



TRABALHO DE CONCLUSÃO DE CURSO

**Tratamento de Sinal de Áudio  
com aplicações em Música Digital**  
*Reverb Shimmer*

**Filipe Miguel Ribeiro**

**Brasília, julho de 2017**

**UNIVERSIDADE DE BRASÍLIA**



UNIVERSIDADE DE BRASÍLIA  
Faculdade de Tecnologia

TRABALHO DE CONCLUSÃO DE CURSO

**Tratamento de Sinal de Áudio  
com aplicações em Música Digital  
*Reverb Shimmer***

**Filipe Miguel Ribeiro**

*Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia  
Elétrica como requisito parcial para obtenção  
do grau de Engenheiro Eletricista*

Banca Examinadora

Prof. Dr. André Café, ENE/UnB  
*Orientador*

---

Prof. Fulano de Tal 2, ENE/UnB  
*Examinador Interno*

---

Prof. Fulano de Tal 2, ENE/UnB  
*Examinador interno*

---

Prof. Fulano de Tal 2, ENE/UnB  
*Examinador interno*

---

## FICHA CATALOGRÁFICA

RIBEIRO, FILIPE MIGUEL

Tratamento de Sinal de Áudiocom aplicações em Música Digital *Reverb Shimmer* [Distrito Federal] 2017.

xvi, 59 p., 210 x 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Elétrica, 2017).

Trabalho de Conclusão de Curso - Universidade de Brasília, Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

1. Efeito Reverb Shimmer

2. Microntrolador MSP430

3. Misturas Convolutivas

4. Separação de sinais de fala

I. ENE/FT/UnB

II. Título (série)

## REFERÊNCIA BIBLIOGRÁFICA

RIBEIRO, F. M. (2017). *Tratamento de Sinal de Áudiocom aplicações em Música Digital Reverb Shimmer*. Trabalho de Conclusão de Curso, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 59 p.

## CESSÃO DE DIREITOS

AUTOR: Filipe Miguel Ribeiro

TÍTULO: Tratamento de Sinal de Áudiocom aplicações em Música Digital *Reverb Shimmer*.

GRAU: Engenheiro Eletricista ANO: 2017

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Conclusão de Curso e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Os autores reservam outros direitos de publicação e nenhuma parte deste Trabalho de Conclusão de Curso pode ser reproduzida sem autorização por escrito dos autores.

---

Filipe Miguel Ribeiro

Depto. de Engenharia Elétrica (ENE) - FT

Universidade de Brasília (UnB)

Campus Darcy Ribeiro

CEP 70919-970 - Brasília - DF - Brasil

## **Dedicatória**

*Dedicatória do autor 1*

*Filipe Miguel Ribeiro*

## **Agradecimentos**

*A inclusão desta seção de agradecimentos é opcional e fica à critério do(s) autor(es), que caso deseje(em) inclui-la deverá(ao) utilizar este espaço, seguindo esta formatação.*

*Filipe Miguel Ribeiro*

---

## **RESUMO**

INSIRA SEU RESUMO AQUI.

---

## **ABSTRACT**

INSERT YOUR ABSTRACT HERE.



# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO .....</b>	<b>1</b>
1.1	CONTEXTUALIZAÇÃO E DEFINIÇÃO DO PROBLEMA .....	1
1.1.1	EFEITO DE AMBIENTIZAÇÃO: <i>Reverb</i> .....	1
1.1.2	TIPOS DE REVERB .....	2
1.1.3	PITCH SHIFTING .....	3
1.2	COMPARATIVO ENTRE SOLUÇÕES DE HARDWARE .....	4
1.2.1	HARDWARES DE REVERB-SHIMMER .....	5
1.2.2	MICROCONTROLADORES E DSP'S .....	6
1.2.3	HARDWARES COMERCIAIS DO EFEITO <i>Reverb - Shimmer</i> .....	7
1.2.4	MICROCONTROLADOR MSP430F5529 E TMS320 .....	7
1.3	OBJETIVOS E DIAGRAMA DE BLOCO DO PROJETO .....	7
<b>2</b>	<b>METODOLOGIA E FERRAMENTAS .....</b>	<b>9</b>
2.1	SINAIS DE TEMPO DISCRETO E TRANSFORMADA DISCRETA DE <i>Fourier</i> ..	9
2.1.1	PROBLEMA DA AMOSTRAGEM DE SINAL CONTÍNUO .....	9
2.1.2	EFEITO DO JANELAMENTO .....	10
2.2	FILTROS DIGITAIS .....	12
2.2.1	CONCEITOS INICIAIS .....	12
2.2.2	FILTROS IIR .....	12
2.2.3	FILTROS FIR .....	13
2.2.4	FILTROS ADAPTATIVOS .....	17
2.2.5	FILTRO PENTE - <i>Comb Filter</i> .....	18
2.3	TITLE .....	21
2.4	CONVERSÃO ANALÓGICA DIGITAL .....	21
2.4.1	CONSIDERAÇÕES INICIAIS .....	21
2.4.2	CONVERSÃO POR APROXIMAÇÕES SUCESSIVAS .....	21
2.4.3	DIFICULDADES PRÁTICAS DE CONVERSORES ADS .....	22
2.5	CONVERSÃO DIGITAL ANALÓGICA .....	22
2.6	COMUNICAÇÃO SERIAL I2C .....	22
2.6.1	FORMATO DE DADOS .....	23
2.7	FERRAMENTAS COMPUTACIONAIS .....	23
2.7.1	<i>Code Compose Studio</i> .....	23
2.7.2	<i>WinFilter</i> .....	23
<b>3</b>	<b>IMPLEMENTAÇÃO DO PROJETO .....</b>	<b>27</b>
3.1	BLOCO 0 - CONVERSÃO A/D NO MSP430 .....	27
3.1.1	ANÁLISE DE <i>Hardware</i> .....	27

3.1.2	REQUISITOS DE HARDWARE PARA A TEMPO DE AMOSTRAGEM.....	28
3.2	BLOCO 1 - PROJETANDO UM FILTRO FIR .....	30
3.2.1	PARA O CÓDIGO EXPERIMENTAL DO MATLAB .....	30
3.2.2	PARA O PROJETO DO $\mu C$ .....	30
3.3	BLOCO 2 - IMPLEMENTANDO O <i>Pitch-Shifter</i> .....	32
3.3.1	ANÁLISE DO ALGORITMO .....	32
3.4	BLOCO 3 - DELAY TIME - REVERB EM CONVOLUÇÃO.....	38
3.5	BLOCO 5 - CONVERSÃO D/A - COMUNICAÇÃO I <sup>2</sup> C E MCP4725.....	39
3.5.1	LIMITAÇÕES DO MSP430F5529LP .....	39
3.5.2	CARACTERÍSTICAS DO <i>Hardware</i> E OPERAÇÃO .....	39
<b>4</b>	<b>SIMULAÇÕES E RESULTADOS .....</b>	<b>40</b>
4.1	DESCRIÇÃO DOS EXPERIMENTOS.....	40
4.2	ANÁLISE DOS RESULTADOS .....	42
4.2.1	DO DESEMPENHO DO CÓDIGO MATLAB.....	42
<b>5</b>	<b>CONCLUSÕES .....</b>	<b>47</b>
5.1	REALIZAÇÃO DO PROJETO.....	47
5.2	AUTO-CRÍTICA .....	47
5.3	MODELOS PARA TRABALHOS FUTUROS .....	48
	<b>REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>49</b>
	<b>APÊNDICES.....</b>	<b>50</b>
I.1	CÓDIGOS EM MATLAB RELACIONADOS AO MÉTODO PROPOSTO.....	51

# LISTA DE FIGURAS

1.1	Ilustração do efeito de Reverberação. ....	1
1.2	Resposta de um Impulso de uma sala pequena. ....	2
1.3	Câmara de reverb de mola de um amplificador ....	2
1.4	Pequeno protótipo de um reverb de placas. ....	3
1.5	Foto modelo Eventide H9 - Effects Processor. ....	6
1.6	Diagrama de Blocos Principal do Projeto.....	8
2.1	Diagrama de fluxo de sinais e diagrama de blocos de um filtro FIR .....	14
2.2	Resposta em frequência do FPB-FIR do exemplo 2.2.1 para $N=51$ . ....	16
2.3	Ação da filtragem do FPB-FIR do exemplo 2.2.1 .....	17
2.4	Resposta em frequência dos modos de Janelamento. ....	17
2.5	Filtro Pente - <i>Comb Filter</i> - Formas de Realimentação .....	19
2.6	Conversor A/D de aproximações sucessivas. ....	21
2.7	Aparência do Programa WinFilter - versão 0.8 de 06-08-2004 .....	24
2.8	Valor lido no gráfico por meio do cursor do mouse. $4ms - 0.854$ .....	25
2.9	Menu Output - programa WinFilter .....	26
3.1	Algumas taxas de amostragem utilizadas na prática .....	29
3.2	Uma pequena ilustração como funciona um <i>ring-buffer</i> .....	31
3.3	Exemplo de arbodagem do <i>pitch-shifter</i> - premissa básica para o efeito <i>shimmer</i> ....	33
3.4	Expansão e compressão do sinal <i>frame</i> por <i>frame</i> . ....	33
3.5	Sinal tornando-se descontínuo quando submetido ao processo de expansão ou compressão no tempo.....	34
3.6	Etapas do algoritmo do <i>phase-vocoder</i> .....	34
3.7	Janelamentos .....	35
3.8	Ondas senoidais com pequena diferença na frequência e fase.....	36
3.9	Reamostragem do Sinal utilizando duas vezes o valor da amostra anterior .....	38
4.1	Quantização do sinal de áudio em 12, 8 e 4 bits PCM .....	41
4.2	Quantização do sinal de áudio em 10, 12 e 16 bits PCM .....	41
4.3	Resposta do filtro FIR-Passa-Baixa com janela <i>Blackman</i> .....	42
4.4	Respostas Impulsionais de Ambiente: Exemplos retirado do <i>website</i> : OpenAir.....	43
4.5	Espectro de frequência do sinal original 12 bits e o sinal filtrado pelo filtro casual FIR. ....	44
4.6	Análise espectral de um sinal puro de guitarra tocada na nota Lá - $A2 = 220Hz$ e o efeito pitch-shifter utilizando deslocamento de 1 oitava. ....	45
4.7	Análise espectral de um áudio de guitarra tocando um acorde dedilhado. Pitch-shifter utilizando deslocamento de 1 oitava. ....	46

# LISTA DE TABELAS

2.1	Função janela ( <i>window</i> ) associadas com filtros FIR .....	14
4.1	Dados relativos as simulações no computador .....	40

# LISTA DE SÍMBOLOS

## Símbolos Latinos

$Q$	Fluxo	[ml/s]
-----	-------	--------

## Símbolos Gregos

$\Delta$	Varição entre duas grandezas similares
$\varepsilon$	Fração muito pequena de uma certa grandeza

## Grupos Adimensionais

$e$	Número de Euler
-----	-----------------

## Subscritos

$max$	Máximo
-------	--------

## Sobrescritos

$—$	Valor médio
-----	-------------

## Siglas

DSP	Processador Digital de Sinais - <i>Digital Signal Processing</i>
LoG	<i>Laplacian of Gaussian</i> (laplaciano da gaussiana)
TE	Tempo de eco
TR	Tempo de repetição
Ts	Período de Amostragem

# 1 INTRODUÇÃO

## 1.1 CONTEXTUALIZAÇÃO E DEFINIÇÃO DO PROBLEMA

### 1.1.1 Efeito de ambientização: *Reverb*

Numa sala, ou em qualquer ambiente acústico, existe um caminho direto pelo qual uma fonte de áudio qualquer pode ser ouvida, no entanto, as respectivas ondas sonoras também podem fluir em caminhos mais longos devido a, por exemplo, reflexão das paredes, do teto, de objetos, antes delas chegarem ao receptor.

A energia envolvida com essas reflexões, as quais viajam essas distâncias maiores do que o som emitido no caminho direto, são parcialmente absorvidas pelas superfícies, logo elas chegam ao receptor com um som mais "fraco" que o som direto.

Essas amostras de som atrasadas e atenuadas ocorridas no evento da emissão do som original é o que denominamos de **reverberação**.

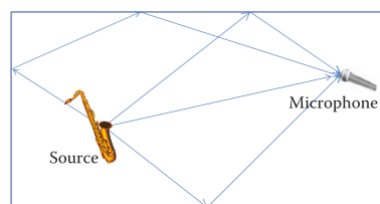


Figura 1.1: Ilustração do efeito de Reverberação.

Muito embora possa parecer ao primeiro momento, o efeito de reverberação é muito mais que uma série de ecos. Um eco, por exemplo, pode ser entendido um resultado de uma distinção, atraso de versões de um som, o qual você pode ouvir com um atraso no mínimo de 40 milissegundos. Já a reverberação de uma sala vazia, por exemplo, há muitas e muitas reflexões, e as primeiras delas que chegam ao receptor são muito mais curtas que termos de tempo de duração. Logo essas reflexões não são tão percebidas ou distinguidas do som da fonte diretamente. Em vez disso, nós percebemos apenas o efeito da combinação de todas essas reflexões.

Nesse sentido, podemos também considerar que a reverberação do som é mais do que um simples dispositivo de *delay* com retorno. No *reverb* a taxa em que as reflexões chegarão muda ao longo do tempo, em oposição de termos apenas que simular reflexões que tenham um intervalo fixo entre elas. Essas reflexões são relacionadas a posição do som que o receptor está na sala, ao tipo de construção da sala (oval, retangular), tamanho e material das paredes.

