Simulador de Codificação de Canal

Dênis Ivan Lenz, Éwerton Piccinini Spezia e Felipe Augusto Hertzer

Comunicação de Dados - Ciências da Computação e Engenharia da Computação
Universidade de Santa Cruz do Sul (UNISC)

96815-900 - Santa Cruz do Sul - RS - Brasil

d.denis.il@gmail.com, ewertonspezia@mx2.unisc.br,
felipeaugustohertzer@gmail.com

Introdução. Este artigo descreve todo o conceito de funcionamento de um Simulador de Codificação de Canal. É estudado os conceitos da camada física do MR-OSI e aplicado e demonstrado técnicas de codificações como, NRZ-L, NRZI, BIPOLAR-AMI, PSEUDOTERNARY, MANCHESTER e DIFERENTIAL MANCHESTER, apresentando suas vantagens e desvantagens com ilustrações e relatos sobre trechos de códigos implementados.

Para o desenvolvimento do Simulador foi utilizado para gerar códigos a partir do número binário a linguagem de programação php e no processo de montagem e demonstração dos gráficos foi utilizada a biblioteca de gráficos ChartJs de javacript.

1.1 NRZ-L (Non return to zero level)

- Usa dois níveis de sinal para representar 0 e 1 (codificação absoluta).
- O Nível do sinal permanece constante durante o intervalo de um bit.

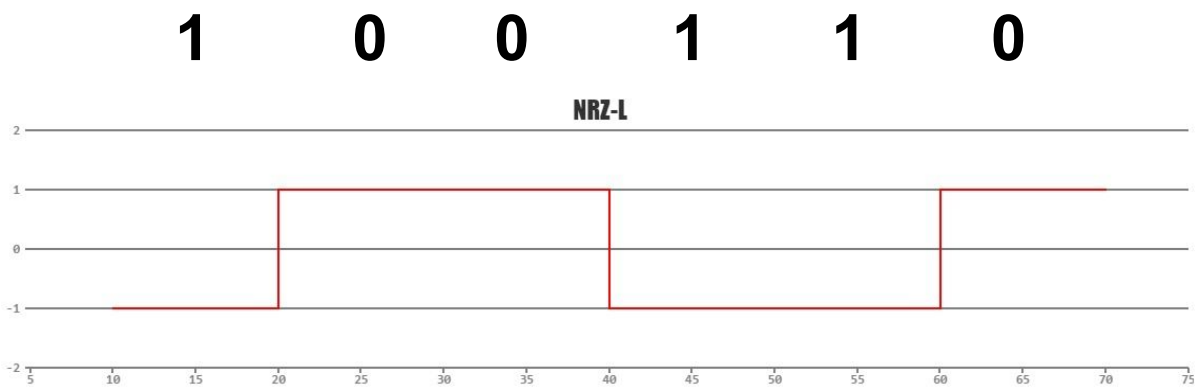
1.2 Vantagens

- Fácil de implementar.
- Boa eficiência espectral.

1.3 Desvantagens

- Não é possível fazer acoplamento.
- Não permite auto-sincronização.

1.4 Ilustração do exemplo 100110



1.5 Trecho de código

```php
public function nrzl($array_binary){
    $clock = $array_bit = 0;
    $array_returno = array();
    foreach($array_binary as $bit){
        $clock = $clock + 10;
        if($bit == 1)
            $array_bit = -1;
        else
            $array_bit = 1;
        $array_returno[$clock] = $array_bit;
    }
    $array_returno[$clock+10] = $array_bit;
    return $array_returno;
}
```

Função que recebe uma sequência binária e faz uma verificação bit a bit para gera o sinal codificado NRZ-L.

2.1 NRZI (Non return to zero inverted)

- Inverte o sinal na presença de bit com valor 1.
- Sinal codificado pela comparação com pulsos adjacentes.
- Na presença de ruído, é mais fácil detectar uma transição do que comparar um valor absoluto com limiar (definira se é 0 ou 1).

