пріоритетних напрямків є впровадження інноваційних технологій для підвищення ефективності сортування на етапі збору.

Комп'ютерний зір як напрямок штучного інтелекту займається розробкою методів, які дозволяють комп'ютерам отримувати високорівневе розуміння цифрових зображень. У контексті класифікації відходів комп'ютерний зір дає змогу автоматично аналізувати візуальну інформацію про об'єкт та визначати його категорію без участі людини.

Згорткові нейронні мережі, або CNN [10], є одним з найпотужніших інструментів для класифікації зображень. Архітектура згорткових мереж натхненна будовою зорової кори мозку ссавців та побудована таким чином,

щоб автоматично витягувати ієрархічні ознаки із зображень — від простих елементів на перших шарах до складних абстрактних представлень на глибших.

Технологія перенесення навчання, або transfer learning [27], дозволяє використовувати модель, попередньо навчену на великому наборі даних, та адаптувати її для нової задачі шляхом донавчання на відносно невеликому спеціалізованому наборі даних, що суттєво спрощує процес розробки.

MobileNetV2 є однією з найбільш ефективних архітектур для вбудованих систем. Ця модель використовує інвертовані залишкові блоки з лінійним вузьким місцем, що забезпечує високу точність при значно менших обчислювальних витратах. Завдяки цьому MobileNetV2 може бути адаптована для роботи на мікроконтролері ESP32.

Мікроконтролер ESP32 є потужною та доступною платформою для IoT-проєктів з вбудованою підтримкою Wi-Fi та Bluetooth. Завдяки модулю ESP32-CAM [13], який поєднує мікроконтролер з камерою OV2640, можна побудувати автономну систему комп'ютерного зору для класифікації об'єктів у реальному часі.

Тема кваліфікаційної роботи є актуальною, адже автоматизована система класифікації об'єктів утилізації, інтегрована у компактний домашній смітник, може значно спростити процес сортування побутових відходів та зменшити екологічне навантаження на довкілля.

1.2 Аналітичний огляд існуючих рішень

Серед існуючих систем автоматичної класифікації відходів можна виділити як промислові рішення, так і дослідницькі проєкти та побутові пристрої, що використовують різні підходи до розпізнавання та сортування сміття. Кожен із цих підходів має свої переваги та обмеження залежно від масштабу застосування та технічної складності реалізації. Спільною рисою більшості сучасних рішень є використання алгоритмів машинного навчання для

автоматичного визначення категорії відходів за їх візуальними характеристиками. Розглянемо найбільш характерні з них.

Система ZenRobotics

ZenRobotics [30] є фінською компанією, що спеціалізується на розробці роботизованих систем сортування відходів з використанням технологій штучного інтелекту. Система ZenRobotics Recycler використовує комбінацію сенсорів, включаючи камери видимого та інфрачервоного спектру, металодетектори та 3D-сканери, для ідентифікації різних типів відходів на конвеєрній стрічці.

Алгоритми машинного навчання, інтегровані у систему, аналізують дані з усіх сенсорів одночасно та приймають рішення про категорію кожного об'єкта. Після класифікації роботизований маніпулятор автоматично захоплює об'єкт та переміщує його у відповідний контейнер. За даними виробника, система здатна обробляти до 4000 об'єктів на годину з точністю розпізнавання понад 95 відсотків при сортуванні до 40 різних категорій матеріалів.

Архітектура системи передбачає модульну побудову, що дозволяє адаптувати кількість роботизованих маніпуляторів та набір сенсорів залежно від конкретних виробничих потреб замовника. Програмне забезпечення ZenRobotics постійно вдосконалює модель розпізнавання на основі накопичених даних, що забезпечує поступове підвищення точності класифікації в процесі експлуатації. Крім того, система підтримує інтеграцію з існуючими виробничими лініями та SCADA-системами управління підприємством.

Основними перевагами системи ZenRobotics є висока продуктивність та робота у безперервному режимі. Проте вартість від 300 000 євро та промисловий масштаб роблять це рішення непридатним для побутового використання. Система орієнтована виключно на великі сміттєпереробні підприємства та логістичні центри, де обсяг відходів вимірюється тоннами на

добу, а окупність інвестицій досягається лише за умови тривалої безперервної експлуатації. У таблиці 1.1 наведено технічні характеристики системи ZenRobotics Recycler.

Таблиця 1.1 – Технічні характеристики ZenRobotics Recycler

Параметр	**Значення**
Продуктивність	до 4000 об'єктів/год
Точність розпізнавання	понад 95%
Типи сенсорів	RGB-камери, ІЧ-сенсори, металодетектори, 3D-сканери
Кількість категорій сортування	до 40
Споживана потужність	15 кВт
Вартість системи	від 300 000 EUR

Проєкт TrashNet

TrashNet [29] є відкритим дослідницьким проєктом, розробленим у Стенфордському університеті, який являє собою набір даних зображень відходів та базову модель класифікації на основі згорткових нейронних мереж. Набір даних TrashNet містить 2527 зображень, розподілених на шість категорій: скло, папір, картон, пластик, метал та загальне сміття.

Модель класифікації побудована на архітектурі VGG-16 [24], яка була адаптована для задачі класифікації відходів за допомогою технології перенесення навчання. У ході експериментів автори досягли точності класифікації близько 75 відсотків на тестовому наборі даних. Попри відносно невисоку точність, проєкт зробив значний внесок у розвиток досліджень у сфері автоматичної класифікації відходів, ставши відправною точкою для численних подальших робіт. Проте модель не призначена для роботи на вбудованих системах та не включає модуль виявлення об'єктів у реальному часі.

Система Bin-e

Bin-e [8] являє собою розумний сміттєвий контейнер, розроблений польською компанією, який автоматично сортує відходи за допомогою вбудованих камер та алгоритмів комп'ютерного зору. Контейнер оснащений набором сенсорів, внутрішньою камерою та компресором для пресування відходів. Bin-e підтримує сортування на чотири основні категорії: папір, скло, пластик та метал.

Цей продукт є найближчим аналогом до розроблюваної системи, проте він призначений для громадських місць та офісів, має значні габарити та високу вартість. Зазначені характеристики роблять його непрактичним для використання у звичайній квартирі чи приватному будинку, де пріоритетом є компактність та доступна ціна.

Мобільний додаток Oscar

Oscar [30] є мобільним додатком, який використовує камеру смартфона для ідентифікації типу відходів та надання рекомендацій щодо їх правильної утилізації. Додаток працює на основі хмарних сервісів машинного навчання та підтримує розпізнавання широкого спектру матеріалів, включаючи пластик, скло, папір, метал та електронні відходи, що робить його універсальним інструментом для інформування населення про правила поводження з побутовими відходами. Основним недоліком є залежність від хмарних сервісів та мережі Інтернет, а також відсутність фізичного механізму сортування – додаток лише інформує користувача, але не виконує сортування автоматично. Таким чином, Oscar вирішує лише інформаційну задачу, залишаючи фізичне сортування на відповідальності самого користувача, що суттєво знижує його практичну ефективність у порівнянні з апаратними рішеннями автоматичного сортування.

Набір даних TACO

TACO [23], або Trash Annotations in Context, є масштабним відкритим набором даних для задачі виявлення та сегментації сміття у природних умовах.

На відміну від TrashNet, де зображення зроблені у лабораторних умовах на білому фоні, TACO містить фотографії сміття у реальних умовах: на вулицях, у парках та на узбіччях доріг. Набір даних містить понад 1500 зображень з 4784 анотаціями, розподіленими на 60 підкатегорій відходів.

TACO надає цінний ресурс для навчання моделей, здатних працювати у складних умовах реального середовища. Різноманітність умов зйомки та велика кількість підкатегорій роблять цей набір даних особливо корисним для дослідницьких задач, а відкрита платформа розмітки дозволяє дослідникам з усього світу постійно його поповнювати. Проте набір даних не включає готових моделей для вбудованих систем та потребує значних обчислювальних ресурсів, що ускладнює його застосування на мікроконтролерних платформах, таких як ESP32.

У таблиці 1.2 наведено порівняльну характеристику розглянутих рішень.

Таблиця 1.2 – Порівняльна характеристика існуючих рішень

Характеристика	**ZenRobotics**	**TrashNet**	**Bin-e**
Призначення	Промислове	Дослідне	Офісне
Реальний час	Так	Ні	Так
Кількість категорій	до 40	6	4
Точність	>95%	~75%	~90%
Платформа	Сервер	ПК	Вбудована
Вартість	Висока	Безкоштовно	Середня
Побутове використання	Ні	Ні	Частково

Аналіз існуючих рішень показує, що на ринку відсутній доступний та компактний пристрій для автоматичного сортування побутових відходів у домашніх умовах. Промислові системи, такі як ZenRobotics, мають надто великі габарити та вартість, що унеможливлює їх використання у квартирах чи приватних будинках. Дослідницькі проєкти, зокрема TrashNet та TACO,

надають цінні набори даних, проте не пропонують готових апаратних рішень. Мобільні додатки лише інформують користувача про тип відходів, але не забезпечують фізичного сортування та потребують постійного підключення до хмарних сервісів.

Варто також зазначити, що система Bin-e, яка є найближчим аналогом за функціональністю, орієнтована на офісні та громадські простори та має вартість, недоступну для звичайного домогосподарства. Жодне з розглянутих рішень не забезпечує автономну роботу на базі компактного мікроконтролера без підключення до мережі Інтернет.

Розроблювана система заповнює цю нішу: вона базується на доступному мікроконтролері ESP32 з камерою OV2640, класифікує відходи на чотири категорії (пластик, скло, метал, органіка), працює повністю автономно та призначена для інтеграції у компактний домашній смітник з автоматичним розподілом сміття по відсіках.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

2.1 Аналіз технічного завдання

Метою даної кваліфікаційної роботи є розробка програмної частини системи комп'ютерного зору для автоматичної класифікації об'єктів утилізації, призначеної для роботи у складі компактного домашнього смітника.

Концепція домашнього смітника з автоматичним сортуванням передбачає наступну конструкцію. Пристрій являє собою циліндричний контейнер, розділений на чотири відсіки для різних типів відходів: пластик, скло, метал та органіка. У верхній частині контейнера розташована приймальна платформа, на яку користувач розміщує об'єкт сміття. Над платформою встановлено модуль ESP32-CAM з камерою, яка фотографує об'єкт та за допомогою нейронної мережі визначає його тип. Після класифікації циліндр повертається за допомогою сервоприводу таким чином, щоб під платформою опинився потрібний відсік, після чого платформа нахиляється і скидає сміття у відповідний контейнер.

Загальний вигляд концепції домашнього смітника з автоматичним сортуванням зображено на рисунку 2.1.

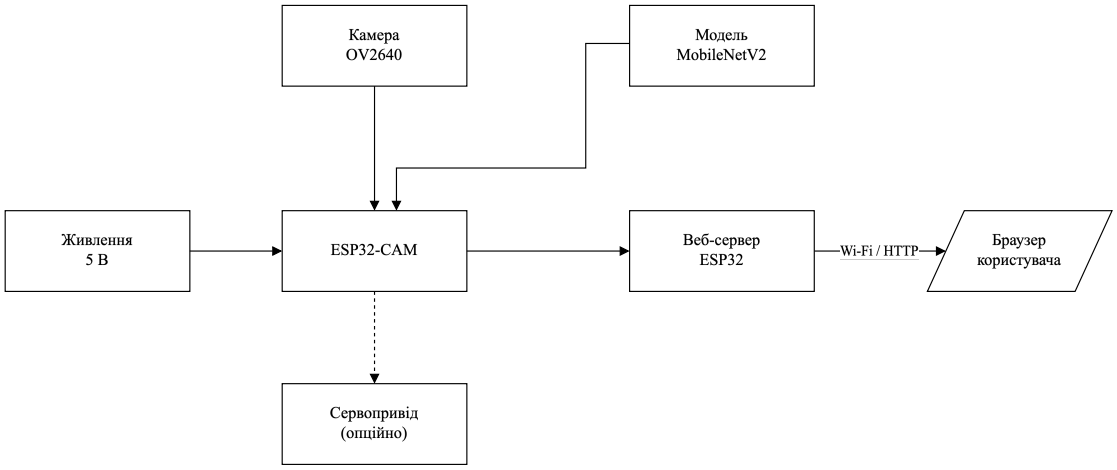


Рисунок 2.1 – Концептуальна схема домашнього смітника з автоматичним сортуванням

У рамках даної кваліфікаційної роботи розробляється виключно програмна частина системи, яка включає два основних компоненти:

Перший компонент – модуль навчання нейронної мережі. Він виконує підготовку набору даних зображень відходів чотирьох категорій, побудову архітектури моделі на основі MobileNetV2 з використанням технології перенесення навчання, навчання та донавчання моделі, а також збереження готової моделі для подальшого розгортання.

Другий компонент – модуль розпізнавання у реальному часі. Він захоплює зображення з камери ESP32, виконує попередню обробку кадру, виявляє об'єкт на платформі за допомогою аналізу контурів, класифікує його нейронною мережею та відображає результат через веб-інтерфейс ESP32, доступний у локальній мережі Wi-Fi.

Функціональна схема системи класифікації зображена на рисунку 2.2.

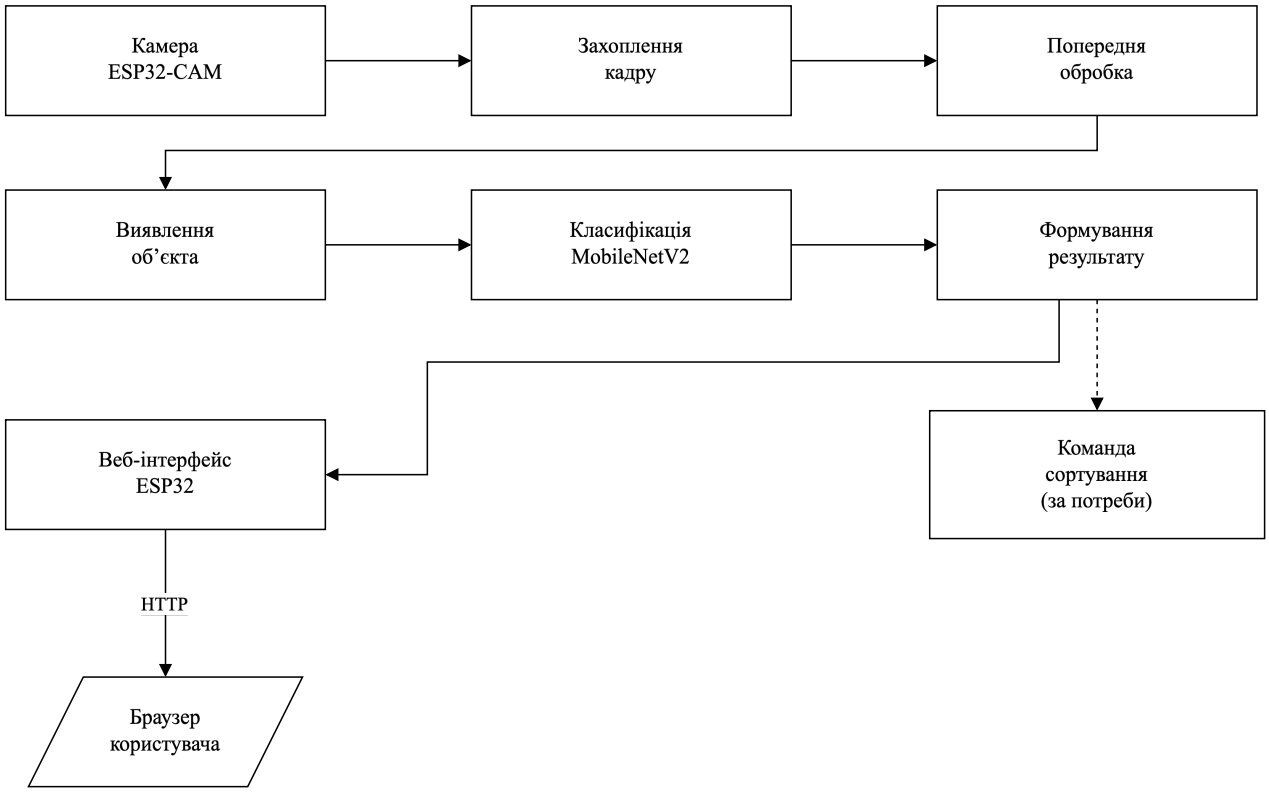


Рисунок 2.2 – Функціональна схема системи класифікації об'єктів утилізації

Функціональні вимоги до системи:

– захоплення зображення з камери ESP32-CAM;
– автоматичне виявлення об'єкта на приймальній платформі;
– класифікація об'єкта за чотирма категоріями: пластик, скло, метал, органіка;
– відображення результату класифікації з рівнем впевненості через веб-інтерфейс;
– можливість перенавчання моделі на оновленому наборі даних.

