

BAN CƠ YẾU CHÍNH PHỦ  
HỌC VIỆN KỸ THUẬT MẬT MÃ

-----



ĐỀ TÀI MÔN HỌC:  
**CHỨNG THỰC ĐIỆN TỬ**

***Đề tài:***

**Tìm hiểu và nghiên cứu mã QR code và xây dựng  
chương trình mô phỏng quá trình sử dụng QR code  
trong giao dịch**

Sinh viên thực hiện:

**Đặng Vũ Hoàng Anh - AT160204**

**Nguyễn Thanh Hải - AT160221**

**Nguyễn Văn Tiến - AT160256**

**Vũ Hồng Phúc - AT160245**

*Giảng viên hướng dẫn:*

**Giảng viên Nguyễn Thị Hồng Hà**

**Hà Nội, 2023**

DANH MỤC HÌNH ẢNH.....	4
DANH MỤC BẢNG .....	7
LỜI MỞ ĐẦU .....	8
LỜI CẢM ƠN.....	9
CHƯƠNG 1. TỔNG QUAN VỀ QR CODE .....	10
1.1. QR code .....	10
1.1.1. Reed-Solomon .....	10
1.2. Cấu tạo của mã QR.....	11
1.3. Phiên bản .....	13
1.4. QR code tĩnh và động .....	14
1.4.1. QR code tĩnh.....	14
1.4.2. QR code động .....	15
1.4.3. So sánh QR code động và QR code tĩnh .....	16
1.5. Các dạng dữ liệu mà QR code có thể lưu trữ .....	16
1.6. Cách tạo QR code và cách quét.....	17
1.6.1. Cách tạo QR code .....	17
1.6.2. Cách quét, giải mã mã Qr code .....	18
1.7. Ứng dụng của mã QR .....	20
CHƯƠNG 2. QR CODE .....	22
2.1. Tổng quan về thuật toán tạo QR code .....	22
2.2. DaAnalysis.....	23
2.3. DaEncoding .....	25
2.4. Error Correction Coding.....	31
2.5. Structure Final Message .....	40
2.6. Module Placement in Matrix .....	48

2.7. DaMasking.....	62
2.8. Format and Version Information .....	70
CHƯƠNG 3. TRIỂN KHAI QR CODE TRONG GIAO DỊCH.....	82
3.1. Môi trường, thư viện.....	82
3.2. Sơ lược về ứng dụng.....	82
3.3. Tìm hiểu mã nguồn.....	87
Tài liệu tham khảo .....	97

## DANH MỤC HÌNH ẢNH

Hình 1 Dấu hiệu phát hiện định vị trong QR code .....	11
Hình 2 Dấu căn chỉnh trong QR code .....	11
Hình 3 Mô-đun thời gian trong QR code .....	12
Hình 4 Thông tin phiên bản của QR code .....	12
Hình 5 Thông tin định dạng của QR code .....	12
Hình 6 Dữ liệu và cơ chế sửa lỗi trong QR code.....	13
Hình 7 Khu vực yên tĩnh trong QR code.....	13
Hình 8 Các version QR code .....	14
Hình 9 goqr.me .....	17
Hình 10 Tạo QR code trên web.....	18
Hình 11 Chỉnh QR code tùy ý .....	18
Hình 12 Quét QR code bằng iPhone .....	19
Hình 13 Quét QR code bằng Android .....	20
Hình 14 Giới hạn với từng loại mode mã hóa .....	26
Hình 15 Các chỉ số mode mã hóa.....	27
Hình 16 Số lượng kí tự với từng version.....	28
Hình 17 Bảng mã hóa chữ số và kí tự .....	28
Hình 18 Chia đa thức cho đa thức .....	34
Hình 19 Cấu trúc mã QR .....	49
Hình 20 Mẫu tìm kiếm.....	50
Hình 21 Vị trí mẫu tìm kiếm phiên bản 1 .....	50
Hình 22 Vị trí mẫu tìm kiếm phiên bản 18.....	51
Hình 23 Bộ phân tách trong QR code .....	52
Hình 24 Mẫu căn chỉnh .....	52
Hình 25 Lỗi mẫu căn chỉnh .....	53

Hình 26 Các mẫu căn chỉnh cho mã QR phiên bản 8.....	53
Hình 27 Mẫu thời gian.....	54
Hình 28 Khu vực thông tin định dạng .....	55
Hình 29 Khu vực thông tin phiên bản .....	56
Hình 30 Hướng đặt dữ liệu tổng quát.....	57
Hình 31 Cách đặt dữ liệu khi cột đi lên.....	57
Hình 32 Minh họa đặt dữ liệu khi cột đi lên.....	58
Hình 33 Cách đặt dữ liệu khi cột đi xuống.....	58
Hình 34 Minh họa đặt dữ liệu khi cột đi xuống .....	59
Hình 35 Đặt dữ liệu khi gặp mẫu chức năng.....	60
Hình 36 Đặt dữ liệu khi gặp mẫu chức năng.....	60
Hình 37 Thêm các module.....	61
Hình 38 Cách đặt dữ liệu khi gặp mẫu thời gian.....	62
Hình 39 Minh họa đặt dữ liệu khi gặp mẫu thời gian.....	62
Hình 40 Điều kiện đánh giá 1 .....	65
Hình 41 Điều kiện đánh giá 2 .....	66
Hình 42 Điều kiện đánh giá 3 .....	67
Hình 43 Điều kiện đánh giá 4 .....	68
Hình 44 Tổng các điểm đánh giá với mỗi mẫu che mặt.....	70
Hình 45 Danh sách các bit sửa lỗi với mỗi mức độ sửa lỗi.....	71
Hình 46 Cách đặt chuỗi định dạng vào QR code .....	75
Hình 47 QR code sau khi đặt QR code.....	75
Hình 48 Thêm modul tối vào QR code .....	76
Hình 49 Chuỗi định dạng được đánh dấu đỏ.....	77
Hình 50 Điền thông tin phiên bản vào QR code .....	78
Hình 51 Cách điền thông tin phiên bản dưới bên trái .....	80
Hình 52 Cách đặt thông tin phiên bản bên phải .....	80

Hình 53 Thông tin được điền vào QR code.....	81
Hình 54 Giao diện chính phía người dùng .....	83
Hình 55 Giao diện chính phía quản trị viên .....	83
Hình 56 Thêm sản phẩm vào giỏ hàng.....	84
Hình 57 Tùy chỉnh số lượng .....	84
Hình 58 Sau khi nhập đủ các thông tin yêu cầu, một mã QR code sẽ được tạo.....	85
Hình 59 Mã QR code và giao diện quét mã QR code xuất hiện .....	85
Hình 60 3 tùy chọn để quét mã QR code: kéo thả, tải lên và sử dụng camera.....	86
Hình 61 Sau khi kéo thả mã QR code vào vùng chọn, xuất hiện nút thanh toán .....	86
Hình 62 Thanh toán thành công và chuyển đến lịch sử đặt hàng.....	87
Hình 63 Ví dụ QR code .....	88
Hình 64 Hàm <code>generateQRCode()</code> .....	92
Hình 65 Server nhận request và bắt đầu xử lý.....	93
Hình 66 Hàm <code>generateQRCode()</code> phía server tại controller.....	93
Hình 67 Đối tượng scanner.....	94
Hình 68 So sánh 2 giá trị <code>scanResult</code> và <code>oldScanResult</code> để hiển thị nút thanh toán.....	95
Hình 69 Hàm <code>tranSuccess()</code> .....	95
Hình 70 Hàm <code>createPayment</code> .....	96

## DANH MỤC BẢNG

Bảng 1 So sánh QR code động và QR code tĩnh.....	16
Bảng 2 Khả năng mã hóa của QR code .....	17
Bảng 3 Mức độ sửa lỗi của Reed-Solomon.....	25
Bảng 4 Mã hoá nhị phân giá trị HELLO WORLD .....	29
Bảng 5 Chuỗi nhị phân giá trị HELLO WORLD được thêm ký tự kết thúc .....	30
Bảng 6 Bảng mẫu dữ liệu mã 5-Q .....	33
Bảng 7 Thông tin mã 5-Q .....	33
Bảng 8 Chia dữ liệu mã 5-Q vào các khối.....	42
Bảng 9 Mã sửa lỗi tương ứng của mỗi khối .....	44
Bảng 10 Quá trình xen kẽ mã dữ liệu.....	45
Bảng 11 Quá trình xen kẽ mã lỗi.....	46
Bảng 12 Danh sách số bit dư cần với mỗi version .....	48
Bảng 13 Các mẫu che mặt của QR code .....	63

## LỜI MỞ ĐẦU

Hiện nay, khi mà công nghệ ngày càng phát triển và len lỏi vào trong mọi lĩnh vực trong cuộc sống, việc ứng dụng công nghệ thông tin và công cuộc quản lý trong doanh nghiệp là một điều tất yếu. Khi mà lời kêu gọi cho công cuộc cách mạng công nghệ 4.0 được đưa ra từ các đơn vị, tổ chức và lĩnh vực ngày càng nhiều. Các doanh nghiệp dần ý thức được tầm quan trọng của công nghệ và có các bước chuyển thay đổi mình.

Ý thức được tầm quan trọng của công nghệ thông tin, rất nhiều công ty, tổ chức hay các cá nhân đã ứng dụng công nghệ thông tin trong các vấn đề trong cuộc sống, trong đó có QR code. QR code đã được ứng dụng rất nhiều trong cuộc sống hằng ngày cũng như đã chứng tỏ được sự tiện lợi mà nó mang lại. Để giúp hiểu rõ hơn về QR code trong đời sống, nhóm bọn em đã chọn đề tài "**Tìm hiểu và nghiên cứu mã QR code và xây dựng chương trình mô phỏng quá trình sử dụng QR code trong giao dịch**".

Do thời gian tìm hiểu đề tài còn hạn chế, cũng như chưa có nhiều kinh nghiệm thực tiễn dẫn đến trong quá trình làm đề tài còn nhiều điểm sai sót và hạn chế. Kính mong cô có thể thông cảm và bổ sung ý kiến đóng góp giúp cho đề tài của nhóm em được hoàn thiện hơn.

Nhóm em xin chân thành cảm ơn!



## LỜI CẢM ƠN

Nhóm chúng em xin chân thành cảm ơn các thầy cô trường Học viện Kỹ thuật mật mã nói chung, quý thầy cô của khoa An toàn thông tin nói riêng đã tận tình dạy bảo, truyền đạt kiến thức cho chúng em trong suốt quá trình học.

Kính gửi đến cô lời cảm ơn chân thành và sâu sắc nhất, cảm ơn cô đã tận tình theo sát, chỉ bảo và hướng dẫn cho nhóm em trong quá trình thực hiện đề tài này. Cô không chỉ hướng dẫn chúng em những kiến thức chuyên ngành, mà còn giúp chúng em học thêm những kỹ năng mềm, tinh thần học hỏi, thái độ khi làm việc nhóm.

Trong quá trình tìm hiểu nhóm chúng em xin cảm ơn các sinh viên đã góp ý, giúp đỡ và hỗ trợ nhóm em rất nhiều trong quá trình tìm hiểu và làm đề tài.

Do kiến thức còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót trong quá trình làm đề tài. Chúng em rất mong nhận được sự đóng góp ý kiến của quý thầy cô để đề tài của chúng em đạt được kết quả tốt hơn.

Chúng em xin chân thành cảm ơn!

# CHƯƠNG 1. TỔNG QUAN VỀ QR CODE

## 1.1. QR code

QR code (Quick Response code, tạm dịch là “mã phản hồi nhanh”) là một loại *mã vạch ma trận* hai chiều, được phát minh vào năm 1994 bởi công ty Nhật Bản Denso Wave để dán nhãn các bộ phận ô tô. Mã vạch là hình ảnh quang học mà máy có thể đọc được, chứa thông tin dành riêng cho mặt hàng được dán nhãn. Trên thực tế, mã QR chứa dữ liệu cho bộ định vị, số nhận dạng và theo dõi khách truy cập trang web.

Để lưu trữ dữ liệu một cách hiệu quả, mã QR sử dụng bốn chế độ mã hóa được tiêu chuẩn hóa:

- Số
- Chữ và số
- Byte hoặc nhị phân
- Chữ Kanji.

*Mã vạch ma trận* còn được gọi là mã vạch 2D (hoặc đơn giản là mã 2D), là một cách thể hiện thông tin hai chiều. Nó tương tự như mã vạch tuyến tính (1 chiều), nhưng có thể biểu thị nhiều dữ liệu hơn trên một đơn vị diện tích.

Mã QR bao gồm các ô vuông màu đen được sắp xếp trong một lưới ô vuông trên nền trắng, bao gồm một số dấu hiệu xác thực, có thể được đọc bởi một thiết bị hình ảnh như máy ảnh và được xử lý bằng cách sử dụng công cụ sửa lỗi Reed–Solomon cho đến khi hình ảnh có thể được diễn giải một cách thích hợp. Dữ liệu cần thiết sau đó được trích xuất từ các mẫu có trong cả thành phần ngang và dọc của hình ảnh.

### 1.1.1. Reed-Solomon

Mã Reed–Solomon là mã vòng sửa lỗi tuyến tính được giới thiệu bởi Irving S. Reed và Gustave Solomon vào năm 1960.

Mã Reed–Solomon hoạt động trên một khối dữ liệu được coi là một tập hợp các phần tử trường hữu hạn được gọi là ký hiệu. Mã Reed–Solomon có thể phát hiện và sửa nhiều lỗi ký hiệu. Bằng cách thêm  $t = n - k$  ký hiệu kiểm tra vào dữ liệu, mã Reed–Solomon có thể phát hiện (nhưng không chính xác) bất kỳ sự kết hợp nào của tối đa  $t$  ký hiệu sai hoặc định vị và sửa tối đa  $\lfloor t/2 \rfloor$  ký hiệu sai tại các vị trí không xác định. Là một mã xóa, nó có thể sửa đến  $t$  xóa tại các vị trí đã biết và được cung cấp cho thuật toán

hoặc thuật toán có thể phát hiện và sửa các kết hợp lỗi và xóa. Mã Reed–Solomon cũng thích hợp làm mã sửa lỗi bit nhiều cụm, vì một chuỗi  $b + 1$  lỗi bit liên tiếp có thể ảnh hưởng đến tối đa hai ký hiệu có kích thước  $b$ . Việc lựa chọn  $t$  tùy thuộc vào người thiết kế mã và có thể được chọn trong giới hạn rộng.

QR code sử dụng tính năng sửa lỗi Reed–Solomon để cho phép đọc chính xác ngay cả khi một phần của mã vạch bị hỏng. Khi máy quét mã vạch không thể nhận ra một mã vạch, nó sẽ coi đó là một sự tẩy xóa.

## 1.2. Cấu tạo của mã QR

- Dấu hiệu phát hiện định vị: Nằm ở ba góc của mỗi mã, nó cho phép máy quét nhận dạng Mã chính xác và đọc ở tốc độ cao, đồng thời cho biết hướng in Mã. Về cơ bản, chúng giúp nhanh chóng xác định sự hiện diện của Mã QR trong hình ảnh và hướng của nó.



Hình 1 Dấu hiệu phát hiện định vị trong QR code

- Đánh dấu căn chỉnh: Nhỏ hơn các điểm đánh dấu phát hiện vị trí, chúng giúp làm thẳng các Mã QR được vẽ trên bề mặt cong. Và, Mã lưu trữ càng nhiều thông tin thì mã càng lớn và càng yêu cầu nhiều mẫu căn chỉnh hơn.



Hình 2 Dấu căn chỉnh trong QR code

- Mô hình thời gian: Mô-đun đen/trắng xen kẽ trên QR code với ý tưởng giúp cấu hình chính xác lưới dữ liệu. Sử dụng những dòng này, máy quét xác định ma trận dữ liệu lớn như thế nào.



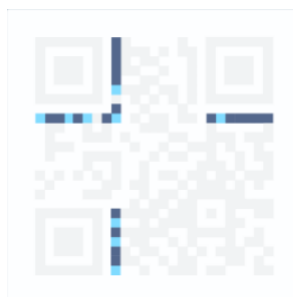
Hình 3 Mô-đun thời gian trong QR code

- Thông tin phiên bản: Với 40 phiên bản Mã QR khác nhau hiện tại, các điểm đánh dấu này chỉ định phiên bản đang được sử dụng. Những cái phổ biến nhất là các phiên bản 1 đến 7.



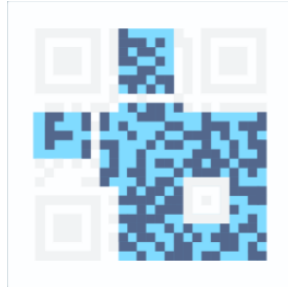
Hình 4 Thông tin phiên bản của QR code

- Thông tin định dạng: Các mẫu định dạng chứa thông tin về khả năng chịu lỗi và mẫu mặt nạ dữ liệu và giúp quét Mã dễ dàng hơn.



Hình 5 Thông tin định dạng của QR code

- Dữ liệu và cơ chế sửa lỗi: Cơ chế sửa lỗi vốn có trong cấu trúc Mã QR là nơi chứa tất cả dữ liệu của người dùng, đồng thời chia sẻ không gian với các khối sửa lỗi cho phép tối đa 30% Mã bị hỏng.



Hình 6 Dữ liệu và cơ chế sửa lỗi trong QR code

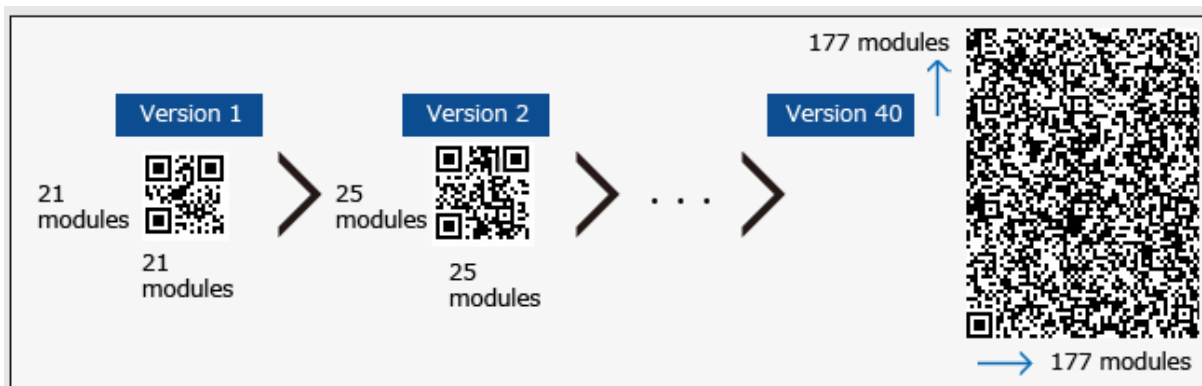
- Khu vực yên tĩnh: Điều này tương tự như tầm quan trọng của khoảng trắng trong thiết kế, đó là nó cung cấp cấu trúc và cải thiện khả năng hiểu. Cho ai hoặc những gì người dùng có thể yêu cầu? Đối với chương trình quét. Để phân biệt Mã QR với môi trường xung quanh, vùng yên tĩnh là rất quan trọng.



Hình 7 Khu vực yên tĩnh trong QR code

### 1.3. Phiên bản

Các phiên bản biểu tượng của Mã QR bao gồm từ Phiên bản 1 đến Phiên bản 40. Mỗi phiên bản có cấu hình mô-đun hoặc số lượng mô-đun khác nhau. (Mô-đun đề cập đến các chấm đen trắng tạo nên Mã QR.) "Cấu hình mô-đun" đề cập đến số lượng mô-đun có trong một biểu tượng, bắt đầu từ Phiên bản 1 (mô-đun  $21 \times 21$ ) cho đến Phiên bản 40 ( $177 \times 177$  mô-đun). Mỗi số phiên bản cao hơn bao gồm 4 mô-đun bổ sung mỗi bên.



Hình 8 Các version QR code

## 1.4. QR code tĩnh và động

### 1.4.1. QR code tĩnh

Mã QR tĩnh chứa thông tin cố định và không thể chỉnh sửa sau khi Mã được tạo. Chúng rất phù hợp cho mục đích sử dụng cá nhân và cho API Mã QR, một chìa khóa để tạo các lô Mã lớn cho ID nhân viên, huy hiệu sự kiện, tài liệu sản phẩm kỹ thuật, v.v. Do đó, QR code tĩnh không thể thay đổi đích đến đã mã hóa bên trong dữ liệu mà phải tạo một mã vạch hoàn toàn mới cho các nhu cầu mã hóa khác.

Đặc điểm của QR code tĩnh:

- Thông tin được mã hóa dưới dạng trực tiếp, cho phép xuất dữ liệu và đi đến đích người tạo mong muốn mà không thông qua bất kỳ bước chuyển hướng nào.
- Thông tin đã mã hóa trong QR code tĩnh sẽ không thể thay đổi hay cập nhật.
- Cho phép tạo miễn phí mà không cần thông qua bất cứ công nghệ hay cấp phép nào.

Ứng dụng của QR code tĩnh: Sự miễn phí và dễ tạo lại mang tính bảo mật cao, dữ liệu lớn là lợi thế mà QR code tĩnh được lựa chọn sử dụng trong nhiều hoạt động như:

- Lưu trữ thông tin cá nhân.
- Mã giảm giá.
- Làm danh thiếp.
- Làm hồ sơ cá nhân.
- Mã hóa những thông tin cố định không thay đổi.
- Mã hóa tài liệu quan trọng.

### ***1.4.2. QR code động***

Mã QR động cho phép người dùng cập nhật, chỉnh sửa và thay đổi loại mã QR nhiều lần, giúp chúng phù hợp nhất cho mục đích kinh doanh và tiếp thị. Càng nhập nhiều thông tin vào Mã QR tĩnh thì cấu trúc càng trở nên lớn hơn và phức tạp hơn. Tuy nhiên, đối với Mã động, nội dung người dùng hiển thị cho máy quét không được chứa trực tiếp trong Mã mà thay vào đó có một URL chuyển hướng ngắn được gán cho nó. Điều đó có nghĩa là mã vẫn còn nhỏ và dễ tích hợp hơn vào vật liệu in và thiết kế bao bì của người dùng.

Nắm bắt và đo lường số liệu thống kê quảng cáo của người dùng mỗi khi Mã động được quét có lẽ là tính năng tốt nhất để tối ưu hóa các chiến dịch tiếp thị. Người dùng có thể có quyền truy cập vào thời gian, địa điểm và với thiết bị nào quá trình quét diễn ra. Người dùng có thể thêm thông tin chiến dịch như phương tiện, ngày bắt đầu và ngày kết thúc, lần in và thậm chí người dùng có thể đặt lại quá trình quét và tải xuống kết quả dưới dạng báo cáo CSV.

Đặc điểm của QR code động:

- Có thể thay đổi dữ liệu đích đã mã hóa mà không cần tác động đến mã vạch.
- Thống kê và theo dõi các dữ liệu.
- URL ngắn được mã hóa trong mã QR động đơn giản cho thời gian truy xuất nhanh.
- Chiếm ít diện tích và rõ ràng kể cả khi được in ấn ở kích thước nhỏ nhờ sự đơn giản trong dữ liệu mã hóa.

Ứng dụng của QR code tĩnh: Mã QR code động được ứng dụng trong nhiều lĩnh vực đặc biệt là các lĩnh vực yêu cầu có sự theo dõi chặt chẽ như:

- Truy xuất nguồn gốc thông tin hàng hóa, thông tin sản phẩm, thông tin cá nhân
- Thay cho các Business Card tại siêu thị, nhà hàng, hội thảo, concert...
- Quảng cáo, thay thế tờ rơi.
- Vé máy bay, tàu, xe, vé xem phim.
- Thanh toán điện tử.
- Kiểm kê hàng hóa, kiểm soát hệ thống phân phối.
- Chống hàng giả.

### 1.4.3. So sánh QR code động và QR code tĩnh

a) Giống nhau:

- Được biểu hiện dưới dạng các ô vuông ma trận bên trong một ô vuông lớn theo quy tắc.
- Truy xuất dữ liệu phản hồi nhanh.
- Mã hóa được lượng dữ liệu lớn và đa dạng.
- Được giải mã bởi điện thoại thông minh, app quét mã vạch và máy quét mã vạch

2D

b) Khác nhau:

	QR code động	QR code tĩnh
Kích cỡ	Nhỏ hơn	Khá lớn và dày đặc
Khả năng chỉnh sửa	Có thể cập nhật và chỉnh sửa thông tin đã mã hóa bên trong ngay cả sau khi in.	Không thể cập nhật và chỉnh sửa.
Tốc độ phản hồi	Nhanh hơn	Chậm hơn
Trường hợp sử dụng	Marketing, kinh doanh, chính phủ	Thường là cá nhân, sử dụng 1 lần

Bảng 1 So sánh QR code động và QR code tĩnh

### 1.5. Các dạng dữ liệu mà QR code có thể lưu trữ

QR code là những ô vuông màu đen có kích thước không đều nhau. Có những ô rất to và ngược lại có những ô rất nhỏ được sắp xếp đan xen lẫn nhau theo quy tắc nhất định. Thông tin mã hóa được đưa vào mã QR càng nhiều thì mật độ các ô vuông càng dày đặc.

Khả năng mã hóa của QR code như bảng sau:



Chế độ đầu vào	Kích thước	Các ký tự khả thi, mã hóa mặc định
Số đơn thuần	Tối đa 7.089 ký tự	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Số và chữ cái	Tối đa 4.296 ký tự	0–9, A–Z (chỉ viết hoa), dấu cách, \$, %, *, +, -, ., /, :
Số nhị phân (8 bit)	Tối đa 2.953 byte	ISO/IEC 8859-1
Kanji/Kana	Tối đa 1.817 ký tự	Shift JIS X 0208

Bảng 2 Khả năng mã hóa của QR code

QR code cho phép truy xuất các thông tin như: URL, thời gian, địa điểm của sự kiện, mô tả, giới thiệu một sản phẩm nào đó,...

Để đảm bảo thông tin chứa trong QR code có thể đọc được cả khi nó bị hư hại, dữ liệu (data) được lưu trữ 2 lần (duplication). Nhờ đó, đến 30% của dữ liệu Code có thể bị hủy mà không ảnh hưởng đến khả năng đọc của Code.

## 1.6. Cách tạo QR code và cách quét

### 1.6.1. Cách tạo QR code

Tạo mã Qr code bằng những trang web miễn phí:

Hiện nay trên thị trường có rất nhiều trang web cho phép tạo QR code miễn phí đơn giản, ví dụ ở dưới hướng dẫn cách tạo mã QR code trên trang web goqr.me

*Bước 1:* Truy cập <https://goqr.me/>, chọn loại QR code muốn tạo.



Hình 9 goqr.me

*Bước 2:* Nhập đường dẫn website vào trong ô “Website address” >> goqr.me sẽ tự động

tạo QR code theo thời gian thực ở phần “Live preview” >> nhấn nút Download.

1. Type

url

2. Contents

Website address

<https://actvn.edu.vn/>

Your QR code data is encrypted during transmission (TLS/SSL) and not stored.

3. Live preview

Add a logo!

Download

Embed

Hình 10 Tạo QR code trên web

*Bước 3:* Chọn kích thước, màu QR code, màu nền, kích thước đường viền, chọn kích thước và tải về dưới dạng hình png/jpeg.

