BEJOM

Brazilian Electronic Journal of Mathematics. 2025, v.6: 74898

https://doi.org/10.14393/BEJOM-v6-2025-74898

ISSN: 26751313 (online edition)

Matemática Aplicada

NÚMEROS BINÁRIOS, CORPOS FINITOS E DIVISÃO POLINOMIAL, CONSTRUINDO UM QR CODE

José Laudelino de Menezes Neto
Universidade Federal da Paraíba, Departamento de Ciências Exatas, Rio Tinto, PB, Brazil.
E-mail: laudelino@dcx.ufpb.br
https://orcid.org/0000-0002-9988-8569

Mathematics Subject Classification (MSC): 00A69, 11T06, 00A09.

Resumo. Por meio de uma revisão bibliográfica, apresentamos todos os processos de como é feito um QR Code modelo 1, versão 1, e as teorias matemáticas envolvidas, sendo possível a criação de um QR Code funcional. Comentamos acerca da escrita de dados na forma de números binários, a correspondência entre escrever um número em base dez e em base dois (binário). Damos ênfase no assunto de corpos finitos, com uma visão polinomial, sendo este tema necessário para escrever os dados do código corretor de erros em um QR Code.

Palavras-chave. Corpos finitos, polinômios, divisão euclidiana.

BINARY NUMBERS, FINITE FIELDS AND POLYNOMIAL DIVISION, MAKING A QR CODE

Abstract. Through a bibliographical review, we present all the processes of how a QR Code model 1, version 1, is made and the mathematical theories involved, making it possible to create a functional QR Code. We comment on how to write the data in the form of binary numbers, the correspondence between writing a number in base ten and in base two (binary). We emphasize the subject of finite fields, with a polynomial view, this subject being necessary to write the error correcting code data into a QR Code.

Keywords. Finite field, polynomial, euclidean division.

NÚMEROS BINARIOS, CAMPOS FINITOS Y DIVISIÓN POLINOMIAL, LA CONSTRUCCIÓN DE UN CÓDIGO QR

Resumen. Mediante una revisión bibliográfica, presentamos todos los procesos involucrados en la creación de un Código QR modelo 1, versión 1, así como las teorías matemáticas que lo hacen posible, permitiendo la generación de un Código QR funcional. Explicamos cómo se escriben los datos en forma de números binarios y la correspondencia entre la escritura de un número en base diez y en base dos (binario). Destacamos el tema de los campos finitos desde una perspectiva

polinomial, siendo este fundamental para la codificación de datos de corrección de errores en un Código QR.

Palabras clave. Campo finito, polinomio, división euclidiana.

1 Introdução

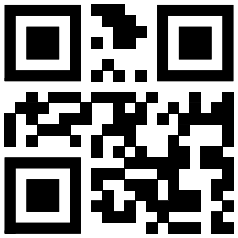
Os QR Codes se tornaram bastante utilizados nos dias atuais, podemos citar as seguintes utilidades: os pagamentos via PIX, divulgação de páginas da internet, modo para se conectar em alguma rede Wi-Fi etc. Porém, você já se perguntou como é a construção de um QR Code? Neste artigo temos o objetivo de mostrar como construir um QR Code e os objetos matemáticos envolvidos, com destaque para corpos finitos e divisão polinomial, utilizados na ferramenta de correção de erro. Existem trabalhos que tratam da estrutura do QR Code e aplicações em sala de aula [1, 2, 3, 4].

A criação do QR Code, ou *Quick Response code*, em uma tradução livre "código de resposta rápida", ocorre no ano de 1994 pela empresa japonesa *Denso Wave*, sua aplicabilidade é o armazenamento de dados de forma simples por meio de uma imagem, e surgiu como uma alternativa aos códigos de barra, mas com possibilidade de maior capacidade de acúmulo de informações [5]. Além disso, os QR Codes possuem uma estrutura de correção de erros, utilizada para caso alguma parte da imagem do QR Code esteja encoberta, ou danificada.

Nos trabalhos [1, 2, 3, 4] é tratado o tema de QR Code com um viés educacional, ensinando as estruturas de um QR Code e como escrever os dados, porém não explica como escrever os dados do código de erro no QR Code. No texto [6] é dado ênfase na parte teórica de códigos corretores de erro, é citado como aplicação a utilização em QR Codes, porém não é detalhado como funciona sua escrita em um QR Code. Portanto, aqui neste artigo explicamos toda estrutura na criação de um QR Code, inclusive a metodologia na escrita do código corretor de erro.

Um exemplo de QR Code contendo a mensagem "Calculo" é exibido na Fig. 1. É necessário um aplicativo para fazer a leitura deste QR Code da Fig. 1, explicações sobre aplicativos leitores de QR Codes podem ser vistas em [1].

Figura 1: QR Code modelo 1, versão 1, com a mensagem "Calculo".



Fonte: Autoria própria.

Por opção, de modo a não entrar em tantos detalhes técnicos, nos restringimos ao QR Code modelo 1, versão 1, com mensagens de até sete caracteres, escritas no modo byte. A criação de QR Codes de outros modelos, versões e utilizando mais caracteres nas mensagens, seguem o mesmo padrão, com algumas mudanças técnicas. O motivo de optarmos por estas restrições, é manter nosso foco em exibir toda matemática por trás de um QR Code, as teorias de números binários, corpos finitos e divisão polinomial.

O restante deste artigo é composto dos seguintes tópicos: na seção 2 se explica toda estrutura de um QR Code modelo 1, versão 1, como é feita a escrita de uma mensagem em um QR Code, e os diferentes modos de dados aceitos em um QR Code; seção 3 trata do tema de Corpos Finitos, com uma abordagem via polinômios, focando no que é necessário em um QR Code; e a seção 4 explica como se escreve o código corretor de erro em um QR Code, sem entrar em pormenores técnicos de funcionamento.

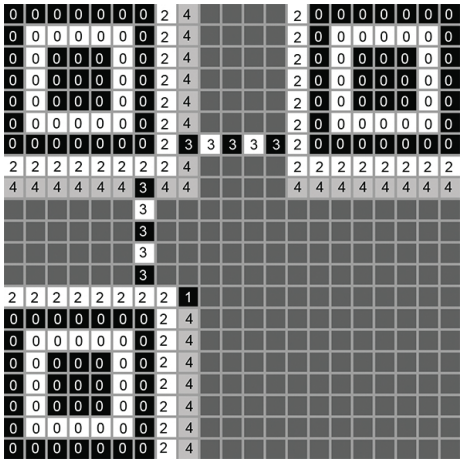
2 Estrutura de um QR Code

Existem vários modelos e versões de QR Code, onde, de forma geral, a grande diferença é a capacidade de armazenamento de dados [7, 4]. Aqui neste trabalho vamos nos ater a construção de um QR Code de modelo 1 e versão 1. De mogo geral, os quadradinhos pretos e brancos que compõem o QR Code são chamados de módulos, um QR Code modelo 1, versão 1, tem tamanho 21×21 módulos.

