Laboratório 2: Códigos Cíclicos

Matheus Vidal de Menezes, *Graduação, ITA,* Thayná Pires Baldão, *Graduação, ITA.*

***Resumo*—Neste artigo serão descritas as implementações com auxílio do *software* MATLAB® de um canal de transmissão, bem como de codificadores e decodificadores de códigos cíclicos para informações compostas por 4, 5, 6, 7, 8 e 10 *bits*, com taxas do sistema semelhantes a 60%. Simulando a transmissão de dados utilizando os codificadores, decodificadores e canal implementados, para diferentes valores de probabilidade p de alterar algum *bit* da palavra-código, conclui-se que, para códigos cíclicos com a mesma distância mínima de constelação, crescer o tamanho do bloco de informação transmitido não varia de forma significativa a probabilidade de erro de *bit* de informação P_b, sendo esta distância mínima o parâmetro mais importante de análise comparativa entre os códigos.**

I. Introdução

Em um mundo extremamente computadorizado, no qual os circuitos e os dados armazenados são digitais, a transmissão de *bits* é de suma importância para a comunicação entre sistemas. Durante a transmissão destes *bits*, todavia, é comum a ocorrência de falhas, de modo que as mensagens transmitidas nem sempre são exatamente as mensagens recebidas.

Sabendo desta problemática, existem diversos artifícios consagrados pela literatura para detectar e corrigir erros de transmissão de *bits*, um deles é a **codificação cíclica**. Essa estratégia será abordada detalhadamente neste artigo e se baseia no tratamento polinomial das informações a serem transmitidas, com o intuito de se obter palavras-código cujo primeiro *bit*, caso esteja corrompido, é fácil de se consertar. Quando o *bit* corrompido não é o primeiro, aproveita-se da propriedade cíclica do código, de modo que basta realizar rotações à direita para posicionar o erro da palavra-código no seu primeira *bit*, a fim de viabilizar a sua correção.

II. Análise Teórica

Na teoria da codificação, um código cíclico é um código de bloco, em que os deslocamentos circulares (rotações) de cada palavra-código fornecem outra palavra que pertence ao código. Uma propriedade importante destes códigos é o fato de que eles possuem propriedades algébricas eficientes para a detecção e correção de erros.

A. Codificação Cíclica

Considere uma palavra-código representada por um vetor v que contém n *bits*, isto é, $v = [v_0, v_1, ..., v_{n-1}]$. Por definição, se $v = [v_0, v_1, ..., v_{n-1}]$ é uma palavra-código, então, a sua rotação à direita $v'(D) = [v_{n-1}, v_0, v_1, ..., v_{n-2}]$ também é uma palavra-código. Neste relatório, utilizaremos a notação $v^{(i)}(D)$, para se referir a vetores que sofreram i rotações à direita.

Note que o vetor v pode ser representado de forma polinomial por meio do polinômio $v(D) = v_0 + v_1 D + v_2 D^2 + ... + v_{n-2} D^{n-2} + v_{n-1} D^{n-1}$, em que D é uma variável *dummy*, cuja função é indicar a posição do termo no vetor v.

Em códigos cíclicos, para se obter uma palavra codificada com n *bits*, a partir de uma informação de k *bits*, utilizaremos o tratamento algébrico supracitado das informações, juntamente com o conceito de **polinômio gerador** $g(D)$.

O polinômio gerador $g(D)$ consiste em uma palavra-código não nula com grau mínimo $n - k$ que possui as seguinte propriedades: $g_0 = 1$; $g(D)$ é único; qualquer combinação linear de $g(D)$ e $g(D)D$, $g(D)D^2$, ..., $g(D)D_{k-1}$ é uma palavra-código; assim como todas as palavras-código podem ser escritas por meio da combinação linear de $g(D)$ e $g(D)D$, $g(D)D^2$, ..., $g(D)D_{k-1}$.