Download QR Code

Error correction code

L (recommended)

Foreground

000000

Background

FFFFFF

Border

1

Download QR Code as

SVG EPS

Size

400

Download QR Code as

PNG JPEG

Note: You can use the QR code completely free of charge (commercial and print usage allowed).

Hình 11 Chỉnh QR code tùy ý

Lúc này chỉ cần mở app quét mã vạch và quét mã QR code vừa tạo, sẽ hiện ra liên kết <https://actvn.edu.vn/>

### 1.6.2. Cách quét, giải mã mã Qr code

a) Quét mã Qr code trên điện thoại

- Trên iPhone:

Đối với người dùng điện thoại iPhone, nếu muốn quét mã QR thì thực hiện như sau: Mở camera iPhone > Quay máy ảnh vào mã QR và tiến hành quét mã QR.



Hình 12 Quét QR code bằng iPhone

- Trên Android:

Một số dòng điện thoại Android có hỗ trợ cách quét mã QR trên camera thì chỉ cần bật máy ảnh, quét mã và nhận thông tin.

Còn nếu như điện thoại của không hỗ trợ tính năng này, thì có thể sử dụng tính năng quét thông qua các app như Zalo hoặc Google Ống Kính.

- Sử dụng Zalo

Truy cập ứng dụng Zalo > Chọn biểu tượng mã QR (phía trên màn hình điện thoại) > Đưa mã QR đến trước máy ảnh và bắt đầu quét mã > Chọn Đồng ý (Yes) để tiếp tục mở liên kết.



Hình 13 Quét QR code bằng Android

## b) Sử dụng máy quét mã vạch 2D quét mã QR code

Máy quét mã vạch 2D cung cấp khả năng chinh phục mọi mã vạch 1D và 2D nhờ công nghệ tia quét Array Image (hoặc Imager) phát ra vùng quét lớn bao trùm lên toàn bộ mã vạch.

Sử dụng máy quét mã vạch 2D để giải mã thông tin trong mã QR code là giải pháp đầu tư tối ưu cho các cửa hàng, doanh nghiệp, siêu thị, nhà sách, rạp chiếu phim, bến tàu,... khi phải làm việc với mã QR thường xuyên.

Máy quét mã vạch QR code cung cấp khả năng giải mã thông tin độ chính xác cao dù mã vạch được mã hóa lượng thông tin lớn, có kích thước nhỏ hay trên bề mặt LCD có độ sáng chói cao.

## 1.7. Ứng dụng của mã QR

- **Hiện thị nội dung đa phương tiện:** Mã QR cũng được sử dụng để hướng người dùng đến nội dung đa phương tiện cụ thể (chẳng hạn như video, âm thanh, hình ảnh, tài liệu và bất kỳ loại nội dung nào có thể truy cập từ web). Loại mã QR này được gọi là "Mã QR đa phương tiện".

- **Hệ điều hành di động:** Mã QR có thể được sử dụng trên các hệ điều hành thiết bị di động khác nhau. iPhone chạy trên iOS 11 trở lên và một số thiết bị Android có thể quét mã QR mà không cần tải xuống ứng dụng bên ngoài. Ứng dụng máy ảnh có thể quét và hiển thị loại mã QR (chỉ trên iPhone) cùng với liên kết (cả trên Android và iPhone). Các thiết bị này hỗ trợ chuyển hướng URL, cho phép mã QR gửi siêu dữ liệu đến các ứng dụng hiện có trên thiết bị. Nhiều ứng dụng trả phí hoặc miễn phí có sẵn với khả năng quét mã và liên kết cứng tới một URL bên ngoài.

- **Cửa hàng ảo:** Mã QR đã được sử dụng để thiết lập "cửa hàng ảo", nơi trưng bày thông tin sản phẩm và mã QR cho khách hàng, ví dụ: trên tường nhà ga. Khách hàng quét mã QR và các sản phẩm được chuyển đến nhà của họ. Việc sử dụng này bắt đầu ở Hàn

Quốc, và Argentina, nhưng hiện đang mở rộng trên toàn cầu. Walmart, Procter & Gamble và Woolworths đã áp dụng khái niệm Cửa hàng ảo.

- **Thanh toán bằng mã QR:** Mã QR có thể được sử dụng để lưu trữ thông tin tài khoản ngân hàng hoặc thông tin thẻ tín dụng hoặc chúng có thể được thiết kế đặc biệt để hoạt động với các ứng dụng của nhà cung cấp dịch vụ thanh toán cụ thể. Có một số ứng dụng thử nghiệm thanh toán bằng mã QR trên toàn thế giới. Ở các nước đang phát triển như Trung Quốc, Ấn Độ và Bangladesh, thanh toán bằng mã QR là một phương thức thanh toán rất phổ biến và tiện lợi. Kể từ khi Alipay thiết kế phương thức thanh toán bằng mã QR vào năm 2011, thanh toán di động đã nhanh chóng được áp dụng ở Trung Quốc. Tính đến năm 2018, khoảng 83% tất cả các khoản thanh toán được thực hiện thông qua thanh toán di động.

- **Đăng nhập trang web:** Có thể sử dụng mã QR để đăng nhập vào các trang web: mã QR được hiển thị trên trang đăng nhập trên màn hình máy tính và khi người dùng đã đăng ký quét mã đó bằng điện thoại thông minh đã được xác minh, họ sẽ tự động đăng nhập. Quá trình xác thực được thực hiện bởi điện thoại thông minh mà liên lạc với máy chủ. Google đã thử nghiệm một phương thức đăng nhập như vậy vào tháng 1 năm 2012.

- **Phát hiện hàng giả:** Mã QR nổi tiếp đã được các thương hiệu và chính phủ sử dụng để cho phép người tiêu dùng, nhà bán lẻ và nhà phân phối xác minh tính xác thực của sản phẩm và giúp phát hiện hàng giả, như một phần của chương trình bảo vệ thương hiệu. Tuy nhiên, mức độ bảo mật của Mã QR thông thường bị hạn chế do Mã QR được in trên các sản phẩm gốc dễ dàng được sao chép trên các sản phẩm giả, mặc dù việc phân tích dữ liệu được tạo ra do quét Mã QR có thể được sử dụng để phát hiện hàng giả và hoạt động bất hợp pháp. Có thể đạt được mức bảo mật cao hơn bằng cách nhúng hình mờ kỹ thuật số hoặc mẫu phát hiện sao chép vào hình ảnh của Mã QR. Điều này làm cho Mã QR an toàn hơn trước các nỗ lực làm giả và các sản phẩm giả có chứa Mã QR giả có thể được phát hiện bằng cách quét Mã QR an toàn bằng một ứng dụng cụ thể (mặc dù bản thân thông báo Mã QR là hợp lệ). Hiệp ước quy định về apostille (giấy tờ có con dấu xác thực), đã được cập nhật để cho phép các quốc gia cấp apostille kỹ thuật số; apostille kỹ thuật số là một tài liệu PDF có chữ ký mã hóa chứa mã QR cho URL chính xác của tài liệu gốc, cho phép người dùng xác minh apostille từ phiên bản in của tài liệu.

- **Vé di động:** Có một hệ thống theo đó mã QR có thể được hiển thị trên một thiết bị như điện thoại thông minh và được sử dụng làm vé vào cửa. Việc sử dụng nó là phổ biến cho J1 League và vé bóng chày chuyên nghiệp Nippon ở Nhật Bản. Trong một số trường hợp, quyền có thể được chuyển nhượng qua Internet.

## CHƯƠNG 2. QR CODE

### 2.1. Tổng quan về thuật toán tạo QR code

Quá trình QR code mã hóa như sau:

- Bước 1: Phân tích dữ liệu - DaAnalysis

Mã QR mã hóa một chuỗi văn bản. Tiêu chuẩn QR có bốn mode để mã hóa văn bản: số, chữ và số, byte và Kanji. Mỗi mode mã hóa văn bản dưới dạng một chuỗi bit (1 và 0), nhưng mỗi mode sử dụng một phương pháp khác nhau để chuyển đổi văn bản thành bit và mỗi phương pháp mã hóa được tối ưu hóa để mã hóa dữ liệu với chuỗi bit ngắn nhất có thể. Do đó, bước đầu tiên của là thực hiện phân tích dữ liệu để xác định liệu văn bản của có thể được mã hóa ở mode số, chữ và số, byte hoặc Kanji hay không, sau đó chọn mode tối ưu nhất cho văn bản của bạn.

- Bước 2: Mã hóa dữ liệu - DaEncoding

Bây giờ đã chọn mode mã hóa thích hợp cho văn bản của mình, bước tiếp theo là mã hóa văn bản. Phần mã hóa dữ liệu mô tả chi tiết quá trình này cho từng mode mã hóa. Kết quả của bước này là một chuỗi bit được chia thành các từ mã dữ liệu mỗi từ 8 bit.

- Bước 3: Mã hóa sửa lỗi - Error Correction Coding

Như đã giải thích ở trên, mã QR sử dụng sửa lỗi. Điều này có nghĩa là sau khi tạo ra chuỗi bit dữ liệu đại diện cho văn bản của mình, phải sử dụng các bit đó để tạo ra các từ mã sửa lỗi bằng cách sử dụng một quá trình gọi là sửa lỗi Reed-Solomon.

Máy quét QR đọc cả các từ mã dữ liệu và các từ mã sửa lỗi. Bằng cách so sánh hai từ mã, máy quét có thể xác định liệu nó đã đọc dữ liệu chính xác hay không và có thể sửa lỗi nếu nó không đọc dữ liệu chính xác. Phần mã hóa sửa lỗi giải thích quá trình tạo ra các từ mã sửa lỗi chi tiết. Để biết thêm thông tin, hãy đọc bài viết của Wikipedia về sửa lỗi Reed-Solomon.

- Bước 4: Cấu trúc thông điệp cuối cùng - Structure Final Message

Các từ mã dữ liệu và sửa lỗi được tạo ra trong các bước trước đó phải được sắp xếp theo thứ tự đúng. Đối với các mã QR lớn, các từ mã dữ liệu và sửa lỗi được tạo ra theo khối và các khối này phải được xen kẽ theo quy định của mã QR. Quá trình này được giải thích trong phần cấu trúc thông điệp cuối cùng.

- Bước 5: Đặt Module trong Ma trận - Module Placement in Matrix

Sau khi tạo ra các từ mã dữ liệu và sửa lỗi và sắp xếp chúng theo thứ tự đúng, phải đặt các bit vào ma trận mã QR. Các từ mã được sắp xếp trong ma trận theo một cách cụ thể. Trong bước này, cũng sẽ đặt các mẫu chung cho tất cả các mã QR, chẳng hạn như các hộp ở ba góc. Quá trình này được giải thích chi tiết trong phần đặt module trong ma trận.

- **Bước 6: Che dữ liệu - DaMasking**

Một số mẫu trong ma trận mã QR có thể khiến máy quét mã QR khó đọc mã chính xác. Để khắc phục điều này, tiêu chuẩn mã QR định nghĩa tám mẫu mặt nạ, mỗi mẫu thay đổi mã QR theo một mẫu cụ thể. phải xác định mẫu mặt nạ nào dẫn đến mã QR có ít đặc điểm không mong muốn nhất. Điều này được thực hiện bằng cách đánh giá từng ma trận đã che dựa trên bốn quy tắc phạt. Mã QR cuối cùng của phải sử dụng mẫu mặt nạ đã dẫn đến điểm phạt thấp nhất. Quá trình che được giải thích trong phần che dữ liệu.

- **Bước 7: Thông tin Định dạng và Phiên bản - Format and Version Information**

Bước cuối cùng là thêm thông tin định dạng và (nếu cần thiết) thông tin phiên bản vào mã QR bằng cách thêm các pixel vào các khu vực cụ thể của mã đã được để trống trong các bước trước. Các pixel định dạng xác định mức sửa lỗi và mẫu mặt nạ được sử dụng trong mã QR này. Các pixel phiên bản mã hóa kích thước của ma trận QR và chỉ được sử dụng trong các mã QR lớn hơn. Để biết chi tiết về bước cuối cùng này, hãy đọc phần thông tin định dạng và phiên bản.

Sau đây nhóm sẽ trình bày chi tiết từng bước.

## **2.2. DaAnalysis**

### **Các mode của QR code**

Có bốn mode mã hóa chuẩn được sử dụng trong mã QR để lưu trữ dữ liệu một cách hiệu quả: mode số, mode chữ số và ký tự đặc biệt, mode byte/binary và mode kanji.

Mode số được sử dụng cho các chữ số thập phân từ 0 đến 9.

Mode chữ số và ký tự đặc biệt được sử dụng cho các chữ số thập phân từ 0 đến 9, cũng như các chữ cái in hoa (không phải in thường!), và các ký hiệu \$, %, \*, +, -, ., /, và : cũng như khoảng trắng. Tất cả các ký tự được hỗ trợ cho mode chữ số và ký tự đặc biệt được liệt kê trong cột bên trái của bảng chữ cái này.

Mode byte mặc định là cho các ký tự từ bộ ký tự ISO-8859-1. Tuy nhiên, một số máy quét mã QR có thể tự động phát hiện xem UTF-8 có được sử dụng trong mode byte không.

Mode kanji được sử dụng cho các ký tự hai byte từ bộ ký tự Shift JIS. Trong khi UTF-8 có thể mã hóa các ký tự kanji, nhưng nó phải sử dụng ba hoặc bốn byte để làm điều này. Shift JIS, ngược lại, chỉ sử dụng hai byte để mã hóa mỗi ký tự kanji, vì vậy mode kanji nén các ký tự kanji một cách hiệu quả hơn. Nếu toàn bộ chuỗi nhập vào chỉ gồm các ký tự trong khoảng hai byte của Shift JIS, hãy sử dụng mode kanji. Cũng có thể sử dụng nhiều mode trong cùng một mã QR, như được mô tả sau trên trang này.

Mode Extended Channel Interpretation (ECI) chỉ định bộ ký tự (ví dụ: UTF-8) trực tiếp. Tuy nhiên, một số máy quét mã QR không hỗ trợ mode ECI và sẽ không hiểu các mã QR sử dụng nó.

Mode Structured Append mã hóa dữ liệu trên nhiều mã QR, tối đa là 16 mã QR. Trong hướng dẫn này sẽ không thảo luận về mode này nhưng có thể thêm thông tin chi tiết sau.

Mode FNC1 cho phép mã QR hoạt động như một mã vạch GS1.

### ***Chọn mode phù hợp***

Để chọn chế độ hiệu quả nhất cho mã QR, hãy xem xét các ký tự trong chuỗi nhập và kiểm tra các điều kiện sau.

- Nếu chuỗi nhập chỉ bao gồm các chữ số thập phân (từ 0 đến 9), hãy sử dụng chế độ số.
- Nếu chế độ số không áp dụng được và nếu tất cả các ký tự trong chuỗi nhập có thể được tìm thấy trong cột bên trái của bảng chữ cái số và ký tự đặc biệt, hãy sử dụng chế độ chữ số và ký tự đặc biệt. Chữ cái thường KHÔNG thể được mã hóa trong chế độ chữ số và ký tự đặc biệt; chỉ có chữ cái in hoa.
- Nếu có một ký tự không nằm trong cột bên trái của bảng chữ cái số và ký tự đặc biệt nhưng có thể được mã hóa trong ISO 8859-1, hãy sử dụng chế độ byte. Như đã đề cập ở trên, các máy quét mã QR có thể nhận ra UTF-8 trong chế độ byte.
- Nếu tất cả các ký tự đều thuộc bộ ký tự Shift JIS, hãy sử dụng chế độ kanji. Các ký tự Shift JIS có thể được mã hóa trong UTF-8 thay vào đó, vì vậy có thể sử dụng chế độ byte cho Kanji, nhưng thông thường hiệu quả hơn là sử dụng Shift JIS và sử dụng chế độ kanji cho các ký tự kanji.

### ***Kết hợp nhiều mode và tối ưu hóa***

Có thể sử dụng nhiều chế độ trong một mã QR duy nhất bằng cách bao gồm bộ chỉ mục chế độ trước mỗi phần của byte sử dụng chế độ đó. Thông số kỹ thuật mã QR giải thích cách chuyển đổi chế độ một cách tối ưu nhất. Hướng dẫn này sẽ giả định rằng sẽ không pha trộn các chế độ trong mã QR của mình.



## 2.3. DaEncoding

### ***Bước 1: Chọn mức độ sửa lỗi***

Trước khi mã hóa dữ liệu, hãy chọn một cấp độ sửa lỗi. Như đã đề cập trong phần giới thiệu, mã QR sử dụng sửa lỗi Reed-Solomon.

Quá trình này tạo ra các từ mã sửa lỗi (byte) dựa trên dữ liệu được mã hóa. Một máy đọc mã QR có thể sử dụng các byte sửa lỗi này để xác định xem nó có đọc được dữ liệu một cách chính xác hay không và các từ mã sửa lỗi có thể được sử dụng để sửa chữa các lỗi đó. Có bốn cấp độ sửa lỗi: L, M, Q, H. Bảng sau liệt kê các cấp độ và khả năng sửa lỗi của chúng.

Mức độ sửa lỗi	Khả năng sửa lỗi
L - Low	Khôi phục 7% dữ liệu
M - Medium	Khôi phục 15% dữ liệu
Q - Quantity	Khôi phục 25% dữ liệu
H - High Level	Khôi phục 30% dữ liệu

Bảng 3 Mức độ sửa lỗi của Reed-Solomon

Hãy nhớ rằng các cấp độ sửa lỗi cao hơn yêu cầu nhiều byte hơn, vì vậy càng cao cấp độ sửa lỗi thì mã QR càng phải lớn hơn.

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

Các kích thước khác nhau của mã QR được gọi là các phiên bản. Có 40 phiên bản có sẵn. Phiên bản nhỏ nhất là phiên bản 1 và có kích thước 21 pixel x 21 pixel. Phiên bản 2 có kích thước 25 pixel x 25 pixel. Phiên bản lớn nhất là phiên bản 40 và có kích thước 177 x 177 pixel. Mỗi phiên bản lớn hơn phiên bản trước đó 4 pixel.

Mỗi phiên bản đều có dung lượng tối đa, tùy thuộc vào chế độ đang sử dụng. Ngoài ra, mức sửa lỗi còn hạn chế dung lượng thêm nữa. Bảng dung lượng ký tự liệt kê dung lượng của tất cả các phiên bản QR cho một chế độ mã hóa và mức sửa lỗi đã cho.

đếm số ký tự cần mã hóa và xác định phiên bản nhỏ nhất có thể chứa số lượng ký tự đó cho chế độ mã hóa và mức độ sửa lỗi mong muốn.

Ví dụ, cụm từ HELLO WORLD có 11 ký tự. Nếu mã hóa nó với mức sửa lỗi cấp Q, bảng dung lượng ký tự cho biết rằng một mã phiên bản 1 sử dụng mức sửa lỗi cấp Q có thể

chứa 16 ký tự ở chế độ chữ và số, vì vậy phiên bản 1 là phiên bản nhỏ nhất có thể chứa số lượng ký tự này. Nếu cụm từ dài hơn 16 ký tự, chẳng hạn như HELLO THERE WORLD (có 17 ký tự) thì phiên bản 2 sẽ là phiên bản nhỏ nhất.

Tham khảo chi tiết tại [Character Capacities by Version, Mode, and Error Correction](#)

### *Giới hạn*

Giới hạn trên Mã QR có dung lượng cao nhất là 40-L (phiên bản 40, mức sửa lỗi cấp L). Dưới đây là một bảng liệt kê dung lượng của mã QR 40-L cho bốn chế độ mã hóa. Đây là số lượng ký tự tối đa mà một mã QR đơn có thể chứa. Các phiên bản 40-M, 40-Q và 40-H có dung lượng thấp hơn vì chúng yêu cầu nhiều không gian hơn để chứa nhiều từ mã sửa lỗi hơn.

Encoding Mode	Maximum number of characters a 40-L code can contain in that mode
Numeric	7089 characters
Alphanumeric	4296 characters
Byte	2953 characters
Kanji	1817 characters

Hình 14 Giới hạn với từng loại mode mã hóa

### ***Bước 3: Thêm chỉ số Mode***

Mỗi chế độ mã hóa đều có một chỉ số chế độ bốn bit xác định nó. Dữ liệu được mã hóa phải bắt đầu bằng chỉ số chế độ thích hợp xác định chế độ được sử dụng cho các bit sau đó. Bảng sau liệt kê các chỉ số chế độ cho mỗi chế độ.

Ví dụ, nếu mã hóa HELLO WORLD ở chế độ chữ và số, chỉ số chế độ là 0010.

Encoding modes	
Indicator	Meaning
0001	Numeric encoding (10 bits per 3 digits)
0010	Alphanumeric encoding (11 bits per 2 characters)
0100	Byte encoding (8 bits per character)
1000	Kanji encoding (13 bits per character)
0011	Structured append (used to split a message across multiple QR symbols)
0111	<a href="#">Extended Channel Interpretation</a> (select alternate character set or encoding)
0101	FNC1 in first position (see <a href="#">Code 128</a> for more information)
1001	FNC1 in second position
0000	End of message (Terminator)

Hình 15 Các chỉ số mode mã hóa

#### ***Bước 4: Thêm chỉ số số lượng ký tự***

Chỉ số số lượng ký tự là một chuỗi bit biểu diễn số lượng ký tự đang được mã hóa. Chỉ số số lượng ký tự phải được đặt sau chỉ số chế độ. Hơn nữa, chỉ số số lượng ký tự phải có độ dài nhất định, tùy thuộc vào phiên bản QR.

Đếm số lượng ký tự trong văn bản đầu vào ban đầu, sau đó chuyển đổi số đó thành nhị phân. Độ dài của chỉ số số lượng ký tự phụ thuộc vào chế độ mã hóa và phiên bản mã QR sẽ được sử dụng. Để làm cho chuỗi nhị phân có độ dài thích hợp, thêm 0 vào bên trái.

Các danh sách sau chứa kích thước của các chỉ số số lượng ký tự cho mỗi chế độ và phiên bản. Ví dụ, nếu mã hóa HELLO WORLD trong một mã QR phiên bản 1 ở chế độ chữ và số, chỉ số số lượng ký tự phải dài 9 bit. Số lượng ký tự của HELLO WORLD là 11. Trong nhị phân, 11 là 1011. Thêm vào bên trái để làm cho nó dài 9 bit: 000001011. Đặt nó sau chỉ số chế độ từ bước 3 để có chuỗi bit sau: 0010 000001011

Number of bits in a length field (Character Count Indicator)			
Encoding	Ver. 1–9	10–26	27–40
Numeric	10	12	14
Alphanumeric	9	11	13
Byte	8	16	16
Kanji	8	10	12

Hình 16 Số lượng kí tự với từng version

Bước 5: Mã hóa sử dụng Mode đã chọn

Mã hóa HELLO WORLD. Với mỗi chế Mode sẽ có cách mode khác nhau. Ở đây HELLO WORLD là Alphanumeric mode nên sẽ làm như sau:

Bước 1: Chia từ ban đầu thành các cặp kí tự: HE, LL, O , WO, RL, D

Đối với chế độ chữ và số, mỗi ký tự chữ và số được biểu diễn bằng một số.

Alphanumeric character codes									
Code	Character	Code	Character	Code	Character	Code	Character	Code	Character
00	0	09	9	18	I	27	R	36	Space
01	1	10	A	19	J	28	S	37	\$
02	2	11	B	20	K	29	T	38	%
03	3	12	C	21	L	30	U	39	*
04	4	13	D	22	M	31	V	40	+
05	5	14	E	23	N	32	W	41	–
06	6	15	F	24	O	33	X	42	.
07	7	16	G	25	P	34	Y	43	/
08	8	17	H	26	Q	35	Z	44	:

Hình 17 Bảng mã hóa chữ số và kí tự

Đối với mỗi cặp ký tự, lấy biểu diễn số (từ bảng chữ và số) của ký tự đầu tiên và nhân nó với 45. Sau đó cộng số đó với biểu diễn số của ký tự thứ hai.

Ví dụ, cặp đầu tiên trong HELLO WORLD là HE.

H → 17

E → 14

Theo các bước trong đoạn văn trước đó, nhân số đầu tiên với 45, sau đó cộng nó với số thứ hai:  $(45 * 17) + 14 = 779$

Bây giờ chuyển đổi số đó thành một chuỗi nhị phân 11 bit, thêm vào bên trái với 0 nếu cần.

779 → 01100001011

Với kí tự không có cặp, chỉ cần mã hóa nó 6 bit.

Như vậy sau khi mã hóa có bảng sau

Chỉ số Mode	Chỉ số đếm kí tự	Dữ liệu đã mã hóa
0010	000001011	01100001011(HE) 01111000110(LL) 10001011100 (O) 10110111000(WO) 10011010100(RL) 001101(D)

Bảng 4 Mã hoá nhị phân giá trị HELLO WORLD

Những chế độ khác tham khảo thêm tại

- [Numeric Mode Encoding](#)
- [Byte Mode Encoding](#)
- [Kanji Mode Encoding](#)

#### ***Bước 5: Chia thành các từ mã 8 bit và thêm byte đệm nếu cần***

Sau khi có được một chuỗi bit bao gồm chỉ số chế độ, chỉ số số lượng ký tự và các bit dữ liệu như được mô tả, có thể cần thêm 0 và byte đệm, vì quy định mã QR yêu cầu chuỗi bit phải hoàn toàn điền đầy dung lượng tổng thể của mã QR. Các phần sau giải thích quá trình thêm 0 và byte đệm vào chuỗi bit.

*Xác định số lượng bit cần thiết cho mã QR này*

Để xác định số lượng bit dữ liệu cần thiết cho một mã QR cụ thể, [tham khảo tại error correction table](#). Tìm phiên bản và mức sửa lỗi đang được sử dụng cho mã QR đang được mã hóa và tìm số trong cột được gắn nhãn “Tổng số từ mã dữ liệu cho phiên bản và mức EC này”. Nhân số này với 8 để có được tổng số bit dữ liệu cần thiết cho phiên bản và mức sửa lỗi này.

Ví dụ, theo bảng, một mã phiên bản 1-Q có 13 từ mã dữ liệu tổng cộng. Do đó, tổng số

bit cần thiết cho mã QR này là  $13 * 8 = 104$  bit.

Thêm ký tự kết thúc 0 nếu cần

Nếu chuỗi bit ngắn hơn tổng số bit cần thiết, một ký tự kết thúc tối đa bốn 0 phải được thêm vào bên phải chuỗi. Nếu vẫn cần nhiều hơn 4 số 0, thêm bốn 0 vào cuối. Nếu chuỗi vẫn thiếu, chỉ cần thêm số lượng 0 cần thiết để đạt đến số lượng bit cần thiết.

Ví dụ, nếu mã hóa HELLO WORLD trong một mã QR phiên bản 1-Q, tổng số bit cần thiết như đã đề cập trong phần trước là 104 bit. Chuỗi bit dữ liệu dài 74 bit. Ký tự kết thúc chỉ có thể dài tối đa 4 bit, vì vậy thêm bốn 0 vào bên phải chuỗi. Chuỗi kết quả vẫn quá ngắn để điền đầy cần 104 bit, nhưng quy định mã QR yêu cầu ký tự kết thúc có độ dài tối đa là bốn 0.

