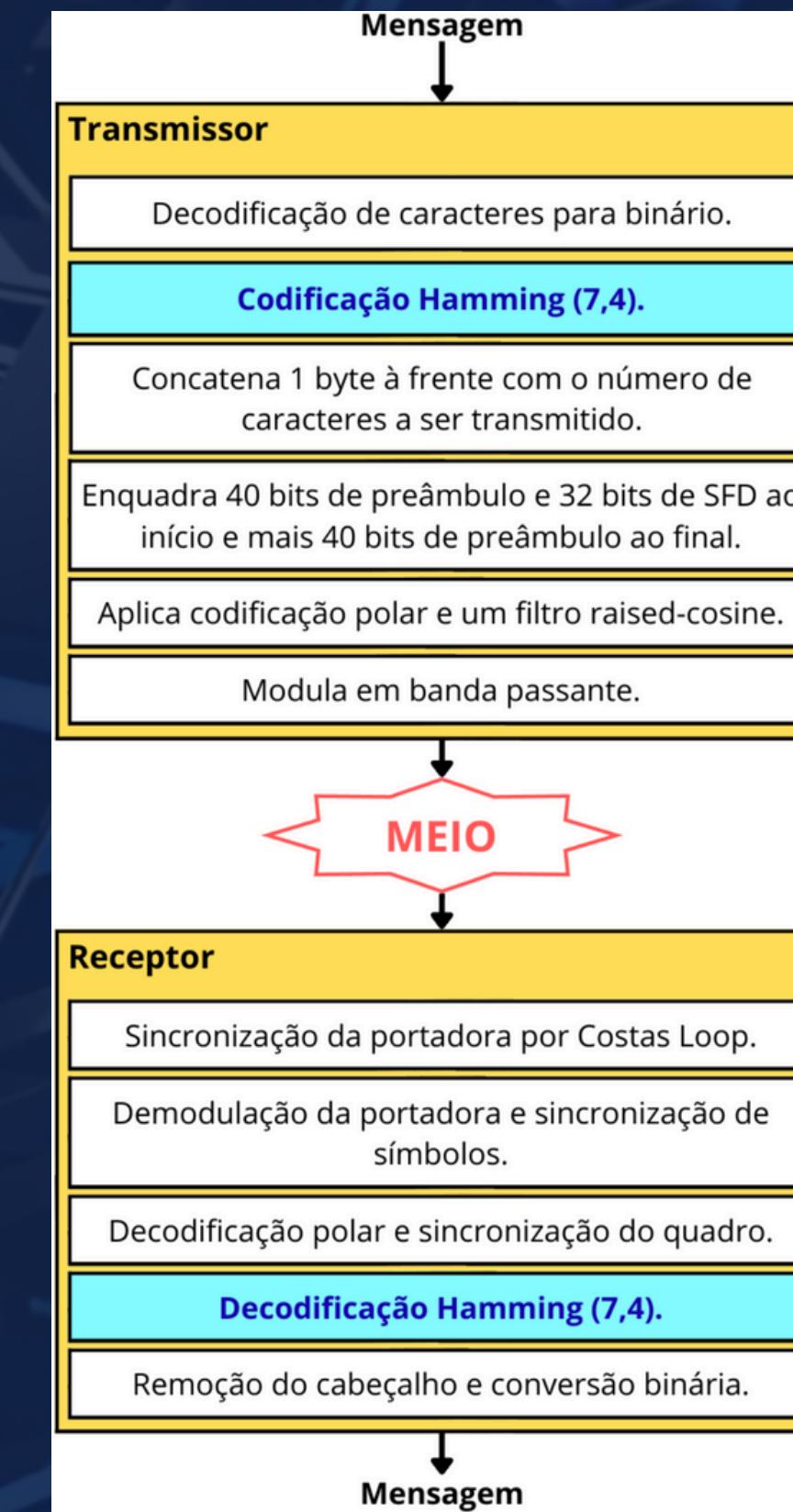


# SISTEMA DIGITAL DE COMUNICAÇÃO DE DADOS

Gil Alves Magalhães  
Luís Pedro D'Avila Zorzi  
Vinícius Gabriel Schultz

# MODEM ACÚSTICO ORIGINAL E MODIFICAÇÕES



# DETECÇÃO E CORRECÇÃO DE ERROS HAMMING(7,4)

Durante a transmissão pelo ar (áudio), ruídos podem inverter bits.

