



**INSTITUTO
FEDERAL**

Santa Catarina

Câmpus
São José

Conversão AD/DA através de PCM NRZ

Sistemas de Comunicação I

Arthur Cadore Matuella Barcella

01 de Maio de 2024

Sumário

1. Introdução	3
2. Fundamentação Teórica:	3
3. Desenvolvimento e Resultados	4
3.1. Conversão AD	4
3.2. Transmissão PCM NRZ	4
3.3. Ruído na Recepção:	5
3.4. Reconstrução do sinal PCM	6
3.5. Conversão DA:	7
4. Scripts e Códigos Utilizados	8
4.1. Conversão AD	8
4.2. Transmissão PCM NRZ	10
4.3. Ruído na Recepção	11
4.4. Reconstrução do sinal PCM	11
4.5. Conversão DA	12
5. Conclusão	13
6. Referências Bibliográficas	13

1. Introdução

O objetivo deste trabalho é simular a conversão de um sinal analógico para digital através dos processos de amostragem e quantização. Em seguida a transmissão do sinal de maneira digital através de um canal de comunicação PCM (Pulse Code Modulation) com codificação não-retornante (NRZ).

Por fim, a conversão do sinal de digital para analógico novamente no receptor, realizando a filtragem do sinal recebido para interpreta-lo corretamente, em seguida, novamente amostragem e quantização e finalmente a conversão da informação recebida novamente para um sinal analógico.

Desta forma, poderemos compreender o funcionamento de um sistema de comunicação digital PCM NRZ e os efeitos da amostragem e quantização no sinal transmitido.

2. Fundamentação Teórica:

- Amostragem e Quantização: A amostragem é o processo de capturar valores de um sinal analógico em intervalos regulares de tempo. A quantização é o processo de discretizar os valores amostrados em níveis de amplitude finitos.
- Ruído AWGN: O ruído AWGN (Additive White Gaussian Noise) é um tipo de ruído que é adicionado ao sinal transmitido, simulando interferências e distorções no sinal. Utilizamos esse tipo de ruído por ser o mais aleatório possível no espectro de frequência, simulando assim o ruído de um canal de comunicação real.
- Ruído de Quantização: O ruído de quantização é o erro que ocorre devido a discretização dos valores amostrados. Quanto maior a quantidade de níveis de quantização, menor será o ruído de quantização, desta forma, devemos priorizar a quantização com alta taxa de bits, para evitar a distorção do sinal interpretado.
- PCM (NRZ): O PCM (Pulse Code Modulation) é um método de modulação digital que consiste em amostrar e quantizar um sinal analógico, e transmitir a informação digitalizada através de um canal de comunicação. O NRZ (Non-Return-to-Zero) é um dos métodos codificação PCM que consiste em manter o sinal em nível alto ou baixo durante todo o período de bit, sem valores no zero.
- Processo de Filtragem: A filtragem é o processo de atenuar frequências indesejadas do sinal recebido, para que o sinal possa ser interpretado corretamente. A filtragem é realizada através de um filtro passa-baixas, que atenua as frequências acima de uma determinada frequência de corte.
- Conversão AD: A conversão AD (Analógico para Digital) é o processo de amostrar e quantizar um sinal analógico, transformando-o em um sinal digital. Utilizaremos este processo no início da simulação, para digitalizar o sinal analógico de entrada.

- Conversão DA: A conversão DA (Digital para Analógico) é o processo de transformar um sinal digital em um sinal analógico. Utilizaremos esse processo no final da simulação, para transformar o sinal digital recebido em um sinal analógico.

