НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Спосіб комбінації кольорів RGB для підвищення ефективності QR-кода»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІО-з11мп
Джевага Павло Дмитрович _____

Науковий керівник:
Доцент, к.т.н.
Русанова Ольга Веніамінівна _____

Консультант з назва розділу:
Проф. каф. ОТ, д.т.н.
Кулаков Юрій Олексійович _____

Рецензент:
Доцент, каф. ІСТ, к.т.н.
Шимкович Володимир Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – другий (магістерський)

Спеціальність *123. Комп'ютерна інженерія*

ОПП *123. Комп'ютерні системи та мережі*

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій Стіренко
(підпис) (ініціали, прізвище)
«____» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Джевага Павло Дмитрович
(прізвище, ім'я, по батькові)

1. Тема дисертації *Спосіб комбінації кольорів RGB для підвищення ефективності QR-кода*
Науковий керівник дисертації *доцент, к.т.н., Русанова О.В.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від *«08» листопада 2022* р. № *4097-с*

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження *процес генерації QR-коду використовуючи кольори RGB*

4. Предмет дослідження *методи компресії, мультиплексування, багатошаровості та спосіб підвищити ефективність QR-коду*

5. Перелік завдань, які потрібно розробити: *проаналізувати предметну область, детально ознайомитися з ключовими поняттями, провести пошук можливих існуючих рішень, визначити особливості існуючих систем або підходів (у разі відсутності схожих систем), провести детальний огляд основних етапів аналізу даних, визначити перелік технологій для розробки модифікованого методу для аналізу кольорового QR-коду з обґрунтуванням їх вибору, оформити пояснювальну записку за результатами проведеної роботи.*

6. Орієнтовний перелік графічного(ілюстративного) матеріалу *Графічні матеріали результатів дослідження надати у вигляді презентації у форматі .ppt*

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Кулаков Ю. О., проф., д.т.н.		
1	Русанова О.В., доц., к.т.н.		
2	Русанова О.В., доц., к.т.н.		
3	Русанова О.В., доц., к.т.н.		
4	Русанова О.В., доц., к.т.н.		

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1	Огляд предметної області	18.07.2022	
2	Підготовка вступної частини	25.07.2022	
3	Аналіз основних понять та пошук можливих існуючих рішень	08.08.2022	
4	Огляд основних етапів аналізу даних	05.09.2022	
5	Розроблення логіки способу аналізу енергоспоживання	19.09.2022	
6	Аналіз технологій для розробки програмної частини дисертації	23.09.2022	
7	Реалізація та тестування програмної частини дисертації	13.11.2022	
8	Оформлення пояснювальної записки	23.11.2022	
9	Захист дисертації	21.12.2022	

Студент _____ Джевага П.Д.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Русанова О.В.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Спосіб комбінації кольорів RGB для підвищення ефективності QR-кода

студентом: Джевагою Павлом Дмитровичем

Магістерська дисертація складається зі вступу, п’ятьох розділів та висновків. Загальний обсяг дисертації становить 89 аркушів основної частини, 34 ілюстрацій, 19 таблиць, 1 додаток на 14 аркушів. Під час написання роботи використовувались матеріали з 9 джерел.

Актуальність теми. Враховуючи стрімкий розвиток інформаційних технологій, в тому числі й технологій цифрових сховищ, створений модифікований алгоритм, який спрощуватиме робочий процес при збереженні даних є досить актуальним. На сьогоднішній день цей процес є переважно «ускладненим», а тому займає багато часу у користувачів, які займаються збереженням даних, що до того ж не є до кінця вивченою, є досить актуальним та корисним.

Мета і задачі дослідження. Метою даного дослідження є підвищення ємності даних кольорового QR-коду шляхом проектування та розробки алгоритмів компресії, мультиплексування та багатошаровості. Мета дослідження досягається наступними завданнями:

1. Розробити алгоритм кодування QR-коду кольору RGB з більшим сховищем даних.
2. Розробити алгоритм декодування QR-коду кольорів RGB, який відновлює всі збережені дані.
3. Розробити алгоритм часткового декодування та перекодування кольорового QR-коду RGB, який дозволяє частково витягти необхідні закодовані дані.

Об’єкт дослідження. Об’єктом дослідження для даної роботи є процес створення модифікованого алгоритму для генерація QR-коду.

Предмет дослідження. Предметом дослідження є спосіб підвищення ефективності QR-коду, який ґрунтується на різноманітних чинниках, що впливають на процес генерація, за допомогою якого створений RGB QR-коду, який ілюструватиме результати модифікованого методу та надаватиме практичні рекомендації щодо потенційного підвищення рівня ефективності QR-коду.

Методи дослідження. Для досягнення мети магістерської дисертації було використано мови програмування Python (для основної логіки), його фреймворками qrcode, pyzbar та PIL. За допомогою вищезгаданих технологій було модифікований метод для генерація RGB QR-коду.

Практичне значення одержаних результатів. Розроблена модель, як і власне спосіб генерація RGB QR-коду можуть бути використані різними типами користувачів, такими як:

1. Профільні підприємства, пов’язані з цифровими сховищами.

2. Інші підприємства, що слідкують за практичністю вільного місця.

3. Фізичні особи, що бажають дізнатися більше інформації стосовно збереження даних у нових типах сховищ, таких як QR-кодах.

Особистий внесок магістранта. Магістерська дисертація є самостійно виконаною роботою, що має в основі авторський підхід для досягнення мети дисертації. Отримано прикладні результати, що відносяться до розв’язання задачі способу підвищення ефективності QR-коду та розвинення його як покращений тип сховища. Формулювання мети та задач дослідження проводилось спільно з науковим керівником.

Ключові слова. QR-код, модифікований метод, використання RGB, сховище.

Пояснювальна записка

до магістерської дисертації

на тему: «Спосіб комбінації кольорів RGB для підвищення ефективності QR-кода»

Київ – 2022

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

QR-code – Quick Response code, код швидкого реагування, - матричний код (двовимірний штрих-код).

PM-code – це 2D-код нового покоління, який дозволяє зберігати цифрові дані у 3-вимірній структурі.

2D – це комп'ютерна генерація цифрових зображень - переважно з двовимірних моделей і за допомогою специфічних для них методів.

3D – це графіка, яка використовує тривимірне представлення геометричних даних, що зберігаються в комп'ютері з метою виконання обчислень і рендерингу цифрових зображень, зазвичай 2D-зображень, але іноді 3D-зображень.

RGB – це адитивна колірна модель, в якій червоний, зелений і синій основні кольори світла додаються разом різними способами для відтворення широкого спектру кольорів.

ASCII – Американський стандартний код для обміну інформацією, стандартний формат кодування даних для електронного зв'язку між комп'ютерами.

ВСТУП

В даний час використання цифрових засобів масової інформації та комунікаційних технологій час від часу стрімко зростає. Але при цьому друковані документи продовжують формувати зручний інтерфейс для людей.

Велика кількість важливих документів, все ще знаходяться в роздрукованому вигляді. Зараз в цифрову еру потрібна одна техніка або механізм для взаємодії з інформацією в друкованих елементах або документах, які можуть бути вбудовані всередину друкованих об'єктів. Таким чином, це може заощадити більше місця в друкованому документі і це безпечно.

На мою думку QR-код може зберігати більше інформації, ніж інші звичайні штрих-коди. Цього можна досягти, використовуючи кольоровий QR-код, мультиплексування даних, видалення пошкоджень та приховування даних. Питання зберігання є основною важливою темою або проблемою для дебатів або обговорення.

Крім того, необхідна кількість QR-коду стає більше, якщо збільшується кількість даних, що підлягають зберіганню. Як наслідок, QR-коди не підходять для представлення на обмежених розмірах друкованих засобів масової інформації.

Метою даного дослідження є підвищення ємності даних кольорового QR-коду шляхом проектування та розробки алгоритмів компресії, мультиплексування та багатошаровості.

Це дослідження сприяє двом сферам, а саме: час зберігання та обробки даних. У сфері зберігання даних це дослідження сприяє тим, хто зацікавлений зберігати дані у формі QR-коду. QR-код може зберігати дані або інформацію без використання будь-яких електронних мікросхем. Його можна використовувати як міні-вторинне сховище. QR-код може знизити витрати на зберігання даних. Цей носій інформації на паперовій основі дешевий і

простий у виробництві. Він також простий у розподілі та перенесенні і не вимагає будь-якої складної технології у використанні. З іншого боку, користувачі можуть легко отримати відповідну інформацію, використовуючи QR-код. Це може покращити бізнес-процеси завдяки швидшому доступу до інформації.

Результатом даного дослідження буде створення модифікованого методу для генерація кольорового QR-коду. Також ця нова модель буде використовувати уже існуючі методи такі як компресія, мультиплексування та багатошаровість. Завдяки цьому дослідженню буде згенерована новий тип кольорового QR-коду, який підвищить своєї ефективності.

РОЗДІЛ 1

АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ

У цьому розділі описуються умови та деякі пов'язані дослідження, проведені з QR-кодом. QR-код став популярним, а його зручність є загальновизнаною; ринок почав вимагати коди, здатні зберігати більше інформації, більше типів символів і які можна було б друкувати на меншому просторі. Шаблони QR є квадратами, крапками, шестикутниками та іншими геометричними фігурами всередині зображення; такий вид штрих-коду називають матричним або двовимірним штрих-кодом. QR є дуже значущим, оскільки несе на собі інформацію. Зберігання даних обмежено через структурований дизайн. Коди містять інформацію як у вертикальній, так і горизонтальній пропорції. Багато дослідників запропонували різні розробки QR-коду, які мають задовольнити розширення ємності зберігання даних.

1.1 QR-КОД

QR-код став широко популярний завдяки своїй швидкості зчитування, точності та функціональним характеристикам [1]. В даний час зростає кількість даних, включаючи електронні листи, фотографії та відео, доступ до яких має бути своєчасним та надійним. Ці дані можуть зберігатися на персональних комп'ютерах, мобільних телефонах або центрах обробки даних по всьому світу. У зв'язку з зростаючими вимогами до даних, зберігання швидко стає важливим чинником у центрі обробки даних обладнання інформаційних технологій. В даний час непереборний потік даних постійно збільшується в обсязі та деталізації, наприклад, соціальні мережі, інтернет речей та мультимедіа. Внаслідок зростання обсягу даних це стає найбільшою проблемою для великих підприємств.

QR-код - це один із засобів, що використовуються для зберігання інформації за допомогою паперу, всередині якого знаходиться код [2], японська компанія з розробки мобільних пристроїв, створила тривимірну (3D) систему

штрих-кодів під назвою або PM-код. Техніка полягає в тому, щоб складати QR-коди в один інтегрований 3D-штрихкод.

Таблиця 1.1

Порівняння щільності даних між деякими двовимірними штрих-кодами, надрукованими з роздільною здатністю 600 dpi.

Двовимірний штрих-код	Щільність даних [біт на сантиметр^2]
QR Code	796
HCCB	2539
HCCB2D	2388
CQR Code-5	2660

Основними цілями удосконалення QR-коду є збільшення простору, доступного для даних, та збереження сильної стійкості та властивостей корекції помилок, аналогічних оригінальному стандарту QR. У зв'язку з цим необхідно розглянути три складові, коли йдеться про ємність даних QR-коду. Цими частинами є методи збільшення, стандарт QR-коду та алгоритми. Методи збільшення ємності даних ґрунтуються на попередніх дослідженнях ємності даних або її збільшенні

При розгляді використання кольорів RGB для розробки кольорового QR-коду необхідно розглянути три аспекти: методи збільшення ємності даних, алгоритми кодування і декодування та стандартний бенчмаркінг. Відповідні методи повинні бути знайдені, розширені та об'єднані для отримання на виході кольорового QR-коду RGB. Крім того, необхідно визначити стандартний еталон (тобто чорно-білий QR-код) на початку процесу збільшення його ємності. Коли вхідні дані визначені, необхідно розробити алгоритми для створення (тобто кодування) QR-коду.

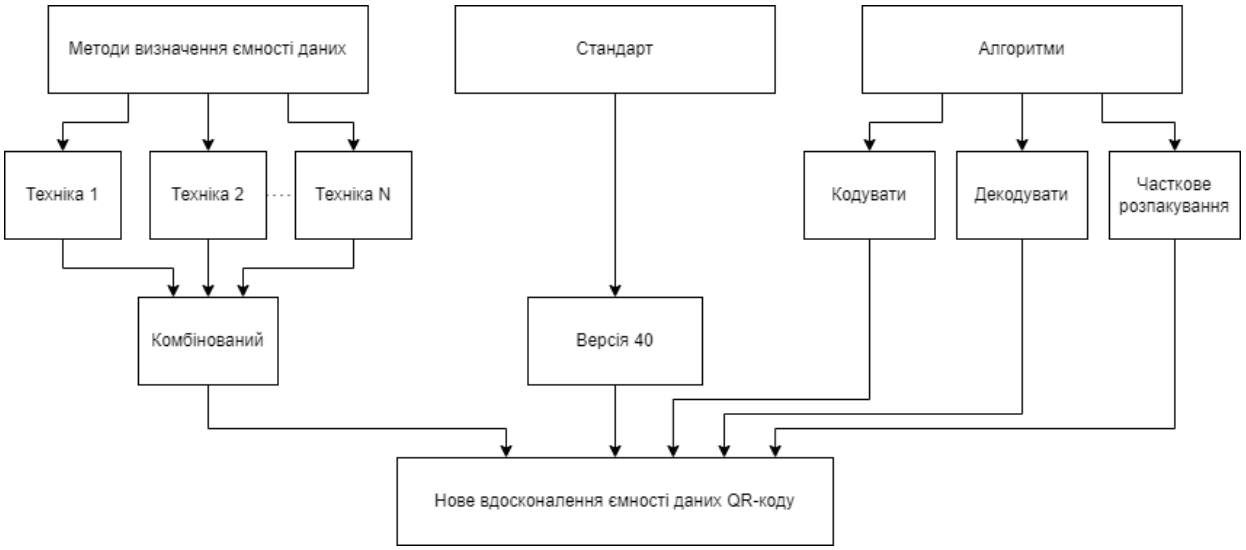


Рис. 1.1. Умовна модель кольорового QR коду RGB для збільшення ємності даних.

Кольоровий QR-код корисний у різних сферах, але головна перевага полягає в тому, що він може виступати як міні-сховище даних для кінцевих користувачів для зберігання постійних або тимчасових даних. Оскільки в сучасну епоху існує інформаційний вибух, кінцеві користувачі перевантажені великою кількістю даних та знань. Деякі з цих даних повинні легко зберігатися та бути доступними.

Одномірні штрих-коди зазвичай зустрічаються на товарах, які скануються на касі при купуванні. Функціональність традиційних штрих-кодів залежить від методу зчитування, тобто зліва направо. Це одне з основних обмежень у 1D штрих-кодах. Тим не менш, QR-код має низку унікальних особливостей, таких як:

1. **Висока ємність кодування даних**. Штрих-код 1D має обмежену ємність і зберігає в основному менше 20 символів.

2. **Невеликі друковані дані.** Якщо 1D-штрих-код та QR-код мають однаковий обсяг даних, то розмір між ними відрізняється на 25%.

3. **Стійкість до забруднення та пошкоджень.** Коли зображення QR-коду забруднене або пошкоджене, дані можуть бути відновлені за допомогою механізму, що називається корекцією та відновленням помилок.

4. **Може бути рахований у багатьох напрямках у порівнянні зі**

звичайним штрих-кодом. Зчитувальний пристрій здатний зчитувати QR-код у напрямку 360 °.

5. **Функція структурованого додавання.** QR-код розроблений для зберігання даних у різних символах QR-коду.

1.1.1 Структура архітектури QR-кодів

QR-коди складаються з різних сегментів, зарезервованих для конкретних цілей. У QR-коді версії 2 сегмент розділений на вісім секцій відповідно до номерів, вказаних на рисунку 1.2, і кожен сегмент має свої специфічні завдання. До специфічних завдань належать:

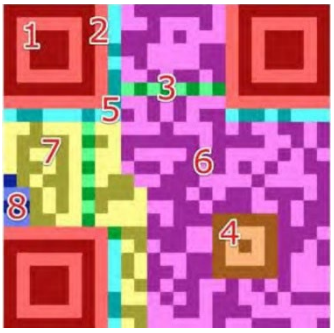


Рис. 1.2. Структура QR-коду версії 2.

1. **Шаблон шукача (1).** Він використовується для виявлення положення QR-коду у додатку-декодері.

2. **Розділювачі (2).** Білі роздільники покращують розпізнаваність шаблонів пошуку, оскільки відокремлюють їхню відмінність від фактичних даних. Ширина роздільників становить один піксель.

3. **Шаблон синхронізації (3).** У програмі декодера паттерн синхронізації використовується визначення координат символу.

4. **Шаблони вирівнювання (4).** Шаблон вирівнювання використовується для визначення сіток вибірки, з яких вилучаються кодові слова, та для забезпечення правильної деформації зображення шаблону.

5. **Інформація про формат (5).** У розділі інформації про формат зберігається інформація про рівень корекції помилок QR-коду та вибраний шаблон маскування.

6. **Дані (6).** Дані зберігаються у вигляді 8-бітових частин (званих

кодовими словами) у секції даних після їх перетворення на бітовий потік.

7. **Корекція помилок (7).** Аналогічно секції даних, коди корекції помилок зберігаються у 8-бітних довгих кодових словах у секції корекції помилок.

8. **Біти залишку (8).** Якщо біти даних і біти корекції помилок не можуть бути поділені на 8-бітові кодові слова без залишку, вони будуть складатися з порожніх бітів.

1.1.2 Типи QR-коду

Були створені різні типи QR-кодів із певними шаблонами. Існує п'ять типів QR-коду, які будуть описані нижче:

1. **QR-код Модель 1 / Модель 2.** Модель 1 здатна зберігати до 1167 цифр і є початковим QR-кодом. Найвища версія коду - 14 з модулями 73 x 73. Модель 2 не поступається моделлю 1, вона здатна зберігати до 7 089 цифрових символів. Версія 40 з 177 x 177 модулями є найбільшою з випускаються. Ця модель 1 є типом QR-коду, який використовується в даний час.

2. **Мікро QR-код.** Він допускає лише одну орієнтацію виявлення малюнка і може бути надрукований на меншому просторі, ніж раніше. Він може зберігати до 35 цифр, найбільший модуль має розмір 17 x 17.

3. **iQR Code.** Код може бути згенерований у два етапи, будь то прямокутні або квадратні модулі. Можна друкувати перевернутий код, чорно-білий інверсійний код або точковий малюнок. Версія 61 може зберігати 40 000 цифрових символів та містить 422 x 422 модулі.

4. **Код SQR.** Має функцію конфіденційності, яка може бути використана для зберігання приватної інформації або управління інформацією компанії. Функція також включає функцію обмеження читання.

5. **Логотип Q** – це вбудований рівень елементів дизайну, таких як ілюстрації, літери та логотипи. Читабельність не порушується, оскільки при генерації цього коду використовується власна логіка.

Таблиця 1.2

Розмір та ємність даних для різних версій QR-коду.

Версія	Модулі на бік	Модулі шаблонів функцій	Модулі формату та версії	Модулі даних	Ємність кодового слова	Модулі нагадувань
1	21	202	31	208	26	0
2	25	235	31	359	44	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮
40	177	1614	67	29648	3706	0

На рисунку 1.3 показано конструкцію різних типів QR-коду, як пояснювалося вище.

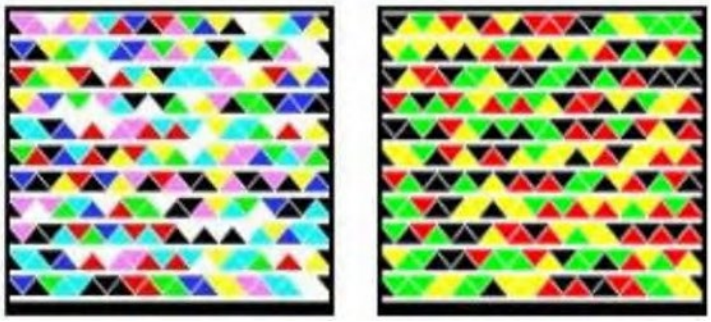
Рис. 1.3. Дизайн QR-кодів [3].

1.2 Кольоровий штрих-код

Деякі двовимірні штрих-коди використовують кольори для створення більшої кількості символів, що призводить до збільшення ємності даних при тому ж розмірі. Прикладами таких штрих-кодів є система кольорових штрих-кодів компанії Imaged Ltd. і найпоширеніша система Microsoft High HCCB. HCCB, також відомий як Microsoft Tag, використовується для різних програм, таких як інформація, розваги та інтерактивний досвід на мобільних телефонах. Основні характеристики HCCB на рисунку 2.4 складаються із рядків, рядків символів (трикутників) чотирьох різних кольорів: чорного, червоного, зеленого та жовтого, а послідовні ряди розділені білою лінією.

Ця техніка враховує лише певні основні кольори, такі як чорний, червоний, зелений та жовтий. Вона не включає комбіновані кольори RGB, які можуть становити безліч типів кольорів. Крім того, пам'ять для восьми кольорів може зберігати лише 84 байти порівняно з QR-версією 40, яка може зберігати до 3 кілобайт.

Основне обмеження коду HCCB пов'язане з крихкістю механізмів виявлення та вирівнювання. Дійсно, процес виявлення працює наступним чином: він починається з точки, яка повинна знаходитись у глибині коду і переходить до квадратів більшого розміру, доки не розпізнає білу межу навколо коду. Після виявлення білої межі починається процес вирівнювання шляхом пошуку нижньої товстої рамки. Крихкість процесу виявлення обумовлена тим, що не всі зображення всередині білого кордону обов'язково є кодами, що призводить до виникнення відкладених збоїв. Слабкість процесу вирівнювання обумовлена тим, що різні нахили на етапі сканування можуть призвести до неможливості правильно розпізнати нижню товсту раму.



8 кольорів штрих-коду, що зберігають 84 RAW байти

4 кольори штрих-коду із зберіганням 58 RAW байт

Рис. 1.4. Кольоровий штрих-код високої ємності від Microsoft [4].

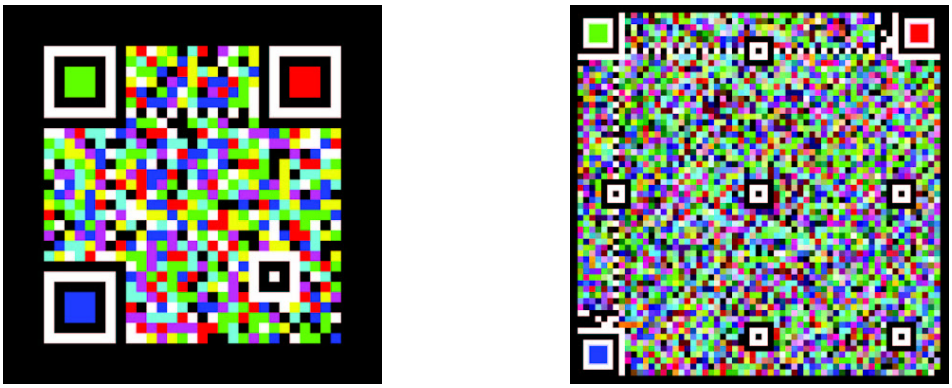


Рис. 1.5. Структури стандартної та IP-технології кодування ПМ

Паперовий код пам'яті (PM-код) відомий як технологія перетворення напівоб'ємних даних, таких як текст, зображення, звук, анімація і т.д., закодовані дані в кольоровому матричному QR-коді. Растрове зображення коду може бути декодовано декодером та відновлено до вихідних даних.

1.3 Кольоровий QR-код

У більшості випадків під час друку використовуються блакитний, жовтий та пурпуровий кольори, але при захопленні зображення використовуються червоний, зелений та синій канали сприйняття. Багато дослідників запропонували кольоровий QR-код, який може запропонувати більшу ємність даних у порівнянні з чорно-білими штрих-кодами [5]. Кольоровий QR-код містить ті ж функціональні шаблони, включаючи шаблони пошуку, роздільники та спокійну зону.

Система кольорів RGB була розроблена на основі комбінації червоного, зеленого та синього кольорів. Існує 16777216 можливих кольорів, які використовують 8-бітовий колір. Формат кольору та розрахунок на основі 24-бітного формату (0...23) показані на рисунку 1.6 нижче.

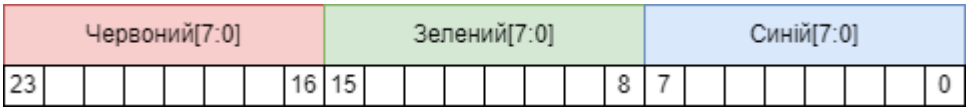


Рис. 1.6. Колірний формат та розрахунок на основі 24-бітного формату (0..23).

Як зазначено на сайті Online Reference and Tool, колірний простір RGB, або колірна система RGB, будує всі кольори з комбінації червоного, зеленого та синього кольорів. Червоний, зелений і синій кольори використовують по 8 біт і мають цілі численні значення від 0 до 255. Загальна кількість можливих кольорів становить 256*256*256=16 777 216 можливих кольорів. Кожен піксель на РК-моніторі відображає кольори таким чином – комбінацією червоних, зелених та синіх світлодіодів. Коли червоний піксель встановлений на 0, світлодіод вимкнено. Коли червоний піксель встановлений на 255, світлодіод увімкнено. Будь-яке значення між ними переводить світлодіод у режим часткового випромінювання світла.

