Conversão AD/DA através de PCM NRZ

Sistemas de Comunicação I

Arthur Cadore Matuella Barcella

01 de Maio de 2024

Sumário

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 possivel 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 inicio 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

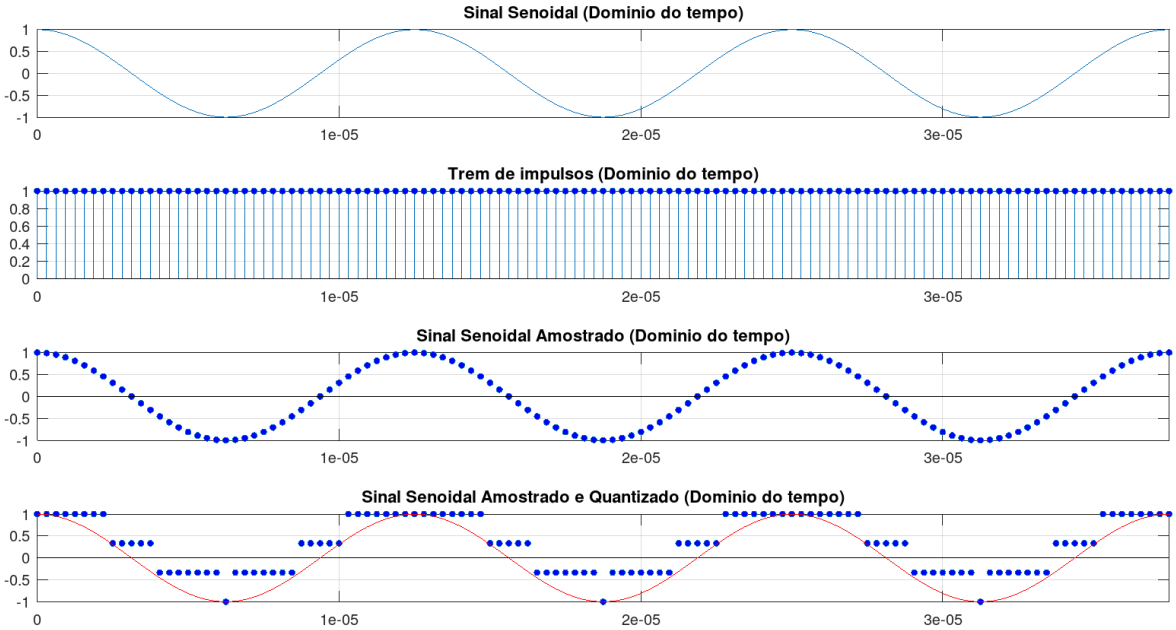
3.1. Conversão AD

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

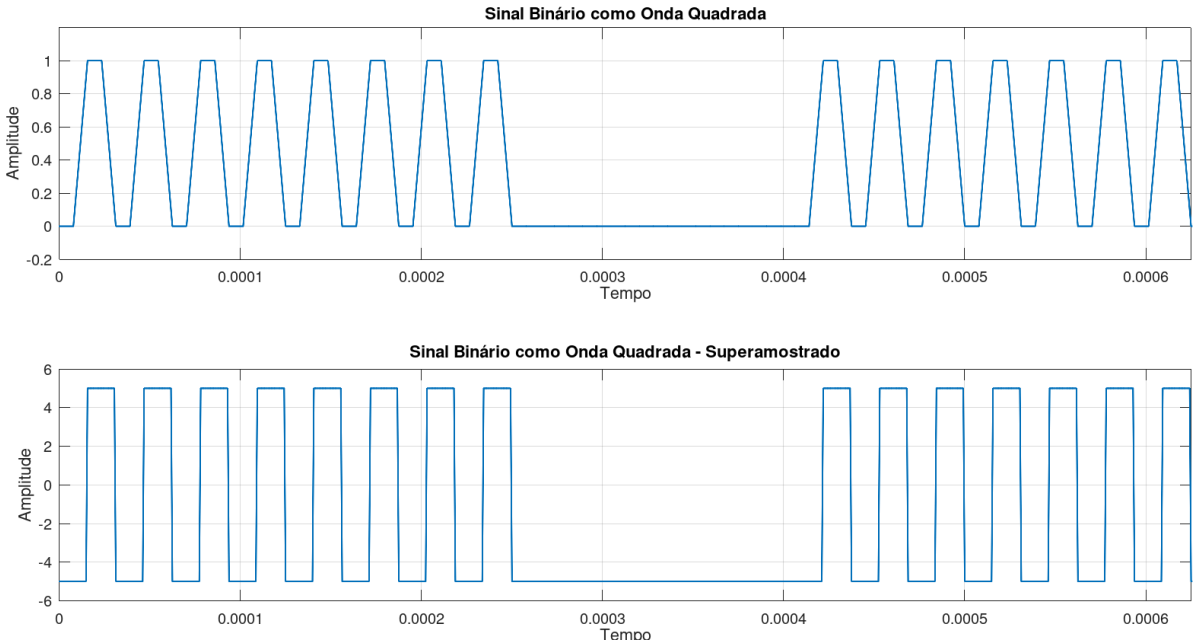