3. Desenvolvimento e Resultados

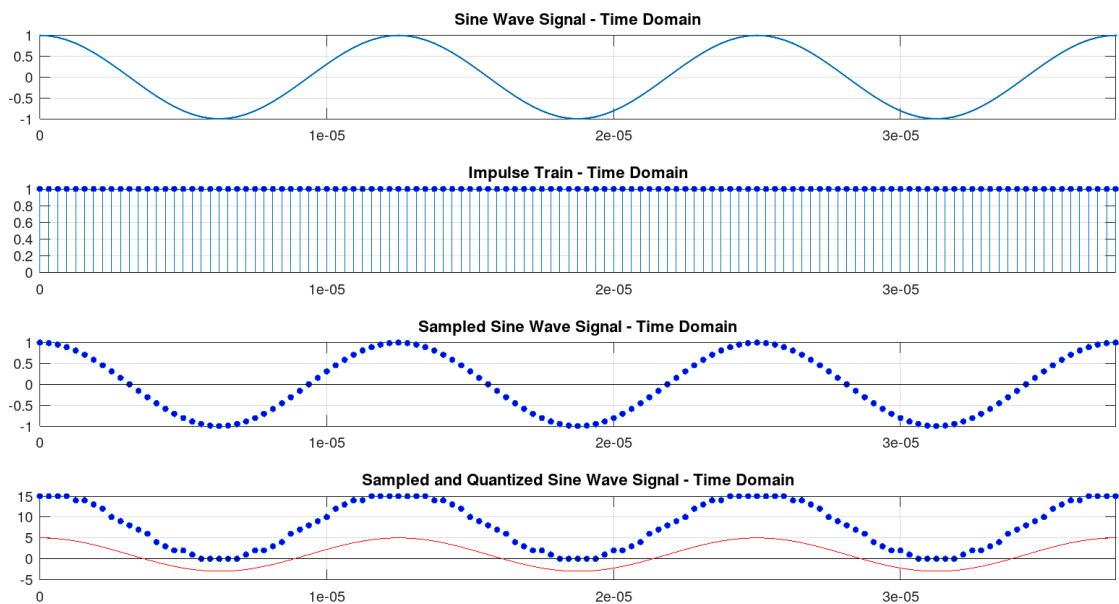
3.1. Conversão AD

Inicialmente, um sinal analógico foi criado para realizar o processo de conversão AD, para isso, foi escolhido um sinal senoidal de 80KHz com amplitude de 1V, neste ponto, é importante escolher um sinal bem comportado pois como conhecemos seu comportamento, podemos analisar melhor os efeitos da amostragem e quantização e qual resultado esperar no receptor após o processo de conversão D/A.

Em seguida, o sinal foi amostrado com uma taxa de amostragem de $40 * F(s)$ (ou seja, 3200KHz) e quantizado em 4 bits, gerando um sinal digital que será transmitido através do canal de comunicação PCM.

A figura abaixo ilustra o processo de amostragem e quantização do sinal analógico:

