Universidade de Brasília

Gabriel Alves Castro - 251030910

Jonathas Wancler Borges dos Santos 200020960

Mario Augusto Vieira dos Santos - 231035778

Teleinformática e Redes 1

Simulador de redes TR

Brasília

2025

SUMÁRIO

1. INTRODUÇÃO

O presente trabalho tem como objetivo o desenvolvimento e a implementação de um simulador de comunicação de dados focado nas camadas Física e de Enlace do modelo OSI.

Descrição do Problema

O problema central abordado por este projeto é a simulação do processo de transmissão de dados entre um emissor e um receptor através de um canal de comunicação sujeito a ruídos. A comunicação digital enfrenta desafios inerentes, como a necessidade de sincronização entre as pontas, a adaptação do sinal digital ao meio físico e a garantia da integridade da informação transmitida. Portanto, o desafio consiste em implementar algoritmos que tratem o sinal bruto, organizem os dados em quadros lógicos e detectem ou corrijam erros que possam surgir durante a transmissão.

Visão Geral do Simulador

O simulador desenvolvido opera sobre uma arquitetura cliente-servidor, composta por três módulos principais: uma Interface Gráfica (GUI), uma camada de enlace e uma camada física com seus respectivos transmissores e receptores. O funcionamento do sistema segue o fluxo abaixo:

1. **Entrada de Dados:** O usuário insere uma mensagem de texto e seleciona os parâmetros de configuração (tipos de modulação, enquadramento e controle de erro) através da interface gráfica.

2. **Processamento no Transmissor:** A mensagem é convertida para binário e processada pela **Camada de Enlace**, onde recebe bits de redundância para controle de erros (Paridade, Checksum, CRC ou Hamming) e é organizada em quadros (Contagem de Caracteres ou Inserção de Bytes/Bits). Em seguida, na **Camada Física**, o sinal é codificado (NRZ, Manchester ou Bipolar) e modulado (ASK, FSK, etc.) para transmissão.

3. **Canal de Comunicação:** O sinal modulado trafega por um meio simulado onde é possível injetar ruído gaussiano, testando a robustez dos algoritmos de correção.

4. **Processamento no Receptor:** O receptor capta o sinal, realiza a demodulação e decodificação de linha. Posteriormente, a Camada de Enlace

verifica a existência de erros e, se possível, os corrige, depois desenquadra os bits e entrega a mensagem final ao usuário

Esta abordagem permite visualizar passo a passo as transformações sofridas pelos dados, oferecendo uma ferramenta didática para a análise de desempenho e confiabilidade dos diferentes protocolos implementados.

2. IMPLEMENTAÇÃO

Esta seção detalha a estrutura técnica e o funcionamento do simulador desenvolvido. A solução foi construída utilizando uma arquitetura modular, dividida em Interface Gráfica, Camada Física e Camada de Enlace, permitindo simular os processos de cada camada.

Para facilitar a compreensão da arquitetura do sistema e do caminho percorrido pelos dados durante a transmissão e recepção, apresentamos os diagramas ilustrativos a seguir.