3.2. Transmissão PCM NRZ

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

O canal de comunicação PCM NRZ nada mais faz do que transmitir um nivel de amplitude especifico para cada valor de bit dado. Portanto, para a tramissão 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

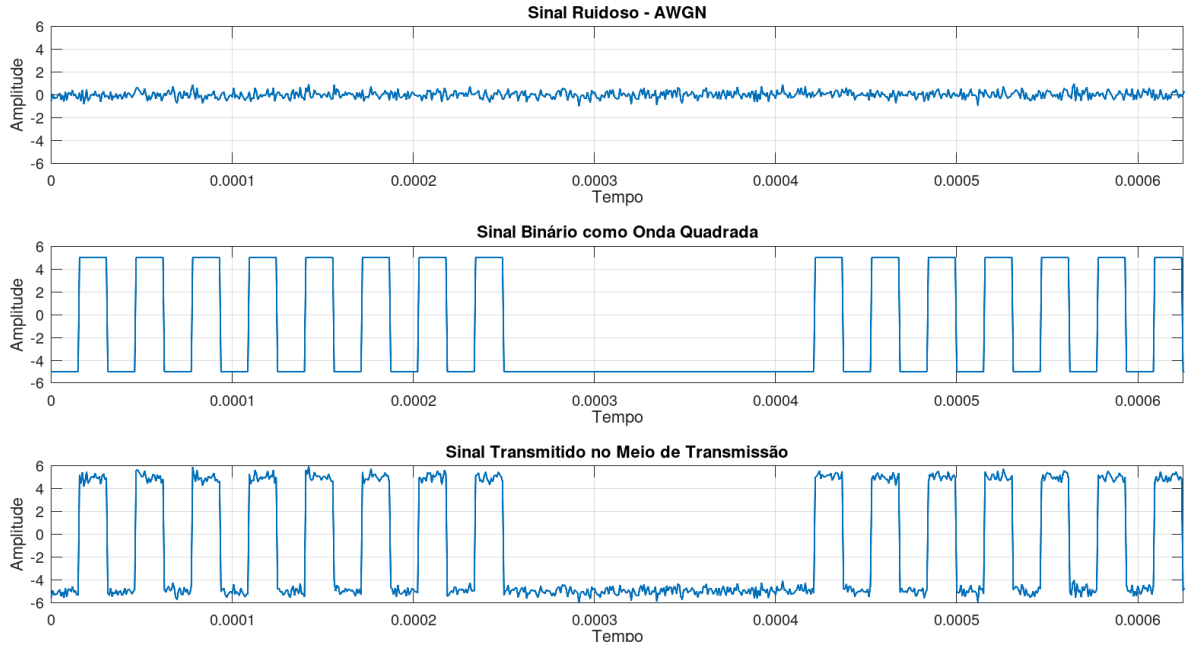
3.3. Ruído na Recepção:

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 aplitude 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

