**Departamento de Engenharia Eletrónica e**

**Comunicações e de Computadores**

**Licenciatura em Engenharia Informática e de Computadores**



**Comunicações**

**Módulo 2**

**Semestre de Inverno 2018/2019**

**Elaborado por:**

André Fonseca nº 32734

Miguel Lourenço nº 38858

Manuel Dias nº 38866

Índice

[Índice 2](#_Toc533275924)

[Exercício 1 3](#_Toc533275925)

[a) 3](#_Toc533275926)

[b) 6](#_Toc533275927)

[Filtro Passa-Baixo 6](#_Toc533275928)

[Filtro Passa-Banda 7](#_Toc533275929)

[Exercício 2 8](#_Toc533275930)

[a) 8](#_Toc533275931)

[b) 9](#_Toc533275932)

[c) 10](#_Toc533275933)

[Exercício 3 11](#_Toc533275934)

[a) 11](#_Toc533275935)

[b) 12](#_Toc533275936)

[c) 12](#_Toc533275937)

[Exercício 4 14](#_Toc533275938)

[a) 14](#_Toc533275939)

[b) 15](#_Toc533275940)

[c) 16](#_Toc533275941)

# Exercício 1

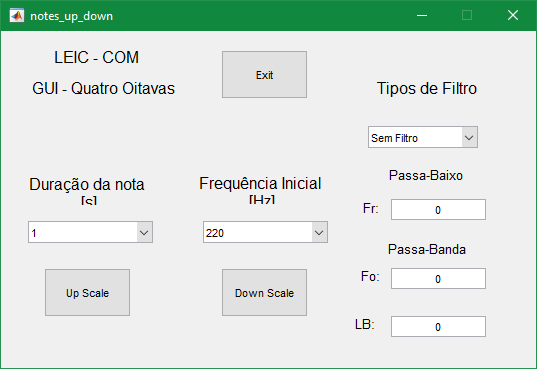
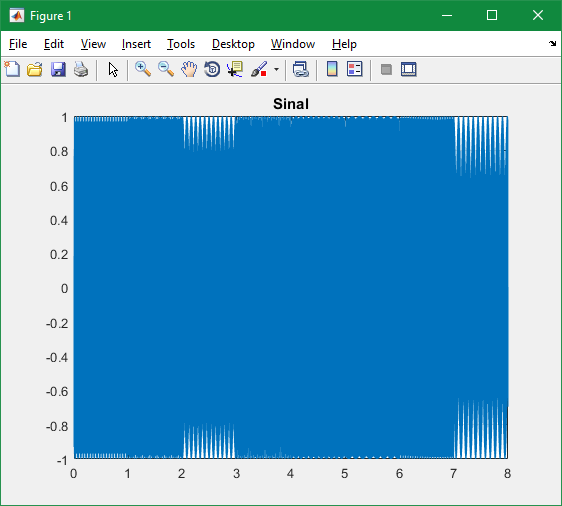


Figura 1 - Figura ilustrativa da UI da aplicação.

## a)

|  |  |  |
| --- | --- | --- |
|  | Sinal 1 | Sinal 2 |
| Duração da nota [s] | 1 | 2 |
| Frequência inicia [Hz] | 220 | 220 |
| Scale | Up | Down |

Tabela 1 - Tabela com os valores iniciais dos 2 sinais.



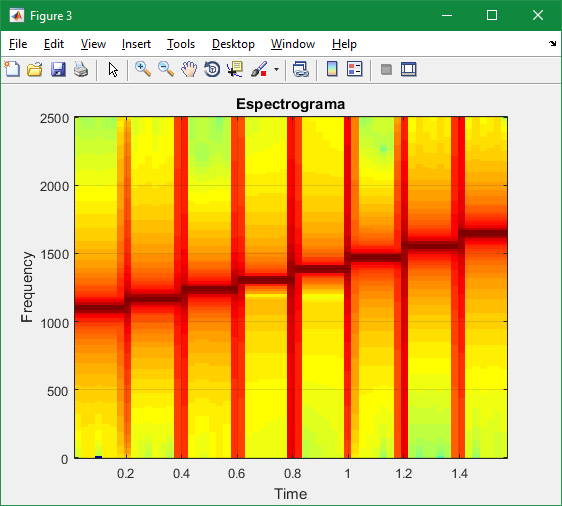
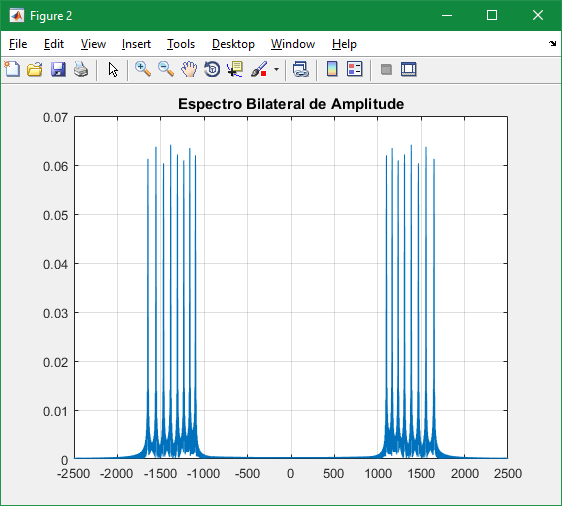


