UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Relatório do Projeto Final
Disciplina de Lógica Reconfigurável

Autores: Gabriel Henrique Linke e João Guilherme Martins Silva

Versão: 05-Julho-2023

1 Introdução

O objetivo deste projeto é criar um sistema com um filtro implementado em VHDL na placa Altera DE2, que se comunica via sockets com um servidor. Esse filtro receberá continuamente por socket um sinal ponto-a-ponto com uma onda e enviará de volta os pontos correspondentes do sinal filtrado. Enquanto isso, o servidor enviará o sinal de entrada lendo-o de um arquivo e depois mostrará em um gráfico o sinal recebido da placa. O sinal de entrada deve seguir o padrão de um arquivo wave de 16 bits, sendo amostrado a uma frequência de 44100Hz e com amplitude compreendida entre 32767 e -32768.

2 Hardware do usuário - Filtro FIR (VHDL)

O filtro em VHDL foi criado com base no filtro do projeto filtro_FIR, que está presente no OpenCore. Esse projeto implementa um filtro genérico, dessa forma, basta definir o tamanho do sinal em bits, a resolução utilizada para os cálculos, a ordem do filtro e os coeficientes do filtro para fazê-lo funcionar. Dessa maneira, para o nosso projeto, foi calculado no matlab um filtro FIR passa alta de ordem 32 e frequência de corte de 500Hz. Esse processo pode ser observado na Figura 1. Depois de criar o filtro no matlab, os coeficientes foram exportados e adicionados no VHDL (arquivo FIR_low_area.vhd), como pode ser visto na Figura 2.

```
% Parâmetros do filtro
ordem = 32;
frequenciaCorte = 500;
frequenciaAmostragem = 44.1e3; % Freq. de amostragem (altere conforme necess

% Projeta o filtro FIR passa-alta
filtro = fir1(ordem, frequenciaCorte / (frequenciaAmostragem / 2), 'high');

% Export filter coefficients to a text file
filename = 'filter_coefficients.txt';
dlmwrite(filename, filtro, 'precision', '%.10f', 'delimiter', '\n');
```

Figura 1: criação dos coeficientes do filtro no Matlab

```
data_length  : NATURAL     := 16;
data_signed  : BOOLEAN     := true;
improv_t     : BOOLEAN     := false;
bits_resol   : NATURAL     := 16;
taps         : NATURAL     := 33;
coefficients : COEFF_ARRAY := (-0.0014480712,
                               -0.0016547757,
                               -0.0021998367,
                               -0.0030872443,
                               -0.0043035648,
                               -0.0058180088,
                               -0.0075834401,
                               -0.0095382907,
                               -0.0116092940,
                               -0.0137148999,
                               -0.0157691965,
                               -0.0176861310,
                               -0.0193838022,
                               -0.0207885931,
                               -0.0218389165,
                               -0.0224883680,
                               0.9787196211,
                               -0.0224883680,
                               -0.0218389165,
                               -0.0207885931,
                               -0.0193838022,
                               -0.0176861310,
                               -0.0157691965,
```

Figura 2: Parâmetros do filtro no VHDL

3 C

O código em C se encarrega de fazer uma requisição DHCP para a rede para obter um IP temporário e em seguida tentar se conectar ao IP e porta especificados no começo do arquivo iniche_init.c.
Ao se conectar a um servidor ele começará a receber os pontos do servidor até um máximo de 5000 caracteres por pacote, passando cada ponto separado por um caracter especial \n para o filtro e enviando um pacote de resposta assim que um ponto for retornado. Para o envio dos pontos para o hardware são utilizadas as funções IOWR_16DIRECT e IORD_16DIRECT.

4 Servidor

O servidor no sistema hospedeiro é dividido em 2 partes: o socket é aberto por um programa em javascript e a interface gráfica é escrita em python, com a biblioteca PyQt.

- server.js:

Abre um socket na porta 5000 e espera um cliente se conectar. Ao receber uma conexão envia de forma assíncrona os pontos presentes no arquivo input.dat e fica escutando por uma resposta. Para cada pacote de dados recebido, os pontos são gravados em um arquivo out.dat e escritas na saída Standard Out.

