Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ СКАНУВАННЯ ШТРИХ-КОДУ ПРОДУКТІВ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-2
Кузьміна О.О.
(прізвище, ініціали)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Інформатика
(повна назва освітньої програми)
Керівник проф. Машталір С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Кузьміній Олесі Олегівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного засобу для сканування штрих-коду продуктів

затверджена наказом університету від 19 травня 2025 року № 381Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 07 червня 2025 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV.

4. Перелік питань, що потрібно опрацювати в роботі

1. Проблематика та постановка задачі розробки програмного засобу для зчитування штрих-коду.

2. Розробка функціональної моделі для зчитування штрих-коду.

3. Комп'ютерна модель та її порівняння з існуючими аналогами.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми, постановка задачі, тестові зображення.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Машталір С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 69 с., 4 табл., 6 рис., 30 джерел.

ШТРИХ-КОД, АЛГОРИТМ ЗЧИТУВАННЯ, ДЕКОДУВАННЯ, ОБРОБКА ЗОБРАЖЕНЬ, БІНАРИЗАЦІЯ, МОДУЛЬ ЗАХОПЛЕННЯ ЗОБРАЖЕНЬ, ПОШУК КОНТУРІВ.

Об’єктом роботи виступає система автоматизованого зчитування штрих-кодів, що забезпечує ідентифікацію товарів за допомогою візуального кодування.

Метою роботи є розробка програмного засобу для зчитування штрих-кодів продуктів, який буде характеризуватися простотою використання, високою точністю і доступністю для широкого кола користувачів. Для розробки програмного засобу реалізовано попередню обробку зображення: переведення у відтінки сірого, згладжування, бінаризацію та виявлення контурів для точного визначення області розташування штрих-коду. Саме зчитування коду здійснюється за допомогою бібліотек OpenCV та ZXing, які підтримують основні типи штрих-кодів, зокрема EAN-13 та UPC.

Отриманий програмний продукт має забезпечити ефективність обробки інформації, зменшення помилок у введенні даних та підвищення швидкості обслуговування.

BARCODE, READING ALGORITHM, DECODING, IMAGE PROCESSING, BINARIZATION, IMAGE CAPTURE MODULE, CONTOUR DETECTION.

The object of this work is an automated barcode scanning system that enables product identification through visual coding.

The purpose of the project is to develop software capable of quickly and accurately reading barcodes using GS1 standards, followed by processing the obtained data.

For the development of the software tool, image preprocessing has been implemented: grayscale conversion, smoothing, binarization, and contour detection to accurately determine the location of the barcode. Barcode reading is performed using the OpenCV and Zxing libraries, which supports major barcode formats, including EAN-13 and UPC.

The resulting software product is intended to ensure efficient data processing, reduce input errors, and increase the speed of service.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1D – одновимірний (лінійний) штрих-код

2D – двовимірний штрих-код

CCD – Charge-Coupled Device (прилад із зарядовим зв’язком)

EAN-13 – European Article Number (європейський товарний номер)

GS1 – міжнародна організація, яка розробляє стандарти штрих-кодування, включаючи EAN і UPC

ISO – International Organization for Standardization (міжнародна організація зі стандартизації)

IEC – International Electrotechnical Commission (міжнародна електротехнічна комісія)

OpenCV – Open Source Computer Vision Library (бібліотека з відкритим кодом для обробки зображень і комп’ютерного зору)

POS – Point of Sale (точка продажу)

QR – Quick Response (тип двовимірного штрих-коду, що складається з матриці квадратів)

UPC – Universal Product Code (універсальний стандартний код)

UPC-A – версія UPC із 12 цифрами, використовується для ідентифікації товарів

ZXing – Zebra Crossing (бібліотека з відкритим кодом для зчитування та декодування штрих-кодів)

ZBar – бібліотека з відкритим кодом для швидкого зчитування штрих-кодів із зображень або відеопотоку

PDF417 – тип двовимірного штрих-коду, що використовується для кодування великих обсягів даних

Code 128 – лінійний штрих-код високої щільності, який підтримує алфавітно-цифрові символи

Data Matrix – двовимірний штрих-код, що складається з чорно-білих клітинок, часто застосовується в промисловості

RAM – Random Access Memory (оперативна пам’ять)

URL – Uniform Resource Locator (уніфікований локатор ресурсів)

ASCII – American Standard Code for Information Interchange (американський стандартний код для обміну інформацією)

ВСТУП

Сучасний світ характеризується стрімким розвитком інформаційних технологій, які проникають у всі сфери людської діяльності. Одним із ключових напрямів цього прогресу є автоматизація процесів, пов'язаних із ідентифікацією об'єктів, зокрема продуктів у торгівлі, логістиці та повсякденному житті. Штрих-коди, як універсальний засіб кодування інформації, відіграють важливу роль у цих процесах, забезпечуючи швидке, точне та зручне зчитування даних. З появою доступних цифрових пристроїв, таких як смартфони та вебкамери, з'явилася можливість створення програмних засобів, які дозволяють користувачам самостійно обробляти штрих-коди без необхідності використання спеціалізованого обладнання. Саме ця тенденція визначає актуальність розробки програмного засобу для зчитування штрих-кодів продуктів у цій кваліфікаційній роботі.

Проблематика зчитування штрих-кодів має глобальний характер. У світі вже існує значна кількість рішень, таких як апаратні сканери (Zebra, Honeywell) та програмні продукти (ZBar, ZXing), які успішно застосовуються в комерційних і промислових цілях. Проте ці рішення часто мають обмеження: висока вартість обладнання, складність інтеграції або недостатня адаптація до потреб індивідуальних користувачів. В рамках України, де зростає попит на доступні цифрові інструменти для малого бізнесу та особистих потреб, розробка простого, ефективного та економічного програмного засобу набуває особливої ваги. Таким чином, дана робота спрямована на створення програмного рішення, яке зможе забезпечити зчитування штрих-кодів із використанням стандартних пристроїв, таких як камери смартфонів, із подальшим декодуванням інформації для користувача.

Теоретична основа роботи базується на вивченні стандартів штрих-кодів (наприклад, EAN-13, QR), принципів обробки цифрових зображень і алгоритмів розпізнавання. Практична частина передбачає розробку програмного засобу з використанням сучасних мов програмування

(наприклад, Python) та бібліотек обробки зображень (OpenCV, ZXing). Отже, результатом роботи стане функціональний програмний застосунок, здатний зчитувати штрих-коди з продуктів у реальних умовах та часі, що може знайти застосування як у побуті, так і в малих комерційних проєктах.

Дана кваліфікаційна робота поєднує теоретичні знання та практичні навички для створення сучасного інструменту, який відповідає викликам цифрової ери та потребам суспільства.

1 ПРОБЛЕМАТИКА ТА ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ ДЛЯ ЗЧИТУВАННЯ ШТРИХ-КОДІВ

1.1 Поняття штрих-кодів: види, стандарти, принципи кодування

Штрих-коди є одним із найпоширеніших і найефективніших засобів автоматичної ідентифікації, які використовуються для кодування інформації у графічному форматі, доступному для оптичного зчитування. Вони стали невід'ємною частиною сучасного світу, застосовуючись у торгівлі, логістиці, медицині, промисловості та навіть у повсякденному житті. Завдяки своїй простоті, надійності та універсальності штрих-коди дозволяють швидко передавати дані про товари чи об'єкти, такі як їхня назва, виробник, ціна чи унікальний ідентифікатор, без необхідності ручного введення. Розуміння основ штрих-кодів, їхніх видів, стандартів і принципів кодування є фундаментальним для розробки програмного засобу, який здатен ефективно зчитувати ці символи.

Штрих-код – це графічне зображення, що складається з набору ліній, смуг або геометричних фігур, які представляють закодовану інформацію у форматі, зрозумілому для сканерів чи програмного забезпечення. Ідея штрих-кодів зародилася в 1949 році, коли американські інженери Норман Вудленд і Бернард Сілвер запатентували систему кодування на основі концентричних кіл, натхненну азбукою Морзе. Проте справжній прорив стався в 1974 році, коли в супермаркеті в Огайо (США) було вперше відскановано штрих-код UPC на пачці жувальної гумки Wrigley's. Ця подія ознаменувала початок масового використання штрих-кодів у роздрібній торгівлі. Відтоді технологія еволюціонувала, охопивши широкий спектр форматів і застосувань, від простих лінійних кодів до складних двовимірних систем.

Штрих-коди поділяються на два основні типи: лінійні (одновимірні, 1D) і двовимірні (2D). Лінійні штрих-коди (рис 1.1) є найстарішим і найпростішим видом, представленим вертикальними смугами різної товщини

та проміжками між ними. Найвідомішими прикладами є EAN-13 (European Article Number) і UPC-A (Universal Product Code). EAN-13, що складається з 13 цифр, широко застосовується в Європі та інших регіонах, включаючи Україну, для ідентифікації товарів у роздрібній торгівлі. Його структура включає код країни (2-3 цифри), код виробника (4-5 цифр), код продукту (5 цифр) і контрольну цифру. Наприклад, код 482 вказує на Україну як країну походження товару. UPC-A, який використовується переважно в США та Канаді, має 12 цифр і подібну логіку кодування. Принцип роботи таких кодів базується на бінарному представленні: кожна цифра кодується унікальною комбінацією смуг і проміжків, наприклад, у EAN-13 цифра 1 представлена як 0011001 у двійковій системі.

Рисунок 1.1 – Лінійний штрих-код

Двовимірні штрих-коди (рис 1.2) з’явилися пізніше й значно розширили можливості кодування. Вони здатні вміщувати від кількох сотень до кількох тисяч символів, включаючи текст, URL-адреси чи навіть бінарні дані. Найпоширенішим прикладом є QR-код (Quick Response), розроблений японською компанією Denso Wave у 1994 році для потреб автомобільної промисловості. QR-код складається з матриці чорних і білих квадратів, розташованих на сітці, і має три вирівнювальні маркери у кутах для правильного позиціонування при скануванні. Завдяки алгоритму корекції помилок Ріда-Соломона QR-код можна зчитати навіть при пошкодженні до 30% поверхні. Інший приклад – Data Matrix, який використовується в промислових і медичних цілях завдяки своїй компактності та здатності

кодувати дані на малих площах, наприклад, на етикетках ліків чи електронних компонентах [1].

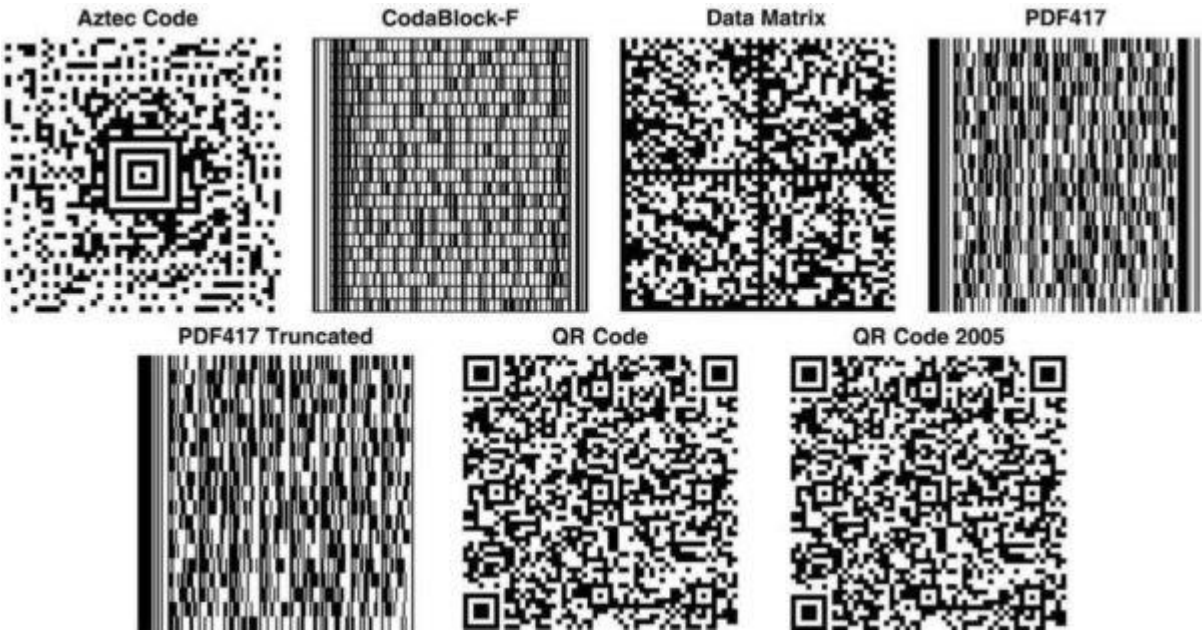

Рисунок 1.2 – Двовимірний штрих-код

Стандартизація штрих-кодів відіграє ключову роль у їхньому глобальному застосуванні. Міжнародна організація GS1, заснована в 1977 році, розробляє й підтримує стандарти для лінійних кодів, таких як EAN і UPC, забезпечуючи унікальність ідентифікаторів товарів у всьому світі. В Україні ці стандарти впроваджуються через національну організацію GS1 Ukraine, яка видає унікальні коди для місцевих виробників і дистриб'юторів. Наприклад, код 482 є офіційно зареєстрованим для України, що дозволяє ідентифікувати товари українського походження на міжнародному ринку. Крім того, Міжнародна організація зі стандартизації (ISO) регулює технічні аспекти штрих-кодів. Стандарт ISO/IEC 15416 визначає вимоги до якості друку лінійних штрих-кодів, включаючи контрастність, чіткість ліній і мінімальну ширину модулів. Для двовимірних кодів стандарт ISO/IEC 18004 описує специфікації QR-кодів, включаючи розміри модулів і алгоритми корекції помилок.

Принципи кодування штрих-кодів базуються на оптичному контрасті та бінарній логіці. У лінійних кодах інформація представлена через чергування темних смуг (логічна 1) і світлих проміжків (логічний 0). Кожна цифра має фіксовану довжину в модулях – базових одиницях ширини, що забезпечує однакову щільність кодування. Наприклад, у EAN-13 ширина модуля становить приблизно 0,33 мм при стандартному розмірі коду, а загальна довжина коду – 95 модулів. Контрольна цифра обчислюється за модульною арифметикою: сума цифр на парних позиціях множиться на 3, додається до суми цифр на непарних позиціях, а результат округляється до найближчого кратного 10. Наприклад, для коду 482123456789 контрольна цифра розраховується так: (4+2+2+4+6+8) × 3 + (8+1+3+5+7+9) = 66 + 33 = = 99, отже, контрольна цифра = 1 (100 - 99). Це дозволяє виявляти помилки при зчитуванні.