2.2 Vantagens

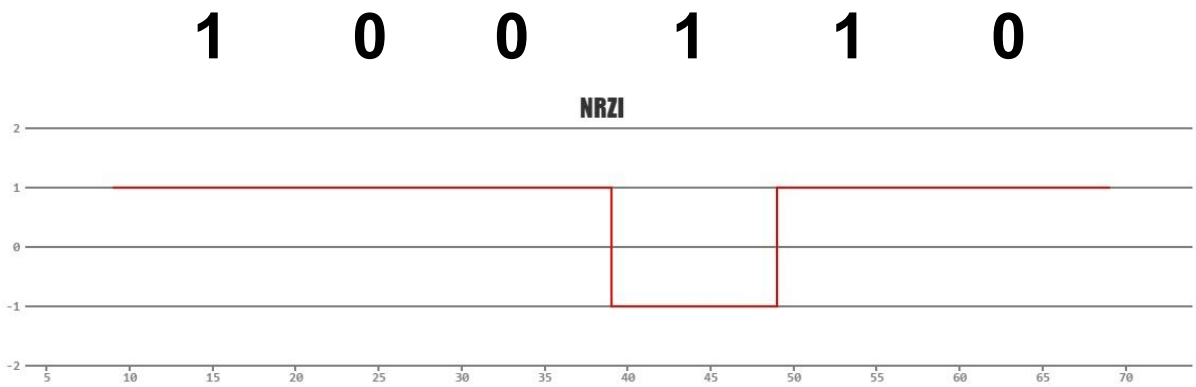
- É simples.

- Utiliza eficientemente a largura de banda.
- Grande parte da energia está concentrada nas componentes centrais.

2.3 Desvantagens

- Falta de mecanismo de sincronização.
- Não detecta erros, pois é um valor absoluto.

2.4 Ilustração do exemplo 100110



2.5 Trecho de código

```php
private function nrzi($array_binary){
    $clock = $array_bit = 0;
    $array_returno = array();
    foreach($array_binary as $bit){
        $clock = $clock + 10;
        if($bit == 1)
            $array_bit = ($array_bit == 1 ? -1 : 1);
        //else
            //$array_bit = $array_bit;
        $array_returno[$clock] = $array_bit;
    }
    $array_returno[$clock+10] = $array_bit;
    return $array_returno;
}
```

Função que recebe uma sequência binária e faz uma verificação bit a bit para gera o sinal codificado NRZI.

3.1 BIPOLAR-AMI (Alternate Mark Inversion)

- 0 é representado pela ausência de sinal.

- 1 é representado por pulso positivo ou negativo (alternados).
- AMI RZ usado no sistema T1 americano (1.544 Mbit/s).

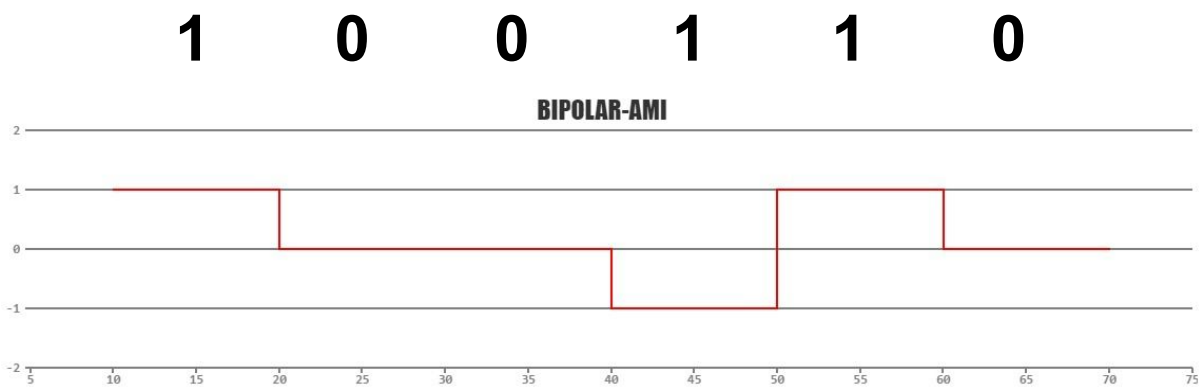
3.2 Vantagens

- Imune a inversão de polaridade.
- Boa eficiência espectral.
- Não perdera sincronização se uma sequência longa de 1s ocorrer.

