TRANSMISSÃO E RECEPÇÃO DE DADOS ATRAVÉS DE MODEM ACÚSTICO

Gil Alves Magalhães, Luís Pedro D'Avila Zorzi, Vinícius Gabriel Schultz
Curso de Engenharia de Computação, Centro de Tecnologia – Universidade Federal de Santa Maria
Disciplina: Comunicação de Dados
e-mail: gil.magalhaes@ecomp.ufsm.br, luispedro.zorzi@ecomp.ufsm.br, vinicius.schultz@ecomp.ufsm.br

Resumo **–** Este trabalho apresenta o desenvolvimento de um modem acústico capaz de transmitir e receber dados digitais por meio de ondas sonoras. O sistema foi inteiramente implementado em MATLAB/Octave e reproduz as principais etapas de um enlace físico real, incluindo modulação BPSK, filtragem com raised-cosine, sincronização de portadora por meio de Costas Loop e recuperação de amostragem de símbolos. Sobre essa camada física, foram incorporadas funcionalidades da camada de enlace, com foco em detecção e correção de erros utilizando o código Hamming(7,4). No transmissor, a mensagem é convertida para bits, enquadrada com preâmbulo e SFD, codificada via Hamming, filtrada e modulada em banda passante. No receptor, o sinal passa por sincronização de portadora, sincronização de símbolos, demodulação, detecção do quadro por correlação com o SFD e decodificação Hamming, que permite corrigir erros de um bit por bloco. Além da função nativa do Octave, foi implementada manualmente a codificação e decodificação Hamming, garantindo compatibilidade e permitindo comparação entre os métodos. Os resultados experimentais demonstram que o modem é capaz de recuperar corretamente mensagens transmitidas mesmo em condições acústicas adversas, evidenciando a eficácia da correção de erros e a aplicabilidade do sistema como ferramenta educacional e de estudo de comunicação digital.

Palavras-Chave **– Comunicação de dados, detecção de erros, correção de erros, hamming, modem acústico.**

I. INTRODUÇÃO

A comunicação digital é fundamental em praticamente todas as tecnologias modernas, desde redes de computadores até sistemas embarcados e dispositivos móveis. Embora normalmente seja realizada por meios eletromagnéticos ou cabeados, a transmissão de informações utilizando ondas sonoras oferece uma forma acessível e didática de compreender os princípios envolvidos nos sistemas de comunicação. Um modem acústico, portanto, permite explorar modulação, demodulação e os efeitos do canal em um ambiente controlado, utilizando apenas microfones e alto-falantes comuns.

O meio acústico, por sua natureza, está sujeito a ruídos ambientais, reflexões, distorções e atenuação, o que o torna um canal propenso a erros. Essas características o tornam ideal para observar, de maneira prática, problemas típicos que afetam sistemas reais, como perda de sincronismo, interferência e corrupção de bits. Dessa forma, estudar comunicação por áudio possibilita compreender tanto os aspectos físicos da transmissão quanto a importância das camadas superiores do modelo de comunicação.

Além dos fenômenos do canal, sistemas reais dependem de mecanismos de preparação, proteção e recuperação da informação. A camada de enlace de dados desempenha papel essencial nessa tarefa, incorporando técnicas como enquadramento, detecção e correção de erros, controle de fluxo e multiplexação. Implementar e testar essas funcionalidades em um modem acústico permite visualizar como elas aumentam significativamente a confiabilidade da comunicação.

O objetivo deste trabalho é desenvolver e analisar um sistema de comunicação por modem acústico, realizando a transmissão e recepção de dados digitais por meio de ondas sonoras. O estudo inclui a observação dos efeitos do meio acústico sobre o sinal recebido e a implementação de funcionalidades da camada de enlace, com foco nas técnicas de detecção e correção de erros aplicadas no código do receptor.

Detecção e correção de erros são essenciais para aumentar a confiabilidade. Como resultado, devemos empregar algum tipo de código de detecção e correção de erros. Ao transferir dados, esses códigos adicionam um ou mais bits extras aos bits de dados; esses bits extras, chamados de bits de paridade, ajudam na detecção de erros[1]. Quando comparado a outros códigos de controle de erro, o código de Hamming é simples de usar e possui alta eficiência tanto na detecção quanto na correção de erros [1].