O código Hamming(7,4) adiciona bits de paridade que permitem:

- Detectar se ocorreu erro
- Descobrir qual bit está errado

Cada bloco de 4 bits da mensagem (dados) é transformado em 7 bits:

- 4 bits de dados: d1 d2 d3 d4
- 3 bits de paridade: p1 p2 p3

A ordem dos bits no bloco transmitido é:

[p1 p2 d1 p3 d2 d3 d4]  
1 2 3 4 5 6 7

Cada paridade cobre um conjunto específico de posições:

- p1 verifica bits 1, 3, 5, 7
- p2 verifica bits 2, 3, 6, 7
- p3 verifica bits 4, 5, 6, 7

# DETECÇÃO E CORREÇÃO DE ERROS

## HAMMING(7,4)

Matriz de paridade (H):

Mostra de forma matemática como o receptor detecta erros

$$\mathbf{H} := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- Cada linha representa uma paridade ( $p_1, p_2, p_3$ ).
- Cada coluna representa a posição no bloco  $cw = [p_1 \ p_2 \ d_1 \ p_3 \ d_2 \ d_3 \ d_4]$ .

Cálculo do síndrome:

$$s = \mathbf{H} \cdot \mathbf{cw}$$

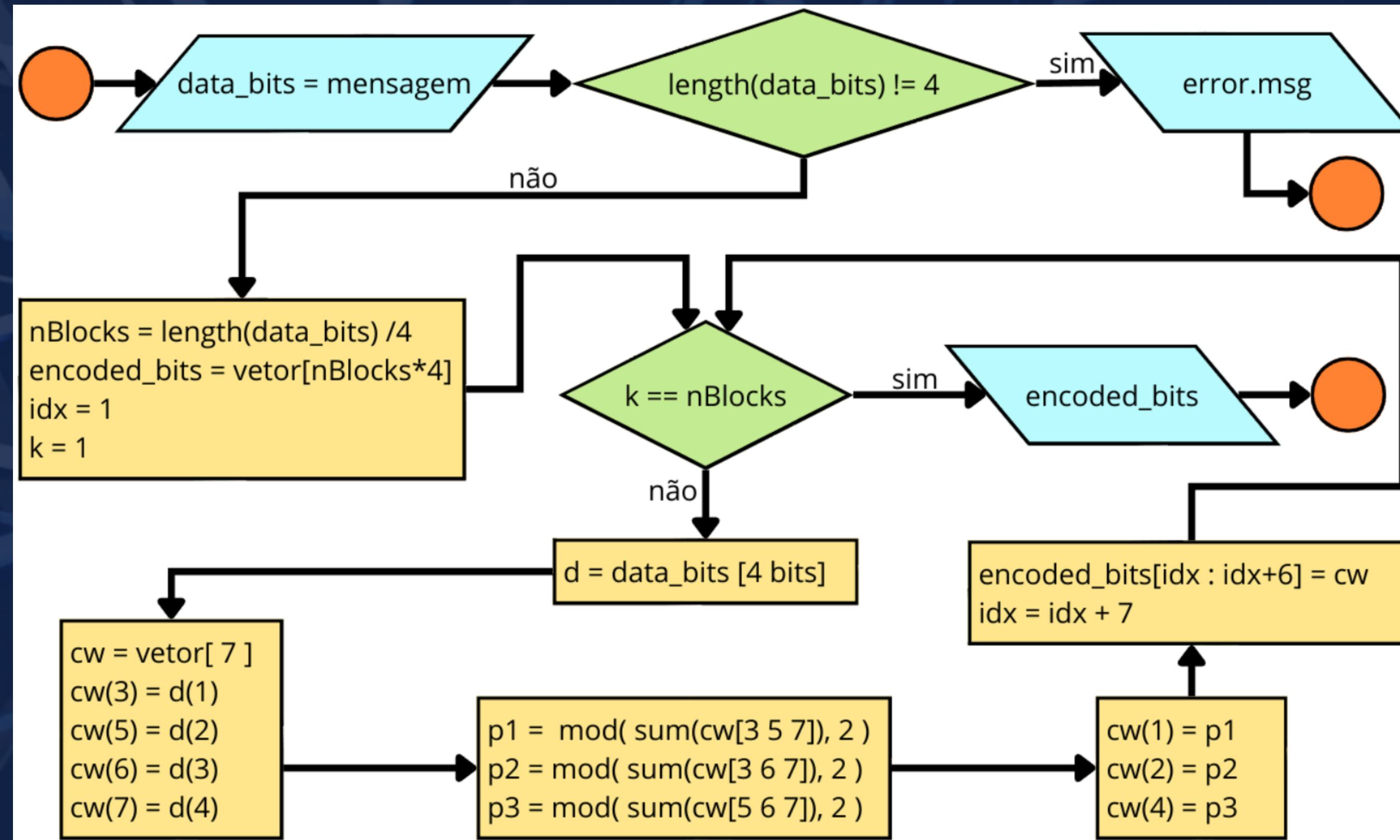
- $s = 000 \rightarrow$  nenhum erro detectado
- $s \neq 000 \rightarrow$  erro detectado, valor indica a posição do bit errado
- O valor do síndrome (1 a 7) indica a posição exata do bit errado.

