ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO ĐỒ ÁN

MÔN: NHẬP MÔN THỊ GIÁC MÁY TÍNH

CHỦ ĐỀ: MÃ HÓA QR-CODE VÀ ĐỌC MÃ QR-CODE THÔNG QUA CAMERA TRÊN THIẾT BỊ DI ĐỘNG

Giảng viên hướng dẫn:

TS. Nguyễn Vinh Tiệp

Sinh viên thực hiện:

Hoàng Ngọc Bá Thi - 19522255

Phan Minh Nhật - 19521956

Phạm Minh Trí - 19522390

Trần Anh Khoa - 19521700

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2021

MỤC LỤC

I. Giới thiệu……2

1. Giới thiệu về QR Code……2

2. Giới thiệu về đồ án……...2

II. Lý thuyết QR Code……...3

1. Mã hóa……..3

1.1 Phân tích dữ liệu……3

1.2 Mã hóa dữ liệu……...4

1.3 Tạo mã sửa lỗi……7

1.4 Hoàn thiện đoạn mã thông điệp……12

1.5 Đặt các module vào ma trận……...17

1.6 Chọn mặt nạ……25

1.7 Tạo chuỗi thông tin định dạng và phiên bản……27

2. Giải mã……34

III. Cài đặt và thực nghiệm……34

1. Cài đặt chương trình Python……35

1.1 Chương trình thực hiện việc tạo mã QR-Code……35

1.2 Chương trình thực hiện việc Đọc và giải mã QR-Code thông qua camera……35

2. Thử nghiệm và đánh giá kết quả……..35

IV. So sánh QR Code với Barcode……..43

Tài liệu tham khảo……...……44

I. Giới thiệu

1. Giới thiệu về QR Code

QR (Quick Response) Code là mã ma trận 2D được thiết kế bởi công ty Denso Wave (Nhật Bản) vào năm 1994. QR Code cung cấp khả năng lưu trữ dữ liệu cao, quét nhanh, khả năng đọc từ nhiều hướng và nhiều ưu điểm khác bao gồm sửa lỗi (mã bị hỏng 1 phần vẫn có thể đọc được) và các loại phiên bản khác nhau. Nhiều loại biểu tượng QR Code khác nhau như QR Code biểu trưng, QR Code được mã hóa, iQR Code cũng có sẵn để người dùng có thể lựa chọn trong số chúng theo nhu cầu của họ. Ngày nay, QR Code được áp dụng trong các luồng ứng dụng khác nhau liên quan đến tiếp thị, bảo mật, học thuật,… và trở nên ngày càng phổ biến. Ngày càng có nhiều người nhận thức được công nghệ này và sử dụng nó một cách phù hợp. Sự phổ biến của QR Code phát triển nhanh chóng cùng với sự gia tăng của người dùng điện thoại thông minh và do đó QR Code nhanh chóng đạt được mức độ chấp nhận cao trên toàn thế giới.

2. Giới thiệu về đồ án

QR Code đang dần trở nên phổ biến và đóng vai trò rất quan trọng trong thời đại công nghệ ngày nay. Ví dụ: các đơn vị quảng cáo sử dụng QR Code để lưu thông tin về công ty/sản phẩm của họ, các thông báo tuyển dụng dán ở các biển quảng cáo có in QR Code dẫn đến form đăng ký điện tử, các cửa hàng đã phổ biến hình thức thanh toán qua việc quét QR Code để liên kết tài khoản điện tử của ngân hàng đến hóa đơn điện tử; đặc biệt là thời gian gần đây, QR Code đã và đang đóng vai trò rất quan trọng trong việc giúp con người kiểm soát đại dịch COVID-19 với việc lưu thông tin về tiêm chủng vắc-xin, về lịch trình di chuyển, về lịch sử tiếp xúc,.... Tuy nhiên, việc xử lý QR Code phải cần đến những phần mềm chuyên dụng được cài đặt trên các thiết bị điện tử. Điều này đặt ra cho nhóm của chúng em vấn đề tìm hiểu về QR Code, cách thức nó được tạo ra và đọc QR Code như thế nào.

Đồ án "Mã hóa và Giải mã QR Code" của chúng em hướng đến mục tiêu sau:

- Hiểu về lý thuyết của QR Code bao gồm việc mã hóa và giải mã.
- Cài đặt các chương trình Python thực hiện việc tạo QR Code, đọc và giải mã QR Code thông qua camera trên thiết bị di động.

II. Lý thuyết QR Code

1. Mã hóa

QR Code sẽ mã hóa được chuỗi văn bản thành một mã ma trận 2D bao gồm các pixel trắng và đen như sau:

Hình 1: Ví dụ về output của một văn bản sau khi được mã hóa

Các bước chi tiết để tạo ra được QR Code sẽ được trình bày chi tiết bên dưới.

1.1 Phân tích dữ liệu

QR Code tiêu chuẩn sẽ có 4 chế độ mã hóa: Numeric, Alphanumeric, Byte, and Kanji. Các chế độ đều chuyển văn bản thành chuỗi bits, tuy nhiên mỗi chế độ lại có một phương thức khác nhau. Mỗi phương thức đều được tối ưu để tạo ra được chuỗi bits ngắn nhất.

* QR Code có 4 chế độ mã hóa chính:

- Chế độ Numeric cho các số thập phân từ 0 đến 9.

- Chế độ Alphanumeric cho các số thập phân từ 0 đến 9, các chữ cái in hoa, các ký tự: $, %, *, +, -, .,/,: và cả khoảng trắng.

- Chế độ Byte cho các ký tự trong bộ ký tự ISO-8859-1 **[9]**.

- Chế độ Kanji cho ký tự double-byte từ bộ ký tự Shift JIS **[10]**.

* Chọn chế độ sao cho hiệu quả nhất:

- Nếu dữ liệu chỉ có số từ 0 đến 9 thì chọn chế độ Numeric.

- Nếu không áp dụng được chế độ Numeric và dữ liệu chỉ bao gồm các ký tự trong alphanumeric table **[5]** thì sử dụng chế độ Alphanumeric.

- Nếu các ký tự không có trong alphanumeric table nhưng có thể mã hóa theo ISO-8859-1 thì dùng chế độ Byte.

- Nếu tất cả các ký tự có trong bộ ký tự Shift JIS thì dùng chế độ Kanji.

1.2 Mã hóa dữ liệu

Mã hóa dữ liệu được thiết kế để tạo chuỗi bit ngắn nhất cho cho các ký tự trong văn bản cần được mã hóa.

*** Bước 1**: Chọn mức độ sửa lỗi (Error Correction Level)

- QR Code sử dụng kỹ thuật sửa lỗi Reed-Solomon để sửa lỗi. Quá trình này sẽ tạo ra mã sửa lỗi và các byte sửa lỗi dựa trên dữ liệu cần được mã hóa.

- Các byte sửa lỗi dùng để phát hiện lỗi và mã sửa lỗi để sửa lỗi.

- Có 4 mức độ sửa lỗi: L(khôi phục 7% dữ liệu), M(khôi phục 15% dữ liệu), Q(khôi phục 25% dữ liệu), H(khôi phục 30% dữ liệu).

- Mức độ sửa lỗi càng cao thì kích thước cần để lưu trữ QR Code càng lớn.

*** Bước 2**: Xác định phiên bản nhỏ nhất cho dữ liệu

- Các kích thước khác nhau của QR Code được gọi là các phiên bản. Có 40 phiên bản, phiên bản nhỏ nhất là 21x21, phiên bản lớn nhất là 177x177, mỗi phiên bản sẽ lớn hơn phiên bản trước đó 4 pixels. Ví dụ: Phiên bản 1: 21x21, Phiên bản 2: 25x25, Phiên bản 3: 29x29, ...)

- Mỗi phiên bản sẽ có sức chứa tối đa tùy thuộc vào chế độ đang được sử dụng. Mức độ sửa lỗi cũng sẽ giới hạn lại sức chứa.

- Để xác định phiên bản nhỏ nhất, ta sẽ đếm số lượng ký tự cần được mã hóa, chọn chế độ mã hóa và mức độ sửa lỗi phù hợp. (dựa trên character capacities table **[6]**)

Ví dụ: HELLO WORLD có 11 ký tự, nếu chọn độ sửa lỗi là Q thì phiên bản nhỏ nhất sẽ là 1.

- Phiên bản 40-L (Phiên bản 40, mức độ sửa lỗi L) có sức chứa cao nhất.

*** Bước 3**: Thêm chỉ báo chế độ (Mode Indicator)

- Mỗi chế độ mã hóa có 4 bit chỉ báo chế độ để biểu diễn.

Tên chế độ	Chỉ báo chế độ
Chế độ Numeric	0001
Chế độ Alphanumeric	0010
Chế độ Byte	0100
Chế độ Kanji	1000

*** Bước 4**: Thêm chỉ báo đếm ký tự (Character Count Indicator)

- Chỉ báo đếm ký tự là chuỗi bit biểu diễn cho số lượng ký tự cần được mã hóa, được đặt ngay sau chỉ báo chế độ.

- Độ dài của chỉ báo đếm ký tự phụ thuộc vào chế độ mã hóa và phiên bản của QR Code được dùng.

- Ví dụ như chuỗi HELLO WORLD có 11 ký tự (bao gồm khoảng trắng), phiên bản 1 và chế độ mã hóa là Alphanumeric, độ dài chuỗi chỉ báo đếm ký tự phải là 9 bits. Trong nhị phân, 11 là 1011, ta sẽ thêm số 0 ở bên trái để chuỗi dài 9 bits: 000001011. Đặt sau chỉ báo chế độ ta được: 0010 000001011.

Phiên bản 1 → 9	Chế độ Numeric: 10 bits Chế độ Alphanumeric: 9 bits Chế độ Byte: 8 bits Chế độ Kanji: 8 bits
Phiên bản 10 → 26	Chế độ Numeric: 12 bits Chế độ Alphanumeric: 11 bits Chế độ Byte: 16 Chế độ Kanji: 10 bits
Phiên bản 26 → 40	Chế độ Numeric: 14 bits Chế độ Alphanumeric: 13 bits Chế độ Byte: 16 bits Chế độ Kanji: 12 bits

*** Bước 5**: Mã hóa sử dụng chế độ đã chọn (Numeric, Alphanumeric, Byte, Kanji)

Lấy ví dụ với chế độ Numeric và Alphanumeric:

- Numeric: 8675309 mã hóa với phiên bản 1 QR Code

+ Bước 1: Chia thành các nhóm 3 ký tự : 867 530 9

+ Bước 2: Biến đổi từng nhóm thành chuỗi nhị phân:

867 → 1101100011

530 → 1000010010

9 → 1001

Sau khi biến đổi ta được chuỗi bit: 1101100011 1000010010 1001

- Alphanumeric: HELLO WORLD sử dụng phiên bản 1

+ Bước 1: Chia chuỗi thành từng cặp HE, LL, O , WO, RL, D

+ Bước 2: Chuyển từng cặp thành chuỗi nhị phân. (xem giá trị các ký tự ở alphanumeric table **[5]**)

Ví dụ cặp đầu tiên HE: H → 17, E → 14

Lấy số đầu tiên nhân 45 rồi cộng với số thứ 2: (45*17) + 14 = 779

779 → 01100001011

Tương tự cho các cặp còn lại.

Sau khi biến đổi tất cả các cặp ta được: 01100001011 01111000110 10001011100 10110111000 10011010100 001101

*** Bước 6**: Chia thành mã 8-bit và thêm Pad Bytes nếu cần

- Sau khi có được chuỗi bit bao gồm chỉ báo chế độ, chỉ báo đếm ký tự và dữ liệu dạng bit, có thể ta sẽ cần phải thêm bit 0 và các pad byte, bởi vì QR Code yêu cầu việc chuỗi bit phải lấp đầy sức chứa của QR Code.

