Universidade Estadual de Campinas – UNICAMP

Faculdade de Tecnologia – FT

Comunicação por Luz Visível: Codificação de canal com controle do fator de escurecimento e cintilação

Bruno Fracasso

Limeira - SP

2020

Universidade Estadual de Campinas – UNICAMP

Faculdade de Tecnologia – FT

Comunicação por Luz Visível:
Codificação de canal com controle do fator de escurecimento e cintilação

Bruno Fracasso

Orientador: Prof. Dr. Jaime Portugheis

TCC apresentado à Faculdade de Tecnologia (FT)
como requisito de conclusão do
Curso de Engenharia de Telecomunicações.

Limeira - SP

2020

Agradecimentos

A minha mãe, em primeiro lugar, que me fez persistir com os estudos no curso mesmo nos momentos mais difíceis.

A minha namorada, por sempre estar do meu lado, me dando conselhos realistas e maduros e me convencendo de que sou maior do que penso.

Ao professor Ivan, que soube me dar um norte sobre que rumo tomar quando a minha situação estava na pior.

Ao meu orientador, Jaime Portugheis, por estar sempre à disposição de ajudar e fornecer materiais de estudo sobre o trabalho.

Ao meu amigo, Pedro Pozelli, que teve participação neste trabalho com estudos focados no controle do fator de escurecimento e foi de imensa ajuda em todos os quesitos.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Resumo

O trabalho faz um estudo sobre sistemas de comunicação por luz visível. Estes sistemas utilizam LEDs que além de prover iluminação, transmitem informação a taxas muito rápidas. O foco do estudo são os sistemas com codificação de canal, com controle do fator de escurecimento e mitigação da cintilação. Diversos sistemas propostos na literatura são comparados para diferentes fatores de escurecimento e taxas de informação. A comparação mostrou que existem esquemas de codificação de canal com a propriedade inerente de mitigação da cintilação.

Palavras-chave: Comunicação por luz visível, codificação de canal, mitigação da cintilação, controle do fator de escurecimento, sequências RLL, códigos LDPC.

Abstract

This work studies visible light communication systems. These systems use LEDs that, in addition to providing lighting, transmit information at very fast rates. Its focus is on channel coding systems with dimming factor control and flickering mitigation. Several systems proposed in the literature are compared for different dimming factors and information rates. The comparison showed that there are channel coding schemes with the inherent flicker mitigation property.

Keywords: Visible light communication, channel coding, flicker mitigation, dimming factor control, RLL sequences, LDPC codes.

Glossário

- IEEE (Institute of Electrical and Electronics Engineers)
- LED (Light Emitting Diode)
- MAC (Medium Access Control)
- OOK (On-Off Keying)
- WDM (Wavelength-division Multiplexing)
- SCM (Subcarrier Multiplexing)
- OFDM (Orthogonal Frequency Division Multiplexing)
- QAM (Quadrature Amplitude Modulation)
- BPL (Broadband over power lines)
- RGB (Red Green Blue)
- RLL (Run-Length Limited)
- LDPC (Low-density Parity-Check)

Capítulo 1

Introdução

1.1. Comunicação por luz visível

O espectro eletromagnético é uma escala de radiações eletromagnéticas de acordo com suas frequências ou comprimentos de onda. O espectro de luz visível é formado pelo intervalo de comprimentos de onda de 380 a 780 nm. A *Figura 1* abaixo ilustra o espectro visível.

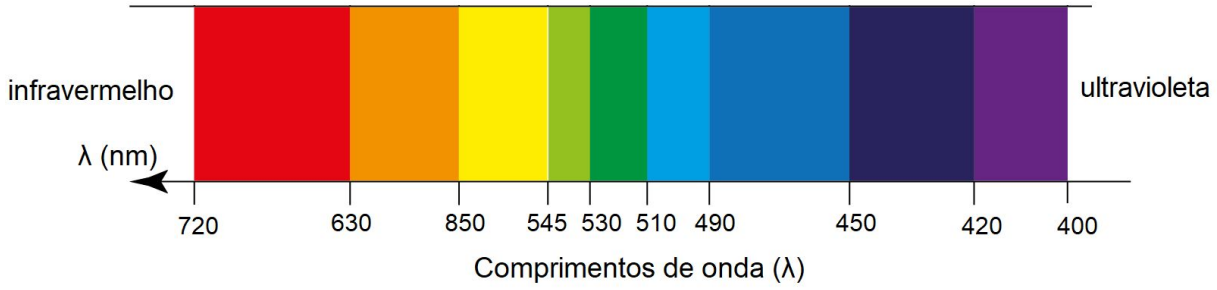

Figura 1: Comprimentos de onda do espectro de luz visível.
Fonte: Adaptada de <https://www.filofima.com.br/fisica/quantica/calculadora-radiancia-espectral.html>

A Comunicação por Luz Visível (ou VLC, do inglês Visible Light Communications) é padronizada pelo Instituto de Engenheiros Eletricistas e Eletrônicos na norma IEEE (do inglês, Institute of Electrical and Electronics Engineers) 802.15.7. Utiliza do espectro visível para a transmissão de dados sem fio de curto alcance, a partir da modulação de fontes reguláveis de luz, com taxas de dados de até 96 Mbps, sem cintilação [1]. Essas fontes de luz podem ser diodos emissores de luz (LEDs, do inglês, Light Emitting Diode) ou diodos laser; a intensidade destes é modulada em velocidades imperceptíveis ao olho humano.

1.2. Motivação

A comunicação em radiofrequências (RF) é amplamente utilizada atualmente. De acordo com [2], algumas limitações nessa tecnologia são interferência, largura de banda limitada, problemas de segurança devidos à fácil penetração das ondas RF nas paredes. Por outro lado, na utilização de VLC, há imunidade à interferência de fontes eletromagnéticas e largura de banda alta. A luz visível não penetra nas paredes, evitando recepção indevida. Uma fonte de luz visível pode ser usada tanto para iluminação quanto para comunicação, resultando em economia de energia.

Outro fator para o aumento do interesse em VLC são as recentes melhorias de eficiência energética e menor tempo de transição dos LEDs [1]. Comparados com métodos de iluminação convencionais, em geral, os LEDs brancos têm menor consumo de potência, menor tensão de operação, maior vida útil, menor tamanho e menor geração de calor [3].

De acordo com [2], algumas aplicações em potencial de VLC incluem Li-Fi (do inglês, Light Fidelity), comunicação entre veículos, comunicação subaquática, comunicação de máquinas em hospitais. O Li-fi usa VLC para prover internet de alta velocidade até 10 Gbps. Algumas aplicações no trânsito (como a prevenção de colisões) necessitam de baixa latência, o que é possível pela grande largura de banda da VLC. Outra aplicação possível é a localização *indoor* [4], visto que o GPS não funciona em locais fechados.

1.3. Desafios na implementação

De acordo com [2], alguns desafios para a implementação de VLC são: a interferência com fontes de luz ambiente e entre dispositivos VLC, e a integração com tecnologias existentes como WiFi (do inglês, Wireless Fidelity). Além disso,

existe um problema simultâneo de transmitir informação garantindo um fator de escurecimento e valores aceitáveis de cintilação [1].

O fator de escurecimento é a proporção do nível real em relação ao nível máximo de iluminação da fonte de luz, geralmente regulado por uma pessoa de acordo com sua comodidade. O seu controle também é importante em termos de economia e eficiência energética.

A cintilação é a flutuação do brilho da luz, e é prejudicial aos olhos humanos; pode ser evitada pela variação do brilho da luz em períodos de tempo curtos (até 5 ms) de modo que a variação seja indetectável aos olhos. A seguir será dada uma visão geral das recomendações feitas pelo IEEE em relação à cintilação.

1.3.1. Recomendação IEEE com relação à cintilação

Todas as fontes de luz estão sujeitas a um certo grau de cintilação por usarem fontes de energia de corrente alternada. O IEEE realizou estudos sobre a cintilação [5] causada pela oscilação da amplitude da tensão em dispositivos que geram iluminação, e criou uma recomendação para mitigar os possíveis danos que esse fenômeno pode causar à saúde das pessoas expostas a essa iluminação. Essa recomendação é chamada IEEE Std 1789™-2015.

Existem riscos à saúde associados à cintilação não controlada dos LEDs. Quando a variação da luz é percebida conscientemente, o que se deve a uma predisposição da pessoa exposta, a cintilação é chamada visível, sendo seus danos mais imediatos. Quando os efeitos da variação da luz são sentidos inconscientemente, a cintilação é chamada invisível, e os riscos estão associados à exposição por longa duração.

De acordo com os pesquisadores desta recomendação, os seguintes efeitos já foram observados:

- Convulsão induzida por luzes intermitentes.
- Enxaqueca ou hemicrania paroxística severa, geralmente associadas a enjoo e perturbações visuais.
- Aumento no comportamento repetitivo em indivíduos com autismo.
- Fadiga ocular, visão embaçada, dores de cabeça e diminuição no desempenho em atividades relacionadas à visão.

As frequências de cintilação na faixa de 3 a 70 Hz são um risco em potencial para convulsões, sendo a probabilidade deste fenômeno maior na faixa de oscilação de 15 a 20 Hz. A cintilação invisível pode causar danos à saúde dos observadores quando na frequência abaixo de 165 Hz.

1.4. Visão geral do trabalho

No capítulo 2, é feita uma breve descrição das camadas de uma arquitetura de sistema VLC; também são discutidos alguns motivos pelos quais se aumentou a utilização de LEDs nesses sistemas.

No capítulo 3, é feita uma introdução aos códigos corretores de erro, alguns destes são discutidos; também se introduz seu papel no fator de escurecimento. No capítulo 4, é feita uma introdução aos códigos de linha que contém certas restrições importantes para evitar a cintilação em um sistema VLC.

No capítulo 5, são apresentados alguns esquemas que dão suporte ao fator de escurecimento e/ou à mitigação da cintilação. Algumas comparações são realizadas, apresentando-se as características positivas e negativas desses esquemas.

No capítulo 6, são apresentados e discutidos os resultados de algumas implementações baseadas em sequências (d, k), e implementações de códigos LDPC (do inglês, Low-density Parity-Check).

Capítulo 2

Arquitetura de um sistema VLC

Como os sistemas de comunicação em geral, um sistema VLC possui duas partes básicas: transmissão e recepção. Estas duas consistem geralmente de três camadas em comum: a camada física, a camada de enlace de dados, e a camada de aplicação [2]. O escopo deste trabalho abrange a camada física e a subcamada MAC (do inglês Medium Access Control) da camada de enlace de dados. A *Figura 2* esquematiza a relação entre essas camadas.