1.3.1 Глибина кольору

Це також відомо як бітова глибина або глибина пікселя, яка є кількістю бітів, що використовуються в пікселі, а один піксель містить його колірну складову [6]. У комп'ютерному дисплеї під цим розуміється кількість бітів на

піксель, що становить певний колір. В даний час майже всі комп'ютери підтримують щонайменше 32-бітовий колір, що дозволяє використовувати до 16,7 мільйонів колірних комбінацій. 48-бітовий колір може підтримуватись операційною системою за умови, що відеокарта комп'ютера підтримує цю глибину кольору. Коли технології та доступні системні ресурси зросли, еволюція глибини кольору також поширилася. Еволюція глибини кольору починається із двоколірної комбінації, відомої як монохромний дисплей. У таблиці 1.3 наведено перелік усіх відмінностей між глибиною кольору.

1.3.2 Кольорова модель

Будь-який метод пояснення властивостей чи поведінки кольору у певному контексті та створення повного спектру кольорів з невеликого набору основних кольорів називається колірною моделлю. Існує два типи колірних моделей, це адитивні колірні моделі, які використовують світло для відображення кольору, та субтрактивні колірні моделі, які застосовуються для друку за допомогою фарб. В даний час колірна модель - це просто спосіб визначення кольору та його відображення на апаратних засобах, таких як кольорові монітори та принтери.

У моделі RGB первинний спектральний компонент для кожного кольору проявляється у червоному, зеленому та синьому кольорах. Адитивні кольори використовують колір світла. Загалом зображення, представлені в колірній моделі RGB, складаються з трьох компонентів (червоний, зелений, синій), і коли ці компоненти потрапляють на монітор RGB, три компоненти зображення об'єднуються на люмінофорному екрані, створюючи складне кольорове зображення, яке потрапляє прямо з монітора в очі.

1.3.3 Піксіляція

Піксилювання виникає при збільшенні растрових зображень, коли окремі пікселі кольору (маленькі кольорові квадратики) видно як елементи, з яких складається растрове зображення. У ранніх графічних додатках це легко помітити різкими краями, які надавали вигнутим об'єктам і діагональним лініям

неприродний вигляд. На рисунку 1.7 показано зображення, збільшене з невеликою ділянкою для ближчого розгляду, на якому можна розрізнити окремими пікселями. Це може статися випадково, коли зображення призначене для відображення на звичайному комп'ютері; при відображенні на великому екрані кожен піксель видно окремо. Пікселі використовуються для деталізації інформації зображення. Вони можуть передавати інформацію про тип кольору кожного пікселя. З комбінації пікселів можна отримати зображення. Чим більше пікселів використовується у зображенні, тим більше деталей зображення виходить.

Рис. 1.7. Більш детальне зображення у збільшеному масштабі.

1.3.4 Багатошаровий колір

Наголосили у своїх дослідженнях [7], що ємність даних може бути покращена шляхом додавання більшої кількості колірних шарів у QR-код. На підставі цього дослідження вони також відзначають, що QR-код з ємністю в дев'ять кілобайт недостатньо хороший відповідно до стандартів користувача і незалежної інтернет-інформацією. Незважаючи на те, що дослідники не створили програму для автоматичної генерації кольорових QR-кодів, вони розробили псевдокод як керівництво.

Кольоровий QR-код - це матричний код, який може бути розширений до системи 3D-штрих-кодів. 3D-штрихкод містить осі x, y та глибини і є розширеною версією традиційного QR-коду. Місткість даних може бути збільшена за рахунок додавання додаткових шарів у систему штрих-коду

шляхом їх укладання. В результаті експерименту кольоровий QR-код був обмежений зберіганням лише дев'яти кілобайт із трьома шарами, які мали три кольори та вісім комбінацій кольорів. Обмеженням даного дослідження є те, що кольори, з якими проводився експеримент, були обмежені лише червоним, зеленим та синім.

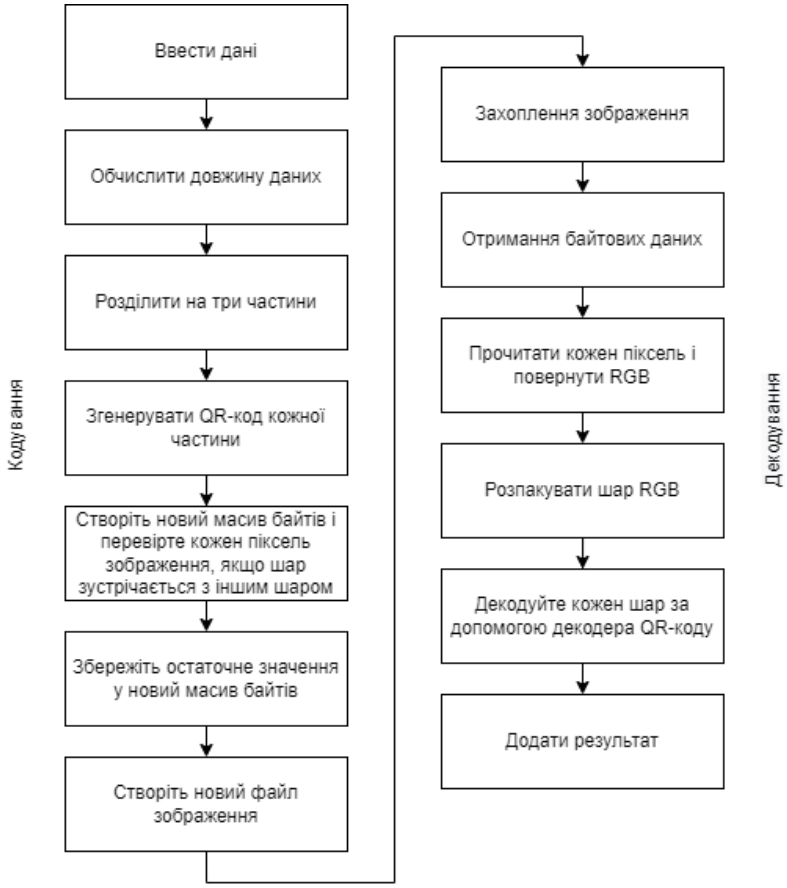


Рис. 1.8. Блок-схема процесів кодування та декодування кольорового QR-коду

Кількість QR-кодів, що вводяться в даному дослідженні, було обмежено трьома монокультурними QR-кодами. Експерименти з кольорами RGB проводилися завдяки здатності коду будувати всі кольори з їхньої комбінації.

Кодування

Перший крок до початку цього процесу – підготовка до генерації одноколірного QR-коду. Його можна згенерувати через Інтернет. Тим часом програма редагування зображень (наприклад, Adobe Photoshop) використовується для об'єднання кольорів. Нижче описано кроки по об'єднанню багатобарвних шарів за допомогою програми редагування зображень шляхом

накладання трьох шарів. Кроки наступні:

1. По-перше, дані поділяються на три частини для зручності кодування. Кожна частина буде закодована в окремий шар кольорового QR-коду.

2. Потім користувач кодує кожну частину QR-код.

3. Після того, як усі три QR-коди згенеровані, їм необхідно присвоїти різні кольори і скласти їх у стопку.

4. Після того, як всім шарам надано колір, кольори необхідно об'єднати в точках перетину шарів, використовуючи для цього програму редагування зображень.

5. У редакторі зображень необхідно створити чотири окремі шари, кожен з яких повинен містити QR-код, причому один шар повинен бути заповнений чорним кольором.

6. Необхідно поєднати кольори шарів у місцях їх перетину.

Процес кодування не реалізований у програмній структурі, тому неможливо провести експеримент, такий як продуктивність та перевірка кольору. Усі комбінації повністю залежить від редактора зображень. Автоматичний генератор кольорового QR-коду не може бути виконаний через обмеження можливостей редактора зображень. Крім того, комбінація кольорів для даного експерименту обмежена лише трьома кольорами як зразок.

Для створення автоматичного генератора кольорового QR-коду було запропоновано псевдокод. Кроки наступні:

1. Отримання вхідних даних. Дані можуть бути числовими, літерно-цифровими, двійковими чи японськими символами.

2. Обчислити довжину даних і поділити їх на три частини.

3. Згенеруйте QR-код для кожної частини даних.

4. Призначте колір кожного шару. Шар 1 - червоний, тому кожна чорна точка згенерованого QR-коду шару 1 змінюється на червону.

5. Створіть новий масив байтів і перевірте кожен піксель зображення, а також перевірте, чи шар перетинається з іншим шаром. Якщо так, додайте значення RGB.

6. Збережіть байтові дані та створіть новий файл зображення. З урахуванням обмежень цього методу закодований QR-код необхідно розділити пропорційно. Розмір QR-коду повинен бути однаковим для кожного шару, але рівень корекції помилок не обов'язковий.

Декодування

Як завжди, першою дією при декодуванні кольорового QR-коду є захоплення зображення. Зображення витягується в байтові дані, і кожен із байтових даних має бути прочитаний і переведений у значення кольору RGB. Кожен піксель RGB кольору буде розділений на моноколорні шари, і цей процес виконується декодером QR-коду. Шари повинні мати вкладену структуру, оскільки зберігання інформації використовується комбінація моноколорних понять. Проблеми під час декодування – це проблема точності через джерело освітлення, обладнання, знебарвлення, роздільну здатність зображення тощо.

Процес декодування є процесом відновлення даних, оскільки при захопленні зображення відсутня перехресна взаємодія між каналами камери (зазвичай колірні канали RGB, що використовуються в пристроях захоплення) і шарами барвників CMY. Тим не менш, барвники для друку CMY можуть бути вилучені із захвату RGB. Після захоплення зображень закодовані дані отримуються за допомогою алгоритмів локалізації та корекції геометрії для монохромних штрих-кодів, щоб отримати RGB QR-код. Після створення RGB QR-коду процес триває з подолання інтерференції між каналами за допомогою алгоритму придушення колірної інтерференції.

1.3.5 Мультиплексування та демультиплексування

Метод мультиплексування означає об'єднання кількох однотипних елементів передачі однієї лінії. При використанні мультиплексування потрібно лише один QR-код. Це призводить до зниження вартості лінії та допомагає відстежувати окремі QR-коди. Тим часом техніка демультиплексування виконує зворотну функцію - поділ об'єднаних однотипних елементів на вихідні елементи.

В дослідженні [8], було згадано про, алгоритм збільшення ємності QR-коду за допомогою мультиплексування та демультиплексування. Концепція спеціального коду символу використовується для збільшення ємності даних у порівнянні з традиційним QR-кодом. Вихідні дані мають бути закодовані та розділені на дрібніші частини. Менші частини генеруються у стандартній формі QR-коду та мультиплексуються для отримання чорно-білого QR-коду зі спеціальними символами всередині.

Кодування мультиплексування

QR-код використовує методи мультиплексування та демультиплексування для збільшення своєї інформації. Спочатку вихідні повідомлення поділяються на невеликі частини та виконуються у вигляді шаблону QR-коду. Процес мультиплексування починається зі зміни модуля кожної частини на спеціальні символи та регенерації QR-коду зі спеціальними символами всередині. Деталі кодуються і представляються для кожного модуля QR-коді спеціальними символами, які мають чорно-білий колір. На рисунку 1.9 показано блок-схему процесів мультиплексування та демультиплексування.

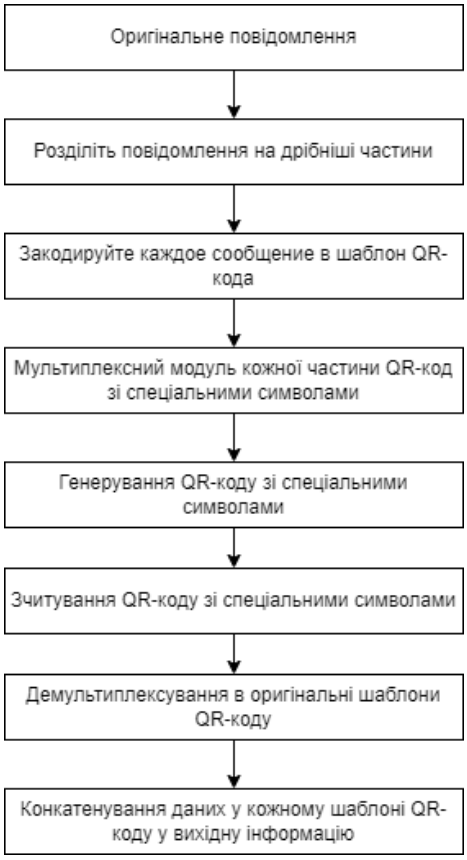


Рис. 1.9. Алгоритми мультиплексування та демультиплексування.

Значення 0 є чорним модулем, а 1 - білим. Коли відповідний модуль в одній і тій же позиції кожного шаблону QR-коду чорний, чорний, чорний (0 0 0), QR-код використовується символ \. Однак у процесах мультиплексування та демультиплексування символи перевіряються лише до 10 символів (\, /, ^, V, >, <, ~ , і !), І існує безліч символів, які можуть бути введені. Незважаючи на те, що в експерименті згадується, що можуть бути використані будь-які символи клавіатури або спеціальні символи, немає жодної інформації про поширені символи, що використовуються в клавіатурі іншої мови.

Процес кодування полягає у введенні даних та їх перетворенні на еквіваленти коду ASCII. Примітивний поліном використовується для створення кінцевих чисел поля, еквівалентних ASCII. Алгоритм кодування використовується для створення кодових слів за допомогою чисел кінцевого поля. Після цього кодове слово перетворюється на двійковий код і розміщується відповідно до шаблону QR-коду. Шаблон QR-коду розглядається як шари кольорів і об'єднується для створення кольорового QR-коду. На рисунку 1.10 показано процес створення кольорового QR-коду.

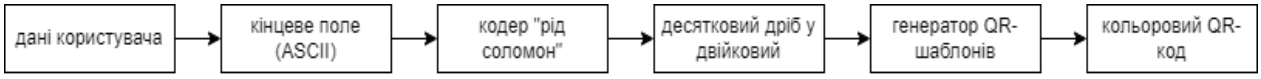


Рис. 1.10. Процес створення кольорового QR-коду.

Треба розділити вихідну інформацію на n частин. n частин - це кількість QR-кодів, що використовуються для процесу мультиплексування, а також для використання як необхідні окремі кольори на основі рівняння 2n.

QR-коди будуть згенеровані з урахуванням n частин. Процес мультиплексування починається з визначення відмінного кольору за допомогою техніки забарвлення коду. Окремий колір буде призначено для кожного можливого шаблону після об'єднання QR-коду. Відмінний колір, що згенерований з комбінації кольорів RGB, виконується з матриці колірної карти в MATLAB.

На рисунку 1.11 показано процес мультиплексування кольорового QR-коду.

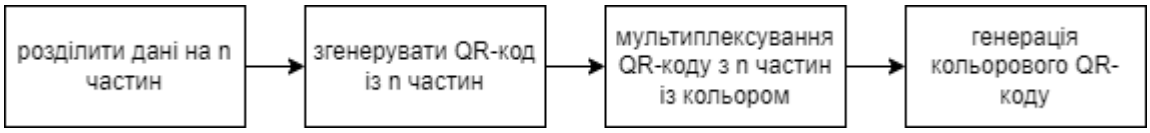


Рис. 1.11. Процес мультиплексування кольорового QR-коду.

Декодування демультиплексування

Фаза декодування включає зчитування QR-коду зі спеціальними символами та демультиплексування його у вихідний шаблон. Символ у кожному модулі (кодовому слові) буде витягнутий назад у кожен шар (шаблон). Значення 0, що представляє чорний модуль, та 1, що представляє білий модуль, будуть відновлені у кожному шарі. Після завершення відновлення модулів вихідний QR-код готовий до зчитування у звичайному режимі. На рисунку 1.12 показано процес демультиплексування QR-коду зі спеціальними символами.

Рис. 1.12. Процес демультиплексування QR-коду із спеціальними символами.

Процес декодування починається з читання кольорового зображення QR-коду та розбиття його на колірні шари RGB. Розділені шари RGB містять двійкову інформацію, і ці шари будуть перетворені на десяткове значення для кожного байта. Десяткові значення використовуються як вхід для декодера Ріда-Соломона. Виходи декодера Ріда-Соломона – це еквіваленти ASCII, і вони будуть перетворені на вихідні дані. На Рисунку 1.13 показано процес декодування.

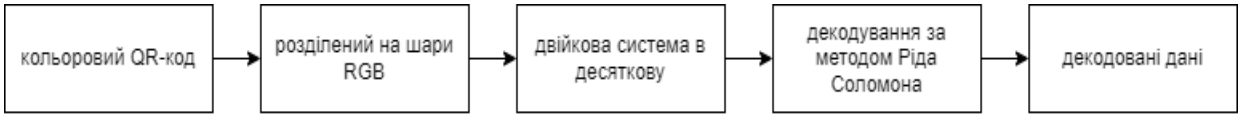


Рис. 1.13. Процес декодування.

Процес демультиплексування включає у собі зворотне завдання методу мультиплексування. Для отримання значень RGB із мультиплексованого кольорового QR-коду використовується таблиця призначених комбінацій кольорів. Оригінальне значення пікселя QR-коду буде відновлено шляхом порівняння з усіма комбінаціями таблиць мультиплексування. Після відновлення

вихідного значення пікселя QR-коду він стає сумісним для сканування пристроями, що зчитують. Отримана інформація може бути об'єднана для формування оригінальної довгої послідовності повідомлень. На рисунку 1.14 показаний процес демультиплексування та декодування.

Рис. 1.14. Процес демультиплексування та декодування.

1.3.6 Компресія

Nancy Victor згадує у своєму дослідженні [9], що для збільшення ємності даних QR-коді, його необхідно спочатку стиснути, використовуючи будь-який тип техніки стиснення даних перед виконанням процесу кодування. У цьому дослідженні пропонується техніка стиснення даних з допомогою двох кроків: (1) перший крок - перетворення текстових даних у двійкову форму; (2) і другий крок - генерація даних хеш-карти цих двійкових даних. В результаті стиснення даних може перевищити більше чотирьох мегабайт даних у QR-коді в порівнянні з чотирма кілобайтами, як це було раніше.

Ідея стиснення корисна тим, що дозволяє знизити споживання ресурсів. При здійсненні процесу розпакування необхідно виконати додаткові обчислювальні процеси. При здійсненні процесу стиснення важливими чинниками є ступінь чи відсоток стискування, ступінь спотворення та обчислювальні ресурси.

Кодування компресії

Ідея створення QR-коду високої ємності полягає в тому, щоб почати з простого стиснення даних перед виконанням процесу кодування. Реалізація кодування здійснюється за допомогою звичайних процесів кодування QR-коду, які полягають у наступному:

1. Проаналізувати дані та перетворити їх на символьні знаки.

Визначити рівень корекції та виявлення помилок.

2. Закодувати дані.
3. Запустити кодування з корекцією помилок.
4. Додавання бітів залишку та шаблонів маскування даних.
5. Генерувати інформацію про формат та інформацію про версію.

На рисунку 1.15 показано блок-схему генерації QR-коду високої ємності.

Рис. 1.15. Блок-схема генерації QR-коду високої ємності.

Кольоровий QR-код може бути розроблений на основі даних хеш-картки. Використовуючи техніку колірного кодування RGB, кожній літері від "A" до "P" будуть присвоєні різні кольори. Щоразу, коли зустрічається такий символ, відповідний колір буде поміщений QR-код. Загальний біт для розподілу кольорів складає чотири біти.

Ще один залишок із чотирьох бітів призначається на розподіл шістнадцяти символів у хеш-карті. Наприклад: від 'a' до 'o' і від 'p' до 'z'. Кожна мапа розподілу шістнадцяти символів буде представлена у вигляді групи колірних кодів. Етапи генерації великої кількості даних QR-коду показані рисунку 1.16.

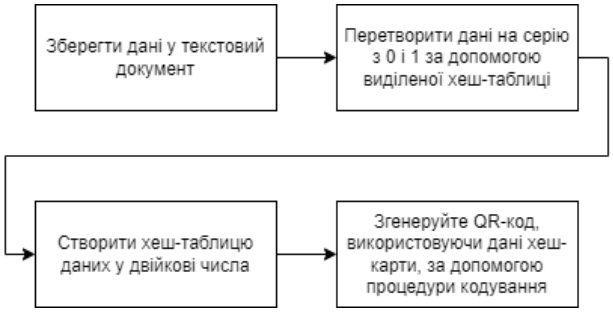


Рис. 1.16. Етапи генерації великої кількості даних QR-коду.

Після завершення вищезгаданих дій можна створити кольоровий QR-код, об'єднавши обидва чотири біти. Колірне відображення обмежено 16 кольорами,

оскільки чотири біти можуть мати лише 16 двійкових розрядів.

Декодування компресії

Процес декодування є зворотним процедурою кодування. Для декодування правильних даних потрібно визначити зону спокою. У процедурі декодування спотворення символів можна виправити за допомогою шаблонів вирівнювання. Рекомендуються два методи, включаючи метод оцінки від краю до схожого краю для перевірки правильності виявленого штрихового шаблону і препроцеси, які проводяться перед виконанням пошуку шаблону.

1.3.7 Гібридне розширення

Є безліч методик, спрямованих збільшення ємності даних QR-коду. Деякі з цих методів – компресії, мультиплексування та багатошаровість. Ці методи можуть бути використані, якщо відповідний метод реалізовано разом.

Для збільшення ємності даних було обрано три методи. Це стиск, багатошаровість та мультиплексування. Водночас у процесі реалізації виникне кілька можливих проблем. У таблиці 1.3 показані можливі проблеми, що виникають під час реалізації пріоритетного обміну.

Таблиця 1.3

Можливий проблемний досвід у разі запровадження пріоритетного обміну.

Пріоритет 1	Пріоритет 2	Пріоритет 3	Примітки
Компресія	Мультип.	Багатошарова	Він підходить, тому що спочатку потрібно стиснути дані, потім виконати чорно-білий QR-код, після чого згенерувати кольоровий QR-код за допомогою багатошарової техніки.
Компресія	Багатошарова	Мультип.	Тим часом техніка мультиплексування охоплює чорно-білий. В результаті, при перетворенні кольорового в чорно-білий

			витрачається більше часу на обробку. Важко здійснити перетворення.
Мультип.	Компресія	Багатошарова	Не підходять, оскільки текстовий файл потрібно спочатку стиснути. Це потребує більше часу для обробки при зворотному декодуванні QR-коду.
Мультип.	Багатошарова	Компресія	
Багатошарова	Мультип.	Компресія	
Багатошарова	Компресія	Мультип.	

Щоб зберегти більше символів, спочатку необхідно створити стислий текст. Текст повинен бути спочатку стиснутий, перш ніж буде згенеровано QR-од. На рисунку 1.17 показані процеси, що відбуваються, коли техніки стиснення, мультиплексування та багатошаровості змінюють свої позиції. З вищезгаданого рисунку видно, що перша техніка показує найменшу кількість процесів у порівнянні з другою та третьою техніками. Основна причина полягає в тому, що коли техніка стиснення переміщується на іншу позицію після тексту, їй потрібний додатковий час на обробку, щоб отримати QR-код високої ємності. Навіть якщо техніка стиснення вміщена в останню позицію, QR-код не може бути згенерований. На закінчення слід зазначити, що перша техніка є найкращим рішенням для отримання більшої ємності даних у запропонованому QR-коді.

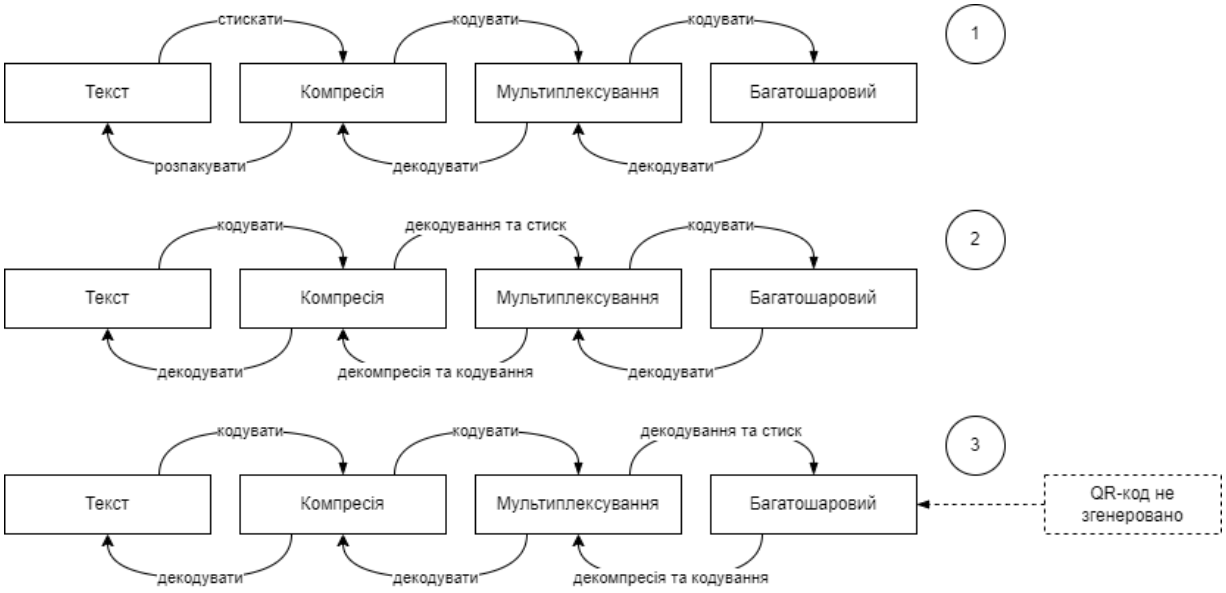


Рис. 1.17. Процеси, що відбуваються при зміні позицій методів стиснення, мультиплексування та багатошаровості.

Пропонується три техніки, які були засновані на попередніх дослідженнях кількох дослідників із деякими змінами. Технічна розбивка виглядає так:

1. Метод компресії. Компресія GNU Zip (GZip) є найкращою утилітою для компресії тексту за допомогою QR-коду. Крім того, вона має бути доповнена кодером/декодером Base64.