- Xác định số số bit cần thiết ta sẽ dựa vào error correction table **[4].** Tìm con số ở cột "Total Number of data codewords for this Version and EC Level", đem nhân với 8 ra sẽ được số bit cần thiết.

- Thêm terminator nếu cần: terminator là 1 chuỗi số 0, tối đa là 4 số, thêm vào trong trường hợp chuỗi bit đang có ngắn hơn số bit yêu cầu. Nếu chuỗi bit ngắn hơn 4 bit hoặc nhiều hơn so với số bit yêu cầu thì ta thêm 4 số 0 ở cuối chuỗi. Còn nếu cần ít hơn 4 bit thì ta chỉ thêm số 0 tương ứng với số bit cần thêm.

- Thêm số 0 nếu chiều dài chuỗi không chia hết cho 8: Nếu tổng số bit không chia hết cho 8, ta sẽ thêm số 0 ở bên phải sao cho chiều dài chuỗi chia hết cho 8.

Ví dụ: HELLO WORLD sau các bước biến đổi trên sẽ dài 78 bits, không chia hết cho 8: 00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 010000

Có 6 bit ở cuối nên ta sẽ thêm 2 số 0 để tạo thành chuỗi 8 bit:

00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 010000**(00)**

- Thêm Pad Bytes nếu chuỗi vẫn quá ngắn: ta chỉ cần 1 chuỗi: 11101100 00010001 ở cuối chuỗi, lặp lại cho đến khi đủ số bit yêu cầu.

Ví dụ: 00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 01000000 **(11101100 00010001 11101100)**

1.3 Tạo mã sửa lỗi

Mã sửa lỗi cho phép bộ đọc QR Code phát hiện và sửa lỗi trong QR Code. Sau đây ta sẽ tìm hiểu cách hình thành mã sửa lỗi sau khi đã mã hóa dữ liệu.

*** Bước 1:** Chia nhỏ mã dữ liệu (là chuỗi bits ta đã mã hóa ở bước phía trên) thành các khối nếu cần.

Trước khi tạo ra mã sửa lỗi, các QR Code phiên bản 2 trở lên sẽ cần được chia nhỏ ra. Lấy ví dụ với 5-Q QR Code và các mã dữ liệu cho trước, dựa vào bảng error correction **[4]**, ta biết được 5-Q QR Code sẽ có 62 mã dữ liệu, mỗi các mã dữ liệu là một chuỗi 8 bits. Ngoài ra, bảng error correction còn cho ta biết được là 62 các mã dữ liệu này sẽ được chia làm 2 nhóm, nhóm 1 sẽ có 2 khối, mỗi khối sẽ có 15 mã dữ liệu, nhóm 2 cũng có 2 khối, mỗi khối có 16 mã dữ liệu, thứ tự của chúng sẽ được giữ nguyên khi bị chia nhỏ ra.

Nhóm	Khối	Các mã dữ liệu trong khối

Nhóm 1	Khối 1	(Mã#1) 01000011 (Mã#2) 01010101 (Mã#3) 01000110 (Mã#4) 10000110 (Mã#5) 01010111 (Mã#6) 00100110 (Mã#7) 01010101 (Mã#8) 11000010 (Mã#9) 01110111 (Mã#10) 00110010 (Mã#11) 00000110 (Mã#12) 00010010 (Mã#13) 00000110 (Mã#14) 01100111 (Mã#15) 00100110
	Khối 2	(Mã#16) 11110110 (Mã#17) 11110110 (Mã#18) 01000010 (Mã#19) 00000111 (Mã#20) 01110110 (Mã#21) 10000110 (Mã#22) 11110010 (Mã#23) 00000111 (Mã#24) 00100110 (Mã#25) 01010110 (Mã#26) 00010110 (Mã#27) 11000110 (Mã#28) 11000111 (Mã#29) 10010010 (Mã#30) 00000110
Nhóm 2	Khối 1	(Mã#31) 10110110 (Mã#32) 11100110 (Mã#33) 11110111 (Mã#34) 01110111 (Mã#35) 00110010 (Mã#36) 00000111 (Mã#37) 01110110 (Mã#38) 10000110

		(Mã#39) 01010111 (Mã#40) 00100110 (Mã#41) 01010010 (Mã#42) 00000110 (Mã#43) 10000110 (Mã#44) 10010111 (Mã#45) 00110010 (Mã#46) 00000111
	Khối 2	(Mã#47) 01000110 (Mã#48) 11110111 (Mã#49) 01110110 (Mã#50) 01010110 (Mã#51) 11000010 (Mã#52) 00000110 (Mã#53) 10010111 (Mã#54) 00110010 (Mã#55) 11100000 (Mã#56) 11101100 (Mã#57) 00010001 (Mã#58) 11101100 (Mã#59) 00010001 (Mã#60) 11101100 (Mã#61) 00010001 (Mã#62) 11101100

Bảng error correction còn cho ta biết mỗi khối sẽ có 18 mã sửa lỗi, với 4 khối thì ta sẽ có tổng cộng 72 mã sửa lỗi.

Tiếp theo, ta sẽ cần phải hiểu được các bước cơ bản của kỹ thuật sửa lỗi Reed-Solomon.

*** Hiểu về Galois Field**

Ta có thể hiểu đơn giản Galois field là một tập hữu hạn các số cùng với một số phép toán để tạo ra các số nằm trong tập hữu hạn này.

QR Code tiêu chuẩn sử dụng Galois field 2^8 hoặc có thể được gọi là Galois field 256 hay GF(256). Các số trong GF(256) sẽ có giá trị từ 0 đến 255. Mọi số trong

khoảng này đều có thể được biểu diễn bởi 8 bits, điều đó có nghĩa là mọi phép tính toán trong GF(256) cho ra kết quả có thể biểu diễn được với 8 bits.

*** Hiểu về toán học trong Galois Field**

Khi các phép toán không nằm trong GF(256) được thực hiện, có thể sẽ xuất hiện con số lớn hơn 255, lúc này ta sẽ cần một phép modulo để khiến cho số đó vẫn nằm trong Galois field.

Trong Galois field, số âm cũng có giá trị như số dương. Điều này có nghĩa là phép cộng và phép trừ là như nhau. Phép cộng và phép trừ trong Galois field sẽ được thực hiện như bình thường nhưng sau đó sẽ thực hiện thêm phép modulo, bởi vì QR Code sử dụng bit-wise modulo 2 nên việc thực hiện phép tính sẽ giống như thực hiện phép XOR.

Ví dụ: 1 + 1 = 2 % 2 = 0 hay 1 ^ 1 = 0

1 + 0 = 1 % 2 = 1 hay 1 ^ 0 = 1

*** Tạo các lũy thừa cơ số 2 sử dụng Byte-Wise Modulo 100011101**

Mọi số trong GF(256) đều có thể được biểu diễn bằng lũy thừa của 2. Như đã nói ở trên, các số trong GF(256) sẽ nằm trong khoảng 0 đến 255, nhưng $2^8 = 256$ sẽ nằm ngoài khoảng này. Ta sẽ tìm hiểu cách để có thể biểu diễn 256 thành một số nằm trong GF(256).

QR Code sử dụng byte-wise modulo 100011101. Tức là khi một số lớn hơn 255 thì số đó sẽ được XOR với 285 (giá trị thập phân của 100011101).

Ví dụ:

$2^8 = 256 \text{ ^ } 285 = 29$

Ta sẽ tạo ra các lũy thừa của 2 lớn hơn bằng cách nhân 2 vào giá trị của lũy thừa ở trước.

Ví dụ:

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

$2^{10} = 58 * 2 = 116$

$2^{11} = 116 * 2 = 232$

Khi mà giá trị của lũy thừa lớn hơn 255, ta sẽ tiếp tục thực hiện phép XOR với 285.

Ví dụ: $2^{12} = 232 * 2 = 464 \wedge 285 = 205$

* **Bước 2:** Tạo đa thức thông điệp và đa thức khởi tạo

- Đa thức thông điệp là đa thức có hệ số là các mã dữ liệu được chuyển sang dạng thập phân.

Ví dụ: Sau bước mã hóa dữ liệu ta có chuỗi các mã dữ liệu của HELLO WORLD với phiên bản 1 và độ sửa lỗi M là:

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

Ta chuyển chuỗi này sang dạng thập phân:

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

Vậy đa thức thông điệp sẽ là:

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

- Đa thức khởi tạo là đa thức có dạng $(x - \alpha^0)(x - \alpha^1) \dots (x - \alpha^{n-1})$ với n là số mã sửa lỗi cần phải tạo (xem ở error correction table [4]), đối với GF(256) thì $\alpha = 2$. Để có thể sử dụng được thì ta cần phải khai triển đa thức khởi tạo, ta có một công cụ để làm điều đó là Generator Polynomial tool **[8]**.

Ví dụ: Trong error correction table, 1-M QR Code sẽ có 10 mã sửa lỗi, ta có đa thức sau:

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

*** Bước 3:** Chia đa thức thông điệp cho đa thức khởi tạo

Đầu tiên, để cho bậc của đa thức không bị quá nhỏ trong quá trình chia, ta sẽ nhân đa thức thông điệp với x^n (n là số mã sửa lỗi cần thiết). Trong ví trên thì ta sẽ nhân với x^{10}, ta sẽ được đa thức như sau:

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

Đa thức khởi tạo cũng phải có cùng bậc với đa thức thông điệp, ta sẽ lấy đa thức khởi tạo tìm được ở trên nhân với x^{15}:

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

Các bước chia sẽ khá phức tạp, tuy nhiên có công cụ để giúp chúng ta thực hiện việc này, đó là Show division steps tool **[11]**.

Sau khi chia ta được chuỗi mã sửa lỗi:

196 35 39 119 235 215 231 226 93 23

Ta sẽ đặt chuỗi này vào sau mã dữ liệu:

32, 91, 11, 120, 209, 114, 220, 77, 67, 64, 236, 17, 236, 17, 236, 17, **196 35 39 119 235 215 231 226 93 23**

1.4 Hoàn thiện đoạn mã thông điệp

Ở mục trước, quá trình áp dụng kỹ thuật sửa lỗi Reed-Solomon đã được giải thích. Ta cũng đã biết các QR Code lớn yêu cầu phải chia các mã dữ liệu thành các khối nhỏ hơn và tạo mã sửa lỗi cho từng khối. Trong trường hợp này, các khối dữ liệu và mã sửa lỗi phải được xếp xen kẽ. Quá trình này sẽ được giải thích kĩ hơn dưới đây.

* **Bước 1:** Xác định xem cần bao nhiêu khối và mã sửa lỗi.

Đối với QR code nhỏ chỉ có một khối các mã dữ liệu thì cũng chỉ có một mã sửa lỗi tương ứng. Trong trường hợp đó ta chỉ đơn giản là đặt mã sửa lỗi sau mã dữ liệu.

Đối với QR code lớn hơn, ta sẽ tham khảo bảng error correction [4].

Lấy ví dụ với 5-Q QR Code với các mã dữ liệu cho trước. Theo bảng error correction thì 5-Q code sẽ có 2 nhóm, nhóm 1 sẽ có 2 khối, mỗi khối sẽ có 15 mã dữ liệu, nhóm 2 cũng có 2 khối, mỗi khối có 16 mã dữ liệu. Ta sẽ chuyển các mã dữ liệu về dạng số nguyên và tính mã sửa lỗi (tính như ở bước phía trên) cho từng khối.

