E-fólio A | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Sistemas em Rede

CÓDIGO: 21106

DOCENTE: Arnaldo Santos

A preencher pelo estudante

NOME: Júlio César Gomes de Barros

N.º DE ESTUDANTE: 1902295

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 24/11/2022

1) Desenvolvidos com o propósito de padronizar o processo de criação de redes, os modelos OSI e TCP/IP baseiam-se em camadas (layers), onde cada camada comunica com a que está acima e abaixo de si, criando uma abstração, e se aqui reside uma semelhança, também aqui se encontra uma diferença pois o modelo OSI tem 7 camadas e o modelo TCP/IP é apresentado com 4 ou por vezes 5 camadas, havendo entre modelos camadas equivalentes, como se representa no quadro seguinte

	Modelo	
camada	OSI	TCP-IP
7	Aplicação	Aplicação
6	Apresentação	
5	Sessão	
4	Transporte	Transporte
3	Rede	Internet
2	Enlace de dados	Enlace
1	Física	Física

Considerando as 5 camadas no modelo TCP/IP, ou no caso de se pensar apenas em 4, mas onde a camada de enlace incorpora a camada física, ambos os modelos oferecem um serviço de transporte ponta a ponta, independente da rede, para que processos possam comunicar entre si. Em ambos os modelos, acima da camada de transporte, fazem uso dela as aplicações, embora o modelo OSI apresente pelo meio as camadas de sessão e apresentação para aceder a esta camada de topo.

O modelo OSI é bastante generalista, continuando a ser usado presentemente, embora os protocolos associados ao modelo não sejam usados. Já o modelo TCP/IP ao invés, é inadequado para descrever qualquer pilha de protocolos para além dos que integram o modelo, mas são esses mesmos protocolos o seu ponto forte, sendo bastante utilizados (HTTP, FTP, TCP, UDP, IP, 802.11…), ainda que, para além dos protocolos TCP e IP, bem estruturados, muitos tenham sido criados ad-hoc.

2) Vantagens da fibra ótica em relação à tecnologia cobre (par trançado), entre outras, serão a maior taxa de transmissão e largura de banda, que só não é maior porque os fotodíodos necessários para converter a luz em sinais elétricos nos recetores a limitam a 100Gbps, sendo que, nas 3 bandas inferiores de transmissão de 850nm, 1300nm e 1550nm a largura de banda é de até 30GHz (Tanenbaum 2014), embora como a fibra ótica é uma tecnologia em franco desenvolvimento seja ingrato falar de números. [1]

[1] Consultar: https://en.wikipedia.org/wiki/Fiber-optic_communication

Por contraposição e considerando uma janela temporal próxima do exposto anteriormente, a tecnologia cobre fica-se por uma taxa de transmissão de 2.5Gbps e largura de banda 120MHz para categoria cobre CAT5e, e 10Gbps/250MHz respetivamente, para categoria CAT6. [2]

Outros valores de comparação referenciados no livro recomendado na UC, são a distância entre repetidores necessários para transmissão a longas distâncias, ser de 50km para fibra (monomodo) devido às baixas perdas, e 5km para o cobre. Também o peso, sendo que um cabo com 1km de comprimento e contendo 1000 pares de cobre, pesa 8 Toneladas, enquanto um cabo com 2 fibras, que tem maior capacidade de transmissão, pesa apenas 100kg.

Outra grande valência da fibra ótica é a imunidade ao ruído eletromagnético, não existindo problemas de cross-talk entre fibras, como acontece no par trançado, onde num cabo CAT.X os pares são trançados com passos diferentes de torção para reduzir o cross-talk, entre pares e entre si (comunicação cruzada ou induzida entre condutores paralelos) e com eventual recurso a malha exterior para minimizar indução de ruído externo. Dado que a fibra é imune a ruídos eletromagnéticos e térmicos, pode-se correr a fibra paralela a cabos de tensão, algo que deve ser evitado com cabos de comunicação em cobre.

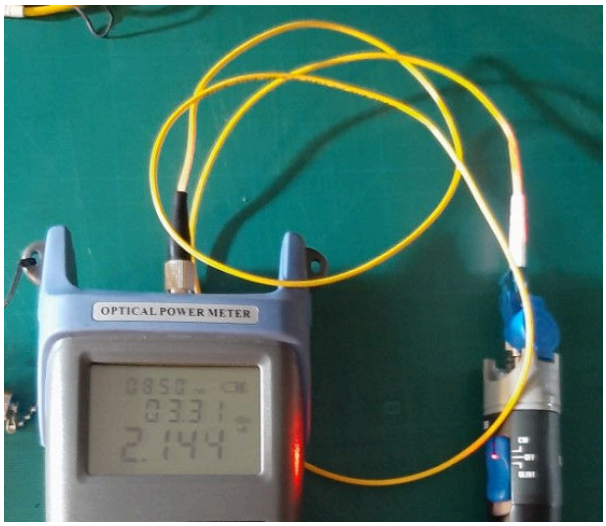
As maiores desvantagens da fibra relativamente ao cobre são a maior dificuldade de manutenção do meio físico de transmissão, pois é mais fácil fazer reparações na cablagem cobre do que na fibra, sendo que esta requer equipamento e um técnico especializado para cortar e fundir a fibra (o vidro), embora haja ligadores mecânicos, mas que devem ser evitados a não ser em casos de reparação de emergência ou instalações menores. Os equipamentos terminais de fibra também encarecem a instalação, mas tudo depende da dimensão da instalação, pois a fibra consegue rapidamente compensar a grandes distâncias e para elevados volumes de tráfego.