Por fim, o trabalho está organizado da seguinte forma: inicialmente são apresentados os princípios do modem acústico junto de suas características de implementação da camada física e de enlace e o funcionamento do transmissor e do receptor. Em seguida, são descritas as modificações propostas em relação ao projeto original do modem. Então são descritas as implementações em código das funções de codificação e decodificação Hamming(7,4). Posteriormente, são apresentados os resultados obtidos e a análise do desempenho do sistema. Por fim, são discutidas as conclusões gerais do estudo.

II. MUDANÇAS NO PROJETO ORIGINAL

O projeto original chamado `modem_acustico` foi fornecido pelo docente e foi codificado em linguagem Matlab/Octave e implementa um sistema de transmissão e recepção de dados por som como meio, utilizando as placas de áudio e microfone do computador pessoal. A decisão se deve pelo fato de que toda a plataforma para a utilização do som é acessível e presente na vida das pessoas, facilitando a etapa de DAC/ADC.

Abaixo, as características da camada física do modem projetado:

- Taxa de Amostragem (F_a): 8000 Hz. Valor padrão para voz digital, suficiente para representar sinais de até 4 kHz.
- Frequência da Portadora (F_p): 2000 Hz. Escolhida por situar-se no centro da faixa de resposta de frequência de áudio, afastada de ruídos de rede elétrica (60 Hz) e de interferências de alta frequência.
- Taxa de Bits (R_B): 100 bps. A escolha de uma taxa baixa visa maximizar a energia por bit (Eb) e minimizar os efeitos de eco e reverberação do ambiente, garantindo maior robustez.
- A modulação empregada é a BPSK (Binary Phase Shift Keying). A implementação utiliza mapeamento polar NRZ (*Non-Return-to-Zero*), onde o bit lógico '1' é mapeado para amplitude +1 (fase 0°) e o bit lógico '0' para amplitude -1 (fase 180°).O sinal passa por um filtro formatador de pulso (*Pulse Shaping*) do tipo Cosseno Levantado para limitar a largura de banda antes da multiplicação pela portadora senoidal.

Quanto à camada de enlace, tem-se o enquadramento do quadro no seguinte formato:

[PREÂMBULO | SFD | TAMANHO | PAYLOAD | PREÂMBULO]

O preâmbulo é composto por 40 bits e utiliza a sequência 0xFFEAAAAAAA. Sua função é permitir que o algoritmo de recuperação de relógio do receptor (sincronização de símbolos) estabilize e ajuste o ganho de entrada (AGC).

O SFD (Start Frame Delimiter) é uma sequência fixa de 32 bits com o valor 0xCE3C38CA. Este campo serve como chave de correlação. O receptor não busca por bits individuais, mas sim pela "assinatura" matemática completa dessa sequência, permitindo encontrar o início exato do quadro mesmo em condições de baixo SNR.

O Campo de Tamanho (Length Byte) é um byte (8 bits) que informa o número de caracteres da mensagem original. A inclusão deste campo dinâmico permite que o receptor realize um "corte cirúrgico" no sinal, processando apenas os dados válidos e descartando o ruído capturado após o fim da transmissão, porém limitado à 255 caracteres.

As modificações possíveis de serem aplicadas no projeto de transmissão se encontram majoritariamente no receptor, porém qualquer mudança nesta etapa reflete também na transmissão dos dados. Portanto, consiste em refinamentos em relação ao original que implementa apenas o necessário para estabelecer uma comunicação.

Na Figura 1 estão indicadas as etapas do sistema de transmissão que estavam implementadas no modem original, além das modificações adicionadas, as quais estão destacadas em azul. No transmissor, foi necessário alterar a codificação para binário utilizada, que anteriormente convertia de ASCII diretamente para binário, enquanto para a nova versão converte utilizando codificação Hamming, colocando três bits de paridade a cada quatro bits de dados, seguindo a seguinte disposição: p1 p2 d1 p3 d2 d3 d4. No receptor foi necessário adicionar a etapa de decodificação Hamming pela qual detecta e corrige os erros de transmissão.

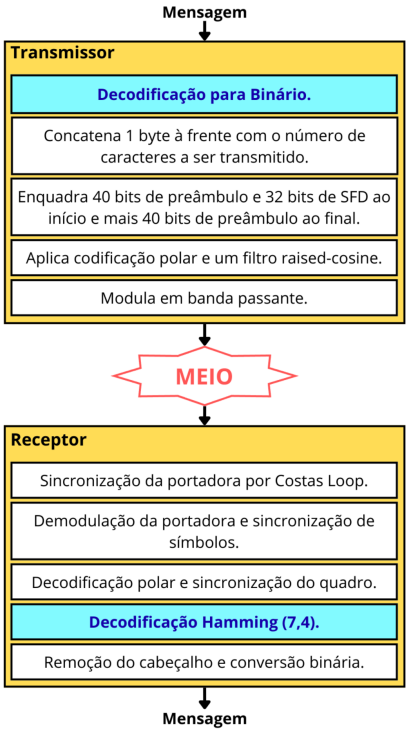

Fig. 1 - Fluxograma de etapas de transmissão do modem acústico.

