

**Área Departamental de Engenharia de Eletrónica e**

**Telecomunicações e de Computadores**

**2º Módulo**

|  |  |
| --- | --- |
|  | **Comunicações Digitais** |
|  |  |

**Estudo e aplicação de conceitos fundamentais sobre SCD, cifra, técninas de deteção de erros e codificação de canal.**

|  |  |
| --- | --- |
| Autores: 48285 | Aureliu Iurcu |
| 48349 | José Silva |

**Licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2021/2022**

04/07/2022

# Introdução

O foco principal deste módulo foi o desenvolvimente de programas e aplicações nas linguagens *Python* e *C* a fim de aprofundarmos os estudos e aplicar os conceitos fundamentais sobre os *Sistemas de Comunicação Digital*, teoria de informação, técninas de deteção de erros,técnicas de codificação de canal e aplicação de cifra.

**Desenvolvimento**

**- Exercício 1**

Neste exercício foram aprofundadas na práctica a aplicação da cifra em ficheiros de texto e de imagem. O foco apareceu na implementação das cifras de *Caeser* e *Vernam.*

A cifra de *Caeser* usa a técnica de deslocamente de um certo número de caractéres positivamente ou negativamente para a codificação da informação e o seu deslocamente negado para a descodificação.

A cifra de *Vernam* aplica a técnica do *XOR* com uma certa chave aleatória para a codificação dos bits e aplica exatamente a mesma técnica para a descodificação da mesma, mas com o pormenor, de a chave ter de ser necessariamente a mesma que foi usada para a codificação da informação.

Para a verificação do funcionamente correto da aplicação de cada uma das cifras foram realizados testes onde os mesmos verificam a entropia de dois ficheiros e criam o seus histogramas.

**1 º Teste ( ficheiros originais)**:Chart, histogram

Description automatically generated

**Chart, histogram

Description automatically generated**

Figura - Histograma "alice29.txt" original

Figura - Histograma "a.txt" original

Entropia do ficheiro “a.txt” -> 5.00507

Entropia do ficheiro “alice29.txt” -> 4.567666

**2 º Teste ( ficheiros cifrados com *Ceaser*)**:

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

Figura - Histograma "alice29.txt" cifrado com Ceaser

Figura - Histograma "a.txt" cifrado com *Ceaser*

Entropia do ficheiro “a\_ciphered\_ceaser.txt” -> 4.959446

Entropia do ficheiro “alice29\_ciphered\_ceaser.txt” -> 4.512862

Chart, bar chart, histogram

Description automatically generated**3 º Teste ( ficheiros decifrados com *Ceaser*)**:

Chart, histogram

Description automatically generated

Figura - Histograma "alice29.txt" decifrado com Ceaser

Figura - Histograma "a.txt" decifrado com *Ceaser*

Entropia do ficheiro “a\_deciphered\_ceaser.txt” -> 5.00507

Entropia do ficheiro “alice29\_dedciphered\_ceaser.txt” -> 4.567666

**2 º Teste ( ficheiros cifrados com *Vernam*)**:

Chart, bar chart

Description automatically generatedChart, histogram

Description automatically generated

Figura - Histograma "alice29.txt" cifrado com *Vernam*

Figura - Histograma "a.txt" cifrado com *Vernam*

Entropia do ficheiro “a\_ciphered\_vernam.txt” -> 6.269962

Entropia do ficheiro “alice29\_ciphered\_ vernam.txt” -> 7.998646

Chart, histogram

Description automatically generatedChart, bar chart, histogram

Description automatically generated**3 º Teste ( ficheiros decifrados com *Vernam*)**:

Figura - Histograma "alice29.txt" decifrado com Vernam

Figura - Histograma "a.txt" decifrado com *Vernam*

Entropia do ficheiro “a\_ciphered\_vernam.txt” -> 5.00507

Entropia do ficheiro “alice29\_ciphered\_ vernam.txt” -> 4.567666

Através das figuras anteriores e as entropias de cada ficheiro em causa podemos observar alguns aspetos importantes e específicos de cada técnina de cifragem:

1. Todos os ficheiros decifrados por qualquer uma das técnicas irão coincidir com o seu ficheiro original, tanto em termos de histograma como em termos de entropia o que leva a concluir que não existiu nenhum erro no processo
2. Aplicando a cifra de *Ceaser* na codificação de qualquer um dos ficheiros, o seu histograma vai se manter igual no seu eixo das ordenadas (occorências) enquanto que no eixo das abcissas (caractér) irá mudar pois todos os carácteres sofreram um deslocamento.
3. Aplicando a cifra de *Vernam* na codificação o histograma do ficheiro irá mudar completamente criando um histograma muito mais equilibrado, sendo isso explicado pelo *XOR* feito dos bits de dados com a chave gerada aleatóriamente, o memso acontece com a entropia que é explicada pelo memso fator. Na aplicação dessa cifra garantimos que os caractéres que irão ser codificados irão ter aproximadamente a mesma ocorrência.

Na análise dos exemplos anteriores podemos concluir que é possível ter segurança perfeita num cenário em que é usado a cifra de *Vernami* e a chave gerada na codificação do memso seja enviada por um canal seguro.

**- Exercício 2**

Neste exercício foram aprofundadas técnicas de deteção de erros, que na práctica resumiu-se no estudo e aplicação do *CRC* (*Cyclic Redundancy Check*).

Para o aprendizado do mesmo foram realizados os métodos “*crc\_file\_compute*” e “*crc\_file\_check*”, onde o “*crc\_file\_compute*” irá ser utilizado para a criação do cabeçalho do ficheiro com o seu respetivo *CRC* calculado, e a função “*crc\_file\_check*” que irá fazer a verificação de integridade do ficherio usando para este efeito o *CRC* recebido no cabeçalho do ficheiro.

Para ser testado o correto funcionamento das funções foi realizada uma função auxiliar que injeta uma certa quantidade de erros no ficheiro que simulamente passou pelo emissor, canal e que no momento se encontra nas mãos do recetor.

Dessa forma foi realizada uma função *main* de teste que cria os ficheiros com a taxas de erros de (0%, 0.01%, 0.1%, 0.5%, 1% e 5%).

Usando um *CRC32* conseguimos obervar os seguintes resultados

Text

Description automatically generated

Figura - *CRC* Test

Analisando a figura podemos concluir que no uso de um *CRC32* consegue detetar erros em qualquer que seja a percentagem de erros presentes no ficheiro.

**- Exercício 3**

O objetivo do exercício em causa foi o entendimento e a correta implementação dos métodos de codificação de canal baseado nas técnincas de *NRZU* (*Non return to zero unipolar*), e *PSK* (*Phase Shift Keying*).

Para se conseguir alcançar o objetivo foram implementadas as funções “*NRZU\_Coder*”, “*PSK\_Modulator*”, “*NRZU\_Decoder*” e “*PSK\_Demodulator*”.

Todas as funções mencinadas anteriormente têm o foco da transformação de sinal digital em sinal analógico e vice\_versa.

Para tal temos as fuções “*NRZU\_Coder*” e “*NRZU\_Decoder*” que tranformam o sinal digital em pulsos quadrados e as funções “*PSK\_Modulator*” e “*PSK\_Demodulator*” que passam o sinal digital para ondas sinusoidais. Sendo as ondas quadradas ou sinusoidais irão ser posteriormente da mesma forma enviados pelo canal.

Para o teste dos métodos mencionados anteriormente foram criadas duas funções auxiliares “*rect\_pulse\_presentation*” e “*sinusoidal\_pulse\_presentation*” que apresentam uma sequência de bits usando o formato *NRZU* e *PSK* respetivamente, as funções em causa também tem parâmetros que especificam a variância da amplitude e o ruído associado ao canal que ajudam a simular um certo canal de transmissão de dados.

No teste realizado foi usada a sequência binária b’10110001’. Os resultados simulando um canal perfeito, com com 75 amostras por bit com amplitude de 5V para o *NRZU* e amplitude máxima e frequência de 2V para a onda *PSK* estão apresentados nas figuras abaixo.

Chart, histogram

Description automatically generated

Chart, bar chart, histogram

Description automatically generated

Figura - *NRZU*

Figura - *PSK*

Foram também realizados testes que simulam canal ruidoso e com variância na amplitude.

Os mesmos estão apresentados a seguir.