Figure 1: Elaborada pelo Autor



Sinal Senoidal sendo Amostrado e Quantizado

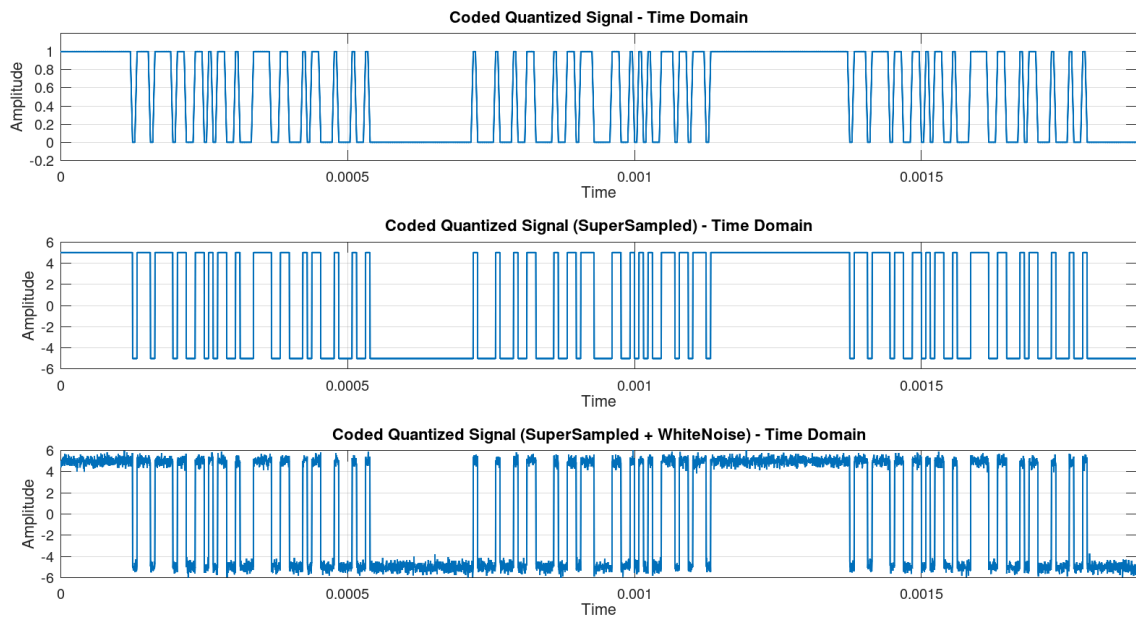
3.2. Transmissão PCM NRZ

Uma vez que o sinal analógico foi amostrado e quantizado, temos como saída um vetor de bits. Através deste vetor, podemos realizar sua transmissão através de um canal de comunicação PCM NRZ.

O canal de comunicação PCM NRZ nada mais faz do que transmitir um nível de amplitude específico para cada valor de bit dado. Portanto, para a transmissão deste sinal, utilizamos -5 para 0 e 5 para 1.

A figura abaixo ilustra o sinal digital transmitido através do canal de comunicação PCM NRZ:

Figure 2: Elaborada pelo Autor



Sinal Digital Transmitido através do Canal PCM NRZ

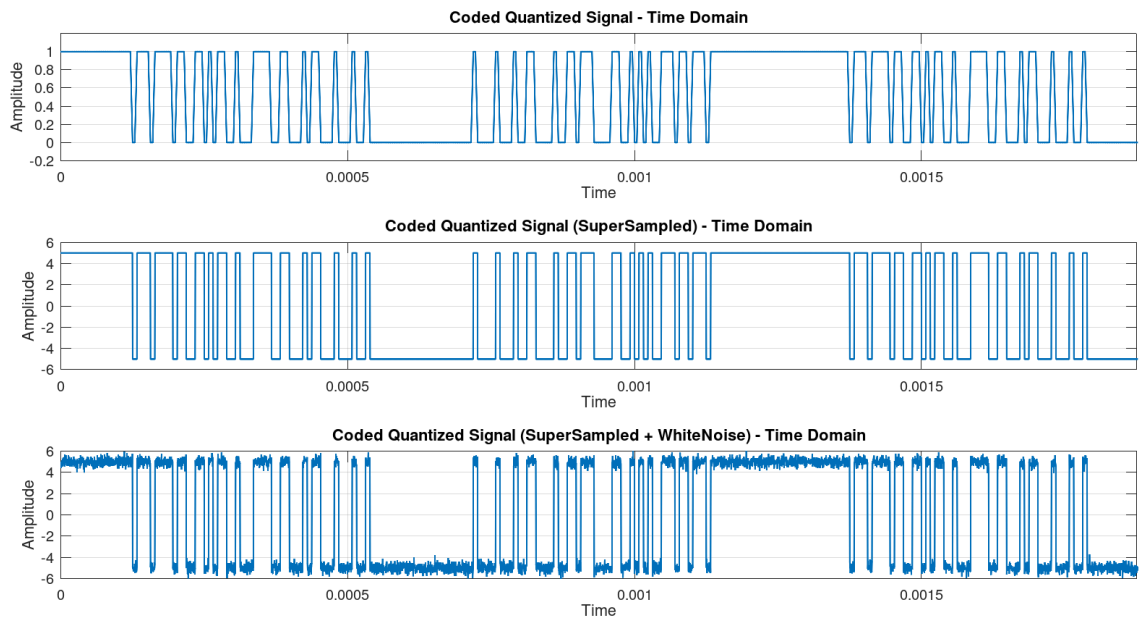
3.3. Ruído na Recepção:

Uma vez o sinal sendo transmitido, é necessário adicionar um ruído AWGN para simular as interferências e distorções que ocorrem em um canal de comunicação real, para isso, geramos um ruído AWGN com uma amplitude de $0.1V$ e realizamos a soma com o sinal transmitido.

O sinal resultante é o sinal recebido no receptor, este agora está com distorções na sua amplitude por conta das componentes adicionadas pelo ruído AWGN.

