U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76001 CARLOS EDUARDO ANTAO 24FRO202 - 1 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111110110101101
00010000000100111110110110110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
---------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

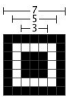
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

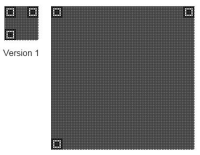
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



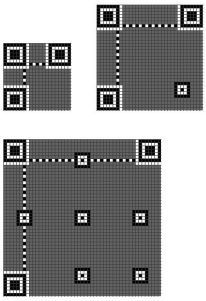
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000111111001O1
00100000011010011001001010010110111000101010O
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001100011101011110101011101001110111l0
2.
000100000001000l1100010000011
3.
00100000001100101110110100111100100101010010010
4.
000100000001000l1001001100000
```

1	2	3	4

202-76001 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76199 DANILO DEPETRIS SOARES 24FRO202 - 2 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
-----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

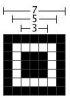
```
00100000001100100000000101111001001010000100110
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

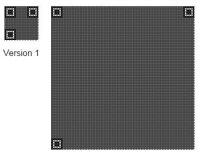
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



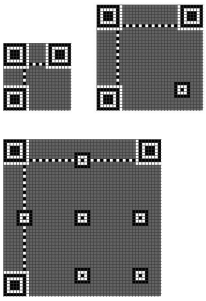
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101100
00010000000100000111111001010
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001100101101101010110000010100111011010
2.
00010000000100010011101110000
3.
00010000000100100100110101010
4.
00100000001100011110000010011010111100110010001
```

1	2	3	4

202-76199 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76018 EDUARDO CAVALHEIRO DIAS 24FRO202 - 3 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
-----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

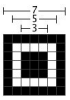
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

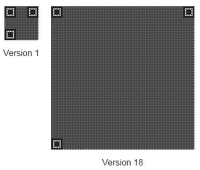
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



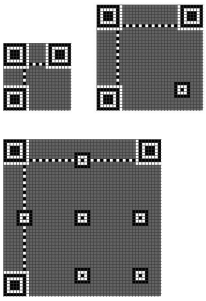
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000111111001O1
00100000011010011001001010010110111000101010O
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00010000000100011010110100l1
2.
00100000001101010010101101101111011111001001
3.
00010000000100001001111110000
4.
00100000001100011110000101101000010010100100l0
```

1	2	3	4

202-76018 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76025 ERIC CORDEIRO 24FRO202 - 4 entregar ate 23/maio / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000011000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

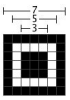
```
00100000001100100000000101111001001010001001l0
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

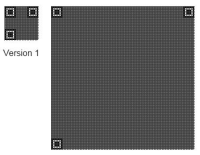
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



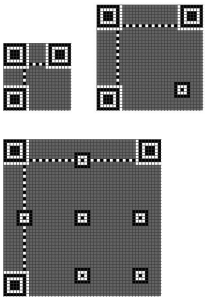
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000111111001O1
00100000011010011001001010010110111000101010O
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001100011111100110100100011101001000011
2.
000100000001000110110000011l
3.
00010000000100100001100101O1
4.
00100000001010111100001110110001110011000
```

1	2	3	4

202-76025 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76032 FELIPE EIJI KANO 24FRO202 - 5 entregar ate 23/maio / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00011000000010100011110110101101
00011000000010011110110110110
00011000000011000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
---------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

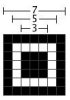
```
00100000001100100000000101111001001010001001l0
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

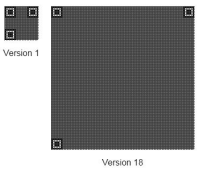
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



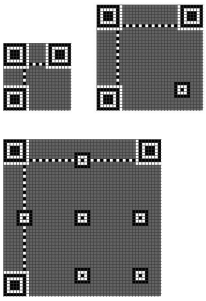
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000111111001O1
00100000011010011001001010010110111000101010O
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001010011110000101100110100011000
2.
00010000000100010100101000l0
3.
00010000000100001111000010O1
4.
00100000001010110111010101111000010011000
```

1	2	3	4

202-76032 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76049 FELIPE JESMIEL LEITE 24FRO202 - 6 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão v é $21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000011000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
-----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

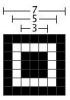
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

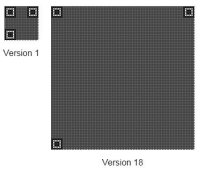
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



