PROFESSOR DE MATEMÁTICA ONLINE
Revista eletrônica da Sociedade Brasileira de Matemática

PMO v.12, n.1, 2024
ISSN: 2319-023X
https://doi.org/10.21711/2319023x2024/pmo125

Explorando QR Codes como Recurso Didático na Educação Matemática

Rodrigo Araujo de Souza Thainá Lopes Ferreira Weslaine Herzog Santos
Alcebiades Dal Col

Resumo

O objetivo principal deste artigo é promover a compreensão do QR Code e indicar sua relação com a Matemática. Nós iniciamos com uma explicação concisa sobre o funcionamento do QR Code, com ênfase nos padrões presentes no QR Code e nos conceitos de *bytes* e máscaras. Nosso propósito é mostrar aos estudantes do ensino médio as conexões entre a matemática escolar e a matemática aplicada. Além disso, este trabalho compartilha uma proposta didática que estimula a compreensão do QR Code. Nós destacamos a importância de despertar a curiosidade dos estudantes, estimulando seu interesse pela Matemática, ao demonstrar como essa disciplina está intrinsecamente ligada ao QR Code.

Palavras-chave: QR Code; codificação; máscara; proposta didática; binário.

Abstract

The main objective of this article is to promote understanding of the QR Code and indicate its relationship with Mathematics. We start with a concise explanation of how the QR code works, with a focus on the patterns present in the QR code and on the concepts of bytes and masks. Our purpose is to demonstrate to students the connections between school mathematics and applied mathematics. Moreover, this work shares a didactic proposal that encourages understanding of the QR code. We emphasize the importance of sparking students' curiosity, stimulating their interest in Mathematics, by showing how this discipline is intrinsically linked to the QR code.

Keywords: QR code; codification; mask; didactic proposal; binaries.

1. Introdução

Na década de 1960, surgiu no Japão a necessidade de um código identificador de mercadorias [1]. Com o rápido crescimento econômico do país e a ausência de um código identificador, os comerciantes eram obrigados a digitar todos os preços dos produtos manualmente, acarretando em alguns problemas de saúde. A fim de solucionar a demanda, foi criado o que conhecemos hoje como código de barras. Porém, com o decorrer dos anos, algumas barreiras vieram à tona, e a principal foi a limitação de caracteres, já que a quantidade máxima era de 20 armazenados.

Após receber muitos pedidos, a *Denso Wave Incorporated* (divisão da *Denso Corporation* encarregada pelo desenvolvimento do código de barras) tomou a decisão de desenvolver um novo código.

SBM SOCIEDADE BRASILEIRA DE MATEMÁTICA

Dado que o código de barras permitia apenas a leitura em uma dimensão, a ideia inicial era desenvolver um código bidimensional. Depois de um ano e meio de trabalho, em 1994, a *Denso Wave Incorporated* anunciou o lançamento do QR Code. QR Code é uma abreviação de *Quick Response Code*, que significa código de resposta rápida em inglês. No cenário atual, devido à disponibilização gratuita das diretrizes de geração do código pela *Denso Wave* e à disponibilidade de *scanners* em *smartphones*, esse código passou a ser utilizado em uma ampla gama de tarefas. Essas aplicações incluem desde o gerenciamento de tarefas e estoque até a emissão de boletos e o armazenamento de informações pessoais.

Com este trabalho, iremos abordar de forma sucinta o funcionamento do QR Code, abrangendo aspectos como o posicionamento dos seus padrões, a codificação de mensagens e o mascaramento de dados. Além disso, ao final, apresentaremos uma proposta de sequência didática para explorar o QR Code em turmas do ensino médio trabalhando as seguintes habilidades presentes na Base Nacional Comum Curricular (BNCC) [2]:

- (EM13MAT105) Utilizar as noções de transformações isométricas (translação, reflexão, rotação e composições destas) e transformações homotéticas para construir figuras e analisar elementos da natureza e diferentes produções humanas (fractais, construções civis, obras de arte, entre outras).
- (EM13MAT405) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

2. Tipos de QR Code

Com a crescente adoção do QR Code, surgiu a necessidade de desenvolver novos tipos de códigos para atender a diferentes demandas. Dentre esses tipos, destacam-se o *micro QR Code*, projetado para códigos de tamanho reduzido, com capacidade máxima de armazenamento de 35 caracteres; o *rMQR Code*, projetado para ser impresso em espaços estreitos, possuindo um formato retangular e capacidade de armazenamento maior que o micro QR Code; o *SQRC*, destinado a dados privados, com aparência semelhante ao QR Code convencional, porém ele só pode ser lido por leitores específicos; o *FRAME QR*, que incorpora uma imagem ao código, sendo exemplos populares o QR Code gerado pelo aplicativo WhatsApp ou Picpay; e, por fim, o *QR Code clássico*, que será o foco dos nossos estudos nas seções seguintes.

O QR Code clássico possui 40 versões, variando em tamanho e complexidade. De uma versão para a versão seguinte tem-se um aumento de 4 *pixels* em cada dimensão do quadrado. A versão 1 tem um tamanho de 21×21, totalizando 441 *pixels*, enquanto a versão 2 possui o tamanho de 25×25, resultando em 625 *pixels*. Seguindo com a progressão vamos até a versão 40 que possui um tamanho de 177×177, totalizando 31.329 *pixels*. A Figura 1 apresenta exemplos de QR Code nas versões 1, 2 e 40. Neste trabalho, a principal versão usada é a de número 3 que tem tamanho 29×29, com um total de 841 pixels. Os pixels são pequenos quadrados que estamos utilizando como unidade de medida de comprimento no QR Code.

Atualmente, nem todo *smartphone* possui o próprio leitor de QR Code, mas, é possível realizar a instalação de aplicativos leitores, como por exemplo: "Leitor de código QR" disponível em [4] e "Scanner de QR Código de Barras" disponível em [5].

3. Padrões no QR Code e suas funções

Figura 1: Exemplos de QR Code nas versões 1, 2 e 40. Fonte: Wikipédia [3].

Uma das principais dificuldades enfrentadas pela empresa durante o desenvolvimento do código foi alcançar uma leitura rápida e eficiente. Como solução, foram estabelecidos padrões que deveriam ser posicionados em locais determinados, com o objetivo de desempenhar funções específicas. Nas subseções a seguir, iremos detalhar a função e o posicionamento dos padrões destacados na Figura 2.