3.3 Desvantagens.

- Sequência de 0s é um problema.
- Requer aproximadamente 3 dB mais potência de sinal do que um sinal de dois níveis para a mesma probabilidade de erro.

3.4 Ilustração do exemplo 100110



3.5 Trecho de código

```php
private function bipolar($array_binary){
    $clock = $array_bit = 0;
    $array_returno = array();
    $last = -1;
    foreach($array_binary as $bit){
        $clock = $clock + 10;
        if($bit == 0)
            $array_bit = 0;
        else
            $array_bit = $last = ($last == -1 ? 1 : -1);
        $array_returno[$clock] = $array_bit;
    }
    $array_returno[$clock+10] = $array_bit;
    return $array_returno;
}
```

Função que recebe uma sequência binária e faz uma verificação bit a bit para gera o sinal codificado BIPOLAR-AMI.

4.1 PSEUDOTERNARY

- 1 é representado pela ausência de sinal.
- 0 é representado por pulso positivo ou negativo (alternados).
- Usado no acesso básico RDIS (equipamento terminal).

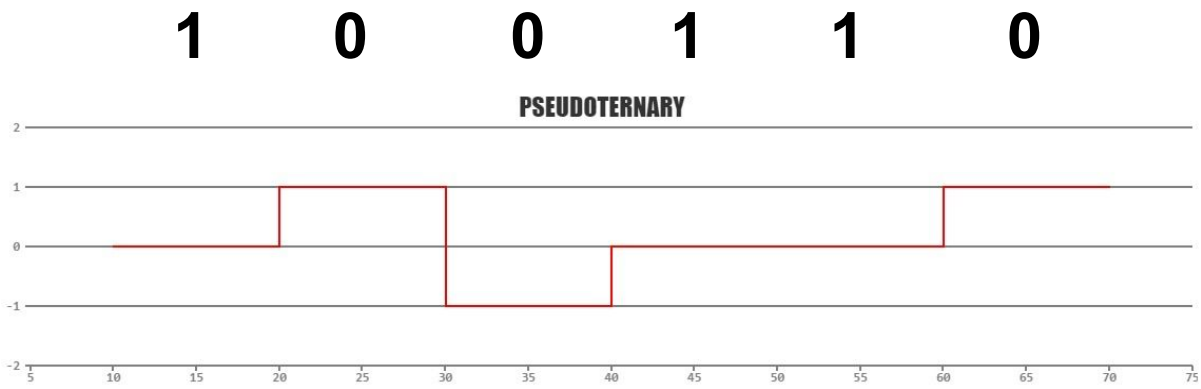
4.2 Vantagens

- Imune a inversão de polaridade.
- Boa eficiência espectral.
- Não perdera sincronização se uma sequência longa de 0s ocorrer.

4.3 Desvantagens.

- Sequência de 1s é um problema.
- Requer aproximadamente 3 dB mais potência de sinal do que um sinal de dois níveis para a mesma probabilidade de erro.

4.4 Ilustração do exemplo 100110



4.5 Trecho de código

```php
private function pseudoternary($array_binary){
    $clock = $array_bit = 0;
    $array_returno = array();
    $last = -1;
    foreach($array_binary as $bit){
        $clock = $clock + 10;
        if($bit == 1)
            $array_bit = 0;
        else
            $array_bit = $last = ($last == -1 ? 1 : -1);
        $array_returno[$clock] = $array_bit;
    }
    $array_returno[$clock+10] = $array_bit;
    return $array_returno;
}
```

Função que recebe uma sequência binária e faz uma verificação bit a bit para gera o sinal codificado PSEUDOTERNARY.