2. Техніка мультиплексування. QR-код, як і раніше, генерується на основі чорно-білого кольору. Більш того, QR-код повинен бути поділений на невеликі частини, перш ніж його можна буде мультиплексувати.

3. Багатошарова техніка. Кольоровий QR-код буде згенерований на основі комбінації червоного, зеленого та синього шарів QR-коду, отриманих у процесі мультиплексування.

1.3.8 Структурований додаток

Основною особливістю структурованого додавання QR Code 2005 є можливість розділити максимум 16 символів структурного формату в одному QR-коді. Ці 16 символів можуть бути розділені на декілька областей даних та QR-символів. Перевага структурованого додавання QR-коді полягає в тому, що воно дозволяє друкувати у вузькій області.

Існує два типи режиму в структурованому append 8-бітного кодового слова, яке розміщується в блоці заголовка. До них відносяться:

1. Положення конкретного символу (4 біти). Розташовується у перших чотирьох бітах кодового слова заголовка та ідентифікується як позиція конкретного символу.

2. Загальна кількість символів, що підлягають конкатенації (4 біти). Знаходиться в останніх чотирьох бітах кодового слова заголовка та ідентифікується як загальна кількість символів, що підлягають конкатенації у форматі структурованого доповнення.

Бітовий шаблон у блоці заголовка є комбінацією позиції та загальної кількості символів. Рисунок 1.18 ілюструє один символ та структуроване додавання символів, закодованих за допомогою "ABCDEFGHIJKLMNOPQRST-UVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"

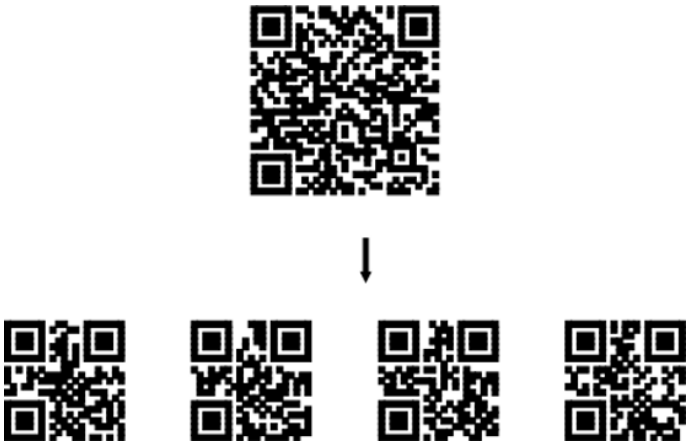


Рис. 1.18. Одиночний символ та структуроване доповнення символів, закодованих за допомогою "ABCDEFGHIJKLMNOPQRSTUVWXYZ-0123456789ABCDEFGHIJKLMNOPQ-RSTUVWXYZ".

1.4 Комбіновані методи ємності даних QR-коду

Робота двомірного штрих-коду ділиться на три області, які полягають у наступному:

1. Інформаційна ємність. Додаючи кольори, двомірний штрих-код може поліпшити свою здатність зберігати більше інформації. Для підвищення інформаційної ємності необхідно змінити правила кодування.

2. Ступінь стиснення інформації. Високий рівень стиснення збільшує загальну кількість символів, які можуть бути збережені. Крім того, QR-код може зберігати бінарні значення, які можуть бути використані у файлі стиснення.

3. Можливість корекції. Високий відсоток відновлення значення є хорошим показником для QR-коду, оскільки це дозволить зробити QR-код більш надійним і точним щодо доступності даних.

Моделі та алгоритми обрані з кількох причин, головні з цих причин:

1. Місткість даних. Новий удосконалений QR-код здатний генерувати чотири мегабайти даних, а приріст даних може досягати трьох разів в одному QR-коді.

2. Кольори. Червоний, Зелений та Синій регулярно тестуються дослідниками. Колірна комбінація шарів тестується лише за допомогою програми для фотозйомок (Adobe Photoshop). Ця техніка не є підходящим

методом для експериментів із квітами та їх поєднанням між шарами.

3. Компресія. Техніка стиснення є додатковою функцією дослідження, оскільки ємність зберігання даних QR-кодів може значно збільшитися, якщо впровадити техніку стиснення.

Незважаючи на вибір трьох досліджень, необхідно ввести зміни, щоб об'єднати вхідні та вихідні дані між трьома методами. Як і третій задачі, часткова інформація може бути створена за допомогою індексного відображення. Це означає, що дані відображатимуться за допомогою вибраного індексу.

1.5 Постановка задачі

Після розгляду предметної області в попередніх розділах настає необхідність в розробці та реалізації QR-кодів з використанням RGB кольорів. Особливість системи полягає в алгоритмах, які будуть розроблені для допомоги компресувати, мультиплексувати та багатошаровістю об’єднати стандартний QR-код з RGB. Для реалізації даної системи буде розроблено додаток, який буде генерувати QR-код та виводити результати.

Для розробки наших оптимізацій ми застосуємо алгоритми компресії, мультиплексування та багатошаровості. Для вхідних даних цих алгоритмів ми повинні будемо з’ясувати пріоритети кожної задачі та використати такі обчислення:

- Процеси які включають ініціалізацію, підрахунок символів, створення чорно-білого QR-коду та розміщення пікселів.
- Відображення алгоритмів, псевдокодів і блок-схем, які можуть допомогти краще зрозуміти перебіг процесів.
- Також впроваджено емпіричний аналітику, щоб надати більшого впливу на результати наступного дослідження.

ВИСНОВКИ ДО РОЗДІЛУ 1

У цьому розділі були представлені фонові дослідження предметів, що розглядаються в цій пропозиції. Першими були розглянуті організаційний процес QR-коду та огляд як допоміжне знання. Дослідження QR-коду були докладно описані, особливо процеси кодування, декодування та часткового вилучення QR-коді. Часткове вилучення було пояснено з погляду його функціональності та можливостей.

РОЗДІЛ 2

ДОСЛІДЖЕННЯ ФРЕЙМВОРКА ТА ВИБІР ТЕХНОЛОГІЇ

У цьому розділі описано методологію, що використовується для досягнення цілей дослідження. Вона вміщує в собі методи та прийоми, які систематично використовують науковці для вирішення проблеми після отримання результатів; наприклад: дослідження методів збору даних, обробки та інструменти. У дослідженні застосовано дедуктивний підхід, який починається із загальної ідеї і потребує коментарів до інших розділів. Як стратегія, кожна фаза має власний метод і результат.

2.1 Методологія дослідження

Процес дослідження проілюстровано на рисунку 2.1.

2.1.1 Перший етап

Перший етап називається попереднім дослідженням, який показано на рисунку 2.1. Завдання включає в себе пошук інформації, пов’язаної з різними джерелами, і вибір відповідних алгоритмів, які можна успішно виконати до того, як буде внесено будь-яку зміну для обраного алгоритму. Під час виконання цього етапу слід враховувати три компоненти, а саме створення теоретичної основи, аналіз уже існуючих алгоритмів і вибір алгоритму для об’єднання.

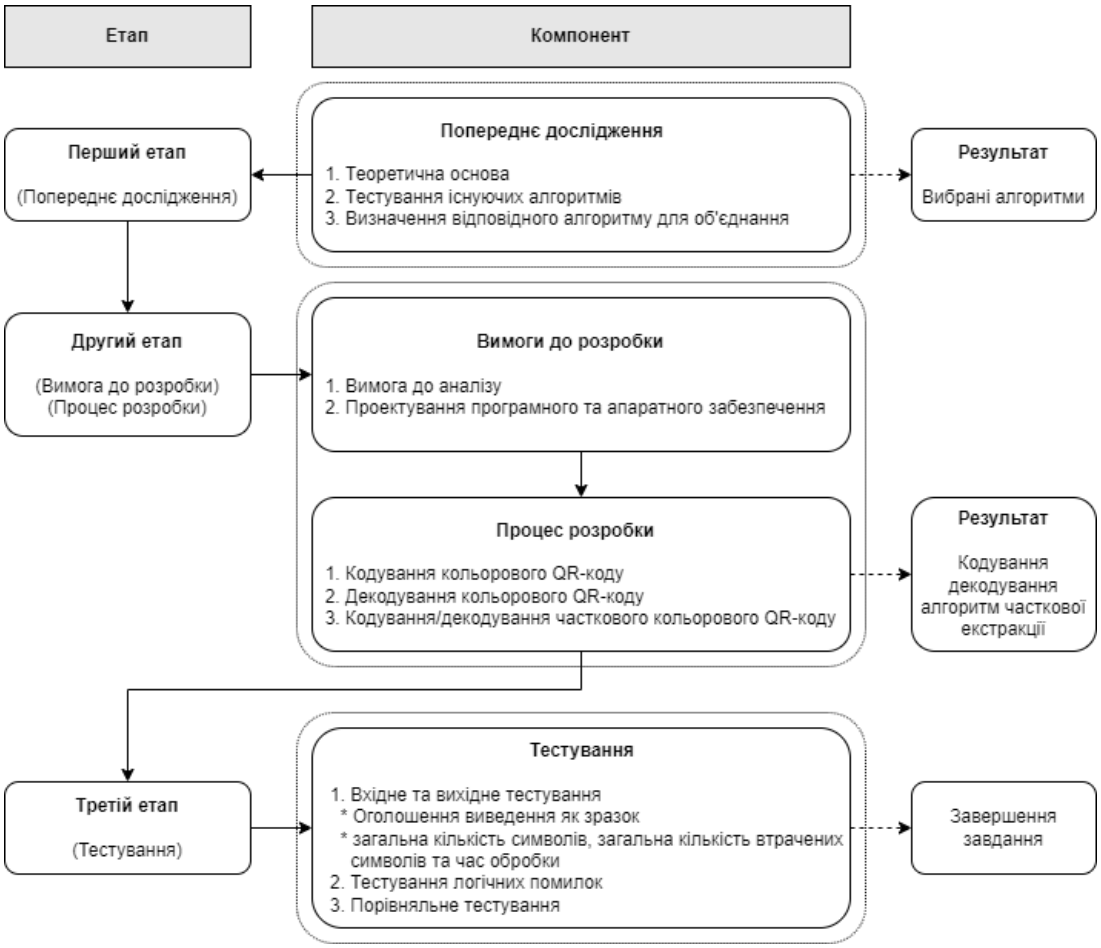


Рис. 2.1. Структура аналізу.

Теоретичні основи

Структура аналізу (рисунок 2.1) зосереджується на питанні експерименту, яке намагається ідентифікувати, проаналізувати, вибрати та синтезувати докази підтвердження, що стосуються цього питання. У цьому аналізі етапи включають наступне:

1. Аналіз методів збільшення пам’яті в QR-кодах.

2. Ідентифікація та збір інформації з однієї або кількох баз даних для пошуку за алгоритмом, що використовується в звичайному та кольоровому QR-кодах, зокрема для збільшення ємності та стиснення даних. Тут базами даних є звичайні електронні журнали, електронні статті, книги, документи, протоколи та інше. Саме аналіз спрямоване на визначення сучасного методу в QR-коді, пов’язаного з ємністю даних і алгоритмом стиснення.

3. Розробка стратегії пошуку для вибору відповідних алгоритмів, створених на основі попередніх досліджень. Причина цієї розробки полягає у визначенні найбільш актуальних алгоритмів. Критеріями вибору є успішне виконання, час обробки та загальна кількість збережених символів.

4. Вибір заголовків, теорії та реалізації на основі включення після їх ідентифікації на попередньому етапі.

На рисунку 2.2 показано теоретичне дослідження першого пункту на першому етапі стратегії дослідження.

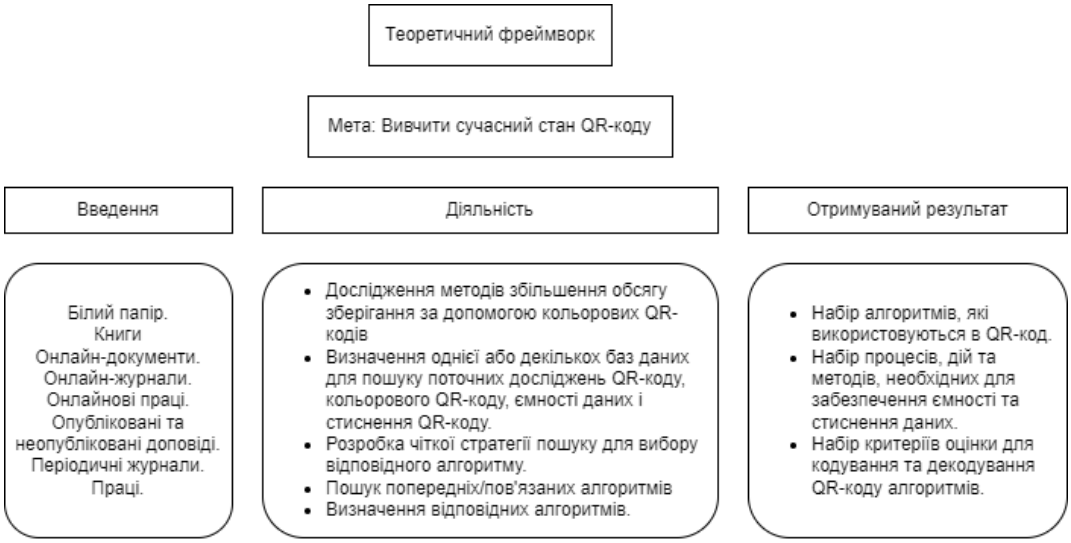


Рис. 2.2. Теоретичний фреймворк

Аналіз вибраних існуючих алгоритмів

Другий компонент більше стосується аналізування існуючих алгоритмів, які включають наступне:

1. Пошук блок-схем, псевдокодів алгоритмів, розроблених для збільшення обсягу даних кольорового QR-коду.

2. Збірка обраного алгоритму як своєрідна підготовка до процесу кодування. Псевдокоди або справжній код разом з блок-схемами організовують у зручний для читання формат, щоб полегшити процес кодування згодом.

3. Ідентифікація обладнання. Для розробки програми використовується персональний комп'ютер. Серед апаратного забезпечення можна виділити наступне:

- процессор Intel i7;

- оперативна пам’ять 24GB DDR3;
- 2TB пам’яті;

4. Ідентифікація ПО. Вибір програмного забезпечення включає в себе системне, прикладне програмне забезпечення та програмне забезпечення мовою програмування.

- Windows 7 і вище;
- Інтегроване середовище розробки Visual Studio Code;
- Python версії 3 і вище;
- ZXing, бібліотека для Python QR Code
- Програма графічного відображення, наприклад Microsoft Paint тощо.

5. Розробка процесів кодування та декодування. Процес розробки відповідає крокам алгоритму, створеного дослідником.

6. Аналіз кожного алгоритму на основі обсягу даних, часу обробки і виправлення помилок. Ці три методи аналузівання є методами, які найбільше оцінюють дослідники у своїх роботах.

На рисунку 2.3 показано дії та результати аналізу для вибору найкращого алгоритму для обсягу даних у QR-коді.

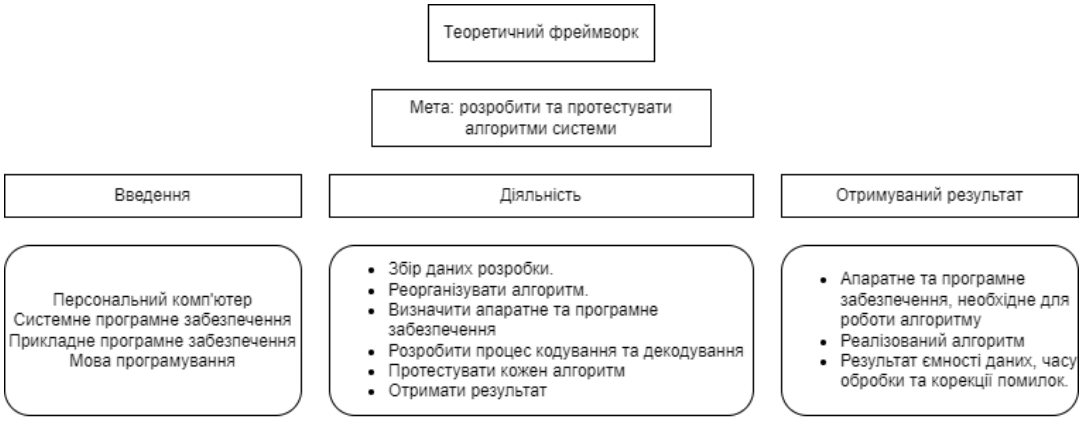


Рис. 2.3. Аналіз алгоритмів.

Об’єднання алгоритмів

В третій частині фокусується на об’єднанні трьох ідентифікованих алгоритмів.

1. Використання результатів аналіз для досягнення сумісності між трьома алгоритмами під час реалізації процесу злиття та отримання максимального обсягу даних, які можна зберігати.

2. Визначення пріоритету алгоритму на основі результатів аналізу.

3. Синхронізація введення та виведення між вибраними алгоритмами.

Рисунок 2.4 ілюструє певні дії з доопрацювання та об’єднання для розробки нового вдосконаленого QR-коду.

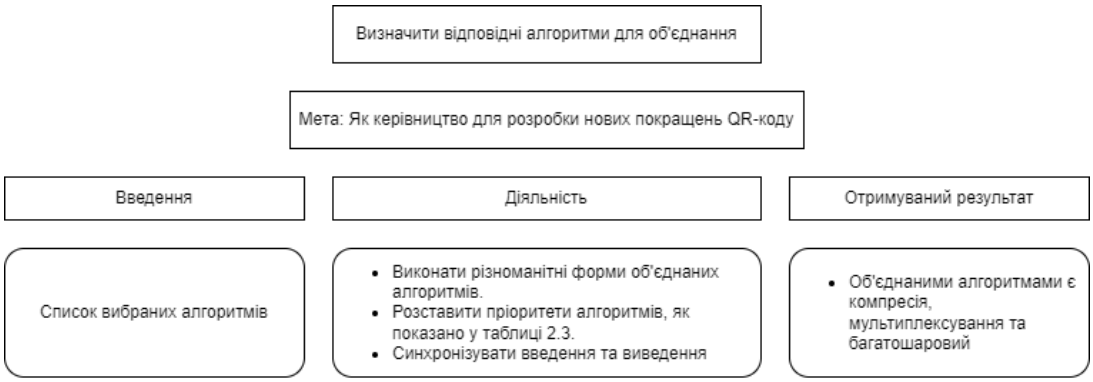


Рис. 2.4. Завершення та об’єднання.

2.1.2 Другий етап

Цей етап включає в себе дизайн і розробку кольорового QR-коду. Він розділений на дві частини: вимоги до розробки та сам процес розробки.

Вимога до розробки

На першому підетапі вимог до розробки досліджуються процеси, пов’язані з розробкою кольорового QR-коду. Цього можна досягнути проаналізувавши всю необхідну інформацію про обладнання, програмне забезпечення та етапи розробки кольорового QR-коду. Алгоритм кольорового QR-коду складається з кількох алгоритмів: стиснення, мультиплексування та багатошаровості. Ці алгоритми потрібні для інтеграції вхідних і вихідних даних. На рисунку 2.5 показано запропонований процес кодування та декодування кольорового QR-коду. Процеси розпакування даних з кольорового QR-коду показано на рисунку 2.6.

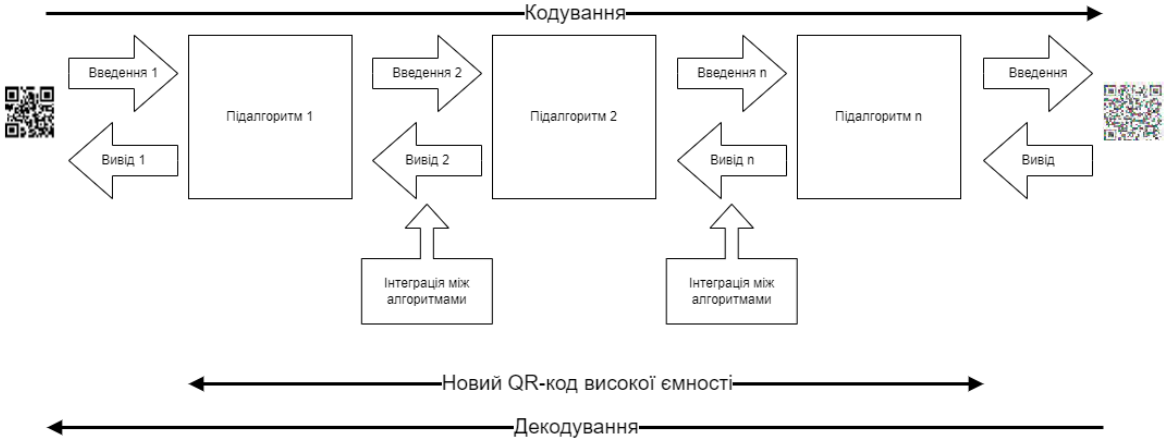


Рис. 2.5. Запропоновані процеси кодування для кольорового QR-коду

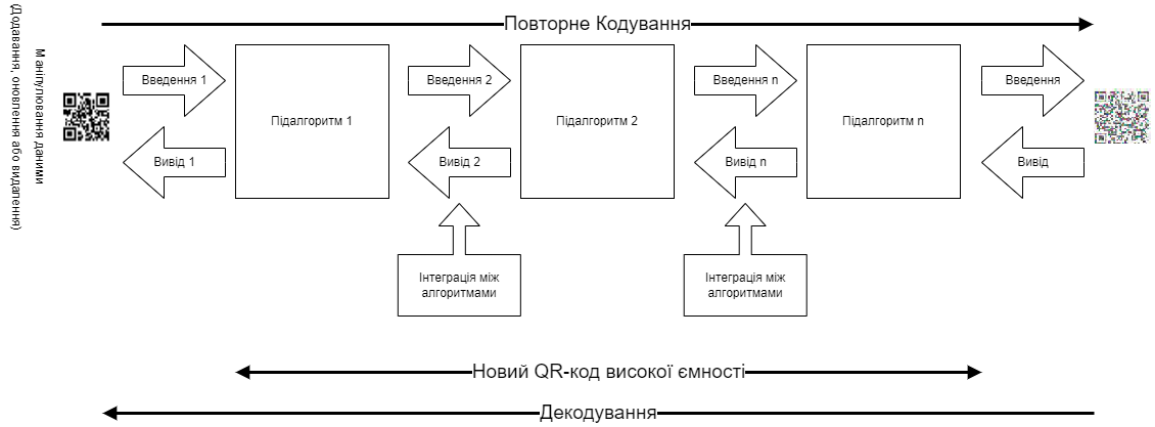


Рис. 2.6. Запропонований процес розпакування даних для кольорового QR-коду

Процес розробки

Етап розробки включає три основні процеси: кодування, декодування кольорового QR-коду та часткове розпакування QR-коду. Ці процеси включають написання окремих програм кодування чи декодування за допомогою бібліотек мови програмування Java та дотримання стилю коментарів Javadoc. Після цього відбувається об’єднання програм кодування та декодування. Далі йде розробка алгоритму часткового розпакування на основі необхідної інформації. Він базуватиметься на запропонованій реалізації кодування та декодування кольорового QR-коду.

На рисунку 2.7 показані етапи процесу розробки. Алгоритм від 1 до n складається з комбінації вибраних алгоритмів, вибраних з першої фази. Алгоритм 1 виконується в першу чергу під час процесу розробки. Тим часом, алгоритм n є останньою частиною, яка буде виконана, й саме він виведе

результат, що є кольоровим QR-кодом. Перед об'єднанням усіх алгоритмів кожен з них окремо розробляється на основі процесів кодування, декодування та часткового розпакування. Оскільки проектування та розробка кодування, декодування та часткового розпакування завершується на другому етапі, то перший, другий та третій етапи дослідження виконуються одночасно.

2.1.3 Третій етап

Після завершення процесів розробки виконується етап тестування. Введення починається зі здатності QR-коду зберігати символи всередині відповідно до його версії та рівня виправлення помилок. Розміщення символів у QR-кодах для версії 40 встановлено з обмеженою кількістю символів відповідно до рівня виправлення помилок, який наведено в таблиці 2.1. Розподіл символів базується на попередньому тесті.

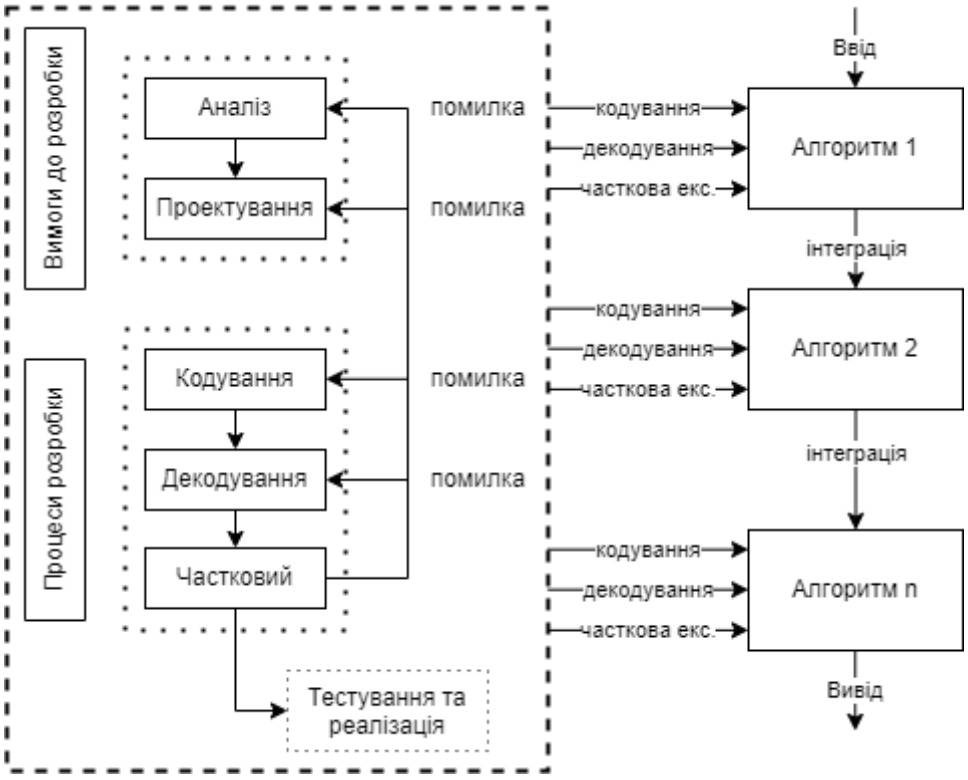


Рис. 2.7. Етапи процесу кодування.