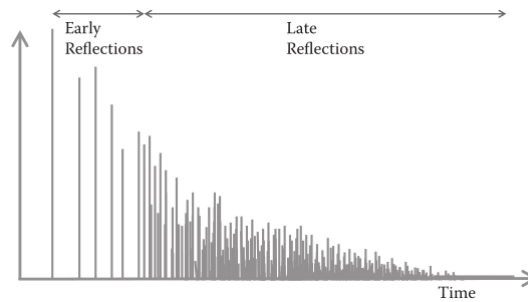


Figura 1.2: *Resposta de um Impulso de uma sala pequena.*

### 1.1.2 Tipos de Reverb

Efeitos de *reverb* são alcançados pela utilização de uma série complexa de *delays* de um mesmo sinal que diminuem em amplitude e clareza de modo a simular o comportamento acústico de um espaço real. Falando sob o aspecto musical e industrial de equipamentos que realizam o processamento e simulação de efeitos de repetição (tal como o reverb, delay) temos no mercado diversos tipos de *reverb*'s, os quais possuem diferentes aplicações que veremos a seguir:

- **Spring** ("Reverb-de-mola") - O efeito de reverberação com mola é um método que foi primeiramente proposto por *Hammond* em 1940s. O reverb de mola foi criado naturalmente, por um sistema mecânico, o qual usa um transdutor e um captador nas extremidades de uma mola, para criar e capturar vibrações dentro dela. Muitos amplificadores de guitarra incluem esse tipo de reverberação dentro de seus projetos. Existem ainda muitos pedais de reverb que oferecem esse efeito emulado digitalmente e apenas algumas companhias, por questões de custo de construção, também lançaram produtos com um sistema real de reverb de molas.

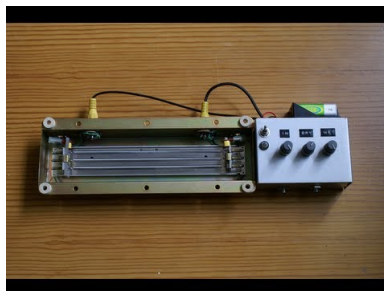


Figura 1.3: *Câmara de reverb de mola de um amplificador*

- **Room** ("Reverb-sala") - Este tipo de reverb é usado para simular um som natural de uma sala acústica, geralmente vazia, que tem um tamanho geralmente pequeno. Esses reverbs usualmente possuem reflexões curtas que desaparecem rapidamente com o tempo.
- **Hall** ("Reverb-palco") - Reverb's dessa categoria são usados para simular um tipo de reverberação encontrado num grande teatro ou até mesmo catedrais ou igrejas. Eles "soam" geralmente

mais fortes do que um *room reverb* por conta da quantidade de reflexões serem significativamente maiores e mais longas. Pode ser encontrado a variação de hall reverbs como "*cathedral*".

- **Plate** ("Reverb-de-placas") - Um plate reverb é definitivamente o efeito que exige uma grande logística a ser empregada. Uma grande máquina conforme mostrado na figura () a qual fornece o áudio dentro de grandes "placas" penduradas de metal que produzem um som de reverb que é mais definido do que o efeito do hall-reverb enquanto ainda é capaz de produzir longos decaimentos no tempo.

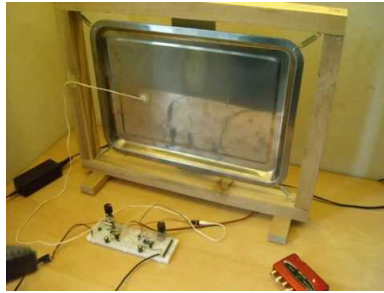


Figura 1.4: Pequeno protótipo de um reverb de placas.

- **Pitch-Shifter Reverb** aka **Shimmer** - Bem esse efeito de reverb será o foco de desenvolvimento desse trabalho. O shimmer tem se tornado bem comum entre pedais de guitarra principalmente nesses últimos anos. A grosso modo, esses reverbs adicionam componentes harmônicos do som original para dar a sensação de ambiência e harmonia e simulam a reverberação desses sons mais agudos na "cauda"(no final do reverb).

Este som tem características bem peculiares, as quais geralmente as pessoas associam a um som de orquestra angelical ou sintetizadores de teclado, dando uma intensa ambiência e atmosfera no som.

Diferente dos demais reverbs, o Shimmer é um efeito eminentemente sintetizado, ou seja, produzido digitalmente por meio de algoritmos embarcados em DSP's e comercializados como equipamentos de produção digital.

### 1.1.3 Pitch Shifting

#### 1.1.3.1 Conceitos Iniciais

Entre os efeitos digitais de áudio (especialmente os realizados em tempo real), esse é um dos que exige algoritmos mais sofisticados, e até recentemente, os resultados não eram convincentes.

A primeira consideração a ser feita sobre esse assunto seria entender o que é um *pitch shifting*. A melhor definição para o termo "pitch" de um som seria o conjunto de frequências de que o som é formado. A melhor maneira de entender esta definição é olhar por exemplo um coral. Nele há tanto mulheres quanto homens cantando mas as mulheres tem um "alcance"/pitch mais alto que



o homem. A mudança das frequências masculinas, por exemplo, para frequências mais altas, o faria parecer como uma voz feminina.

O efeito funciona comprimindo e, posteriormente expandindo o sinal que está sendo processado. A grosso modo, para transpor um som uma frequência mais acima, o sinal é tocado mais rápido, o que o torna mais curto. Então é preciso copiar segmentos do sinal processado e adicioná-lo ao sinal resultante para eliminar essa diferença temporal.

Para tornar um som mais grave, o sinal é reproduzido mais lentamente, o que requer o corte de algumas seções do sinal para diminuir sua duração. Ou seja, *pitch shifters* estão constantemente cortando ou colando pequenas porções do áudio a ser processado.

### 1.1.3.2 Escala de Frequência Musical - *Pitch Shift*

É sabido que a escala musical é dividida em várias oitavas. Cada oitava é composta por 12 (doze) semitons, também são referenciados como meio salto.

Cada nota corresponde a uma frequência fundamental que a compõe. Essa frequência é definida pela equação 1.1, onde  $p$  corresponde ao número de semitons e  $f$  a frequência em Hertz.

$$p = 69 + 12 \cdot \log(f/440) \quad (1.1)$$

Sob a óptica de sinais e sistemas, um pitch shifting consiste em deslocar a frequência fundamental e suas harmônicas por um fator específico, conforme mostrado na equação (). A equação consiste na obtenção da frequência final  $f_{final}$  dado a frequência inicial  $f_{inicial}$ , e os números de semitons " $s$ " os quais pretende deslocar.

$$f_{final} = 2^{(s/12)} \cdot f_{inicial} \quad (1.2)$$

Como já mencionado, existem 12 semitons por oitava. Isso implica que cada transposição para cima ou para baixo uma oitava é equivalente na escala do espectro a multiplicar por 2 ou 1/2 respectivamente. A figura () ilustra as componentes de frequência (com um fator de  $2^{(4/12)}$ ) o qual equivale a um *pitch shift* de 4 semitons acima (de C para E).

## 1.2 COMPARATIVO ENTRE SOLUÇÕES DE HARDWARE

De acordo com a seção 1.1.1, numa sala de concerto, o som que espectador ouve contém tanto o som original produzido pela fonte (voz, instrumento acústico, sistema de sonorização, etc) quanto às milhares de reflexões desse som original, que bate no chão, paredes e teto, até chegar aos ouvidos, com um pequeno atraso. Essas reflexões são como milhares de ecos do sinal direto que, devido à sua grande quantidade, não são percebidas exatamente como ecos, mas sim

como o efeito de “reverberação”.

Baseados na reflexividade de um ambiente podemos distinguir os materiais de que ele é composto. Em salas grandes com paredes elevadas de tijolo a reverberação geralmente é muito pesada e precisa de algum tempo até cessar. Já uma sala pequena, com muitos objetos dentro, possui uma reverberação muito pequena, em geral nem percebida como tal. Entretanto, essa pequena reverberação de fato existe, e por essa razão é que os projetistas de processadores de efeitos incluem vários tipos básicos de reverberações, dando a eles nomes de tipos diferentes de “salas” - room reverb. É muito natural, por exemplo, que uma programação de reverb chamada “Catedral” - reverb hall produza uma reverberação longa e muito densa, enquanto uma programação chamada de “Room” represente a acústica de uma sala muito menor (ALBAR, 2007).

Diante dessa realidade, no mercado, atualmente, várias são as empresas que possuem em seu portfólio esse tipo de efeito com essas características clássicas. Contudo, o efeito *shimmer* é um dos mais complexos a serem implementados na prática devido ao seu alto custo de processamento. Todos os detalhes serão abordados mais a frente neste trabalho.

#### NOTA

Vale destacar que o efeito puramente de reverberação já é notoriamente conhecido e manipulado por diversos hardwares e softwares no mercado. Note-se que estamos falando do efeito adicional incorporado ao produto que é o chamado *shimmer*.

### 1.2.1 Hardwares de Reverb-Shimmer

Apesar dos equipamentos ora aqui listados serem conservacionalmente populares no meio musical, tendo diversas aplicações em projetos musicais, infelizmente não se pôde obter detalhes técnicos específicos, em termos de performance de *hardware*, além do que consta em seus manuais ou algum *review* na internet, os quais foram de imensa ajuda principalmente para o produto da empresa Strymon (*BigSky Reverb*).

1. **Eventide H9:** A Eventide, Inc. (também conhecida anteriormente como Eventide *Clock Works Inc.*, ou hoje simplesmente como Eventide) é uma companhia de áudio, transmissões, comunicações e aviação dos Estados Unidos cuja divisão de áudio produz processadores de áudio, software DSP e efeitos de guitarra. Eventide foi uma das primeiras companhias a produzir processadores de áudio digital e seus produtos são fundamentais em gravação e reprodução de som, pós-produção e estúdios de transmissão. Ela possui um pedal de guitarra chamado H9 (1.5) "revolucionário" que embarca diversos tipos de efeito e possui grande poder de processamento e efeito com qualidade profissional num formato relativamente compacto. Neste pedal é possível importar algoritmos de efeitos de modulação, de repetição e diversas ambiências tal como o efeito de reverb shimmer tudo através de sua entrada serial universal (USB) utilizando um *software* proprietário.



Figura 1.5: Foto modelo Eventide H9 - Effects Processor.

2. **Strymon BigSky:** A empresa Strymon, fundada em meados de 2008, é considerada uma das empresas mais bem sucedidas no universos de pedais de guitarra de linha exclusiva (*boutique-pedals*) que integram algoritmos de tratamento de sinais de áudio extremamente avançados.

Vale salientar que com tantas grandes empresas produzindo diversos produtos, sendo por vezes um réplica do outro, é impressionante ver uma pequena empresa produzir um produto de qualidade notoriamente alta. Foi, de fato, uma reinvenção da roda em termos de qualidade dos efeitos e o resultado esperado pelo usuário final.

(a) **Qualidade de Áudio:**

- Baixo ruído na entrada do dispositivo, alta performance de áudio com resolução de 24-bit e taxa de amostragem de 96kHz, tanto para o conversor Analógico digital quanto para o conversor digital analógico.
- 115 db de relação sinal-ruído em 50

(b) **Processador:**

### 1.2.2 Micontroladores e DSP's

O termo sistemas embarcados constituem circuitos eletrônicos que utilizam processadores digitais (microprocessadores ou microcontroladores, etc.) em aplicações dedicadas para determinado equipamento ou produtos.

Os microcontroladores, diferentemente dos microprocessadores, em geral, possuem todos os periféricos necessários num único chip. Seu tamanho também é muito pequeno, mesmo contendo vários periféricos como: memórias, barramentos, *timers*, portas de comunicação, conversores de sinal analógicos para digital etc.

Por outro lado, esses dispositivos possuem um desempenho menor que os microprocessadores, mas são ideais em aplicações que necessitam de menores dimensões, tempo e custos.

As linguagens de programação das unidades processadores de sistemas embarcados podem variar, mas em geral, se limitam às linguagens C/C++, *Assembly* e *Java*.

Nessa linha, temos ainda o processador digital de sinais (DSP - *Digital Signal Processing*) e pode definir tanto o processador quanto o processo em si. Esse tipo de tratamento exige um alto desempenho para aplicações numéricas em tempo real.

Os DSP's são construídos para computar de forma eficiente equações de diferenças e algoritmos de transformadas diversas (como a *Fast Fourier Transform* - FFT). As aplicações dos DSP's, em suma, estão relacionadas com sistemas de controle de alta velocidade, realizações de filtros digitais, transformadas rápidas de *Fourier*, processamento de sons e imagens, entre outras.

### **1.2.3 Hardwares Comerciais do Efeito *Reverb* - *Shimmer***

### **1.2.4 Microcontrolador MSP430F5529 e TMS320**

Foram utilizados ao longo do projeto como soluções de hardware essencialmente o *microcontrolador launchpad MSP430F5529*.

Não obstante foram apontados na subseção anterior a respeito das limitações do hardware para tratamento de sinais de áudio em tempo real, o escopo do trabalho levou em consideração a economicidade do hardware em questão, bem como a utilização de operações no domínio do tempo, sem utilização de etapas intermediárias como realização da transformada de *Fourier* para manipulação do sinal de áudio.

Além disso, o projeto teve uma participação do desempenho do TMS320 C2000 o qual, devido ao tempo de aprendizagem do DSP para correta aplicabilidade no projeto, não fora possível.

