МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Мобільний додаток роботи з QR-кодами

Освітній ступінь "бакалавр"
Спеціальність 123 – "комп'ютерна інженерія"

Керівник проекту: _____ (підпис) Щербаков Є.В. (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Критська Я.О. (ініціали, прізвище)

Здобувач вищої освіти: _____ (підпис) Тінаєв О.О. (ініціали, прізвище)

Група: КІ-15бд

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній ступінь бакалавр
Напрям підготовки _____
(шифр і назва)
Спеціальність 123 – комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри КНІ
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Тінаєву Олегу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Мобільний додаток роботи з QR-кодами

керівник проекту (роботи) Щербаков Євген Іванович, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "13" 05 2019 р. № 84/15.15
2. Термін подання студентом роботи 16.06.2019
3. Вихідні дані до роботи Зображення для обробки, текстові дані для кодування інформації, перелік використовуваних програмних засобів: Android Studio

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Питання актуальності qr-кодів у сучасному світі, алгоритм генерації qr-коду, програмна реалізація мобільного додатку, охорона праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст. викл. Критська Я.О.		

7. Дата видачі завдання 30.04.2019

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз завдання та робота з літературою	05.05.2019 - 13.05.2019	
2	Аналіз технічних засобів	14.05.2019 - 22.05.2019	
3	Розробка алгоритму	22.05.2019 - 02.06.2019	
4	Програмна реалізація	02.06 .2019- 11.06.2019	
5	Оформлення пояснювальної записки та	11.06.2019 - 16.06.2019	
	електронних плакатів		

Здобувач вищої освіти _____ Тінаєв О.О.
(підпис) (прізвище та ініціали)

Керівник _____ Щербаков Є.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка дипломної роботи: 97 с., 46 рис., 4 табл., 15 джерел.

Робота присвячена розробці та реалізації мобільного застосунку для генерування та розпізнавання QR-кодів. Розпізнавання полягає в аналізі зображення для встановлення чи є QR-код присутній на зображенні. Рішення щодо розпізнавання приймається на основі знахождення обов'язкових елементів QR-коду, які дозволяють встановити, що даний об'єкт є QR-код. Після знаходження QR-коду на зображенні відбувається зчитування даних, які були закодовани та розміщенні на QR-коді. Процес генерації QR-кодів полягає у встановленні типу даних, які користувач бажає закодувати, та з застосуванням необхідних алгоритмів для маскування, встановлення бітів корекції та версії і типу QR-коду, вони розміщуються на QR-коді відповідно до алгоритму розміщення інформації на QR-коді.

Алгоритми генерування та розпізнавання були використані для створення спеціалізованого мобільного застосунку, який буде виконувати генерування та розпізнавання QR-кодів.

Ключові слова: РОЗПІЗНАВАННЯ QR-КОДІВ, ДЕКОДУВАННЯ ІНФОРМАЦІЇ, ГЕНЕРАЦІЯ QR-КОДІВ, МОБІЛЬНИЙ ЗАСТОСУНОК.

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А,. м. Сєвєродонецьк, 93400с.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

QR-code – quick response code – код швидкої відповіді

APK – android package kit – формат архівних файлів-застосунків android

BCH - коди Боуза — Чоудхури — Хоквингема

DEX – dalvik executable file – скомпільований файл кода android застосунку

OS – operating system – операційна система

JVM – java virtual machine – віртуальна машина java

ВСТУП

Кожен з нас в своєму повсякденному житті зустрічається з величезною кількістю зображень QR-коду. Це й не дивно. Після представлення QR-коду широкій публіці багато виробників і маркетологів стали використовувати QR-код як засіб швидкого доступу до інформації, так як для отримання інформації про продукт або послугу споживачеві потрібно просто навести камеру мобільного пристрою, що має сканер QR-коду, на QR-код і він буде перенаправлений на веб-сайт з усією запропонованою інформацією.

QR-код – двомірний код, який може містити як цифрову так і алфавітну інформацію. Може являти собою посилання на веб-ресурс, контактну інформацію, геолокацію, умови для підключення до wifi, та sms або email повідомлення. Через широкий спектр послуг, які надає QR-код своїм користувачам, він здобув широкого застосування та попиту. Вперше QR-код був використаний виробниками автомобілів для відстеження автомобілів і деталей.

QR-коди більш досконалі, ніж одномірний штрих-код, - вони мають більш високу ємність сховища і можуть зберігати різні типи символів. По суті, ці коди схожі на фізичні гіперпосилання, оскільки при їх скануванні користувач переходить на зовнішнє посилання або сайт.

На ринку існує досить рішень, але вони не покривають всього спектра можливостей використання та обробки QR-кодів. Існує багато програм, які надають функції або сканування або генерування кодів, не поєднуючи це в один додаток. Так само існують додатки, які виконують декодування кодів, але не можуть обробити контактну інформацію або інформацію яка містіть повідомлення з електронною поштою. Або мобільні застосунки які не можуть виконувати сканування якщо пристрій не має підключення до мережі інтернет.

Саме тому існує необхідність у створенні мобільного застосунку, який би дозволив людині відсканувати QR-код за найкоротший час, обробив би

отриманий результат та при наяві бажання, надав би можливість генерувати код самостійно з тими даними, які бажає користувач.

Ця робота присвячена розробці мобільного android застосунку, який би виконував розпізнавання QR-кодів, декодування інформації, та генерував би у QR-код інформацію, введену користувачем. Розроблений мобільний застосунок виконує усі функції незалежно від інших сервісів та застосунків, що робить його зручним у використанні у будь-яких умовах та з будь-якими даними.

1 ПИТАННЯ АКТУАЛЬНОСТІ QR-КОДІВ У СУЧАСНОМУ СВІТІ

QR-код був винайден тому що звичайні одномірні штрихові коди не могли містити у собі необхідну кількість інформації. Та японською компанією розробником було прийнято рішення почати експерементувати зі способами кодування інформації у графічній формі. Після того, як QR-код почали використовувати у промисловості та маркетингу, він набув широкої популярності серед користувачів, через його незвичний вигляд та простоту використання, що призвело до поширення використання QR-кодів.

QR-код може містити символи, цифри і текст. Через те, що QR-коди є горизонтальними і вертикальними, вони можуть містити ту ж інформацію, що і штрих-код, але займаючи лише 1/10 місця, яке знадобиться штрих-коду. Таким чином QR-коди є більш привабливим рішенням, ніж штрих-коди.

На відміну від старого штрихкоду, який сканують тонким променем, QR-код визначається датчиком або камерою як двовимірне зображення. Три квадрата в кутах зображення і менші синхронизирующие квадратики по всьому коду дозволяють нормалізувати розмір зображення і його орієнтацію, а також кут, під яким датчик розташований до поверхні зображення.

Але QR-код популярний не тільки в Японії, країні, де він був винайден, QR-код також знайшов користувачів серед інших країн. Наприклад, у Китаї майже всі платіжні системи використовують QR-код. Це дуже практично, тому що людині потрібно просто отсканувати зображення, яке містить інформацію про товар та його ціну, із спеціального програмного забезпечення, що гарантує безпеку при проведенні платежів, та підтвердити платіж онлайн. Це зберігає час та автоматизує багато процесів.

Проводити платежі не єдина функція та галузь застосування QR-кодів. QR-коди можуть містити інформацію різних форматів – контактну інформацію, інформацію для підключення до мережі Wifi, інформацію для sms або email повідомлення, геолокацію, посилання на веб-ресурси і також інформацію з номером телефону. Що користується попитом у багатьох

маркетологів. Зараз майже на кожному оголошенні можна побачити невеличкий QR-код, який би містив інформацію про місцезнаходження, контактні дані або посилання на веб-ресурс.

Також QR-коди використовуються для допомоги при навігації по місту або як туристичний довідник. У деяких містах QR-коди розташовані на асфальті біля пам’ятки, щоб будь-яка бажаюча людина могла переглянути інформацію та розширити свій кругозір. Також QR-коди наносять на знаки з вказання вулиць, щоб, отсканувавши, людина могла побачити конкретно карту з позначкою місцязнаходження.

Одна з переваг QR-коду - це легке розпізнавання скануючим обладнанням, що дає можливість використання в торгівлі, виробництві, логістиці [10].

QR-коди застосовуються для розміщення в рекламі, пошуку інформації, привітань зі святами, обміну контактами, авторизації та реєстрації в сервісах, підключення до WiFi в публічних місцях — для будь-яких дій, що мають на увазі інтерактив з власником смартфону.Для користувача код дав спромогу переходити за посиланнями, для рекламодавця — можливість інтеграції зовнішньої і ТБ-реклами з інтерактивом, доступним тільки в цифровому середовищі.

Таке швидке зростання популярності та широке використання багатьма рекламодавцями та маркетологами для просування інформації в маси зайвий раз доводить, що QR-коди не лише популярні на даний момент, але мають великий потенціал до розвитку у майбутньому.

1.1 Види двомірних кодів

Двомірні символіки були розроблені для кодування великого обсягу інформації. Розшифровка такого коду проводиться в двох вимірах (по горизонталі і по вертикалі).

Двомірні коди поділяються на багаторівневі і матричні. Багаторівневі штрих-коди з'явилися історично раніше, і являють собою поставлені один на одного кілька звичайних лінійних кодів. Матричні ж коди більш щільно упаковують інформаційні елементи по вертикалі.

1.1.1 Aztec Code

Aztec Code - двовимірний матричний штрихкод.

Aztec Code можливо застосовувати там, де площа для нанесення коду сильно обмежена. В першу чергу це обумовлено тим, що він не вимагає вільного місця навколо коду. Aztec Code може бути тільки квадратним. Сторона квадрата містить від 15 до 151 модуля. Код може об'єднуватися в блоки. Зрозуміло, як і інші типи кодів, Aztec Code підтримує корекцію даних за принципом Ріда-Соломона. При цьому він дозволяє налаштувати надмірність даних від 5% до 95%.

У багатьох країнах цей код використовується залізничними компаніями в електронних квитках. Крім того, він був обраний міжнародною асоціацією повітряного транспорту для електронних квитків. Приклад Aztec Code продемонстровано на рисунку 1.1:

Рисунок 1.1 – Aztec Code

Побудова відбувається на квадратній регулярної сітці.

На рисунку 1.2 показані області повної версії Aztec Code.

Червоним і чорним кольором позначені калібрувальні елементи. Структура цих областей незмінна і положення інших областей при зчитуванні обчислюється щодо них. Кількість пунктирних прямих може змінюватися в залежності від розміру використовуваного символу.

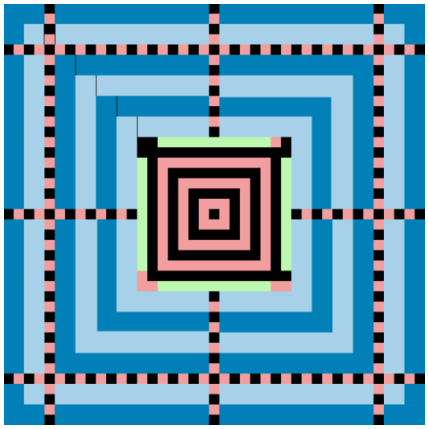


Рисунок 1.2 - області повної версії Aztec Code

Зеленим кольором позначені області для зберігання службової інформації, всього 40 біт: по одному десятібітовому блоку на кожній з чотирьох сторін.

Блакитним та синім кольорами позначені шари зберігання даних, що радіально розходяться від центру. Темна точка в цій області кодує логічну одиницю, світла - логічний нуль, дані в кожному шарі записуються по спіралі за годинниковою стрілкою, на ілюстрації тонкою лінією зліва зверху кожного шару показано його початок.

Завдяки навігаційним маркерами код не залежить від просторової орієнтації, і може бути лічений не тільки при будь-якому куті повороту, але і навіть при дзеркальному відображенні малюнка.

Розмір коду може варіюватися від квадрата 15x15 до квадрата 151x151. Найменший може містити в собі до 13 цифр або 12 букв англійського алфавіту, а найбільший - 3832 цифр або 3067 букв англійського алфавіту або 1914 байт даних. При цьому не потрібно порожнього простору навколо малюнка коду.

До особливостей Aztec Code можно віднести:

– наявність особливої системи розмітки, мішені, дозволяє зчитувати інформацію навіть з спотвореного зображення. Наприклад, повернутого або розтягнутого;

– у коді застосовується кодування Ріда-Соломона, що дозволяє успішно зчитувати код при частковому пошкодженні його поверхні. Стандартний рівень надмірності при кодуванні 23%, при цьому його можна змінювати від 5% до 95%;

– радіальне розташування шарів інформації дозволяє збільшувати обсяг інформації, що зберігається, просто розширюючи область кодування.

1.1.2 DataMatrix

DataMatrix - являє собою двомірну матрицю з чорно-білих точок або модулів. У коді повинно міститися парне число таких модулів як по вертикалі, так і по горизонталі. DataMatrix код може складатися як з одного, так і з декількох блоків. У кожному блоці обов'язково міститься дві суцільні пересічні лінії у вигляді букви L - так званий «шаблон пошуку», який допомагає зрозуміти орієнтацію коду для пристрою, що зчитує. Дві інші сторони блоку складаються з чорних і білих точок, що чергуються. Приклад DataMatrix наведено на рисунку 1.3.

Нова версія DataMatrix ECC 200 використовує коди Ріда-Соломона для запобігання помилок і відновлення стертої інформації. Це уможливлює відновлення всієї послідовності закодованої інформації, коли символ містить 30% пошкоджень, припускаючи, що матриця все ще розташована в точності правильно.

Рисунок 1.3 – Двомірний код DataMatrix

Символи мають парну кількість рядів і парна кількість стовпців. Більшість символів квадратні з розмірами модулів від 10x10 до 144x144. Однак деякі символи прямокутні і мають розміри від 8x18 до 16x48 модулів (тільки парні значення). Всі символи, що підтримують виправлення помилок, можуть бути розпізнані по верхньому правому кутовому модулю, який одного кольору з фоновим.

До особливостей нової версії DataMatrix можна віднести:

– зворотний порядок читання символів (світле зображення на темному фоні);

– специфікація набору символів;

– прямокутні символи;

– структурне приєднання (з'єднання до 16 символів для кодування більшої кількості інформації).

Головною перевагою цього різновиду двомірних кодів - є його надмалий розмір. За допомогою DataMatrix можна помістити інформацію в 50 символів на площу розміром в два квадратних міліметри. При цьому, код може бути нанесений на поверхню великою кількістю способів. Крім того, у коду є дві можливі форми: квадрат і прямокутник. Це дозволяє ще ефективніше використовувати доступну для розміщення коду площу.

Ці відмінності, присутні у DataMatix, роблять цей різновид двомірних кодів підхожий для маркування малих об'єктів, наприклад, мікросхем. І саме тому DataMatrix дуже активно застосовується в промисловості

Приклади застосування DataMatrix у промисловості та в повсякденному житті наведені на рисунках 1.4, 1.5.

Рисунок 1.4 – Застосування DataMatrix у промисловості

Рисунок 1.5 – Застосування DataMatrix у повсякденному житті для сортування пошти

1.1.3 MaxiCode

MaxiCode - загальнодоступний, машиночитаний двомірний штриховий код. Підходить для відстеження та управління відвантаженням пакетів, він нагадує штриховий код, але використовує точки, розташовані в гексагональній сітці замість смуг.

MaxiCode з'являється як 1 дюймовий квадрат з “мішенню” в центрі, оточений шаблоном гексагональних точок. Він може зберігати близько 93 символів інформації, і максимум 8 символів MaxiCode можуть бути об’єднані для передачі більшої кількості даних. “Мішень” яка знаходиться в центрі MaxiCode робить розпізнавання розташування MaxiCode та його орієнтації

простіше, та це дозволяє сканувати MaxiCode навіть на пакеті, який швидко рухається. Приклад MaxiCode наведено на рисунку 1.6:

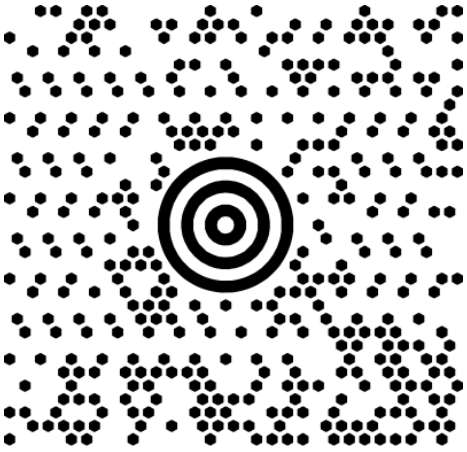


Рисунок 1.6 - MaxiCode

Режими MaxiCode:

– режим 0 - застарілий режим замінено режимами 2 і 3;

– режим 1 - режим застарілий, заміщений режимом 4;

– режим 2 - формовані дані, що містять структуроване повідомлення перевізника з цифровим поштовим кодом. (Первинне використання - це місцеві напрямки США.);

– режим 3 - формовані дані, що містять структуроване повідомлення перевізника з буквено-цифровим поштовим кодом. (Первинне використання є міжнародними напрямками.);

– режим 4 - неформатовані дані зі стандартною корекцією помилок;

– режим 5 - неформатовані дані з покращеною корекцією помилок;

– режим 6 - використовується для програмування апаратних пристроїв.

Елементи UPS використовують режим 2 або режим 3 MaxiCode.

Символи MaxiCode, що використовують режими 2 та 3, містять структуроване повідомлення перевізника, що містить ключову інформацію про пакет. Ця інформація захищена сильним кодом виправлення помилок Ріда-Соломона, що дозволяє його читати, навіть якщо частина символу пошкоджена. Ці поля включають:

– 4-бітна індикація режиму, що використовується, в даний час або режим 2, або режим 3;

– національний чи міжнародний поштовий індекс. MaxiCode підтримує як цифрові поштові коди (наприклад, поштовий індекс), так і буквено-цифрові поштові коди;

– 3-значний код країни, закодований на ISO 3166;

– 3-значний клас службового коду, присвоєний перевізником;

Структурована частина повідомлення зберігається у внутрішній частині символу поруч із зображенням “мішені”(у режимах, які не містять структуровану частину, внутрішня область просто зберігає початок повідомлення.). Приклад структури MaxiCode наведено на рисунку 1.7:

Рисунок 1.7 – Структура MaxiCode

Незалежно від режиму, велика кількість інформації про програму може бути закодована в символі MaxiCode. Формат цих додаткових даних не є чітко визначеним, але серед іншої інформації може бути:

– номер замовлення на поставку;

– клієнтська довідка;

– номер накладної;

– номер відстеження;

– індикатор вихідного перевізника.

1.1.4 PDF417

PDF417 – двомірний штрих-код, який підтримує кодування до 2710 знаків. В даний час PDF417 широко застосовується в ідентифікації особистості, обліку товарів, при здачі звітності до контролюючих органів та інших областях. Формат PDF417 відкритий для загального використання.