Таблиця 2.1

Максимальна кількість символів залежить від рівня корекції помилок.

Рівень корекції помилок	Загальна кількість символів для кожного QR-коду
Мала	2952

Середня	2330
Висока	1662
Дуже висока	1270

Виявлення помилок та спотворення кольору не розглядаються у цьому дослідженні. Рівень корекції помилок, густина даних та час обробки використовуються для перевірки нової запропонованої ємності даних QR-коду.

Тестування

Тестування включає використання декількох вхідних і вихідних даних. Це необхідно для того, щоб переконатися, що запропонований алгоритм реалізовано відповідним чином. Тестування дозволяє визначити, що програма виконується добре, ефективно, стійко та надійно.

У цьому дослідженні використовується наступна метрика експерименту:

1. Щільність даних. Для перевірки загальної кількості ASCII-кодів, які можуть бути збережені у кольоровому QR-коді. Чим більше значення, тим краще алгоритм. Воно формулюється з урахуванням того, скільки символів успішно закодовано. Одиницею вимірювання, що використовується у цьому тестуванні, є загальна кількість символів, збережених у кольоровому QR-коді.

2. Порядок точності за рівнем корекції помилок. Для перевірки швидкості відновлення даних.

Чим вище значення, тим краще алгоритм. Одиницею вимірювання, яка використовується в даному тестуванні точності, є відсоток втрачених символів, який становить:

*(Загальне декодування символів одиниці виміру / Загальне кодування символів одиниці виміру) * 100*

1. **Час обробки.** Для перевірки загальної кількості часу, витраченого виконання процесу. Чим менше значення, тим краще алгоритм. Цей час обробки формулюється на основі:

Час закінчення завдання - Час початку завдання

Метрична одиниця, яка використовується при вимірі часу обробки - мілісекунду.

Тестування включає:

1. Тестування введення та виведення. Як вхідні дані використовується кілька символів. Це робиться для того, щоб перевірити максимальну кількість символів, яка може вмістити QR-код. Вихідні дані також повинні бути перевірені, оскільки вони допоможуть підтвердити збережену загальну кількість символів. Символи включають алфавітно-цифрові, цифрові, двійкові та кілька звичайних символів ASCII. Загальна кількість символів має бути ідентифікована у процесі декодування, а загальний час обробки буде розрахований у процесі кодування, декодування та часткового вилучення. Цей результат використовується як зразок для порівняння з іншими системами.

2. Логічні помилки. Це тестування проводиться для перевірки ненавмисного чи небажаного висновку чи іншої поведінки в алгоритмах, що призведе до їхньої неправильної роботи.

3. Порівняльне тестування. Порівняльне тестування включає вивчення результатів роботи запропонованих алгоритмів у порівнянні з QR версії 40 та іншими кольоровими QR-кодами високої ємності. QR версії 40 використовується як зразок для всіх порівняльних тестів, включаючи кольоровий QR-код високої ємності.

Після проведення тестування в рамках третьої фази мета дослідження 4 буде досягнута після завершення цього процесу.

ВИСНОВКИ ДО 2 РОЗДІЛУ

У цьому розділі висвітлено процедури дослідження, в яких дано докладний опис кроків, зроблених для завершення дослідження у цій дисертації. Процедури включають етапи попереднього дослідження, вимоги до розробки, процес розробки та аналізу. Крім того, в цьому розділі також обговорювався дизайн запропонованого алгоритму, який є кольоровим QR-кодом, що використовує комбінацію кольорів RGB. Такий внесок (див. четвертий розділ) включає деталі процесів кодування, декодування та часткового вилучення. Процес кодування забезпечує запропонований підхід, який може значно покращити щільність даних та час обробки кольорового QR-коду порівняно із звичайним QR-кодом. Процес декодування дозволяє отримати всі символи, які були успішно закодовані. Нарешті, процес часткового вилучення здатний поліпшити час обробки регенерації кольорового QR-коду для маніпулювання даними. З іншого боку, у цьому дослідженні також вивчається поєднання алгоритмів до створення кольорового QR-коду з більшою ємністю пам'яті. Надалі він тестується на основі трьох вимірів: щільність даних, точність та час обробки. Ця схема дослідження є керівництвом до виконання всіх поставлених завдань дослідження.

РОЗДІЛ 3

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОЄКТУ

У цьому розділі пояснюється дизайн та розробка кольорового QR-коду, починаючи з кодування, декодування та часткового вилучення. Проектування та розробка базуються на методології з другого розділу. Процес починається з проектування алгоритму кодування, за яким слідує розробка алгоритму кодування. Після цього буде продовжено проектування та розробку для декодування та часткового вилучення. Процеси містять ініціалізацію, підрахунок символів, створення чорно-білого QR-коду, розміщення пікселя тощо. Алгоритм, псевдокод та блок-схема відображаються, що може принести більше розуміння про перебіг процесів. Після того, як всі процеси будуть завершені, буде впроваджено емпіричний аналітик, щоб надати більший вплив на наступний результат дослідження.

3.1 Алгоритм кодування

Алгоритм кодування - це покроковий процес створення кольорового QR-коду. У цьому розділі представлені результати, отримані в ході експерименту, що зосереджується на кодуванні модулів на основі розробленого алгоритму. На початку всі модулі пояснюються в режимі псевдокоду, оскільки це легше пояснити. Потім всі псевдокоди будуть перетворені в алгоритми. Висновки базуються на часі обробки, загальній кількості успішно закодованих символів за типом рівня корекції помилок, загальному обсязі текстового файлу після стиснення та загальній кількості байт кольорового зображення QR-коду. У наступних розділах детально описані результати експерименту.

3.1.1 Модуль кодування

Загалом, модулі кодування, що використовуються в цьому дослідженні, застосовуються для створення повного кольорового QR-коду, який може зберігати велику кількість даних (зосереджуючись на текстових символах в

якості вхідних даних). У цьому дослідженні модулі кодування можна побачити на рисунках 2.5 та 2.6, які містять вимоги до розробки та процеси розробки на другому етапі. Технічно, другий етап містить підготовку до отримання запропонованої моделі для генерації кольорового QR-коду високої щільності. Процес кодування кольорового QR-коду включає стиснення, мультиплексування та багатошарові підмодулі.

3.1.2 Етапи кодування

При кодуванні кольорового QR-коду необхідно враховувати деякі кроки. Ці кроки включають наступні три підмодулі:

1. Стиснення. Етапи стиснення здійснюються шляхом забезпечення необхідної підготовки тексту та виконання стиснення.

2. Мультиплексування. Коли стиснення є успішним, процес мультиплексування буде підготовлений шляхом перетворення двійкового коду в текст, перетворення коду Американського національного інституту стандартів (ANSI) в Unicode Transformation Format 8 bit (UTF-8) та його мультиплексування.

3. Багатошаровий. Після завершення мультиплексування багатошаровий процес об'єднає червоний, зелений і синій QR-коди в кольоровий QR-код. Послідовність процесу кодування показана на рисунку 3.1.

У попередніх дослідженнях (див. підрозділи 1.3.2, 1.3.3 та 1.3.4) процеси кодування виконувалися на основі одного модуля, як пояснено в Розділі 1. Дослідники не зосереджувалися на об'єднанні інших модулів, щоб отримати додатковий обсяг пам'яті для QR-коду. Деякі з методів, що використовуються в кожному модулі, відрізняються від цього дослідження. Назви модулів подібні до цього дослідження, а результат зосереджений на створенні кольорового QR-коду.

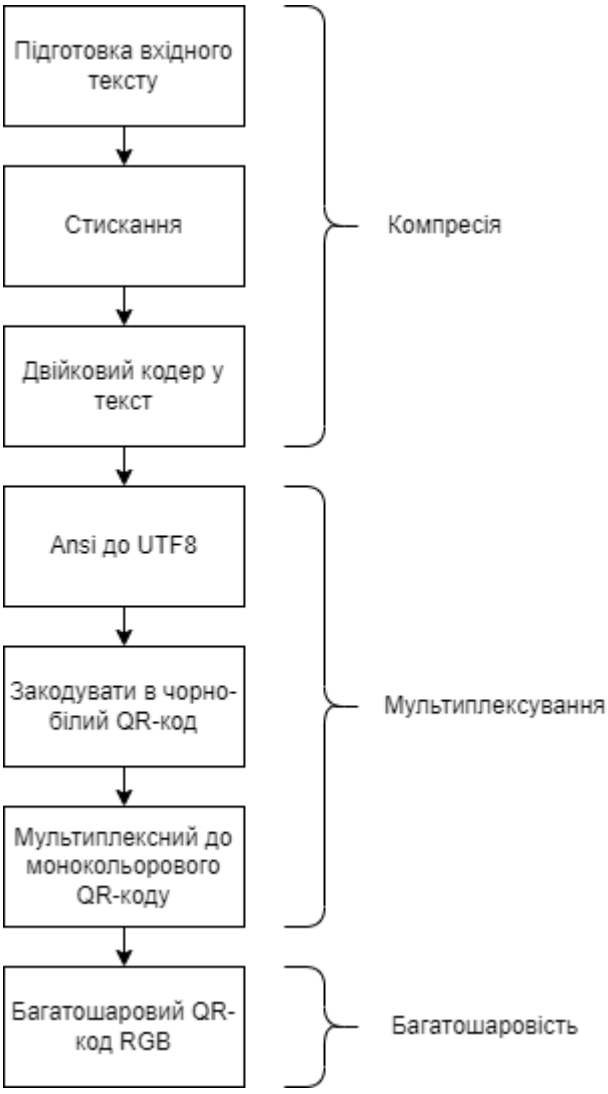


Рис. 3.1. Процес кодування потоку.

Модулі стиснення кодування

Алгоритм стиснення коду складається з двох модулів: модуля підрахунку символів і модуля стиснення. Ці модулі розділені через різні завдання, які вони виконують. Перший модуль фокусується на підрахунку модуля для визначення кількості ASCII символів, що друкуються та керуючих символів, тоді як другий модуль фокусується на стисненні файлу.

Модуль підрахунку символів

Це перший крок для виконання процедури стиснення, відомої як підготовка вхідного тексту. Процес стиснення розпочався з підрахунку загальної кількості символів до початку фактичного процесу стиснення. Символи були

підраховані в першу чергу, щоб уникнути надлишку символів, який міг би призвести до невдачі створення QR-коду. Карта кодів символів у цьому дослідженні відноситься до символів ASCII для друку (літери, цифри та символи) та двох керуючих символів, а саме повернення каретки та переведення рядка. Інші типи символів, крім зазначених вище, не підтримуються, наприклад, арабські, японські, китайські тощо.

Робота модуля починається з установки лічильника для підрахунку символів у файлі та ініціалізації його цілим нулем. Вхідний файл було ідентифіковано та ініціалізовано. Додатково для підрахунку групи унікальних символів використовується таблиця деревовидної карти символів. Символи повинні були надходити у форматі UTF-8. Після завершення ініціалізації процес починався з зчитування першого рядка вхідного файлу, і кожен символ у рядку розділявся на групу унікальних символів. Якщо зчитаний символ збігався з символом у таблиці деревовидної карти символів, то лічильник цього символу збільшувався на одиницю. Цей процес повторювався шляхом послідовного зчитування рядка до кінця файлу. Оскільки символи зчитувалися в режимі друку ASCII, то підраховувалися і контрольні символи ASCII. Етапи цього процесу починалися з читання першого рядка вхідного файлу. Якщо рядок містив символ повернення каретки або переведення рядка, то лічильник збільшувався на одиницю, виходячи з того, що було знайдено в рядку. Нарешті, вся інформація про кількість унікальних символів надсилалася до головної програми. Потоковий процес підрахунку символів показано на рисунку 3.2

Модуль компресії

Цей модуль стискає текстовий файл за допомогою інструментів стиснення. Утиліта стиснення використовувалася для зменшення обсягу текстового файлу. Текстовий файл був перетворений в бінарний файл за допомогою утиліти стиснення. Інструмент стиснення GZip був обраний на основі експерименту для отримання найкращого інструменту стиснення. Експеримент був розділений на два етапи. На першому етапі для стиснення текстового файлу в якості

експерименту було використано декілька утиліт стиснення. Серед них: Lempel-Ziv- Welch (LZW) (Victor, 2012), Zip (Kattan & Poli), GZip (Husain, Bakhtiari, & Zainal), Huffman Coding (Shahbahrami, Bahrampour, Rostami, & Mostafa Ayoubi), Huffman Coding об'єднали з GZip (Oswal, Singh & Kumari), і Huffman Coding об'єднали з Zip (Pathak et al.). Вхідні дані були взяті з символів друкованого режиму ASCII і були повторені 20 разів для досягнення мінімального значення стиснення. Після завершення стиснення всі стиснуті елементи зберігалися в QR-коді з рівнем корекції помилок H. Стиснуті дані всередині QR-коду мають двійковий тип файлу.

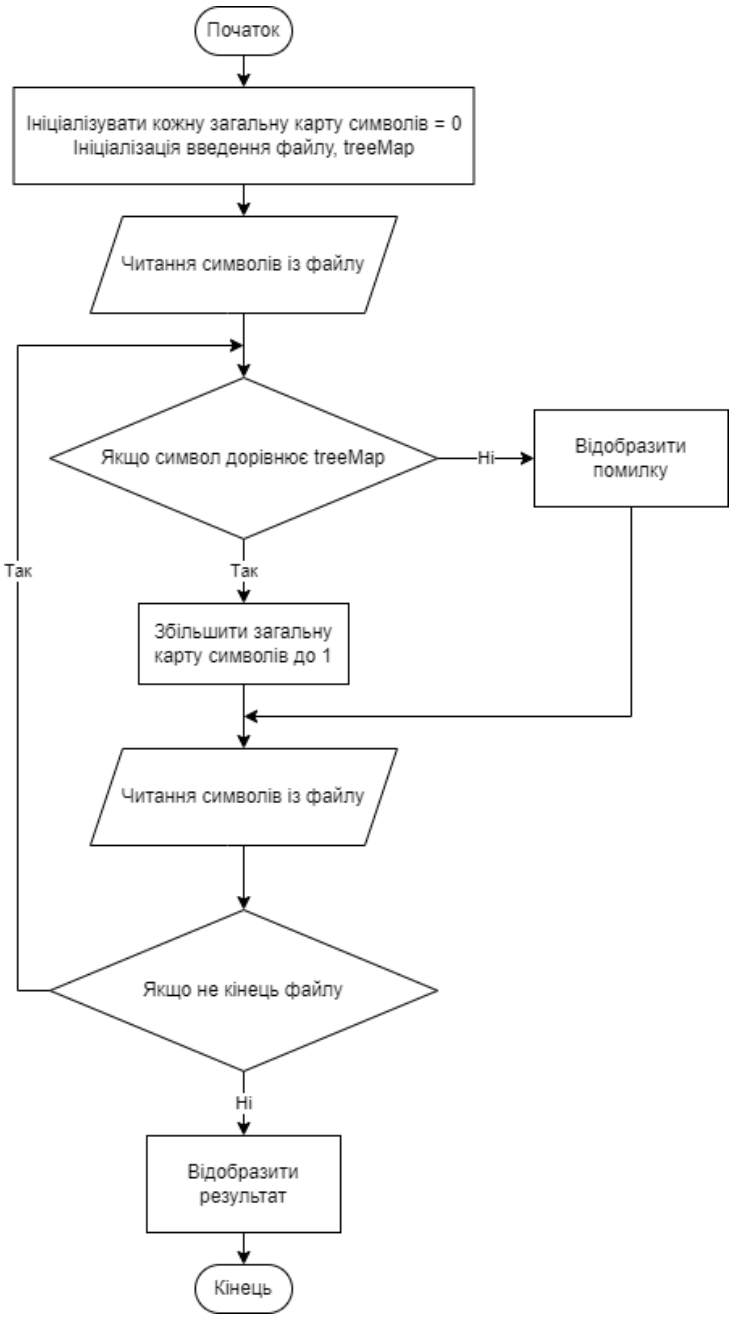


Рис. 3.2. Блок-схема модуля підрахунку символів.

Результати цього етапу наведені в таблиці 3.1, мінімальне значення загальної кількості символів з 20-кратного повторення експерименту з рівнем корекції помилок H.

Рівняння 3.1 може бути сформульоване так, як показано нижче:

$$A = \{a_i\}^N{}_{i=0} \tag{3.1}$$

A = набір елементів

a = елемент

i = індекс

N = загальна кількість елементів

яка позначається min A або міні ai, і дорівнює першому елементу відсортованого символу (тобто по порядку) з множини A.

Таблиця 3.1

Мінімальне значення загальної суми символу від 20-кратного повторного експерименту з рівнем виправлення помилок H.

Нормальний	Zip	GZip	LZW	Кодування Huffmann	Huffmann і GZip
1271	469	632	433	109	466

Метод без стиснення є найкращим способом зберігання даних у QR-коді. Він може зберігати до 1,271 символів у порівнянні зі стисненим файлом. Причина, чому цей результат не може забезпечити більше символів для зберігання в QR-коді, полягає в тому, що QR-код має більше можливостей для зберігання даних в алфавітно-цифровому форматі в порівнянні з двійковим. У таблиці 3.1 показано кількість символів, які можуть бути збережені в чорно-білому QR-коді версії 40 за типами символів.

Псевдокод для стиснення текстових даних за допомогою утиліт стиснення, був розроблений як орієнтир для виконання фактичної дії.

Таблиця 3.2

Кількість символів, які можна зберігати в чорно-білому QR-коді версії 40 за типом символів.

Режим введення	Макс. символи	Можливі символи, кодування за замовчуванням
Тільки числові	7,089	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Буквено-цифровий	4,296	0–9, A–Z (лише верхній регістр), пробіл, $, %, *,+, -, ., /, :
Двійковий/байт	2,953	ISO 8859-1
Кандзі/Кана	1,817	Shift JIS X 0208

На другому етапі на стиснуті файли було накладено кодер/декодер Base64. Причина полягає в тому, щоб отримати більше символів для стиснення, щоб QR-код міг зберігати більше даних. Схеми кодування Base64 можуть бути використані для кодування двійкових даних у текстові дані, які необхідно зберігати та передавати через носії. Існують різні кодувальники, які можуть бути використані в цьому експерименті, але кодер/декодер Base64 був обраний тому, що час кодування та декодування є швидшим, ніж у інших декодерів.

Фіксований склад символів був закладений як тестові дані. Результати були розділені за рівнем виправлення помилок, як показано в таблиці 3.3. Кожен експеримент проводився лише один раз, оскільки у вхідному файлі використовувалися символи фіксованого складу, а алгоритм стиснення генерував файли однакового розміру. З результатів, наведених у таблиці 3.3, видно, що найбільший сумарний символ виходить при стисненні GZip. Наприклад, QR-код може містити до 1 784 символів на рівні H у версії 40. Стиснення GZip може перевищувати більш ніж на 40% стиснення даних у порівнянні зі звичайним текстом.

Таблиця 3.3

Максимальна загальна кількість символів, що зберігаються в QR-коді за рівнем помилки.

Рівень помилок	Нормальний	Zip	Gzip	LZW	Huffman Кодування	Huffman І Gzip	Huffman І Zip
H	1270	1560	1784	1167	212	1364	1166
Q	1662	2114	2405	1627	282	1827	1639
M	2330	3188	3470	2441	392	2607	2425

L	2952	4226	4480	3253	503	3323	3095

Модулі кодування мультиплексування

На цьому етапі експеримент починається з розподілу символів у відповідні файли відповідно до кількості символів, зазначеної перед генерацією чорно-білих QR-кодів. Цей процес завершується розробкою червоного, зеленого та синього QR-кодів. Послідовність процесу включає в себе перетворення ANSI в UTF-8, створення пустих файлів, розподіл символів по пустих файлах, створення чорно-білих QR-кодів та створення червоних, зелених та синіх QR-кодів.

Перетворення модуля ANSI на модуль UTF-8

Конвертація була необхідна, оскільки кодер Base64 при перетворенні двійкового коду в текст видавав формат ANSI. Як завжди, при створенні чорно-білого QR-коду, текстові дані повинні були бути у форматі UTF-8. Причини використання формату UTF-8 наступні:

1. Більшість пристроїв сканера використовують формат JIS8 (QR 2000) для сканування даних, сумісних з форматом UTF-8. JIS - це японські промислові стандарти кодування японської мови, і він забезпечує підтримку перетворення набору коду для UTF-8.

2. *Бібліотека ZXing*, яка використовувалася для створення QR-коду, може отримувати тільки формат UTF-8. Це обмеження для даного експерименту і єдиним рішенням є перетворення всіх виходів з отриманого кодера в формат UTF-8.

При перетворенні ANSI в UTF-8 використовувався клас, наданий бібліотекою Java під назвою *Charset* з кодуванням методу.

Модуль створення порожніх файлів

Після завершення процесу кодування наступним кроком було створення порожніх файлів. Цей процес як один із попередніх процесів перед фактичним виконанням процесу мультиплексування. Порожні файли повинні були бути створені, оскільки процес мультиплексування повинен був бути підготовлений з

певною кількістю порожніх файлів, перш ніж вхідні дані можуть бути вбудовані в пов'язані файли. У цьому експерименті *були* створені N значення порожніх файлів, які розділилися на три групи, які представляли по вісім файлів.

Причина, по якій файл повинен бути створений у восьми файлах кожен, буде згадана в наступному пункті. Три речі були враховані під час ініціалізації порожніх файлів, які були загальною максимальною кількістю символів, загальною кількістю використаних файлів та розташуванням файлу, яке потрібно записати. В основному виділення максимальної кількості символів базувалося на попередньому результаті другого стовпця таблиці 3.3. Рівень виправлення помилок L споживав 2,952 максимум символів і він використовувався в експерименті для мультиплексування. Тим часом, загальні файли, використані в цьому експерименті, були порожніми *файлами значень N*, і вони були розташовані в поточному каталозі виконання програми. Під час процесу створення порожніх файлів з'явилася інструкція додати символи 'пробіл' до всіх порожніх файлів. Іншими словами, в кожен файл було вбудовано 2,952 символи "пробілу". Причина, чому ця дія була вжита, полягає в тому, щоб гарантувати, що всі QR-коди стануть версією 40 після завершення процесу створення чорно-білих QR-кодів, навіть якщо інформація не була виконана для повного *значення N* чорних і білих QR-кодів. По суті, версія QR-коду з цього експерименту буде створена на основі загальної кількості вбудованих символів.

Модуль розділення символів

Для наступної частини символи з вхідних файлів були розділені певними виділеними загальними символами. При цьому для кожного файлу було виділено 2 952 символи. Таблиця 3.3 коротко показує розподіл файлів символів. Було висунуто кілька умов, щоб уникнути логічної помилки при виконанні. Умови включають, що якщо загальний вбудований символ перевищив більше, ніж файли значень N, програма автоматично не зможе продовжити. Відобразиться повідомлення про помилку. Більше того, якби вбудовані символи не перевищували (менше), ніж файли значень N, програма продовжувала б

виконуватися, але зберігала б збалансованість символів "пробілу" всередині файлу, який був вбудований раніше. Усі символи "пробілу" були замінені під час вставки закодованих символів.

Таблиця 3.4

Розподіл файлів символів.

	Один файл	Вісім файлів	Двадцять чотири файли	N файлів
Символів	2,952	8 * 2,952 23,616	24 * 2,952 70,848	N * 2,952 2,952*N

Два розрахунки були зроблені ще до того, як процес почав виконуватися. Вони використовувалися для ідентифікації точної загальної кількості файлів , які будуть використовуватися. Перший розрахунок - це:

Загальна кількість файлів = загальна кількість символів */ максимальна кількість символів у кожному файлі*

Якби значення мало залишок, воно зайняло б тільки ціле значення. Після цього наступним розрахунком було отримання значення мода, яке становить:

Загальне значення мода = загальна кількість символів % максимальної кількості символів у кожному файлі

Коли ці розрахунки були завершені, це збільшило б загальний файл на одиницю, якби загальне значення мода мало залишок. Решта були символами балансу для останнього файлу. Якби символів було менше 2,952 символів, решта символів "пробілу" в останньому файлі залишалася б назавжди разом із символами балансу, щоб завершити 2,952 загальних символів в одному файлі.

Коли файли були готові до отримання символів, всі символи були розділені на 2,952 символи кожен і поміщені у файл, поки символи не були повністю переміщені до всіх пов'язаних файлів. Якщо символи не досягли достатньої кількості 2,952 символів наприкінці процесу, баланс буде замінено символами "пробілу" назавжди та збережено в останньому файлі. В кінці процес відправив загальну кількість файлів, завершених для наступного процесу.

Створення чорно-білого модуля QR-коду

У цій частині він зосередиться на створенні *значення N* чорних і білих QR-кодів. Він використовував файл, який містив розділені символи, як вхідні дані. Першим кроком була ініціалізація введення тексту і графічного вихідного файлу. Графічним вихідним файлам були присвоєні імена, оскільки це полегшило б наступний сеанс виконання. Після завершення ініціалізації процес витягнув дані з вхідних файлів, прочитавши їх і ввівши в масив даних на модулі створення чорно-білих QR-кодів. Процес відкликав (об'єктно-орієнтований) той самий модуль як частину для створення чорно-білого QR-коду, оскільки кількість вхідних файлів була більше одного.