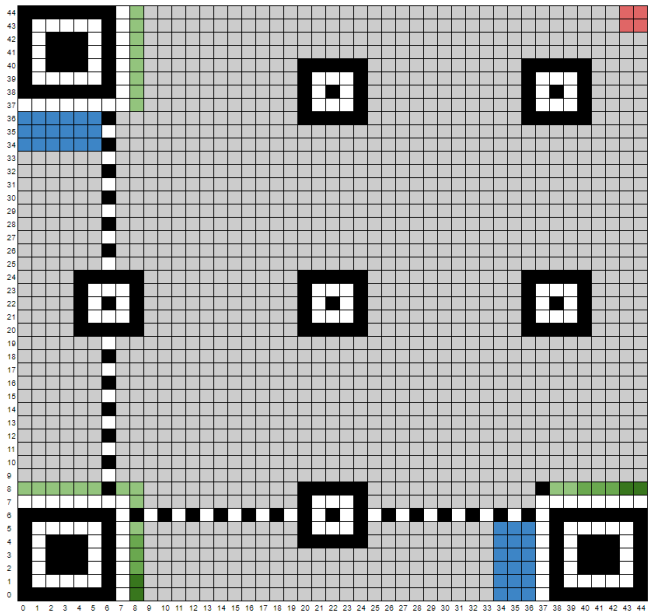


Figura 2: Posicionamento dos padrões. Fonte: Produção dos próprios autores.

3.1. Padrões de localização

Os padrões de localização estão destacados na Figura 3 e são blocos quadrados de tamanho 7×7 *pixels* posicionados no canto superior esquerdo, canto inferior direito e canto inferior esquerdo, totalizando três blocos. Cada bloco é composto por três quadrados de mesmo centro: o quadrado externo preto de tamanho 7 *pixels* por 7 *pixels*, um quadrado branco de 5 *pixels* por 5 *pixels* e um

quadrado preto sólido de 3 *pixels* por 3 *pixels*.

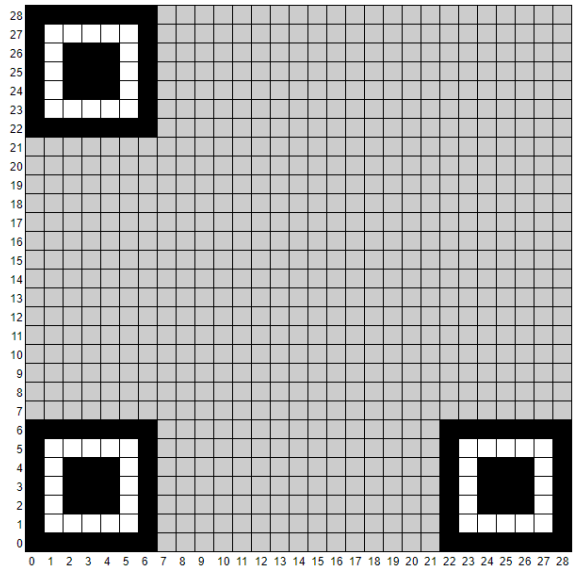


Figura 3: Posicionamento dos padrões de localização. Fonte: Produção dos próprios autores.

A função dos padrões de localização é orientar o *scanner* quanto à direção em que a leitura deve ser realizada. O leitor de QR Code é capaz de identificar o padrão de localização por meio da detecção da proporção de $1:1:3:1:1$, conforme ilustrado na Figura 4, por isso, é possível ler QR Codes rotacionados.

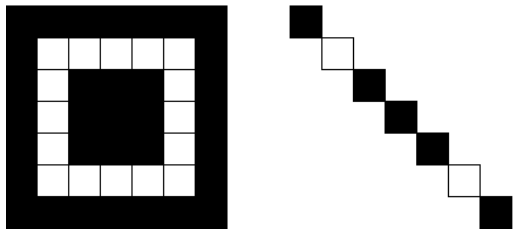


Figura 4: Proporção $1:1:3:1:1$. Fonte: Produção dos próprios autores.

3.2. Separadores

Os separadores são conjuntos de *pixels* brancos, com um *pixel* de largura, que são colocados ao redor dos padrões de localização para separá-los do restante do QR Code. Esse padrão está destacado em branco na Figura 5.

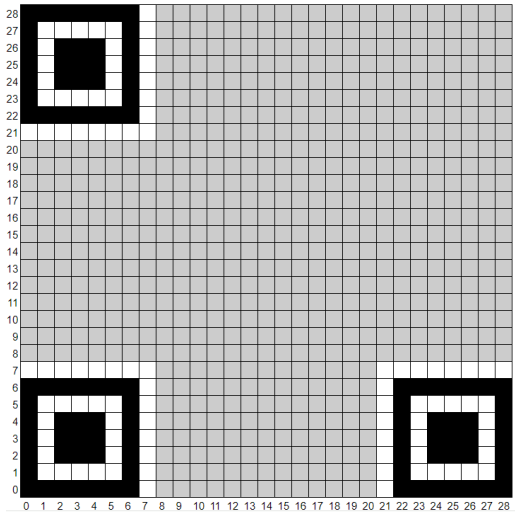


Figura 5: Posicionamento dos separadores. Fonte: Produção dos próprios autores.

3.3. Padrões de alinhamento

Os padrões de alinhamento consistem em blocos de 5 *pixels* por 5 *pixels* e eles devem estar presentes em códigos de versão maiores ou iguais a 2. O bloco é composto por um único *pixel* preto inscrito em um quadrado branco de 3 *pixels* por 3 *pixels* que por sua vez está contido em um quadrado externo preto de tamanho 5 *pixels* por 5 *pixels*, todos com o mesmo centro. A função dos padrões de alinhamento é semelhante à dos padrões de localização.

O posicionamento dos padrões de alinhamento segue uma configuração predefinida, determinada com base na Tabela 1. A tabela determina onde o centro do padrão deve ser disposto, os números representam coordenadas de linha e coluna.