É possível demonstrar que $g(D)$ é um fator de $(1 + D^n)$. **Portanto, $\mathbf{g(D)}$ pode ser obtido por meio da combinação linear de fatores irredutíveis de $\mathbf{(1 + D^n)}$. Note que, desta maneira, não é possível obter $\mathbf{g(D)}$ com qualquer grau, mas apenas com os graus dos polinômios resultantes da combinação linear de fatores irredutíveis de $\mathbf{(1 + D^n)}$. Não sendo possível obter $\mathbf{g(D)}$ para qualquer grau $\mathbf{n - k}$ desejado, também não será possível obter codificações para qualquer tamanho de bloco $\mathbf{(n, k)}$.**

Sabendo que todas as palavras-código podem ser escritas por meio da combinação linear de $\mathbf{g(D)}$ e $\mathbf{g(D)D}$, $\mathbf{g(D)D^2}, ..., \mathbf{g(D)D_{k-1}}$, é fácil perceber que a multiplicação de uma informação $\mathbf{u(D)}$ contendo $\mathbf{k}$ *bits*, pelo polinômio gerador $\mathbf{g(D)}$ de grau $\mathbf{n - k}$, resulta em um polinômio $\mathbf{v(D)}$ contendo $\mathbf{n}$ *bits*, com a palavra codificada. Portanto, o processo de codificação cíclica pode ser descrito pela Equação (1).[1]

$$v(D) = g(D) \cdot u(D) \tag{1}$$

É importante ressaltar que a multiplicação presente na Equação (1) pode gerar números que ultrapassam 1. Portanto, após realizar a operação descrita na Equação (1) é preciso calcular o módulo 2 dos coeficientes de $v(D)$ para se obter uma palavra-código contendo apenas números binários, conforme é esperado.

Além disso, note que é possível que existam vários polinômios $g(D)$ resultantes da combinação linear de fatores irredutíveis de $(1 + D^n)$ com o mesmo grau $n - k$. Desta maneira, para sanar o problema de escolha de $g(D)$, sempre que houver mais de um $g(D)$ com grau $n - k$ possível, escolheremos o $g(D)$ que possui a maior distância mínima entre dois pontos da constelação que ele gera.

B. Modelagem do Canal

Para que a comunicação entre dois sistemas ocorra é necessário utilizar-se um canal de transmissão. Justamente nesta fase mais física do processo é que os *bits* de informação podem ser corrompidos, isto é, os ruídos presentes podem

[1] Resposta à pergunta 4 requisitada no roteiro desta atividade laboratorial.

perturbar o sinal de tal forma que seu significado mude de 0 para 1 ou de 1 para 0.

Para as análises a serem feitas neste artigo, será utilizado um canal de transmissão do tipo **BSC** (*Binary Symmetric Channel*), em que a probabilidade p de que um *bit* de informação seja transmitido de forma incorreta, independe da informação originalmente contida neste *bit*.

C. Decodificação Cíclica

Após a passagem da palavra-código $v(D)$ pelo canal de transmissão, ela será transformadas em uma mensagem recebida $r(D)$, contendo, possivelmente, *bits* corrompidos, conforme descrito pela Equação (2), em que $e(D)$ é uma palavra com todos os *bits* nulos, com exceção dos *bits* que possuem erro, que serão unitários.

$$r(D) = v(D) + e(D) \tag{2}$$

O resto da divisão de $r(D)$ por $g(D)$ é chamado de **síndrome** e é representado pelo polinômio $s(D)$, conforme descrito na Equação (3). A obtenção das síndromes é um processo bastante importante para a decodificação cíclica, pois é possível associar os padrões de erro detectáveis pelo sistema de decodificação com uma síndrome ao mapear, entre todos os erros que causam esta síndrome, o mais provável.

$$s(D) \equiv r(D) \pmod{g(D)} \tag{3}$$

O decodificador cíclico implementado neste laboratório é capaz de identificar erros na primeira posição das palavras recebidas, desde que estas palavras possuam o número de *bits* corrompidos inferior a $\frac{d_{min}}{2}$, em que d_{min} corresponde à distância mínima da constelação gerada por meio de $g(D)$.

Sendo assim, para realizar o processo de decodificação é necessário realizar uma etapa de pré-processamento em que são identificadas todas as síndromes que ocorrem com as palavras recebidas contendo *bits* corrompidos na primeira posição de $r(D)$, bem como contendo até, no máximo, $\frac{d_{min}}{2}$ *bits* corrompidos. Então, essas síndromes são armazenadas em um **conjunto A**.

Após realizar este pré-processamento, é possível começar a decodificar palavras. A primeira etapa do processo de decodificação consiste em identificar-se a síndrome associada à palavra recebida $r(D)$, por meio da Equação (3).

