Roteiro do Projeto: Implementação do Algoritmo Ami Pseudoternário

Este roteiro detalha as etapas necessárias para a implementação do projeto, considerando as exigências fornecidas e o funcionamento do algoritmo Ami Pseudoternário.

Fase 1: Planejamento e Definição de Tecnologias

- **1.1 Escolha da Linguagem e Framework:**
  - **Recomendação:** Python com bibliotecas para GUI (Tkinter, PyQt, Kivy) e comunicação de rede (sockets). Python é versátil, possui bibliotecas robustas para todas as necessidades do projeto e é amplamente utilizado em ambientes desktop.
  - **Alternativas:** Java (Swing/JavaFX para GUI, Sockets para rede), C# (.NET/WPF para GUI, Sockets para rede), C++ (Qt para GUI, Sockets para rede).
- **1.2 Escolha da Biblioteca de Gráficos:**
  - **Recomendação:** Matplotlib (Python) ou similar. É excelente para plotar gráficos de forma de onda e é bem documentada.
- **1.3 Definição do Protocolo de Comunicação:**
  - **Recomendação:** TCP/IP via sockets. Garante entrega confiável e ordenada dos dados, essencial para a transmissão da mensagem. UDP pode ser considerado para cenários onde a velocidade é mais crítica que a confiabilidade, mas exigiria tratamento de perda de pacotes.
- **1.4 Pesquisa e Escolha do Algoritmo de Criptografia:**
  - **Recomendação:** Algoritmos simétricos como AES (Advanced Encryption Standard) ou DES (Data Encryption Standard) são boas opções para iniciantes devido à sua ampla disponibilidade em bibliotecas e bom desempenho. Para maior simplicidade inicial, um algoritmo de cifra de substituição simples (como Cifra de César ou Vigenère) pode ser usado para fins didáticos, mas AES/DES seriam mais adequados para o "ponto extra" de dificuldade.
  - **Considerações:** A criptografia deve ser implementada dentro do aplicativo, não usando ferramentas externas. A explicação do método escolhido é uma exigência.

Fase 2: Implementação Core do Algoritmo Ami Pseudoternário

- **2.1 Conversão de Texto para Binário (T5):**
  - Implementar função para converter strings (incluindo caracteres especiais e acentuados) para sua representação binária usando a tabela ASCII estendido (UTF-8 é uma boa escolha para cobrir o ASCII estendido e além).
  - Exemplo: `str.encode('utf-8')` em Python.
- **2.2 Codificação Ami Pseudoternário (T6):**
  - Desenvolver a lógica para aplicar o algoritmo Ami Pseudoternário à sequência binária. Isso inclui a alternância de polaridade para os bits '1' e a representação de '0's como zero.
  - Manter o estado da última polaridade utilizada para o bit '1'.
- **2.3 Decodificação Ami Pseudoternário (T8 - Parte 1):**
  - Implementar a lógica inversa para converter o sinal Ami Pseudoternário de volta para a sequência binária original.
- **2.4 Geração da Forma de Onda (T2):**
  - Criar uma função que gere os dados para plotagem do gráfico da forma de onda do sinal Ami Pseudoternário. Isso envolverá mapear cada bit codificado para um valor de tensão (+V, -V, 0V) por um determinado período de tempo.
  - A forma de onda deve ser gerada tanto para o processo de codificação quanto para o de decodificação.

Fase 3: Desenvolvimento da Interface Gráfica (GUI) e Visualização (T1)

- **3.1 Design da Interface:**
  - Criar uma interface intuitiva que permita ao usuário digitar o texto de entrada.
  - Exibir os seguintes campos de saída:
    - Mensagem escrita original.
    - Mensagem criptografada.
    - Mensagem em binário.
    - Mensagem com o algoritmo Ami Pseudoternário aplicado (representação numérica do sinal).
- **3.2 Integração do Gráfico (T1, T2):**
  - Incorporar o gráfico da forma de onda na GUI. O gráfico deve ser atualizado dinamicamente à medida que a mensagem é processada.
  - Garantir que o gráfico mostre claramente a forma de onda do sinal codificado e, no lado receptor, a forma de onda do sinal decodificado.

Fase 4: Implementação da Criptografia (T4)

- **4.1 Implementação do Algoritmo de Criptografia:**
  - Codificar o algoritmo de criptografia escolhido (ex: AES, DES ou um algoritmo mais simples para fins didáticos).
  - Implementar as funções de criptografia e descriptografia.
- **4.2 Integração com o Fluxo do Projeto:**
  - A criptografia deve ser aplicada antes da conversão para binário no Host A.
  - A descriptografia deve ser aplicada depois da conversão de binário para texto no Host B.

Fase 5: Comunicação em Rede (T3, T7)

- **5.1 Configuração de Sockets:**
  - Implementar a comunicação de rede usando sockets TCP/IP.
  - Configurar um "Host A" (cliente/servidor, dependendo da arquitetura) e um "Host B" (servidor/cliente).
  - Garantir que a comunicação funcione entre dois ou mais computadores reais, não apenas `localhost`.
- **5.2 Envio e Recepção da Mensagem (T7):**
  - No Host A, enviar a mensagem codificada pelo Ami Pseudoternário através da rede.
  - No Host B, receber a mensagem da rede.

Fase 6: Processo Inverso e Desmontagem (T8)

- **6.1 Desmontagem Completa no Host B:**
  - No Host B, após receber o sinal, aplicar o processo inverso do Ami Pseudoternário para obter a sequência binária.
  - Converter a sequência binária de volta para texto ASCII estendido.
  - Aplicar o algoritmo de descriptografia para obter a mensagem original.
- **6.2 Exibição da Mensagem Original (T8):**
  - Exibir a mensagem original decodificada na interface gráfica do Host B.

Fase 7: Testes e Refinamentos

- **7.1 Teste com Sequência Binária da Aula:**
  - Realizar o teste exigido com a sequência binária fornecida na aula (sem criptografia).
- **7.2 Testes de Ponta a Ponta:**
  - Testar o fluxo completo: digitação, criptografia, codificação Ami, envio, recepção, decodificação Ami, descriptografia, exibição.

    - Testar com diferentes mensagens, incluindo caracteres especiais e acentuados.
    - Testar a comunicação entre múltiplos computadores.
- **7.3 Validação da Interface e Gráficos:**
    - Verificar se todos os campos da GUI são exibidos corretamente.
    - Assegurar que os gráficos da forma de onda são claros e precisos, mostrando ambos os processos (codificação e decodificação).
- **7.4 Documentação:**
    - Preparar a explicação do algoritmo de criptografia utilizado.
    - Documentar o código e o processo de instalação/execução.

Recomendações Técnicas Adicionais:

- **Modularização:** Dividir o código em módulos (funções/classes) para cada etapa (conversão binária, codificação Ami, decodificação Ami, criptografia, rede, GUI). Isso facilita o desenvolvimento, teste e manutenção.
- **Tratamento de Erros:** Implementar tratamento de erros para comunicação de rede (conexões perdidas, dados corrompidos) e para entradas inválidas.
- **Persistência de Configurações:** Se aplicável, permitir que o usuário salve configurações de rede (endereço IP, porta).
- **Feedback Visual:** Fornecer feedback visual ao usuário sobre o status da transmissão (enviando, recebendo, erro).

Este roteiro serve como um guia. As etapas podem ser iteradas e refinadas conforme o desenvolvimento avança.