Universidade de Brasília
Gustavo Pereira Chaves – 19/0014113
David Gonçalves Mendes – 19/0056967

Simulador da Camada Física

Brasília
2022

1. Introdução

Nesse trabalho foi desenvolvido um simulador de Camada Física do Modelo OSI na linguagem C++, que busca transmitir e receber determinada mensagem codificada.

Para enviar os dados, foi implementada uma Camada de Aplicação que transforma a mensagem passada em bits para a Camada Física. Essa, por sua vez, possui três protocolos de codificação (Manchester, Binário e Bipolar) que vão transformar esse trem de bits para serem enviados pelo canal de comunicação (no caso do programa implementado, é apenas uma função que copia os bits transmitidos para a Camada Física Receptora).

Já na recepção da mensagem, são também implementados protocolos de decodificação da mensagem, baseados também no tipo de codificação utilizado. Com isso, obtêm-se o trem de bits original, que é passado para a Camada Física receptora, convertendo-a novamente para caracteres legíveis e mostrando a mensagem original.

2. Implementação

O simulador foi implementado de forma que cada protocolo foi descrito como uma rotina, de forma que após determinada transformação é chamada uma nova função que recebe a mensagem transformada. Foram implementadas também funções auxiliares para evitar a repetição de certas partes do código.

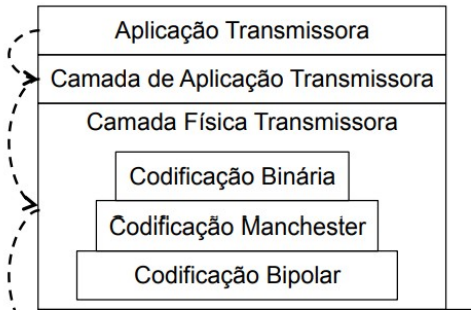
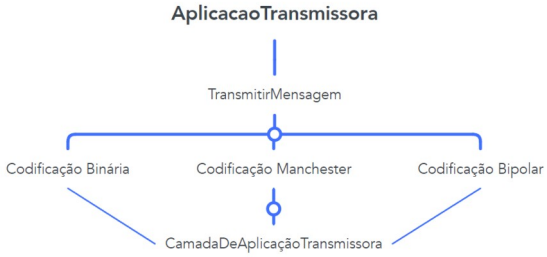
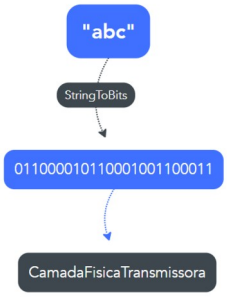


Figura 1: Representação das Funções Implementadas e suas Saída

Inicialmente é chamada a função **AplicacaoTransmissora** que lê uma mensagem e chama a função auxiliar **TransmitirMensagem**, alterando uma variável global criada para a codificação a ser utilizada em todo o programa. A função aplicação transmissora realiza essa chamada para todas as codificações a serem testadas.

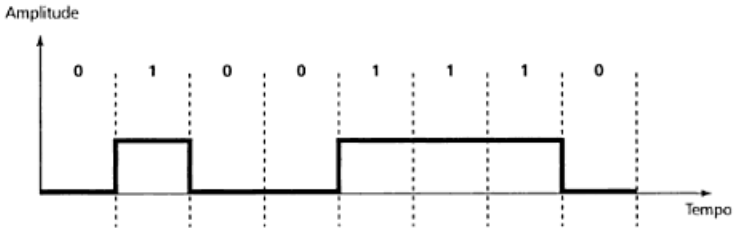


Nessa camada a mensagem é convertida para um trem de bits utilizando a função auxiliar **StringToBits**, que gera um vetor de inteiros de 1's e 0's. Após isso, os bits gerados são enviados para a função **CamadaFisicaTransmissora**.



Na camada física transmissora o trem de bits recebido é codificado de acordo com o valor da variável global definida, sendo três casos possíveis:

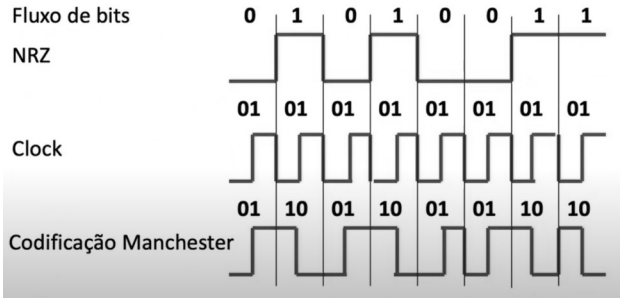
- Codificação Binária:



Para cada bit 1 gera-se uma tensão de V e para o bit 0 uma tensão zero. É uma codificação que pode apresentar grandes sequências de 1's e 0's podendo ocorrer perda de sincronia entre a transmissão e recepção da mensagem.