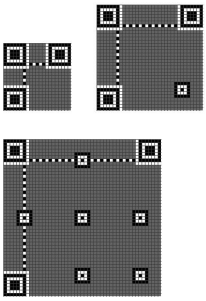
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000111111001O1
00100000011010011001001010010110111000101010O
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00010000000100010101110100O0
2.
00100000001011011000010101001100100011000
3.
00100000001010100000011101110111011011000
4.
0001000000010001000111010111
```

1	2	3	4

202-76049 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76056 GABRIELA BERNARDO PASQUALOTT 24FRO202 - 7
entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão v é $21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

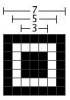
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

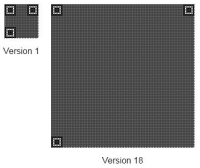
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



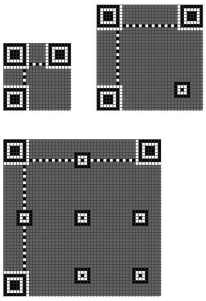
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000111111001O1
00100000011010011001001010010110111000101010O
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001101001100100101001011011100010101O0
2.
000100000001000111010110010O
3.
00100000001011011000010101001100100011000
4.
0001000000010001010101110111
```

1	2	3	4

202-76056 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76063
GUILHERME VITOR DA SILVA 24FRO202 - 8
entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000011000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
-----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

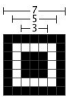
```
00100000001100100000000101111001001010001001l0
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

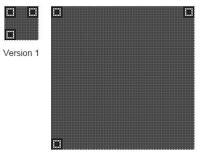
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



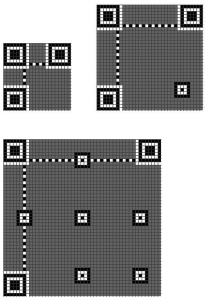
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000111111001O1
00100000011010011001001010010110111000101010O
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00010000000100001111000101O1
2.
00100000001010110111010101111000010011000
3.
00100000001100100011010001010001011011000101O1
4.
00010000000100100001110l0110
```

1	2	3	4

202-76063 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76087 LAURA PACHECO DELATTRE 24FRO202 - 9 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão v é $21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
-----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

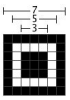
```
00100000001100100000000101111001001010000100110
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

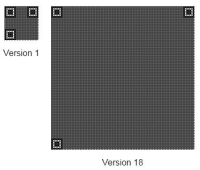
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



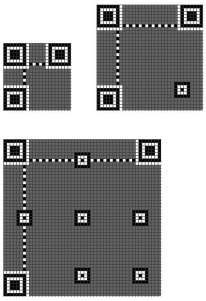
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000111111001O1
00100000011010011001001010010110111000101010O
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001100011110000010011010111100110010O1
2.
000100000001000111110100000l
3.
00100000001101000111110110011011100100010101010O
4.
000100000001001000000010l000
```

1	2	3	4

202-76087 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76106 LUIZ MIGUEL OLINEK 24FRO202 - 10 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
-----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

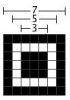
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

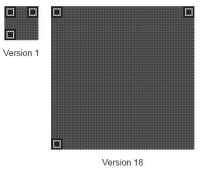
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



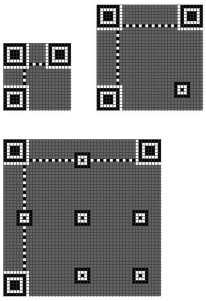
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101100
00010000000100000111111001101
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00010000000100100000101100110
2.
00100000001100100010011010000011001011101111011
3.
00010000000100011111001101001
4.
00100000001010011101011110101011101001010
```

1	2	3	4

202-76106 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76113 NICOLE DA SILVA DE AZEVEDO 24FRO202 - 11 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000011000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
---------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

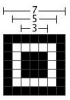
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

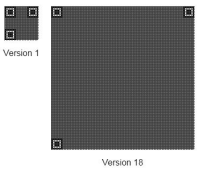
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



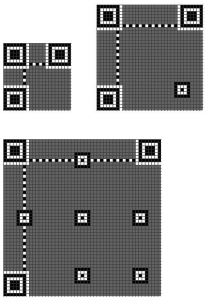
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101100
00010000000100000111111001010
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00010000000100010000011010011
2.
00100000001011011000010101001100100011000
3.
00100000001100100010011010000011001011101110011
4.
00010000000100010001101101l0
```