PDF417 використовується у багатьох програмах як комерційних, так і державних організацій. PDF417 - це один із форматів (разом із DataMatrix), який може використовуватися для друку поштових повідомлень. PDF417 також був вибраний стандартом Bar Codeed Boarding Pass (BCBP) у галузі авіакомпанії як символ 2D штрих-коду для пасажирських посадкових талонів. PDF417 - це стандарт, який був обран у деяких країнах, як машинозчитувана технологія для ідентифікаційного посвідчення водія та посвідчення особи. Штриховий код PDF417 наведено на рисунку 1.8:

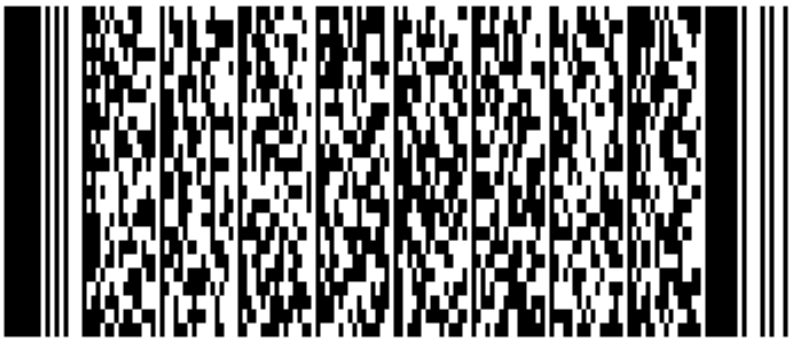


Рисунок 1.8 - Штриховий код PDF417

Штриховий код PDF417 складається з 3 до 90 рядків, кожен з яких схожий на малий лінійний штрих-код. Кожен рядок містить:

– тиха зона. Це стандарна мінімальна кількість пробілів до початку штрих-коду;

– початковий малюнок/шаблон, який визначає формат PDF417;

– кодове слово "рядок ліворуч", що містить інформацію про рядок (наприклад, номер рядка та рівень корекції помилок);

– кодові слова даних 1-30: кодове слово - це група смуг та пробілів, що представляють собою один чи більше цифр, літер або інших символів;

– кодове слово "правий рядок" з додатковою інформацією про рядок;

– схема зупинки;

– інша тиха зона.

Структура PDF417 наведена на рисунку 1.9:

Рисунок 1.9 – Структура коду PDF417

PDF417 використовує базове кодування 929. Кожне кодове слово представляє цифру від 0 до 928.

Кодові слова представлені шаблонами темного (полоска) та світлого (простір) областей. Кожен з цих моделей містить чотири чорні полоси і чотири білих полоси (простір). Загальна ширина в 17 разів перевищує ширину найменшої дозволеної вертикальної смуги. Кожен шаблон починається з чорної смуги і закінчується пробілом (біла смуга).

Висота лінії повинна бути мінімум як 3 мінімальної ширини смуги.

Існує три чітких шаблону просторів, що використовуються для відображення кожного коду слова. Ці моделі складаються з трьох груп, відомих як кластери. Кластери позначені як 0, 3 та 6. У шаблоні пробіл не використовується більше, ніж в одному кластері. Рядки символів повторюються через три кластери.

Однією з цілей трьох кластерів є визначення того, який рядок входить в кодове слово. Кластери дозволяють прочитати частини символу, використовуючи одну лінію сканування, яка може бути перевернута з

горизонталі. Наприклад, сканування може починатися з рядка 6 на початку рядка, але закінчується на рядку 10. На початку сканування сканер бачить початковий шаблон, а потім відображає символи в кластері 6. При перекрученому скануванні він перетинає рядки 6 і 7, тоді сканер бачить шум. Коли сканування відбувається в рядку 7, сканер бачить символи в кластері 0. Отже, сканер знає напрямок перекосу. До того часу, коли сканер досяг правого, він знаходиться на 10 рядку, так що він бачить кластер 0 шаблонів. Сканер також побачить схему зупинки.

З 929 доступних кодових слів, 900 використовуються для даних, а 29 для спеціальних функцій, таких як переміщення між основними режимами. Три основні режими кодують різні типи даних різними способами і можуть бути змішані при необхідності в межах одного штрих-коду:

– байт: кожна група з 5 кодових слів представляє 6 байт. (Оскільки 9005> 2566.) Додаткові байти кодуються по одному для кодового слова;

– цифри: n цифри закодовані в ⌊n / 3⌋ + 1 кодове слово, максимум 44 цифри в 15 кодових словах;

– текст: кожне кодовое слово представляє два байти-30 цифр, які використовуються системою з чотирьох підмоделів, що представляють друковані символи ASCII (а також CR, LF і HT):

1) верхній регістр: A-Z, SP, змінити в нижньому регістрі, змінити на змішаний, інтерпретувати наступну цифру як пунктуацію;
2) нижній регістр: a-z, SP, інтерпретувати наступну цифру у верхньому регістрі, змінити на змішаний, інтерпретувати наступну цифру як пунктуацію;
3) змішаний: 0-9, &, CR, HT, кома,:, #, -, період, $, /, +,%, *, =, ^, змінити пунктуацію, SP, змінити в нижньому регістрі, інтерпретувати наступну цифру як пунктуацію;
4) пунктуація:;, <,>, @, [, \,], _, `, ~,!, CR, HT, комами,:, LF, -, точка, $, /,", |, *, (,),?, {,}, ', змінити у верхньому регістрі.

Коли створюється PDF417, додаються кодові слова та слова корекції помилок кількістю від 2 до 512. PDF417 використовує корекцію помилок Ріда-Соломона. Коли символ сканується, максимальна кількість виправлень, що можуть бути зроблені, дорівнює кількості доданих кодових слів, однак стандарт рекомендує затримати два кодових слова, щоб забезпечити надійність виправленої інформації.

1.1.5 Microsoft Tag

Microsoft Tag - двомірний кольоровий штрихкод (High Capacity Color Barcode - HCCB), розроблений Microsoft. Був спеціально розроблений для розпізнавання за допомогою фотокамер, вбудованих в мобільні телефони. Призначений для швидкої ідентифікації та отримання на пристрій заздалегідь підготовленої інформації, яка прив'язана до коду і зберігається на сервері компанії Microsoft.

Просунуті методики обробки зображень дозволяють декодувати навіть розфокусовані знімки, зроблені з використанням об'єктивів з нерухомою, сфокусованої на нескінченність оптикою, які присутні більшості стільникових телефонів. Приклад Microsoft Tag наведено на рисунку 1.10.

Особливості Microsoft Tag, переваги і недоліки :

– коди є кольоровими, отже, вимагають застосування кольорових друкувальних і відеоконтрольних пристроїв, камер що знімають. Якщо кольорові монітори і камери в даний час складають переважну більшість, чорно-білі принтери все ще досить широко поширені, але створення чорно-білих тегів підтримується штатно;

– використання кольорових трикутників дозволяє зберігати більшу кількість інформації при тому ж фізичному розмірі елементів зображення;

– на поточний момент вся інформація зберігається на серверах Microsoft, а отже вимагає підключення до Інтернету мобільного пристрою;

– інформацію несуть виключно невеликі круги в центрах трикутників і кінці синхронізаційних ліній. На всьому іншому просторі може бути що завгодно, тому можливі MS Tag'i з малюнками;

– технологія чутлива до якості і чистоті рамки коду, наприклад, напис, зроблений відразу під чорної рамкою коду, робить його невалідним і може привести до збою розпізнавання;

– коди чутливі до геометричних спотворень, тому для їх використання потрібно пласка поверхня. З цієї ж причини на розпізнавання впливає кут, під яким код видно зчитувальних пристроїв.

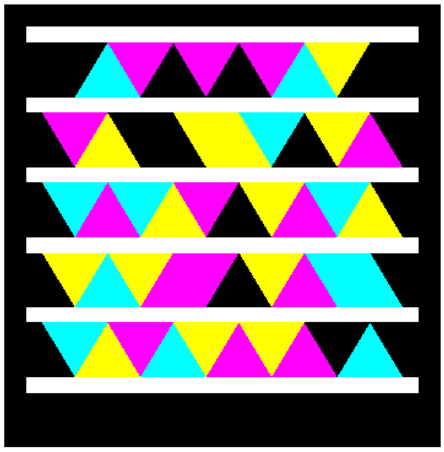


Рисунок 1.10 – Штриховий код Microsoft Tag

1.1.6 QR-код

QR-код - товарний знак для типу матричних штрих-кодів (або двовимірних штрих-кодів), від самого початку розроблених для автомобільної промисловості Японії.

Приклад QR-коду наведено на рисунку 1.11:

Рисунок 1.11 – Двомірний штриховий код QR-код

Система QR-кодів стала популярною за межами автомобільної промисловості завдяки можливості швидкого зчитування і більшої місткості в порівнянні зі штрих-кодами стандарту UPC.

QR-код складається з чорних квадратів, розташованих у квадратної сітці на білому тлі, які можуть зчитуватися за допомогою пристроїв обробки зображень, таких як камера, і оброблятися з використанням кодів Ріда - Соломона до тих пір, поки зображення не буде належним чином розпізнано. Потім необхідні дані витягуються з шаблонів, які присутні в горизонтальних і вертикальних компонентах зображення. Структуру QR-коду наведено на рисунку 1.12.

Існує декілька версій (розмірів) QR – кодів. Найменший QR-код (версія 1) має розмір 21 × 21 піксель (без урахування полів), найбільший (версія 40) - 177 × 177 пікселів. На відміну від більш старих одномірних штрих-кодів, які були розроблені для механічного сканування вузьким променем світла, QR-код виявляється двомірним датчиком цифрового зображення, а потім цифровим чином аналізується запрограмованим процесором. Процесор знаходить три відмінні квадрати в кутах зображення QR-коду, використовуючи менший квадрат (або кілька квадратів) біля четвертого кута для нормалізації зображення за розміром, орієнтацією та кутом перегляду.

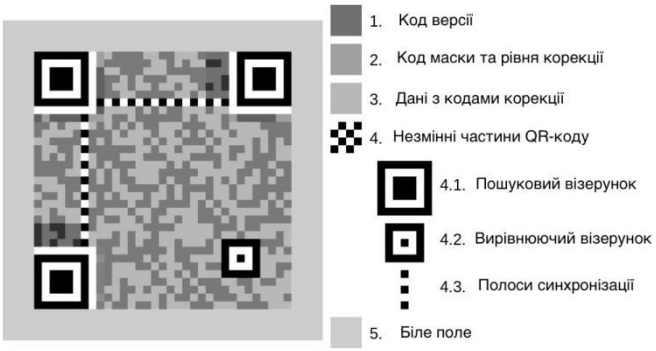


Рисунок 1.12 – Структура QR-коду

Невеликі крапки по всьому QR-коду потім перетворюються на двійкові числа та перевіряються з алгоритмом виправлення помилок.

Існує чотири основних типів кодування QR-кодів, вони наведені у таблиці 1.1:

Таблиця 1.1 – основні типи кодування символів у QR - код

Тип	Максимальна кількість символів	Біт/символ	Можливі символи, кодування за замовчуванням
Цифрове	7,089	3,333	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Алфавітно-цифрове	4,296	5,5	0–9, A–Z (верхній регістр), пробіл, $, %, *, +, -, ., /, :
Байтове	2,953	8	дані в будь-якому зручному кодуванні (за замовчуванням ISO 8859-1)
Кандзі	1,817	13	японські ієрогліфи згідно зі стандартами Shift JIS, X 0208

Цифрове кодування - цей тип кодування вимагає 10 біт на 3 символу. Вся послідовність символів розбивається на групи по 3 цифри, і кожна група

(тризначне число) переводиться в 10-бітове двійкове число і додається до послідовності біт. Якщо загальна кількість символів не кратне 3, то якщо в кінці залишається 2 символи, отримане двозначне число кодується 7 битами, а якщо 1 символ, то 4 бітами. Наприклад, є рядок «12345678», яку треба закодувати. Послідовність розбивається на числа: 123, 456 і 78, потім кожне число переводиться в двійковий вигляд: 0001111011, 0111001000 і 1001110, і об'єднується це в один потік бітів: 000111101101110010001001110.

Алфавітно-цифрове - на відміну від цифрового кодування, для кодування 2 символів потрібно 11 біт інформації. Послідовність символів розбивається на групи по 2, в групі кожен символ кодується згідно таблиці «Значення символів в буквено-цифровому кодуванні». Значення першого символу множиться на 45, потім до цього твору додається значення другого символу. Отримане число переводиться в 11-бітове двійкове число і додається до послідовності біт. Якщо в останній групі залишається один символ, то його значення кодується 6-бітовим числом. Розглянемо на прикладі: «PROOF». Розбиваємо послідовність символів на групи: PR, OO, F. Знаходимо відповідні значення символам до кожної групи (дивимося в таблицю): PR- (25,27), OO- (24,24), F- (15). Знаходимо значення для кожної групи: 25 * 45 + 27 = 1152, 24 * 45 + 24 = 1104, 15 = 15. Переводимо кожне значення в двійковий вигляд: 1152 Отримати = 10010000000, 1104 = 10001010000, 15 = 001111. Об'єднуємо в одну послідовність: 10010000000100010100000001111.

Байтове кодування - таким способом кодування можна закодувати будь-які символи. Вхідний потік символів кодується в будь-якому кодуванні (рекомендовано в UTF-8), потім переводиться в двійковий вигляд, після чого об'єднується в один потік бітів.

Наприклад, слово «Мир» кодируем в Unicode (HEX) в UTF-8: М - D09C; і - D0B8; р - D180. Переводимо кожне значення в двійкову систему числення: D0 = 11010000, 9C = 10011100, D0 = 11010000, B8 = 10111000, D1

= 11010001 і 80 = 10000000; об'єднуємо в один потік біт: 11010000 10011100 11010000 10111000 11010001 10000000.

Кандзі - в основі кодування ієрогліфів (як і інших символів) лежить візуально сприймається таблиця або список зображень ієрогліфів з їх кодами. Така таблиця називається «character set». Для японської мови основне значення мають дві таблиці символів: JIS 0208: 1997 і JIS 0212: 1990. Друга з них є доповненням до першої. JIS 0208: 1 997 розбита на 94 сторінки по 94 символу. Наприклад, сторінка 4 - хирагана, 5 - катакана, 7 - кирилиця, 16-43 - кандзі рівня 1, 48-83 - кандзі рівня 2. Кандзі рівня 1 («JIS Дайіті суйдзюн кандзі») впорядковані по Онам. Кандзі рівня 2 (JIS Дайна суйдзюн кандзі) впорядковані по ключам, і всередині них - за кількістю рис.

Також існує декілька різновидів QR-кодів. Найбільш популярним з них є Micro QR Code.

Micro QR Code - менша версія стандарту QR-коду для випадків, де розмір символу обмежений. Є чотири різних версії (розміри) Micro QR-кодів: найменший - 11 × 11 модулів; найбільший може вмістити 35 цифрових символів. Приклад Micro QR Code наведено на рисунку 1.13.

IQR код є альтернативою існуючим QR-кодами, розробленими компанією, що створила QR-коди. Коди IQR можуть бути створені в квадратних або прямокутних формаціях; це призначене для ситуацій, коли прямокутний штрих-код був би доречнішим, наприклад, циліндричні об'єкти. Коди IQR можуть відповідати такої ж кількості інформації та займати на 30% менше місця.

Рисунок 1.13 - Micro QR Code

Є 61 версія квадратних кодів IQR та 15 версій прямокутних кодів. Для квадратів мінімальний розмір - 9x9 модулів; прямокутники мають мінімум 19x5 модулів. Коди IQR додають рівень корекції помилок S, що дозволяє корекцію помилок 50%. Коди IQR ще не отримали специфікації ISO, і лише запатентовані продукти Denso Wave можуть створювати або читати коди IQR. Приклад IQR наведено на рисунку 1.14:

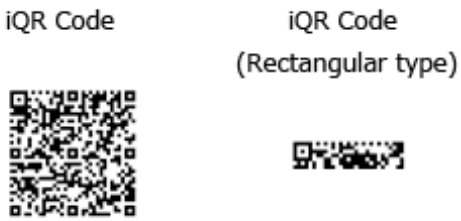


Рисунок 1.14 – IQR код

SQRC - це тип QR-коду з функцією обмеження читання. Це може бути використано для зберігання приватної інформації та управління внутрішньою інформацією компанії тощо. Зовнішній вигляд SQRC не відрізняється від звичайного QR-коду.Всі функції, що постачаються зі звичайним QR-кодом, включаючи функцію корекції помилок, всі зберігаються. Приклад SQRC наведено на рисунку 1.15:

Рисунок 1.15 – Приклад SQRC коду

Frame QR - це QR-код з "областю полотна", який можна гнучко використовувати. У центрі цього коду є область полотна, де можна гнучко розташувати графіку, літери та інші об’єкти, що дозволяє викласти код без втрати дизайну, ілюстрацій, фотографій тощо.

1.2 Постановка задачі

Метою даної роботи є створення мобільного застосунку, здатного розпізнавати і генерувати QR-коди. Для створення мобільного застосунку необхідно вирішити такі завдання:

– розробити архітектуру мобільного застосунку;

– розробити дизайн мобільного застосунку;

– розробити web view для мобільного застосунку;

– розробити базу даних;

– дослідити методи розпізнавання ключових точок QR-коду на зображенні;

– дослідити методи декодування даних, які були закодовані у QR-код;

– дослідити методи обробки інформації в процесі декодування QR-коду;

– дослідити методи кодування інформації у QR-код;

– дослідити методи обробки інформаціх у процесі кодування;

– програмно реалізувати метод для розпізнавання QR-коду на зображенні;

– програмно реалізувати метод для декодування інформації з QR-коду;

– програмно реалізувати метод по обробці декодованої інформації;

– програмно реалізувати метод для генерування QR-коду спираючись на дані, введені користувачем;

– програмно реалізувати метод для генерування QR-коду спираючись на тип введених даних.

2 АЛГОРИТМ ГЕНЕРАЦІЇ QR-КОДУ

2.1 Кодування інформації

QR код підтримує кілька способів кодування даних, в залежності від того, які символи використовуються: цифрове, алфавітно-цифрове, кандзі (китайсько-японські ієрогліфи) і побайтове кодування [7, 8]. Цифрове кодування має на увазі використання тільки цифр від 0 до 9, алфавітно-цифрове - прописні букви латинського алфавіту, цифри і символи $% * + -. /: І пробіл. Спочатку вам треба створити порожню послідовність біт, яка далі буде заповнюватися.

Для того, щоб кодувати дані за допомогою одного з цих методів, ми спочатку вказуємо, який метод ми використовуємо та скільки даних ми зберігаємо. Ми вказуємо метод з чотирма бітами - цифровий - 0001, алфавітно-цифровий - 0010, бінарний - 0100, а кандзі - 1000.

Метод кодування визначає, скільки бітів ми використовуємо для позначення довжини даних. Деталі наведено в таблиці 2.1 нижче. Як приклад, якщо ми кодуємо 5 бінарних символів версії 1, то двійкові дані, з яких ми починаємо, буде 0100 00000101. (0100 для позначення бінарного типу кодування, а 00000101 - це 8-бітове зображення 5 для позначення довжини даних.)