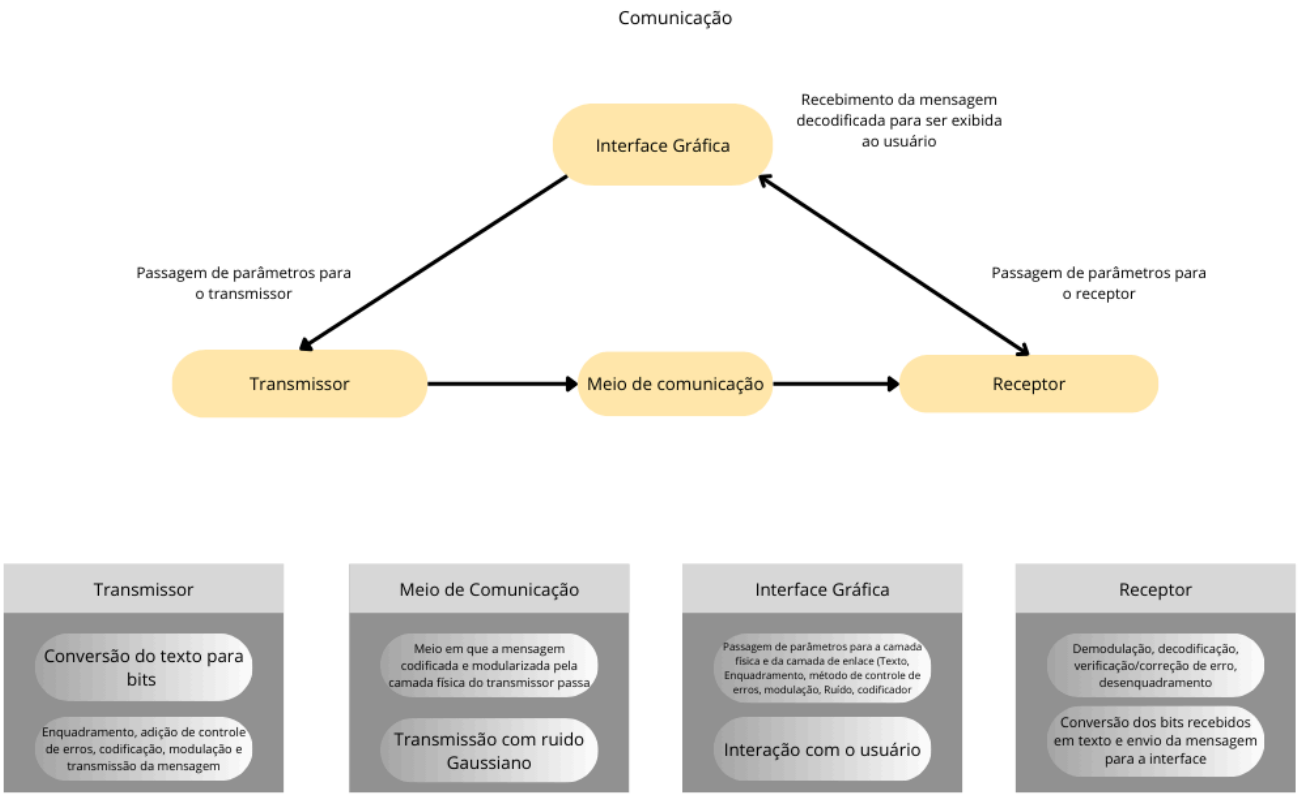

Diagrama 1: Comunicação entre a interface gráfica e o transmissor/receptor

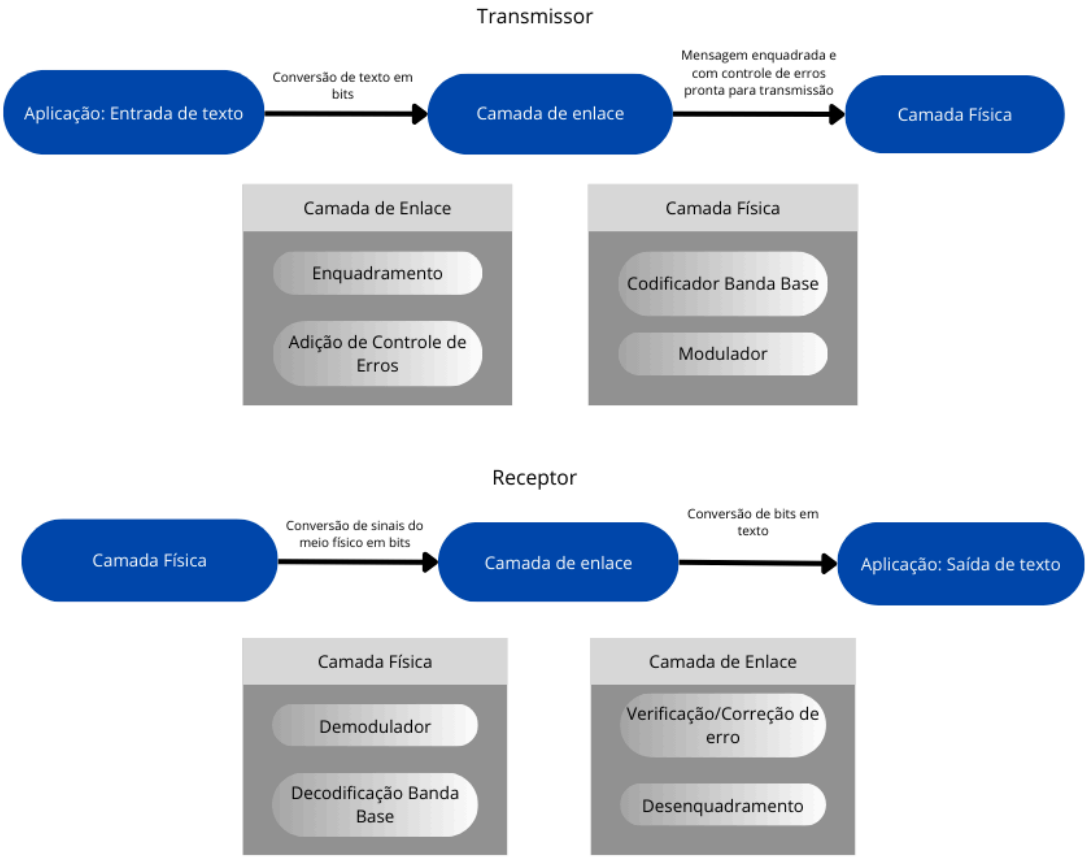

Diagrama 2: Funcionamento do transmissor e do receptor

2.1. Camada de enlace

A Camada de Enlace de Dados (*Data Link Layer*) é responsável pela transferência confiável de informações através do canal físico. No âmbito deste simulador, ela atua como uma interface entre a aplicação (texto) e a camada física (sinais), executando duas funções primordiais: o **Enquadramento** (*Framing*), que delimita o início e o fim das mensagens, e o **Controlo de Erros** (Detecção e correção), que garante a integridade dos dados recebidos.

2.1.1. Métodos Implementados

O sistema foi desenvolvido de forma modular, permitindo a seleção dinâmica entre diferentes protocolos:

- **Enquadramento:** Contagem de Caracteres, Inserção de Bytes (*Byte Stuffing*) e Inserção de Bits (*Bit Stuffing*).
- **Detecção de Erros:** Bit de Paridade Par, Checksum (Soma de Verificação) e CRC-32.

- **Correção de Erros:** Código de Hamming.

2.1.2. Análise da Implementação

A implementação foi realizada em Python, utilizando manipulação direta de *strings* de bits e listas de inteiros para simular o processamento lógico dos dados. O fluxo de dados segue uma arquitetura de *pipeline*: os dados brutos passam primeiro pelo bloco de Controle de Erros e, subsequentemente, pelo bloco de Enquadramento antes de serem enviados à Camada Física.

Algoritmos de Enquadramento