Nhóm	Khối	Các mã dữ liệu trong khối	Tính toán
Nhóm 1	Khối 1	(Mã#1) 01000011 (Mã#2) 01010101 (Mã#3) 01000110	Dạng số nguyên của các mã dữ liệu: 67, 85, 70, 134, 87, 38, 85, 194, 119 , 50, 6, 18, 6, 103 , 38

		(Mã#4) 10000110 (Mã#5) 01010111 (Mã#6) 00100110 (Mã#7) 01010101 (Mã#8) 11000010 (Mã#9) 01110111 (Mã#10) 00110010 (Mã#11) 00000110 (Mã#12) 00010010 (Mã#13) 00000110 (Mã#14) 01100111 (Mã#15) 00100110	Mã sửa lỗi: 213 199 11 45 115 247 241 223 229 248 154 117 154 111 86 161 111 39
	Khối 2	(Mã#16) 11110110 (Mã#17) 11110110 (Mã#18) 01000010 (Mã#19) 00000111 (Mã#20) 01110110 (Mã#21) 10000110 (Mã#22) 11110010 (Mã#23) 00000111 (Mã#24) 00100110 (Mã#25) 01010110 (Mã#26) 00010110 (Mã#27) 11000110 (Mã#28) 11000111 (Mã#29) 10010010 (Mã#30) 00000110	Dạng số nguyên của các mã dữ liệu: 246, 246, 66, 7, 118, 134, 242, 7, 38, 86, 22, 198, 199, 146, 6 Mã sửa lỗi: 87 204 96 60 202 182 124 157 200 134 27 129 209 17 163 163 120 133
Nhóm 2	Khối 1	(Mã#31) 10110110 (Mã#32) 11100110 (Mã#33) 11110111 (Mã#34) 01110111 (Mã#35) 00110010 (Mã#36) 00000111 (Mã#37) 01110110 (Mã#38) 10000110 (Mã#39) 01010111 (Mã#40) 00100110 (Mã#41) 01010010	Dạng số nguyên của các mã dữ liệu: 182, 230, 247, 119, 50, 7, 118, 134, 87, 38, 82, 6, 134, 151, 50, 7 Mã sửa lỗi: 148 116 177 212 76 133 75 242 238 76 195 230 189 10 108 240 192 141

		(Mã#42) 00000110 (Mã#43) 10000110 (Mã#44) 10010111 (Mã#45) 00110010 (Mã#46) 00000111	
	Khối 2	(Mã#47) 01000110 (Mã#48) 11110111 (Mã#49) 01110110 (Mã#50) 01010110 (Mã#51) 11000010 (Mã#52) 00000110 (Mã#53) 10010111 (Mã#54) 00110010 (Mã#55) 11100000 (Mã#56) 11101100 (Mã#57) 00010001 (Mã#58) 11101100 (Mã#59) 00010001 (Mã#60) 11101100 (Mã#61) 00010001 (Mã#62) 11101100	Dạng số nguyên của các mã dữ liệu: 70, 247, 118, 86, 194, 6, 151, 50, 16, 236, 17, 236, 17, 236, 17, 236 Mã sửa lỗi: 235 159 5 173 24 147 59 33 106 40 255 172 82 2 131 32 178 236

*** Bước 2:** Sắp xếp các khối xen kẽ nhau.

Ta sẽ làm theo các bước sau:

- Lấy mã dữ liệu đầu tiên ở khối thứ nhất
- Lấy mã dữ liệu đầu tiên ở khối thứ hai
- Lấy mã dữ liệu đầu tiên ở khối thứ ba
- Lấy mã dữ liệu đầu tiên ở khối thứ tư
- Lấy mã dữ liệu thứ hai ở khối thứ nhất
- Lấy mã dữ liệu thứ hai ở khối thứ hai

…

Ta sẽ lặp lại cho đến khi tất cả các mã dữ liệu được lấy ra, lúc này ta đã xếp chúng xen kẽ với nhau.

Sau đó ta cũng làm tương tự các bước như trên đối với mã sửa lỗi.

Ví dụ:

- Đối với mã dữ liệu:

- Đầu tiên ta sẽ lấy mã dữ liệu đầu tiên của từng khối: 67, 246, 182, 70.
- Tiếp theo ta sẽ lấy mã dữ liệu thứ 2 của từng khối: 67, 246, 182, 70, 85, 246, 230, 247.
- Lặp lại cho đến khi đủ hết các mã dữ liệu. Ở vòng lặp cuối chỉ có 2 số, ta đơn giản thêm 2 con số đó vào dãy theo thứ tự như các vòng lặp trước. Lúc này ta sẽ được dãy: 67, 246, 182, 70, 85, 246, 230, 247, 70, 66, 247, 118, 134, 7, 119, 86, 87, 118, 50, 194, 38, 134, 7, 6, 85, 242, 118, 151, 194, 7, 134, 50, 119, 38, 87, 16, 50, 86, 38, 236, 6, 22, 82, 17, 18, 198, 6, 236, 6, 199, 134, 17, 103, 146, 151, 236, 38, 6, 50, 17, 7, 236

- Đối với mã sửa lỗi:

Cũng tương tự như trên, sau khi đi qua các bước ta được dãy: 213, 87, 148, 235, 199, 204, 116, 159, 11, 96, 177, 5, 45, 60, 212, 173, 115, 202, 76, 24, 247, 182, 133, 147, 241, 124, 75, 59, 223, 157, 242, 33, 229, 200, 238, 106, 248, 134, 76, 40, 154, 27, 195, 255, 117, 129, 230, 172, 154, 209, 189, 82, 111, 17, 10, 2, 86, 163, 108, 131, 161, 163, 240, 32, 111, 120, 192, 178, 39, 133, 141, 236

Để được dãy hoàn thiện, ta sẽ thêm dãy mã sửa lỗi vào sau dãy mã dữ liệu: 67, 246, 182, 70, 85, 246, 230, 247, 70, 66, 247, 118, 134, 7, 119, 86, 87, 118, 50, 194, 38, 134, 7, 6, 85, 242, 118, 151, 194, 7, 134, 50, 119, 38, 87, 16, 50, 86, 38, 236, 6, 22, 82, 17, 18, 198, 6, 236, 6, 199, 134, 17, 103, 146, 151, 236, 38, 6, 50, 17, 7, 236, **213, 87, 148, 235, 199, 204, 116, 159, 11, 96, 177, 5, 45, 60, 212, 173, 115, 202, 76, 24, 247, 182, 133, 147, 241, 124, 75, 59, 223, 157, 242, 33, 229, 200, 238, 106, 248, 134, 76, 40, 154, 27, 195, 255, 117, 129, 230, 172, 154, 209, 189, 82, 111, 17, 10, 2, 86, 163, 108, 131, 161, 163, 240, 32, 111, 120, 192, 178, 39, 133, 141, 236**

*** Bước 3:** Chuyển dãy trên thành dãy nhị phân.

Lấy ví dụ với 4 mã đầu tiên 67, 246, 182 và 70**:**

67 = 01000011

246 = 11110110

182 = 10110110

70 = 01000110

Ta sẽ hợp các chuỗi trên thành 1 chuỗi liên tục 32 bits như sau:

01000011111101101011011001000110

Chuỗi hoàn thiện:

010000111111011010110110010001100101010111110110111001101111011
101000110010000101111011101110110100001100000011101110111010101
100101011101110110001100101100001000100110100001100000011100000
110010101011111001001110110100101111100001000000111100001100011
001001110111001001100101011100010000001100100101011000100110111
011000000011000010110010100100001000100010010110001100000011011
101100000001101100011110000110000100010110011110010010100101111
110110000100110000001100011001000010001000001111110110011010101
010101111001010011101011110001111100110001110100100111110000101
101100000101100010000010100101101001111001101010010101101011100
111100101001001100000110001111011110110110100001011001001111110
001011111000100101100111011110111111001110111110010001000011110
010111001000111011100110101011111000100001100100110000101000100
110100001101111000011111111110111010110000001111001101010110010
011010110100011011110101010010011011110001000100001010000000100
101011010100011011011001000000111010000110100011111100000000100000
011011110111100011000000101100100010011110000101100011011110110
0

* **Bước 4:** Thêm Remainder bits nếu cần.

Ở một số phiên bản QR code, ta sẽ cần thêm số các số 0 ở cuối chuỗi để chuỗi dài đủ số bit yêu cầu. Những số 0 thêm vào đó gọi là Remainder bits.

Đối với QR Code version 5 như ví dụ ở trên, ta sẽ cần thêm 7 Remainder bits. Ví dụ lấy 8 bits cuối từ chuỗi trên: 11101100, thêm 7 số 0 vào cuối chuỗi ta được: 11101100**0000000.**

Dưới đây là bảng số lượng Remainder bits cho từng phiên bản QR Code:

Phiên bản QR Code	Remainder bits yêu cầu
Version 1, 7 → 13, 35 → 40	0
Version 2 → 6	7

Version 14 → 20, 28 → 34	3
Version 21 → 27	4

1.5 Đặt các module vào ma trận

Ở bước trước thì chúng ta đã có được chuỗi bits hoàn thiện. Ở bước này chúng ta sẽ đưa chuỗi bits này vào ma trận QR Code cùng với một số mẫu chức năng (function pattern).

*** Tổng quan về mẫu chức năng:**

QR Code bắt buộc phải có các mẫu chức năng. Đây là những hình phải được đặt ở các khu vực cụ thể của QR Code để đảm bảo rằng khi quét mã có thể xác định chính xác để giải mã.

- ***Các mẫu tìm kiếm (Finder patterns)*** là 3 khối nằm ở các góc trên bên trái, góc dưới bên trái và góc trên bên phải.

- ***Khoảng phân cách (Separators)*** là các khoảng trắng xung quanh các mẫu tìm kiếm.

- ***Các mẫu canh chỉnh (Alignment patterns)*** có hình dạng tương tự như các mẫu tìm kiếm nhưng nhỏ hơn và nằm nhiều nơi bên trong QR Code. Các mẫu canh chỉnh sẽ được dùng trong QR Code phiên bản 2 trở lên và vị trí của chúng sẽ phụ thuộc vào từng phiên bản của QR Code.

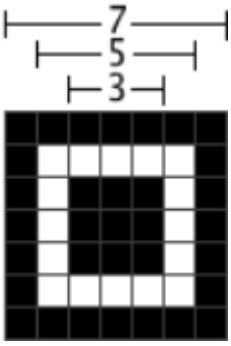
- ***Các mẫu Timing (Timing patterns)*** là các đường chấm nối các mẫu tìm kiếm.

- ***Module đen (Dark Module)*** là một module màu đen nằm bên cạnh khoảng phân cách của mẫu tìm kiếm góc dưới bên trái.

Chi tiết từng mẫu chức năng sẽ được mô tả chi tiết hơn ở bên dưới.

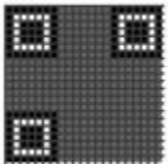
*** Bước 1: Thêm các mẫu tìm kiếm**

- Mẫu tìm kiếm bao gồm hình vuông 7x7 tạo từ các module màu đen bao quanh bên ngoài, bên trong là hình vuông tạo từ các module màu trắng với kích thước 5x5 và ở trung tâm sẽ là khối vuông 3x3 tạo từ các module đen.

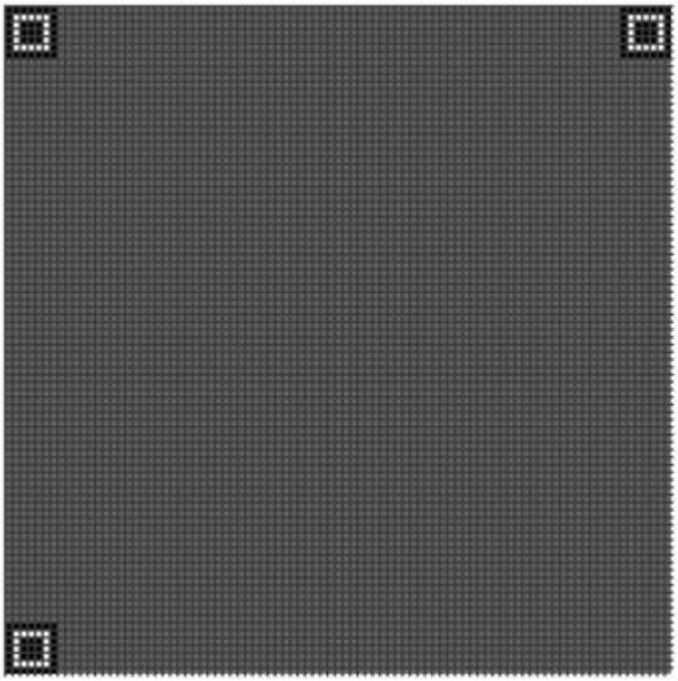


Hình 2: Hình ảnh minh họa các phần của một mẫu tìm kiếm

- Các mẫu tìm kiếm luôn nằm ở các góc trên bên trái, góc dưới bên trái và góc trên bên phải bất kể phiên bản nào của QR Code.



