# Codificação/Decodificação

**Maria Eduarda Casanova**
**Bruno Selau**

## Introdução

Este relatório descreve a alternativa dada ao problema proposto na disciplina de Introdução à Redes de Computadores. A questão consiste em, dado um número em hexadecimal, codifica-lo utilizando as seguinte técnicas: NRZ-I, Manchester Diferencial, HDB3, 8B6T, 6B/8B. E dado o sinal codificado, decodificá-lo para hexadecimal. É apresentada uma solução de codificação e decodificação para cada técnica e apresentados exemplos de execução.

## Implementação

- ### NRZI

#### Codificação

A codificação implementada para a técnica NRZI começa a partir da passagem de parâmetro de um número hexadecimal, o script principal já passa como parâmetro para função de encode do NRZI um número binário. Após isso o primeiro sinal codificado é informado a partir do primeiro número binário, caso seja 1 será “+”, caso seja zero será “-”. A partir disso, sempre que informado um dígito zero, o sinal repetirá o sinal antecessor e nos casos de dígito 1, o sinal será invertido.

#### Decodificação

A decodificação implementada para a técnica NRZI começa a partir da passagem de parâmetro de uma sequência de sinais, seguindo de forma muito parecida a codificação dessa mesma técnica. Começando a partir do primeiro sinal, caso seja “+” será 1, caso seja “-” será 0. A partir disso, sempre que um sinal se repetir, o dígito correspondente será 0, e caso o novo sinal seja diferente o anterior teremos 1 como dígito.

```
ada@MacBook-Pro-de-Maria-3 T1 % python3 encode.py nrzi 616263
-+-----++-++++---+----+-
ada@MacBook-Pro-de-Maria-3 T1 % python3 decode.py nrzi -+-----++-++++---+----+-
0x616263
ada@MacBook-Pro-de-Maria-3 T1 % python3 encode.py nrzi 726564657331
-+-+++---+---++--+---++++-+++--++-+---+---+----
ada@MacBook-Pro-de-Maria-3 T1 % python3 decode.py nrzi -+-+++---+---++--+---++++-+++--++-+---+---+----+
0x726564657331
```

- ### Manchester Diferencial

#### Codificação

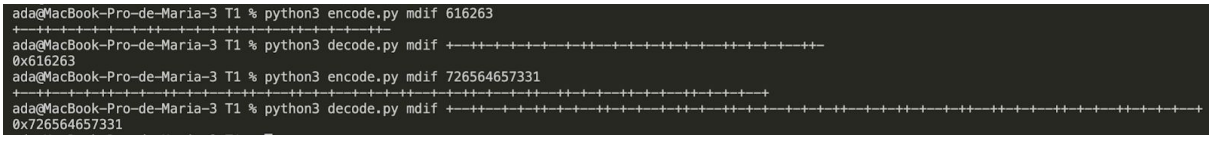
A codificação implementada para a técnica Manchester Diferencial começa a partir da passagem de parâmetro de um número hexadecimal, o script principal já passa como parâmetro para função de encode do Manchester Diferencial um número binário.

Após isso o primeiro sinal codificado é informado a partir do primeiro número binário, caso seja 1 será “-+”, caso seja zero será “+-”. A partir disso, sempre que informado um dígito zero, o sinal inverterá o sinal antecessor e será acrescido de seu sinal oposto. Nos casos de dígito 1, o sinal manterá o sinal antecessor e será acrescido do seu oposto.

### Decodificação

A decodificação implementada para a técnica Manchester Diferencial começa a partir da passagem de parâmetro de uma sequência de sinais, seguindo de forma muito parecida a codificação dessa mesma técnica.

Começando a partir dos primeiros dois sinais, caso seja “+-” será 0, caso seja “-+” será 0. A partir disso, a cada dois sinais sempre que um sinal se repetir, o dígito correspondente será 1, e caso o novo sinal seja diferente o anterior teremos 0 como dígito.




- HDB3

### Codificação

A codificação implementada para a técnica HDB3 utiliza um contador de zeros para identificar bit de verificação e sinalização, além de guardar sempre qual foi o último bit de informação. Então ao encontrar um bit 1, deve-se zerar o contador de zeros e escrever um sinal inverso ao salvo como último bit de informação, e com isso atualizar qual foi o último bit de informação. Toda vez que um zero for encontrado e o contador for menor que três, apenas é necessário incrementar o contador. Caso seja encontrado bit 0 e o contador for igual a 3, deve-se modificar o primeiro e o último sinal da sequência de zeros para o sinal inverso do último bit de informação, e então deve-se atualizar o bit de informação.