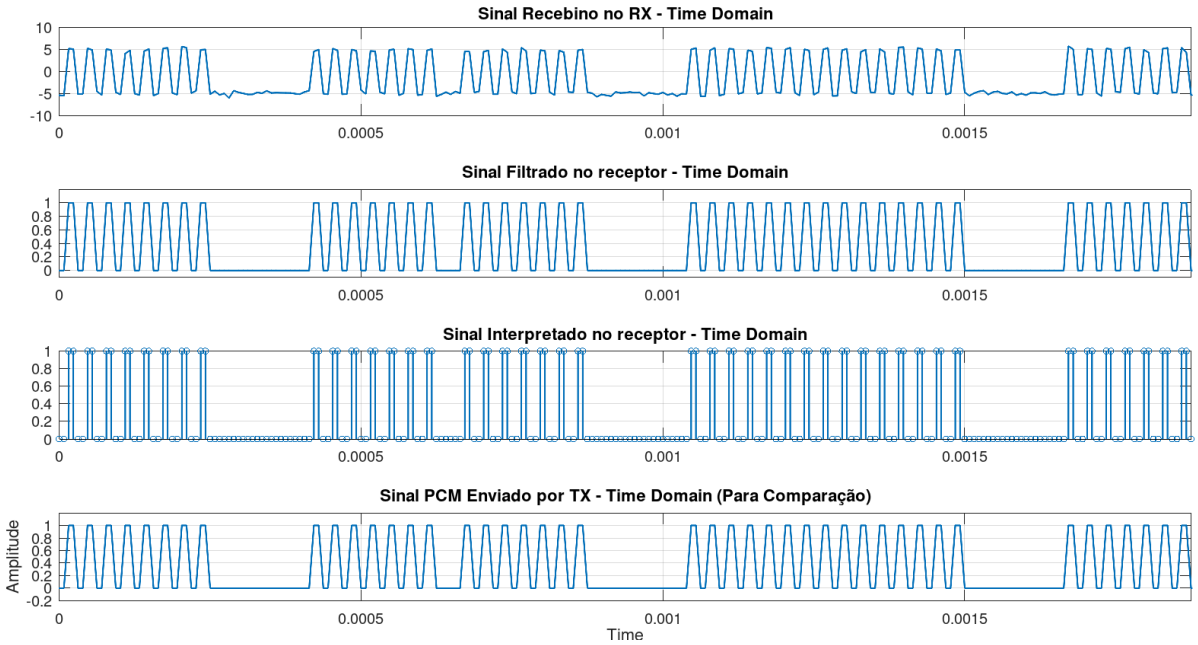
3.4. Reconstrução do sinal PCM

O sinal PCM transmitido é então reconstruido 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 reconstruido no receptor, note que abaixo do sinal PCM recebido e reconstruido há um plot do sinal transmitido antes de ser corrompido pelo ruído, para que possamos compara-los:

Figure 4: Elaborada pelo Autor



Sinal PCM Reconstruido no Receptor

3.5. Conversão DA:

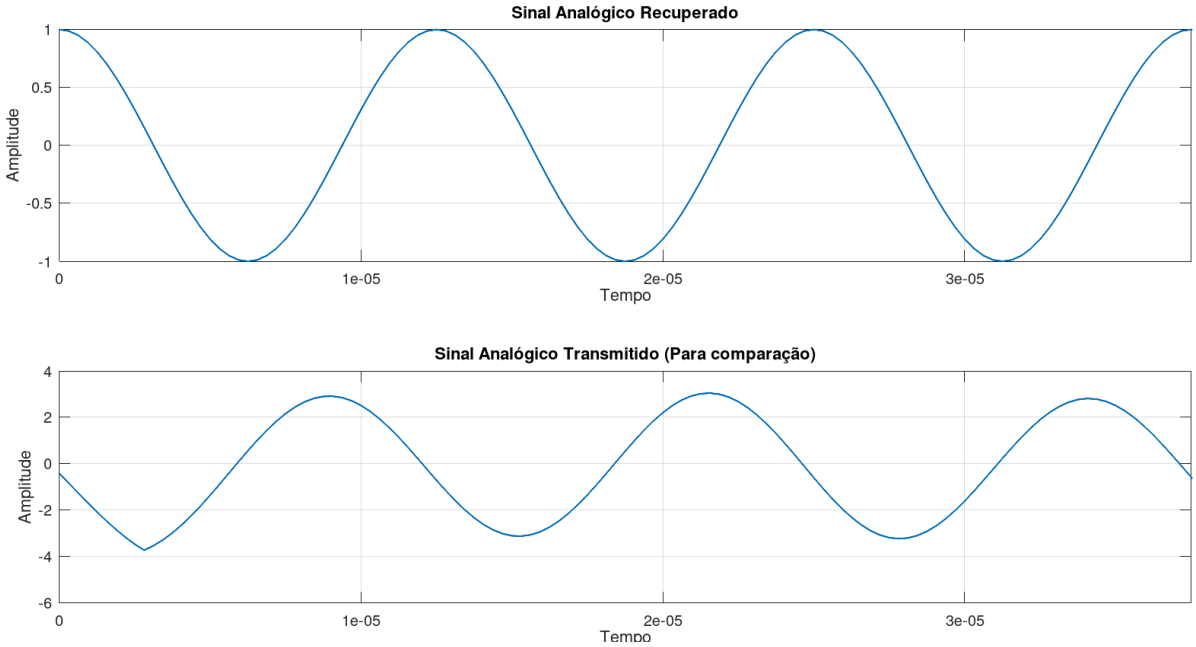
Uma vez com o trem de bits reconstruido, é necessário converter o sinal digital para analógico, para isso, realizamos um agrupamento dos bits de acordo com a ordem de transmissão.