Đây là chuỗi ví dụ HELLO WORLD với ký tự kết thúc được thêm vào:

Chỉ số Mode	Chỉ số đếm kí tự	Dữ liệu đã mã hóa	Kí tự kết thúc
0010	000001011	01100001011(HE) 01111000110(LL) 10001011100 (O ) 10110111000(WO) 10011010100(RL) 001101(D)	0000

Bảng 5 Chuỗi nhị phân giá trị HELLO WORLD được thêm ký tự kết thúc

*Thêm nhiều số 0 để biến độ dài thành bội số của 8*

Sau khi thêm ký tự kết thúc, nếu số lượng bit trong chuỗi không phải là bội số của 8, trước tiên thêm 0 vào bên phải chuỗi để làm cho độ dài chuỗi là bội số của 8.

Ví dụ, sau khi thêm ký tự kết thúc vào chuỗi HELLO WORLD, độ dài trở thành 78 bit. Đây không phải là bội số của 8. Chuỗi bit được hiển thị ở đây được chia thành các byte nhị phân 8 bit:

00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101  
01000011 010000

Có sáu bit cuối cùng. Thêm hai 0 để tạo thành một byte nhị phân 8 bit:

00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101  
01000011 01000000

*Thêm byte đệm nếu chuỗi vẫn quá ngắn*

Nếu chuỗi vẫn chưa đủ dài để điền đầy dung lượng tối đa, thêm các byte sau vào cuối

chuỗi, lặp lại cho đến khi chuỗi đạt độ dài tối đa: 11101100 00010001

Các byte này tương đương với 236(11101100) và 17(00010001). Chúng được yêu cầu cụ thể bởi quy định mã QR để được thêm vào nếu chuỗi bit quá ngắn ở giai đoạn này.

Ví dụ, chuỗi HELLO WORLD trên có độ dài 80 bit. Dung lượng cần thiết cho một mã 1-Q, như đã nói trước đó trên trang, là 104 bit. Số lượng bit phải được thêm vào để điền đầy dung lượng còn lại là  $104 - 80$ , hoặc 24. Chia số này cho 8:  $24 / 8 = 3$ . Do đó, ba byte đệm phải được thêm vào cuối chuỗi dữ liệu. Điều này được hiển thị dưới đây: 00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 01000000 11101100 00010001 11101100

## 2.4. Error Correction Coding

### *Bước 1: Chia từ mã dữ liệu thành các khối nếu cần*

Trước khi tạo từ mã sửa lỗi, nếu mã QR có version lớn hơn 2, phải chia dữ liệu thành các khối nhỏ hơn. Ở đây sẽ làm với mã 5-Q.

Thông tin về từ mã sửa lỗi và khối tại [đây](#).

Bảng sửa lỗi đề cập đến "group 1" và "group 2", cũng như "số lượng khối". Điều này có nghĩa là các từ mã dữ liệu phải được chia thành tối đa hai nhóm và trong mỗi nhóm, các từ mã dữ liệu có thể được chia thành các khối. Các từ mã dữ liệu được chia theo thứ tự (tức là bắt đầu với từ mã 1, sau đó là từ mã 2 và cứ thế.)

Đối với một mã 5-Q, nó nói rằng có hai nhóm, nhóm đầu tiên phải được chia thành 2 khối mỗi khối chứa 15 từ mã dữ liệu và nhóm thứ hai phải được chia thành 2 khối mỗi khối chứa 16 từ mã dữ liệu. Lưu ý rằng  $15 + 15 + 16 + 16 = 62$ , đó là tổng số từ mã dữ liệu. Dưới đây là các từ mã trên, được chia thành các nhóm và khối chính xác để minh họa cách thực hiện bước này.

Như vậy có bảng sau với mẫu

Group Number	Block Number	DaCodewords in the Group	Group Number	Block Number	DaCodewords in the Group
Group 1	Block 1	01000011	Group 2	Block 1	10110110
		01010101			11100110
		01000110			11110111
		10000110			01110111

		01010111			00110010
		00100110			00000111
		01010101			01110110
		11000010			10000110
		01110111			01010111
		00110010			00100110
		00000110			01010010
		00010010			00000110
		00000110			10000110
		01100111			10010111
		00100110			00110010
					00000111
	Block 2	11110110		Block 2	01000110
		11110110			11110111
		01000010			01110110
		00000111			01010110
		01110110			11000010
		10000110			00000110
		11110010			10010111
		00000111			00110010
		00100110			11100000
		01010110			11101100
		00010110			00010001
		11000110			11101100
		11000111			00010001



		10010010 00000110			11101100 00010001 11101100
--	--	----------------------	--	--	----------------------------------

Bảng 6 Bảng mẫu dữ liệu mã 5-Q

Trong bảng, hãy chú ý rằng các khối trong nhóm 1 bao gồm 15 từ mã dữ liệu và các khối trong nhóm 2 bao gồm 16 từ mã dữ liệu như được quy định trong bảng sửa lỗi. Cũng lưu ý rằng chúng được chia theo thứ tự. Tức là, chúng có cùng thứ tự như trước khi được tách thành các khối.

Tìm thông tin mã 5-Q trong bảng

Versio n và EC Level	Tổng số dacodewor d cho Version và EC level	EC code word trong mỗi Bloc k	Số lượng Block trong Grou p 1	Số lượng dacodewor d trong mỗi khối của group 1	Số lượng Block trong Grou p 2	Số lượng dacodewor d trong mỗi khối của group 2	Tổng số dacodewor d
5-Q	62	18	2	15	2	16	$(15*2) + (16*2) = 62$

Bảng 7 Thông tin mã 5-Q

Có 18 từ mã sửa lỗi cho mỗi khối. Trong ví dụ này, có bốn khối, vì vậy sẽ có bốn tập hợp 18 từ mã sửa lỗi, tổng cộng là 72 từ mã sửa lỗi.

Như đã hiển thị trong bảng sửa lỗi, các mã QR nhỏ hơn không yêu cầu các từ mã dữ liệu phải được chia tách hoàn toàn, vì vậy, đối với một mã 1-M (ví dụ) tất cả 16 từ mã dữ liệu sẽ được sử dụng làm một khối duy nhất và chỉ cần tạo ra 10 từ mã sửa lỗi.

Trước khi tiếp tục ví dụ 5-Q, cần giải thích các bước cơ bản của sửa lỗi Reed-Solomon.

### ***Bước 2: Hiểu cách chia đa thức dài***

$$\begin{array}{r}
 x^3 - x^2 - 7x + 3 \quad | \quad x - 3 \\
 \underline{-(x^3 - 3x^2)} \phantom{+ 3} \\
 2x^2 - 7x + 3 \\
 \underline{-(2x^2 - 6x)} \phantom{+ 3} \\
 -x + 3 \\
 \underline{-(-x + 3)} \\
 0
 \end{array}$$

Hình 18 Chia đa thức cho đa thức

### ***Bước 3: Hiểu về Trường Galois***

Như đã đề cập trong phần trước, để tạo ra các từ mã sửa lỗi, quá trình này sử dụng một phương pháp gọi là sửa lỗi Reed-Solomon. Cùng với phép chia đa thức dài, phương pháp này sử dụng một trường Galois, đó là một tập hợp hạn chế các số, cũng như một số phép toán toán học tạo ra các số vẫn nằm trong tập hợp đó.

Tiêu chuẩn mã QR nói rằng sử dụng số học modulo 2 theo bit và số học modulo 100011101 theo byte. Điều này có nghĩa là sử dụng Trường Galois 28, hay nói cách khác Trường Galois 256, đôi khi được viết là GF(256).

Các số trong GF(256) sẽ nằm trong khoảng từ 0 đến 255 (bao gồm cả hai). Lưu ý rằng đây là cùng một khoảng số có thể được biểu diễn bằng một byte tám bit (byte tám bit lớn nhất có thể là 11111111, tương đương với 255).

Điều này có nghĩa là tất cả các phép toán toán học trong GF(256) sẽ cho ra kết quả là các số có thể được biểu diễn dưới dạng byte tám bit.

GF(256) chứa các số từ 0 đến 255 (bao gồm cả hai). Các phép toán toán học trong GF(256) có tính chu kỳ, có nghĩa là nếu một phép toán toán học được thực hiện trong GF(256) mà kết quả là một số lớn hơn 255, sẽ cần sử dụng phép chia lấy dư để có được một số vẫn nằm trong Trường Galois.

Trong Trường Galois, các số âm có cùng giá trị với các số dương, vì vậy  $-n = n$ . Nói cách khác, luôn sử dụng giá trị tuyệt đối của các số liên quan đến số học Trường Galois.

Điều này có nghĩa là phép cộng và phép trừ trong Trường Galois là cùng một thứ. Phép cộng và phép trừ trong Trường Galois được thực hiện bằng cách thực hiện phép cộng hoặc phép trừ như bình thường, nhưng sau đó thực hiện phép chia lấy dư. Và vì đang sử dụng số học modulo 2 theo bit (như đã đề cập trong quy định mã QR), điều này giống

như thực hiện phép XOR. Ví dụ:

$$1 + 1 = 2 \% 2 = 0$$

hoặc

$$1 \wedge 1 = 0$$

và

$$0 + 1 = 1 \% 2 = 1$$

hoặc

$$0 \wedge 1 = 1$$

Đối với mục đích mã hóa một mã QR, tất cả các phép cộng và phép trừ trong GF(256) được thực hiện bằng cách XOR hai số lại với nhau.

Bước 4: Tạo Lũy thừa của 2 sử dụng Byte-Wise modulo 100011101

Phép chia lấy dư theo Byte 100011101 Tất cả các số trong GF(256) có thể được biểu diễn dưới dạng lũy thừa của 2. Cụ thể, tất cả các số trong GF(256) có thể được biểu diễn dưới dạng  $2^n$ , trong đó  $0 \leq n < 256$ . Tuy nhiên,  $2^8$  có vẻ như quá lớn cho Trường Galois vì nó bằng 256.

Các lũy thừa của 2 từ 0 đến 8 là:

The powers of 2 from 0 to 8 are:

$$2^0 = 1$$

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8$$

$$2^4 = 16$$

$$2^5 = 32$$

$$2^6 = 64$$

$$2^7 = 128$$

$$2^8 = 256$$

Thông số QR code nói rằng sử dụng phép chia lấy dư theo byte 100011101 (i 100011101 là một số nhị phân tương đương với 285 trong hệ thập phân). Điều này có nghĩa là khi

một số là 256 hoặc lớn hơn, nó sẽ được XOR với 285. Điều này dẫn đến các giá trị bất ngờ cho  $2^8$  và lớn hơn.

Nói cách khác:  $2^8 = 256 \wedge 285 = 29$

Lưu ý rằng khi tiếp tục đến  $2^9$ , không lấy giá trị thông thường của nó là 512 và XOR với 285 (điều này sẽ dẫn đến một số quá lớn). Thay vào đó, vì  $2^9 = 2^8 * 2$ , hãy sử dụng giá trị của  $2^8$  được tính toán trong bước trước.

Do đó

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

Tiếp tục sử dụng lũy thừa trước đó của 2 để tạo ra các lũy thừa tiếp theo của 2:  $2^{10} = 2^9 * 2 = 58 * 2 = 116$   $2^{11} = 2^{10} * 2 = 116 * 2 = 232$

Bất cứ khi nào có giá trị lớn hơn hoặc bằng 256 được thu được, một lần nữa XOR với 285:  $2^{12} = 2^{11} * 2 = 232 * 2 = 464 \wedge 285 = 205$

Nói chung, các giá trị luôn bằng với hai lần lũy thừa trước đó và nếu giá trị đó là 256 hoặc lớn hơn, nó sẽ được XOR với 285.

Sử dụng quy trình này, tất cả các số trong GF(256) có thể được biểu diễn với  $2^n$ , trong đó n là một số trong phạm vi từ 0 đến 255.

#### ***Bước 4: Hiểu về đa thức sinh***

Mã hóa sửa lỗi sử dụng phép chia đa thức dài. Để làm điều đó, cần hai đa thức. Đa thức đầu tiên được sử dụng được gọi là đa thức thông điệp. Đa thức thông điệp sử dụng các từ mã dữ liệu từ bước mã hóa dữ liệu làm hệ số của nó. Ví dụ, nếu các từ mã dữ liệu, được chuyển đổi thành số nguyên, là 25, 218 và 35, thì đa thức thông điệp sẽ là  $25x^2 + 218x + 35$ . Trong thực tế, các đa thức thông điệp thực tế cho các mã QR chuẩn rất dài hơn nhiều.

Đa thức thông điệp sẽ được chia cho một đa thức sinh. Đa thức sinh là một đa thức được tạo ra bằng cách nhân

$$(x - \alpha^0) \dots (x - \alpha^{n-1})$$

trong đó n là số lượng các từ mã sửa lỗi phải được tạo ra (xem bảng sửa lỗi). Như đã đề cập trong phần trước,  $\alpha$  (alpha) bằng 2.

Thông số QR code liệt kê các đa thức sinh trong Phụ lục A, bắt đầu từ 2 và kết thúc với 68. Mặc dù các mã QR chuẩn luôn yêu cầu nhiều hơn 2 từ mã sửa lỗi cho mỗi khối, trang này sẽ chỉ ra cách tính toán đa thức sinh cho 2 từ mã sửa lỗi, vì nó minh họa quá trình

tính toán tất cả các đa thức sinh còn lại.

Ví dụ có đa thức thông điệp

$$x^2 + 3x^1 + 2x^0$$

Thì đa thức sinh với  $\alpha$  có dạng

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

Tra thêm thông tin tại [bảng](#).

### ***Bước 5: Tạo Error Correction Codewords***

*Đa thức thông điệp*

Dacodeword của HELLO WORLD dạng 1-M như sau

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

Chuyển từ nhị phân sang dạng thập phân

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

Tiếp đó tạo đa thức thông điệp

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

Tạo đa thức sinh

1-M code chỉ cần 10 error correction codewords. Do đó đa thức sinh có dạng

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

*Chia đa thức thông điệp cho đa thức sinh*

Bây giờ là lúc chia đa thức thông điệp cho đa thức sinh. Điều này được thực hiện gần giống như phép chia đa thức dài được hiển thị trước đó trên trang. Dưới đây là các bước chia, được cập nhật để tính toán số học Trường Galois:

1. Tìm nhân số phù hợp để nhân đa thức sinh. Kết quả của phép nhân nên có cùng thuật ngữ đầu tiên với đa thức thông điệp (trong bước nhân đầu tiên) hoặc phần dư (trong tất cả các bước nhân tiếp theo).
2. XOR kết quả với đa thức thông điệp (trong bước nhân đầu tiên) hoặc phần dư (trong tất cả các bước nhân tiếp theo).

3. Thực hiện các bước này n lần, trong đó n là số lượng các từ mã dữ liệu.

Chú ý sự khác biệt giữa các bước này và các bước của phép chia đa thức dài bình thường. Thay vì trừ sau bước nhân, thực hiện một XOR (điều này, trong GF(256), là cùng một điều).

Quan trọng hơn, sau khi chia hai đa thức, sẽ có một phần dư. Các hệ số của phần dư này là các từ mã sửa lỗi.

Phép chia sẽ được minh họa từng bước trong phần tiếp theo.

Giờ sẽ tiến hành chia như sau

Bước đầu tiên của phép chia là chuẩn bị đa thức thông điệp cho phép chia. Đa thức thông điệp đầy đủ là:

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

Để đảm bảo rằng số mũ của thuật ngữ dẫn không trở nên quá nhỏ trong quá trình chia, nhân đa thức thông điệp với  $x^n$  trong đó n là số lượng các từ mã sửa lỗi cần thiết. Trong trường hợp này n là 10, cho 10 từ mã sửa lỗi, vì vậy nhân đa thức thông điệp với  $x^{10}$ , điều này:

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

Thuật ngữ dẫn của đa thức sinh cũng nên có cùng số mũ, vì vậy nhân với  $x^{15}$  để được

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

Bây giờ có thể thực hiện các bước chia lặp đi lặp lại. Số lượng bước trong phép chia phải bằng số lượng các thuật ngữ trong đa thức thông điệp. Trong trường hợp này, phép chia sẽ mất 16 bước để hoàn thành. Điều này sẽ dẫn đến một phần dư có  $26 - 16 = 10$  thuật ngữ. Những thuật ngữ này sẽ là các từ mã sửa lỗi cần thiết.

#### Bước 5.1a: Nhân đa thức sinh với số hạng đầu của đa thức thông điệp

Hạng tử dẫn trong trường hợp này là  $32x^{25}$ . Vì ký hiệu alpha giúp thực hiện phép nhân dễ dàng hơn, nên nó được khuyến nghị chuyển đổi  $32x^{25}$  sang ký hiệu alpha. Theo bảng log antilog, với giá trị nguyên 32, số mũ alpha là 5. Vì vậy  $32 = \alpha^5$ . Nhân đa thức sinh bởi  $\alpha^5$ :

Các số mũ của các alpha được cộng lại với nhau. Trong trường hợp này, ít nhất một trong

các số mũ lớn hơn 255, vì vậy với những số mũ lớn hơn 255 thực hiện modulo 255:

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

Kết quả là:

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

Bây giờ, chuyển đổi sang ký hiệu nguyên:

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

Bước 5.1b: XOR kết quả với đa thức thông điệp

Từ kết quả của bước trên, XOR kết quả đó với đa thức thông điệp

$$(32 \oplus 32)x^{25} + (91 \oplus 2)x^{24} + (11 \oplus 101)x^{23} + (120 \oplus 10)x^{22} + (209 \oplus 97)x^{21} + (114 \oplus 197)x^{20} + (220 \oplus 15)x^{19} + (77 \oplus 47)x^{18} + (67 \oplus 134)x^{17} + (64 \oplus 74)x^{16} + (236 \oplus 5)x^{15} + (17 \oplus 0)x^{14} + (236 \oplus 0)x^{13} + (17 \oplus 0)x^{12} + (236 \oplus 0)x^{11} + (17 \oplus 0)x^{10}$$

được kết quả

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

Loại bỏ 0

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

Bước 5.2a: Nhân đa thức sinh với số hạng đầu của kết quả phép XOR ở bước trước

Số hạng đầu là  $89x^{24}$ . Chuyển sang dạng  $\alpha$ , tra [bảng](#) thấy  $89 = \alpha^{210}$ .

Tiến hành nhân

$$(\alpha^{210} * \alpha^0)x^{24} + (\alpha^{210} * \alpha^{251})x^{23} + (\alpha^{210} * \alpha^{67})x^{22} + (\alpha^{210} * \alpha^{46})x^{21} + (\alpha^{210} * \alpha^{61})x^{20} + (\alpha^{210} * \alpha^{118})x^{19} + (\alpha^{210} * \alpha^{70})x^{18} + (\alpha^{210} * \alpha^{64})x^{17} + (\alpha^{210} * \alpha^{94})x^{16} + (\alpha^{210} * \alpha^{32})x^{15} + (\alpha^{210} * \alpha^{45})x^{14}$$

Với những số mũ lớn hơn 255 tiến hành mod với 255

$$\alpha^{210}x^{24} + \alpha^{(461 \% 255)}x^{23} + \alpha^{(277 \% 255)}x^{22} + \alpha^{(256 \% 255)}x^{21} + \alpha^{(271 \% 255)}x^{20} + \alpha^{(328 \% 255)}x^{19} + \alpha^{280 \% 255}x^{18} + \alpha^{(274 \% 255)}x^{17} + \alpha^{304 \% 255}x^{16} + \alpha^{242}x^{15} + \alpha^{255}x^{14}$$

Kết quả thu được

$$\alpha^{210}x^{24} + \alpha^{206}x^{23} + \alpha^{22}x^{22} + \alpha^1x^{21} + \alpha^{16}x^{20} + \alpha^{73}x^{19} + \alpha^{25}x^{18} + \alpha^{19}x^{17} + \alpha^{304 \% 255}x^{16} + \alpha^{242}x^{15} + \alpha^{255}x^{14}$$

Chuyển sang kí hiệu số

$$89x^{24} + 83x^{23} + 234x^{22} + 2x^{21} + 76x^{20} + 202x^{19} + 3x^{18} + 90x^{17} + 140x^{16} + 176x^{15} + 1x^{14}$$

Bước 5.2b: XOR kết quả với kết quả bước 1b

$$(89 \oplus 89)x^{24} + (110 \oplus 83)x^{23} + (114 \oplus 234)x^{22} + (176 \oplus 2)x^{21} + (183 \oplus 76)x^{20} + (211 \oplus 202)x^{19} + (98 \oplus 3)x^{18} + (197 \oplus 90)x^{17} + (10 \oplus 140)x^{16} + (233 \oplus 176)x^{15} + (17 \oplus 1)x^{14} + (236 \oplus 0)x^{13} + (17 \oplus 0)x^{12} + (236 \oplus 0)x^{11} + (17 \oplus 0)x^{10}$$

Kết quả thu về sau khi bỏ đi số hạng bằng 0

$$61x^{23} + 152x^{22} + 178x^{21} + 251x^{20} + 25x^{19} + 97x^{18} + 159x^{17} + 134x^{16} + 89x^{15} + 16x^{14} + 236x^{13} + 17x^{12} + 236x^{11} + 17x^{10}$$

Cứ như vậy lặp thêm  $16 - 2 = 14$  bước nữa, mỗi bước gồm nhân đa thức thu được với số hạng đầu của kết quả phép XOR của bước trước đó, sau đó XOR kết quả thu được với kết quả của phép XOR bước trước.

Kết quả cuối cùng thu được có dạng

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

Từ kết quả đa thức cuối cùng, sẽ lấy các hạng tử của phần dư làm mã sửa lỗi. Như vậy thu được mã sửa lỗi như sau:

196 35 39 119 235 215 231 226 93 23

## 2.5. Structure Final Message

***Bước 1: Xác định số lượng khối và mã sửa lỗi cần thiết***

Bảng sửa lỗi cho thấy số lượng khối dữ liệu và mã sửa lỗi cho mỗi khối được yêu cầu cho mỗi phiên bản và mức độ sửa lỗi.



*Đối với các mã nhỏ hơn, sử dụng dữ liệu và mã sửa lỗi như thế nào*

Xin lưu ý rằng các mã QR nhỏ chỉ bao gồm một khối của các mã dữ liệu, với một tập hợp các mã sửa lỗi cho khối đó. Trong trường hợp này, không cần xen kẽ. Chỉ cần đặt các mã sửa lỗi sau các mã dữ liệu và chuyển sang bước tiếp theo, đặt module trong ma trận.

*Chia nhỏ các mã QR lớn hơn*

Trên trang mã hóa sửa lỗi, bảng sửa lỗi nói rằng nhóm đầu tiên của mã 5-Q bao gồm 2 khối, với 15 mã dữ liệu cho mỗi khối, và nhóm thứ hai bao gồm 2 khối, với 16 mã dữ liệu cho mỗi khối. Đây lại là bảng được hiển thị trên trang mã hóa sửa lỗi đã chỉ ra mã 5-Q được chia thành các nhóm và khối.

Group Number	Block Number	DaCodewords in the Group	Group Number	Block Number	DaCodewords in the Group
Group 1	Block 1	01000011	Group 2	Block 1	10110110
		01010101			11100110
		01000110			11110111
		10000110			01110111
		01010111			00110010
		00100110			00000111
		01010101			01110110
		11000010			10000110
		01110111			01010111
		00110010			00100110
		00000110			01010010
		00010010			00000110
		00000110			10000110
		01100111			10010111
		00100110			00110010
					00000111

	Block 2	11110110		Block 2	01000110
		11110110			11110111
		01000010			01110110
		00000111			01010110
		01110110			11000010
		10000110			00000110
		11110010			10010111
		00000111			00110010
		00100110			11100000
		01010110			11101100
		00010110			00010001
		11000110			11101100
		11000111			00010001
		10010010			11101100
		00000110			00010001
					11101100

Bảng 8 Chia dữ liệu mã 5-Q vào các khối

Lưu ý rằng bảng sửa lỗi cho biết đối với mỗi khối, mã 5-Q phải có 18 mã sửa lỗi. Bây giờ khi đã giải thích về mã hóa sửa lỗi, các mã sửa lỗi cho dữ liệu trên có thể được tính toán.

Có bốn khối, vì vậy nên tạo ra bốn tập hợp 18 mã sửa lỗi. Cập nhật cho thấy các mã sửa lỗi dữ liệu của mỗi khối được chuyển đổi thành một danh sách các số nguyên và 18 mã sửa lỗi được tạo ra cho mỗi khối.

Group Number	Block Number	DaCodewords in the Group	DaCodewords dạng số nguyên
			Error Correction Codewords cho mỗi khối
Group 1	Block 1	01000011	DaCodewords từ trái sang phải dưới dạng số:

		01010101 01000110 10000110 01010111 00100110 01010101 11000010 01110111 00110010 00000110 00010010 00000110 01100111 00100110	67,85,70,134,87,38,85,194,119,50,6,18,6,103,38  Error Correction Codewords (tính tại <a href="#">đây</a> ): 213 199 11 45 115 247 241 223 229 248 154 117 154 111 86 161 111 39
	Block 2	11110110 11110110 01000010 00000111 01110110 10000110 11110010 00000111 00100110 01010110 00010110 11000110 11000111 10010010 00000110	DaCodewords từ trái sang phải dưới dạng số: 246,246,66,7,118,134,242,7,38,86,22,198,199,146,6  Error Correction Codewords : 87 204 96 60 202 182 124 157 200 134 27 129 209 17 163 163 120 133
Group Number	Block Number	DaCodewords in the Group	DaCodewords dạng số nguyên
			Error Correction Codewords cho mỗi khối
Group 2	Block 1	10110110	DaCodewords từ trái sang phải dưới dạng số:

		11100110 11110111 01110111 00110010 00000111 01110110 10000110 01010111 00100110 01010010 00000110 10000110 10010111 00110010 00000111	182,230,247,119,50,7,118,134,87,38,82,6,134,151,50,7  Error Correction Codewords : 148 116 177 212 76 133 75 242 238 76 195 230 189 10 108 240 192 141
	Block 2	01000110 11110111 01110110 01010110 11000010 00000110 10010111 00110010 11100000 11101100 00010001 11101100 00010001 11101100 00010001 11101100	DaCodewords từ trái sang phải dưới dạng số: 70,247,118,86,194,6,151,50,16,236,17,236,17,236,17,236  Error Correction Codewords : 235 159 5 173 24 147 59 33 106 40 255 172 82 2 131 32 178 236

Bảng 9 Mã sửa lỗi tương ứng của mỗi khối

### ***Bước 2: Xen kẽ các khối***

Các khối được xen kẽ bằng cách thực hiện như sau:

1. lấy mã dữ liệu đầu tiên từ khối đầu tiên
2. tiếp theo là mã dữ liệu đầu tiên từ khối thứ hai
3. tiếp theo là mã dữ liệu đầu tiên từ khối thứ ba
4. tiếp theo là mã dữ liệu đầu tiên từ khối thứ tư
5. tiếp theo là mã dữ liệu thứ hai từ khối đầu tiên
6. và cứ thế

Mẫu này được lặp lại, đi qua các khối, cho đến khi tất cả các mã dữ liệu đã được xen kẽ.