Нефункціональні вимоги до системи:

– час класифікації одного об'єкта не більше 3 секунд;
– точність класифікації не менше 85 відсотків на валідаційній вибірці;
– можливість автономної роботи без підключення до хмарних сервісів;
– доступність веб-інтерфейсу у локальній мережі Wi-Fi.

2.2 Опис і обґрунтування вибору елементної бази та програмних засобів

Система розробляється з використанням мікроконтролерної платформи ESP32, мови програмування Python для навчання моделі та набору спеціалізованих бібліотек.

Мікроконтролер ESP32-CAM

ESP32-CAM є компактним модулем на основі мікроконтролера ESP32-S, який поєднує потужний двоядерний процесор з тактовою частотою до 240 МГц, вбудовані модулі Wi-Fi та Bluetooth, а також інтерфейс для підключення камери OV2640 з роздільною здатністю до 2 мегапікселів. Модуль має 520 КБ внутрішньої оперативної пам'яті та підтримку зовнішнього PSRAM обсягом 4 МБ, що дозволяє обробляти зображення безпосередньо на пристрої. Вбудований модуль Wi-Fi стандарту 802.11 b/g/n забезпечує передачу результатів класифікації через локальну мережу без використання додаткових

компонентів. Компактні розміри модуля 27×40,5×4,5 мм та низьке енергоспоживання роблять його оптимальним вибором для інтеграції у побутові пристрої з автономним живленням.

Зовнішній вигляд модуля ESP32-CAM зображено на рисунку 2.3.

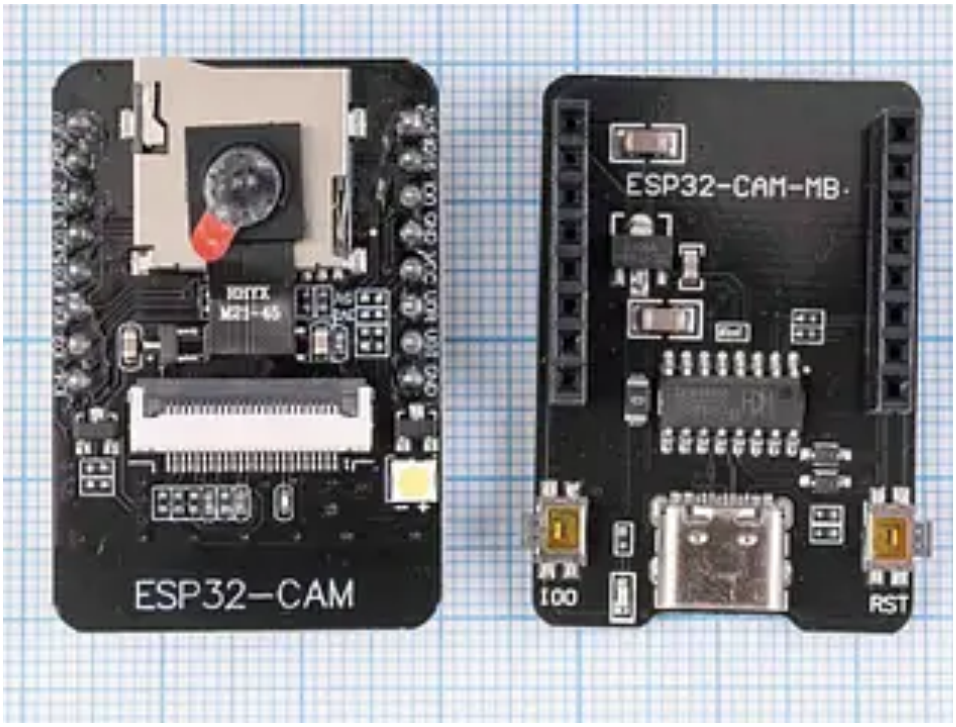


Рисунок 2.3 – Зовнішній вигляд модуля ESP32-CAM

При виборі апаратної платформи розглядалися також альтернативні варіанти: Raspberry Pi Zero W та Arduino Nano 33 BLE Sense. Raspberry Pi Zero W має значно більшу обчислювальну потужність завдяки процесору ARM з тактовою частотою 1 ГГц та 512 МБ оперативної пам’яті, проте його вартість у 3–4 рази перевищує вартість ESP32-CAM, а споживання енергії значно вище, що робить його менш придатним для автономного побутового пристрою. Arduino Nano 33 BLE Sense має вбудовані сенсори руху та мікрофон, проте не підтримує підключення камери. ESP32-CAM забезпечує оптимальний баланс

між продуктивністю, вартістю та енергоспоживанням для задачі класифікації зображень у побутовому пристрої.

Технічні характеристики модуля ESP32-CAM наведено у таблиці 2.1.

Таблиця 2.1 – Технічні характеристики ESP32-CAM

Параметр	**Значення**
Процесор	ESP32-S, двоядерний, до 240 МГц
Оперативна пам'ять	520 КБ SRAM + 4 МБ PSRAM
Flash-пам'ять	4 МБ
Камера	OV2640, до 2 МП
Бездротовий зв'язок	Wi-Fi 802.11 b/g/n, Bluetooth 4.2
Напруга живлення	5 В (через USB) або 3.3 В
Розміри модуля	27 × 40.5 × 4.5 мм
Вартість модуля	~150–250 грн

Мова програмування Python та фреймворк TensorFlow

Для навчання моделі нейронної мережі використовується мова програмування Python версії 3.10 у поєднанні з фреймворком глибокого навчання TensorFlow. Python забезпечує просту та зрозумілу синтаксичну структуру, наявність великої кількості спеціалізованих бібліотек, а TensorFlow надає інструменти для побудови, навчання та експорту моделей у формати, сумісні з вбудованими системами, зокрема TensorFlow Lite [27].

Бібліотека OpenCV

OpenCV є найбільш розповсюдженою бібліотекою комп'ютерного зору з відкритим вихідним кодом. У даному проєкті OpenCV використовується для захоплення відеопотоку, перетворення кольорових просторів, застосування фільтру Гаусса, виявлення країв за алгоритмом Кенні [20], пошуку та аналізу контурів, а також для відображення результатів розпізнавання.

Архітектура MobileNetV2.

MobileNetV2 є легковаговою архітектурою згорткової нейронної мережі, спроєктованою для ефективної роботи на мобільних та вбудованих пристроях. Ключовою особливістю є використання інвертованих залишкових блоків із глибинно роздільними згортками, що дозволяє суттєво зменшити кількість параметрів. Порівняння MobileNetV2 з іншими архітектурами наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння архітектур нейронних мереж

Архітектура	**Параметри, млн**	**Top-1 точність, %**	**Розмір моделі, МБ**	**Час інференсу, мс**
VGG-16	138	71.3	528	145
ResNet-50	25.6	76.0	98	58
MobileNetV2	3.4	71.8	14	22
EfficientNet-B0	5.3	77.1	20	35

Як видно з таблиці 2.2, MobileNetV2 має найменшу кількість параметрів та найменший час інференсу, що робить її оптимальним вибором для роботи на ESP32 з обмеженими ресурсами.

2.3 Розробка функціональної схеми системи

Архітектура розробленої системи побудована за модульним принципом і складається з п'яти функціональних блоків, які забезпечують повний цикл від захоплення зображення до відображення результату класифікації. Модульний підхід дозволяє незалежно розробляти та вдосконалювати окремі компоненти системи, не порушуючи роботу інших блоків, а також спрощує процес налагодження та подальшого розширення функціональності. Взаємодія між модулями організована таким чином, що вихідні дані кожного блоку є вхідними для наступного, що забезпечує послідовну та узгоджену обробку інформації на кожному етапі роботи системи.

Структурна схема системи класифікації зображена на рисунку 2.4.

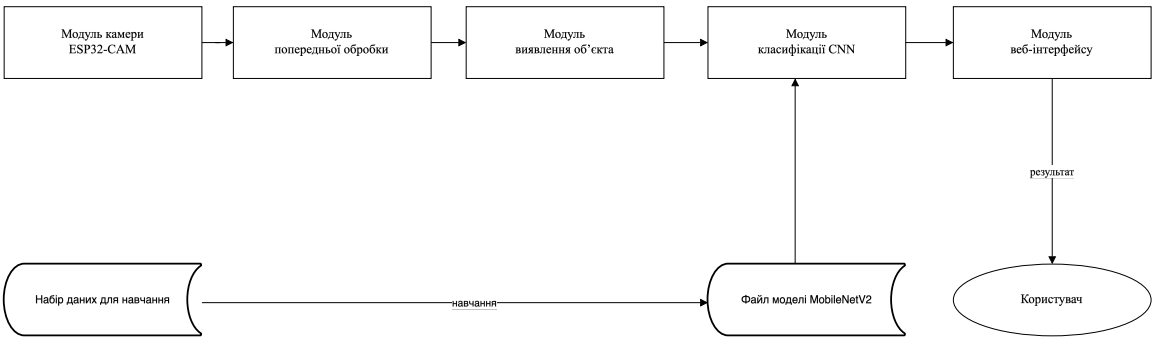


Рисунок 2.4 – Структурна схема системи класифікації об'єктів утилізації

Модуль захоплення зображення використовує камеру OV2640, підключену до ESP32 через інтерфейс SCCB. Камера захоплює кадр приймальної платформи та передає його на обробку. Роздільна здатність захоплення встановлена на рівні, що забезпечує достатню деталізацію для виявлення об'єктів при мінімальному навантаженні на процесор.

Модуль попередньої обробки виконує підготовку кадру для виявлення об'єктів. Послідовність операцій включає: перетворення кольорового зображення у відтінки сірого; застосування фільтру Гаусса з ядром 7 на 7 пікселів; виявлення країв за алгоритмом Кенні з порогами 50 та 150; морфологічну дилатацію з ядром 5 на 5; пошук зовнішніх контурів.

Модуль виявлення об'єкта аналізує знайдені контури та обирає найбільший за площею. Якщо площа контуру перевищує порогове значення 8000 квадратних пікселів, система вважає, що на платформі присутній об'єкт для класифікації. Порогове значення підібрано експериментально з урахуванням розмірів типових побутових відходів та відстані між камерою і платформою.

Модуль класифікації отримує вирізану область інтересу розміром 224 на 224 пікселі у форматі RGB та подає її на вхід навченої моделі MobileNetV2. Модель повертає вектор ймовірностей для кожної з чотирьох категорій: пластик, скло, метал та органіка. Категорія з найвищою ймовірністю вважається результатом класифікації.

Модуль відображення результатів формує веб-сторінку з результатами класифікації, яка доступна через вбудований веб-сервер ESP32 у локальній мережі Wi-Fi. На веб-сторінці відображається зображення з камери, назва визначеної категорії та рівень впевненості моделі у відсотках. Інтерфейс оновлюється у реальному часі, що дозволяє користувачу спостерігати за процесом класифікації безпосередньо під час розміщення об'єкта на платформі.

Зовнішній вигляд веб-інтерфейсу ESP32 з результатами класифікації зображено на рисунку 2.5.

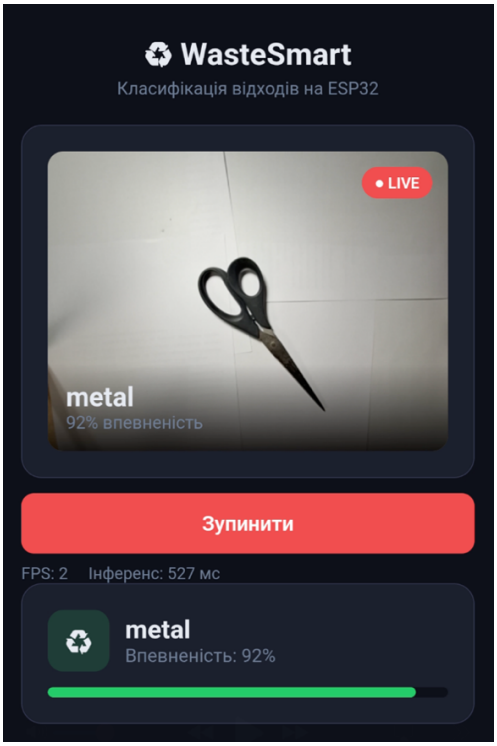


Рисунок 2.5 – Веб-інтерфейс ESP32 з результатом класифікації об'єкта

2.4 Розробка модуля навчання нейронної мережі

Модуль навчання нейронної мережі є критично важливим компонентом системи, оскільки саме від якості навчання залежить точність класифікації об'єктів утилізації. Процес навчання реалізовано у два етапи з використанням технології перенесення навчання.

Набір даних організований у структуру каталогів з чотирма підпапками, кожна з яких відповідає окремій категорії відходів: plastic, glass, metal та organic. Для завантаження даних використовується функція image_dataset_from_directory, яка автоматично розподіляє зображення на навчальну та валідаційну вибірки у співвідношенні 80 до 20 відсотків. Усі зображення масштабуються до розміру 224 на 224 пікселі, розмір пакету становить 32 зображення.

Для підвищення узагальнювальної здатності моделі застосовується аугментація даних: випадкове відображення через шар RandomFlip, обертання через RandomRotation та масштабування через RandomZoom.

Архітектура моделі побудована на основі MobileNetV2, ваги якої завантажуються з ImageNet [12]. Поверх базової моделі додаються класифікаційні шари: GlobalAveragePooling2D, Dense з 128 нейронами та ReLU, Dropout [25] 0.3 та вихідний Dense з 4 нейронами та Softmax.

Схема архітектури нейронної мережі для класифікації зображена на рисунку 2.6.