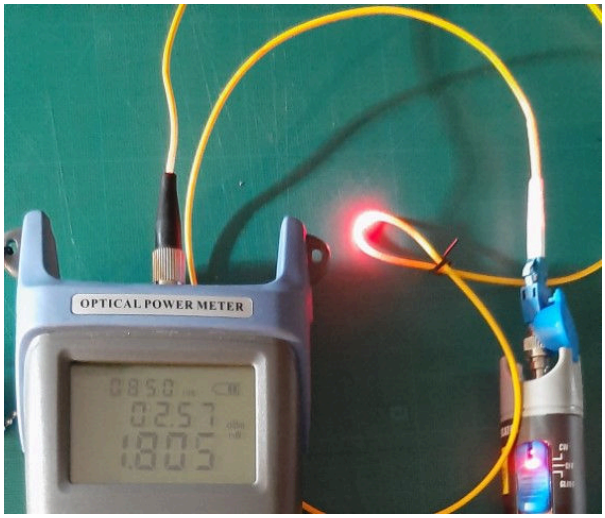
Por outro lado, a fibra também requer alguns cuidados mecânicos. Embora tenha bom comportamento relativamente à tração (quando se puxa por exemplo um cabo dentro de tubagem na altura da instalação), tem as suas fragilidades quando à dobragem, ou raio de curvatura máximo admissível. Mesmo que a fibra não parta no interior do cabo por dobragem excessiva, será induzida perda por refração, pois a luz dentro da fibra pode ultrapassar o ângulo crítico de reflexão e a luz começa a ser refratada produzindo perdas.

De seguida apresentam-se 2 imagens onde se mede a potência luminosa num patch de FO, no comprimento de onda λ de 850nm, e no caso da imagem da esquerda com a fibra apenas enrolada normalmente os valores medidos são de 3.31dB/2.144mW e na

[2] Consultar: https://en.wikipedia.org/wiki/Twisted_pair

imagem da direita com a fibra dobrada medem-se os valores de 2.57dB/1.805mW, portanto, com perdas relativamente ao caso anterior, e pode ver-se a luz a “escapar” da fibra na zona onde esta está excessivamente dobrada.



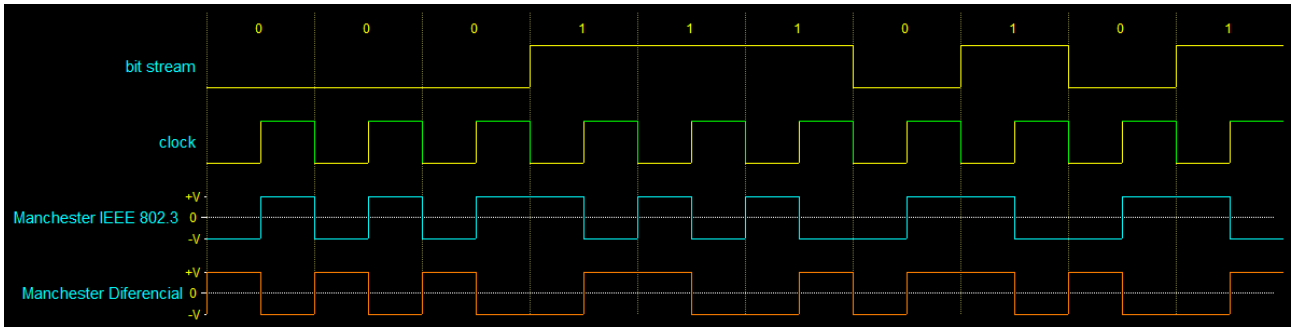


3) As codificações Manchester e Manchester diferencial, são polares, ou seja, os níveis de sinal atingem níveis de tensão simétricos, positivos e negativos, eliminando a componente DC do sinal, graças também ao facto de serem codificações bifase por conterem transições a meio do bit o que significa que, seja qual for o estado lógico, um bit estará sempre durante meio tempo com tensão positiva e o outro meio tempo com tensão negativa, garantindo assim uma média de tensão nula, portanto, sem componente contínua DC. Por outro lado, a inversão a meio do bit permite a sincronização. A frequência de transições é definida pelo sinal de clock e esta é o dobro da frequência do sinal de dados (ou bit stream), o que facilita a sincronização do lado do descodificador, pois o sinal de clock está “embutido” no sinal codificado transmitido. No caso da codificação Manchester IEEE os dados enviados sofrem uma operação XOR com o clock. A codificação Manchester Diferencial carece de uma abordagem mais elaborada[3].