Algumas regiões de módulos são inerentes ao QR Code por padrão, vamos elencar as regiões do QR Code modelo 1, versão 1, estas regiões são dadas a seguir [8].

- Região de procura (módulos numerados com zero na Fig. 2);
- Módulo escuro (módulo numerado com 1 na Fig. 2), chamado em inglês de *dark module*. É um único módulo na cor preta posicionado estrategicamente, existe uma fórmula para posicionamento deste módulo escuro em cada modelo e versão de QR Code [1, 8].
- Região de espaçamento (módulos numerados com 2 na Fig. 2).
- Região de tempo (módulos numerados com 3 na Fig. 2). São linhas pontilhadas para conectar as regiões de procura. Estes módulos permanecem com a coloração padrão apresentada (veja também na Fig. 3).
- Região reservada para escrita de informações a respeito do QR Code (módulos numerados com 4 na Fig. 2).
- Região onde são escritos os dados a serem codificados (região de módulos sem numeração, pintados na cor cinza escuro na Fig. 2).

Figura 2: QR Code modelo 1, versão 1. Módulos na cor preta e/ou branca são as regiões de procura (módulos numerados com zero), módulo escuro (módulo numerado com 1), regiões de espaçamento (módulos numerados com 2), regiões de tempo (módulos numerados com 3). Regiões na cor cinza clara são para definir características de modelo, versão, máscara e código de erro (módulos numerados com 4). Região na cor cinza escuro é onde se escreve os dados a serem lidos (módulos sem numeração).



Fonte: Autoria própria.

Ao redor do QR Code, deve-se deixar, no mínimo, uma borda branca com espaçamento do tamanho de um módulo.

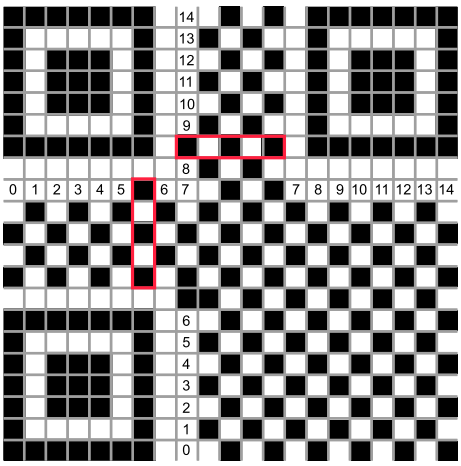
Existem oito padrões de máscara de um QR Code, enumerados de 0 (zero) a 7 (sete), que é a forma padrão de pintar a região onde são escritos os dados (região de módulos na cor cinza escuro e sem numeração na Fig. 2), ou seja, uma máscara é a coloração da região dos dados sem nada escrito nela. Mais detalhes sobre os padrões de máscara são encontrados em [1, 3, 8]. Existe toda uma metodologia para escolher a melhor máscara, detalhes são encontrados em [1, 2], porém, para não entrar em tantos pormenores, optamos por utilizar a máscara 0 (zero) na construção do nosso QR Code modelo 1, versão 1. Sendo assim, sem nenhum dado escrito, o QR Code fica conforme a Fig. 3. A numeração de 0 a 14 na Fig. 3 é a ordem de preenchimento na região reservada para escrita de informações a respeito do QR Code (região de módulos numerados com 4 na Fig. 2) [8].

Observação 1. *A aplicação da máscara só é feita na região de escrita dos dados (região de módulos cinza escuro sem numeração na Fig. 2).*

2.1 Escrita de dados e números binários

A escrita dos dados de um QR Code é feita utilizando números binários, números escritos em base dois, ou seja, números em que seus dígitos têm apenas os algarismos 0 (zero) e 1 (um).

Figura 3: QR Code versão 1, modelo 1, sem nada escrito na região de dados, e utilizando a máscara de tipo 0 (zero). A numeração de 0 a 14 é a ordem de preenchimento na região reservada para escrita de informações a respeito do QR Code. Os módulos de tempo estão destacados por retângulos vermelhos, e a coloração dos módulos de tempo permanece como está por padrão.



Fonte: Autoria própria.

Um número escrito em base dez, é a forma usual na escrita de um número, com dez algarismos, de 0 (zero) a 9 (nove), ou seja com os algarismos 0,1,2,3,4,5,6,7,8 e 9.

Aqui abrimos um parenteses explicando o método de como passar qualquer número inteiro positivo N em base dez para base dois, binário, só precisamos utilizar a divisão euclidiana, também chamada de divisão com restos. O procedimento inicial é dividir N por 2, usando a divisão euclidiana, mais precisamente o Algoritmo da Divisão de números inteiros [9]. Assim, obtemos um quociente q_0 e um resto r_0, com $0 \leq r_0 \leq 1$ e

$$N = 2 \cdot q_0 + r_0.$$

Se $q_0 = 0$, então terminamos e N escrito em binário é r_0. Caso $q_0 \neq 0$, o próximo passo é dividir q_0 por 2, obtendo um quociente q_1 e resto r_1,

$$q_0 = 2 \cdot q_1 + r_1, \quad 0 \leq r_1 \leq 1.$$

Repetimos este processo até obter um quociente $q_k = 0$, quando isto ocorre, temos que o número N escrito em binário é

$$r_k r_{k-1} r_{k-2} \ldots r_2 r_1 r_0. \tag{1}$$

Para fazer o caminho inverso, passar um número binário como em (1) para base dez, basta

fazermos a seguinte operação,

$$r_k \cdot 2^k + r_{k-1} \cdot 2^{k-1} + r_{k-2} \cdot 2^{k-2} + \ldots + r_2 \cdot 2^2 + r_1 \cdot 2^1 + r_0 \cdot 2^0.$$

Exemplo 1. *Para escrever o número* $N = 116$ *em binário, efetuamos a sequência de divisões euclidianas,*

$$\begin{aligned}
116 &= 2 \cdot 58 + 0 \qquad (q_0 = 58,\ r_0 = 0),\\
58 &= 2 \cdot 29 + 0 \qquad (q_1 = 29,\ r_1 = 0),\\
29 &= 2 \cdot 14 + 1 \qquad (q_2 = 14,\ r_2 = 1),\\
14 &= 2 \cdot 7 + 0 \qquad (q_3 = 7,\ r_3 = 0),\\
7 &= 2 \cdot 3 + 1 \qquad (q_4 = 3,\ r_4 = 1),\\
3 &= 2 \cdot 1 + 1 \qquad (q_5 = 1,\ r_5 = 1),\\
1 &= 2 \cdot 0 + 1 \qquad (q_6 = 0,\ r_6 = 1).
\end{aligned}$$