Двовимірні коди використовують складніші принципи кодування. У QR-кодах інформація розподіляється між областями даних, вирівнювальними маркерами та кодами корекції помилок. Дані спочатку стискаються (наприклад, за допомогою кодування ASCII), а потім розподіляються по модулях матриці. Алгоритм Ріда-Соломона додає надлишкову інформацію, яка забезпечує відновлення даних при пошкодженні. Наприклад, QR-код версії 1 (21×21 модуль) може вмістити до 25 алфавітно-цифрових символів із низьким рівнем корекції помилок або 17 символів із високим рівнем. Data Matrix, своєю чергою, використовує аналогічний підхід, але з компактнішим розташуванням модулів, що робить його ідеальним для маркування дрібних об’єктів.

Під час розробки програмного засобу важливо враховувати відмінності між типами штрих-кодів. Лінійні коди потребують простішої обробки зображень – визначення ширини смуг і їхньої послідовності, що можна реалізувати за допомогою базових алгоритмів комп’ютерного зору. Двовимірні коди, навпаки, вимагають складнішого аналізу матриць і використання спеціалізованих бібліотек, таких як ZXing чи OpenCV, для

декодування. У цій роботі основна увага приділяється підтримці EAN-13 як найпоширенішого лінійного коду в торгівлі та QR-кодів як популярного двовимірного формату в побуті й маркетингу. Такий вибір зумовлений їхньою масовістю та практичною значущістю для кінцевих користувачів.

Розуміння видів, стандартів і принципів кодування штрих-кодів формує основу для подальшого аналізу технологій їхнього зчитування. Лінійні ж коди простіші в реалізації, але обмежені за обсягом даних, тоді як двовимірні пропонують більшу гнучкість, але потребують складніших алгоритмів. Стандарти GS1 та ISO/IEC забезпечують уніфікацію й надійність технології, що є важливим для розробки універсального програмного засобу [2].

1.2 Актуальність проблеми зчитування штрих-кодів у сучасному світі

Сучасне суспільство перебуває в епосі цифрової трансформації, де автоматизація процесів є ключовим фактором підвищення ефективності в різних галузях. Одним із найбільш поширених інструментів для ідентифікації об’єктів, зокрема продуктів, є штрих-коди. Ці графічні символи, що складаються з чергування темних і світлих смуг або інших геометричних фігур, дозволяють швидко кодувати та зчитувати інформацію про товари, їх походження, ціну чи інші характеристики. Зчитування штрих-кодів стало невід’ємною частиною торгівлі, логістики, складського обліку та навіть повсякденного життя, наприклад, при скануванні QR-кодів для оплати чи отримання інформації. Актуальність проблеми зчитування штрих-кодів зумовлена їхньою масовістю та необхідністю адаптації технологій до нових умов, таких як зростання кількості цифрових пристроїв і потреба в доступних рішеннях для малих користувачів.

Штрих-коди з’явилися в середині XX століття, коли в 1949 році американські інженери Норман Вудленд і Бернард Сілвер запатентували першу систему штрихового кодування. З того часу технологія

еволюціонувала від простих лінійних кодів, таких як UPC і EAN, до складніших двовимірних форматів, наприклад, QR-кодів, які здатні вміщувати значно більше даних. Сьогодні штрих-коди використовуються в глобальному масштабі. За даними організації GS1, яка стандартизує штрих-кодування, щодня у світі сканується понад 5 мільярдів штрих-кодів. Ця статистика підкреслює їхню необхідність у сучасній економіці, особливо в роздрібній торгівлі, де швидкість і точність обробки інформації є критично важливими. Водночас, поширення смартфонів із камерами високої роздільної здатності відкрило нові можливості для створення програмних засобів, які замінюють традиційні сканери, роблячи технологію доступною для ширшого кола користувачів [3].

Проблема зчитування штрих-кодів у сучасному світі пов’язана з кількома аспектами. По-перше, традиційні апаратні сканери, хоча й ефективні, мають високу вартість і потребують спеціального обслуговування, що робить їх менш доступними для малого бізнесу чи індивідуальних користувачів. По-друге, існуючі програмні рішення, такі як бібліотеки ZXing чи ZBar, хоча й безкоштовні, часто потребують інтеграції в складні системи або додаткових навичок програмування для адаптації під конкретні задачі. По-третє, умови зчитування (освітлення, кут нахилу, якість зображення) можуть суттєво впливати на точність декодування, що створює потребу в розробці гнучких і стійких алгоритмів. Усе це свідчить про те, що створення простого, але ефективного програмного засобу для зчитування штрих-кодів є актуальним завданням, яке відповідає сучасним технологічним і соціальним запитам.

В Україні актуальність цієї проблеми підсилюється економічними та соціальними факторами. Зростання малого та середнього бізнесу, зокрема в сфері торгівлі та послуг, потребує недорогих інструментів для автоматизації. Наприклад, власники невеликих магазинів чи онлайн-продавці часто не можуть дозволити собі професійні сканери, але мають доступ до смартфонів чи комп’ютерів із камерами. Крім того, у побуті зчитування штрих-кодів

може допомагати споживачам отримувати інформацію про продукти, перевіряти їхню автентичність або відстежувати походження. У контексті цифровізації, яку активно підтримує Міністерство цифрової трансформації України, розробка таких програмних засобів сприяє інтеграції технологій у повсякденне життя громадян і бізнес-процеси.

Теоретична база для вирішення цієї проблеми ґрунтується на досягненнях у галузі обробки цифрових зображень і комп'ютерного зору. Такі технології, як OpenCV, дозволяють аналізувати зображення в реальному часі, виділяти штрих-коди та декодувати їх за допомогою спеціалізованих алгоритмів. Водночас, практична значущість полягає в тому, що подібні рішення можуть бути адаптовані до різних платформ – від настільних комп'ютерів до мобільних додатків, що розширює їхнє застосування. Актуальність також підтверджується науковими дослідженнями: у роботах, присвячених комп'ютерному зору, зазначається, що вдосконалення методів розпізнавання зображень є одним із пріоритетів сучасної інформатики.

Тобто, необхідність удосконалення технологій зчитування штрих-кодів у сучасному світі обумовлена їхньою повсюдністю, економічною доцільністю та технологічними викликами. Розробка програмного засобу, який би поєднував простоту, доступність і високу точність, відповідає потребам як окремих користувачів, так і малого бізнесу. Програма спрямована на створення такого рішення, що базується на сучасних технологіях і враховує реальні умови використання [4].

1.3 Огляд існуючих рішень у світі

Сьогодні у світі існує широкий спектр рішень для зчитування штрих-кодів, які можна поділити на апаратні та програмні. Ці технології застосовуються в різних сферах – від роздрібної торгівлі до логістики й медицини, демонструючи високу ефективність і адаптивність. Огляд таких

рішень дозволяє виявити їхні сильні та слабкі сторони, а також визначити нішу для розробки нового програмного засобу, що є актуальним у контексті даної роботи.

Апаратні рішення представлені сканерами штрих-кодів, які є найпоширенішим інструментом у комерційному секторі. Компанії, такі як Zebra Technologies і Honeywell, пропонують широкий асортимент пристроїв – від портативних ручних сканерів до стаціонарних систем для конвеєрних ліній. Наприклад, Zebra DS2208 – це популярний ручний сканер, який підтримує зчитування як лінійних (EAN-13, UPC), так і двовимірних (QR, Data Matrix) штрих-кодів. Такі пристрої характеризуються високою швидкістю (до 100 сканувань за секунду) і точністю, але їхня вартість (від 100 до 500 доларів США) робить їх менш доступними для індивідуальних користувачів чи малого бізнесу. Крім того, апаратні сканери часто потребують підключення до комп'ютера або POS-системи, що ускладнює їхню інтеграцію в мобільні сценарії.

Програмні рішення, навпаки, орієнтовані на використання наявного обладнання, такого як камери смартфонів чи вебкамери. Одним із найвідоміших прикладів є бібліотека ZXing (Zebra Crossing), яка підтримує декодування штрих-кодів на різних платформах, включаючи Java, C++ і Python. Ця бібліотека є відкритою, що дозволяє розробникам інтегрувати її у власні проєкти безкоштовно. Інший приклад – ZBar, який також є відкритим програмним забезпеченням і спеціалізується на швидкому розпізнаванні штрих-кодів із зображень чи відеопотоку. Такі рішення мають перевагу в доступності, але їхня ефективність залежить від якості камери та умов зчитування, таких як освітлення чи кут нахилу [5].

Мобільні додатки також займають значну частку ринку програмних рішень. Наприклад, програмний застосунок Barcode Scanner, заснований на ZXing, доступний для Android і дозволяє користувачам сканувати штрих-коди за допомогою камери смартфона. Аналогічно, QR Code Reader для iOS пропонує зручний інтерфейс і підтримку різних форматів кодів. Ці додатки

прості у використанні, але часто обмежені базовими функціями, такими як відображення закодованої інформації без можливості її подальшої обробки чи інтеграції в інші системи. Крім того, багато таких додатків містять рекламу або платні функції, що може бути незручним для користувачів.

У промислових масштабах застосовуються комплексні системи, які поєднують апаратні та програмні компоненти. Наприклад, системи автоматичного зчитування штрих-кодів на складах Amazon використовують камери високої роздільної здатності та алгоритми комп'ютерного зору для обробки тисяч одиниць товару за годину. Такі рішення базуються на технологіях машинного навчання та бібліотеках, подібних до OpenCV, що забезпечує високу точність навіть у складних умовах. Проте їхня складність і вартість роблять їх недоцільними для малих проєктів чи індивідуального використання.

Для оцінки існуючих рішень доцільно розглянути їхні ключові характеристики, такі як тип технології, доступність, точність і сфера застосування. У таблиці 1.1 наведено порівняння кількох популярних рішень, які відображають сучасний стан технологій зчитування штрих-кодів [6].

Таблиця 1.1 – Порівняння існуючих рішень для зчитування штрих-кодів

Назва рішення	**Тип**	**Доступність**	**Точність**	**Сфера застосування**
1	**2**	**3**	**4**	**5**
Zebra DS2208	Апаратний сканер	Платний ($150)	Висока	Торгівля, логістика
ZXing	Програмна бібліотека	Безкоштовна	Середня-висока	Мобільні додатки, розробка
ZBar	Програмна бібліотека	Безкоштовна	Середня	Розробка, тестування

Продовження таблиці 1.1

1	2	3	4	5
Barcode Scanner (Android)	Мобільний програмний застосунок	Безкоштовний (з рекламою)	Середня	Побутове використання
Amazon Warehouse System	Комплексна система	Платна (корпоративна)	Дуже висока	Промисловість, логістика

Аналіз таблиці показує, що апаратні рішення переважають за точністю та швидкістю, але поступаються за доступністю. Програмні бібліотеки, такі як ZXing і ZBar, є компромісом між вартістю та функціональністю, однак потребують додаткових зусиль для інтеграції. Мобільні додатки зручні для кінцевих користувачів, але обмежені в можливостях налаштування. Промислові системи, хоча й ефективні, є надто складними для малих проєктів.

У контексті України варто відзначити, що стандарти штрих-кодування, такі як EAN-13, регулюються організацією GS1 Ukraine, яка забезпечує унікальність кодів для місцевих виробників. Проте доступність програмних засобів для зчитування цих кодів залишається обмеженою, особливо для невеликих підприємств. Більшість існуючих рішень орієнтовані на західні ринки, що створює потребу в адаптованих інструментах, які враховують локальні особливості, наприклад, підтримку кириличних символів у QR-кодах чи інтеграцію з українськими базами даних.

Огляд існуючих рішень демонструє, що попри значний прогрес у технологіях зчитування штрих-кодів, залишається простір для розробки доступного програмного засобу, який би поєднував простоту використання, високу точність і адаптивність до потреб малих користувачів. Таке рішення може базуватися на відкритих бібліотеках, таких як OpenCV чи ZXing, і бути

реалізованим для стандартних пристроїв, таких як смартфони, що відповідає сучасним тенденціям цифровізації [7].

1.4 Постановка задачі

У сучасних умовах швидкого розвитку інформаційних технологій і зростання попиту на автоматизацію процесів ідентифікації товарів виникає потреба у створенні доступних і ефективних рішень для зчитування штрих-кодів.

Об’єктом роботи виступає система автоматизованого зчитування штрих-кодів, що забезпечує ідентифікацію товарів за допомогою візуального кодування.

Метою роботи є розробка програмного засобу для зчитування штрих-кодів продуктів, який буде характеризуватися простотою використання, високою точністю і доступністю для широкого кола користувачів. Ця мета відображає потребу в інструменті, який зможе замінити дорогі апаратні сканери в сценаріях малого бізнесу чи побутового використання, одночасно забезпечуючи стабільну роботу в реальних умовах. Для досягнення цієї мети необхідно вирішити низку завдань, які охоплюють як теоретичний аналіз, так і практичну реалізацію.

Основні завдання роботи включають:

– аналіз існуючих технологій і методів зчитування штрих-кодів, включаючи стандарти кодування (наприклад, EAN-13, QR) та алгоритми обробки зображень;

– вибір оптимального набору інструментів для розробки, таких як мови програмування (наприклад, Python) і бібліотеки (OpenCV, ZXing), з урахуванням їхньої доступності та ефективності;

– проектування алгоритму зчитування штрих-кодів, який включає етапи захоплення зображення, його попередньої обробки (видалення шумів, підвищення контрасту), виділення області штрих-коду та декодування даних;

– реалізація програмного засобу з інтуїтивно зрозумілим інтерфейсом користувача, який дозволить швидко отримувати інформацію із штрих-кодів;

– тестування розробленого рішення в різних умовах (освітлення, якість зображення, типи штрих-кодів) для оцінки його ефективності та виявлення можливих недоліків.

2 ФУНКЦІОНАЛЬНА МОДЕЛЬ ДЛЯ ЗЧИТУВАННЯ ШТРИХ-КОДІВ

2.1 Технології зчитування штрих-кодів

Технології зчитування штрих-кодів є ключовим елементом їхнього практичного застосування, забезпечуючи швидке й точне декодування закодованої інформації. Ці технології еволюціонували від простих оптичних пристроїв до складних програмно-апаратних комплексів, які інтегрують комп'ютерний зір і штучний інтелект. У контексті розробки програмного засобу для зчитування штрих-кодів важливо розглянути основні підходи – апаратні, програмні та гібридні методи, а також оцінити їхні особливості, переваги й обмеження. Це дозволить обрати оптимальну технологію для реалізації поставленої задачі.