Passos da correção

1. Identificar a posição do erro a partir do síndrome.
2. Inverter o bit nessa posição ( $0 \rightarrow 1$  ou  $1 \rightarrow 0$ ).
3. Após a inversão, o bloco de 7 bits está corrigido.

Extração dos dados originais  $d_1, d_2, d_3, d_4$

# FLUXOGRAMA DO ENCODER



# TRANSMISSOR.m

```
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);
```

```
% 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];
```

# ENCODER

```
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
```

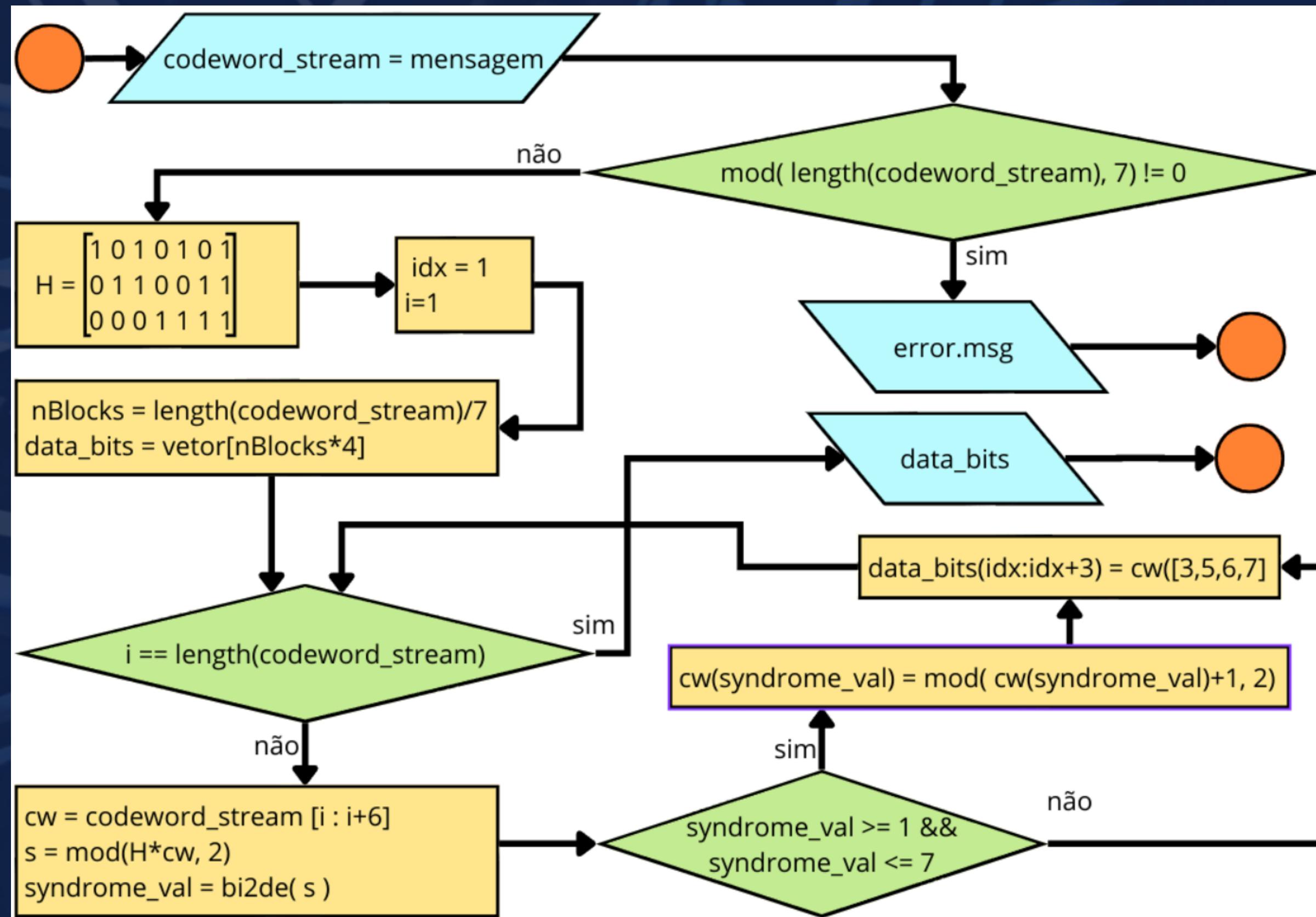


```
% Calcula os bits de paridade
p1 = mod(sum(cw([3 5 7])),2);
p2 = mod(sum(cw([3 6 7])),2);
p3 = mod(sum(cw([5 6 7])),2);

% 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;
```

# FLUXOGRAMA DO DECODER



# RECEPTOR\_2.m

```
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';
```

# DECODER

```
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 codificada
    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
```

# TRANSMISSOR

A mensagem original "Comunicação de Dados" 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.

Taxa de Amostragem (Fa) = 8000 amostras por segundo.

Taxa de Bits (RB) = 100 bits por segundo.

Amostras por Bit = 80 amostras

Totalizando  $400 \times 80 = 32000$  amostras

$400\text{bits}/100\text{bps} = 4\text{s}$  de áudio

>> tx

Tamanho do quadro: 400 símbolos

Tamanho do quadro: 400 bits

Tamanho do sinal: 32401 amostras

# RECEPTOR

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 e 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.