5.1 MANCHESTER

- Transição no meio de cada bit.
- Bit 1 transição ascendente.
- Bit 0 transição descendente.
- Utilizado no padrão de rede IEEE 802.3 (Ethernet).

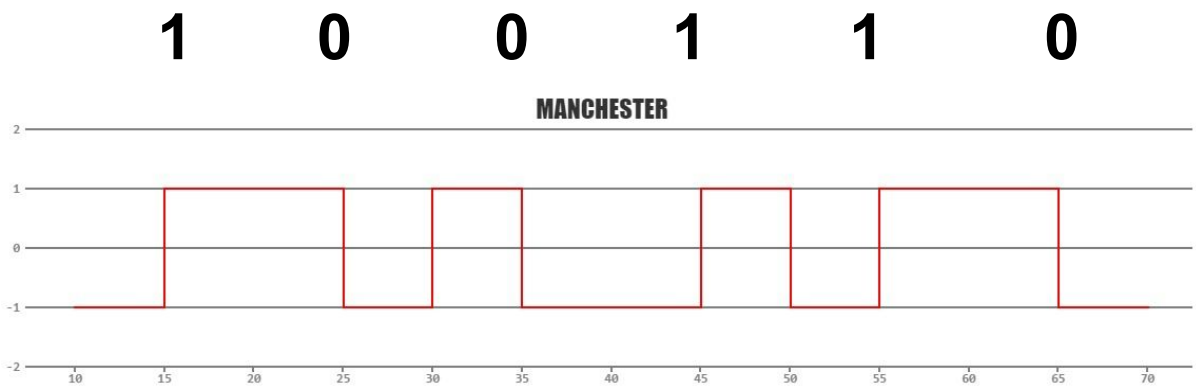
5.2 Vantagens

- Propriedade de auto-sincronização.
- Ausência de componentes espectrais de baixa frequência.

5.3 Desvantagens

- O baud rate é duplo do bit rate.
- Reque maior largura de banda do que o NRZ.

5.4 Ilustração do exemplo 100110



5.5 Trecho de código

```
private function manchester($array_binary){
    $clock = 0;
    $array_returno = array();
    foreach($array_binary as $bit){
        $clock = $clock + 10;
        if($bit == 0) {
            $array_returno[$clock] = 1;
            $array_returno[$clock+5] = -1;
        } else {
            $array_returno[$clock] = -1;
            $array_returno[$clock+5] = 1;
        }
    }
    $array_returno[$clock+10] = $array_returno[$clock+5];
    return $array_returno;
}
```

Função que recebe uma sequência binária e faz uma verificação bit a bit para gera o sinal codificado MANCHESTER.

6.1 DIFERENTIAL MANCHESTER

- Transição no meio de cada bit.
- Bit 0 transição no inicio do bit.
- Bit 1 ausência de transição no início do bit.
- Usado na LAN IEEE 802.5 (Token Ring).

6.2 Vantagens

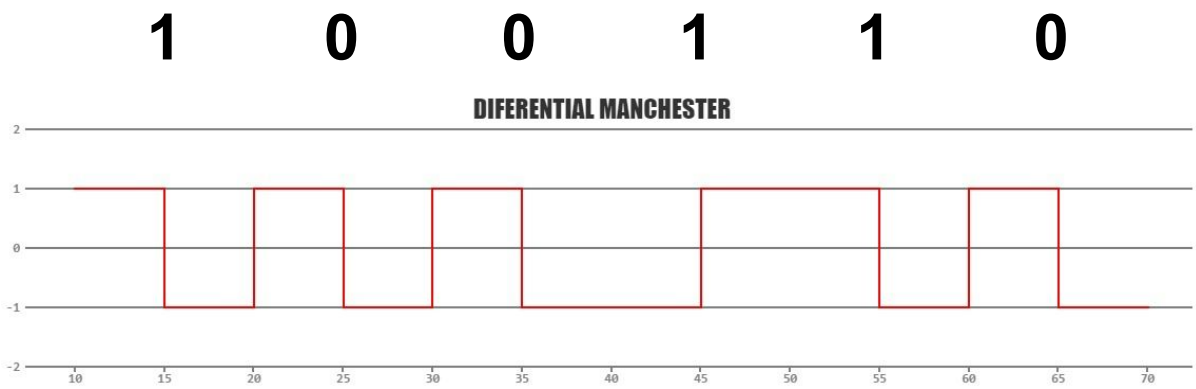
- Propriedade de auto-sincronização.