Деякі критерії потрібно було застосувати перед створенням чорно-білого QR-коду. Цими критеріями є ініціалізація висоти, ширини, кольору, набору символів, рівня виправлення помилок та поля. Після завершення цих параметрів чорно-білий QR-код був створений за допомогою бібліотеки *ZXing*, яка містила клас *QRCodeWriter*. Цей клас створив байтову матрицю, яка містила графічні пікселі QR-коду. Ця байтова матриця використовувалася для генерації зображення QR-коду з виконанням буферного зображення та графічного зображення. Дизайн чорно-білого QR-коду був сформований шляхом вказівки розташування осі x та пікселя осі y на піксель.

Створення червоного, зеленого та синього модуля QR-коду

Цей метод є останнім методом для процесу мультиплексування. Завдання процесу була розділена на три підпроцеси. Підпроцеси включали читання зображення та перетворення його на 8-бітну структуру, збір інформації та створення червоних, зелених та синіх QR-кодів. На першому кроці була захоплена вся інформація щодо кольорового пікселя з чорно-білого QR-коду, створеного раніше.

Зображення були прочитані після ініціалізації імен файлів зображень , загальної кількості файлів зображень та розміру зображень (висота та ширина). У цьому експерименті *значення N* чорних та білих QR-кодів було використано для створення червоних, зелених та синіх QR-кодів. Було призначено масив

буферного зображення та збережено інформацію щодо кольорів пікселів *значення N* чорно-білих QR-кодів. Іншими словами, масив буферного зображення зберігав інформацію піксельних кольорів на основі загальної кількості чорно-білих QR-кодів. Масив буферного зображення зберігав інформацію колірного типу за допомогою RGB-схеми в точці осі x і осі y чорно-білого QR-коду. У схемі RGB було використано лише два типи кольорів, які були чорно-білими.

Після того, як ця інформація була зібрана, наступним процесом є перетворення кольору в цифру. У цьому випадку чорний колір був призначений як 1, а білий - як 0. Код у схемі RGB для чорного кольору був RGB(000 000 000), а білого — RGB(255 255 255). Наприклад, вісім чорно-білих QR-кодів були обрані та об'єднані через розташування осі x та індексу осі y, щоб отримати червоний QR-код. Індекси осі x та осі y з восьми чорно-білих QR-кодів у місці розташування (0,0) були об'єднані як 8 біт двійкового числа, оскільки їх було лише 0 та 1. Після цього двійкові числа були перетворені в десяткове число, яке знаходилося між діапазоном від 0 до 254. Ці цифри розташовувалися в першій колонці схеми RGB. Другий і третій стовпці були представлені зі значенням 0. Ця інформація була представлена в місці розташування пікселів (*0,0*). Для іншого розташування він буде слідувати тим же методом до кінця розташування пікселя, яке було розташуванням пікселя (ширина -1*, висота -1*). Для синього та зеленого QR-кодів процеси все ще були такими ж, як і створення червоного QR-коду. У цьому випадку для мультиплексування він згенерував три QR-коди, а саме червоний, зелений та синій QR-коди. На рисунку 3.3 наведено приклад першого процесу перетворення двійкового числа в десяткову кому в індексному розташуванні (0,0) для кожного чорно-білого QR-коду та присвоєння значення розташуванню індексу (0,0) за червоним QR-кодом. Створення червоних, зелених та синіх QR-кодів відбувалося з масиву графічних зображень, які містили колір кожного пікселя. Процес мультиплексування використовував вісім зображень і створив одне зображення, об'єднавши двійкові числа в 8 біт.

Кодування багатошарових модулів

Після завершення процесу мультиплексування багатошаровий процес перейде на об'єднання червоного, зеленого та синього QR-кодів. Цей процес зчитує весь кольоровий код з кожного місця індексу, починаючи з (0,0) до (ширина - 1, висота -1) зображень з червоного, зеленого та синього *QR-кодів*. Те саме індексне розташування червоних, зелених та синіх QR-кодів було змішано в колірній схемі RGB, як *Колір (*x, y, z), який вказував *x* як кольоровий код червоного QR-коду, *y* для зеленого QR-коду, *a z* представляв синій QR-код. З цього експерименту багатошарові QR-коди складалися з 93 987 процесів, які обчислювалися як:

(177 (розмір піксельної осі x) * *177 (розмір піксельної осі y)) * 3 (червоні, зелені та сині QR-коди*)).

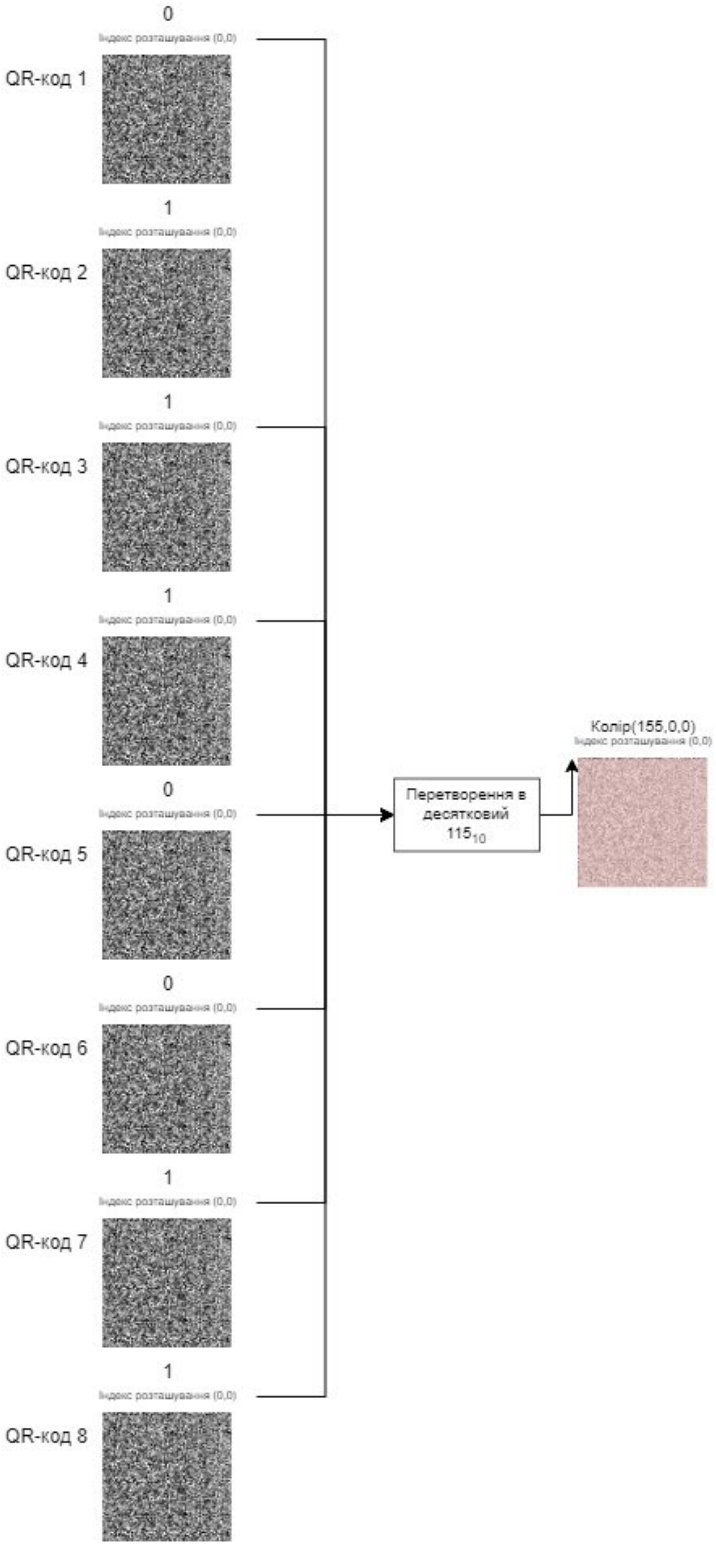


Рис. 3.3. Приклад першого процесу перетворення двійкового числа в десяткову кому в розташуванні індексу (0,0) для кожного чорно-білого QR-коду та присвоєння значення розташуванню індексу (0,0) за червоним QR-кодом .

На початку були ініціалізовані загальні файли зображень, імена файлів зображень (червоні, зелені та сині QR-коди), розташування зображення та кінцевий багатошаровий файл зображення. Однак загальні файли зображень,

імена файлів зображень (червоний, зелений та синій QR-коди) та розташування зображення були встановлені як параметри для багатошарового алгоритму. Буферне зображення для кінцевого багатошарового файлу зображення ініціалізувало розмір ширини та висоти, а також тип кольору. Вся піксельна інформація з червоного, зеленого та синього QR-кодів була об'єднана (як згадувалося в попередньому пункті) та присвоєна графічному зображенню. Після завершення процесів графічне зображення присвоїло графічному файлу значення кольору, щоб створити зображення.

3.2 Алгоритм декодування

У цьому розділі описано процес розшифровки кольорового QR-коду. Всі модулі будуть пояснені у вигляді псевдокоду і будуть перетворені в алгоритми. Експеримент проведеного експерименту базується на витраті часу, загальній кількості витягнутих QR-кодів та рівні виправлення помилок. Результати ґрунтуються на експерименті, проведеному за методами, використаними в даному дослідженні. Псевдокоди розробляються як орієнтир перед розробкою реальної програми. У цьому розділі також наведено певний результат моделювання, якщо реалізовано кольоровий канал та глибину кольору. Наступні розділи детально описують висновки експерименту з розшифровки.

3.2.1 Декодування QR-коду

Процес декодування полягає в отриманні фактичних даних, які були закодовані раніше. По суті, кольоровий QR-код є кінцевим фізичним результатом процесу кодування. У ньому міститься деяка інформація, закладена всередині нього. Інформація представлена кольоровим QR-кодом. Коли кольоровий QR-код успішно закодований, його потрібно розшифрувати, щоб отримати дані, що зберігаються в ньому. Це тому, що коли користувачі отримали кольоровий QR-код, вони повинні знати, яка інформація всередині. Користувачі не можуть перевести зображення кольорового QR-коду у фізичну форму, через що його потрібно перекласти за допомогою методу декодування.

Загалом, розшифровка починається з демультишарового процесу від кольорового QR-коду до червоного, зеленого та синього QR-кодів. Після цього процес демультиплексування перейде до перетворення червоного, зеленого та синього QR-кодів у 24 чорно-білі QR-коди. Нарешті, процес декомпресії перетворить вміст всередині чорно-білих QR-кодів у оригінальний текст.

3.2.2 Етапи декодування

Процес декодування - це зворотний процес методу кодування. Є кілька етапів, які потрібно обробити при підтримці процесів декодування, а саме:

1. **Демультишарування**. Цей крок є першим кроком для процесу декодування. Як правило, кольоровий QR-код буде перетворено на червоні, зелені та сині QR-коди. Цей процес розбиває колірний код RGB на окремі унікальні коди червоного, зеленого та синього кольорів.

2. **Демультиплексування**. Після завершення процесу демультишару процес демультиплексування буде підготовлений шляхом перетворення червоного, зеленого та синього QR-кодів у чорно-білі QR-коди. Загальна кількість чорно-білих QR-кодів становить 24.

3. **Декомпресія**. Останній процес полягає у вилученні вмісту всередині чорно-білих QR-кодів. По-перше, йому потрібно розшифрувати чорно-білі QR-коди. В результаті двійковий файл GZip виробляється і готовий до вилучення.

Розшифровка потоку представлена на рисунку 3.4.

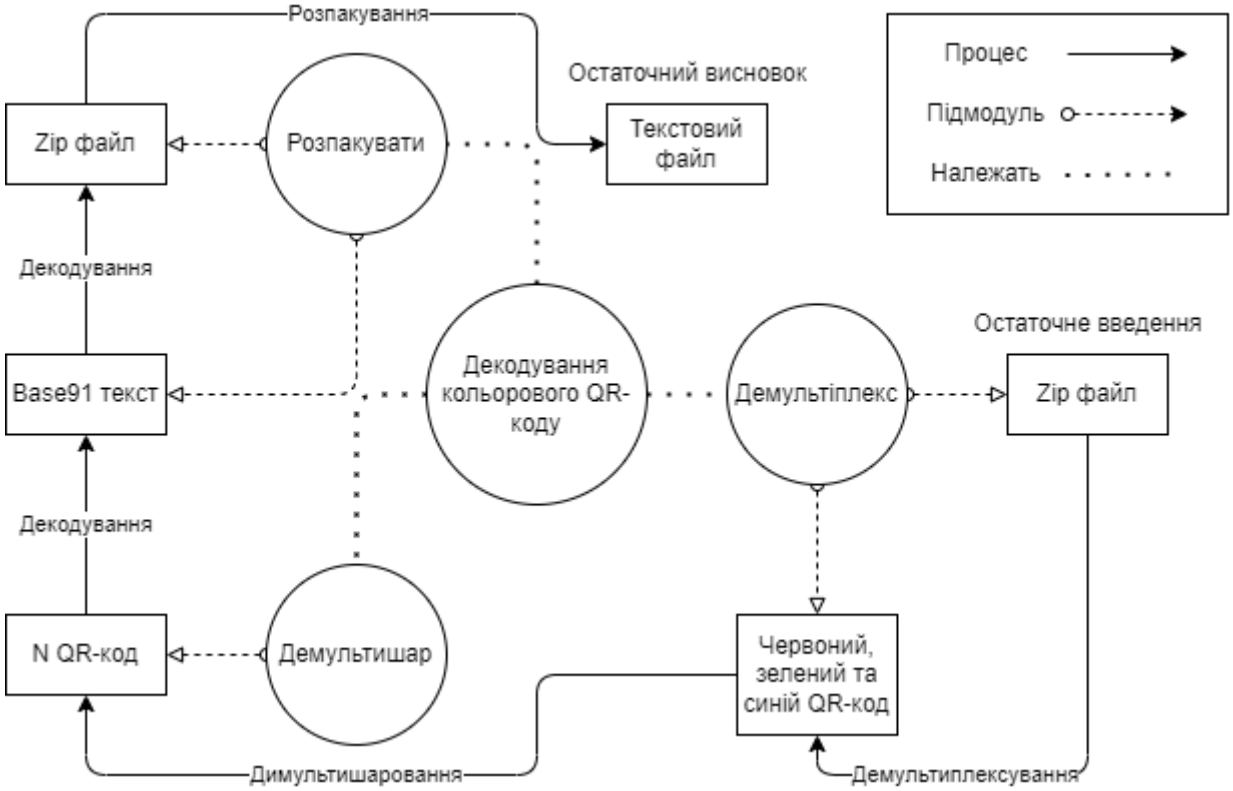


Рис. 3.4. Процес потоку декодування.

Основний алгоритм був розроблений з деякою модифікацією процесу кодування. Незважаючи на те, що процес декодування є оберненим до процесу кодування, деякі процеси потребують реінжинірингу. Основний алгоритм містить деталі демультишарового, демультиплексування та декомпресії.

Модуль ініціалізації

Це перший крок, який необхідно виконати, перш ніж завершити всі процеси. Декодування демультишарового псевдокоду слід спочатку ініціалізувати, перш ніж він зможе перейти до виведення червоного, зеленого та синього QR-кодів. Значення складалися з типу файлу, введення файлу, виведення файлу та розташування файлу кольорової інформації QR-коду. Наведена вище інформація була використана для того, щоб розбити кольоровий QR-код на три монокольорові QR-коди. Потім він отримав інформацію про колір пікселів з червоного, зеленого та синього QR-кодів як вихід. Вся наведена вище вхідна інформація повинна була бути використана в наступному модулі демультиплексування.

Модуль демультишарового декодування

Детально, кольоровий QR-код був згенерований комбінацією зображень файлу QR-коду червоного, зеленого та синього кольорів. Зображення файлів були створені на основі унікальних кольорів червоного, зеленого та синього.

Завдання починалися з призначення вмісту інформації про колір пікселя всередині кольорового QR-коду. На основі цієї інформації файли вихідних зображень були створені спочатку для зручності розподілу пов'язаних кольорів пікселів на пов'язані червоні, зелені та сині QR-коди. Весь вміст інформації про колір пікселів всередині кольорового QR був перетворений у буфер зображення, перш ніж перейти до пов'язаних червоних, зелених та синіх QR-кодів. Спосіб передачі інформації про колір пікселя полягав у поділі кожного елемента колірної кодової схеми RGB на червоні, зелені та сині елементи, як обговорювалося раніше. Загальна кількість кольорових пікселів базувалася на розмірі кольорового зображення QR-коду.

Експеримент порядку часу , що минув , за рівнем корекції помилок при декодуванні демультишару був виконаний успішно і результат наведено в таблиці 3.5. З результатів минулі часи вважалися мало чим відрізняються за часовим діапазоном серед рівнів виправлення помилок. Виправлення помилок рівня M виконало своє завдання за 0с 099мс.

Таблиця 3.5

Експеримент з упорядкування минулого часу за рівнем корекції помилок.

Рівень виправлення помилок	Демультишарування
L	0с 103мс
M	0с 100мс
Q	0с 101мс
H	0с 099мс

Модуль декодування демультиплексування

Наступним процесом є демультиплексування, яке полягало в демультиплексуванні з червоного, зеленого та синього QR-кодів на *N* значень

чорних та білих QR-кодів. Цей процес вилучив кожне значення пікселя, яке було в десятковому значенні, кожного червоного, зеленого та синього QR-кодів кольору RGB у двійкові числа 8 біт. Після того як були визначені двійкові числа , їх розбили на вісім окремих одиниць 8-бітних двійкових чисел. Кожна окрема цифра двійкових чисел служила білим (0) або чорним (1) піксельним кольорами кожного чорно-білого QR-коду. Наприклад, скажімо, що розташування пікселів (0,0) червоного QR-коду має RGB(45,0,0). Тоді 8-бітове двійкове число 00101101. Перша цифра двійкової - 0, яка представляє білий колір першого чорно-білого QR-коду в місці розташування пікселів (0,0). Те ж саме і з другим і восьмим чорно-білими QR-кодами. Цей метод показує одну з багатьох концепцій, які були реалізовані в процесі демультиплексування.

Процес потоку розпочався з ідентифікації червоної, зеленої та синьої інформації про зображення QR-коду та створення групи *Mn* чорних та білих QR-кодів кожного червоного, зеленого та синього QR-кодів. Процес демультиплексування розпочався з вилучення десяткового значення червоного елемента червоного колірного коду QR RGB для кожного місця розташування пікселя, починаючи з місця розташування (0,0) до (177,177). Останнє розташування пікселів визначалося розміром ширини і висоти зображення. Наступний процес продовжився зеленими та синіми QR-кодами, використовуючи умовну заяву, оскільки вони повинні виконуватися один за одним. Десяткові значення мали місце лише для пов'язаних між собою елементів червоного, зеленого та синього кольорів. Наприклад, якщо RGB для червоного QR-коду дорівнює RGB(46,0,0), він присвоїть місце тимчасового зберігання десяткове значення 46. Вся кольорова інформація червоного QR-коду була зібрана та збережена в місці тимчасового зберігання пам'яті. Тимчасове сховище зібрало всю інформацію про зелений і синій QR-коди в десятковому значенні. Коли були зібрані всі десяткові значення, наступним процесом було перетворення десяткового значення в двійкове число. Десяткові значення були призначені як цілочисельні значення і перетворені в рядковий режим, щоб легко маніпулювати отриманими символами. Розрахунок перетворення був

ідентифікований нормальним математичним розрахунком. Десяткове значення ділили на два, а потім записували залишок. Цей процес повторювався до тих пір, поки його вже не можна було розділити на два. Для прикладу візьмемо десяткове значення 157, як показано в таблиці 3.6.

Таблиця 3.6

Десятковий до двійкового процесу.

Ділення значення (÷)	**Значення модуля (%)**
157 ÷ 2 = 78	із залишком 1
78 ÷ 2 = 39	із залишком 0
39 ÷ 2 = 19	із залишком 1
19 ÷ 2 = 9	із залишком 1
9 ÷ 2 = 4	із залишком 1
4 ÷ 2 = 2	із залишком 0
2 ÷ 2 = 1	із залишком 0
1 ÷ 2 = 0	із залишком 1

Далі значення залишків знизу вгору представляло значення двійкового числа, яке було 10011101.

Після того, як двійкові числа були зібрані з кожного червоного, зеленого та синього QR-коду, наступним кроком було розбивку їх на один біт та позначення розташування індексу кожного біта. Коли один біт був ідентифікований, колір пікселів позначався або чорним, або білим. Кольоровий піксель розпізнавався, коли один біт давав значення 0 або 1, при цьому 0 було для білого, а 1 - для чорного. Під час цього процесу при визначенні кольору було враховано три пункти: тип кольору QR-коду, розташування пікселів та індекс розриву 8 бітів двійкового числа на одинарну комбінацію цифр 0 (білий) або 1 (чорний). Процеси комбінування, як показано нижче:

Тип QR-код x = червоний; y = зелений; z = синього Розташування пікселя = a(0,0) ~ *a(177,177)*

Розташування індексу 8 біт на одинарний біт= p(0) ~ *p(7)*

Визначення чорного або білого кольору можна показати у вигляді набору {тип QR коду}{розташування пікселя}{одиночний біт}, який детально називається $\{x_i\}\{a_{ij}\}\{p_i\}$. На рисунку 3.5 нижче представлена блок-схема при визначенні чорних або білих пікселів чорно-білих QR-кодів.

Декодування чорно-білих QR-кодів

Коли 24 чорно-білі QR-коди були повністю згенеровані, наступним кроком було декодування їх у текстовий файл. Процес декодування полягав у розшифровці кожного чорно-білого QR-коду, об'єднанні декодованих текстів та збереженні в логічному файлі. Процес розшифровки QR-коду почався з виклику першого чорно-білого QR-коду з червоного QR-коду і розшифровки його в текст. Текст, що міститься в кожному чорно-білому QR-коді, був частиною повного тексту під час попереднього процесу кодування. Цей процес тривав до останнього чорно-білого QR-коду зеленого QR-коду. Відповідним модулем декодування чорно-білого QR-коду є *MultiFormatReader*.

Після того як процес декодування тексту був завершений, наступним кроком об'єднання текстів. Тексти були об'єднані за допомогою класу рядків, який міг зарезервувати близько 2 147 483 647 (2^{31} - 1) символів загалом. Кожній текстовій інформації, що зберігається в кожному чорному QR-коді, було присвоєно за допомогою оператора '+'. Нарешті, вся текстова інформація, що зберігається в класі буферних рядків, зберігалася в каталозі файлів за допомогою класу *File*. Текстовий файл у файловому каталозі містив текстовий формат Base64, який повинен бути декодований його декодером.

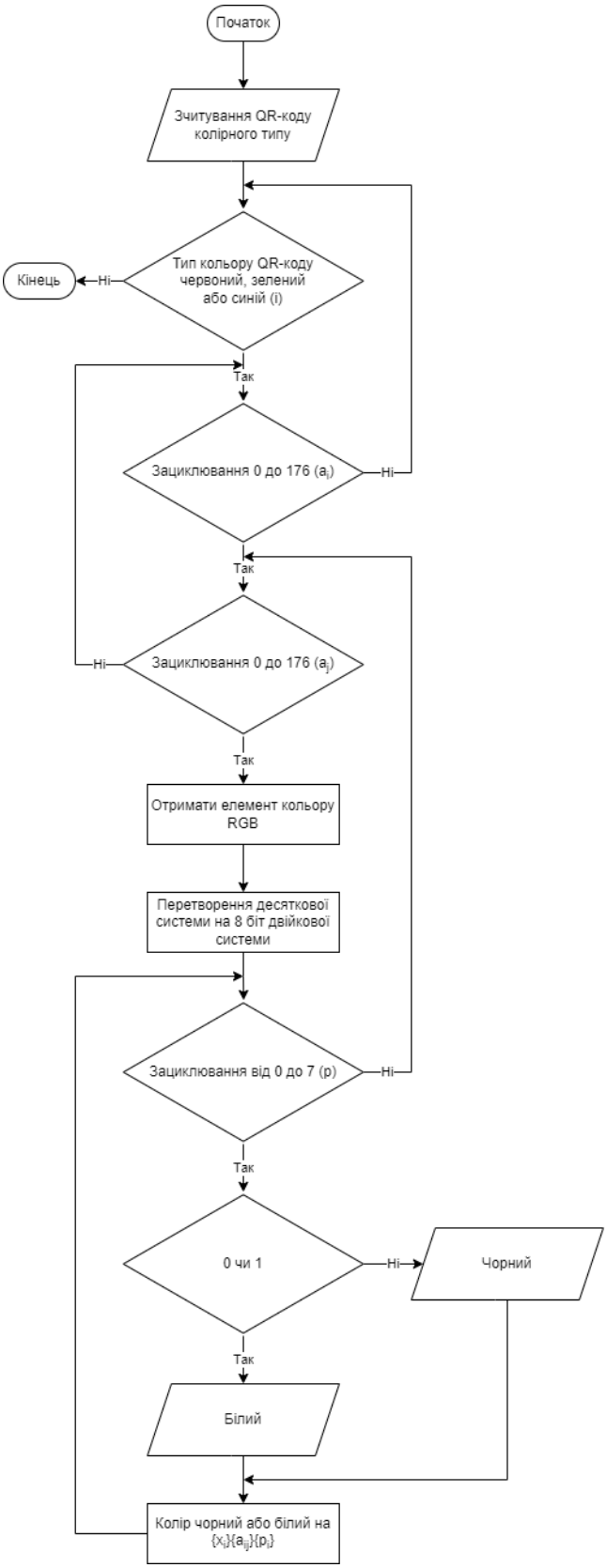


Рис. 3.5. Технологічна схема процесу визначення чорних або білих пікселів чорно-білих QR-кодів.