Sau đó, làm như sau:

1. lấy mã sửa lỗi đầu tiên từ khối đầu tiên
2. tiếp theo là mã sửa lỗi đầu tiên từ khối thứ hai
3. tiếp theo là mã sửa lỗi đầu tiên từ khối thứ ba
4. tiếp theo là mã sửa lỗi đầu tiên từ khối thứ tư
5. tiếp theo là mã sửa lỗi thứ hai từ khối đầu tiên
6. và cứ thế

Làm điều này cho đến khi tất cả các mã sửa lỗi đã được sử dụng hết.

Quá trình xen kẽ mã dữ liệu được chi tiết hóa dưới đây.

	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10	Col 11	Col 12	Col 13	Col 14	Col 15	Col 16
Block 1	67	85	70	134	87	38	85	194	119	50	6	18	6	103	38	
Block 2	246	246	66	7	118	134	242	7	38	86	22	198	199	146	6	
Block 3	182	230	247	119	50	7	118	134	87	38	82	6	134	151	50	7
Block 4	70	247	118	86	194	6	151	50	16	236	17	236	17	236	17	236

Bảng 10 Quá trình xen kẽ mã dữ liệu

Như đã mô tả ở trên, lấy mã dữ liệu đầu tiên từ khối đầu tiên, tiếp theo là mã dữ liệu đầu tiên từ khối thứ hai, tiếp theo là mã dữ liệu đầu tiên từ khối thứ ba, cuối cùng là mã dữ liệu đầu tiên từ khối thứ tư.

Hoặc, như được hiển thị trong bảng trên, lấy các mã dữ liệu từ cột 1, bắt đầu từ khối 1 và kết thúc với khối 4. Các số từ cột 1 là 67, 246, 182 và 70.

Sau đó, đặt các số từ cột 2, bắt đầu từ khối 1 và kết thúc với khối 4. Dữ liệu xen kẽ cho đến nay:

67, 246, 182, 70, 85, 246, 230, 247

Làm tương tự với cột 3.

Dữ liệu xen kẽ :

67, 246, 182, 70, 85, 246, 230, 247, 70, 66

Cột số 16 chỉ còn 2 số nên chỉ cần đặt 2 số này vào cuối dữ liệu xen kẽ.

Tương tự như vậy với xen kẽ mã lỗi

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
B1	213	199	11	45	115	247	241	223	229	248	154	117	154	111	86	161	111	39
B2	87	204	96	60	202	182	124	157	200	134	27	129	209	17	163	163	120	133
B3	148	116	177	212	76	133	75	242	238	76	195	230	189	10	108	240	192	141
B4	235	159	5	173	24	147	59	33	106	40	255	172	82	2	131	32	178	236

Bảng 11 Quá trình xen kẽ mã lỗi

Tiến hành đặt mã lỗi đã xen kẽ đằng sau mã dữ liệu đã được xen kẽ thu được thông điệp cuối

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 sang dạng nhị phân***

Chuyển các số trong thông điệp thành dạng nhị phân 8 bit

Ví dụ với 4 số đầu:

67 = 01000011

246 = 11110110

182 = 10110110

70 = 01000110

Sau đó chỉ cần ghép chúng lần lượt theo thứ tự xuất hiện

01000011111101101011011001000110

Tương có được thông điệp cuối dạng nhị phân

010000111111011010110110010001100101010111110110111001101111011101000110  
010000101111011101110110100001100000011101110111010101100101011101110110  
001100101100001000100110100001100000011100000110010101011111001001110110  
100101111100001000000111100001100011001001110111001001100101011100010000  
001100100101011000100110111011000000011000010110010100100001000100010010  
110001100000011011101100000001101100011110000110000100010110011110010010  
100101111110110000100110000001100011001000010001000001111110110011010101  
010101111001010011101011110001111100110001110100100111110000101101100000  
101100010000010100101101001111001101010010101101011100111100101001001100  
000110001111011110110110100001011001001111110001011111000100101100111011  
110111111001110111110010001000011110010111001000111011100110101011111000  
100001100100110000101000100110100001101111000011111111110111010110000001  
111001101010110010011010110100011011110101010010011011110001000100001010  
000000100101011010100011011011001000001110100001101000111111000000100000  
0110111101111000110000001011001000100111100001011000110111101100

#### ***Bước 4: Thêm các bit dư nếu cần***

Đối với một số phiên bản QR, thông điệp nhị phân cuối cùng không đủ dài để điền vào số lượng bit yêu cầu. Trong trường hợp này, cần thêm một số lượng nhất định các số 0 vào cuối thông điệp cuối cùng để làm cho nó có độ dài chính xác. Những số 0 thêm vào này được gọi là các bit dư. Một mã QR phiên bản 5, như trong ví dụ này, phải có 7 bit dư được thêm vào cuối.

8 bit cuối cùng từ chuỗi trong hộp văn bản ở trên là:

11101100

Với 7 bit dư được thêm vào cuối, các bit cuối cùng của chuỗi là:

111011000000000

Dưới đây là danh sách phiên bản và yêu cầu số bit dư

QR Version	1	2	3	4	5	6	7	8	9	10
Số bit dư cần	0	7	7	7	7	7	0	0	0	0
QR Version	11	12	13	14	15	16	17	18	19	20
Số bit dư cần	0	0	0	3	3	3	3	3	3	3
QR Version	21	22	23	24	25	26	27	28	29	30
Số bit dư cần	4	4	4	4	4	4	4	3	3	3
QR Version	31	32	33	34	35	36	37	38	39	40
Số bit dư cần	3	3	3	3	0	0	0	0	0	0

Bảng 12 Danh sách số bit dư cần với mỗi version

## 2.6. Module Placement in Matrix

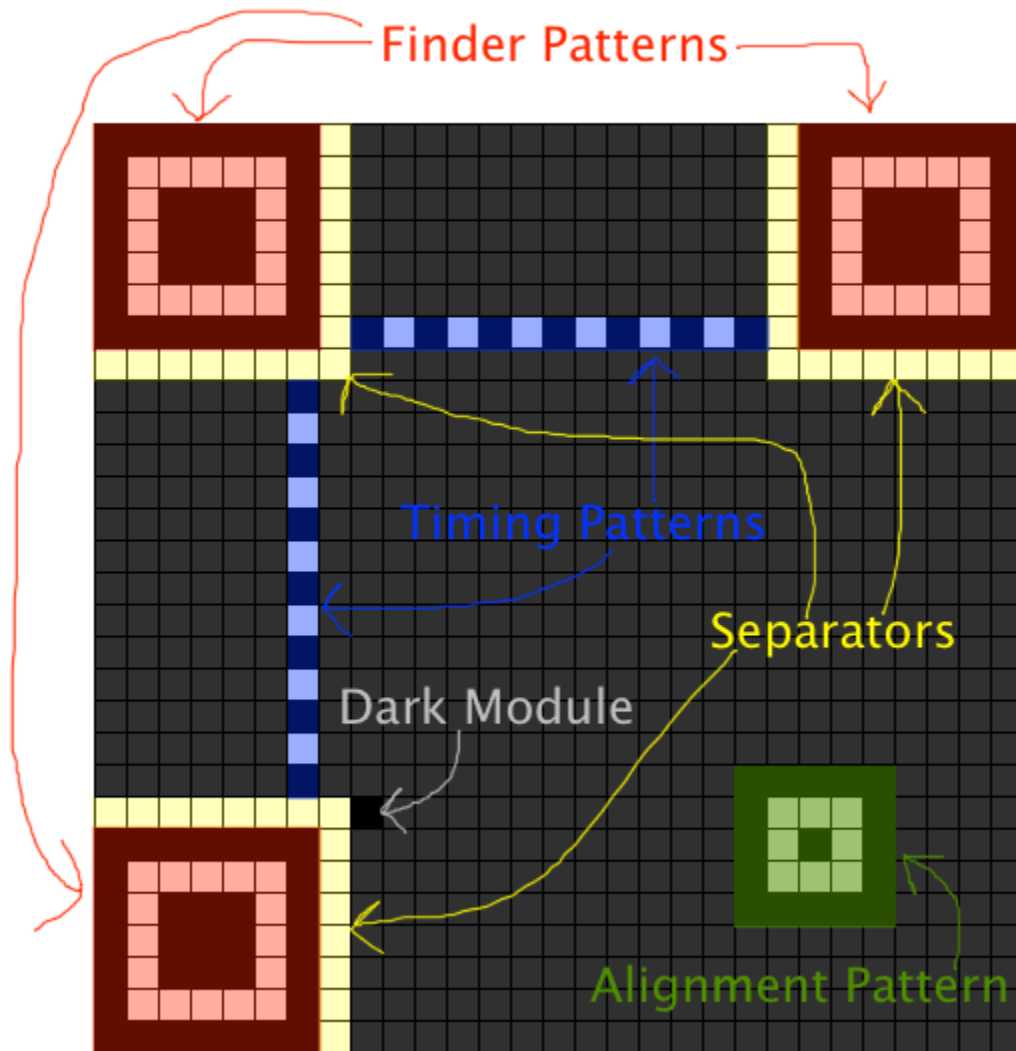
### *Pixel so với Module*

Ở đây gọi các ô vuông đen và trắng của mã QR là module thay vì pixel. Điều này nhằm phân biệt giữa các pixel trên màn hình và các ô vuông đen và trắng của mã QR. Ví dụ, một mã QR phiên bản 1 luôn có kích thước 21 module x 21 module, ngay cả khi nó chiếm 42 x 42 pixel trên màn hình máy tính, hoặc 105x105, v.v.

Tổng quan về Mẫu chức năng

Mã QR phải bao gồm các mẫu chức năng. Đây là các hình dạng phải được đặt ở các vị trí cụ thể của mã QR để đảm bảo rằng máy quét mã QR có thể xác định và định hướng chính xác mã để giải mã. Hình ảnh sau đây cho thấy ví dụ về các mẫu chức năng và vị trí của chúng.





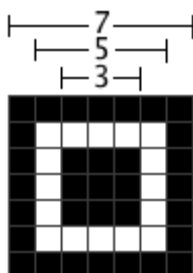
Hình 19 Cấu trúc mã QR

- Mẫu tìm kiếm là ba khối ở các góc của mã QR ở trên bên trái, trên bên phải và dưới bên trái.
- Các bộ phận tách là các khu vực không có gì bên cạnh các mẫu tìm kiếm.
- Các mẫu căn chỉnh tương tự như các mẫu tìm kiếm, nhưng nhỏ hơn và được đặt khắp mã. Chúng được sử dụng trong phiên bản 2 và lớn hơn và vị trí của chúng phụ thuộc vào phiên bản mã QR.
- Các mẫu thời gian là các đường chấm chấm kết nối các mẫu tìm kiếm.
- Mô-đun tối là một mô-đun đen đơn lẻ luôn được đặt bên cạnh mẫu tìm kiếm dưới bên trái.

Các phần dưới đây giải thích chi tiết hơn về cách định vị các mẫu chức năng.

### ***Bước 1: Thêm các Mẫu Tìm kiếm***

Đầu tiên, đưa các mẫu tìm kiếm vào ma trận. Mẫu tìm kiếm bao gồm một hình vuông đen bên ngoài có kích thước 7 mô-đun x 7 mô-đun, một hình vuông trắng bên trong có kích thước 5 mô-đun x 5 mô-đun và một hình vuông đen rắn ở trung tâm có kích thước 3 mô-đun x 3 mô-đun.

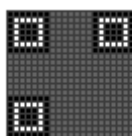


Hình 20 Mẫu tìm kiếm

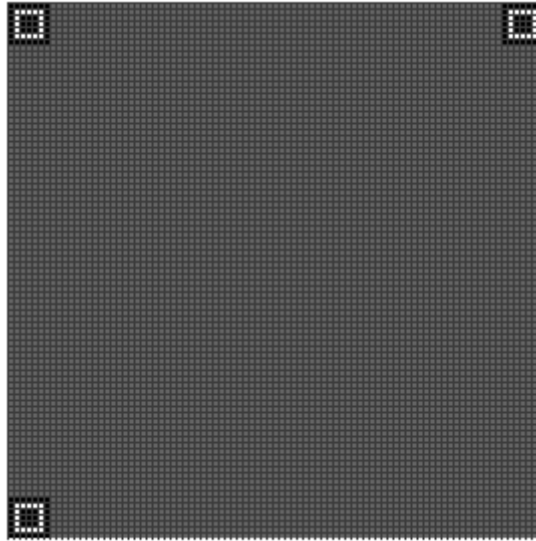
Mẫu tìm kiếm được thiết kế để là một mẫu không có khả năng xuất hiện trong các phần khác của mã QR. Các chiều rộng mô-đun của mẫu tìm kiếm có tỷ lệ 1:1:3:1:1. Máy quét mã QR có thể tìm kiếm tỷ lệ này của các mô-đun sáng và tối để phát hiện các mẫu tìm kiếm và định hướng chính xác mã QR để giải mã.

Các mẫu tìm kiếm luôn được đặt ở góc trên bên trái, trên bên phải và dưới bên trái của mã QR, không phân biệt phiên bản nào đang được sử dụng.

Để minh họa điều này, các hình ảnh sau đây cho thấy vị trí của các mẫu tìm kiếm. Hình ảnh đầu tiên là mã phiên bản 1 và hình ảnh thứ hai là mã phiên bản 18.



Hình 21 Vị trí mẫu tìm kiếm phiên bản 1



Hình 22 Vị trí mẫu tìm kiếm phiên bản 18

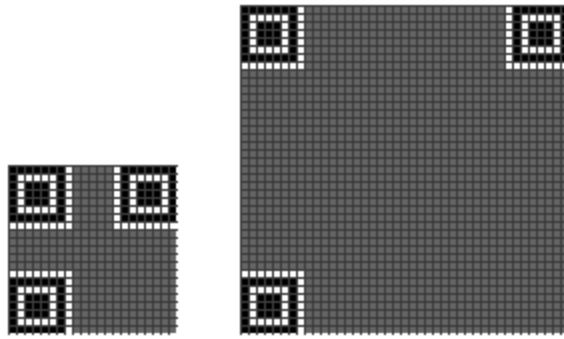
Kích thước của mã QR có thể được tính bằng công thức  $((V-1)*4)+21$ , trong đó  $V$  là phiên bản mã QR.

Ví dụ, phiên bản 32 là  $((32-1)*4)+21$  hoặc 145 mô-đun x 145 mô-đun. Do đó, vị trí của các mẫu tìm kiếm có thể được tổng quát hóa như sau:

- Góc trên bên trái của mẫu tìm kiếm trên bên trái luôn được đặt tại (0,0).
- Góc trên bên TRÁI của mẫu tìm kiếm trên bên phải luôn được đặt tại  $[(((V-1)*4)+21) - 7], 0)$
- Góc trên bên TRÁI của mẫu tìm kiếm dưới bên trái luôn được đặt tại  $(0, [(((V-1)*4)+21) - 7])$

### ***Bước 2: Thêm các Bộ phận Tách***

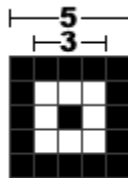
Các bộ phận tách là các dòng mô-đun trắng, rộng một mô-đun, được đặt bên cạnh các mẫu tìm kiếm để tách chúng ra khỏi phần còn lại của mã QR. Các bộ phận tách chỉ được đặt bên cạnh các cạnh của các mẫu tìm kiếm chạm vào phần trong của mã QR. Ví dụ:



Hình 23 Bộ phân tách trong QR code

### ***Bước 3: Thêm mẫu căn chỉnh***

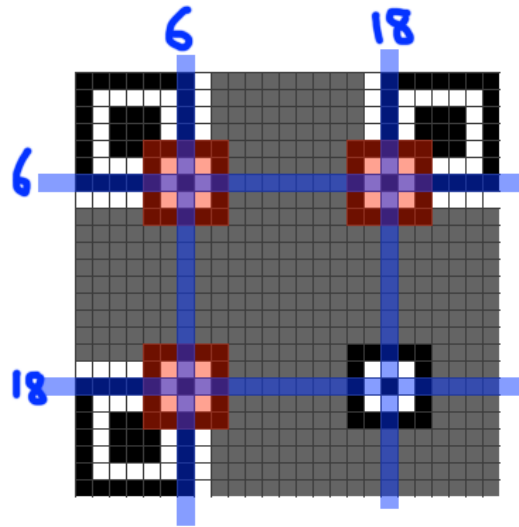
Mã QR phiên bản 2 trở lên được yêu cầu phải có các mẫu căn chỉnh. Một mẫu căn chỉnh, được hiển thị dưới đây, bao gồm một hình vuông đen 5 mô-đun x 5 mô-đun, một hình vuông trắng bên trong 3 mô-đun x 3 mô-đun và một mô-đun đen đơn lẻ ở trung tâm.



Hình 24 Mẫu căn chỉnh

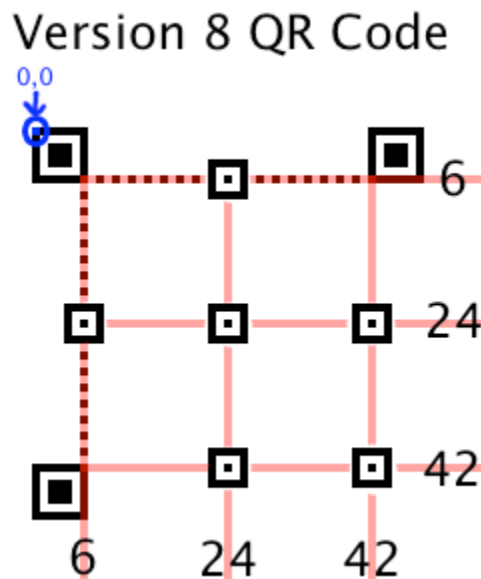
Các vị trí mà các mẫu căn chỉnh phải được đặt được xác định trong bảng vị trí mẫu căn chỉnh. Các số được sử dụng như là cả tọa độ hàng và cột. Ví dụ, Phiên bản 2 có các số 6 và 18. Điều này có nghĩa là các mô-đun trung tâm của các mẫu căn chỉnh phải được đặt tại (6, 6), (6, 18), (18, 6) và (18, 18).

Tuy nhiên, các mẫu căn chỉnh phải được đưa vào ma trận sau khi các mẫu tìm kiếm và bộ phận tách đã được đặt và các mẫu căn chỉnh không được chồng lên các mẫu tìm kiếm hoặc bộ phận tách. Các hình ảnh sau đây cho thấy mã phiên bản 2, được miêu tả trong đoạn văn trước đó là có các mẫu căn chỉnh được định tâm tại (6, 6), (6, 18), (18, 6) và (18, 18). Tuy nhiên, như hình bên trái cho thấy, các mẫu căn chỉnh được đánh dấu bằng màu đỏ không được đặt vào ma trận vì chúng chồng lên các mẫu tìm kiếm và bộ phận tách. Các mẫu căn chỉnh chồng lên các mẫu tìm kiếm hoặc bộ phận tách chỉ được loại bỏ khỏi ma trận.



Hình 25 Lỗi mẫu căn chỉnh

Để minh họa thêm về việc đặt mẫu căn chỉnh. Bảng vị trí mẫu căn chỉnh liệt kê 6, 24 và 42 là các vị trí mẫu căn chỉnh cho phiên bản 8. Tất cả các tổ hợp của ba số này được sử dụng làm tọa độ cho các mẫu căn chỉnh.

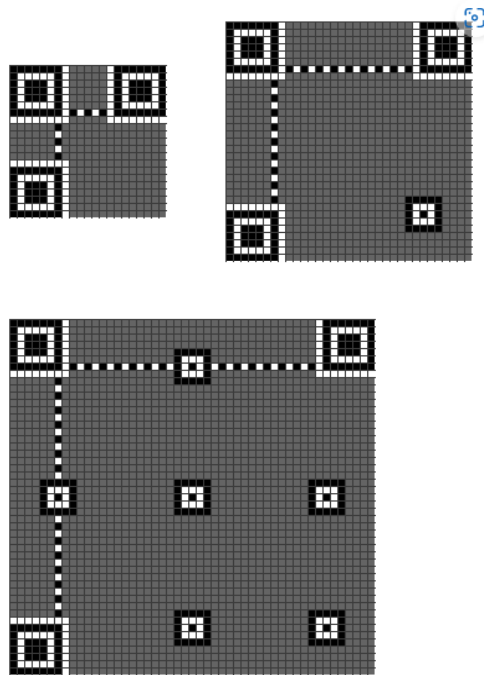


Hình 26 Các mẫu căn chỉnh cho mã QR phiên bản 8

#### ***Bước 4: Thêm mẫu Thời gian***

Mẫu thời gian là hai dòng, một ngang và một dọc, xen kẽ giữa các mô-đun tối và sáng. Mẫu thời gian ngang được đặt trên hàng thứ 6 của mã QR giữa các bộ phận tách. Mẫu thời gian dọc được đặt trên cột thứ 6 của mã QR giữa các bộ phận tách. Các mẫu thời gian luôn bắt đầu và kết thúc với một mô-đun tối. Các mẫu căn chỉnh có thể chồng lên

các mẫu thời gian vì các mô-đun sáng và tối của chúng luôn trùng với các mô-đun sáng và tối của các mẫu thời gian, như có thể thấy trong hình ảnh bên phải. Các hình ảnh sau đây hiển thị các mẫu thời gian trên các phiên bản khác nhau của mã QR.



Hình 27 Mẫu thời gian

#### ***Bước 5: Thêm Mô-đun Tối và Khu Vực Dành Riêng***

Gần đến lúc thêm các bit dữ liệu vào ma trận mã QR. Tuy nhiên, trước khi làm điều đó, mô-đun tối phải được thêm vào và có những khu vực của ma trận phải được dành riêng cho các bit định dạng và phiên bản, sẽ được thêm vào sau.

##### ***Mô-đun Tối***

mẫu tìm kiếm dưới bên trái. Cụ thể hơn, mô-đun tối luôn nằm ở tọa độ  $((4 * V) + 9, 8)$  nơi V là phiên bản của mã QR.

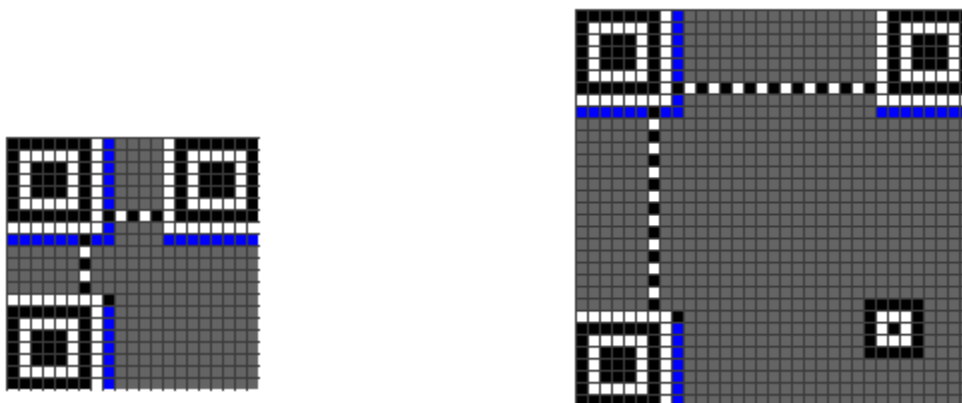
##### ***Dành Riêng Khu Vực Thông Tin Định Dạng***

Một dải các mô-đun bên cạnh các bộ phận tách biệt phải được dành riêng cho khu vực thông tin định dạng như sau:

- Gần mẫu tìm kiếm trên bên trái, một dải mô-đun phải được dành riêng dưới và bên phải của bộ phận tách biệt.

- Gần mẫu tìm kiếm trên bên phải, một dải mô-đun phải được dành riêng dưới bộ phận tách biệt.
- Gần mẫu tìm kiếm dưới bên trái, một dải mô-đun phải được dành riêng bên phải của bộ phận tách biệt.

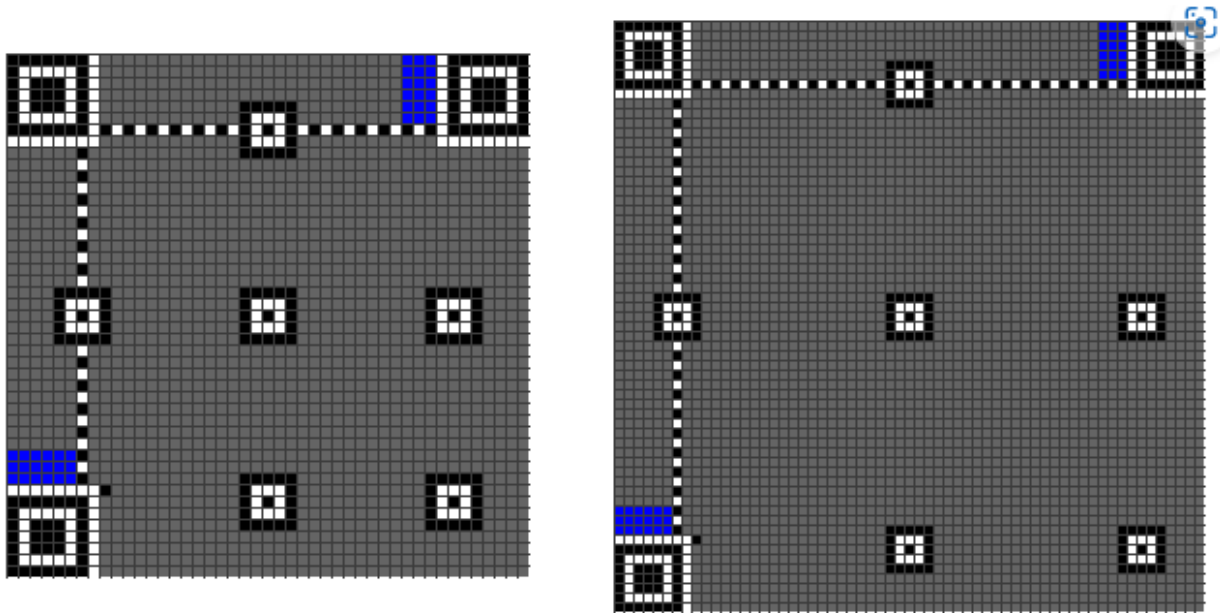
Hình ảnh sau đây cho thấy các khu vực được dành riêng trong màu xanh lam. Chúng luôn được đặt dọc theo các bộ phận tách biệt, không quan trọng phiên bản mã QR là gì.



Hình 28 Khu vực thông tin định dạng

#### *Dành Riêng Khu Vực Thông Tin Phiên Bản*

Các mã QR phiên bản 7 trở lên phải chứa hai khu vực nơi các bit thông tin phiên bản được đặt. Các khu vực là một khối 6x3 trên mẫu tìm kiếm dưới bên trái và một khối 3x6 bên trái của mẫu tìm kiếm trên bên phải. Hình ảnh sau đây cho thấy vị trí của các khu vực được dành riêng trong màu xanh lam.