Logo, $N = 116$ *em binário é* 1110100*. Para retornar o número binário* 1110100 *em base dez, fazemos*

$$1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 116.$$

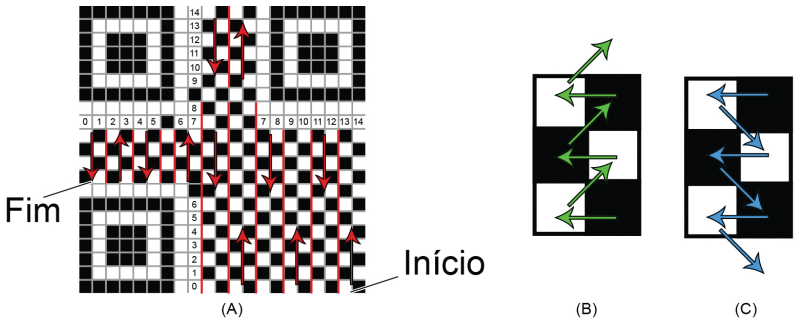
Como complementação sobre este tema de sistema binário e sistema em base dez de numeração, sugerimos a dissertação [1].

Voltando à escrita de dados em um QR Code, na região, onde se tem ou não a máscara aplicada, se formos escrever o algarismo zero, mantemos a cor do módulo como está, cor branca se o módulo por padrão for cor branca, cor preta se o módulo por padrão for cor preta.

No local onde não se usa máscara, na região da escrita de definição de um QR Code (módulos numerados de 0 a 14 na Fig. 3), se o algarismo a ser escrito for 1, então pintamos o módulo na cor preta. Já na região dos dados, onde se tem máscara (região de módulos sem numeração e cinza escuro na Fig. 2), se o algarismo a ser escrito for 1, fazemos uma inversão na cor do módulo: se o módulo estiver na cor branca, pintamos na cor preta; se o módulo estiver na cor preta, pintamos na cor branca.

A sequência de escrita dos dados na região de definição de um QR Code, os módulos numeradas de 0 a 14 na Fig. 3, segue-se a sequência de números já escrita, partindo do zero e finalizando no 14.

Figura 4: Na imagem (A), as setas vermelhas indicam o preenchimento dos dados em blocos de módulos de largura 2. Imagem (B) indica o preenchimento nos blocos de módulos de largura 2 quando a seta vermelha está subindo. Imagem (C) indica o preenchimento nos blocos de módulos de largura 2 quando a seta vermelha está descendo.



Fonte: Autoria própria.

Na região dos dados (região de módulos sem numeração e nas cores cinza escuro na Fig. 2), a escrita segue um padrão de zigue-zague, alternando de baixo para cima, e de cima para baixo, conforme as setas e as barras nas cores vermelhas da Fig. 4(A), iniciando na inscrição Início, e finalizando na inscrição Fim. O preenchimento nos blocos de módulo de largura dois, segue o padrão exibido na Fig. 4(B) quando a escrita é de baixo para cima, e o padrão da Fig. 4(C) quando a escrita é de cima para baixo [1, 2, 4, 8].

2.2 Definição de um QR Code

Vimos que tem duas regiões de módulos no QR Code onde se deve escrever as definições escolhidas quanto a modelo, versão, tipo de máscara e tipo de código corretor de erro (módulos numerados de 0 a 14 na Fig. 3). Escrevemos um mesmo número em binário pré determinado, com 15 dígitos, nestas duas regiões. Para criação do nosso QR Code, já ficou acertado que nos mantemos no modelo 1, versão 1, e usamos a máscara do tipo 0 (zero), resta definir o tipo de código corretor de erro.

O código corretor de erro utilizado nos QR Codes é o de Reed-Solomon (*Reed-Solomon error correction*), esse assunto é abordado com minuciosidade na seção 4. Existem quatro níveis de correção de erro: L, M, Q, H. Optamos pelo nível H. Portanto, com estas configurações, seguindo uma tabela pré determinada, devemos preencher na sequência de 0 a 14 apresentada na Fig. 3 o número binário 001011010001001. A geração deste número segue um padrão técnico e informações mais detalhadas sobre como gerar este número binário são encontradas na seção *Format and Version Information* da página [8].

Seguindo o padrão descrito na seção 2.1 e a sequência dada pelos números de 0 a 14 na Fig. 3, o preenchimento se dá pela Tabela 1, onde **Posição** se refere ao módulo com numeração na Fig. 3, **Dígito** se refere ao dígito do número binário 001011010001001, que representa nossas

escolhas quanto a modelo, versão, tipo de máscara e tipo de código corretor de erro, e **Cor** se refere à cor a ser pintada no módulo com B representando a cor branca, e P representando a cor preta.

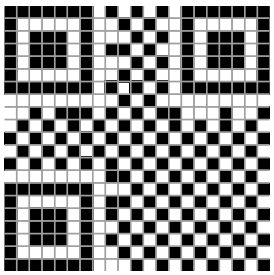
Tabela 1: Forma de preenchimento dos dígitos do número binário 001011010001001 nos módulos destacados com os números de 0 a 14 na Fig. 3. Na **Posição** n ($n \in \{0, 1, 2, \ldots, 13, 14\}$) se o **Dígito** for 0 (zero) o módulo é pintado na cor branca (B), e se o **Dígito** for 1 (um) o módulo é pintado na cor preta (P).

Posição	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Dígito	0	0	1	0	1	1	0	1	0	0	0	1	0	0	1
Cor	B	B	P	B	P	P	B	P	B	B	B	P	B	B	P

Fonte: Autoria própria.

Após esta escrita da definição das característica do nosso QR Code, sua estrutura fica conforme ilustrado na Fig. 5.

Figura 5: QR Code preenchido com o número binário 001011010001001 na região onde se define suas características quanto a modelo, versão, máscara, e tipo de código corretor de erro.



Fonte: Autoria própria.

2.3 Modo de dados e tamanho dos dados

Assim como na criptografia, em que se deve escolher um alfabeto na codificação das mensagens, isto é a escolha dos caracteres permitidos na escrita das mensagens, o mesmo é feito em um QR Code, porém com a escolha de caracteres limitada a cinco opções de modos. Esta escolha de alfabeto é a primeira etapa na escrita dos dados na região onde são escritos os dados a serem codificados (na Fig. 2 é a região de módulos sem numeração, pintados na cor cinza escuro). Em outras palavras, a primeira mensagem a ser escrita é descrever qual modo de alfabeto estamos utilizando.

- Modo numérico, escrita de números utilizando os algarismos de 0 a 9. Neste caso, deve-se escrever o número binário 0001.
- Modo alfanumérico, os caracteres seguem o padrão alfanumérico de codificação e aceita as letras de A a Z (apenas em maiúsculos), algarismos de 0 a 9, além de alguns caracteres especiais [1]. Neste caso, deve-se escrever o número binário 0010.
- Modo byte, onde os caracteres seguem uma codificação padrão em byte. Neste caso, deve-se escrever o número binário 0100.
- Modo kanji, que é o sistema de escrita do idioma japônes. Neste caso, deve-se escrever o número binário 1000.
- Modo *Extended Channel Interpretation* (ECI), não iremos tratar deste caso, ele só é elencado aqui. Porém, se for feita a escolha do modo ECI, deve-se escrever o número binário 1111.