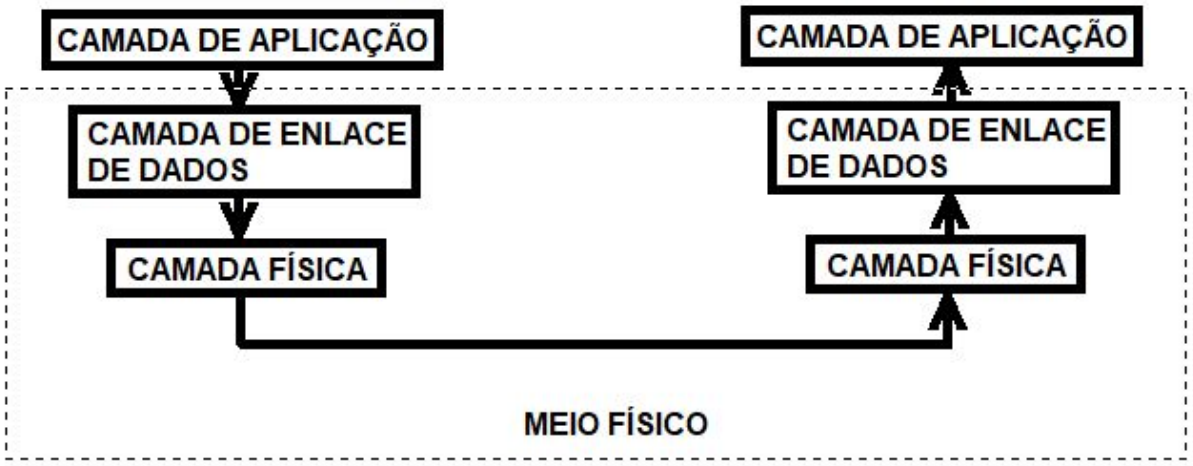

Figura 2: Camadas de um sistema VLC.
Fonte: Adaptada de [2].

2.1. Camada física

De acordo com [2], a camada física faz: (1) a especificação física do dispositivo, e (2) a relação entre o dispositivo e o meio. Esta camada apresenta estrutura sequencial representada na *Figura 3*.

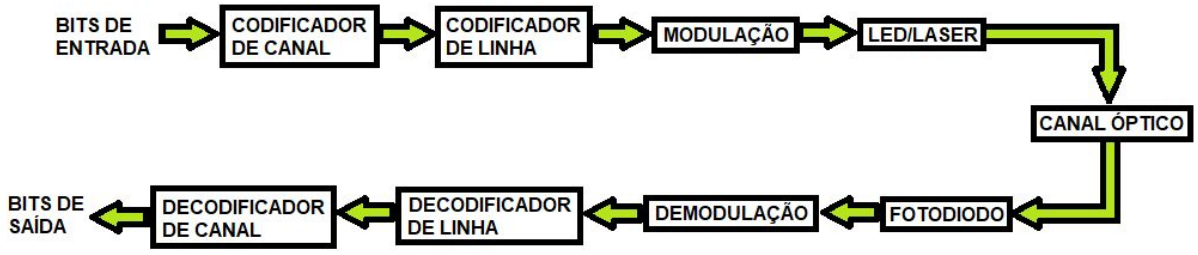

Figura 3: Camada física de um sistema VLC
Fonte: Adaptada de [2]

Inicialmente, um fluxo de bits passa pelo codificador de canal, o que pode melhorar o desempenho do sistema VLC [2], em especial por possibilitar a correção de erros na transmissão. Na sequência, esse fluxo passa por um codificador de linha, ou modulação em banda base, o que, dentre outros, melhora o sincronismo com o receptor e torna possível a mitigação da cintilação .

A modulação, conceito amplamente conhecido nas telecomunicações, é basicamente um processo em que um sinal a ser transmitido modifica ondas eletromagnéticas de uma forma que estas carregam sua informação. Em VLC, a modulação é realizada por variações na intensidade da luz que correspondem à informação a ser transmitida.

A multiplexação por divisão de comprimento de onda (WDM, do inglês, Wavelength-division Multiplexing) e a multiplexação por subportadora (SCM, do inglês, Subcarrier Multiplexing) podem ser usadas para transmissão bidirecional. Além disso, a multiplexação por divisão de frequência ortogonal (OFDM, do inglês, Orthogonal Frequency Division Multiplexing) e a modulação de amplitude em quadratura (QAM, do inglês, Quadrature Amplitude Modulation) podem ser usadas para aumentar a taxa de dados [2].

Dos fatores a serem considerados no projeto do esquema de modulação para VLC incluem o fator de escurecimento e a cintilação [2]. A luz deve ser modulada

numa taxa maior que 200 Hz para evitar efeitos prejudiciais à saúde [9]. Os diferentes esquemas de modulação propostos na norma IEEE 802.15.7 permitem um bom compromisso entre taxas de transmissão de dados e diferentes níveis de escurecimento [9]. O esquema de modulação OOK (do inglês, On-Off Keying) tem como principal vantagem a facilidade de implementação [2].

Tanto LEDs como lasers podem ser usados em sistemas VLC, sendo o LED usado quando a comunicação e a iluminação são realizadas por um mesmo dispositivo [2]. Um transmissor típico em VLC é composto por um conjunto de luminárias (LEDs), um circuito para o controle dos LEDs e módulos para a conexão do circuito de controle com uma rede ethernet ou de tecnologia BPL (do inglês, Broadband over power lines) [6]. O LED gera o sinal luminoso que se transmite pelo canal óptico.

Em geral, os receptores são compostos por elementos ópticos como filtros, lentes, fotodetectores, amplificadores e circuitos de recuperação do sinal, módulos para conexão com a rede Ethernet ou BPL. O diodo PIN, o fotodiodo de silicone e o fotodiodo de avalanche são usados em VLC [2]. A filtragem é importante para eliminar componentes DC como luz solar e outras fontes de luz [6].

O sinal luminoso que se transmite pelo canal óptico é detectado por um fotodiodo, e então é convertido em fotocorrente. Os processos de modulação, codificação de linha e codificação de canal são revertidos, respectivamente, pela demodulação, decodificação de linha e decodificação de canal. Nesta última etapa, o objetivo é tentar recuperar a informação original.

2.1.1. A utilização de LEDs

A evolução do LED o tornou mais confiável e mais eficiente em termos de potência e luminosidade. Este pode ser branco, como um LED RGB (do inglês, Red

Green Blue), que permite três fluxos de dados independentes, ou azul, encapsulado com uma camada de fósforo, que é mais lento para transmissão [6].

Ilustrado na *Figura 4* abaixo, o LED tricromático RGB é o mais comum para a geração de luz branca, podendo vir em três *chips* ou em um único *chip*. Uma vantagem do LED RGB é a alta largura de banda que resulta em taxas de transmissão mais altas; uma desvantagem é a dificuldade de modulação [2].

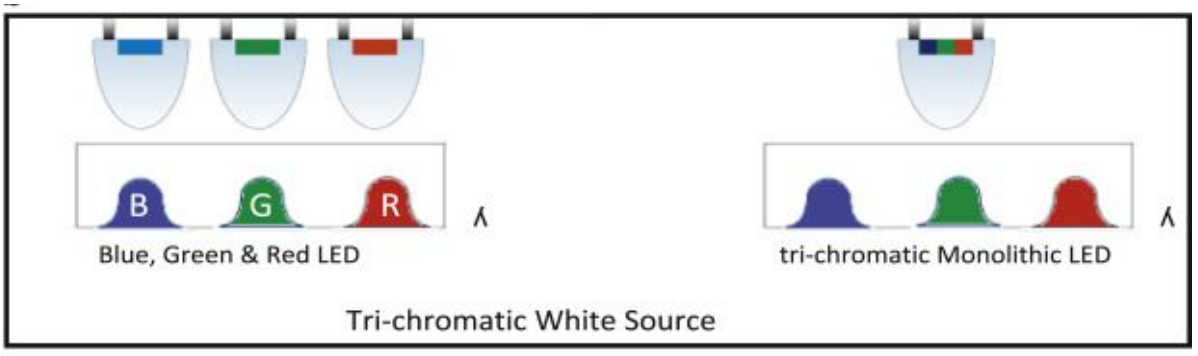

Figura 4: LED tricromático RGB usado para a geração de luz branca.
Fonte: Adaptada de [2]

2.2. Camada de enlace de dados

A camada de enlace possui diversas atribuições, sendo geralmente dividida em subcamadas. Aqui abordamos a subcamada MAC, que possibilita, dentre outros, a mobilidade e a visibilidade dos dispositivos, associações confiáveis entre estes e, especificamente em relação a VLC, o fator de escurecimento e esquemas para mitigar a cintilação [4].

As topologias suportadas pela subcamada MAC são ilustradas na *Figura 5* abaixo, sendo essas: ponto-a-ponto, estrela, e broadcast. Nas topologias ponto-a-ponto e estrela, a comunicação é bidirecional.

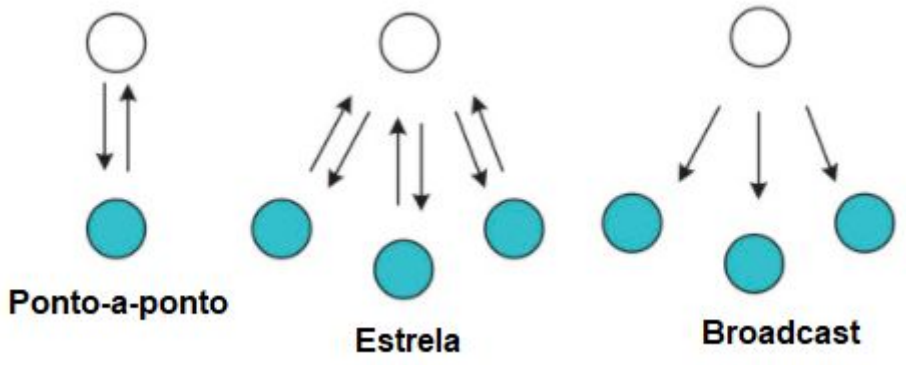

Figura 5: Topologias suportadas pela camada MAC
Fonte: Adaptada de [4]

1. Ponto-a-ponto: A topologia ponto-a-ponto envolve um dispositivo atuando como coordenador para o enlace entre dois dispositivos, sendo que a comunicação entre si é possível já que ambos apresentam enlaces de subida e descida.