Figura 2 - Sinal 1 com o seu espetro bilateral e respetivo espetrograma.

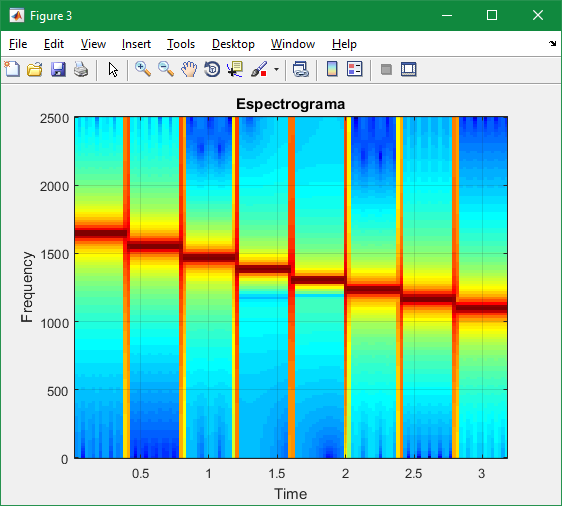
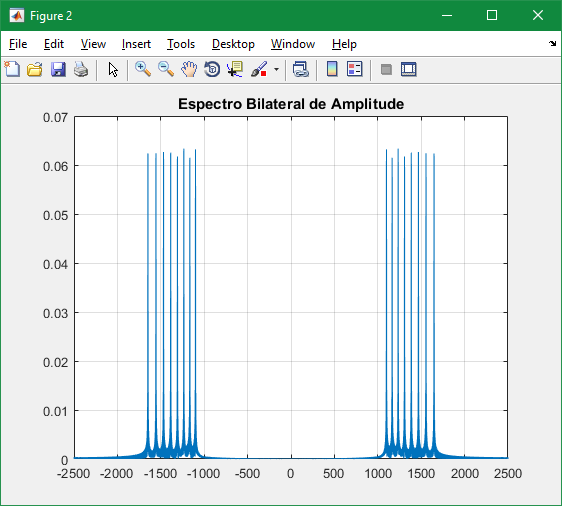
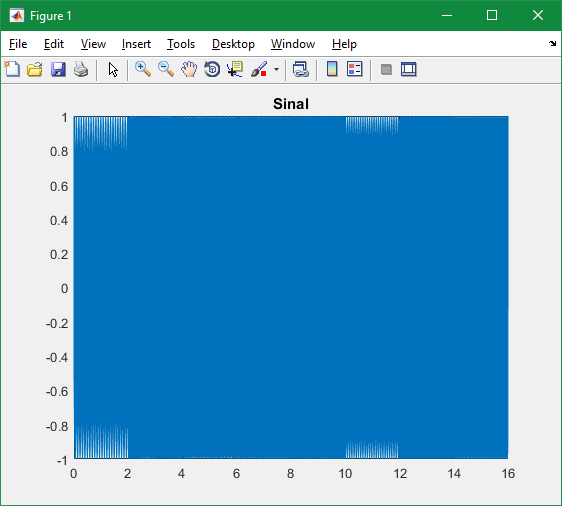


Figura 3 - Sinal 2 com o seu espetro bilateral e respetivo espetrograma.