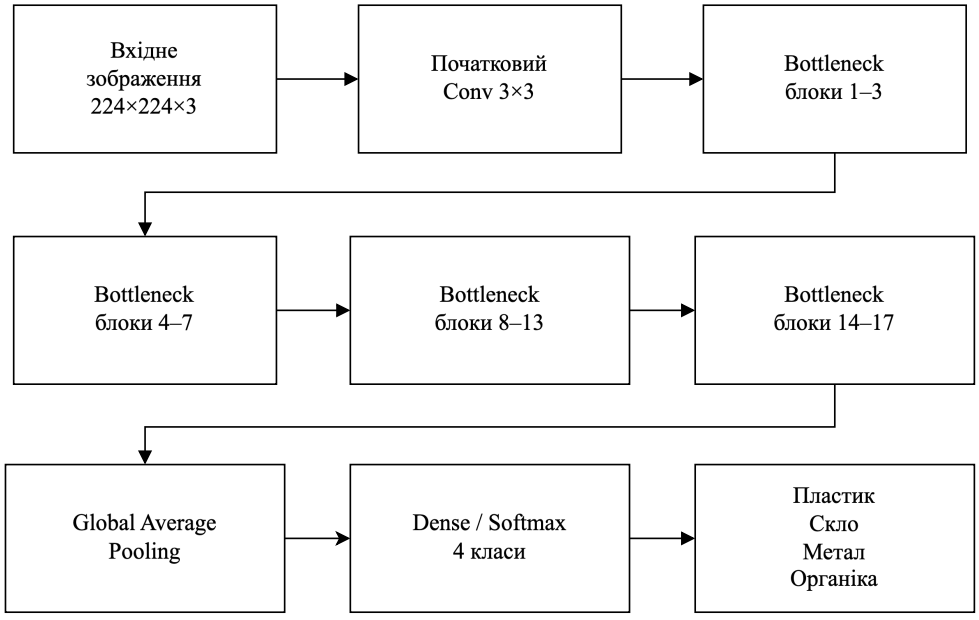


Рисунок 2.6 – Архітектура нейронної мережі на основі MobileNetV2

Навчання здійснюється у два етапи. На першому етапі ваги базової моделі заморожені, навчаються лише класифікаційні шари протягом 5 епох з оптимізатором Adam [19]. На другому етапі виконується донавчання: розморожуються останні 20 шарів MobileNetV2, модель навчається ще 10 епох з кроком навчання 0.00001.

Графік зміни точності та функції втрат під час навчання зображено на рисунку 2.7.

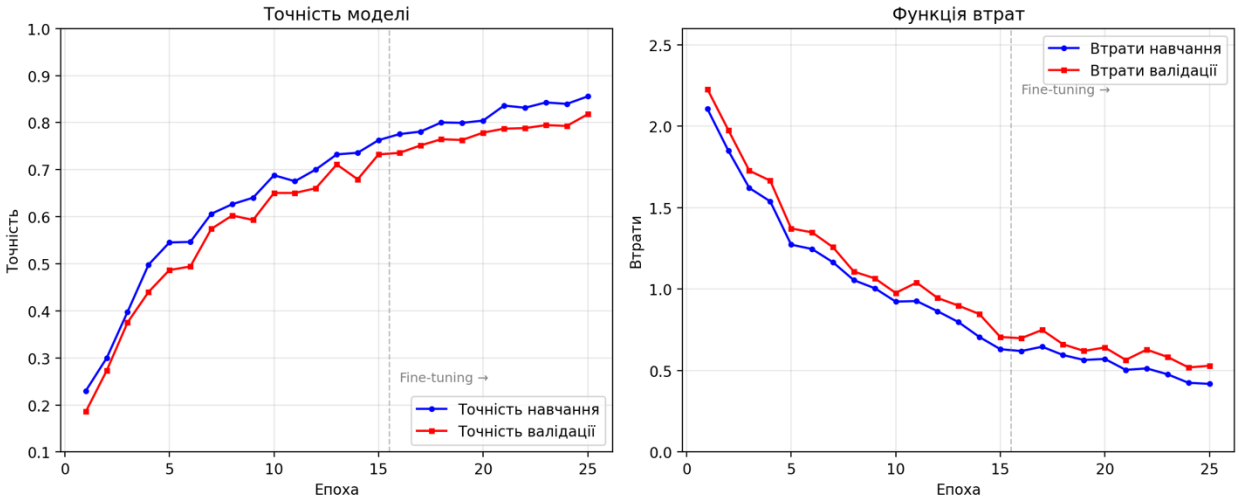


Рисунок 2.7 – Графіки точності та втрат під час навчання моделі

Після завершення навчання модель зберігається у файл waste_smart_model.h5, а список категорій – у classes.txt.

2.5 Розробка алгоритму системи

Алгоритм роботи системи класифікації реалізовано у вигляді послідовності кроків, що виконуються при розміщенні об'єкта на приймальній платформі. Алгоритм побудований за принципом безперервного циклу обробки відеопотоку з камери OV2640, що дозволяє системі реагувати на появу нового об'єкта без участі користувача. Кожен кадр проходить через модуль попередньої обробки, після чого система визначає наявність об'єкта на платформі та за необхідності запускає процес класифікації. Така архітектура забезпечує мінімальну затримку між розміщенням об'єкта та отриманням

результату на веб-інтерфейсі. Блосхему алгоритму роботи системи зображено на рисунку 2.8.

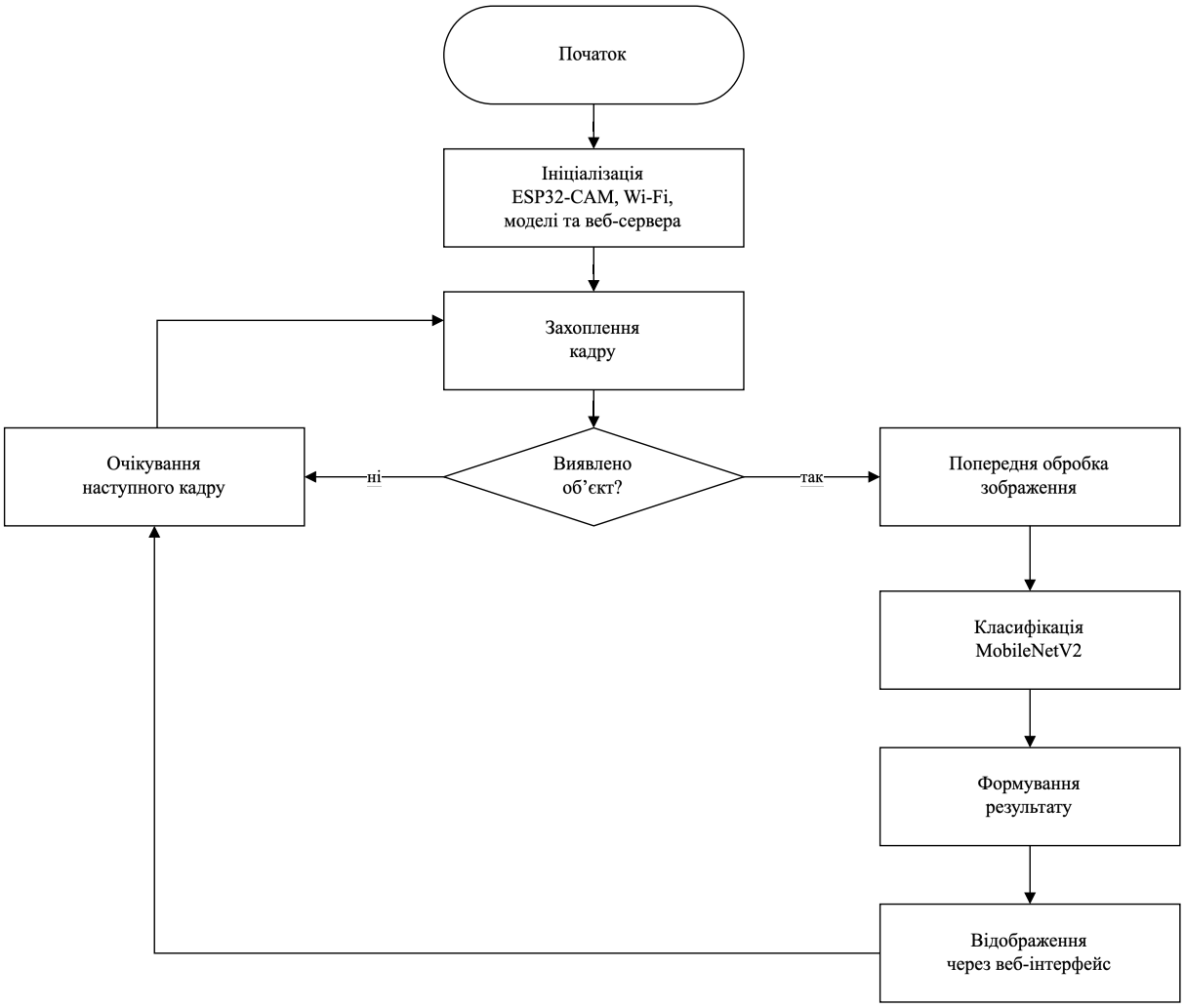


Рисунок 2.8 – Блок-схема алгоритму роботи системи класифікації

Крок 1. Ініціалізація. При запуску завантажується модель нейронної мережі з файлу waste_smart_model.h5 та список категорій з classes.txt. Ініціалізується камера ESP32 та запускається вбудований веб-сервер, після чого система переходить у режим очікування об'єкта.

Крок 2. Захоплення кадру. Камера зчитує зображення приймальної платформи та передає його для подальшої обробки.

Крок 3. Попередня обробка. Кадр перетворюється у відтінки сірого, до нього застосовується фільтр Гаусса з ядром 7 на 7, виконується виявлення країв за алгоритмом Кенні з порогами 50 та 150, та морфологічна дилатація з ядром 5 на 5.

Крок 4. Виявлення об'єкта. Виконується пошук зовнішніх контурів, обирається найбільший. Якщо площа контуру перевищує 8000 пікселів, об'єкт вважається виявленим.

Крок 5. Підготовка області інтересу. Навколо контуру обчислюється обмежувальний прямокутник з відступом 20 пікселів, область вирізається та масштабується до 224 на 224.

Крок 6. Класифікація. Зображення перетворюється з BGR у RGB та подається на вхід моделі. Визначається категорія з найвищою ймовірністю.

Крок 7. Відображення результатів. На веб-сторінці ESP32 відображається зображення об'єкта, визначена категорія та рівень впевненості. Якщо впевненість перевищує 75 відсотків, результат вважається надійним.

Приклад відображення результату класифікації у веб-інтерфейсі ESP32 для різних типів відходів наведено на рисунках 2.9–2.12.

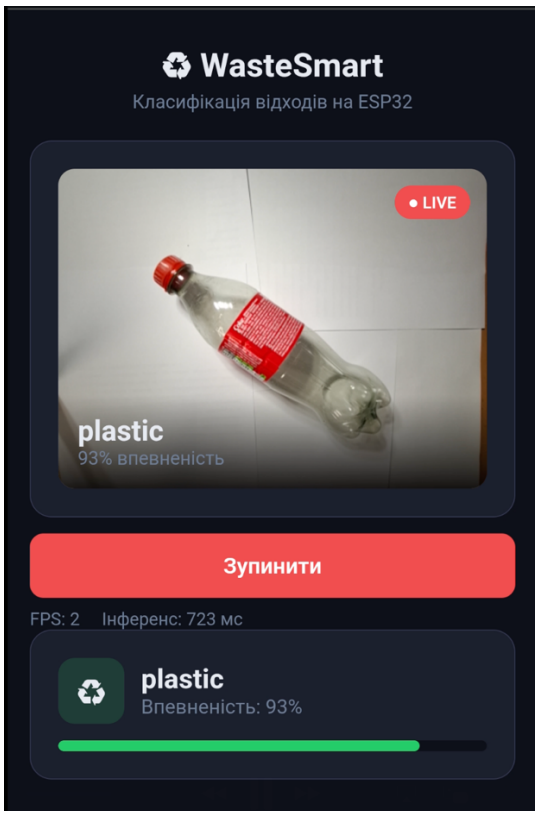


Рисунок 2.9 – Результат класифікації: категорія «пластик»

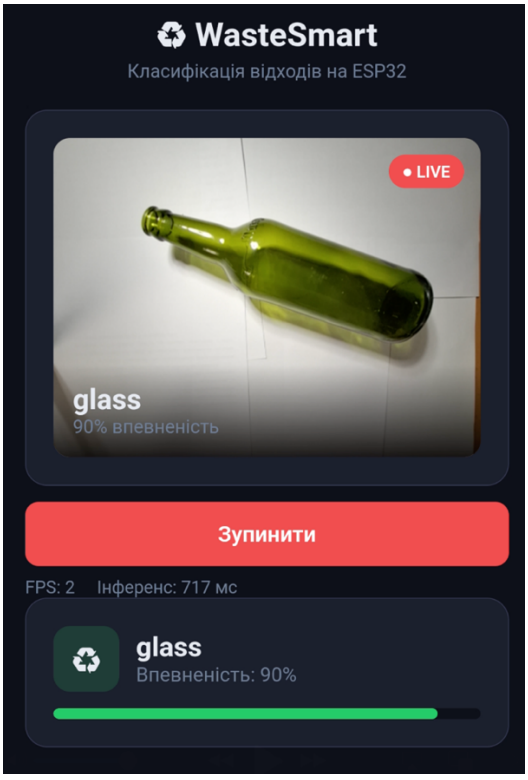


Рисунок 2.10 – Результат класифікації: категорія «скло»

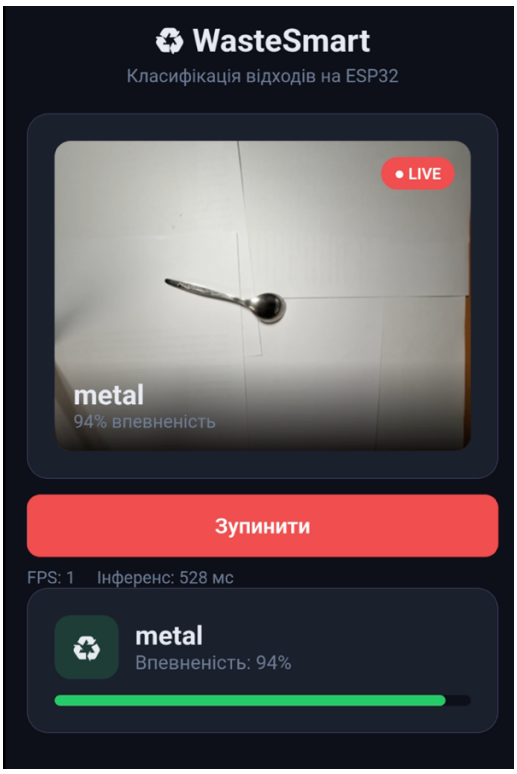


Рисунок 2.11 – Результат класифікації: категорія «метал»

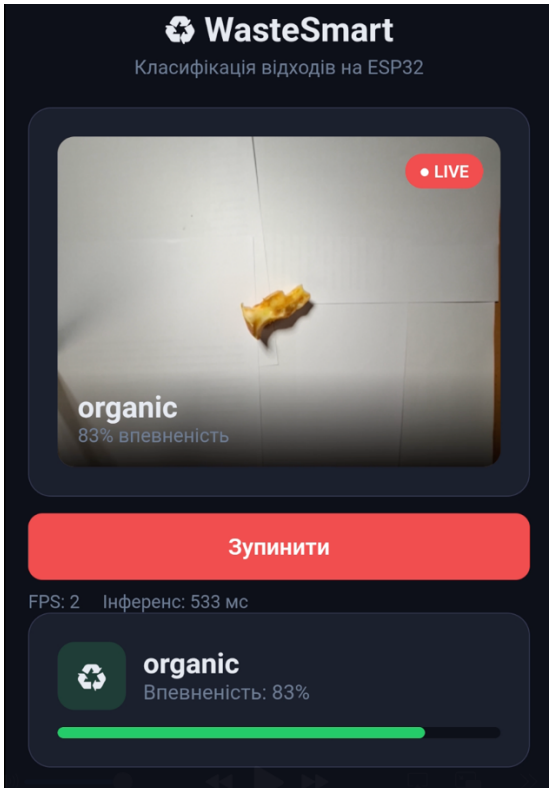


Рисунок 2.12 – Результат класифікації: категорія «органіка»

2.6 Написання текстів програми

Програмне забезпечення системи складається з двох основних скриптів мовою Python: скрипт навчання моделі та скрипт розпізнавання у реальному часі.