A duplicação da frequência como resultado da codificação Manchester relativamente à frequência de entrada de dados, implica o aumento de largura de banda necessária à transmissão do sinal codificado, mas tal não constitui um problema se usado cabo coaxial, tendo sido esta codificação usada na rede local Ethernet original (IEEE 802.3).

A variante da codificação Manchester G. E. Thomas, não considerada aqui, pode ser conseguida à conta de um XNOR, portanto com estados simétricos à variante IEEE.

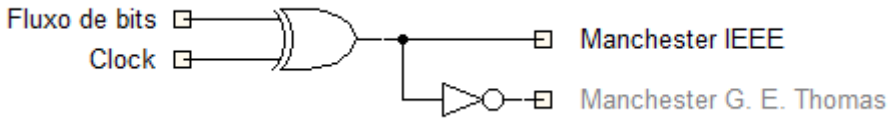
[3] http://ijsetr.com/uploads/234651IJSETR15244-878.pdf



Na mesma figura[4] acima, temos:

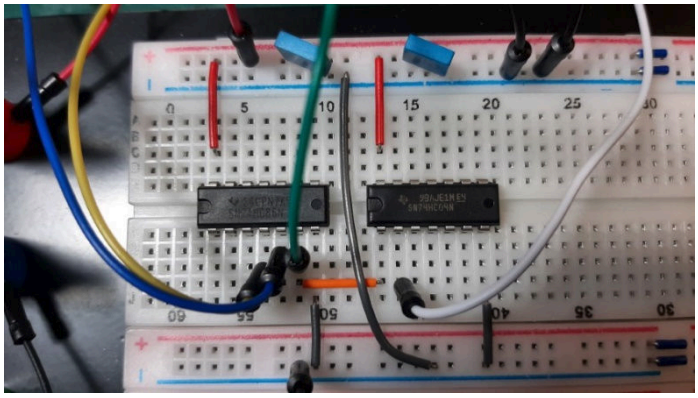
a) A azul o resultado da <u>codificação Manchester</u> do fluxo de bits (bit stream) 0001110101. A codificação é conseguida pela realização de um XOR entre o fluxo de bits a amarelo e o sinal de relógio (ou clock) em traço alternado amarelo-verde.

Bit St	Clock	XOR
0	0	0
0	1	1
1	0	1
1	1	0



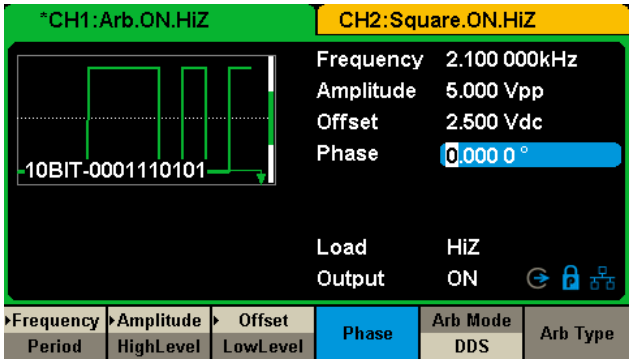
Por observação gráfica verifica-se que, sempre que no fluxo de bits temos um 0, como resultado da codificação Manchester surge uma transição de 0 para 1, e quando no fluxo de bits está presente 1, como resultado da codificação ocorre uma transição de 1 para 0.

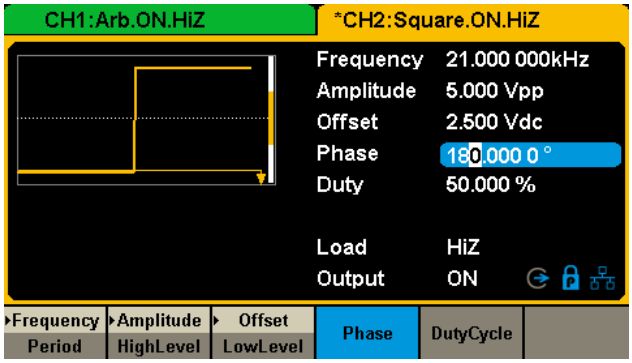
A título de curiosidade foi realizada uma montagem simples com um 74HC86 (XOR TTL), um 74HC04 (NOT TTL) e dois condensadores MKT de 100nF para desacoplamento.