Versão	Centro	Versão	Centro	Versão	Centro
2	6 18	**15**	6 26 48 70	**28**	6 26 50 74 98 122
3	6 22	**16**	6 26 50 74	**29**	6 30 54 78 102 126
4	6 26	**17**	6 30 54 78	**30**	6 26 52 78 104 130
5	6 30	**18**	6 30 56 82	**31**	6 30 56 82 108 134
6	6 34	**19**	6 30 58 86	**32**	6 34 60 86 112 138
7	6 22 38	**20**	6 34 62 90	**33**	6 30 58 83 114 142
8	6 24 42	**21**	6 28 50 72 94	**34**	6 34 62 90 118 146
9	6 26 46	**22**	6 26 50 74 98	**35**	6 30 54 78 102 126 150
10	6 28 50	**23**	6 30 54 78 102	**36**	6 24 50 76 102 128 154
11	6 30 54	**24**	6 28 54 80 106	**37**	6 28 54 80 106 132 158
12	6 32 58	**25**	6 32 58 84 110	**38**	6 32 58 84 110 136 162
13	6 34 62	**26**	6 30 58 86 114	**39**	6 26 54 82 110 138 166
14	6 26 46 66	**27**	6 34 62 90 118	**40**	6 30 58 86 114 142 170

Tabela 1: Posicionamento dos padrões de alinhamento. Fonte: Produção dos próprios autores.

Por exemplo, os números vinculados à versão 3 são 6 e 22, então os centros dos padrões devem

ser colocados em (6, 6), (6, 22), (22, 6) e (22, 22), lembrando que a contagem começa da linha 0 e coluna 0 (canto inferior esquerdo). Porém, os padrões de alinhamento não podem sobrepor os padrões de localização ou os separadores. A Figura 6 exibe a localização desses padrões na versão 3 do QR Code. Os três padrões de alinhamento destacados em vermelho sobrepõe os localizadores, por isso, apenas o padrão de centro (22, 22) é mantido.

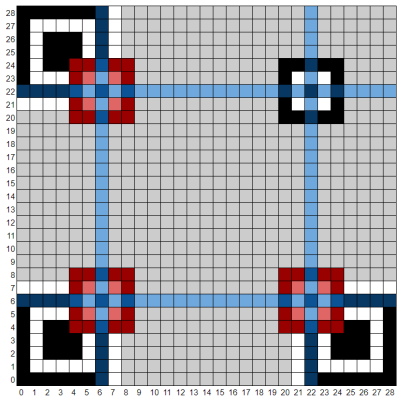


Figura 6: Posicionamento dos padrões de alinhamento na versão 3 do QR Code. Fonte: Produção dos próprios autores.

3.4. Padrões de temporização

Os padrões de tempo são duas linhas de um *pixel* de largura que conectam os padrões de localização. Essas linhas intercalam entre as cores pretas e brancas, começando e terminando com um *pixel* da cor preta. Seu comprimento depende da versão usada, mas o padrão horizontal sempre é colocado na 6ª linha do QR Code, enquanto o vertical sempre é colocado na 6ª coluna do QR Code, como mostrado na Figura 7. A função desse padrão é auxiliar na leitura das linhas e colunas, bem como confirmar a versão do código.

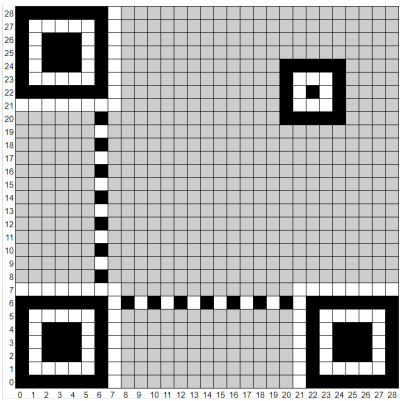


Figura 7: Posicionamento dos padrões de temporização. Fonte: Produção dos próprios autores.

3.5. Correção de erro

Os códigos corretores de erros detectam e corrigem erros. Os *pixels* destacados na Figura 8 são destinados ao armazenamento do nível de correção de erro que será usado na codificação da mensagem. O método de correção de erros usado no QR Code é o chamado código de Reed-Solomon [6].

A Figura 9 mostra os níveis de correção de erro listados em ordem crescente. É importante ressaltar que quanto maior o nível de correção de erro, maiores serão as chances dos leitores de QR Code decodificarem com sucesso a mensagem contida em um QR Code danificado.

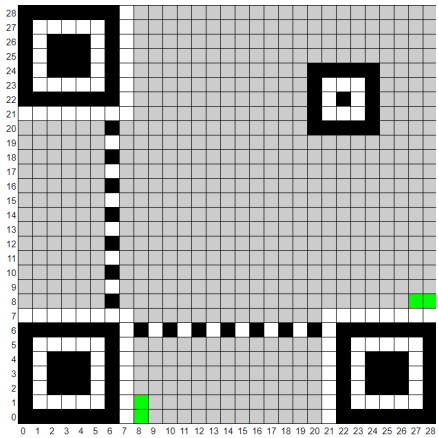


Figura 8: Posicionamento do padrão de correção de erro. Fonte: Produção dos próprios autores.

NÍVEL DE CORREÇÃO DE ERRO	BITS	PIXELS	PORCENTAGEM DA CORREÇÃO
L (LOW)	01		7%
M (MEDIUM)	00		15%
Q (QUARTILE)	11		25%
H (HIGH)	10		30%

Figura 9: Níveis de correção de erro e sua representação em *pixels*. Fonte: Produção dos próprios autores.

3.6. Tipo de armazenamento de dados

No canto superior direito é posicionado um quadrado de 2 *pixels* por 2 *pixels*, como destacado na Figura 10, cuja função é determinar o tipo de caracteres contido na mensagem. A Figura 11 mostra os principais tipos de decodificação de dados usados e suas representações em *pixels*.

O modo *numérico* é utilizado para representar dígitos decimais de 0 a 9. Ele é ideal para codificar informações numéricas, como números de telefone ou códigos de identificação. Já o modo *alfanumérico* permite representar uma variedade maior de caracteres, incluindo algarismos, letras maiúsculas, alguns caracteres especiais e espaços. Esse modo é adequado para codificar mensagens que contenham combinações de números, letras maiúsculas e certos símbolos. O modo *byte* é mais abrangente, permitindo números, letras maiúsculas e minúsculas, símbolos, dentre outros caracteres presentes na codificação de caracteres ISO 8859-1 [7, 8]. Por fim, o modo *Kanji* é específico para a língua japonesa, permitindo a codificação de caracteres Kanji. É utilizado principalmente em QR Codes destinados ao público japonês.

Para selecionar o tipo de armazenamento de dados mais adequado, é necessário analisar as necessidades específicas da mensagem. Por exemplo, se a mensagem consistir apenas de letras maiúsculas, o modo alfanumérico será suficiente, embora seja possível codificar a mensagem usando o modo bytes.