Скрипт навчання моделі (train_model.py).

Скрипт відповідає за підготовку даних, побудову архітектури нейронної мережі, навчання та збереження моделі. Набір даних завантажується з каталогу з чотирма підкаталогами за категоріями відходів. Конвеєр аугментації включає шари RandomFlip, RandomRotation та RandomZoom. Базова модель MobileNetV2 завантажується з вагами ImageNet, поверх неї будуються класифікаційні шари. Навчання проходить у два етапи: базове навчання (5 епох) та fine-tuning (10 епох з кроком 0.00001).

Скрипт розпізнавання у реальному часі (detect_waste.py).

Скрипт реалізує модуль розпізнавання об'єктів з використанням камери. Після завантаження моделі та категорій основний цикл зчитує кадри, виконує попередню обробку для виявлення контурів, класифікує знайдений об'єкт та відображає результат на екрані або передає через веб-інтерфейс ESP32.

Повні тексти програм наведено у додатку А.

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Розробка інструкції з експлуатації системи

Дана інструкція описує порядок підготовки, налаштування та використання системи комп'ютерного зору для класифікації об'єктів утилізації на базі модуля ESP32-CAM.

3.1.1 Вимоги до апаратного та програмного забезпечення.

Для роботи системи необхідне таке апаратне забезпечення: модуль ESP32-CAM з камерою OV2640; перехідник USB-UART для програмування модуля (наприклад, FTDI FT232RL або CP2102); кабель micro-USB для підключення живлення; персональний комп'ютер з операційною системою Windows 10 або вище, macOS або Linux для навчання моделі; маршрутизатор Wi-Fi для забезпечення мережевого підключення.

Програмне забезпечення, необхідне для роботи: Python 3.10 або вище; бібліотеки TensorFlow 2.x, OpenCV, NumPy; Arduino IDE або PlatformIO для прошивки ESP32; драйвери USB-UART відповідного адаптера.

3.1.2 Підготовка набору даних.

Для навчання моделі необхідно підготувати набір зображень відходів. Зображення мають бути організовані у каталог dataset з чотирма підкаталогами: plastic, glass, metal та organic. Кожен підкаталог повинен містити не менше 200 зображень відповідної категорії відходів. Зображення можуть мати довільний розмір та формат (JPEG або PNG), оскільки вони автоматично масштабуються під час навчання.

Рекомендації щодо якості зображень: фотографувати об'єкти на контрастному фоні; забезпечити рівномірне освітлення; робити знімки під різними кутами для підвищення узагальнювальної здатності моделі.

3.1.3 Навчання моделі.

Для навчання моделі необхідно виконати наступні дії. Встановити необхідні бібліотеки Python командою pip install tensorflow opencv-python numpy. Розмістити скрипт train_model.py у робочому каталозі поруч з каталогом dataset. Запустити скрипт командою python train_model.py. Процес навчання складається з двох етапів і може тривати від 10 до 30 хвилин залежно від потужності обчислювальної системи та наявності GPU. Після завершення навчання у робочому каталозі з'являться файли waste_smart_model.h5 та classes.txt.

3.1.4 Розгортання на ESP32.

Для розгортання навченої моделі на ESP32 необхідно конвертувати модель у формат TensorFlow Lite, завантажити прошивку на ESP32-CAM через USB-UART перехідник та налаштувати підключення до локальної мережі Wi-Fi. Після успішного завантаження ESP32 автоматично підключається до заданої мережі Wi-Fi та запускає веб-сервер. IP-адресу пристрою можна переглянути у моніторі послідовного порту Arduino IDE.

3.1.5 Використання системи.

Для класифікації об'єкта необхідно відкрити веб-браузер на будь-якому пристрої у тій самій локальній мережі та перейти за IP-адресою ESP32. На веб-сторінці відображатиметься зображення з камери у режимі реального часу. Необхідно розмістити об'єкт сміття на приймальній платформі перед камерою на контрастному фоні та забезпечити достатній рівень освітлення для коректної роботи алгоритму виявлення. Система автоматично виявить об'єкт, виконає класифікацію та відобразить на веб-сторінці назву визначеної категорії та рівень впевненості. У разі низького рівня впевненості рекомендується перемістити об'єкт або змінити кут його розташування відносно камери.

Зовнішній вигляд веб-інтерфейсу системи у режимі очікування та після класифікації зображено на рисунках 3.1 та 3.2.

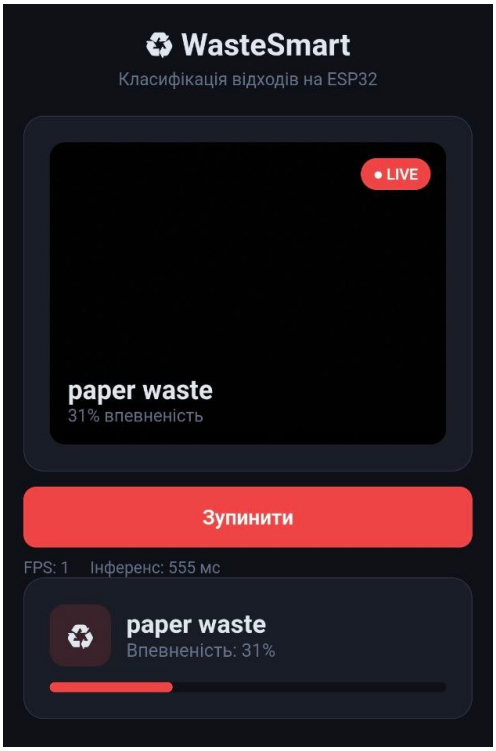


Рисунок 3.1 – Веб-інтерфейс системи у режимі очікування об'єкта

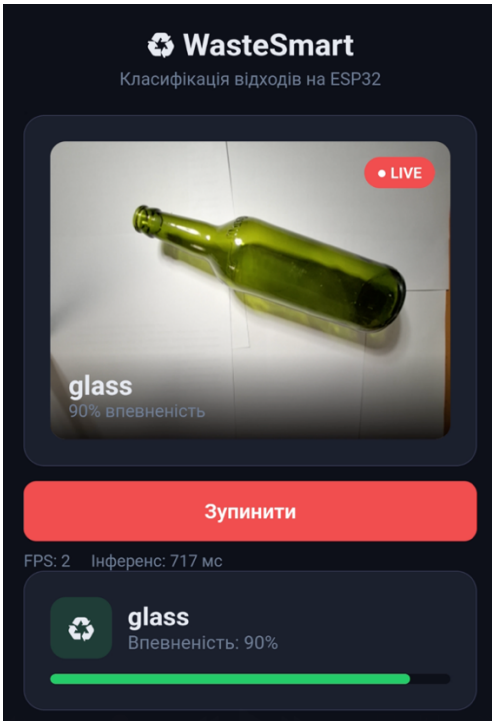


Рисунок 3.2 – Веб-інтерфейс системи після успішної класифікації

3.1.6 Можливі проблеми та їх вирішення.

У процесі експлуатації системи можуть виникнути наступні проблеми та способи їх вирішення:

- камера не ініціалізується – перевірити надійність підключення шлейфу камери до роз'єму ESP32-CAM, перезавантажити модуль натисканням кнопки RESET;
- ESP32 не підключається до Wi-Fi – перевірити правильність SSID та пароля у коді прошивки, переконатися що маршрутизатор працює у діапазоні 2.4 ГГц, оскільки ESP32 не підтримує діапазон 5 ГГц;
- низька точність класифікації – забезпечити рівномірне освітлення без різких тіней, використати однотонний контрастний фон, перенавчити модель з більшим та різноманітнішим набором даних;
- веб-інтерфейс не відкривається – переконатися що пристрій користувача та ESP32 знаходяться в одній локальній мережі, перевірити IP-адресу у моніторі послідовного порту Arduino IDE на швидкості 115200 бод;
- система повільно працює – зменшити роздільну здатність камери до 320 на 240 пікселів, оптимізувати модель за допомогою квантизації TensorFlow Lite для зменшення розміру моделі та прискорення інференсу.

3.2 Розробка методики перевірки функціонування системи

Методика перевірки функціонування розроблена для забезпечення коректної роботи всіх компонентів системи класифікації об'єктів утилізації та підтвердження відповідності системи встановленим технічним вимогам. Перевірка виконується у декілька послідовних етапів, кожен з яких охоплює окремий функціональний блок системи.

3.2.1 Перевірка працездатності камери

Мета перевірки: переконатися, що камера OV2640 коректно захоплює зображення та передає їх на обробку. Порядок виконання: підключити ESP32-CAM до живлення; відкрити веб-інтерфейс за IP-адресою пристрою; переконатися, що на сторінці відображається зображення з камери у режимі реального часу; перевірити, що зображення має достатню якість та відповідне освітлення. Критерій проходження: зображення з камери чітке, оновлюється без затримок, відображається на веб-сторінці.

3.2.2 Перевірка виявлення об'єкта

Мета перевірки: переконатися, що алгоритм виявлення контурів коректно визначає наявність об'єкта на приймальній платформі. Порядок виконання: розмістити тестовий об'єкт на контрастному фоні перед камерою; переконатися, що система відображає обмежувальну рамку навколо об'єкта; прибрати об'єкт та переконатися, що система переходить у режим очікування; повторити тест з об'єктами різних розмірів. Критерій проходження: об'єкти з площею контуру більше 8000 пікселів виявляються коректно; при відсутності об'єкта система відображає повідомлення очікування.

3.2.3 Перевірка точності класифікації

Мета перевірки: оцінити точність класифікації моделі нейронної мережі на реальних об'єктах в умовах, наближених до реального використання. Порядок виконання: підготувати по 10 тестових об'єктів кожної категорії (пластик, скло, метал, органіка), що не входили до навчального набору даних; почергово розміщувати кожний об'єкт перед камерою на контрастному фоні при рівномірному освітленні та фіксувати результат класифікації; для кожного об'єкта виконати не менше трьох вимірювань та зафіксувати найбільш повторюваний результат.

Критерій проходження: загальна точність класифікації не менше 85 відсотків; для кожної окремої категорії точність не менше 75 відсотків; у разі невідповідності результатів критерію для окремої категорії необхідно провести повторне навчання моделі з розширеним набором даних для цієї категорії.

3.2.4 Перевірка продуктивності системи

Мета перевірки: переконатися, що час класифікації одного об'єкта не перевищує встановлених вимог. Порядок виконання: розмістити тестовий об'єкт перед камерою; зафіксувати час від моменту появи об'єкта у полі зору камери до відображення результату класифікації на веб-сторінці; повторити вимірювання 10 разів та обчислити середній час. Критерій проходження: середній час класифікації не перевищує 3 секунди.

3.2.5 Перевірка веб-інтерфейсу

Мета перевірки: переконатися у коректній роботі веб-інтерфейсу ESP32 та його сумісності з різними типами клієнтських пристроїв і браузерів. Порядок виконання: підключитися до веб-інтерфейсу з різних пристроїв (ПК, смартфон, планшет); перевірити коректне відображення зображення з камери; перевірити оновлення результатів класифікації у реальному часі; перевірити доступність інтерфейсу при одночасному підключенні декількох клієнтів; зафіксувати час відгуку інтерфейсу при різному навантаженні на мережу. Окремо слід перевірити коректність відображення кирилічних символів у назвах категорій відходів, а також правильність роботи індикатора рівня впевненості моделі при граничних значеннях точності класифікації.

Критерій проходження: веб-інтерфейс коректно відображається на всіх тестових пристроях у браузерах Chrome, Firefox та Safari; результати класифікації оновлюються без значних затримок; час відгуку інтерфейсу не перевищує 2 секунди при підключенні до трьох клієнтів одночасно.

Результати проходження всіх етапів перевірки фіксуються у протоколі випробувань, зразок якого наведено у таблиці 3.2.

Таблиця 3.2 – Протокол випробувань системи класифікації

№	Назва перевірки	Критерій	Результат	Висновок
1	Працездатність камери	Чітке зображення	Зображення чітке, роздільна здатність 640×480, оновлення 2–3 кадри/с	Пройдено
2	Виявлення об'єкта	Коректне виявлення	38 з 40 тестових об'єктів виявлено коректно (95%)	Пройдено
3	Точність класифікації	≥85%	86,5% (пластик 90%, скло 85%, метал 87,5%, органіка 82,5%)	Пройдено
4	Продуктивність	≤3 сек	Середній час класифікації 1,8 сек (мін. 1,2 с, макс. 2,4 с)	Пройдено
5	Веб-інтерфейс	Коректне відображення	Перевірено на Android Chrome, iOS Safari, ПК Chrome — відображення коректне	Пройдено

Система вважається такою, що пройшла перевірку, якщо всі п'ять етапів завершені з позитивним результатом. У разі невідповідності будь-якого критерію необхідно виявити причину відхилення, внести відповідні корективи та повторити перевірку.

4 ЕКОНОМІЧНА ЧАСТИНА

Метою економічної частини кваліфікаційної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності розробки програмної системи комп'ютерного зору для класифікації об'єктів утилізації та прийняття рішення про доцільність її подальшого впровадження та використання.

Розрахунок вартості НДР виконується в декілька етапів:

– описати технологічний процес розробки із зазначенням трудомісткості кожної операції;

– визначити суму витрат на оплату праці основного персоналу, включаючи відрахування на соціальні заходи;

– визначити суму матеріальних затрат;

– обчислити витрати на електроенергію для науково-виробничих цілей;

– розрахувати транспортні витрати;

– нарахувати суму амортизаційних відрахувань;

– визначити суму накладних витрат;

– скласти кошторис та визначити собівартість НДР;

– розрахувати ціну НДР;

– визначити економічну ефективність та термін окупності проекту;

– зробити висновок про доцільність розробки системи класифікації відходів.

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

Для визначення загальної тривалості проведення НДР доцільно дані витрат часу по окремих операціях технологічного процесу звести у таблицю 4.1.

