**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**

Neste relatório é abordado o processo de desenvolvimento e teste do miniprojeto de Introdução à Análise e Processamento de Sinal.

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**

O modo de funcionamento do sistema desenvolvido 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 sons de frequências definidas (cada uma de entre 4 possíveis) para o envio da informação, o “protocolo” desenvolvido usa 4 sons, o que permite a codificação de 256 símbolos em vez de 16. Este novo sistema permite também a comunicação em 2 canais separados 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 informação, mas os resultados revelaram-se muito inconsistentes, não tendo esta sido utilizada por esse motivo.

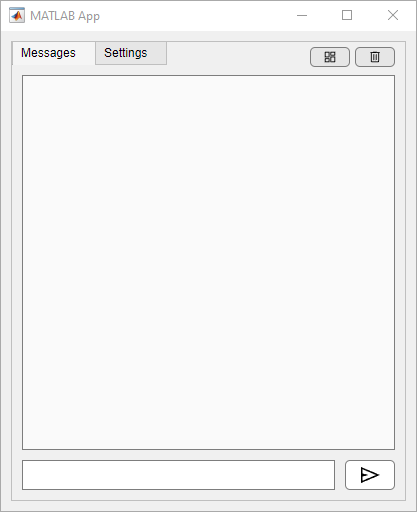
ESCOLHA DAS FREQUENCIAS PORQUE?

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](https://www.asciitable.com/), 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:

 Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

*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 código desenvolvido está organizado vários blocos, sendo alguns deles simétricos entre si por existir codificação de dados no emissor e descodificação do recetor, existindo um script que controla todo o funcionamento dos mesmos: msg.m.

**Transmissor**

**Recetor**

**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 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 é feito através da análise 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 |

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 fa/2) uma vez que nestes são usadas as 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 descodifica a mensagem.

**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, o que permite obter um “sinal analógico” que embora com algum ruído permite distinguir os diferentes símbolos. Depois de obter o “sinal analógico” é preciso torná-lo num sinal digital booleano e para isso…

**Funções asccii\_to\_symbol() e symbol\_to\_ascii()**

**Funções encrypt() e decrypt()**

**Função sender()**

**Resultados**

a velocidade máxima de transmissão sem falhas atingida em ambiente de teste (controlado) foi de cerca de 50 Bytes por segundo, em cada canal, totalizando 100 B/s de velocidade combinada.

PODERIA SER IMPLEMENTADO UM MODO QUE USA AMBOS OS CANAIS AO MESMO TEMPO

**Conclusão**

TABELAS TEM QUE TER MARGENS DE 0.1