Para representar essa codificação foi criada a função **CamadaFisicaTransmissoraCodificacaoBinaria**, que gera um vetor com as tensões que esse sinal deveria gerar.

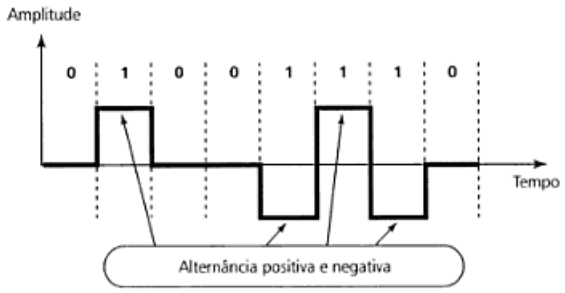
- Codificação Manchester:



É realizado um XOR entre o clock e o trem de bits da mensagem, gerando-se uma tensão V ou zero. Essa codificação ainda apresenta problemas de sincronização para grandes sequências de 1's e 0's.

A função implementada para simular esse comportamento é a **CamadaFisicaTransmissoraCodificacaoManchester**.

- Codificação Bipolar:



Nessa codificação, o bit 0 é convertido para a tensão zero e os bits 1's são sempre alternados: ora valem a tensão V, ora -V. Dessa forma essa codificação é chamada de balanceada.

Foi criada a função **CamadaFisicaTransmissoraCodificacaoBipolar** para realizar essa transformação, que gera o vetor de tensões assim como explicitado.

Após a mensagem passar por uma das três codificações, é chamada a função **MeioDeComunicação**, que apenas copia as informações de um ponto A (camada física transmissora) para um ponto B (camada física receptora), representados por dois vetores.

Esse vetor do ponto B é passado para a função **CamadaFisicaReceptora** que tem por objetivo, decodificar a mensagem, utilizando um dos três processos seguintes:

- Decodificação Binária:

Sendo o processo mais simples, para a decodificação desse sinal basta converter a tensão V para bits 1's e a tensão zero para bits 0's

- Decodificação Manchester:

Devemos analisar qual foi a saída da codificação Manchester para cada bit de entrada. Realizando o XOR de "1" com o clock obtemos a saída "10". Já para a entrada "0" obtemos "01". Dessa forma, pode-se perceber que o primeiro bit de cada saída carrega a informação original e o segundo deve ser ignorado. Assim, para decodificar a mensagem deve-se ler um bit e pular um bit sucessivamente até o fim do trem de bits.

- Decodificação Bipolar:

Os bits zeros sempre valem zero, logo são facilmente decodificáveis. Os bits 1's são sempre alternados entre V e -V. Logo, para qualquer tensão diferente de zero, o bit decodificado deve ser "1". Dessa forma obtemos a resposta facilmente.

Por fim, é chamada a função **CamadaAplicacaoReceptora**, com o resultado da operação anterior, que converterá a mensagem recebida em bits para uma string. Essa saída é passada para a **AplicacaoReceptora**, que apresenta a mensagem final no console.

3. Atividades desenvolvidas por cada Membro

- Gustavo Pereira Chaves:
  - Desenvolvimento dos algoritmos de decodificação e funções auxiliares;
  - Organização dos arquivos do projeto;
  - Confecção da parte inicial do relatório (Introdução e parte da implementação);

- David Gonçalves Mendes:
  - Desenvolvimento dos algoritmos de codificação;
  - Confecção da parte final do relatório (Parte da implementação e conclusão)

4. Conclusão

Nesse trabalho, pudemos simular com êxito o processo do fluxo de bits na camada física através da implementação de funções de codificação e decodificação para protocolos distintos (Manchester, Bipolar e Binária), que são usadas, respectivamente, na transmissão e na recepção do quadro de bits.

A principal dificuldade encontrada ao longo do projeto foi a abstração dos algoritmos de codificação e decodificação para realizar as suas respectivas transposições para o formato de código. Além disso, outro impedimento encontrado foi ler a mensagem como bits, sendo necessário, para contornar esse problema, criar uma função auxiliar para realizar essa conversão.