- **Contagem de Caracteres:** A função `enquadrar_contagem_caracteres` calcula o comprimento total da mensagem em bytes. Este valor é convertido num cabeçalho binário de 8 bits e anexado ao início do quadro. A implementação inclui uma função auxiliar (`_bits_para_lista_de_bytes`) que garante o alinhamento dos dados, adicionando *padding* (zeros à esquerda) caso o número de bits não seja múltiplo de 8.
- **Inserção de Bytes (*Byte Stuffing*):** Este método utiliza a *flag* `0x7E` (padrão PPP) para delimitar o quadro. A implementação percorre a lista de bytes da mensagem e aplica o mecanismo de escape: sempre que um byte de dados coincide com a *flag* (`0x7E`) ou com o próprio byte de escape (`0x7D`), o sistema insere um byte de escape extra antes dele. Isso garante a transparência dos dados, permitindo que qualquer padrão binário seja transmitido no *payload*.
- **Inserção de Bits (*Bit Stuffing*):** Implementado para operar em nível de bit, este método utiliza a sequência `01111110` como delimitador. O algoritmo de transmissão varre a string de bits e insere um '0' sempre que encontra cinco '1's consecutivos nos dados (`11111` → `111110`). O receptor (`desenquadrar_bit_stuffing`) monitora o fluxo e remove este '0' artificial, recuperando a sequência original e prevenindo falsas detecções de fim de quadro.

Algoritmos de Controle de Erros (Detecção e correção)

- **Bit de Paridade Par:** Implementado através de operações

lógicas XOR. O transmissor faz a iteração sobre todos os bits da mensagem; se a contagem de '1's for ímpar, o bit de paridade (anexado ao final) é definido como '1', tornando o total par. O receptor verifica se a paridade do quadro recebido se mantém par e retorna o valor.