- Ausência de componentes espectrais de baixa frequência.

6.3 Desvantagens

- O baud rate é duplo do bit rate.
- Reque maior largura de banda do que o NRZ.

6.4 Ilustração do exemplo 100110



6.5 Trecho de código

```php
private function diferential_manchester($array_binary){
    $clock = 0;
    $sync = false;
    $array_returno = array();
    foreach($array_binary as $bit){
        $clock = $clock + 10;
        $sync = ($bit == 1 ? $sync ? false : true : $sync);
        if($sync) {
            $array_returno[$clock] = 1;
            $array_returno[$clock + 5] = -1;
        } else {
            $array_returno[$clock] = -1;
            $array_returno[$clock + 5] = 1;
        }
    }
    $array_returno[$clock+10] = $array_returno[$clock+5];
    return $array_returno;
}
```

Função que recebe uma sequência binária e faz uma verificação bit a bit para gera o sinal codificado DIFERENTIAL MANCHESTER.

7. Trechos de Códigos Complementares.

```php
public function resultado()
{
    $data['title'] = "Gráfico(s) para ".$_POST['binary'];
    $string = preg_replace("/[^0-1]/", "",preg_replace('/\s+/', '', $_POST['binary']));
    $array_binary = str_split($string);
    if(!empty($_POST['tipo'])) {
        foreach ($_POST['tipo'] as $t) {
            switch ($t) {
                case 0:
                    // NRZ-L
                    $data['resultado']['nome'][] = "NRZ-L";
                    $data['resultado']['retorno'][] = self::nrzl($array_binary);
                    break;
                case 1:
                    // NRZI
                    $data['resultado']['nome'][] = "NRZI";
                    $data['resultado']['retorno'][] = self::nrzi($array_binary);
                    break;
                case 2:
                    // BIPOLAR-AMI
                    $data['resultado']['nome'][] = "BIPOLAR-AMI";
                    $data['resultado']['retorno'][] = self::bipolar($array_binary);
                    break;
                case 3:
                    // PSEUDOTERNARY
                    $data['resultado']['nome'][] = "PSEUDOTERNARY";
                    $data['resultado']['retorno'][] = self::pseudoternary($array_binary);
                    break;
                case 4:
                    // MANCHESTER
                    $data['resultado']['nome'][] = "MANCHESTER";
                    $data['resultado']['retorno'][] = self::manchester($array_binary);
                    break;
                case 5:
                    // DIFERENTIAL MANCHESTER
                    $data['resultado']['nome'][] = "DIFERENTIAL MANCHESTER";
                    $data['resultado']['retorno'][] = self::diferential_manchester($array_binary);
                    break;
                default:
                    $data['resultado']['nome'][] = "NRZ-L";
                    $data['resultado']['retorno'][] = self::nrzl($array_binary);
                    break;
            }
        }
    } else {
        $data['resultado']['nome'][] = "NRZ-L";
        $data['resultado']['retorno'][] = self::nrzl($array_binary);
    }

    View::renderTemplate('header', $data);
    View::render('Welcome/Resultado', $data);
    View::renderTemplate('footer', $data);
}
```