Був протестований експеримент минулого часового порядку рівня виправлення помилок, як показано в таблиці 3.7, і він зосередився лише на

процесі демультиплексування. З табличних даних рівень виправлення помилок Q має мінімальний загальний час декодування демультиплексування, що минув, який становить 96 мілісекунд. Тим часом, рівні виправлення помилок M і H мають максимальний час декодування демультиплексування, що минув за 101 мілісекунду. З результату, час, витрачений на декодування кольорового QR-коду до текстового файлу Base64, не був великою проблемою, оскільки час діапазону між ними не був занадто довгим. На завершення процесу пішло приблизно 0,1 секунди.

Таблиця 3.7

Час декодування процесу демультиплексування.

Рівень виправлення помилок	Демульти-плексування RGB	Чорно-біле демульти-плексування	Повне мульти-плексування
L	0с 045мс	0с 052мс	0с 097мс
M	0с 049мс	0с 052мс	0с 101мс
Q	0с 046мс	0с 050мс	0с 096мс
H	0с 048мс	0с 053мс	0с 101мс

Декодування тексту Кодер/Декодер для модуля декомпресії

Формат текстового файлу Base64, тобто файл текстового декодера ASCII, який був створений в результаті процесу демультиплексування, буде використовуватися як перший процес в методі декомпресії. У цьому процесі текст Base64 потрібно перетворити в файл стиснення, який є файлом GZip. Файл GZip - це двійковий файл, який може розпаковувати та створювати оригінальний текстовий файл. У цьому методі беруть участь два процеси, які є інструментом декодування тексту ASCII та декомпресії.

Метод почався з читання текстового файлу Base64 до кінця файлу. Функція читання повинна бути здатна читати друковані та керувати символами всередині текстового файлу Base64. Після завершення процесу читання символи були перетворені в двійковий файл за допомогою методу *Base64.decode.* Двійковий файл стиснення був повністю згенерований, коли процес декодування Base64 успішно конвертував його.

Інструмент стиснення декодування для модуля декомпресії

Останнім кроком є розпакування бінарного файлу стиснення в оригінальний текстовий файл. Бінарний файл стиснення в даному випадку відноситься до файлу GZip, який використовувався в якості інструменту стиснення. Відповідний клас декомпресії, який використовується в цьому дослідженні, - клас *GZip*. Зазвичай цей клас розроблявся постачальником інструментів стиснення. Крок декомпресії почався з підготовки файлу вихідного потоку. Потім дані розпакували всередині бінарного файлу стиснення і помістили їх у вихідний текстовий файл всередині відповідного внутрішнього каталогу. На рисунку 3.6 представлена технологічна схема методу декомпресії.

Таблиця 3.8 показує минувший час процесу декомпресії. Процес декомпресії починався від текстового декодера (Base64) до декомпресії (GZip). Рівень виправлення помилок H мав можливість завершити процес декомпресії в найшвидший час у порівнянні з іншими рівнями виправлення помилок, який становив 0 секунд і 8 мілісекунд.

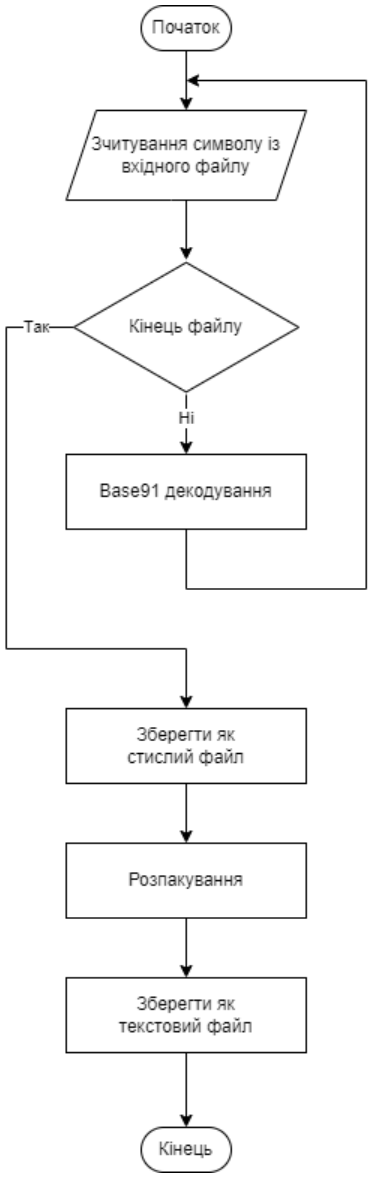


Рис. 3.6. Технологічна схема методу декомпресії.

З цієї точки зору можна зробити висновок, що час, витрачений на завершення цього процесу, не сприяв розміру файлу QR-коду для цього дослідження.

Таблиця 3.8

Минувший час процесу декомпресії.

Рівень виправлення помилок	Декомпресія
L	0с 012мс
M	0с 011мс
Q	0с 009мс
H	0с 008мс

3.3 Алгоритм часткового вилучення

У цьому розділі описується техніка декодування та повторного кодування кольорового QR-коду під час половини шляху процесу. Це називається частковим вилученням. Необхідність в ньому полягає в цілях маніпулювання даними. Маніпуляція включає оновлення, вставку та видалення. Всі модулі записуються в режимі псевдокоду і розташовуються у вигляді індексів модулів. Потім алгоритми розробляються і показуються в додатках D, E, F і G. Із запропонованого модуля часткового вилучення можна довести, що час обробки може бути скорочено. Модуль базується на поточному модулі декодування, але з деякою модифікацією процесів потоку. Висновки ґрунтуються на тому, наскільки швидко можна маніпулювати процесом та кількістю інформації.

Кольоровий QR-код був розроблений на основі абстрактної моделі, як показано на рисунку 3.7. QR-коди, які були виготовлені в цій моделі, є моноколорними (червоний, зелений, синій), чорним і білим, потім вони діляться на два рівні при виконанні часткового вилучення. Перший рівень включає маніпулювання чорно-білим QR-кодом, а другий рівень - монокольоровий QR-код. Кожен рівень містить процеси декодування та перекодування, і кожен з них буде задіяний на обох рівнях.

3.3.1 Модуль декодування 1-го рівня

Цей процес розпочався з визначення того, яким чорним індексом QR-коду буде маніпулювати. Крім того, повинні були бути описані значення глибини кольору і загальна кількість колірних каналів з колірної моделі. Коли було класифіковано індексне розташування чорно-білого QR-коду, глибину кольору та кольорові канали, наступним процесом було декодування кольорового QR-коду за допомогою демультишарового процесу. Наприклад, якщо індекс чорно-білого QR-коду дорівнює 24, глибина кольору становить 8 біт і використовує кольоровий канал RGB, наступний процес слід віднести до синього моноколірного QR-коду після демультишарового процесу. Формула для

визначення того, яке індексне розташування монокольорного QR-коду буде посилатися з індексу чорно-білого QR-коду, наведена нижче:

Індекс розташування моноколора = індекс розташування чорно-білого QR-коду / загальна глибина кольору

який, уникаючи залишку, приймає тільки цілочисельне значення. З прикладу, розглянутого раніше, індексне розташування монокольорного QR-коду дорівнює 23 / 8 = 2,875, в якому місце розташування індексу дорівнює 2.

Коли монокольоровий QR-код був створений, наступним процесом було визначення індексного розташування монокольорового QR-коду, який потрібно витягти. Розраховане індексне розташування монокольорового QR-коду було використано як орієнтир для вилучення восьми чорно-білих QR-кодів. Після того, як було ідентифіковано індексне розташування монокольорового QR-коду, наступний процес витягнув монокольоровий QR-код у вісім чорно-білих QR-кодів. Індексне розташування чорно-білого QR-коду, який був описаний або введений, повинен був використовуватися для визначення того, який чорно-білий QR-код потрібно витягти в символи. Процес порівняння використовувався для того, щоб знайти вибране індексне розташування чорно-білого QR-коду. Ця частина відома як процес демультиплексування.

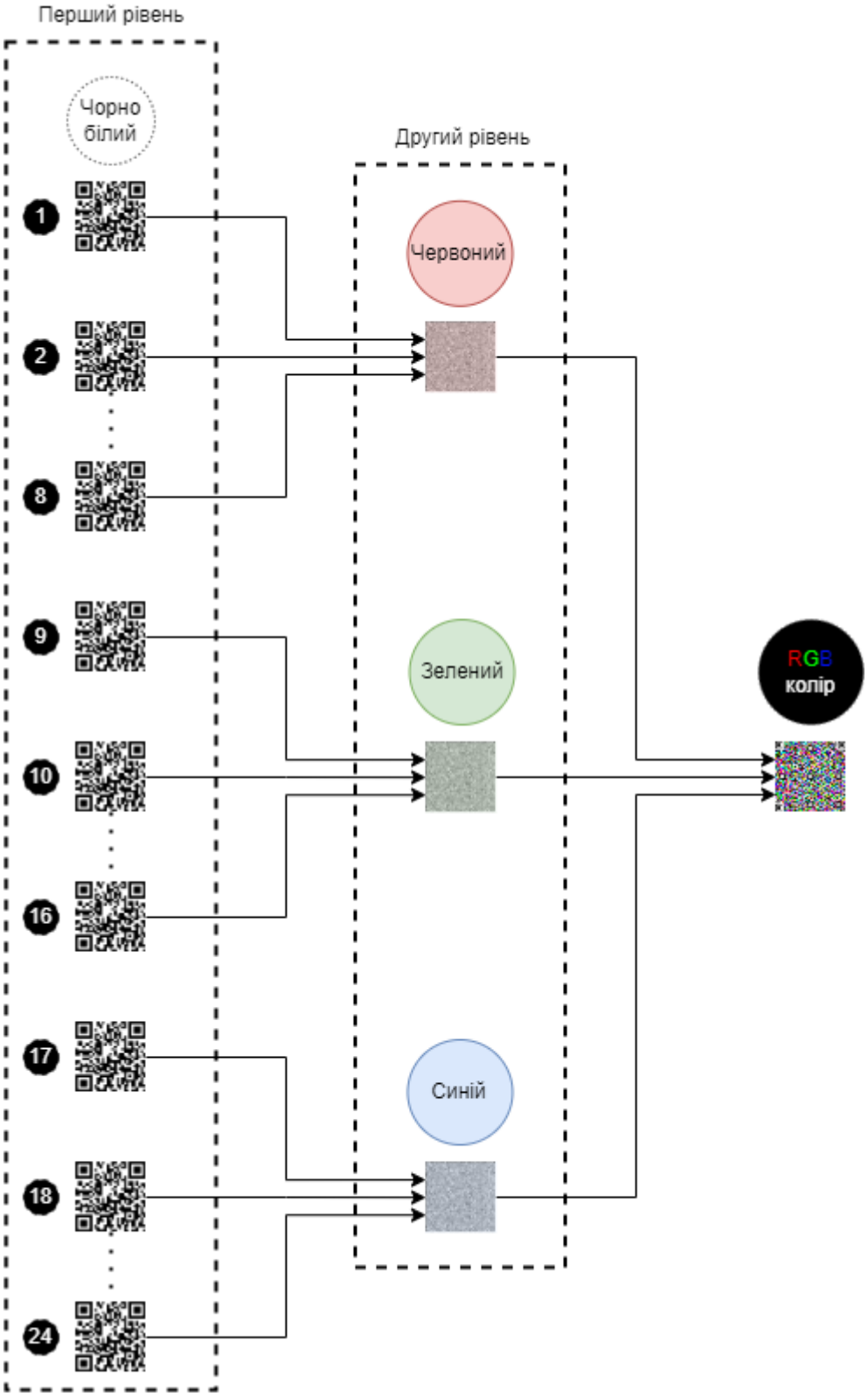


Рис. 3.7. Абстрактна модель 8-бітової глибини кольору та 3-канального RGB-кольору моделі.

Коли був обраний чорно-білий QR-код, наступний процес вилучення його в текст символів за допомогою декодера QR-коду. Символи з чорно-білого QR-коду, який був створений, були в нечитабельному текстовому форматі, який

потрібно розшифрувати. На етапі процесу декомпресії символи були декодовані в двійковий файл, який представляв собою формат типу стиснення. Оскільки процес декомпресії мав два процеси, наступним кроком було розпакування їх на читабельні символи, які були вихідними символами. У процесі декомпресії використовувалися інструменти декомпресії. Текстом, створеним за допомогою інструментів декомпресії, можна маніпулювати, наприклад, додавати, оновлювати або видаляти.

3.3.2 Модуль повторного кодування 1-го рівня

Коли всі тексти були змінені, наступним процесом було перекодування їх до кольорового QR-коду. Процес перекодування почався зі стиснення нових маніпульованих символів. Він включав лише ідентифіковане розташування індексу чорно-білих QR-кодів, а решта чорно-білих QR-кодів залишалися як зображення у файлі комп'ютерного каталогу. Решта кодів будуть використовуватися в процесі мультиплексування. Як завжди, нові маніпульовані символи стискалися інструментами стиснення (GZip), а потім з двійкового файлу стиснення, який був викликаний, вони були перетворені з двійкового в текст за допомогою декодера (Base64). Режим нового тексту був змінений з ANSI на UTF-8. Потім текст з декодера був закодований в чорно-білі QR-коди.

Решта та нові чорно-білі QR-коди були знову об'єднані в процесі мультиплексування та використані як вхід для розробки монокольорових QR-кодів. Після створення монокольорових QR-кодів наступним кроком було створення кольорового QR-коду з багатошаровим процесом. Методи перекодування мультиплексування і багатошаровості були тими ж методами, використовуваними в попередньому розділі.

3.3.3 Модуль декодування 2-го рівня

Розшифровка псевдокоду 2-го рівня передбачає лише часткове вилучення монокольорового QR-коду, який потрібно ідентифікувати в першу чергу. Процеси 1 і 2 рівнів мають подібний потік завдань; тим не менш, рівень 2 витягує

монокольорові QR-коди, тоді як рівень 1 витягує чорно-білі QR-коди. Перевага вилучення монокольорових QR-кодів полягає в тому, що вилучення даних у чорно-білих QR-кодах можна зробити для більш ніж одного чорно-білого QR-коду. Це залежить від загальної глибини кольору, яка використовувалася раніше.

Процес почався з ідентифікації того, яким індексним розташуванням монокольорового QR-коду доводиться маніпулювати. Наступним кроком була ініціалізація розташування індексу текстових файлів на основі розташування індексу монокольорового QR-коду. Використовуючи цю функцію, загальна кількість чорно-білого QR-коду базувалася на типі глибини кольору одного колірного каналу, що використовується. Наприклад, якщо глибина кольору становить 8 біт для кольорового каналу, то загальна кількість чорно-білого QR-коду становить 8 для моноколірного QR-коду. Щоб виділити індексне розташування чорно-білих QR-кодів, формула виглядає наступним чином:

Перше індексне розташування чорно-білого QR-коду = індексне розташування моноколірного QR-коду x тип глибини кольору одного кольорового каналу

Потім індексне розташування чорно-білих QR-кодів було збільшено на 1 і цей приріст повторювався + 1 раз на основі розміру глибини кольору одного кольорового каналу. Процес продовжився генеруванням (демультишаровим) групи моноколірних QR-кодів з кольорових QR-кодів. Загальна кількість монокольорових QR-кодів базувалася на використовуваній кольоровій моделі. Коли були створені монокольорові QR-коди, було зроблено заяву про умову, щоб визначити, який моноколор буде використовуватися для їх вилучення. У той момент, коли був ідентифікований монокольоровий QR-код, він потім був розшифрований (демультиплекс) у чорно-білі QR-коди на основі глибини кольору. Зображення чорно-білих QR-кодів було ініціалізовано та розміщено в конкретному каталозі розташування. Всі зображення чорно-білих QR-кодів були розшифровані в текстовий файл, який містив декодовані символи (двійкові в текст). Після завершення процесу декодування від чорно-білого QR-коду до текстового файлу декодовані текстові файли були перетворені в двійкові файли

стиснення. Для цього процесу використовувався інструмент декодера. Двійкові файли, які були створені в результаті попереднього процесу, були розпаковані. Це був останній процес, щоб отримати частину оригінальних текстів або символів. Частиною оригінальних текстів маніпулювали за допомогою процесів додавання, оновлення або видалення. В цілому, етапи декодування і перекодування на рівні 2 були тим же процесом, що і обговорювалися в попередньому розділі, але з різницею в реалізації.

3.3.4 Модуль повторного кодування 2-го рівня

Після того, як група символів була змінена, наступним процесом є повторне кодування їх у кольоровий QR-код. Процес перекодування включає в себе групу текстів з обраних монокольорових QR-кодів. Вигода від цього процесу полягає в тому, що великою кількістю символів можна маніпулювати в порівнянні з реалізацією рівня 1.

Першим кроком було управління символами в оновленому текстовому файлі. Розмір текстового файлу повинен бути аналогічним попередньому процесу декомпресії. Текстовий файл був названий на основі індексного розташування першого чорно-білого QR-коду, і він був поміщений в поточний конкретний каталог, який використовувався в програмі для отримання цих файлів. Наступним процесом було стиснення групи оновлених текстових файлів за допомогою інструменту стиснення. Оновлений текстовий файл, який потрібно стиснути, залежав від того, які символи були оновлені. Якщо текстовий файл не оновлювався, процеси стиснення і двійкового в текстове декодування не були обов'язковими або необхідними для виконання. Його можна використовувати зі старою версією чорно-білого QR-коду. Припустимо, що група текстових файлів оновлена, інструмент стиснення стиснує текстовий файл. Процес продовжився кодуванням типу стискання файлу в закодовані символи за допомогою інструмента кодера. В результаті була створена купа символів, які були готові до поділу та розміщення у відповідному чорно-білому QR-коді версії 40 максимальна місткість символів а також рівень виправлення помилок. Символи

були перетворені з формату ANSI в UTP8 через обмеження вбудовування символів в чорно-білі QR-коди. Закодовані символи в кожному файлі містили максимальну загальну кількість символів чорно-білого QR-коду версії 40 і були представлені у вигляді зображень чорно-білих QR-кодів. Загальна кількість файлів, які були розділені, повинна була бути схожа на загальну кількість чорно-білих зображень QR-кодів, які були розшифровані раніше. Якщо ні, система не зможе перейти до наступного кроку. Коли всі критерії були виконані, закодовані файли символів були закодовані в зображення чорно-білого QR-коду. Назва зображень чорно-білих QR-кодів була заснована на тому, що використовувався монокольоровий QR-код , і вони розташовувалися в одному каталозі розташування оригінального чорно-білого QR-коду. Наступним кроком став процес мультиплексування, який перетворив групу оновлених чорно-білих QR-кодів в єдиний оновлений монокольоровий QR-код. Тип моноколірної моделі був ідентифікований на основі процесу декодування, який був зроблений раніше. Старі та поточні монокольорові QR-коди були об'єднані для створення кольорового QR-коду за допомогою багатошарового процесу. Це був останній процес перекодування псевдокоду 2-го рівня.

ВИСНОВОК ДО РОЗДІЛУ 3

У цьому розділі подано детальну інформацію про потокові процеси кодування, декодування та часткового вилучення. Основною основою цієї тези є надання та доведення алгоритмів, які були розроблені та розроблені, можуть бути збільшені ємність даних кольорового QR-коду. Ці алгоритми сприяють запропонованому кольоровому QR-коду, який наведений на рисунку 1 Вступу. Оскільки дослідження в цій дисертації включає як псевдокод, так і блок-схему, яку потрібно представити, отже, це повна деталь, як були розроблені кодування, декодування та часткове вилучення. З третьої глави розробка цієї системи базується на розробленому алгоритмі. Він продовжить процес отримання знахідки в четвертому розділі після завершення розробки системи.

РОЗДІЛ 4

ТЕСТУВАННЯ

Цей розділ охоплює детальний результат експерименту модулів кодування, та декодування, які також був включений до методології тестування третьої фази. Кожен модуль містить процеси стиснення, мультиплексування та багатошаровості. Наступний розділ детально описують результати експерименту.

4.1 Експеримент з кодуванням

Експерименти охоплюють отримання загальних символів, загальної кількості втрачених символів та часу обробки під час перетворення з текстового файлу на кольоровий QR-код. Кожен модуль буде протестований, і буде проведено порівняння з чорно-білим QR-кодом. Результати поділяються на різні рівні виправлення помилок, а саме L, M, Q та H.

4.2 Результат експерименту з кодуванням модулів

Експеримент із запропонованими методиками генерації кольорового QR-коду виконується за трьома критеріями: рівнем виправлення помилок, щільністю даних та часом обчислень. Вхідні дані використовуються в цьому експерименті, а текст, заснований на короткій історії. На рисунку 4.1 представлена вхідного тексту, яка має різні типи символів, такі як цифрові, алфавіти та кілька символів.

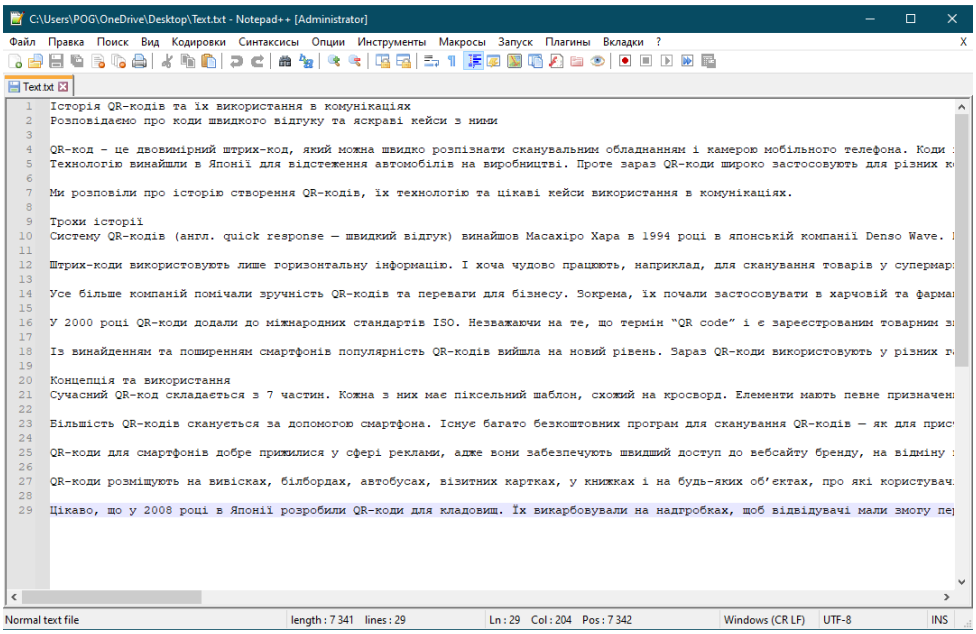


Рис. 4.1. Коротка історія QR-коду.

Використовуваним еталонним методом є QR-код версії 40. У цьому експерименті розподіл символу на кожному QR-коді було встановлено з обмеженою кількістю символів відповідно до рівня виправлення помилок. Отже, максимальна кількість символів, що зберігаються в кожному чорно-білому QR-коді версії 40, наведена в таблиці 4.1. Показано, що рівень виправлення помилок L має перевагу або найвищий за загальною кількістю символів серед інших рівнів виправлення помилок. Він може зберігати 2952 символи для чорно-білого QR-коду версії 40.

Таблиця 4.1

Максимальна кількість символів, що зберігаються в кожному QR-коді версії 40.

Рівень виправлення помилок	Загальна кількість символів кожного -коду
L	2,952
M	2,330
Q	1,662
H	1,270

У таблиці 4.2 наведено час, що минув під час процесу стиснення кодування. Процес кодування починається від стиснення GZip до кодера Base64. Рівень виправлення помилок H має можливість завершити процес стиснення кодування швидше, ніж інші рівні виправлення помилок, які витратили 21 мілісекунду на завершення процесів. Причина, чому рівень виправлення

помилок H витрачає менше часу на обробку, оскільки він містить менше символів у порівнянні з іншими, що менше часу на обробку. З цієї точки зору можна зробити висновок, що час, витрачений на завершення процесу, сприяє загальній кількості символів QR-коду для цього дослідження.

Таблиця 4.2

Минувший час процесу компресії кодування.

Рівень виправлення помилок	Компресія
L	0с 064мс
M	0с 029мс
Q	0с 024мс
H	0с 021мс

Процес кодування мультиплексування запускається, коли процес стиснення завершений. Цей процес завершиться, коли будуть створені червоні, зелені та сині QR-коди. Час, витрачений на цей процес, був перевірений для того, щоб інформація про ефективність на який рівень виправлення помилок витрачається найдовше протягом цього процесу. Таблиця 4.3 наочно показує, що корекція помилок рівня H має можливість завершити процес відразу в порівнянні з іншими. Мається на увазі час, що минув. Можна зробити висновок, що рівні виправлення помилок H, Q і M здатні виконати більш ніж на 50% менше, ніж рівень виправлення помилок L.

Таблиця 4.3

Минувший час процесу кодування мультиплексування.

Рівень виправлення помилок	Мультиплексування
L	0с 238мс
M	0с 176мс
Q	0с 154мс
H	0с 131мс

Нарешті, останнім процесом у цій моделі є пошук часу, використаного в багатошаровому процесі, за допомогою різних рівнів виправлення помилок. Цей процес починається після створення червоного, зеленого та синього QR-кодів. Ці

QR-коди будуть об'єднані для створення кольорового QR-коду. З проведеного експерименту багатошаровий процес вимірюється виходячи з часу в комп'ютерній системі. Таблиця 4.4 зведена в таблицю результату багатошарового процесу в секунді і мілісекунді. Часовий діапазон між рівнями виправлення помилок M, Q і H не так сильно відрізняється, але рівень корекції помилок L має значно різний час виконання. Можна зробити висновок, що рівень виправлення помилок L не підходить для використання, якщо користувачеві потрібно негайно завершити процес (багатошаровий).