Para ilustrar o sinal recebido e o ruído AWGN adicionado, temos a figura abaixo, note que a amplitude do sinal recebido foi distorcida pelo ruído AWGN:

Figure 3: Elaborada pelo Autor



Sinal Recebido com Ruído AWGN

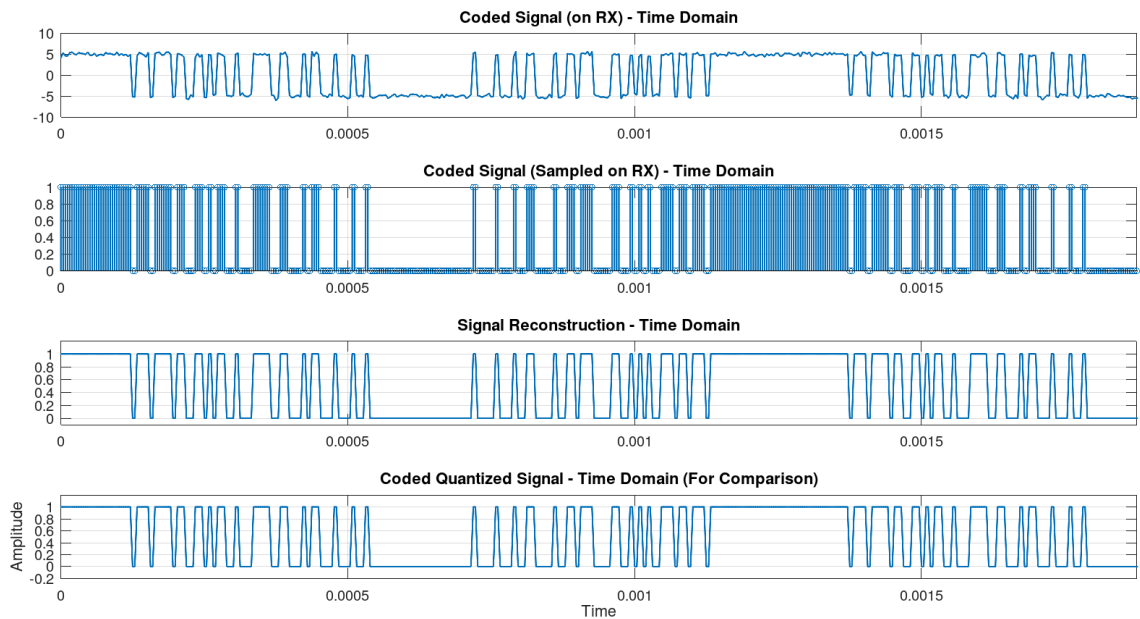
3.4. Reconstrução do sinal PCM

O sinal PCM transmitido é então reconstruído no receptor, para isso, é utilizado um limiar, neste caso $0V$, para decidir qual o valor binário correspondente do sinal em relação ao sinal propriamente recebido.

Uma vez recebido coletado, o sinal binário é superamostrado e filtrado, para que possamos visualizar o sinal corretamente com mais amostras para cada bit recebido.

Abaixo temos a figura do sinal PCM reconstruído no receptor, note que abaixo do sinal PCM recebido e reconstruído há um plot do sinal transmitido antes de ser corrompido pelo ruído, para que possamos compara-los:

Figure 4: Elaborada pelo Autor



Sinal PCM Reconstruído no Receptor

3.5. Conversão DA:

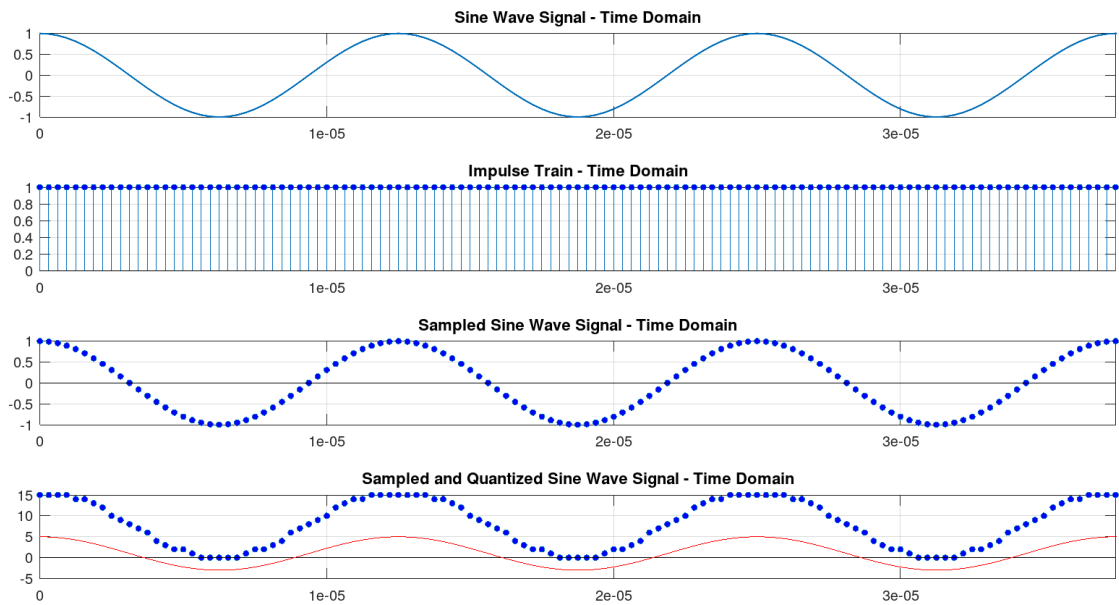
Uma vez com o trem de bits reconstruído, é necessário converter o sinal digital para analógico, para isso, realizamos um agrupamento dos bits de acordo com a ordem de transmissão.

Para um cenário real, a taxa de quantização deve ser igual entre as partes e previamente configurada, assim quando os bits forem recebidos o destinatário saberá como agrupá-los.

Uma vez agrupados, definimos um limiar para cada nível, no caso de 4bits, são 2^b níveis, portanto diferentes níveis possíveis para o sinal analógico na saída do D/A.

Desta forma, mapeamos cada agrupamento de 4bits em sua correspondente amplitude de saída, e assim, reconstruindo o sinal analógico:

Figure 5: Elaborada pelo Autor



Sinal Analógico Reconstruído no Receptor

4. Scripts e Códigos Utilizados

4.1. Conversão AD

O código abaixo apresenta a etapa de conversão AD, onde um sinal senoidal é amostrado e quantizado, gerando um sinal digital que será transmitido através do canal de comunicação PCM NRZ.

```
1 close all; clear all; clc;
2 pkg load signal;
3 pkg load communications;
4
5 % Altera o tamanho da fonte nos plots para 15
6 set(0, 'DefaultAxesFontSize', 20);
7
8 % Definindo a amplitude do sinal senoidal
9 A_signal = 1;
10
11 % Definindo a frequência do sinal senoidal
12 f_signal = 80000;
13
14 % Definindo a frequência de amostragem
15 fs = 40*f_signal;
16 Ts = 1/fs;
17 T = 1/f_signal;
18
19 % Definindo o tempo inicial e final do sinal
20 t_inicial = 0;
21 t_final = 0.01;
22
23 % Vetor de tempo
24 t = [t_inicial:Ts:t_final];
```



```

25
26 % Criando o sinal senoidal
27 signal = A_signal*cos(2*pi*f_signal*t);
28
29 % Criando um trem de impulsos com período de 2T
30 impulse_train = zeros(size(t));
31 impulse_train(mod(t, 1/fs) == 0) = 1;
32
33 % Amostragem do sinal senoidal
34 signal_sampled = signal .* impulse_train;
35
36 % Quantidade de níveis desejada (tirando o 0)
37 n=4;
38 num_levels = 2^n;
39
40 % Gerando os níveis de quantização automaticamente
41 levels = linspace(-1, 1, num_levels);
42
43 % Verifica se o vetor possui algum elemento com "0".
44 % Se sim, remove o elemento e sai do loop
45 for i = 1:length(levels)
46     if levels(i) == 0
47         levels(i) = [];
48         break;
49     end
50 end
51
52 % Quantização do sinal com 2^n níveis
53 quantized_signal = zeros(size(signal_sampled));
54 for i = 1:length(signal_sampled)
55     for j = 1:length(levels)
56         if signal_sampled(i) <= levels(j)
57             quantized_signal(i) = levels(j);
58             break;
59         end
60     end
61 end
62
63 % Plotando os sinais
64 figure(1)
65 subplot(411)
66 plot(t,signal)
67 grid on;
68 xlim([0 3*T])
69 title('Sinal Senoidal (Dominio do tempo)')
70
71 subplot(412)
72 stem(t,impulse_train, 'MarkerFaceColor', 'b')
73 grid on;
74 xlim([0 3*T])
75 title('Trem de impulsos (Dominio do tempo)')
76
77 subplot(413)
78 stem(t,signal_sampled, 'LineStyle','none', 'MarkerFaceColor', 'b')
79 grid on;
80 xlim([0 3*T])
81 title('Sinal Senoidal Amostrado (Dominio do tempo)')
82
83 subplot(414)