Após a escrita do modo de alfabeto utilizado, descreve-se o tamanho da mensagem em número binário. Como optamos por fazer um QR Code modelo 1, versão 1, com código corretor do tipo H, existe uma limitação na quantidade de caracteres da mensagem para cada tipo de modo:

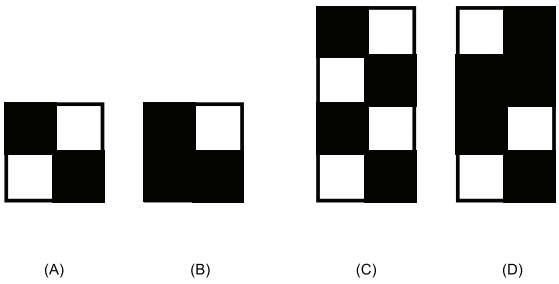
- até 17 caracteres no modo numérico, ou seja, um número na base dez, escrito com até 17 dígitos;
- até 10 caracteres no modo alfanumérico;
- até 7 caracteres no modo byte;
- até 4 caracteres no modo kanji.

No caso, escolhemos o modo de escrita em byte, logo escrevemos o número binário 0100 no espaço de dados, justamente ocupando o bloco 2×2 de módulos posicionado na inscrição Início na Fig. 4(A). Na Fig. 6(A) temos o bloco 2×2 em branco, com a máscara 0 (zero), e na Fig. 6(B) o bloco 2×2 escrito com o número binário 0100 conforme a explicação da seção 2.1.

Na sequência, escrevemos o tamanho da mensagem codificada no QR Code, a título de exemplo, escrevemos a mensagem “Calculo”, que possui 7 (sete) caracteres no modo byte. Logo, escrevemos em binário o número 7 utilizando oito dígitos. A depender do modo de escrita e do modelo e versão do QR Code, a quantidade de dígitos no número binário para informar o tamanho da mensagem muda (veja mais detalhes em [8] seção *Data Encoding, Step 4: Add the Character Count Indicator*). O número 7 em binário é 111, e usando oito dígitos fica 00000111. Portanto, nos próximos oito módulos, logo em seguida de onde escrevemos o modo de caracteres, escrevemos 00000111. Na Fig. 6(C) temos os oito módulos em branco, e na Fig.

6(D) temos os oito módulos com o número binário 00000111 escrito, seguindo a metodologia dada na seção 2.1.

Figura 6: Imagem (A), sequência de quatro módulos com máscara zero sem nada escrito. Imagem (B), sequência de quatro módulos com máscara zero, com o número binário 0100 escrito. Imagem (C), sequência de oito módulos com máscara zero, sem nada escrito. Imagem (D), sequência de oito módulos com máscara zero, com o número binário 00000111 escrito.



Fonte: Autoria própria.

2.4 Escrevendo a mensagem

Como escolhemos o modo byte, escrevemos a mensagem "Calculo" utilizando uma tabela ASCII, onde cada caractere possui uma associação com um número binário de oito dígitos, ou seja um byte. A tabela ASCII completa pode ser vista em [10]. Terminando de escrever a mensagem, deve-se encerrar com uma sequência de quatro zeros, 0000. Logo, a mensagem "Calculo" em byte fica na forma

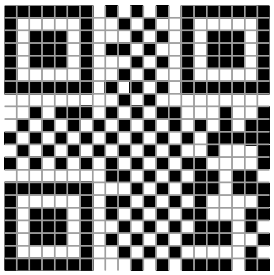
$$\underbrace{01000011}_{\text{C}}\underbrace{01100001}_{\text{a}}\underbrace{01101100}_{\text{l}}\underbrace{01100011}_{\text{c}}\underbrace{01110101}_{\text{u}}\underbrace{01101100}_{\text{l}}\underbrace{01101111}_{\text{o}}0000. \tag{2}$$

No QR Code, logo após o número binário de oito dígitos informando o tamanho da mensagem, escrevemos a mensagem dada em (2).

Caso a mensagem tenha menos de sete caracteres, adicione caracteres de forma a completar os sete caracteres. Por exemplo, a mensagem "Sim!" possui quatro caracteres, então adicionamos três pontos "..." ao final, obtendo "Sim!..." com sete caracteres.

Após esta etapa inicial, de estruturação do QR Code escrevendo definições do QR Code (vide seção 2.2), escolha do modo de mensagem (modo byte), tamanho da mensagem (mensagem com sete caracteres em byte), e a mensagem em si ("Calculo"), o QR Code fica conforme exibido na Fig. 7. Feito isso, passamos à próxima etapa que é a do código corretor de erro, mas antes tratamos do assunto de corpos finitos.

Figura 7: QR Code definido, com a escrita do tipo de modo (modo byte), tamanho da mensagem (sete caracteres), e a escrita da mensagem "Calculo", restando preencher o código corretor de erro.



Fonte: Autoria própria.

3 Corpos Finitos

Nesta seção, explicamos sobre a teoria de corpos finitos, o necessário do que é utilizado no código corretor de erro em um QR Code.

Em linhas gerais, um corpo K é um conjunto fechado com duas operações, usualmente as operações de soma e produto [9]. Os racionais, $\mathbb{Q}$, é um corpo com as operações usuais de soma e multiplicação, o mesmo ocorre com os reais, $\mathbb{R}$.

Um corpo finito é um conjunto F que é corpo, porém com uma quantidade finita de elementos. Todo corpo finito tem quantidade de elementos p^n, onde p é um número primo, e $n \in \mathbb{Z}_+^*$ [11]. Um corpo finito com exatamente p elementos é isomorfo a $\mathbb{Z}_p$, conjunto das classes de equivalência de congruência módulo p [9]. Em particular, $\mathbb{Z}_2 = \{0, 1\}$ é um corpo e valem as operações,

$$0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 1 = 0, \quad 0 \cdot 0 = 0, \quad 0 \cdot 1 = 0, \quad 1 \cdot 1 = 1.$$

Um corpo finito com p^n elementos é denotado por $GF(p^n)$, onde a sigla *GF* significa *Galois Field*, em uma tradução livre "corpo de Galois", em homenagem ao matemático francês Évariste Galois (1811-1832) [12]. Esta mesma terminologia com *GF* também é aplicada aos corpos finitos com p elementos, ou seja $GF(p) \simeq \mathbb{Z}_p$, onde o símbolo $\simeq$ denota que existe um isomorfismo entre os corpos [9].

