Miniprojeto de IAPS

Desenvolvimento de um Sistema de Comunicação Full-Duplex através de sinais sonoros de frequências audíveis

Por: Paulo Fernandes nº 108678 [Turma P6]

Introdução

O tema do projeto desenvolvido assenta sobre o ponto nº 3 das propostas apresentadas, tendo como objetivo principal o desenvolvimento de um sistema de comunicação full-duplex capaz de realizar a transferência de tramas de bits entre computadores através das suas placas de som e a posterior utilização desse sistema para a implementação de uma aplicação de mensagens.

Funcionamento do CODEC

Este sistema funciona de forma semelhante ao **DTMF** (Dual-Tone Multi-Frequency), inicialmente desenvolvido para a discagem de números telefónicos, e ainda muito usado em sistemas de resposta interativa (**IVR**). Diferentemente do **DTMF**, que como o nome indica, usa combinações de 2 tons de frequências definidas (cada uma de entre 4 possíveis) para o envio da informação, o "protocolo" desenvolvido usa 4 tons, o que permite a codificação de **256 símbolos** em vez de 16. Este novo sistema permite também a comunicação em **2 canais** separados e em simultâneo (que usam 2 gamas de frequências distintas). Foram também feitos testes relativos à utilização da **fase** de cada sinal para a codificação de mais símbolos, que acabou por não ser utilizada pela elevada inconsistência dos resultados obtidos.

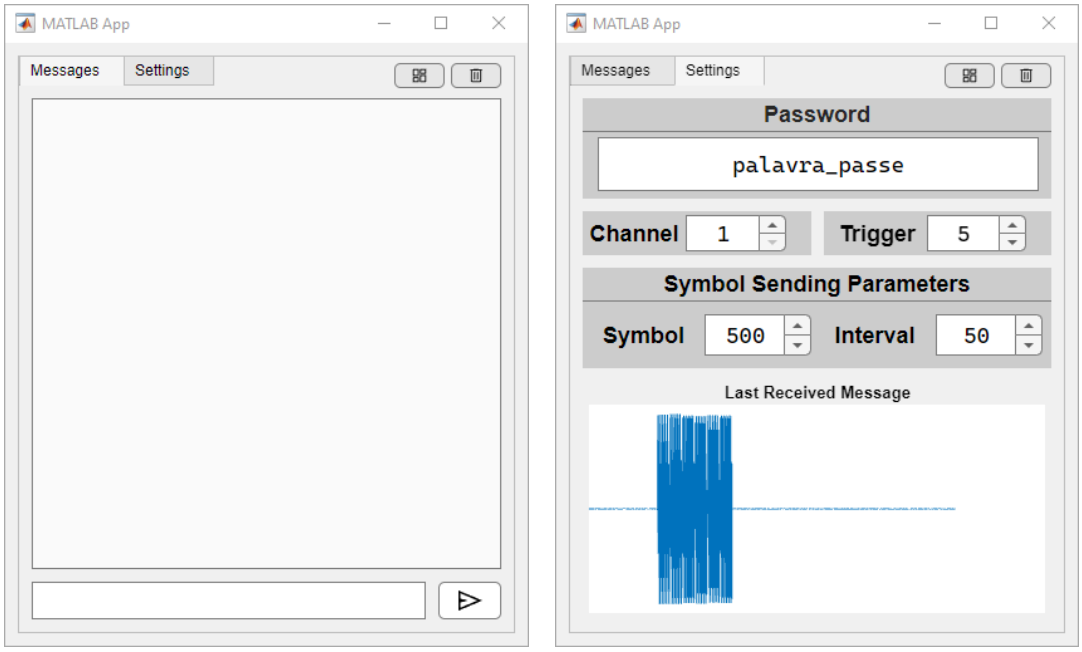
As frequências usadas para a transmissão da informação são múltiplas de 240 Hz pertencem aos intervalos [240, 3600] e [8160,11520] para **CH1** e **CH2**, respetivamente, e foram escolhidas de modo a otimizar transmissão e aumentar a sua velocidade. Enumeram-se algumas das razões pela escolha do número 240:

1. Para que todas as frequências possam ser corretamente identificadas na análise do sinal pela Transformada de Fourier é necessário que a resolução em frequência seja adequada ao seu espaçamento e quanto maior for a velocidade de transmissão, menor será esta resolução disponível, pelo que um espaçamento elevado entre frequências é ideal;
2. Sabendo que a frequência de amostragem usada foi de 24000 Hz, o uso do 240 permite que as frequências usadas apareçam descritas exatamente para todas as Transformadas de Fourier que usem múltiplos de 100 amostras, o que facilita a sua identificação.