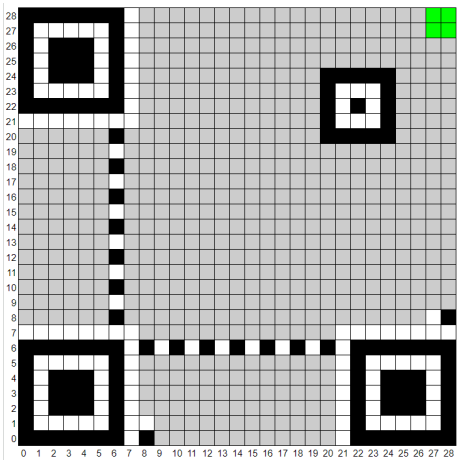


Figura 10: Posicionamento do padrão de armazenamento de dados. Fonte: Produção dos próprios autores.

TIPO	BITS	PIXELS
NÚMERICO	0001	
ALFANUMÉRICO	0010	
BYTE	0100	
KANJI	1000	
ECI	0111	

Figura 11: Tipos de armazenamento de dados. Fonte: Produção dos próprios autores.

3.7. Dark module, string de formato, informação de versão e Quiet Zone

O *dark module* é um único *pixel* preto posicionado na coordenada $(((4 * V) + 9), 8)$, onde V é a versão do QR Code. Assim, o *dark module* sempre fica localizado acima do canto superior esquerdo do separador inferior direito. A Figura 12 mostra a versão 7 do QR Code com o *dark module* na coordenada $(37, 8)$.

A *string de formato* é um conjunto de *pixels* composto por 15 *pixels*, como destacado em verde na Figura 12. Sua função é fornecer informações sobre a correção de erros, máscara (estudaremos na Seção 5) e versão do código. A criação da *string* é feita a partir da sequência de 5 *pixels* correspondentes à correção de erro e máscara, com o uso do método de correção de erros *Reed-Solomon*. Não iremos entrar em detalhes sobre esse processo por envolver diversas divisões polinomiais, mas as possíveis combinações estão listadas no site de tutorial [9].

A informação de versões são dois blocos localizados próximos ao padrão de localização, como destacado em azul na Figura 12, sendo que um bloco tem tamanho 6 *pixels* por 3 *pixels* e outro bloco tem tamanho 3 *pixels* por 6 *pixels*. Apenas as versões 7 ou maiores possuem tal padrão. Sua função é indicar a capacidade de armazenamento de dados e a versão do QR Code.

Os três padrões mencionados nesta subseção são adicionados quando as outras etapas responsáveis pela criação do QR Code estiverem feitas.

Por fim, a *Quiet Zone*, que significa Zona Tranquila em inglês, é uma área em branco ao redor do QR Code que fornece espaço de segurança e ajuda na leitura correta do código pelos leitores.

4. Bytes

Uma vez selecionado o tipo de codificação de dados a ser utilizado, é realizada a etapa de composição da mensagem de acordo com a tabela de codificação associada ao modo escolhido. É importante ressaltar que a mensagem codificada deve ser transformada da base 10 para a base

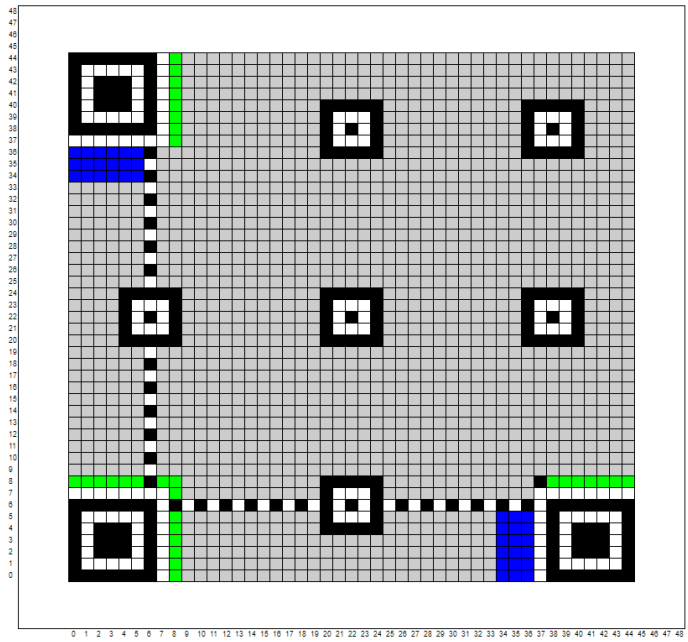


Figura 12: Posicionamento dos padrões *dark module, string* de formato e informação de versão. Fonte: Produção dos próprios autores.

2. Para exemplificar o procedimento, tomaremos a palavra "engraçadinho" (Tabela 2). Como a mensagem é formada por letras minúsculas e caracteres especiais, o modo de codificação preferível é o modo *byte*, por isso seguiremos a tabela ISO 8859 [7, 8].

Letra	Codificação em decimal	Codificação em binário
e	101	01100101
n	110	01101110
g	103	01100111
r	114	01110010
a	97	01100001
ç	231	11100111
a	97	01100001
d	100	01100100
i	105	01101001
n	110	01101110
h	104	01101000
o	111	01101111

Tabela 2: Codificação da palavra "engraçadinho". Fonte: Produção dos próprios autores.

Após o processo de codificação dos dados, a palavra-código deve ser transformada em *byte*. Cada caractere é armazenado em um *byte*, e cada *byte* é composto por 8 *bits*. Caso o caractere não

detenha 8 *bits*, são acrescentados 0's à esquerda, até que o *byte* esteja completo. Antes de prosseguirmos, é necessário salientar que o número 0 representa um *pixel* branco, enquanto o número 1 representa um *pixel* preto.

Os *bits* são lidos em um padrão de ziguezague percorrendo-se os módulos individuais do QR Code de acordo com a direção escolhida, que pode ser de baixo para cima ou de cima para baixo. Na Figura 13 apresentamos as possibilidades de leitura dos *bits* e na Figura 14 mostramos um exemplo.

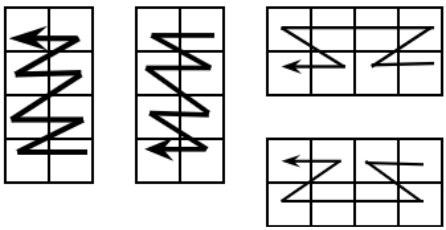


Figura 13: Padrão ziguezague. Fonte: Produção dos próprios autores.