- serverGui.py:

O controle do servidor é feito por um botão que inicia ou mata um processo executando o arquivo server.js. Os dados são obtidos lendo o standard out do processo gerado, que também é escrito na caixa de texto da interface. Devido ao encoding utilizado pela placa é necessário sanitizar a entrada retirando caracteres especiais que não são removidos pela decodificação e checando se a entrada pode ser convertida para um inteiro. Os pontos

válidos da entrada são agrupados em uma lista e um sinal é disparado avisando qualquer elemento conectado da mudança, com a lista de pontos detectados como parâmetro.

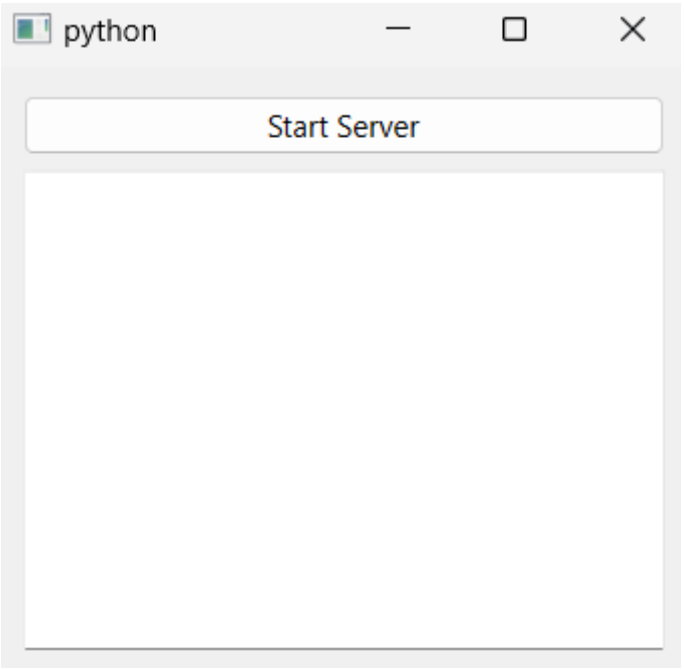

Figura 3 - Interface Gráfica do servidor

- realtime-plot.py:

Implementa uma interface gráfica para iniciar o servidor e para observar o sinal. Essa janela apresenta 2 gráficos, um para o sinal original(extraído do arquivo input.dat) e outro para ser alimentado em tempo real pelo sinal filtrado. A classe definida em serverGui.py é incorporada à janela e o gráfico é configurado para atualizar sempre que o sinal do servidor for emitido.

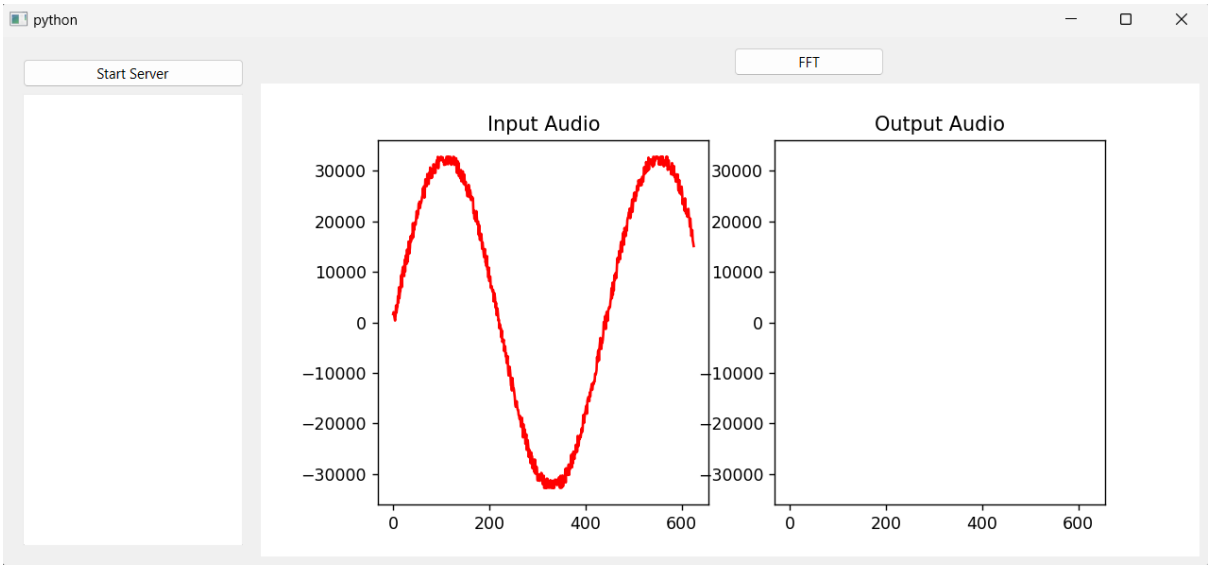

Figura 4 - Sinal Original no Tempo

A visualização pode ser alternada entre o domínio do tempo e o domínio da frequência, com a fft do sinal original já calculada e armazenada desde o começo da execução, enquanto a fft do sinal filtrado é calculada apenas no momento da troca.