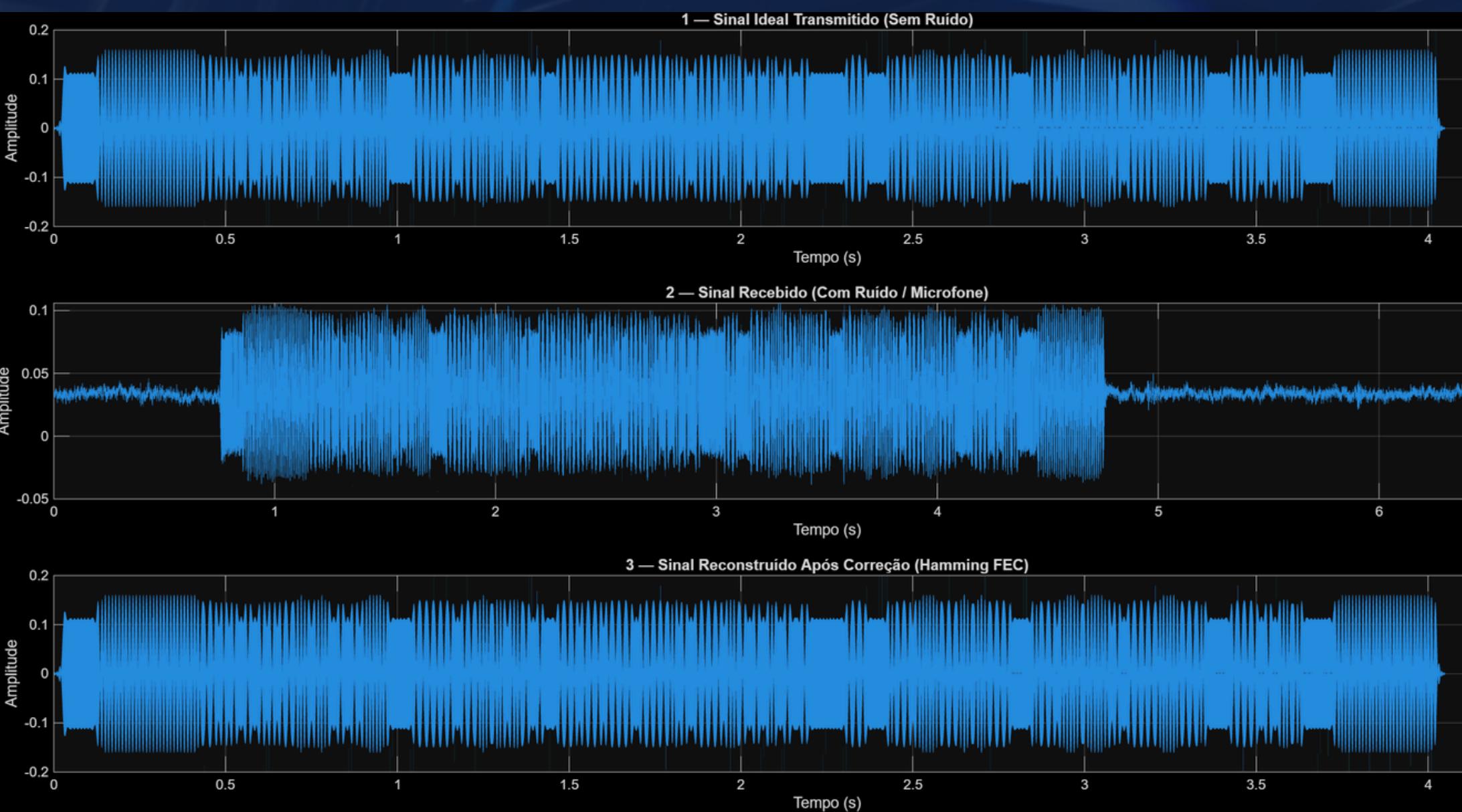
```
Iniciando captura de áudio...
Aguardando som... Nível mínimo: 0.01
Gravação concluída, analisando sinal...
Nível de sinal detectado: 0.10599
Sinal forte detectado! Salvando arquivo...
*****
*** MENSAGEM DECODIFICADA (HAMMING) ***
*** (Erros de 1 bit por bloco foram corrigidos) ***
*****
    "Tamanho da mensagem recebida: "    "20"    " bytes"
    "Mensagem recebida: "    "Comunicação de Dados"
Tamanho do quadro: 400 simbolos
Tamanho do quadro: 400 simbolos
```

# GRÁFICOS DO SINAL

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: Os 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.