2. Estrela: Nesta topologia, muitos dispositivos clientes se conectam a um dispositivo coordenador. Um uso típico é feito nas redes VLC de acesso sem fio.

3. Broadcast: Diferentemente daqueles na topologia estrela, os dispositivos em uma topologia broadcast podem apenas receber dados do transmissor. É uma topologia que simplifica o projeto da camada MAC ao não possibilitar comunicação bidirecional.

Capítulo 3

Codificação de canal

(O capítulo é fruto de um trabalho conjunto com Pedro H. de S. Pozelli.)

3.1. Introdução aos códigos corretores de erro

Os códigos de bloco binários formam grupos de bits chamados “palavras código”, a partir da codificação de blocos de bits da informação a ser transmitida, processo conhecido como “codificação de canal”. Na transmissão de uma palavra código por um meio físico, podem ocorrer erros, isto é, a palavra código recebida pode ter bits diferentes daqueles presentes na palavra original. Assim, a codificação de canal acrescenta redundância à informação na forma de bits extras, chamados bits de verificação, possibilitando a verificação de erros e, consequentemente, tornando mais eficiente a transmissão da informação[7].

A codificação de canal desempenha papel importante no controle do fator de escurecimento. Por exemplo, na já mencionada modulação OOK, um LED é desligado ou ligado de acordo com cada bit de uma palavra código; geralmente, o estado ligado é representado pelo bit 1, e o estado desligado pelo bit 0.

Os chamados “símbolos de compensação” são bits inseridos na transmissão de tal forma que se pode regular o fator de escurecimento ao nível que for necessário, pois este é consequência da porcentagem de bits 1 em uma palavra código.

Em relação à notação, usa-se aqui a representação $(n,\ k)$ para os códigos de bloco binários, sendo k o número de bits do bloco de informação a ser transmitido (também chamado “comprimento da mensagem”), n o número de bits na palavra código (também chamado “comprimento da palavra código”), e $m = n - k$ o número

de bits de verificação. Assim, cada uma das 2^k palavras de mensagem pode ser representada por uma das 2^n palavras código disponíveis.

A taxa de um código, Rc, é dada por:

$$R_C = \frac{k}{n} , \quad (1)$$

sendo tanto menor quanto mais redundância for introduzida.

A taxa efetiva de um código é dada por:

$$R_A = \frac{k}{n+n_C} , \quad (2)$$

em que n_C é a quantidade de símbolos de compensação.

A capacidade de uma família de códigos é sua taxa máxima (isto é, quando o comprimento n tende a infinito) e é geralmente representada pela letra C.

Para que seja possível detectar a presença de até t bits transmitidos incorretamente nas palavras código, cada uma destas deve ser codificada de uma forma que haja, em relação às outras, no mínimo $t+1$ bits diferentes; desta forma, mesmo que uma palavra código recebida contenha até t bits incorretos, esta ainda não será igual a nenhuma das outras. O número de bits distintos entre duas palavras de um código é a distância entre estas palavras. Para que qualquer padrão de t bits incorretos possam ser corrigidos, a menor distância entre quaisquer duas palavras do código precisa ser então no mínimo igual a $2t + 1$. Esta distância é chamada de distância mínima e é denotada por d_{min}.

Uma matriz de verificação de paridade, também chamada matriz H, pode ser montada de tal forma que a multiplicação de qualquer palavra do código pela transposta desta matriz seja igual a um vetor nulo, de forma que, quando o resultado for diferente do valor nulo, a presença de erros possa ser detectada. Ou seja,

$$H^T x \neq 0 \ , \tag{3}$$

em que x é qualquer palavra código. Neste caso, algum critério deve ser definido para que o receptor decida qual é a palavra mais provável de ter sido transmitida. Havendo k bits de informação em cada palavra código de comprimento n, a matriz H contém $(n-k)$ linhas e n colunas.

Para cada matriz H corresponde uma matriz G chamada de geradora do código. A palavra código x pode ser obtida através da multiplicação da palavra mensagem u (com k bits) pela matriz G, ou seja,

$$uG = x$$

3.2. Códigos Reed-Muller

Os códigos Reed-Muller são códigos lineares com notação RM(r, m), sendo r a ordem e 2^m o comprimento, com m positivo e $0 \leq r \leq m$. A distância mínima é 2^{m-r}. O número de bits de informação é: $k = 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r}$. Um código RM($r$, m) binário consiste de todas as combinações lineares de funções booleanas em m variáveis que são monômios com grau menor ou igual a r. Dessa forma, os códigos RM são lineares mas não são cíclicos [8].

Os códigos RM(1, m) são códigos binários que podem ser alcançados recursivamente. O código RM (r, m) é obtido a partir do código RM (1, $m-1$).

Assim, a partir de RM (1, 1), todos os códigos RM(1, m) podem ser gerados.

O conceito de matriz geradora é uma matriz que dá origem a todas as palavras código possíveis através de combinações lineares de suas linhas.

Uma matriz geradora de RM (1, 1) é

$$G = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{4}$$

A matriz geradora G_{m+1} para o código RM ($1, m+1$) é formada a partir da matriz geradora G_m do código RM (1, m) da seguinte forma:

$$G_{m+1} = \begin{bmatrix} G_m & G_m \\ 0\ldots0 & 1\ldots1 \end{bmatrix} \tag{5}$$

assim, por exemplo, a matriz geradora de RM (1, 3) obtida será

$$G_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{6}$$

3.3. Códigos LDPC

Os códigos LDPC devem seu nome a sua matriz de verificação de paridade ser esparsa, contendo poucos valores diferentes de zero. Um código LDPC pode ser

denotado por (n, w_c, w_r), em que n é comprimento do bloco, w_c é o número de uns em uma coluna, e w_r é o número de uns em uma linha da matriz H.O código é chamado regular pois w_c e w_r são constantes nas colunas e linhas. Em [11] e [13], é mostrado que matrizes irregulares muitas vezes são usadas na busca por maior desempenho.

Enquanto a esparsidade da matriz H possibilita que a decodificação seja eficiente, sua aleatoriedade melhora a capacidade de correção de erros [12].

A codificação LDPC envolve duas ideias principais: (1) Construir uma matriz de verificação esparsa; e (2) gerar palavras código a partir dessa matriz. Esses códigos exibem alto desempenho, permitindo uma boa relação de compromisso entre desempenho e complexidade de decodificação. Por outro lado, seus métodos de codificação geralmente apresentam complexidade com relação quadrática sobre o comprimento do bloco, apresentando então maior custo em termos de recursos computacionais.

Em [12], um algoritmo para a construção de codificações LDPC eficientes foi desenvolvido, sendo a eficiência do código gerado devida à esparsidade da matriz H. O codificador realiza permutações de linhas e colunas de forma a levar a matriz H aproximadamente à forma triangular inferior.

Alguns autores propõem métodos de codificação e decodificação em tempo linear; por exemplo, em [13] é sugerido forçar a matriz de verificação de paridade a ter forma triangular inferior, garantindo complexidade de codificação em tempo linear, porém, perdendo em desempenho.

Uma subclasse dos códigos LDPC, com desempenho satisfatório, é formada pelos códigos RA (do inglês Repeat-Accumulate) podem ser gerados de forma simples, como descrito em [14].

Capítulo 4

Codificação de linha

4.1. Sequências (d, k)

O termo run-length se refere ao comprimento de uma sequência de um mesmo valor de bit. Assim, para uma sequência RLL (do inglês, Run-Length Limited), quaisquer subsequências de bits iguais têm seus comprimentos mínimo e máximo limitados. Uma sequência de bits que contenha pelo menos d e no máximo k zeros entre dois uns consecutivos constitui uma sequência de bits de um código (d, k), isto é, os limites mínimos e máximo de *run-lengths* para o bit zero são, respectivamente, d e k [17].

Para implementar uma sequência (d, k), pode ser usada a máquina de estados finitos ilustrada na *Figura 6,* cuja implementação pode ser feita por meio de registradores de deslocamento, e cujos estados são indicados por s. A seta tracejada indica uma continuidade do processo até que o *run-length* seja adequado, o que significa que no mínimo d e no máximo k zeros serão gerados antes que o bit um possa ou deva ser gerado.

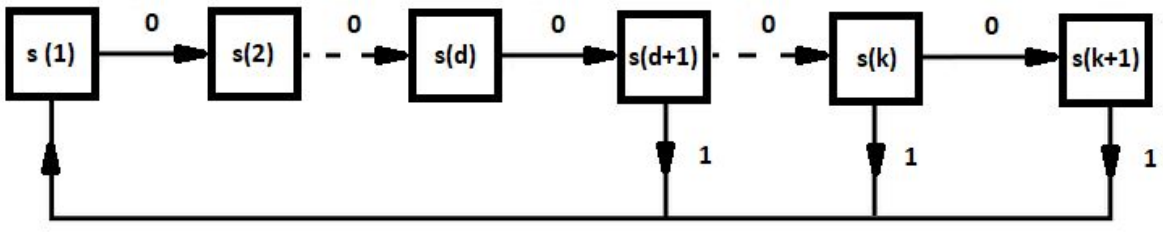

Figura 6: Máquina de estados que gera uma sequência (d, k).

Fonte: Adaptada de [17]

No estado $s(d+1)$, um total de d zeros foi gerado. Podem então sair mais zeros (com o limite de k), ou então o um, que a qualquer momento faz a máquina voltar ao estado inicial. Se o estado $s(k+1)$ é atingido, o próximo bit a sair deve ser o um, caracterizando a sequência $(d,\ k)$.

Para um código $(d,\ k)$, são dados $N(n)$, o número de sequências de comprimento n, e $C(d,k)$, a capacidade do código, pelas equações abaixo [18],

$$
\begin{aligned}
N(n) &= 0,\ se\ n < 0 \\
N(0) &= 1 \\
N(n) &= n+1,\ se\ 1 \le n \le d+1 \\
N(n) &= N(n-1) + N(n-d-1),\ se\ d+1 \le n \le k \\
N(n) &= d+k+1-n+\sum_{i=d}^{k} N(n-i-1),\ se\ k < n \le d+k \\
N(n) &= \sum_{i=d}^{k} N(n-i-1),\ se\ n > d+k
\end{aligned}
\tag{7}
$$

$$
C(d,k) = lim\ _{n\to\infty}\ \tfrac{1}{n}\ log\ _2\ N(n) \tag{8}
$$

Alguns valores de capacidade são exibidos na *Tabela 1* abaixo.