Таблиця 4.4

Результат багатошарового процесу в секундах і мілісекундах.

Рівень виправлення помилок	Багатошаровий час, що минув
L	0с 325 мс
M	0с 149 мс
Q	0с 092 мс
H	0с 090 мс

4.2.1 Загальний результат експерименту з кодуванням

Загалом, процеси стиснення, мультиплексування та багатошарового кодування є найкращим способом збільшити ємність QR-коду. Ці процеси можуть вносити додаткові дані або інформацію, щоб потрапити в зображення, яке є QR-кодом. Щоб досягти додаткової ємності сховища, потрібно дотримуватися процедури, щоб досягти мети. На рисунку 5.2 показаний потоковий процес кодування процесів стиснення, мультиплексування і багатошаровості. Він починається від звичайного текстового файлу, поки не буде створено кольоровий QR-код. Ці покрокові процеси є висновком з розділів, розглянутих раніше.

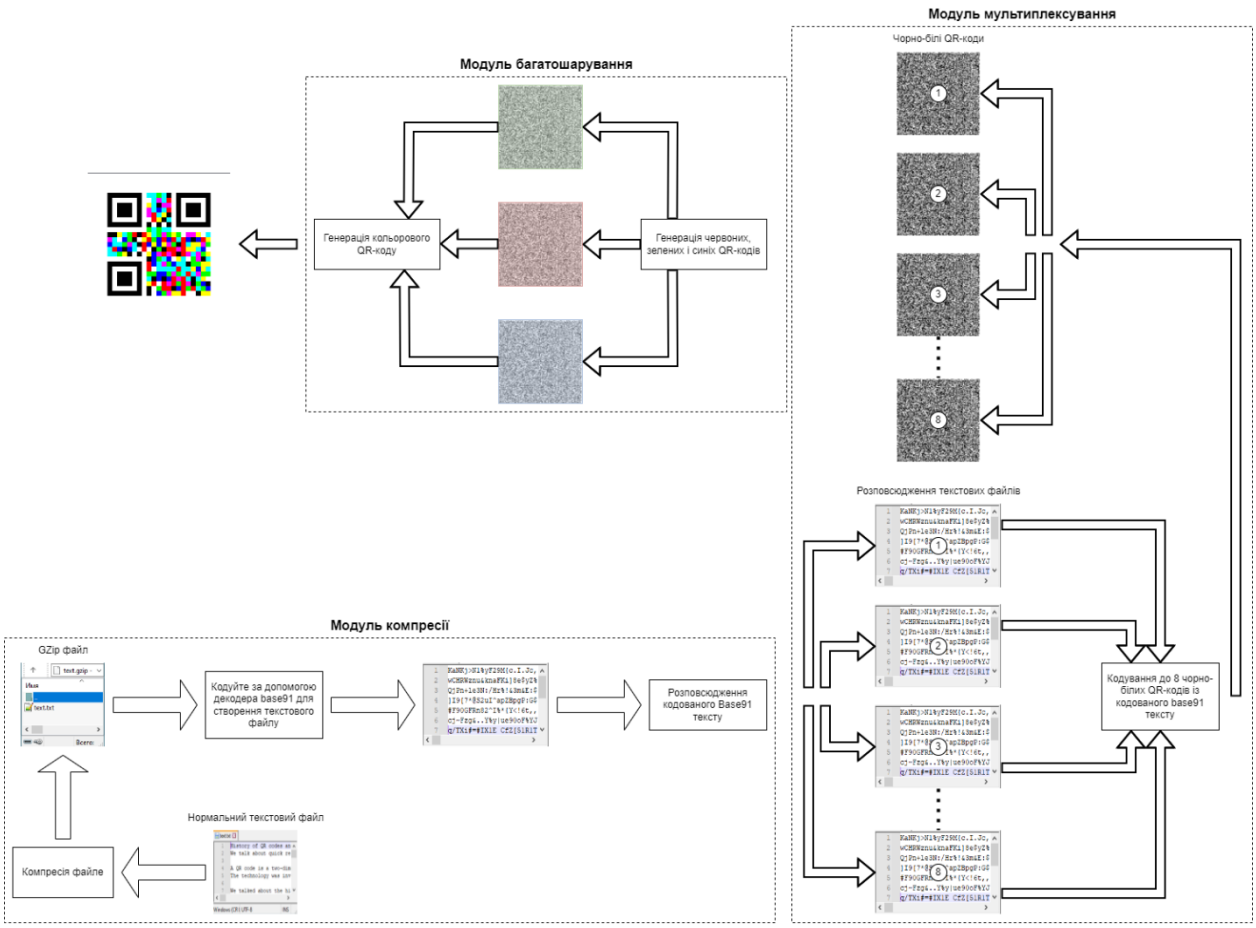


Рис. 4.2. Потокові процеси кодування модулів компресії, мультиплексування та багатошаровості.

Традиційний QR-код версії 40 був для порівняння з новим запропонованим кольоровим QR-кодом у вигляді ємності даних. Наведена таблиця 4.5 показує різницю текстової ємності між QR-кодом версії 40 та запропонованим кольоровим QR-кодом. З тестів рівень виправлення помилок L має велику різницю між двома QR-кодами. Він більший у 3 разів (для кольорового QR-коду) порівняно з традиційним QR-кодом версії 40. Звичайно, новий запропонований QR-код має велику можливість розширити символи максимум 8,859символів.

Таблиця 4.5

Різниця текстової ємності між QR-кодом версії 40 та запропонованим кольоровим QR-кодом.

Рівень виправлення помилок	Версія 40 (максимальна)	Пропонований кольоровий QR-код (Максимум)
L	2,953	8,859
M	2,331	6,993

Q	1,663	4,989
H	1,273	3,819

Запропонований закодований QR-код, який використовував методи стиснення, мультиплексування та багатошаровості, показав збільшення зберігання даних QR-коду. Експеримент був реалізований на тексті на основі короткого оповідання, яке містило не більше 8,860 знаків для виправлення помилок рівня L. Оскільки проведений експеримент дав хороші результати, він підходить для зберігання або вбудовування опису продукту в рекламних цілях також було б успішною реалізацією

4.3 Експеримент з декодуванням

Експеримент з декодуванням можна провести після завершення процесу кодування. Результати експерименту свідчать про витратну глибину кольору та колірний канал. Кожен модуль також тестується, наприклад, стиснення та декомпресія, поділ чорного та білого QR-коду та порівняння часу обробки та інструменту компресії з іншим дослідником.

4.3.1 Результат експерименту з декодуванням модулів

Процес декомпресії є останнім кроком для відновлення оригінального тексту. Від початку і до кінця процесів декодування обчислювався час, що минув, щоб отримати фактичний час для завершення процесу. У таблиці 4.6 наведено складання часу загальних процесів декодування.

З таблиці 4.6 загальний час обробки становить майже 0,3 секунди для завершення процесу декодування. Швидше всіх виправити помилки рівня Q вдалося завершити раніше.

Таблиця 4.6

Складання минувшого часу загальних процесів декодування.

Рівень виправлення помилок	Демультишарування	Демультиплексування	Декомпресія	Загально
L	0с 113 мс	0с 184 мс	0с 012 мс	0с 309 мс
M	0с 099 мс	0с 183 мс	0с 011 мс	0с 293 мс

Q	0с 101 мс	0с 179 мс	0с 09 мс	0с 289 мс
H	0с 100 мс	0с 183 мс	0с 08 мс	0с 291 мс

, що є найшвидшим серед інших трьох рівнів виправлення помилок. Загальний підсумок стовпця містить всю інформацію про процес, починаючи від першої команди і закінчуючи кінцевим рядком програми. Стовпці модулів демультишару, демультиплексування та декомпресії містять час, витрачений на виконання конкретного завдання модуля.

Цей метод може розширити зберігання даних QR-коду, якщо незначна модифікація проводиться в модулях стиснення, мультиплексування та багатошаровості. Модифікація передбачає вибір найкращої технології стиснення, розробку детальної глибини кольору та розширення колірного каналу.

4.4 Порівняння з існуючим QR-кодом

Всі дані для перевірки взяті з попередніх досліджень кольорового QR-коду та деяких комерційних продуктів. Аналіз порівняння запропонованого кольорового QR-коду з існуючим кольоровим QR-кодом базується на ємності даних для кожного з них. Результати порівнюються з терміном загальної кількості текстових символів, що зберігаються збережених байтах. Запропонований кольоровий QR-код (колір 8 біта) використовуватиме рівень виправлення помилок L, оскільки він може зберігати більше даних порівняно з іншими рівнями виправлення помилок. У таблиці 4.7 наведено порівняння місткості тексту між запропонованим кольоровим QR-кодом та існуючим QR-кодом (чорно-білим та кольоровим).

Таблиця 4.7

Ємність тексту порівняння між запропонованим кольоровим QR-кодом та існуючим QR-кодом (чорно-білим та кольоровим)

№	Тип QR-коду/ Дослідники	Тип	Загальна кількість символів
1	Пропонований QR-код (колір 8 біт)	Кольоровий	8,859
2	QR-код версії 40	Чорно-білий	2,953

3	Nancy Victor (2012)	Чорно-білий	1,365
4	Antonio Grillo, Alessandro Lentini, Marco Querinni, and Giuseppe F. Italiano (2012)	Кольоровий	643
5	Microsoft High Capacity Color Barcode	Кольоровий	84
6	Max E. Vizearra, Alexandre Zaghetto, Bruno Macchiavello and Anderson C. A. Nascimento (2012)	Кольоровий	43

З даних таблиці 4.7, видно, що запропонований кольоровий QR-код зберігає найбільшу загальну кількість символів у порівнянні з іншими. Він може зберігати 8,859 символів. Другим за величиною є стандартний QR-код 40, який може зберігати 2,953 символи. Перш ніж отримати висновки, було з'ясовано, що більшість кольорових QR-кодів пропонують більшу ємність пам'яті порівняно з чорними та білими QR-кодами. З цих результатів можна зробити висновок, що метод стиснення, мультиплексування та багатошаровості, який використовується в цьому дослідженні, є найкращим способом досягнення найбільшої ємності текстових даних у кольоровому QR-коді.

ВИСНОВОК ДО РОЗДІЛУ 4

Процес кодування включав в себе три основні процеси, а саме стиснення, мультиплексування і багатошаровість. Кінець цього процесу полягає у створенні кольорового QR-коду, який може зберігати більше даних всередині нього порівняно зі звичайним QR-кодом. У процесі стиснення використовуються два типи інструментів, а саме: інструмент стиснення та текстовий кодер. Поточний інструмент стиснення можна змінити на інший інструмент стиснення, якщо він здатний стискати більше даних і має додаткові можливості, такі як більший відсоток стиснення тексту або швидка транзакція в порівнянні з попереднім. Аналогічно, текстовий кодувальник також можна змінити на інший текстовий кодер, який здатний кодувати більше символів, але з меншим представленням символів. З наведеної вище інформації показано, що модуль стиснення є гнучким модулем, оскільки інструменти всередині нього можна змінювати відповідно до можливостей інструментів. Процес мультиплексування використовується для об'єднання багатьох QR-кодів в один QR-код. Цей метод допомагає мінімізувати *n* загальної кількості QR-кодів в єдине представлення QR-коду. Представлення QR-коду використовує моноколор, який можна розширити до інших типів моноколорів, таких як червоний, зелений та синій моноколор QR-коду. Загальна кількість QR-кодів n також може бути представлена кожним монокольоровим представленням. Це означає, що можна створити більше QR-кодів. Останнім процесом є багатошаровий процес, який складає кожен монокольоровий QR-код і змішує кольори між ними. Цей метод може заощадити більше моноколірного розподілу QR-коду, щоб QR-код можна було згенерувати з моноколору в кольорову презентацію.

Процес декодування є оберненим до процесу кодування і складається з процесів демультиплексування, демультишару та декомпресії. Мета полягає в тому, щоб повернути закодовані дані у введення даних. З результату показано, що дані можуть бути відновлені назад, як вони були введені раніше, без будь-яких відсутніх символів. Максимально збережені дані становлять до 8,859 символів, як показано в таблиці 4.6 з рівнем виправлення помилок L і 8 бітами

кожного кольорового каналу. В цілому, символи можуть бути збільшені, якщо в цій моделі реалізована глибина кольору і колірний канал. Як імітовано, один кольоровий QR-код може містити символи, подібні до книги або дослідницької дисертації. Деякі рівняння були опубліковані як еталон для отримання відповідного загального чорно-білого QR-коду.

Експеримент проводився в двох частинах. Перша частина включала експеримент для вимірювання витрат часу на вилучення одного чорно-білого QR-коду з точки зору того, чи може він зайняти менше часу, ніж звичайний метод. Методика називається декодуванням і перекодуванням рівня 1. Друга частина полягала в оцінці витрат часу на вилучення монокольорового QR-коду. Зміст даних були більшими, ніж попередні частини, оскільки монокольоровий QR-код міг містити багато чорно-білих QR-кодів. Він відомий як декодування та повторне кодування рівня 2.

В результаті показано, що змінення інформації в QR-коді підходить для виконання цим методом у порівнянні з існуючим методом (тобто QR-кодом версії 40), оскільки він витрачав менше обчислювального часу. З таким досягненням QR-код може бути розширений в управлінні оновленими даними всередині коду. Інформація, що зберігається в коді, завжди може бути змінена відповідно до потреб користувача.

Загалом, порівняння було зроблено в розділі тестів завдяки тому, що запропонований кольоровий QR-код має можливість зберігати більше символів серед інших QR-коді. Запропонований QR-код використовує виправлення помилок L, яке не враховує критерії режиму відновлення. Дослідження та комерційні QR-коди були використані для порівняння із запропонованими кольоровими QR-кодами. Тести показали, що найвище сховище даних належить запропонованому кольоровому QR-коду порівняно з іншими QR-кодами. Таким чином, загалом можна зробити висновок, що запропонований кольоровий QR-код забезпечує найкраще зберігання даних для кольорового QR-коду.

РОЗДІЛ 5

МОДЕЛЮВАННЯ СТАРТАП ПРОЄКТУ

5.1 Інформаційна картка проєкту

№	Характеристика	Зміст характеристики
1	Назва проєкту	Сканер кольорового QR-коду
2	Автори проєкту	Русанова Ольга Веніамінівна, Джевага Павло Дмитрович
3	Коротка анотація	Метою даного проєкту є створення універсальної програми для сканування кольорового QR-коду. Код швидкого відгуку (QR) - це двомірний штрих-код, який зберігає символи і може зчитуватися камерою будь-якого смартфона. Алгоритм, що включає процеси кодування та декодування, заснований на техніці червоного, зеленого та синього кольорів (RGB), яка використовується для створення кольорового QR-коду високої ємності. Отже, кольоровий QR-код має потенціал стати корисним міні-сховищем даних, оскільки він не залежить від підключення до Інтернету.
4	Термін реалізації	Три квартали (9 місяців)
5	Необхідні фінансові ресурси	Заробітні плати працівникам (щомісячні) на етапі розробки програмного забезпечення: - Розробники = 2.500$; - Тестувальник = 1.800$; - Проджект-менеджер = 2.000$; - Бізнес-аналітик = 2.500$; - Юрист = 1.200$; Всього щомісяця = 10.000$; Оформлення торгової марки = 22.000 грн. Рекламна просування = 100.000 грн.
6	Необхідні матеріальні ресурси	Відсутні через віддалену форму роботи. Також відсутня техніка, котру необхідно закупляти.
7	Необхідні людино-ресурси	- 2 Розробники (Back-End, Front-End); - 1 Тестувальник (Manual); - 1 Проджект-менеджер; - 1 Бізнес-аналітик; - 1 Юрист;
8	Опис проблеми, яку вирішує проєкт	Ємність даних QR-коду може бути покращена шляхом об'єднання декількох досліджень у ємності даних QR-коду. На основі цієї ідеї буде пояснено багато інформації, починаючи з фонового дослідження до існуючих досліджень QR-коду. Деякі огляди в певних дослідженнях були зроблені, щоб довести проблеми, які можуть виникнути в разі реалізації цієї структури. Тим не менш, з цього дослідження це може сприяти рішенню для зміни ємності даних QR-коду.

9	Головні цілі та завдання	Ціль даного дослідження є підвищення ємності даних кольорового QR-коду шляхом проектування та розробки алгоритмів компресії, мультиплексування та багатошаровості. Завдання дослідження досягається наступними завданнями: • Розробити алгоритм кодування QR-коду кольору RGB з більшим сховищем даних. • Розробити алгоритм декодування QR-коду кольорів RGB, який відновлює всі збережені дані. • Розробити алгоритм часткового декодування та перекодування кольорового QR-коду RGB, який дозволяє частково витягти необхідні закодовані дані.
10	Очікувані результати	За низькою ціною користувач отримує безліч можливостей, таких як: - Модуль кодування вхідних даних починаючи текстами та базами даних, закінчуючи різними медіа форматів файлів; - Модуль компресії, для зменшення розмірів файлів без втрати початкових даних ; - Модуль мультиплексування, котрий розбиває стиснуті файли між чорно-білими QR-кодами; - Модуль багатошаровості, який створює кольорові QR-коди з певними частками інформації; - Модуль генерація RGB QR-коду; - Модуль дебагатошаровості, який розбиває генерований RGB QR-код на кольорові QR-коди; - Модуль демультиплексування, котрий повертає чорно-білі QR-коди з кольорових QR-кодів - Модуль декомпресії, що відновлює закодовану інформацію з чорно-білих QR-кодів у початковий файл.

5.2 Склад команди стартап-проєкту

№	Роль	Спеціальність	Кількість
1	Розробник	Back-End	1 люд.
2	Розробник	Front-End	1 люд.
3	Тестувальник	Manual	1 люд.
4	Менеджер	Project Manager	1 люд.
5	Аналітик	Business Analyst	1 люд.
6	Юрист	Юрист	1 люд.
Разом			6 людей

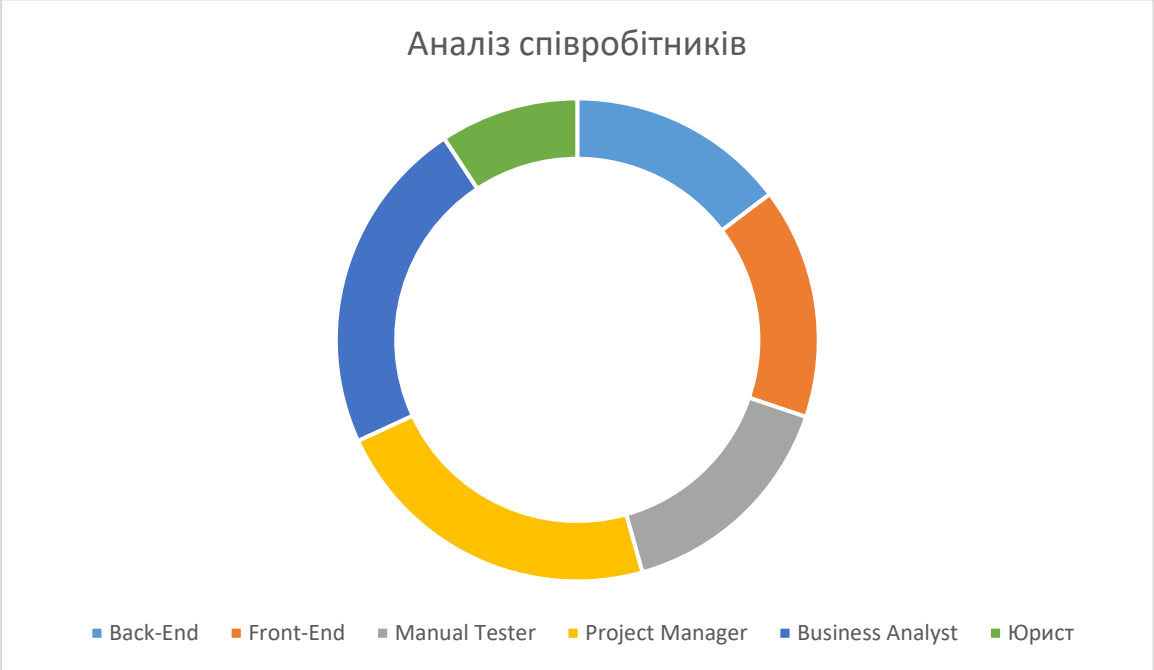
Отже, на початку розробки команда буде складатися з 6 людей. Далі потрібно визначити важливість факторів щодо вкладень у реалізацію проєкту. Результат запишемо до таблиці:

№	Фактор	Вага важливості (1-10 балів)
1	Створення та продвигання ідеї	7
2	Формування бізнес-плану	6

3	Компетентність співробітників	9
4	Виконання обов’язків	10
5	Співпраця та комунікація	8

Фінальним кроком у аналізі співробітників є оцінювання особистого внеску співробітника у справу. Для цього внесемо всі дані у таблицю та побудуємо графік, виходячи з результатів аналізу:

Фактор	Back-End	Front-End	Manual Tester	Project Manager	Business Analyst	Юрист	Разом
Створення та продвигання ідеї	43	34	11	91	95	0	
Формування бізнес-плану	12	20	35	100	100	0	
Компетентність співробітників	100	100	100	100	100	100	
Виконання обов’язків	100	100	100	100	100	100	
Співпраця та комунікація	63	81	90	95	93	1	
Разом	318	335	336	486	488	201	2164
Відсоток	14.7%	15.5%	15.5%	22.5%	22.6%	9.3%	100%



Отже, виходячи з результатів, можна стверджувати, що кожен член команди рівно важливий, як і всі інші. І лише спільна плідна праця може привести до результату.

5.3 Формулювання основної ідеї проєкту

Незважаючи на те, що цифровізація бізнесу — зовсім молодий процес, який ще досі не набрав повних обертів, але вже існує достатньо багато проєктів-аналогів, які були проаналізовані в пункті 1. Але після детального аналізу можна виділити явні плюси та мінуси кожного з конкурентів, які наявні на ринку станом на сьогоднішній день. Тому, за умов подальшого спостерігання за списками оновлень конкурентних продуктів — маємо всі шанси створити конкурентоспроможний продукт, який увійде до лідерів арени подібних програмних додатків.

У пункті 2 було виведено утопічну формулу, яка дозволить реалізувати програмне забезпечення з цінами, нижчими, ніж у аналогів, а функціоналом — значно більшим. І все це завдяки тому, що код проєкту уніфікований, спроєктований не під конкретний бізнес, а направлений на максимально велику кількість різноманітних бізнесів. За рахунок цього отримуємо можливість вкласти більше ресурсів на етапі проєктування і написання програмного

продукту, щоб потім на кожному наступному клієнті отримувати набагато більший прибуток, ніж написання нового продукту.

Але це не остання ідея максимального прибутку, який можна отримати з даної розробки. Значно більшою мірою ринок можна охопити, перевикориставши деякі з мікросервісів, які були написані для іншого напрямку бізнесу. Наприклад, монолітний програмний продукт для перукарні ніяк не може бути використаний для розробки проєкту для невеличких кав'ярень. Але якщо на моменті проєктування обрати правильний підхід — написання ПЗ з використанням архітектури незалежних мікросервісів, то це дозволить просто взяти програмний код деяких мікросервісів і перевикористати їх, не вносячи жодних змін, або ж з мінімальними змінами.

Тобто, якщо відштовхуватись з прикладу «Перукарня-кав'ярня» - можливо перевикористати як мінімум модуль авторизації, обліку персоналу та модулю контролю за складом.

Отже, підсумовуючи усі плюси та мінуси, можна навести висновки у таблиці:

№	Плюси	Мінуси
1	Перевикористання коду	Складність проєктування
2	Економія на розробці (з другої ітерації)	Складність розробки (перша ітерація)
3	Економія на тестуванні (з другої ітерації)	Мінімальна кастомізація
4	Економія на підтримці	
5	Охоплення великої кількості напрямів бізнесу	
6	Низька ціна для клієнтів	
7	Зменшення навантаження на співробітників за рахунок одноманітного підходу до всіх клієнтів	
Сума	7	3

Для успішної реалізації ідеї необхідно визначити всі сильні, нейтральні та слабкі техніко-економічні характеристики і пропрацювати слабкі, якщо вони мають вагомий вплив на фінальний результат.

Наведемо результати аналізу у таблиці:

№	Характеристика	Власна розробка	Аналоги			W	N	S
			1	2	3			
1		✔	-	-	-	-	-	✔

2		✔	-	-	-		-	✔
3	Можливість проведення відео-аудіо дзвінків	-	-	✔	-	✔	-	-
4		✔	-	-	-	-	-	✔
	Простий і неперезавантажений інтерфейс	✔	✔		-	-	✔	-
6		✔	-	-	-	-	-	✔
7		✔	-		✔	-	-	✔

Проаналізувавши результати техніко-економічної характеристики, можемо створити підґрунтя для формування конкурентоспроможності власного проєкту. Можна стверждувати, що наш проєкт має досить багато сильних характеристик, пов'язаних з кастомізацією під кожного клієнта, а це в бізнесі має досить вагоме значення, адже брендинг є дуже важливою складовою престижності кожної компанії. А користування власною системою з власним логотипом та підлаштованим інтерфейсом - невід'ємна частина політики брендингу компаній.

Також бачимо, що у нашій розробці відсутня можливість проведення відео та аудіо конференцій, але це зумовлено тим, що існує досить багато спеціалізованих програмних продуктів, які вже створили максимально зручні умови для проведення аудіо-відеоконференцій, тож немає жодних підстав збільшувати вартість проєкту та перевантажувати інтерфейс заради такого функціоналу.