## b)

### Filtro Passa-Baixo

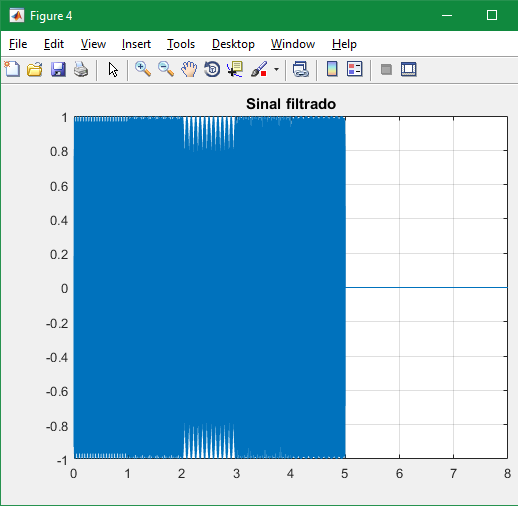
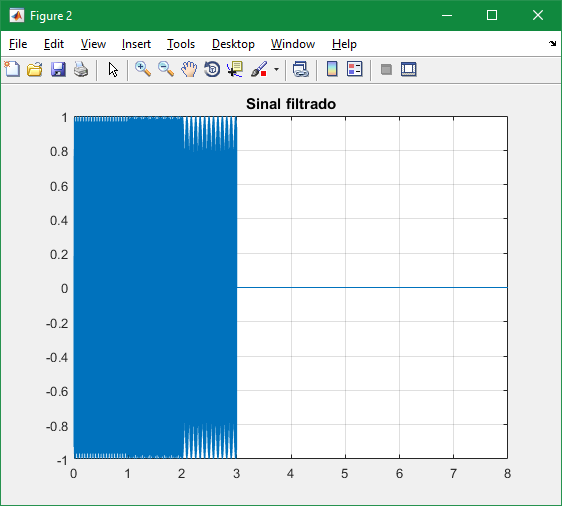
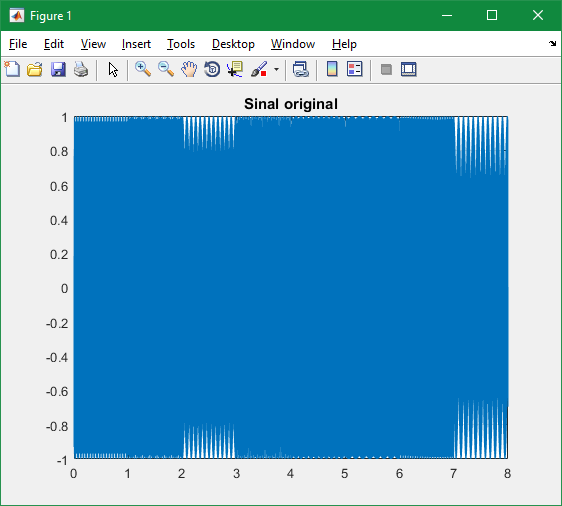


Figura 4 - Resultados experimentais do filtro passa-baixo com frequência de corte em 250 e 280 [Hz].

### Filtro Passa-Banda

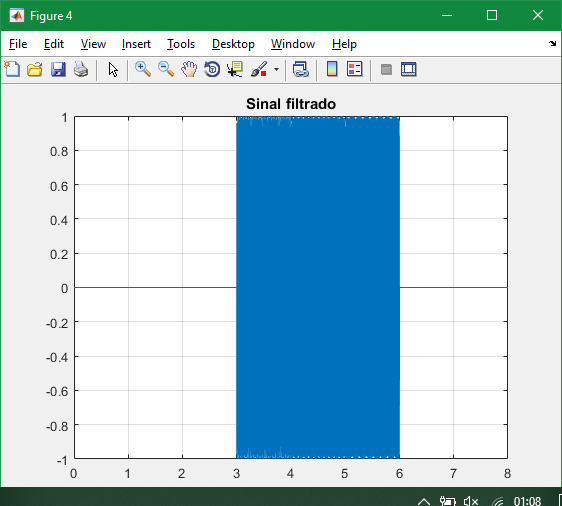
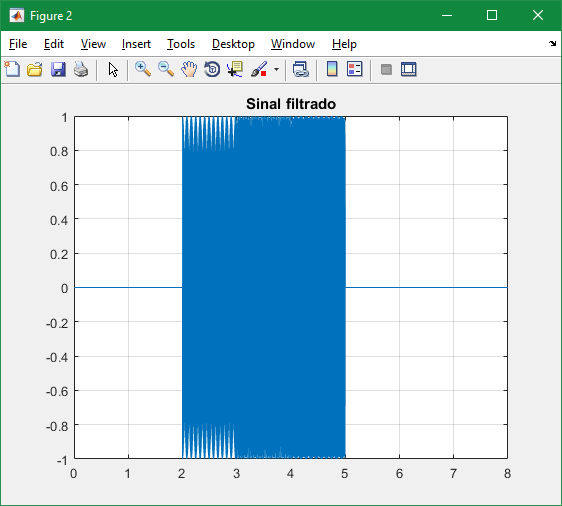
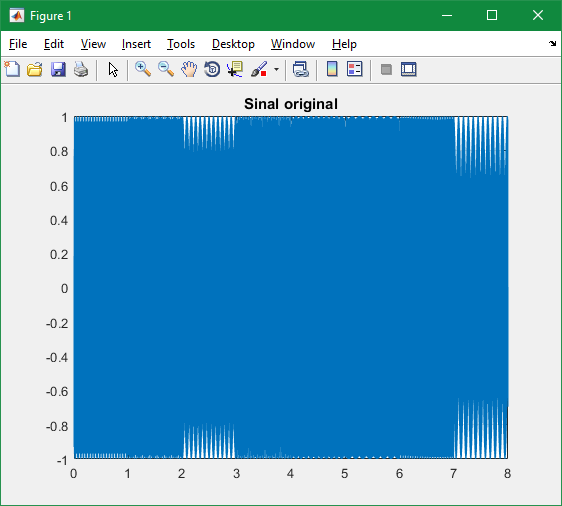