Таблиця 2.1 – Кількість бітів для методів кодування

Тип	Версія 1 - 9	Версія 10 – 26	Версія 27 - 40
Цифрове	10 біт	12 біт	14 біт
Алфавітно-цифрове	9 біт	11 біт	13 біт
Байтове	8 біт	16 біт	16 біт

Після того, як ми вказали, що ми зберігаємо, тепер настав час дійсно додати дані. Бінарний метод найпростіший - просто 8-бітове представлення символу та додаємо його до набору даних. Для числових даних ви берете набори з трьох цифр. Для кожного набору з трьох ви кодуєте їх безпосередньо до їх 10-бітних двійкових уявлень та додаєте до послідовності біт. Якщо в кінці даних залишилась 1 цифра, кодуємо її на чотири біти, а якщо залишилося 2 цифри, кодуємо їх на сім бітів.

Алфавітно-цифрові дані трохи складніші. У цьому випадку на 2 символи потрібно 11 біт інформації. Вхідний потік символів розділяється на групи по 2, в групі кожен символ кодується згідно таблиці 2.2:

Таблиця 2.2 – Значення символів у алфавітно-цифровому кодуванні

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
U	V	W	X	Y	Z	Пробіл	$	%	*	+	-	.	/	:
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44

Значення першого символу в групі множиться на 45 і додається до значення другого символу. Отримане число переводиться в 11-битное двійкове число і додається до послідовності біт. Якщо в останній групі 1

символ, то його значення безпосередньо кодується 6-бітним числом і додається до послідовності біт.

2.2 Додавання службової інформації

На цьому етапі треба визначитися з рівнем корекції - чим вищий цей рівень, тим більше допустимий рівень пошкодження QR коду, на якому його ще можна відновити, і тим менше інформації поміститься на QR коді фіксованого розміру. Всього є 4 рівня корекції: L (допустимо максимум 7% пошкоджень), M (15%), Q (25%) і H (30%). Найчастіше використовується рівень M.

Ще одна властивість QR коду - його версія (чим вона більша, тим більше розмір QR коду). Всього існує 40 версій. Номер версії залежить від кількості інформації для кодування і від рівня корекції. У таблиці 2.3 зазначено максимальну кількість корисної інформації разом із службовою (в бітах), яку можна закодувати в QR коді певної версії. З цієї таблиці визначається версія QR коду.

Таблиця 2.3 – Максимальна кількість інформації QR-коду згідно з його версією

	1	2	3	4	5	6	7	8	9	10
L	152	272	440	640	864	1088	1248	1552	1856	2192
M	128	224	352	512	688	864	992	1232	1456	1728
Q	104	176	272	384	496	608	704	880	1056	1232
H	72	128	208	288	368	480	528	688	800	976

Продовження таблиці 2.3

	11	12	13	14	15	16	17	18	19	20
L	2592	2960	3424	3688	4184	4712	5176	5768	6360	6888
M	2032	2320	2672	2920	3320	3624	4056	4504	5016	5352
Q	1440	1648	1952	2088	2360	2600	2936	3176	3560	3880
H	1120	1264	1440	1576	1784	2024	2264	2504	2728	3080
	21	22	23	24	25	26	27	28	29	30
L	7456	8048	8752	9392	10208	10960	11744	12248	13048	13880
M	5712	6256	6880	7312	8000	8496	9024	9544	10136	10984
Q	4096	4544	4912	5312	5744	6032	6464	6968	7288	7880
H	3248	3536	3712	4112	4304	4768	5024	5288	5608	5960
	31	32	33	34	35	36	37	38	39	40
L	14744	15640	16568	17528	18448	19472	20528	21616	22496	23648
M	11640	12328	13048	13800	14496	15312	15936	16816	17728	18672
Q	8264	8920	9368	9848	10288	10832	11408	12016	12656	13328
H	6344	6760	7208	7688	7888	8432	8768	9136	9776	10208

Тепер треба перед послідовністю біт, отриманої в попередньому пункті, додати на початку два поля: спосіб кодування і кількість даних. Спосіб кодування - поле довжиною 4 біта, яке має наступні значення: 0001 для цифрового кодування, 0010 для алфавітно-цифрового і 0100 для побайтового. Кількість даних - це кількість кодованих символів, а для побайтового кількість байт (а не біт в отриманої послідовності), у вигляді двійкового числа, довжина якого визначається по таблиці 2.1.

Кількість даних - це рядок бітів, що представляє кількість символів, що кодуються. Кількість даних слід розміщувати після індикатора режиму кодування. Крім того, індикатор кількості даних має бути певною кількістю бітів довжиною, залежно від QR-версії.

Перерахуємо кількість символів у початковому вхідному тексті, а потім перетворимо це число в двійкове. Довжина індикатора кількості символів залежить від режиму кодування та версії QR-коду, яка буде використовуватися. Якщо після перетворення кількості символів у бінарний вигляд, отримана кількість символів не відповідає довжині, вказаної для заданого типу кодування, доповніть її зліва нулями.

Наприклад, якщо кодувати HELLO WORLD в QR-коді версії 1 в алфавітно-цифровому режимі, показник кількості символів повинен бути 9 біт (згідно таблиці 2.1). Кількість символів HELLO WORLD - 11. У бінарному вигляді 11 - 1011. Зліва доповнюємо нулямі, щоб зробити послідовність довжиною 9 біт: 000001011. Перед отриманою послідовністю для кількості символів дописуємо спосіб кодування, та отримуємо: 0010 000001011.

Тепер в нас є послідовність закодованих біт даних, інформація о способі кодування та кількість символів. Якщо бітовий рядок коротший за загальну кількість необхідних бітів, то до правої сторони рядка повинен бути доданий темінатор з максимум чотирьох нулів. Якщо бітовий рядок ніж на чотири біта коротший, ніж потрібна кількість бітів, додаємо до кінця чотири 0. Якщо бітовий рядок менше, ніж чотири біта коротший, додаємо лише кількість 0, необхідних для досягнення потрібної кількості бітів. Ми розбиваємо послідовність на групи по 8 біт, якщо після додавання термінатора кількість бітів у рядку не кратна 8, потрібно перевірити чи остання група з 8 біт дійсно містить 8 біт, якщо ні – доповнити нулями доки група не буде містити 8 біт. Якщо навіть після цього послідовніть не є того розміру, який вказано для обраної версії QR-коду, треба доповнити її байтами що чергуються 11101100 00010001. Ці байти еквівалентні 236 і 17, відповідно. Вони спеціально вказані специфікацією QR-коду, щоб додати, якщо бітовий рядок на цьому етапі є занадто коротким.

2.3 Розділення інформації на блоки

Послідовність байт, отримана в попередньому етапі, розділяється на певну кількість блоків, наведених у таблиці 2.4, згідно з версію та рівнем корекції.

Таблиця 2.4 - кількість блоків для певних версій QR-коду

	1	2	3	4	5	6	7	8	9	10
L	1	1	1	1	1	2	2	2	2	4
M	1	1	1	2	2	4	4	4	5	5
Q	1	1	2	2	4	4	6	6	8	8
H	1	1	2	4	4	4	5	6	8	8
	11	12	13	14	15	16	17	18	19	20
L	4	4	4	4	6	6	6	6	7	8
M	5	8	9	9	10	10	11	13	14	16
Q	8	10	12	16	12	17	16	18	21	20
H	11	11	16	16	18	16	19	21	25	25
	21	22	23	24	25	26	27	28	29	30
L	8	9	9	10	12	12	12	13	14	15
M	17	17	18	20	21	23	25	26	28	29
Q	23	23	25	27	29	34	34	35	38	40
H	25	34	30	32	35	37	40	42	45	48
	31	32	33	34	35	36	37	38	39	40
L	16	17	18	19	19	20	21	22	24	25
M	31	33	35	37	38	40	43	45	47	49
Q	43	45	48	51	53	56	59	62	65	68
H	51	54	57	60	63	66	70	74	77	81

Для того, щоб визначити кількість байт у кожному блоці необхідно розділити всю кількість байт (можна визначити кількість байт в даних або розділити число з таблиці 2.3 на вісім) на кількість блоків даних. Якщо це число не ціле, то треба визначити залишок від ділення. Цей залишок визначає скільки блоків з усіх доповнені (такі блоки, кількість байт в яких більше на один ніж в інших). Всупереч сподіванням, доповненими блоками повинні бути не перші блоки, а останні.

Блок заповнюється байтами з даних повністю. Коли поточний блок повністю заповнюється, черга переходить до наступного. Байтів даних повинно вистачити рівно на всі блоки, ні більше і ні менше.

2.4 Створення байтів корекції

Алгоритм Ріда-Соломона застосовується до кожного блоку даних (якщо блок даних один, то просто до даних).

При побудові коду Ріда-Соломона задається пара чисел N, K, де N – загальне кількість символів, а K - «корисне» кількість символів, решта N-K символів є надлишковий код, призначений для відновлення помилок. Такий код матиме так звану «відстань Хеммінга» D = N - K +1;

Відстань Хеммінга є параметром коду і визначається як мінімальне число відмінностей між двома різними кодовими словами. Відповідно до теорії кодування, код, який має відстань Хеммінга D = 2t + 1, дозволяє відновлювати t помилок.

Алгорітм Ріда-Соломона використовує Поле Галуа для підрахування кодових слів корекції.

Стандарт QR-коду говорить про те, що використовується побітове складання за модулем 2 та побайтове складання за модулем 100011101. Це означає використання поля Галуа 2^8 або, іншими словами, поле Галуа 256, яке іноді називається $GF(256)$ [9].

Для побудови поля Галуа ми можемо уявити усі числа у $GF(256)$ як 2^n, де $0 \le n \le 255$. Але усі значення поля Галуа повинні бути у диапазоні $(0.....255)$, тому 2^8 має значення, яке виходить за цей діапазон, так как $2^8 = 256$.

Специфікація QR-коду говорить про те, що у таких випадках треба використовувати побайтове складання за модулем 100011101 (де 100011101 є двійковим числом, еквівалентним 285 в десятковій формі).

Це означає, що коли число 256 або більше, треба застосувати операцію “логічне або” з 285. Наприклад:

$$2^8 = 256 \oplus 285 = 29;$$

$$2^9 = 2^8 * 2 = 29 * 2 = 58.$$

Так само треба продовжувати доки ми не знайдемо 2^{255}.

Після того як ми знайшли усі значення для поля Галуа наступним кроком є генерування поліному.

Кодове слово за допомогою алгоритма Ріда-Соломона генерується за допомогою спеціального полінома. Всі дійсні кодовані слова точно поділяються на многочлен генератора. Загальна форма порождуючого многочлена:

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1})...(x - \alpha^{b+2t}).$$

Якщо ми хочемо згенерувати поліном для версії QR-коду, яка містить 2 кодових слова корекції, то він матиме вигляд:

$$(x - \alpha^0) * (x - \alpha^1) -> (\alpha^0 x - \alpha^0) * (\alpha^0 x - \alpha^1).$$

Перемножуємо кожний множник з першої різниці на кожний множник другої.

$$(\alpha^0 x - \alpha^0) + (\alpha^0 x^0 * \alpha^0 x^1) + (\alpha^0 x^1 * \alpha^1 x^0) + (\alpha^0 x^0 * \alpha^1 x^0);$$

$$(\alpha^{(0+0)} x^{(1+1)}) + (\alpha^{(0+0)} x^{(0+1)}) + (\alpha^{(0+1)} x^{(1+0)}) + (\alpha^{(0+1)} x^{(0+0)});$$

$$\alpha^0 x^2 + \alpha^0 x^1 + \alpha^1 x^1 + \alpha^1 x^0.$$

Виконуємо групування, так як ми маємо два x^1

$$\alpha^0 x^2 + (\alpha^0 + \alpha^1) x^1 + \alpha^1 x^0;$$

$$x^2 + (1+2) x^1 + 2x^0 -> x^2 + 3x^1 + 2x^0.$$

Використовуючи значення обратного поля Галуа, ми перетворюємо числову формулу до формули з α.

$$\alpha^0 x^2 + \alpha^{25} x^1 + \alpha^1 x^0.$$

Це і буде поліном для двох кодових слов корекції.

Для трьох кодових слов корекції поліном буде розраховуватись так:

Поліном з попереднього кроку треба помножити на $(x - \alpha^2)$

$$(\alpha^0 x^2 + \alpha^{25} x^1 + \alpha^1 x^0) * (\alpha^0 x^1 + \alpha^2 x^0);$$

$$(\alpha^0 x^2 * \alpha^0 x^1) + (\alpha^{25} x^1 * \alpha^0 x^1) + (\alpha^1 x^0 * \alpha^0 x^1) +$$
$$+ (\alpha^0 x^2 * \alpha^2 x^0) + (\alpha^{25} x^1 * \alpha^2 x^0) + (\alpha^1 x^0 * \alpha^2 x^0);$$

$$(\alpha^{(0+0)} + x^{(2+1)}) + (\alpha^{(25+0)} + x^{(1+1)}) + (\alpha^{(1+0)} x^{(0+1)}) +$$
$$+ (\alpha^{(0+2)} x^{(2+0)}) + (\alpha^{(25+2)} x^{(1+0)}) + (\alpha^{(1+2)} x^{(0+0)});$$

$$\alpha^0 x^3 + \alpha^{25} x^2 + \alpha^1 x^1 + \alpha^2 x^2 + \alpha^{27} x^1 + \alpha^3 x^0;$$

$$\alpha^0 x^3 + (\alpha^{25} + \alpha^2) x^2 + (\alpha^1 + \alpha^{27}) x^1 + \alpha^3 x^0;$$

$$\alpha^0 x^3 + (3+4) x^2 + (2+12) x^1 + \alpha^3 x^0;$$

$$\alpha^0 x^3 + 7x^2 + 14x^1 + \alpha^3 x^0;$$

$$\alpha^0 x^3 + \alpha^{198} x^2 + \alpha^{199} x^1 + \alpha^3 x^0.$$

За таким принципом можливо порахувати поліноми для необхідної кількості кодових слів корекції, згідно з кількістю слів, вказаною у таблиці стандарту для QR-кодів.

Для того, щоб згенерувати байти корекції для закодованих даних треба:

– закодовані дані згідно з одним із методів кодування інформації у QR-код;

– конвертирувати ці дані із бінарного представлення у десятичне;

– побудувати поліном даних, де коефіцієнтами виступають десяткові значення закодованих даних;

– згідно версії QR-коду порахувати поліном;

– поділити поліном даних на поліном версії.

Якщо ми закодуємо QR-код версії 1 з рівнем корекції Q у алфавітно-цифровому режимі кодування. Рівень корекції Q у алфавітно-цифровому режимі може містити 16 символів.

Нехай дані будуть строкою “HELLO WORLD”. Після використання алфавітно-цифрового алгоритму кодування, дані будуть мати вигляд у двійковій формі:

00100000 01011011 00001011 01111000 11010001
01110010 11011100 01001101 01000011 01000000
11101100 00010001 11101100 00010001 11101100
00010001.

У десятичній:

32, 91, 11, 120, 209, 114, 220, 77, 67, 64, 236, 17, 236, 17, 236, 17.

Згідно з десятичної формою, поліном даних буде мати вигляд:

$$32x^{15} + 91x^{14} + 11x^{13} + 120x^{12} + 209x^{11} + 114x^{10} +$$
$$+220x^{9} + 77x^{8} + 67x^{7} + 64x^{6} + 236x^{5} + 17x^{4} +$$
$$+236x^{3} + 17x^{2} + 236x^{1} + 17.$$

Згідно з даними стандарту QR-коду для 1-Q треба створити поліном для 10 байтів корекції. Такий поліном буде мати вигляд:

$$x^{10} + \alpha^{251}x^9 + \alpha^{67}x^8 + \alpha^{46}x^7 + \alpha^{61}x^6 + \alpha^{118}x^5 + \alpha^{70}x^4 + \\ + \alpha^{64}x^3 + \alpha^{94}x^2 + \alpha^{32}x + \alpha^{45}.$$

Поліном даних множимо на x^n , де n - кількість байтів корекції для даної версії QR-коду.

$$32x^{25} + 91x^{24} + 11x^{23} + 120x^{22} + 209x^{21} + 114x^{20} + \\ + 220x^{19} + 77x^{18} + 67x^{17} + 64x^{16} + 236x^{15} + 17x^{14} + \\ + 236x^{13} + 17x^{12} + 236x^{11} + 17x^{10}.$$

Перший символ полінома байтів корекції повинен бути у той ж ступіні як і перший символ поліному даних, тому помножимо поліном байтів корекції на x^{15} :

$$\alpha^0 x^{25} + \alpha^{251}x^{24} + \alpha^{67}x^{23} + \alpha^{46}x^{22} + \alpha^{61}x^{21} + \alpha^{118}x^{20} + \alpha^{70}x^{19} + \\ + \alpha^{64}x^{18} + \alpha^{94}x^{17} + \alpha^{32}x^{16} + \alpha^{45}x^{15}.$$

$32x^5$ - є першим символом поліному даних. Згідно зі значеннями поля Галуа, 32 дорівнює α^5 . Множимо поліном байтів корекції на α^5:

$$\alpha^5 x^{25} + \alpha^{(256\%255)}x^{24} + \alpha^{72}x^{23} + \alpha^{51}x^{22} + \alpha^{66}x^{21} + \alpha^{123}x^{20} + \\ + \alpha^{75}x^{19} + \alpha^{69}x^{18} + \alpha^{99}x^{17} + \alpha^{37}x^{16} + \alpha^{50}x^{15}.$$

Що дорівнює:

$$\alpha^5 x^{25} + \alpha^1 x^{24} + \alpha^{72}x^{23} + \alpha^{51}x^{22} + \alpha^{66}x^{21} + \alpha^{123}x^{20} + \\ + \alpha^{75}x^{19} + \alpha^{69}x^{18} + \alpha^{99}x^{17} + \alpha^{37}x^{16} + \alpha^{50}x^{15}.$$

У десятичному вигляді це буде:

$$32x^{25} + 2x^{24} + 101x^{23} + 10x^{22} + 97x^{21} + 197x^{20} +$$
$$+15x^{19} + 47x^{18} + 134x^{17} + 74x^{16} + 5x^{15}.$$

Наступним кроком виконуємо операцію “логічне або”:

$$(32 \oplus 32)x^{25} + (91 \oplus 2)x^{24} + (11 \oplus 101)x^{23} + (120 \oplus 10)x^{22} +$$
$$+(209 \oplus 97)x^{21} + (114 \oplus 197)x^{20} + (220 \oplus 15)x^{19} +$$
$$+(77 \oplus 47)x^{18} + (67 \oplus 134)x^{17} + (64 \oplus 134)x^{16} +$$
$$+(236 \oplus 5)x^{15} + (17 \oplus 0)x^{14} + (236 \oplus 0)x^{13} + (17 \oplus 0)x^{12} +$$
$$+(236 \oplus 0)x^{11} + (17 \oplus 0)x^{10}.$$

Результат:

$$0x^{25} + 89x^{24} + 110x^{23} + 114x^{22} + 176x^{21} + 183x^{20} +$$
$$+211x^{19} + 98x^{18} + 197x^{17} + 10x^{16} + 233x^{15} + 17x^{14} +$$
$$+236x^{13} + 17x^{12} + 236x^{11} + 17x^{10}.$$

Перший елемент дорівнює нулю, тому відкидуємо його. Оновлений поліном має вигляд:

$$89x^{24} + 110x^{23} + 114x^{22} + 176x^{21} + 183x^{20} +$$
$$+211x^{19} + 98x^{18} + 197x^{17} + 10x^{16} + 233x^{15} + 17x^{14} +$$
$$+236x^{13} + 17x^{12} + 236x^{11} + 17x^{10}.$$

Тепер першим елементом є $89x^{24}$. Згідно зі значеннями поля Галуа - $89 = \alpha^{210}$.