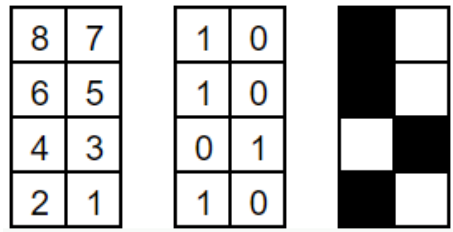


Figura 14: Exemplo de codificação da letra "e". Fonte: Produção dos próprios autores.

Após gerar a mensagem final em *bits* e organizá-la na ordem correta, é necessário colocar os *bits* na matriz do QR Code, seguindo uma estrutura específica. Antes de proceder com a estruturação da mensagem na matriz, é preciso montar o primeiro *byte*, conhecido como *indicador de contagem de caracteres*. Esse *byte* representa o número de caracteres que estão sendo codificados somado um, pois o indicador deve ser considerado parte da mensagem, e é uma sequência de *bits* derivada da conversão desse número para sua representação binária. Por exemplo, se a mensagem original contém 12 letras, o indicador de contagem de caracteres irá mostrar o valor 13 que em binário, é representado por "00001101". A Figura 15 mostra o indicador de contagem na cor cinza.

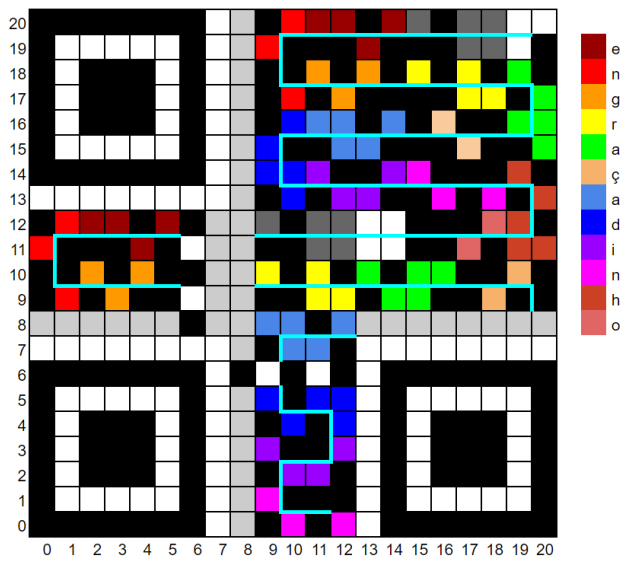


Figura 15: Estruturação da mensagem no QR Code. Fonte: Produção dos próprios autores.

Retomando a estruturação da mensagem na matriz, à esquerda do código que indica o tipo de dados utilizado, inserimos o indicador de contagem de caracteres, seguido da mensagem original. Em algumas versões, a cadeia de *bits* final pode não ser longa o suficiente para preencher o número total de *bits* na matriz. Nesse caso, é necessário adicionar um certo número de 0 ao final da mensagem para alcançar o comprimento correto. Esses 0 adicionais são conhecidos como *bits restantes*.

Para estruturar a mensagem da Tabela 2, escolhemos a versão 1 do QR Code, pois ela possui capacidade suficiente para acomodar a nossa mensagem. Nessa versão, são adicionados 4 *bits* restantes ao final da mensagem. Se a sequência de *bits* restantes for menor que 4 *bits* para atingir o final do QR Code, devemos adicionar apenas o número de zeros necessários para atingir o comprimento desejado. Após a sequência de zeros, a mensagem é repetida.

Durante a inserção dos *bits* na matriz, quando um padrão de função é encontrado, todos os módulos ocupados são ignorados até que o próximo módulo não utilizado seja alcançado. Da mesma forma, quando o padrão de alinhamento é encontrado, os módulos que fazem parte desse padrão são pulados, e a inserção dos *bits* continua em uma direção descendente.

A Figura 15 apresenta um exemplo da estruturação de uma mensagem no QR Code. O caminho percorrido pelos leitores de QR Code ao ler a mensagem é representado pela linha ciano. A leitura da mensagem começa a partir do padrão de armazenamento de dados e segue em ziguezague. Ao "caminhar" sobre o primeiro segmento da linha um padrão de localização deve estar à esquerda.

5. Máscaras

O mascaramento de informações é uma técnica aplicada aos dados codificados no QR Code, com o propósito de torná-los mais legíveis e evitar grandes proporções de *pixels* da mesma cor juntos. Existem oito padrões de máscaras disponíveis, cada um com o objetivo de modificar a aparência final do QR Code de diferentes formas. Cada máscara possui uma fórmula que determinará se o *pixel* será preto ou branco (veja a Tabela 3); se a fórmula funcionar, então o pixel será preto.

Máscara	**Bits**	**Fórmula**
0	000	$(i+j)\%2 = 0$
1	001	$i\%2 = 0$
2	010	$(j)\%3 = 0$
3	011	$(i+j)\%3 = 0$
4	100	$([(i/2] + [j/3])\%2 = 0$
5	101	$((i*j)\%2) + ((i*j)\%3) = 0$
6	110	$(((i*j)\%2) + ((i*j)\%3))\%2 = 0$
7	111	$(((i+j)\%2) + ((i*j)\%3))\%2 = 0$

Tabela 3: Fórmula e representação em *bits* das máscaras. Fonte: Produção dos próprios autores.

Na Tabela 3, o símbolo i representa a linha, o j representa a coluna, a % representa o resto da divisão e o símbolo [] representam o "maior inteiro menor que".

Cada *pixel* da máscara possui uma função específica: o *pixel* preto inverte a cor do *pixel* original, enquanto o *pixel* branco mantém a cor do *pixel* original. Essa técnica pode ser compreendida como a sobreposição da máscara sobre a mensagem original, resultando na soma dos valores correspondentes a cada *pixel* na mensagem. Nesse contexto, o *pixel* preto representa o valor 1, e por isso

ocorre a inversão de cor, enquanto o *pixel* branco representa o valor 0, mantendo a cor original, lembrando que na base 2, $1 + 1 = 0$.

É importante ressaltar que os padrões de máscara devem ser aplicados exclusivamente na mensagem, essa restrição garante que os padrões relevantes para a interpretação e decodificação da mensagem não sejam afetados pelo mascaramento.