## **1.3 OBJETIVOS E DIAGRAMA DE BLOCO DO PROJETO**

Nosso objetivo neste trabalho é o estudo e a realização de um projeto, desenvolvido em *software* e futuramente aplicado em *hardware*, de um efeito digital de áudio seletivo em frequência que tem como característica musical a repetição assíncrona do som (*reverb*) com destaque na utilização de componentes de frequências (harmônicas) na realização desse efeito (*shimmer*).

Nesta parte do trabalho, será mostrado de maneira abstrata os blocos necessários para que obtenha a saída do sistema desejado, ou seja, o efeito do *Reverb* com *Shimmer* implementado em software e executado seja em software ou em hardware.

Temos aqui na figura 1.6 um sinal analógico oriundo de um instrumento musical - guitarra, que passa por um processador de som digital - DSP, o qual procede com todo o trabalho de tratamento desse sinal analógico ora convertido em amostras digitais. Aplicando, portanto, nesse contexto, um filtro de resposta finita; um algoritmo de seleção de frequências denominado *pitch-shifter* e por fim realimentando o sistema acrescentando um pequeno atraso temporal (*delay*).

Não obstante ao modelo apresentado, também será objeto de estudo os blocos que compõem a

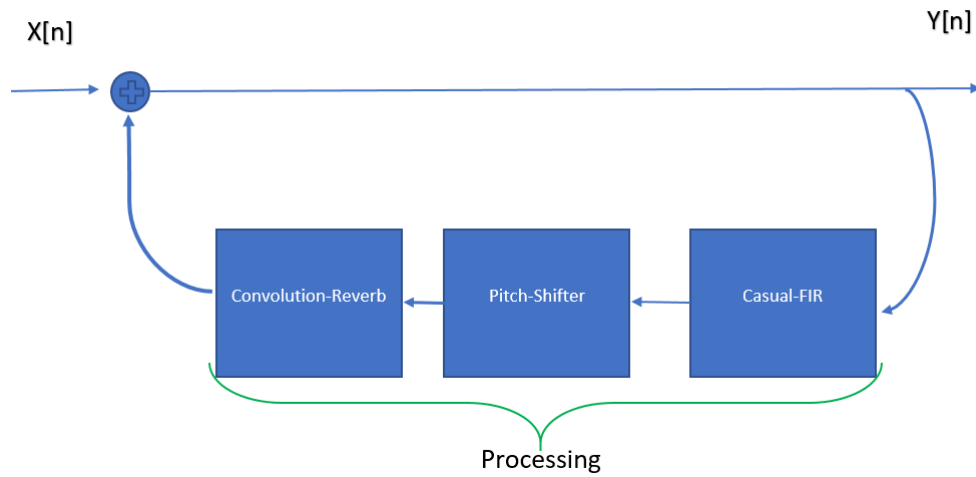


Figura 1.6: *Diagrama de Blocos Principal do Projeto.*

conversão do sinal analógico para a realidade de amostras digitais (conversão A/D), bem como o caminho inverso que propõe o envio das amostras a um conversor gerador de níveis de tensão elétrica correspondente (conversão D/A).

## 2 METODOLOGIA E FERRAMENTAS

### 2.1 SINAIS DE TEMPO DISCRETO E TRANSFORMADA DISCRETA DE *FOURIER*

#### 2.1.1 Problema da Amostragem de Sinal Contínuo

Antes da abordagem do efeito seletivo em frequência em estudo, devemos abordar de forma sucinta um importante teorema que serve de ponte entre os mundos do tempo contínuo e de tempo discreto<sup>1</sup>: O teorema da Amostragem.

Sabe-se que a transformada de Fourier de um sinal de tempo contínuo  $x(t)$  é dada por:

$$X(f) = \mathcal{F}[x(t)] = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft} dt \quad (2.1)$$

Na maioria das vezes, um sinal de áudio dificilmente terá uma representação analítica, como geralmente ocorre com os sinais no mundo real. Assim, teremos que representar o sinal  $X(f)$  numa quantidade finita de frequências, em outras palavras, é possível determinar somente um número finito de amostras de  $X(f)$ .

Nesse caso, uma forma de amostrar um sinal é através do concatenamento do sinal original  $x(t)$  por um trem de impulsos periódicos  $\delta_{T_s}(t)$ . O período,  $T_s$ , é chamado de intervalo de amostragem: é o espaçamento entre amostras consecutivas tomadas de  $x(t)$ .

O sinal amostrado  $x_s(t)$  é dado matematicamente pela seguinte equação, para  $n \in \mathbb{Z}$ :

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) \quad (2.2)$$

A transformada de Fourier neste caso para o sinal  $x_s(t)$ , pode ser obtida da seguinte forma:

$$\begin{aligned} X_s(f) &= \mathcal{F}[x(t)\delta_{T_s}(t)] = X(f) * \Delta_{T_s}(f) = \\ X(f) * f_s \sum_{k=-\infty}^{\infty} \delta(f - kf_s) &= \boxed{f_s \sum_{k=-\infty}^{\infty} X(f - kf_s)} \end{aligned} \quad (2.3)$$

Nota-se que o sinal  $X_s(f)$  é periódico com período  $f_s$  hertz. Considerando uma largura espectral do sinal  $X(f)$  como  $B$ , em que  $X(f) = 0$  para  $|f| \geq f_s/2$  - isto é,  $B \leq f_s/2$ , ou,

---

<sup>1</sup>Um sinal contínuo amostrado é uma sequência de impulsos, enquanto que um sinal em tempo discreto apresenta a mesma informação em uma sequência de números. Todos os conceitos aplicados a sinais amostrados se aplicam a sinais em tempo discreto.(HAYKIN; VEEN, 2001)

de forma análoga  $f_s \geq 2B$ .

Dessa forma podemos avaliar o sinal  $X(f)$  como sendo:

$$X(f) = T_s \cdot X_s(f) \quad (2.4)$$

, para  $|f| < f_s/2$ , se  $f_s \geq 2B$

Ou seja, se  $f_s \geq 2B$ , é possível obter  $X(f)$  a partir do sinal discreto  $X_s(f)$  que, por consequência, reconstruir  $x(t)$  a partir de  $x_s(t)$ . Esse resultado é conhecido como **teorema da amostragem**.

No nosso caso concreto, teremos que utilizar esses conceitos para avaliar uma taxa de amostragem coerente bem como conceitos relativos ao tipo de método usado para representar digitalmente amostras de um sinal analógico, por exemplo o som da guitarra ou de uma amostra musical.

Vale também salientar que o espectro  $X_s(f)$  pode ser obtido calculando-se diretamente a transformada de Fourier de  $x_s(t)$  usando-se as equações (2.1) e (2.2), pode-se mostrar que:

$$X_s(f) = \int_{-\infty}^{\infty} \underbrace{x_s(t)}_{x(nT_s)\delta(t-nT_s)} e^{-j2\pi ft} dt = \sum_{n=-\infty}^{\infty} x(nT_s) e^{-j2\pi f n T_s} \quad (2.5)$$

Substituindo na equação (2.4), para  $|f| < f_s/2$ , se  $f_s \geq 2B$  temos:

$$X(f) = T_s \sum_{n=-\infty}^{\infty} x(nT_s) e^{-j2\pi f n T_s} \quad (2.6)$$

Nota-se que a equação (2.6) pode ser usada também para o cálculo numérico de  $X(f)$ <sup>2</sup>.

### 2.1.2 Efeito do Janelamento

Apesar dessa discretização do sinal ainda tratamos de um sinal com somatório infinito, que, em geral, não será possível de ser calculado. Portanto, esse somatório precisa ser truncado, e consequentemente, não se terá mais a igualdade da equação (2.6), mesmo que  $X(f) = 0$  para  $|f| \geq f_s/2$ .

Uma consideração acerca desse truncamento é dada pela equação abaixo, para  $n \in [0, N-1]$ :

$$\hat{X}(f) = T_s \sum_{n=0}^{N-1} x(nT_s) e^{-j2\pi f n T_s} \quad (2.7)$$

Uma forma de tratar o truncamento, por questões de continuidade manter a continuidade do

---

<sup>2</sup>A expressão dada equação (2.6) para  $X(f)$  é a mesma que se obtém utilizando a regra do trapézio para solucionar numericamente a equação (2.1), contudo com essa abordagem é mais difícil visualizar com clareza aspectos importantes e limitações desse cálculo numérico

sinal amostrado, é considerá-lo como resultante do truncamento do sinal  $x(t)$ , realizado por meio da multiplicação desse sinal com um sinal  $w(t)$  de duração finita, geralmente  $w(t)$  é um sinal retangular. Chamaos o sinal  $w(t)$  de sinal-janela ou, simplesmente janela.

Assim definimos o sinal  $x_w(t)$  como sendo:

$$x_w(t) = w(t)x(t) \quad (2.8)$$

Escolhendo um tempo de janelamento  $T_w = NT_s$ , tem-se que  $x_w(nT_s) = 0$  para  $n \notin [0, N - 1]$ . Assim, substituindo na equação (2.6) podemos obter uma aproximação  $\hat{X}(f)$ , para  $|f| < f_s/2$ , dada por:

$$\hat{X}(f) = T_s \sum_{n=0}^{N-1} x_w(nT_s) e^{-j2\pi f n T_s} \quad (2.9)$$

Podemos considerar que a aproximação da equação (2.9) é idêntica a fornecida pela equação (2.7) pois, considerando que o sinal  $w(t)$  é uma janela retangular<sup>3</sup>, então  $x_w(nT_s) = x(nT_s)$ , para  $n \in [0, N - 1]$ .

Todavia o somatório dessas equações, embora finito, não poderá ser transformado em uma expressão analítica válida para qualquer  $|f| < f_s/2$ . Isto é, para cada valor de  $f$  dado, para o qual se deseje determinar  $\hat{X}(f)$ , será preciso calcular o referido somatório, o que não é nada prático.

Uma alternativa é o cálculo de  $\hat{X}(f)$  para valores de  $f$  espaçados de:

$$\Delta f = \frac{f_s}{N} \quad (2.10)$$

Ou seja, para  $f = k\Delta f$ ,  $k \in \mathbb{Z}$

Assim, usando a equação<sup>4</sup> (2.9), para  $|k| \leq N/2$ :

$$\hat{X}(k\Delta f) = T_s \sum_{n=0}^{N-1} x_w(nT_s) e^{-j2\pi k \frac{f_s}{N} n T_s} = T_s \sum_{n=0}^{N-1} x_w(nT_s) e^{-j \frac{2\pi}{N} k n} \quad (2.11)$$

Finalmente definindo o sinal de tempo discreto  $x_w[n] = x_w(nT_s)$ , a equação (2.11) pode ser reescrita como:

$$\hat{X}(k\Delta f) = T_s \sum_{n=0}^{N-1} x_w[n] e^{-j2\pi k \frac{f_s}{N} n} \quad (2.12)$$

---

<sup>3</sup> $w(t) = \text{rect}\left(\frac{t-T_w/2}{T_w}\right) = \begin{cases} 1, & 0 \leq t < T_w \\ 0, & \text{c.c} \end{cases}$

<sup>4</sup>Sendo  $|f| < f_s/2$  e  $f = k\Delta f$ , portanto  $|f| = |k|f_s/2 \Rightarrow |k| > N/2$ .



O somatório dessa equação é a transformada de Fourier discreta (DFT - *discrete Fourier transform*).

O conhecimento desse conceito matemático, bem como suas limitações quanto aos problemas de janelamento e de sobreposição espectral (aliasing) são fundamentais para entendimento das próximas atividades efetuadas.

## 2.2 FILTROS DIGITAIS

### 2.2.1 Conceitos Iniciais

Os filtros digitais não contém uma implementação física em si, diferentemente dos filtros analógicos constituídos, geralmente, de associação de resistores e capacitores. Eles são construídos através de algoritmos.

Para que isso possa ocorrer é necessário que o sinal de áudio (analógico) seja devidamente convertido em um sinal digital. esse sinal portanto convolui por um algoritmo de filtro adequado.

De maneira geral, o projeto de um filtro consiste em obter os coeficientes para os filtros. Isso é realizado através de uma equação chamada de equação das diferenças. O processo pode ser simplesmente realizado pela equação (2.13):

$$\text{Saida} = \sum_{1}^n \text{Coeficiente}_n \text{do filtro} * \text{Amostra}_n \quad (2.13)$$

Assim, o contexto de um filtro digital estará associado a equações de diferenças (ou funções de transferência no domínio Z) cujo parâmetros (coeficientes) serão calculados com o objetivo de discriminar (extrair, atenuar, etc.) determinadas componentes espectrais presentes em um sinal ou uma informação no mesmo sentido dos filtros analógicos, sem a necessidade de um circuito (*hardware*) adicional. Em outras palavras, o filtro digital será uma rotina adicional agregada ao algoritmo responsável pela realização do sistema proposto em questão.

### 2.2.2 Filtros IIR

Os filtros digitais de resposta infinita ao impulso (*Infinite Impulse Response - IIR*), também conhecidos como filtros recursivos ou autorregressivos, são modelados pela equação de diferença (2.14) ou pela função de transferência (2.15), em que basicamente os valores dos coeficientes dos modelos define a natureza do filtro (passa-baixa; passa-alta; passa-faixa; rejeita-faixa).