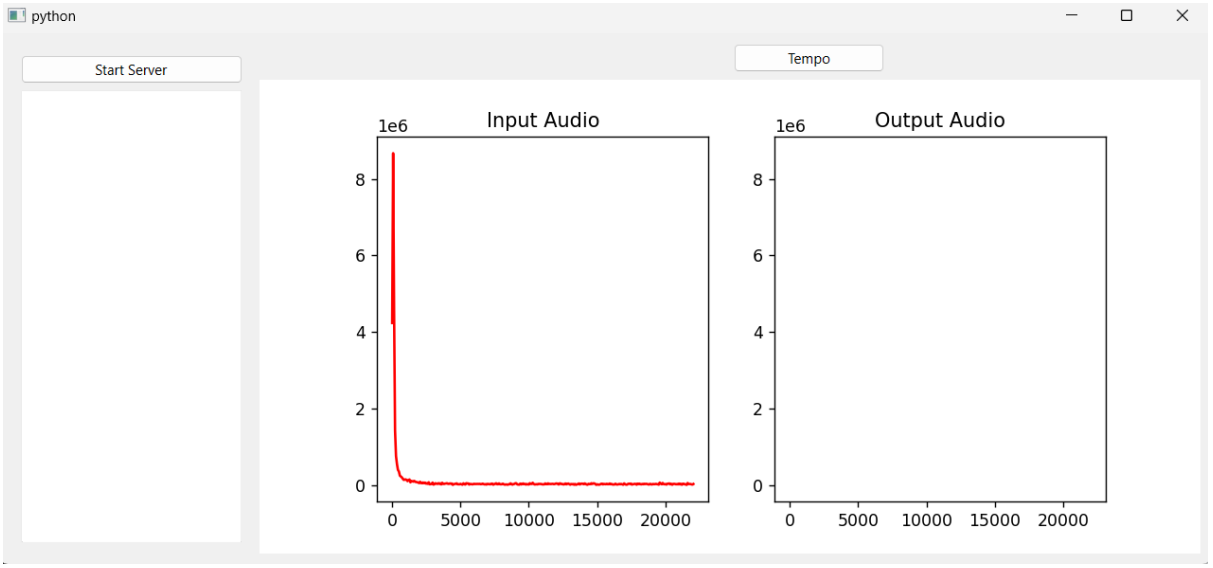

Figura 5 - Sinal Original na frequência

5 Teste do funcionamento

Para testar o funcionamento, é possível enviar sinais com frequências conhecidas e verificar no Matlab se a resposta é similar à esperada. Para começar, será enviado um sinal de 100Hz, que fica abaixo da frequência de corte de 500Hz. É possível verificar na Figura 6 que o sinal recebido da placa é atenuado, como esperado. Também é possível observar as raias em frequência na Figura 7, que mostra a diminuição da amplitude do sinal de 100Hz. Por fim, usando o mesmo filtro no Matlab para filtrar uma senoide de 100Hz, é possível observar na Figura 8 que a amplitude do sinal de saída é muito similar à do nosso sistema.