Hình 29 Khu vực thông tin phiên bản

### ***Bước 6: Đặt các Bit Dữ Liệu***

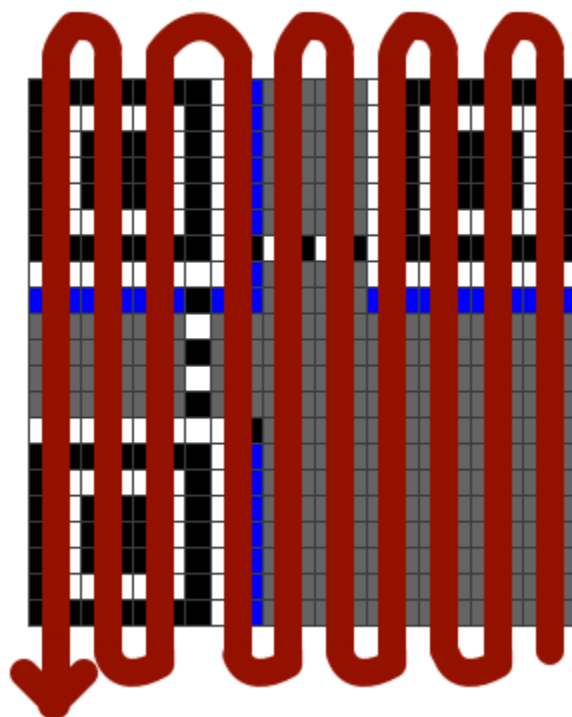
Bây giờ là lúc thêm các bit dữ liệu vào ma trận mã QR. Các bit được đặt theo một mẫu cụ thể.

#### ***Mẫu Đặt***

Các bit dữ liệu được đặt bắt đầu từ phía dưới bên phải của ma trận và tiến lên trên trong một cột rộng 2 mô-đun. Sử dụng các pixel trắng cho 0 và các pixel đen cho 1. Khi cột đó đến đỉnh, cột 2 mô-đun tiếp theo bắt đầu ngay bên trái của cột trước đó và tiếp tục xuống dưới. Bất cứ khi nào cột hiện tại đến mép của ma trận, chuyển sang cột 2 mô-đun tiếp theo và thay đổi hướng. Nếu gặp mẫu chức năng hoặc khu vực dành riêng, bit dữ liệu được đặt trong mô-đun chưa sử dụng tiếp theo.

Hình ảnh sau đây cho thấy mẫu đặt các bit dữ liệu trong mã QR. Lưu ý rằng khi đạt được mẫu thời gian dọc, cột tiếp theo bắt đầu bên trái của nó.

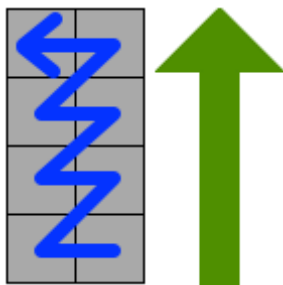




Hình 30 Hướng đặt dữ liệu tổng quát

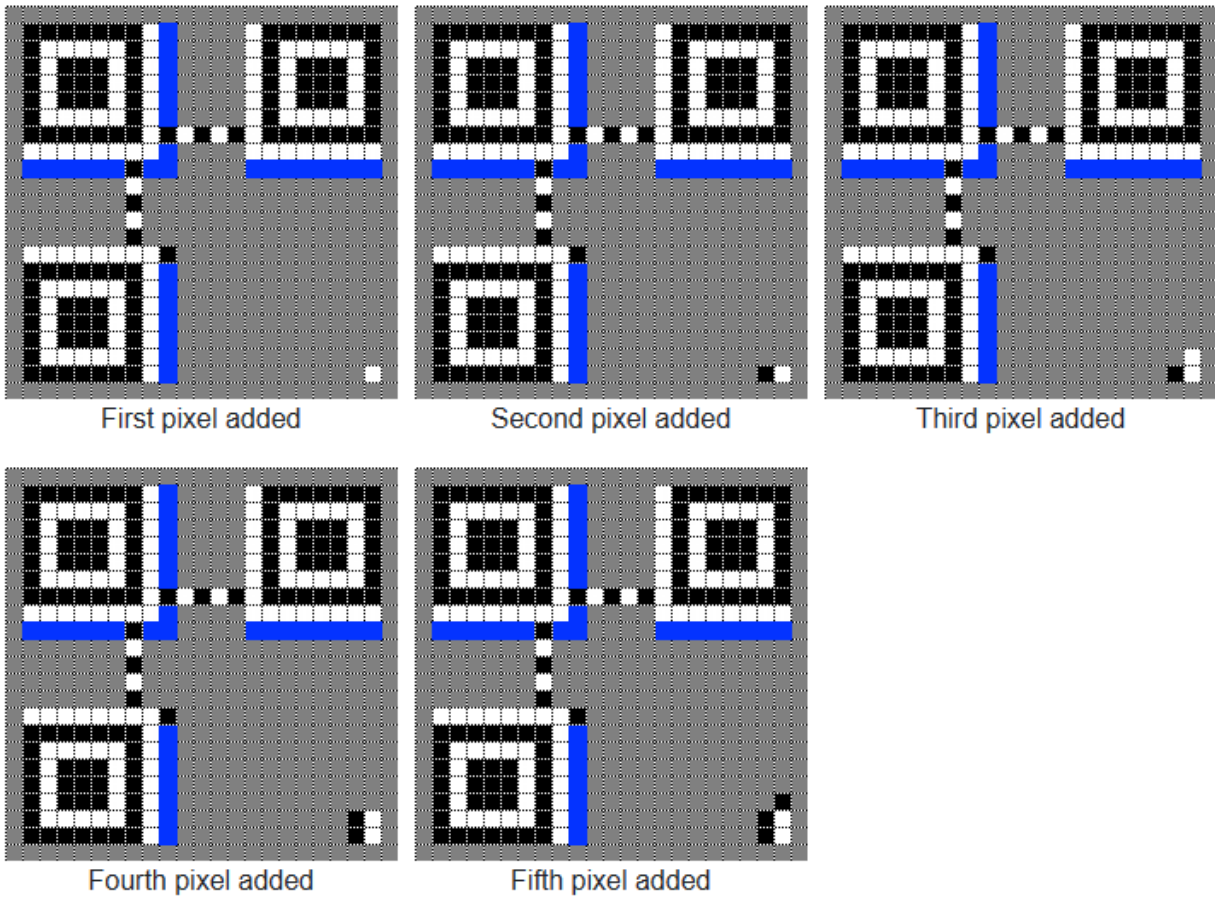
*Vị trí hướng lên*

Hình ảnh sau đây cho thấy thứ tự đặt các bit dữ liệu khi cột đi lên.



Hình 31 Cách đặt dữ liệu khi cột đi lên

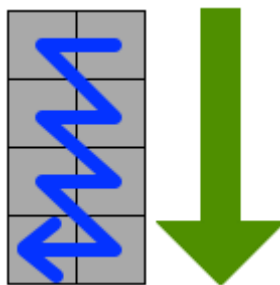
Ví dụ sau minh họa vị trí của các bit trong cột hướng lên đầu tiên.



Hình 32 Minh họa đặt dữ liệu khi cột đi lên

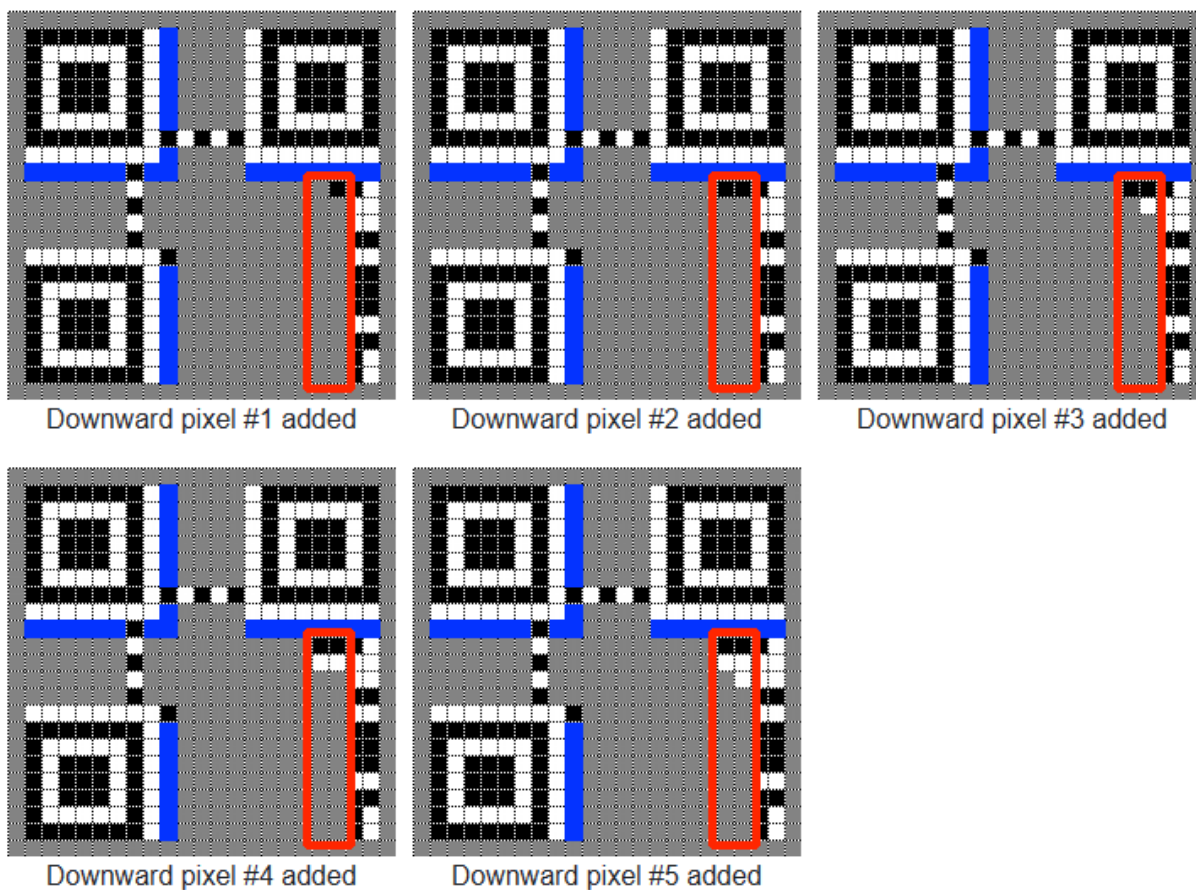
*Vị trí hướng xuống*

Hình ảnh sau đây minh họa vị trí các bit dữ liệu khi cột đi xuống.



Hình 33 Cách đặt dữ liệu khi cột đi xuống

Vị trí minh họa các bit dữ liệu khi cột đi xuống

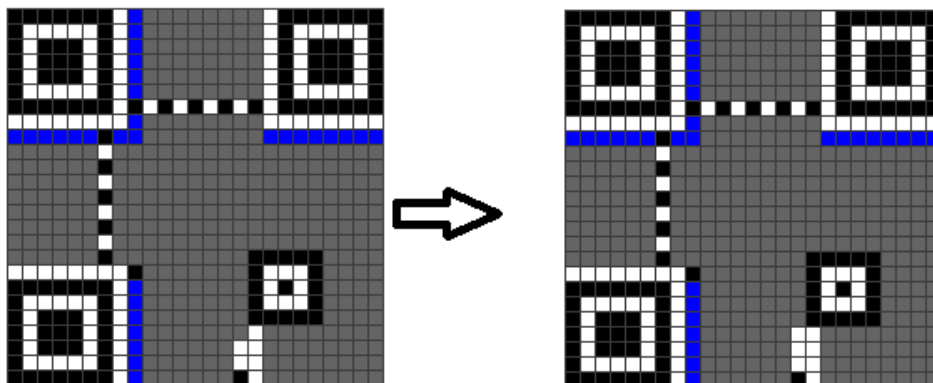


Hình 34 Minh họa đặt dữ liệu khi cột đi xuống

### *Bỏ qua mẫu chức năng*

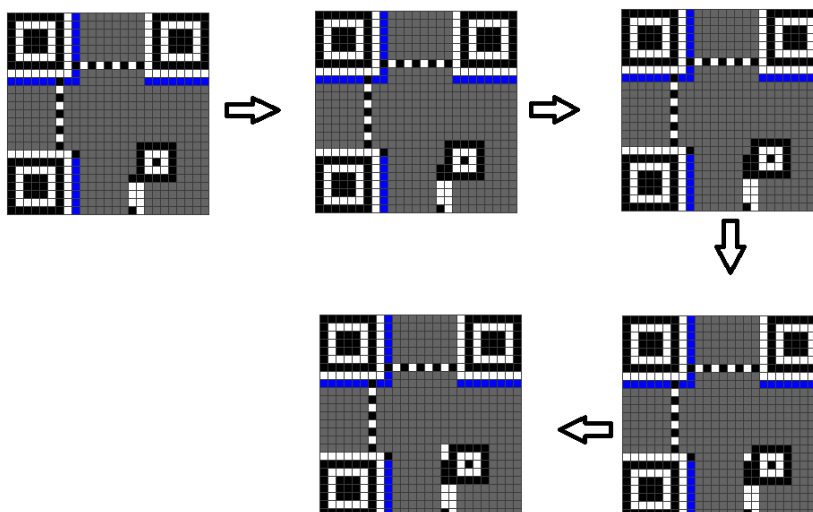
Khi gặp một mẫu chức năng, bỏ qua bất kỳ mô-đun nào đã được sử dụng cho đến khi đến mô-đun chưa sử dụng tiếp theo.

Trong các hình ảnh sau đây, các bit dữ liệu dưới mẫu căn chỉnh đang tiến lên trong một cột lên trên. Lưu ý rằng cột đó chồng lên mẫu căn chỉnh.



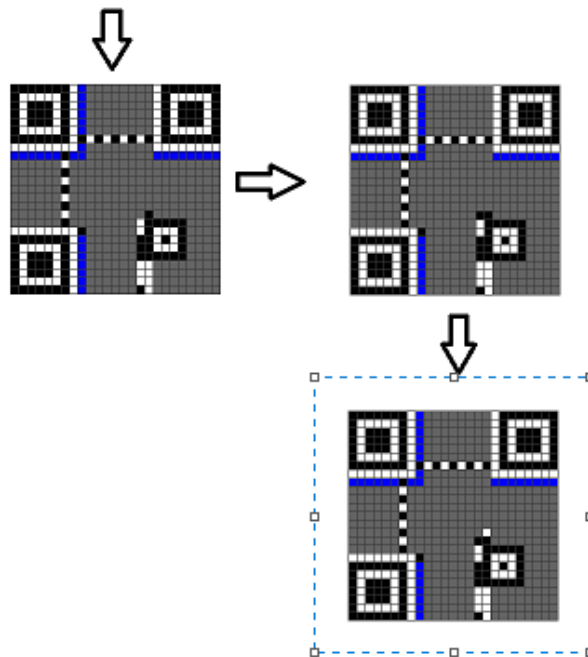
Hình 35 Đặt dữ liệu khi gấp mẫu chức năng

Khi gấp mẫu căn chỉnh, chỉ cần bỏ qua module trong phần mẫu căn chỉnh nó và vượt qua nó.



Hình 36 Đặt dữ liệu khi gấp mẫu chức năng

Tiếp tục thêm các module như cách thông thường



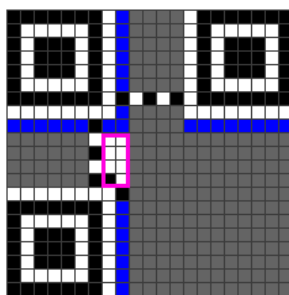
Hình 37 Thêm các module

Tóm lại, luôn tiến hành bình thường dọc theo các cột, bỏ qua bất kỳ mô-đun nào được sử dụng bởi các mẫu chức năng hoặc khu vực dành riêng. Ngoại lệ duy nhất là mẫu thời gian dọc, như đã giải thích dưới đây.

*Ngoại lệ: Mẫu Thời Gian Dọc*

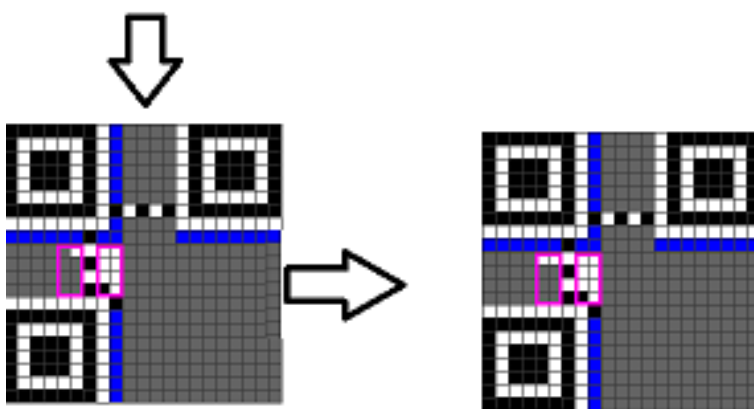
Mẫu thời gian dọc là ngoại lệ duy nhất đối với quy tắc này. Khi đạt được mẫu thời gian dọc, luôn bắt đầu cột tiếp theo bên trái của nó. Không có cột nào được chồng lên mẫu thời gian dọc.

Trong hình ảnh sau đây, cột trước đó (lên trên) được đánh dấu. Phần còn lại của các mô-đun trên cột này được chiếm bởi các mẫu chức năng và khu vực dành riêng, vì vậy cột tiếp theo phải được bắt đầu, đi xuống.



Hình 38 Cách đặt dữ liệu khi gặp mẫu thời gian

Như được hiển thị trong các hình ảnh sau đây, cột tiếp theo bắt đầu bên trái của mẫu thời gian dọc. Cột không chồng lên mẫu thời gian dọc.



Hình 39 Minh họa đặt dữ liệu khi gặp mẫu thời gian

## 2.7. DaMasking

Nếu một mô-đun trong mã QR được "che mặt", điều này đơn giản có nghĩa là nếu nó là một mô-đun sáng, nó sẽ được thay đổi thành một mô-đun tối và nếu nó là một mô-đun tối, nó sẽ được thay đổi thành một mô-đun sáng. Nói cách khác, che mặt chỉ đơn giản là chuyển đổi màu sắc của mô-đun.

### *Tổng quan về các mẫu che mặt*

Thông số mã QR xác định tám mẫu che mặt có thể được áp dụng cho mã QR. Ví dụ, đối với mẫu che mặt #1, mỗi hàng có số chẵn trong ma trận QR được che và đối với mẫu che mặt #2, mỗi cột thứ ba trong ma trận QR được che.

Mẫu mask	Nếu công thức bên dưới đúng với tọa độ hàng/cột đã cho, hãy chuyển bit ở
----------	--

	tọa độ đó sang bit còn lại (0 ->1, 1->0)
0	$(\text{row} + \text{column}) \bmod 2 == 0$
1	$(\text{row}) \bmod 2 == 0$
2	$(\text{column}) \bmod 3 == 0$
3	$(\text{row} + \text{column}) \bmod 3 == 0$
4	$(\text{floor}(\text{row} / 2) + \text{floor}(\text{column} / 3)) \bmod 2 == 0$
5	$((\text{row} * \text{column}) \bmod 2) + ((\text{row} * \text{column}) \bmod 3) == 0$
6	$((\text{row} * \text{column}) \bmod 2) + ((\text{row} * \text{column}) \bmod 3) \bmod 2 == 0$
7	$((\text{row} + \text{column}) \bmod 2) + ((\text{row} * \text{column}) \bmod 3) \bmod 2 == 0$

Bảng 13 Các mẫu che mặt của QR code

### ***Những gì cần che***

Các mẫu che chỉ được áp dụng cho các module dữ liệu và module sửa lỗi. Nói cách khác:

- Không che các mẫu chức năng (mẫu tìm kiếm, các chuẩn thời gian, các bộ phận tách biệt, các chuẩn căn chỉnh)
- Không che các khu vực dự trữ (khu vực thông tin định dạng, khu vực thông tin phiên bản)

### ***Xác định mẫu che tốt nhất***

Sau khi áp dụng một mẫu che cho ma trận QR, nó sẽ nhận được điểm phạt dựa trên bốn điều kiện đánh giá được xác định trong thông số mã QR. Một bộ mã hóa QR phải áp dụng tất cả tám mẫu che và đánh giá từng cái. **Bất kể mẫu che nào có điểm phạt thấp nhất là mẫu che phải được sử dụng cho đầu ra cuối cùng.**

### ***Cách Đánh giá Các Khu vực Dự trữ***

Lưu ý rằng toàn bộ ma trận (bao gồm cả các mẫu chức năng và các khu vực dự trữ) được đánh giá, ngay cả khi việc che chỉ được áp dụng cho các module dữ liệu và sửa lỗi.

### ***Bốn Quy tắc Phạt***

Bốn quy tắc phạt có thể được tóm tắt như sau:

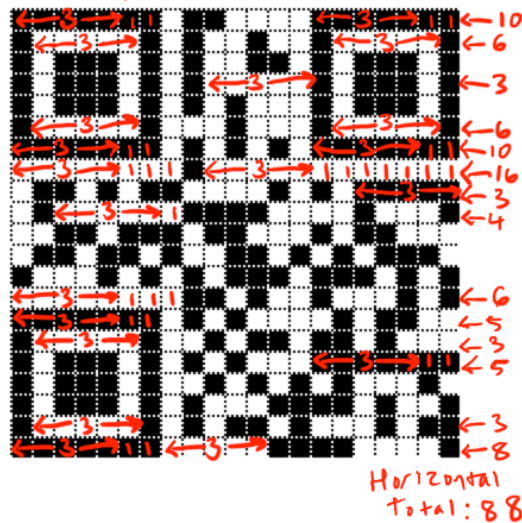
- Quy tắc đầu tiên cho mã QR phạt cho từng nhóm năm hoặc nhiều module cùng màu trong hàng (hoặc cột).
- Quy tắc thứ hai cho mã QR phạt cho từng khu vực 2x2 của các module cùng màu trong ma trận.
- Quy tắc thứ ba cho mã QR phạt lớn nếu có các chuẩn trông giống như các chuẩn tìm kiếm.
- Quy tắc thứ tư cho mã QR phạt nếu hơn phân nửa các module là tối hoặc sáng, với phạt lớn hơn cho sự khác biệt lớn hơn.

#### *Điều kiện đánh giá #1*

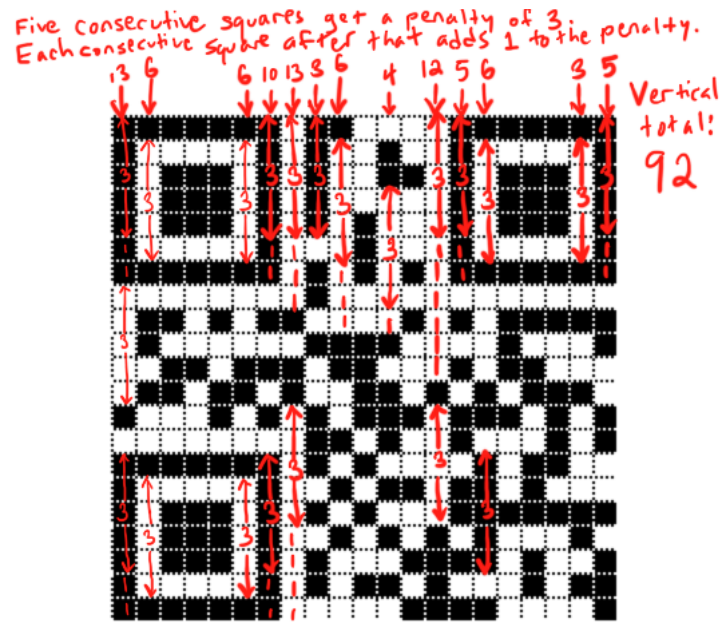
Đối với điều kiện đánh giá đầu tiên, kiểm tra từng hàng một. Nếu có năm module liên tiếp cùng màu, thêm 3 vào điểm phạt. Nếu có nhiều module cùng màu sau khi có năm module đầu tiên, thêm 1 cho mỗi module cùng màu bổ sung. Sau đó, kiểm tra từng cột một, kiểm tra cùng một điều kiện. Thêm tổng số ngang và dọc để thu được điểm phạt #1.

Các hình ảnh sau minh họa quá trình đánh giá mã QR theo cách này. Trong ví dụ này, điểm phạt ngang là 92 và điểm phạt dọc là 88. Vì vậy, điểm phạt #1 là  $92 + 88 = 180$ .

Five consecutive squares get a penalty of 3.  
Each consecutive square after that adds 1 to the penalty.







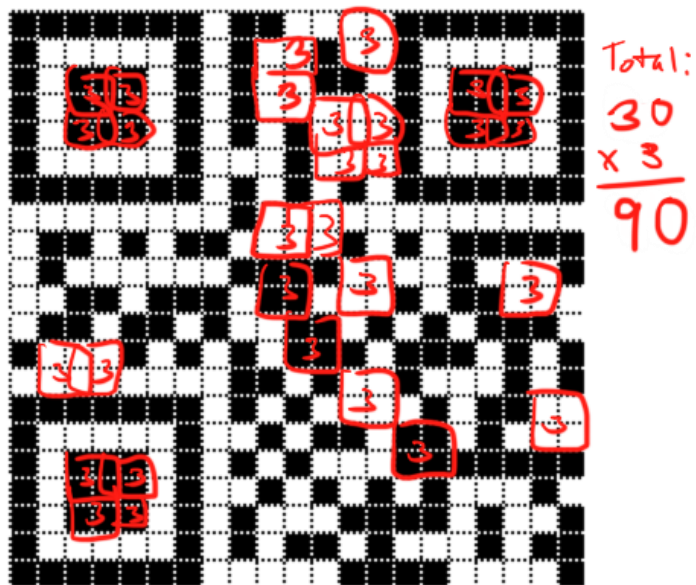
Hình 40 Điều kiện đánh giá 1

### Điều kiện đánh giá #2

Đối với điều kiện đánh giá thứ hai, tìm kiếm các khu vực cùng màu có kích thước ít nhất 2x2 module hoặc lớn hơn. Thông số mã QR nói rằng đối với một khối màu đồng nhất có kích thước  $m \times n$ , điểm phạt là  $3 \times (m - 1) \times (n - 1)$ . Tuy nhiên, thông số mã QR không chỉ định cách tính điểm phạt khi có nhiều cách chia các khối màu đồng nhất.

Vì vậy, thay vì tìm kiếm các khối màu đồng nhất lớn hơn 2x2, chỉ cần thêm 3 vào điểm phạt cho mỗi khối 2x2 cùng màu trong mã QR, đảm bảo đếm các khối 2x2 chồng lên nhau. Ví dụ, một khối 3x2 cùng màu nên được tính là hai khối 2x2, một chồng lên trên cái kia.

Hình ảnh sau minh họa cách tính quy tắc phạt #2.