- **CRC-32 (IEEE 802.3):** A implementação utiliza operações *bitwise* (deslocamento e XOR) para simular a divisão polinomial pelo padrão `0x104C11DB7`.
    - *Estratégia de Padding:* Baseado na lógica estudada, o algoritmo inclui uma verificação de tamanho mínimo. Se a mensagem for muito curta (menos de 64 bits), é adicionado um *padding* determinístico (sequência alternada de 0s e 1s) antes do cálculo do CRC. Isso visa manter a estabilidade da sincronização na camada física. O tamanho deste *padding* é calculado e transmitido num cabeçalho dedicado, permitindo que o receptor descarte os bits extras após validar o CRC.
- **Checksum (Soma de Verificação):** Implementado utilizando a aritmética de complemento de um. A mensagem é dividida em blocos de 8 bits que são somados. O *carry* (estouro) de cada soma é adicionado de volta ao bit menos significativo. O resultado final é invertido (complemento) e enviado como *checksum*. O receptor realiza a mesma soma (incluindo o *checksum*); se o resultado for uma sequência de 1s (ou `0xFF` em hexadecimal), a integridade é confirmada.
- **Código de Hamming:** O algoritmo foi desenvolvido para correção de erro único (SEC). O código calcula a quantidade necessária de bits de redundância (r) e os insere nas posições que são potências de 2 (1, 2, 4, 8...). Os valores destes bits são derivados da paridade dos dados que eles "cobrem". No receptor, o recálculo das paridades gera uma "síndrome"; se a síndrome for zero, os dados estão corretos; caso contrário, o valor da síndrome aponta o índice exato do bit errado, que é invertido e corrigido via software.

2.2. Camada física

A Camada Física é responsável pela conversão entre bits/quadros e sinais

analógicos que trafegam pelo meio. No simulador, ela implementa a geração (modulação) e recuperação (demodulação) de formas de onda digitais e por portadora, além de regras básicas de amostragem e alinhamento temporal. Atua diretamente sobre vetores de amostras (numpy arrays) que representam sinais discretizados no tempo.

2.2.1 Métodos Implementados

Modulações Digitais em Nível de Símbolo:

- NRZ-Polar: mapeamento binário para níveis +A / −A com repetição de 100 amostras por bit.
- Manchester: codificação por transição dentro do símbolo (metade do símbolo positivo/negativo).
- Bipolar AMI: pulsos alternados para ‘1’ e zero para ‘0’ (AMI), com repetição por amostras.

Modulações por Portadora:

- ASK (On-Off Keying): presença/ausência de portadora de amplitude A por bit.
- FSK: duas frequências f1/f2 (uma por valor de bit).
- PSK: fase 0 / π por bit (BPSK).
- QPSK: mapeamento Gray de pares de bits para componentes I/Q; modulação coerente com portadora seno/cosseno.
- 16-QAM: mapeamento Gray para 4 níveis I e Q (−3,−1,1,3) e combinação com portadora.

Demodulação (recepção):

- NRZ/AMI/ASMK: média por bloco de 100 amostras e decisão por limiar.
- Manchester: demodulação pela forma (metade positiva/negativa).
- FSK/PSK: correladores simples com formas de referência; FSK usa duas referências (f1, f2).
- QPSK/QAM: correlador coerente com cosseno/seno, normalização por energia e quantização ao nível mais próximo; QAM utiliza mapa Gray inverso para recuperar bits.

2.2.2 Análise da Implementação

Representação e Amostragem:

- Todos os sinais são vetores NumPy; cada bit/símbolo é representado por 100 amostras por padrão (parametrizável).

- A escolha de 100 amostras por símbolo simplifica sincronização (segmentação por índices inteiros) e testes, mas pressupõe sincronismo de tempo perfeito entre transmissor e receptor.

Decisões de Projeto:

- Uso de correladores coerentes para PSK/QPSK/QAM melhora robustez contra ruído quando a fase/freq carr. é conhecida.
- Para esquemas amplitude-sensíveis (ASK, bipolar) a demarcação por média e limiar permite implementação simples.
- QPSK e 16-QAM usam mapeamento Gray para reduzir probabilidade de erro de bit por erro de símbolo.
- Implementação modular: funções separadas para cada esquema tornam fácil selecionar/modificar técnicas.

Tratamento de Ruído e Robustez:

- O código contém pontos de extensão para injeção de ruído (AWGN) — podem ser adicionados antes da demodulação.
- Tolerâncias simples (p.ex. threshold relativo a A) são usadas; isso funciona com SNR moderado, mas falha em baixo SNR ou deslocamentos de ganho.
- A demodulação QAM usa decisão por “nearest level” (mínima distância) em I/Q; imprime valores estimados (debug) que ajudam a ajustar parâmetros.