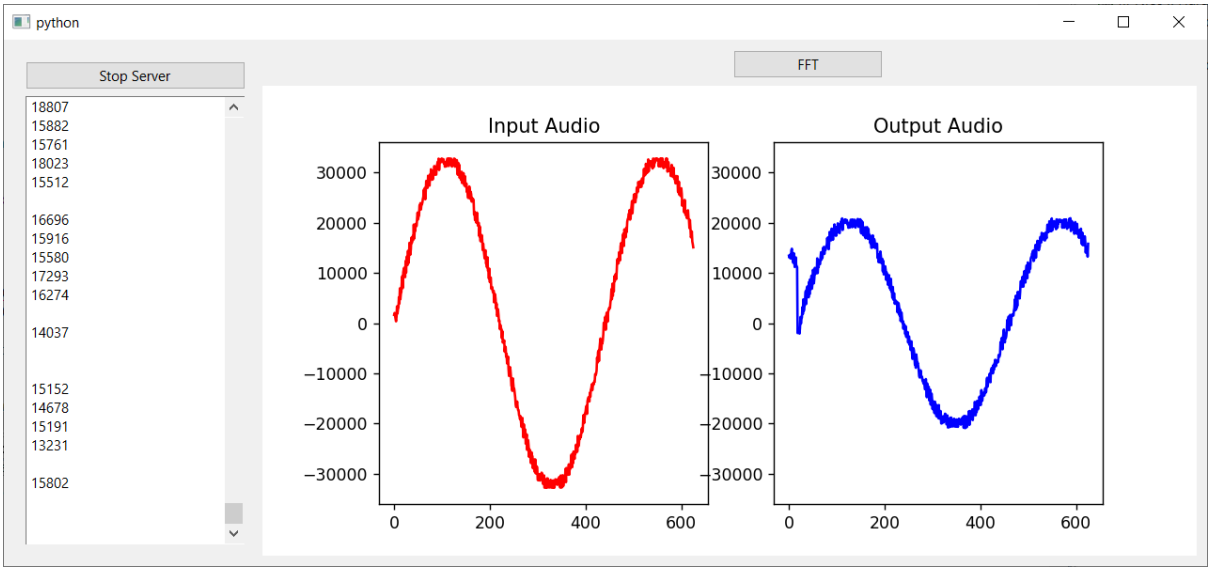

Figura 6 - Sinal de 100Hz filtrado

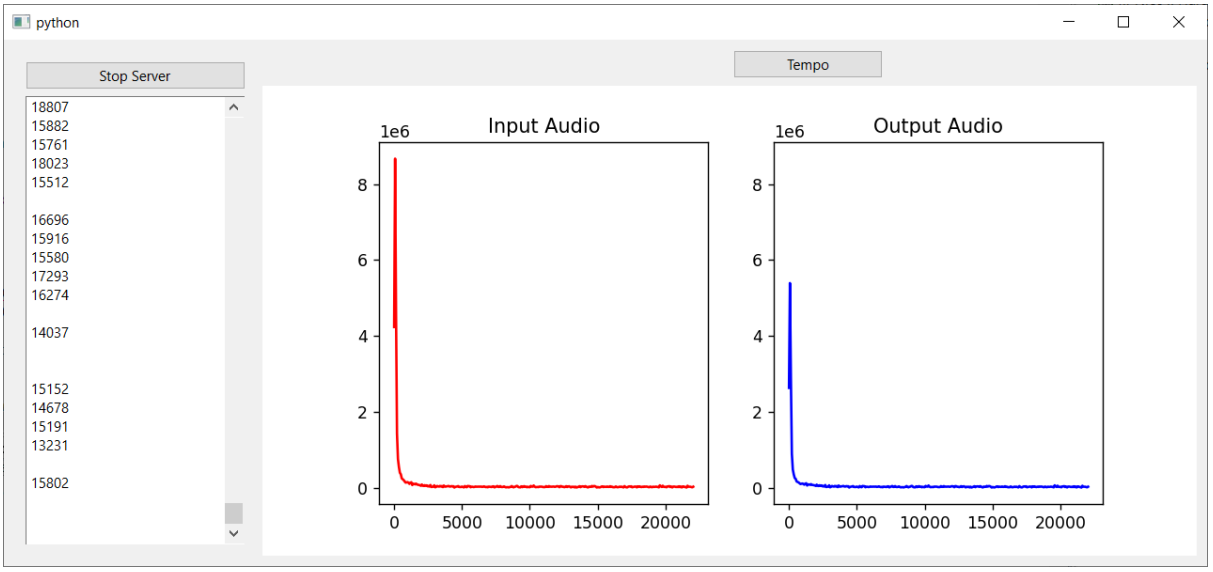

Figura 7 - FFT do sinal de entrada e de saída

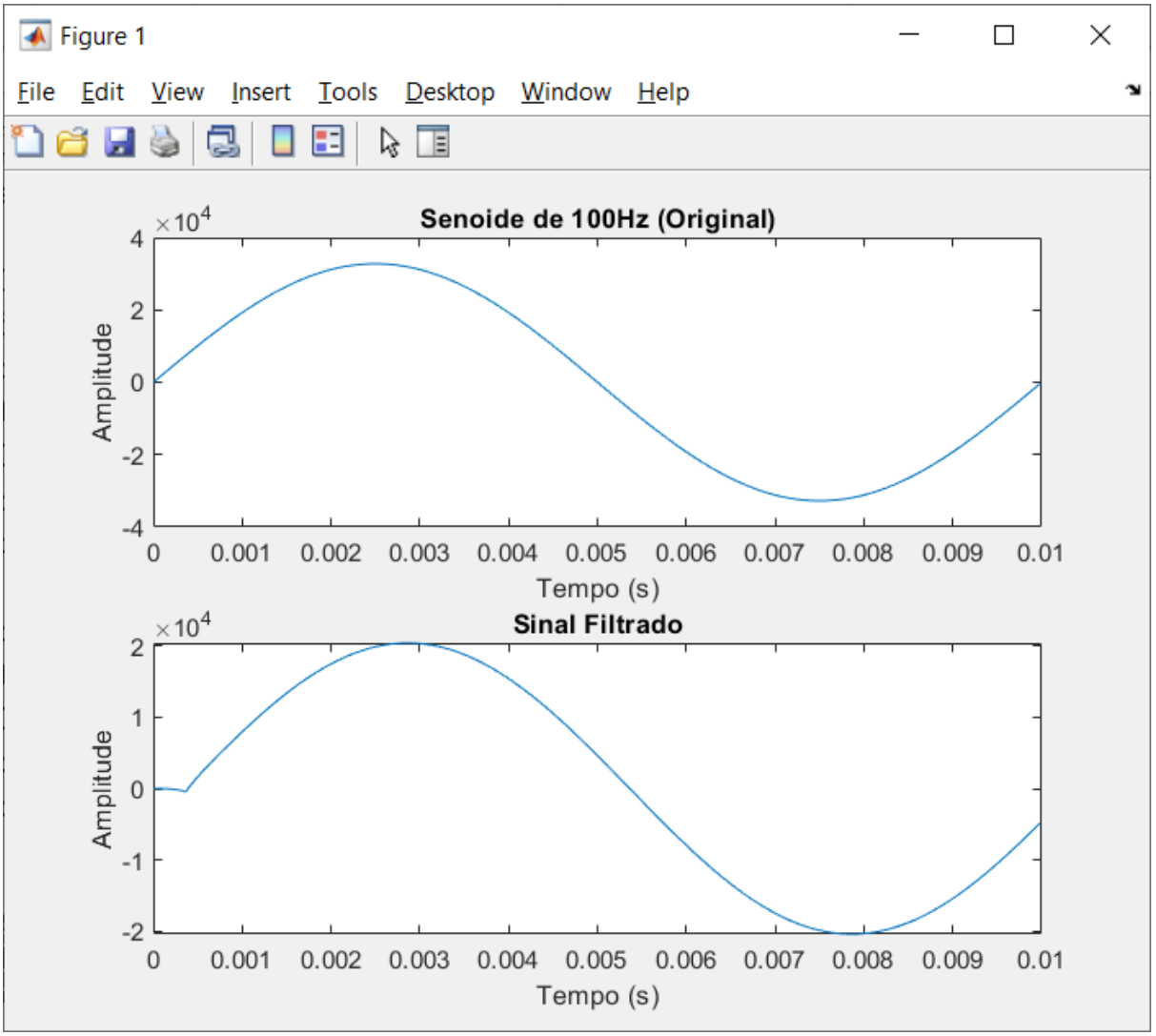

Figura 8: Sinal de 100Hz filtrado no Matlab

Como um segundo teste, será enviado um sinal de 2000Hz misturado com mais alguns sinais de frequências maiores, que ficam acima da frequência de corte de 500Hz. É possível verificar na Figura 9 que o sinal recebido da placa é apenas levemente atenuado, como esperado. Também é possível observar as raias em frequência na Figura 10, que mostra que o sinal segue muito similar. Por fim, usando o mesmo filtro no Matlab para filtrar uma senoide de 2000Hz, é possível observar na Figura 11 que a amplitude do sinal de saída é muito similar à do nosso sistema.