Caso o polinômio $s(D)$ obtido não seja um polinômio pertencente ao conjunto A, sabe-se que há algum erro na palavra recebida $r(D)$. Além disso, sabe-se que este erro não está no primeiro *bit*, do contrário, a síndrome de $r(D)$ pertenceria ao conjunto A. Neste caso, iremos nos aproveitar do fato de estarmos trabalhando com um código cíclico e tentaremos deslocar o *bit* errado para a primeira posição, utilizando, para isso, rotações à direita. Neste processo, iremos rotacionar tanto a palavra recebida $r(D)$ quanto a síndrome obtida para aquela palavra $s(D)$.

Para rotacionar a palavra recebida $r(D)$ à direita, basta armazenar o coeficiente do termo D^{n-1} em uma variável auxiliar, fazer um *shift* à direita de todos os outros coeficientes, isto é, o coeficiente do termo D^i passa a ser o coeficiente do termo D^{i+1}. Após realizar este processo, basta atribuir ao coeficiente do termo D^0 o valor armazenado na variável auxiliar e obter-se-á $r'(D)$.

Já para rotacionar a síndrome $s(D)$ à direita, o procedimento é um pouco mais complexo. Além de se fazer o *shift* à direita de todos os coeficientes de $s(D)$, zera-se o primeiro *bit* e, então, realiza-se a adição módulo 2 do polinômio obtido por meio deste procedimento com o polinômio que corresponde a $g(D)$ truncado para o grau $n-1$. O polinômio resultante é o $s'(D)$.

Estes procedimentos de rotação à direita da palavra recebida e da síndrome serão repetidos até que a síndrome encontrada pertença ao conjunto A, ou até que $s(D)$ seja um polinômio nulo ou ainda até que n rotações tenham sido realizadas, o que significa que todas as rotações possíveis foram realizadas e, apesar disso, não conseguiu-se alocar o *bit* errado na primeira posição, de modo que nenhuma síndrome pertence ao conjunto A.

Se o polinômio $s(D)$ obtido for um polinômio pertencente ao conjunto A, sabe-se que o *bit* corrompido está na primeira posição, logo, basta consertá-lo e repetir o cálculo da síndrome, mas agora com $r(D)$ ajustado, com o intuito de verificar se a correção do primeiro *bit* de $r(D)$ foi suficiente para se obter uma síndrome correspondente ao polinômio nulo.

Já no caso de a síndrome obtida ser o polinômio nulo, isso significa que o polinômio não possui *bits* corrompidos detectáveis e, portanto, $r(D) = v(D)$ e, para se obter a informação original $u(D)$ decodificada, caso nenhuma rotação tenha sido realizada na palavra recebida $r(D)$, basta dividir $r(D)$ por $g(D)$, conforme descrito pela Equação (4).

$$u(D) = \frac{r(D)}{g(D)} \tag{4}$$

É importante ressaltar, contudo, que se a palavra recebida $r(D)$ tiver sido rotacionada à direita para forçar que o *bit* errado fosse o primeiro, é preciso rotacionar à esquerda $r(D)$ a mesma quantidade de vezes que $r(D)$ foi rotacionada à direita, antes de se utilizar a Equação (4). Por causa disso, é importante manter um contador contabilizando o número x de rotações realizadas até se encontrar a síndrome cujo polinômio é nulo.

III. Implementação

Baseado na teoria detalhada na Seção II, foram implementadas as codificações cíclicas requisitados pelo roteiro [1] deste laboratório, para palavras-código com tamanhos $n = \{8, 10, 12, 14, 16\}$. A taxa escolhida foi de aproximadamente de 60%, o que ocasionou a codificação de informações $u(D)$ de k *bits*, com $k = \{5, 6, 7, 8, 10\}$.

A. Codificador Cíclico

A fim de que a codificação fosse feita, foi necessária a realização de uma etapa de pré-processamento: a determinação do polinômio gerador $g(D)$, conforme descrito na Subseção II-A. Para tanto, dado um n e a $taxa = k/n$, primeiramente, gerou-se as 2^k possíveis informações $u(D)$ contendo k *bits*. Então, foram obtidos todos os possíveis polinômios candidatos a $g(D)$ com o auxílio da função `cyclpoly(n,k,'all')` própria do *MATLAB®*.