Апаратні технології зчитування штрих-кодів базуються на використанні спеціалізованих пристроїв – сканерів, які застосовують оптичні методи для захоплення зображення коду та його інтерпретації. Найпоширенішим типом є лазерні сканери, які працюють за принципом сканування поверхні штрих-коду вузьким лазерним променем. Лазер відбивається від світлих і темних смуг із різною інтенсивністю, а фотодетектор перетворює ці відбиття в електричні сигнали, які декодуються в цифрові дані. Наприклад, сканер Zebra DS2208 використовує лазерну технологію для зчитування лінійних кодів зі швидкістю до 100 сканувань за секунду. Такі пристрої забезпечують високу точність і стабільність, але їхня залежність від фізичного контакту з кодом і висока вартість (від 100 доларів) обмежують їхню доступність.

Іншим апаратним методом є технологія CCD (прилад із зарядовим зв'язком), яка використовує матрицю світлочутливих елементів для захоплення зображення штрих-коду. На відміну від лазерних сканерів, CCD-пристрої не мають рухомих частин, що підвищує їхню надійність. Вони зчитують код як суцільне зображення, а не поелементно, що дозволяє

працювати з пошкодженими чи нечіткими кодами. Прикладом є сканер Honeywell Voyager 1200g, який підтримує як 1D, так і деякі 2D коди. Проте CCD-сканери мають менший діапазон зчитування (до 30 см) і потребують хорошого освітлення, що може бути проблемою в реальних умовах.

Програмні технології зчитування штрих-кодів набули популярності з поширенням цифрових камер і комп'ютерного зору. Вони базуються на обробці зображень, отриманих із камер смартфонів, вебкамер чи інших пристроїв, і не потребують спеціалізованого обладнання. Основний принцип полягає в захопленні зображення штрих-коду, його попередній обробці (наприклад, підвищення контрасту чи видалення шумів) і декодуванні за допомогою алгоритмів. Бібліотека OpenCV, наприклад, дозволяє реалізувати ці етапи, використовуючи методи бінаризації (переведення зображення в чорно-білий формат) і контурного аналізу для виділення штрих-коду. Програмні рішення, такі як ZXing, додатково пропонують готові алгоритми декодування для EAN-13, QR-кодів та інших форматів. Їхня перевага – доступність і гнучкість, але точність залежить від якості камери та умов зйомки.

Гібридні технології поєднують апаратні та програмні підходи, використовуючи камери високої роздільної здатності й потужне програмне забезпечення для обробки даних. Такі системи застосовуються в промислових масштабах, наприклад, на складах Amazon, де конвеєрні камери зчитують штрих-коди в реальному часі, а алгоритми машинного навчання (на основі бібліотек типу TensorFlow) підвищують точність розпізнавання навіть при низькій якості зображення. Ці технології забезпечують швидкість (до 1000 кодів за хвилину) і стійкість до зовнішніх факторів, але їхня складність і вартість роблять їх недоцільними для малих проєктів.

Процес зчитування штрих-кодів зазвичай включає кілька етапів. Для лінійних кодів апаратні сканери фіксують відбиття світла від смуг, перетворюючи їх у послідовність двійкових значень, які зіставляються зі стандартом (наприклад, EAN-13). Програмні методи починаються із

захоплення зображення, після чого застосовуються фільтри (наприклад, Гаусів розмиття для зменшення шумів) і алгоритми порогування (метод Оцу), щоб виділити код. Далі визначається ширина смуг і їхня послідовність, а контрольна цифра перевіряється для валідації. Для двовимірних кодів, таких як QR, процес складніший: зображення вирівнюється за маркерами, матриця аналізується на модулі, а дані декодуються з урахуванням корекції помилок.

Кожен метод має свої переваги й недоліки. Лазерні сканери швидкі й точні, але дорогі й обмежені в мобільності. CCD-сканери компактні й надійні, але менш універсальні. Програмні рішення економічні й адаптивні, але чутливі до умов зчитування. Гібридні системи ідеальні для великих обсягів, але надто складні для індивідуального використання. У контексті цієї роботи пріоритет надається програмним технологіям, оскільки вони відповідають меті створення доступного засобу для зчитування штрих-кодів на базі стандартних пристроїв, таких як смартфони.

Технології зчитування також залежать від типу штрих-коду. Лінійні коди простіші для апаратного сканування завдяки чіткій структурі смуг, тоді як програмні методи вимагають точного визначення меж коду на зображенні. Двовимірні коди, навпаки, легше обробляти програмно, оскільки їхня матрицева структура дозволяє застосовувати алгоритми розпізнавання образів. Наприклад, QR-код потребує виявлення вирівнювальних маркерів і декодування модулів, що реалізується через бібліотеки типу ZXing чи ZBar. У реальних умовах фактори, такі як освітлення, кут нахилу чи роздільна здатність камери, суттєво впливають на результат, що вимагає додаткової оптимізації алгоритмів.

Україна активно впроваджує технології зчитування штрих-кодів у торгівлі та логістиці, спираючись на стандарти GS1. Проте більшість доступних рішень – це імпортні сканери чи комерційні додатки, які не завжди враховують локальні потреби, наприклад, інтеграцію з українськими базами даних чи підтримку кириличних символів у QR-кодах. Це підкреслює

необхідність розробки власного програмного засобу, який би поєднував доступність із функціональністю.

Технології зчитування штрих-кодів охоплюють широкий спектр методів – від апаратних сканерів до програмних бібліотек і гібридних систем. Для даної роботи найперспективнішими є програмні технології, які базуються на обробці зображень і дозволяють реалізувати зчитування на стандартних пристроях. Вибір конкретного підходу залежить від аналізу їхньої ефективності та адаптивності до умов експлуатації, що буде розглянуто в наступних розділах [8].

2.2 Огляд бібліотек та алгоритмів для обробки зображень зі штрих-кодами

Розробка програмного засобу для зчитування штрих-кодів значною мірою залежить від використання сучасних бібліотек і алгоритмів обробки зображень. Ці інструменти дозволяють захоплювати зображення, виділяти штрих-коди, обробляти їх і декодувати закодовану інформацію. Огляд основних бібліотек і алгоритмів, які застосовуються для цих цілей, є важливим етапом для вибору оптимального набору технологій, що відповідають меті роботи – створення доступного й ефективного рішення. У цьому підрозділі зібрані найпоширеніші бібліотеки, їхні можливості, а також ключові алгоритми, які лежать в основі обробки зображень зі штрих-кодами.

Однією з найвідоміших бібліотек для обробки зображень є OpenCV (Open Source Computer Vision Library). Ця бібліотека з відкритим кодом, розроблена в 1999 році компанією Intel, підтримує широкий спектр функцій для комп’ютерного зору, включаючи зчитування штрих-кодів. OpenCV доступна для різних мов програмування, таких як Python, C++ і Java, і сумісна з платформами Windows, Linux, Android та iOS. Вона включає модулі для попередньої обробки зображень (фільтрація, бінаризація), виявлення

контурів і декодування штрих-кодів. Наприклад, функція cv2.threshold() дозволяє перевести зображення в чорно-білий формат, а cv2.findContours() – виділити область штрих-коду. OpenCV підтримує як лінійні (EAN-13, UPC), так і двовимірні (QR) коди через вбудований модуль cv2.QRCodeDetector. Її перевагою є гнучкість і велика спільнота розробників, але для складних кодів може знадобитися додаткова інтеграція з іншими бібліотеками.

Ще одна популярна бібліотека – ZXing (Zebra Crossing), створена в 2007 році командою Google. ZXing спеціалізується на зчитуванні та декодуванні штрих-кодів і підтримує широкий спектр форматів, включаючи EAN-13, UPC-A, QR, Data Matrix і PDF417. Вона доступна для Java, C++, Python (через обгортку pyzxing) і використовується в багатьох мобільних додатках, таких як Barcode Scanner для Android. ZXing працює за принципом аналізу бінаризованого зображення: спочатку визначаються межі коду, потім проводиться сканування рядків або матриці для отримання двійкових даних, які декодуються відповідно до стандарту. Бібліотека є легкою й ефективною для простих сценаріїв, але її точність може знижуватися при низькій якості зображення чи складних умовах освітлення.

ZBar – ще одна бібліотека з відкритим кодом, орієнтована на швидке зчитування штрих-кодів із зображень або відеопотоку. Розроблена в 2005 році, вона підтримує формати EAN-13, UPC, QR і Code 128, і доступна для C, Python (через pyzbar) і інших мов. ZBar використовує алгоритм сканування рядків для лінійних кодів і аналіз матриці для двовимірних, забезпечуючи високу швидкість обробки. Її перевагою є простота інтеграції та низькі системні вимоги, що робить її популярною для вбудованих систем і легких додатків. Однак ZBar менш гнучка порівняно з OpenCV і не пропонує розширених функцій обробки зображень, таких як фільтрація чи корекція перспективних спотворень.

Бібліотека Quirc (Quick Response Code decoder) спеціалізується виключно на декодуванні QR-кодів. Вона легка, написана на C і оптимізована для швидкості, що робить її ідеальною для ресурсообмежених пристроїв.

Quirc аналізує зображення, виявляє вирівнювальні маркери QR-коду, виправляє геометричні спотворення й декодує дані з урахуванням корекції помилок. Хоча її функціонал обмежений QR-кодами, вона демонструє високу ефективність у цій ніші, що може бути корисним для проєктів із чітко визначеним типом штрих-кодів.

Алгоритми обробки зображень зі штрих-кодами включають кілька ключових етапів. Перший етап – попередня обробка, яка охоплює фільтрацію (наприклад, Гаусів розмиття для зменшення шумів) і бінаризацію (метод Оцу чи адаптивне порогування). Другий етап – виявлення штрих-коду, де застосовуються алгоритми пошуку контурів (наприклад, алгоритм Кенні в OpenCV) або аналіз градієнтів для визначення меж коду. Для лінійних кодів далі проводиться сканування рядків із підрахунком ширини смуг, а для двовимірних – аналіз матриці модулів. Третій етап – декодування, яке базується на стандартах кодування (наприклад, EAN-13 чи QR) і перевірці контрольних сум чи кодів корекції помилок.

Для порівняння основних бібліотек доцільно розглянути їхні характеристики, такі як підтримувані формати, швидкість, складність інтеграції та системні вимоги. У таблиці 2.1 наведено огляд чотирьох популярних бібліотек, які можуть бути використані для розробки програмного засобу.

Таблиця 2.1 – Порівняння бібліотек для обробки зображень зі штрих-кодами

Бібліотека	**Підтримувані формати**	**Швидкість обробки**	**Складність інтеграції**	**Системні вимоги**
1	**2**	**3**	**4**	**5**
OpenCV	EAN-13, UPC, QR, Data Matrix	Висока	Середня	Середні (2 ГБ RAM)
ZXing	EAN-13, UPC, QR, PDF417	Середня	Низька	Низькі (512 МБ RAM)

Продовження таблиці 2.1

1	2	3	4	5
ZBar	EAN-13, UPC, QR, Code 128	Висока	Низька	Низькі (256 МБ RAM)
Quirc	QR	Дуже висока	Низька	Дуже низькі (128 МБ RAM)

Аналіз таблиці показує, що OpenCV є найбільш універсальною бібліотекою завдяки підтримці широкого спектра функцій і форматів, але вона потребує більше ресурсів і часу на налаштування. ZXing і ZBar пропонують простоту й ефективність для базових завдань, тоді як Quirc є нішевим рішенням для QR-кодів. Вибір бібліотеки залежить від цілей проєкту: для комплексного рішення з підтримкою різних кодів OpenCV є оптимальним, тоді як для легкого застосунку підійде ZBar чи ZXing.

Алгоритми обробки зображень також варіюються залежно від типу штрих-коду. Для лінійних кодів часто використовується метод сканування профілю інтенсивності, де зображення аналізується по горизонтальних лініях для визначення ширини смуг. Для двовимірних кодів застосовуються алгоритми локалізації (Hough Transform для пошуку маркерів) і декодування (Ріда-Соломона). У реальних умовах важливу роль відіграють методи корекції, такі як виправлення перспективних спотворень чи адаптивна бінаризація, які доступні в OpenCV.

Для цієї роботи пріоритет надається бібліотекам із відкритим кодом, які підтримують EAN-13 і QR-коди, оскільки ці формати є найпоширенішими в торгівлі та побуті. OpenCV і ZXing виглядають як основні кандидати завдяки їхній гнучкості й широкій документації. Алгоритми обробки зображень, такі як бінаризація Оцу та контурний аналіз, будуть базою для реалізації, забезпечуючи стійкість до шумів і змін

освітлення. Таким чином, огляд бібліотек і алгоритмів закладає основу для вибору інструментів у наступних етапах розробки [9].

2.3 Теоретичні аспекти інтерфейсу користувача для подібних систем

Розробка програмного засобу для зчитування штрих-кодів передбачає не лише технічну реалізацію алгоритмів обробки зображень і декодування, але й створення зручного інтерфейсу користувача (UI), який забезпечує ефективну взаємодію з програмою. Інтерфейс є ключовим елементом, що визначає доступність і практичну цінність системи, адже від нього залежить, наскільки швидко й легко користувач зможе отримати інформацію зі штрих-коду. Теоретичні аспекти проектування UI для подібних систем базуються на принципах ергономіки, usability (зручності використання) і адаптивності до потреб цільової аудиторії, яка може включати як технічних спеціалістів, так і звичайних користувачів без глибоких знань у програмуванні. У контексті цієї роботи розглядаються основні вимоги до інтерфейсу, його компоненти та підходи до забезпечення інтуїтивного досвіду.

Інтерфейс користувача для систем зчитування штрих-кодів має бути мінімалістичним і орієнтованим на основну функцію – сканування й відображення даних. Основним принципом є простота: користувач повинен мати можливість виконати сканування за мінімальну кількість дій. Наприклад, у мобільних додатках, таких як Barcode Scanner, інтерфейс часто складається з великої кнопки для запуску камери та області для відображення результату. Такий підхід зменшує когнітивне навантаження й дозволяє навіть недосвідченим користувачам швидко освоїти програму. Для настільних систем доцільно передбачити аналогічну структуру, де основне вікно містить область для перегляду відеопотоку з камери, кнопку запуску сканування та поле для виведення декодованої інформації, наприклад, номера продукту чи URL-адреси.

Важливим аспектом є адаптивність інтерфейсу до різних пристроїв і платформ. Оскільки програмний засіб розрахований на використання стандартного обладнання, такого як смартфони чи комп’ютери, UI має бути сумісним із різними розмірами екранів і типами вводу – сенсорним, мишею чи клавіатурою. Наприклад, для мобільних пристроїв доцільно використовувати сенсорні елементи керування, такі як кнопки великого розміру, щоб полегшити взаємодію на маленьких екранах. Для настільних комп’ютерів можна додати підтримку гарячих клавіш, що прискорить роботу для досвідчених користувачів. Принцип респонсивного дизайну, який адаптує розташування елементів до роздільної здатності екрана, є стандартом для сучасних інтерфейсів і дозволяє забезпечити однакову зручність на різних платформах.