III. METODOLOGIA

Para a implementação das mudanças indicadas na Figura 1 foi utilizado o projeto em código Matlab `modem_acustico` disponibilizado pelo docente, o qual abrigou as alterações necessárias. As modificações foram implementadas por meio de duas funções próprias: uma codifica a mensagem no transmissor e outra a decodifica no receptor.

Quanto aos *softwares* utilizados, para as simulações computacionais foram utilizados tanto Matlab quanto Octave, enquanto as imagens e *slides* foram produzidos pelo Canva e GIMP. A integração foi feita utilizando um repositório no GitHub[2].

IV. DESENVOLVIMENTO

Inicialmente foi desenvolvida a função que implementa a codificação da mensagem em Hamming(7,4), incluindo os bits de paridade. A linguagem Matlab/Octave já oferece uma função que supre essa necessidade, como indicado abaixo:

```
var = encode(msg_bits', 7, 4, 'hamming');
```

porém optou-se por não utilizá-la e criar outra semelhante. Na Figura 2 está indicado o fluxograma da função do codificador criada para substituir a `encode`.

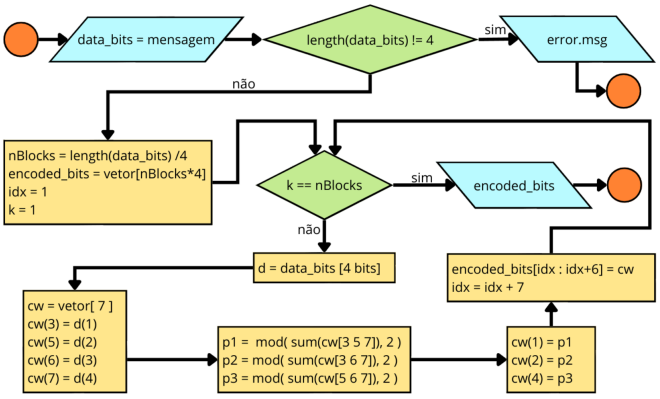

Fig. 2 - Fluxograma do código da função de codificação Hamming(7,4).

A função inicia verificando se o tamanho da mensagem a ser enviada é múltiplo de 4 para garantir que não faltou nenhum bit. Em seguida ele obtém o número de blocos de 4 bits a serem codificados e aloca um vetor para abrigar todos os bits da mensagem. Então, entra em um laço de repetição que varre todos os blocos da mensagem. Em cada bloco a função extrai os bits de dados e salva no vetor `cw`, que armazena temporariamente o bloco codificado, nas posições 3, 5, 6 e 7. Após ele calcula a paridade dos bit, somando combinações de 3 deles e calculando o resto da divisão por 2, valores estes que são salvos nas posições 1, 2 e 4 de `cw`. Por fim, concatena na variável `encoded_bits` o `cw` com o bloco atual codificado. Caso terminem os blocos, o laço de repetição é finalizado e a variável `encoded_bits` é retornada.

Em relação à implementação em código, inicialmente no arquivo `transmissor.m` a modificação da Figura 3 foi necessária para passar a não incluir nesta etapa o byte com o tamanho da mensagem, caso contrário seria codificado com Hamming junto dos blocos da mensagem.

```
l = length(msg);
msg = double(msg);
msg = reshape(de2bi([l msg],8)',1,8*(l+1));
```

```
l = length(msg);
msg_decimal = double(msg);
msg_bits = reshape(de2bi(msg_decimal, 8)', 1, 8*l);
```

Fig. 3 - Modificação inicial em `transmissor.m`.

Após, a função de codificação é chamada, é obtido o byte de tamanho e formada a mensagem codificada, como indicado na Figura 4. O restante do código permanece igual.

```
% Chama a função para codificação Hamming
encoded_msg_bits = hamming74_encode_stream(msg_bits);

% Converte para binário o byte de tamanho
length_bits = reshape(de2bi(l, 8)', 1, 8);

% 4. Monta a mensagem codificada
payload = [length_bits encoded_msg_bits];
```

Fig. 4 - Modificação final em `transmissor.m`.