De um modo geral, vale o seguinte isomorfismo

$$GF(p^n) \simeq \frac{\mathbb{Z}_p[x]}{< f(x) >} \quad (n \in \mathbb{Z}_+^*),$$

onde $\mathbb{Z}_p[x]$ é o anel de poliômios na variável x com coeficientes em $\mathbb{Z}_p$, $f(x)$ é um polinômio irredutível de grau n em $\mathbb{Z}_p[x]$, e $< f(x) >$ representa o ideal gerado por $f(x)$ em $\mathbb{Z}_p[x]$ [11]. Logo, qualquer elemento de $GF(p^n)$ pode ser visto como um polinômio em $\mathbb{Z}_p[x]$ módulo um

polinômio irredutível $f(x)$ de grau n [11]. A depender da escolha deste polinômio irredutível, a menos de isomorfismo, temos o mesmo corpo, ou seja, se $f(x)$ e $g(x)$ são polinômios irredutíveis em $\mathbb{Z}_p[x]$, distintos, de grau n, então

$$GF(p^n) \simeq \frac{\mathbb{Z}_p[x]}{< f(x) >} \simeq \frac{\mathbb{Z}_p[x]}{< g(x) >}.$$

Dado $n \in \mathbb{Z}_+^*$, sempre é possível obter um polinômio irredutível de grau n em $\mathbb{Z}_p[x]$ [11].

Sejam $w(x)$ e $q(x)$ dois elementos em $GF(p^n)$, a soma entre $w(x)$ e $q(x)$ denotada por $w(x) + q(x)$ funciona como soma usual de polinômios, lembrando que seus coeficientes estão em $\mathbb{Z}_p$. Quanto ao produto entre $w(x)$ e $q(x)$, escrito como $w(x) \cdot q(x)$, também funciona de forma usual como um produto de polinômios, atentando, mais uma vez, ao fato de que seus coeficientes estão em $\mathbb{Z}_p$. Caso o grau de um polinômio $h(x)$ em $GF(p^n)$ seja igual ou exceda o grau n do polinômio irredutível $f(x)$, deve-se efetuar a divisão de $h(x)$ por $f(x)$, e usar o resto da divisão para representar $h(x)$ em $GF(p^n)$.

Existem, ao menos, mais duas formas de lidar com os elementos de $GF(p^n)$ [11, 12]. Uma delas é através de um elemento chamado de elemento primitivo $b \in GF(p^n)$, tal que qualquer elemento $d \neq 0$ de $GF(p^n)$ é uma potência de b, isto é, $d = b^k$ para algum k inteiro positivo. A outra forma é através de matrizes. Não iremos tratar destas outras formas aqui.

No código corretor de erro de Reed-Solomon, para o caso de QR Codes, só vamos necessitar trabalhar com o corpo finito composto por 256 elementos, $GF(256) = GF(2^8)$. Para este caso específico, utilizamos o polinômio irredutível $f(x) = x^8 + x^4 + x^3 + x^2 + 1$ [12, 13], definindo assim o corpo

$$GF(256) \simeq \frac{\mathbb{Z}_2[x]}{< x^8 + x^4 + x^3 + x^2 + 1 >}.$$

Além disso, adotamos a seguinte representação dos elementos em $GF(256)$, dado

$$N \in \{0, 1, 2, \ldots, 254, 255\},$$

com $r_7r_6r_5r_4r_3r_2r_1r_0$ a representação em binário de N, então

$$r_7x^7 + r_6x^6 + r_5x^5 + r_4x^4 + r_3x^3 + r_2x^2 + r_1x + r_0 \in GF(256).$$

Isto é, podemos pensar os elementos de $GF(256)$ como os números inteiros de 0 a 255, e quando for necessário efetuar uma operação de soma ou produto entre eles, utilizamos a representação de polinômio por meio de sua representação em binário.

Exemplo 2. *Escrevemos as representações em binário, com oito dígitos, dos números* 64 *e* 198*, respectivamente,*

$$01000000 \qquad e \qquad 11000110,$$

então 64 *é representado pelo polinômio*

$$0x^7 + 1x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 0 = x^6, \tag{3}$$

e 198 *é representado pelo polinômio*

$$1x^7 + 1x^6 + 0x^5 + 0x^4 + 0x^3 + 1x^2 + 1x^1 + 0 = x^7 + x^6 + x^2 + x. \tag{4}$$

A soma e o produto entre os elementos 64 *e* 198 *em* $GF(256)$ *é feita utilizando suas representações polinomiais dadas, respectivamente, em* (3) *e* (4)*, levando em conta que seus coeficientes estão em* $\mathbb{Z}_2$*. Assim, a soma* $64 + 198$ *em* $GF(256)$ *fica igual a*

$$x^7 + x^2 + x,$$

com a equivalência com o número binário 10000110*, que na base dez é o número* 134*. Portanto,* $64 + 198 = 134$ *em* $GF(256)$*.*

Quanto ao produto entre 64 *e* 198 *em* $GF(256)$*, obtemos*

$$x^6(x^7 + x^6 + x^2 + x) = x^{13} + x^{12} + x^8 + x^7.$$

Sendo o grau do polinômio resultante maior ou igual a oito, logo devemos efetuar a divisão em $\mathbb{Z}_2[x]$ *pelo polinômio irredutível* $x^8 + x^4 + x^3 + x^2 + 1$*, e usar o resto da divisão para sua representação em* $GF(256)$*. Obtemos como resto da divisão de* $x^{13} + x^{12} + x^8 + x^7$ *por* $x^8 + x^4 + x^3 + x^2 + 1$*, o polinômio* $x^7 + x^6 + x^4 + x^2 + x + 1$*, que tem equivalência com o número binário* 11010111*, sendo o número* 215 *em base dez. Portanto,* $64 \cdot 198 = 215$ *em* $GF(256)$*.*

Observamos que, se em vez do polinômio irredutível $f(x) = x^8 + x^4 + x^3 + x^2 + 1$*, for utilizado outro polinômio irredutível em* $\mathbb{Z}_2[x]$*, por exemplo* $g(x) = x^8 + x^4 + x^3 + x + 1$ *[12], então apesar de*

$$GF(256) \simeq \frac{\mathbb{Z}_2[x]}{< x^8 + x^4 + x^3 + x^2 + 1 >} \simeq \frac{\mathbb{Z}_2[x]}{< x^8 + x^4 + x^3 + x + 1 >},$$

as operações de produto entre elementos de $GF(256) \simeq \frac{\mathbb{Z}_2[x]}{<x^8+x^4+x^3+x+1>}$ *utilizando a representação em base dez tem uma configuração diferente. Por exemplo,* $64 \cdot 198 = 125$ *em* $GF(256) \simeq \frac{\mathbb{Z}_2[x]}{<x^8+x^4+x^3+x+1>}$*. Por isso, é importante fixar o polinômio irredutível utilizado em todas as operações.*

Observação 2. *A operação de soma em* $GF(2^n)$ *não depende da escolha do polinômio irredutível, além disso o inverso aditivo de qualquer elemento* $w(x)$ *em* $GF(2^n)$ *é ele mesmo, ou seja* $w(x) + w(x) = 0$*.*

Uma maneira de gerar todos os elementos de $GF(2^n)$ e observar a diferença na escolha do

polinômio irredutível de grau n, é pegar os elementos $0, 1, x$, e para gerar os demais elementos, utilizar potências de x. Quando o grau desta potência de x for maior ou igual a n, efetuar a divisão pelo polinômio irredutível escolhido.