# RESULTADOS

Destaca-se, no Gráfico 1, uma redução de 90% na taxa de erros de transmissão do modem com correção de erros, enquanto no Gráfico 2 o overhead de número de bits necessários para realizar a comunicação causado pela implementação, assumindo um comportamento praticamente linear de 1,76% mais 3,86% para cada caractere da mensagem.

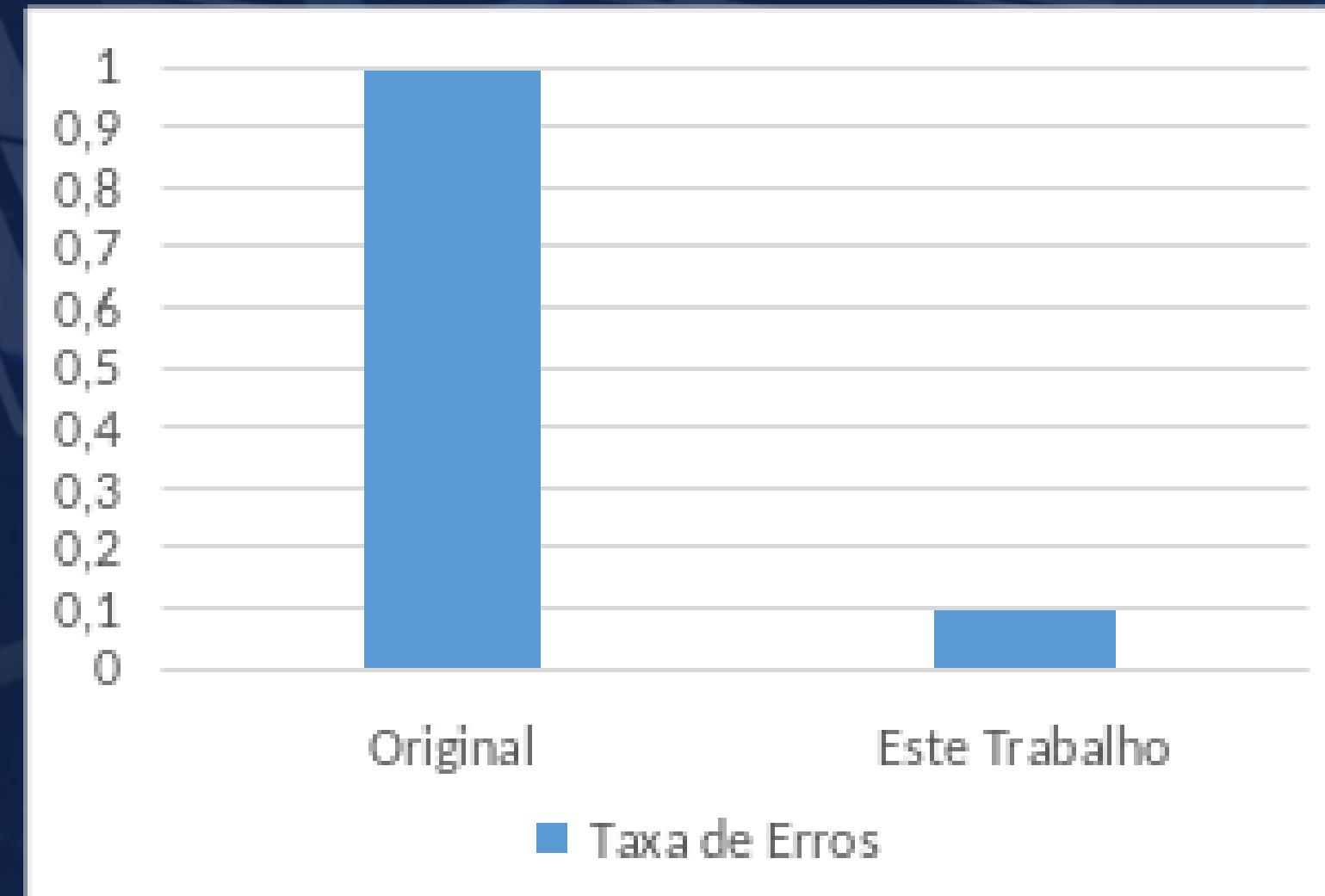


Gráfico 1 - Redução da taxa de erros.