Figura 5 - Resultados experimentais do filtro passa-banda com frequência fundamental de 260 e 280 e largura de banda de 40 [Hz]

# Exercício 2

## a)

O codificador, *Codification*, é uma função com os seguintes parâmetros de entrada:

1. A -> Amplitude dos pulsos [V];
2. Rb -> Ritmo binário [bits/s];
3. L -> nº de bits iguais para inserir um bit de *stuffing*;
4. Fs -> Frequência de amostragem [Hz];
5. Message -> Mensagem para codificar;

E como parâmetros de saída:

1. S -> vetor do sinal;
2. T -> vetor de tempo;
3. messageStuffed -> mensagem com os bits de *stuffing*;

Para implementar esta função primeiro foi necessário calcular o tempo de bit e inserir todos os bits de *stuffing* no vetor de retorno. Com o tempo de bit calculado, foram instanciados os vetores de tempo e de sinal, com este último tudo a 0. Finalmente foi inserido no vetor de sinal os pulsos conforme o vetor da mensagem já com os bits de *stuffing*;

O descodificador, *Descodification*, é uma função com os seguintes parâmetros de entrada:

1. s -> vetor do sinal da mensagem;
2. t -> vetor de tempo;
3. Rb -> Ritmo binário [bit/s];
4. L -> nº de bits iguais para inserir um bit de *stuffing*;
5. Fs -> Frequência de amostragem [Hz];

E como parâmetro de saída, um vetor com a mensagem já descodificada.

Para implementar esta função primeiro foi necessário calcular o tempo de bit. Com este pode-se calcular a energia de bit e é a partir deste que se pode concluir se o bit é 1 ou 0. Também é declarado 2 variáveis auxiliares, value e count, que estão responsáveis por saber qual foi o último bit a ser lido e quantas vezes já apareceu. Se este bit já apareceu L vezes, quer dizer que o próximo bit é um bit de *stuffing* e é ignorado.

## b)

|  |  |  |
| --- | --- | --- |
|  | Sinal 1 | Sinal 2 |
| A | 5 | 3.3 |
| Rb | 1000 | 100 |
| L | 3 | 6 |
| Fs | 8000 | 8000 |
| Msg | [100111100010] | [1001111111111111111100010] |

Tabela 2 - Tabela com os valores dos 2 sinais.