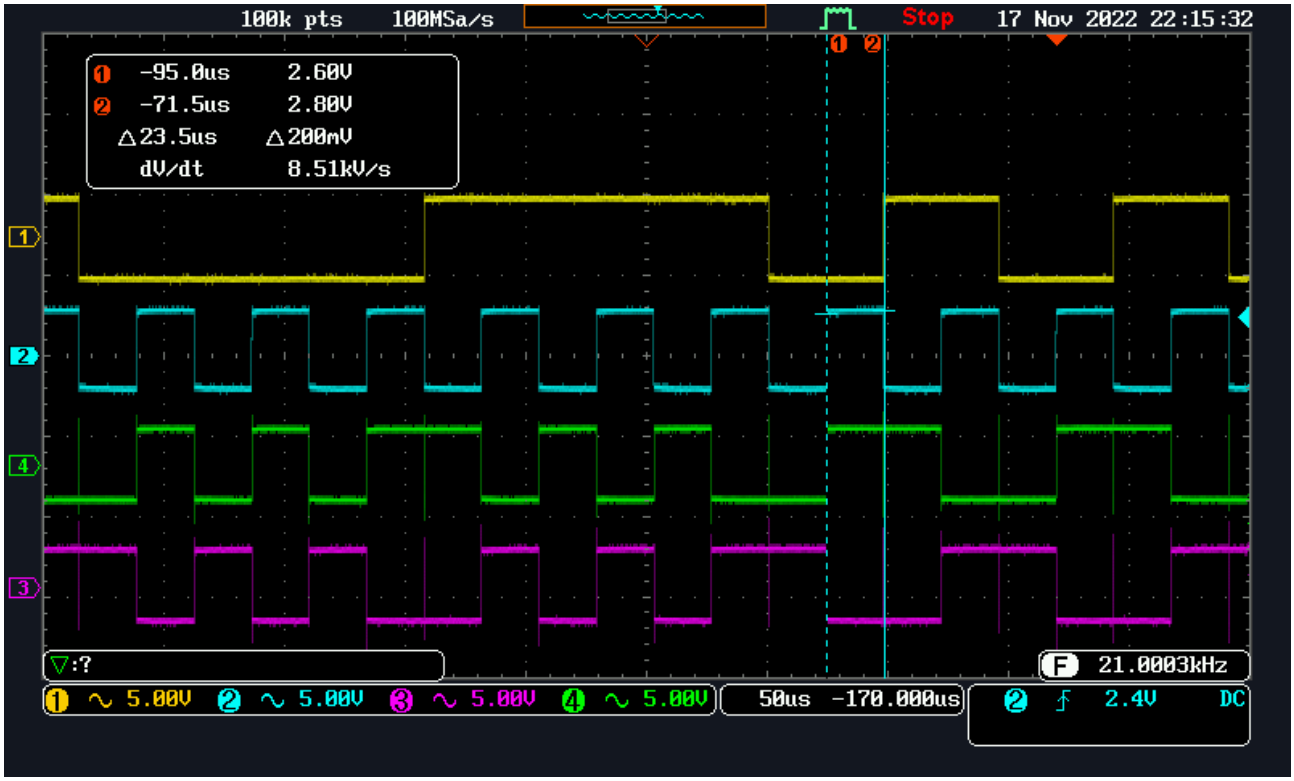
[4] https://waveme.weebly.com/

Nas entradas das portas XOR foi injetado um sinal previamente construído num Gerador de forma de onda arbitrária (AWG) com 2 canais, ligando uma das entradas do XOR ao canal 1 (CH1) do gerador que forneceu o fluxo de 10 bits 0001110101, num bloco com frequência de 2.1KHz, o que dá o equivalente a uma frequência de 21KHz, com período de bit de 47.6μs. Na outra entrada da porta do XOR foi injetado do canal 2 (CH2) do gerador um sinal quadrangular de frequência 21KHz (clock), com representação de 2 estados (0 e 1) por ciclo, ou seja, com período de bit de 23.8μs (ver medidas de cursor no osciloscópio).





O resultado foi observado num osciloscópio digital, como se mostra na imagem a seguir. No canal 1 do osciloscópio temos representado o fluxo de bits 0001110101 com traço amarelo, aplicado numa das entradas do XOR. No canal 2, com traço azul, temos o sinal de clock aplicado na outra entrada do XOR. No canal 4 a verde, temos o resultado da codificação Manchester, que mais não é que o XOR das duas entradas.



Note-se que no canal 4 (e no 3) há quase sempre um indesejado fino traço vertical visível quando o bit (do fluxo de bits) mantém o estado, onde na realidade ocorre uma curta transição, porque entre os 2 sinais de entrada (fluxo de bits e clock) há um ligeiro desfasamento temporal e por breves instantes os sinais são momentaneamente iguais ou diferentes, o que produz um pico a 0 ou a 1.

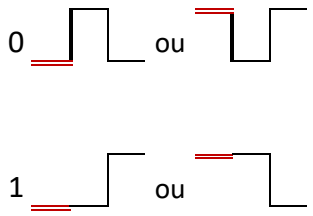
No canal 3, a magenta, está representado o Manchester G. E. Thomas, obtido pela negação da variante IEEE à custa de um 74HC04, negação aplicada após o XOR (⟺ XNOR).

b) Na 1ª figura desta questão 3, a cor de laranja, apresenta-se o sinal resultante da <u>codificação Manchester Diferencial</u>, para o mesmo fluxo de bits 0001110101.

Esta codificação impõe que o estado atual dependa do anterior, pelo que, tendo sido considerado que a linha se encontrava no estado baixo, ou seja, no nível 0, e o 1º bit do sinal a codificar também é o 0, começa por ocorrer logo uma transição de nível de 0 para 1. Com esta codificação, um bit 0, produz imediatamente uma comutação de nível, para 1 se o estado anterior era 0, e se o estado anterior era 1, comuta para 0. Depois, tal como na codificação Manchester IEEE, a meio do período do bit de entrada, ocorre uma transição de estado no sinal de saída.