Tabela 1: Capacidade em função dos parâmetros (d, k)

k	d=0	d=1	d=2	d=3	d=4	d=5	d=6
2	0,8791	0,4057					
3	0,9468	0,5515	0,2878				
4	0,9752	0,6174	0,4057	0,2232			

5	0,9881	0,6509	0,4650	0,3218	0,1823		
6	0,9942	0,6690	0,4979	0,3746	0,2269	0,1542	
7	0,9971	0,6793	0,5174	0,4057	0,3142	0,2281	0,1335
8	0,9986	0,6853	0,5293	0,4251	0,3432	0,2709	0,1993
9	0,9993	0,6888	0,5369	0,4376	0,3620	0,2979	0,2382
10	0,9996	0,6909	0,5418	0,4460	0,3746	0,3158	0,2633
11	0,9998	0,6922	0,5450	0,4516	0,3833	0,3285	0,2804
12	0,9999	0,6930	0,5471	0,4555	0,3894	0,3369	0,2924
13	0,9999	0,6935	0,5485	0,4583	0,3937	0,3432	0,3011
14	0,9999	0,6938	0,5495	0,4602	0,3968	0,3478	0,3074
15	0,9999	0,6939	0,5501	0,4615	0,3991	0,3513	0,3122
∞	1,000	0,6942	0,5515	0,4650	0,4057	0,3620	0,3282

Fonte: Adaptada de [17]

4.2. Geração de sequências (d, k)

Na construção de uma sequência (d, k), seus blocos de entrada e saída podem ter tamanho fixo ou variável. O primeiro caso é mais simples, e será tratado em um primeiro momento.

Em uma codificação de tamanho fixo, k bits de entrada dão origem a n bits de saída. Definido o tamanho do bloco de saída n, é possível obter 2^n combinações, porém, estas devem ser filtradas, pois nem todas vão atender às restrições (d, k) escolhidas. Também, algumas sequências são descartadas por se notar que quebram a regra quando concatenadas com outras.

Ao se escolherem os valores dos parâmetros de tamanho de bloco, k e n, e as restrições $(d,\ k)$, a eficiência do código, Ef, é obtida pela razão entre a taxa de código R_C utilizada e a capacidade $C(d,\ k)$, ou seja,

$$Ef = \frac{R_C}{C(d, k)} \tag{9}$$

4.2.1. Códigos de bloco de tamanho fixo

A seguir é descrito um exemplo, utilizando $(d,k) = (0,2)$, k*=2, n=3. Um total de $2^3 = 8$ sequências são possíveis: $000, 001, 010, 011, 100, 101, 110, 111$. (k*: Este se refere ao tamanho do bloco de entrada.)

Como há sequências que terminam e/ou começam com zero, no processo de concatenação haveria violação das restrições $(d,\ k)$ (mais que dois zeros consecutivos), portanto, após uma filtragem devem sobrar as seguintes sequências: $010,\ 011, 101, 110, 111$.

Em seguida, quatro das sequências são relacionadas às quatro entradas de dois bits (tamanho do bloco de entrada), como resumido na *Tabela 2* abaixo, tendo-se então $C(0,2) = 0,8791$, $R_C = \frac{2}{3}$ e $Ef = 0,76 = 76\%$.

Tabela 2: Codificação (d, k) para d=0, k=2, n=3

Entrada	Saída
00	010
01	011
10	110
11	111

Um novo exemplo é dado para $d = 0,\ k = 5,\ n = 5$. Trinta e duas sequências são possíveis, mas, por inspeção visual, pode se verificar que apenas 27 não violam as restrições mesmo quando concatenadas. Um máximo de 4 bits de entrada podem ser codificados, utilizando 16 combinações dentre as 27, como mostra a *Tabela 3* abaixo. Para o exemplo, $C(0,5) = 0,9881$, $R_C = \frac{4}{5}$ e $Ef = 0,81 = 81\%$.

Tabela 3: Codificação para d=0, k=5, n=5.

Entrada	Saída Codificada	Entrada	Saída Codificada
0000	00100	1000	10010
0001	01100	1001	11010
0010	00010	1010	10110
0011	01010	1011	11110
0100	00110	1100	01001
0101	01110	1101	00101
0110	10100	1110	01101
0111	11100	1111	11111

4.2.2. Códigos de bloco de tamanho variável

O exemplo a seguir demonstra um código de tamanho de entrada variável e tamanho de saída fixo, com $d = 0,\ k = 2, n = 2$.

Utiliza-se uma máquina de estados, para que o codificador faça distinção entre o 0 isolado, o que precede o 1 (em 01) e o que sucede o 1 (em 10). Assim, têm-se as relações: 0→01**,** 10→10**,** 11→11**.** Para um melhor entendimento, um diagrama de estados foi construído na *Figura 7*.

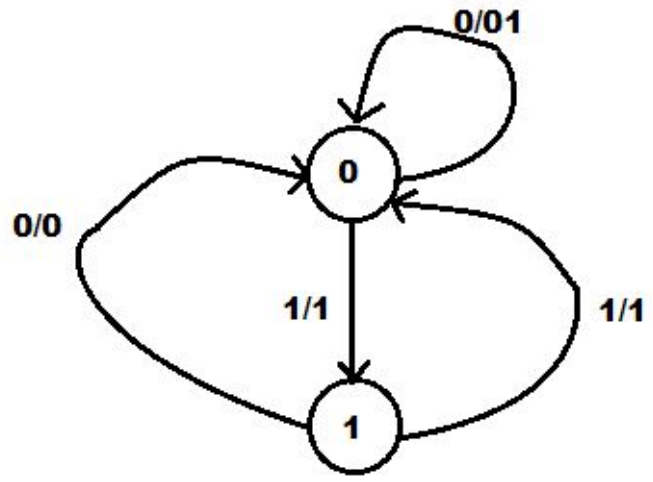

Figura 7 : Máquina de estados para um código (d, k) de blocos de tamanho variável.

Este modelo consiste em utilizar dois estados, 0 e 1, acionados em casos específicos. Quando ocorrer bit 0: se o estado atual é o zero, envia-se 01 para a saída; se o estado atual é o um, envia-se 0 para a saída e ocorre transição para o estado zero. Quando ocorrer 1: se o estado atual é o zero, envia-se 1 para a saída e ocorre transição para o estado um; se o estado atual é o um, envia-se 1 para a saída e ocorre transição para o estado zero.

Foi realizada uma implementação em linguagem de programação, para demonstrar seu funcionamento. Uma aplicação para o caso de entrada contínua também é possível, e foi implementada por meio de flip-flops JK. Essas implementações se encontram na seção 6.1.

Capítulo 5

Suporte ao fator de escurecimento e à mitigação da cintilação

(O capítulo é fruto de um trabalho conjunto com Pedro H. de S. Pozelli.)

Se, por um lado, a quantidade de uns numa palavra código representa a intensidade média de luminosidade dos LEDs, a quantidade de uns sucessivos em uma fração da palavra código representa a intensidade luminosa em um certo intervalo de tempo. Se este for maior que o tempo de persistência dos olhos, a mudança de luminosidade é sentida pelas pessoas expostas, o que deve ser evitado.

Por proposta do padrão IEEE para VLC [9], o suporte ao dimming geralmente é realizado pela inserção de símbolos de compensação, e a mitigação da cintilação dos LEDs é obtida pela codificação de linha RLL, vista no capítulo 4, que evita longas sequências de zeros.

A seguir, são apresentados e comparados alguns esquemas que dão suporte ao fator de escurecimento e/ou à mitigação da cintilação.

5.1. Esquema típico para VLC com suporte ao fator de escurecimento

A *Figura 8* ilustra um esquema típico para VLC com suporte ao fator de escurecimento. Como se observa, k bits definem a mensagem a ser transmitida, **m** , e esta passa por um codificador LDPC (o qual foi explicado na seção 3.3), onde são acrescentados bits de paridade, passando a ter comprimento n', formando a palavra código, **t**. Em seguida, para se evitar a cintilação, t passa pela codificação de linha RLL, dando origem a **y** com comprimento n'' bits. E finalmente, este último recebe a inserção de n_{CS} símbolos de compensação que ajustam o fator de escurecimento, d. Um interleaver (ou entrelaçador) tem por objetivo espalhar os bits, **x**, que são

vizinhos uns dos outros para quebrar a memória do sistema, evitando assim, que no caso de erro, este não troque erroneamente muitos bits vizinhos, corrompendo a informação. Depois de todas estas etapas a informação está pronta para ser transmitida pelo canal óptico através do modulador OOK.

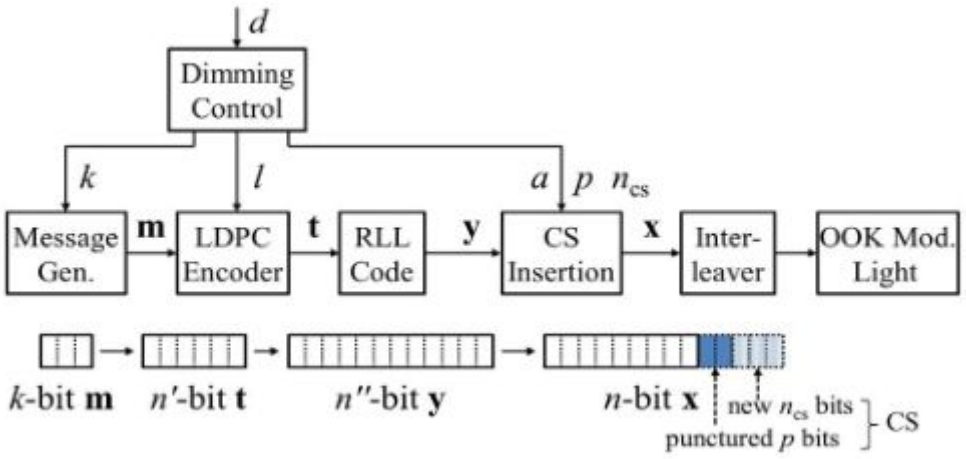

Figura 8: Diagrama de blocos de um típico sistema VLC.
Fonte: Extraída de [10].