A denominação de IIR se deve que a saída do modelo decai para um valor nulo em um tempo infinito em resposta a um impulso aplicado na entrada filtro correspondente.

$$y(k) = \frac{1}{a_0} \left( \sum_{m=0}^M b_m x(k-m) - \sum_{n=1}^N a_n y(k-n) \right) \quad (2.14)$$

$$D(z) = \frac{y(z)}{x(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_m z^{-m}}{a_1 z^{-1} + a_n z^{-n}} \quad (2.15)$$

Resumidamente, a forma usual de calcular os coeficientes de um filtro digital IIR consiste em utilizar o modelo de um filtro analógico, e aplicar uma transformada Z via aproximação retangular ou trapezoidal (OPPENHEIN, 1998).

Notoriamente, uma das vantagens na utilização dos filtros IIR é que eles resultam em comprimentos (quantidade de coeficientes) de filtro menor do que o filtro FIR correspondente, porém, esta melhoria é obtida às custas de distorção de fase e um transitório que não se limita a um intervalo de tempo finito (ROBERTS, 1987).

Adicionalmente, conforme veremos na seção (??) deste capítulo, um exemplo clássico de um filtro IIR é o Filtro Pente, pois se estável, a resposta simplesmente consiste em repetir 'series de impulsos que decrescem em amplitude com o tempo.

### 2.2.3 Filtros FIR

Os filtros de resposta finita ao impulso (*Finite Impulse Response* - FIR), também conhecidos como filtros não recursivos ou de média nível, são modelados pela equação de diferenças (2.16) ou pela função de transferência (2.17), em que basicamente os valores dos coeficientes dos modelos define a natureza do filtro (passa-baixa; passa-alta; passa-faixa; rejeita-faixa). A denominação FIR se deve a saída do modelo decair para um valor nulo em um tempo finito em resposta a um impulso aplicado na entrada do filtro correspondente. Como características dessa categoria de filtros pode-se citar que são inerentemente estáveis (ao contrário dos filtros IIR), e apresentam variação de fase linear na faixa de frequência de operação (uma condição apropriada para aplicações em sistemas com modulação/demodulação de fase). Podemos ainda visualizar através da figura 2.1 a representação do filtro em diagrama de fluxo de sinais (figura 2.1(a)) e diagrama de blocos (figura 2.1(b)).

$$x(n) = \sum_{k=0}^{N-1} b_k x(n-k) \quad (2.16)$$

$$D(z) = \frac{y(z)}{x(z)} = b_0 + b_1 z^{-1} + \dots + b_n z^{-n} \quad (2.17)$$

A forma usual de calcular os coeficientes de um filtro digital FIR consiste em utilizar modelos de respostas ideais dos filtros e funções janelas associadas.

Observando o exemplo da função *Sinc*, nota-se que os valores das amplitudes desta tendem

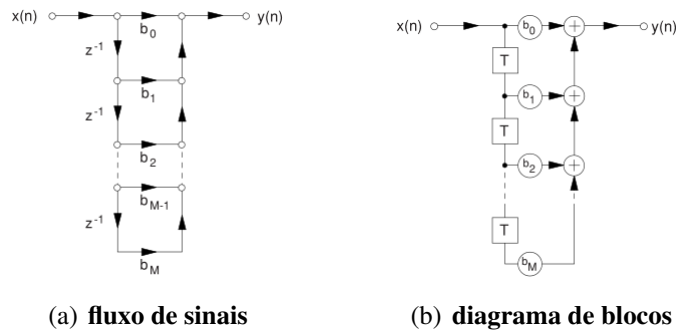


Figura 2.1: Diagrama de fluxo de sinais e diagrama de blocos de um filtro FIR

teoricamente aos limites não realizáveis  $-\infty < i < \infty$ , conforme já mencionamos na seção 2.1.1. Com a finalidade de truncar os valores correspondentes dos coeficientes dos filtros FIR associados em valores finitos, são empregadas funções janelas com este propósito e que também influenciam as atenuações resultantes dos filtros. A tabela 2.1 mostra as principais funções janelas que são utilizadas nos projetos de filtros FIR, em que o valor prático de atenuação  $\beta$  é indicado para cada função janela.

Tipo de Janela	$\beta[dB]$	Função Janela: $w(n)$
Retangular	-21	1
Bartlett	-25	$1 - 2 \left( \frac{2 n - \frac{N}{2} }{N} \right)$
Hanning	-44	$0,5 + 0,5 \cdot \cos \left( \frac{2\pi n}{N} \right)$
Hamming	-53	$0,5 + 0,46 \cdot \cos \left( \frac{2\pi n}{N} \right)$
Blackman	-74	$0,42 + 0,5 \cdot \cos \left( \frac{2\pi n}{N} \right) + 0,5 \cdot \cos \left( \frac{4\pi n}{N} \right)$

Tabela 2.1: Função janela (window) associadas com filtros FIR

Os valores das atenuações  $\beta$  dependem também do numero de coeficientes  $N$  utilizados em

um filtro FIR, em que quanto menor esse numero mais compacta será a realização computacional do filtro correspondente.

Vantagens gerais do filtro FIR:

- (a) Um filtro não recursivo como o filtro FIR é inerentemente estável. Conforme pode ser visto na função de transferência 2.17 ela é especificada em termos do zeros apenas no plano-z. Logo não há grandes preocupações em termos da escolha dos coeficientes que possam causar instabilidade no sistema, posto que seu LGR encontra-se estritamente dentro do semi-plano esquerdo do domínio-z.
- (b) O filtro FIR (resposta ao impulso) tem forma simétrica no seu espectro de frequência. Isso produz uma característica de fase linear ideal, ou seja, é equivalente a puramente um atraso temporal em todas as componentes de frequência passando pelo filtro. Em outras palavras, podemos dizer que o filtro FIR não tem *distorção de fase* (LYNN; FUERST, 1998).

Diante dessa análise de escolha do janelamento o procedimento típico de cálculo de coeficientes de um filtro FIR, é essencial especificar os valores de frequências e atenuações associadas aos tipos de janelas a serem utilizadas, o número de coeficientes e o tempo de amostragem (ou frequência de amostragem  $f_a = 1/T$  correspondente).

**Exemplo 2.2.1** *Um pequeno exemplo de aplicação do projeto de filtro digital FIR (passa-baixa) e uma função de janela Blackman, considerando uma quantidade de 7 (sete coeficientes) chamado comumente de Tap's.*

*Considerando uma taxa de amostragem igual a  $f_a = 200Hz$  e uma frequência de corte  $f_c = f_1 = 40Hz$ , têm-se:  $f_{norm} = f_1/(f_a/2) = 0.4$ ;  $\omega_1 = 0.4\pi$ . Computando a resposta ideal do filtro em questão, num intervalo  $-(N-1)/2 \leq n \leq (N-1)/2$  com a função janela especificada  $J(n)$  (Blackman - tabela 2.1).*

*(for n=-3; n < 4; n++) 0,42 + 0,5.cos( $\frac{2\pi n}{7}$ ) + 0,5.cos( $\frac{4\pi n}{7}$ )*

*Pode-se computar a resposta ideal de um filtro passa baixa cuja função é descrita como:*

$$h(n) = \frac{\sin(w_1 n)}{\pi n} \quad h(0) = \frac{w_1}{\pi} \quad (2.18)$$

*Daí, finalmente, temos os valores dos coeficientes do filtro:*

$$b(n) = J(n).h(n)$$

```
1      %%Codigo para calculos dos coeficientes de um filtro FIR
2      N = 51; f1 = 40; fa=200; fn= f1/(fa/2); Wl=pi*fn; ...
      %parametros e normalizacao
```

```

3      n = -(N-1)/2:(N+1)/2; h= (W1/pi)*sinc((W1/pi)*n); ...
      %Resposta ideal do FPB
4      %% Funcao Janela
5      J = 0.42 + 0.5*cos(2*pi*n/N) + 0.08*cos(4*pi*n/N); b = ...
      J.*h;
6      freqz(b,1,[0:1:100],fa); figure %Resposta em frequencia ...
      do filtro
7      t= 0:(1/fa):0.5; x = 2 + 0.1*sin(2*pi*60*t); %Entrada ...
      do filtro
8      y = filter(b,1,x); %acao da filtragem;
9      plot(t,x,'k:'); hold
10     stairs(t,y,'k'); hold off
11     xlabel('t[s]'); ylabel('x(t)="..."'; y(t)= "___" ');

```

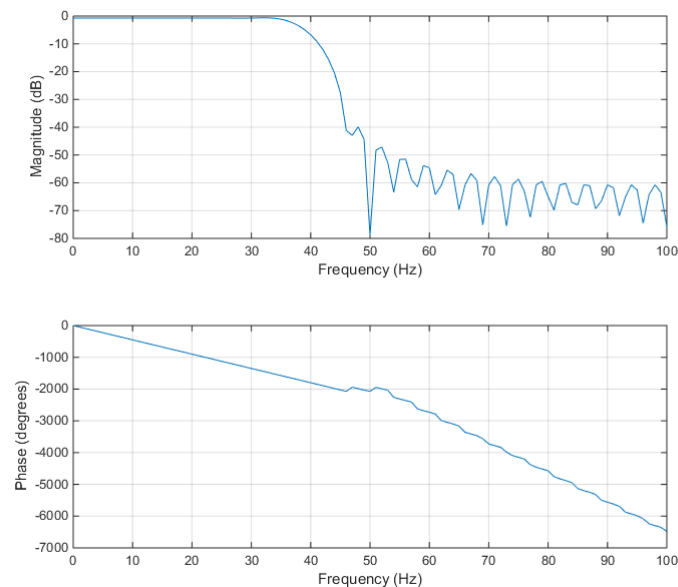


Figura 2.2: Resposta em frequência do FPB-FIR do exemplo 2.2.1 para  $N=51$ .

Vale destacar nesse exemplo que, por conta da janela *Blackman* ter 1 termo a mais na sua expressão de cálculo de coeficientes comparado a outros modos de janela (um termo cosseno extra), em outras palavras, mais termos indica mais acurácia nos resultados. Ademais, esse termo adicional também reduz os termos dos lóbulos laterais, melhorando assim o tamanho do lóbulo principal (CHAKRABORTY, 2013).

Outro detalhe importante, conforme visto na figura 2.4, é que o lóbulo principal, por exemplo, da janela retangular tem aproximadamente a metade da largura do lóbulo principal da janela *Hamming*;

De maneira análoga, os lóbulos laterais da janela *Hamming*, em relação ao lóbulo principal, são muitos reduzidos em comparação com os da janela retangular. Especificamente, o pico de

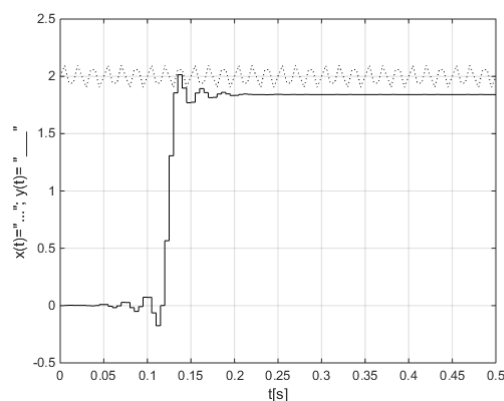


Figura 2.3: Ação da filtragem do FPB-FIR do exemplo 2.2.1

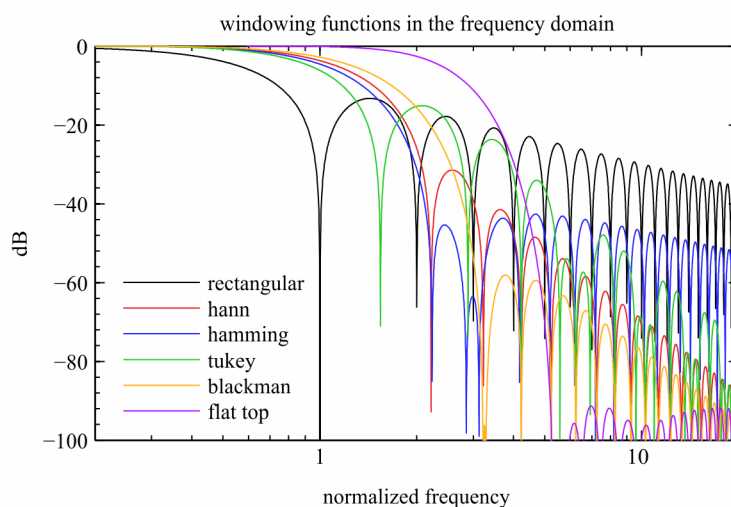


Figura 2.4: Resposta em frequência dos modos de Janelamento.

amplitude do primeiro lóbulo lateral da janela retangular está somente aproximadamente 13 dB abaixo do pico do lóbulo principal, ao passo que o valor correspondente para a janela *Hamming* é de aproximadamente 40 dB.

É devido a esta último ponto que a janela *Hamming* reduz as oscilações na resposta em frequência de um filtro digital FIR. Todavia, há um preço a ser pago por esta melhoria, a saber, uma faixa de transição mais larga no espectro do filtro.(HAYKIN; VEEN, 2001)

## 2.2.4 Filtros Adaptativos

Os filtros adaptativos são constituídos, geralmente, por estruturas FIR, em que os coeficientes dos modelos associados são modificados conforme um procedimento adaptativo. essa modalidade de filtro geralmente é empregada nos seguintes contextos (*lista não exaustiva*):

- Como procedimento alternativo na obtenção de valores dos coeficientes de um determinado

filtro FIR, em que padrões de entrada e saída conhecidos são utilizados para estabelecer os valores dos coeficientes do filtro em questão;

- Cancelamento ou redução de ecos/barulhos de um determinado ambiente;
- Na modelagem de sistemas dinâmicos; e
- Como modelagem básica de representações de redes neurais artificiais.