Quando o bit de entrada é 1, a codificação mantém o nível do estado anterior, continuando a 0 se este era 0, ou a 1 se o estado anterior era 1, com a espectável transição de nível a meio do período do bit de entrada.

Síntese de estados, com o traço vermelho a representar o nível do estado anterior, e com traço preto grosso a necessária transição logo no início para a representação do 0.



4) O Código de **Hamming** tem como propósito a deteção e correção de erros na transmissão de um fluxo de dados. Para tal, recorre a bits de paridade inseridos na mensagem a transmitir, em posições que são potências de 2, intervalados com os restantes bits do fluxo de dados (mx) a transmitir, em posições definidas (Px).[5]

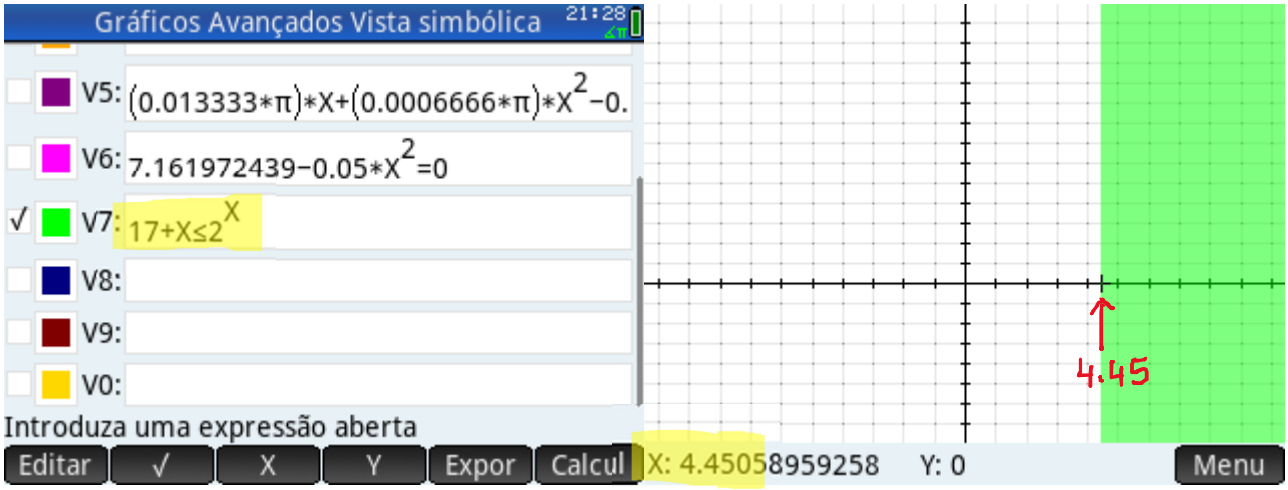
[5] Algoritmo do código de Hamming descrito em: https://en.wikipedia.org/wiki/Hamming_code

Seja *m* a dimensão da mensagem, *r* a quantidade de bits de paridade (check bits), *n* a dimensão da palavra a transmitir, tal que $n=m+r$.

O nº de bits de paridade *r*, necessários para formar a palavra a transmitir com código de Hamming para correção de erros, deve ser o menor nº inteiro que satisfaça a condição: $(m + r + 1) \leq 2^r$

Para $m = 16bits$ a inequação fica $17 + r \leq 2^r$

Resolvendo com o emulador da calculadora HP Prime[6] temos que $r \geq 4.45$ ou seja $r = 5$



Como os bits de paridade a usar têm de ser potências de 2, teremos:

$$P1 = 2^0 = 1\ (00001_2), \qquad P2 = 2^1 = 2\ (00010_2), \qquad P4 = 2^2 = 4\ (00100_2),$$
$$P8 = 2^3 = 8\ (01000_2) \quad e \quad P16 = 2^4 = 16\ (10000_2)$$

Sendo a mensagem a codificar:

INPUT mensagem:	1	1	0	1	0	0	1	1	0	0	1	1	0	1	0	1

Atendendo a que $m + r = n \Leftrightarrow 16 + 5 = 21$ e dispondo os espaços para os bits de paridade (*check bits* ou *bits de verificação*), numerados da esquerda para a direita, nas posições 1, 2, 4, 8 e 16, e os *m* bits da mensagem nos restantes espaços da palavra de código **1101 0011 0011 0101** de dimensão *n*, temos para começar:

P1	P2	m3	P4	m5	m6	m7	P8	m9	m10	m11	m12	m13	m14	m15	P16	m17	m18	m19	m20	m21
		1		1	0	1		0	0	1	1	0	0	1		1	0	1	0	1
00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110	01111	10000	10001	10010	10011	10100	10101

[7]

Agora temos de considerar que é requerido que a codificação seja par, portanto todas as somas afetas a cada um dos bits de paridade, com o próprio incluído, tem de ser par.