Parâmetros e Assunções

- Amplitude de referência A e frequência de portadora f são parâmetros de entrada para módulos de modulação/demodulação.
- samples_per_symbol fixo em 100 (deve ser par para Manchester).
- Assume-se sincronização de símbolo (início dos blocos conhecido) e, para demodulação coerente, sincronização de portadora (fase/frequência conhecidas/estáveis).
- Não há implementação completa de AGC, sincronismo fino, ou equalização de canal no código atual.

2.2.3 Limitações e Possíveis Melhoria

Limitações conhecidas:

- **Sincronismo**: o receptor exige alinhamento por símbolos e sincronização de portadora; sem isso o desempenho cai rapidamente.
- **Decisões simples**: limiares fixos e correlação sem normalização fina podem falhar em condições com variação de ganho/SNR.

- **NRZ, bipolar e Manchester Incoerentes na integração:** As funções de manchester, bipolar e NRZ foram corretamente implementadas utilizando 100 amostras por símbolo por padrão. No entanto, devido à falta de tempo, foram utilizadas versões simplificadas com apenas uma amostra por símbolo para essas três funções, o que limitou a simulação de ruídos na camada física. O código completo da camada física, sem nenhuma limitação e como descrito nesse relatório se encontra na pasta “camada_fisica_completa”, e ele não foi utilizado completamente para a confecção do simulador final.
- **Ruído:** No simulador, não foi possível, devido ao tempo, adicionar ruído gaussiano para as modulações digitais. No entanto, isso foi realizado em código, no transmissor para todas as modulações por portadora. Foi adicionado no simulador, um ruído aleatório para testar a camada de enlace, mas este, é separado do ruído gaussiano simulando um meio físico real. Tal como o que foi implementado, de maneira limitada, nas modulações por portadora.

3. Simulador

O **Módulo de Simulação** é responsável pela criação, configuração e coordenação das duas principais entidades funcionais do sistema: o **Transmissor** e o **Receptor**. Essas entidades representam os pontos de comunicação da simulação e fazem uso direto das funções já implementadas nas camadas Física e de Enlace.

A função central do módulo é **integrar os componentes desenvolvidos** nas camadas inferiores e posicioná-los dentro de um fluxo completo de transmissão e recepção de dados. Para isso, o módulo inicializa e gerencia tanto o ambiente de comunicação quanto as rotinas de processamento necessárias para simular o comportamento real de um enlace digital.

3.1. responsabilidades

1. Instanciação das entidades
    a. Cria os objetos correspondentes ao Transmissor e ao Receptor.
    b. Configura seus parâmetros iniciais, como métodos de modulação, técnicas de enquadramento e algoritmos de detecção/correção de erros.

2. Orquestração do processo de comunicação
    a. Coordena a execução das etapas da camada Física (codificação e decodificação de sinais).
    b. Garante a correta interação com a camada de Enlace (enquadramento, inserção de bits de verificação, correção de erros, etc.).

c. Controla o fluxo da simulação, desde a preparação da mensagem até a entrega final dos dados reconstruídos.

3. Gerenciamento da comunicação via Socket
    a. Estabelece o canal de comunicação entre Transmissor e Receptor utilizando sockets TCP.
    b. Simula o processo de envio e recepção em um ambiente semelhante a uma rede real, possibilitando testes de confiabilidade e desempenho.
4. Integração com interface gráfica
    a. Possibilita que ações do usuário (como enviar mensagens, escolher modulações e habilitar mecanismos de detecção de erros) acionem diretamente as rotinas da simulação.
    b. Atualiza a interface com informações de debug e estados da transmissão.

3.2. Transmissor

O transmissor é responsável por preparar a mensagem para envio, transformando dados textuais em um sinal modulável pela camada física. Seu funcionamento segue um pipeline organizado que integra as funções das camadas de Enlace e Física.

3.2.1. Pipeline do Transmissor

1. Conversão da mensagem em bits
   Texto UTF-8 é convertido em uma sequência binária.

2. Enquadramento
   Define os limites do quadro a ser transmitido, aplicando técnicas como Contagem de Caracteres, Bit-Stuffing ou Byte-Stuffing.