Função que determina qual das funções de codificações chamar para gerar o gráfico e logo no começo previne a entrada de somente números binários.

```php
<div class="page-header">
    <h1><?=$title;?></h1>
</div>
<?php
$form = new \Helpers\Form();
echo $form->open(array("action" => "resultado", "method" => "post"));
echo '<div class="form-group">';
echo $form->input(array("name"=>"binary", "placeholder" => "Digite o código binário", "class" => "form-control", "required" => true));
echo '</div><div class="form-group">';
echo $form->checkbox(array(
    0=>array('id'=>'0', 'name'=>'tipo[]', 'value'=>'0', 'label'=>'NRZ-L' ),
    1=>array('id'=>'1', 'name'=>'tipo[]', 'value'=>'1', 'label'=>'NRZI' ),
    2=>array('id'=>'2', 'name'=>'tipo[]', 'value'=>'2', 'label'=>'BIPOLAR-AMI' ),
    3=>array('id'=>'3', 'name'=>'tipo[]', 'value'=>'3', 'label'=>'PSEUDOTERNARY' ),
    4=>array('id'=>'4', 'name'=>'tipo[]', 'value'=>'4', 'label'=>'MANCHESTER' ),
    5=>array('id'=>'5', 'name'=>'tipo[]', 'value'=>'5', 'label'=>'DIFERENTIAL MANCHESTER')
    ));
echo '</div>';
echo $form->submit(array("name" => "submit", "value" => "Gerar", "class" =>"btn btn-success"));
echo $form->close(); ?>
```

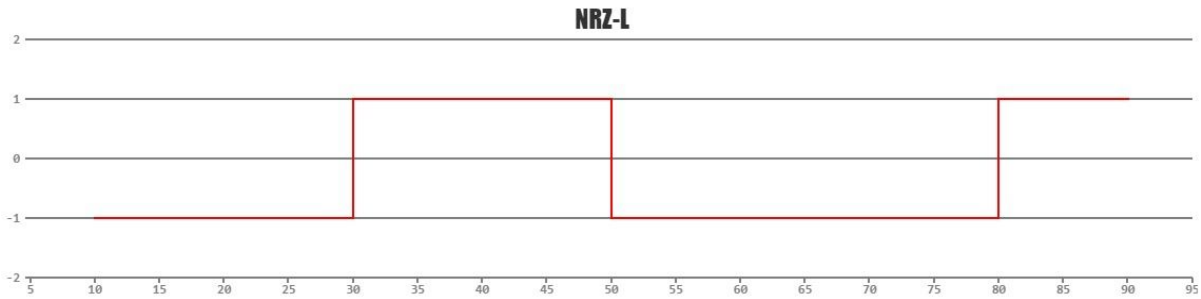
Código php que gera a página do simulador na web.

8.1 Testes e Resultados Analisados.

Nos testes realizados para o exemplo 11001110 podemos observar uma grande diferença no sinal produzido para os Binários NRZ (NRZ-L e NRZI), Binários Multinível (BIPOLAR-AMI e PSEUDOTERNARY) e Bifásicos (MANCHESTER e DIFERENTIAL MANCHESTER), devido a sua implementação, quanto a sua capacidade de encontrar erros e também evitar erros.