A equação 2.19 representa o modelo de um filtro FIR, em que  $W_m(k)$  denota os valores dos coeficientes do filtro em um instante de tempo  $k$ .

$$y(k) = \sum_{m=0}^M W_m(k)x(k-m) \quad (2.19)$$

A diferença ou erro  $e(k)$  entre o valor de padrão desejado  $d(k)$  para a resposta do filtro e a informação da saída atual  $y(k)$  do modelo associado é expressa pela equação 2.20:

$$e(k) = d(k) - y(k) \quad (2.20)$$

Basicamente para ajustar os valores dos coeficientes de um filtro adaptativo tipicamente utiliza o método do gradiente para essa finalidade (—inserir fonte—), sendo o critério da somatória do erro quadrático de  $e(k)$  frequentemente utilizado na etapa de adaptação.

Vale salientar, que alguns sistemas de comunicação de voz utilizam filtros adaptativos com o objetivo de cancelar ou reduzir ecos ou barulhos do ambiente. Nesse contexto, foi pensado inicialmente a utilização desse modelo de filtro para o projeto. No entanto, será explicado mais a frente a não adoção desse modelo, bem como pela utilização de um filtro FIR típico.

### 2.2.5 Filtro Pente - *Comb Filter*

Em termos práticos *comb filter* ou “filtro pente” é uma versão atrasada do mesmo sinal, causando uma interferência construtiva ou destrutiva de dois sons tocados simultaneamente, porém com atraso entre um para o outro.

Nesse sentido, há basicamente dois tipos de *comb filters*: o (*feedback comb filter*) e (*feed-forward comb filter*). Resumidamente, os nomes referem-se à direção em que os sinais são atrasados antes de serem adicionados à entrada. O primeiro tipo considera em adicionar a saída do filtro a entrada imediatamente posterior, enquanto o segundo considera apenas adicionar à saída do filtro as entradas presente e a mesma entrada atrasada.

Como já informado, o caso do filtro pente com realimentação na saída, é um caso especial de um Filtro de Resposta Infinita (??), posto que pode ser observado uma figura de atraso (*delay*) no sentido de realimentação no sistema. Este filtro pode ser um modelo físico computacional

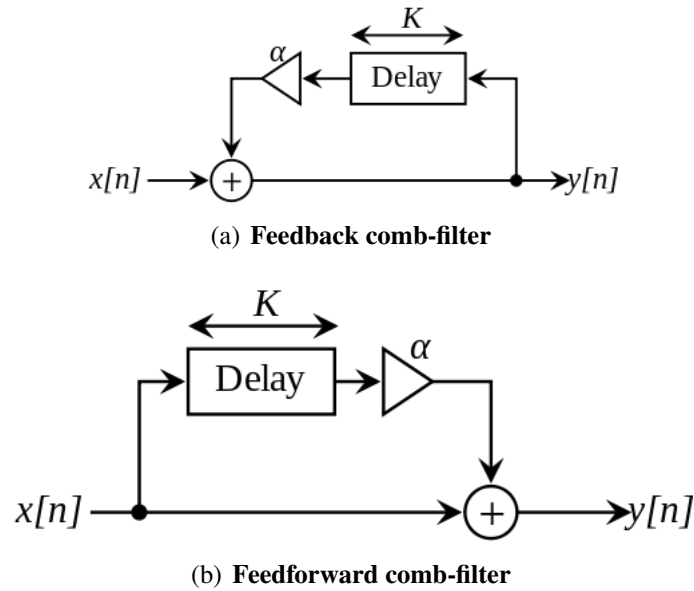


Figura 2.5: Filtro Pente - Comb Filter - Formas de Realimentação

de "séries de ecos", os quais decaem exponencialmente, bem como espaçados uniformemente no tempo (III, 2010).

Continuando a análise do modelo do referido filtro, ve-se que a estrutura geral de um sistema de realimentação de um *feedback comb filter* pode ser mostrado através da figura 2.5(a). Além disso pode ser descrito pela seguinte equação<sup>5</sup> de diferenças 2.21:

$$y[n] = x[n] + \alpha y[n - K] \quad (2.21)$$

Se rearranjarmos os termos da equação para que todos as variáveis em  $y$  fiquem do mesmo lado da equação e, na sequência, aplicamos a transformada Z em ambos os membros, teremos:

$$(1 - \alpha z^{-K})Y(z) = X(z) \quad (2.22)$$

Finalmente temos a função de transferência (2.23) correspondente ao sistema:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - \alpha z^{-K}} = \frac{z^K}{z^K - \alpha} \quad (2.23)$$

Uma das formas de estimar a resposta de magnitude em função do valor do tempo de delay  $K$  e fator de atenuação/ganho  $\alpha$  é expressar  $H(z)$  em termos de módulo  $|H(K, \alpha)|$ , bem como, de maneira conveniente, utilizando uma substituição  $z = e^{j\omega}$ .

<sup>5</sup>Para um critério de estabilidade o coeficiente  $\alpha$  da equação 2.21 tem que ser menor ou igual a 1 (0db).



$$|H(e^{j\omega})| = \frac{1}{|1 - \alpha e^{-j\omega K}|}, \quad -\pi \leq \omega \leq \pi \quad (2.24)$$

Nota-se pela expressão do ganho dada pela equação (2.24) que seu valor tem valores de resposta **periódica** indo de um valor mínimo e subindo para um valor máximo.

Supondo o valor de  $\alpha = 1$  (0 dB), para simplificar os cálculos, a amplitude da resposta reduz-se a:

$$H(w) = \frac{1}{2|\sin(\omega M/2)|} \quad (2.25)$$

Note ainda que para  $\alpha > 0$  são produzidos picos de ressonância em:

$$\begin{aligned} \sin(\omega K/2) &= 0 \\ \omega \frac{K}{2} &= p\pi, \quad p = 0, 1, 2, \dots, K-1 \\ w_p &= 2\pi \frac{p}{K} \end{aligned} \quad (2.26)$$

Ou seja, em todos harmônicos pares do filtro.

Ademais, esses filtros são utilizados em toda sorte de efeitos de sons, principalmente no universo de instrumentos musicais. Vários desses filtros podem ser usados por exemplo para simular uma reverberação.

Por último, vamos falar do modelo proposto para o projeto deste trabalho, no qual consiste de uma versão mais generalista do *feedback comb-filter*. Este modelo propõe a alocação de um filtro casual  $H_l(z)$  na malha de realimentação, em vez de apenas um ganho de atenuação  $\alpha$ . A função de transferência 2.22 pode ser então reescrita como:

$$H(z) = \frac{1}{1 - H_l(z)z^{-K}} \quad (2.27)$$

#### NOTA

Para o critério de estabilidade do sistema, a amplitude de resposta do filtro  $H_l(z)$  deverá ser menor que 1 em todas as frequências, i.e.,  $|H_l(e^{j\omega T})| < 1, \forall \omega T \in [-\pi, \pi)$

## 2.3 TITLE

## 2.4 CONVERSÃO ANALÓGICA DIGITAL

### 2.4.1 Considerações Iniciais

### 2.4.2 Conversão Por Aproximações Sucessivas

Resumidamente, conforme pode ser observado na figura 2.6 este tipo de conversor utiliza uma técnica de realimentação para relacionar uma voltagem analógica de entrada com um código digital correspondente (conforme os  $N$  bits de resolução do conversor). No início do processo de conversão o *shift register* e o *holding register* são zerados. Na primeira etapa de conversão o MSB (bit mais significativo) do *holding register* é colocado em nível alto (1 lógico) e os demais mantidos em nível baixo (0 lógico). É, então, realizada uma comparação entre o resultado de saída do conversor D/A ( $V_O$ ) e o sinal de entrada ( $V_{IN}$ ). Se  $V_O < V_{IN}$ , o nível “1” é mantido para o MSB, caso contrário é substituído por “0”. A etapa seguinte repete o mesmo processo para o 2-SB. Isso continua até que todos os  $N$  bits tenham sido verificados. A decisão de manter o nível lógico “1” ou substituir por “0” é realizada pelo comparador e pelo registrador de aproximação sucessiva. O controle lógico controla o início e o fim de cada etapa de aproximação e o resultado destas etapas são retidas no *holding register*. O sinal de saída é válido apenas quando todo o processo for concluído e isto é sinalizado pelo sinal de *status* do controle lógico.

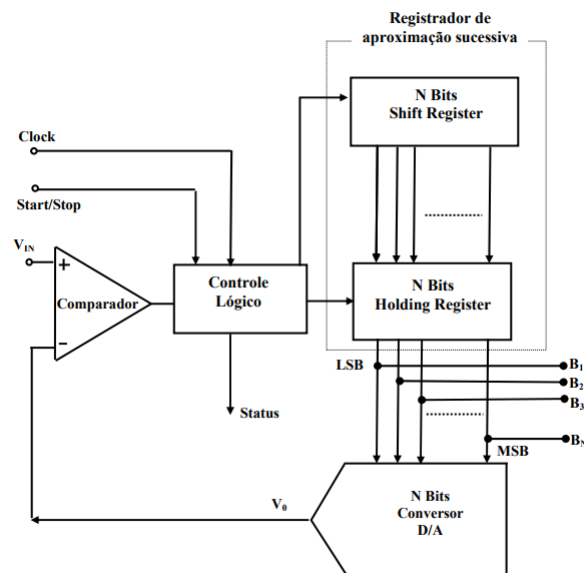


Figura 2.6: Conversor A/D de aproximações sucessivas.

### 2.4.3 Dificuldades Práticas de Conversores ADs

Apesar do princípio de funcionamento de um conversor AD ser relativamente simples, em suma, a operação desse dispositivo não é uma simples conexão de um sinal analógico na entrada de um microcontrolador. Listamos aqui algumas das dificuldades naturalmente encontradas:

- **Intervalo de Tensão de Entrada:** Bem, se conectarmos uma guitarra que produz em seus captadores digamos valores em  $\pm 100mV$  não será possível conectar diretamente essa tensão num conversor de um microcontrolador que espera valores entre  $1V$  a  $3V$ . Neste caso é necessário que o sinal da guitarra seja amplificado e modificado para que ele "ajuste-se" ao ADC.
- **Tensão de Referência:** Um ADC não tem uma noção absoluta de tensão, em outras palavras, sua tarefa é comparar sua entrada com a tensão de referência e sua saída numa proporção de 2. Assim, uma saída precisa recair na qualidade da referência e isso limita a acurácia<sup>6</sup> em geral.
- **Ruído e filtragem:** Sinais no mundo real contém frequências indesejadas, no nosso caso, uma guitarra ligada numa cadeira de pedais com fontes de alimentação cuja frequência da rede elétrica gira torno de  $50 - 60Hz$ . Nesse caso filtros são necessários para remover esse ruído ainda de iniciar a conversão ADC propriamente dita. Vale apontar, conforme seção 2.1.2 que um filtro passa-baixa é também necessário para evitar-se o efeito de *aliasing*.

## 2.5 CONVERSÃO DIGITAL ANALÓGICA

## 2.6 COMUNICAÇÃO SERIAL I2C

O protocolo  $I^2C$  (Inter Integrated Communication ou Comunicação entre Integrados) é um protocolo síncrono half duplex a dois fios do tipo mestre-escravo, criado pela *Philips* com objetivo de facilitar o desenvolvimento de sistemas modulares para televisores e outros aparelhos eletrônicos de consumo geral (BRAGA, 2012).

O funcionamento do protocolo baseia-se em alguns princípios básicos e usa duas linhas bidirecionais:

- Serial data (SDA);
- Serial Clock (SCL).

1. A informação presente na linha de dados (SDA) somente é lida durante a fase alta da linha de clock (SCL);

---

<sup>6</sup>text

2. Somente é permitido alterar o nível da linha de dados (SDA) durante a fase baixa da linha de *clock* (SCL).
3. Quando o barramento não está em uso, ambas as linhas permanecem desligadas (e, portanto forçadas as em nível "1" pelos resistores de *pull-up*).

### 2.6.1 Formato de Dados

Como o protocolo em questão permite a coexistência de diversos dispositivos em um mesmo barramento, é necessário que cada um possua identificação ou endereço próprio.

O formato básico de um comando  $I^2C$  é constituído por 7 *bits* de endereço<sup>7</sup>, utilizados para especificar o dispositivo escravo a ser acessado, seguidos por um *bit* indicador de leitura/escrita, em geral é o LSB, chamado de  $R/\overline{W}$

- $R/\overline{W} = 0$ : significa que o modo mestre escreverá dados no dispositivo em modo escravo. O mestre é o transmissor e o escravo o receptor<sup>8</sup>.

Normalmente o endereço de 7 *bits* é composto por duas partes: a primeira, de 4 *bits*, especifica o tipo de dispositivo escravo a ser acessado, no nosso caso o endereço do dispositivo MCP 4725. A segunda, de 3 *bits*, especifica um entre até oito dispositivos daquele tipo, o qual poderá ser acessado.

## 2.7 FERRAMENTAS COMPUTACIONAIS

### 2.7.1 Code Compose Studio

### 2.7.2 WinFilter

O *WinFilter* é uma ferramenta de *software* criada com o objetivo de provê a realização de projetos de filtros digitais. Ele possui uma interface gráfica bastante simples e bem intuitiva (KUNDERT, 2004). Esse *software* pode facilmente projetar filtros IIR, bem como filtros FIR e ainda pode gerar códigos do projeto em linguagem C e também em código VHDL. Os modelos de filtro digitais, no entanto, são limitados a apenas esses modelos abaixo:

- *Butterworth*;
- *Chebyshev I*;

---

<sup>7</sup>Existem uma extensão de endereçamento utilizando 10 – *bits* mas são raras as redes de dispositivos  $I^2C$  que consigam esgotar a quantidade de endereços de 7 – *bit*, ou seja, raramente essa possibilidade é utilizada.