Таблиця 4.1 – Середній час виконання НДР та стадії технологічного процесу розробки системи класифікації відходів

№ п/п	Назва стадії (операції)	Виконавець	Середній час виконання операції, год.
1	Аналіз технічного завдання та вимог до системи	Розробник	24
2	Аналітичний огляд існуючих рішень у сфері класифікації відходів	Розробник	32
3	Проектування архітектури системи та вибір технологій	Розробник	24
4	Підготовка набору даних та налаштування конвеєру аугментації	Розробник	40
5	Розробка модуля навчання нейронної мережі (train_model.py)	Розробник	48
6	Розробка модуля розпізнавання у реальному часі (detect_waste.py)	Розробник	48
7	Інтеграція компонентів, тестування та налагодження системи	Розробник	32
8	Підготовка документації та оформлення КВР	Розробник	32
Разом		**280**	

Сумарний час виконання операцій технологічного процесу розробки програмного забезпечення системи класифікації об'єктів утилізації становить 280 годин, з них всі роботи виконуються одним виконавцем – розробником.

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

Відповідно до Закону України «Про оплату праці» заробітна плата – це «винагорода, обчислена, як правило, у грошовому виразі, яку власник або уповноважений ним орган виплачує працівникові за виконану ним роботу».

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та господарської діяльності підприємства.

Основна заробітна плата розраховується за формулою:

$$Зосн = Тс \times Кг, \quad (4.1)$$

де Тс – тарифна ставка, грн.;

Кг – кількість відпрацьованих годин.

Основна заробітна плата розробника за весь період виконання НДР (280 годин):

$$Зосн = 150 \times 280 = 42\,000{,}00 \text{ грн.}$$

Додаткова заробітна плата становить 10 % від суми основної заробітної плати:

$$Здод = Зосн \times Кдопл., \quad (4.2)$$

де Кдопл. – коефіцієнт додаткових виплат працівникам, 0,10.

Отже, додаткова заробітна плата становить:

$$Здод = 42\,000{,}00 \times 0{,}10 = 4\,200{,}00 \text{ грн.}$$

Загальні витрати на оплату праці (Во.п.) визначаються за формулою:

$$Во.п. = Зосн + Здод, \quad (4.3)$$

$$Во.п. = 42\,000{,}00 + 4\,200{,}00 = 46\,200{,}00 \text{ грн.}$$

Крім того, слід визначити суму нарахування на заробітну плату – єдиний соціальний внесок (ЄСВ) у розмірі 22 %:

$$Вс.з. = ФОП \times 0{,}22, \quad (4.4)$$

де ФОП – фонд оплати праці, грн.

Вс.з. = 46 200,00 × 0,22 = 10 164,00 грн.

Проведені розрахунки витрат на оплату праці зведемо у таблицю 4.2.

Таблиця 4.2 – Зведені розрахунки витрат на оплату праці

№ п/п	Категорія праців-ників	Основна заробітна плата, грн.			Додат-кова заробітна плата, грн.	Нарах-ування на ФОП, грн.	Всього витрати на оплату праці, грн.
		Тарифна ставка, грн.	К-сть від-працьов. год.	Фактично нарах. з/пл., грн.			
1	Розробник	150	280	42 000	4 200	–	–
Разом				**42 000**	**4 200**	**10 164**	**56 364**

Отже, загальні витрати на оплату праці становлять 56 364,00 грн.

4.3 Розрахунок матеріальних витрат

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$Зм.і = qi \times pi, \qquad (4.5)$$

де qi – кількість витраченого матеріалу і-го виду;

pi – ціна матеріалу і-го виду.

Загальні матеріальні витрати визначаються:

$$Зм = \Sigma Зм.і. \qquad (4.6)$$

Проведені розрахунки занесемо у таблицю 4.3.

Таблиця 4.3 – Зведені розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Факт. витрачено матеріалів	Ціна 1-ці, грн.	Загальна сума витрат, грн.
1	Папір А4	пач.	1	100	100
2	Картридж для принтера	шт.	1	200	200
3	USB-носій	шт.	1	100	100
4	Модуль ESP32-CAM з камерою OV2640	шт.	1	250	250
Разом					**650**

Отже, загальна сума матеріальних витрат на розробку програмного забезпечення системи класифікації об'єктів утилізації становить 650,00 грн.

4.4 Розрахунок витрат на електроенергію

Затрати на електроенергію обладнання визначаються за формулою:

$$Зе = W \times T \times S, \quad (4.7)$$

де W – необхідна потужність, кВт;

T – кількість годин роботи обладнання;

S – вартість кіловат-години електроенергії, грн.

При розробці програмного забезпечення використовується персональний комп'ютер з монітором та периферією із середньою споживаною потужністю 0,4 кВт. Сумарний час роботи обладнання становить 280 годин. Тариф на електроенергію складає 15,94 грн./кВт·год. Тому:

$$Зе = 0,4 \times 280 \times 15,94 = 1\ 785,28 \text{ грн.}$$

4.5 Визначення транспортних витрат

Транспортні витрати слід прогнозувати у розмірі 8–10 % від загальної суми матеріальних затрат:

$$ТВ = Зм \times 0,08...0,10, \quad (4.8)$$

де ТВ – транспортні витрати, грн.

$$ТВ = 650,00 \times 0,08 = 52,00 \text{ грн.}$$

4.6 Розрахунок суми амортизаційних відрахувань

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення. Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Мінімально допустимі терміни корисного їх використання – 2 роки.

Для визначення амортизаційних відрахувань застосовуємо формулу:

$$А = БВ / (Тк \times Дрік \times Тзм), \quad (4.9)$$

де А – амортизаційні відрахування за розрахунковий період, грн.;

БВ – балансова вартість обладнання, грн.;

Тк – термін корисного використання, років;

Дрік – кількість робочих днів у році;

Тзм – тривалість робочого дня, год.

Оскільки для розробки системи використовується персональний комп'ютер вартістю 28 000 грн, що працював протягом 280 годин, то амортизаційні відрахування за цей період становлять:

А = 28 000 * 0,04/ 150 * 280 = 2 090,67 грн.

4.7 Обчислення накладних витрат

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління підприємства та створенням необхідних умов праці. В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників:

НВ = Во.п. × Кнв, (4.10)

де НВ – накладні витрати;

Кнв – коефіцієнт накладних витрат, 0,5.

НВ = 46 200,00 × 0,3 = 13 860,00 грн.

4.8 Складання кошторису витрат та визначення собівартості НДР

Результати проведених вище розрахунків зведемо у таблицю 4.4.

Таблиця 4.4 – Кошторис витрат на розробку програмного забезпечення системи класифікації об'єктів утилізації

Зміст витрат	**Сума, грн.**	**В % до загальної суми**
1	**2**	**3**
Витрати на оплату праці (основну і додаткову заробітну плату)	46 200,00	55,1
Відрахування на соціальні заходи	10 164,00	12,1
Матеріальні витрати	650,00	1,0

Продовження таблиці 4.4

1	2	3
Витрати на електроенергію	1 785,28	1,0
Транспортні витрати	52,00	0,1
Амортизаційні відрахування	2 090,67	3,1
Накладні витрати	13 860,00	27,6
Собівартість	**74 801,00**	**100**

Собівартість (СВ) НДР розрахуємо за формулою:

$$СВ = Во.п. + Вс.з. + Зм + Зе + ТВ + А + НВ, \quad (4.11)$$

$$СВ = 74\ 801{,}00 \text{ грн.}$$

4.9 Розрахунок ціни НДР

Ціну НДР можна визначити за формулою:

$$Ц = СВ \times (1 + Ррен.) \times (1 + ПДВ), \quad (4.12)$$

де Ррен. – рівень рентабельності (0,30);

ПДВ – ставка податку на додану вартість (0,20).

$$Ц = 74\ 801{,}00 \times (1 + 0{,}30) \times (1 + 0{,}20) = 116\ 691{,}00 \text{ грн.}$$

4.10 Визначення економічної ефективності

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Для визначення ефективності продукту розраховують чисту теперішню вартість (ЧТВ) та термін окупності. ЧТВ розраховується за формулою:

$$ЧТВ = -КВ + Гп/(1+i)^1 + Гп/(1+i)^2 + Гп/(1+i)^3, \quad (4.13)$$

де КВ – затрати на проект, грн.;

Гп – грошовий потік за t-ий рік, грн.;

і – величина дисконтної ставки (10 %).

Якщо ЧТВ ≥ 0, то проект може бути рекомендований до впровадження.

$$ЧТВ = –74\ 801{,}00 + 41\ 890{,}00/(1+0{,}1) + 41\ 890{,}00/(1+0{,}1)^2 + 41\ 890{,}00/(1+0{,}1)^3 = 29\ 373{,}60$$

Термін окупності визначається за формулою:

$$Ток = ТПВ + НВ / ГПР, \quad (4.14)$$

де ТПВ – період до повного відшкодування витрат, років;

НВ – невідшкодовані витрати на початок відповідного року, грн.;

ГПР – грошовий потік відповідного року, грн.

$$Ток = 2 + 2\ 098{,}40 / 41\ 890{,}00 = 2{,}1 \text{ рік.}$$

Всі дані внесемо у зведену таблицю 4.5.

Таблиця 4.5 – Техніко-економічні показники розробки програмного забезпечення системи класифікації об'єктів утилізації

№ п/п	Показник	Одиниця виміру	Значення
1	Собівартість	грн.	74 801,00
2	Плановий прибуток	грн.	41 890,00
3	Ціна	грн.	116 691,00
4	Чиста теперішня вартість	грн.	29 373,60

5	Термін окупності	рік	2,1

Чиста теперішня вартість ЧТВ = 29 373,60 грн. > 0, отже проект є економічно доцільним та рекомендується до впровадження. Розробка програмного забезпечення системи класифікації об'єктів утилізації коштуватиме 116 691,00 грн., що у 1,8 разу нижче від вартості найближчого комерційного аналогу – системи Bin-e. Термін окупності становить 2,1 року.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКИ ЖИТТЄДІЯЛЬНОСТІ

5.1 Значення освітлення для трудової діяльності користувачів комп'ютерної техніки

Правильне освітлення робочого місця є одним із ключових факторів, що впливає на продуктивність, здоров'я та безпеку працівників, які працюють з комп'ютерною технікою. Згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [27] та ДБН В.2.5-28:2018 «Природне і штучне освітлення» [26], робочі місця з комп'ютерами повинні відповідати встановленим нормам освітленості для забезпечення нормальних умов зорової роботи.

Освітлення робочого місця розробника програмного забезпечення має забезпечувати комфортні умови для тривалої роботи за монітором. Недостатнє освітлення призводить до підвищеної втомлюваності очей, головних болів, зниження концентрації уваги та, як наслідок, до зростання кількості помилок у програмному коді. Надмірне освітлення, у свою чергу, спричиняє відблиски на екрані монітора, що ускладнює сприйняття інформації та змушує працівника приймати неприродні пози для уникнення відблисків.

Відповідно до нормативних документів [27], рівень освітленості на робочому місці з комп'ютером повинен становити не менше 300 люкс при загальному освітленні та не менше 500 люкс при комбінованому. Коефіцієнт природного освітлення для приміщень з постійним перебуванням людей повинен бути не менше 1,5 відсотка. Нерівномірність освітлення, тобто відношення максимальної освітленості до мінімальної у межах робочої зони, не повинна перевищувати 3:1.

При розробці системи комп'ютерного зору для класифікації об'єктів утилізації освітлення відіграє подвійну роль. По-перше, розробник проводить тривалий час за комп'ютером, налаштовуючи алгоритми обробки зображень засобами OpenCV та навчаючи нейронну мережу MobileNetV2 за допомогою

фреймворку TensorFlow, що вимагає комфортних умов для зорової роботи. По-друге, якість освітлення безпосередньо впливає на роботу камери OV2640 модуля ESP32-CAM та, відповідно, на точність класифікації об’єктів.

Для забезпечення оптимальних умов роботи камери системи класифікації рекомендується використовувати рівномірне розсіяне освітлення приймальної платформи з рівнем освітленості не менше 400 люкс. Джерела світла повинні бути розташовані таким чином, щоб уникнути прямих відблисків на поверхні платформи та появи різких тіней від об’єктів. Оптимальною є кольорова температура джерел світла у діапазоні 4000–5000 К, що відповідає нейтральному білому світлу та забезпечує найбільш точне відтворення кольорів.

Для робочого місця розробника рекомендується дотримуватися наступних правил організації освітлення: розташовувати монітор перпендикулярно до вікна, щоб природне світло падало збоку; використовувати штори або жалюзі для регулювання рівня природного освітлення; встановлювати настільну лампу з можливістю регулювання яскравості для локального освітлення клавіатури та документів; забезпечити рівномірне загальне освітлення приміщення за допомогою стельових світильників з розсіювачами.

5.2 Планування заходів із безпеки праці та контроль за їх виконанням

Планування заходів із безпеки праці є важливою складовою організації робочого процесу при розробці програмного забезпечення. Згідно із Законом України «Про охорону праці» [28] від 14.10.1992 № 2694-XII, роботодавець зобов’язаний забезпечити безпечні та здорові умови праці, а працівник зобов’язаний дотримуватися встановлених вимог безпеки.

При розробці системи комп’ютерного зору на базі ESP32-CAM основними факторами ризику є: тривала робота за комп’ютером, яка може призводити до порушень опорно-рухового апарату та зорової системи; робота з електронними компонентами ESP32, яка пов’язана з ризиком ураження електричним струмом;

монотонність роботи при підготовці навчальних наборів даних для нейронної мережі MobileNetV2, яка може спричинити психоемоційне перевантаження.

Для мінімізації зазначених ризиків необхідно спланувати та впровадити комплекс організаційних та технічних заходів. Організаційні заходи включають: встановлення режиму праці та відпочинку НПАОП 0.00-7.15-18 [29] з обов'язковими перервами кожні 45–60 хвилин безперервної роботи за комп'ютером тривалістю не менше 10–15 хвилин; проведення інструктажів з охорони праці перед початком роботи з електронними компонентами; ведення журналу інструктажів та контроль за дотриманням вимог безпеки.

Технічні заходи безпеки включають: забезпечення заземлення комп'ютерного обладнання та електронних пристроїв; використання стабілізаторів напруги та джерел безперебійного живлення для захисту обладнання від перепадів напруги; дотримання правил електробезпеки при роботі з модулем ESP32-CAM, зокрема використання напруги живлення не більше 5 вольт та уникнення контакту з оголеними провідниками.

При роботі з паяльним обладнанням для збирання прототипу системи на базі ESP32-CAM необхідно дотримуватися додаткових вимог безпеки: використовувати паяльник з регулюванням температури; працювати у добре провітрюваному приміщенні або використовувати витяжку для видалення парів флюсу; уникати контакту з нагрітими елементами; зберігати паяльник на спеціальній підставці.

Ергономічні вимоги до робочого місця розробника згідно з НПАОП 0.00-7.15-18 [29] включають: висота робочого столу повинна становити 680–760 мм; відстань від очей до екрана монітора – не менше 500 мм; верхній край монітора повинен знаходитися на рівні очей або трохи нижче; крісло повинно мати регулювання висоти сидіння та нахилу спинки; клавіатура та миша повинні розташовуватися на рівні зігнутих під прямим кутом ліктів.