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

Uma vez agrupados, definimos um limiar para cada nivel, no caso de 4bits, são 2^b niveis, portanto diferentes níveis possiveis 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 Reconstruido 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.

```
close all; clear all; clc;
pkg load signal;
pkg load communications;

% Altera o tamanho da fonte nos plots para 15
set(0, 'DefaultAxesFontSize', 20);

% Definindo a amplitude do sinal senoidal
A_signal = 1;

% Definindo a frequência do sinal senoidal
f_signal = 80000;

% Definindo a frequência de amostragem
fs = 40*f_signal;
Ts = 1/fs;
T = 1/f_signal;

% Definindo o tempo inicial e final do sinal
t_inicial = 0;
t_final = 0.01;

% Vetor de tempo
```

```
t = [t_inicial:Ts:t_final];

% Criando o sinal senoidal
signal = A_signal*cos(2*pi*f_signal*t);

% Criando um trem de impulsos com período de 2T
impulse_train = zeros(size(t));
impulse_train(mod(t, 1/fs) == 0) = 1;

% Amostragem do sinal senoidal
signal_sampled = signal .* impulse_train;

% Quantidade de níveis desejada (tirando o 0)
n=4;
num_levels = 2^n;

% Gerando os níveis de quantização automaticamente
levels = linspace(-1, 1, num_levels);

% Verifica se o vetor possui algum elemento com "0".
% Se sim, remove o elemento e sai do loop
for i = 1:length(levels)
    if levels(i) == 0
        levels(i) = [];
        break;
    end
end

% Quantização do sinal com 2^n niveis
quantized_signal = zeros(size(signal_sampled));
for i = 1:length(signal_sampled)
    for j = 1:length(levels)
        if signal_sampled(i) <= levels(j)
            quantized_signal(i) = levels(j);
            break;
        end
    end
end

% Plotando os sinais
figure(1)
subplot(411)
plot(t,signal)
grid on;
xlim([0 3*T])
title('Sinal Senoidal (Dominio do tempo)')

subplot(412)
stem(t,impulse_train, 'MarkerFaceColor', 'b')
grid on;
xlim([0 3*T])
title('Trem de impulsos (Dominio do tempo)')

subplot(413)
stem(t,signal_sampled, 'LineStyle','none', 'MarkerFaceColor', 'b')
grid on;
xlim([0 3*T])
title('Sinal Senoidal Amostrado (Dominio do tempo)')

```

```
subplot(414)
stem(t,quantized_signal, 'LineStyle','none', 'MarkerFaceColor', 'b')
xlim([0 3*T])
hold on;
plot(t,signal, 'r')
xlim([0 3*T])
grid on;
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.