Conhecendo os possíveis candidatos a $g(D)$, para encontrar o polinômio $g(D)$ que atendia ao requisito de possuir a maior distância mínima entre dois pontos da constelação por ele gerada, foi preciso gerar os conjuntos de todas as palavras-código

$v(D)$ possíveis (constelação) para cada um dos candidatos a $g(D)$. Então, analisou-se separadamente cada uma destas constelações, com o intuito de se encontrar a distância mínima entre dois pontos pertencentes a ela. Como a constelação é um conjunto fechado, a distância entre duas palavras-código pertencentes a ela também é uma palavra-código da constelação. Desta maneira, para se encontrar a distância mínima entre dois pontos da constelação não foi preciso, efetivamente, calcular a distância entre dois pontos, mas apenas somou-se os *bits* de cada uma das palavras-código $v(D)$ da constelação e obteve-se o mínimo dentre estes valores. Posteriormente, foi necessário encontrar a maior dentre as distâncias mínimas obtidas para as constelações distintas e selecionar o $g(D)$ associado a esta máxima distância mínima. **Obter o $g(D)$ por meio deste método específico foi a maior dificuldade encontrada pelos autores deste artigo para se obter a codificação cíclica**[1].

Note que este algoritmo de pré-processamento possui complexidade de tempo de $O(2^k)$, pois é preciso gerar todas as palavras-código das constelações associadas a um número constante de candidatos a $g(D)$ (no máximo 2 candidatos para os valores de (n, k) implementados).

De posse do polinômio gerador $g(D)$, foi implementado um codificador de canal para o código cíclico. Basicamente, a codificação é dada pela Equação (1) seguida da obtenção do módulo 2 de todos os coeficientes obtidos para $v(D)$. **Desta maneira, o algoritmo codificador, desconsiderando o pré-processamento de determinação de $\mathbf{g(D)}$ (que ocorre apenas na realização da codificação da primeira informação) é $\mathbf{O((n-k)\cdot k) = O(n\cdot k)}$ em complexidade de tempo**.[2]

B. Decodificador Cíclico

Para detectar erros de transmissão no processo de decodificação foram implementadas correções via síndrome, tal como explicado na Subseção II-C. Desta forma, foi necessário como pré-processamento da decodificação a determinação de uma matriz A definida como o conjunto das síndromes nas quais há erro no primeiro *bit* (conjunto A).

Aqui, vale uma ressalva: esta matriz A **não** armazena somente as síndromes associadas a erros **apenas** no primeiro *bit*, mas sim todas as síndromes contendo erros no primeiro *bit* com o número de *bits* errados correspondentes até no máximo a metade da distância mínima (aquela encontrada na determinação do polinômio gerador). Desta maneira, houve casos nos quais a matriz A era composta por uma única síndrome (*e.g.*, $\mathbf{n = 8}$, com $\mathbf{d_{min} = 2}$), mas também houve casos, em que a matriz A era composta por mais de uma síndrome (*e.g.*, $\mathbf{n = 12}$, com $\mathbf{d_{min} = 4}$). É possível visualizar todos os casos na Tabela I. **Obter a matriz A desta maneira foi a maior dificuldade encontrada pelos autores deste artigo para se obter a decodificação cíclica.**[3]

Esse pré-processamento para a determinação da matriz A gera todas as palavras-código associadas à $g(D)$, bem como todas as suas síndromes e, em um primeiro momento, adiciona todas essas síndromes na matriz A. Posteriormente, retiram-se da matriz A as síndromes provenientes de palavras que não possuem o primeiro *bit* corrompido, assim como retiram-se as síndromes provenientes de palavras com número de *bits* corrompidos superior a $\frac{d_{min}}{2}$. Por causa disso, esse pré-processamento é $O(2^k)$ em termos de complexidade de tempo.

De posse da matriz A, o processo de decodificação inicia-se com o cálculo da síndrome da palavra recebida. Caso a síndrome obtida seja o polinômio nulo, a palavra recebida não possui erros e basta utilizar a Equação (4) para se obter a palavra decodificada. Já caso a síndrome não seja nula, mas seja pertencente à matriz A, identifica-se um erro no primeiro *bit* e, portanto, inverte-se o seu valor a fim de corrigi-lo, e, então, calcula-se a síndrome da palavra corrigida para verificar se a síndrome obtida é o polinômio nulo. Por último, caso a síndrome obtida não seja nula, e não pertença à matriz A, rotaciona-se à direita a palavra recebida e a síndrome.