E estas são as modificações citadas anteriormente, na Figura 1, relacionadas à codificação binária. Passando para a implementação da função `hamming74_encode_stream`, está indicada na Figura 5, com comentários que a relacionam com seu fluxograma da Figura 2.

```
function encoded_bits = hamming74_encode_stream(data_bits)
    % Converte data_bits em linha
    data_bits = data_bits(:).';

    % Testa se data_bits é múltiplo de 4
    if mod(length(data_bits),4) ~= 0
        error("Número de bits não múltiplo de 4.");
    end

    % Obtém o número de blocos
    nBlocks = length(data_bits) / 4;
    % Aloca um vetor para abrigar toda a mensagem codificada
    encoded_bits = zeros(1, nBlocks * 7);
    idx = 1;

    % Laço de repetição que varre todos os blocos
    for k = 1:nBlocks
        % Extrai os bits de dados do bloco
        d = data_bits((k-1)*4 + (1:4));

        % Posiciona os bits de dados no bloco codificado
        % p1 p2 d1 p3 d2 d3 d4
        cw = zeros(1,7);
        cw(3) = d(1); % d1
        cw(5) = d(2); % d2
        cw(6) = d(3); % d3
        cw(7) = d(4); % d4

        % Calcula os bits de paridade
        p1 = mod(sum(cw([3 5 7])),2);
        p2 = mod(sum(cw([3 6 7])),2);
        p3 = mod(sum(cw([5 6 7])),2);

        % Posiciona os bits de paridade no bloco codificado
        cw(1) = p1;
        cw(2) = p2;
        cw(4) = p3;

        % Concatena o bloco codificado à mensagem codificada
        encoded_bits(idx:idx+6) = cw;
        idx = idx + 7;
    end
end
```

Fig. 5 - Código de implementação do codificador Hamming.

Em relação ao decodificador, inicialmente foi necessário modificar o arquivo `receptor_2.m`, como indicado na Figura 6, adaptando para obter a largura da mensagem de

acordo com a codificação Hamming, além de chamar a função de decodificação. O restante do código permanece igual.

```
x = x(b+1:end);
l = bi2de(x(1:8)');
x = x(9:min(end,(l+1)*8));
```

```
encoded_length = (l * 8 / 4) * 7;
encoded_msg_bits = x(9 : 8 + encoded_length);
decoded_msg_bits_col = hamming74_decode_stream(encoded_msg_bits);
x = decoded_msg_bits_col';
```

Fig. 6 - Modificações no arquivo `receptor_2.m`.

A função do decodificador recebe a mensagem recebida, sem preâmbulo e SFD, codificada por Hamming e extrai apenas os bits originais da mensagem. O fluxograma da função está indicado na Figura 7. Ela inicia verificando se o tamanho da mensagem a ser enviada é múltiplo de 7 para garantir que não faltou nenhum bit. Após ela define a matriz de paridade H, que indica a forma matemática como o receptor detecta os erros. Em seguida ele obtém o número de blocos de 7 bits a serem codificados e aloca um vetor para abrigar todos os blocos da mensagem. Então entra em um laço de repetição que varre todos os blocos de 7 bits. Para cada bloco ele começa salvando os seus bits na variável `cw`, multiplica esses bits por H para detectar os erros e salva o número do bit errado em `syndrome_val`. Então entra em uma condicional que verifica se `syndrome_val==0` que indica que não ocorreu erro, caso tenha ocorrido o valor da variável é o bit que deve ser invertido, por meio do cálculo do resto da divisão por 2 deste bit somado com 1. Por fim, concatena na variável `data_bits` o `cw` com o bloco atual decodificado. Caso terminem os blocos, o laço de repetição é finalizado e a variável `data_bits` é retornada.

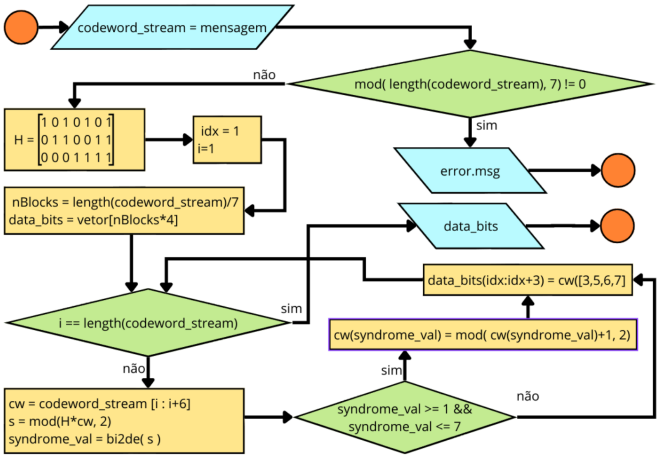

Fig. 7 - Fluxograma do código da função de decodificação Hamming(7,4).