5.2. Códigos Reed-Muller modificados

O esquema do sistema de transmissão descrito em [15], cuja proposta é utilizar uma classe de códigos chamada “códigos Reed-Muller modificados”, tem como propriedade produzir a mesma probabilidade de ocorrência de zeros e uns. Assim, menos símbolos de compensação são necessários para suportar diferentes níveis de escurecimento.

Na *Tabela 4* abaixo, esse esquema é comparado com códigos RM de primeira ordem convencionais que contém símbolos de compensação, em que D é a taxa de informação, R_C é a taxa de código antes, e R_A é a taxa de código depois da adição de símbolos de compensação. Na comparação, $m = n - k = 5$.

Como se nota na *Tabela 4*, para obterem os mesmos valores de R_A que os códigos RM convencionais, os códigos RM modificados requerem menos símbolos de compensação, pois têm menor R_C. A tabela ainda mostra a principal propriedade dos códigos RM modificados, a de produzir exatamente a mesma ocorrência de uns e zeros, garantindo que o fator de escurecimento seja 50% sem a inserção símbolos de compensação.

Tabela 4: Códigos Reed-Muller modificados comparado com RM convencionais

Fator de escurecimento (%)	**Convencionais**	**Modificados**	**R_A**	**D (Mbps)**
	R_C	**R_C**		
50	0,313	0,156	0,156	2,344
25 ou 75	0,5	0,250	0,125	1,875
12,5 ou 87,5	0,75	0,375	0,093	1,406

Fonte: Adaptada de [15]

5.3. Códigos polares

Ainda que uma codificação seja eficiente, a eficiência desta e a taxa de transmissão são reduzidas pela utilização de codificação de linha RLL e pela inserção de símbolos de compensação. Levando isso em conta, o esquema de códigos polares descrito em [10] mostra que estes códigos atingem a capacidade de um canal de transmissão binário simétrico e sem memória.

O esquema apresenta as seguintes características: (a) probabilidades iguais para uns e zeros na palavra código; (b) pequeno comprimento de sequência; (c) alta eficiência de codificação alcançada com uma estrutura simples.

O funcionamento do sistema é resumido a seguir e ilustrado na *Figura 9* abaixo. Uma palavra com k bits passa por um codificador de códigos polares para

gerar uma palavra código de n' bits. Na sequência, n_C símbolos de compensação são inseridos para formar a palavra código final com n bits, de acordo com o fator de escurecimento pretendido.

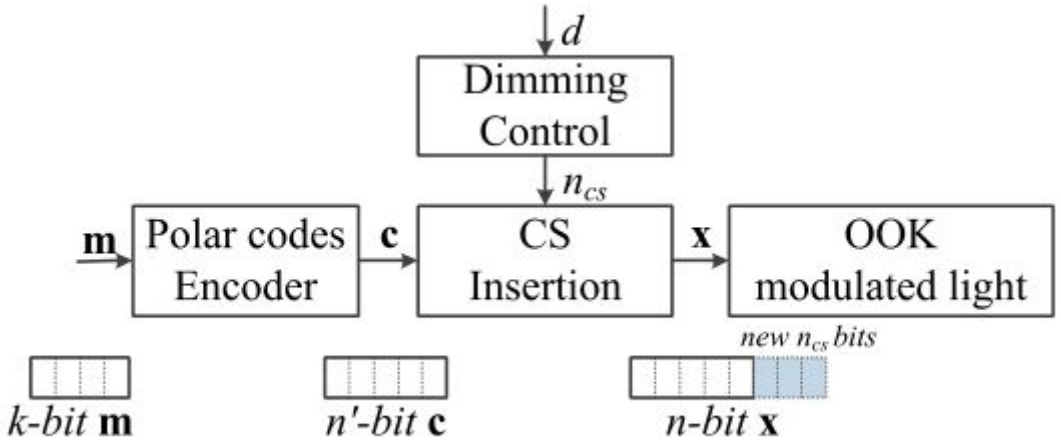

Figura 9: Diagrama de blocos para um sistema VLC com códigos polares.
Fonte: Extraída de [10].

A taxa de código efetiva do esquema é dada por:

$$R_A = \frac{k}{n' + n_C} \tag{10}$$

5.4. Comparação de códigos que suportam fator de escurecimento

A *Tabela 5* a seguir faz uma comparação simplificada de esquemas de codificação para VLC com suporte a fator de escurecimento. Cabe notar que, conhecida na teoria da codificação, *puncturing* é uma técnica que elimina bits de verificação das palavras código, sendo oposta à inserção de símbolos de compensação.

Tabela 5: Comparação de esquemas de codificação que suportam fator de escurecimento

Esquema	**Mitigação da cintilação**	**Suporte a escurecimento**
Códigos RS (IEEE)	Códigos de linha RLL	Símbolos de compensação
Códigos RM	Códigos de linha RLL	Símbolos de compensação
Códigos LDPC	Códigos de linha RLL	Símbolos de compensação e puncturing
Códigos polares	É inerente ao código	Símbolos de compensação

Fonte: Extraída e adaptada de [10]

Em seus testes em [10], o esquema de códigos polares utilizou três taxas de código: 0,25, 0,5 e 0,75, e palavras código com comprimento 2048. Assim, respectivamente, os comprimentos das palavras código e das mensagens, parâmetros (n, k), são: (2048, 512), (2048, 1024) e (2048, 1536).

Em um teste com 10.000 palavras código, foram obtidos os seguintes resultados:

- 99,8% das palavras código têm quantidade de bits 1 no intervalo [972, 1072].
- 0,2% das palavras código têm quantidade de bits 1 nos intervalos [944, 972] ou [1072, 1104].
- Porcentagem média de uns nas palavras código igual a 50%

- Desvio padrão igual a 1,1% do comprimento da palavra código (isto é, 22,55 bits).
- No máximo até 24 bits consecutivos iguais, e isto só ocorreu duas vezes nos 10000 x 2048 bits. Ainda assim, a transmissão de 24 bits (na taxa de 200 kHz) leva apenas 0,12 ms, o que dispensa a preocupação com a cintilação, pois, ainda que todas as palavras código sejam formadas apenas por sequências de 24 bits repetidos, a frequência de transição de luminosidade ainda é 8,333 KHz, o que é muito maior que a frequência de 200 Hz, já considerada segura aos olhos.
- Para a taxa de código 0,5, e fatores de escurecimento de 12,5%, 25% 50%, 75% e 87,5%, a taxa de código do esquema é aproximadamente duas vezes maior que a de um esquema LDPC e a de um esquema RS.
- Para taxas de código 0,75, a eficiência de código é aproximadamente três vezes maior que a de um esquema LDPC.

A *Tabela 6* abaixo descreve as comparações acima apresentadas. Nesta, n_C indica a quantidade símbolos de compensação inseridos, sendo então $n_{TOTAL} = n + n_C$.

Tabela 6: Comparação entre códigos RM, RS, LDPC, e polares

Escureci-mento	**Esquema**	**k**	**n**	n_C	**n_{TOTAL}**	**k / n**	**k / n_{TOTAL}**
50%	Códigos RM	5	32	0	32	0,156	0,156
	Códigos RS	32	64	64	128	0,5	0,25
	Códigos LDPC	576	1152	1152	2304	0,5	0,25
	Códigos polares	512	2048	0	2048	0,25	0,25
		1024				0,5	0,5
		1536				0,75	0,75
	Códigos RM	4	16	16	32	0,25	0,125

25% ou 75%	Códigos RS	32	64	192	256	0,5	0,125
	Códigos LDPC	288	576	1728	2304	0,5	0,125
	Códigos polares	256	1024	1024	2048	0,25	0,125
		512				0,5	0,25
		768				0,75	0,375
12,5% ou 87,5%	Códigos RM	3	8	24	32	0,375	0,094
	Códigos RS	32	64	448	512	0,5	0,063
	Códigos LDPC	144	288	2016	2304	0,5	0,06
	Códigos polares	128	512	1536	2048	0,25	0,063
		256				0,5	0,125
		384				0,75	0,188

Fonte: Adaptada de [10]

5.5. Esquema de codificação sem símbolos de compensação e códigos RLL

Em [22] é descrito um esquema de codificação e modulação que se aproxima da capacidade de um canal VLC com restrição no fator de escurecimento. A codificação é realizada com um codificador LDPC binário e o modulador utiliza modulação PAM (do inglês, Pulse Amplitude Modulation) com M níveis. A aproximação da capacidade de canal é feita através da geração de níveis com probabilidades diferentes. Assim, um mapeador “vários para um” é utilizado entre o codificador e o modulador.

A *Figura 10* mostra um diagrama de blocos do esquema e a *Tabela 7* mostra um exemplo de mapeamento para 2-PAM. Os k bits na entrada do codificador são igualmente prováveis e, sendo o codificador linear, os n bits na sua saída também serão. O mapeador recebe estes n bits e separa em grupos de três bits. Dos oito possíveis grupos, apenas um é mapeado no nível zero e os outros sete são mapeados no nível 1. Isto resulta num fator de escurecimento de 12,5 %.

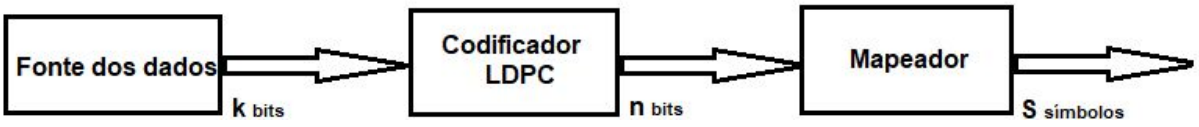

Figura 10: Diagrama de blocos de um sistema M-PAM codificado com níveis não equiprováveis.

Fonte: Adaptada de [22]

Tabela 7: Mapeamento M-PAM para k *=3, M=2,* $p_0 = \frac{1}{8}$ *e* $p_1 = \frac{7}{8}$ *.*

Combinação	Símbolo
0,0,0	**0**
0,0,1	
0,1,0	
0,1,1	**1**
1,0,0	
1,0,1	
1,1,0	
1,1,1	

Fonte: Adaptada de [22]

Comparando a *Figura 10* com a *Figura 9* podemos perceber que o esquema de [22] não necessita de símbolos de compensação para controlar o fator de escurecimento: o controle é realizado pelo mapeador que resulta em níveis da modulação com probabilidades distintas. O mapeamento “vários para um” implica num aumento da complexidade do processo de decodificação no receptor.

Veremos no capítulo 6 que os códigos LDPC também possuem a propriedade inerente de mitigação da cintilação. Sendo assim, não é necessário introduzir codificação RLL.