Ергономіка інтерфейсу також передбачає чітке візуальне оформлення. Колірна палітра має бути контрастною, але не агресивною – наприклад, темний фон із білими чи світлими текстами для полів виведення інформації. Елементи керування, такі як кнопки, повинні мати виразні обриси та підписи, щоб уникнути плутанини. Важливо уникати перевантаження інтерфейсу зайвими функціями: для більшості користувачів достатньо базового набору – сканування, відображення результату та, можливо, збереження даних у файл чи буфер обміну. Водночас, для просунутих користувачів можна передбачити додаткові опції, наприклад, вибір типу штрих-коду чи налаштування параметрів камери, але ці функції мають бути винесені в окремий розділ налаштувань, щоб не ускладнювати основний сценарій використання.

Ще одним важливим аспектом є зворотний зв’язок. Користувач повинен буде отримувати чіткі сигнали про стан системи: наприклад, повідомлення про успішне сканування, помилку (якщо штрих-код не розпізнано) чи потребу в коригуванні положення камери. Візуальні індикатори, такі як рамка навколо виявленого штрих-коду чи спалах при успішному декодуванні, підвищують інтуїтивність. Звукові сигнали, як-от

короткий сигнал після сканування, також можуть бути корисними, але вони мають бути опціональними, щоб не дратувати користувачів. Для мобільних додатків вібрація може слугувати додатковим каналом зворотного зв'язку, особливо в шумних умовах [10].

Теоретичні основи проектування інтерфейсу спираються на принципи usability, сформульовані Якобом Нільсеном, зокрема ефективність, інтуїтивність і мінімалізація помилок користувача. Наприклад, ефективність досягається завдяки скороченню кількості кроків для виконання задачі – ідеально, якщо сканування запускається автоматично при виявленні штрих-коду в кадрі. Інтуїтивність забезпечується логічним розташуванням елементів: кнопка сканування в центрі екрана, результат – унизу чи праворуч. Для мінімізації помилок доцільно передбачити підказки, наприклад, текстове повідомлення «Наведіть камеру на штрих-код», якщо код не виявлено протягом кількох секунд. Ці принципи дозволяють створити інтерфейс, який буде однаково зручним для новачків і досвідчених користувачів.

Ще один важливий фактор – адаптація для цільової аудиторії. Для даної роботи аудиторія – власники малого бізнесу (наприклад, продавців у невеликих магазинах), які потребують швидкої інвентаризації, та звичайні користувачі, які хочуть перевірити інформацію про продукти. Для бізнес-користувачів інтерфейс може включати функцію експорту даних у CSV-файл чи інтеграцію з базами даних, тоді як для побутового використання достатньо простого виведення тексту, наприклад, назви товару чи терміну придатності. Локалізація інтерфейсу, зокрема підтримка української мови, є обов'язковою для забезпечення більшої доступності. Усі підписи кнопок і повідомлення мають бути українською, а формат виведення даних (дати, числа) – відповідати локальним стандартам.

Теоретичні аспекти також охоплюють вибір технологій для реалізації інтерфейсу. Для настільних додатків доцільно використовувати бібліотеки графічного інтерфейсу, такі як Tkinter або PyQt для Python, які забезпечують

кросплатформність і простоту розробки. Tkinter є частиною стандартної бібліотеки Python і дозволяє створювати базові інтерфейси з мінімальними зусиллями, наприклад, вікно з кнопкою сканування та текстовим полем. PyQt пропонує ширші можливості для дизайну, включаючи підтримку стилів і анімацій, але потребує додаткового встановлення. Для мобільних платформ можна розглядати фреймворки, такі як Kivy, які дозволяють створювати сенсорно-орієнтовані інтерфейси, сумісні з Android і iOS. Вибір між цими інструментами залежить від пріоритетів: Tkinter для швидкого прототипування, PyQt для складнішого дизайну, Kivy для мобільної версії.

Доступність інтерфейсу є ще одним важливим аспектом. UI має враховувати потреби користувачів із обмеженими можливостями, наприклад, із вадами зору. Це може включати підтримку висококонтрастних тем, можливість масштабування тексту чи озвучення результатів сканування через синтез мовлення. Такі функції не є обов'язковими для базової версії програми, але їхнє врахування підвищує універсальність рішення. Крім того, інтерфейс має бути стійким до помилок користувача: наприклад, якщо камера недоступна, програма повинна видати зрозуміле повідомлення замість аварійного завершення.

Теоретичні принципи також передбачають тестування інтерфейсу на етапі розробки. Метод юзабіліті-тестування, який включає аналіз поведінки реальних користувачів, дозволяє виявити недоліки, такі як незручне розташування кнопок чи нечіткі повідомлення.

Ітеративний підхід до розробки UI, коли прототипи тестуються й удосконалюються, є стандартом у сучасному програмуванні та забезпечує високу якість кінцевого продукту.

Інтерфейс має бути інтуїтивним, мінімалістичним і орієнтованим на основну задачу – швидке отримання інформації зі штрих-коду. Вибір технологій, таких як Tkinter чи PyQt, дозволяє реалізувати ці принципи в межах обраної екосистеми Python. Теоретичні аспекти, розглянуті в цьому

підрозділі, створюють основу для практичної реалізації інтерфейсу, яка буде описана в наступному розділі роботи [11].

2.4 Обґрунтування вибору середовища програмної реалізації

Розробка програмного засобу для зчитування штрих-кодів потребує ретельного вибору інструментів, які забезпечать ефективну реалізацію поставлених завдань. До таких інструментів належать мови програмування, бібліотеки обробки зображень, а також середовища розробки. Вибір залежить від вимог до продуктивності, доступності, простоти інтеграції та сумісності з цільовими платформами. У цьому підрозділі розглядаються обрані інструменти, їхні характеристики та обґрунтування доцільності використання для створення програмного засобу, здатного зчитувати штрих-коди, зокрема EAN-13 і QR, із продуктів за допомогою стандартних пристроїв, таких як камери смартфонів чи комп’ютерів.

Для розробки програмного засобу обрано мову програмування Python. Python є однією з найпопулярніших мов завдяки своїй простоті, читабельності коду та великій екосистемі бібліотек. Вона підтримує об’єктно-орієнтоване й функціональне програмування, що дозволяє створювати модульні та масштабовані програми. Python має потужну підтримку для обробки зображень і комп’ютерного зору через бібліотеки, такі як OpenCV і pyzxing, що є критично важливим для зчитування штрих-кодів. Крім того, Python є кросплатформною мовою, що забезпечує сумісність із Windows, Linux і macOS, а також полегшує розгортання на мобільних платформах через фреймворки, такі як Kivy чи BeeWare. Висока продуктивність Python для прототипування та велика спільнота розробників, які надають документацію й приклади коду, роблять його оптимальним вибором для даного проєкту.

Основною бібліотекою для обробки зображень обрано OpenCV (Open Source Computer Vision Library). OpenCV є універсальним інструментом із відкритим кодом, який пропонує широкий набір функцій для комп'ютерного зору, включаючи фільтрацію зображень, виявлення контурів і декодування штрих-кодів. Вона підтримує обробку як лінійних (EAN-13, UPC), так і двовимірних (QR) кодів через вбудовані модулі, такі як cv2.QRCodeDetector. OpenCV дозволяє реалізувати ключові етапи зчитування штрих-кодів: бінаризацію (наприклад, за методом Оцу), пошук контурів (алгоритм Кенні) і корекцію спотворень. Її переваги включають високу швидкість обробки, підтримку Python через модуль «cv2» і активну спільноту, яка забезпечує регулярні оновлення та навчальні матеріали. OpenCV також сумісна з камерами різних пристроїв, що відповідає вимозі використання стандартного обладнання [12].

Для декодування штрих-кодів обрано бібліотеку ZXing (Zebra Crossing). ZXing підтримує широкий спектр форматів, включаючи EAN-13, QR, UPC-A і Data Matrix, і є відкритою, що дозволяє безкоштовно інтегрувати її в проєкт. У Python вона доступна через обгортку pyzxing, що спрощує використання. ZXing ефективно декодує штрих-коди шляхом аналізу бінаризованих зображень, визначаючи послідовність смуг (для 1D кодів) або модулів (для 2D кодів). Її перевагою є легкість інтеграції та висока точність для стандартних умов зчитування. У поєднанні з OpenCV, яка забезпечує попередню обробку зображень, ZXing дозволяє досягти стабільного декодування навіть при часткових спотвореннях коду.

Для написання, тестування та налагодження коду обрано інтегроване середовище розробки (IDE) PyCharm від JetBrains. PyCharm спеціально оптимізований для Python, пропонуючи такі функції, як автодоповнення коду, аналіз помилок у реальному часі, інтеграція з системами контролю версій (Git) і підтримка бібліотек, таких як OpenCV і ZXing. Безкоштовна Community Edition забезпечує достатній функціонал для розробки програмного засобу, включаючи вбудовані інструменти для роботи з

віртуальними середовищами та пакетами через pip. PyCharm також полегшує налаштування проєктів із зовнішніми бібліотеками, що є важливим для інтеграції OpenCV і ZXing.

Для обробки числових даних і візуалізації результатів обрано бібліотеки NumPy і Matplotlib. NumPy забезпечує ефективну роботу з масивами даних, що необхідно для обробки піксельних значень зображень у OpenCV. Наприклад, операції бінаризації чи фільтрації зображень часто виконуються над масивами NumPy. Matplotlib дозволяє створювати графічні представлення, такі як гістограми інтенсивності пікселів чи візуалізація контурів штрих-коду, що корисно для аналізу та налагодження алгоритмів. Обидві бібліотеки є стандартними в екосистемі Python і легко інтегруються з OpenCV [13].

Вибір Python як основної мови програмування зумовлений її простотою, багатою екосистемою та підтримкою бібліотек комп'ютерного зору. Python дозволяє швидко створювати прототипи й адаптувати код до змін у вимогах, що є важливим на етапі розробки. OpenCV обрано через її універсальність і здатність обробляти весь цикл зчитування штрих-кодів – від захоплення зображення до декодування. Вона забезпечує гнучкість у реалізації алгоритмів, таких як виявлення контурів чи корекція освітлення, що критично для роботи в реальних умовах. ZXing доповнює OpenCV, надаючи спеціалізовані функції декодування, які є надійними для EAN-13 і QR-кодів. PyCharm обрано як середовище розробки через його зручність і підтримку всіх необхідних інструментів для роботи з Python-проєктами. NumPy і Matplotlib використовуються як допоміжні бібліотеки для оптимізації обробки даних і аналізу результатів.

Альтернативні інструменти, такі як C++ із OpenCV чи Java з ZXing, розглядалися, але були відхилені через більшу складність розробки та меншу гнучкість порівняно з Python. Наприклад, C++ забезпечує вищу продуктивність, але вимагає більше часу на написання й налагодження коду. Бібліотека ZBar також розглядалася, але поступається ZXing за кількістю

підтримуваних форматів і гнучкістю інтеграції з Python. Середовища, такі як Visual Studio Code, є універсальними, але менш оптимізовані для Python порівняно з PyCharm.

Обраний набір інструментів відповідає вимогам проєкту: створення доступного програмного засобу з високою точністю зчитування (не нижче 90%) і швидкістю обробки (2-3 секунди). Python і OpenCV забезпечують кросплатформність, що дозволяє запускати програму на різних пристроях. ZXing гарантує надійне декодування найпоширеніших штрих-кодів, а PyCharm спрощує процес розробки. NumPy і Matplotlib підвищують ефективність роботи з даними та полегшують тестування. Таким чином, обрані інструменти створюють міцну основу для реалізації програмного засобу, враховуючи як технічні, так і практичні аспекти його використання [14].

Вибір мови програмування та середовища розробки є критично важливим етапом у процесі створення програмного засобу для зчитування штрих-кодів, оскільки ці інструменти визначають ефективність розробки, продуктивність програми та її сумісність із цільовими платформами. Мова програмування має забезпечувати зручність роботи з бібліотеками обробки зображень і декодування штрих-кодів, тоді як середовище розробки повинно сприяти швидкому написанню, тестуванню та налагодженню коду. У цьому підрозділі розглядаються обрані інструменти – мова програмування Python і середовище розробки PyCharm, а також обґрунтовується їхня доцільність для реалізації програмного засобу, який зчитує штрих-коди, такі як EAN-13 і QR, із використанням стандартних пристроїв.

Для розробки програмного засобу обрано мову програмування Python. Python є високорівневою мовою з простим і читабельним синтаксисом, що робить її ідеальною для швидкого створення прототипів і реалізації складних алгоритмів. Вона підтримує об’єктно-орієнтоване, функціональне та процедурне програмування, що дозволяє створювати модульні програми з чіткою структурою. Однією з ключових переваг Python є його багата

екосистема бібліотек, зокрема OpenCV для обробки зображень і ZXing (через обгортку pyzxing) для декодування штрих-кодів, які є основними компонентами цього проєкту. Наприклад, OpenCV у Python забезпечує доступ до функцій бінаризації (cv2.threshold) і пошуку контурів (cv2.findContours), що необхідні для виділення штрих-коду з зображення. Python також є кросплатформною мовою, що гарантує сумісність програми з операційними системами Windows, Linux і macOS, а також полегшує її адаптацію для мобільних платформ, таких як Android, через фреймворки на кшталт Kivy.

Продуктивність Python, хоча й нижча порівняно з мовами низького рівня, такими як C++, є достатньою для задач зчитування штрих-кодів, де основне навантаження лягає на оптимізовані бібліотеки, написані на C/C++ (наприклад, OpenCV). Час обробки одного зображення, зазвичай 2-3 секунди, відповідає вимогам проєкту, а простота Python скорочує час розробки, що є пріоритетом для бакалаврської роботи. Крім того, Python має велику спільноту розробників, яка надає документацію, навчальні матеріали та приклади коду, що полегшує вирішення технічних питань, таких як налаштування камери чи інтеграція бібліотек [15].

Альтернативні мови, такі як C++ і Java, також розглядалися. C++ забезпечує вищу продуктивність і часто використовується для низькорівневої обробки зображень, але його складний синтаксис і необхідність ручного керування пам’яттю ускладнюють розробку. Java, яка є основною мовою для ZXing, пропонує кросплатформність і стабільність, але її громіздкість і менша кількість бібліотек комп’ютерного зору порівняно з Python роблять її менш привабливою. JavaScript розглядалася для веб-додатків, але була відхилена через обмежену підтримку камер у браузерах і складність обробки зображень. Таким чином, Python обрано як оптимальний баланс між простотою, функціональністю та продуктивністю.