Всі етапи ділення повторюються з попереднього кроку. Таких кроків повинно буті 16, тому що згідно стандарту QR-коду, 1-Q має містити 16 байт даних. Після виконання усіх 16 кроків ми отримуємо такий поліном:

$$196x^9 + 35x^8 + 39x^7 + 119x^6 + 235x^5 + 215x^4 +$$
$$+231x^3 + 226x^2 + 93x^1 + 23.$$

Та кодовими словами корекції для даних будуть:

$$196353911923521523122693223.$$

2.5 Об’єднання блоків

На цьому етапі ми маємо декілька блоків даних та стільки ж блоків байтів корекції, треба їх об’єднати в один спільний поток байт.

Робиться це в такий спосіб: з кожного блоку даних по черзі береться один байт інформації, коли черга доходить до останнього блоку, з нього береться байт і черга переходить до першого блоку. Так триває до тих пір, поки в кожному блоці не закінчаться байти. Якщо в поточному блоці вже немає байт, то він пропускається (таке відбувається, коли звичайні блоки вже порожні, а в доповнених ще є по одному байту). Аналогічним чином треба зробити з блоками байтів корекції. Вони беруться в тому ж порядку, що і відповідні блоки даних. Якщо ж згідно стандарту дана версія QR-коду має лише один блок з кодовими словами корекції. У цьому випадку немає необхідності у процедурі об’єднання блоків. Просто записується послідовність із закодованих даних а за нею послідовність із слов корекції.

2.6 Розміщення інформації на QR-коді

У нас є послідовніть байт даних, яка готова для того, щоб її помістили на полотно. Полотно складається з модулів - елементарних квадратів, рисунок 1.12.

Основними базовими елементами QR-коду є:

Пошукові шаблони - це візерунки, які вдають із себе чорний квадрат розміром 3 на 3 модуля, який оточений рамкою з білих модулів, яка виділена рамкою з чорних модулів, яка виділена рамкою з білих модулів тільки з тих сторін, де немає відступу. Пошукові візерунки розташовуються у верхніх і лівих кутах (всього 3).

Вирівнюючі шаблони - Використовуються починаючи з 2-ї версії, вдають із себе чорний квадрат розміром 1 на 1 модуль, який оточений рамкою з білих модулів, яка виділена рамкою з чорних модулів, в результаті цей візерунок має розмір 5 на 5. Є одна важлива умова: вирівнюють візерунки не повинні нашаровуватися на пошукові візерунки.

Смуги синхронізації - Тут все просто, згідно з рисунком 2.1, вони починаються від самого нижнього правого чорного модуля верхнього лівого пошукового візерунка і йдуть, чергуючи чорний і білий модуль, вниз і вправо до протилежної пошукового модуля. При нашаруванні на вирівнюючий модуль він повинен залишитися без змін.

Код версії - ці елементи використовуються починаючи з 7-ї версії QR-коду. Код версії дублюється в 2-х місцях, причому дзеркально, тобто вказавши колір модуля в координатах (x, y) , можна сміливо вказувати такий же колір в координатах (y, x). Модулі в цих місцях розташовуються згідно зі стандартом для QR-коду.

Після розміщення базових елементів на полотні QR-коду можна починати заповнювати його даними. Вільний простір на полотні розбивається на стовпчики: кожнен по 2 модуля, не важливо що знаходиться в цих модулях, крім вертикальної смуги синхронізації, яка просто пропускається. Заповнення починається з правого нижнього кута, йде в межах стовпчика справа наліво, знизу вгору. Якщо поточний модуль зайнятий (наприклад смугою синхронізації або вирівнює візерунком), то він просто пропускається. У випадку досягнення верх стовпчика, то рух триває з верхнього правого кута стовпчика, який розташований лівіше, і йде зверху вниз. Досягнувши низу, рух триває від нижнього правого кута стовпчика, який розташований лівіше, і йде від низу до верху. І так далі, поки весь вільний простір не буде заповнено. Процедура заповнення полотна даними продемонстрована на рисунку 2.1.

Заповнення відбувається біт за бітом з байтів даних, при цьому 1 це чорний модуль, а 0 - білий. Якщо даних не вистачає, то залишившийся

простір заповнюється нульовими модулями. При цьому на кожен модуль накладається маска. Згідно зі стандратом для QR-коду існує 8 масок (від 0 до 7). Маска вибирається по різному: деякі завжди використовують одну і ту ж, інші щораз випадкову, але специфікація наполягає, щоб кожна маска оцінювалася і вибиралася найоптимальніша. Від обраної маски залежить код маски і рівня корекції.

Вибір найоптимальнішої маски полягає у тому, що треба згенерувати QR-код з кожною з восьми масок, нарахувати кожної штрафні очки за певними правилами, вказаними у стандарті, і вибрати маску з найменшою кількістю очок.

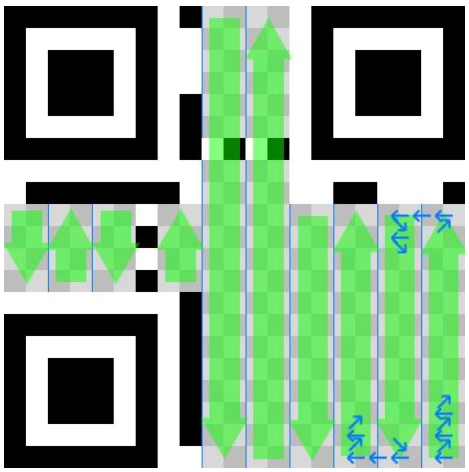


Рисунок 2.1 – Процедура заповнення QR-кода

Інформація про формат додається після того, як була обрана маска, тому що вона містить інформацію о коді маскування та коді рівня корекції, а маскування визначається вже після того, як дані були нанесені на полотно QR-коду.

Інформація про формат являє собою послідовність з 15 бітів, що містить 5 біт даних і 10 біт виправлення помилок, що обчислюються за кодом Боуза-Чоудхурі-Хоквінгема (BCH). Поліном, коефіцієнти якого біти рядка даних(дані про маску та рівень корекції), ділять на породжующий поліном, який має вигляд:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

У бінарному вигляді це буде:

10100110111.

Ділення виконується наступним чином:

У нас є інформація про маску та рівень корекції у бінарному вигляді, наприклад рівень корекції L та 4 маска:

$$01 -> L;$$
$$100 -> 4 mask.$$

Разом:

01100.

Через те що інформація про формат містить 15 бітів, а послідовність інформації про маску та рівень корекції разом будуть довжиною 5 бітів, нам треба зробити її довжиною 15 бітів, дописавши необхідну кількість нулів:

011000000000000.

Тепер убираємо нулі з лівої частини:

11000000000000.

Ділення виконується доки результуюча строка формату буде 10 бітів чи коротша. Перед кожною новою процедурою ділення треба перевірити чи поточна строка формату довжиною 11 або більше бітів, тому що 11 – довжина породжуючого поліному.

Після цього можно починати ділення, але строка з бінарним уявленням інформації про формат довше за бінарну послідовність породжуючого поліному, тому нам треба дописати 0 у кінці полінома, щоб зробити їх однакової довжини:

11000000000000;
10100110111000.

Тепер застосовуємо операцію складання за модулем 2:

$$11000000000000 \oplus 10100110111000 = 01100110111000.$$

Також прибираємо нулі з лівої сторони:

$$01100110111000 -> 1100110111000.$$

Отримана строка довжиною 13 бітів, це більше 11, тому ми продовжуємо ділення. Так продовжується до четвертої операції ділення, тому що на четвертому кроці розмір послідовності дорівнюватиме 10.

Отримана послідовність на 4 кроці:

1000111101.

Тепер ми дописуємо 5 біт інформації про маску та рівень корекції до отриманої строки з 10 бітів виправлення помилок:

011001000111101.

Згідно зі стандартом для QR-кодів, до результуючої строки інформації про формат треба застосувати операцію складання по модулю 2 зі строкою:

101010000010010.

Результатом буде:

$$011001000111101 \oplus 101010000010010 = 110011000101111.$$

Це і є рядок з інформацією про формат, який записується на полотно QR-коду у місця, позначені на рисунку 1.12.

Інформація про версію QR-кода - являє собою 18-бітову послідовність, що складається з 6 бітів даних і 12 бітів виправлення помилок за кодом Голея. Специфікація QR-коду говорить про використання коду Голея(18, 6) для рядка інформації про версію. Породжуючим поліномом, згідно зі стандартом для QR-коду, буде поліном вигляду:

$$x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^2 + 1.$$

Кроки ділення такі самі як при розрахуванні інформації про формат, за виключенням того, що в даному випадку довжина рядку повинна бути 18 бітів, а не 15, та ми припиняємо ділити якщо поточний рядок довжиною 12 чи менше бітів.

Після отримання дворічного рядку з інформацією про формат, треба нанести його на полотно QR-коду, згідно з його стандартним місцезнаходженням, яке продемонстровано на рисунку 1.12.

Також кожен QR-код повинен містити чорний піксель, який не повинен перекриватися іншими елементами QR-коду. Місцезнаходження цього пікселю обчислюється згідно з версією та стандартом для QR-коду, та для кожної версії він знаходиться біля правого верхнього кута пошукового візерунку, що знаходиться у нижньому лівому куті, як продемонстровано на рисунку 2.2:

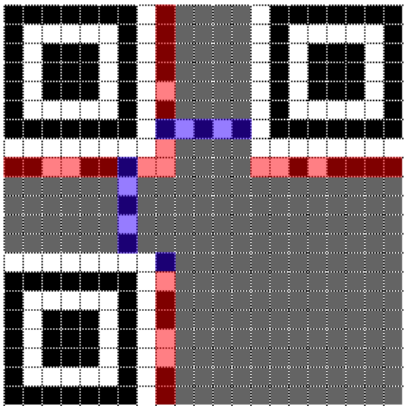


Рисунок 2.2 – Положення чорного пікселю на полотні QR-кода

2.7 Декодування QR-коду

Етапи декодування, починаючи з читання символу QR-коду і закінчуючи формуванням вихідного рядка знаків, є зворотними по відношенню до процедури кодування.

1) Визначається місце розташування та отримується зображення символу. Темні і світлі модулі представляються як масив бітів “0”, так і “1”. Визначається полярність коефіцієнтів відображення по модулям шаблона пошуку.

2) Зчитується інформація про формат. Реалізується шаблон маски і проводиться процедура виправлення помилок на модулях інформації про формат, якщо необхідно; у разі успіху символ має звичайну орієнтацію, в іншому випадку вживають спробу декодування інформації про формат для зображення в дзеркальному відображенні. Визначаємо рівень виправлення помилок у символах QR Code - безпосередньо.

3) Зчитується інформація про версію (де це можливо), і визначається версія символу.

4) З інформації про формат отримується маска шаблона покажчика і реалізується процедура маскування даних, застосовуючи операцію XOR до бітів області даних з бітами шаблону маски.

5) Зчитуються знаки символу відповідно до правил розміщення для моделі та відновлюють слова даних та виправлення помилок у повідомленні.

6) Визначається кількість кодових слів виправлення помилок на підставі рівня виправлення помилок і знаходимо помилкові кодові слова, використовуючи кодові слова виправлення помилок. Якщо помилки були виявлені, виправляють їх.

7) Розділити кодові слова даних на сегменти відповідно до індикаторів режимів та індикаторів числа знаків.

8) У заключенні декодуються знаки даних відповідно до використовуваних режимів і виводиться результат.

За допомогою даного рекомендованого алгоритму декодування знаходять на зображенні символи QR Code і декодують їх.

Алгоритм розпізнавання, що відноситься до визначення темних і світлих точок в зображенні.

1) Визначають глобальний поріг, використовуючи отримані значення коефіцієнта відображення, як середнє між мінімальним і максимальним коефіцієнтом відображення в зображенні; перетворюють зображення в масив темних і світлих пікселів на основі глобального порога.

2) Визначають місце знаходження шаблону пошуку. Шаблон пошуку для символів QR Code складається з трьох однакових шаблонів, розташованих в трьох з чотирьох кутів символу. Шаблон пошуку для сім волів Micro QR Code складається з єдиного шаблону. модулі в кожному шаблоні пошуку формують послідовність темний-світлий-темний-світлий-темний, відносна значення ширини кожного елемента якої знаходяться в співвідношенні 1:1:3:1:1. Для цілей даного алгоритму допускається відхилення кожного значення ширині до 0,5 (тобто в діапазоні від 0,5 до 1,5 модулів для області шириною в 1 модуль і від 2,5 до 3,5 модулів для квадратної області шириною 3 модуля).

а) при виявленні області-кандидата, відзначають позиції першої і останньої точки (A і B відповідно) на лінії з пікселів, яка

перетинається з зовнішніми кордонами шаблону пошуку на зображенні (рис 2.3). Повторюють вищевказане для суміжних ліній пікселів в зображенні до тих пір, поки не будуть ідентифіковані всі лінії, які перетинають центральний блок шаблону пошуку по осі X зображення.

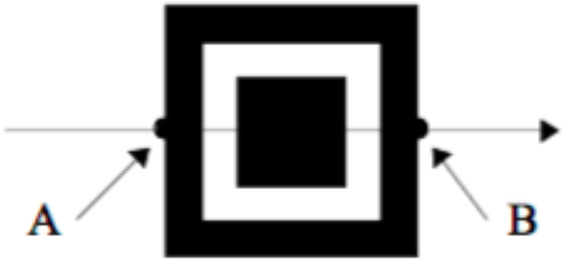


Рисунок 2.3 – Лінія сканування у шаблоні пошуку

б) повторюють етап а) для стовпців пікселів, які перетинають центральний блок знайденого шаблону пошуку по осі У.

в) визначають місце знаходження центру шаблону. Проводять лінію через точки, що знаходяться по середині між точками A і B для ліній, що перетинають центральний блок шаблону пошуку уздовж осі X, побудованих на етапі 1). Проводять лінію через точки, що знаходяться посередині між точками A і B для ліній, що перетинають центральний блок шаблону пошуку уздовж осі Y, побудованих на етапі b). Центр шаблону пошуку повинен знаходитися в точці перетину цих двох ліній.

г) повторюють кроки а) - с), щоб знайти центри двох інших шаблонів виявлення позиції.

3) Визначають кутову орієнтацію символу, аналізуючи координати центрів знайдених шаблонів пошуку для уточнення, який з шаблонів знаходиться в лівому верхньому кутку символу, і таким чином знаходять кут повороту символу.

4) визначають:

а) відстань D на лінії, що перетинає символ і проходить через центри лівого верхнього і правого верхнього шаблонів;

б) ширину двох цих шаблонів W_{UL}, W_{UR} , як показано на рис 2.4.

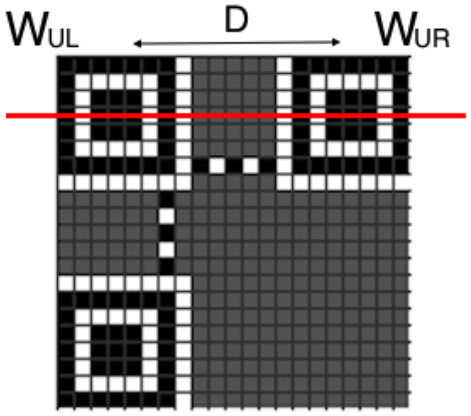


Рисунок 2.4 – Шаблони пошуку

5) Обчислюють номінальний розмір X для символу

$$X = (W_{UL} + W_{UR}) / 14.$$

6) Обчислюють приблизну версію символу V

$$V = \frac{(D / X) - 10}{4}.$$

7) Якщо приблизна версія символу менше 6, її приймають в якості остаточної версії. Якщо приблизна версія 7 або більше, декодують інформацію про версії наступним чином:

а) ділять ширину верхнього лівого шаблону W_{UR} на 7, що б обчислити ширину одного модуля - CP_{UR}

$$CP_{UR} = W_{UR} / 7.$$

б) знаходять напрямні лінії AC і AB, що проходять через центри трьох шаблонів пошуку A, B і C згідно з рис 2.5. Сітку вибірки центрів модулів блоку 1 інформації про версії визначають на

основі ліній, паралельних напрямних лініях, координат центрів шаблонів пошуку та раз міра модуля CP_{UR}. Визначають двійкові значення 0 і 1 по комбінації світлих і темних 51ік селів на сітці вибірці.

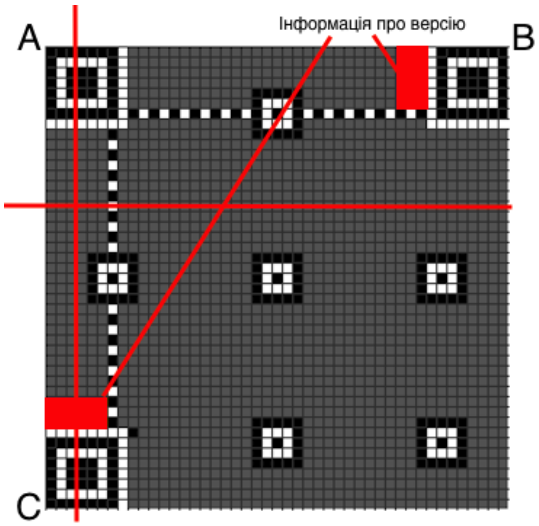


Рисунок 2.5 – Шаблони пошуку та інформація про версію

в) визначають версію, використовуючи виявлення та виправлення помилок, 51рунтуючись на корекції помилок BCH.

г) якщо число виявлених помилок перевищує здатність до їх виправлення, обчислюють ши рину лівого нижнього шаблону W_{DL} і виконують дії, зазначені на етапах a), b) та c).

8) Для символу версії 1, повторно визначають розмір X як середній інтервал між центрами темних і світлих модулів у верхньому шаблоні синхронізації. Подібним чином обчислюють розмір Y як середній інтервал між центрами темних і світлих модулів в лівому шаблоні синхронізації. Визначають сітку вибірки на підставі:

а) базової горизонтальної лінії, проведеної через верхній шаблон синхронізації, спільно з лініями, паралельними базової горизонтальної лінії на відстані Y, включаючи шість ліній

вище базової горизонтальної лінії і стільки ліній нижче цієї лінії, скільки потрібно для даної версії;

б) вертикальної лінії, проведеної по лівій стороні шаблону синхронізації спільно з лініями, паралельними цієї лінії і проведеними на расстоянии X, включаючи шість ліній лівіше цієї лінії і стільки ліній правіше цієї лінії, скільки потрібно для даної версії.

Для символів версії 2 і більше, визначають координати центрів всіх напрямних шаблонів а потім формують сітку вибірки з лініями, проведеними на рівному віддаленні від цих точок.