Capítulo 6

Resultados, discussões e conclusões

(O capítulo é fruto de um trabalho conjunto com Pedro H. de S. Pozelli.)

6.1. Resultados de compilação de códigos (d, k)

Como mencionado na seção 4.2, foram realizadas implementações para demonstrar o funcionamento dos modelos de geração de sequência (d, k) com blocos de código de tamanho fixo e de tamanho variável.

Duas das implementações foram feitas em linguagem de programação C. Os programas em C dos *Apêndices I e II* foram executados e os resultados são mostrados na *Tabela 8*.

Tabela 8: Geração de sequência (d, k) com blocos de código de tamanho fixo e variável.

Código	Entrada	Saída
Tamanho fixo (d=0, k=5, n=5)	0 0 0 0 0 0 1 0 1 0 0 0	00100 00010 10010
Tamanho variável (d=0, k=2)	1 0 0 0 1 1 1 1 1 1 1 0	1 0 01 01 1 1 1 1 1 1 1 0

Tendo em vista que o máximo de zeros consecutivos para os códigos é 5 e 2 respectivamente, pode-se dizer que o código obteve sucesso ao quebrar as sequências de zero que foram propositalmente inseridas na entrada.

A terceira implementação foi por meio de flip-flops JK. Para ser possível trabalhar com taxas de código diferentes, uma variável auxiliar S foi utilizada. A tabela verdade construída para este modelo se encontra na *Tabela 9*.

Tabela 9: Tabela verdade construída para o modelo de entrada contínua.

S	Estado	Entrada	Saída	Próximo estado	Próximo S
0	0	0	0	0	1
0	1	0	0	0	0
0	0	1	1	1	0
0	1	1	1	0	0
1	X	X	1	0	0

A notação “X” indica que o campo pode admitir qualquer valor (0 ou 1).

A lógica da *Tabela 9* engloba a mesma ideia do diagrama de estados da *Figura 7*, porém, com a adição da variável auxiliar S, que tem por função enviar para a saída o bit “1” no caso da entrada “0”.

O próximo passo para chegar no modelo desejado é parametrizar os valores que assumem as variáveis de saída, próximo estado, e próximo S. Em seguida, relaciona-se isto com as entradas J e K do flip-flop JK. Logo, é preciso que mais colunas sejam adicionadas à tabela original para analisar cada caso.

Na *Tabela 10*, Js e Ks correspondem a J e K do primeiro flip-flop; Jest e Kest correspondem a J e K do segundo flip-flop; Jent e Kent correspondem a J e K do terceiro flip-flop. Assim, as variáveis J e K estão associadas cada uma a um flip-flop. O mecanismo de decisão dos valores de J e K funciona a partir da comparação do valor atual da variável em questão com o seu valor futuro.

Tabela 10: Tabela verdade do Exemplo 3 expandida para contemplar J e K.

S	Est	Ent	Saí	Próx. estado	Próx. S	Js	Ks	Jest	Kest	Jent	Kent
0	0	0	0	0	1	1	X	0	X	0	X
0	1	0	0	0	0	0	X	X	1	0	X
0	0	1	1	1	0	0	X	1	X	X	0
0	1	1	1	0	0	0	X	X	1	X	0
1	0	0	1	0	0	X	1	0	X	1	X
1	0	1	1	0	0	X	1	0	X	X	0
1	1	0	1	0	0	X	1	X	1	1	X
1	1	1	1	0	0	X	1	X	1	X	0

Partindo-se da transição de bit que ocorre e da tabela verdade de um flip-flop JK na *Tabela 11*, J e K são escolhidos para gerar a resposta requerida, como mostrado abaixo:

$$0 \rightarrow 1 : \ J = 1, \ K = X$$
$$0 \rightarrow 0 : \ K = 0, \ K = X$$
$$1 \rightarrow 0 : \ J = X, \ K = 1$$
$$1 \rightarrow 1 : \ J = X, \ K = 0$$

Tabela 11 : Tabela verdade de um flip-flop JK.

J	K	Clock	Saída
0	0	Subida	Inalterada
0	1	Subida	Q=0
1	0	Subida	Q=1
1	1	Subida	Comuta

Fonte: Adaptada de [19]

Pelo método do mapa de Karnaugh [20], obtêm-se as equações simplificadas:

$$Js = NOT\ (Estado\ OR\ Entrada);$$

$$Ks = 1;$$

$$Jest = NOT(S)\ AND\ Entrada;$$

$$Kest = 1;$$

$$Jent = S;$$

$$Kent = NOT(Ent)\ .$$

Assim, projetou-se o modelo na *Figura 11* abaixo na ferramenta Falstad [21]:

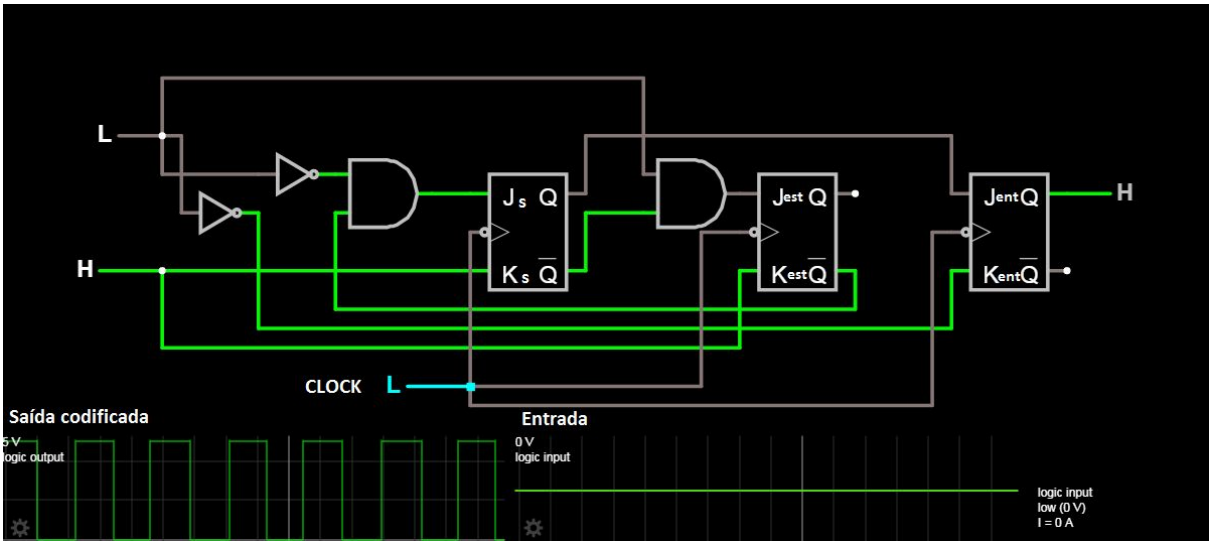

Figura 11: Implementação de sequência (d, k) com taxa de código variável.

Com a taxa de código variável, se as três diferentes entradas forem equiprováveis, calcula-se a eficiência média, $Ef(média)$, deste código por:

$$Ef(média) = \frac{Rc(média)}{C(d,k)} \tag{11}$$

Assim, o controle da cintilação é possível através de códigos RLL do tipo (d,k), seja por blocos de tamanho fixo ou variável, com diferentes taxas e eficiências. No caso de blocos variáveis, mostrou-se que sua eficiência pode ser bem superior à de tamanho fixo, porém com maior complexidade de projeto e implementação.

6.2. Geração de códigos LDPC

Para a geração de palavras código de códigos LDPC, inicialmente procurou-se entender os algoritmos para a construção dos mesmos. Em resultado de estudos de [12], foi criado um código que está no *Apêndice III*, com base no método “Low-Triangular Form” não aproximado.

A partir de uma matriz de verificação de paridade esparsa H que seja fornecida, um vetor de bits de entrada pode ser codificado. Os valores de n, k, m, e de todos os outros necessários são calculados a partir da matriz H e do vetor de bits de entrada. Embora não seja foco deste trabalho, o código também calcula o chamado vetor de síndrome, importante na detecção e correção dos erros ocorridos sobre a palavra código transmitida.

Por exemplo, a entrada da palavra mensagem $m =$ (111) resulta na palavra código $t =$ (111101010101).

Após o entendimento inicial da criação de códigos LDPC, optou-se por utilizar códigos de programação já existentes, robustos e de livre utilização para a geração desse tipo de código. Os códigos utilizados são encontrados em [16]. Em seguida, foram desenvolvidos códigos de programação com o objetivo de obter histogramas de pesos das palavras-código geradas (mostrados nos *Apêndices IV e V*).

A *Figura 12* abaixo mostra o histograma de pesos para 5.000 palavras-código com parâmetros n=2048, k=1024.

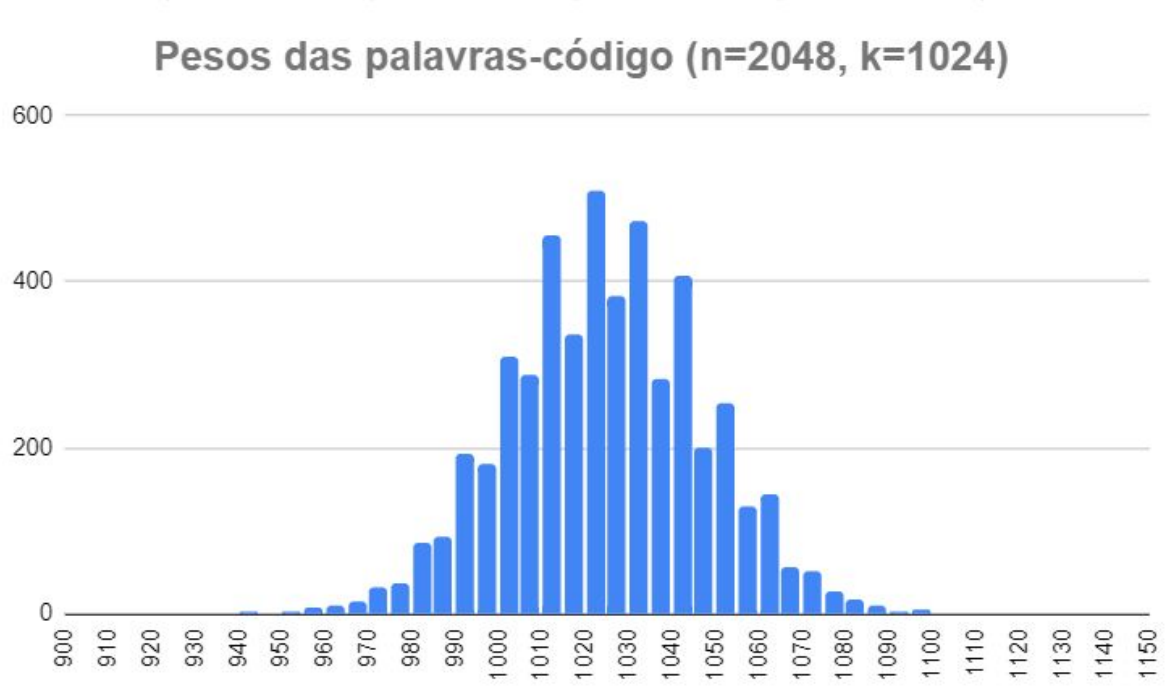

Figura 12: Histograma de pesos das palavras-código geradas do tipo LDPC
Fonte: Do próprio autor