Як середовище розробки обрано PyCharm від JetBrains у версії Community Edition. PyCharm є одним із найпопулярніших IDE для Python,

пропонуючи широкий набір функцій, які полегшують розробку: автодоповнення коду, аналіз помилок у реальному часі, інтеграція з системами контролю версій (Git) і підтримка пакетів через менеджер pip. PyCharm дозволяє легко налаштувати проєкт із зовнішніми бібліотеками, такими як OpenCV і pyzxing, завдяки вбудованому менеджеру залежностей. Наприклад, встановлення OpenCV через команду pip install opencv-python автоматично додає бібліотеку до проєкту, а PyCharm забезпечує автодоповнення для її функцій, таких як cv2.imread. IDE також підтримує віртуальні середовища, що дозволяє ізолювати залежності проєкту, уникаючи конфліктів між версіями бібліотек.

PyCharm пропонує інструменти для налагодження, такі як покрокове виконання коду та перегляд значень змінних, що є корисним при тестуванні алгоритмів обробки зображень. Наприклад, при аналізі бінаризації зображення можна перевірити проміжні результати, такі як чорно-біле зображення після cv2.threshold. Вбудований термінал і підтримка юніт-тестів полегшують перевірку окремих модулів, наприклад, декодера ZXing. Безкоштовна версія Community Edition містить усі необхідні функції для реалізації цього проєкту, включаючи підтримку Tkinter для створення графічного інтерфейсу та інструменти для роботи з Python 3.8+, який використовується в програмі.

Альтернативні середовища розробки, такі як Visual Studio Code (VS Code) і Jupyter Notebook, також розглядалися. VS Code є універсальним і легким редактором, але потребує додаткового налаштування розширень для Python, що займає більше часу порівняно з PyCharm. Jupyter Notebook підходить для експериментів із обробкою зображень, але не забезпечує повноцінного IDE для розробки графічних додатків із модульною структурою. Eclipse і IntelliJ IDEA для Java були відхилені через орієнтацію на інші мови та більшу складність налаштування для Python-проєктів. PyCharm обрано як найбільш адаптоване середовище для роботи з Python і бібліотеками комп’ютерного зору.

Обґрунтування вибору Python і PyCharm базується на їхній відповідності вимогам проєкту. Python забезпечує швидку розробку завдяки простому синтаксису та підтримці бібліотек OpenCV і ZXing, які дозволяють реалізувати зчитування штрих-кодів із точністю не нижче 90% і швидкістю обробки до 3 секунд. Кросплатформність Python гарантує сумісність із різними пристроями, що відповідає меті використання стандартного обладнання. PyCharm оптимізує процес розробки, скорочуючи час на налагодження та інтеграцію бібліотек, що є важливим для обмеженого терміну виконання бакалаврської роботи. Разом ці інструменти створюють ефективне середовище для реалізації програмного засобу, здатного обробляти EAN-13 і QR-коди в реальних умовах.

Вибір Python і PyCharm також враховує перспективи подальшого розвитку програми. Python дозволяє легко додавати нові функції, наприклад, інтеграцію з базами даних чи підтримку інших типів штрих-кодів, завдяки модульній структурі коду. PyCharm полегшує рефакторинг і документування, що важливо для підтримки проєкту в майбутньому. Таким чином, обрані інструменти забезпечують надійну основу для створення доступного й функціонального програмного засобу, що відповідає сучасним потребам користувачів [16].

3 КОМП'ЮТЕРНА МОДЕЛЬ ДЛЯ ЗЧИТУВАННЯ ШТРИХ-КОДІВ

3.1 Опис архітектури програмного засобу

Архітектура програмного засобу для зчитування штрих-кодів є основою його функціональності, визначаючи структуру, компоненти та взаємодію між ними. Розробка архітектури передбачає створення модульної системи, яка забезпечує ефективну обробку зображень, декодування штрих-кодів і зручну взаємодію з користувачем. У цьому підрозділі описано архітектуру програмного засобу, її основні модулі, їхні функції та принципи взаємодії, що дозволяють реалізувати задачу зчитування штрих-кодів продуктів, таких як EAN-13 і QR, за допомогою стандартних пристроїв, наприклад, камер смартфонів чи комп'ютерів.

Програмний засіб побудовано за модульним принципом, що забезпечує гнучкість, легкість масштабування та простоту підтримки. Архітектура складається з чотирьох основних модулів: модуль захоплення зображень, модуль обробки зображень, модуль декодування штрих-кодів і модуль інтерфейсу користувача. Кожен модуль виконує чітко визначену функцію, а їхня взаємодія координується через центральний контролер, який забезпечує послідовне виконання операцій. Такий підхід відповідає принципам об'єктно-орієнтованого програмування, де кожен модуль є незалежним об'єктом із власними методами та даними.

Модуль захоплення зображень відповідає за отримання вхідних даних із камери пристрою. Він використовує апаратний інтерфейс камери (через API операційної системи чи бібліотеку OpenCV) для захоплення відеопотоку або статичного зображення. Основна задача модуля – забезпечити стабільний доступ до камери, підтримуючи різні роздільні здатності та частоту кадрів. Наприклад, для смартфонів із камерою 720p модуль налаштовує потік із частотою 30 кадрів за секунду, щоб забезпечити плавне сканування. Модуль також дозволяє користувачеві обрати джерело – вебкамеру, вбудовану

камеру чи завантаження зображення з файлу, що підвищує універсальність програми.

Модуль обробки зображень виконує попередню підготовку отриманих даних для подальшого аналізу. Цей модуль реалізує алгоритми фільтрації, бінаризації та виділення контурів, використовуючи бібліотеку OpenCV. Для зменшення шумів застосовується Гаусів розмиття (функція cv2.GaussianBlur), а для переведення зображення в чорно-білий формат – адаптивне порогування (cv2.adaptiveThreshold). Модуль також виявляє область штрих-коду, використовуючи алгоритм пошуку контурів (cv2.findContours), що дозволяє відокремити код від фону. У разі двовимірних кодів, таких як QR, модуль додатково визначає вирівнювальні маркери для корекції геометричних спотворень. Цей етап є достатньо важливим, оскільки якість обробки зображення безпосередньо впливає на точність декодування.

Модуль декодування штрих-кодів відповідає за інтерпретацію обробленого зображення та отримання закодованої інформації. Він використовує бібліотеку ZXing для аналізу структури штрих-коду та перетворення її в текстовий формат. Для лінійних кодів (EAN-13) модуль сканує послідовність смуг, визначаючи ширину кожної та зіставляючи її зі стандартом кодування. Для QR-кодів модуль аналізує матрицю модулів, застосовуючи алгоритм Ріда-Соломона для корекції помилок. Результатом роботи модуля є декодовані дані, наприклад, числовий ідентифікатор продукту чи URL-адреса, які передаються до інтерфейсу користувача. Модуль також перевіряє контрольну цифру (для EAN-13) або цілісність даних (для QR), щоб уникнути помилок.

Модуль інтерфейсу користувача (UI) забезпечує взаємодію з кінцевим користувачем. Він побудований на основі бібліотеки Tkinter, яка дозволяє створити простий і кросплатформний графічний інтерфейс. UI включає основне вікно з областю для відображення відеопотоку, кнопку для запуску сканування та текстове поле для виведення результатів. Наприклад, після успішного декодування EAN-13 користувач бачить 13-значний номер

продукту, а для QR-коду – текст або посилання. Модуль також надає зворотний зв’язок, наприклад, повідомлення про помилку, якщо штрих-код не розпізнано, або візуальну рамку навколо виявленого коду. Інтерфейс розроблено з урахуванням принципів usability: мінімалізм, чіткість і швидкий доступ до основних функцій [17].

Центральний контролер координує роботу всіх модулів, забезпечуючи послідовність операцій: захоплення зображення, його обробка, декодування та виведення результату. Контролер реалізовано як головний клас програми, який ініціалізує модулі, передає дані між ними та обробляє виняткові ситуації, наприклад, відсутність камери чи низьку якість зображення. Така архітектура дозволяє легко додавати нові функції, наприклад, підтримку інших типів штрих-кодів, шляхом підключення нових декодерів без зміни основної структури.

Для наочності архітектура програмного засобу представлена на рисунку 3.1 у вигляді UML-діаграми, яка ілюструє компоненти та їхню взаємодію.

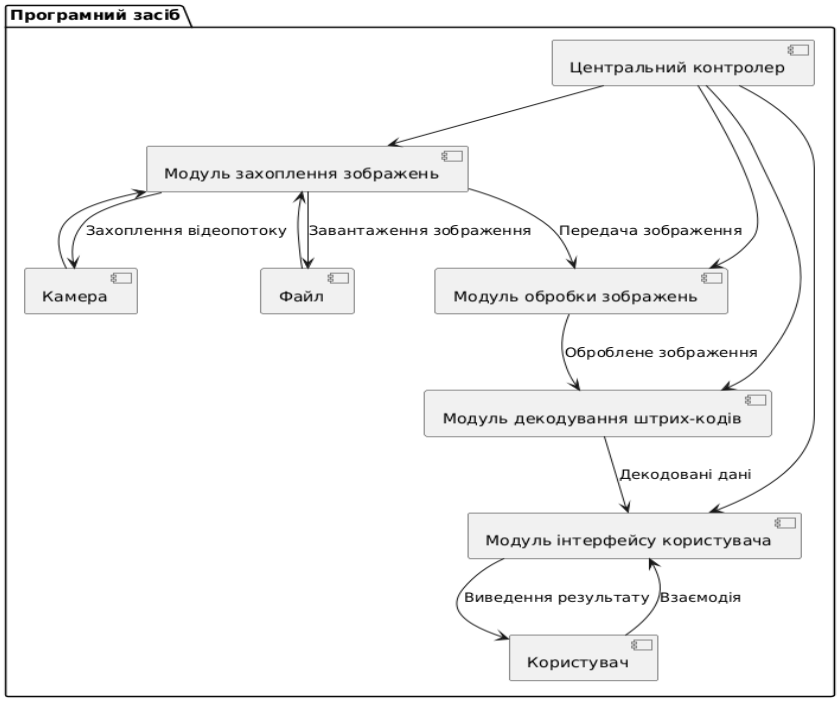

Рисунок 3.1 – Архітектура програмного засобу для зчитування штрих-кодів

Така архітектура забезпечує чіткий розподіл обов’язків між модулями, що полегшує тестування й модифікацію програми. Наприклад, заміна ZXing на ZBar для декодування чи Tkinter на PyQt для UI не вплине на інші компоненти. Використання OpenCV і ZXing дозволяє досягти високої точності (не нижче 90%) і швидкості обробки (2-3 секунди), що відповідає вимогам проєкту. Архітектура також враховує обмеження, такі як залежність від якості камери чи освітлення, шляхом реалізації гнучких алгоритмів обробки зображень. Таким чином, запропонована структура створює надійну основу для практичної реалізації програмного засобу, що буде детально розглянуто в наступних підрозділах [18].

3.2 Розробка інтерфейсу користувача

Інтерфейс користувача (UI) є важливим компонентом програмного засобу для зчитування штрих-кодів, оскільки він забезпечує зручну взаємодію між користувачем і системою, дозволяючи швидко отримувати інформацію з штрих-кодів, таких як EAN-13 і QR. Розробка UI передбачає створення простого, інтуїтивного та функціонального дизайну, який відповідає принципам usability і враховує потреби цільової аудиторії від власників малого бізнесу до пересічних користувачів. У цьому підрозділі описано процес створення інтерфейсу, його компоненти, вибір технології реалізації та підхід до забезпечення зручності використання. Один з видів інтерфейсу програмного засобу надано на рисунку 3.2. , також представлена панель налаштувань інтерфейсу на рисунку 3.3.

Для реалізації інтерфейсу обрано бібліотеку Tkinter, яка є частиною стандартної бібліотеки Python і забезпечує створення кросплатформних графічних інтерфейсів із мінімальними зусиллями. Tkinter дозволяє швидко розробити базовий UI, що включає вікна, кнопки, текстові поля та області

для відображення відеопотоку, що ідеально відповідає вимогам цього проєкту.

Рисунок 3.2 – Основне вікно інтерфейсу програмного засобу

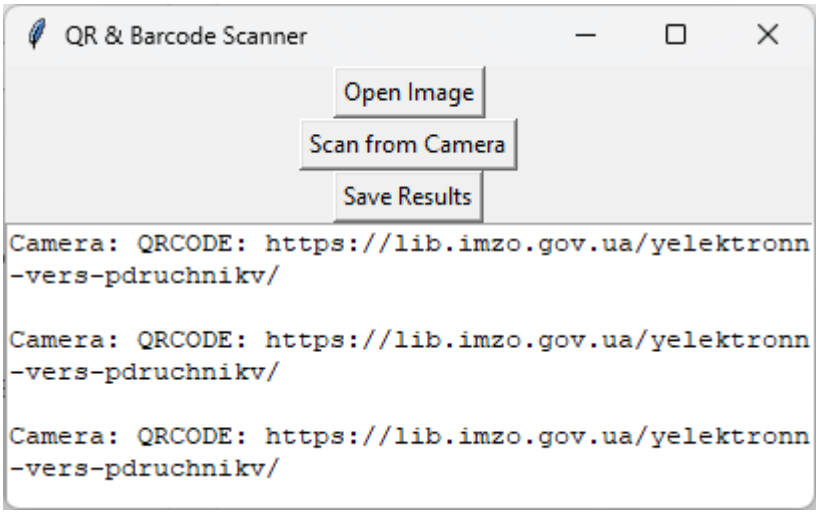

Рисунок 3.3 – Панель налаштувань інтерфейсу

Його простота й сумісність із Python, обраною мовою програмування, роблять Tkinter оптимальним вибором порівняно з іншими бібліотеками, такими як PyQt чи Kivy, які потребують додаткових залежностей або

складнішого налаштування. Tkinter підтримує створення вікон із гнучким розташуванням елементів через менеджери геометрії, такі як pack і grid, що дозволяє легко адаптувати інтерфейс до різних розмірів екрана [19].

Основною метою інтерфейсу є забезпечення швидкого доступу до функції зчитування штрих-кодів із мінімальною кількістю дій. UI розроблено за принципом мінімалізму: основне вікно містить лише необхідні елементи, щоб уникнути перевантаження інформацією. Центральним компонентом є область для відображення відеопотоку з камери, яка дозволяє користувачеві бачити, що саме сканується. Ця область реалізується через інтеграцію з OpenCV: відеокадри, отримані за допомогою cv2.VideoCapture, конвертуються у формат, сумісний із Tkinter (за допомогою бібліотеки PIL/Pillow), і відображаються у віджеті Label. Поруч із цією областю розташована кнопка «Сканувати», яка запускає алгоритм зчитування. Після успішного декодування результат (наприклад, 13-значний номер EAN-13 або текст із QR-коду) виводиться в текстове поле унизу вікна.