<sup>8</sup>Em todos os casos o dispositivo *master* é responsável por toda a transmissão

- Bessel;
- Retagunlar; e
- Cosseno Levantado.

Todos os filtros IIR são transformados do domínio  $S$  para o domínio  $Z$  com uma transformação bilinear<sup>9</sup> e o método de correção de dobramento espectral (*aliasing*).

O WinFilter é composto por uma caixa de diálogo à esquerda, conforme visto na figura ??, e a direita uma série de disposição dos gráficos de resposta do filtro correspondente.

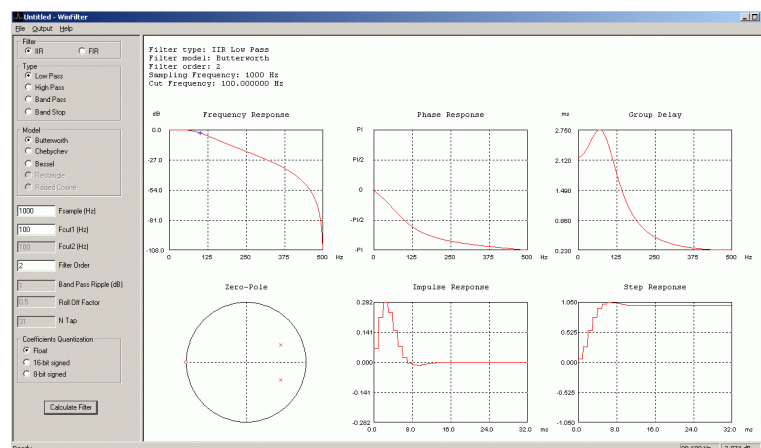


Figura 2.7: Aparência do Programa WinFilter - versão 0.8 de 06-08-2004

Como pode ser observado, há também botões para que seja aumentado ou atenuado alguns parâmetros do filtro, tais como taxa de amostragem, frequência de corte 1, frequência de corte 2, ordem do filtro dentro outros. O botão "**Calculate**" pega os parâmetros selecionados e os mostram através do display de gráficos à direita do programa.

A resposta em frequência, fase e grupo de delay são calculados. Os zeros e polos são plotados no plano  $Z$ . A resposta do filtro ao impulso unitário também é mostrada.

Uma importante ferramenta é que você pode avaliar nos gráficos a direita com o cursor do mouse os valores do gráfico em determinado ponto especificamente, conforme visto na figura ??:

<sup>9</sup>Bilinear Transform: [https://en.wikipedia.org/wiki/Bilinear\\_transform](https://en.wikipedia.org/wiki/Bilinear_transform)

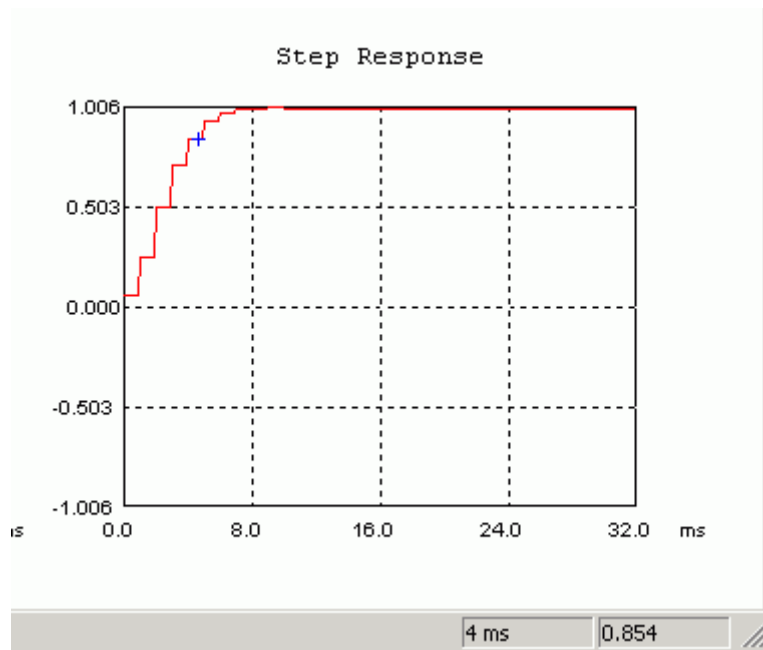
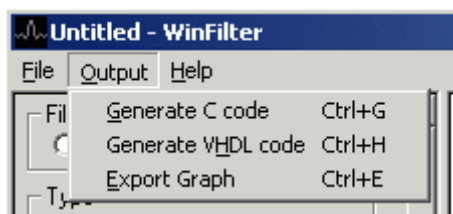
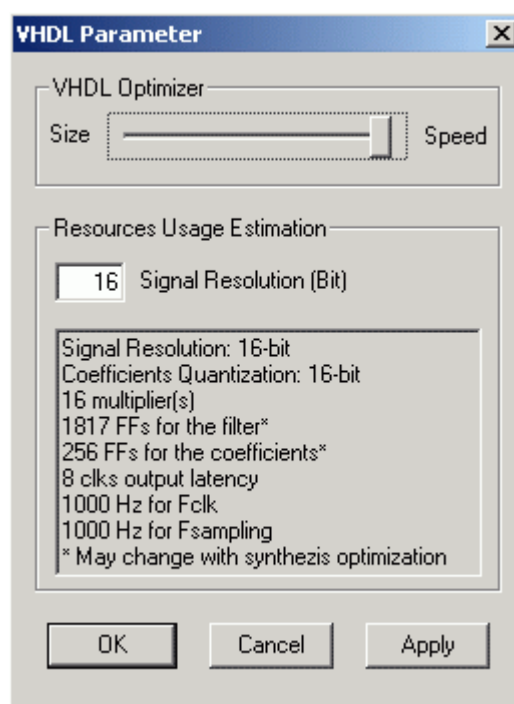


Figura 2.8: Valor lido no gráfico por meio do cursor do mouse. 4ms – 0.854

No menu *Output* (figura 2.9(a)) o programa permite que o código na linguagem C pode ser gerados bem como os gráficos ora mostrados na região à direita do programa também possam ser exportados em um *Graphic Interchange Format* - GIF. Além disso pode ser exportado o código gerado via VHDL em dois modelos (figura 2.9(b)): Sized ou Speed Optimized (modo de velocidade otimizada). Ambos os modos conseguem velocidades acima de 160 Mhz num dispositivo, por exemplo, modelo Ep1c3t144 Cyclone (grade-4) FPGA



(a) Opções de exportação do Programa



(b) Opções VHDL

Figura 2.9: Menu Output - programa WinFilter

# 3 IMPLEMENTAÇÃO DO PROJETO

## 3.1 BLOCO 0 - CONVERSÃO A/D NO MSP430

### 3.1.1 Análise de *Hardware*

Antes do sinal chegar ao microcontrolador para o correto processamento do sinal de áudio, será necessário, conforme já relatamos, amostrar esse sinal através de valores corretamente quantizados. Para isso, conforme abordado na seção 2.4 deste trabalho, é necessário utilizar um Conversor Analógico Digital - ADC.

O modelo do MSP 430 em questão (MSP430F5529LP) possui um ADC interno de 12 (doze) bits (ADC12). Ou seja, é possível representar em níveis de pulso até 4096 níveis. Algumas características podemos listar pois será objeto de avaliação dentro do projeto (DAVIES, 2008).

- Resolução de 12 bits monotônica, sem perdas de código;
- Velocidade nominal de até 200.000 amostras por segundo (200 Ksps), utilizando a técnica de aproximação sucessivas (SAR);
- Operação de com diversas referências internas de tensões: 1.5V, 2.0V, ou 2.5V com consumo típico de aproximadamente  $250\mu A$  quando em operação;
- Canais de entrada exclusivos para sensor de temperatura interno, tensão de alimentação e tensões de referências externas;
- 16 memórias de conversão com controle independente de cada uma, inclusive com a capacidade de especificar o canal de entrada e referência;
- Fonte de *clock* selecionáveis por softwares;

O coração de funcionamento desse módulo do microcontrolador consiste basicamente no seguinte:

- (i) O processador usa dois níveis de tensões selecionáveis:  $V_{R+}$  e  $V_{R-}$  afim de determinar o valor mínimo e máximo do conversor;
- (ii) Uma saída digital ( $N_{ADC}$ ) é setado no nível máximo ( $4095 = 0FFFh$ ) quando o valor de entrada for igual ou até mesmo maior que  $V_{R+}$ . De modo semelhante o valor digital ( $N_{ADC}$ ) será zero quando o valor de entrada for igual ou menor que  $V_{R-}$ .



(iii) A equação básica da conversão é:

$$N_{ADC} = 4095 \cdot \frac{V_{in} - V_R^-}{V_R^+ - V_R^-} \quad V_{in} : \text{Tensão de entrada.} \quad (3.1)$$

- (iv) Em especial, o módulo ADC12\_A é configurado por dois registros de controle: o ADC12CTL0 e ADC12CTL1.
- (v) O modo de funcionamento do conversor (conversão simples ou sequência de canais) pode ser configurado pelos *bits* CONSEQx (registrador ADC12CTL1);
- (vi) Após isso, seleciona-se o endereço inicial da memória de conversão, pelos *bits* CSTARTADDx (registrador ADC12CTL1);
- (vii) Por fim, liga-se o conversor (*bit* ADC12CTL0:ADC12ON=1) e habilitam-se as conversões (*bit* ADC12CTL0:ENC=1).<sup>7</sup>

### 3.1.2 Requisitos de Hardware para a Tempo de Amostragem

#### 3.1.2.1 Contextualização

Um ponto importante a ser considerado, quando convertemos um sinal analógico qualquer numa sequência de valores digitais, é a precisão que estes valores representam o sinal original.

É claro que, na prática, é conveniente, para a escolha da taxa de amostragem, usar frequências muito maiores do que 2 vezes a do sinal e isso ocorre, por exemplo, quando possível. Como é o caso de sinais de áudio, especialmente em CDs players, em que a frequência de amostragem é de 44,1 kbytes por segundo onde o que nos leva a 2 vezes a frequência máxima que podemos ouvir que é de 20 kHz.

Por outro lado, o microcontrolador, em especial para a aplicação ora pretendida, são equipamentos alimentados por bateria e o consumo do dispositivo, nesse caso, é um requisito muito importante. Como a complexidade na obtenção das amostras aumenta em função da quantidade de amostragem que pode fazer e a potencialidade do microprocessador usado, é extremamente importante a escolha viável de uma taxa de amostragem (figura ??) que atendam aos requisitos do projeto, ponderados com as limitações do equipamento em questão (BRAGA, 2012).

#### 3.1.2.2 Tempo de Amostragem do ADC12 do $\mu C$ MSP430

O conversor ADC do  $\mu C$  aguarda a ocorrência de um sinal de disparo de conversão (SAMP-CON) que pode ser originado de uma das quatro fontes<sup>1</sup> selecionadas pelos bits SHSx (registrador

---

<sup>1</sup>SHS\_1 por exemplo é proveniente da interrupção de um evento de comparação em um dos canais do timer A (saída TA1).

Taxa de amostragem (amostragens por segundo)	Uso
8k	Walk-talkies, intercomunicadores, microfones sem fio
11,025 k	Áudio MPEG, subwoofer e outros de menor fidelidade
16k	Extensão para telefone, VoIP e VVoip
22,050 k	PCM e MPEG de baixa qualidade
32k	NICAM, Rádio Satélite, e microfones sem fio de alta qualidade
44,056k	Áudio do sistema NTSC de TV
44,1 k	Áudio CD, MPEG1 em MP3, VCD, SVCD, PAL e microfones sem fio encriptados
47,5	Gravador de som PCM – primeiros tipos
48 k	Som do vídeo digital profissional, TV digital, som do DVD e filmes – SDI e HD-SDI, compressores e outros equipamentos de som
50 k	Primeiros gravadores de som digital como os da 3M dos anos 70
50,4 k	Gravador Mitsubishi X-800
88,2 k	Equipamentos de som digital, visando CDs de 44,1 k, mixers, equalizadores, câmaras de eco, etc.
176,4 k	Gravadores de som HD-CD e produção de CDs
192 k	Áudio em DVD, DVD LPCM, Som Blu-ray e som HD-DVD
352,8 k	Digital eXtreme Definition usado na gravação de Super Audio CDs
2 822,8 k	SADC com modulação delta-sigma processo conhecido como Direct Stream Digital da Sony e Philips
5 644,8 k	RSD de Taxa dupla, no sistema de 1 bit Direct Stream Digital x2 – Usado em gravadores profissionais DSD

Figura 3.1: Algumas taxas de amostragem utilizadas na prática

ADC12CTL1). Quando  $SAMCOMP = 0$ , todas as entradas analógicas ficam em alta impedância, caso contrário ( $SAMCOMP = 1$ ), uma entrada  $A_x$  pode ser modelada como um circuito  $RC$ , passa baixa, durante o tempo de amostragem da conversão ( $t_{sample}$ ).