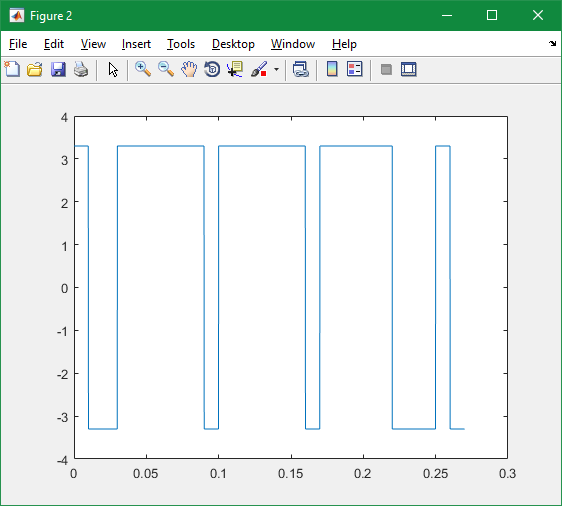
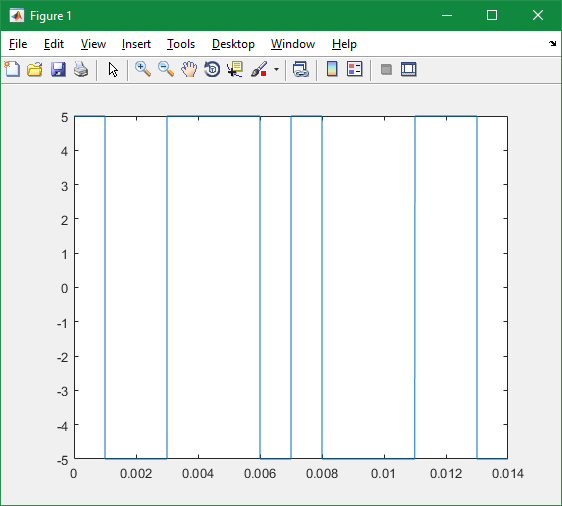


Figura 6 - Sinais resultantes da codificação das 2 sequências binárias.

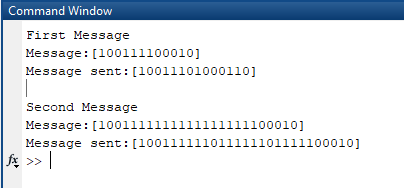


Figura 7 - Output com os valores das mensagens originais e codificadas.

## c)

|  |  |
| --- | --- |
|  | Sinal |
| A | 5 |
| Rb | 1000 |
| L | 3 |
| Fs | 8000 |
| Msg | [1 0 0 1 1 1 0 0 0 1 0] |

Tabela 3 - Tabela com os valores do sinal.

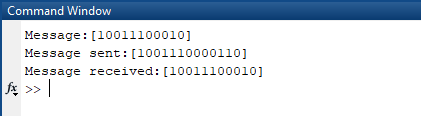


Figura 8 – Output com o resultado da mensagem enviada e recebida depois de codificada.

# Exercício 3

## a)

Para implementar a função *emitterQPSK*, foi decidido usar como parâmetros de entrada:

1. Code -> A sequência de bits a modular;
2. A -> A amplitude do sinal [V];
3. Fs -> A frequência de amostragem [Hz];
4. Fo -> A frequência fundamental [Hz];
5. Rb -> O ritmo binário [bits/s];

E como parâmetros de saída 2 vetores, o vetor de sinal e o vetor de tempo.

A constelação usada está presente na Figura 9. Foi decidido usar esta constelação apenas por ser intuitivo. No quadrante positivo, primeiro quadrante, temos o símbolo 1 1, e no quadrante negativo, terceiro quadrante, temos o símbolo 00. No segundo temo 0 1, porque cos é negativo e sin é positivo, e no quarto quadrante temo 1 0, pois cos é positivo e sin é negativo.