5.4 Технологічний аналіз проєкту

Для мінімальних затрат на підтримку проєкту важливо обирати найновітніші технології, які будуть мати довготривалу підтримку і не будуть мати кардинальних змін у оновленнях, що спричинить неминучі зміни в коді нашого продукту. Тому проаналізуємо та оберемо стек технологій, який буде використаний у створенні нашого програмного продукту та занесемо дані до таблиці:

№	Технологія	Версія	Сфера використання	Доступність

1	Visual Studio Code	3.1	Головне середовище для розробки	Безкоштовно
2	Python	3	Мова фреймворку	Безкоштовно
3	Python libraire	3	Фреймворк бібліотек	Безкоштовно
4	pythare	15	Бібліотека для управління оптимізацією проєктом	Безкоштовно
5	qrcode	3.1	Бібліотека роботи з QR-кодами	Безкоштовно
6	PIL	3.4	Бібліотека для роботи з фотографіями	Безкоштовно

Отже, основний стек обраний повністю від компанії Microsoft, що пояснюється ідеально налаштованою комунікацією та найширшим функціоналом для комунікації між різними сервісами.

5.5 Аналіз ринкових можливостей для запуску стартап-проєкту

Для коректного аналізу можливостей ринку перш за все необхідно провести характеристику потенційного ринку для нашого майбутнього стартап-проєкту. Для цього відобразимо результати у таблиці:

№	Характеристика	Значення
1	Кількість аналогів на ринку	3 шт
2	Обсяг запланованої реклами	1.000 рекламних показів * 100 грн (за 1 показ) = 100.000 грн щомісячно
3	Якісна оцінка динаміки ринку	Ринок зростає, попит стає все більшим з кожним днем, так як кількість потенційних клієнтів збільшується з динамікою відкриттів нових бізнесів
4	Обмеження для входу на ринок	Обмеження відсутні
5	Вимоги та стандарти	- Найголовнішою вимогою є швидкість відображення та постійна актуальність даних – генерація QR-коду не може займати більше, ніж 3 секунди - Максимальна простота інтерфейсу для швидкого розуміння новими користувачами - Мінімальний людський фактор з метою зменшення кількості помилок - Коректність статистичних даних - Використання загальноприйнятих форматів відображення даних згідно з країною використання (формат дат, формат чисел тощо) - Єдиний стиль веб-сайту, включаючи стиль шрифтів
6	Середня рентабельність в галузі	(735000 / 1000000) * 100 = 73,5 % річних

Аналізуючи потенційних клієнтів, можна стверджувати, що цільовою аудиторією продажів будуть бізнеси у сфері обслуговування. Так як додаток уніфікований - він не має жодних відмінностей у певній цільовій групі клієнтів та всі вимоги до додатку будуть співпадати з основними вимогами ринку, описані у таблиці вище.

Не можна нехтувати й існуючими загрозами, які можуть завадити розвитку стартап-проєкта. Наведемо всі фактори ризику у таблиці:

№	Фактор	Опис загрози	Відповідна реакція компанії
1	Зріст конкуренції на ринку	Поява на ринку нових компаній, функціонал яких подібний, або ширший за наявний	Внесення до порядку денного оновлення продукту, доповнивши функціонал подібними можливостями, як у конкурентів та пропозиція оновити раніше продані продукти вже наявним клієнтам безкоштовно
2	Неякісна рекламна компанія	Неправильно обрана цільова аудиторія реклами, недостатня зацікавленість та мале розповсюдження	Змінити фірму, котра надає послуги SMM та обрати нову стратегію розповсюдження рекламної продукції
3	Неякісна продукція	Знайдені баги в додатках, які вже знаходяться в стадії використання	Внесення до порядку денного виправлення знайдених проблем, негайний деплоймент виправлень та, за потреби, цікава пропозиція для обурених клієнтів

Проаналізувавши наявні загрози, потрібно також проаналізувати й можливі перспективи, які можуть чекати програмний додаток у майбутньому. Для цього запишемо фактори можливостей у таблицю:

№	Фактор	Опис можливості	Відповідна реакція компанії
1	Відсутність конкурентів	На ринку не з’являються нові проєкти, здатні скласти конкуренцію	Продовження розвитку в обраному напрямку без відволікання на “гонку аналогів”
2	Створення дерева розвитку програмного додатку	Створення та доповнення концепції перспектив розвитку програмного додатку	Залучення коштів для розвитку власного програмного продукту
3	Фідбек-сесії з наявними клієнтами	Підтримка постійного двостороннього зв’язку “розробник - клієнт” з метою вдосконалення продукту	Реагування та обробка запитів про вдосконалення від клієнтів, які використовують програмний додаток

Для глибокого аналізу ринку необхідно розглянути усі конкурентні середовища. Результати занесемо до таблиці:

Характеристика конкурентного середовища	Сфера впливу характеристики	Дії компанії для забезпечення конкурентоспроможності
Цінова	Конкуренція за рахунок зменшення ціни попри численні оновлення та вдосконалення програми	Зменшення ціни на програмний додаток
Товарно-родова	Конкуренція продуктів різного типу	Вдосконалення наявного функціоналу з метою покриття більшої кількості сфер бізнесу
Чиста	Відсутність будь-яких конкурентнів на ринку	Захищеність бізнесу від конкуренції
Немарочна	Мінімальна роль торгової марки	Виграш у кількості клієнтів за рахунок якості продукту, а не назви бренду
Національна	Поява конкурентів за межами країни	Створення мультимовного інтерфейсу

Проведемо аналіз сильних та слабких сторін згідно факторів конкурентоспроможності та занесемо результати до порівняльної таблиці:

№	Фактор конкурентоспроможності	Бали (1-20)	Рейтинг аналогів у порівнянні з власним проєктом						
			-3	-2	-1	0	1	2	3
1	Відсутність аналогів	15						+	
2	Ціна	20							+
3	Задовільнення потреб користувачів	20			+				

У якості альтернативного ринкового впровадження найкращим варіантом є транслювання реклами на просторі соціальних мереж, таких як “Facebook”, “Instagram”, “Telegram”, “Twitter” та “Linkedin” з метою збільшення кількості потенційних клієнтів. Цей варіант найдієвіший через те, що існує можливість максимально коректно налаштувати фільтри потенційної аудиторії, яка являє собою спільноту власників середніх та малих бізнесів у сфері обслуговування. Також у даних соціальних мережах буде створено профіль компанії та продемонстровано весь спектр послуг, які покриває стартап.

5.6 Моделювання виробничого плану

Однією з найголовніших частин моделювання є створення календаря плана-графіка реалізації проєкту

№	Назва етапу реалцізації	Період реалізації						
		1-й рік				2-й рік	3-й рік	N+ рік
		1-й кв.	2-й кв.	3-й кв.	4-й кв.			
1	Науково-дослідницькі роботи	+						
2	Розробка технічного завдання	+						
3	Робоче проєктування	+	+	+				
4	Тестування проміжних та кінцевих результатів		+	+				
5	Створення компанії, оформлення юридичних документів			+				
6	Маркетингові дослідження			+				
7	Аналіз поточної ринкової ситуації			+				
8	Рекламна компанія			+	+	+		
9	Впровадження системи				+			
10	Продаж продукції				+	+	+	+

Великим плюсом є ідея з уникненням необхідності закупівлі будь-якого обладнання та робочої площі, а також оренди офісних приміщень, так як усі співробітники компанії працюють віддалено, як мінімум на перших етапах розробки. Також вони використовують власну техніку для роботи, що значно зменшує витрати на старт проєкту. Це дозволяє не завишати ціни на готову продукцію, чим спричинить більшу конкурентоспроможність.

Проаналізуємо витрати на найманих працівників, котрі будуть допомагати з розробкою на початкових етапах стартап-проєкту. Занесемо дані в таблицю:

№	Категорія працівників	Чисельність	Заробітна плата за місяць, тис. $	Податки (ФОП, 5% від ЗП)	Витрати на оплату праці, тис. $		
					1 рік	2 рік	N+ рік
1	Розробник	2	2.5	125	60	30	30
2	Тестувальник	1	1.8	90	21	21	21
3	Проджект-менеджер	1	2	100	24	24	24
4	Бізнес-аналітик	1	2.5	125	30	-	-
5	Юрист	1	1.2	60	14.4	-	-
Разом	4	5	10	535	149.4	75	75

ВИСНОВОК ДО РОЗДІЛУ 5

У даному розділі магістрської дисертації було проаналізовано та змодельовано стартап-проєкт на основі запропонованої у роботі ідеї. Підібрано мінімальний склад команди з шести осіб - back-end девелопера, front-end девеломера, manual тестувальника, проджект менеджера, бізнес аналітика та юриста. Також розроблено план затрат на заробітні плати, включно з податками на момент розробки стартап-проєкту.

У ході роботи також було проаналізовано наявну ситуацію на ринку, ідею порівняно з аналогами, винайшовши плюси та мінуси ідеї, а також сильні та слабкі сторони ідеї.

Було проведено аналіз ринкових можливостей та продумано шлях альтернативного ринкового впровадження. Також створено мапу виробничого плану з конкретними датами та дедлайнами на кожен етап запуску. Усі результати було занесено до таблиць та зображено у вигляді діаграм різного типу.

І в результаті отримали готову заповнену інформаційну картку стартап-проєкту, котра дає зрозуміти короткі відомості про кожен аспект майбутнього стартапу.

ВИСНОВОК

Розглянуто модифікований алгоритм, який розкриває та вирішує проблеми в розширенні сховища QR-коду. З минулих досліджень було реалізовано багато методів для збільшення зберігання QR-коду, таких як компресія, мультиплексування тощо. На основі ідей з минулого (які були зосереджені на розширенні даних), було виявлено кілька способів розширити більше текстових символів, які будуть вбудовані в QR-код.

Запропонована модель може допомогти користувачам швидше збільшити обсяг пам'яті, використовуючи QR-код як носій інформації в одному зображенні QR-код. До цього був проведений огляд літератури для вивчення проблеми дослідження і прийняття рішення проблеми. Ця модель вимагає набору тестових даних або введення для виконання динамічного експерименту зі зберігання даних в QR-коді. Таким чином, дані допомагають довести розроблену модель, чи можна її використовувати для збільшення даних в QR-коді. Використовувалися різні автоматизовані методи, такі як час пропускної здатності, загальні символи та рівень виправлення помилок.

Таким чином, проведені експериментальні досліджені, які показують модель для виведення та генерації великої ємності даних для зберігання текстової інформації за допомогою методів стиснення, мультиплексування та багатошаровості. Якщо надати текстові дані, які відповідають специфікаціям створеної моделі рішення, ця модель дає позитивні результати ємності даних, використовуючи цей критерій. Модель здатна збільшити зберігання даних QR-коду, з точки зору використання модифікований метод, а саме із методів компресії, мультиплексування та багатошаровості.

Очікується, що ця модель дасть надію тим, хто зацікавлений у збільшенні даних в єдиному QR-коді. Ця перевага модифікованого методу може зменшити обмеження часу через меншу пропускну здатність та зменшити використання великої кількості QR-кодів.

На основі результатів, зібраних з проведеного експериментального дослідження, значно доводить, що модель збільшила ємність даних у порівнянні

з тим, що було використано в поточному дослідженні QR-коду. Крім того, підкріплений висновками, показано, що вміст QR-коду можна налаштувати швидше в порівнянні з традиційною технікою. Цей експеримент також показує, що запропонована модель здатна використовувати інші вхідні дані, такі як зображення, інші типи символів. Його можна прийняти в галузі контролю процесів, опису продукту, офлайн-електронної книги тощо.

Усі завдання з цього експерименту були виконані, а саме були розроблені: алгоритм кодування QR-коду використовуючи кольори RGB для збільшення сховища даних, алгоритм декодування, який відновлює всі збережені дані та алгоритм часткового декодування/перекодування, який дозволяє частково експортувати необхідні закодовані дані. Усі завдання з цього експерименту були виконані, а саме був розроблені: алгоритм кодування QR-коду використовуючи кольори RGB для збільшення сховища даних, алгоритм декодування, який відновлює всі збережені дані.

4. Розробити алгоритм часткового декодування та перекодування кольорового QR-коду RGB, який дозволяє частково витягти необхідні закодовані дані.

, і всі дослідницькі питання були вирішені. Оскільки цей експеримент повністю завершено, ця модель, яка виробляє кольорові QR-коди, готова до використання індустрією QR-кодів для економії ємності зберігання в майбутньому.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. «Історія QR-кодів та їх використання». [Електронний ресурс]. — Режим доступу: https://bazilik.media/istoriia-qr-kodiv-ta-ikh-vykorystannia-v-komunikatsiiakh/ — Дата доступу : Листопад 2022.

2. Content Idea of Asia Co. Ltd. «PM-code» [Електронний ресурс]. — Режим доступу : http://pm-code.com/eng/ — Дата доступу : Листопад 2022

3. «QR code.com». [Електронний ресурс]. — Режим доступу: https://www.qrcode.com/en/ — Дата доступу : Листопад 2022

4. «High Capacity Color Barcodes (HCCB)». [Електронний ресурс]. — Режим доступу: https://www.microsoft.com/en-us/research/project/high-capacity-color-barcodes-hccb/ — Дата доступу : Листопад 2022

5. Kim, H. M., Kim, W., & Cho, D. A New Color Transforming for RGB Coding. In ICIP. [Text] / K. H. Pandya, H.J. Galiyawala — International Journal of Emerging Technology and Advanced Engineering, 2014. — P. 258-262.

6. Pandya, K. H., Galiyawala, H. J. A Survey on QR Codes : in context of Research and Application [Text] / H. M. Kim, , W. Kim, D. Cho — International Conference, april 2004. — P. 107-110.

7. Taveerad, N., Vongpradhip, S. Development of Color QR Code for Increasing Capacity. [Text] / N. Taveerad, S. Vongpradhip — Proceedings - 11th International Conference on Signal-Image Technology and Internet-Based Systems, 2015. — P. 645-648.

8. Vongpradhip, S. Use Multiplexing to Increase Information in QR Code [Text] / S. Vongpradhip — The 8th International Conference on Computer Science & Education, 2013. — P. 361-364.

9. Victor, N. Enhancing the Data Capacity of QR Codes by Compressing the Data before Generation. [Text] / N. Victor — International Journal of Computer Applications, 2012. — P. 17-21.

ДОДАТОК А

Спосіб комбінації кольорів RGB для підвищення ефективності QR-кода

Лістинг фрагментів коду програми

Аркушів 14

Київ – 2022

- **Клас для запуску програми**

```
from Encoder.encode import Encoder
from Decoder.decode import Decoder
from PIL import Image

encoder = Encoder(error_correction="LOW")
decoder = Decoder()

f = open("Txt files/test.txt", "r")
text = encoder.encode(str(f.read()))
text.save("PNG/main.png")
outPutImage = Image.open("PNG/main.png")
result = decoder.decode(outPutImage)
print(result)
```

- **Клас Encoder**

```
# ecl: Error Correction Level(L,M,Q,H)
def encode(ver, ecl, str):
    mode_encoding = {
            'numeric': numeric_encoding,
            'alphanumeric': alphanumeric_encoding,
            'byte': byte_encoding,
            'kanji': kanji_encoding
            }

    ver, mode = analyse(ver, ecl, str)
```

```
    print('line 16: mode:', mode)

    code = mode_indicator[mode] + get_cci(ver, mode, str) +
mode_encoding[mode](str)

    # Add a Terminator
    rqbits = 8 * required_bytes[ver-1][lindex[ecl]]
    b = rqbits - len(code)
    code += '0000' if b >= 4 else '0' * b

    # Make the Length a Multiple of 8
    while len(code) % 8 != 0:
        code += '0'

    # Add Pad Bytes if the String is Still too Short
    while len(code) < rqbits:
        code += '1110110000010001' if rqbits - len(code) >= 16 else '11101100'

    data_code = [code[i:i+8] for i in range(len(code)) if i%8 == 0]
    data_code = [int(i,2) for i in data_code]

    g = grouping_list[ver-1][lindex[ecl]]
    data_codewords, i = [], 0
    for n in range(g[0]):
        data_codewords.append(data_code[i:i+g[1]])
        i += g[1]
    for n in range(g[2]):
        data_codewords.append(data_code[i:i+g[3]])
        i += g[3]
```

```
    return ver, data_codewords

def analyse(ver, ecl, str):
    if all(i in num_list for i in str):
        mode = 'numeric'
    elif all(i in alphanum_list for i in str):
        mode = 'alphanumeric'
    else:
        mode = 'byte'

    m = mindex[mode]
    l = len(str)
    for i in range(40):
        if char_cap[ecl][i][m] > l:
            ver = i + 1 if i+1 > ver else ver
            break

    return ver, mode

def numeric_encoding(str):
    str_list = [str[i:i+3] for i in range(0,len(str),3)]
    code = ''
    for i in str_list:
        rqbin_len = 10
        if len(i) == 1:
            rqbin_len = 4
        elif len(i) == 2:
            rqbin_len = 7
        code_temp = bin(int(i))[2:]
        code += ('0'*(rqbin_len - len(code_temp)) + code_temp)
```

```
    return code

def alphanumeric_encoding(str):
    code_list = [alphanum_list.index(i) for i in str]
    code = ''
    for i in range(1, len(code_list), 2):
        c = bin(code_list[i-1] * 45 + code_list[i])[2:]
        c = '0'*(11-len(c)) + c
        code += c
    if i != len(code_list) - 1:
        c = bin(code_list[-1])[2:]
        c = '0'*(6-len(c)) + c
        code += c

    return code

def byte_encoding(str):
    code = ''
    for i in str:
        c = bin(ord(i.encode('iso-8859-1')))[2:]
        c = '0'*(8-len(c)) + c
        code += c
    return code

def kanji_encoding(str):
    pass

# cci: character count indicator
def get_cci(ver, mode, str):
    if 1 <= ver <= 9:
```

```
        cci_len = (10, 9, 8, 8)[mindex[mode]]
    elif 10 <= ver <= 26:
        cci_len = (12, 11, 16, 10)[mindex[mode]]
    else:
        cci_len = (14, 13, 16, 12)[mindex[mode]]

    cci = bin(len(str))[2:]
    cci = '0' * (cci_len - len(cci)) + cci
    return cci

if __name__ == '__main__':
    s = '123456789'
    v, datacode = encode(1, 'H', s)
    print(v, datacode)
```

- **Клас Decoder**

```
def decoder(ver, ecl, data_codewords):
    en = ecc_num_per_block[ver-1][lindex[ecl]]
    ecc = []
    for dc in data_codewords:
        ecc.append(get_ecc(dc, en))
    return ecc

def get_ecc(dc, ecc_num):
    gp = GP_list[ecc_num]
    remainder = dc
    for i in range(len(dc)):
        remainder = divide(remainder, *gp)
    return remainder
```

```
def divide(MP, *GP):
    if MP[0]:
        GP = list(GP)
        for i in range(len(GP)):
            GP[i] += log[MP[0]]
            if GP[i] > 255:
                GP[i] %= 255
            GP[i] = po2[GP[i]]
        return XOR(GP, *MP)
    else:
        return XOR([0]*len(GP), *MP)



def XOR(GP, *MP):
    MP = list(MP)
    a = len(MP) - len(GP)
    if a < 0:
        MP += [0] * (-a)
    elif a > 0:
        GP += [0] * a

    remainder = []
    for i in range(1, len(MP)):
        remainder.append(MP[i]^GP[i])
    return remainder

def get_qrmatrix(ver, ecl, bits):
    num = (ver - 1) * 4 + 21
    qrmatrix = [[None] * num for i in range(num)]
```

```
    #  [([None] * num * num)[i:i+num] for i in range(num * num) if i % num ==
0]

    # Add the Finder Patterns & Add the Separators
    add_finder_and_separator(qrmatrix)

    # Add the Alignment Patterns
    add_alignment(ver, qrmatrix)

    # Add the Timing Patterns
    add_timing(qrmatrix)

    # Add the Dark Module and Reserved Areas
    add_dark_and_reserving(ver, qrmatrix)

    maskmatrix = [i[:] for i in qrmatrix]

    # Place the Data Bits
    place_bits(bits, qrmatrix)

    # Data Masking
    mask_num, qrmatrix = mask(maskmatrix, qrmatrix)

    # Format Information
    add_format_and_version_string(ver, ecl, mask_num, qrmatrix)

    return qrmatrix

def add_finder_and_separator(m):
    for i in range(8):
```

```
        for j in range(8):
            if i in (0, 6):
                m[i][j] = m[-i-1][j] = m[i][-j-1] = 0 if j == 7 else 1
            elif i in (1, 5):
                m[i][j] = m[-i-1][j] = m[i][-j-1] = 1 if j in (0, 6) else 0
            elif i == 7:
                m[i][j] = m[-i-1][j] = m[i][-j-1] = 0
            else:
                m[i][j] = m[-i-1][j] = m[i][-j-1] = 0 if j in (1, 5, 7) else 1

def add_alignment(ver, m):
    if ver > 1:
        coordinates = alig_location[ver-2]
        for i in coordinates:
            for j in coordinates:
                if m[i][j] is None:
                    add_an_alignment(i, j, m)

def add_an_alignment(row, column, m):
    for i in range(row-2, row+3):
        for j in range(column-2, column+3):
            m[i][j] = 1 if i in (row-2, row+2) or j in (column-2, column+2) else 0
    m[row][column] = 1

def add_timing(m):
    for i in range(8, len(m)-8):
        m[i][6] = m[6][i] = 1 if i % 2 ==0 else 0

def add_dark_and_reserving(ver, m):
    for j in range(8):
```

```
        m[8][j] = m[8][-j-1] = m[j][8] = m[-j-1][8] = 0
    m[8][8] = 0
    m[8][6] = m[6][8] = m[-8][8] = 1

    if ver > 6:
        for i in range(6):
            for j in (-9, -10, -11):
                m[i][j] = m[j][i] = 0

def place_bits(bits, m):
    bit = (int(i) for i in bits)

    up = True
    for a in range(len(m)-1, 0, -2):
        a = a-1 if a <= 6 else a
        irange = range(len(m)-1, -1, -1) if up else range(len(m))
        for i in irange:
            for j in (a, a-1):
                if m[i][j] is None:
                    m[i][j] = next(bit)
        up = not up

def mask(mm, m):
    mps = get_mask_patterns(mm)
    scores = []
    for mp in mps:
        for i in range(len(mp)):
            for j in range(len(mp)):
                mp[i][j] = mp[i][j] ^ m[i][j]
        scores.append(compute_score(mp))
```

```
    best = scores.index(min(scores))
    return best, mps[best]

def get_mask_patterns(mm):
    def formula(i, row, column):
        if i == 0:
            return (row + column) % 2 == 0
        elif i == 1:
            return row % 2 == 0
        elif i == 2:
            return column % 3 == 0
        elif i == 3:
            return (row + column) % 3 == 0
        elif i == 4:
            return (row // 2 + column // 3) % 2 == 0
        elif i == 5:
            return ((row * column) % 2) + ((row * column) % 3) == 0
        elif i == 6:
            return (((row * column) % 2) + ((row * column) % 3)) % 2 == 0
        elif i == 7:
            return        (((row + column) % 2) + ((row * column) % 3)) % 2 == 0

    mm[-8][8] = None
    for i in range(len(mm)):
        for j in range(len(mm)):
            mm[i][j] = 0 if mm[i][j] is not None else mm[i][j]
    mps = []
    for i in range(8):
        mp = [ii[:] for ii in mm]
        for row in range(len(mp)):
```

```
                for column in range(len(mp)):
                    mp[row][column] = 1 if mp[row][column] is None and formula(i, row,
column) else 0
            mps.append(mp)

        return mps

    def compute_score(m):
        def evaluation1(m):
            def ev1(ma):
                sc = 0
                for mi in ma:
                    j = 0
                    while j < len(mi)-4:
                        n = 4
                        while mi[j:j+n+1] in [[1]*(n+1), [0]*(n+1)]:
                            n += 1
                        (sc, j) = (sc+n-2, j+n) if n > 4 else (sc, j+1)
                return sc
            return ev1(m) + ev1(list(map(list, zip(*m))))

        def evaluation2(m):
            sc = 0
            for i in range(len(m)-1):
                for j in range(len(m)-1):
                    sc += 3 if m[i][j] == m[i+1][j] == m[i][j+1] == m[i+1][j+1] else 0
            return sc

        def evaluation3(m):
            def ev3(ma):
```

```
            sc = 0
            for mi in ma:
                j = 0
                while j < len(mi)-10:
                    if mi[j:j+11] == [1,0,1,1,1,0,1,0,0,0,0]:
                        sc += 40
                        j += 7
                    elif mi[j:j+11] == [0,0,0,0,1,0,1,1,1,0,1]:
                        sc += 40
                        j += 4
                    else:
                        j += 1
                return sc
            return ev3(m) + ev3(list(map(list, zip(*m))))

        def evaluation4(m):
            darknum = 0
            for i in m:
                darknum += sum(i)
            percent = darknum  / (len(m)**2) * 100
            s = int((50 - percent) / 5) * 5
            return 2*s if s >=0 else -2*s

        score = evaluation1(m) + evaluation2(m)+ evaluation3(m) + evaluation4(m)
        return score

    def add_format_and_version_string(ver, ecl, mask_num, m):
        fs = [int(i) for i in format_info_str[lindex[ecl]][mask_num]]
        for j in range(6):
            m[8][j] = m[-j-1][8] = fs[j]
```

```
    m[8][-j-1] = m[j][8] = fs[-j-1]
m[8][7] = m[-7][8] = fs[6]
m[8][8] = m[8][-8] = fs[7]
m[7][8] = m[8][-7] = fs[8]

if ver > 6:
    vs = (int(i) for i in version_info_str[ver-7])
    for j in range(5, -1, -1):
        for i in (-9, -10, -11):
            m[i][j] = m[j][i] = next(vs)
```