Está indicado na Figura 8 a implementação da função `hamming74_decode_stream`, com comentários que o relacionam com seu fluxograma da Figura 7. Com isso, a etapa destacada na Figura 1 como decodificação Hemming está concluída.

```
function data_bits = hamming74_decode_stream(codeword_stream)
    % Converte codeword_stream em linha
    codeword_stream = codeword_stream(:).';

    % Testa se data_bits é múltiplo de 7
    if mod(length(codeword_stream),7) ~= 0
        error("Tamanho não é múltiplo de 7.");
    end

    % Cria a matriz de paridade
    H = [1 0 1 0 1 0 1;
         0 1 1 0 0 1 1;
         0 0 0 1 1 1 1];

    % Obtém o número de blocos
    nBlocks = length(codeword_stream)/7;
    % Aloca um vetor para abrigar toda a mensagem codifica
    data_bits = zeros(1, nBlocks*4);
    idx = 1;

    % Laço de repetição que varre todos os blocos
    for i = 1:7:length(codeword_stream)
        % Extrai os bits de dados do bloco
        cw = codeword_stream(i:i+6);

        % Detecta o bit que ocorreu erro
        s = mod(H*cw.',2);
        syndrome_val = bi2de(s.',"left-msb");

        % Se ocorreu erro inverte esse bit
        if syndrome_val >= 1 && syndrome_val <= 7
            cw(syndrome_val) = mod(cw(syndrome_val)+1,2);
        end

        % Concatena os bits de dados à mensagem decodificada
        data_bits(idx:idx+3) = cw([3 5 6 7]);

        idx = idx + 4;
    end
end
```

Fig. 8 - Código de implementação do decodificador Hamming.

V. RESULTADOS

A fig 2. mostra o log de execução do transmissor (tx.m) capturado durante a transmissão da mensagem "Comunicação de Dados"

```
>> tx
Tamanho do quadro: 400 simbolos
Tamanho do quadro: 400 bits
Tamanho do sinal: 32401 amostras
```

Fig 2. saída do Command Window do tx.m

Com base na Figura 1, observa-se que a mensagem original contém 20 caracteres, correspondendo a 160 bits de informação. No entanto, o tamanho final do quadro transmitido é de 400 bits. Essa expansão justifica-se pelo hamming(7,4) que transforma cada 4 bits em 7, resultando em 280 bits. Soma-se 1 byte (8 bits) de cabeçalho, 80 bits de preâmbulo de início e fim e SFD de 32 bits, totalizando 400 bits.

A fig 3. mostra o log de execução do receptor (rx.m) capturado durante a transmissão da mensagem "Comunicação de Dados".

```
Iniciando captura de áudio...
Aguardando som... Nível mínimo: 0.01
Gravação concluída, analisando sinal...
Nível de sinal detectado: 0.10599
Sinal forte detectado! Salvando arquivo...
**********************************
*** MENSAGEM DECODIFICADA (HAMMING) ***
*** (Erros de 1 bit por bloco foram corrigidos) ***
**********************************
    "Tamanho da mensagem recebida: "    "20"    " bytes"

    "Mensagem recebida: "    "Comunicação de Dados"

Tamanho do quadro: 400 simbolos
Tamanho do quadro: 400 simbolos
```

Fig 3. saída do Command Window do rx.m

Como mostrado na figura 3, o sinal foi detectado e o hamming decodificou e corrigiu os erros na mensagem.

Na sequência observa-se que o receptor identificou corretamente o SFD (Start Frame Delimiter) em meio ao ruído inicial, travando o início da leitura de dados e, sequencialmente, já leu o primeiro byte após o SFD, identificando corretamente que a mensagem possuía 20 caracteres.

Ao final a mensagem foi apresentada corretamente, indicando que o algoritmo de Hamming(7,4) foi capaz de corrigir eventuais inversões de bit causadas pelo ruído do canal acústico.