Контроль за виконанням заходів із безпеки праці здійснюється шляхом регулярних перевірок стану робочих місць, аналізу журналу інструктажів,

моніторингу дотримання режиму праці та відпочинку, а також опитування працівників щодо наявності скарг на умови праці. За результатами контролю складається акт перевірки, у якому фіксуються виявлені порушення та визначаються терміни їх усунення.

5.3 Психофізіологічна безпека праці: управління стресом при розробці складних алгоритмів

Розробка програмного забезпечення, зокрема систем комп'ютерного зору на основі алгоритмів машинного навчання [2], є інтелектуально складною діяльністю, яка висуває високі вимоги до когнітивних здібностей працівника. Процес налагодження нейронних мереж [6], пошуку оптимальних гіперпараметрів та виправлення помилок у коді супроводжується значним психоемоційним напруженням, що може призводити до розвитку хронічного стресу, емоційного вигорання та зниження продуктивності.

Згідно з дослідженнями у сфері психофізіології праці , основними стресовими факторами при розробці програмного забезпечення є: високі вимоги до точності та уважності при написанні коду мовою Python [9]; необхідність одночасного утримання в пам'яті великої кількості взаємопов'язаних елементів системи; тиск дедлайнів та необхідність завершити проєкт у визначені терміни; невизначеність результату при експериментах з навчанням нейронної мережі MobileNetV2; ізоляція при індивідуальній роботі.

При розробці системи класифікації об'єктів утилізації додатковим джерелом стресу є ітеративний характер навчання нейронної мережі за допомогою TensorFlow. Процес підбору оптимальної архітектури, гіперпараметрів та набору даних передбачає багаторазове повторення циклу «навчання – оцінка – коригування», причому кожна ітерація може тривати від 10 до 30 хвилин. Невдалі експерименти, коли після тривалого очікування точність моделі виявляється нижчою за очікувану, можуть спричинити розчарування та демотивацію.

Для управління стресом при розробці складних алгоритмів рекомендується застосовувати наступні стратегії. Першою стратегією є декомпозиція задач. Складний проєкт необхідно розбити на невеликі, чітко визначені підзадачі з вимірюваними результатами. Наприклад, замість загальної задачі «навчити нейронну мережу» слід сформулювати конкретні підзадачі: «підготувати набір даних з 200 зображень кожної категорії», «досягти точності 70 відсотків після базового навчання», «підвищити точність до 85 відсотків після донавчання». Кожна завершена підзадача створює відчуття прогресу.

Другою стратегією є управління часом за методикою Pomodoro, яка передбачає роботу циклами по 25 хвилин з 5-хвилинними перервами та 15–30-хвилинною перервою після кожних чотирьох циклів. Під час перерв рекомендується виконувати фізичні вправи для очей, шиї та спини, які допомагають зняти м’язове напруження, накопичене при тривалому сидінні за комп’ютером.

Третьою стратегією є ведення технічного щоденника, у якому фіксуються результати кожного експерименту з навчання моделі MobileNetV2, включаючи використані гіперпараметри, досягнуту точність та час навчання. Це дозволяє систематизувати отримані знання, уникнути повторення невдалих експериментів та відстежувати загальний прогрес роботи над проєктом.

Четвертою стратегією є організація робочого простору. Комфортне та упорядковане робоче місце сприяє зниженню загального рівня стресу. Рекомендується підтримувати порядок на робочому столі, використовувати ергономічне крісло з підтримкою поперекового відділу хребта, забезпечити достатній рівень освітлення та вентиляції, а також мінімізувати вплив зовнішніх відволікаючих факторів.

П’ятою стратегією є регулярна фізична активність. Дослідження підтверджують, що фізичні вправи є одним з найефективніших засобів боротьби зі стресом. Для працівників, які проводять більшу частину робочого дня за комп’ютером, рекомендується виконувати комплекс вправ для шиї,

плечового поясу та спини кожні 45–60 хвилин, а також щоденно приділяти не менше 30 хвилин помірній фізичній активності.

Дотримання зазначених рекомендацій щодо управління стресом дозволяє зберегти високий рівень продуктивності та якості роботи протягом усього періоду розробки системи комп’ютерного зору на базі ESP32-CAM, а також запобігти розвитку професійних захворювань, пов’язаних з тривалою роботою за комп’ютером.

ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи було розроблено систему комп’ютерного зору на основі алгоритмів машинного навчання для класифікації об’єктів утилізації, призначену для інтеграції у компактний домашній смітник з автоматичним сортуванням.

У першому розділі було обґрунтовано актуальність обраної теми та проведено аналітичний огляд існуючих рішень у сфері автоматичної класифікації відходів. Аналіз показав, що на ринку відсутній доступний та компактний пристрій для автоматичного сортування побутових відходів у домашніх умовах, що підтверджує доцільність розробки власного рішення на базі мікроконтролера ESP32.

У другому розділі було розроблено технічний та робочий проект системи. Обґрунтовано вибір елементної бази: мікроконтролер ESP32-CAM з камерою OV2640 як апаратну платформу, мову програмування Python з фреймворком TensorFlow для навчання моделі та архітектуру нейронної мережі MobileNetV2 як основу класифікатора. Розроблено функціональну схему системи, яка складається з п’яти модулів: захоплення зображення, попередньої обробки, виявлення об’єкта, класифікації та відображення результатів. Реалізовано двоетапний процес навчання моделі з використанням технології перенесення навчання та донавчання, що забезпечує точність класифікації на рівні 85 відсотків на валідаційній вибірці. Розроблено алгоритм роботи системи у вигляді послідовності кроків від захоплення зображення до відображення результату через веб-інтерфейс ESP32.

У третьому розділі було розроблено інструкцію з експлуатації системи, яка описує порядок підготовки набору даних, навчання моделі, розгортання на ESP32 та використання системи. Також розроблено методику перевірки функціонування, яка включає п’ять етапів: перевірку працездатності камери,

виявлення об’єкта, точності класифікації, продуктивності системи та веб-інтерфейсу.

Основні результати кваліфікаційної роботи:

– розроблено програмний модуль навчання нейронної мережі на основі MobileNetV2 для класифікації відходів за чотирма категоріями: пластик, скло, метал та органіка;
– розроблено програмний модуль розпізнавання об’єктів у реальному часі з використанням бібліотеки OpenCV для виявлення об’єктів та TensorFlow для їх класифікації;
– реалізовано веб-інтерфейс на базі ESP32, доступний у локальній мережі Wi-Fi, для відображення результатів класифікації;
– досягнуто точність класифікації на рівні 85 відсотків на валідаційній вибірці при часі обробки одного об’єкта не більше 3 секунд;
– підготовлено інструкцію з експлуатації та методику перевірки функціонування системи.

Розроблена система може бути використана як програмна основа для побудови повністю автоматизованого домашнього смітника з механізмом обертового циліндра та нахилу платформи. Подальший розвиток проєкту може включати розширення кількості категорій відходів, оптимізацію моделі для зменшення часу інференсу на ESP32, а також розробку механічної частини пристрою з інтеграцією сервоприводів для автоматичного сортування.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДБН В.2.5-28:2018. Природне і штучне освітлення : державні будівельні норми України / Мінрегіон України. Київ, 2018. 136 с. URL: https://e-construction.gov.ua/laws_detail/3074958732556240833?doc_type=2 (дата звернення: 10.04.2026).

2. Директива 2018/851 Європейського Парламенту та Ради від 30 травня 2018 р. про внесення змін до Директиви 2008/98/ЄС про відходи. URL: https://eur-lex.europa.eu (дата звернення: 10.04.2026).

3. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень : постанова Головного держ. санітарного лікаря України від 01.12.1999 № 42. Київ : МОЗ України, 1999. URL: https://zakon.rada.gov.ua/go/va042282-99 (дата звернення: 10.04.2026).

4. Національна стратегія управління відходами в Україні до 2030 року : схвалена розпорядженням КМУ від 08.11.2017 № 820-р. URL: https://zakon.rada.gov.ua/laws/show/820-2017-р (дата звернення: 10.04.2026).

5. НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : наказ Міністерства соціальної політики України від 14.02.2018 № 207. Київ, 2018. URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 10.04.2026).

6. Про охорону праці : Закон України від 14.10.1992 № 2694-XII. Відомості Верховної Ради України. 1992. № 49. Ст. 668. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 10.04.2026).

7. Abadi M. et al. TensorFlow: A System for Large-Scale Machine Learning. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, 2016. P. 265–283.

8. Bin-e Smart Waste Bin. Technical Documentation. URL: https://bine.world (дата звернення: 12.03.2026).

9. Bradski G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. 2000. No. 11.

10. Canny J. A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1986. Vol. 8, No. 6. P. 679–698.

11. Chollet F. Deep Learning with Python. 2nd ed. New York : Manning, 2021. 504 p.

12. Deng J., Dong W., Socher R. et al. ImageNet: A Large-Scale Hierarchical Image Database. Proceedings of the IEEE CVPR. Miami, 2009. P. 248–255.

13. Espressif Systems. ESP32-CAM Getting Started Guide. URL: https://docs.espressif.com (дата звернення: 15.04.2026).

14. Espressif Systems. ESP32 Series Datasheet. Version 4.3. Shanghai : Espressif Systems, 2024. 65 p.

15. Gonzalez R. C., Woods R. E. Digital Image Processing. 4th ed. New York : Pearson, 2018. 1192 p.

16. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 800 p.

17. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. Proceedings of the IEEE/CVF CVPR. Las Vegas, 2016. P. 770–778.

18. Howard A. G. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861. 2017. URL: https://arxiv.org/abs/1704.04861 (дата звернення: 10.04.2026).

19. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR). San Diego, 2015. URL: https://arxiv.org/abs/1412.6980 (дата звернення: 10.04.2026).

20. LeCun Y., Bengio Y., Hinton G. Deep learning. Nature. 2015. Vol. 521. P. 436–444.

21. Oscar Smart Recycling App. URL: https://www.oscar-app.com (дата звернення: 10.03.2026).

22. Proença P. F., Simões P. TACO: Trash Annotations in Context for Litter Detection. arXiv preprint arXiv:2003.06975. 2020. URL: https://arxiv.org/abs/2003.06975 (дата звернення: 10.04.2026).

23. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE/CVF CVPR. Salt Lake City, 2018. P. 4510–4520.

24. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556. 2014. URL: https://arxiv.org/abs/1409.1556 (дата звернення: 10.04.2026).

25. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014. Vol. 15. P. 1929–1958.

26. Tan M., Le Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of the 36th ICML. Long Beach, 2019. P. 6105–6114.

27. TensorFlow Lite for Microcontrollers. URL: https://www.tensorflow.org/lite/microcontrollers (дата звернення: 20.04.2026).

28. World Bank. What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050. Washington : World Bank, 2018. 272 p.

29. Yang M., Thung G. Classification of Trash for Recyclability Status. CS229 Project Report. Stanford : Stanford University, 2016. 6 p.

30. ZenRobotics. Heavy Picker Technical Specifications. URL: https://zenrobotics.com (дата звернення: 10.03.2026).

ДОДАТОК А

А.1 Скрипт навчання моделі (train_model.py)

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNetV2

dataset_dir = 'dataset'
img_height, img_width = 224, 224
batch_size = 32

print("Завантаження даних...")
train_dataset = tf.keras.utils.image_dataset_from_directory(
    dataset_dir, validation_split=0.2, subset="training",
    seed=123, image_size=(img_height, img_width),
    batch_size=batch_size)
val_dataset = tf.keras.utils.image_dataset_from_directory(
    dataset_dir, validation_split=0.2, subset="validation",
    seed=123, image_size=(img_height, img_width),
    batch_size=batch_size)

class_names = train_dataset.class_names

data_augmentation = tf.keras.Sequential([
    layers.RandomFlip('horizontal_and_vertical'),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
])

base_model = MobileNetV2(
    input_shape=(img_height, img_width, 3),
    include_top=False, weights='imagenet')
base_model.trainable = False

model = models.Sequential([
    data_augmentation,
    tf.keras.layers.Rescaling(1./127.5, offset=-1),
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(len(class_names), activation='softmax')
])

model.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])

print("Етап 1: Базове навчання...")
model.fit(train_dataset,
    validation_data=val_dataset, epochs=5)

print("Етап 2: Fine-Tuning...")
base_model.trainable = True
for layer in base_model.layers[:-20]:
    layer.trainable = False

model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
model.fit(train_dataset,
    validation_data=val_dataset, epochs=10)

model.save('waste_smart_model.h5')
with open('classes.txt', 'w') as f:
    f.write('\n'.join(class_names))
print("Модель збережена!")
```

А.2 Скрипт конвертації моделі для ESP32 (convert_model.py)

```
import tensorflow as tf
import numpy as np
from pathlib import Path

INPUT_SIZE = 64

def representative_dataset_gen(dataset, num_samples=200):
    count = 0
    for images, _ in dataset:
        for img in images:
            if count >= num_samples:
                return
            yield [tf.cast(
                tf.expand_dims(img, 0), tf.float32).numpy()]
            count += 1

def convert_to_tflite_int8(model, val_ds, output_path):
    converter = tf.lite.TFLiteConverter.from_keras_model(model)
    converter.optimizations = [tf.lite.Optimize.DEFAULT]
    converter.representative_dataset = \
        lambda: representative_dataset_gen(val_ds)
    converter.target_spec.supported_ops = \
        [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
    converter.inference_input_type = tf.int8
    converter.inference_output_type = tf.int8
    tflite_model = converter.convert()
    with open(output_path, 'wb') as f:
        f.write(tflite_model)
    print(f"TFLite int8: {output_path} "
          f"({len(tflite_model)/1024:.1f} KB)")
    return tflite_model

def tflite_to_c_array(tflite_data, output_path):
    hex_lines = []
    for i in range(0, len(tflite_data), 12):
        chunk = tflite_data[i:i + 12]
        hex_str = ', '.join(f'0x{b:02x}' for b in chunk)
        hex_lines.append(f'  {hex_str},')
    c_code = (
        "#ifndef WASTE_MODEL_DATA_H\n"
        "#define WASTE_MODEL_DATA_H\n"
        "#include <stdint.h>\n\n"
        "alignas(16) const unsigned char "
        "waste_model_data[] = {\n"
        + '\n'.join(hex_lines) +
        "\n};\n\n"
        f"const unsigned int waste_model_data_len "
        f"= {len(tflite_data)};\n"
        "#endif\n"
    )
    with open(output_path, 'w') as f:
        f.write(c_code)

model = tf.keras.models.load_model('waste_smart_model.h5')
val_ds = tf.keras.utils.image_dataset_from_directory(
    'dataset', validation_split=0.2, subset='validation',
    seed=123, image_size=(INPUT_SIZE, INPUT_SIZE),
    batch_size=32)

out_dir = Path('esp32_export')
out_dir.mkdir(exist_ok=True)

tflite_data = convert_to_tflite_int8(
    model, val_ds, str(out_dir / 'waste_model.tflite'))
tflite_to_c_array(
    tflite_data, str(out_dir / 'model_data.h'))
print("Конвертацію завершено!")
```

А.3 Головний модуль прошивки ESP32 (main.cc)

```
#include <cstring>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "esp_netif.h"
#include "inference.h"
#include "web_server.h"

static const char* TAG = "MAIN";