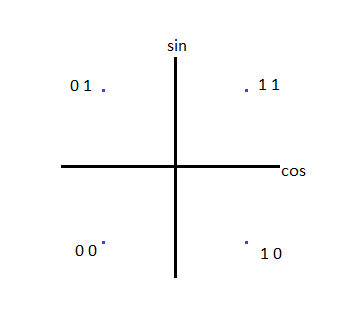


Figura 9 Constelação usada para implementar a modulação QPSK

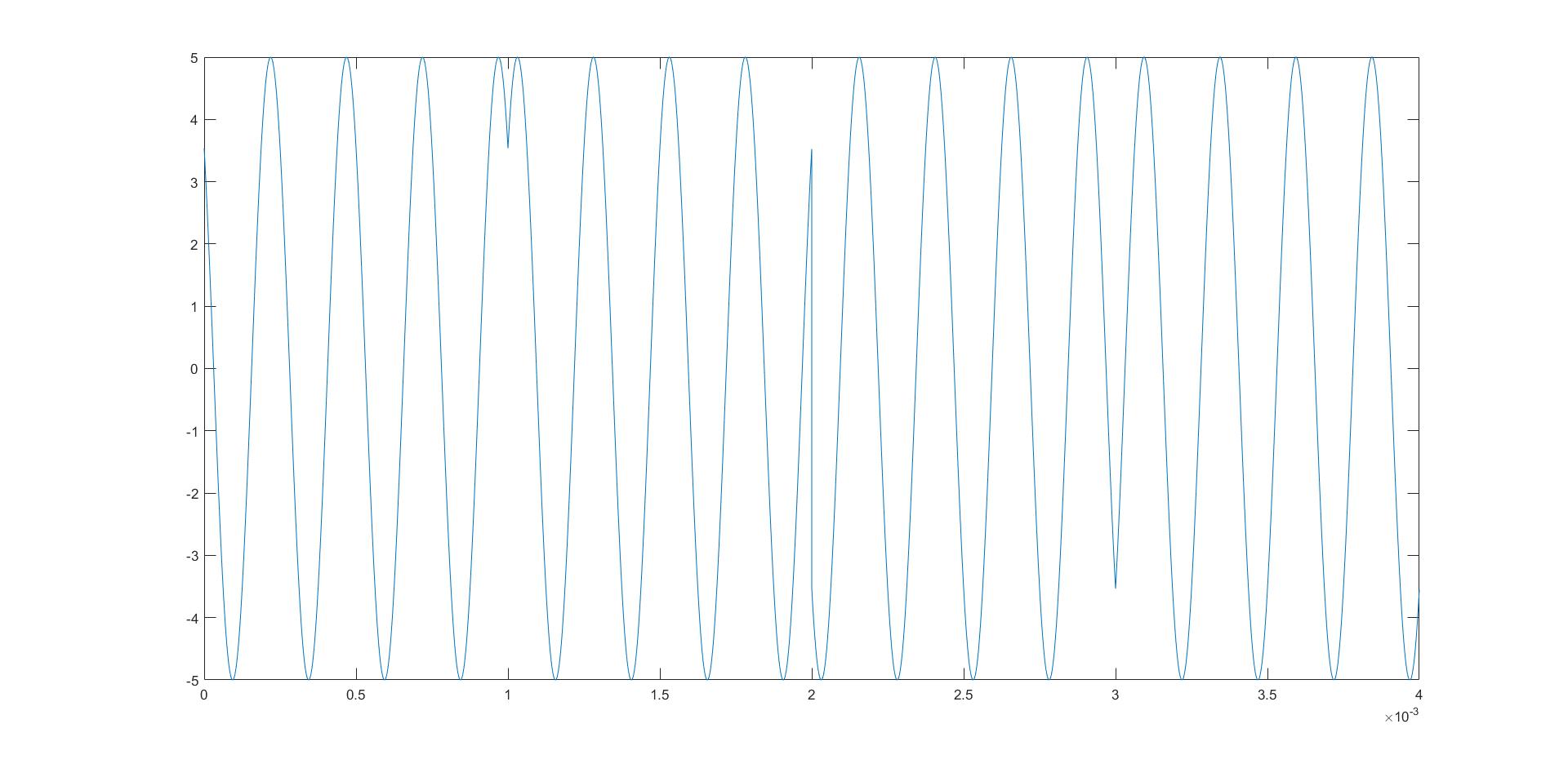


Figura 10 Sinal resultante da codificação da sequência 10110001

## b)

Para implementar função *receiverQPSK*, foi decidido usar como parâmetros de entrada:

1. S -> vetor do sinal da mensagem transmitida;
2. T -> vetor de tempo;
3. Fs -> Frequência de amostragem [Hz];
4. Fo -> Frequência fundamental [Hz];

E como parâmetro de saída a mensagem descodificada.

Para converter do sinal recebido para o digital, usou-se o algoritmo dado nos slides nas aulas. É primeiro calculado o vetor de tempo de 1 símbolo e o vetor de sinal desse símbolo através da frequência de amostragem. Depois gera-se um sinal cos e sin de referência consoante a frequência fundamental e para cada referência multiplica-se o sinal de símbolo. A partir dessa multiplicação, calcula-se a energia de cada um. Por fim compara-se os valores da energia, neste caso, se a energia vinda do sinal de referência cos for maior ou igual a 0, sabemos que o bit mais à esquerda é 1, caso contrário é 0, e faz-se a mesma coisa com o cálculo da energia vinda do sinal de referência sin.