а) Ділять ширину W_{UL} левого верхнього шаблону пошуку P_{UL} на 7, щоб обчислити розмір модуля CP_{UL}

$$CP_{UL} = W_{UL} / 7.$$

б) Визначають приблизні координати центрів напрямних $P1, P2$ шаблонів на основі координат центру лівого верхнього шаблону пошуку (рис 2.6)

в) сканує напрямні шаблони починаючи з приблизною координати цін трального пікселя до контуру білого квадрата, щоб знайти фактичні координати центру

г) визначають приблизні координати центру направляючого шаблону $P3$ на основі координат центру верхнього лівого шаблону пошуку і фактичних координат центрів напрямних шаблонів $P1, P2$

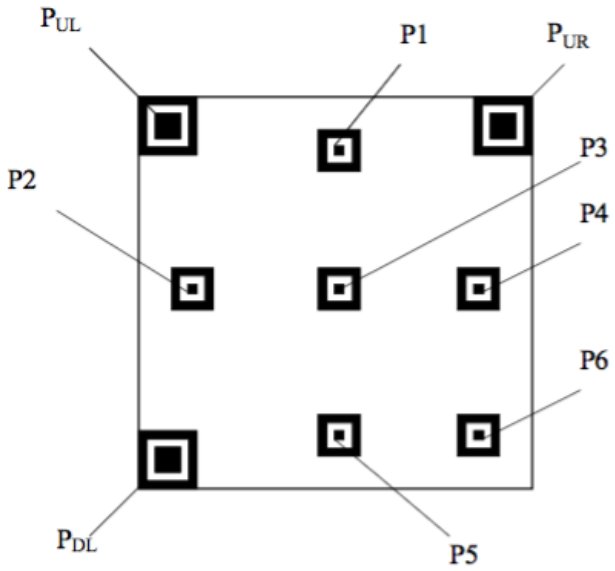


Рисунок 2.6 – Напрямні шаблони

д) Знаходять фактичні координати центру направляючого шаблону $P3$ як у пункті с)

е) Знаходять Lx - відстань по горизонталі між центрами напрямних шаблонів $P2, P3$ і Ly, відстань по вертикалі між центрами напрямних шаблонів $P1, P3$. Ділять Lx, Ly на AP - відстань в модулях між центрами напрямних шаблонів, для отримання розмірів модуля CPx по лівій стороні та CPy по правій стороні у верхній лівій області символу:

$$CPx = Lx / AP;$$

$$CPy = Ly / AP.$$

ж) таким же чином знаходять Lx', відстань по горизонталі між центрами верхнього лівого шаблону пошуку P_{UL} і направляючого шаблону $P1$ і Ly', відстань по вертикалі між центрами верхнього лівого шаблону пошуку P_{UL} і направляючого шаблону $P2$. Ділять Lx' і Ly', для отримання розмірів модуля CPx' по верхній стороні і CPy' по лівій стороні у верхній лівій області символу за формулою:

$CPx' = Lx'$ / (координата стовпця центрального модуля направляючого шаблону $P1$ - координата стовпця центрального модуля верхнього лівого шаблону пошуку P_{UL});

$CPy' = Ly'$ / (координата стовпця центрального модуля направляючого шаблону $P2$ - координата стовпця центрального модуля верхнього лівого шаблону пошуку P_{UL}).

з) Визначають сітку вибірки, яка покриває верхню ліву область символу на основі розмірів модулів CPx, CPx', CPy і CPy', представлених по кожній стороні верхньої лівої області символу.

к) Для направляючого шаблону $P6$ (рис.2.6) визначають приблизні координати цін трального модуля з розмірів модулів CPx' і CPy', значення яких визначені з відстаней між напрямними шаблонами $P3, P4$ і $P5$, шляхом проведення допоміжних ліній через центри направляючих шаблонів $P3, P4$ і через $P3, P5$ відповідно і обліку центральних координат цих шаблонів.

л) Повторюють етапи e) - h) для визначення сітки вибірки для нижньої правої області символу.

м) Таким же чином визначають сітки вибірки для неохоплених областей символу.

9) Переглядають області зображення розміром $3x3$ пікселів з центрами на перетинах ліній сітки вибірки і визначають, є цей модуль світлим або темним на підставі глобального порога. Формують бітову матрицю, привласнюючи двійкову 1 темним модулям і двійковий 0 світлим модулям.

10) Розшифровують інформацію про формат біля верхнього лівого шаблону виявлення позиції, щоб отримати рівень корекції помилок та шаблон маски, застосовані до символу. Якщо кількість виявлених помилок

перевищує здатність їх виправлення, повторюють цю процедуру для декодування інформації про формат, суміжну з верхнім правим і нижнім лівим шаблонами пошуку.

11) Виконують процедуру маскування, застосовуючи операцію XOR шаблону маски даних до області кодування символу, і відновлюють знаки символу, що представляють дані і кодові слова виправлення помилок. Ця процедура є зворотньою процесу маскування даних, застосованної в процесі кодування.

12) Визначають кодові слова символу згідно з правилами розміщення.

13) Перегруповують кодові слова по блокам, як потрібно для даної версії символу і рівня виправлення помилок.

14) Проводять декодуючими процедуру виявлення і виправлення помилок, щоб виправити таке число помилок і стирань, яке максимально можливо для даної версії символу і рівня виправлення помилок.

15) Відновлюють початковий двійковий потік повідомлення, збираючи блоки даних в послідовність.

16) Поділяють двійковий потік даних на сегменти, кожен з яких починається з індикатора режиму і індикатора числа знаків.

17) Декодують кожен сегмент відповідно правилам, дійсними для даного режиму.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках дипломної роботи був розроблений мобільний застосунок для генерації та розпізнавання QR-кодів. Мобільний застосунок призначений для роботи з мобільними пристроями під керуванням мобільної операційної системи Android [2, 4]. Для реалізації цього застосунку було обране середовище Android Studio 3.0.1 та мова програмування Java [1, 3]. Це обумовлено тим, що Android Studio було створено саме для програмування під платформу Android і ця IDE підтримує всі необхідні функції для роботи саме з мобільною операційною системою Android, що робить процес розробки більш комфортним та зручним. Також Android Studio підтримує систему збірки Gradle, яка не тільки відповідає за збірку проекту, але також є дуже потужним інструментом для підключення різних бібліотек та плагінів до проекту.

Деякими із головних переваг Android Studio є:

– візуальний редактор макету – дозволяє швидко та зручно додавати необхідні об’єкти до макету (будь то текстове поле, поле для зображення і т.д.), шляхом перетягування необхідних об’єктів із Palette - редактору макету (рис 3.1) та також дозволяє редагувати розмір та положення елементів на макеті у реальному часі, шляхом перетягування необхідних елементів безпосередньо на макеті (рис. 3.2, рис. 3.3) або використовуючи XML редактор;

– Android Studio містить APK Analyzer, який забезпечує негайне розуміння складу APK мобільного застосунку після завершення процесу побудови. Використання APK Analyzer дозволяє зменшити час, на виявлення проблем з DEX-файлами та ресурсами у додатку, а також зменшує розмір файлу APK.

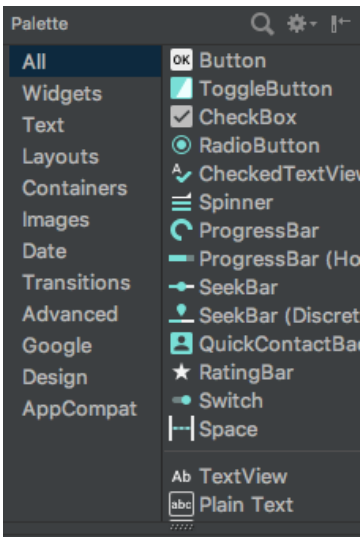


Рисунок 3.1 – Palette - палітра з елементами для додавання на макет мобільного застосунку

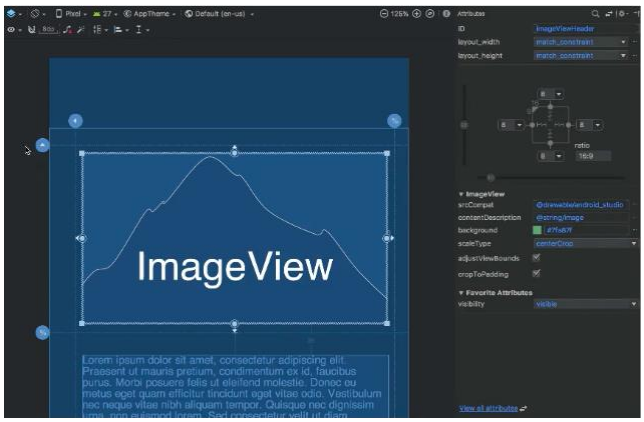


Рисунок 3.2 – Приклад редагування ImageView на макеті

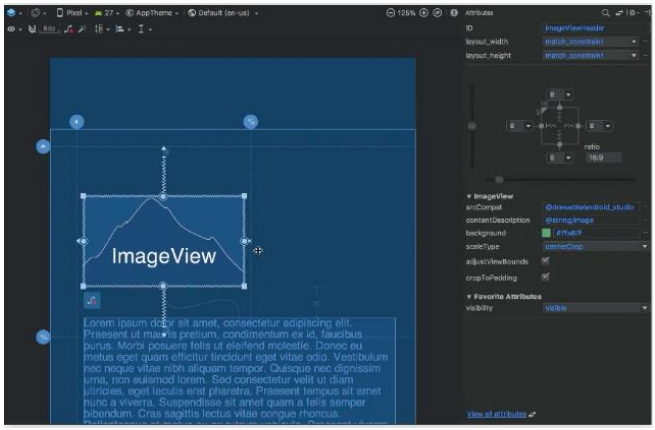


Рисунок 3.3 – Приклад редагування ImageView на макеті

За допомогою APK Analyzer можно виконати наступне:

– переглянути абсолютний та відносний розмір файлів у APK, такі як DEX-файли та файли ресурсів Android;

– зрозуміти склад DEX-файлів;

– мати швидкий доступ до остаточних версій файлів в APK, таких як AndroidManifest.xml;

– порівнювати 2 файли APK на одному екрані.

– Android Emulator – імітує різні пристрої під керуванням Android(Android смартфони, планшети, Wear OS, Android TV) на комп’ютері. Поставляється з попередньо визначеними конфігураціями для популярних типів пристроїв та може передавати дані швидше, ніж пристрій, підключений до комп’ютеру через USB. Android Emulator надає майже всі можливості та функції реального пристрою Android. На Android Emulator можно імітувати вхідні дзвінки, текстові повідомлення, вказувати геолокацію пристрою, тестувати підключення до інтернет-мережі та користуватися функціями зміни орієнтації екрану, камери, мати доступ до магазину Android застосунків – Google Play Store.

– гнучка система подубови застосунків – використовуючи Gradle система збірки в Android Studio дозволяє кастомізувати побудову проекту, шоб генерувати декілька варіантів побудови для різних типів пристроїв для одного проекту.

– Android Profiler – надає дані у реальному часі про використання пам’яті, CPU, та активності у мережі пристрою, на якому запущений застосунок (рис. 3.4). Щоб показати розширені дані про застосунок під час запуску, на пристроях з ОС Android 7.1 або новішої версії, Android Studio має ввести логіку моніторингу у програму яка була скомпільована. Розширений моніторинг застосунку містить хронологію подій всіх вікон, кількість виділених об'єктів у Memory Profiler, події збору сміття у Memory Profiler та інформацію про всі передані файли у Network Profiler.

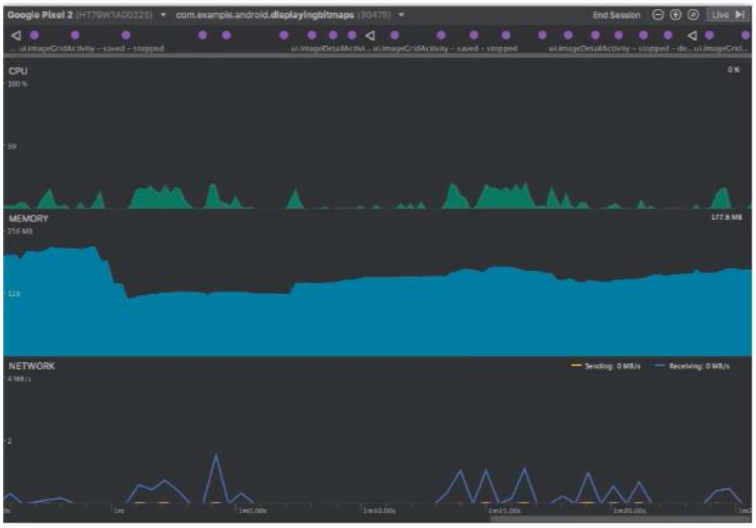


Рисунок 3.4 – Android Profiler

3.2 Мова програмування для розробки застосунку

Java – високорівнева мова програмування загального призначення, була спеціально розроблена щоб бути як можна менше залежною від реалізації. Головною ідеєю Java при створенні була – “write once, run anywhere”, що означає, що скомпільований код Java може працювати на всіх платформах, підтримуючих Java, без необхідності перекомпіляції. Програми, які написані на Java, як правило, збираються до байт-коду, який може бути запущений на будь-якій віртуальній машині Java (JVM) незалежно від архітектури комп’ютера.

Також треба відзначити, що для розробки Android застосунків необхідно використовувати Android SDK – інструмент розробника, який містить в собі комплексний набір інструментів розробки, включаючи дебагер, API бібліотеки, емулятор, документацію, приклади коду та рекомендації по використанню.

Мова Java є ключовим стовпом в Android - мобільній операційній системі з відкритим кодом. Хоча Android, побудован на ядрі Linux, написан переважно в C, Android SDK використовує мову Java як основу для Android застосунків. Мова байт-коду, підтримувана Android SDK, несумісна з байт-кодом Java і працює на власній віртуальній машині, оптимізованій для

пристроїв з низькою пам'яттю, таких як смартфони та планшетні комп'ютери. Залежно від версії Android, байт-код або інтерпретується віртуальною машиною Dalvik або компілюється у Android Runtime.

3.3 Програмна реалізація

Для програмної реалізації мобільного застосунку окрім стандартних засобів та бібліотек Java було використано бібліотеку ZXing [5, 6].

ZXing – бібліотека з відкритим кодом для обробки одновимірних та двовимірних штрихових кодів, написана на Java та була портована на інші мови програмування.

ZXing було обрано за декількох причин:

– ZXing використовується Google для деяких програм та для пошуку книг, що підкреслює якість бібліотеки;

– ZXing є бібліотекою з відкритим кодом, має активні сторінки для вирішення проблем та команда розробників продовжує покращувати якість та швидкість роботи бібліотеки своєчасними оновленнями бібліотеки;

– незважаючи на існування бібліотек які мають функції для обробки двовимірних штрихових кодів схожі на ZXing, ZXing вважається кращою тому що для роботи вона не потребує постійного підключення до інтернет мережі, не потребує сторонніх сервісів та не потребує встановлення додаткових застосунків. ZXing є цілком незалежною від інших сервісів, що робить її кращою альтернативою на ринку.

Також розробники ZXing адаптували роботу з математичними алгоритмами з полями Галуа та кодами Ріда-Соломона у бібліотеці, що робить бібліотеку більш комфортною у використанні, так як в ній присутні класи для виконування деяких математичних обчислень, необхідних для роботи з QR-кодами.

Далі будуть описані кроки реалізації функцій генерування та розпізнавання QR-кодів, які були використані для розробки даного мобільного застосунку. Описані кроки математично розглянуто у розділі 2.

3.3.1 Реалізація процесу розпізнавання QR-кодів

Для процесу розпізнавання QR-кодів використовується два головних класи - ReadActivity та ScannerActivity. ReadActivity забезпечує запуск процесу сканування, викликом відповідного activity, у даному випадку – ScannerActivity, та забезпечує обробку результату, отриманого після завершення сканування. ScannerActivity виконує сканування QR-коду, та результат відправляє до ReadActivity.

Для початку сканування потрібно ініціалізувати View, який відповідає за спеціальне прев'ю камери, де реалізовано функції визначення та розпізнавання QR-коду на зображенні. Після успішної ініціалізації камери та прев'ю для визначення чи є QR-код присутній на зображенні, починається процес сканування інформації. У процесі сканування встановлюються параметри про набір символів, закодованих у QR-код, інформацію про формат та додаткова інформація, яка містить в собі допоміжні дані які можуть прискорити процес сканування та декодування QR-коду (рис. 3.5).

Після того як необхідні параметри для декодування були встановлені, створюється декодер (рис. 3.6), у ньому відокремлюються білі та чорні модулі QR-коду для декодування та проводяться маніпуляції для отримання додаткової інформації про QR-код, щоб прискорити декодування.

Наступним кроком відбувається встановлення рівню корекції QR-коду, його версії. Із даних, встановлених на попередньому кроці, окремо виділяються кодові слова та окремо виділяються блоки з даними.

```
public Decoder createDecoder(Map<DecodeHintType, ?> baseHints) {
    Map<DecodeHintType, Object> hints = new EnumMap<>(DecodeHintType.class);

    hints.putAll(baseHints);

    if(this.hints != null) {
        hints.putAll(this.hints);
    }

    if(this.decodeFormats != null) {
        hints.put(DecodeHintType.POSSIBLE_FORMATS, decodeFormats);
    }

    if (characterSet != null) {
        hints.put(DecodeHintType.CHARACTER_SET, characterSet);
    }

    MultiFormatReader reader = new MultiFormatReader();
    reader.setHints(hints);

    return inverted ? new InvertedDecoder(reader) : new Decoder(reader);
```

Рисунок 3.5 – Створення декодеру для розпізвання QR-коду

```
public DecoderResult decode(BitMatrix bits, Map<DecodeHintType,?> hints)
        throws FormatException, ChecksumException {

    // Construct a parser and read version, error-correction level
    BitMatrixParser parser = new BitMatrixParser(bits);
        // Revert the bit matrix
        parser.remask();

        // Will be attempting a mirrored reading of the version and format info.
        parser.setMirror(true);

        // Preemptively read the version.
        parser.readVersion();

        // Preemptively read the format information.
        parser.readFormatInformation();

        // Prepare for a mirrored reading.
        parser.mirror();

        DecoderResult result = decode(parser, hints);

        // Success! Notify the caller that the code was mirrored.
        result.setOther(new QRCodeDecoderMetaData( mirrored: true));

        return result;
```

Рисунок 3.6 – Декодер для отримання білих та чорних модулів з даними із матриці QR-коду

Інформація з кодовими словами, рівнем корекції та версією QR-коду зберігається у окремому блоку зі службовими даними. Підраховується кількість блоків з закодованими даними та ці блоки об’єднуються з блоками службової інформації у один спільний потік байт.

```
private DecoderResult decode(BitMatrixParser parser, Map<DecodeHintType,?> hints)
        throws FormatException, ChecksumException {
    Version version = parser.readVersion();
    ErrorCorrectionLevel ecLevel = parser.readFormatInformation().getErrorCorrectionLevel();

    // Read codewords
    byte[] codewords = parser.readCodewords();
    // Separate into data blocks
    DataBlock[] dataBlocks = DataBlock.getDataBlocks(codewords, version, ecLevel);

    // Count total number of data bytes
    int totalBytes = 0;
    for (DataBlock dataBlock : dataBlocks) {
        totalBytes += dataBlock.getNumDataCodewords();
    }
    byte[] resultBytes = new byte[totalBytes];
    int resultOffset = 0;

    // Error-correct and copy data blocks together into a stream of bytes
    for (DataBlock dataBlock : dataBlocks) {
        byte[] codewordBytes = dataBlock.getCodewords();
        int numDataCodewords = dataBlock.getNumDataCodewords();
        correctErrors(codewordBytes, numDataCodewords);
        for (int i = 0; i < numDataCodewords; i++) {
            resultBytes[resultOffset++] = codewordBytes[i];
        }
    }

    // Decode the contents of that stream of bytes
    return DecodedBitStreamParser.decode(resultBytes, version, ecLevel, hints);
```

Рисунок 3.7 – Розділення закодованих даних та службової інформації, яка містить дані для декодування, для виконання процесу декодування інформації

У функцію декодування інформації у тектовий вигляд передаються всі параметри, встановлені на попередніх кроках (рис. 3.8).

```
static DecoderResult decode(byte[] bytes,
                            Version version,
                            ErrorCorrectionLevel ecLevel,
                            Map<DecodeHintType,?> hints) throws FormatException {
```

Рисунок 3.8 – Функція перетворення закодованої інформації у текстовий вигляд

Як параметри ця функція приймає байти з закодованою інформацією, встановлену інформацію про версію самого QR-коду, рівень корекції та додаткову інформацію, яка може впливати на процес декодування.