1	2	3	4

202-76113 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76120 PAULO JOSE FREIRE CORREA 24FRO202 - 12
entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000011000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
---------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

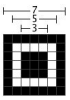
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

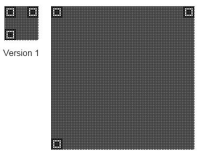
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



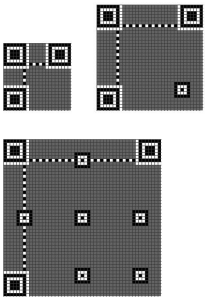
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101100
00010000000100000111111001010
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00010000000100100010110001100
2.
00010000000100001000011010100
3.
00100000001010100000011101110111011011100
4.
00100000001101001100100101001011011100010101000
```

1	2	3	4

202-76120 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76137 PEDRO HENRIQUE PIEKARSKI DE 24FRO202 - 13
entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
-----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

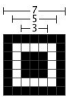
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

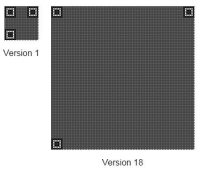
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



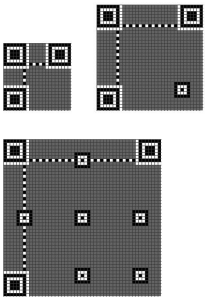
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101100
00010000000100000111111001010
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001100011101011110101011101100010101000
2.
00010000000100010111111000011
3.
00100000001010100010110100111100001001110
4.
00010000000100000011010110011
```

1	2	3	4

202-76137 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76144 RAUL KNAPKI CUNHA 24FRO202 - 14 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

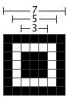
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

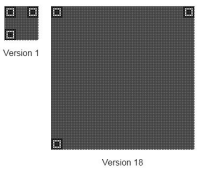
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



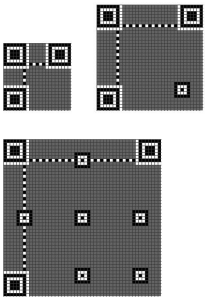
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101100
00010000000100000111111001010
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00010000000100000111000010000
2.
00100000001100011101011110101011101100010101000
3.
00010000000100010100011100l0
4.
00100000001011100011010101000000101000l010
```

1	2	3	4

202-76144 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76168 VICTOR BRONDANI DOS SANTOS 24FRO202 - 15
entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
---------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

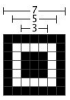
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

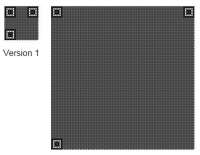
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



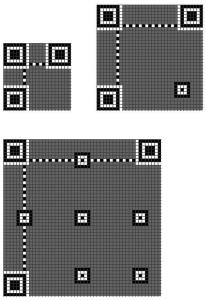
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101100
00010000000100000111111001010
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001101000111110110011011100100010101000
2.
00010000000100001100101001100
3.
00010000000100011000011110001
4.
00100000001010011100110110011010001010101
```

1	2	3	4

202-76168 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76175 VICTOR HUGO DOS SANTOS DE CA 24FRO202 - 16 entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão v é $21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000011000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
-----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

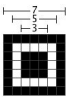
```
00100000001100100000000101111001001010001001l0
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

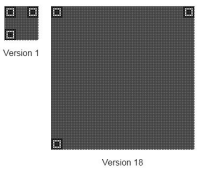
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



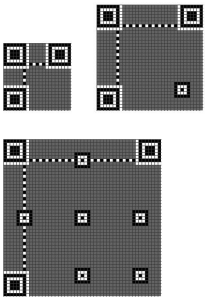
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101l0
00010000000100000011111100101
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00010000000100000111100101O1
2.
00100000001010011110000001110111011011100
3.
00100000001100101110110100111100100101010010010
4.
00010000000100011011001001l1
```

1	2	3	4

202-76175 - ga/ a

U Positivo - UTFPR - PUCPr – 20/05/2024 - 10:30:39.4
Prof Dr P Kantek (pkantek@gmail.com)
QR Code VIVX571a, V: 1.06 76182 VITOR EDUARDO DE GOES FONTES 24FRO202 - 17
entregar ate 23/maio _____ / _____ / _____