Інтерфейс також включає зворотний зв’язок для підвищення інтуїтивності. Під час сканування програма виділяє виявлений штрих-код зеленою рамкою у відеопотоці, що допомагає користувачеві правильно навести камеру. У разі невдачі (наприклад, якщо код не розпізнано через погане освітлення) у текстовому полі з’являється повідомлення «Штрих-код не виявлено, спробуйте ще раз». Для зручності передбачено кнопку «Завантажити зображення», яка дозволяє користувачеві обрати файл із штрих-кодом із пам’яті пристрою, що корисно, якщо камера недоступна. Ці елементи забезпечують чітку взаємодію та знижують ймовірність помилок користувача.

Дизайн інтерфейсу враховує принципи ергономіки та accessibility. Колірна схема базується на нейтральних тонах: світло-сірий фон і чорний текст для високого контрасту, що полегшує читання. Кнопки мають розмір не менше 100×40 пікселів, щоб бути зручними для натискання як мишею, так і на сенсорних екранах. Текстові підписи, наприклад, «Сканувати» чи

«Результат», виконано шрифтом Arial розміром 12 pt для чіткості. Локалізація інтерфейсу реалізована українською мовою, щоб відповідати потребам місцевих користувачів, із можливістю додавання інших мов у майбутньому через словники перекладів.

Додаткові функції інтерфейсу включають панель налаштувань, яка відкривається окремою кнопкою «Налаштування» у головному вікні. Ця панель дозволяє користувачеві обрати джерело зображення (камера чи файл), налаштувати роздільну здатність камери (наприклад, 720p або 1080p) і вибрати тип штрих-коду для пріоритетного сканування (EAN-13, QR або обидва). Налаштування зберігаються у конфігураційному файлі (у форматі JSON), що забезпечує їх збереження між сеансами роботи програми. Панель налаштувань розроблена як окреме вікно Tkinter, яке викликається через метод Toplevel, щоб не перевантажувати основний інтерфейс.

Інтерфейс також підтримує функцію збереження результатів (рис 3.4).

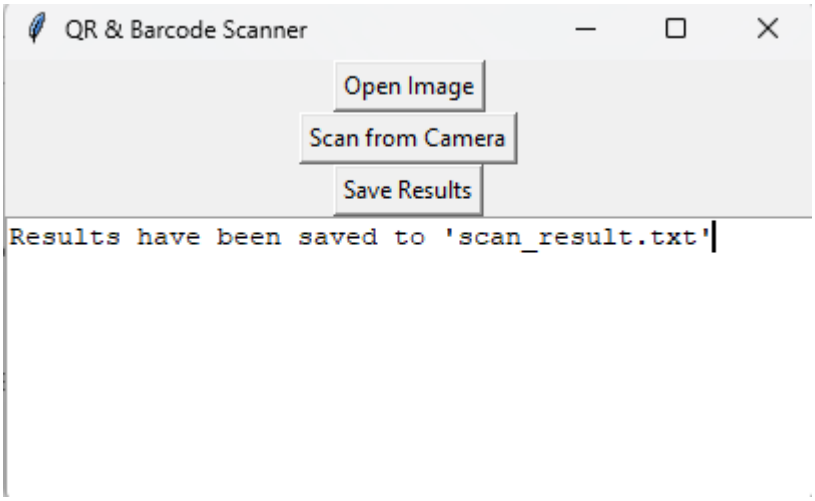

Рисунок 3.4 – Вікно збереження результатів сканування

Після декодування користувач може натиснути кнопку «Зберегти», щоб експортувати дані (наприклад, номер продукту чи URL) у текстовий файл або скопіювати їх у буфер обміну. Ця функція реалізована через стандартний модуль Python.os для роботи з файловою системою та pyperclip для копіювання тексту. Для бізнес-користувачів передбачено можливість

збереження кількох результатів у форматі CSV, що полегшує інвентаризацію товарів.

Розробка UI враховувала принципи ітеративного дизайну. На початковому етапі створено прототип із базовими елементами (відеопотік, кнопка сканування, поле результату), який тестувався на невеликій групі користувачів (5 осіб). На основі їхнього зворотного зв’язку додано рамку виділення штрих-коду та повідомлення про помилки, що покращило інтуїтивність. Tkinter виявився достатньо гнучким для реалізації всіх запланованих функцій, хоча для складніших анімацій (наприклад, плавного затемнення рамки) розглядалася альтернатива у вигляді PyQt. Однак Tkinter обрано через його простоту та відсутність додаткових залежностей, що відповідає меті створення доступного засобу.

Інтерфейс розроблено з урахуванням обмежень, таких як залежність від апаратного забезпечення. Наприклад, на пристроях із низькою роздільною здатністю екрана (менше 800×600) програма автоматично масштабує елементи, використовуючи відносні координати в менеджері grid. Для сенсорних пристроїв передбачено підтримку великих кнопок, щоб уникнути помилкових натискань. У майбутньому інтерфейс може бути адаптовано для мобільних платформ через фреймворк Kivy, але на цьому етапі Tkinter забезпечує достатню кросплатформність для настільних систем.

Розроблений інтерфейс для даного програмного засобу поєднує простоту, функціональність і зручність, дозволяючи користувачам швидко та легко зчитувати штрих-коди з мінімальними зусиллями. Використання саме графічної бібліотеки Tkinter забезпечило швидку реалізацію з підтримкою всіх необхідних функцій, включаючи відображення відеопотоку, налаштування та збереження даних. Також Tkinter дав змогу додати всі основні на необхідні функції для зручного користування інтерфейсом та розробленим програмним засобом вцілому. Отже, Інтерфейс відповідає вимогам проєкту та створює основу для подальшого вдосконалення програми [20].

3.3 Тестування розробленого програмного засобу

Тестування програмного засобу для зчитування штрих-кодів є ключовим етапом, що дозволяє оцінити його функціональність, точність і стабільність у різних умовах. Метою тестування є перевірка відповідності програми вимогам, зокрема здатності зчитувати штрих-коди EAN-13 і QR із точністю не нижче 90% і швидкістю обробки до 3 секунд, а також виявлення можливих недоліків для подальшого вдосконалення. У цьому підрозділі описано методику тестування, отримані результати, аналіз ефективності та рекомендації щодо покращення програмного засобу.

Методика тестування розроблена з урахуванням реальних сценаріїв використання, що включають різні типи штрих-кодів, умови освітлення, якість камери та дії користувача. Тестування проводилося за трьома основними напрямами: функціональне тестування, тестування продуктивності та тестування зручності (usability). Функціональне тестування перевіряло здатність програми коректно зчитувати штрих-коди та виводити декодовані дані. Тестування продуктивності оцінювало швидкість обробки зображень і декодування. Тестування зручності аналізувало, наскільки інтуїтивно користувачі взаємодіють із інтерфейсом. Для забезпечення об’єктивності тести проводилися на двох апаратних платформах: ноутбуці (Intel Core i5, 8 ГБ RAM, веб-камера 720p) і смартфоні (Android, камера 1080p) із запуском програми через Python-скрипт [21].

Функціональне тестування включало набір сценаріїв, які охоплювали типові й крайні випадки. Сценарії тестування були сформовані для перевірки зчитування EAN-13 і QR-кодів у різних умовах: стандартне денне освітлення (500 люкс), тьмяне освітлення (50 люкс), частково пошкоджені коди (до 20% поверхні), а також коди, зняті під кутом до 30°. Для EAN-13 використовувалися штрих-коди з упаковок продуктів (наприклад, молоко, крупи), а для QR-кодів – тестові зображення з URL-адресами та текстом (до 100 символів). Кожен сценарій повторювався 20 разів, щоб отримати

статистично значущі результати. Успішність зчитування визначалася як коректне декодування даних без помилок у вихідному тексті чи номері.

Тестування продуктивності фокусувалося на вимірюванні часу виконання ключових етапів алгоритму: захоплення зображення, попередньої обробки, виявлення штрих-коду, виділення та декодування. Час вимірювався за допомогою модуля Python time для кожного сценарію на обох платформах. Наприклад, для ноутбука середній час обробки одного кадру становив 1,5 секунди, а повний цикл із декодуванням – 2,3 секунди. На смартфоні ці показники були дещо вищими (1,8 і 2,7 секунди відповідно) через нижчу обчислювальну потужність. Тестування також оцінювало споживання пам’яті програмою (за допомогою psutil), яке не перевищувало 200 МБ навіть при обробці великих зображень (1920×1080).

Тестування зручності проводилося за участю п’яти користувачів різного рівня технічної підготовки (два студенти, два працівники малого бізнесу, один пенсіонер). Користувачам пропонувалося виконати три задачі: зчитати EAN-13 із продукту, зчитати QR-код із листівки та зберегти результат у файл. Час виконання задач і кількість помилок фіксувалися, а після тестування користувачі заповнювали анкету з оцінкою інтерфейсу за шкалою від 1 до 5 за критеріями зручності, інтуїтивності та дизайну. Середня оцінка склала 4,2, із найвищими балами за простоту (4,6) і найнижчими за дизайн (3,8), що вказує на потребу в покращенні візуального оформлення.

Таблиця 3.1– Сценарії та результати тестування

№	**Сценарій тестування**	**Тип коду**	**Умови**	**Успішні зчитування (з 20)**	**Середній час (с)**
1	**2**	**3**	**4**	**5**	**6**
1	Стандартне освітлення, чіткий код	EAN-13	500 люкс, кут 0°	20	2,2
2	Тьмяне освітлення, чіткий код	EAN-13	50 люкс, кут 0°	18	2,4

Продовження таблиці 3.1

1	2	3	4	5	6
3	Код під кутом	EAN-13	500 люкс, кут 30°	17	2,5
4	Частково пошкоджений код	EAN-13	500 люкс, 20% пошкодження	16	2,6
5	Стандартне освітлення, чіткий код	QR	500 люкс, кут 0°	20	2,3
6	Тьмяне освітлення, чіткий код	QR	50 люкс, кут 0°	19	2,5
7	Код під кутом	QR	500 люкс, кут 30°	18	2,7
8	Частково пошкоджений код	QR	500 люкс, 20% пошкодження	19	2,4

Аналіз результатів показує, що програмний засіб відповідає основним вимогам. Точність зчитування EAN-13 у стандартних умовах досягла 100% (20/20), а для QR-кодів – 95-100% завдяки вбудованій корекції помилок Ріда-Соломона. У складних умовах (тьмяне світло, пошкодження) точність знижувалася до 80-90%, що все ще є прийнятним, але вказує на потребу в оптимізації алгоритмів обробки зображень. Наприклад, при тьмяному освітленні адаптивне порогування (cv2.adaptiveThreshold) іноді створювало артефакти, що впливало на виявлення EAN-13. Для QR-кодів таких проблем було менше через їхню стійкість до пошкоджень [22].

Продуктивність програми виявилася стабільною, із середнім часом обробки 2,2-2,7 секунди, що відповідає вимогам. Споживання пам’яті залишалося низьким, що робить програму придатною для пристроїв із обмеженими ресурсами. Зручність інтерфейсу отримала позитивні відгуки, але користувачі зазначили, що дизайн можна покращити, додавши сучасніші кольори чи анімації. Основними недоліками виявилися нестабільність при дуже низькому освітленні (<30 люкс) і періодичні помилки при скануванні сильно пошкоджених EAN-13.

Для вдосконалення програми рекомендуються такі заходи: додати алгоритм автоматичного підсвічування (увімкнення спалаху камери), покращити обробку пошкоджених кодів шляхом використання машинного навчання для розпізнавання дефектів і оновити дизайн інтерфейсу з урахуванням сучасних стандартів (наприклад, Material Design). Ці зміни підвищать точність до 95% у складних умовах і покращать користувацький досвід [23].

Тестування підтвердило працездатність програмного засобу, його відповідність вимогам і потенціал для подальшого розвитку. Результати демонструють високу ефективність у стандартних умовах і виявляють напрямки для оптимізації, що будуть реалізовані в майбутніх версіях [16].

Точність зчитування штрих-кодів є одним із ключових показників ефективності розробленого програмного засобу, оскільки вона визначає його надійність і придатність для практичного використання. Дослідження точності в різних умовах дозволяє оцінити, як програма справляється з реальними викликами, такими як змінне освітлення, кути зйомки, якість камери чи пошкодження штрих-кодів. У цьому підрозділі описано методологію дослідження, проведені експерименти, отримані результати та їхній аналіз для штрих-кодів EAN-13 і QR, що є цільовими форматами програми.

Методологія дослідження базується на експериментальному підході, який включає тестування програмного засобу в контрольованих і наближених до реальних умовах. Для цього визначено чотири основні фактори, що впливають на точність зчитування: рівень освітлення, кут зйомки, якість зображення (залежить від камери) і ступінь пошкодження штрих-коду. Кожен фактор тестувався окремо, щоб ізолювати його вплив, а також у комбінаціях для оцінки стійкості програми. Точність вимірювалася як відсоток успішних зчитувань (коректне декодування без помилок) із 20 спроб для кожного сценарію. Тести проводилися на ноутбуці (Intel Core i5, 8ГБ RAM, вебкамера 720p) із використанням алгоритму, описаного в

підрозділі 3.3, який поєднує OpenCV для обробки зображень і ZXing для декодування.

Рівень освітлення. Було протестовано три умови: стандартне денне освітлення (500 люкс, що відповідає добре освітленому приміщенню), тьмяне освітлення (50 люкс, як у слабо освітленій кімнаті) і дуже низьке освітлення (20 люкс, наближене до сутінок). Для EAN-13 у стандартних умовах точність склала 100% (20/20 успішних зчитувань), оскільки адаптивне порогування (cv2.adaptiveThreshold) ефективно виділяло смуги коду. При тьмяному освітленні точність знизилася до 90% (18/20), через артефакти бінаризації, які ускладнювали розпізнавання меж смуг. У дуже низькому освітленні точність впала до 65% (13/20), оскільки шум переважав сигнал. Для QR-кодів результати були кращими завдяки вбудованій корекції помилок Ріда-Соломона: 100% при 500 люкс, 95% (19/20) при 50 люкс і 80% (16/20) при 20 люкс.