3. Correção e detecção de erros
   O transmissor pode aplicar:

    a. Correção (Hamming)

    b. Detecção (Paridade, Checksum, CRC-32)

4. Inserção de ruído (opcional)
   Permite simular erros e avaliar robustez das técnicas utilizadas.

5. Modulação digital
   Os bits são convertidos em formas de onda digitais (NRZ, Bipolar, Manchester) ou símbolos (QPSK, 16QAM).

6. Modulação em portadora
   O sinal digital é deslocado para uma portadora usando ASK, FSK, PSK, QPSK ou 16QAM.

7. Envio via socket
   O transmissor empacota o sinal e os parâmetros utilizados e os envia ao receptor via TCP.

3.2.2. Características Gerais

1. Pipeline modular e configurável

2. Suporte a múltiplas técnicas de enlace e modulação

3. Capacidade de simular ruído

4. Registro estruturado das transformações aplicadas

3.3. Receptor

O receptor executa o fluxo inverso ao transmissor desfazendo todas as operações até recuperar a mensagem original. Ele reconstroi, avalia e interpreta os dados recebidos.

3.3.1. Pipeline do Receptor

1. Recebe o sinal modulado e a configuração enviada pelo transmissor.

2. Demodulação da portadora
   Recupera símbolos digitais a partir do sinal analógico recebido.

3. Demodulação digital
   Converte símbolos ou formas de onda em bits (NRZ, Bipolar, Manchester, QPSK, 16QAM).

4. Detecção e correção de erros
   Verifica integridade usando Paridade, Checksum ou CRC-32
   Corrige erros utilizando Hamming quando presente.

5. Desenquadramento
   Remove técnicas aplicadas no transmissor e restaura o quadro original.

6. Conversão para texto
   Bits → bytes → texto UTF-8, reconstruindo a mensagem final.

7. Retorno ao transmissor
   Envia a mensagem decodificada e o histórico das etapas realizadas.

3.3.2. Características Gerais

- Total compatibilidade com todas as técnicas do transmissor
- Pipeline completo de demodulação, verificação e reconstrução
- Registro detalhado das operações de RX
- Tratamento seguro de dados e recuperação de erros

3.3.3. conjunto Transmissor-Receptor

O módulo de simulação implementa um ciclo completo de comunicação digital, integrando:

- Enquadramento e desenquadramento
- Detecção e correção de erros
- Modulação digital e em portadora
- Adição de ruído
- Comunicação via socket
- Reconstrução da mensagem no destino

A arquitetura modular permite simular diferentes cenários, alterar técnicas da camada física e de enlace e observar seus efeitos na comunicação, oferecendo um ambiente versátil para experimentação e estudo.

3.4. Interface Gráfica

O simulador conta com uma **interface gráfica interativa**, que permite ao usuário:

- Escolher técnicas de enquadramento, detecção de erros e modulação
- Enviar mensagens em tempo real

- Visualizar logs de debug de cada etapa
- Acompanhar o fluxo completo entre transmissor e receptor
- Controlar parâmetros como nível de ruído e opções de modulação

A interface facilita o uso do simulador, reduz a complexidade de configuração e torna o processo de análise mais intuitivo e visual.

4. Membros

- Jonathas implementou a camada de enlace, auxiliou na concepção e organização do código, revisou todo o código e confeccionou a sessão da camada de enlace deste relatório.
- Gabriel implementou a camada física, auxiliou na concepção e organização do código, revisou todo o código e confeccionou a sessão da camada física deste relatório.
- Mario implementou o simulador e a interface GUI, auxiliou na concepção e organização do código, revisou todo o código e confeccionou a sessão do simulador deste relatório.

5. conclusão

O simulador desenvolvido permitiu compreender de forma prática o funcionamento integrado das camadas Física e de Enlace, demonstrando como técnicas de enquadramento, detecção/correção de erros e modulação influenciam a confiabilidade da comunicação digital.
Entre as principais dificuldades encontradas destacam-se:

- A sincronização entre transmissor e receptor;
- A integração entre GUI, sockets e as rotinas de sinal.
- A adição de ruído branco

Apesar das limitações, o simulador cumpre os objetivos propostos e fornece uma ferramenta didática útil para visualizar erros, ruídos e protocolos em funcionamento. Melhorias futuras podem expandir ainda mais suas capacidades. Uma das principais possibilidades é a introdução de sincronismo automático entre transmissor e receptor, reduzindo a dependência de configurações manuais e tornando o processo de recepção mais robusto a atrasos e variações do canal.