І у класі DecoderResult, на основі типу символів, закодованих у QR-код, відбувається декодування (рис. 3.9).

```
private static void decodeNumericSegment(BitSource bits,
                                         StringBuilder result,
                                         int count) throws FormatException {
  // Read three digits at a time
  while (count >= 3) {
    // Each 10 bits encodes three digits
    if (bits.available() < 10) {
      throw FormatException.getFormatInstance();
    }
    int threeDigitsBits = bits.readBits( numBits: 10);
    if (threeDigitsBits >= 1000) {
      throw FormatException.getFormatInstance();
    }
    result.append(toAlphaNumericChar( value: threeDigitsBits / 100));
    result.append(toAlphaNumericChar( value: (threeDigitsBits / 10) % 10));
    result.append(toAlphaNumericChar( value: threeDigitsBits % 10));
    count -= 3;
  }
  if (count == 2) {
    // Two digits left over to read, encoded in 7 bits
    if (bits.available() < 7) {
      throw FormatException.getFormatInstance();
    }
    int twoDigitsBits = bits.readBits( numBits: 7);
    if (twoDigitsBits >= 100) {
      throw FormatException.getFormatInstance();
    }
    result.append(toAlphaNumericChar( value: twoDigitsBits / 10));
    result.append(toAlphaNumericChar( value: twoDigitsBits % 10));
  } else if (count == 1) {
    // One digit left over to read
    if (bits.available() < 4) {
      throw FormatException.getFormatInstance();
    }
    int digitBits = bits.readBits( numBits: 4);
    if (digitBits >= 10) {
      throw FormatException.getFormatInstance();
    }
    result.append(toAlphaNumericChar(digitBits));
  }
}
```

Рисунок 3.9 – Декодування алфавітно-цифрової інформації

Після вдалого декодування інформації у рядок робиться висновок про формат цієї інформації – чи ця інформація містить номер телефону, електронну пошту, смс, контактну інформацію. Розкодована інформація далі оброблюється відповідно до її типу (рис. 3.10, рис 3.11)

```
case ADDRESSBOOK:
    Log.d(ResultTAG,  msg: "ADDRESSBOOK: " + parserdResult.getDisplayResult());

    resultHandlerClass.addMeCard(parserdResult);
```

Рисунок 3.10 – Виклик методу який обробляє інформацію що зберігає дані про контактну інформацію

```
final void addContact(String[] names, String[] phoneNumbers, String[] emails,
                      String address, String org, String title) {

    intent = new Intent(ContactsContract.Intents.Insert.ACTION, Contacts.People.CONTENT_URI);
    putExtra(intent, ContactsContract.Intents.Insert.NAME, names != null ? names[0] : null);

    linearLayout.addView(createATextView(LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT, RelativeLayout.ALIGN_PARENT_RIGHT,
            names != null ? names[0] : null, fontSize: 20, textStyle: 1, margin: 10, padding: 20));



    int phoneCount = Math.min((phoneNumbers != null) ? phoneNumbers.length : 0,
            PHONE_KEYS.length);
    for (int x = 0; x < phoneCount; x++) {
        putExtra(intent, PHONE_KEYS[x], phoneNumbers[x]);

        linearLayout.addView(createATextView(LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT, RelativeLayout.ALIGN_PARENT_RIGHT,
                phoneNumbers[x], fontSize: 20, textStyle: 0, margin: 10, padding: 20));

    }

    int emailCount = Math.min((emails != null) ? emails.length : 0, EMAIL_KEYS.length);
    for (int x = 0; x < emailCount; x++) {
        putExtra(intent, EMAIL_KEYS[x], emails[x]);
        //emailMecard.setText(emails[x]);

        linearLayout.addView(createATextView(LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT, RelativeLayout.ALIGN_PARENT_RIGHT,
                emails[x], fontSize: 20, textStyle: 0, margin: 10, padding: 20));

    }
```

Рисунок 3.11 – Метод для обробки контактної інформації

3.3.2 Реалізація процесу генерування QR-кодів

QR-код підтримує декілька форматів даних, які можуть бути закодовані. У зв’язку з цим було розроблено окремі методи для перетворення інформації, яка була введена користувачем, у формат, згідно з форматом даних для генерування. Було реалізовано генерування QR-кодів для електронної пошти, відправлення смс, контактної інформації, здійснення телефонних дзвінків, тексту, підключення до WIFI, посилання на інтернет ресурс.

Для прикладу розглянемо процес генерації QR-коду для контактної інформації – MeCard. Кожен тип інформації для кодування у QR-код має окреме activity, для контактної інформації це клас MeCardActivity.

MeCardActivity має декілька полів для заповнення користувачем. Після введення бажаної інформації користувачем, введена інформація записується до відповідних рядків та передається у функцію, яка перетворить рядки на формат, відповідний до контактної інформації. Після перетворення рядків з

даними до текстової інформації відповідно до вимог формату MeCard, фінальний рядок з даними передається у функцію, яка відповідає за кодування інформації у QR-код (рис. 3.12).

```
public void createBarcode (String qrToEncode) throws WriterException {
    Map<EncodeHintType, Object> hints = new EnumMap<>(EncodeHintType.class);

   com.google.zxing.common.CharacterSetECI eci = CharacterSetECI.UTF8;
   hints.put(EncodeHintType.CHARACTER_SET, eci);

    hints.put(EncodeHintType.ERROR_CORRECTION, ErrorCorrectionLevel.H);

    BitMatrix bMatrix = null;

    MultiFormatWriter multiFormatWriter = new MultiFormatWriter();

    bMatrix = multiFormatWriter.encode(qrToEncode, BarcodeFormat.QR_CODE,  width: 200,  height: 200, hints);

    Bitmap imageBitmap = Bitmap.createBitmap( width: 200,  height: 200, Bitmap.Config.RGB_565);
    for (int y = 0; y < 200; y++){
        for (int x = 0; x < 200; x++){
            imageBitmap.setPixel(x, y, bMatrix.get(x,y) ? BLACK : WHITE);
        }
    }
```

Рисунок 3.12 – функція для генерування QR-коду

Основним класом для створення QR-коду із введеної інформації є клас MultiFormatWriter. Використовується метод encode, який отримує як параметри на вході інформацію про вхідні дані, які треба закодувати, формат, ширину і висоту бажаного QR-коду, та додаткові параметри, які треба враховувати при генерації.

У процесі генерації на основі вхідних даних та додаткової інформації встановлюється режим кодування символів (рис. 3.13).

```
String encoding = DEFAULT_BYTE_MODE_ENCODING;
boolean hasEncodingHint = hints != null && hints.containsKey(EncodeHintType.CHARACTER_SET);
if (hasEncodingHint) {
  encoding = hints.get(EncodeHintType.CHARACTER_SET).toString();
}

// Pick an encoding mode appropriate for the content. Note that this will not attempt to use
// multiple modes / segments even if that were more efficient. Twould be nice.
Mode mode = chooseMode(content, encoding);
```

Рисунок 3.13 – Встановлення режиму кодування символів

Окремо кодується основна інформація та служебна інформація. На основі розміру вхідних даних підраховується необхідна кількість бітів у QR-коді та встановлюється його розмір (рис. 3.14).

```
BitArray dataBits = new BitArray();
appendBytes(content, mode, dataBits, encoding);

Version version;
if (hints != null && hints.containsKey(EncodeHintType.QR_VERSION)) {
  int versionNumber = Integer.parseInt(hints.get(EncodeHintType.QR_VERSION).toString());
  version = Version.getVersionForNumber(versionNumber);
  int bitsNeeded = calculateBitsNeeded(mode, headerBits, dataBits, version);
  if (!willFit(bitsNeeded, version, ecLevel)) {
    throw new WriterException("Data too big for requested version");
  }
} else {
  version = recommendVersion(ecLevel, mode, headerBits, dataBits);
}
```

Рисунок 3.14 - Відділення основної інформації та розрахування версії QR-коду

В останню чергу об’єднуються у єдиний масив біти з інформацією та біти зі служебною інформацією QR-коду. Розраховується кількість блоків корекції. Блоки корекції об’єднуються з масивом закодованої інформації (рис. 3.15).

```
Version.ECBlocks ecBlocks = version.getECBlocksForLevel(ecLevel);
int numDataBytes = version.getTotalCodewords() - ecBlocks.getTotalECCodewords();

// Terminate the bits properly.
terminateBits(numDataBytes, headerAndDataBits);

// Interleave data bits with error correction code.
BitArray finalBits = interleaveWithECBytes(headerAndDataBits,
                                           version.getTotalCodewords(),
                                           numDataBytes,
                                           ecBlocks.getNumBlocks());
```

Рисунок 3.15 – Генерування блоків корекції та об’єднання блоків корекції з закодованою інформацією

В останню чергу до масиву, який містить закодовану інформацію, службеу інформацію та інформацію з блокамі корекції, додається шаблон маски (рис. 3.16).

```
int dimension = version.getDimensionForVersion();
ByteMatrix matrix = new ByteMatrix(dimension, dimension);
int maskPattern = chooseMaskPattern(finalBits, ecLevel, version, matrix);
qrCode.setMaskPattern(maskPattern);

// Build the matrix and set it to "qrCode".
MatrixUtil.buildMatrix(finalBits, ecLevel, version, maskPattern, matrix);
qrCode.setMatrix(matrix);
```

Рисунок 3.16 – Розрахування маски QR-коду та застосування її до блоків з закодованою інформацією

Після успішного кодування інформації, за допомогою методу Bitmap.createBitmap(int width, int height, Bitmap.Config config), який повертає змінний растровий малюнок із зазначеною шириною та висотою. Його початкова щільність відповідна до getDensity(). Створений растровий малюнок знаходиться в колірному просторі sRGB. Та методу Bitmap.setPixel(int x, int y, int color), який використовує вказаний колір у растровому малюнку (припускаючи, що він є змінним) у координатах x, y. Колір має бути неперебільшеною величиною ARGB у колірному просторі sRGB. Отриманий Bitmap і є згенерованим QR-кодом, який користувач побачить на екрані.

3.4 Інструкція користувача

При запуску мобільного застосунку користувач бачить екран, який наведено на рисунку 3.17, на якому прямокутником відмічена область куди слід помістити зображення QR-коду для початку сканування.

Після вдалого процесу сканування QR-коду користувач побачить екран з результатом сканування. На рисунках 3.18 та 3.19 наведено приклад

відображення результату сканування QR-коду, який містить інформацію з електронною поштою.

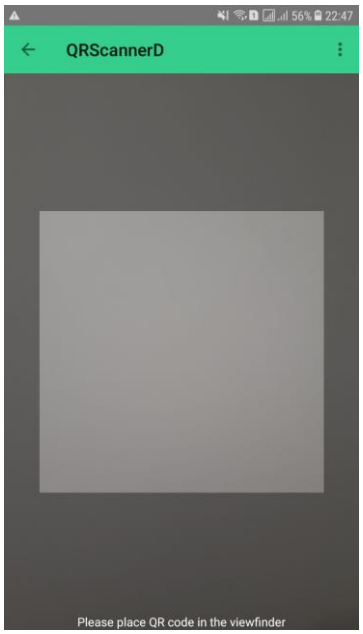


Рисунок 3.17 – Початковий екран сканування, який користувач бачить при запуску мобільного застосунку

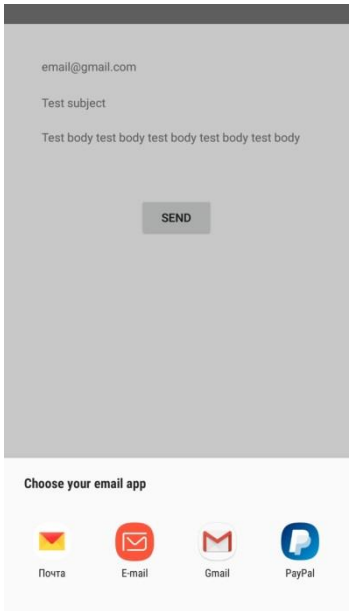


Рисунок 3.18 – Надання можливості користувачу обрати зручний застосунок електронної пошти

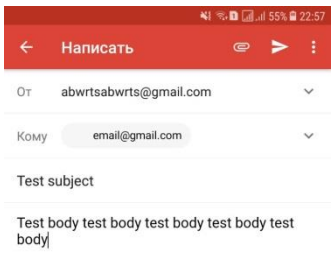


Рисунок 3.19 – Відображення відсканованної інформації

Для того щоб згенерувати QR-код користувачу потрібно обрати відповідний пункт меню та у меню обрати тип інформації, яку він бажає закодувати. Це продемонстровано на рисунках 3.20, 3.21.

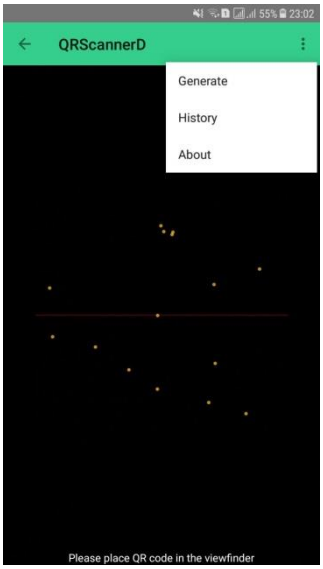


Рисунок 3.20 – Меню для вибору опції генерування

Після того як користувач обрав тип інформації, натиснувши на відповідну іконку, йому буде запропоновано ввести дані для кодування.

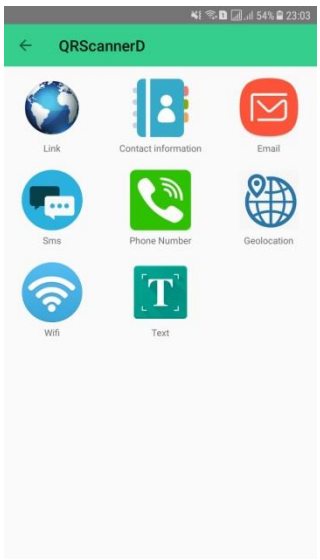


Рисунок 3.21 – Меню для вибору типу інформації яку користувач бажає закодувати

На рисунку 3.22 продемонстровано введення інформації для кодування електронної пошти.

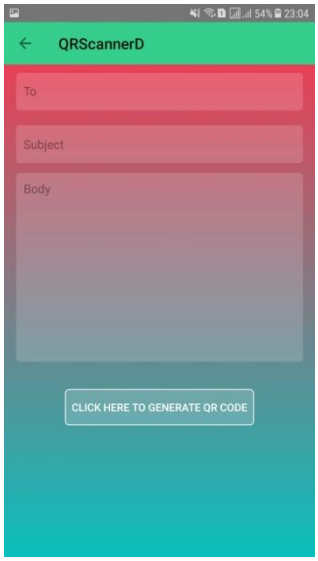


Рисунок 3.22 – Кодування електронної пошти

Після вдалого процесу кодування інформації користувач побачить згенерований QR-код, наведений на рисунку 3.23.

Рисунок 3.23 – Згенерований QR-код

Натиснувши на кнопку “Share” користувач отримає можливість відправити згенерований код через любий месенджер, клієнт електронної пошти або застосунок для передачі файлів, встановлений на його мобільному пристрої (рис. 3.24).

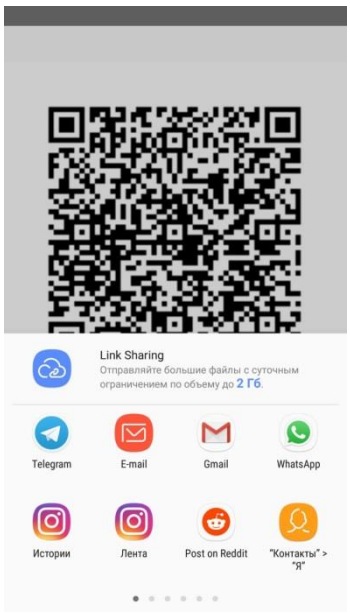


Рисунок 3.24 – Користувачу запропоновано відправити згенерований QR-код

Натиснувши на опцію меню “History” (рис. 3.20), користувач побачить історію відсканованих QR-кодів (рис. 3.25).

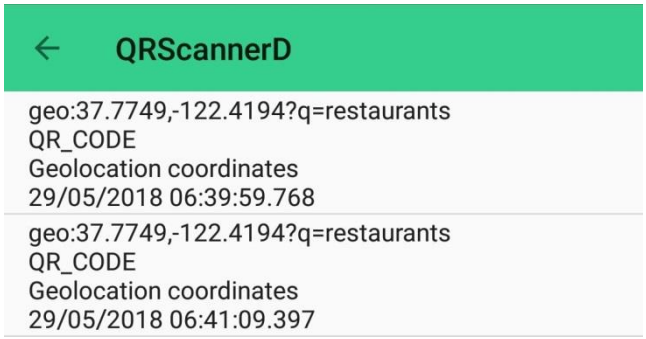


Рисунок 3.24 – Історія відсканованих QR-кодів

4 ОХОРОНА ПРАЦІ

4.1 Аналіз потенційно небезпечних і шкідливих виробничих факторів , що впливають на персонал

Персональні ЕОМ типу IBM PC AT має наступні характеристики:

– споживана потужність 350 Вт;

– робоча напруга 220 В;

– напруга джерел живлення +12 В, -12 В, 5 В;

– робоча частота 50 Гц.

Виходячи з приведених характеристик, очевидно, що для користувача існує небезпека поразки електричним струмом у разі недбалого поводження з комп'ютером і порушення правил експлуатації (невиконання огляду відкритих частин ПЕВМ, що знаходяться під напругою або знятих для ремонту вузлів і т. д.).