Hình 3: Hình ảnh minh họa các mẫu tìm kiếm trong QR Code phiên bản 1



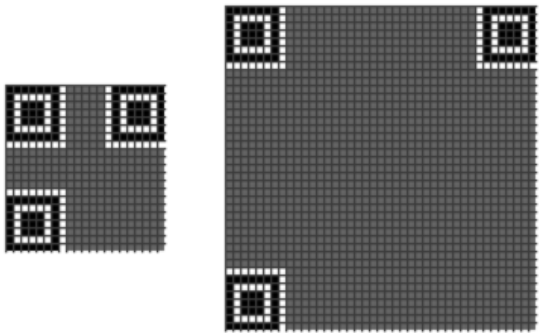
Hình 4: Hình ảnh minh họa các mẫu tìm kiếm trong QR Code phiên bản 18

- Ta có thể tính được tọa độ của module nằm góc trên bên trái của từng mẫu tìm kiếm:
  - Góc trên bên trái: (0, 0).

- Góc trên bên phải: ([(((V-1)*4)+21) - 7], 0).
- Góc dưới bên trái: (0,[(((V-1)*4)+21) - 7]).
- Với V là phiên bản của QR Code.

*** Bước 2: Thêm các khoảng phân cách**

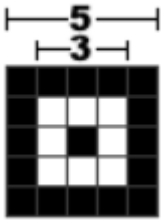
- Khoảng phân cách là các đường màu trắng bao xung quanh các mẫu tìm kiếm, ngăn cách các mẫu tìm kiếm với phần còn lại của QR Code.



Hình 5: Hình ảnh minh họa các khoảng phân cách trong QR Code phiên bản 1 và 18

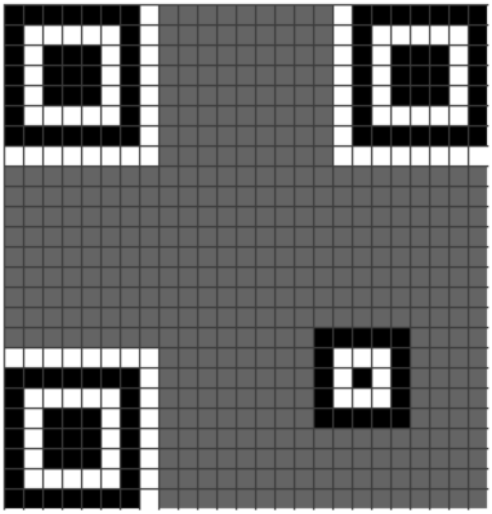
*** Bước 3: Thêm các mẫu canh chỉnh**

- QR Code phiên bản 2 trở lên sẽ yêu cầu có các mẫu canh chỉnh.
- Mẫu canh chỉnh bao gồm hình vuông 5x5 tạo từ các module màu đen bao quanh bên ngoài, hình vuông 3x3 tạo từ các module màu trắng ở trong và một module màu đen ở chính giữa.



Hình 6: Hình ảnh minh họa các thành phần của một mẫu canh chỉnh

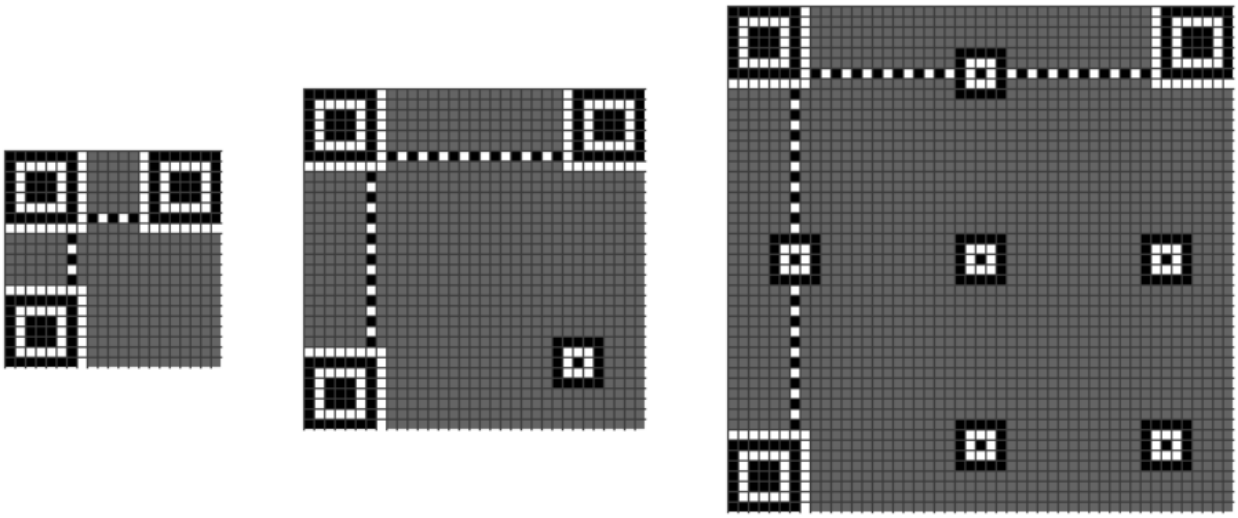
- Các mẫu canh chỉnh sẽ được đặt vào trong QR Code sau khi các mẫu tìm kiếm và các khoảng phân cách đã được thêm vào. Các mẫu canh chỉnh không được nằm chồng lên các mẫu tìm kiếm và khoảng phân cách.
- Ta có thể xem tọa độ của module trung tâm của các mẫu canh chỉnh ở alignment patterns locations table [7].



Hình 7: Ví dụ về mẫu canh chỉnh trong QR Code

* Bước 4: Thêm các mẫu Timing

- Các mẫu Timing là các đường thẳng, một đường ngang và một đường dọc được tạo thành từ các module đen và trắng xen kẽ nhau.
- Mẫu Timing ngang được đặt ở hàng thứ 7 nối giữa 2 khoảng phân cách.
- Mẫu Timing dọc được đặt ở cột thứ 7 nối giữa 2 khoảng phân cách.
- Các mẫu Timing luôn bắt đầu và kết thúc với module màu đen.
- Các mẫu canh chỉnh có thể nằm chồng lên các mẫu Timing vì các module trắng và đen của chúng luôn trùng nhau.



Hình 8: Ví dụ về mẫu Timing trong các phiên bản QR Code

* Bước 5: Thêm module đen và vùng dự trữ

- Module đen

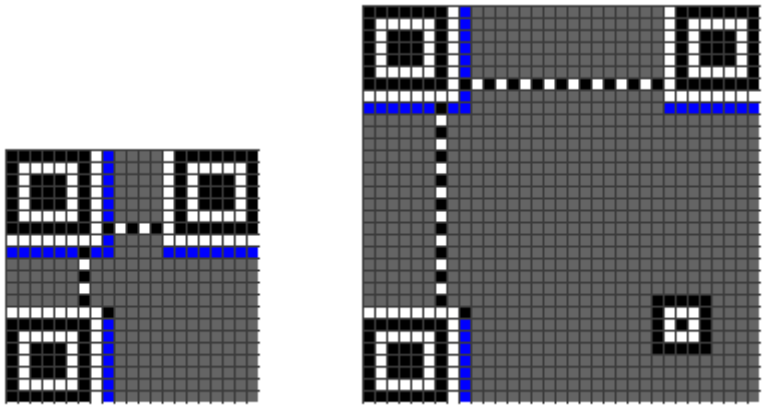
- Là module màu đen nằm bên cạnh khoảng phân cách của mẫu tìm kiếm góc dưới bên trái.
- Module đen luôn có tọa độ là ([(4 * V) + 9], 8) (V là phiên bản của QR Code).

- Vùng dự trữ cho thông tin định dạng (Format Information)

Các dải module nằm bên cạnh các khoảng phân cách sẽ là vùng dự trữ cho thông tin định dạng. Ở mẫu tìm kiếm góc trên bên trái, dải module sẽ được lưu trữ ở phía dưới và bên phải khoảng phân cách.

- Ở mẫu tìm kiếm góc trên bên phải, dải module sẽ được lưu trữ ở phía dưới khoảng phân cách.
- Ở mẫu tìm kiếm góc dưới bên trái, dải module sẽ được lưu trữ ở bên phải khoảng phân cách.

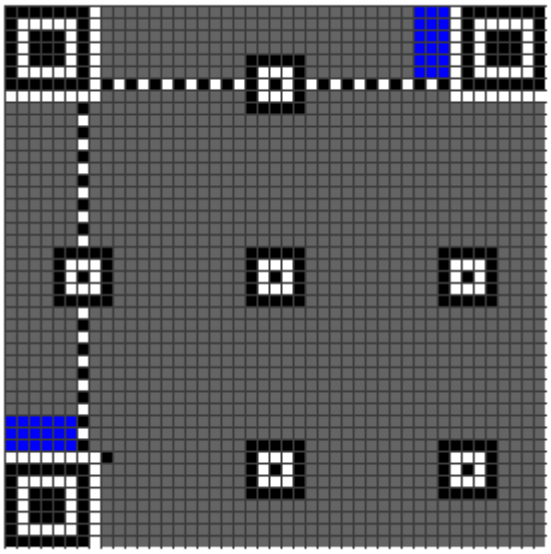
Vùng lưu trữ thông tin định dạng sẽ được biểu diễn bằng màu xanh trong hình ảnh minh họa ở dưới.



Hình 9: Hình minh họa mẫu cho vùng lưu trữ thông tin định dạng trong QR Code

- Vùng dự trữ cho thông tin phiên bản (Version Information)

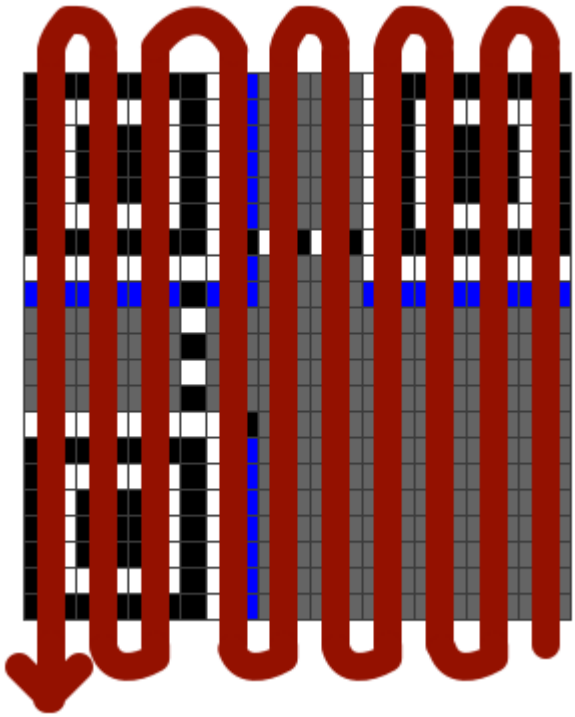
QR Code phiên bản 7 trở lên cần phải có 2 vùng để đặt các bits thông tin phiên bản. Vùng thứ nhất là khối 6x3 nằm trên Mẫu tìm kiếm góc dưới bên trái, vùng thứ hai là khối 3x6 nằm bên trái của Mẫu tìm kiếm góc trên bên phải.