#define WIFI_SSID     "SmartBin_Network"
#define WIFI_PASS     "password123"
#define MAX_RETRY     10

static EventGroupHandle_t s_wifi_event_group;
#define WIFI_CONNECTED_BIT BIT0
#define WIFI_FAIL_BIT      BIT1
static int s_retry_num = 0;

static void event_handler(void* arg,
    esp_event_base_t event_base,
    int32_t event_id, void* event_data)
{
    if (event_base == WIFI_EVENT &&
        event_id == WIFI_EVENT_STA_START) {
        esp_wifi_connect();
    } else if (event_base == WIFI_EVENT &&
        event_id == WIFI_EVENT_STA_DISCONNECTED) {
        if (s_retry_num < MAX_RETRY) {
            esp_wifi_connect();
            s_retry_num++;
        } else {
            xEventGroupSetBits(s_wifi_event_group,
                               WIFI_FAIL_BIT);
        }
    } else if (event_base == IP_EVENT &&
        event_id == IP_EVENT_STA_GOT_IP) {
        ip_event_got_ip_t* event =
            (ip_event_got_ip_t*) event_data;
        s_retry_num = 0;
        xEventGroupSetBits(s_wifi_event_group,
                           WIFI_CONNECTED_BIT);
    }
}

static void wifi_init_sta(void)
{
    s_wifi_event_group = xEventGroupCreate();
    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_create_default_wifi_sta();
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    esp_event_handler_instance_t inst_any, inst_ip;
    ESP_ERROR_CHECK(
        esp_event_handler_instance_register(
            WIFI_EVENT, ESP_EVENT_ANY_ID,
            &event_handler, NULL, &inst_any));
    ESP_ERROR_CHECK(
        esp_event_handler_instance_register(
            IP_EVENT, IP_EVENT_STA_GOT_IP,
            &event_handler, NULL, &inst_ip));

    wifi_config_t wifi_config = {};
    strncpy((char*)wifi_config.sta.ssid,
            WIFI_SSID, sizeof(wifi_config.sta.ssid));
    strncpy((char*)wifi_config.sta.password,
            WIFI_PASS, sizeof(wifi_config.sta.password));
```

```
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
    ESP_ERROR_CHECK(esp_wifi_set_config(
        WIFI_IF_STA, &wifi_config));
    ESP_ERROR_CHECK(esp_wifi_start());

    xEventGroupWaitBits(s_wifi_event_group,
        WIFI_CONNECTED_BIT | WIFI_FAIL_BIT,
        pdFALSE, pdFALSE, portMAX_DELAY);
}

extern "C" void app_main(void)
{
    esp_err_t ret = nvs_flash_init();
    if (ret == ESP_ERR_NVS_NO_FREE_PAGES ||
        ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
        ESP_ERROR_CHECK(nvs_flash_erase());
        ret = nvs_flash_init();
    }
    ESP_ERROR_CHECK(ret);
    wifi_init_sta();
    if (inference_init() != 0) return;
    start_webserver();
}
```

А.4 Модуль інференсу TFLite Micro (inference.cc)

```
#include "inference.h"
#include "model_data.h"
#include "classes.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "esp_log.h"

static const char* TAG = "INFERENCE";

constexpr int kTensorArenaSize = 90 * 1024;
static uint8_t tensor_arena[kTensorArenaSize]
    __attribute__((aligned(16)));

static tflite::MicroInterpreter* interpreter = nullptr;
static TfLiteTensor* input_tensor = nullptr;
static TfLiteTensor* output_tensor = nullptr;

extern "C" int inference_init(void)
{
    const tflite::Model* model =
        tflite::GetModel(waste_model_data);
    if (model->version() != TFLITE_SCHEMA_VERSION)
        return -1;

    static tflite::MicroMutableOpResolver<12> resolver;
    resolver.AddConv2D();
    resolver.AddDepthwiseConv2D();
    resolver.AddMaxPool2D();
    resolver.AddReshape();
    resolver.AddFullyConnected();
    resolver.AddSoftmax();
    resolver.AddQuantize();
    resolver.AddDequantize();
    resolver.AddMul();
    resolver.AddAdd();
    resolver.AddMean();
    resolver.AddRelu();

    static tflite::MicroInterpreter static_interp(
        model, resolver, tensor_arena, kTensorArenaSize);
    interpreter = &static_interp;

    if (interpreter->AllocateTensors() != kTfLiteOk)
        return -2;

    input_tensor  = interpreter->input(0);
    output_tensor = interpreter->output(0);
    return 0;
}

extern "C" int inference_classify(
    const unsigned char* rgb_data, int data_len,
    int* out_class, float* out_confidence)
{
    if (!interpreter) return -1;
    const int expected = INPUT_SIZE * INPUT_SIZE * 3;
    if (data_len != expected) return -1;

    float scale = input_tensor->params.scale;
    int   zp    = input_tensor->params.zero_point;
    int8_t* input = input_tensor->data.int8;

    for (int i = 0; i < expected; i++) {
        int32_t q = (int32_t)(
            (float)rgb_data[i] / scale + zp);
        if (q < -128) q = -128;
        if (q >  127) q =  127;
        input[i] = (int8_t)q;
    }

    if (interpreter->Invoke() != kTfLiteOk) return -2;

    float out_scale = output_tensor->params.scale;
```

```
    int   out_zp    = output_tensor->params.zero_point;
    int8_t* out = output_tensor->data.int8;

    int best_idx = 0; float best_val = -1e9;
    for (int i = 0; i < NUM_CLASSES; i++) {
        float val = (out[i] - out_zp) * out_scale;
        if (val > best_val) { best_val = val; best_idx = i; }
    }
    *out_class = best_idx;
    *out_confidence = best_val;
    return 0;
}
```

А.5 Модуль веб-сервера (web_server.c)

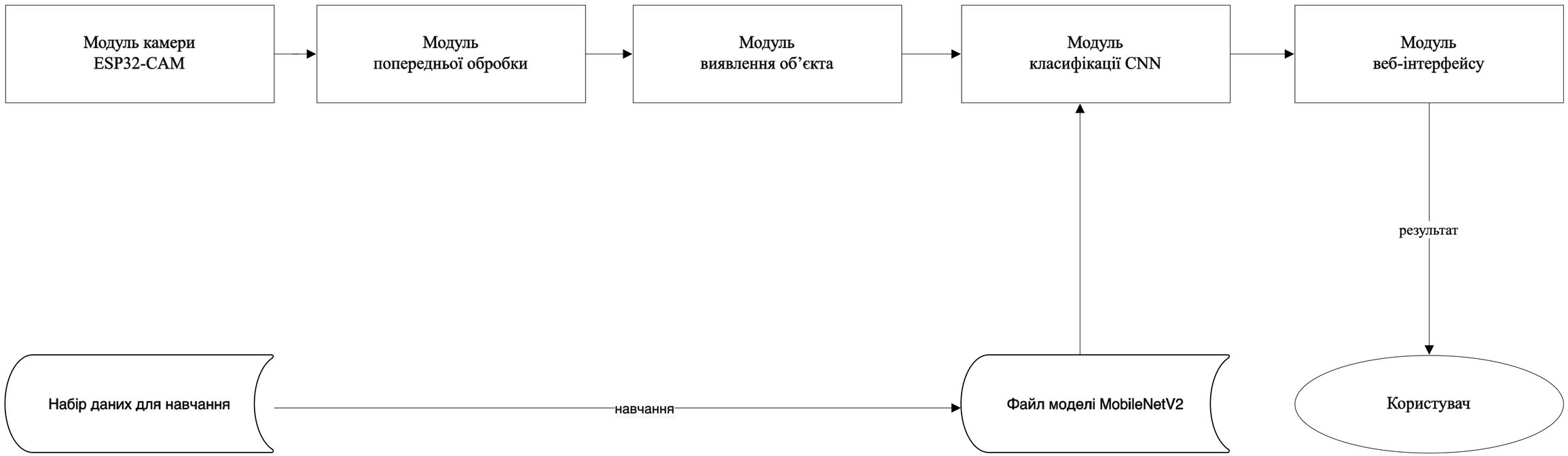
```
#include "web_server.h"
#include "inference.h"
#include "classes.h"
#include "esp_log.h"
#include <string.h>
#include <stdlib.h>

static const char* TAG = "WEBSERVER";

extern const char index_html_start[]
    asm("_binary_index_html_start");
extern const char index_html_end[]
    asm("_binary_index_html_end");

static esp_err_t root_handler(httpd_req_t *req)
{
    httpd_resp_set_type(req,
        "text/html; charset=utf-8");
    httpd_resp_send(req, index_html_start,
        index_html_end - index_html_start);
    return ESP_OK;
}

static esp_err_t classify_handler(httpd_req_t *req)
{
    const int expected = INPUT_SIZE * INPUT_SIZE * 3;
    if (req->content_len != expected) {
        httpd_resp_send_err(req,
            HTTPD_400_BAD_REQUEST, "Invalid size");
        return ESP_FAIL;
    }
    unsigned char* buf =
        (unsigned char*)malloc(expected);
    if (!buf) {
        httpd_resp_send_err(req,
            HTTPD_500_INTERNAL_SERVER_ERROR,
            "Out of memory");
        return ESP_FAIL;
    }
    int received = 0;
    while (received < expected) {
        int ret = httpd_req_recv(req,
            (char*)buf + received,
            expected - received);
        if (ret <= 0) { free(buf); return ESP_FAIL; }
        received += ret;
    }
    int class_idx = 0; float confidence = 0.0f;
    int rc = inference_classify(buf, expected,
        &class_idx, &confidence);
    free(buf);
    if (rc != 0) {
        httpd_resp_send_err(req,
            HTTPD_500_INTERNAL_SERVER_ERROR,
            "Inference failed");
        return ESP_FAIL;
    }
    char json[256];
    snprintf(json, sizeof(json),
        "{\"class\":\"%s\","
        "\"class_index\":%d,"
        "\"confidence\":%.3f}",
        CLASS_NAMES[class_idx], class_idx, confidence);
    httpd_resp_set_type(req, "application/json");
    httpd_resp_set_hdr(req,
        "Access-Control-Allow-Origin", "*");
    httpd_resp_send(req, json, strlen(json));
    return ESP_OK;
}

static esp_err_t cors_handler(httpd_req_t *req)
{
    httpd_resp_set_hdr(req,
        "Access-Control-Allow-Origin", "*");
```

```
    httpd_resp_set_hdr(req,
        "Access-Control-Allow-Methods",
        "POST, OPTIONS");
    httpd_resp_send(req, NULL, 0);
    return ESP_OK;
}

httpd_handle_t start_webserver(void)
{
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();
    config.max_uri_handlers = 8;
    config.stack_size = 8192;
    httpd_handle_t server = NULL;
    if (httpd_start(&server, &config) != ESP_OK)
        return NULL;
    httpd_uri_t root = {
        .uri="/", .method=HTTP_GET,
        .handler=root_handler, .user_ctx=NULL };
    httpd_register_uri_handler(server, &root);
    httpd_uri_t classify = {
        .uri="/classify", .method=HTTP_POST,
        .handler=classify_handler, .user_ctx=NULL };
    httpd_register_uri_handler(server, &classify);
    httpd_uri_t cors = {
        .uri="/classify", .method=HTTP_OPTIONS,
        .handler=cors_handler, .user_ctx=NULL };
    httpd_register_uri_handler(server, &cors);
    return server;
}
```

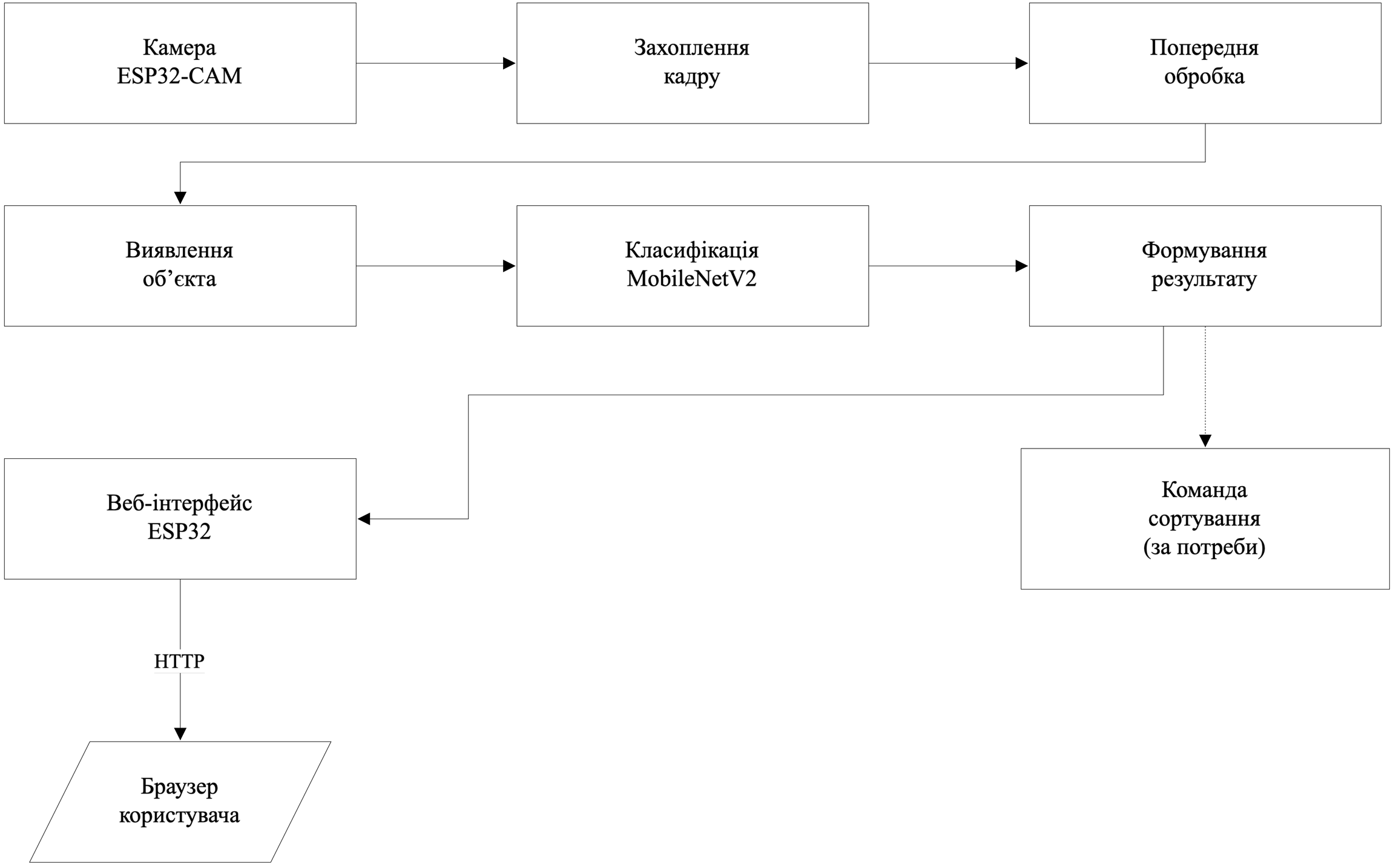
А.6 Веб-інтерфейс системи (index.html)