A *Figura 13* abaixo mostra o histograma de pesos para 5000 palavras-código com parâmetros n=10000, k=5000:

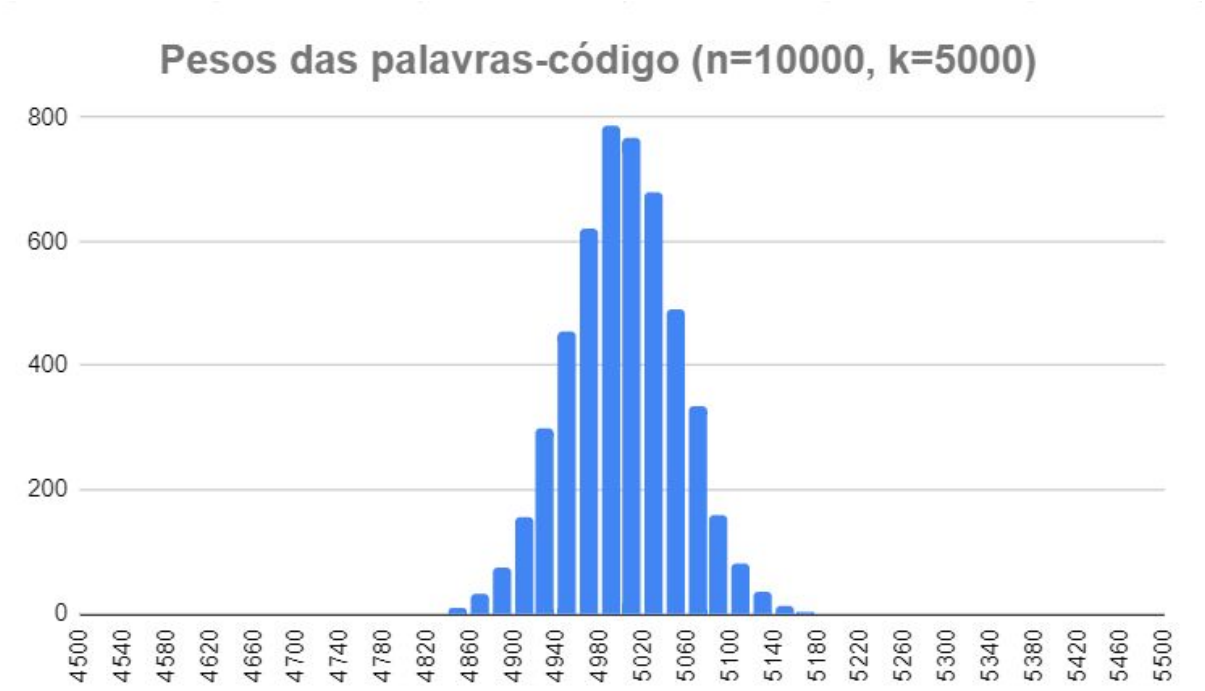

Figura 13: Histograma de pesos das palavras-código geradas do tipo LDPC
Fonte: Do próprio autor

Nos dois histogramas acima, o peso médio das palavras-código é o próprio valor de k, que é a metade do respectivo n. Assim, o fator de escurecimento com média de 50% é garantido.

No primeiro histograma, o menor peso de palavra código é 940, o maior é 1118. Além disso, 78,56% das palavras-código têm peso no intervalo [997, 1.053], estando cerca de 10% abaixo do peso 997 e cerca de 10% acima do peso 1053. No intervalo [973, 1.075], que representa variação entre -5% e +5% em relação ao peso médio 1024, estão 97,52% das palavras-código. O desvio padrão é de 2,20%, equivalente a 23 bits.

No segundo histograma, o menor peso é 4812, o maior peso é 5.216. Além disso, 78,52% das palavras-código têm peso no intervalo [4937, 5063], estando cerca de 10% abaixo do peso 4.937 e cerca de 10% acima do peso 5.063. No intervalo [4900, 5100], que representa variação entre -2% e +2% em relação ao peso

médio 5.000, estão 95,3% das palavras-código. No intervalo [4850, 5150], que representa variação entre -3% e +3% em relação ao peso médio 5000, estão 99,7% das palavras-código. O desvio padrão é de 1,01%, equivalente a 51 bits.

Com base nos resultados acima, podemos agora analisar a cintilação, considerando o pior cenário. Para isso, foi desenvolvido um código em Python (mostrado no *Apêndice VI*) para calcular o máximo de bits repetidos nos 5.000 blocos gerados. Para o teste com $k = 1024$, o máximo de bits repetidos obtido foi igual a 26 bits '1'. Para o teste com $k = 5000$, o máximo de bits repetidos obtido foi igual a 25 bits '0'.

Em uma taxa de 200 kHz, a transmissão de 25 bits leva apenas 0,125 ms, e a transmissão de 26 bits leva apenas 0,13 ms. Com base nisso, a preocupação com a cintilação é dispensada, pois, ainda que todas as palavras código sejam formadas apenas por sequências de 25 ou 26 bits repetidos, a frequência de transição de luminosidade ainda é de 7,7 kHz (para 26 bits) a 8 kHz (para 25 bits), o que é muito maior que a frequência de 200Hz que, conforme já mencionado, é considerada segura à saúde.

6.3. Conclusões

Os LEDs mostram ser a tendência para o futuro da iluminação de uma forma geral, substituindo os atuais modelos de lâmpadas, por serem mais vantajosos tanto no sentido econômico como no energético. Apresentam ainda rapidez de transição entre os estados ligado e desligado, o que é altamente vantajoso para as comunicações por luz visível.

O controle do fator de escurecimento é importante em um sistema VLC porque propicia o ajuste do nível de luminosidade em um ambiente, o que tem relação até mesmo com a saúde e o conforto dos indivíduos expostos à luz. A

cintilação, por sua vez, pode ser controlada ou até mesmo eliminada com a utilização de códigos (d, k) e, posteriormente, o equivalente RLL, seja por blocos de tamanho fixo ou variável, diferenciando-se na taxa de código e eficiência. Tal preocupação deve ser encorajada, uma vez que estudos do IEEE provam que há riscos à saúde, seja pela exposição curta ou prolongada.

As simulações para a análise dos códigos LDPC mostraram que, assim como os códigos polares, esses podem ser implementados para apresentar fator de escurecimento desejado ao mesmo tempo que dispensam a preocupação com a cintilação, o que sempre deve ser considerado.

Como comentado na seção 5.1, um esquema [22] que realiza o chamado “mapeamento vários para um” torna possível ajustar o fator de escurecimento para valores diferentes de 50%, sem a necessidade de símbolos de compensação. Os autores deste trabalho pensam em estudar a utilização desse esquema em trabalhos futuros.

REFERÊNCIAS

[1] Rajagopal, S. “IEEE 802.15.7 visible light communication: modulation schemes and dimming support,” IEEE Communications Magazine, p. 72-82, março 2012.

[2] Khan, L. ”Visible light communication: Applications, architecture, standardization and research challenges,” Digital Communications and Networks, Vol. 3, 2° Edição, p. 78-88, maio 2017.

[3] Komine, T.; Nakagawa, M. “Fundamental Analysis for Visible Light Communication System using LED lights,” IEEE Transactions on Consumer Electronics, p. 100-107, junho 2004.

[4] Pathak, P. et al., “Visible Light Communication, Networking and Sensing: A Survey, Potential and Challenges,” IEEE Communications Surveys & Tutorials Vol. 17, 4° Edição, Quarto trimestre de 2015.

[5] Instituto de Engenheiros Eletricistas e Eletrônicos,“IEEE Recommended Practices for Modulating Current in High-Brightness LEDs for Mitigating Health Risks to Viewers,” IEEE, junho 2015.

[6] Junior, H. “Comunicação de Dados Utilizando Luz Visível,” Revista de Engenharia da Universidade Católica de Petrópolis, 2011.

[7]: Lathi, B. “Modern Digital and Analog Communication Systems,” Imprensa da Universidade de Oxford, 3° edição, Cap. 16, 1998.

[8]: Moon, T. “Error Correcting Coding: Mathematical Methods and Algorithms,” New Jersey: John Wiley & Sons Inc., p. 375-392, maio 2005.

[9] Instituto de Engenheiros Eletricistas e Eletrônicos, “IEEE Standard for Local and Metropolitan Area Networks–Part 15.7, Short-Range Wireless Optical Communication Using Visible Light,” IEEE, setembro. 2011, p. 1–309.

[10] Fang, J. et al. “An Efficient Flicker-Free FEC Coding Scheme for Dimmable Visible Light Communication Based on Polar Codes,” IEEE Photonics Journal, Vol. 9, Num. 3, p. 1-10, junho 2017.

[11] Tu, Z.; Zhang, S. “Overview of LDPC Codes,” IEEE, p. 469-474, 2007.