Hình 10: Hình minh họa mẫu cho vùng lưu trữ thông tin phiên bản trong QR Code

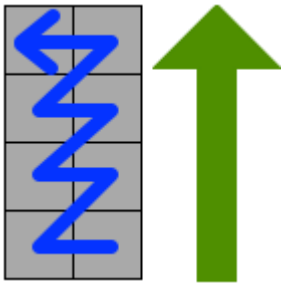
* Bước 6: Thêm các bits dữ liệu

Bit 0 sẽ được biểu diễn bằng màu trắng, bit 1 sẽ được biểu diễn bằng màu đen. Các bits dữ liệu sẽ được thêm vào từ góc dưới bên phải của ma trận và được thêm theo chiều đi lên trong một cột rộng 2 module. Khi cột lên này tới đỉnh của ma trận, cột tiếp theo nằm bên trái cột trước đó sẽ được bắt đầu thêm các bits dữ liệu theo chiều xuống. Khi chạm tới cạnh của ma trận thì lại bắt đầu với cột tiếp theo và đổi chiều. Nếu gặp các mẫu chức năng và các vùng lưu trữ, ta sẽ bỏ qua và đặt các bits dữ liệu ở các module trống. Khi tới mẫu Timing dọc, ta sẽ tiếp tục thêm bits dữ liệu ở cột bên trái của mẫu Timing này.

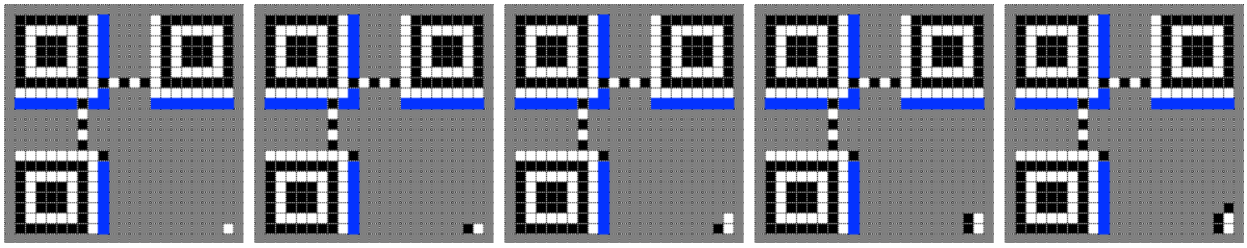


Hình 11: Hình ảnh minh họa thứ tự thêm dữ liệu vào ma trận QR Code

Thêm bit dữ liệu theo chiều đi lên sẽ được thêm theo thứ tự như sau:

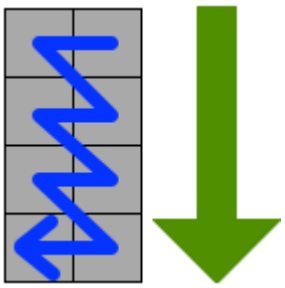


Hình 12: Hình ảnh minh họa thứ tự thêm dữ liệu theo chiều đi lên

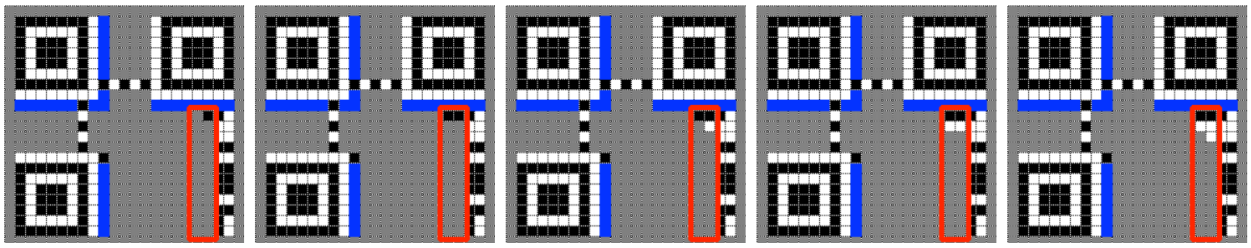


Hình 13: Ví dụ từng bước thêm dữ liệu theo chiều đi lên (thực hiện từ trái sang phải)

Thêm bit dữ liệu theo chiều đi xuống sẽ được thêm theo thứ tự như sau:

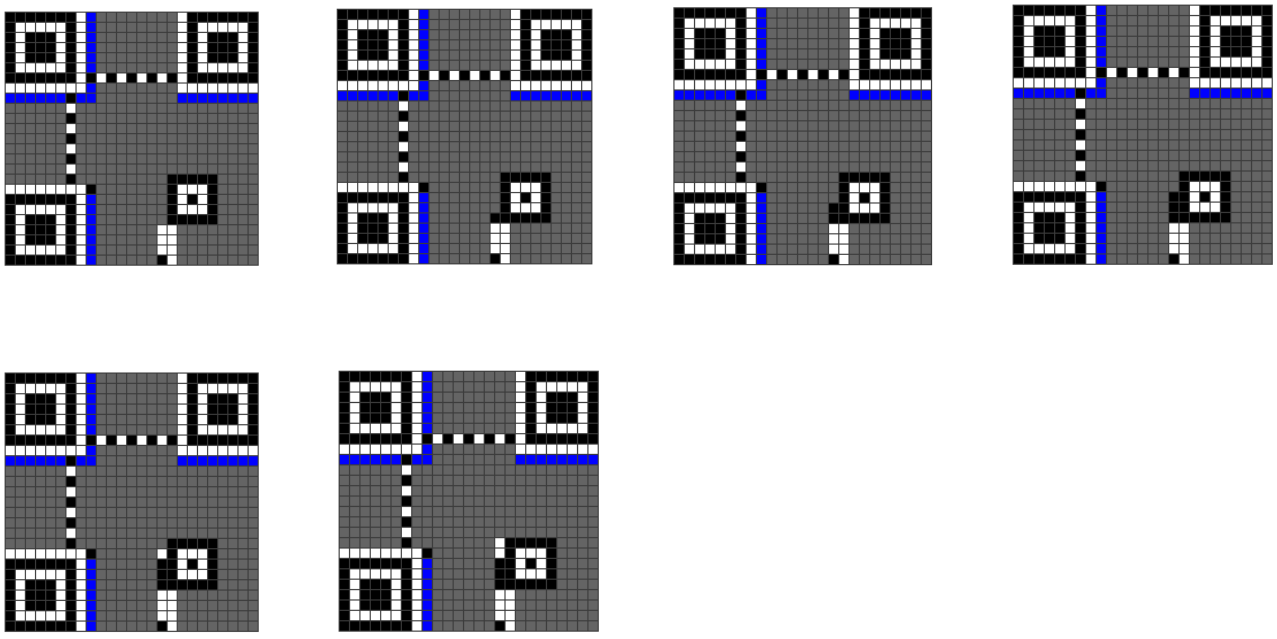


Hình 14: Hình ảnh minh họa thứ tự thêm dữ liệu theo chiều đi xuống



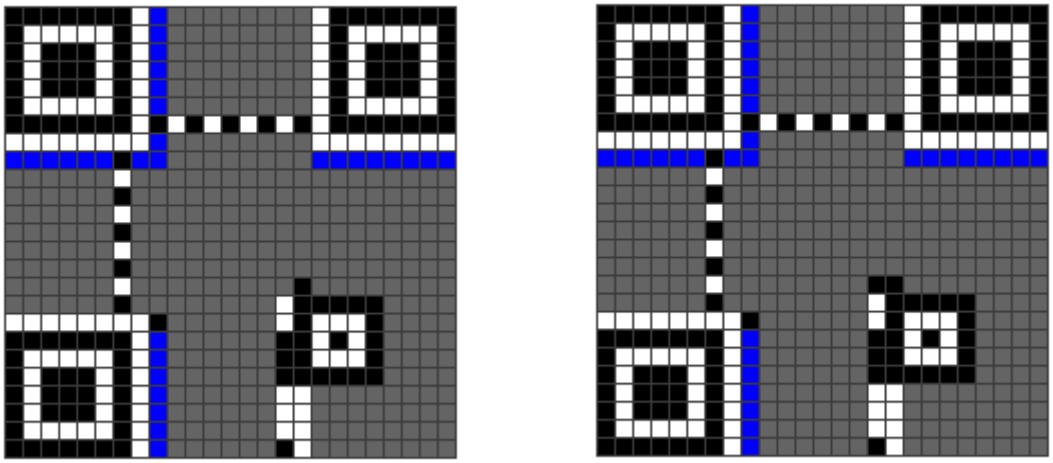
Hình 15: Ví dụ từng bước thêm dữ liệu theo chiều đi xuống (thực hiện từ trái sang phải)

Nếu gặp các mẫu chức năng và các vùng lưu trữ, ta sẽ bỏ qua và đặt các bits dữ liệu ở các module trống tiếp theo như sau:



Hình 16: Ví dụ từng bước thêm dữ liệu khi gặp mẫu chức năng (các bước thực hiện từ trái sang phải, từ trên xuống dưới)

Sau khi đã đi qua mẫu canh chỉnh thì thêm bits dữ liệu như bình thường.



Hình 17: Ví dụ từng bước thêm dữ liệu sau khi đi qua mẫu chức năng

1.6 Chọn mặt nạ

Mục tiêu: Các khối cùng màu có kích thước lớn có thể gây khó khăn cho các công cụ scan QR vì vậy ta cần biến đổi các khối cùng màu này thành các module màu trắng đen xen kẽ nhau bằng các công thức nghịch đảo màu các điểm trong ma trận.

Cách hoạt động:

Xét một điểm màu Q[i][j] với i, j lần lượt là chỉ số hàng và cột của điểm màu, các điểm có chỉ số hàng, cột thỏa điều kiện của mẫu mặt nạ được chọn sẽ thực hiện nghịch đảo màu.

Có 8 mẫu mặt nạ có thể chọn để áp dụng lên ma trận QR của ta:

- *Mẫu 0:* (i + j) % 2 = 0
- *Mẫu 1:* i % 2 = 0
- *Mẫu 2:* j % 3 = 0
- *Mẫu 3:* (i + j) % 3 = 0
- *Mẫu 4:* [floor(i / 2) + floor(j / 3)] % 2 = 0 *(floor(): làm tròn xuống)*
- *Mẫu 5:* (i * j) % 2 + (i * j) % 3 = 0
- *Mẫu 6:* [(i * j) % 2 + (i * j) % 3] % 2 = 0
- *Mẫu 7:* [(i + j) % 2 + (i * j) % 3] % 2 = 0

Lưu ý: Quy trình biến đổi các module theo mẫu mặt nạ chỉ áp dụng trên các thành phần liên quan đến dữ liệu đầu vào và khu vực mã hóa sửa lỗi, còn các thành phần vận hành của QR Code (các vùng lưu trữ phiên bản, định dạng và các mẫu chức năng) không được sửa đổi.

Làm sao để chọn mẫu mặt nạ tối ưu nhất ?

Để chọn mặt nạ trước hết ta cần tính điểm phạt (penalty) của 8 mẫu mặt nạ khi áp dụng từng mặt nạ lên ma trận QR. Mặt nạ cho giá trị điểm phạt nhỏ nhất sẽ là mặt nạ ta chọn. Chú ý rằng dù các mặt nạ chỉ được áp dụng trên một vùng của ma trận QR nhưng bước kiểm tra lại tiến hành trên toàn bộ ma trận QR.

Quy trình tính điểm sẽ thông qua 4 bước, kết quả cuối cùng sẽ là tổng điểm phạt của 4 bước.

Bước 1: Kiểm tra từng hàng một của ma trận, nếu xuất hiện chuỗi 5 module màu liên tiếp trùng nhau, giá trị điểm sẽ tăng 3. Đối với chuỗi nhiều hơn 5 module, mỗi module tăng thêm sẽ cộng thêm 1 vào giá trị điểm phạt. Sau khi kiểm tra từng hàng, ta tiếp tục kiểm tra từng cột với quy tắc như trên hàng.