A Figura 16 mostra a aplicação da máscara 2 em uma parte da nossa mensagem e a Figura 17 mostra a posição onde o padrão de máscara deve ser colocado.

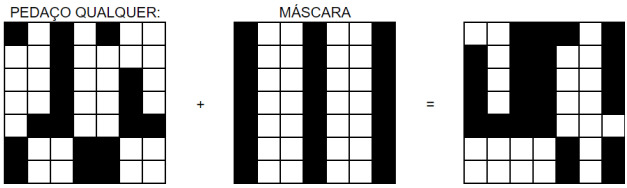


Figura 16: Exemplo de aplicação da máscara 010. Fonte: Produção dos próprios autores.

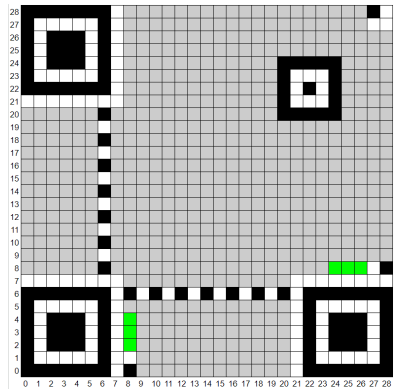


Figura 17: Posição do padrão de máscara. O padrão é definido pela coluna Bits da Tabela 3. Fonte: Produção dos próprios autores.

O gerador de QR Code seleciona a máscara mais adequada aplicando uma série de penalidades. Para determinar a melhor máscara, o gerador testará individualmente cada uma delas e escolherá aquela que apresentar a menor penalidade. A Tabela 4 apresenta essas penalidades.

Grau da penalidade	Característica
1	Grupo de cinco ou mais *bits* da mesma cor em uma linha (ou coluna).
2	Área 2×2 de módulos da mesma cor na matriz.
3	Padrões semelhantes aos padrões de localização.
4	Grande quantidade de *pixels* pretos ou brancos.

Tabela 4: Penalidades. Fonte: Produção dos próprios autores.

É importante destacar que as penalidades são aplicadas em todo o código, incluindo todos os padrões mencionados anteriormente, mesmo que a máscara não seja aplicada nesses padrões específicos. Isso significa que as penalidades são consideradas globalmente, levando em conta a estrutura completa do QR Code, a fim de garantir a melhor legibilidade e detecção do código pelos leitores.

A Figura 18 mostra o QR Code final da nossa mensagem. É possível fazer a leitura deste QR Code com um leitor e obter a mensagem "engraçadinho".

Versão do QR Code: Versão 1;
Canto inferior esquerdo dos padrões localizadores posicionados em (0,0), (14,0) e (0,14);
Tipo de armazenamento de dados: *Byte*;
Dark module posicionado em (13,8);
Nível de correção de erro: M;
Máscara: 110.

Figura 18: QR Code final da mensagem que construímos gerado por aplicativo [10].

6. Proposta didática

A proposta didática foi organizada em quatro partes, cada uma correspondendo a uma aula. A primeira aula aborda a conversão decimal-binário, a segunda aula trata da transformação binário-bytes, a terceira aula concentra-se na construção do QR Code, e a quarta aula aborda o mascaramento da mensagem. Ao fim da proposta, foi organizada uma atividade complementar opcional destinada ao aprofundamento do entendimento sobre matrizes com o auxílio do QR Code. Recomenda-se que as atividades sejam conduzidas com alunos do ensino médio, sendo aconselhável a divisão da turma em grupos para promover uma participação mais ativa e colaborativa durante as aulas.

6.1. Aula 1: Conversão decimal - binário

O objetivo dessa aula é compreender o processo de conversão de um número que está na base decimal para a base binária, onde serão apresentadas duas abordagens diferentes de conversão, a fim de enriquecer a compreensão do processo.

Atividade disparadora para aula: Escolha um número e transforme-o em binário.

- **Método das divisões sucessivas**

  O método de conversão consiste em dividir o número, que está em base decimal, sucessivamente por 2 e analisando o resto resultante de cada divisão. A seguir, expomos o passo a passo usando o número decimal 101 como exemplo. Como mostrado na Figura 19, o número 101 será dividido por 2, de modo que obtemos quociente 50 e resto 1 (circulado em vermelho); em seguida, o número 50 será dividido por 2, gerando quociente 25 e resto 0; o método segue assim até chegarmos em um quociente igual a 0.

  Agora, lemos os restos da divisão de baixo para cima, ou seja o resto da última divisão até a primeira, para obter o número binário equivalente: $101 = (1100101)_2$. Uma boa maneira de relembrar a direção em que o número deve ser lido é pensando na maior e menor potência. O resto da última divisão corresponde à maior potência de 2, logo faz sentido imaginar que esse resto equivale ao dígito mais à esquerda.

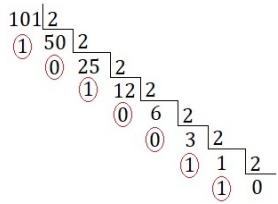


Figura 19: Método das divisões sucessivas. Fonte: Produção dos próprios autores.

- **Método de decomposição em potências de base 2**

  Esse método consiste em decompor um número na base decimal em suas potências de 2, partindo da maior potência possível que ainda "caiba" no número e, em seguida, utilizando potências menores até alcançar o valor desejado.

  Para o passo a passo usaremos o número na base decimal 101 como exemplo novamente. A potência de 2 mais próxima do decimal que será convertido é $2^6 = 64$. Agora pensaremos em outra potência menor que somado ao 64 chegue próximo ao 101, $2^5 = 32$. Assim, nós já temos $64 + 32 = 96$, porém ainda não alcançamos 101, que é nosso objetivo, então continuaremos fazendo o mesmo. Observe que não é possível utilizar 2^4 pois equivale a 16 e $64 + 32 + 16 = 112$, então teríamos que recorrer a potências menores. Repetindo o mesmo pensamento seria possível utilizar: $2^6, 2^5, 2^2, 2^0$. Dessa forma, $64 + 32 + 4 + 1 = 101$. Veja a Tabela 5.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	0	1	0	1

Tabela 5: Decomposição do decimal 101. Fonte: Produção dos próprios autores.

Esta atividade possibilita a apresentação da base binária de forma contextualizada. Esta linguagem é usada para gravar comandos no processador de um computador (EM13MAT405).

6.2. Aula 2: Transformação binário - bytes

Inicialmente, os alunos devem escolher uma mensagem para codificar e acessar o *site* [7] para identificar qual número na base decimal está associado a cada caractere selecionado.