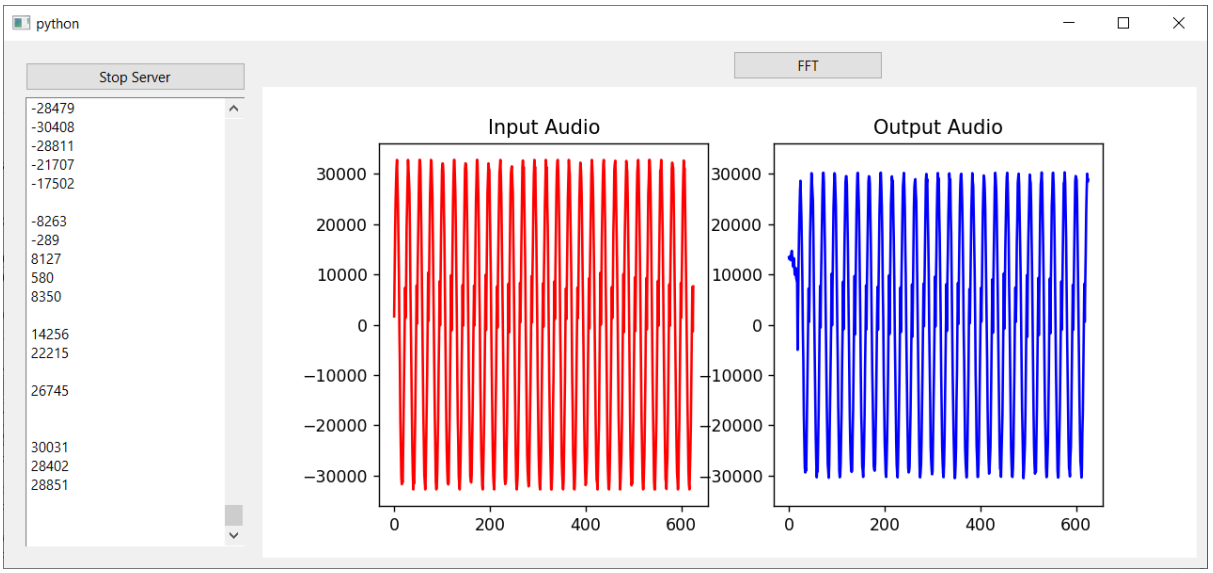

Figura 9 - Sinal de 2000Hz filtrado

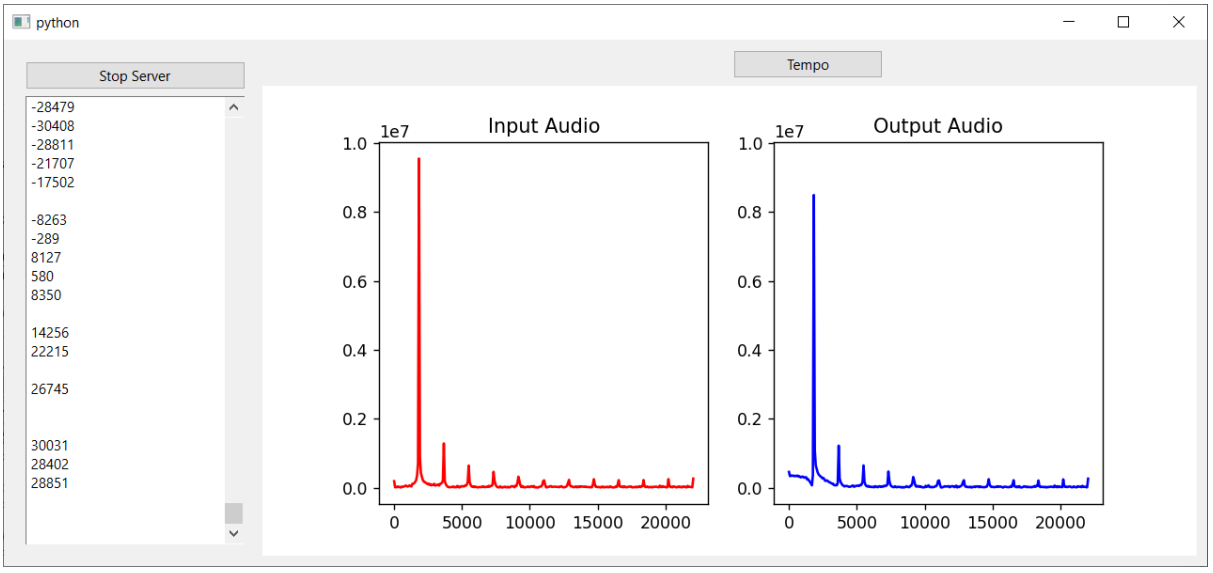