Hình 41 Điều kiện đánh giá 2

### Điều kiện đánh giá #3

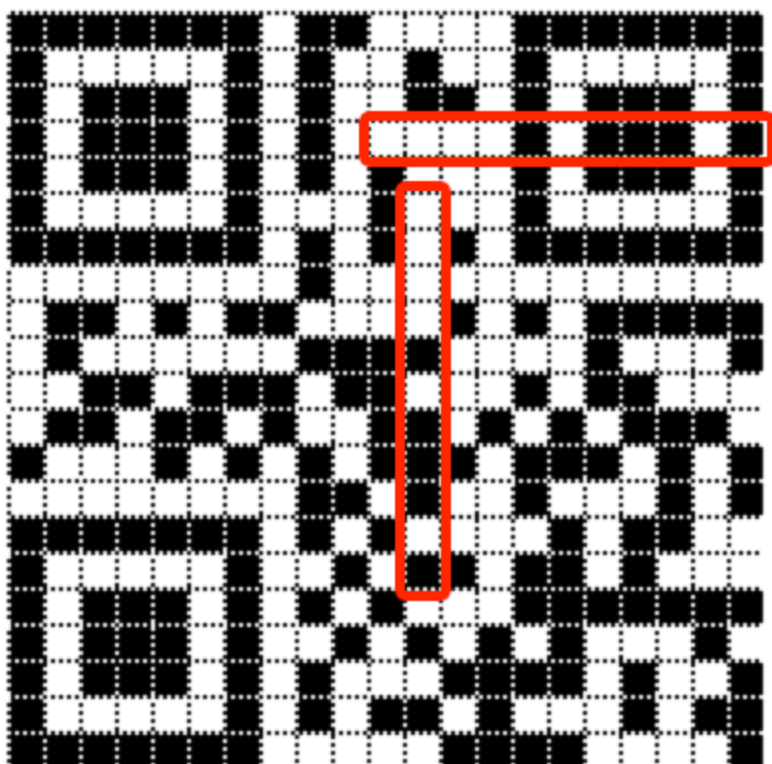
Quy tắc phạt thứ ba tìm kiếm các mẫu tối-sáng-tối-tối-tối-sáng-tối có bốn mô-đun sáng ở hai bên. Nói cách khác, nó tìm kiếm bất kỳ hai mẫu sau đây:



HOẶC



Mỗi lần tìm thấy mẫu này, thêm 40 vào điểm phạt. Trong ví dụ dưới đây, có hai mẫu như vậy. Vì vậy, điểm phạt #3 là 80.



Hình 42 Điều kiện đánh giá 3

#### *Điều kiện đánh giá #4*

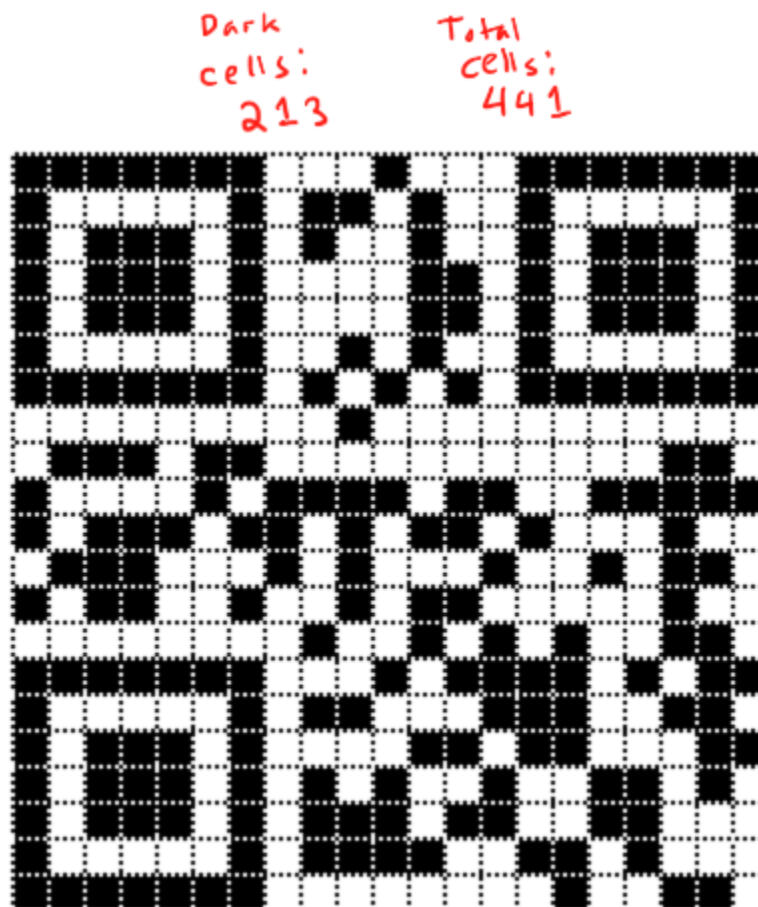
Điều kiện đánh giá cuối cùng dựa trên tỷ lệ các module sáng so với các module tối. Để tính quy tắc phạt này, thực hiện các bước sau:

1. Đếm tổng số lượng các module trong ma trận.
2. Đếm số lượng các module tối trong ma trận.
3. Tính phần trăm các module trong ma trận là tối:  $(\text{module tối} / \text{tổng module}) * 100$
4. Xác định bội số trước và tiếp theo của năm của phần trăm này. Ví dụ, đối với 43 phần trăm, bội số trước của năm là 40 và bội số tiếp theo của năm là 45.
5. Trừ 50 từ mỗi bội số này của năm và lấy giá trị tuyệt đối của kết quả. Ví dụ,  $|40 - 50| =$

$|-10| = 10$  và  $|45 - 50| = |-5| = 5$ .

6. Chia mỗi số này cho năm. Ví dụ,  $10/5 = 2$  và  $5/5 = 1$ .

7. Cuối cùng, lấy số nhỏ nhất trong hai số và nhân nó với 10. Trong ví dụ này, số thấp hơn là 1, vì vậy kết quả là 10. Đây là điểm phạt #4. Ví dụ khác, trong hình dưới đây, tổng số lượng các module là 441 và tổng số lượng các module tối là 213.



Hình 43 Điều kiện đánh giá 4

Phần trăm các module tối là  $(213 / 441) * 100 \approx 48.2993$

Bội số trước của năm là 45 và bội số tiếp theo của năm là 50.

Trừ đi 50 và lấy giá trị tuyệt đối của mỗi số:

$$|45 - 50| = |-5| = 5$$

$$|50 - 50| = 0$$

Chia mỗi số cho năm:

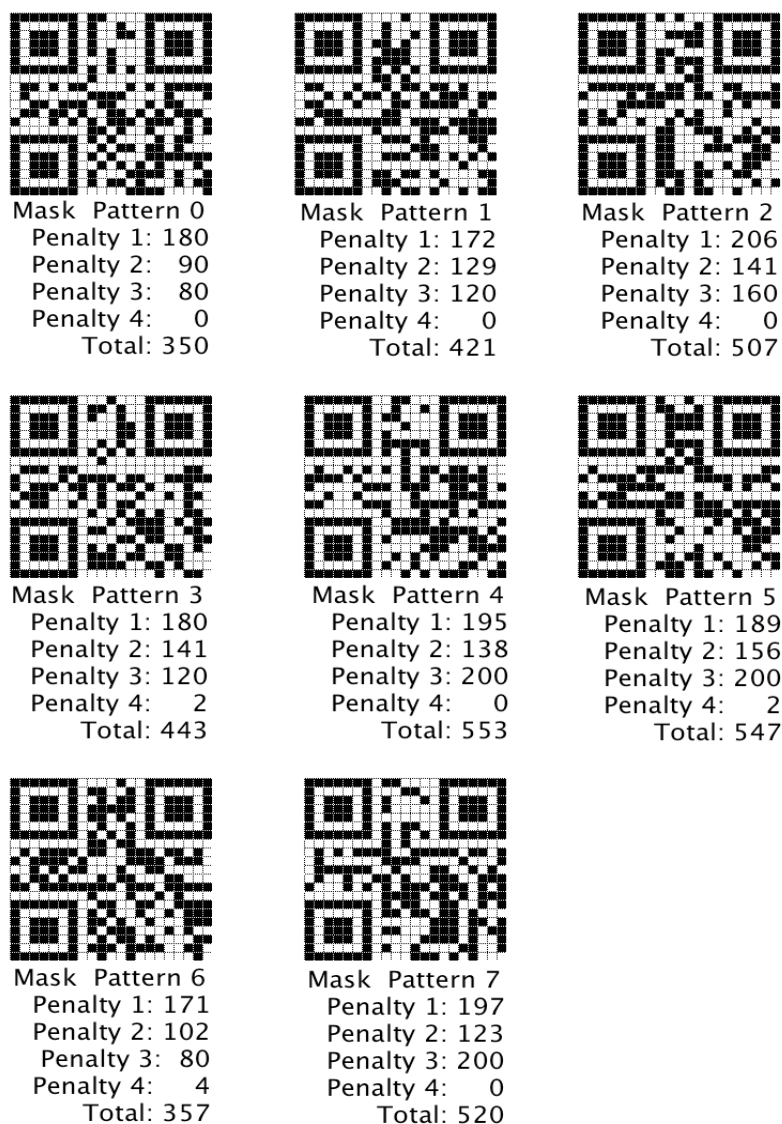
$$5/5 = 1$$

$$0/5 = 0$$

Số nhỏ nhất trong hai số này là 0. Nhân với 10, điều này vẫn là 0. Vì vậy, điểm phạt #4 trong ví dụ này là 0.

*Tổng điểm phạt*

Để hoàn thành việc đánh giá một mã QR, cộng bốn điểm phạt. Tổng số là điểm phạt tổng thể của mã QR.



Hình 44 Tổng các điểm đánh giá với mỗi mẫu che mặt

Có thể thấy mẫu 0 có điểm thấp nhất nên sẽ chọn mẫu 0.

## 2.8. Format and Version Information

### *Chuỗi Định Dạng*

Chuỗi thông tin định dạng mã hóa mức độ sửa lỗi và mẫu mặt nạ nào đang được sử dụng trong mã QR hiện tại. Vì có bốn mức độ sửa lỗi có thể (L, M, Q và H) và tám mẫu mặt nạ có thể, có 32 (4 lần 8) chuỗi thông tin định dạng có thể. Phần tiếp theo giải thích cách tạo ra các chuỗi định dạng này. Để biết danh sách đầy đủ của 32 chuỗi định dạng, tham khảo thêm tại [đây](#).

### *Tạo Chuỗi Định Dạng*

Chuỗi định dạng luôn dài 15 bit. Để tạo chuỗi, trước tiên tạo một chuỗi năm bit mã hóa mức độ sửa lỗi và mẫu mặt nạ đang được sử dụng trong mã QR này. Sau đó sử dụng năm bit đó để tạo ra mười bit sửa lỗi. Mười lăm bit kết quả được XOR với mẫu mặt nạ 101010000010010. Quá trình này được giải thích chi tiết dưới đây.

### *Các Bit Sửa Lỗi*

Bước đầu tiên để tạo chuỗi định dạng là lấy hai bit chỉ định mức độ sửa lỗi đang được sử dụng trong mã QR. Bảng sau đây cho thấy các chuỗi bit cho mỗi mức độ sửa lỗi.

Error Correction Level	Bits	Integer Equivalent
L	01	1
M	00	0
Q	11	3
H	10	2

Hình 45 Danh sách các bit sửa lỗi với mỗi mức độ sửa lỗi

### *Các Bit Mẫu Mặt Nạ*

Đối với các mẫu mặt nạ, hãy tham khảo trang [mẫu mặt nạ](#) mã QR để tìm số mặt nạ đi kèm với mỗi mẫu. Chuyển đổi số thành một chuỗi nhị phân ba bit.

Ví dụ, nếu chúng đã sử dụng mức độ sửa lỗi L và mẫu mặt nạ 4, chuỗi nhị phân năm bit được tạo như sau:

- 01 (chỉ báo cho mức độ sửa lỗi L)
- 100 (nhị phân cho 4, tức là mẫu mặt nạ 4)

Kết quả: 01100

### *Tạo Các Bit Sửa Lỗi cho Chuỗi Định Dạng*

Bây giờ chúng đã có năm bit cho chuỗi định dạng, chúng phải sử dụng nó để tạo ra mười bit sửa lỗi. Bước này sử dụng Sửa Lỗi Reed-Solomon, nhưng nó hơi dễ hơn vì trong trường hợp này, các đa thức không có nhiều hơn mười lăm hạng tử và hệ số của chúng đều là 1 hoặc 0.

### *Lấy Đa Thức Sinh*

Khi tạo các từ mã sửa lỗi định dạng của mã QR, thông số mã QR nói rằng sử dụng đa thức sinh sau:

$$x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

Có thể chuyển đổi nó thành một chuỗi nhị phân bằng cách chỉ lấy hệ số của mỗi hạng tử. Hệ số của  $x^{10}$  là 1, hệ số của  $x^9$  là 0 vì  $x^9$  không có trong đa thức và cứ thế.

Nói cách khác, chuỗi nhị phân đại diện cho đa thức sinh cho bước này là 10100110111

### *Tính toán các Bit Sửa Lỗi*

Bước tiếp theo là chia các bit chuỗi định dạng (01100 từ bước trước) cho đa thức sinh (10100110111 từ bước trước).

Để làm điều này, trước tiên tạo một chuỗi 15 bit bằng cách đặt mười số 0 vào bên phải của chuỗi định dạng, như sau:

01100 -> 011000000000000

Bây giờ xóa bất kỳ số 0 nào từ phía trái:

011000000000000 -> 11000000000000

Bây giờ thực hiện phép chia. Các bước (được mô tả chi tiết hơn dưới đây) là:

1. Đệm chuỗi đa thức sinh bên phải với số 0 để làm cho nó cùng chiều dài với chuỗi định dạng hiện tại.
2. XOR chuỗi đa thức sinh được đệm với chuỗi định dạng hiện tại.
3. Xóa số 0 từ phía trái của kết quả.

**Phải chia các đa thức cho đến khi chuỗi định dạng kết quả có 10 bit hoặc ít hơn.** Do đó, trước mỗi bước chia, kiểm tra để đảm bảo rằng chuỗi định dạng hiện tại có 11 bit hoặc dài hơn (11 bit là chiều dài của đa thức sinh). Chuỗi hiện tại: 11000000000000, dài 14 bit.

### *Phép Chia Đầu Tiên*

Đầu tiên, đệm chuỗi đa thức sinh bên phải với số 0 để làm cho nó cùng chiều dài với chuỗi định dạng hiện tại:

10100110111 -> 10100110111000

11000000000000 (chuỗi định dạng)



Bây giờ, XOR chuỗi định dạng với chuỗi đa thức sinh được đệm:

$$11000000000000 \wedge 10100110111000 = 01100110111000$$

Sau đó xóa bất kì số 0 đứng ở bên trái

$$01100110111000 \rightarrow 1100110111000$$

### ***Phép Chia Thứ Hai***

Chuỗi kết quả 1100110111000 dài 13 bit, vì vậy vì nó có 11 bit hoặc dài hơn.

Đệm đa thức sinh bên phải để làm cho cùng chiều dài với chuỗi định dạng hiện tại:

$$10100110111 \rightarrow 1010011011100$$

$$1100110111000 \text{ (kết quả của phép chia đầu)}$$

XOR chuỗi định dạng hiện tại với chuỗi đa thức sinh được đệm:

$$1100110111000 \wedge 1010011011100 = 110101100100$$

### ***Phép Chia Thứ Ba***

Chuỗi kết quả 110101100100 dài 12 bit, vì vậy vì nó có 11 bit hoặc dài hơn.

Đệm đa thức sinh bên phải để làm cho cùng chiều dài với chuỗi định dạng hiện tại:

$$10100110111 \rightarrow 101001101110$$

$$110101100100 \text{ (kết quả của phép chia thứ 2)}$$

XOR chuỗi định dạng hiện tại với chuỗi đa thức sinh được đệm:

$$110101100100 \wedge 101001101110 = 011100001010$$

Và xóa số 0 từ phía trái của kết quả:

$$011100001010 \rightarrow 11100001010$$

### ***Phép Chia Thứ Tư***

Chuỗi kết quả 11100001010 dài 11 bit, vì vậy vì nó có 11 bit hoặc dài hơn.

Không cần đệm đa thức sinh, vì chuỗi định dạng hiện tại có cùng chiều dài.

XOR chuỗi định dạng hiện tại với đa thức sinh:

$$11100001010 \wedge 10100110111 = 1000111101$$

Kết quả dài 10 bit, hoàn thành bước chia. Nếu kết quả nhỏ hơn 10 bit, sẽ đệm nó bên

TRÁI với số 0 để làm cho nó dài 10 bit.

### ***Đưa các Bit Định Dạng và Sửa Lỗi lại với nhau***

Tạo một chuỗi bằng cách sử dụng năm bit ban đầu của thông tin định dạng (chỉ báo mức độ sửa lỗi và chỉ báo mẫu mặt nạ), theo sau là các bit sửa lỗi vừa tạo ra.

Chuỗi định dạng năm bit: 01100

Chuỗi sửa lỗi mười bit từ bước chia: 1000111101

Chuỗi kết hợp: 01**100**1000**111101**

### ***XOR với Chuỗi Mặt Nạ***

Thông số mã QR nói rằng XOR kết quả với chuỗi nhị phân sau: 101010000010010

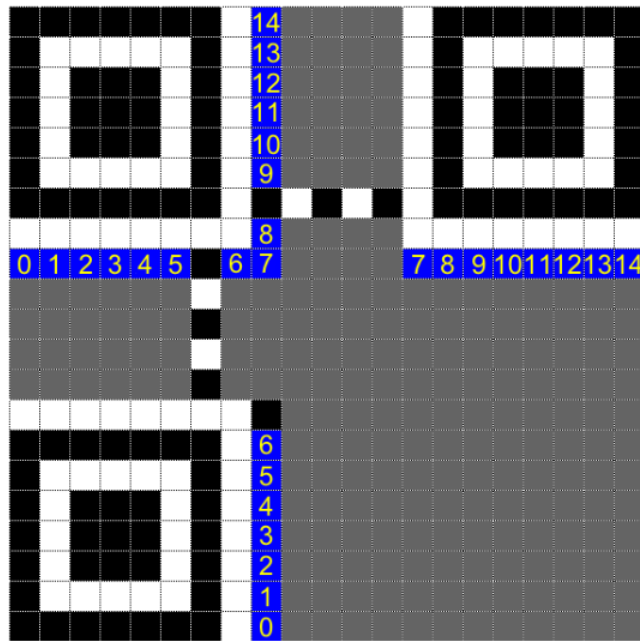
$011001000111101 \wedge 101010000010010 = 110011000101111$

Chuỗi định dạng cuối cùng cho một mã có mức độ sửa lỗi L và mẫu mặt nạ 4 là 110011000101111

### ***Đặt chuỗi định dạng vào QR code***

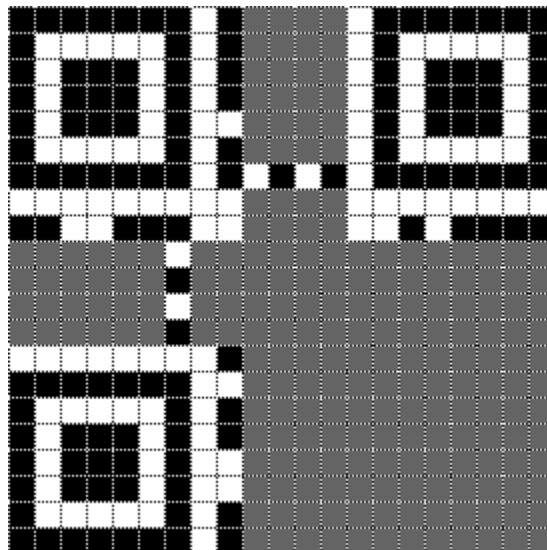
Chuỗi thông tin định dạng được đặt dưới các mẫu tìm kiếm trên cùng và bên phải của các mẫu tìm kiếm bên trái nhất, như hình dưới đây. Số 0 trong hình ảnh đề cập đến bit có trọng số lớn nhất của chuỗi định dạng và số 14 đề cập đến bit có trọng số nhỏ nhất. Nói cách khác, sử dụng chuỗi định dạng 110011000101111 từ ví dụ trên, các số trong hình ảnh dưới đây tương ứng với các bit chuỗi định dạng 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 46 Cách đặt chuỗi định dạng vào QR code

Theo cách sắp xếp đó, hình ảnh điền thông tin chuỗi định dạng 110011000101111 trong QR code version 1 sẽ như sau:



Hình 47 QR code sau khi đặt QR code

### ***Mô-đun Tối***

Mọi mã QR đều phải có một pixel tối, còn được gọi là mô-đun tối, tại tọa độ  $(8, 4 * \text{phiên bản} + 9)$ .

Nghĩa là, tọa độ y của mô-đun tối là

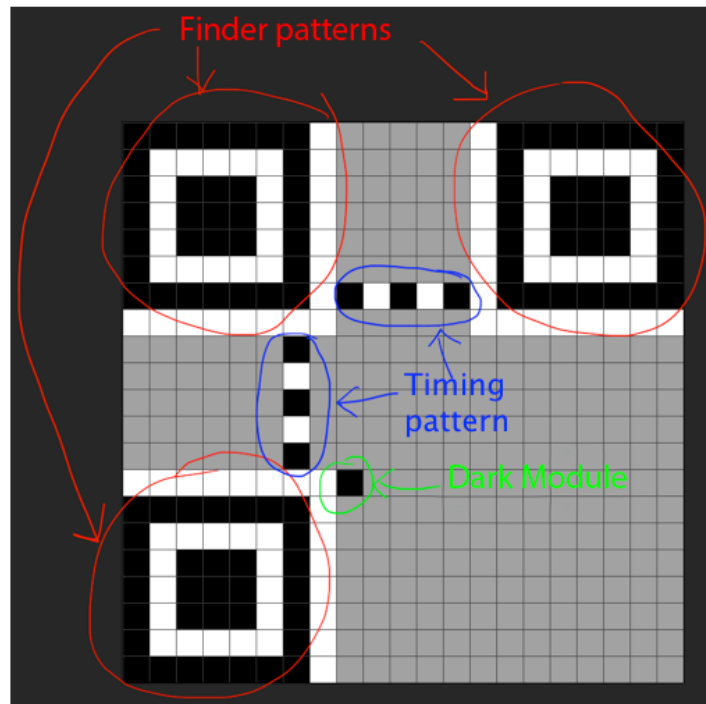
phiên bản 1:  $4 * 1 + 9 = 13$

phiên bản 2:  $4 * 2 + 9 = 17$

phiên bản 3:  $4 * 3 + 9 = 21$

và cứ thế.

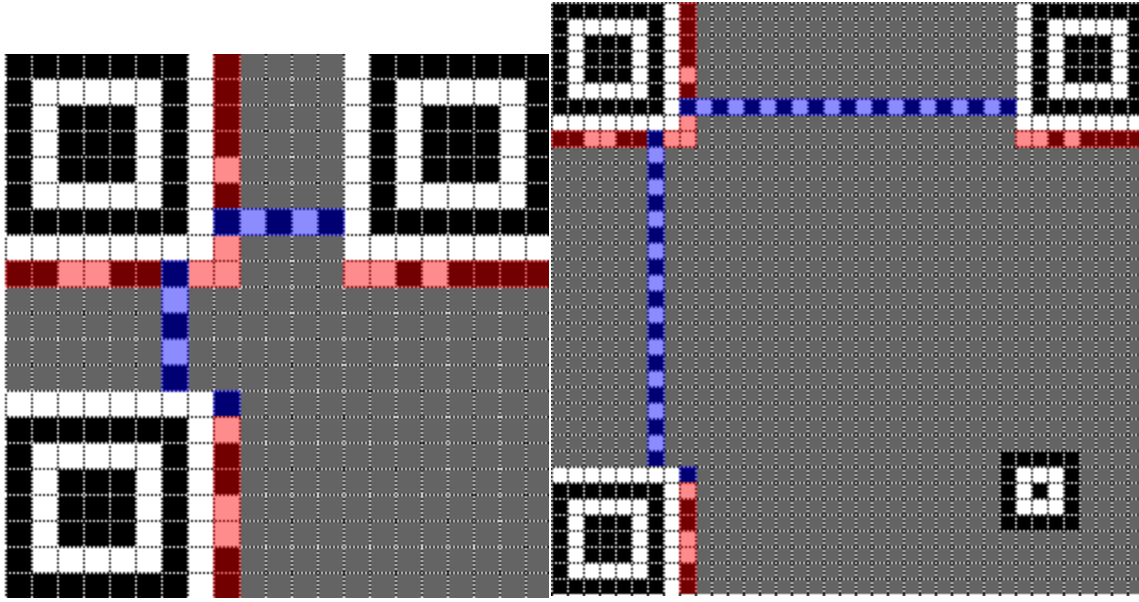
Điều này có nghĩa là mô-đun tối luôn nằm bên phải góc trên cùng bên phải của mẫu tìm kiếm dưới bên trái. Điều này được hiển thị trong hình dưới đây.



Hình 48 Thêm modul tối vào QR code

### ***Đối với các Mã Lớn hơn***

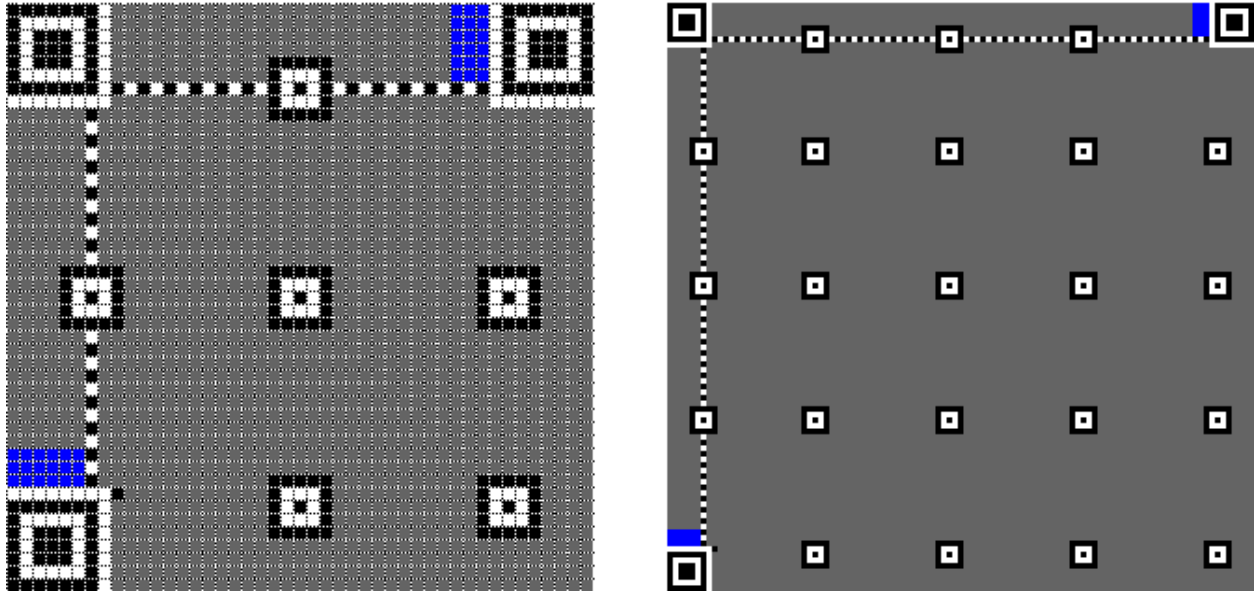
Không quan trọng kích thước của mã QR là bao nhiêu, các bit chuỗi định dạng được đặt dưới các mẫu tìm kiếm trên cùng và bên phải của các mẫu tìm kiếm bên trái nhất. Lưu ý rằng cả hai hình ảnh đều có cùng một chuỗi định dạng, 110011000101111. Chuỗi định dạng được đánh dấu màu đỏ và mẫu thời gian và mô-đun tối được đánh dấu màu xanh lam.