Com os 256 símbolos possíveis de gerar é possível o envio de 8 bits de informação por cada símbolo enviado, podendo este sistema ser usado para o envio de qualquer tipo de dados, desde que codificados digitalmente. Para o envio de mensagens através deste sistema é necessário fazer a transferência de caracteres entre dispositivos de forma digital, tendo para isso sido usado o sistema de codificação Extended ASCII, que usa 8 bits para codificar os caracteres, fazendo com que seja possível o envio de 1 caracter por símbolo.

Interface Gráfica

Para um funcionamento mais simplificado ao utilizador decidi desenvolver uma interface gráfica simples com que permite configurar algumas funcionalidades:



Legenda: *Interface do programa desenvolvido para o projeto:* ***menu tab*** *e* ***settings tab****, respetivamente*

Como é possível ver pela imagem acima a aplicação possui uma aba principal, onde se escrevem e recebem as mensagens, bem como uma aba de definições onde podem ser configurados vários parâmetros que alteram a forma como o sinal é enviado/recebido:

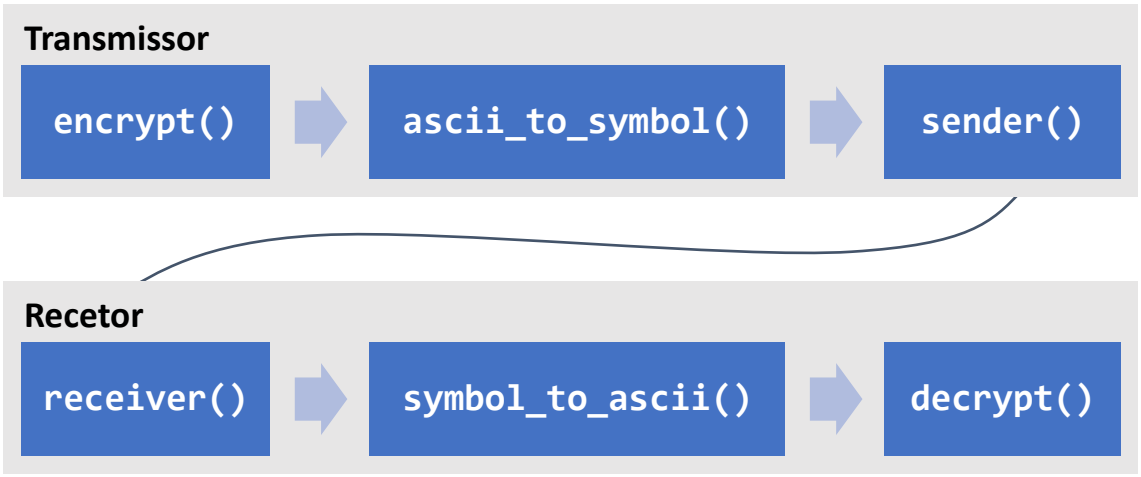
- **Password:** permite usar uma chave partilhada para encriptar o sinal enviado;
- **Channel:** seleciona o canal (1 ou 2) em que a comunicação está a ser feita;
- **Trigger:** determina a sensibilidade do sistema para a receção de mensagens, que embora seja determinada automaticamente no início do programa pode ser ajustada caso as mensagens não estejam a ser recebidas, ou estejam a ser identificadas transmissões falsas por conta do ruído do canal de comunicação;
- **Symbol:** especifica o número de samples enviados por cada símbolo;
- **Interval:** especifica o número de samples de intervalo enviado entre cada símbolo.

Além das configurações disponíveis é também possível visualizar o gráfico da última mensagem recebida, que permite avaliar o efeito das configurações acima definidas.

Quando uma mensagem é enviada, o seu conteúdo é mostrado na caixa de mensagens com o símbolo `[→]` e as mensagens recebidas aparecem com o símbolo `[←]`. Existem também 2 botões na parte superior de ambas as abas que permitem abrir o menu de sons do computador e apagar todas as mensagens recebidas e enviadas, respetivamente.