Uma equação, conforme o Guia do Usuário da Família de Microcontroladores MSP430x5xx, pode ser usada para cálculo do valor mínimo da taxa de amostragem para uma conversão de  $n$ -bits, no qual  $n$  é igual a quantidade de bits de resolução do conversor.

$$t_{sample} > (R_s + R_i) \cdot \ln(2^{n+1}) \cdot C_i + 800ns \quad (3.2)$$

A equação 3.2 é oriunda de uma análise de um circuito  $RC$  em descarregamento cuja resposta no domínio do tempo da tensão de saída é dada por:

$$V_{out} = V_{in} e^{-t/RC}$$

Temos no caso, para uma quantização dos valores de tensão de entrada valores que podemos mensurá-los em MSB e LSB. Considerando o caso mais simples onde a tensão de entrada estaria entre um bit 0 e 1 de uma quantização com apenas  $n$  byte ( $2^n$  LSB's) obtemos:

$$\begin{aligned} \frac{V_f}{V_i} &= e^{-t/RC} \Rightarrow \ln\left(\frac{V_f}{V_i}\right) = -\frac{t}{RC} \\ \ln\left(\frac{\frac{1}{2}LSB}{2^n LSB}\right) &= -\frac{t}{RC} \Rightarrow \boxed{t = RC \cdot \ln(2^{n+1})} \end{aligned} \quad (3.3)$$

Por conseguinte, com SHP = 1 temos o modo de amostragem temporizada. O tempo de amostragem será determinado por um *timer* interno, que é configurado de acordo com os bits SHT1x e SHT0x (registrador ADC12CTL0). Os *bits* SHT1x determinam o tempo de amostragem (em ciclos de clock do conversor) para as memórias ADC12MEM8 até ADC12MEM15, tal como os *bits* SHT0x determinam o tempo de amostragem para as memórias ADC12MEM0 até ADC12MEM7.

Em suma, calculado o tempo de amostragem

## 3.2 BLOCO 1 - PROJETANDO UM FILTRO FIR

### 3.2.1 Para o código Experimental do MATLAB

Para o código que foi elaborado com o objetivo de análise do efeito *shimmer* e suas limitações em termos da discretização do sinal no domínio do tempo na perspectiva de síntese e análise do sinal.

Nesse caso, o filtro escolhido foi um filtro FIR-passa baixa cuja frequência de corte do projeto foi no valor de 2kHz. Tendo esse filtro utilizado a quantidade de 51 *Taps* (tamanho da janela), cujo janelamento foi feito com a janela *Blackman* (equação 3.4), cujos benefícios já foram explicados anteriormente.

$$J = 0.42 + 0.5 \cos((2\pi t)/Ntaps) + 0.08 \cos((4\pi t)/Ntaps); \quad (3.4)$$

Com isso bastou-se multiplicar a resposta da janela com um filtro ideal passa-baixa ideal (equação 3.5), cuja frequência normalizada vale  $W_1 = \pi \cdot \left(\frac{2000}{44100/2}\right)$ :

$$h = (W_1/\pi) \cdot \text{sinc}((W_1/\pi)t); \quad (3.5)$$

Após a multiplicação basta agora efetuar a operação de convolução com o sinal original de 12 bits resultando num sinal de 12 bits filtrado em 2kHz.

### 3.2.2 Para o projeto do $\mu C$

Com respeito ao projeto do microcontrolador, sabe-se que há uma limitação do próprio hardware em trabalhar com valores em ponto flutuante. Nesse sentido a utilização do software WinFil-

ter foi de suma importância pois ele fornece um vetor de coeficientes do filtro FIR correspondente de acordo com os parâmetros do projeto, algo extremamente prático.

Além disso ele fornece uma pequena função que já calcula a convolução das amostras do filtro, no entanto a implementação é ineficiente devido que para cada nova amostra que chega do sinal, ele desloca todas as amostras para então realocar espaço para amostra que chega:

Listing 3.1: Código Gerado no WinFilter para um filtro FIR

```
1 //shift the old samples
2 for(n=Ntap-1; n>0; n--)
3 x[n] = x[n-1];
4
5 //Calculate the new output
6 x[0] = NewSample;
7 for(n=0; n<Ntap; n++)
8 y += FIRCoef[n] * x[n];
9 return y / DCgain;
```

Nesse caso, há um desperdício de processamento o qual é solucionado com a implementação de um ponteiro que sempre sobrescreva as amostras antigas assim que ele chegar no fim do vetor (*buffer*) predeterminado para carregar as amostras vindas do conversor A/D. Essa forma é conhecida como *ring-buffer* (buffer em anel - figura 3.2).

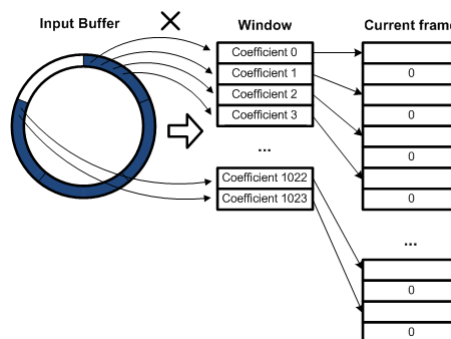


Figura 3.2: Uma pequena ilustração como funciona um ring-buffer

Depois de superada essa parte do processamento a convolução das amostras se dá na segunda parte do código 3.1.

## 3.3 BLOCO 2 - IMPLEMENTANDO O *PITCH-SHIFTER*

### 3.3.1 Análise do Algoritmo

#### 3.3.1.1 Conceito Preliminar - O que é necessário implementar?

Nós sabemos que o objetivo agora nesse bloco é alterar a frequência do sinal sem alterar sua duração. Uma forma que aparentemente funcionaria é que: se gravarmos um som e então faz-lo tocar duas vezes mais rápido, então a frequência desse sinal seria dobrada e então deslocada a frente 1 oitava. Isso está correto, no entanto o sinal é **duas vezes também mais curto**.

A maneira de fazer isso corretamente é pegar o som gravado, dobra-lo de tamanho sem afetar sua frequência natural e então toca-lo duas vezes mais rápido, logo todas as frequências dobrariam de valor e então, devidamente deslocadas, bem como a duração do sinal nesse caso estaria mantida conforme o som original. O algoritmo do MATLAB, nesse caso se baseia nesse princípio. O que será explicado na próxima seção.

#### 3.3.1.2 Implementação do Algoritmo

##### Visão Geral

Um fato de escala já visto na seção 1.1.3 mostra que um fator de escala é definido como um fator usado para expandir ou comprimir o espectro para ajustar as frequências do sinal de modo que esse deslocamento seja corretamente ajustado. Uma vez feito isso, nós poderemos reconstruir o sinal e voltar a sua duração inicial com o efeito *pitch-shifting* realizado.

De fato, se queremos deslocar o sinal um semitom, será preciso utilizar um fator de escala igual a  $2^{1/12}$  o qual corresponde a 1.0594 (figura 3.3). Isto é, precisamos esticar o sinal sem alterar seu espectro de frequência a qual a duração é agora multiplicado por 1.0594. Uma vez feito isso, somente será preciso tocar o sinal 1.0594 vezes mais rápido (DEMERS FRANCOIS GRONDIN, 2009).

Uma das etapas de implementação do algoritmo é a separação do sinal original em uma grande quantidade de *frames* (pedaços). Esses *frames* (figura 3.4) são obtidos do sinal original os quais são "sobrepostos" uns com os outros por um fator de 75% de sobreposição. Esse *frames* estarão então espaçados dependendo se o sinal, no processamento, será esticado ou comprimido no tempo na sua saída.

No entanto, a medida que essa operação é realizada teremos um problema muito comum, pois o espaço entre os *frames* são feitos em determinados trechos do sinal inadvertidamente. Nesse caso, é criado diversas discontinuidades. Isso é mostrado na figura 3.5 que mostra os espaços entre os frames é dado como tamanho  $x$  e torna-lo tamanho  $y$ .

Esse procedimento produz "defeitos" no som (espaços indesejados) que são perceptíveis ao

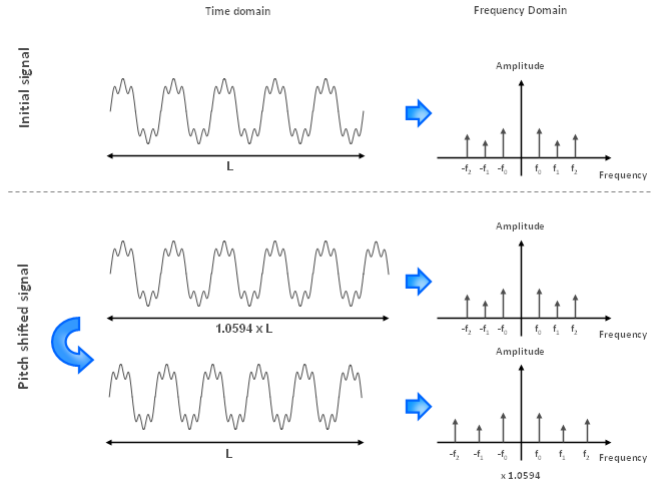


Figura 3.3: Exemplo de arbodagem do pitch-shifter - premissa básica para o efeito shimmer.

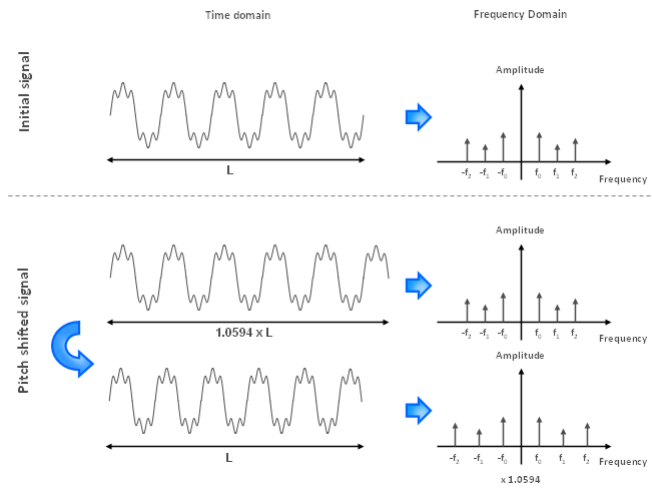


Figura 3.4: Expansão e compressão do sinal frame por frame.

ouvido humano. Desse modo é preciso de alguma forma fazer essa descontinuidade desaparecer. Por conta disso precisamos de uma técnica de processamento denominada de *phase vocoder*.

### Phase Vocoder

O *phase vocoder* é uma variação na Transformada Rápida de Fourier (FFT) que usa informações de fase para melhorar as estimativas de frequência (SETHARES, 1999). É ideal para uso em aplicações como o alongamento ou a compressão de tempo de áudio, embora existam vários outros efeitos especiais que podem ser implementados usando a estratégia de *phase vocoder*.

Além disso, o *phase vocoder* é geralmente apresentado como uma solução de alta qualidade para a modificação da escala de tempo (ou frequências), o pitch-shift é geralmente implementado como uma combinação de escala de tempo e conversão da taxa de amostragem (LAROCHE, 1999).

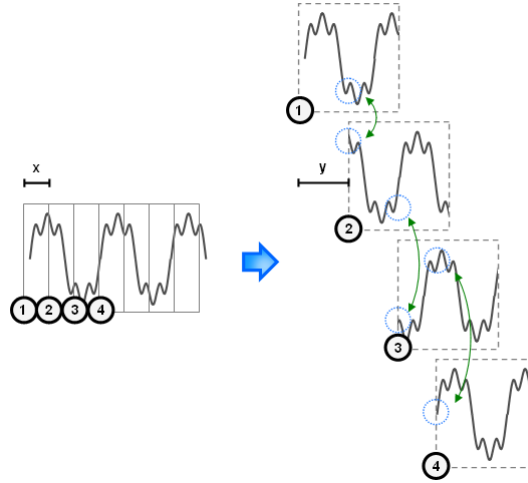
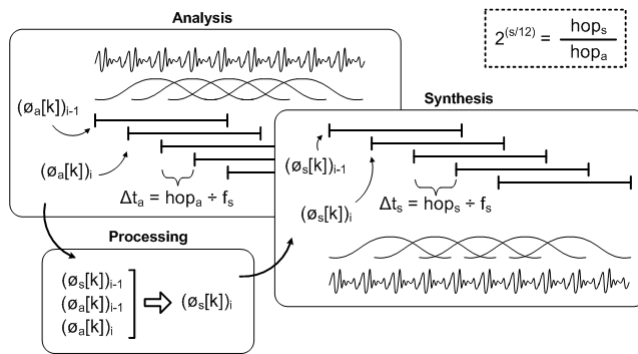
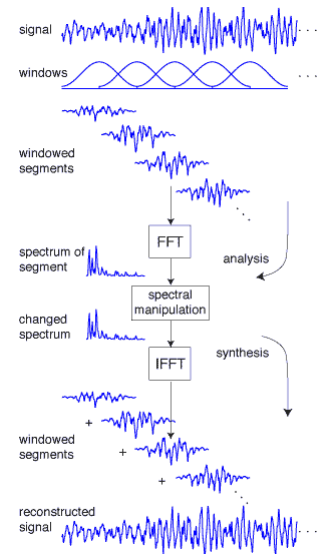


Figura 3.5: Sinal tornando-se descontínuo quando submetido ao processo de expansão ou compressão no tempo.

A figura 3.6 mostra como um algoritmo phase vocoder funciona tanto de maneira geral (figura 3.6(b)), tal como com a implementação do algoritmo escrito em MATLAB (figura 3.6(a)). Ele basicamente consiste em 3 (três) estágios: análise, processamento e síntese (PORTNOFF, 1976).