Bước 2: Ở bước này ta dò tìm tất cả các khối 2×2 cùng màu trên toàn bộ ma trận tính luôn cả các khối chồng lên nhau (chẳng hạn khối 3×2 được tính là 2 khối 2×2). Kết quả điểm phạt ở bước này sẽ bằng tổng số khối nhân cho 3.

Bước 3: Ta tìm trong toàn bộ ma trận QR các dãy ở cả phương dọc và phương ngang có thứ tự đen-trắng-đen-đen-đen-trắng-đen-trắng-trắng-trắng hoặc có màu ngược lại.

Hình 18: Minh họa dãy có thứ tự như được nhắc đến ở trên

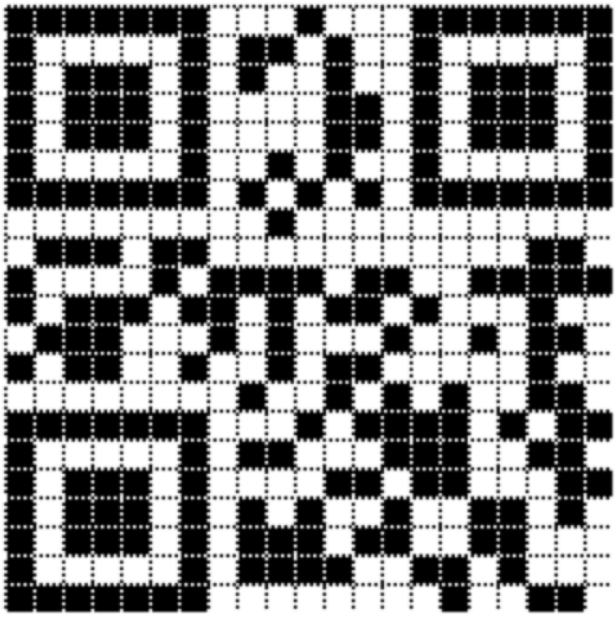
Hình 19: Minh họa dãy có thứ tự ngược lại dãy ở **Hình 18**

Mỗi dãy tìm được sẽ tăng giá trị điểm phạt thêm 40

Bước 4:

1. Đầu tiên ta tính tỉ số phần trăm module màu đen trong tổng số các module trong ma trận.

2. Sau đó tìm 2 giá trị bội số của 5 liền trước và liền sau giá trị phần trăm vừa tính ở trên.
3. Lần lượt lấy trị tuyệt đối của hiệu hai giá trị này với 50, sau đó chia cả hai cho 5
4. Cuối cùng lấy giá trị nhỏ nhất trong 2 giá trị tìm được nhân cho 10 ta sẽ tính ra điểm điểm phạt trong bước 4.



Hình 20: Minh họa ma trận của QR Code có 213 module màu đen trong toàn bộ 441 module

Tính điểm phạt bước 4 cho QR Code ở Hình 20:

1. *H = (213/441)*100 =* ***48.2993%***
2. *Hai giá trị bội số 5 liền trước và liền sau của H là* ***45*** *và* ***50***
3. ***a*** *= |45 - 50|/5 =* ***1*** *&* ***b*** *= |50 - 50|/5 =* ***0***
4. ***b nhỏ hơn a*** *vì vậy* ***điểm phạt = 0 × 10 = 0***

1.7 Tạo chuỗi thông tin định dạng và phiên bản

Bước cuối cùng sẽ là tạo ra chuỗi định dạng và chuỗi phiên bản, sau đó đặt chúng vào đúng chỗ trong QR Code.

Chuỗi định dạng : Chuỗi này sẽ bao gồm thông tin về mức độ sửa lỗi và mẫu mặt nạ đang được sử dụng trong QR Code. Do có 4 mức độ sửa lỗi (L, M, Q, H) và 8 loại mẫu mặt nạ nên sẽ có 32 chuỗi thông tin định dạng có thể được hình thành.

chuỗi định dạng sẽ dài 15 bits. Để có được chuỗi này, đầu tiên ta sẽ tạo chuỗi 5 bits mã hóa của mức độ sửa lỗi và mẫu mặt nạ được sử dụng, tiếp đến dùng 5 bits đó để tạo ra 10 bits sửa lỗi. Sau đó thực hiện phép XOR chuỗi 15 bits vừa tạo với chuỗi 101010000010010 được để có chuỗi định dạng .

Sau đây là các bước chi tiết để tạo ra được chuỗi định dạng :

- **Bước 1:** Dùng 2 bits để chứa thông tin về mức độ sửa lỗi.

Mức độ sửa lỗi	Bits	Số nguyên tương ứng
L	01	1
M	00	0
Q	11	3
H	10	2

- **Bước 2:** Chuyển con số của mẫu mặt nạ thành 3 bits.

Ví dụ: QR Code có mức độ sửa lỗi là L và mẫu mặt nạ 4.

Mức độ sửa lỗi: 01.
Mẫu mặt nạ 4: 100 (là dạng nhị phân của số 4).
Ta được chuỗi 5 bits: 01100.

- **Bước 3:** Tạo ra chuỗi bits sửa lỗi:

- Khi khởi tạo mã sửa lỗi của chuỗi định dạng , QR Code sẽ dùng đa thức khởi tạo sau: $x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$. Ta có thể chuyển chuỗi này thành chuỗi nhị phân bằng cách lấy hệ số của các phần tử. Ví dụ như x^{10} sẽ có hệ số là 1, x^9 sẽ có hệ số là 0 bởi vì nó không có trong đa thức. Vậy đa thức khi biểu diễn dưới dạng nhị phân sẽ có dạng như sau: 10100110111.
- Tiếp theo ta lấy chuỗi định dạng chia cho chuỗi được tạo từ đa thức khởi tạo. Để có thể chia được thì ta sẽ phải thêm số 0 bên phải của chuỗi định dạng sao cho đủ 15 bits, sau đó bỏ số 0 ở bên trái:

  01100 → 11000000000000
- Ta sẽ thực hiện các bước chia như sau:
  + Thêm số 0 (nếu cần) vào bên phải chuỗi đa thức khởi tạo sao cho chiều dài chuỗi bằng với chuỗi định dạng .
  + Thực hiện phép XOR với chuỗi định dạng với chuỗi đa thức khởi tạo vừa xử lý.

+ Xóa toàn bộ số 0 bên trái của chuỗi kết quả từ phép XOR.

Ví dụ phép chia đầu tiên:
Chuỗi đa thức khởi tạo: 10100110111
Chuỗi định dạng : 11000000000000
Ta sẽ phải thêm số 0 vào bên phải chuỗi đa thức khởi tạo :

10100110111 → 10100110111000

Bây giờ ta sẽ XOR 2 chuỗi này:
11000000000000 ^ 10100110111000 = 01100110111000
Sau đó bỏ số toàn bộ số 0 ở bên trái:
01100110111000 -> 1100110111000

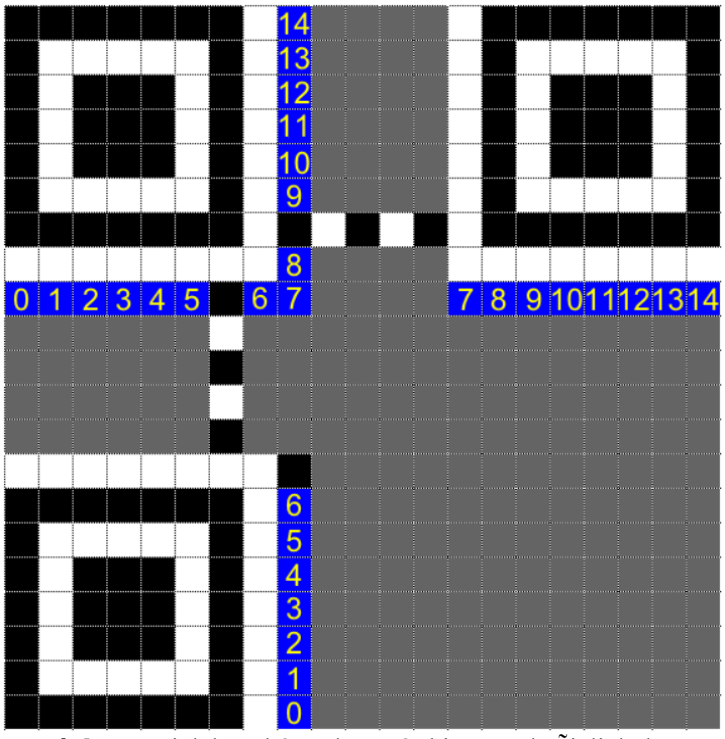
Chuỗi kết quả này sẽ là chuỗi định dạng sử dụng cho các bước chia sau. Ta sẽ thực hiện các bước chia này cho đến khi chuỗi định dạng còn 10 bits hoặc ít hơn. Nếu ít hơn 10 bits thì ta sẽ thêm số 0 vào bên phải chuỗi cho đủ 10 bits.

- Sau 4 lần chia ta sẽ được chuỗi: 1000111101. Kết hợp với 5 bits tìm được ở bước 2 ta được chuỗi: 011001000111101.

- **Bước 4:** Ta sẽ thực hiện phép XOR chuỗi vừa tìm được với chuỗi 101010000010010.
  011001000111101 ^ 101010000010010 = 110011000101111
  Vậy chuỗi chuỗi định dạng của QR Code với mức độ sửa lỗi L và mẫu mặt nạ 4 là: 110011000101111.

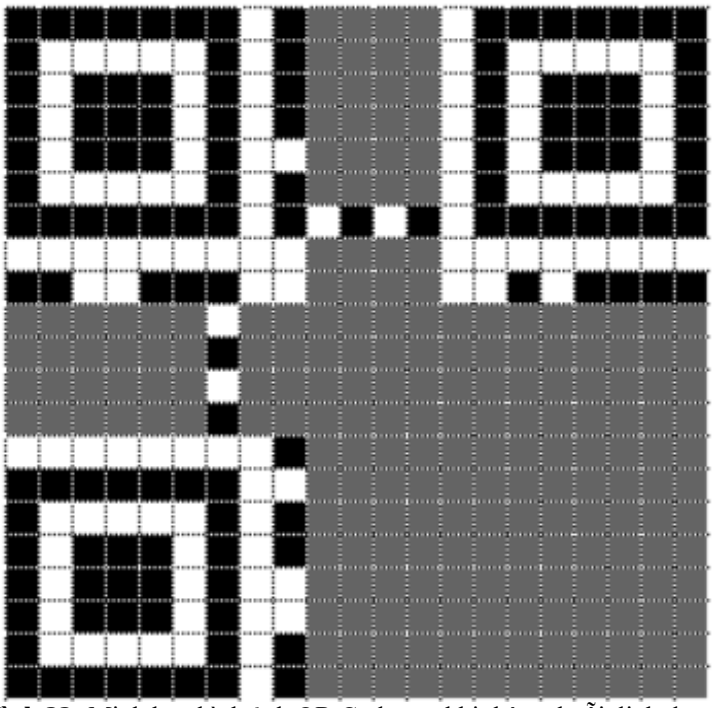
  Tiếp theo ta sẽ đưa chuỗi này vào trong QR Code theo thứ tự 0 đến 14 như sau:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	0	0	1	1	0	0	0	1	0	1	1	1	1



Hình 21: Minh họa thứ tự thêm các bits của chuỗi định dạng

Đây là QR Code (lấy ví dụ với phiên bản 1) sau khi điền chuỗi 110011000101111 ở trên:



Hình 22: Minh họa hình ảnh QR Code sau khi thêm chuỗi định dạng

Chuỗi phiên bản: QR Code dùng mã (18, 6) Golay cho chuỗi thông tin phiên bản. Chuỗi này sẽ có 18 bits, 6 bits mã hóa của phiên bản QR Code và 12 bits sửa lỗi.
- Mã hóa phiên bản của QR Code thành chuỗi 6 bits.
Ví dụ: QR Code phiên bản 7: 000111 (dạng nhị phân của số 7)

- Tạo ra chuỗi bits sửa lỗi:
- Ở bước này, ta sẽ dùng đa thức khởi tạo sau:
  $x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^2 + 1$
  Tương tự như trên thì ta sẽ có chuỗi tạo ra từ đa thức khởi tạo:
  1111100100101
- Thực hiện phép chia tương tự như các bước ở chuỗi định dạng. Có 1 số điểm khác là chuỗi khởi đầu sẽ dài 18 bits thay vì 15 bits như trên, phép chia sẽ dừng khi chuỗi dài 12 bits hoặc ít hơn thay vì 10.