Código Desenvolvido e Seu Funcionamento

O programa desenvolvido está organizado vários blocos, que são executados em cadeia, como mostra a figura abaixo. Estes módulos têm efeitos simétricos entre os transmissor e o recetor por ocorrer a codificação de dados no envio e a sua posterior descodificação quando são recebidos. Todo este processo é coordenado pelo script principal: **msg.m**



Script de controlo: msg.m

Quando o programa é iniciado é chamada a função **selectMic()** que permite escolher o método de entrada de áudio. De seguida a interface gráfica é aberta, é iniciada a gravação e amostrado 1 segundo de áudio para calcular a sua energia e assim determinar o valor de trigger para a posterior distinção entre ruído e a receção de mensagens. A partir deste momento a função deste script é analisar o buffer de áudio para perceber se está a ser recebida alguma mensagem no canal selecionado na aplicação. Este processo de seleção é feito através do cálculo da energia dos últimos 0.1 segundos de áudio guardados no buffer, depois de terem sido filtrados para apenas serem relevantes as frequências respetivas de cada canal:

```
[num_ch1, den_ch1] = butter(3,0.5,"low");  % LowPass Butter Filter for CH1
[num_ch2, den_ch2] = butter(3,0.5,"high"); % HighPass Butter Filter for CH2
```

```
if(app.channel.Value == 1)
        audio_chunk = filter(num_ch1,den_ch1,audio_chunk);
    else
        audio_chunk = filter(num_ch2,den_ch2,audio_chunk);
    end
```

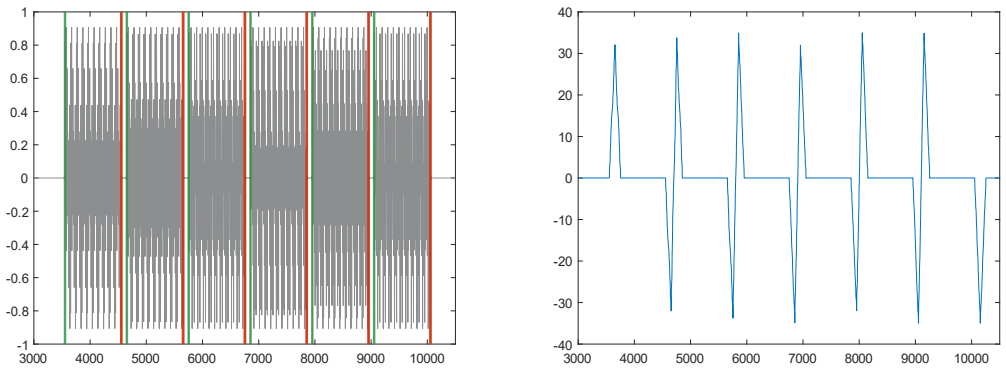
Legenda: *Definição dos filtros para ambos os canais e Filtragem do sinal de acordo com o canal selecionado*

Os filtros definidos para os canais **CH1** e **CH2** são respetivamente **passa-baixo** e **passa-alto** com frequências de corte de 50% (em relação a metade da frequência de amostragem) uma vez que nestes são usadas, respetivamente, a primeira e segunda metade do espetro de frequências disponíveis para a transmissão de símbolos.

Quando a energia calculada ultrapassa o valor de trigger é guardada a posição de início com uma pequena tolerância e o programa aguarda que a trama de símbolos seja enviada. O fim da transmissão é identificado pelo terceiro intervalo cuja energia calculada é inferior ao valor de trigger. Depois de ter detetado o fim da transmissão, este programa envia o vetor de amostras correspondente para a função **receiver()**, que dá início à descodificação a mensagem. Por fim, é reiniciada a gravação e é retomada a análise do áudio para a deteção de novas mensagens.

Receção de Símbolos: função receiver()

É nesta função que é feito o processamento do sinal recebido. Para poder descodificar todos os símbolos recebidos em cada mensagem é preciso primeiro isolá-los da mensagem completa. Para isso aplica-se primeiro no **módulo do sinal de áudio** um **filtro de média** (passa-baixo) muito longo, que permite obter um "sinal analógico" que embora com algum ruído permite distinguir os diferentes símbolos. Para que deste vetor de dados possam ser retiradas informações das posições de cada dígito é necessário calcular a sua derivada, que permite identificar o início de um dígito com um pico positivo e o seu fim com um pico negativo, como se pode ver pela figura abaixo:



Legenda: *Gráficos e audio, e de an_detect_diff obtidos na receção da mensagem "108678" por receiver()*

Depois de identificados os picos e corrigidos os offsets causados pelos filtros causais aplicados, obtêm-se as posições de início e de fim de cada símbolo, como mostrado na figura acima com as cores verde e vermelho, respetivamente. Com esta informação é agora possível isolar cada um dos símbolos e processá-los individualmente.