## c)

Para descodificar a mensagem vinda do ficheiro, foi feito a função *decodeQPSK* cujo os parâmetros de entrada são:

1. S -> vetor do sinal codificado;
2. Rb -> Ritmo binário [bit/s];
3. Fs -> Frequência de amostragem [Hz];
4. Fo -> Frequência fundamental [Hz];
5. Eb -> Média da energia de bit [J];

Como parâmetros de saída, a mensagem descodificada num vetor binário.

Para realizar a descodificação da mensagem contida no ficheiro *QPSKstr.mat*, foi realizado um script, *c.m*, em que este chama a função *decodeQPSK* e com o vetor recebido desta função chama a função *bin2char* que converte um vetor de bits num vetor de caracteres.

Para realizar a descodificação foi usado o mesmo método que foi usado na função *receiverQPSK*, a diferença foi obter os vetores de tempo e de sinal de 1 símbolo a partir do tempo de símbolo. Este tempo foi calculado a partir do ritmo binário.

A mensagem descodificada está presente na Figura 11.

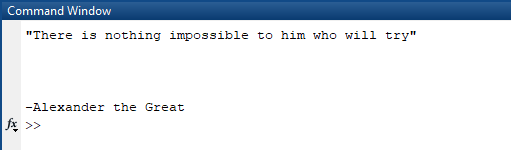


Figura 11 Mensagem codificada do ficheiro QPSKstr.mat

# Exercício 4

## a)

Para desenvolver os códigos Hamming (7,4) e Hamming (8,4), foi necessário criar equações de paridade, para os bits de paridade dos códigos. O número de bits de paridade do código Hamming (7,4) são 3 e o número de bits de paridade do código Hamming (8,4) são 4.

Sendo assim, uma mensagem transmitida usando o código Hamming (7,4) tem o padrão [], no qual os bits são os bits de mensagem e os bits são os bits de paridade. Um bit de paridade é compreendido por uma operação *exclusive OR* entre 3 bits de mensagem. A equação de paridade de cada um destes bits é única e nenhuma delas pode ser igual à outra. Na realização da codificação e da descodificação do código Hamming (7,4) usaram-se as seguintes equações:



Através destas equações pode-se então criar a tabela de síndromas para o código Hamming (7,4).

|  |  |  |
| --- | --- | --- |
| Síndroma | Padrão de Erro | Observações |
| 000 | 0000000 | Ausência de erro |
| 011 | 1000000 | 1.º bit em erro |
| 110 | 0100000 | 2.º bit em erro |
| 101 | 0010000 | 3.º bit em erro |
| 111 | 0001000 | 4.º bit em erro |
| 100 | 0000100 | 5.º bit em erro |
| 010 | 0000010 | 6.º bit em erro |
| 001 | 0000001 | 7.º bit em erro |

Tabela 4 Tabela de Síndromas para o código Hamming (7,4)

Uma mensagem transmitida usando o código Hamming (8,4) tem o padrão []. Na realização da codificação e da descodificação do código Hamming (8,4) usaram-se as seguintes equações:



Através destas equações pode-se então criar a tabela de síndromas para o código Hamming (8,4).

|  |  |  |
| --- | --- | --- |
| Síndroma | Padrão de Erro | Observações |
| 0000 | 00000000 | Ausência de erro |
| 0001 | 00000001 | 8.º bit em erro |
| 0010 | 00000010 | 7.º bit em erro |
| 0011 | - - - - - - - - | Bits em erro ≥ 2 |
| 0100 | 00000100 | 6.º bit em erro |
| 0101 | - - - - - - - - | Bits em erro ≥ 2 |
| 0110 | - - - - - - - - | Bits em erro ≥ 2 |
| 0111 | 00010000 | 4.º bit em erro |
| 1000 | 00001000 | 5.º bit em erro |
| 1001 | - - - - - - - - | Bits em erro ≥ 2 |
| 1010 | - - - - - - - - | Bits em erro ≥ 2 |
| 1011 | 00100000 | 3.º bit em erro |
| 1100 | - - - - - - - - | Bits em erro ≥ 2 |
| 1101 | 01000000 | 2.º bit em erro |
| 1110 | 10000000 | 1.º bit em erro |
| 1111 | - - - - - - - - | Bits em erro ≥ 2 |

Tabela 5 Tabela de Síndromas para o código Hamming (8,4)

Com estas equações de paridade e as tabelas de Síndromas é possível fazer um codificador e um descodificador que deteta e corrige erros de transmissão.