Atividade disparadora para aula: Defina uma mensagem e transforme em *bytes* Figura 20.

LETRA	BINÁRIO	BYTE	LETRA	BINÁRIO	BYTE	LETRA	BINÁRIO	BYTE	LETRA	BINÁRIO	BYTE

Figura 20: Exemplo de tabela suporte. Fonte: Produção dos próprios autores.

Já com a mensagem escrita em binário, preencha a tabela da Figura 20 lembrando da disposição dos *bits* em *bytes* como visto na Seção 4. A seguir temos um exemplo do que deve ser feito usando a mensagem "engraçadinho".

LETRA	BINÁRIO	BYTE	LETRA	BINÁRIO	BYTE	LETRA	BINÁRIO	BYTE	LETRA	BINÁRIO	BYTE
e	1100101		r	1110010		a	1100001		n	1101110	
n	1101110		a	1100001		d	1100100		h	1101000	
g	1100111		ç	11100111		i	1101001		o	1101111	

Figura 21: Gabarito da atividade com a mensagem "engraçadinho". Fonte: Produção dos próprios autores.

Esta atividade permite que os estudantes entendam como um computador organiza os *bits* em sua memória (EM13MAT405).

6.3. Aula 3: Construção do QR Code no plano.

Neste ponto da sequencia didática, veremos o QR Code tomando forma. Queremos preencher uma região plana limitada, cujo tamanho será definido pela versão do QR Code, com *pixels* pretos e brancos que levarão em consideração os padrões e a mensagem que temos.

Atividade disparadora para aula: Construa o QR Code no plano seguindo as coordenadas referentes a cada padrão, registre esses dados e estruture a mensagem no plano cartesiano.

A Tabela 6 contém informações de coordenadas de alguns padrões, onde V representa a versão do QR Code escolhida pelo aluno. Observe que algumas células estão em branco, no entanto, é possível inferir as coordenadas referentes a essas células a partir dos padrões que estão destacados, e para descobrir as coordenadas dos padrões de alinhamento basta seguir a Tabela 6.

Padrão	**Coordenada**
Canto inferior esquerdo do padrão localizador inferior esquerdo	(0,0)
Canto inferior esquerdo do padrão localizador inferior direito	$([(((V-1)\cdot 4)+21)-7],\ 0)$
Canto inferior esquerdo do padrão localizador superior	$(0,\ [(((V-1)*4)+21)-7])$
Dark module	$([(4*V)+9],8)$
Pixel qualquer de cada separador	
Pixel qualquer de cada padrão de temporização	
Pixels do padrão de correção de erro	
Pixels do padrão de armazenamento de dados	
Pixel qualquer da *string* de formato	

Tabela 6: Tabela de coordenadas. Fonte: Produção dos próprios autores.

Em seguida, com tabela similar à da Figura 20 e a Tabela 6 preenchidas, estruture o QR Code no plano cartesiano da Figura 22. É importante lembrar das áreas reservadas e estruturação adequada da mensagem, como visto nas Seções 3.7 e 4, respectivamente.

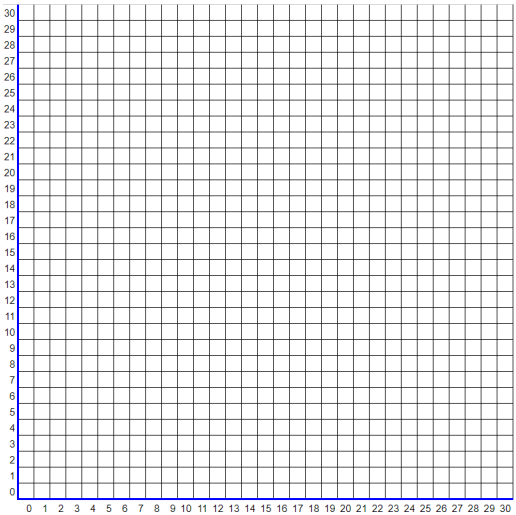


Figura 22: Exemplo do plano cartesiano. Fonte: Produção dos próprios autores.

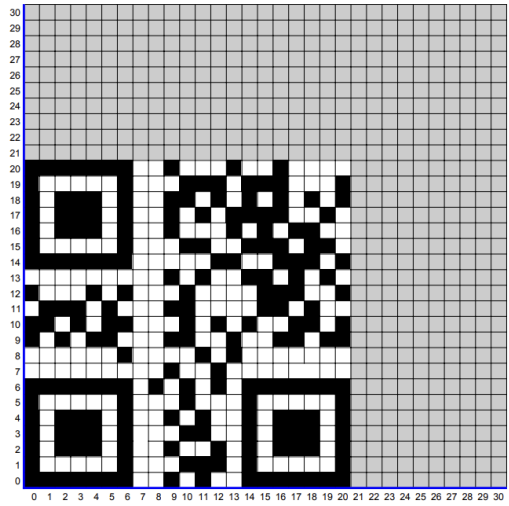


Figura 23: Gabarito da atividade com a mensagem "engraçadinho". Fonte: Produção dos próprios autores.

Esta atividade exige que posicionemos os *bytes* correspondentes a cada caractere da mensagem no QR Code por meio de translações, reflexões e rotações (EM13MAT105).

6.4. Aula 4: Mascaramento da mensagem

O último passo para finalizar a sequência didática será a aplicação de uma das máscaras apresentadas anteriormente na Tabela 3.

Atividade disparadora para aula: Aplique a máscara determinada ao seu grupo à sua mensagem.

Cada grupo receberá uma das oito máscaras presentes na Tabela 3 e deverá aplicá-la na estrutura montada no plano cartesiano seguindo as orientações da Seção 5. Para finalizar, adiciona-se as strings de formato e informação de versão.

Repare que o QR Code que construímos, presente na Figura 26 condiz com o QR Code da Figura 18.

Esta atividade estimula a realização de operações na base binária. Tais operações são realizadas rotineiramente pelos computadores em todos os seus processos (EM13MAT405).

A fim de promover a reprodutibilidade do nosso trabalho, disponibilizamos as planilhas usadas na construção dos QR Codes apresentados neste trabalho [11]. Desse modo, outros estudantes e professores poderão reproduzir e adaptar os QR Codes aos seus propósitos.