O processo de descodificação passa primeiro por verificar se o intervalo de cada dígito tem a duração mínima para a qual a identificação é válida (foi escolhido um mínimo de 200 amostras que correspondem a uma resolução em frequência de 120 Hz), sendo descartados os intervalos que não obedecem à condição. Depois disto é calculada a **FFT** do respetivo intervalo e obtidas as 4 frequências correspondentes às posições de cada intervalo cuja amplitude é máxima. Por fim o valor de cada frequência estimada é comparado com as frequências possíveis para cada uma das 4 zonas, e escolhida a posição da mais próxima. Cada símbolo é então identificado por um vetor das 4 posições das frequências detetadas. No final da descodificação a função devolve uma matriz de 4 linhas com todos os símbolos identificados. No caso do **CH2**, depois de as frequências serem estimadas é lhe aplicado um offset de -7920 Hz para que a construção do vetor de símbolos possa ser baseada na matriz de frequências do **CH1**.

Conversões: funções asccii_to_symbol() e symbol_to_ascii()

Estas funções implementam a conversão entre uma **string** (mensagem) e a matriz dos respetivos **símbolos**, e servem para codificar/descodificar os dados enviados/recebidos pelas funções sender() e receiver(). A geração dos símbolos consiste na conversão do código ASCII de cada dígito (8 bits) para a base 4, separação dos dígitos e o seu incremento em 1 unidade, sendo o processo de restauro da string, o inverso deste. Na tabela abaixo mostram-se 2 exemplos:

Character	Extended ASCII code		Symbol
'P'	50_{16}	1100_4	[2 2 1 1]
'F'	46_{16}	1012_4	[2 1 2 3]

Encriptação: Funções encrypt() e decrypt()

Nestas funções é feita a encriptação/desencriptação da mensagem trocada evitando a sua receção por dispositivos alheios à comunicação. O mecanismo implementado usa **criptografia simétrica**, que usa uma **chave partilhada** de tamanho variável pelo recetor e emissor para ambos os processos de encriptação e desencriptação. Esta chave é repetida até ter o mesmo tamanho da mensagem a enviar e os dígitos sofrem um offset positivo ou negativo baseado na chave, dependendo da operação que se está a realizar.

Envio de símbolos: função sender()

Esta função é chamada pela aplicação da interface gráfica quando o utilizador clica no botão de enviar e recebe como argumento de entrada a **matriz de símbolos** a serem enviados, o **canal** de envio, e os tamanhos de cada símbolo e da duração dos intervalos em número de samples. Com estes parâmetros é gerado um vetor vazio com a dimensão final do som a enviar e são preenchidas as zonas correspondentes a cada símbolo. Como as somas de diferentes frequências geram sons com amplitudes diferentes é necessário normalizá-las para que o som seja mais facilmente descodificado no recetor. Depois de o áudio estar completamente construído, é enviado para a placa de som do computador.

Resultados

Em testes realizados em ambiente virtual (cabo virtual que conecta uma saída de áudio a uma entrada permitindo a interação entre as 2 componentes do sistema através de som) o sistema funciona sem falhas para velocidades de transmissão de até cerca de 45 Bytes/s, funcionando quase sem perdas em até 50 Bytes/s, individualmente em cada canal.

Foi também testado o funcionamento em sistemas reais (conectando 2 computadores através de um cabo de áudio) que embora tenha tido algum sucesso apenas funciona quase sem perdas para velocidades de transmissão menores que 30 Bytes/s. Os piores resultados obtidos nesta situação explicam-se pela introdução de vários fatores que dificultam a comunicação, como:

- Introdução intermédia das placas de som dos computadores leva a diversos problemas:
  - Sistema afetado com ruído gaussiano de elevada amplitude
  - Não linearidade das respostas em frequência que afeta a intensidade de algumas frequências enviadas/recebidas;
  - Inconsistência na cadência de envio das amostras para a saída de áudio causando distorções da frequência e paragens bruscas na transmissão
  - Impossibilidade do controlo do ganho dos microfones que leva a distorções da amplitude do sinal
- Interferências de fatores exteriores no cabo de comunicação;

Os problemas encontrados na utilização do sistema em condições reais poderiam ter sido amenizados se tivesse sido implementado um controlo eletrónico de regulação de potência do sinal transmitido entre os computadores.

Conclusão

Com a realização deste projeto foi possível uma maior familiarização com os processos de geração e análise e processamento de sinais em geral usando o MATLAB, e mais concretamente um melhor entendimento sobre o funcionamento de sistemas de comunicação através de sons audíveis e desafios inerentes a esse modo de comunicação em sistemas reais.