Vale ressaltar que nem sempre o decodificador será capaz de corrigir os *bits* corrompidos da palavra recebida, já que ele realiza a correção apenas quando a síndrome rotacionada à direita recai em uma das síndromes mapeadas pela matriz A. **Assim, da mesma forma que o código 8/14 implementado no laboratório anterior não era capaz de corrigir todos os erros, por falta de mapeamento de síndromes, analogamente na decodificação cíclica há casos nos quais o decodificador implementado rotacionará a palavra recebida n vezes e não corrigirá *bit* corrompido algum.** [4] Desta maneira, **como o algoritmo de decodificação no pior caso de execução executa $\mathbf{n}$ rotações na palavra recebida, com as rotações sendo, cada uma, $\mathbf{O(n)}$ em complexidade de tempo, a complexidade de tempo do algoritmo de decodificação cíclica, desconsiderando o pré-processamento da determinação da matriz $\mathbf{A}$ (que ocorre apenas na decodificação da primeira informação), é $\mathbf{O(n^2)}$.**[5]

IV. Resultados e Discussões

De posse das implementações, foram obtidas as curvas de probabilidade de erro de *bit* de informação, em função do parâmetro p, o qual corresponde à probabilidade de se corromper algum *bit* da palavra-código. Foram adotados valores: $p = 0.5, 0.2, 0.1, 0.05, 0.02, 0.01, 0.005, ..., 2\cdot 10^{-6}$.

Além dos valores de n propostos, **foi também gerado, de modo extra, o resultado para código cíclico com $\mathbf{n = 7}$, a fim de compará-lo com a codificação de *Hamming* também para $(n, k) = (7, 4)$. O resultado geral para todos os valores de n simulados encontra-se na Figura 1, na qual observa-se um comportamento muito semelhante entre a codificação de *Hamming* e a cíclica, ambos para $\mathbf{n = 7}$. Isso é justificado, pelo fato de o código Hamming ser um código cíclico.** [6]

É possível observar na Figura 1 que todas as probabilidades de erro de *bit* de transmissão caem, à medida que a probabilidade p diminui, o que é esperado, já que quanto menor for o número de *bits* corrompidos, mas eficiente o código cíclico será em detectar e corrigir os erros de transmissão. **Além disso, é possível observar que, apesar de todas as codificações apresentadas possuírem taxas próximas,**

[1] Resposta à pergunta 1 requisitada no roteiro desta atividade laboratorial.
[2] Resposta à pergunta 6 requisitada no roteiro desta atividade laboratorial.
[3] Resposta à pergunta 1 requisitada no roteiro desta atividade laboratorial.
[4] Resposta à pergunta 2 requisitada no roteiro desta atividade laboratorial.
[5] Resposta à pergunta 6 requisitada no roteiro desta atividade laboratorial.
[6] Resposta à pergunta 3 requisitada no roteiro desta atividade laboratorial.

o código de *Hamming* com $(n,k) = (7,4)$ foi o que obteve melhor resultado, dado que apresentou menores probabilidades de erro de transmissão de *bit*. Isso pode ser justificado por três fatores: a distância mínima entre as palavras-código geradas por este código é uma das mais altas, este é o código com o menor valor de n, assim como a taxa deste código não é de exatamente 60%, mas sim de $\approx$ 57%. [1]

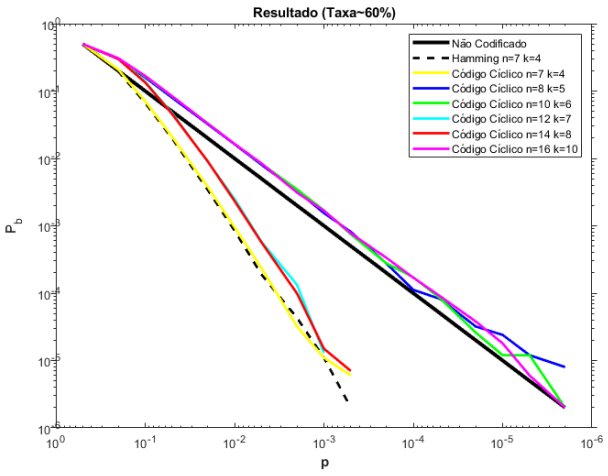

Fig. 1. Resultados das simulações realizadas, por meio do *software* MATLAB®, em escala logarítmica.