Джерелами підвищеної небезпеки можуть служити наступні елементи:

розподільний щит;

джерела живлення;

У відповідності з [14] до легкої фізичної роботи відносяться всі види діяльності, вироблювані сидячи і не вимагаючи фізичної напруги. Робота користувача розробленого пакету програм відноситься до категорії Іа.

Згідно з [13] приміщення для ПЕОМ по ступеню небезпеки поразки людини електричним струмом відноситься до приміщень без підвищеної небезпеки (немає струмопровідної полови, вогкості, підвищеної температури, можливості одночасного дотику до корпусів устаткування з “землею” і до струмонесучих частин).

У відповідності з [12] при обслуговуванні ПЕВМ мають місце фізичні і психофізичні небезпечні, а також шкідливі виробничі чинники:

– підвищене значення напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини;

– підвищений рівень статичної електрики;

– підвищений рівень електромагнітних випромінювань;
– підвищена або знижена температура повітря робочої зони;
– підвищена або знижена рухливість повітря;
– підвищена або знижена вогкість повітря;
– відсутність або недолік природного світла;
– підвищена пульсація світлового потоку;
– недостатня освітленість робочого місця;
– підвищений рівень шуму на робочому місці;
– розумове перенапруження;
– емоційні навантаження;
– монотонність праці.

Щодо до впливу на довкілля, то програмний засіб, який було розлоблено під час дипломного прокету на довкілля ніяк не впливає.

Діяльність за темою магістерської роботи в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища», Законом України «Про забезпечення санітарного та епідемічного благополуччя населення», Законом України «Про відходи», Законом України «Про охорону атмосферного повітря», Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», Водний кодекс України.

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на відалення (знешкодження), утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці не оказує, бо не має в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

В процесі діяльності виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

– Батарейки та акумулятори (малі) - III клас небезпеки

– Макулатура - IV клас небезпеки

– Матеріали пакувальні, що не вміщують целюлозу - IV клас небезпеки

– Матеріали пакувальні, що вміщують п/ет, п/пр - IV клас небезпеки

– Змінні носії інформації - IV клас небезпеки

Наводяться вимоги зберігання виявлених за своєю роботою відходів відповідно до вимог Державних санітарних правил і норм ДСанПіН 2.2.7.029.

Відходи в міру їх накопичення збирають у тару, відповідну класу небезпеки, з дотриманням правил безпеки, після чого доставляють до місця тимчасового зберігання відходів відповідно до затвердженої схеми їх розміщення Зазначені для зберігання відходів місця чи об'єкти повинні використовуватися лише для заявлених відходів.

Не допускається зберігання відходів у невстановлених схемою місцях, а також перевищення норм тимчасового зберігання відходів.

Способи тимчасового зберігання відходів визначаються видом, агрегатним станом і класом небезпеки відходів:

– - Відходи III класу небезпеки зберігаються в тарі, яка забезпечує локалізацію зберігання, дозволяє виконувати вантажно-розвантажувальні і транспортні роботи і виключає поширення в ОС шкідливих речовин;

– - Відходи IV класу небезпеки можуть зберігатися відкрито на промисловому майданчику у вигляді конусоподібної купи, звідки їх автонавантажувачем перевантажують у самоскид і доставляють на місце утилізації або захоронення.

В разі часового зберігання відходів у стаціонарних складах або промислових приміщеннях повинні бути забезпечені санітарно-гігієнічними етичні вимоги до повітря робочої зони згідно з ДСН 3.3.6.042-99.

Не допускається змішування відходів різних видів і класів небезпеки з будівельними і побутовими відходами, відходами дерев'яної, металевої,

синтетичної тари, відходами текстильних матеріалів (старий спецодяг, ганчірки) і інш.

Проведення заготовки, здачі, переробки та реалізації металобрухту встановлені окремо Законом України «Про металобрухт».

Всі відходи, що утворюються в процесі діяльності/роботи, підлягають обліку.

Вимоги безпеки при поводженні з відходами:

Під час роботи з відходами (прибирання виробничих приміщень, збір і сортування, навантаження, транспортування, розвантаження та ін.) працівники та обслуговуючий персонал підприємства повинні бути забезпечені засобами індивідуального захисту та дотримуватися вимог інструкцій з охорони праці, що діють на підприємстві.

Наведено перелік деяких відходів, які передаються на утилізацію організаціям, які мають ліцензію на поводження з відходами як вторинної сировини:

- лом і кускові відходи міді, бронзи, латуні, алюмінію, свинцю;
- брухт чорних металів;
- макулатура;
- склобій;
- матеріали текстильні вторинні;
- відходи деревини кускові
- відпрацьовані фільтрувальні засоби індивідуального захисту
- відпрацьовані вогнегасники
- матеріали пакувальні вторинні

Відвантаження таких відходів здійснюється відповідно до договору (контракту).

Побутові та будівельні відходи вивозяться на полігон твердих побутових відходів міста, також відповідно до договору з комунальним дорожньо-експлуатаційним управлінням.

Особи, винні в порушенні встановленого порядку поводження з відходами (порушення правил обліку відходів, самовільне складування і видалення відходів, передача відходів в інші підприємства/організації з порушенням встановлених правил), згідно законодавства несуть дисциплінарну, адміністративну або кримінальну відповідальність.

4.2 Заходи щодо техніки безпеки

Основним небезпечним чинником при роботі з ЕОМ є небезпека поразки людини електричним струмом, яка усугубляє тим, що органи чуття людини не можуть на відстані знайти наявності електричної напруги на устаткуванні.

Проходячи через тіло людини, електричний струм надає на нього складну дію, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон і інших органів тканин організму) дій.

Ступінь ураження людини електричним струмом залежить від наступних факторів:

– значення сили струму;

– електричного опору тіла людини і тривалості протікання через нього струму;

– роду і частоти струму;

– індивідуальних властивостей людини і навколишнього середовища.

Даним проектом передбачаються наступні технічні способи і засоби, застережливі поразки людини електричним струмом:

– заземлення електроустановок;

– занулення;

– захисне відключення;

– електричне розділення сітей;

– використовування малої напруги;

– ізоляція струмоведучих частин;

– огорожа електроустановок.

Проведемо розрахунок заземлюючого пристрою.

Початкові дані для розрахунку заземлюючого пристрою:

– напруга установки, що заземляється, - 220В;

– режим нейтралу мережі - з ізольованою нейтралою;

– питомий опір ґрунту – 100 Ом·м(суглинок);

– гранично допустимий опір заземлюючого пристрою - 4 Ом;

– характеристика кліматичної зони (III):

а) середня багаторічна низька температура, ^{o}C - від –14 до -10;

б) тривалість замерзання вод, дні - 150;

в) коефіцієнт сезонності для вертикального електроду завдовжки 3м -1,5.

Визначимо розрахунковий опір ґрунту (Ом·м) по формулі (4.1).

$$\rho_{расч} = \psi \cdot \rho = 1{,}5 \cdot 100 = 150 Ом \cdot м \qquad (4.1)$$

де ρ - питомий опір ґрунту;

ψ_i – кліматичний коефіцієнт, що враховує стан ґрунту під час вимірювань (таблиця 4 [12]).

Розрахуємо опір розтіканню одиночного трубчастого заземлювача по формулі (4.2).

$$R_{з.1} = (\frac{\rho_{расч}}{2 \cdot \pi \cdot l}) \cdot \ln(4 \cdot \frac{l}{d}) \qquad (4.2)$$

де l – довжина заземлювача (l=5м);

d – діаметр труби і стрижня (d=0,05м);

$$R_{з.1} = (\frac{\rho_{расч}}{2 \cdot \pi \cdot l}) \cdot \ln(4 \cdot \frac{l}{d}) = (\frac{150}{2 \cdot 3{,}14 \cdot 5}) \cdot \ln(4 \cdot \frac{5}{0{,}05}) = 28{,}6 \text{ Ом}$$

Розрахуємо кількість паралельно сполучених одиночних заземлювачей по формулі (5.3).

$$n = \frac{R_{з.1}}{R_{доп} \cdot \eta} = \frac{28{,}6}{4 \cdot 0.47} = 15{,}2 \qquad (4.3)$$

де Rдоп=4. – самий допустимий опір заземлюючого пристрою;

η - коефіцієнт використовування ґрунтового заземлення (для шістки заземлювачей η=0,47).

Округлятимемо отримане значення у більшу сторону n=[15,2]=16.

Розрахуємо довжину горизонтальної сполучної смуги по формулі (4.4).

$$L = a \cdot (n-1) = 3 \cdot (16-1) = 45м \qquad (4.4)$$

де a – відстань між вертикальними заземлювачами (a=3м);

n – кількість вертикальних заземлювачей (n=16).

Розрахуємо опір сполучної смуги по формулі (4.5).

$$R_n = \frac{\rho_{расч}}{2 \cdot \pi \cdot l} \cdot \ln(\frac{L^2}{d \cdot h}) \qquad (4.5)$$

де d – еквівалентний діаметр смуги шириною l=5 (d=0,05м);

h – глибина заставляння смуги (h=0,8м).

$$R_n = \frac{\rho_{расч}}{2 \cdot \pi \cdot l} \cdot \ln(\frac{L^2}{d \cdot h}) = \frac{150}{2 \cdot 3{,}14 \cdot 5} \cdot \ln(\frac{45^2}{0{,}05 \cdot 0{,}8}) = 51{,}7 \text{ Ом}$$

Розрахуємо результуючий опір заземлюючого електроду з урахуванням

сполучної смуги по формулі (5.6).

$$R_{гр} = \frac{R_{з.1} \cdot R_n}{R_{з.1} \cdot \eta_n + R_n \cdot n \cdot \eta_з} \le R_{доп} \qquad (4.6)$$

де η_n – коефіцієнт використовування сполучної смуги (для 6-і заземлювачей $\eta_{n=0,27}$).

$$R_{гр} = \frac{R_{з.1} \cdot R_n}{R_{з.1} \cdot \eta_n + R_n \cdot n \cdot \eta_з} = \frac{26{,}6 \cdot 51{,}7}{26{,}6 \cdot 0{,}27 + 51{,}7 \cdot 16 \cdot 0{,}47} = 3{,}47 \text{ Ом}$$

$3{,}47 < 4 \Rightarrow$ умова забезпечення електробезпеки персоналу виконується.

Таким чином, остаточна кількість заземлювачей 15 шт.

4.3 Заходи, що забезпечують виробничу санітарію і гігієну праці

Підвищення працездатності людини і збереження його здоров'я забезпечується стабільними метеорологічними умовами.

Мікроклімат виробничих приміщень – це поєднання температури, вогкості і швидкості руху повітря, а також температури навколишніх поверхонь. Значне коливання параметрів мікроклімату приводить до порушення систем кровообігу, нервової і пітовидільної, що може викликати підвищення або пониження температури тіла, слабкість, запаморочення і навіть непритомність.

В приміщенні для виконання робіт операторського типу, пов'язаних з нервово-емоційною напругою, проектом передбачається дотримання наступних нормованих величин параметрів мікроклімату (див. таблицю 4.1).

Таблиця 4.1 - Оптимальні параметри мікроклімату в робочій зоні виробничого приміщення для категорії робіт 1

Період року	Температура, оС	Відносна вогкість %	Швидкість руху повітря, м/с
Холодний	22.24	40.60	0,1
Теплий	23.25	40.60	0,1

Оскільки в приміщенні немає джерел виділення шкідливих речовин, можна використовувати природну вентиляцію. Площа приміщення складає 32 м2 . Для забезпечення прийнятних параметрів мікроклімату в приміщенні з такою площею можна використовувати 1 кондиціонер типу БК-2000.

Спектр випромінювання монітора комп'ютера включає рентгенівську, ультрафіолетову, інфрачервону області, а також широкий діапазон хвиль інших частот. Небезпека рентгенівського проміння нехтує мала, оскільки цей вид випромінювання поглинається речовиною екрану.

Для зниження дії електромагнітного випромінювання пропонується захист часом і відстанню. Захист часом передбачає обмеження часу перебування людини в зоні дії полів. Тривалість роботи на ПЕОМ повинна складати не більше 3.5–4.5 години.

Також необхідно забезпечити раціональне освітлення в робочому приміщенні. В проекті, що розробляється, передбачається використовувати суміщене освітлення. В світлий час доби приміщення освітлюватиметься через віконні отвори, в решту часу використовуватиметься штучне освітлення.

Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла в світильниках загального освітлення, оскільки люмінесцентні лампи володіють високою світловою віддачею до 75 Лам/Вт і більш, тривалим терміном служби до

10000 годин, спектральним складом випромінюваного світла, близьким до сонячного.

Зорова робота оператора ПЕВМ відповідно до [13] відноситься до розряду Va з світловим потоком Фл=3120 кожна. Нормована освітленість на робочому місці (Ен) при загальному освітленні складає 200 лк.

Проведемо розрахунок кількості світильників в робочому приміщенні завдовжки a=6 м, шириною b=3 м, заввишки c=4 м. Формула розрахунку штучного освітлення при горизонтальній робочій поверхні методом світлового потоку (4.7):

$$Фл = \frac{Ен \cdot S \cdot Z \cdot K}{N \cdot U \cdot M} \qquad (4.7)$$

де *Фл* – світловий потік, Лм;

Ен – нормована освітленість;

S – площа підлоги, кв.м;

Z=1.1–1.3 - поправочний коефіцієнт світильника (для стандартних світильників);

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації світильників;

N – число світильників;

U=0.55–0.6 – коефіцієнт використовування, залежний від типу світильника, показника індексу приміщення і др.;

M – число ламп в світильнику.

З формули (5.7) виразимо N і визначимо кількість світильників для даного приміщення:

$$N = \frac{Ен \cdot S \cdot Z \cdot K}{Фл \cdot U \cdot M}$$

$$N = \frac{200 \cdot 18 \cdot 1{,}2 \cdot 1{,}5}{3120 \cdot 0{,}6 \cdot 2} = 1{,}7 \qquad (4.7)$$

Виходячи з цього, рекомендується використовувати 2 світильники. Світильники слід розміщувати рядами, бажано паралельно стіні з вікнами. Схема розташування світильників зображена на рис. (4.1).

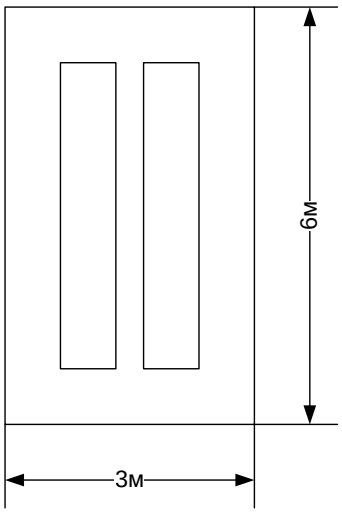


Рисунок 4.1 – Схема розташування світильників

Зниження шуму можна добитися раціонально розпланувавши приміщення, установкою устаткування на спеціальні амортизуючи прокладки. Згідно вимогам [12] рівні звуку не повинні перевищувати 50 дБ.

Для зниження стомлюваності обслуговуючого персоналу в приміщеннях, де розташовані обчислювальні засоби передбачаються використовувати спокійні колірні поєднання і покриття, що не дають відблисків. Від електромагнітного випромінювання, витікаючого від ПЕОМ, використовуються захисні екрани.

Для забезпечення чистоти повітря і відповідних мікрокліматичних умов пропонується застосувати приточування-витяжну вентиляцію. Для зменшення дії шкідливих речовин і загазованості для роботи з розплавленими матеріалами робоче місце забезпечується примусовою витяжною вентиляцією. Цей метод забезпечує притоку потрібної кількості свіжого повітря (30 мм^3 /ч на одного працюючого).

Кількість повітря, яка необхідна подавати в приміщення для забезпечення необхідних параметрів повітряного середовища, визначається на підставі кількість тепла, вологи і шкідливих речовин, що поступають в приміщення, а також враховуючи видалення повітря місцевими відсмоктуваннями від устаткування, загальнообмінною вентиляцією.

4.4 Рекомендації по пожежній профілактиці

Пожежі представляють небезпеку для життя людини і зв'язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки, що спричиняє за собою порушення ходу технологічного процесу.

Горючими матеріалами в приміщенні, де розташовані ПЕОМ, є:

– поліамід - матеріал корпусу мікросхеми. Горюча речовина. Температура самозапалення 420 ^{o}C, енергія запалення 2мДж;

– полівінілхлорид - ізоляційний матеріал. Горюча речовина. Температура самозаймання 480 єС, енергія запалення 50мДж;

– склостоліт ДЦ - матеріал друкарської платні. Складногорючий матеріал;

– пластикат кабельний No.489 - матеріал ізоляції кабелю. Складногорючий матеріал. Температура самозаймання 1500 єС;

– плита деревостружкова - будівельний і обробний матеріал, матеріал з якого виготовлені меблі. Складнозапалений матеріал. Показник горючості 1.8;

– папір – довідкова і робоча документація, література. Горючий матеріал. Показник горючості більше 2.1.

Відповідно до [14] приміщення відноситься до категорії В (пожежовибухонебезпечної).

Джерелами запалення можуть бути:

– іскри при замиканні і розмиканні ланцюгів;

– іскри і дуги коротких замикань;

– перегріви від тривалого перевантаження і наявності перехідного опору;

– розряди статичної електрики.

Для того, щоб зупинити реакцію горіння, порушують умови її виникнення і підтримки. Звичайно для гасіння використовуються порушення двох основних умов сталого стану – пониження температури і режим руху газів. Пониження температури може бути досягнутий шляхом введення речовин, які поглинають багато тепла в результаті випаровування і дисоціації (наприклад, вода, порошки).

При повному тому, що згоряє органічних сполук утворюються С, SO, Н Про, N, а при тому, що згоряє неорганічних з'єднань – оксиди. Залежно від температури плавлення і тривалості реакції можуть знаходиться або у вигляді розплавів (Al O, Ti O), або підійматися в повітря у вигляді диму (P O, Na Про, MgO).

Склад продуктів неповного згоряє горючих речовин складений і різноманітний. Це можуть бути горючі речовини:

– Н, З, СН;

– атомарний водень і кисень;

– різні радикали – ВІН, СН .

Продуктами неповного згоряє можуть бути також оксиди азоту, спирти, альдегіди, кетони і високотоксичні з'єднання, наприклад, синильна кислота.

Для захисту персоналу від дій небезпечних і шкідливих чинників пожежі проектом передбачено застосування промислового фільтруючого протигаза з коробкою марки В (жовтий).

До системи запобігання пожежі відносяться: запобігання утворення горючого середовища і освіти в горючому середовищі джерел запалення, забезпечення пожежебеспеки устаткування.