[12] Richardson, T.; Urbanke, R. “Efficient Encoding of Low-Density Parity-Check Codes,” Vol. 47, Num. 2, p. 638-656, fevereiro 2001.
[13] MacKay, D.; Wilson, S.; Davey, M. “Comparison of constructions of irregular Gallager codes,” Vol. 47, Num. 10, págs 1449-1454, outubro 1999.
[14] MacKay, D. “Information Theory, Inference and Learning Algorithms,” Imprensa da Universidade de Cambridge, Cap. 49, setembro 2003.
[15] Kim, S. S. Jung, “Novel FEC Coding Scheme for Dimmable Visible Light Communication Based on the Modified Reed–Muller Codes,” IEEE Photonics Technology Letters, Vol. 23, Num. 20, p. 1514-1516, outubro 2011.
[16] Neal, R. “Software for Low Density Parity Check Codes,” Disponível em: <https://www.cs.utoronto.ca/~radford/ftp/LDPC-2012-02-11/index.html>. Último acesso em 28 out. 2019.
[17] Proakis, J.; Salehi, M. “Digital Communications,” McGraw-Hill Education, 4° Edição, p. 568-576, agosto 2000.
[18] Immink, K. “Run length-limited sequences,” Proceedings of the IEEE, dezembro 1990.
[19] Flip flop JK com clock. Disponível em: <http://eletronworld.com.br/eletronica/flip-flop-jk-com-clock/>. Último acesso em 01 nov. 2019.
[20] Rushdi, A. “Karnaugh Map,” Encyclopaedia of Mathematics, Volume Suplementar I. 1., janeiro 1997.
[21] Falstad circuits. Disponível em: <https://www.falstad.com/circuit/circuitjs.html>. Último acesso em 20 out. 2019.
[22] Belli, R.; Runge, C.; Portugheis, J. “Esquema de codificação para sistema M-PAM VLC com escurecimento,” XXXVI Simpósio Brasileiro de telecomunicações e processamento de sinais, p. 809-813, setembro 2018.

Apêndices

Implementações dos exemplos citados no Capítulo 4 para sequências (d, k).

I. Sequência (d, k) para blocos de tamanho fixo em linguagem C.

```
#include <iostream>
using namespace std;
#include <stdio.h>
#include <stdlib.h>
int main() {
char entrada[13]="000000101000";
string saida="";
int c=0;
while(c!=12)
{
if(entrada[c]=='0' && entrada[c+1]=='0' && entrada[c+2]=='0'&&
entrada[c+3]=='0')
{
  saida=saida+"00100";
  c=c+4;
}
if(entrada[c]=='0' && entrada[c+1]=='0' && entrada[c+2]=='0'&&
entrada[c+3]=='1')
{
  saida=saida+"01100";
  c=c+4;
}
if(entrada[c]=='0' && entrada[c+1]=='0' && entrada[c+2]=='1'&&
entrada[c+3]=='0')
{
  saida=saida+"00010";
  c=c+4;
}
if(entrada[c]=='0' && entrada[c+1]=='0' && entrada[c+2]=='1'&&
entrada[c+3]=='1')
{
  saida=saida+"01010";
  c=c+4;
}
```

```
if(entrada[c]=='0' && entrada[c+1]=='1' && entrada[c+2]=='0'&&
entrada[c+3]=='0')
{
  saida=saida+"00110";
  c=c+4;
}
if(entrada[c]=='0' && entrada[c+1]=='1' && entrada[c+2]=='0'&&
entrada[c+3]=='1')
{
  saida=saida+"01110";
  c=c+4;
}
if(entrada[c]=='0' && entrada[c+1]=='1' && entrada[c+2]=='1'&&
entrada[c+3]=='0')
{
  saida=saida+"10100";
  c=c+4;
}
if(entrada[c]=='0' && entrada[c+1]=='1' && entrada[c+2]=='1'&&
entrada[c+3]=='1')
{
  saida=saida+"11100";
  c=c+4;
}
if(entrada[c]=='1' && entrada[c+1]=='0' && entrada[c+2]=='0'&&
entrada[c+3]=='0')
{
  saida=saida+"10010";
  c=c+4;
}
if(entrada[c]=='1' && entrada[c+1]=='0' && entrada[c+2]=='0'&&
entrada[c+3]=='1')
{
  saida=saida+"11010";
  c=c+4;
}
if(entrada[c]=='1' && entrada[c+1]=='0' && entrada[c+2]=='1'&&
entrada[c+3]=='0')
{
  saida=saida+"10110";
```

```
  c=c+4;
}
if(entrada[c]=='1' && entrada[c+1]=='0' && entrada[c+2]=='1'&&
entrada[c+3]=='1')
{
  saida=saida+"11110";
  c=c+4;
}
if(entrada[c]=='1' && entrada[c+1]=='1' && entrada[c+2]=='0'&&
entrada[c+3]=='0')
{
  saida=saida+"01001";
  c=c+4;
}
if(entrada[c]=='1' && entrada[c+1]=='1' && entrada[c+2]=='0'&&
entrada[c+3]=='1')
{
  saida=saida+"00101";
  c=c+4;
}
if(entrada[c]=='1' && entrada[c+1]=='1' && entrada[c+2]=='1'&&
entrada[c+3]=='0')
{
  saida=saida+"01101";
  c=c+4;
}
if(entrada[c]=='1' && entrada[c+1]=='1' && entrada[c+2]=='1'&&
entrada[c+3]=='1')
{
  saida=saida+"11111";
  c=c+4;
}
cout << entrada;
cout << " \n"<< saida;
return 0;
}
}
```

II. Sequência (d, k) para blocos de tamanho variável em linguagem C.

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;

int main()
{
    int tam=12;
    int c,estado;
    string sai[tam];
    estado=0;//0(estado 1) e 1(estado 2)
    int ent[tam];
    srand(time(0));
    for(c=0;c<tam;c++)
    {
        ent[c]=rand() % 2;
        switch(ent[c])
        {
            case 0:
            if (estado==0)
            sai[c]="01";
            else
        {
            sai[c]="0";
            estado=0;
        }
            break;
            case 1:
            sai[c]="1";
            if (estado==0)
            estado=1;
            else
            estado=0;
            break;
        }

    }
    printf("Entrada:");
```

```
      for(c=0;c<tam;c++)
      {
         cout << ent[c] << " ";
      }
      printf("\n");
      printf("Saida:  ");
      for(c=0;c<tam;c++)
      {
         cout << sai[c] << " ";
      }


    return 0;
}
```

Implementações que dizem respeito à geração de códigos corretores de erros.

III. Código em JavaScript para geração de palavras-código a partir de uma matriz de verificação de paridade

```
print = function (a){
     console.log(a);
     document.write(a+'<br/><br/>');
}
// Matriz H no exemplo 1.16 da página 12 de:
http://sigpromu.org/sarah/SJohnsonLDPCintro.pdf

// Parâmetros que precisam ser definidos: H, s
// Os valores de n, k, m e todos os outros são calculados a
partir de H e S
msg = 'Palavra mensagem: ';
pal = 'Palavra codificada: ';
sind = 'Síndrome: ';

H = [
[1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0],
[0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0],
[1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0],
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1]
];
```

```
s = [1, 1, 1];

n = H[0].length;
k = s.length;
m = n-k;
palavra = new Array(n);
bits_paridade = new Array(m);
sindrome = new Array(m);

function calcula_bit_l(l){
    var soma = 0;

    for(var j=0; j<k; j++)
        soma += H[l][j]*s[j];
    for(var j=0; j<l; j++)
        soma += H[l][j+k]*bits_paridade[j];

    return (soma%2);
}

for (var i=0; i<k; i++)
    palavra[i] = s[i];

for(var l=0; l<m; l++){
    bits_paridade[l] = calcula_bit_l(l);
    palavra[k+l] = bits_paridade[l];
}

for(var a=0; a<m; a++){
    sindrome[a] = 0;
```

```
    for(var b=0; b<n; b++){
        sindrome[a] += H[a][b]*palavra[b];
    }
    sindrome[a] = sindrome[a] % 2;
}

function formata(s, sep){
    return s.toString().split(',').join(sep || '');
}

hstr = '';
for(var i=0; i<H.length; i++)
    hstr += ((i==Math.floor(H.length/2))?'H =
 |':'  |')+formata(H[i], ' ')+'|<br>';
print(hstr);
print(msg+'<br/>'+formata(s));
print(pal+'<br/>'+formata(palavra));
print(sind+'<br/>'+formata(sindrome));
```

IV. Comandos utilizados para a geração dos códigos LDPC

```
set -e  # Encerrar se houver erros

# Código LDPC (2048,1024) com 3 verificações por bit e 6 bits por verificação
./make-ldpc ex-ldpc36-1024a.pchk 1024 2048 1 evenboth 3 no4cycle
./make-gen  ex-ldpc36-1024a.pchk ex-ldpc36-1024a.gen dense
./rand-src  ex-ldpc36-1024a.src 1 1024x5000
./encode    ex-ldpc36-1024a.pchk ex-ldpc36-1024a.gen ex-ldpc36-1024a.src
ex-ldpc36-1024a.enc

# Código LDPC (10.000,5.000) com 3 verificações por bit e 6 bits por verificação
./make-ldpc ex-ldpc36-5000a.pchk 5000 10000 1 evenboth 3 no4cycle
./make-gen  ex-ldpc36-5000a.pchk ex-ldpc36-5000a.gen dense
./rand-src  ex-ldpc36-5000a.src 1 5000x5000
./encode    ex-ldpc36-5000a.pchk ex-ldpc36-5000a.gen ex-ldpc36-5000a.src
ex-ldpc36-5000a.enc
```

V. Código em linguagem Python para cálculo do peso de cada palavra código gerada

```
text = open("ex-ldpc36-XXXXa.enc", "r") # XXXX = 5000 ou 1024
with open('histograma.txt', 'w') as f:
    for line in text:
        peso = len(line.strip().split('1'))-1
        f.write('%d\n' % peso)
```

VI. Código em linguagem Python para calcular o máximo de bits repetidos em uma lista de palavras-código

```
def maxZeros(str):
     l = len(str)
     resultado = 0
     for i in range(l):
          if (str[i]!='0'):
               continue
          contador = 1
          for j in range(i + 1, l):
               if (str[j]!='0'):
                    break
               contador += 1

          if contador > resultado:
               resultado = contador
     return resultado

def maxUns(str):
     l = len(str)
     resultado = 0
     for i in range(l):
          if (str[i]!='1'):
               continue
          contador = 1
          for j in range(i + 1, l):
               if (str[j]!='1'):
                    break
               contador += 1
```

```
            if contador > resultado:
                resultado = contador
        return resultado

maximo = 0
text = open("ex-ldpc36-5000a.enc", "r") # substituir por
"1024" quando necessário

with open('ocorrencias.txt', 'w') as f:
        for linha in text:
            linha = linha.strip()
            maior = max(maxZeros(linha), maxUns(linha))
            if maior>maximo:
                maximo = maior
        print('Maximo de bits repetidos:%d' % maximo)
```