Hình 49 Chuỗi định dạng được đánh dấu đỏ

### ***Thông Tin Phiên Bản***

Nếu mã QR là phiên bản 7 hoặc lớn hơn, phải bao gồm một chuỗi thông tin phiên bản 18 bit ở góc dưới bên trái và góc trên bên phải của mã QR. (Để biết danh sách đầy đủ của tất cả các chuỗi thông tin phiên bản có thể, hãy tham khảo tại [đây](#).) Các khu vực thông tin phiên bản là các hình chữ nhật xanh lam 6x3 được hiển thị trong các hình ảnh dưới đây. Thông tin phiên bản được đặt bên cạnh các mẫu tìm kiếm không quan trọng kích thước của mã QR là bao nhiêu. Hình ảnh bên trái là một mã phiên bản 7 và hình ảnh bên phải là một mã QR phiên bản 22.



Hình 50 Điền thông tin phiên bản vào QR code

### ***Tạo Chuỗi Thông Tin Phiên Bản***

Thông số mã QR nói rằng sử dụng mã Golay (18, 6) cho chuỗi thông tin phiên bản. Như vậy, chuỗi thông tin phiên bản là một chuỗi 18 bit bao gồm một chuỗi nhị phân sáu bit mã hóa phiên bản QR, theo sau là một chuỗi 12 bit sửa lỗi. Toàn bộ chuỗi dài 18 bit.

### ***Lấy Đa Thức Sinh***

Thông số mã QR nói rằng sử dụng đa thức sinh sau cho bước này:

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

Như đã giải thích trong phần chuỗi định dạng trước đó trên trang này, có thể đại diện cho đa thức sinh này với chuỗi nhị phân sau:

1111100100101

### ***Thực hiện Phép Chia***

Từ đây, có thể làm theo các bước chia giống như đã sử dụng để tạo chuỗi thông tin định dạng, ngoại trừ trong trường hợp này đệm các chuỗi ban đầu để có độ dài 18 bit thay vì 15 và dừng lại khi chuỗi bit hiện tại có 12 bit hoặc ít.

Bắt đầu bằng cách tạo một chuỗi nhị phân sáu bit biểu diễn số phiên bản. Ví dụ, cho một mã phiên bản 7, tương đương nhị phân sáu bit của 7 là:

000111

Chuyển nó thành một chuỗi 18 bit bằng cách đệm bên phải với số 0:

000111000000000000

Và xóa số 0 từ phía trái:

1110000000000000

Bây giờ đệm đa thức sinh bên phải với số 0 để có cùng chiều dài.

1111100100101 -> 111110010010100 (đa thức sinh)

1110000000000000

XOR chuỗi phiên bản và đa thức sinh được đệm:

11100000000000 ^ 111110010010100 = 110010010100

Chuỗi này đã có chiều dài yêu cầu là 12, vì vậy không cần chia thêm. Giống như chuỗi thông tin định dạng, nếu kết quả nhỏ hơn 12, nó phải được đệm TRÁI với số 0 để làm cho nó dài 12 bit.

Cuối cùng, đặt chuỗi phiên bản sáu bit ban đầu vào bên trái của kết quả từ bước cuối cùng.

- Chuỗi phiên bản: 000111
- Chuỗi sửa lỗi từ trên: 110010010100

Chuỗi thông tin phiên bản cuối cùng: 000111110010010100

### ***Đặt Chuỗi Phiên Bản trong Mã QR***

Có hai khu vực hình chữ nhật nơi chuỗi thông tin phiên bản phải được đặt: một phía dưới bên trái và một phía trên bên phải.

#### ***Khối thông tin phiên bản dưới bên trái***

Khối thông tin phiên bản dưới bên trái cao 3 modulo và rộng 6 modulo. Bảng sau đây giải thích cách sắp xếp các bit của chuỗi thông tin phiên bản trong khu vực thông tin phiên bản dưới bên trái. Số 0 đại diện cho bit phải nhất (ít quan trọng nhất) của chuỗi thông tin phiên bản và số 17 đại diện cho bit trái nhất (quan trọng nhất) của chuỗi thông tin phiên bản.

00	03	06	09	12	15
01	04	07	10	13	16
02	05	08	11	14	17

Hình 51 Cách điền thông tin phiên bản dưới bên trái

### ***Khởi thông tin phiên bản trên bên phải***

Là một khối modulo rộng 3 modulo và cao 6 modulo. Bảng sau đây giải thích cách sắp xếp các bit của chuỗi thông tin phiên bản trong khu vực thông tin phiên bản góc trên bên phải. Số 0 đại diện cho bit bên PHẢI nhất (ít quan trọng nhất) của chuỗi thông tin phiên bản và số 17 đại diện cho bit bên TRÁI nhất (quan trọng nhất) của chuỗi thông tin phiên bản.

00	01	02
03	04	05
06	07	08
09	10	11
12	13	14
15	16	17

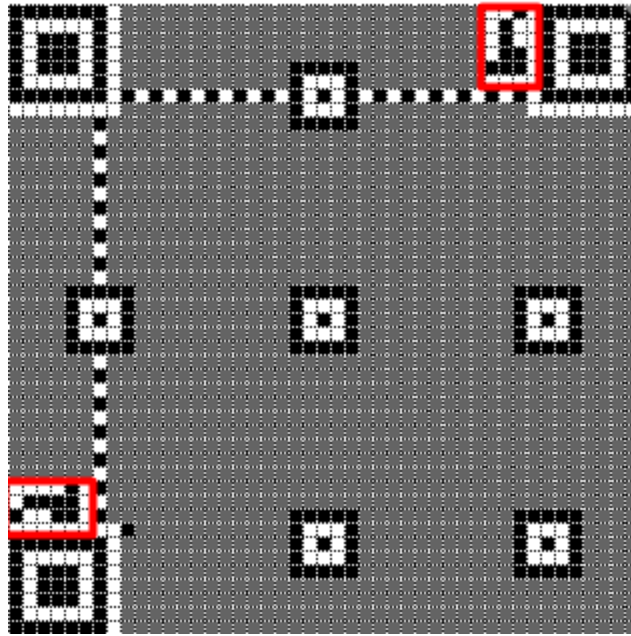
Hình 52 Cách đặt thông tin phiên bản bên phải

Chuỗi thông tin 000111110010010100 sử dụng version 7, các số trong bảng sẽ được đánh theo bit của chuỗi đó

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

Sau đó theo các bảng trên sẽ điền thông tin vào QR code





Hình 53 Thông tin được điền vào QR code

## CHƯƠNG 3. TRIỂN KHAI QR CODE TRONG GIAO DỊCH

### 3.1. Môi trường, thư viện

Ứng dụng website bán hàng sử dụng thư viện React phía client, và chạy trên môi trường Node.js phía server.

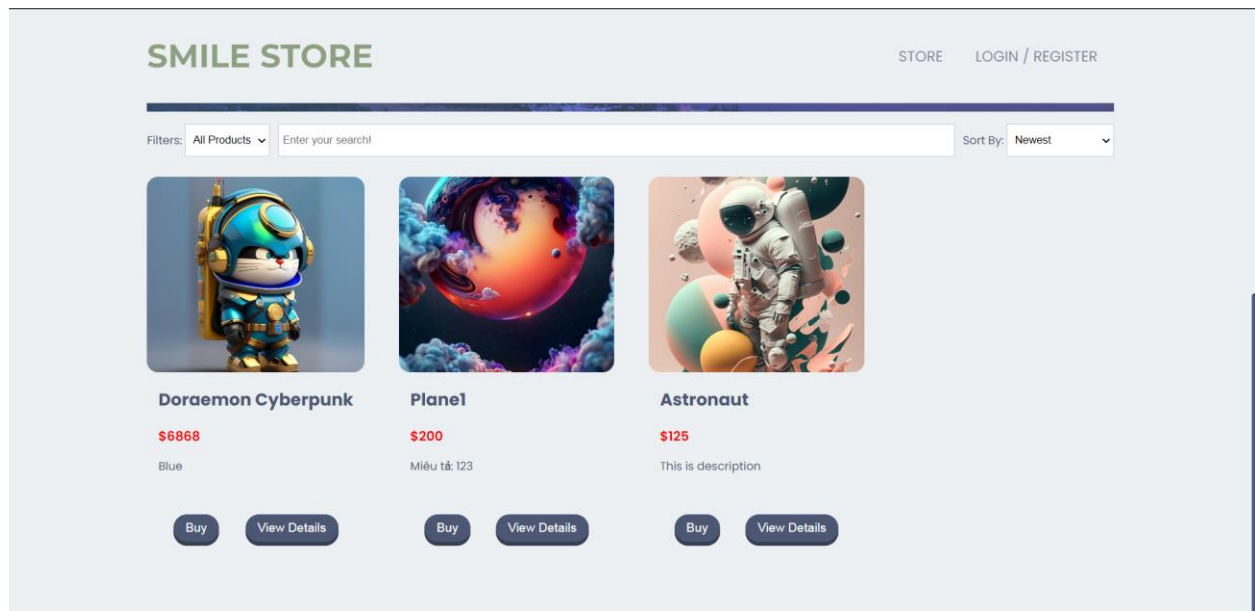
Ngoài ra hai thư viện chính liên quan đến đề tài là thư viện **qrcode** được cài đặt phía server và thư viện **html5-qrcode** được cài đặt phía client.

Thông tin thêm về hai thư viện **qrcode** và **html5-qrcode**:

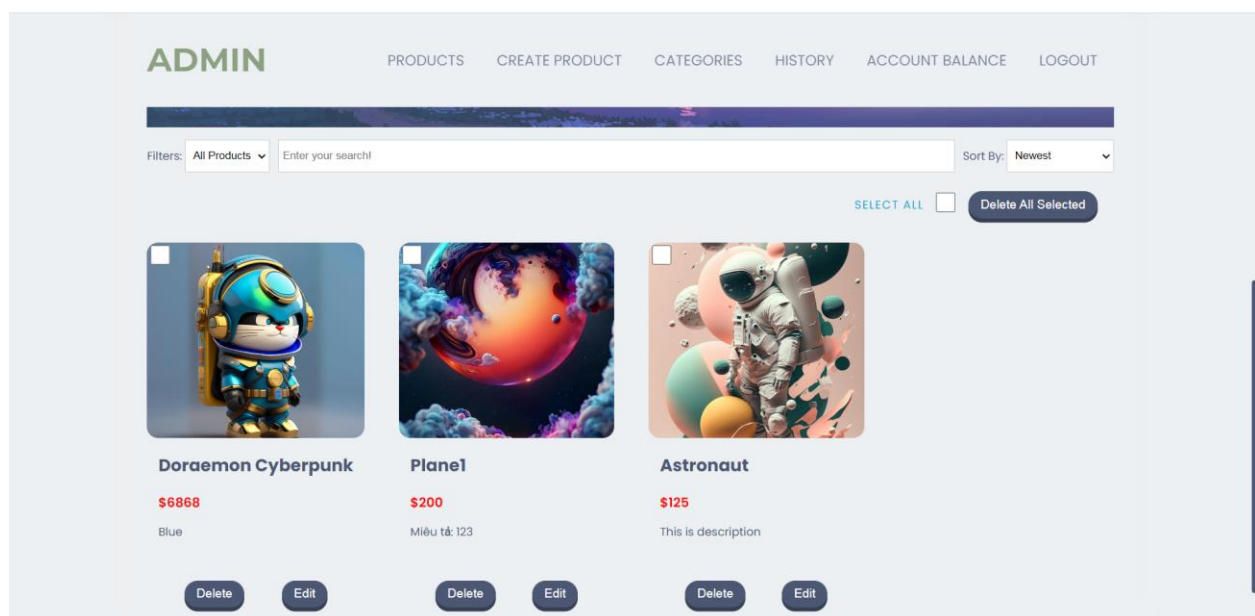
- **qrcode**: là một thư viện mã nguồn mở cho phép tạo mã QR code dễ dàng trong các ứng dụng web. Thư viện này hỗ trợ tạo mã QR code với nhiều tùy chọn tùy chỉnh, bao gồm kích thước, màu sắc, nội dung và kiểu dữ liệu.
- **html5-qrcode**: là một thư viện JavaScript mã nguồn mở dùng để quét mã QR code trên trình duyệt web sử dụng các API hỗ trợ của HTML5. Thư viện này cho phép tích hợp chức năng quét mã QR code vào ứng dụng web của mình một cách dễ dàng.

### 3.2. Sơ lược về ứng dụng

Là một website bán hàng, ứng dụng bao gồm hai loại người dùng là quản trị viên – người bán hàng và người dùng bình thường – người mua hàng. Website có các chức năng chính như thêm, sửa xóa sản phẩm, danh mục sản phẩm, xem lịch sử các đơn hàng đối với quản trị viên và đặt hàng, xem lịch sử đặt hàng đối với người dùng bình thường.



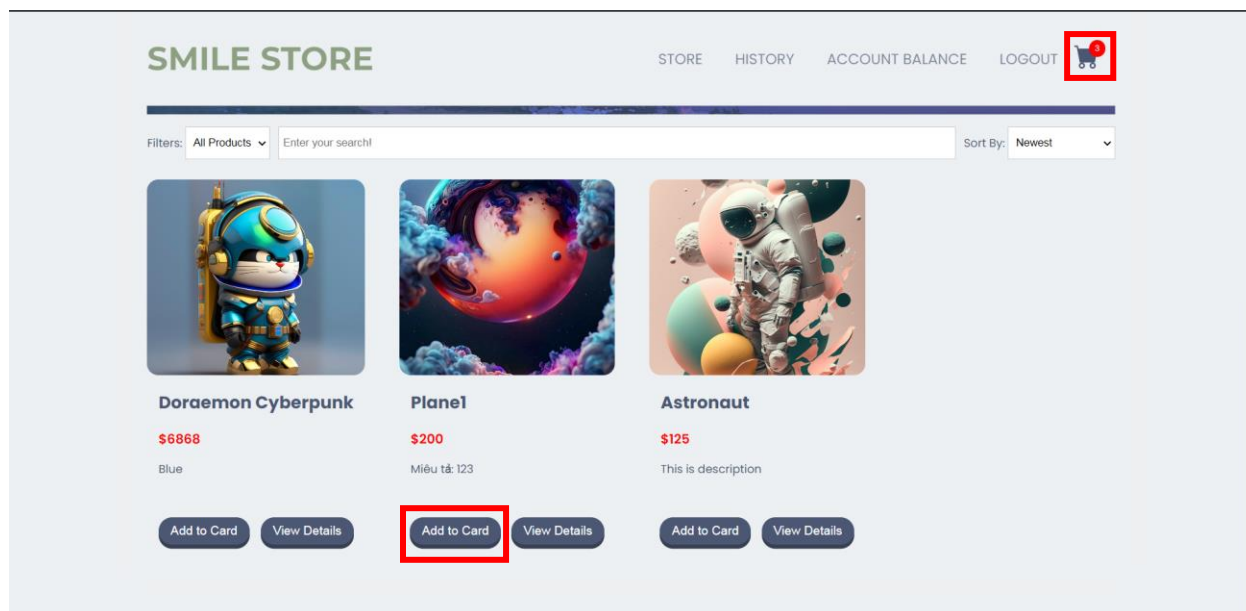
Hình 54 Giao diện chính phía người dùng



Hình 55 Giao diện chính phía quản trị viên

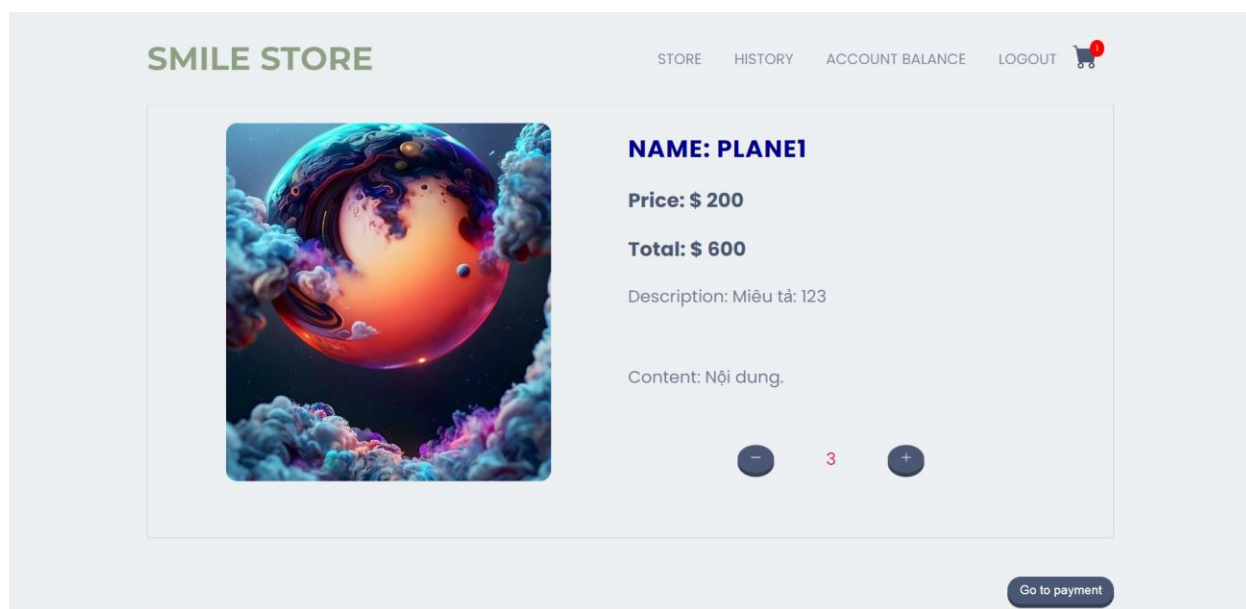
Quy trình đặt hàng đối với người mua hàng

1. Thêm sản phẩm vào giỏ hàng



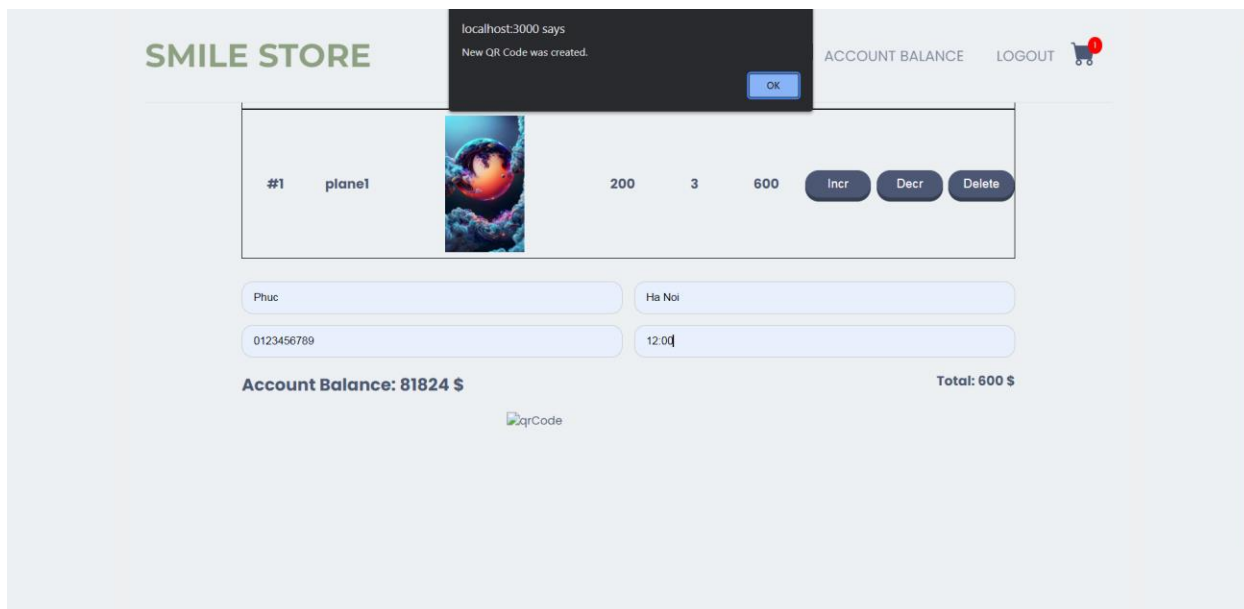
Hình 56 Thêm sản phẩm vào giỏ hàng

## 2. Vào giỏ hàng và chọn số lượng

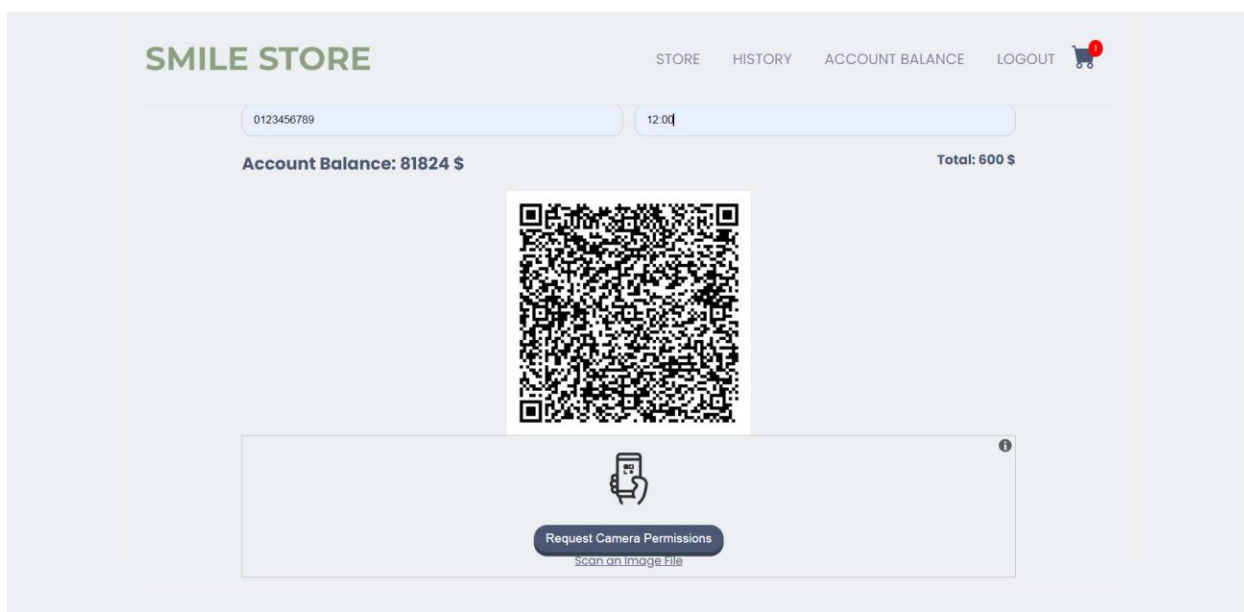


Hình 57 Tuỳ chỉnh số lượng

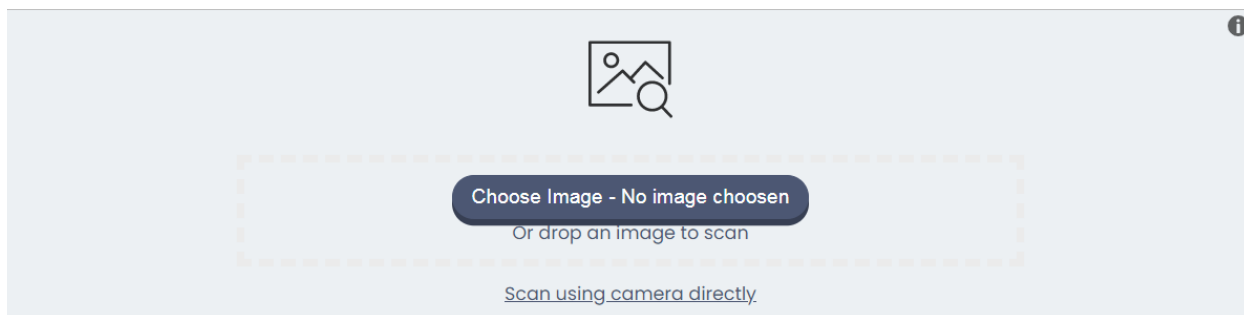
## 3. Tiến tới trang thanh toán



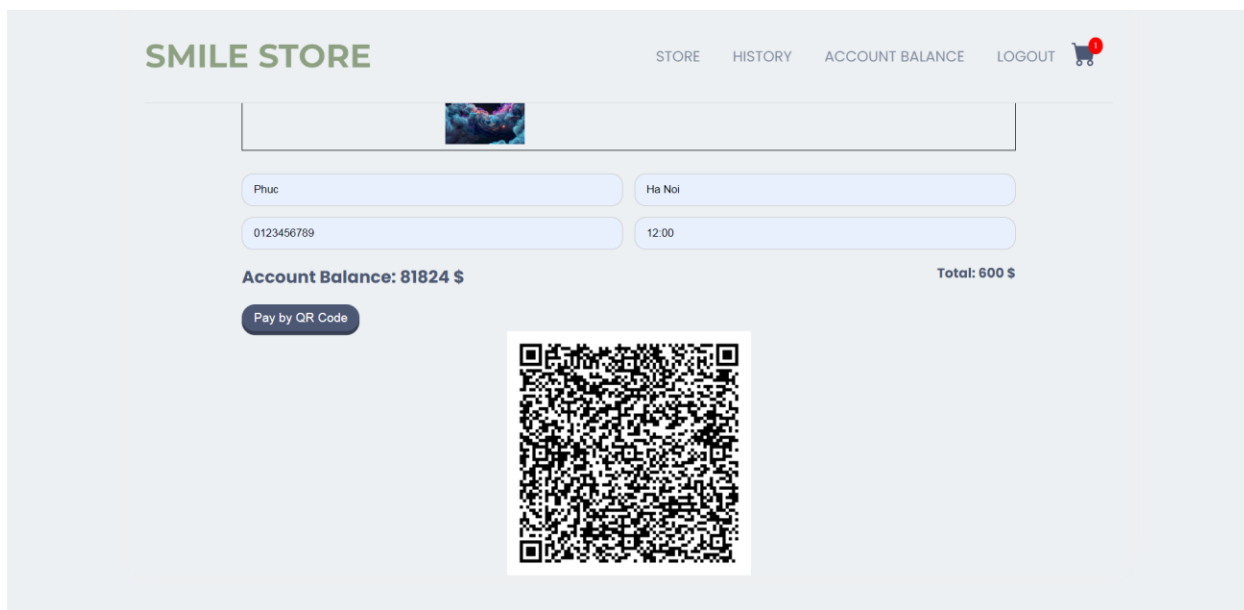
Hình 58 Sau khi nhập đủ các thông tin yêu cầu, một mã QR code sẽ được tạo



Hình 59 Mã QR code và giao diện quét mã QR code xuất hiện




Hình 60 3 tùy chọn để quét mã QR code: kéo thả, tải lên và sử dụng camera



Hình 61 Sau khi kéo thả mã QR code vào vùng chọn, xuất hiện nút thanh toán

### 3. Lịch sử đặt hàng

SMILE STORE			STORE	HISTORY	ACCOUNT BALANCE	LOGOUT	
HISTORY							
YOU HAVE 13 ORDERED							
Payment ID			Date Of Purchased				
6471a54bac46d0534f07a6e0_ima2foxu			5/29/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_ejqb7pe6			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_15kkren4			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_a7ocg35n			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_2qedlry8			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_lm0clhru			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_6dbnbmxs			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_laj9mca			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_2m0qq4au			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_eikq7zkl			5/28/2023		<a href="#">View</a>		
6471a54bac46d0534f07a6e0_jjrekzxs			5/28/2023		<a href="#">View</a>		

Hình 62 Thanh toán thành công và chuyển đến lịch sử đặt hàng

### 3.3. Tìm hiểu mã nguồn

Trước tiên, cần tìm hiểu mã QR code trên chứa những thông tin gì. Trong trường hợp này, là một website bán hàng nên mã QR code sẽ chứa thông tin đơn hàng của người đặt bao gồm các loại sản phẩm và số lượng của chúng, ngoài ra còn có thông tin của người nhận như tên, địa chỉ, số điện thoại và thời gian nhận hàng.