Exemplo 3. *Para $GF(2^3) = GF(8)$, os polinômios $x^3 + x^2 + 1$ e $x^3 + x + 1$ são irredutíveis de grau 3 em $\mathbb{Z}_2[x]$, então*

$$GF(8) \simeq \frac{\mathbb{Z}_2[x]}{< x^3 + x^2 + 1 >} \simeq \frac{\mathbb{Z}_2[x]}{< x^3 + x + 1 >}.$$

Na construção dos elementos de $GF(8)$ com cada polinômio irredutível, a Tabela 2 mostra esta diferença, onde na coluna do meio temos os elementos $0, 1, x$ e as potências de x, na coluna mais à esquerda os elementos obtidos a partir da divisão pelo polinômio irredutível $x^3 + x^2 + 1$ e na coluna mais à direita os elementos obtidos a partir da divisão pelo polinômio irredutível $x^3 + x + 1$.

Tabela 2: Elementos de $GF(8)$ obtidos por meio do polinômio irredutível x^3+x^2+1 na coluna mais à esquerda, e elementos de $GF(8)$ obtidos por meio do polinômio irredutível $x^3 + x + 1$ na coluna mais à direita.

$\frac{\mathbb{Z}_2[x]}{< x^3 + x^2 + 1 >}$		$\frac{\mathbb{Z}_2[x]}{< x^3 + x + 1 >}$
0	0	0
1	1	1
x	x	x
x^2	x^2	x^2
$x^2 + 1$	x^3	$x + 1$
$x^2 + x + 1$	x^4	$x^2 + x$
$x + 1$	x^5	$x^2 + x + 1$
$x^2 + x$	x^6	$x^2 + 1$

Fonte: Autoria própria.

Olhando a Tabela 2, por um lado temos $x \cdot x^2 = x^2 + 1$ em $GF(8) \simeq \frac{\mathbb{Z}_2[x]}{<x^3+x^2+1>}$, por outro lado $x \cdot x^2 = x + 1$ em $GF(8) \simeq \frac{\mathbb{Z}_2[x]}{<x^3+x+1>}$.

Observação 3. *Trabalhando com QR Code, para efetuar as operações em $GF(256)$ é obrigatório o uso do polinômio irredutível em $\mathbb{Z}_2[x]$ dado por $f(x) = x^8 + x^4 + x^3 + x^2 + 1$, ou seja, consideramos*

$$GF(256) \simeq \frac{\mathbb{Z}_2[x]}{x^8 + x^4 + x^3 + x^2 + 1}.$$

O motivo de fixar esta escolha de polinômio irredutível se deve ao fato desse polinômio ser o padrão na implementação computacional de $GF(256)$ [14, 8]. Portanto, daqui por diante,

todas as operações feitas em $GF(256)$ *utilizam o polinômio irredutível em* $\mathbb{Z}_2[x]$ *definido por* $f(x) = x^8 + x^4 + x^3 + x^2 + 1$.

3.1 Trabalhando com $GF(256)$ computacionalmente

Nesta subseção, explicamos como se trabalha com $GF(256)$ de maneira computacional (veja [8], seção *Error Correction Coding*, ou [14]). Os elementos de $GF(256)$ são vistos como números de 0 a 255. Para somar dois elementos N_1 e N_2 de $GF(256)$, escrevemos cada um em forma binária e somamos utilizando a operação de XOR ("ou exclusivo"), denotada por $\oplus$, satisfazendo para cada dígito em binário

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 1 = 0.$$

Por exemplo, somando 64 com 198, escrevemos a representação binária de cada um, respectivamente, 01000000 e 11000110. Somamos usando a operação de XOR,

$$01000000 \oplus 11000110 = 10000110,$$

e 10000110 é o número 134 em base dez. Logo, $64 + 198 = 134$ em $GF(256)$. Equivalente a operação de soma explicado no Exemplo 2.

Para fazer o produto entre dois números de $GF(256)$, escrevemos cada elemento de 1 a 255 de $GF(256)$ como uma potência de 2, criando uma tabela

$$2^0 = 1, \quad 2^1 = 2, \quad 2^2 = 4, \quad 2^3 = 8, \quad 2^4 = 16, \quad 2^5 = 32, \quad 2^6 = 64, \quad 2^7 = 128,$$

como $2^8 = 256$ e $256 \notin GF(256)$, neste caso escrevemos 256 em binário, 100000000, e somamos usando a operação de XOR com o número 285 em binário, 100011101,

$$100000000 \oplus 100011101 = 000011101,$$

e 000011101 em base dez é 29, logo $2^8 = 29$ em $GF(256)$. Continuando, $2^9 = 2 \cdot 2^8 = 2 \cdot 29 = 58$ em $GF(256)$, $2^{10} = 2 \cdot 2^9 = 2 \cdot 58 = 116$ em $GF(256)$, $2^{11} = 2 \cdot 2^{10} = 2 \cdot 116 = 232$ em $GF(256)$, $2^{12} = 2 \cdot 2^{11} = 2 \cdot 232 = 464$ e $464 \notin GF(256)$, repetimos o procedimento feito em 2^8, transformamos 464 em binário, e somamos usando a operação de XOR com 285, também em binário,

$$111010000 \oplus 100011101 = 011001101,$$

e 011001101 é 205 em base dez, logo $2^{12} = 205$ em $GF(256)$. Continuamos este procedimento, para as demais potencias, $2^{13}, 2^{14}, \ldots, 2^{254}$, até chegar a $2^{255} = 1$ em $GF(256)$, cobrindo todos os elementos não nulos de $GF(256)$. Com esta tabela de potencias de 2, representando cada elemento não nulo de $GF(256)$, para fazer o produto, por exemplo, entre 116 e 205 em $GF(256)$,

procuramos na tabela a representação em potencia de 2 de cada número em $GF(256)$, e fazemos

$$116 \cdot 205 = 2^{10} \cdot 2^{12} = 2^{10+12} = 2^{22} = 234.$$

Caso obtenha uma potencia 2^k com $k > 255$, fazemos a divisão euclidiana de k por 255, obtendo $k = 255 \cdot q + r$, com $0 \leq r < 255$, assim

$$2^k = 2^{255 \cdot q + r} = \left(2^{255}\right)^q \cdot 2^r = 1^q \cdot 2^r = 2^r.$$

Em [8], seção *QR Code Log Antilog Table for Galois Field 256*, existe uma tabela com todas as potências de 2 em $GF(256)$.