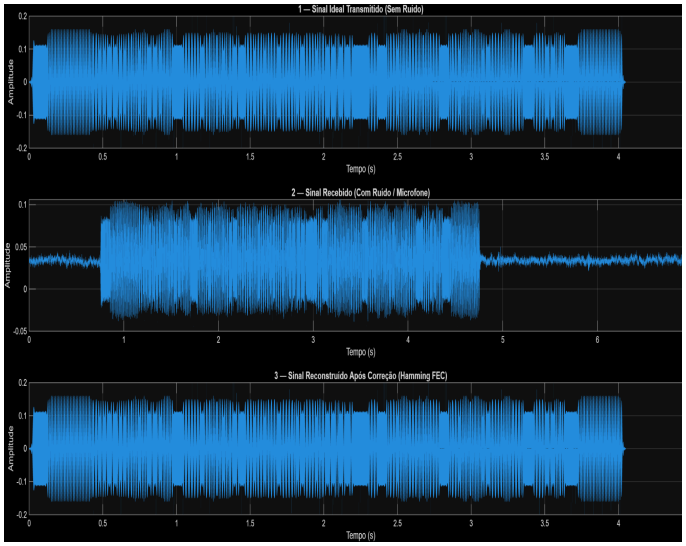

fig 4. gráfico do sinal de entrada (1), sinal recebido, sinal reconstruído

Na figura 4 temos 3 gráficos nela:

Gráfico 1 - Sinal Ideal Transmitido (Sem Ruído. Observa-se uma forma de onda limpa, com amplitude constante e centrada na frequência da portadora de 2 kHz.

Gráfico 2 - Sinal Recebido (Com Ruído / Microfone). Ilustra o sinal capturado pelo microfone após trafegar pelo canal aéreo. A amplitude máxima do sinal recebido (aprox. 0.1) é inferior à do sinal transmitido, evidenciando a atenuação natural do canal acústico. Além disso, nas regiões onde não há transmissão (ex: 0s a 0.8s), observa-se um "piso de ruído" causado pelo ruído ambiente e térmico do hardware de áudio.

Gráfico 3 - Sinal Reconstruído Após Correção (Hamming FEC). Representa a "re-modulação" da mensagem final que foi decodificada pelo receptor. Embora o sinal real (Gráfico 2) tenha sofrido distorções, o algoritmo de Hamming(7,4) foi capaz de identificar e corrigir eventuais bits invertidos.

V. CONCLUSÃO

O projeto alcançou com êxito o objetivo de implementar um sistema de transmissão de dados digitais via canal acústico não guiado, integrando conceitos de processamento de sinais (Camada Física) e controle de fluxo de dados (Camada de Enlace).

A implementação do protocolo de Enquadramento (Framing) mostrou-se essencial para a viabilidade do sistema. A utilização de um Delimitador de Início de Quadro (SFD) de 32 bits, aliada à técnica de correlação cruzada, permitiu ao receptor identificar o início da transmissão com precisão de amostra, superando a aleatoriedade do ruído ambiente. Adicionalmente, a inclusão do cabeçalho de tamanho possibilitou um processamento eficiente, evitando a decodificação de ruído residual após o término da mensagem.

No que tange à confiabilidade, a adoção do código Hamming(7,4) para Correção de Erros (FEC) validou-se como uma estratégia robusta. Os testes demonstraram que o sistema foi capaz de recuperar a integridade da mensagem original mesmo quando bits individuais eram corrompidos durante a propagação pelo ar.

Em suma, a combinação de modulação BPSK robusta com técnicas de Enlace de Dados garantiu a entrega correta da informação ("Comunicação de Dados"), validando a aplicação prática dos conceitos teóricos estudados.

REFERÊNCIAS

[1] MARY, Delphine P.; SIMRAN, A. *Design and Implementation of Hamming Code with Error Correction Using Xilinx.* International Journal of Scientific Research in Computer Science, Engineering and Information Technology, v. 10, n. 4, p. 158-166, Jul./Aug. 2024. DOI: 10.32628/CSEIT24104117.

[2] SCHULTZ, Vinícius. ModemAcustico. GitHub, 2025. Disponível em: https://github.com/VenixBR/ModemAcustico.git. Acessado em: 23 nov. 2025.

[3] LATHI, B. P., “Modern Digital And Analog Communications Systems”, Oxford University Press, 3rd Ed, 1998.

[4] HAYKIN, Simon., Digital Communications, Jonh Wyley & Sons, 4th edition, 2007

[5] STALLINGS, William., “Data and Computer Communications. Prentice Hall”, 6th edition, 2000

[6] SOARES, Luiz Fernando Gomes., “Redes de Computadores. Das LANs, MANs e WANs `as Redes ATM”, Editora Campus, 6 a edi¸c˜ao, 1995.