### Decodificação

A decodificação implementada para a técnica HDB3 preenche primeiramente 0 nos índices que o sinal tem 0, e 1 nos índices que o sinal tem “+” ou “-”. Após isso ocorre o tratamento desses bits que foram preenchidos com 1, logo ao encontrar um 0, apenas somamos o contador de zeros.

Caso seja encontrado um bit 1 temos que verificar o seguinte:

- Se passamos por dois zeros, e o sinal que gerou esse 1 é igual ao sinal que gerou o último 1, logo, deveremos mudar esses dois bits para zero;
- Se passamos por 3 zeros, e o sinal que gerou esse 1 é igual ao sinal que gerou o último 1, logo, deveremos mudar esse bit para zero;

```
ada@MacBook-Pro-de-Maria-3 T1 % python3 encode.py hdb3 616263
0+-+00+-0+-000+00-+000-+
ada@MacBook-Pro-de-Maria-3 T1 % python3 decode.py hdb3 0+-+00+-0+-000+00-+000-+
0x616263
ada@MacBook-Pro-de-Maria-3 T1 % python3 encode.py hdb3 726564657331
0+-+00-00+-00+0-0+-00+000-+00-0+0-+-00+-00+-000+
ada@MacBook-Pro-de-Maria-3 T1 % python3 decode.py hdb3 0+-+00-00+-00+0-0+-00+000-+00-0+0-+-00+-00+-000+
0x726564657331
```

- 8B6T

## Codificação

A codificação implementada para a técnica 8B6T utiliza inicialmente um dicionário que representa a tabela referente a essa técnica. Com isso dividiremos o binário informado em intervalos, para cada intervalo é necessário verificar se a disparidade dos intervalo anterior aumenta após a inserção deste novo intervalo. Se a disparidade dos intervalos anteriores não cresce, então devemos apenas informar intervalo de sinais disponibilizado pela tabela. Caso esta disparidade cresça, deveremos pegar este os sinais disponibilizados pela tabela e inverter todos os sinais diferentes de 0.

## Decodificação

A decodificação implementada para a técnica 8B6T utiliza inicialmente um dicionário que representa a tabela referente a essa técnica. Com isso dividiremos o binário informado em intervalos, para cada intervalo é necessário verificar este intervalo está presente na tabela de dessa técnica. Caso o intervalo esteja presente basta procurar na tabela os valores correspondentes, caso contrário deveremos inverter todos os sinais diferentes de 0.

```
ada@MacBook-Pro-de-Maria-3 T1 % python3 encode.py 8b6t 616263
+0+-00-0-0+0+0+00-
ada@MacBook-Pro-de-Maria-3 T1 % python3 decode.py 8b6t +0+-00-0-0+0+0+00-
616263
ada@MacBook-Pro-de-Maria-3 T1 % python3 encode.py 8b6t 726564657331
000-++--0+000++00---0+00000+000+--+0
ada@MacBook-Pro-de-Maria-3 T1 % python3 decode.py 8b6t 000-++--0+000++00---0+00000+000+--+0
726564657331
```

- 6B/8B

## Codificação

A codificação implementada para a técnica 6B8B começa a partir da passagem de parâmetro de um número hexadecimal, o script principal já passa como parâmetro para função de encode do 6B8B um número binário.

Após isso o binário é dividido em partes de tamanho 6, para cada uma dessas partes é calculada a disparidade do binário, isto é, a soma das ocorrências do bit 1 menos a soma das ocorrências do bit 0.

Caso a disparidade seja igual a 0, é acrescentado um prefixo de "10" ao binário. Caso a disparidade seja igual a 2, é acrescentado um prefixo de "00" ao binário. Caso a disparidade seja igual a -2, é acrescentado um prefixo de "11" ao binário. Se a disparidade não se encaixar a nenhum dos casos citados acima, o binário é substituído pelo seu equivalente em uma tabela pré estabelecida.

Por fim a cadeia binário resultado é transformada em sinal utilizando a técnica NRZ-I.

## Decodificação

A decodificação implementada para a técnica 6B8B começa a partir da passagem de parâmetro de uma sequência de sinais, seguindo de forma muito parecida a codificação dessa mesma técnica.

Começando a partir da separação em partes de tamanho 8, para cada uma dessas partes é verificado se elas coincidem com os seguintes casos:

- Primeiros dois bit iguais a "10" e últimos seis com paridade igual a 0.
- Primeiros dois bit iguais a "11" e últimos seis com paridade igual a -2.
- Primeiros dois bit iguais a "10" e últimos seis com paridade igual a 2.

Se coincidir com um desses casos o binário de resultante é acrescido com os seis últimos bits. Senão o binário é substituído pelo seu equivalente em uma tabela pré estabelecida.

Por fim o binário resultante é transformado em hexadecimal.