```
% Desloca o vetor quantizado para 0 ou mais:

min_value = min(quantized_signal);
quantized_signal = quantized_signal - min_value;

% Multiplica os valores quantizados para que sejam números inteiros

% Número de intervalos entre os níveis
num_intervals = num_levels - 1;

% Multiplicando os valores quantizados para que sejam números inteiros
quantized_signal_int = quantized_signal * num_intervals;

% Convertendo valores quantizados inteiros para binário
binary_signal = de2bi(quantized_signal_int, n);

% Concatenando os valores binários em um único vetor
binary_signal_concatenated = reshape(binary_signal.', 1, []);

% Vetor de tempo
t = linspace(0, 1, length(binary_signal_concatenated) * 2);

% Repetindo cada valor do sinal
repeated_signal = reshape(repmat(binary_signal_concatenated, 2, 1), 1, []);

% ========================================================
% realizando a superamostragem do sinal com a função upper

% Superamostragem
n = 10;
amplitude =5;
repeated_signal_up = upsample(repeated_signal, n);

filtr_tx = ones(1, n);
filtered_signal  =  filter(filtr_tx,  1,  repeated_signal_up)*2*amplitude-
amplitude;

% criando um novo vetor de t para o sinal filtrado
t_super = linspace(0, 1, length(filtered_signal));

var_noise = 0.1;
```

```
transmission_noise = sqrt(var_noise)*randn(1,length(filtered_signal));

% Gerando o sinal transmitido no meio de comunicação multiplicando o sinal de transmissão pelo ruído do meio.
transmitted_signal = filtered_signal + transmission_noise;

% Plotando o sinal
figure(2)
subplot(311)
plot(t,repeated_signal, 'LineWidth', 2);
ylim([-0.2, 1.2]);
xlim([0, 50*T]);
xlabel('Tempo');
ylabel('Amplitude');
title('Sinal Binário como Onda Quadrada');
grid on;

subplot(312)
plot(t_super,filtered_signal, 'LineWidth', 2);
xlim([0, 50*T]);
ylim([-amplitude*1.2 , amplitude*1.2]);
xlabel('Tempo');
ylabel('Amplitude');
title('Sinal Binário como Onda Quadrada');
grid on;

subplot(313)
plot(t_super,transmitted_signal, 'LineWidth', 2);
xlim([0, 50*T]);
ylim([-amplitude*1.2 , amplitude*1.2]);
xlabel('Tempo');
ylabel('Amplitude');
title('Sinal Binário como Onda Quadrada');
grid on;
```

4.3. Rúido 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.

```
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.

```
% Definindo o limiar (valor que vai decidir se o sinal é 0 ou 1)

limiar = 0;
% amostrando o sinal recebido
received_signal = transmitted_signal(n/2:n:end);

```

```
% Verifica se o sinal recebido é maior ou menor que o limiar.
% Se for maior, o sinal é 1, se for menor, o sinal é 0.
received_binary = received_signal > limiar;

% Vetor de tempo para o sinal recebido
t_received = linspace(0, 1, length(t_super)/n);

% Plotando o sinal
figure(3)
subplot(211)
plot(t_received, received_signal);
xlim([0, 50*T]);
subplot(212)
stem(t_received, received_binary);
xlim([0, 50*T]);

% Calculando a taxa de erro de bit
num_erro = sum(xor(received_signal, received_binary))
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.

```
% Vetor de tempo para o sinal recebido
t_received = linspace(0, 1, length(received_signal));

% Interpolando o sinal para restaurar a taxa de amostragem original
received_signal_interp = interp(received_signal, n);

% Filtrando o sinal interpolado para remover artefatos
filtr_rx = ones(1, n);
received_signal_filtered = filter(filtr_rx, 1, received_signal_interp) / n;

% Plotando o sinal recuperado
figure(4);
subplot(211);
plot(t_super, filtered_signal, 'LineWidth', 2, 'DisplayName', 'Sinal Original');
hold on;
plot(t_received, received_signal_filtered(1:length(t_received)), '--', 'LineWidth', 2, 'DisplayName', 'Sinal Recuperado');
xlim([0, 50*T]);
ylim([-amplitude*1.2 , amplitude*1.2]);
xlabel('Tempo');
ylabel('Amplitude');
title('Sinal Original e Sinal Recuperado');
legend;
grid on;

% Convertendo o sinal recuperado para o formato analógico
received_analog_signal = received_signal_filtered(1:length(t_received)) * (2*amplitude) - amplitude; % Normalização da amplitude

```

```
% Plotando o sinal analógico recuperado
subplot(212);
plot(t_received, received_analog_signal, 'LineWidth', 2);
xlim([0, 50*T]);
xlabel('Tempo');
ylabel('Amplitude');
title('Sinal Analógico Recuperado');
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