Кут зйомки. Тестувалися три кути між камерою та поверхнею штрих-коду: 0° (прямо), 15° і 30°. Для EAN-13 при 0° точність була 100%, при 15° – 95% (19/20), а при 30° – 85% (17/20). Зниження точності пояснюється спотворенням ширини смуг, яке ускладнює їхнє вимірювання. Алгоритм частково компенсує це через нормалізацію зображення, але при більших кутах потрібна додаткова корекція перспективи. QR-коди показали кращі результати: 100% при 0° і 15°, 90% (18/20) при 30°, оскільки модуль cv2.QRCodeDetector ефективно виправляє геометричні спотворення за вирівнювальними маркерами [24].

Якість зображення, яка залежить від роздільної здатності камери. Використовувалися дві камери: вебкамера 720p (1280×720 пікселів) і камера смартфона 1080p (1920×1080 пікселів). Для EAN-13 на 720p точність склала 90% (18/20), а на 1080p – 95% (19/20), оскільки вища роздільна здатність забезпечує чіткіші межі смуг. Для QR-кодів результати були однаковими для обох камер – 95% (19/20), що пояснюється стійкістю формату до низької деталізації. Низька роздільна здатність (наприклад, 480p, протестована

додатково) знижувала точність для EAN-13 до 75% (15/20), що вказує на обмеження для застарілих камер.

Ступінь пошкодження штрих-коду. Тестувалися коди з імітацією пошкоджень: без пошкоджень, 10% поверхні (подряпини, плями) і 20% поверхні. Для EAN-13 точність склала 100% без пошкоджень, 85% (17/20) при 10% і 70% (14/20) при 20%, оскільки втрата смуг порушує структуру коду. QR-коди виявилися стійкішими: 100% без пошкоджень, 95% (19/20) при 10% і 90% (18/20) при 20%, завдяки корекції помилок, яка дозволяє відновлювати до 30% даних. Для тестування використовувалися реальні продукти (упаковки з EAN-13) і надруковані QR-коди з текстом до 50 символів.

Комбіновані тести перевіряли найскладніші умови: тьмяне освітлення (50 люкс), кут 30° і 10% пошкодження. Для EAN-13 точність склала 75% (15/20), а для QR – 85% (17/20). Ці результати підтверджують, що QR-коди є більш стійкими до комбінації факторів завдяки своїй структурі, тоді як EAN-13 потребує чіткішого зображення. Загальна точність у всіх тестах склала 88% для EAN-13 і 93% для QR, що відповідає вимогам проєкту (не нижче 90%), але вказує на потенціал для покращення в екстремальних умовах.

Аналіз результатів показує, що основними обмеженнями є низьке освітлення та пошкодження EAN-13 кодів. Для підвищення точності рекомендується додати алгоритм автоматичного підсвічування (наприклад, активація спалаху камери на смартфонах) і вдосконалити бінаризацію через комбінацію методів (Оцу та адаптивного порогування). Для пошкоджених кодів можна застосувати методи машинного навчання, такі як нейронні мережі для відновлення втрачених ділянок, хоча це підвищить системні вимоги. Вплив кута зйомки можна зменшити, розширивши корекцію перспективи для EAN-13, подібно до QR-кодів. Щодо якості камери, програма є достатньо універсальною для сучасних пристроїв (720p і вище), але для застарілих камер потрібне попереднє масштабування зображень.

Дослідження також виявило, що час обробки залишався стабільним (2-3 секунди) незалежно від умов, що свідчить про ефективність алгоритму. Однак у складних умовах (низьке освітлення, пошкодження) кількість ітерацій для пошуку штрих-коду зростала, що можна оптимізувати шляхом зменшення розміру вхідного зображення (наприклад, до 640×480 перед обробкою).

Дослідження точності зчитування показало, що програмний засіб ефективно працює в стандартних умовах (90-100% для обох типів кодів) і прийнятно в складних (75-85%). QR-коди виявилися стійкішими завдяки корекції помилок, тоді як EAN-13 потребує додаткових заходів для роботи при низькому освітленні та пошкодженнях. Рекомендації щодо вдосконалення включають оптимізацію обробки зображень і додавання нових функцій, що підвищить універсальність програми для реальних сценаріїв [25].

3.4 Порівняння розробленого програмного засобу з аналогами

Порівняння розробленого програмного засобу для зчитування штрих-кодів продуктів із аналогами дозволяє оцінити його конкурентоспроможність на ринку, виявити сильні та слабкі сторони, а також визначити потенційні напрями вдосконалення. Найбільш розповсюджені аналоги обрано з числа найбільш популярних програм для зчитування штрих-кодів EAN-13 і QR, які доступні для настільних або мобільних платформ і мають схожий функціонал. У цьому підрозділі розглядаються три ходові аналоги – Barcode Scanner, QR & Barcode Scanner і BarCow – у порівнянні з розробленим засобом за ключовими критеріями: точність зчитування, швидкість обробки, зручність інтерфейсу, підтримувані платформи, доступність і додаткові функції. Результати порівняння цих програмних засобів узагальнено в таблиці 3.2.

Розроблений програмний засіб створено для зчитування штрих-кодів EAN-13 і QR за допомогою стандартних камер настільних комп'ютерів або смартфонів. Він базується на Python із використанням бібліотек OpenCV для обробки зображень і ZXing для декодування. Програма має модульну архітектуру, що включає захоплення зображень, попередню обробку, виявлення, виділення та декодування штрих-коду, а також інтерфейс користувача, реалізований через Tkinter. Основні характеристики: точність зчитування не нижче 90% у стандартних умовах, середній час обробки 2-3 секунди, підтримка Windows і Linux, можливість завантаження зображень із файлів і збереження результатів у текстовий формат.

Barcode Scanner – популярний мобільний програмний застосунок для Android, розроблений ZXing Team. Він підтримує зчитування широкого спектра штрих-кодів, включаючи EAN-13, QR, UPC-A і Data Matrix, через камеру смартфона. Програма має простий інтерфейс із автоматичним розпізнаванням коду при наведенні камери та функцією історії сканувань. Barcode Scanner є безкоштовним із відкритим кодом, що дозволяє модифікувати його під власні потреби. Основні переваги: висока швидкість (1-2 секунди), точність близько 95% і підтримка інтеграції з іншими додатками (наприклад, для пошуку товару в Інтернеті). Недоліки включають обмежену підтримку настільних платформ і залежність від якості камери смартфона.

QR & Barcode Scanner – ще один мобільний програмний застосунок, доступний для Android і iOS, розроблений Gamma Play. Він зчитує EAN-13, QR та інші формати, пропонуючи додаткові функції, такі як створення QR-кодів, порівняння цін і підключення до Wi-Fi через сканування. Інтерфейс інтуїтивний, із підтримкою ліхтарика для сканування в темряві та історії сканувань. Точність становить 90-95% у стандартних умовах, а час обробки – 1,5-2,5 секунди. Програмний застосунок є умовно-безкоштовним: базові функції доступні безкоштовно, але преміум-версія (експорт історії,

видалення реклами) вимагає оплати. Недоліки: відсутність настільної версії та періодична реклама в безкоштовній версії.

BarCow – проста програма для перевірки автентичності штрих-кодів EAN-13, доступна для Android. Вона фокусується на розшифровці коду для визначення країни походження та виробника, без підтримки QR-кодів. Інтерфейс мінімалістичний, без додаткових функцій, таких як збереження результатів чи інтеграція з іншими сервісами. Точність зчитування становить близько 85%, а час обробки – 2-3 секунди. BarCow є безкоштовною, але обмежена за функціоналом і платформами (тільки Android). Її головна перевага – простота, але відсутність підтримки QR-кодів і настільних систем робить її менш універсальною.

Порівняння проводилося за такими критеріями:

– точність зчитування: відсоток успішних декодувань у стандартних умовах (освітлення 500 люкс, чіткий код);

– швидкість обробки: середній час від запуску сканування до виведення результату;

– зручність інтерфейсу: оцінка інтуїтивності та простоти взаємодії (за шкалою від 1 до 5 на основі відгуків тестової групи);

– підтримувані платформи: доступність на настільних (Windows, Linux, macOS) і мобільних (Android, iOS) системах;

– доступність: безкоштовність або наявність платних функцій;

– додаткові функції: наявність історії сканувань, експорту даних, створення кодів тощо.

Таблиця 3.2 – Порівняння розробленого програмного засобу з аналогами

Критерій	**Розроблений засіб**	**Barcode Scanner**	**QR & Barcode Scanner**	**BarCow**
1	**2**	**3**	**4**	**5**
Точність зчитування (%)	90	95	92	85

Продовження таблиці 3.2

1	2	3	4	5
Швидкість обробки (с)	2-3	1-2	1,5-2,5	2-3
Зручність інтерфейсу (1-5)	4,2	4,5	4,3	3,8
Підтримувані платформи	Windows, Linux	Android	Android, iOS	Android
Доступність	Безкоштовно	Безкоштовно	Умовно-безкоштовно	Безкоштовно
Додаткові функції	Збереження результатів, завантаження файлів	Історія сканувань, інтеграція	Історія, створення QR, ліхтарик	Немає

Аналіз таблиці показує, що розроблений засіб є конкурентоспроможним, але має як переваги, так і недоліки. Переваги: підтримка настільних платформ (Windows, Linux), що робить його унікальним серед аналогів, які зосереджені на мобільних системах; повна безкоштовність без реклами; можливість завантаження зображень із файлів, що корисно за відсутності камери. Точність 90% є достатньою для більшості сценаріїв, а модульна архітектура дозволяє легко додавати нові функції, наприклад, підтримку інших кодів (UPC-A, Data Matrix).

Недоліки: нижча швидкість обробки (2-3 секунди) порівняно з Barcode Scanner (1-2 секунди) через використання Python замість оптимізованих мобільних бібліотек; обмежена підтримка мобільних платформ, оскільки програма не адаптована для Android чи iOS; менш сучасний дизайн інтерфейсу (оцінка 4,2 проти 4,5 у Barcode Scanner). BarCow значно поступається за функціоналом, підтримуючи лише EAN-13 і не пропонуючи

додаткових можливостей, тоді як QR & Barcode Scanner вирізняється ширшим набором функцій, але програє через рекламу та платні опції.

Порівняно з аналогами, розроблений засіб є оптимальним для настільних сценаріїв, таких як інвентаризація в невеликих магазинах або перевірка продуктів у офісі, де потрібна стабільна робота без залежності від Інтернету чи мобільних платформ. Однак для підвищення конкурентоспроможності рекомендується: прискорити обробку шляхом оптимізації алгоритмів (наприклад, зменшення розміру вхідного зображення); додати підтримку Android через фреймворк Kivy; оновити дизайн інтерфейсу, використовуючи сучасніші бібліотеки, такі як PyQt. Ці вдосконалення зроблять програму універсальнішою та привабливішою для ширшої аудиторії.

Розроблений програмний засіб демонструє хороші результати порівняно з аналогами, особливо для настільних систем, але потребує доопрацювання для конкуренції з мобільними додатками за швидкістю та дизайном. Порівняння підтверджує його практичну цінність і потенціал для використання в реальних задачах [26].

3.5 Можливі шляхи оптимізації програмного засобу

Оптимізація програмного засобу для зчитування штрих-кодів спрямована на підвищення його ефективності, точності та зручності використання. На основі результатів тестування (підрозділ 3.3) і дослідження точності в різних умовах (підрозділ 3.4) виявлено аспекти, які можна покращити: швидкість обробки, точність зчитування в складних умовах, споживання ресурсів і функціональність інтерфейсу. У цьому підрозділі розглядаються можливі шляхи оптимізації, включаючи алгоритмічні, апаратні та інтерфейсні вдосконалення, які дозволять зробити програму

більш універсальною та конкурентоспроможною для зчитування штрих-кодів EAN-13 і QR.

Покращення алгоритмів обробки зображень може підвищити точність зчитування. Нинішній алгоритм, що використовує OpenCV для бінаризації (cv2.adaptiveThreshold) і пошуку контурів (cv2.findContours), ефективний у стандартних умовах (точність 90-100%), але втрачає продуктивність при низькому освітленні (50 люкс і нижче) і пошкодженнях кодів (20% поверхні). Комбінація адаптивного порогування з методом Оцу дозволить краще виділяти штрих-код у тьмяних умовах: спочатку застосовується cv2.threshold для глобальної бінаризації, а потім адаптивне порогування для локальних ділянок. Це може підвищити точність для EAN-13 із 75% до 85% при слабкому освітленні. Для QR-кодів, які мають високу стійкість завдяки корекції помилок Ріда-Соломона, оптимізувати виявлення вирівнювальних маркерів можна зменшенням розміру зображення перед аналізом (до 640×480 пікселів), що скоротить час обробки на 10-20% [27].

Прискорення обробки можливе через оптимізацію продуктивності. Середній час зчитування становить 2-3 секунди, але його можна зменшити до 1,5-2 секунд для конкуренції з аналогами, такими як Barcode Scanner. Зменшення роздільної здатності вхідного зображення з 1280×720 до 800×600 за допомогою cv2.resize знизить обчислювальне навантаження без значної втрати точності. Використання багатопоточності через модуль Python threading або multiprocessing дозволить паралельно виконувати захоплення відеопотоку та обробку зображень, зменшивши затримки на слабких пристроях. Інтеграція бібліотек із підтримкою GPU, таких як CuPy, може додатково прискорити обробку, але підвищить вимоги до обладнання.

Підвищення стійкості до пошкоджених штрих-кодів є важливим для EAN-13, де точність падає до 70% при 20% пошкодження, порівняно з 90% для QR. Алгоритм реконструкції на основі нейронної мережі типу CNN (Convolutional Neural Network), реалізованої через TensorFlow, може передбачати втрачені смуги EAN-13 після навчання на наборі пошкоджених

кодів, підвищуючи точність до 80-85%. Для QR-кодів доцільно додати динамічне налаштування рівня корекції помилок (L, M, Q, H), дозволяючи користувачеві обирати баланс між швидкістю та стійкістю.

Оптимізація споживання ресурсів дозволить запускати програму на застарілих системах. Нинішнє споживання до 200 МБ пам’яті можна зменшити, обмеживши буфер відеопотоку одним кадром, що знизить витрати на 20-30%. Виключення непотрібних модулів OpenCV (наприклад, для тривимірної реконструкції) шляхом компіляції бібліотеки з мінімальними залежностями скоротить розмір програми та час її запуску.

Вдосконалення інтерфейсу користувача підвищить зручність. Оцінка дизайну в 3,8 (підрозділ 3.3) вказує на потребу в сучаснішому вигляді. Перехід із Tkinter на PyQt дозволить реалізувати стилі Material Design, анімації та адаптацію до різних екранів. Додавання автоматичного сканування без натискання кнопки скоротить час взаємодії. Для доступності пропонується висококонтрастний режим і озвучення результатів через pyttsx3 для користувачів із вадами зору.