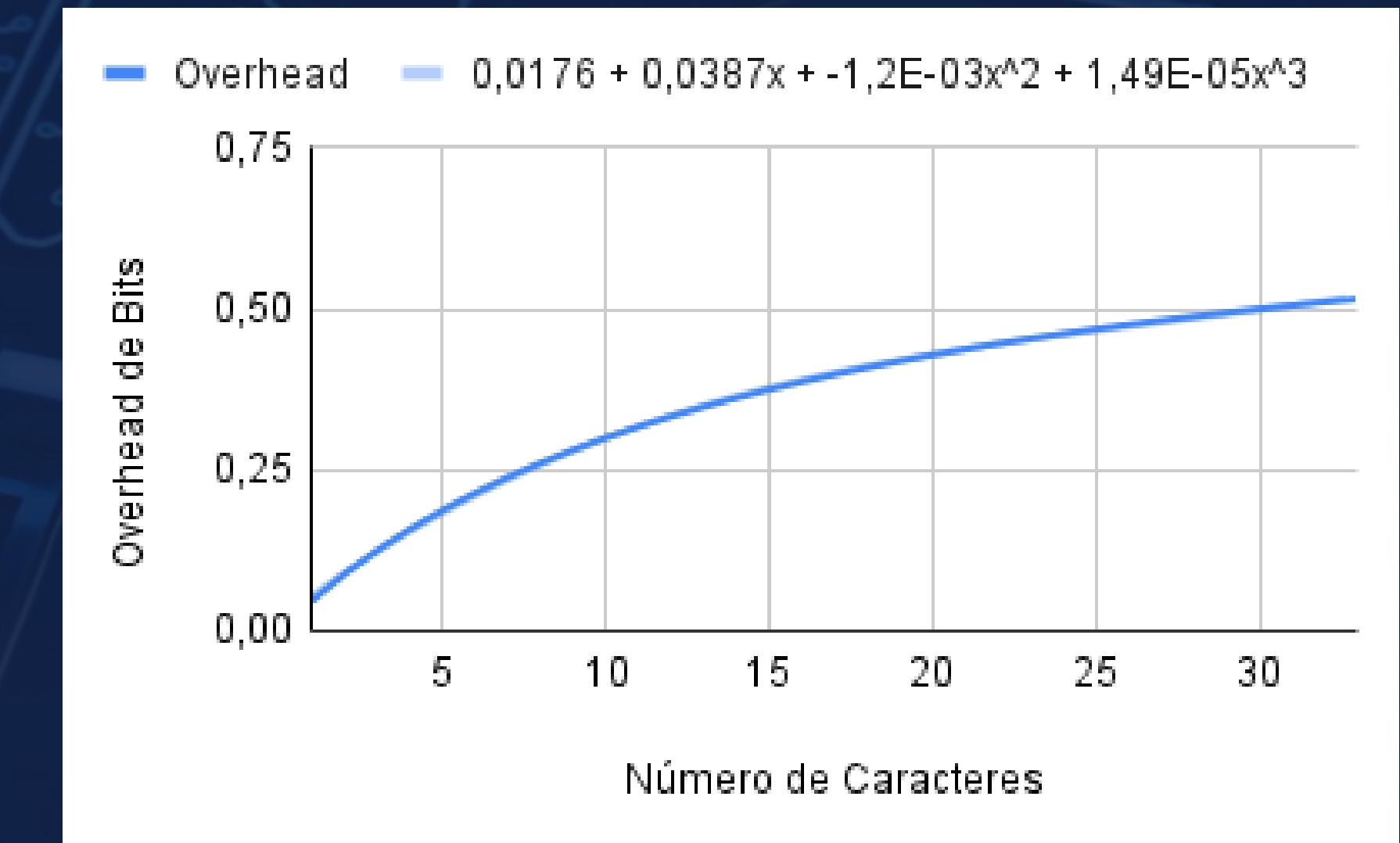
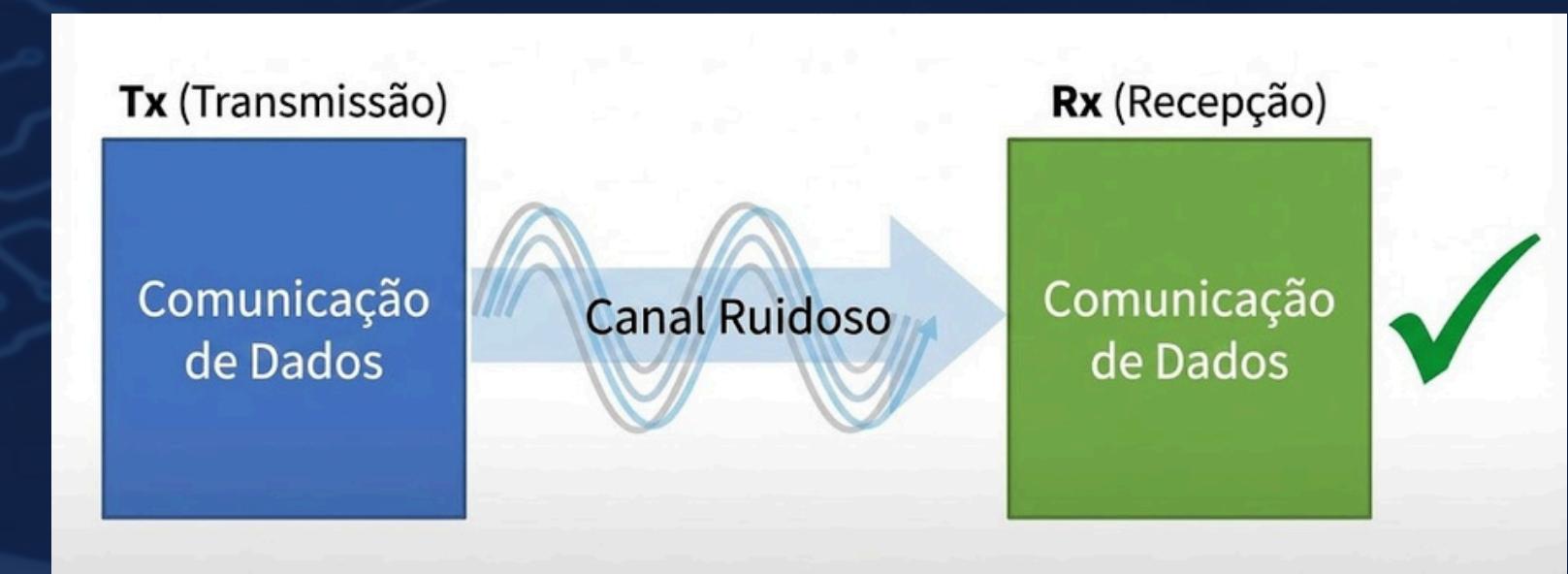


Gráfico 2 - Overhead de número de bits por caractere da mensagem.

# CONCLUSÃO

**HAMMING(7,4) PARA CORREÇÃO DE ERROS (FEC) VALIDOU-SE COMO UMA ÓTIMA ESTRATÉGIA. 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.**



# THANK YOU!



<https://github.com/VenixBR/ModemAcustico.git>