Chart, histogram

Description automatically generated**Teste 1 (alpha = 1 && n(t) in -0.8..0.8):**

Chart

Description automatically generated

Figura - *PSK* with noise

Figura - *NRZU* with noise

A screenshot of a computer

Description automatically generated with medium confidence

Chart, histogram

Description automatically generated**Teste 2 (alpha < 1 && n(t) = 0):**

Chart, histogram

Description automatically generated

Figura - *NRZU* with amplitude variance and no noise

A screenshot of a computer

Description automatically generated with medium confidence

Figura - *PSK* with amplitude variance and no noise

Através dos exemplos apresentados anteriormente podes chegar as seguintes conclusões:

1. Caso a variância ultrapasse a metade da amplitude no sinal *NRZU* irá existir uma troca de 50% dos bits uma vez que apenas os bits iguais a 0 serão descodificados com sucesso enqunto que no sinal *PSK* a chance de erra um bit caso a variância aumenta não é tão grande desde a mesma não seja grande demais.
2. Em termos do ruído podemos concluir que quanto maior o ruído maior a chance de um bit ser mal descoficado, aplicando-se isto para qualquer que seja o canal de transmissão.

**- Exercício 4**

Neste exercício o pricipal objetivo foi aprender a criar um *SCD* na práctica onde consigamos detetar o emissor, canal e o recetor.

O exercício consiste na criação de uma conexão entre dois dispositiovos, em que no caso específico seria entre um *Arduino* e um *PC* em que o *Arduino* é o emissor que estaria conectado ao *PC,* que é o nosso emissor, através de um cabo *USB* que seria o nosso canal.

 A conexão física está apresentada na figura abaixo

Figura - Conexão física entre o *Arduino* e o *PC*

Para conseguirmos enviar dados do *Arduino* para o *PC* foi usado o IDE *Arduino* para o controlo da informação presente no mesmoe a linguagem *Python* para o controlo da informação recebida no *PC.*

O processo foi dividido em 3 fases:

1. Criação de um programa em *Arduino* que envia uma sequência de 3 Strings (“Begin”, “Processing data...” e “Finished”) com um delay de 1s entre o envio de cada uma delas.
2. Configuração do canal de transmissão do *Arduino*

Graphical user interface, text, application

Description automatically generated

Figura - Conexão ao canal de transmissão de dados *COM3*

1. Criação de um programa usando a linguagem *Python* e a biblioteca *serial*. O programa em *Python* tem como objetivo ler os dados transmitidos pelo *Arduino* e excrever os mesmos na consola. A biblioteca *serial* é usada para possibilitar a conexão ao *Arduino* através do canal acima mencionado

Graphical user interface, text, application, email

Description automatically generatedNas figuras a seguir demostrasse o correto funcionamento que comprova a conexão entre os dois dispositivos .

Figura - Texto apresentado na consola logo após o inicio do programa

Graphical user interface, text

Description automatically generated with medium confidence

Figura - Texto apresentado na consola 1s após o envio de "Begin"

Graphical user interface, text, application

Description automatically generated

Figura - Texto apresentado na consola 1s após o envio de "Processing data..."

**Conclusão**

Durante a realização desse módulo, fomos capazes de aplicar conceitos fundamentais sobre SCD, cifra, codificação de canal, implementação de códigos de controlo de erros, desempenho de sistemas de transmissão digital, ao nível físico com código de linha e modulação digital, sem esquecer a manipulação de dados e a interação com a plataforma Arduíno. Acreditamos que após a realização desse módulo somos capazes de selecionar e aplicar códigos detetores e corretores de erros; reconhecer, identificar e aplicar as principais técnicas de codificação de dados com e sem perda, de acordo a matéria lecionada; e projetar e realizar sistemas que incluam comunicação digital entre dispositivos. Sem dúvidas que a resolução dos exercícios deste módulo serviu para a consolidação da segunda parte da matéria, que agora, temos muito maior domínio. Não obstante, não nos foi possível cumprir o desafio de realizar todos os exercícios propostos pelo enunciado, quer seja pela necessidade da utilização de conteúdos que não chegamos a implementar no módulo anterior, ou por não termos conseguido resolvê-los a tempo da data da entrega. Procuramos adotar as melhores soluções para as questões contidas no módulo em questão, apesar de nem todas serem as melhores do ponto de vista das complexidades (espaciais e temporais). Em suma, foi um módulo bastante desafiador, e fomos capazes de cumprir a maioria dos requisitos impostos, aplicando devidamente os conceitos aprendidos nas aulas