Figura 10 - FFT do sinal de entrada e de saída

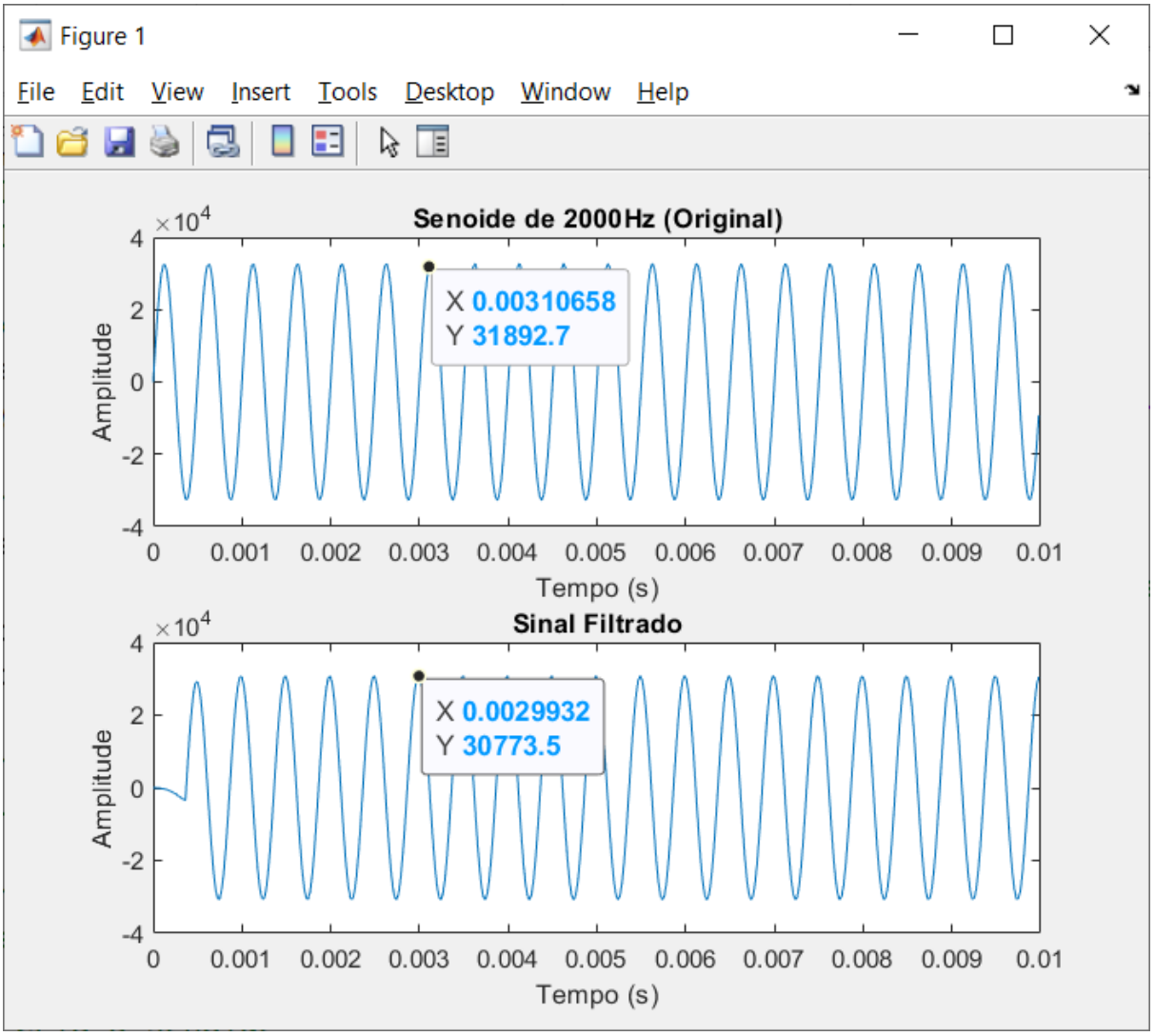

Figura 11: Sinal de 2000Hz filtrado no Matlab