[6] http://www.hp-prime.de/en/category/13-emulator
[7] Notação usada igual à da figura.6, página.206 – Computer Networks, 5th.edition - Tanenbaum

Os cálculos são efetuados como se segue, sempre com a posição do bit de paridade Px associado deixado primariamente em VAZIO, para que após o cálculo do somatório dos bits que lhe são afetos, venha a tomar o valor 0 ou 1, conforme seja ou não necessário adicionar 1 para que o resultado final esteja de acordo com a paridade pretendida, par:

Como P1, a começar por si, é afetado por todos os bits com endereço ímpar na 1ª posição (00001_2), temos:

P1 = VAZIO+m3+m5+m7+m9+m11+m13+m15+m17m+19m+m21

P1 = 8, logo é par, pelo que ***P1 = 0***, pois assim mantém o somatório par.

Realizando operações similares para P2, considerando que é afetado, a começar por si, por todos os bits com endereço ímpar na 2ª posição (00010_2), temos:

P2 = VAZIO+m3+m6+m7+m10+m11+m14+m15+m18+m19 = 5

Logo, ímpar, é necessário que ***P2 = 1*** para tornar o somatório par.

P4, será afeto, a começar por si, por todos os bits com endereço ímpar na 3ª posição (00100_2), e assim:

P4 = VAZIO+m5+m6+m7+m12+m13+m14+m15+m20+m21 = 5, ímpar, então ***P4 = 1***

Seguindo o mesmo raciocínio para P8 (01000_2), e P16 (10000_2), afetos pelos bits até ao limite *n* da dimensão da palavra codificada, teremos:

P8 = VAZIO+m9+m10+m11+m12+m13+m14+m15 = 3, ímpar, então ***P8 = 1***

P16 = VAZIO+m17+m18+m19+m20+m21 = 3, ímpar, então ***P16 = 1***

Hamming (21,16) - deteção e correção de erros (paridade PAR)

INPUT mensagem:	1	1	0	1	0	0	1	1	0	0	1	1	0	1	0	1
	m3	m5	m6	m7	m9	m10	m11	m12	m13	m14	m15	m17	m18	m19	m20	m21

P1	P2	m3	P4	m5	m6	m7	P8	m9	m10	m11	m12	m13	m14	m15	P16	m17	m18	m19	m20	m21
		1		1	0	1		0	0	1	1	0	0	1		1	0	1	0	1
00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110	01111	10000	10001	10010	10011	10100	10101

P1 = VAZIO+m3+m5+m7+m9+m11+m13+m15+m17m+19m+m21 P1 = 8 PAR **P1 = 0**

P1	P2	m3	P4	m5	m6	m7	P8	m9	m10	m11	m12	m13	m14	m15	P16	m17	m18	m19	m20	m21
0		1		1	0	1		0	0	1	1	0	0	1		1	0	1	0	1
00001		00011		00101		00111		01001		01011		01101		01111		10001		10011		10101

P2 = VAZIO+m3+m6+m7+m10+m11+m14+m15+m18+m19 P2 = 5 ÍMPAR **P2 = 1**

P1	P2	m3	P4	m5	m6	m7	P8	m9	m10	m11	m12	m13	m14	m15	P16	m17	m18	m19	m20	m21
0	1	1		1	0	1		0	0	1	1	0	0	1		1	0	1	0	1
	00010	00011			00110	00111			01010	01011			01110	01111			10010	10011		

P4 = VAZIO+m5+m6+m7+m12+m13+m14+m15+m20+m21 P4 = 5 ÍMPAR **P4 = 1**

P1	P2	m3	P4	m5	m6	m7	P8	m9	m10	m11	m12	m13	m14	m15	P16	m17	m18	m19	m20	m21
0	1	1	1	1	0	1		0	0	1	1	0	0	1		1	0	1	0	1
			00100	00101	00110	00111					01100	01101	01110	01111					10100	10101

P8 = VAZIO+m9+m10+m11+m12+m13+m14+m15 P8 = 3 ÍMPAR **P8 = 1**

P1	P2	m3	P4	m5	m6	m7	P8	m9	m10	m11	m12	m13	m14	m15	P16	m17	m18	m19	m20	m21
0	1	1	1	1	0	1	1	0	0	1	1	0	0	1		1	0	1	0	1
							01000	01001	01010	01011	01100	01101	01110	01111						

P16 = VAZIO+m17+m18+m19+m20+m21 P16 = 3 ÍMPAR **P16= 1**

P1	P2	m3	P4	m5	m6	m7	P8	m9	m10	m11	m12	m13	m14	m15	P16	m17	m18	m19	m20	m21
0	1	1	1	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	0	1
															10000	10001	10010	10011	10100	10101

OUTPUT mensagem enviada:																				
0	1	1	1	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	0	1
00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110	01111	10000	10001	10010	10011	10100	10101

Como resultado, a palavra codificada transmitida é: **0 1111 0110 0110 0111 0101**

A codificação de **Hamming** também poderia ser resolvida na <u>forma matricial</u>[8]

Para tal, recorre-se a uma matriz geradora *G(n,m)* a que se multiplica o vetor vertical composto pelos bits a transmitir *M(m),* onde:

n – é composto pelos dados *(mx)* e bits de paridade *(Px)* numerados de cima para baixo, e os bits que são potências de 2 (1, 2, 4, 8, 16…) são bits de paridade, e os restantes (3, 5, 6, 7, 9, 10,…) são preenchidos com os *m* bits de dados que compõem a mensagem.