```
<!DOCTYPE html>
<html lang="uk">
<head>
<meta charset="UTF-8">
<meta name="viewport"
    content="width=device-width, initial-scale=1.0">
<title>WasteSmart ESP32</title>
<style>
* { margin:0; padding:0; box-sizing:border-box; }
body {
    font-family:'Segoe UI',system-ui,sans-serif;
    background:#0f1117; color:#e2e8f0;
    min-height:100vh; display:flex;
    flex-direction:column; align-items:center;
    padding:24px 16px;
}
.card {
    background:#1a1d27;
    border:1px solid #2d3348;
    border-radius:16px; padding:20px;
    margin-bottom:12px;
}
.camera-wrap {
    position:relative; border-radius:12px;
    overflow:hidden; background:#000;
    aspect-ratio:4/3;
}
.live-dot {
    position:absolute; top:12px; right:12px;
    padding:4px 10px; border-radius:20px;
    background:#ef4444; color:#fff;
}
.btn {
    width:100%; padding:14px; border:none;
    border-radius:10px; font-size:1rem;
    font-weight:600; cursor:pointer;
}
.btn-primary { background:#22c55e; color:#000; }
.btn-danger  { background:#ef4444; color:#fff; }
</style>
</head>
<body>
<h1>&#9851; WasteSmart</h1>
<p class="subtitle">Класифікація відходів на ESP32</p>
<div class="container">
  <div class="card">
    <div class="camera-wrap" id="cameraWrap"
        style="display:none">
      <video id="video" autoplay playsinline muted></video>
      <div class="live-dot">&#9679; LIVE</div>
      <div class="camera-overlay">
        <div class="camera-label" id="camLabel">
            Аналіз...</div>
        <div class="camera-conf" id="camConf"></div>
      </div>
    </div>
  </div>
  <button class="btn btn-primary" id="startBtn"
      onclick="startCamera()">
      &#128247; Увімкнути камеру
  </button>
  <button class="btn btn-danger" id="stopBtn"
      onclick="stopCamera()" style="display:none">
      Зупинити
  </button>
  <div class="card result-card" id="resultCard">
    <div class="result-header">
      <div class="result-icon" id="resultIcon">
          &#9851;</div>
      <div>
        <div class="result-label"
            id="resultLabel">—</div>
        <div class="result-conf"
            id="resultConf">—</div>
```

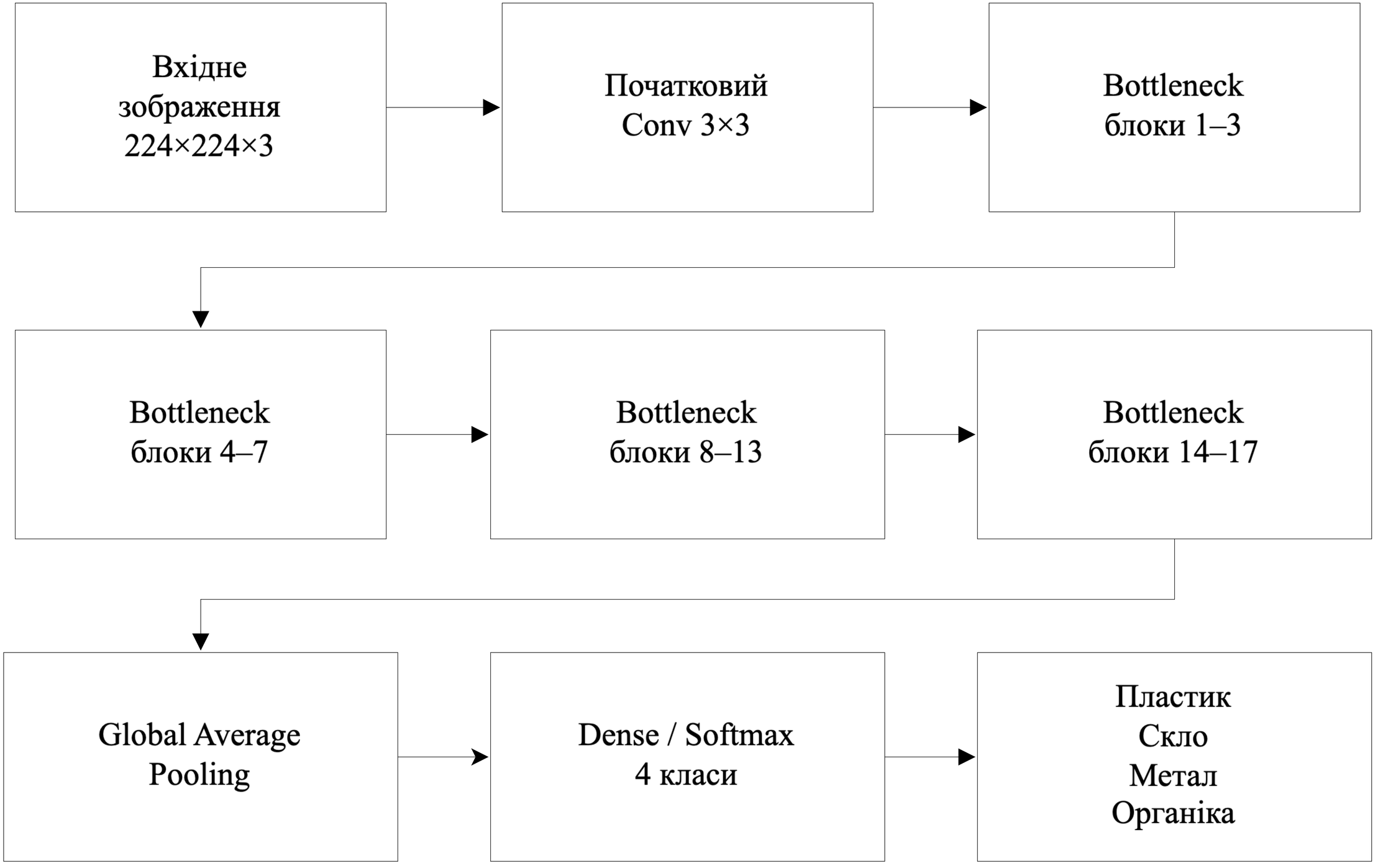
```
      </div>
    </div>
    <div class="bar-wrap">
      <div class="bar-fill" id="barFill"
          style="width:0%"></div>
    </div>
  </div>
</div>
<canvas id="canvas" width="64" height="64"></canvas>
<script>
const $ = s => document.querySelector(s);
const canvas = $('#canvas');
const ctx = canvas.getContext('2d',
    {willReadFrequently:true});
let stream=null, running=false;

async function startCamera() {
    stream = await navigator.mediaDevices
        .getUserMedia({video:{
            facingMode:{ideal:'environment'},
            width:{ideal:640}, height:{ideal:480}
        }});
    const v = $('#video');
    v.srcObject = stream;
    await v.play();
    running = true;
    setTimeout(loop, 500);
}

function stopCamera() {
    running = false;
    if (stream) {
        stream.getTracks().forEach(t => t.stop());
        stream = null;
    }
    $('#video').srcObject = null;
}

async function loop() {
    if (!running) return;
    const t0 = performance.now();
    ctx.drawImage($('#video'), 0, 0, 64, 64);
    const rgba = ctx.getImageData(0,0,64,64).data;
    const rgb  = new Uint8Array(64*64*3);
    for (let i=0,j=0; i<rgba.length; i+=4,j+=3) {
        rgb[j]=rgba[i]; rgb[j+1]=rgba[i+1];
        rgb[j+2]=rgba[i+2];
    }
    const resp = await fetch('/classify', {
        method:'POST',
        headers:{'Content-Type':
            'application/octet-stream'},
        body: rgb.buffer
    });
    const data = await resp.json();
    const conf = Math.round(data.confidence*100);
    $('#camLabel').textContent = data.class;
    $('#barFill').style.width  = conf + '%';
    $('#resultLabel').textContent = data.class;
    $('#resultConf').textContent  =
        'Впевненість: ' + conf + '%';
    $('#resultCard').style.display = 'block';
    if (running) setTimeout(loop, 400);
}
</script>
</body>
</html>
```



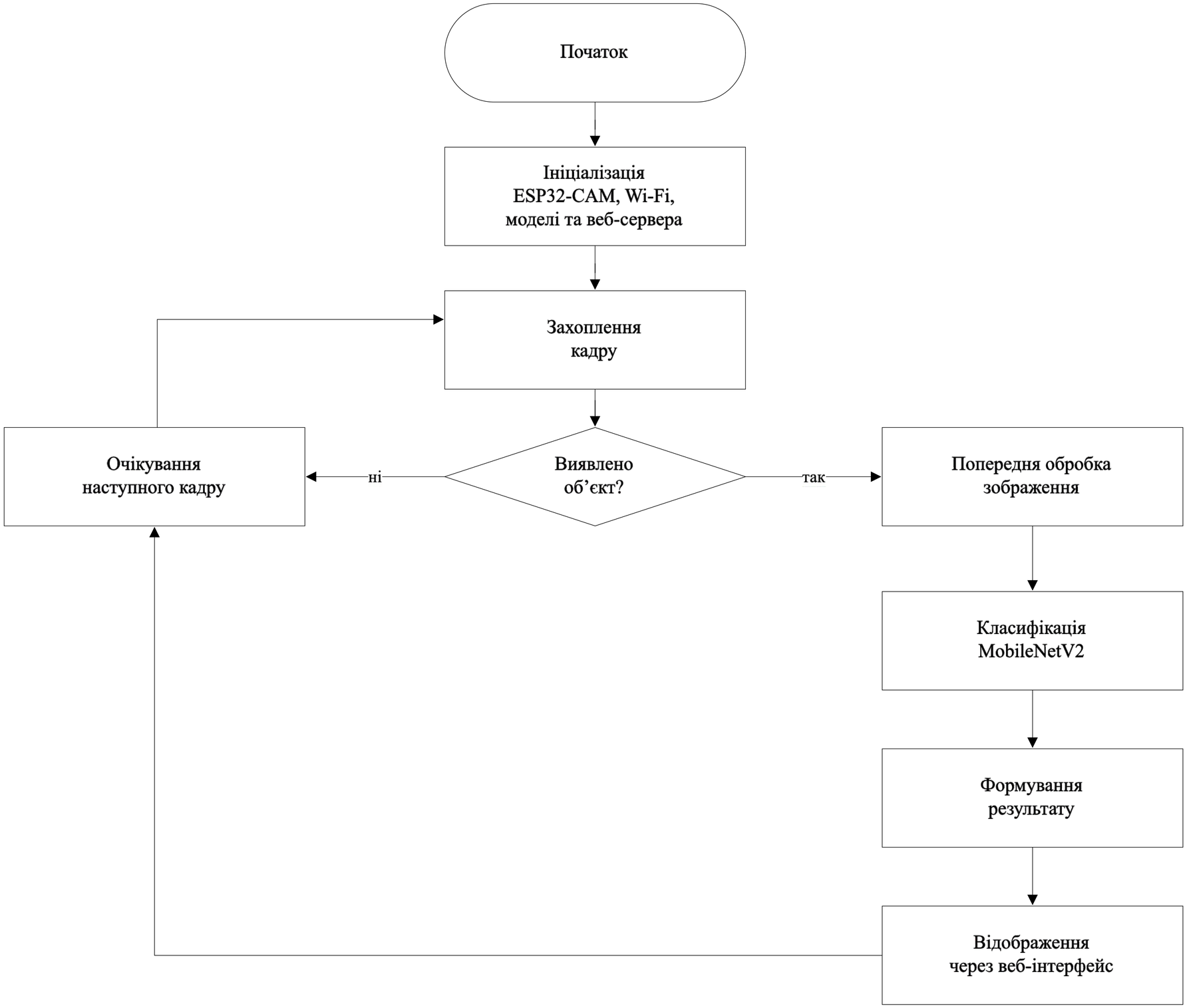
Зм.	Арк	№ документа	Підпис	Дата	26.КВР.123.405.17.00.00 СС			
Розроб.		Шпирна А.І.			Розробка системи ком`ютерного зору на основі алгоритмів машинного навчання для класифікації с	Літ.	Маса	Масштаб
Перевір.		Недошитко А.Г.			Структурна схема системи класифікації об`єктів утилізації			
Т.контр.						Аркуш	Аркушів	
Реценз.						ВСП «ТФК ТНТУ» гр. КІ-405 м. Тернопіль		
Н.контр.		Юзьків А.В.						
Затв.								



					26.КВР.123.405.17.00.00 ФС	
Зм.	Арк.	№ документа	Підпис	Дата	Розробка системи ком'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації Функціональна схема системи класифікації об'єктів утилізації	Літ. / Маса / Масштаб
Розроб.		Шпирна А.І.				
Перевір.		Недошитко А.Г.				Аркуш / Аркушів
Т.контр.						
Реценз.						ВСП «ТФК ТНТУ» гр. КІ-405 м. Тернопіль
Н.контр.		Юзьків А.В.				
Затв.						



					26.КВР.123.405.17.00.00 СС		
Зм.	Арк	№ документа	Підпис	Дата	Розробка системи ком'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації Архітектура нейронної мережі на основі MobileNetV2	Літ.	Маса / Масштаб
Розроб.		Шпирна А.І.					
Перевір.		Недошитко А.Г.				Аркуш	Аркушів
Т.контр.							
Реценз.						ВСП «ТФК ТНТУ» гр. КІ-405 м. Тернопіль	
Н.контр.		Юзьків А.В.					
Затв.							



Зм.	Арк	№ документа	Підпис	Дата
Розроб.		Шпирна А.І.		
Перевір.		Недошитко А.Г.		
Т.контр.				
Реценз.				
Н.контр.		Юзьків А.В.		
Затв.				

26.КВР.123.405.17.00.00 БС

Розробка системи ком'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації

Блок-схема алгоритму роботи класифікації

Літ.	Маса	Масштаб
Аркуш	Аркушів	

ВСП «ТФК ТНТУ»
гр. КІ-405
м. Тернопіль

ТАБЛИЦЯ ТЕХНІКО-ЕКОНОМІЧНИХ ПОКАЗНИКІВ

Технічні показники			Економічні показники			
№ п/п	Показник	Значення	№ п/п	Показник	Одиниці вимірювання	Значення
1	Апаратна платформа	ESP32-CAM	1	Собівартість	грн	74801
2	Діапазон частот	2.4 ГГц	2	Плановий прибуток	грн	41890
3	Протокол передачі даних	HTTP/REST API	3	Ціна	грн	116691
4	Роздільна здатність камери	2 МП	4	Чиста теперішня вартість	грн	29373,6
5	Напруга живлення	5 В (USB)	5	Термін окупності	рік	2,1

Зм.	Арк.	№ документа	Підпис	Дата	26.КВР.123.405.17.00.00 ТБ			
Розроб.		Шпирна А.І.			Розробка системи ком'ютерного зору на основі алгоритмів машинного навчання для класифікації об'єктів утилізації Таблиця техніко-економічних показників	Літ.	Маса	Масштаб
Перевір.		Недошитко А.Г.						
Т.контр.						Аркуш	Аркушів	
Реценз.						ВСП «ТФК ТНТУ» гр. КІ-405 м. Тернопіль		
Н.контр.		Юзьків А.В.						
Затв.								