Observação 4. *No caso em que os números* $256, 464 \notin GF(256)$ *e foi necessário transformá-los em binário para fazer a soma usando XOR com o número* 285 *em binário, isto equivale a executar o que foi exposto no início desta seção 3. Seguindo a explicação da primeira parte desta seção 3, para o número* 256 *que em binário é* 100000000 *equivalente ao polinômio* x^8*, e* 285 *em binário é* 100011101 *equivalente ao polinômio* $x^8 + x^4 + x^3 + x^2 + 1$*. Fazendo a divisão de* x^8 *por* $x^8 + x^4 + x^3 + x^2 + 1$*, obtemos o resto* $x^4 + x^3 + x^2 + 1$*, associando a binário com* 11101 *que equivale ao número* 29 *em base dez. Sendo mais direto, para o número* 464*, associamos com o polinômio* $x^8 + x^7 + x^6 + x^4$ *e dividimos pelo polinômio* $x^8 + x^4 + x^3 + x^2 + 1$*, obtendo o resto* $x^7 + x^6 + x^3 + x^2 + 1$*, que equivale ao número em base dez* 205*. Em resumo, quando o número* N *inteiro positivo não está em* $GF(256)$*, transformar em binário e somar usando a operação de XOR com* 285 *em binário, equivale a usar a representação polinomial de* N *em* $GF(256) \simeq \frac{\mathbb{Z}_2}{<x^8+x^4+x^3+x^2+1>}$ *conforme explicado na primeira parte da seção 3. Isso justifica o que foi dito na Observação 3.*

4 Código corretor de erro

Mencionamos anteriormente que o código corretor de erro utilizado em um QR Code é o de Reed-Solomon, que funciona da seguinte forma [13]: dada uma quantidade de dados, em um QR Code temos os dados do método de escrita da mensagem, byte, a quantidade de caracteres utilizada, sete caracteres, e a mensagem armazenada "Calculo", então adicionamos mais alguns dados extras, justamente o código corretor de erros, de forma que caso uma parte da mensagem seja corrompida, ainda seja possível recuperar parte dela.

Os quatro níveis de correção do código corretor de Reed-Solomon garante a quantidade máxima de dados a serem recuperados, nível L recupera até 7% de dados, nível M até 15%, nível Q até 25% e nível H até 30%. Não entramos nos pormenores sobre o funcionamento do código corretor de erro de Reed-Solomon, nos restringimos a explicar como funciona a escrita dos dados deste código em um QR Code. No artigo [6] a parte teórica sobre código corretores de erro de Reed-Solomon é tratada com mais detalhes.

Em um QR Code, a partir dos dados escritos, criamos um poliômio $M(X)$ em

$GF(256)[X]$, utilizamos um polinômio $G(X)$ em $GF(256)[X]$ chamado de polinômio gerador, e efetuamos a divisão de $M(X)$ por $G(X)$ em $GF(256)[X]$, os coeficientes do polinômio obtido como resto nesta divisão, $R(X)$, são os dados do código corretor de erro a ser escrito no restante do QR Code.

No QR Code que estamos escrevendo, até o momento temos escrito na sua parte de dados o tipo de codificação da mensgem, byte, dado pelo número binário 0100, a quantidade de caracteres da mensagem, sete caracteres, em binário 00000111, e a mensagem "Calculo" finalizada com a sequência em binário 0000, dada pela equação (2). Juntamos todos estes dados em binário,

$$01000000011101000011011000010110110001100011011101010110110001101111 0000. \quad (5)$$

Dividimos o número em binário em (5) em blocos de oito dígitos,

$$\begin{array}{lll} 01000000 & 01110100 & 00110110 \\ 00010110 & 11000110 & 00110111 \\ 01010110 & 11000110 & 11110000. \end{array} \quad (6)$$

Cada bloco de oito dígitos em (6) é um elemento em $GF(256)$, são eles, respectivamente,

$$64, \quad 116, \quad 54, \quad 22, \quad 198, \quad 55, \quad 86, \quad 198, \quad 240.$$

Estes números são os coeficientes do polinômio $M(X) \in GF(256)[X]$ definido por

$$M(X) = 64X^8 + 116X^7 + 54X^6 + 22X^5 + 198X^4 + 55X^3 + 86X^2 + 198X + 240. \quad (7)$$

Para o nosso caso, de um QR Code modelo 1, versão 1, com uma mensagem de sete caracteres escrita em byte, e usando o nível H no código corretor de erro, o polinômio gerador $G(X)$ é sempre o mesmo, e é definido por

$$G(X) = \prod_{j=0}^{16} X + 2^j = (X + 2^0)(X + 2^1)(X + 2^2)\dots(X + 2^{14})(X + 2^{15})(X + 2^{16}). \quad (8)$$

Mais detalhes sobre como obter o polinômio $G(X)$ pode ser visto em [8], na seção *Error Correction Coding, Step 7: Understanding The Generator Polynomial*.

Expandimos a expressão de $G(X)$ em (8) no corpo $GF(256)[X]$,

$$\begin{aligned} G(X) \;=\; & X^{17} + 119X^{16} + 66X^{15} + 83X^{14} + 120X^{13} + 119X^{12} \\ & +22X^{11} + 197X^{10} + 83X^9 + 249X^8 + 41X^7 + 143X^6 \\ & +134X^5 + 85X^4 + 53X^3 + 125X^2 + 99X + 79. \end{aligned} \quad (9)$$

Portanto, sempre que for criar um QR Code modelo 1, versão 1, com uma mensagem de sete caracteres escrita em byte, e usando o nível H no código corretor de erro, o polinômio gerador $G(X)$ a ser usado é este dado em (9).