m – é composto apenas pelos dados *(mx)* numerados da esquerda para a direita

Da multiplicação de *G(n,m)*M(m)* resulta o vetor *T(n)* que é a mensagem a transmitir pretendida, **0 1111 0110 0110 0111 0101** depois de convertidos os bits para binário (mod2)

Matriz geradora: G

	m3	m5	m6	m7	m9	m10	m11	m12	m13	m14	m15	m17	m18	m19	m20	m21
P1	1	1	0	1	1	0	1	0	1	0	1	1	0	1	0	1
P2	1	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
m3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P4	0	1	1	1	0	0	0	1	1	1	1	0	0	0	1	1
m5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m6	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
m7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
P8	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
m9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
m10	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
m11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
m12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
m13	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
m14	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
m15	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P16	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
m17	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
m18	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
m19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
m20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
m21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

G*M=T

	input M			output T	
m3	1		8	0	P1
m5	1		5	1	P2
m6	0		1	1	m3
m7	1		5	1	P4
m9	0		1	1	m5
m10	0		0	0	m6
m11	1		1	1	m7
m12	1		3	1	P8
m13	0		0	0	m9
m14	0	=	0	0	m10
m15	1	mod2 =>	1	1	m11
m17	1		1	1	m12
m18	0		0	0	m13
m19	1		0	0	m14
m20	0		1	1	m15
m21	1		3	1	P16
			1	1	m17
			0	0	m18
			1	1	m19
			0	0	m20
			1	1	m21

[8] Para mais detalhes ver: https://eaulas.usp.br/portal/video?idItem=7727

A mensagem pode depois ser verificada e descodificada no recetor, recorrendo a uma matriz verificadora *H(k,n)* que será multiplicada pela mensagem recebida *T(n),* do que resulta após conversão para binário um vetor de verificação *V(k),* isto é *H(k,n)*T(n)=V(k)*

Se não houver erros na receção, *V(k)* será = 00000_2

Matriz verificadora: H																					
	P1	P2	m3	P4	m5	m6	m7	P8	m9	m10	m11	m12	m13	m14	m15	P16	m17	m18	m19	m20	m21
P1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
P2	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
P4	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
P8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0
P16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

	recebida T	V 54321			
	0	00001		8	V1
	1	00010		6	V2
X	1	00011	=	6	V3
	1	00100		4	V4
	1	00101		4	V5
	0	00110		mod2	
	1	00111		Verificação: V	
	1	01000		0	0	0	0	0
	0	01001		V5	V4	V3	V2	V1
	0	01010						
	1	01011						
	1	01100						
	0	01101						
	0	01110						
	1	01111						
	1	10000						
	1	10001						
	0	10010						
	1	10011						
	0	10100						
	1	10101						

Sendo o resultado da descodificação a mensagem original *M(m)*

Mensagem descodificada sem erros															
m3	m5	m6	m7	m9	m10	m11	m12	m13	m14	m15	m17	m18	m19	m20	m21
1	1	0	1	0	0	1	1	0	0	1	1	0	1	0	1
00011	00101	00110	00111	01001	01010	01011	01100	01101	01110	01111	10001	10010	10011	10100	10101

Se a matriz verificadora *H(k,n)* for multiplicada por um vetor que contenha uma mensagem recebida com erro, o resultado será um vetor *V(k)* com valor diferente de zero, e que indicará a posição do bit errado.

A matriz verificadora *H(k,n)* não será novamente representada pois é a mesma. Vamos apenas multiplicá-la por uma mensagem *T(n)* com erro no 6º bit, para demonstração.

	recebida	V	input	<--- inserir bits para teste de erro
	T	54321	E	
	0	00001		
	1	00010		
X	1	00011		=
	1	00100		
	1	00101		
	1	00110	1	
	1	00111		
	1	01000		
	0	01001		
	0	01010		
	1	01011		
	1	01100		
	0	01101		
	0	01110		
	1	01111		
	1	10000		
	1	10001		
	0	10010		
	1	10011		
	0	10100		
	1	10101		

8	V1
7	V2
7	V3
4	V4
4	V5

mod2

Verificação: V				
0	0	1	1	0
V5	V4	V3	V2	V1

É assinalado um erro no 6º bit da mensagem recebida *T(x),* com endereço dado pelo vetor de verificação *V(k)*, agora diferente de zero, ou seja 00110_2 ou 6_{10}, o que corresponde ao 3º bit *(m6)* da mensagem original *M(m),* pelo que a mensagem descodificada agora terá o 3º bit errado.

Mensagem descodificada contém erros

m3	m5	m6	m7	m9	m10	m11	m12	m13	m14	m15	m17	m18	m19	m20	m21
1	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1
00011	00101	00110	00111	01001	01010	01011	01100	01101	01110	01111	10001	10010	10011	10100	10101

5.a) A um fluxo de bits 11 0101 1111 transmitido pelo método **CRC padrão** corresponde o polinómio $M(x) = 1x^9 + 1x^8 + 0x^7 + 1x^6 + 0x^5 + 1x^4 + 1x^3 + 1x^2 + 1x^1 + 1x^0$

Entre o emissor e recetor é acordada uma chave, ou polinómio gerador para a codificação.