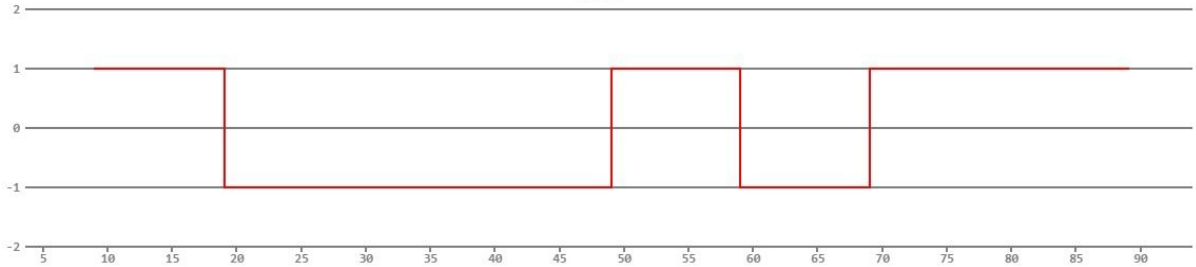
8.2 Resultados

1 1 0 0 1 1 1 0



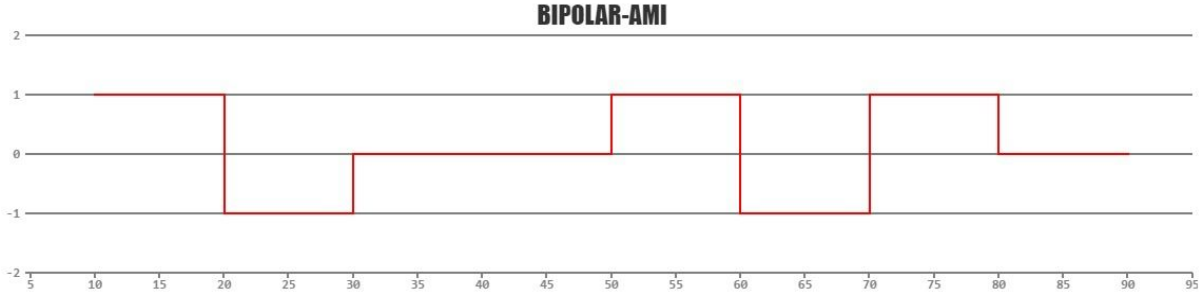
1 1 0 0 1 1 1 0

NRZI



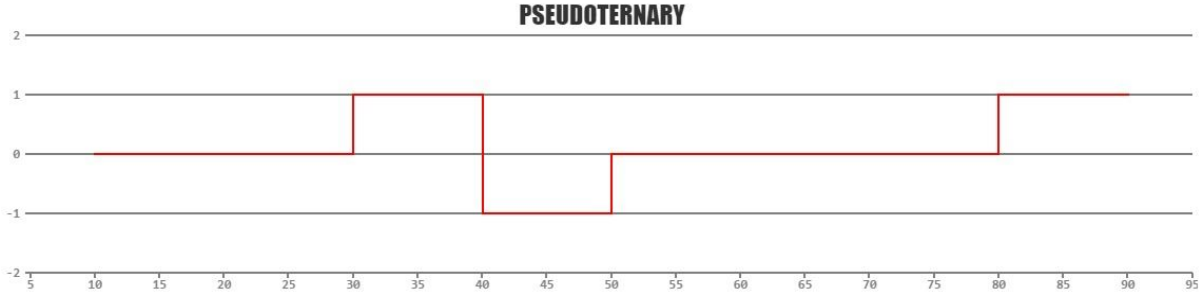
1 1 0 0 1 1 1 0

BIPOLAR-AMI



1 1 0 0 1 1 1 0

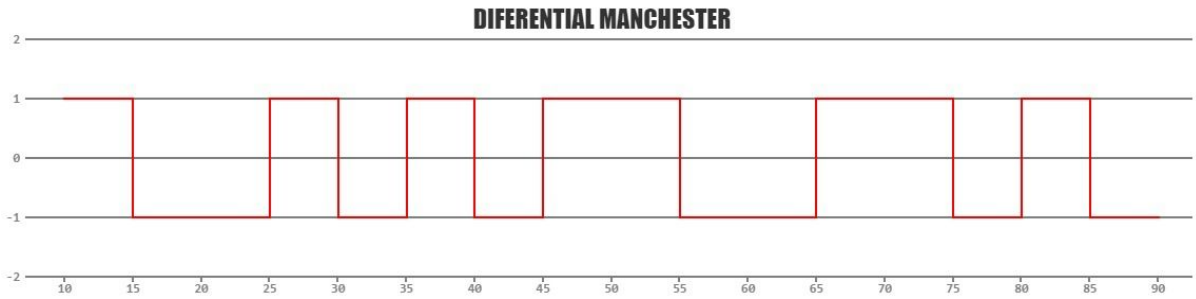
PSEUDOTERNARY



1 1 0 0 1 1 1 0

MANCHESTER

1 1 0 0 1 1 1 0



9. Referências Bibliográficas

https://web.fe.up.pt/~mricardo/02_03/cdrc1/teoricas/codificacao_v5.pdf

Simulador: http://comunicacao.16mb.com/

PDF Aula 5