```

```

84 stem(t,quantized_signal, 'LineStyle','none', 'MarkerFaceColor', 'b')
85 xlim([0 3*T])
86 hold on;
87 plot(t,signal, 'r')
88 xlim([0 3*T])
89 grid on;
90 title('Sinal Senoidal Amostrado e Quantizado (Dominio do tempo)')

```

4.2. Transmissão PCM NRZ

O código abaixo apresenta a etapa de transmissão do sinal digital através do canal de comunicação PCM NRZ, onde o sinal digital é transmitido com níveis de amplitude específicos para cada valor de bit.

```

1  % Desloca o vetor quantizado para 0 ou mais:
2
3  min_value = min(quantized_signal);
4  quantized_signal = quantized_signal - min_value;
5
6  % Multiplica os valores quantizados para que sejam números inteiros
7
8  % Número de intervalos entre os níveis
9  num_intervals = num_levels - 1;
10
11 % Multiplicando os valores quantizados para que sejam números inteiros
12 quantized_signal_int = quantized_signal * num_intervals;
13
14 % Convertendo valores quantizados inteiros para binário
15 binary_signal = de2bi(quantized_signal_int, n);
16
17 % Concatenando os valores binários em um único vetor
18 binary_signal_concatenated = reshape(binary_signal.', 1, []);
19
20 % Vetor de tempo
21 t = linspace(0, 1, length(binary_signal_concatenated) * 2);
22
23 % Repetindo cada valor do sinal
24 repeated_signal = reshape(repmat(binary_signal_concatenated, 2, 1), 1, []);
25
26 % =====
27 % realizando a superamostragem do sinal com a função upper
28
29 % Superamostragem
30 n = 10;
31 amplitude = 5;
32 repeated_signal_up = upsample(repeated_signal, n);
33
34 filtr_tx = ones(1, n);
35 filtered_signal = filter(filtr_tx, 1, repeated_signal_up)*2*amplitude-
    amplitude;
36
37 % criando um novo vetor de t para o sinal filtrado
38 t_super = linspace(0, 1, length(filtered_signal));
39
40 var_noise = 0.1;
41 transmission_noise = sqrt(var_noise)*randn(1,length(filtered_signal));
42

```

```

43 % Gerando o sinal transmitido no meio de comunicação multiplicando o sinal
    de transmissão pelo ruído do meio.
44 transmitted_signal = filtered_signal + transmission_noise;
45
46 % Plotando o sinal
47 figure(2)
48 subplot(311)
49 plot(t,repeated_signal, 'LineWidth', 2);
50 ylim([-0.2, 1.2]);
51 xlim([0, 50*T]);
52 xlabel('Tempo');
53 ylabel('Amplitude');
54 title('Sinal Binário como Onda Quadrada');
55 grid on;
56
57 subplot(312)
58 plot(t_super,filtered_signal, 'LineWidth', 2);
59 xlim([0, 50*T]);
60 ylim([-amplitude*1.2 , amplitude*1.2]);
61 xlabel('Tempo');
62 ylabel('Amplitude');
63 title('Sinal Binário como Onda Quadrada');
64 grid on;
65
66 subplot(313)
67 plot(t_super,transmitted_signal, 'LineWidth', 2);
68 xlim([0, 50*T]);
69 ylim([-amplitude*1.2 , amplitude*1.2]);
70 xlabel('Tempo');
71 ylabel('Amplitude');
72 title('Sinal Binário como Onda Quadrada');
73 grid on;