O polinómio gerador fornecido é: $G(x) = 1x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0 = 10011$

Para aplicar o método CRC padrão, serão acrescentados zeros à direita de $M(x)$, tantos quanto o grau do polinómio gerador $G(x)$, que sendo de grau 4, implica que a palavra de dados aumentada seja $x^r M(x) = x^4 M(x) = 11010111110000$.

$x^4 M(x)$ será dividido pelo polinómio gerador $G(x)$, e do resto desta divisão resultará o CRC (checksum), que será então posteriormente concatenado à direita do fluxo de dados, isto é, do polinómio $M(x)$, para então formarem a palavra a transmitir $T(x) = x^r M(x) + CRC$

A aritmética de polinómios é feita em módulo 2, pelo que a adição ou a subtração são operações idênticas ao XOR. A divisão dos polinómios é feita do mesmo modo que me binário, mas as subtrações são feitas em módulo 2, que como referido, equivale a realizar uma operação XOR. Realiza-se então a operação $x^4M(x)/\mathrm{G(x)}$

	M(x)										r zeros				G(x)				
	1	1	0	1	0	1	1	1	1	1	0	0	0	0	1	0	0	1	1
⊕	1	0	0	1	1										1	1	0	0	0
	0	1	0	0	1	1													
	⊕	1	0	0	1	1													
		0	0	0	0	0	1												
		⊕	0	0	0	0	0												
			0	0	0	0	1	1											
			⊕	0	0	0	0	0											
				0	0	0	1	1	1										
				⊕	0	0	0	0	0										
					0	0	1	1	1	1									
					⊕	0	0	0	0	0									
						0	1	1	1	1	0								
						⊕	1	0	0	1	1								
							0	1	1	0	1	0							
							⊕	1	0	0	1	1							
								0	1	0	0	1	0						
								⊕	1	0	0	1	1						
									0	0	0	0	1	0					
									⊕	0	0	0	0	0					
								CRC = Resto		0	0	0	1	0					

A string de bits realmente transmitida $T(x) = x^4M(x) + CRC$ é **11 0101 1111 0010**

5.b) No caso de ser invertido o 3º bit a partir da esquerda de $T(x)$ durante a transmissão, o que chega ao recetor é a string de bits: **11 1101 1111 0010**

Como o polinómio gerador $G(x)$ é partilhado pelos intervenientes, o recetor divide por ele a string de bits recebida, e tem a confirmação de sucesso na transmissão se o resto dessa divisão for zero. Caso contrário, houve erro na transmissão, como se verifica $R = 1110 \neq 0$

					T(x) + E(x)								CRC				G(x)									
	1	1	1	1	0	1	1	1	1	1	0	0	1	0	1	0	0	1	1							
⊕	1	0	0	1	1										1	1	1	0	0	1	0	1	0	0		
	0	1	1	0	1	1																				
	⊕	1	0	0	1	1																				
		0	1	0	0	0	1																			
		⊕	1	0	0	1	1																			
			0	0	0	1	0	1																		
			⊕	0	0	0	0	0																		
				0	0	1	0	1	1																	
				⊕	0	0	0	0	0																	
					0	1	0	1	1	1																
					⊕	1	0	0	1	1																
						0	0	1	0	0	0															
						⊕	0	0	0	0	0															
							0	1	0	0	0	0														
							⊕	1	0	0	1	1														
								0	0	0	1	1	1													
								⊕	0	0	0	0	0													
									0	0	1	1	1	0												
									⊕	0	0	0	0	0												
										Resto!=0	0	1	1	1	0											

Se a receção da string de bits fosse a correta, isto é, **11 0101 1111 0010** o resto seria 0

	1	1	0	1	0	1	1	1	1	1	0	0	1	0	1	0	0	1	1							
⊕	1	0	0	1	1										1	1	0	0	0	0	1	1	1	0		
	0	1	0	0	1	1																				
	⊕	1	0	0	1	1																				
		0	0	0	0	0	1																			
		⊕	0	0	0	0	0																			
			0	0	0	0	1	1																		
			⊕	0	0	0	0	0																		
				0	0	0	1	1	1																	
				⊕	0	0	0	0	0																	
					0	0	1	1	1	1																
					⊕	0	0	0	0	0																
						0	1	1	1	1	0															
						⊕	1	0	0	1	1															
							0	1	1	0	1	0														
							⊕	1	0	0	1	1														
								0	1	0	0	1	1													
								⊕	1	0	0	1	1													
									0	0	0	0	0	0												
									⊕	0	0	0	0	0												
										Resto=0	0	0	0	0	0											

Bibliografia:

- Computer Networks, 5th. Edition, international;
  Tanenbaum & Wetherall; Pearson Education Limited (2014)

- Data Communications, Computer Networks and Open Systems, 4th Edition;
  Fred Halsall; Addison-Wesley Publishing Company (1996)

- Data Communications and Networking 5th Edition;
  Behrouz A. Forouzan; McGraw-Hill (2013)

- Outras referências expostas em notas de rodapé ao longo do texto