Conforme pode ser observado na Tabela I, extraem-se os seguintes casos: $d_{min} = 2, d_{min} = 3$ e $d_{min} = 4$, cujas configurações para duas palavras-código são dadas respectivamente pelas Figuras 2, 3 e 4.

Para $d_{min} = 2$, são somente corrigidos erros de peso igual a 1, o que cai diretamente no ponto médio entre os sinais, vide Figura 2. Disso resulta que a probabilidade de acerto ou erro fica com um comportamento acima, porém ainda muito próximo da reta de não codificação, o que corresponde ao que, de fato, se observa para $n = \{8, 10, 16\}$ na Figura 1.

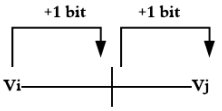

Fig. 2. Configuração genérica para dois sinais codificados ciclicamente para $n = \{8, 10, 16\}$, com $d_{min} = 2$.

Para $d_{min} = 3$, também são somente corrigidos erros de peso igual a 1, o que cai diretamente num ponto mais próximo de v_i do que no caso anterior. Disso resulta que a probabilidade de acerto ou erro fica com um comportamento abaixo da reta de não codificação, que é o que, de fato, ocorre para $n = \{14\}$ na Figura 1.

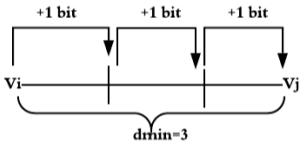

Fig. 3. Configuração genérica para dois sinais codificados ciclicamente para $n = \{14\}$, com $d_{min} = 3$.

[1]Resposta à pergunta 3 requisitada no roteiro desta atividade laboratorial.

Para $d_{min} = 4$, são corrigidos apenas erros de peso igual a 1 ou 2, o que pode cair num ponto mais próximo de v_i ou diretamente no ponto médio. Por conta do efeito do canal de não codificação para correção de 2 *bits*, acaba que o resultado obtido com $n = 12$ é muito similar àquele para $n = 14$.

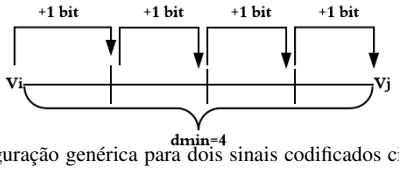

Fig. 4. Configuração genérica para dois sinais codificados ciclicamente para $n = \{12\}$, com $d_{min} = 4$.

TABELA I
RESULTADOS DOS PARÂMETROS.

n	$g(D)$	d_{min}	A
8	[1 1 1 1]	2	[1 0 0]
10	[1 1 1 1 1]	2	[1 0 0 0]
12	[1 0 1 1 0 1]	4	[1 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0]
14	[1 0 0 0 1 0 1]	3	[1 0 0 0 0 0]
16	[1 0 1 0 1 0 1]	2	[1 0 0 0 0 0]

V. CONCLUSÃO

Visando entender mais sobre a codificação cíclica de informações e seus efeitos em sistemas de comunicação, foram implementados e comparados codificadores/decodificadores cíclicos para vários tamanhos de bloco. Inclusive foi evidenciado experimentalmente que o código de Hamming, é também um código cíclico.

Conforme pode ser observado na Figura 1, com todas as codificações com taxa semelhantes a 60%, para códigos cíclicos com a mesma distância mínima de constelação, crescer o tamanho do bloco de informação transmitido não varia de forma significativa a probabilidade de erro de *bit* de informação P_b, sendo esta distância mínima o parâmetro mais importante de análise comparativa entre os códigos. Desta forma, os códigos que possuem as maiores distâncias mínimas corrigem mais *bits*, conforme o esperado e, portanto, têm menores probabilidades de erro.

Por causa disso, em ordem de desempenho crescente, obteve-se os valores de n: 7, $\{12, 14\}$ e $\{8, 10, 16\}$. Neste contexto, saber escolher qual código utilizar, de acordo com os parâmetros apresentados na Tabela I, é um questionamento importante a se fazer no projeto de sistemas de telecomunicação envolvendo códigos cíclicos. [2]

REFERÊNCIAS

[1] Roteiro do Laboratório 2, 2019, online, disponível em: http://www.ele.ita.br/~manish/ele32/Documentos/ELE32-20180911-R2.pdf, consultado em 27 de Setembro de 2019.

[2]Resposta à pergunta 5 requisitada no roteiro desta atividade laboratorial.