Щоб запобігти пожежі в обчислювальних центрах, проектом пропонується виконання наступних вимог:

– електроживлення ЕОМ має автоматичне блокування відключення електроенергії на випадок перегріву системи, що може бути результатом зупинки системи охолоджування і кондиціонування;

– система вентиляції обчислювальних центрів обладнується блокуючими пристроями, що забезпечують її відключення на випадок пожежі. Система обладнується вогнеперегороджуючими клапанами;

– застосування устаткування, що задовольняє вимогам електростатичної іскробезпеки [15];

– після закінчення роботи, перед закриттям приміщення, всі електроустановки і персональні комп'ютери відключаються від сіті електроживлення;

– в приміщеннях обчислювальних центрів забороняється:

1) влаштовувати електророзетки на основах, що згоряють;

2) використовувати синтетичні доріжки і килими;

3) користуватися побутовими електронагрівальними приладами;

4) захаращувати евакуаційні виходи і проходи;

5) влаштовувати на вікнах глухі грати;

6) залишати без нагляду включену в електромережу апаратуру, що використовується для вимірювань і нагляду.

Для протипожежного захисту проектом пропонується обладнати приміщення площею 18 м2, яке відноситься до категорії В, автоматичною протипожежною сигналізацією із застосуванням датчиків сповіщення РІД-1 (оповіщувач димовий іонізаційний) в кількості 1 штуки і застосовується в первинних засобах пожежегасінні. Площа контрольована оповіщувачем 150 м2.

Крім того, необхідно проводити навчання робочого персоналу правилам пожежної безпеки.

Розрахуємо вірогідність виникнення пожежі у виробничому приміщенні у разі запалювання транзистора:

$$Q = л \cdot T \cdot Ркз/отк \cdot Qвоспл \cdot Рзащ \qquad (5.8)$$

де л – інтенсивність відмов пожежеопасних EPI;

T – час роботи пожежеопасного EPI за оцінюваний інтервал часу;

Ркз/отк - умовна вірогідність виходу EPI в стан короткого замикання при його відмові;

Qвоспл - вірогідність запалювання EPI, що знаходиться в стані короткого замикання;

Рзащ – вірогідність відмови захисту пожежеопасного EPI. Якщо захист відсутній, Рзащ приймається рівній 1.

Вірогідність виникнення пожежі у разі запалювання транзистора:

$$Q = 1 \cdot 10^{-6} \cdot 1 \cdot 10^{-4} \cdot 0.1 \cdot 1 \cdot 10^{-4} = 1 \cdot 10^{-15}$$

Розрахована вірогідність виникнення пожежі значно менше допустимої, яка складає $1 \cdot 10^{-6}$.

В даному розділі були проаналізовані небезпечні і шкідливі виробничі чинники, що роблять вплив на персонал, розроблені заходи щодо техніки безпеки, заходу, забезпечуючи виробничу санітарію і гігієну праці, а також заходи щодо пожежної профілактики.

ВИСНОВКИ

У рамках дипломної роботи був розроблений і реалізований мобільний застосунок для генерації та розпізнавання QR-кодів. Для цього були вирішені наступні завдання:

– були досліджені методи розпізнавання на зображенні ключових елементів для встановлення чи є QR-код присутній на зображенні;

– була розроблена архітектура мобільного застосунку;

– був розроблений дизайн мобільного застосунку;

– був розроблений web view для мобільного застосунку;

– була розроблена база даних;

– були досліджені методи розпізнавання ключових точок QR-коду на зображенні;

– були досліджені методи декодування даних, які були закодовані у QR-код;

– були досліджені методи обробки інформації в процесі декодування QR-коду;

– були досліджені методи кодування інформації у QR-код;

– були досліджені методи обробки інформаціх у процесі кодування;

– був програмно реалізован метод для розпізнавання QR-коду на зображенні;

– був програмно реалізован метод для декодування інформації з QR-коду;

– був програмно реалізован метод по обробці декодованої інформації;

– був програмно реалізован метод для генерування QR-коду спираючись на дані, введені користувачем;

– був програмно реалізован метод для генерування QR-коду спираючись на тип введених даних;

Не дивлячись на те, що на ринку програмних продуктів існує величезна кількість програм по роботі з QR-кодами, у розробленого в рамках цього дипломного проекту програмного продукту є одна незаперечна перевага - він об'єднує кодування і декодування QR-коду в один додаток. Крім того, даний застосунок дозволяє обробляти контактну інформацію, інформацію з електронною поштою, геолокацію, смс повідомленнями, підключенням до мережі wifi, відкривати посилання на веб-ресурси. Особливістю даного мобільного застосунку є те, що він не залежить від сторонніх сервісів, здійснює процеси сканування та декодування не залежачи від сторонніх додатків і підключення до мережі інтернет.

В рамках даної роботи було реалізовано мобільний додаток, що дозволяє швидко відсканувати код, обробити результат і сгенерувати код з даними, введеними користувачем.

З огляду на все можливості і переваги розробленого мобільного додатка, я впевнена, що він буде зручним і корисним для користувача.

У четвертому розділі було проаналізовано умови праці, виявлені причини травматизму і захворювань, можливі небезпечні й шкідливі виробничі фактори. Також було проведено ряд розрахунків щодо виконання вимог охорони праці в приміщенні відділу програмного забезпечення. Дотримання цих вимог є важливим для збереження працездатності та здоров’я працівників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Шилдт, Г. Java 8. Полное руководство [Текст] / Шилдт Г. – М.: Вильямс-М., 2015. – 1337 с.

2) Phillips, B. Android programming the Big nerd ranch guide [Text]: / B. Phillips, C. Steward, K. Marsicano. – Indianapolis: Pearson technology group, 2017. – 587 p.

3) Bloch, J. Effective Java [Text]: / J. Bloch. – Boston: Addison Wesley, May 2008. – 369 p.

4) Simon, J. Head First Android Development [Text]: / J. Simon. - Sebastopol, California: O'Reilly Media Inc., 2011. – 512 p.

5) Zxing barcode image processing library. [Electronic resourse] – [Access mode]: www/ URL: https://github.com/zxing/zxing

6) Zxing barcode image processing library documentation. [Electronic resourse] – [Access mode]: www/ URL: https://zxing.github.io/zxing/apidocs/

7) Yunhua, G., Weixiang, Z. QR Code Recognition Based On Image Processing [Text] / G. Yunhua, Z. Weixiang // International Conference on Information Science and Technology. – 2011.- 733-736 p.

8) Torawane, C.V., Torawane, A.V. Mobile Barcode for Event Schedule using Android [Text] / C.V Torawane, A.V. Torawane // International Journal of Computer Applications. – 2014. vol 105, no 10. 16 – 20 p.

9) ISO/IEC 18004:2005. Information technology — Automatic identification and data capture techniques — QR Code 2005 bar code symbology specification [Електронний ресурс] - [Режим доступа]: www/ URL: http://www.arscreatio.com/repositorio/images/n_23/SC031-N-1915-18004Text.pdf - 2004 p. - 109 с.

10) Guide to Barcodes vs. QR Codes: In-Depth Comparison and Analysis of Both Label Types. [Electronic resourse]: – [Access mode]: www/ URL: https://www.mpofcinci.com/blog/barcode-vs-qr-code/

11) ДСН 3.3.6.042-99 Державні санітарні норми мікроклімату виробничих. Режим доступу: https://dnaop.com/html/34094/doc-ДСН_3.3.6.042-99

12) ДСН 3.3.6.037-99 Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку. Режим доступу: https://zakon.rada.gov.ua/rada/show/va037282-99

13) ДБН В.2.5-28:2018 Природнє і штучне освітлення. Режим доступу: https://okna.ua/img_all/oknaua/dbn-V-2-5-28-2018-ed.pdf

14) ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. Режим доступу: https://dbn.co.ua/load/normativy/dstu/dstu_b_v_1_1_36/5-1-0-1759

15) ДСТУ Б А.3.2-13:2011 Система стандартів безпеки праці. Будівництво. Електробезпечність. Загальні вимоги (ГОСТ 12.1.013-78, MOD) Режим доступу: http://online.budstandart.com/ru/catalog/doc-page?id_doc=27973

ДОДОТОК А.
ЕЛЕКТРОННІ ПЛАКАТИ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

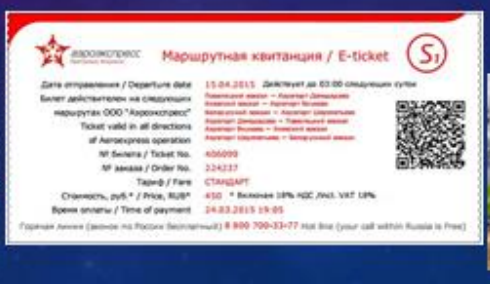
МОБІЛЬНИЙ ДОДАТОК РОБОТИ З QR-КОДАМИ

ТІНАЄВ О.О.

АКТУАЛЬНІСТЬ

Кожен з нас в своєму повсякденному житті стикається з величезним QR-коду. Це й не дивно.

Після представлення QR-коду доступні публіці багато виробників і маркетологи стали використовувати QR-код як засіб швидкого доступу до інформації, так як для отримання необхідної інформації потрібно тільки наявність мобільного пристрою, що має QR-код, і він буде перенаправлений на веб-сайт з усією запропонованою інформації.



QR КОД

QR-код - (код швидкого реагування) це двовимірний штрих код, який спочатку був розроблений для автомобільної промисловості Японії.

QR-код був використаний виробниками автомобілів для відстеження автомобілів і деталей.

QR-коди більш досконалі, ніж одномірний штрих-код, - вони мають більш високу ємність сховища і можуть зберігати різні типи символів. По суті, ці коди схожі на фізичні гіперпосилання, оскільки при їх скануванні користувач переходить на зовнішнє посилання або сайт.

Штрих-код містить тільки інформацію в горизонтальному напрямку, QR-код зберігає інформацію як в горизонтальному, так і у вертикальному напрямках (звідси і назва «2-мірний код»). Через цю структурної різниці QR-код зберігає в сотні разів більше інформації, ніж штрих-код, і має потенціал для зберігання великих обсягів інформації в меншому об'ємі, ніж штрих-код.

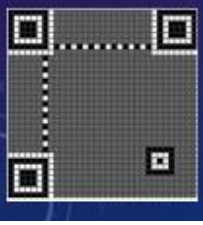
QR КОД

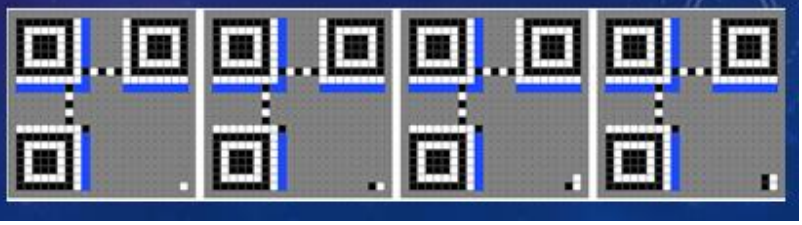
Зміна зовнішнього вигляду і корекція помилок - це, безумовно, основна перевага QR-кодів над традиційними штрих-кодами. QR-коди мають коефіцієнт похибки від 7 до 30%.

Простими словами, навіть якщо упаковка відповідного продукту або надрукованого коду пошкоджена або забруднена, QR-код буде як і раніше працювати. Ця функція активно експлуатується компаніями і підприємствами. Функція виправлення помилок дає можливість поміщати невеликий логотип або зображення в QR-код, щоб зробити його дизайн більш ефективним для бізнесу.

АЛГОРИТМ РОЗПІЗНАВАННЯ

- Встановлюється чи є QR-код на зображенні, шляхом знаходження пошукових візерунків (для кодів версії 2 і більше так же знаходяться вирівнюють візерунки) і перевірки візерунків на відповідність параметрам для QR-коду.
- Якщо було встановлено, що зображення має QR-код, то алгоритм починає зчитувати службову інформацію, яка містить версію QR-коду, його маску і рівень корекції для того щоб встановити подальший алгоритм для зчитування інформації
- Починаючи з версії 7 інформація про формат знаходиться біля правого верхнього пошукового візерунка і зверху лівого нижнього пошукового візерунка. Алгоритм спочатку обробляє правий верхній блок із службовою інформацією. Якщо не вдається її розкодувати, алгоритм обробляє лівий нижній блок зі службовою інформацією.
- Після того як версія, рівень корекції і маска QR-коду були прочитані, починається процес декодування інформації. Для цього до бітам даних застосовується маска, яка була отримана з службової інформації, потім дані зчитуються відповідно до правил заповнення QR-коду, виконується пошук помилкових кодових слів і застосовується до них код корекції, якщо помилки були виявлені, вони виправляються.
- На останньому етапі біти декодируются відповідно до алгоритму типу даних, які були закодовані, і виводиться результат.





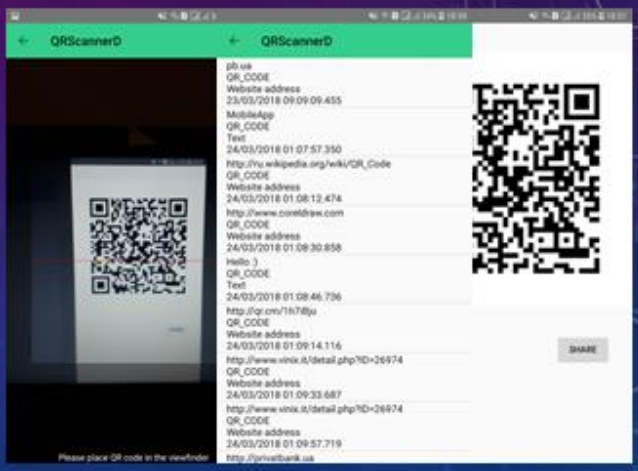
АЛГОРИТМ ГЕНЕРАЦІЇ

- Для кодування інформації першим кроком визначається тип інформації, яка була введена.
- Далі додається інформація про рівень корекції і версії QR-коду, яка визначається виходячи з кількості введених символів.
- Після встановлення необхідної службової інформації та кодування даних, закодована інформація поділяється на блоки, їх кількість залежить від рівня корекції та версії QR-коду. Для закодованої інформації розділеної на блоки розраховуються слова корекції. Далі блоки слів корекції об'єднуються з блоками закодованих даних в єдину послідовність.
- Починається процес заповнення полотна QR-коду послідовністю біт, отриманої після об'єднання даних і слів корекції. Спочатку на полотно наносяться пошукові візерунки, які вирівнюють візерунки, смуги синхронізації і код версії. Далі порожні області полотна розбиваються на стовпчики по 2 модуля. Заповнення починається з правого нижнього кута, рухається в межах стовпчика праворуч-ліворуч, знизу-вгору. У разі, якщо заповнення дісталося до верху стовпця, заповнення відбувається зверху вниз.
- Після нанесення інформації на полотно QR-коду до неї застосовується маска. Застосування маски полягає в зверненні (інвертуванні) квітів бітів оригінального повідомлення відповідно до обраного шаблоном маски.
- Останнім етапом є додавання на полотно інформації про формат і версію QR-коду.

РЕАЛІЗАЦІЯ ANDROID-ДОДАТКУ

Основними функціями android-додатки є:

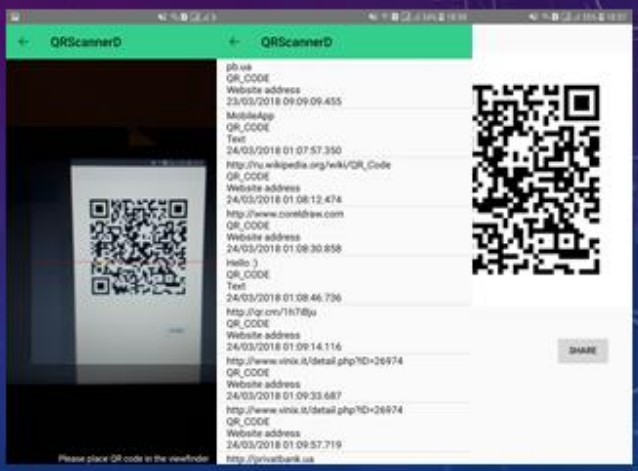
- Сканування QR-кодів
- Генерація QR-кодів
- Збереження результатів сканування
- для подальшого використання

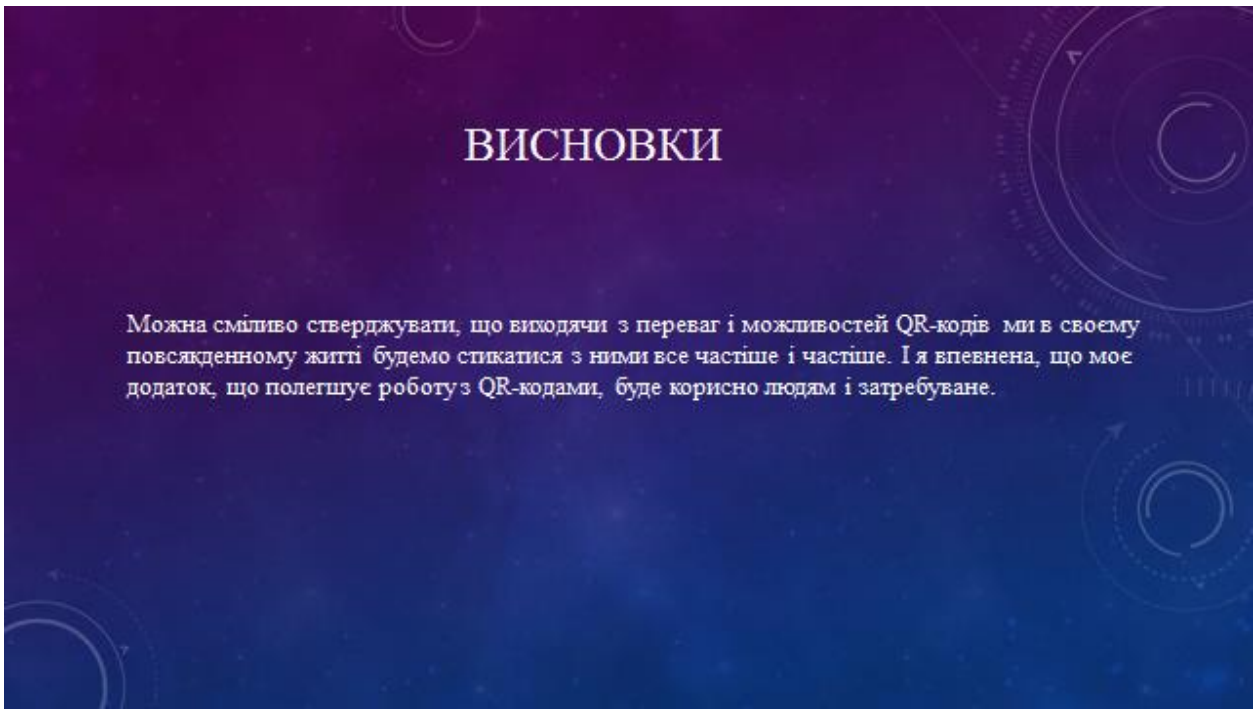


РЕАЛІЗАЦІЯ ANDROID-ДОДАТКУ

Основними функціями android-додатки є:

- Сканування QR-кодів
- Генерація QR-кодів
- Збереження результатів сканування
- для подальшого використання




ВИСНОВКИ
Можна сміливо стверджувати, що виходячи з переваг і можливостей QR-кодів ми в своєму повсякденному житті будемо стикатися з ними все частіше і частіше. І я впевнена, що моє додаток, що полегшує роботу з QR-кодами, буде корисно людям і затребуване.