- Ví dụ:
- Ta sẽ bắt đầu với chuỗi biểu diễn cho phiên bản QR Code. Ở đây sẽ lấy ví dụ với phiên bản 7: 000111
- Biến đổi chuỗi trên thành chuỗi 18 bits bằng cách thêm số 0 bên phải:
  000111 → 000111000000000000
- Xóa toàn bộ số 0 bên phải chuỗi ta được: 111000000000000.
- Thêm số 0 vào bên phải để chuỗi đa thức khởi tạo dài bằng chuỗi phiên bản: 1111100100101 → 111110010010100.
- XOR chuỗi phiên bản với chuỗi đa thức khởi tạo:
  1110000000000 ^ 111110010010100 = 110010010100

Do chuỗi đã có độ dài 12 bits nên không cần phải tiếp tục chia nữa. Trong trường hợp ít hơn 12 bits thì ta sẽ thêm số 0 bên phải cho đủ 12 bits.

Kết hợp với chuỗi 6 bits ở trên ta được chuỗi: 000111110010010100.

Tiếp theo ta sẽ đưa chuỗi này vào QR Code theo thứ tự từ 0 đến 17 như sau:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	1	1	0	0	1	0	0	1	0	1	0	0

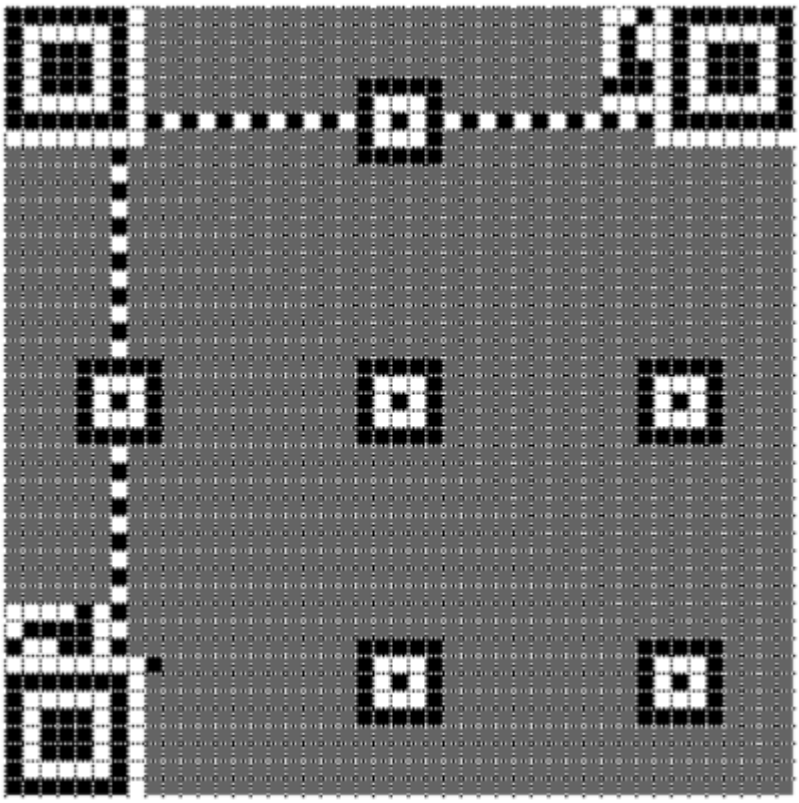
- Đối với khối ở phía trên bên phải:

0	1	2
3	4	5
6	7	8
9	10	11
12	13	14
15	16	17

- Đối với khối phía dưới bên trái:

0	3	6	9	12	15
1	4	7	10	13	16
2	5	8	11	14	17

Sau khi điền vào ma trận ta sẽ được QR Code như sau:



Hình 23: Minh họa hình ảnh QR Code sau khi thêm chuỗi phiên bản

Sau khi thêm chuỗi định dạng và chuỗi phiên bản thì ta sẽ sử dụng mẫu mặt nạ đã được xác định là có điểm phạt thấp nhất để hoàn thiện ma trận QR.

QR Code yêu cầu cần có vùng trắng (Quiet zone) bao quanh ma trận QR, đây là vùng màu trắng rộng 4 module. Sau khi thêm vùng trắng ta sẽ được một QR code hoàn chỉnh.

Hình 24: 1-Q QR Code được mã hóa từ chuỗi HELLO WORLD ở chế độ Alphanumeric

2. Giải mã

Các bước tổng quát để giải mã QR Code:

- Xác định vị trí của các mẫu tìm kiếm.

- Tính kích thước của QR Code dựa vào vị trí của 3 mẫu tìm kiếm này.

- Từ kích thước của QR Code suy ra được thông tin phiên bản.

- Căn cứ vào số phiên bản ta xác định số mẫu canh chỉnh và vị trí các điểm trung tâm của chúng.

- Ước lượng các tọa độ của bất kỳ ô (cell) nào.

- Nhị phân hóa các ô (cell) và sắp xếp lại chúng thành một chuỗi bit.

- Áp dụng thuật toán dò tìm/sửa lỗi cho chuỗi bit vừa lấy được.

- Giải mã chuỗi đã chỉnh thành các kí tự tùy thuộc vào định dạng mã hóa.

III. Cài đặt và thực nghiệm

1. Cài đặt chương trình Python

1.1 Chương trình thực hiện việc tạo mã QR-Code

Chương trình được cài đặt trong file "*qr_encoder.py*", đính kèm theo thư mục báo cáo "*CS231 Project QR Code\Source Code\Encoder*". Trong chương trình này, chúng em sử dụng thư viện có tên "qrcode"[12] để mã hóa thông tin dạng văn bản vào QR Code và lưu lại ảnh QR Code dưới định dạng PNG.

1.2 Chương trình thực hiện việc Đọc và giải mã QR-Code thông qua camera.

Chương trình được cài đặt trong file "*qr_decoder.py*", đính kèm theo thư mục "*CS231 Project QR Code\Source Code\Decoder*". Trong chương trình này, chúng em sử dụng thư viện xử lý mã QR có tên là "pyzbar"[13] kết hợp với thư viện xử lý hình ảnh "OpenCV"[14] để thực hiện việc quét mã và giải mã QR Code thông qua camera. Thông điệp được giải mã từ QR Code sẽ được lưu vào file "output.txt" có hỗ trợ định dạng UTF-8 để đọc và ghi tiếng Việt cũng như các ký tự đặc biệt.

2. Thử nghiệm và đánh giá kết quả

Cả 2 chương trình trên được chạy thực nghiệm trên máy tính cá nhân có cấu hình như sau:

- Bộ xử lý: Intel Core i5-3317U (4 CPUs) 1.70GHz.
- Bộ nhớ RAM: 4096MB
- Bộ xử lý đồ họa: Intel HD Graphics 4000 1792MB, DirectX 12
- Hệ điều hành: Windows 10 Professional 64-bit Build 15063
- Môi trường lập trình: Visual Studio Code 1.63.2
- Camera: đi kèm theo laptop, độ phân giải 640x480 (0.3 megapixel)

Dưới đây là những QR Code được tạo, đính kèm trong thư mục "*CS231 Project QR Code\Source Code\encoder\demo*":

* Ảnh "*test1.png*" với thông điệp tiếng Anh: "CS231 Introduction to Computer Vision"

test1.png

* Ảnh "*test2.png*" với thông điệp Tiếng Việt (ký tự dạng unicode) "Nhập Môn Thị Giác Máy Tính"

test2.png

* Ảnh "*test3.png*" với thông điệp là đường dẫn đến trang web chính của trường Đại Học Công Nghệ Thông Tin TPHCM "https://www.uit.edu.vn/"

test3.png

* Ảnh "*test4.png*" với thông điệp dài hơn, mô tả thông tin về trường ĐH CNTT TP.HCM: "Trường Đại học Công nghệ Thông tin (ĐH CNTT), ĐHQG-HCM là trường đại học công lập chuyên ngành CNTT được thành lập theo Quyết định số 134/2006/QĐ-TTg ngày 08/06/2006 của Thủ tướng Chính phủ trên cơ sở Trung tâm Phát triển Công nghệ Thông tin"

test4.png

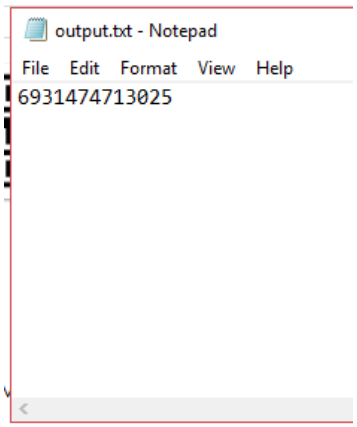
* Ảnh “test5.png” với thông điệp tiếng Nga “Информационные технологии” (dịch là: “Công nghệ Thông tin”):

test5.png

* ***Nhận xét:*** Có thể thấy, tùy vào độ dài và định dạng của thông điệp mà QR Code có kích cỡ và độ phức tạp khác nhau, thông điệp càng dài, càng phức tạp thì QR Code sử dụng phiên bản càng cao, có kích thước càng lớn, và càng phức tạp.

Tiếp đến, chúng em đã tiến hành thử nghiệm chương trình Đọc và Giải mã QR Code “*qr_decoder.py*” và đưa ra nhận xét với các trường hợp sau:

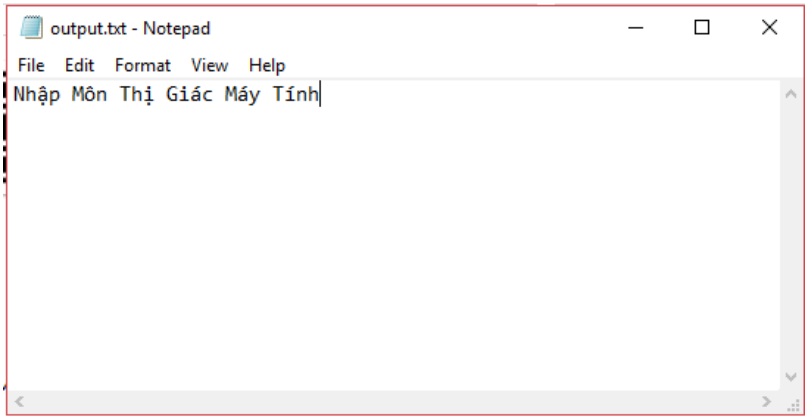
**** Quét QR Code in trên giấy, điều kiện ánh sáng bình thường:***



Hình ảnh từ camera *Thông điệp đọc được*

Nhận xét: Có thể thấy camera cho chất lượng hình ảnh khá thấp khiến QR Code bị mờ nhòe, tuy nhiên chương trình vẫn có thể đọc được thông điệp.