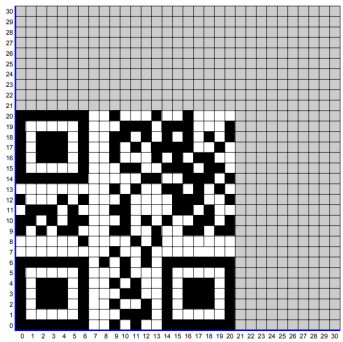


Figura 24: QR Code original. Fonte: Produção dos próprios autores.

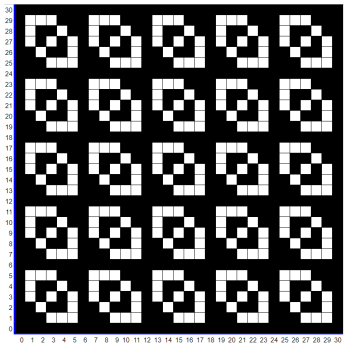


Figura 25: Máscara (110). Fonte: Produção dos próprios autores.

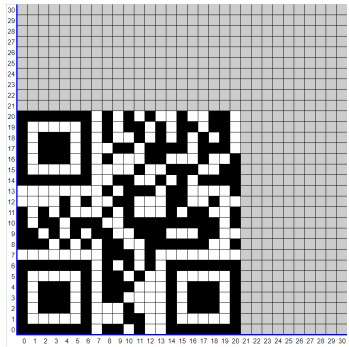
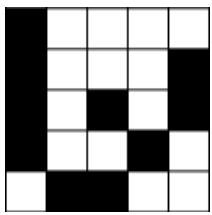


Figura 26: QR Code final. Fonte: Produção dos próprios autores.

6.5. Aula extra

Esta aula pode ser usada como uma aula completar para que os alunos vejam outras funcionalidades das matrizes e de suas operações.

Orientações: Como sugestão de atividade complementar, podemos explorar a representação de um QR Code como uma matriz. Nessa atividade, cada *pixel* do QR Code será associado a uma entrada em uma matriz, em que o branco será representado pelo número 0 e o preto pelo número 1.



$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Figura 27: Matriz associada ao canto superior direito do QR Code da Figura 23. Fonte: Produção dos próprios autores.

Além disso, podemos aplicar o mascaramento do QR Code utilizando a soma de matrizes na base 2. No exemplo a seguir, a primeira matriz é a mesma da Figura 27, a segunda matriz é associada à máscara 110 e a terceira é o resultado do mascaramento.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

7. Conclusão

Nossa expectativa ao elaborar esse artigo foi de, por um lado, explicar como funciona um instrumento presente no nosso cotidiano, e, por outro lado, mostrar onde a Matemática encontra-se nesse assunto e como ela contribui para a eficácia e utilidade do QR Code. Ao explorar a interseção entre a tecnologia e a Matemática, pudemos perceber como os algoritmos e os princípios matemáticos contidos são essenciais para a criação e decodificação dos QR Codes. À medida que continuamos a avançar em um mundo cada vez mais digital, o QR Code certamente desempenhará um papel fundamental em nossa interação com a informação. Portanto, esperamos que com este artigo, tenhamos fornecido informações suficientes para cativar professores e estudantes.

Referências

[1] Answers to your questions about the QR Code. Disponível em: <https://www.qrcode.com/en/>. Acesso em: 11 de junho de 2023.

[2] BRASIL. Ministério da Educação. *Base Nacional Comum Curricular*. Brasília, 2018.

[3] Código QR. Wikipédia. Disponível em: <https://pt.wikipedia.org/wiki/C%C3%B3digo_QR>. Acesso em: 11 de junho de 2023.

[4] Leitor de código QR. Google Play. Disponível em <https://play.google.com/store/apps/details?id=com.teacapps.barcodescanner>. Acesso em: 20 de julho de 2023.

[5] Scanner de QR Código de Barras. Google Play. Disponível em <https://play.google.com/store/apps/details?id=com.gamma.scan>. Acesso em: 20 de julho de 2023.

[6] ROUSSEAU.C; SAINT-AUBIN, Y. *Matemática e atualidade* volume 1. Rio de Janeiro: SBM, 2015 (Coleção PROFMAT).

[7] Tabela ISO/IEC 8859-1. Wikipédia. Disponível em: <https://pt.wikipedia.org/wiki/ISO/IEC_8859-1>. Acesso em: 11 de junho de 2023.

[8] ISO/IEC 8859-1 (codificação de caracteres do alfabeto latino). Disponível em: <https://www.iso.org/standard/28245.html>. Acesso em: 02 de janeiro de 2024.

[9] Format and Version String Tables. Thonky, 2023. Disponível em: <https://www.thonky.com/qr-code-tutorial/format-version-table>. Acesso em: 11 de junho de 2023.

[10] QR Code Monkey. Disponível em: <https://www.qrcode-monkey.com/pt/#text>. Acesso em: 11 de junho de 2023.

[11] Planilhas usadas na construção dos QR Codes. Disponível em: <https://docs.google.com/spreadsheets/d/1ACuWwtCYjvtCrUIZNw4CeN7a2emljqKw7fXH22Q2X5E/edit?usp=sharing>.

[12] QR Code Tutorial. Disponível em: <https://www.thonky.com/qr-code-tutorial/>. Acesso em: 11 de junho de 2023.

[13] Silva, Alberto Renan Dias. A matemática do código de barra e QR code. Orientador: Silas Fantin. 2021. 70f. Dissertação, Pós-graduação em matemática, Profmat, Unirio, Rio de Janeiro. 2021. Disponível em: <https://sca.profmat-sbm.org.br/profmat_tcc.php?id1=6556&id2=171055699>. Acesso em: 11 de junho de 2023.

[14] Stroski, Pedro Ney. Código QR: O que é e como funciona? Electrical e library, 2019. Disponível em: <https://www.electricalelibrary.com/2019/11/22/codigo-qr-o-que-e-e-como-funciona/>. Acesso em: 11 de junho de 2023.

Rodrigo Araujo de Souza
Universidade Federal do Espírito Santo
<rodrigoaraujo8@live.com>

Thainá Lopes Ferreira
Universidade Federal do Espírito Santo
<profathainaferreira@gmail.com>

Weslaine Herzog Santos
Universidade Federal do Espírito Santo
<weslainehs@outlook.com>

Alcebiades Dal Col
Universidade Federal do Espírito Santo
<alcebiades.col@ufes.br>

Recebido: 20/04/2020
Publicado: 08/03/2024