## b)

Através do script *b.m*, obtemos os seguintes resultados para a codificação:

|  |  |  |
| --- | --- | --- |
| Mensagem | Mensagem Codificada  Hamming(7, 4) | Mensagem Codificada  Hamming(8, 4) |
| [1 0 0 1] | [1 0 0 1 1 0 0] | [1 0 0 1 1 0 0 1] |
| [1 1 1 1] | [1 1 1 1 1 1 1] | [1 1 1 1 1 1 1 1] |
| [1 0 1 0] | [1 0 1 0 1 1 0] | [1 0 1 0 0 1 0 1] |
| [0 0 0 0] | [0 0 0 0 0 0 0] | [0 0 0 0 0 0 0 0] |

Tabela 6 Codificação de mensagens usando os códigos Hamming

Usando o mesmo script, obtemos os seguintes resultados para a descodificação sem e com bits em erro, em modo correção:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Mensagem | Hamming(7, 4) | | | | | Hamming(8, 4) | | | | |
| Com Erro | | | Sem Erro | | Com Erro | | | Sem Erro | |
| Mensagem Descodificada | Síndroma | Bit Erro | Mensagem Descodificada | Síndroma | Mensagem Descodificada | Síndroma | Bit Erro | Mensagem Descodificada | Síndroma |
| [1 0 0 1] | [1 0 0 1] | [0 1 1] | 1 | [1 0 0 1] | [0 0 0] | [1 0 0 1] | [1 1 1 0] | 1 | [1 0 0 1] | [0 0 0 0] |
| [1 1 1 1] | [1 1 1 1] | [1 1 0] | 2 | [1 1 1 1] | [0 0 0] | [1 1 1 1] | [1 1 0 1] | 2 | [1 1 1 1] | [0 0 0 0] |
| [1 0 1 0] | [1 0 1 0] | [1 0 1] | 3 | [1 0 1 0] | [0 0 0] | [1 0 1 0] | [1 0 1 1] | 3 | [1 0 1 0] | [0 0 0 0] |
| [0 0 0 0] | [0 0 0 0] | [1 1 1] | 4 | [0 0 0 0] | [0 0 0] | [0 0 0 0] | [0 1 1 1] | 4 | [0 0 0 0] | [0 0 0 0] |

Tabela 7 Descodificação de mensagens usando os códigos Hamming

## c)

No seguimento da execução da função *crc16*, foi decidido ter os seguintes parâmetros de entrada:

1. Message -> mensagem a codificar/descodificar;
2. G -> polinómio gerador do código CRC16;
3. Q -> O grau do polinómio;
4. Operation -> Operação desejada, codificação (valor > 0) ou descodificação (valor ≤ 0);

Como parâmetros de retorno, temos a mensagem codificada/descodificada e a síndroma. Estes são os resultados:

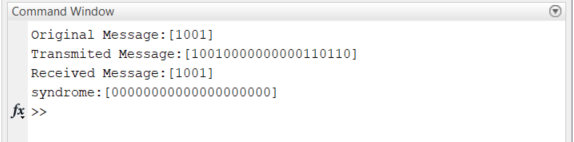


Figura 12 Codificação e Descodificação de uma mensagem sem erros

Na Figura 12 pode-se verificar primeiro a mensagem original e a mensagem que é transmitida depois do cálculo dos bits de paridade da codificação com CRC16. Depois pode-se verificar a mensagem recebida e a síndroma. Como a síndroma é 0 então pode-se concluir que não houve erros na transmissão e a mensagem recebida é a mensagem que o emissor quis enviar.

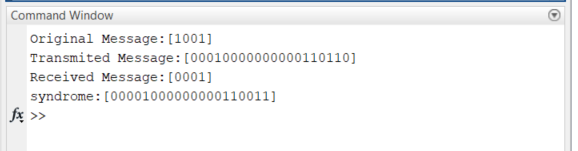


Figura 13 Codificação e Descodificação de uma mensagem com erros

Na Figura 13, pode-se verificar o mesmo processo que na Figura 12, no entanto a transmissão foi feita com um erro no primeiro bit da mensagem. Quando é feito a descodificação da mensagem recebida, pela síndroma dá para verificar que de facto houve um erro na transmissão, pois esta é diferente de 0, e é preciso retransmitir a mensagem.