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão v é $21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivx570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados. Eis alguns exemplos: (O modo pode ser n=numérico, alf=alfanumérico, byte=binário e kanji)

v.		nív	bits	n.	alf.	byte	kanji
1	21	L	152	41	25	17	10
	x	M	128	34	20	14	8
	21	Q	104	27	16	11	7
		H	72	17	10	7	4
...							
10	57 x	L	2192	652	395	271	167
	57	H	976	288	174	119	74
...							
40	177 x	L	23648	7089	4296	2953	1817
	177	H	10208	3057	1852	1273	784

Análise O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

O string começa com o indicador de modo. É um número binário de 4 bits a saber: 0001=numérico, 0010=alfanumérico, 0100=byte, 1000=kanji e 0111=ECI (significa Extended Channel Interpretation, para outros tipos de codificação). Depois disso, vem um número variável de bits indicando o comprimento das informações. A quantidade de bits desta segunda informação varia dependendo da versão e do modo da codificação. Assim:

versões	n	a	b	k
1 a 9	10 bits	9	8	8
10 a 26	12	11	16	10
27 a 40	14	13	16	12

A maneira de codificar a informação, varia dependendo do modo empregado. Se o modo é numérico, os digitos são agrupados de 3 em 3. Com isso o último grupo pode ficar com 1, 2 ou 3 dígitos. A seguir cada grupo é convertido em um número binário e o número é transcrito normalmente. A seguir, a conversão de 12345, 9876 e 555 cujas respostas serão

```
00010000000101000111101101011101
00010000000100111101101101110
00010000000111000101011
```

Se o modo é alfanumérico, os caracteres são agrupados de 2 em 2 e cada caracter é transformado em um número usando a tabela:

```
          1         2         3         4
0123456789012345678901234567890123456789012345
----------------------------------------------
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:
```

Agora cada par 2 de números é convertido em um único número multiplicando o primeiro por 45 e somando com o segundo. Depois o resultado é convertido em binário e transcrito normalmente. Por exemplo, sejam os textos BILOCA e VIVXS, cujas conversões darão:

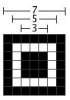
```
00100000001100100000000101111001001010001001100
00100000001011011000010110110001011011100
```

Agora, é necessário decidir o nível de detecção e correção de erros. Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

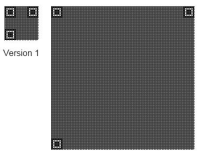
Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Criação do desenho

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7 × 7, fora de uma linha branca de 5 × 5, que fica fora de um quadrado preto de 3 × 3. Veja o desenho:



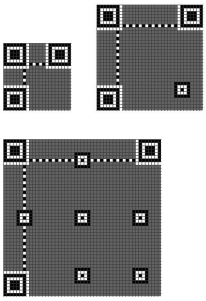
Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento. Veja a seguir na figura em que se descreve o padrão de tempo. Agora o padrão a incluir é chamado *timing patterns* que são umas linhas alternando brancos e pretos colocadas entre os *find patterns*. Veja como fica este padrão junto com o padrão de alinhamento (eventualmente nenhum ou mais de um padrão de alinhamento, dependendo da versão).



Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo deste exercício, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

Em Resumo

Os primeiros 4 dígitos informam o modo. Daí, se modo=n (0001), e versão=1 os próximos 10 bits indicam o tamanho. Daí cada 10 bits são convertidos em 3 dígitos. No final podem sobrar 2 ou 1 dígitos que são representados em 7 ou 4 bits. Se modo=a (0010), segue-se um tamanho de 9 bits e depois os caracteres aparecem de 2 em 2. Aqui chamados de xy. Cada conjunto de 2 ocupa 11 bits. (O último ao final pode ser um caracter isolado ocupando 6 bits). Deve-se converter esse número de 11 bits em decimal ($\rightarrow N$). Daí fazer N mod 45 = y (o segundo caracter) e N div 45 = x (o primeiro caracter do grupo).

Exemplos

Suponha as mensagens `BANCO`, `3716`, `1265` e `RADIOS`. Elas serão codificadas como

```
00100000001010011111100110000010111011000
00010000000100010111001101100
00010000000100000011111100101
00100000001101001100100101001011011100010101000
```

☞ Para você fazer

Suponha que ao decodificar o conteúdo de um determinado QR CODE voce recebeu os seguintes 4 strings de bits. Cada um deles representa uma palavra ou um número. Descubra-os:

```
1.
00100000001010101011010110000010101010101
2.
00100000001010101101101010100110001011100
3.
00010000000100000011011110000
4.
00010000000100011111001100000
```

1	2	3	4

202-76182 - ga/ a