Para fazer o processo de divisão de $M(X)$ por $G(X)$, antes devemos multiplicar $M(X)$ em (7) pelo monômio X^{17}, onde 17 é o grau de $G(X)$ em (9), obtendo, então, a versão final de $M(X)$,

$$M(X) = 64X^{25} + 116X^{24} + 54X^{23} + 22X^{22} + 198X^{21} + 55X^{20} + 86X^{19} + 198X^{18} + 240X^{17}. \tag{10}$$

Efetuamos a divisão em $GF(256)[X]$ de $M(X)$ em (10) por $G(X)$ em (9), obtendo o quociente $Q(X)$ e resto $R(X)$ tais que

$$M(X) = Q(X) \cdot G(X) + R(X),$$

onde

$$\begin{aligned} Q(X) &= 64X^8 + 248X^7 + 8X^6 + 241X^5 + 222X^4 + 88X^3 \\ &\quad +16X^2 + 101X + 154, \end{aligned}$$

$$\begin{aligned} R(X) &= 70X^{16} + 95X^{15} + 31X^{14} + 143X^{13} + 87X^{12} + 199X^{11} \\ &\quad +214X^{10} + 18X^9 + 99X^8 + 133X^7 + 202X^6 + 234X^5 \\ &\quad +58X^4 + 28X^3 + 181X^2 + X + 12. \end{aligned} \tag{11}$$

Os coeficientes

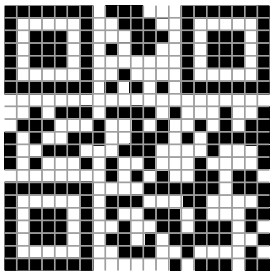
$$\begin{array}{cccccccccc} 70, & 95, & 31, & 143, & 87, & 199, & 214, & 18, & 99, \\ 133, & 202, & 234, & 58, & 28, & 181, & 1, & 12, & \end{array} \tag{12}$$

nesta ordem, do polinômio $R(X)$ em (11), formam o código corretor de erro a ser escrito no restante do QR Code exibido na Fig. 7. A escrita deve ser em binário, portanto, os coeficientes em (12) são dados, respectivamente, pelos números binários de oito dígitos,

$$\begin{array}{llll} B_{16} = 01000110, & B_{15} = 01011111, & B_{14} = 00011111, & B_{13} = 10001111, \\ B_{12} = 01011111, & B_{11} = 11000111, & B_{10} = 11010110, & B_9 = 00010010, \\ B_8 = 01100011, & B_7 = 10000101, & B_6 = 11001010, & B_5 = 11101010, \\ B_4 = 00111010, & B_3 = 00011100, & B_2 = 10110101, & B_1 = 00000001, \\ B_0 = 00001100. & & & \end{array}$$

Escrevemos estes números em binário acima na sequência $B_{16}, B_{15}, B_{14}, \ldots, B_2, B_1, B_0$, no restante do QR Code da Fig. 7, obtendo, assim, o QR Code dado pela Fig. 8.

Figura 8: QR Code final, definido com a escrita do tipo de modo (modo byte), tamanho da mensagem (sete caracteres), mensagem "Calculo" escrita, e com o código corretor de erro preenchido.



Fonte: Autoria própria.

A versão final do QR Code modelo 1, versão 1, com a mensagem "Calculo", onde é possível fazer a leitura com um leitor de QR Code, é exibido na imagem da Fig. 1.

5 Conclusão

Neste artigo, tratamos de todos os requisitos necessários na elaboração de um QR Code modelo 1, versão 1, onde construímos do início ao fim o QR Code funcional exposto na Fig. 1, podendo ser lido por qualquer aplicativo leitor de QR Code. Apesar desta restrição de modelo e versão, os leitores interessados tem toda base para compreensão de outros modelos e versões de QR Codes, visto que a diferença entre eles são detalhes estritamente técnicos. Devido a utilização das teorias de corpos finitos e de códigos corretores de erros, observamos que QR Codes podem servir como motivação no aprofundamento do estudo destas teorias.

Conflitos de Interesse

Os autores declaram que não têm conflitos de interesse.

Financiamento

Este trabalho foi realizado sem apoio financeiro.

Aprovação do Comitê de Ética

Não se aplica.

Licença

As obras submetidas ao jornal BEJOM estão sujeitas à licença CC BY 4.0. Sob esta licença, os autores concedem aos leitores o direito de compartilhar, adaptar e utilizar as obras, inclusive para fins comerciais, desde que o crédito apropriado seja dado aos autores. Quaisquer modificações devem ser indicadas. Não há restrições adicionais além das estabelecidas pela licença.

Referências

[1] Silva, A. R. D. A matemática do código de barras e QR Code. Master's thesis. 2021. URL: http://www.repositorio-bc.unirio.br:8080/xmlui/bitstream/handle/unirio/13368/TCC%20-%20QR%20CODE%20-%20Vers%c3%a3o%20FInal.pdf?sequence=1&isAllowed=y (acesso em 16/08/2024).

[2] Silva, A. R. D. e Fantin, S. A Matemática do QR Code. Em: *Revista Ensin@ UFMS* 2.Esp. (15 de dez. de 2021), pp. 374–399. URL: https://periodicos.ufms.br/index.php/anacptl/article/view/13902 (acesso em 20/04/2024).

[3] Sousa, D. P. de. Dos hieroglifos ao QR Code: códigos como ferramenta na sala de aula. Master's thesis. Universidade Estadual do Sudoeste da Bahia, 2016. URL: https://www2.uesb.br/ppg/profmat/wp-content/uploads/2018/11/Dissertacao_DEIVISON_PORTO_DE_SOUSA.pdf (acesso em 16/08/2024).

[4] Souza, R. A. et al. Explorando QR Codes como Recurso Didático na Educação Matemática. Em: *Revista Professor de Matemática Online* 12.1 (2024), pp. 72–90. DOI: 10.21711/2319023x2024/pmo125.

[5] *History of QR Code*. 2024. URL: https://www.qrcode.com/en/history/ (acesso em 20/04/2024).

[6] Cornelissen, M. e Moura, R. Códigos de Reed Solomon. Em: *Revista de Matemática da UFOP* 2 (2021), pp. 1–18. URL: https://periodicos.ufop.br/rmat/article/view/5087 (acesso em 16/08/2024).

[7] *Types of QR Code*. 2024. URL: https://www.qrcode.com/en/codes/ (acesso em 20/04/2024).

[8] *QR Code Tutorial*. 2024. URL: https://www.thonky.com/qr-code-tutorial/ (acesso em 20/04/2024).

[9] Gonçalves, A. Introdução à Álgebra. 5ª ed. Rio de Janeiro: IMPA, 2006.

[10] *Tabela ASCII*. 2024. URL: https://www.ime.usp.br/~kellyrb/mac2166_2015/tabela_ascii.html (acesso em 22/04/2024).

[11] Lidl, R. e Niederreiter, H. Introduction to Finite Fields and Their Applications. Cambridge University Press, 1994.

[12] Mullen, G. L. e Panario, D. Handbook of Finite Fields. Boca Raton: CRC Press, 2013.

[13] Plank, J. S. A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems. Em: *Software: Practice & Experience* 29.9 (1997), pp. 995–1012. URL: https://web.eecs.utk.edu/~jplank/plank/papers/SPE-9-97.html (acesso em 26/04/2024).

[14] Artizzu, M. *Let's develop a QR Code Generator, part III: error correction*. 2021. URL: https://dev.to/maxart2501/let-s-develop-a-qr-code-generator-part-iii-error-correction-1kbm (acesso em 29/04/2024).

Corresponding Author:
José Laudelino de Menezes Neto, *laudelino@dcx.ufpb.br*

Submitted: August 16, 2024
Accepted: February 24, 2025
Published: May 28, 2025
https://seer.ufu.br/index.php/BEJOM/index