```

4.3. Ruído na Recepção

O código abaixo apresenta a etapa de adição de ruído AWGN ao sinal transmitido, simulando as interferências e distorções que ocorrem em um canal de comunicação real.

```

1 a

```

4.4. Reconstrução do sinal PCM

O código abaixo apresenta a etapa de reconstrução do sinal PCM no receptor, onde o sinal recebido é comparado com um limiar para decidir qual o valor binário correspondente do sinal.

```

1 % Definindo o limiar (valor que vai decidir se o sinal é 0 ou 1)
2
3 limiar = 0;
4 % amostrando o sinal recebido
5 received_signal = transmitted_signal(n/2:n:end);
6
7 % Verifica se o sinal recebido é maior ou menor que o limiar.

```

```

8 % Se for maior, o sinal é 1, se for menor, o sinal é 0.
9 received_binary = received_signal > limiar;
10
11 % Vetor de tempo para o sinal recebido
12 t_received = linspace(0, 1, length(t_super)/n);
13
14 % Plotando o sinal
15 figure(3)
16 subplot(211)
17 plot(t_received, received_signal);
18 xlim([0, 50*T]);
19 subplot(212)
20 stem(t_received, received_binary);
21 xlim([0, 50*T]);
22
23 % Calculando a taxa de erro de bit
24 num_erro = sum(xor(received_signal, received_binary))
25 taxa_erro = num_erro/length(t_super)

```

4.5. Conversão DA

O código abaixo apresenta a etapa de conversão DA, onde o sinal digital é convertido para analógico, reconstruindo o sinal original.

```

1 % Vetor de tempo para o sinal recebido
2 t_received = linspace(0, 1, length(received_signal));
3
4 % Interpolando o sinal para restaurar a taxa de amostragem original
5 received_signal_interp = interp(received_signal, n);
6
7 % Filtrando o sinal interpolado para remover artefatos
8 filtr_rx = ones(1, n);
9 received_signal_filtered = filter(filtr_rx, 1, received_signal_interp) / n;
10
11 % Plotando o sinal recuperado
12 figure(4);
13 subplot(211);
14 plot(t_super, filtered_signal, 'LineWidth', 2, 'DisplayName', 'Sinal Original');
15 hold on;
16 plot(t_received, received_signal_filtered(1:length(t_received)), '--',
17 'LineWidth', 2, 'DisplayName', 'Sinal Recuperado');
18 xlim([0, 50*T]);
19 ylim([-amplitude*1.2, amplitude*1.2]);
20 xlabel('Tempo');
21 ylabel('Amplitude');
22 title('Sinal Original e Sinal Recuperado');
23 legend;
24 grid on;
25
26 % Convertendo o sinal recuperado para o formato analógico
27 received_analog_signal = received_signal_filtered(1:length(t_received)) *
28 (2*amplitude) - amplitude; % Normalização da amplitude

```

```

29 subplot(212);
30 plot(t_received, received_analog_signal, 'LineWidth', 2);
31 xlim([0, 50*T]);
32 xlabel('Tempo');
33 ylabel('Amplitude');
34 title('Sinal Analógico Recuperado');
35 grid on;

```

5. Conclusão

A partir dos conceitos vistos e dos resultados obtidos podemos concluir que a amostragem e quantização são processos fundamentais para a conversão de um sinal analógico para digital, e que a transmissão de um sinal digital através de um canal de comunicação PCM NRZ é possível e eficiente.

Além disso, a adição de ruído AWGN no sinal transmitido simula as interferências e distorções que ocorrem em um canal de comunicação real, e a reconstrução do sinal PCM no receptor é possível através da comparação do sinal recebido com um limiar.

Por fim, a conversão do sinal digital para analógico é possível através da conversão dos valores binários em níveis de amplitude, e a filtragem do sinal recebido é necessária para remover artefatos e distorções.

6. Referências Bibliográficas

Para o desenvolvimento deste relatório, foi utilizado o seguinte material de referência:

- Software Defined Radio Using MATLAB & Simulink and the RTL-SDR, de Robert W. Stewart