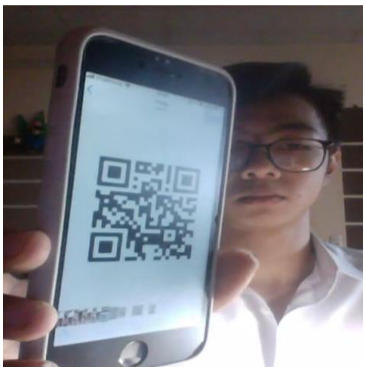
**** Trường hợp quét QR Code từ màn hình điện thoại ở góc quay thẳng:***

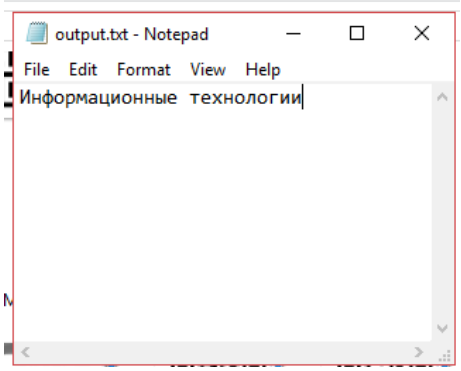


Hình ảnh từ camera *Thông điệp được ghi vào file output.txt*

Nhận xét: Các chi tiết của QR Code bị mờ và mất mát do sự ảnh hưởng từ cả ánh sáng màn hình và ánh sáng tự nhiên, chương trình vẫn đọc được thông điệp.

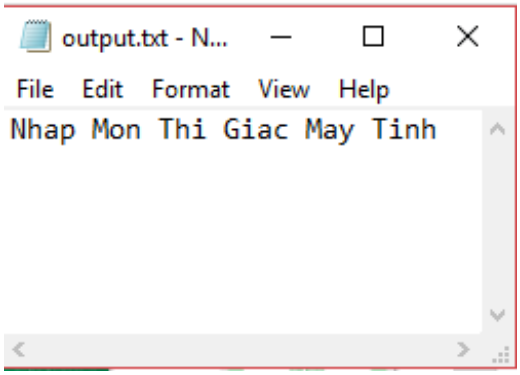
**** Trường hợp góc quay chéo:***





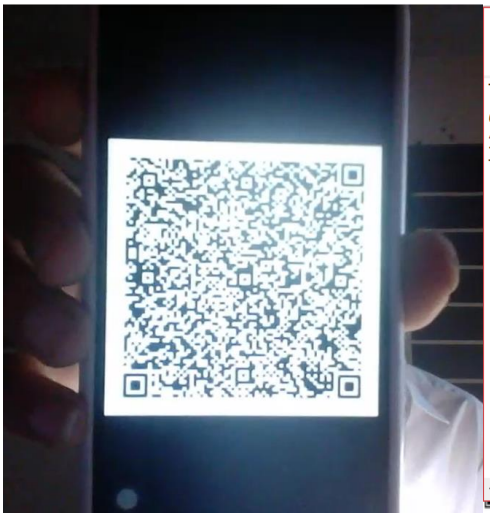
Hình ảnh từ camera *Thông điệp đọc được*

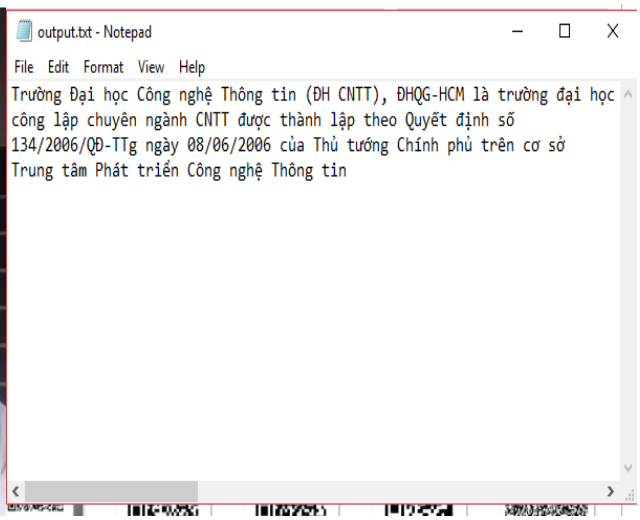
**** Trường hợp mã QR Code nằm ngang:***



Hình ảnh từ camera *Thông điệp đọc được*

**** Trường hợp QR Code quay ngược:***

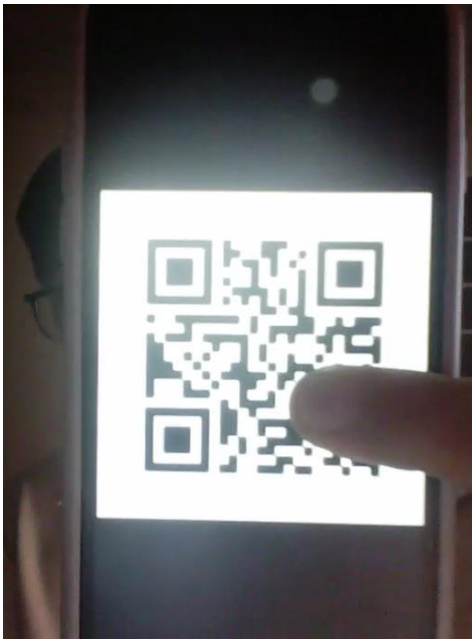




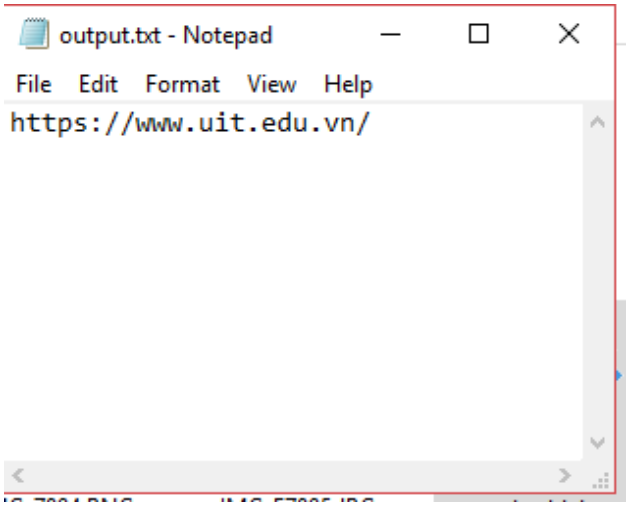
Hình ảnh từ camera *Thông điệp đọc được*

Nhận xét: Ở góc quay chéo, nằm ngang, quay ngược, chương trình vẫn đọc được thông điệp khi xác định được các Mẫu tìm kiếm và Mẫu canh chỉnh.

**** Trường hợp che khuất Mẫu canh chỉnh, một góc nhỏ bị che khuất:***



Hình ảnh từ camera



Thông điệp đọc được

**** Tuy nhiên, có những trường hợp khiến chương trình không đọc được QR Code:***

- Mẫu tìm kiếm bị che khuất:

Hình ảnh từ camera

- Hình ảnh thu được quá mờ, nhòe do chất lượng in hoặc do ở khoảng cách quá xa so với khả năng ghi hình của camera:

Hình ảnh từ camera

- QR Code bị che khuất/mất mát quá nhiều, có thể hiểu là độ mất mát vượt quá khả năng sửa lỗi của QR Code:

Hình ảnh từ camera

NHẬN XÉT CHUNG:

- Với việc tạo QR Code, chương trình "*qr_encoder.py*" đã thực hiện tốt, các mẫu QR Code tạo được có thể đọc tốt trên các thiết bị di động có ứng dụng hỗ trợ quét QR Code.

- Thông điệp càng dài, định dạng ký tự càng phức tạp hơn (các ký tự dạng Unicode) thì QR Code sẽ có phiên bản lớn hơn, kích thước lớn hơn và trông phức tạp hơn.

- Việc quét QR Code nhìn chung rất thuận tiện, mặc dù camera được sử dụng có độ phân giải khá thấp khiến ảnh có độ mờ nhòe nhẹ, nhưng nhờ vào khả

năng chịu lỗi/sửa lỗi tốt của QR Code mà chương trình vẫn có thể đọc được trong các trường hợp quay chéo, quay ngang, quay ngược, che khuất 1 phần. Và chương trình "*qr_decoder.py*" cũng đã làm việc tốt trong việc đọc và ghi lại các thông điệp có ký tự Unicode.

- Tuy nhiên, QR Code sẽ không thể quét được trong các trường hợp sau:

+ Các mẫu tìm kiếm bị che khuất.
+ Mẫu canh chỉnh bị che khuất kèm theo diện tích che khuất lớn.
+ Diện tích che khuất quá nhiều, vượt quá khả năng chịu lỗi của mẫu QR Code.
+ Khoảng cách từ QR Code đến camera quá xa, vượt quá khả năng làm việc tốt của camera.
+ Chất lượng in không tốt, sử dụng ảnh mẫu có độ phân giải thấp, không lựa chọn đúng kích thước in ấn (in quá to hoặc quá nhỏ so với độ phân giải ảnh), cũng khiến cho QR Code không thể sử dụng được.

IV. So sánh QR Code với Barcode

- Khả năng lưu trữ thông tin: Trong khi Barcode chỉ giữ được thông tin theo chiều ngang , QR Code có thể giữ thông tin cả chiều ngang và chiều dọc. Với sự khác biệt về cấu trúc này, QR Code có thể lưu trữ thông tin nhiều gấp hàng trăm lần so với Barcode, do đó nó có thể lưu trữ thông tin tốt hơn trong 1 khoảng diện tích nhỏ hơn so với Barcode.

- Khả năng chịu lỗi: Đây chính là ưu điểm vượt trội của QR Code so với Barcode. QR Code có khả năng chịu lỗi từ 7-30%. Tức là, trong trường hợp QR Code in trên sản phẩm bị bẩn hay trầy xước, trong mức cho phép 7-30%, chúng ta vẫn có thể lấy được thông tin trên đó 1 cách chính xác. Nhờ tính năng chịu lỗi vô cùng lớn này, nhiều công ty đã đưa logo hay hình ảnh của họ vào code để phòng trường hợp có bất kì câu hỏi nào liên quan.

- Một điều nhỏ nữa là QR Code có thể đọc từ mọi hướng, dễ dàng và nhanh hơn so với barcode.

HẾT

Tài liệu tham khảo:

[1] QR Code Tutorial - Thonky.com. https://www.thonky.com/qr-code-tutorial/.

[2] “QR Code vs Barcode: Differences Between QR Code and Barcode”. Science ABC, 29 March 2016. https://www.scienceabc.com/innovation/whats-qr-code-how-its-different-from-barcode.html.

[3] S. Tiwari, "An Introduction to QR Code Technology", 2016 International Conference on Information Technology (ICIT). https://www.computer.org/csdl/proceedings-article/icoit/2016/07966807/12OmNqzu6Vk.

[4] QR Code Tutorial - Thonky.com. https://www.thonky.com/qr-code-tutorial/error-correction-table.

[5] QR Code Tutorial - Thonky.com. https://www.thonky.com/qr-code-tutorial/alphanumeric-table.

[6] QR Code Tutorial - Thonky.com. https://www.thonky.com/qr-code-tutorial/character-capacities.

[7] QR Code Tutorial - Thonky.com. https://www.thonky.com/qr-code-tutorial/alignment-pattern-locations.

[8] QR Code Tutorial - Thonky.com.

https://www.thonky.com/qr-code-tutorial/generator-polynomial-tool.

[9] "ISO/IEC 8859-1 - Wikipedia", En.wikipedia.org, 2021. https://en.wikipedia.org/wiki/ISO/IEC_8859-1.

[10] "Shift JIS - Wikipedia", En.wikipedia.org, 2021. https://en.wikipedia.org/wiki/Shift_JIS.

[11] QR Code Tutorial - Thonky.com. https://www.thonky.com/qr-code-tutorial/show-division-steps

[12] “Pure python QR Code generator” https://github.com/lincolnloop/python-qrcode

[13] “pyzbar”. https://github.com/NaturalHistoryMuseum/pyzbar/

[14] “OpenCV”. https://opencv.org/