Ví dụ một mã QR code như sau:





aa63bxVprrdvFWmut2w8vqUwVvk8onVUwVvk8pJxaTyhMpU8UTFN6m8oXJS8YbKExWTylQxqZ  
xUTCqTyiepnFRMKIPFicpJxRMqU8VU8YTKJ1V808Vaa63bxVprrdvFWmut2w9fVvFJKk9UPFEx  
qTyhMlVMKk+oTBWTyknFpHJS8YTKVDGpTBVPqJyonFRMKicVk8pUMamcVDxR8YbKVDGpnF  
ScqDxR8YTKEyeTxRsXa621bhdrrbVuF2uttW72iw9SeaNiUpkqTITeqDhRmSomlZOKSWWq+CS  
VqWJSmSomlaniCZWTihOVqeJE5aTiDZVPqphUnqh4Q2WqmFROKv5LKIPFGxdrbVuF2uttW4  
Xa621bj98WcWk8kTFN1WcqJyoTBUnKicqU8UTKicqU8Wk8oTKScU3qUwVT6icVEwV31RxonK  
iMlVMKk9UnKhMFU+oPFHxTRdrbVuF2uttW4Xa621bj/8sYpJZVI5qZhUvqniCZWpYqqYVkaKS  
eWk4qRiUpIUpooTlaliUpIu3qiYVkaKE5WTiknlROWk4kRIqphUTireqJhUJpWpYII5UTmpmCr+  
JRdrbVuF2uttW4Xa621bvaLD1L5plpJZao4UTmpOFGZKiaVqeIJIJZOKSeWkYIJ5o+JEZaqYVkaK  
E5WpYII5qZhUpopJZaqYVkaKJ1SeqJhUpopJ5aTim1S+qWJSmSo+6WKttdbtYq211u1irbXWzX  
7xgspUMalMFZPKVDGpnFScqEwVT6hMFScqU8WkclLxhMpJxaTyRsWJylRxoJjVnKhMFScqU8  
UTKp9UcaLyRsWkMlVMKicVk8onVUwqb1S8cbHWWut2sdZa63ax1IrrZr/4IJWTijdU3qg4UZkq  
JpWp4gmVqeJE5ZMqJpWTihOVqWJSOamYVD6pYIJ5o+IJlaniCZU3KiaVqeKTVKaKSeWJiknlpO  
KNi7XWWreLtdZat4u11lq3H15SeUJlqphUnqiYVkaKT6r4L1U8ofJExRMVk8pJxaQyVUwqU8Ub  
FZPKX1KZKk4qnlCZVkaKJ1SmipOKJyomlUllqphUPulirbXW7WKttdbtYq211s1+8UEqn1RxovJE  
xaQyVUwqJxUnKIPFpDJVTCpPVEwqU8WkclJxoJjVTCrfVDGpTBWTyknFpHJScalyVTyhMIWcqE  
wVJypTxYnKv6Tiky7WWmvdLtZaa90u1lpr3ewXL6hMFZPKJ1WcqJxUnKhMFZPKScUTKk9UfJL  
KVDGpTBUnKIPFicpU8UkqU8UbKIPFpDJVTCpTxaQyVZyoTBWTyhsVk8obFZPKJ1W8cbHWWu  
t2sdZa63ax1IrrZr/4lpWpYII5qThReaPiL6IMFZ+kcIlxqZxUTCpTxRMqT1S8oTJVnKicVEwqT1RM  
KIPFpHJSMalMFZPKVDGpnFRMKicVk8pU8YbKVPHGxVprrdvFWmut28Vaa62b/eKDVkaKSeW  
kYII5qZhUpopJ5ZMqJpWp4kRIqjhROal4QmWqmFSmihOVk4pJZaqYVE4qTIROKp5QmSpOVE4  
q/ksqT1RMKIPFEypTxaTyRMUbF2uttW4Xa621bhdrrbVu9osXVL6pYIJ5o+JEZao4UZkqTISmikl  
qphUpopJ5aTiCZWtiidUTio+SeWk4kTliYpvUnmiYIKZKp5QmSqeUDmp+C9drLXWul2stda6Xay  
11rrZL15Q+aSKSeWk4kTlpGJSmSo+SWWqOFGZKk5U3qh4Q+WkYIJ5o2JSmSq+SeWkYIKZKiaV  
qeINlaniRGWqeEPLiYoTlZOKNy7WWmvdLtZaa90u1lpr3ewX/yGVqeIJIaliUjmpOFF5ouINIU+q  
mFSeqJhUpopJ5aTiCZUnKiaVk4pJZaqYVE4qnlCZKiaVk4pJZar4JJWp4r+kMIW8cbHWWut2sdZ  
a63ax1IrrZr94QeWk4kTlmyomlW+qOFH5X1JxoJjVPKEyVUwqU8UnqXxTxaTyRsWkMIU8oXJS  
caLyL6l442KttdbtYq211u1irbXW7YeXKp5QOal4QmWqeKLiROWk4kTlpOJE5YmKJ1ROVkaKT6  
o4qZhU3qg4qXhCZao4qXhCZVKZKiaVqWJSOamYVkaKk4onVE4q/tLFWmut28Vaa63bxVprds  
PH6YyVUwVvk8qJylRxoJjVnKicVEwqJypvqJxUTConKIPFScWJylQxqZxUTCqfVPFJKIPFicpUMak8  
UXGiMIWcVJyonKg8oTJVnFRMKk9UvHGx1IrrdrHWWut2sdZa6/bDI6IMFU9UPFFxUjGpTBWT  
yhsVk8pJxYnKExWfVPFExRMVT6icqLxR8UTFpDJVvKEyVZyovFFxonJS8UbFpPJNF2uttW4Xa62  
1bhdrrbVu9osXVP6XVTyh8kbFpDJVPKHYSRVPqEwVn6TyRsWJyidVTCqfVPFJKk9UTCqfVDGpP  
FHxxsVaa63bxVprrdvFWmut2w8fVjGpTBWfpDJVnKhMKicV/xKVNyomIROVqWKqeENlqpgqJ  
pWpYIKZVkaKqeJfUvGEyidVTCqfVDGpPFHxTRdrbVuF2uttW4Xa621bj98mMobKicVJyrfVDG  
pnFS8oXJS8YTKVHFS8YTKExUnKIPFpDJVvKEyVUwqT1RMKIPFpDJVnFScqEwVk8qk8i9TOal44  
2KttdbtYq211u1irbXW7YeXKiaVT6qYVE4qJpWp4gmVqWKq+CSVJ1SmijdUTiomlZOKJ1Smiknl  
CZWpYIJ5ouKbKp5QmSqmpOKJ1Smiknlkyr+0sVaa63bxVprrdvFWmutm/3ii1ROKiaVJyomlZO  
KSWWqeELIjYoTlaliUpkqPkllqvglaliUpkq3ICZKiaVqWJSOak4UTmpmFROKiaVk4o3VE4qTlSm  
ihOVqWJSmSreuFhrrXW7WGutdbtYa611++HDVE4qnqiYVCaVJ1ROVE4qTireUJkqPknIk1Smik

+qmFSmihOVE5U3KiaV4oTlaliUplUpopJZVKZKiaVqWKqeELICZUnKj7pYq211u1irbXW7WKtt  
dbthw+rmFQmIZOKSWWqOFE5qZhUpopJZVKZKiaV4pJZaqYVkaKJ1SeqPgkISdU3ICZKp5QOa  
k4qZhUJpU3KiaVSeWk4pNUTiqeqDhRmSo+6WKttdbtYq211u1irbXWzX7xQSpTxaTyRMWkMI  
W8oXJSMak8UTGpTBWTylQxqUwV4k8pJxYnKJ1V8kspJxaQyVZyoTBUUnKicVT6g8UXGi8kTFpDJ  
VPKHrRsWJylTxxsVaa63bxVprdrvFWmutm/3iBZU3Kk5UpoonVE4qJpWTikllqnhCZap4QuWTK  
iaV4oTlaliUpkqJpWTikllqphUvqniCZWpYII5qZhUTir+ZSonFZPKVPHGxVprdrvFWmut28Vaa6  
2b/eKDVE4qJpWp4ptUpopJ5ZMqJpWp4kTlpOIJlaliUnmj4kRlqphUpooTlaliUpkqJpWTikllqph  
UnqiYVkaKN1SmiknliYonVkaKSeWk4i9drLXWul2stda6Xay11rrZL/6QyhsV4k8pJxaTySRVPqEw  
V4k8pU8UkqJxUnKm9UTCpTxaQyVUwqJxWTyknFEypTxYnKVPGEyITxhspU8YTKScUTKicV33Sx  
1lrrdrHWWut2sdZa6/bDSypVEwqU8UbFW9UTCpTxYnKVHGiMIVMKIPFGxVPqEwV4k8pU8UT  
FpPJNFZPKicobKIPFpDJVTConKicVJxVvVEwqk8pUMaIMFScqU8UnXay11rpdrLXWul2stda62S  
9eUJkqPknIjYq/pPJGxRMqU8WkcllxqTxRMamcVEwqB1RMKIPFpHJS8YbKVHGi8kbFicpJxaQy  
VXySylRxoJVTCPtxRsXa621bhdrbVuF2uttW4/vFQxqUwVb1Q8ofJJKIPFScUbKIPFpHKiMIU8  
UTGpvFHxRsWkMqmcqEwVJypTxaQyVUwV4k8pJxaQyVZyoPFHx1SeUJkq/tLFWmut28Vaa63b  
xVprdrsPL6IMFZPKVDGpTBWTylTtSpTxaQyVUwqB1ScVEwqU8WkMIVMKIPFScVJxYnKVHGi  
MIWcqEwVJyonKIPFpDJVTBUUnKicqU8VUMak8oTJVTConFZPKN6IMFZ90sdZa63ax1lrrdrHWW  
uv2w0sVT6icqEwVJyonKicVn1RxovKEyhMqT1RMKk+oTBUUnFW+ofFLFicpfqphUJpWp4qRiUp  
kqJpWpYII5qThRmVSmiknIRGWqeONirbXW7WKttdbtYq211u2HP1YxqZyoPFHxRsUbKIPFScW  
kMIWcqDyh8kkVT6g8UfGEyqTySSonKIPFpPJGxUnFScVJxUnFpPKXKiaVT7pYa611u1hrrXW7W  
Gutdfvhj6mcVDyh8oTKN1U8oTJVTCPtxUnFEypTxYnKVPFExYnKpHJScVlxqZxUPKEyVZxUTC  
oFScqJxVvqEwVJxVPqJxUTCrfdLHWWut2sdZa63ax1lrr9sM/TmWqeELIjYo3VE4qnIB5QmWq  
OFH5JJWpYIKZKk5UTiomiLaliUjIRmSpOVE4qpopJZao4qXhD5QmVJ1SmijcqJpVPulhrrXW7WG  
utdbtYa611++EllanipGJSOal4QmWqOFF5Q+WTVKaKE5WTim9SOak4qXhD5aRiUnmi4omKJ1  
ROVE4qTIROKp5QeaLiDZW/dLHWWut2sdZa63ax1lrrZr/4D6l8UsWkMIV8kspfqphUPqniRGW  
qmFROKiaVqeJEZao4UfkvVUwqU8UbKicVJyonFZPKX6r4SxdrrbVuF2uttW4Xa621bj98mcpUM  
VVMKIPFv6xiUjpmFmihOVqWJSmsqeUDmpeKLiL6mcVJyoTBUUnKIPFScUTKIPFDGpPFHxR  
sWk8kTFpPJfulhrrXW7WGutdbtYa611++ElIZOKE5WpYIKZKk5UpoonVkaKNyomlaniRGWqmF  
SmiknlpOKbVE4qnqh4Q+UNlaliUnmiYIJ5QmWqmFROKiaVqWKqOKI4QmWqmFSeqHjjYq211  
u1irbXW7WKttdbNfvE/ROWNihOVqeJEZao4UZkqJpUnKiaVqWJSOamYVkaKSeWk4ptUpopJZ  
ap4QmWqeEJlqnhC5ZsqTIROKk5UpooTlaniL12stda6Xay11rpdrLXWuv3wkspJxaQyVUwqT1R  
MKIPFpHJS8UTFpHJSMan8pYoTlScqnIB5omJSmSomlSdUpoqp4kTICZUnKv6SyknFicpUcalyVZ  
yoTBWfdLHWWut2sdZa63ax1lrrZr/4IJUnKiaVqWJSOal4QmWqOFGZKiaVqWJSmSomlTcqTIS  
mihOVqeJEZaqYVkaKN1SmikllqjhRmSomlZOKT1I5qZhU3qiYVJ6oeEPLiYpPulhrrXW7WGutdbt  
Ya611s1/8w1SmiidUnqh4QuWkYIKZKp5QOamYVkaKE5WTijdUnqiYVkaKSeVfUvGEyknFpDJV  
vKEyVZyoTBWTyhmV4k8oTFW9crLXWul2stda6Xay11rr98GEqn1QxqTxRcalyqZxUTBVPVJyonF  
RMKicV4k8pJxTdVTCpTxV+qOFGZKiaVqWJSmSomlZOKSWWqmFTEqJhUnlCZKp5QmSomlW+6  
WGutdbtYa611u1hrrXX74SWVqeJIROV4k8pJ5ZMqJpWpYIJ5o+KbKk5U3qg4UZkqnqiYVN5QO  
ak4qZhUpopJZaqYVE4q3qh4omJSmSpOVkaKJ1Smim+6WGutdbtYa611u1hrrXX74Y+pTBVPqE  
wqT6j8lYpOVkaKSWWqmFSmik+qmFQmlaliqnhD5aRiUpkqJpUTlaliUpkqTiomiLaliUjmpeELlp  
OKkYIKZKk5Unqj4SxdrrbVuF2uttW4Xa621bj+8VDGpPKEyVUwqU8WkMIVMKIPFpHKicqlyVZy

oTBUnt+kMIWcqJxUTConFZPKScWkMqIMFW9UTCpTxaTyRMWkMIU8UTGpTBWTyqTylypO  
VJ6oeONirbXW7WKttdbtYq211s1+8YLVKVDGpvFHxhMobFScqT1S8oTJVPKEyVUwqU8U3qUw  
VT6icVJyoTBUntKIPFicJxaQyVUwqT1RMKm9UnKicVEwqJxX/pYu11lq3i7XWWreLtdZaN/vFCy  
pTxRsqt1RMKIPFpPJExaRyUjGpfFLFpDJVnKhMFZPKExWTylQxqUwV8k8pU8YTKScWk8kTFpHJ  
SMaIMFW+oTBVPqDxRcaLyTRWTylTxxsVaa63bxVprdrvFWmutm/3iBZWp4kTliYpJ5aTiRGWq  
mFSmihOVb6r4JJUnKiaV64oTIZOKE5VPqphUpooTlaniL6mcVEwqJxWTylTxxSpTxYnKVPFJF2u  
ttW4Xa621bhdrbVu9osXVE4qJpWpYII5qThRmSomlZOKN1SmikllqphUTipOVKaKSeWTKt5Q  
mSomlanim1ROKk5UpopJ5YmKE5Wp4kTlkyomlZOKKE5WpYII5qXjjYq211u1irbXW7WKttdbth  
5cq3ICZKk5UTiomlaniCZWp4gmVqeKJiicqJpWTihOVE5WpYII5QuWTVJ6omFQ+qeINlaliUnmj  
4kRIUpkqnIB5omJS+aSLtdZat4u11lq3i7XWWj7xQsqU8UTKIPFpPJFScqU8WJylTxhMoTFZPK  
X6qYVKAkV6RyUnGi8i+peEPlpGJSeaJiUvmXVLxxsdZa63ax1lrrdrHWWutmv/gfpjJVTCPtXaQy  
VXySyknFpHJSMaIMFU+oTBXfphJScalyVZyoPFHxhMpUMamcVJyofFLFGyonFU+onFRMKIPFJ  
12stda6Xay11rpdrlXWuv3wkspfqpgqTiomlaliUpkqJpWpYIKZKiaV64pPUpkqTISeqJhUTiomlR  
OVJ1SmiknlCZWp4kTIDZWp4kRIqphUTISeqJhUTISmiidUTISmijcu1lpr3S7WWmvdLtZaa91++L  
CKT1I5UXmiYIJ5omJSOVGZKiaV64p4omJSmSpOKp6oeKJiUpkqPqniiYo3KiaVqWKqOKmYVP5S  
xRMV/6WLtdZat4u11lq3i7XWWj7xQsqU8Wk8kTFpDJVTCqfVHGIMIVMKIPFpHJSMa8pYpv  
Ujnp+Esqn1RxonJSMaIMFScqU8WJylQxqfxLKiaVqeKni7XWWreLtdZat4u11lq3H/6Pq3hCZao  
4UZkqTipOVE4qJpWp4gmVE5Wp4n+JylTxRMWkcqJyUnGiMIVMKicVJyonKicVJyonFZPKScVfu  
lhrrXW7WGutdbtYa611++H/mlpJ5YmKSWWq+CSVqWJSmVSmiknlkyqeUDmpmFROVKAKE5  
Wp4qRiUplUpopJZaqYVCaVqeKJihOVk4pJ5aTiRGWqeKJiUplUpopvulhrrXW7WGutdbtYa611  
++HLKr6pYIJ5oulNlaliUpkqpopPqjhRmSomlaniRGWqmFQmlaliUpkqJpWTijcqJpU3Kp6omFS  
mim+qmFSmiidUTipOKv7SxVprdrvFWmut28Vaa63bDx+m8pdUTiqeUJkqTiomlW+qmFSmikll  
qphUpooTlaliUpkqJpVJ5Y2KJ1SmiknlROUJlaliUpkqTISmiidU3ICZKk4qTISmiknl12stda6Xay1  
1rpdrlXWutkv1lprcbHWWut2sdZa63ax1lrrdrHWWut2sdZa63ax1lrrdrHWWut2sdZa63ax1lrr  
drHWWut2sdZa63ax1lrrdrHWWut2sdZa63ax1lrr9v8AgUxHDv6LOYsAAAAASUVORK5CYII=

Cấu trúc của URI QR Code dạng data bao gồm các thành phần sau:

### 1. Tiền tố "data:"

- Đây là tiền tố bắt buộc để chỉ định rằng đây là dữ liệu dạng data.

### 2. Loại dữ liệu (MIME type):

- Sau tiền tố "data:", tiếp theo là loại dữ liệu (MIME type) của nội dung được mã hóa.
- Ví dụ: "text/plain" cho văn bản thuần, "image/png" cho hình ảnh PNG, "application/pdf" cho tệp PDF, v.v.

### 3. Kiểu mã hóa (encoding):

- Tiếp theo sau loại dữ liệu là kiểu mã hóa (encoding) được sử dụng để mã hóa dữ liệu.

- Có hai kiểu mã hóa phổ biến là "base64" và "utf-8".

#### 4. Dấu phân cách (;):

- Sau loại dữ liệu và kiểu mã hóa, sẽ có một dấu phân cách là dấu chấm phẩy (;) để ngăn cách với nội dung dữ liệu.

#### 5. Nội dung dữ liệu:

- Cuối cùng, sau dấu phân cách, là nội dung dữ liệu thực tế được mã hóa theo kiểu mã hóa đã chỉ định.
- Ví dụ: văn bản, hình ảnh, tệp tin, v.v.

Ngoài dạng data, QR code còn có dạng URL trong trạng thái query, nhưng có lẽ không nên dùng vì sẽ lộ API.

Tiếp đến tìm hiểu về luồng hoạt động. Sau khi người dùng đã nhập đủ các thông tin yêu cầu hợp lệ, ứng dụng sẽ gửi toàn bộ dữ liệu về phía server thông qua giao thức POST đến API paymentQR qua hàm **generateQRCode()**:

```
const generateQRCode = async () => {
  try {
    const res = await axios.post('/api/paymentQR', { cart: [...cart], address, receiver, phone, time, total }, {
      headers: { Authorization: token }
    });
    setOldSrc(src)
    setSrc(res.data)
    alert("New QR Code was created.")
  } catch (err) {
    alert(err.response.data.msg)
  }
}
```

Hình 64 Hàm **generateQRCode()**

Hàm này nếu thành công sẽ gán giá trị data cho biến src, nếu sai thì thông báo lỗi. Mục đích của việc có 2 giá trị src và oldSrc là cập nhật mã QR code khi đơn hàng có sự thay đổi từ người dùng.

Đi tới server, cụ thể tại router, khi nhận được request tới API /paymentQR, server sẽ gọi đến hàm **generateQRCode** trong controller để xử lý:

```

const router = require('express').Router()
const paymentCtrl = require('../controllers/paymentCtrl')
const auth = require('../middleware/auth')
const authAdmin = require('../middleware/authAdmin')

router.get('/payment', auth, authAdmin, paymentCtrl.getPayments)
router.post('/payment', auth, paymentCtrl.createPayment)
router.post('/paymentQR', auth, paymentCtrl.generateQRCode)

module.exports = router

```

Hình 65 Server nhận request và bắt đầu xử lý

Tại payment controller, hàm generateQRCode() sẽ tiếp nhận dữ liệu từ request body tức dữ liệu người dùng nhập vào. Trước hết sẽ xác thực người dùng, sau đó tất cả dữ liệu sẽ được chuẩn hoá sang định dạng JSON thuận tiện cho việc truyền tải dữ liệu giữa server và client. Thư viện **qrcode** đã được import phía trên, gọi đến hàm **toDataURL()**. Giá trị trả về src chính là giá trị URI QR code dạng data đã được nêu phía trên.

```

generateQRCode: async (req, res) => {
  try {
    const user = await Users.findById(req.user.id).select('name email')
    if (!user) return res.status(400).json({ msg: "User does not exist." })
    const { cart, address, receiver, phone, time, total } = req.body;
    const newCart = cart.map((el, i) => (
      {
        "id": i + 1,
        "idProduct": el._id,
        "name": el.title,
        "price": el.price,
        "quantity": el.quantity,
        "sold": el.sold,
        "url": el.images.url
      }
    ))
    const { _id } = user;

    const dataQRCode = JSON.stringify({ userId: _id, receiver, cart: newCart, address, phone, time, total })
    const src = await QRCode.toDataURL(dataQRCode)

    res.json(src)
  } catch (error) {
    return res.status(500).json({ msg: error.message })
  }
},

```

Hình 66 Hàm generateQRCode() phía server tại controller

Trở lại phía client, sau khi hàm giá trị data gán vào biến src, một vùng chọn cho phép quét mã QR code hiện ra. Đối tượng scanner này được kế thừa dựa trên đối tượng Html5QrcodeScanner là một API của thư viện **Html5-qrcode**, nó sẽ tạo ra vùng chọn

với thẻ có id reader đã được chỉ định. Với hàm success, tức là khi quét thành công, giá trị trả về (đã được trình bày phía trên) sẽ được gán cho biến scanResult và xóa phần vùng chọn đi.

```
useEffect(() => {
  if (oldSrc !== src) {
    const scanner = new Html5QrcodeScanner('reader', {
      qrbox: {
        width: 250,
        height: 250,
      },
      fps: 5
    });

    scanner.render(success, error);

    function success(result) {
      setOldScanResult(scanResult)
      setScanResult(result)
      setOldSrc(src)
      scanner.clear();
    }

    function error(err) {
      console.warn(err)
    }
  }
}, [oldSrc, src])

const generatePaymentId = () => {
  return Math.floor(Math.random() * Date.now()).toString(36);
};
```

Hình 67 Đối tượng scanner

Sau khi đã có giá trị scanResult, ứng dụng sẽ so sánh scanResult với oldScanResult để hiển thị nút thanh toán cho người dùng, tránh trường hợp khi người dùng thay đổi thông tin đơn hàng, nút thanh toán vẫn là đường dẫn chứa thông tin cũ.

```
useEffect(() => {
  if (oldScanResult !== scanResult) {
    setOpenPayment(true)
  } else {
    setOpenPayment(false)
  }
}, [oldScanResult, scanResult])
```

Hình 68 So sánh 2 giá trị scanResult và oldScanResult để hiển thị nút thanh toán

Khi giá trị openPayment là true tức là nút thanh toán đã được hiển thị, người dùng nhấn vào sẽ gọi đến hàm **tranSuccess()** chứa dữ liệu đơn hàng.

```
const tranSuccess = async (payment) => {
  if (balance < total) {
    alert("Your balance is not available.")
  } else {
    var payment = JSON.parse(payment);
    const paymentID = payment.userId + '_' + generatePaymentId();
    const { address, cart, receiver, phone, time, total } = payment;
    console.log('test');
    await axios.post('/api/payment', { cart, paymentID, address, phone, time, receiver, total, balance }, {
      headers: { Authorization: token }
    })

    setCart([])
    addToCart([])
    alert("You have successfully placed an order.")
    window.location = '/history'
  }
}
```

Hình 69 Hàm **tranSuccess()**

Hàm này sẽ kiểm tra số dư tài khoản, sau đó chuẩn hoá giá trị payment được truyền vào từ dạng JSON trở về thành 1 đối tượng. Giá trị paymentID được tạo ngẫu nhiên, và sau đó kết hợp các thông tin người đặt hàng, đơn hàng gửi lên server.

Tại server ở phần controller, hàm **createPayment** sẽ được gọi để xử lý. Hàm này sẽ xác thực người dùng, kiểm tra số dư, sau đó tạo một đối tượng đơn hàng mới, cập nhật số lượng sản phẩm đã bán và số dư người dùng.

```

createPayment: async (req, res) => {
  try {
    const user = await Users.findById(req.user.id).select('name email')
    if (!user) return res.status(400).json({ msg: "User does not exist." })
    const { cart, paymentID, address, phone, time, receiver, total, balance } = req.body;

    const { _id, email } = user;
    if (balance < total) {
      res.status(400).json({ msg: "Balance is not available." })
    } else {
      const newPayment = new Payments({
        user_id: _id, name: receiver, email, cart, paymentID, address, phone, time, total, status: true
      })

      cart.filter(item => {
        return sold(item.idProduct, item.quantity, item.sold)
      })

      await newPayment.save()
      updateBalance(req.user.id, balance, total)
      res.json({ msg: "Payment Success!" })
    }
  } catch (err) {
    return res.status(500).json({ msg: err.message })
  }
},

```

Hình 70 Hàm `createPayment`



## **Tài liệu tham khảo**

- [1] [QR Code Tutorial: Introduction](#)
- [2] [https://en.wikipedia.org/wiki/QR\\_code#Design](https://en.wikipedia.org/wiki/QR_code#Design)
- [3] <https://www.qr-code-generator.com/qr-code-marketing/qr-codes-basics/>
- [4] <https://thegioimavach.com/qr-code-dong-bien-doi-qr-tinh-la-gi-so-sanh>
- [5] <https://www.qrcode.com/en/about/version.html>