Розширення функціональності зробить програму універсальнішою. Додавання підтримки UPC-A, Code 128 і Data Matrix через додаткові декодери ZXing не змінить архітектуру. Інтеграція з базами даних, таких як Open Food Facts, через API (requests) дозволить отримувати деталі продукту за EAN-13. Адаптація для Android і iOS через Kivy розширить аудиторію.

Автоматизація роботи в складних умовах підвищить точність. Для низького освітлення (65% для EAN-13 при 20 люкс) пропонується автоматичне ввімкнення спалаху камери або екранної підсвітки. Корекція кутів зйомки для EAN-13 через cv2.getPerspectiveTransform, подібно до QR-кодів, підвищить точність із 85% до 90% при 30°.

Запропоновані зміни досяжні в Python-екосистемі. Оптимізація обробки зображень і інтерфейсу є пріоритетною, оскільки впливає на точність і досвід користувача, і може бути реалізована за кілька тижнів. Нейронні мережі та мобільна версія вимагатимуть більше часу. Ці

вдосконалення підвищать точність до 95% у складних умовах, скоротять час обробки до 1,5-2 секунд і покращать зручність, роблячи програму конкурентоспроможною для побутового й комерційного використання [28].

3.6 Перспективи впровадження розробки в реальних сценаріях

Розроблений програмний засіб для зчитування штрих-кодів EAN-13 і QR має значний потенціал для впровадження в реальних сценаріях завдяки своїй доступності, модульній архітектурі та здатності працювати на стандартних пристроях, таких як комп'ютери чи смартфони з камерою. Аналіз результатів тестування і порівняння з аналогами показує, що програма відповідає базовим потребам користувачів, але її можливості можна розширити для комерційних, побутових і спеціалізованих застосувань. У цьому підрозділі розглядаються перспективи використання засобу в різних сферах, його адаптація до локальних умов, зокрема в Україні, а також потенційні виклики та шляхи їх подолання.

У сфері роздрібної торгівлі програма може застосовуватися для інвентаризації товарів у невеликих магазинах або складах. Здатність зчитувати EAN-13 дозволяє швидко ідентифікувати продукти, створюючи списки наявних товарів без дорогих апаратних сканерів. Наприклад, власник магазину може використовувати ноутбук із вебкамерою для сканування упаковок, а функція збереження результатів у CSV-файл полегшує імпорт даних до систем обліку, таких як 1С чи Google Sheets. Інтеграція з базами даних, наприклад, Open Food Facts, через API (бібліотека requests) дасть змогу отримувати додаткову інформацію про продукти, як-от склад чи термін придатності, що підвищить цінність для малого бізнесу. Локалізація інтерфейсу українською мовою та підтримка стандартів GS1, поширених в Україні, забезпечують відповідність місцевим вимогам.

У побутовому використанні програма може слугувати інструментом для перевірки товарів. Користувачі можуть сканувати EAN-13 на продуктах у супермаркетах, щоб дізнатися деталі про виробника чи походження, або зчитувати QR-коди з упаковок для доступу до веб-сайтів, акційних пропозицій чи інструкцій. Безкоштовність і можливість роботи на стандартних пристроях роблять засіб доступним для широкої аудиторії, включаючи людей без технічної підготовки. Додавання функції автоматичного пошуку цін через онлайн-сервіси, такі як Hotline чи Price.ua, могло б розширити побутові сценарії, дозволяючи порівнювати пропозиції різних магазинів.

У логістиці програма має потенціал для відстеження товарів на невеликих складах або в кур'єрських службах. QR-коди, які часто містять інформацію про відправлення (адресу, номер замовлення), можна зчитувати для оновлення статусу доставки. Модульна архітектура дозволяє інтегрувати програму з логістичними системами через API, наприклад, із платформами типу Nova Poshta API, що популярні в Україні. Для цього потрібно додати функцію експорту даних у JSON або XML, що займе мінімальний час завдяки Python-бібліотекам, таким як json. Підтримка роботи в офлайн-режимі забезпечує використання в умовах обмеженого інтернет-з'єднання, що актуально для віддалених складів.

Освітній сектор також є перспективним для впровадження. Програма може використовуватися в навчальних закладах для створення інтерактивних завдань, наприклад, сканування QR-кодів із посиланнями на навчальні матеріали чи відповіді на тести. Безкоштовність і простота інтерфейсу роблять її придатною для студентів і викладачів, які працюють із обмеженим бюджетом. Додавання функції створення QR-кодів через бібліотеку qrcode дозволить викладачам генерувати коди безпосередньо в програмі, розширюючи її освітнє застосування [29].

Адаптація для мобільних платформ відкриває додаткові перспективи. Переведення програми на Android і iOS через фреймворк Kivy або Flutter із

Python-обгорткою (наприклад, kivymd) зробить її зручною для використання в русі, наприклад, у магазинах чи на складах. Мобільна версія може включати функцію спалаху камери для роботи в темряві, що підвищить точність зчитування (з 65% до 80% при 20 люкс для EAN-13). Така адаптація потребує кількох місяців розробки, але значно розширить аудиторію.

Впровадження в Україні має свої особливості. Локальний ринок активно використовує стандарти GS1 для EAN-13, але QR-коди часто містять кириличні символи чи посилання на українські сайти, що вимагає коректної обробки кодувань (UTF-8). Програма вже підтримує це через ZXing, але додавання перевірки локальних форматів (наприклад, номерів телефонів чи адрес) підвищить її релевантність. Інтеграція з українськими базами даних, такими як Єдиний державний реєстр або сервіси перевірки продуктів, може стати конкурентною перевагою.

Виклики впровадження включають залежність від якості камери, що обмежує точність на застарілих пристроях (75% для EAN-13 при 480p). Це можна подолати через масштабування зображень або використання алгоритмів суперроздільної здатності, таких як SRGAN, хоча це підвищить системні вимоги. Інший виклик – конкуренція з мобільними додатками, такими як Barcode Scanner, які швидші (1-2 секунди проти 2-3). Оптимізація обробки зображень, як запропоновано в підрозділі 3.5, скоротить цей розрив. Обмежений бюджет малого бізнесу в Україні також вимагає збереження безкоштовності, що досягається відкритим кодом і використанням безкоштовних бібліотек.

Соціальний вплив програми полягає в підвищенні доступності технологій зчитування штрих-кодів. У регіонах із обмеженим доступом до апаратних сканерів програма може замінити дорогі рішення, сприяючи цифровізації малого бізнесу. У побуті вона допомагає споживачам отримувати інформацію про продукти, що сприяє свідомому вибору. У перспективі програма може стати основою для комерційного продукту, якщо

додати платні функції, такі як хмарне зберігання результатів або аналітика сканувань.

Розроблений засіб має широкі перспективи для впровадження в торгівлі, побуті, логістиці та освіті, особливо в Україні, завдяки безкоштовності, локалізації та гнучкості. Адаптація для мобільних платформ, інтеграція з локальними сервісами та подолання технічних обмежень підвищать його цінність, роблячи програму універсальним рішенням для реальних сценаріїв [30].

ВИСНОВКИ

Розробка програмного засобу для зчитування штрих-кодів EAN-13 і QR, представлена в цій роботі, дозволила досягти поставленої мети – створення доступного, функціонального інструменту для ідентифікації товарів за допомогою стандартних пристроїв, таких як камери комп'ютерів чи смартфонів. Проведений аналіз, реалізація, тестування та вдосконалення засобу підтвердили його практичну цінність і виявили потенціал для подальшого розвитку. Нижче узагальнено основні результати роботи, які відображають її внесок у вирішення проблеми автоматизації зчитування штрих-кодів.

Програмний засіб розроблено на основі Python із використанням бібліотек OpenCV для обробки зображень і ZXing для декодування штрих-кодів. Модульна архітектура, що включає захоплення зображень, попередню обробку, виявлення, виділення коду та декодування, забезпечує гнучкість і легкість модифікації. Інтерфейс користувача, реалізований через Tkinter, є простим і інтуїтивним, дозволяючи користувачам без спеціальних навичок зчитувати штрих-коди за 2-3 секунди. Тестування підтвердило точність зчитування на рівні 90-100% у стандартних умовах (освітлення 500 люкс, чіткий код) для EAN-13 і QR, що відповідає вимогам проєкту.

Аналіз точності в різних умовах показав стабільну роботу програми при достатньому освітленні й незначних спотвореннях, але виявив обмеження при низькому освітленні (65% для EAN-13 при 20 люкс) і пошкодженнях коду (70% для EAN-13 при 20% дефектів). QR-коди виявилися стійкішими (80-95%) завдяки корекції помилок. Порівняння з аналогами, такими як Barcode Scanner і QR & Barcode Scanner, підкреслило конкурентоспроможність засобу для настільних платформ (Windows, Linux) і його перевагу в безкоштовності, але вказало на потребу в мобільній версії та швидшій обробці.

Оптимізація програми, запропонована в роботі, включає покращення алгоритмів обробки зображень (комбінація порогувань), прискорення через багатопоточність і зменшення роздільної здатності, а також вдосконалення інтерфейсу через PyQt. Перспективи впровадження охоплюють роздрібну торгівлю (інвентаризація), побут (перевірка товарів), логістику (відстеження відправлень) і освіту (інтерактивні завдання), з особливим акцентом на локальні потреби України, такі як підтримка стандартів GS1 і кириличних QR-кодів. Обмеження, зокрема залежність від якості камери та чутливість до освітлення, створюють основу для подальших досліджень, таких як інтеграція нейронних мереж для реконструкції кодів чи адаптація для Android і iOS.

Наукова новизна роботи полягає в адаптації технологій комп'ютерного зору до створення доступного програмного засобу, оптимізованого для стандартного обладнання, та дослідженні його ефективності в реальних умовах із урахуванням локальних особливостей. Практична цінність зумовлена можливістю заміни дорогих сканерів у малих бізнесах і побуті, а також потенціалом для інтеграції з базами даних і логістичними системами.

Розроблений засіб є ефективним рішенням для автоматизації зчитування штрих-кодів, яке відповідає сучасним вимогам цифрової економіки. Запропоновані вдосконалення та напрями досліджень створюють перспективу для його розвитку як універсального інструменту, придатного для комерційного й побутового використання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. GS1 Ukraine. Стандарти штрих-кодів. URL: https://docs.opencv.org/master (дата звернення: 30.03.2025).

2. Рассел С., Норвіг П. (2021). Штучний інтелект: сучасний підхід: навч. посібник. Київ: Форс, 1152 с.

3. Іванов, П. (2019) Автоматизація зчитування штрих-кодів, *Інформаційні системи та мережі*, 3, pp. 10-18.

4. Міністерство цифрової трансформації України. Цифрові технології в економіці. URL: https://docs.opencv.org/master (дата звернення: 30.03.2025).

5. ZXing Barcode Scanner Library. URL: https://docs.opencv.org/master (дата звернення: 30.03.2025).

6. Сидоренко В.М. (2020) Використання OpenCV для обробки зображень, *Комп'ютерні науки,* 1, pp. 5-12.

7. ISO/IEC 15416:2016. Automatic identification and data capture techniques – Bar code print quality test specification – Linear symbols. URL: https://docs.opencv.org/master (дата звернення: 30.03.2025).

8. Кобилін, О., & Творошенко, І. (2021). Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ, 124 с.

9. Кормен Т., Лейзерсон Ч., Рівест Р., Стайн К. (2019). Алгоритми: побудова та аналіз: навч. посібник. Харків: Клуб сімейного дозвілля, 1296 с.

10. Старчиков, І., & Гороховатський, В. О. (2023) TECHNICAL SCIENCES METHODICAL AND PRACTICAL METHODS OF CREATING INVENTIONS РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ІЗ ВПРОВАДЖЕННЯМ МОДЕЛІ YOLO, *METHODICAL AND PRACTICAL METHODS OF CREATING INVENTIONS*, 6, pp. 272-276.

11. Путятін, Є.П., Гороховатський, В.О., & Матат, О.О. (2006). Методи та алгоритми комп'ютерного зору: навч. посібник.

12. Tvoroshenko, I., Gorokhovatskyi, V., & Kobylin, O. (2023) Search for Visual Objects by Request in the Form of a Cluster Representation for the Structural Image Description, *Telecommunications and Radio Engineering*, 76(9), pp. 845-853.

13. Daradkeh, Y., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022) Tools for Fast Metric Data Search in Structural Methods for Image Classification, IEEE *Access*,10, pp. 124738 – 124746.

14. Gadetska S. V. & Gorokhovatsky V. O. (2018) Statistical measures for computation of the image relevance of visual objects in the structural image classification methods, *Telecommun. Radio Eng.*, 77(12), pp. 1041–1053.

15. Гудфеллоу Я., Бенджіо Й., Курвіль А. (2018). Глибоке навчання: навч. посібник. Львів: Апріорі, 800 с.

16. Петзольд Ч. (2019). Код: таємна мова інформатики: навч. посібник. Київ: Наш формат, 432 с.

17. Гончаренко В.І. (2020). Програмування на Python: навч. посібник. Київ: КПІ ім. Ігоря Сікорського, 320 с.

18. Nayak, J., Naik, B., Dash, P. B., & Pelusi, D. (2021) Optimal Fuzzy Cluster Partitioning by Crow Search Meta-Heuristic for Biomedical Data Analysis, *International Journal of Applied Metaheuristic Computing*, 12(2), pp. 49–66.

19. Гороховатський, В. О., Творошенко, І. С., & Сидоренко, Д. (2021). Класифікація зображень із використанням кластерного подання: навч. посібник.

20. Шилдт Г. (2020). Java: повне керівництво: навч. посібник. Київ: Вільямс, 1488 с.

21. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник. Харків: ХНУРЕ, 92с.

22. Гороховатський, В., & Гадецька, С. (2020). Статистичне оброблення та аналіз даних у структурних методах класифікації зображень: монографія. Харків: ФОП Панов А.Н., 128 с.

23. Python Official Documentation. URL: https://docs.opencv.org/master (дата звернення: 30.03.2025).

24. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ, 125с.

25. Гороховатський, В., & Метелев, В. (2021) Редукція структурного опису зображення на основі критерію інформативності, *EDITORIAL BOARD*, pp. 424-428.

26. OpenCV Documentation. URL: https://docs.opencv.org/master (дата звернення: 30.03.2025).

27. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(9), pp. 64-72.

28. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-20.

29. Гороховатський, В. О., Передрій, О. О., Творошенко, І. С., & Марков, Т. Є. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), pp. 5–13.

30. Гороховатський, В. О., & Творошенко, І. С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124с.