(a) Ilustração Específica das Etapas do algoritmo do *phase-vocoder*



(b) Ilustração Geral da Etapa do algoritmo *phase-vocoder*

Figura 3.6: Etapas do algoritmo do *phase-vocoder*

**Análise:** Como já mencionado nesse trabalho na seção 2.1.2 que um janelamento refere-se a falar de um pequeno *frame* de um sinal infinitamente maior (se considerarmos apenas 1 pequena janela comparado a todo o sinal amostrado). O processamento dessa janela no espectro do sinal pode ser minimizada. Em outras palavras, para reduzir o efeito de janelamento do sinal na sua representação no domínio da frequência, é utilizado uma janela do tipo *Hanning* ou *Blackman*

de tamanho  $N$ . O porquê disso? Bem, de acordo com a figura 3.7, uma multiplicação de uma janela retangular no domínio do tempo com o sinal é equivalente a uma convolução no domínio da frequência. Desse modo a janela retangular possui mais energia nos lóbulos laterais em comparação aos demais janelamentos, enquanto a janela *Hanning* ou *Blackman* tem um foco de concentração de sua energia espectral no seu lóbulo principal, ou seja, perto do nível DC do sinal.

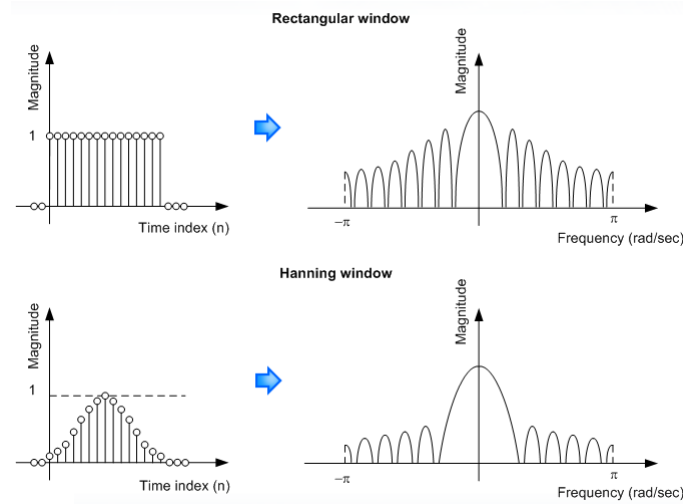


Figura 3.7: Janelamentos

Esse frame é, o qual é formado por  $N$  amostras, é então transformado através de uma Transformada Rápida de Fourier - *Fast Fourier Transform* (FFT) conforme a equação 3.6

$$(X_a[k])_i = \sum_{n=0}^{N-1} x[n + i \cdot (hop_a)] w[n] e^{-j(\frac{2\pi kn}{N})} \quad k = 0, 1, 2, \dots, N_1 \quad (3.6)$$

Na equação acima,  $x[n]$  é a amostra do sinal,  $w[n]$  representa o janelamento *Hanning* ou *BlackMan* e  $(X_a[k])_i$  representa o espectro discreto do *frame*  $i$ . Afim de aumentar a resolução do espectro, as janelas são sobrepostas com um fator de 75%. O número de amostras entre duas sucessivas janelas é referenciado pelo "número de saltos" entre cada janela - termo *hop size* ( $hop_a$ ) e é igual a  $\frac{N}{4}$  de uma sobreposição de 75%.

**Processamento:** Já é sabido que aplicando a FFT num sinal de tamanho  $N$  resulta em  $N$  segmentos espectrais que começam do valor 0 até o valor  $\frac{(N-1)}{N} f_s$  com um intervalo de  $\frac{f_s}{N}$ , onde  $f_s$  é a nossa frequência de amostragem do sinal. Em termos de módulo, um sinal que possua uma frequência intermediária entre os dois segmentos espectrais terá seu valor não computado e a energia contida nessa frequência, distribuída entre os dois segmentos vizinhos. A informação de fase do sinal é usado para melhorar a acurácia do estimado dessas frequências entre cada segmento espectral. A figura 3.8 mostra duas ondas senoidais com uma pequena diferença em valores de frequência. Conforme o método, os frames do sinal são divididos dentro das  $N$  amostras. As janelas nesse caso não estão sobrepostas para facilitar a explicação. O primeiro sinal tem



a frequência de  $\frac{f_s}{N}$  e ela, de fato, cai exatamente na primeira frequência do primeiro segmento espectral. A segunda senoide, por sua vez, tem uma frequência levemente maior que o primeiro sinal. Observe que a amostra não está centrado no primeiro segmento, embora sua energia esteja principalmente no dentro dele.

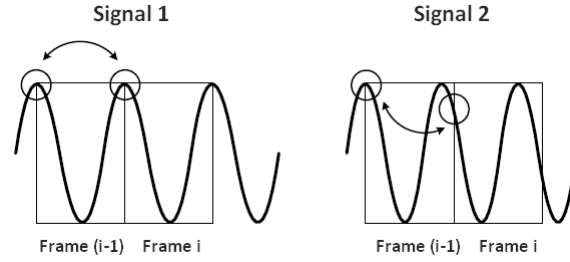


Figura 3.8: Ondas senoidais com pequena diferença na frequência e fase.

A informação de fase de dois *frames* sucessivos é significativamente relevante. Pois o primeiro sinal, não há diferença de fases entre os dois frames. Por outro lado, para o segundo sinal, a fase do primeiro segmento é maior que zero. Isso implica que a componente do sinal que correspondente para este segmento é maior que a frequência desse segmento. A diferença de fase entre dois frames é referenciado no código como *phase-shift*  $(\Delta\phi_a[k])_i$  no intervalo de  $-\pi$  a  $\pi$ . Sem a sobreposição, a frequência real  $(\omega_{true}[k])_i$  pode ser obtida de um deslocamento de fase  $(\Delta\phi_a[k])_i$  e um intervalo de tempo  $\Delta t_a$  entre dois *frames* conforme visto na equação 3.7. O intervalo de tempo  $\Delta t_a$  é o número de saltos  $hop_a$ , já visto no págrfo de análise, dividido pela frequência de amostragem  $f_s$ .

$$(\omega_{true}[k])_i = \frac{(\Delta\phi_a[k])_i}{\Delta t_a} \quad (3.7)$$

A equação supracitada é menos complexa pois trata-se de um sinal não sobreposto. No caso mais complexo, o desvio de frequência para o primeiro segmento é calculado e então sobreposto. Essa quantidade é adicionada ao segmento de frequência para obter a real frequência daquele componente dentro do *frame*. As equações 3.8, 3.9 e 3.10 ilustram esse procedimento. A variável  $(\phi_a[k])_{i-1}$  e  $(\phi_a[k])_i$  correspondem a fase do frame anterior e o frame presente respectivamente. Também,  $\omega_{bin}[k]$  corresponde a frequência do segmento,  $(\Delta\omega[k])_i$  o desvio de frequência e  $(\Delta\omega_{wrapped}[k])_i$  corresponde o desvio de frequência sobreposta (*wrap-frequency*).

$$(\Delta\omega[k])_i = \frac{(\phi_a[k])_i - (\phi_a[k])_{i-1}}{\Delta t_a} - \omega_{bin}[k] \quad (3.8)$$

$$(\Delta\omega_{wrapped}[k])_i = \text{mod} [((\Delta\omega[k])_i + \pi), 2\pi] - \pi \quad (3.9)$$

$$(\omega_{true}[k])_i = \omega_{bin}[k] + (\Delta\omega_{wrapped}[k])_i \quad (3.10)$$

A nova fase de cada fragmento pode ser então calculada adicionando a fase deslocada requerida para assim evitar-se a descontinuidade dos segmentos. Isso é feito multiplicando a frequência real com o intervalo de tempo do estágio de síntese, conforme mostrado na equação 3.11:

$$(\phi_s[k])_i = (\phi_s[k])_{i-1} + \Delta_{t_s} \cdot (\omega_{true}[k])_i \quad (3.11)$$

A fase do frame anterior, por síntese, já é conhecida desde que ela á foi calculada pelo algoritmo de recursividade. Por fim, o novo espectro é então obtido conforme a equação 3.12:

$$|(X_s[k])_i| = |(X_a[k])_i| \quad \angle(X_s[k])_i = (\phi_s[k])_i \quad (3.12)$$

**Síntese:** Agora, como já gerenciado o ajuste de fase no domínio da frequência para as sequências de frames, será preciso agora retornar ao domínio do tempo. Para isso aplicamos a Transformada Inversa Discreta de Fouriter (IDFT) para cada frame do espectro. O resultado é então um janelamento com a janela apropriada (*Hanning* ou *Blackman*) obtendo  $q_i[n]$ . O janelamento é usado dessa vez para suavizar o sinal. Esse processo é descrito na equação 3.13:

$$q_i[n] = \left( \frac{1}{N} \sum_{k=0}^{N-1} (X_s[k])_i e^{-j \frac{2\pi kn}{N}} \right) w[n] \quad n = 0, 1, 2, \dots, N-1 \quad (3.13)$$

Cada *frame* é estão sobreposto conforme mostra a equação 3.14. A variável  $L$  corresponde ao número de *frames* e  $u[n]$  representa a função degrau unitário.

$$y[n] = \sum_{i=0}^{L-1} q_i[n - i \cdot hop_s] \{u[n - i \cdot hop_s] - u[n - i \cdot hop_s - L]\} \quad (3.14)$$

### 3.3.1.3 Reamostragem do Sinal

Agora que temos o nosso sinal sem descontinuidades podemos agora estica-lo ou comprimi-lo no tempo e suas componentes de frequências não serão alteradas. Agora, podemos re-amostrar o sinal e volta-lo para a duração inicial e então deslocar a sua frequência. Supondo que dado uma taxa de amostragem queremos dobrar a frequência - 1pitch de 1 oitava. A maneira mais fácil de fazer é escolher apenas uma amostra de duas e produzir o resultado. Isso é fácil, porque quando dobramos a frequência, lidamos com um fator de escala que é um número inteiro. Para fatores de escala não inteiros (1 semitom por exemplo - 1.0594) é usado uma interpolação linear para aproximar a amostra que deveria estar naquele local.

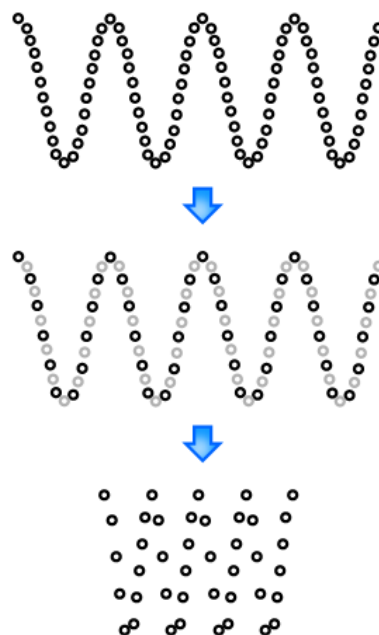


Figura 3.9: *Reamostragem do Sinal utilizando duas vezes o valor da amostra anterior*

### 3.4 BLOCO 3 - DELAY TIME - REVERB EM CONVOLUÇÃO

A onda sonora emitida pelo instrumento se espalha no ambiente. primeiro, chega diretamente ao ouvinte. mas segue seu caminho, para várias direções, e atinge superfícies. reflete. para outras direções. passa de novo pelo ouvinte. continua adiante, reflete em outras superfícies, e em outras mais, atinge o ouvinte. Como já mencionado na seção (), num curto espaço de tempo, o campo sonoro dentro da sala estará um verdadeiro caos. Milhares de ondas sonoras, de diferentes intensidades e fases, seguem o som original da fonte sonora, criando o que chamamos de reverberação.

O fato é que o som final (resultante do som direto, das reflexões primárias, fortes e espaçadas e do campo reverberante que soa como uma cauda), depende totalmente da posição da fonte sonora, da posição do ouvinte (ou do microfone), da geometria da sala, de seu tamanho e dos revestimentos existentes. Cada sala tem o seu próprio tipo de reverb. Para efeitos deste texto (mesmo porque cada literatura trata o assunto de uma maneira diferente), chamaremos de *Reverb* todo o som que segue o original. Portanto, o conjunto de reflexões primárias e tardias, que dependem das características acústicas da sala.

Em outras palavras, podemos dizer que toda e qualquer sala possui uma “impressão digital”, única, característica, que determina como o som se comportará dentro dela. Algumas impressões digitais são bem conhecidas, sendo que a grande maioria das pessoas identifique facilmente se um som foi gravado ou está sendo gerado dentro de um ginásio de esportes, ou de uma catedral, cavernas, banheiros, salas de concerto, etc.

A grande vantagem nesse caso de usar o reverb de convolução é justamente utilizar uma forma

de resposta impulsional daquele ambiente e depois "filtrar"(convoluir) o sinal original com essa resposta, muito semelhante ao que é feito com o filtro de resposta finita.

## 3.5 BLOCO 5 - CONVERSÃO D/A - COMUNICAÇÃO I<sup>2</sup>C E MCP4725

### 3.5.1 Limitações do MSP430F5529LP

Dentro do contexto do projeto de avaliação de desempenho de um microcontrolador no processamento de sinais digitais de áudio, o modelo em questão não possui um conversor Digital Analógico Integrado. Dessa forma é necessário um módulo externo responsável para a conversão das amostras digitais do  $\mu C$ .

### 3.5.2 Características do *Hardware* e Operação

A interface I<sup>2</sup>C implementada nos MSP430 possui as seguintes características, dentre outras:

- Transferencia de *byte* e *word*;
- Suporte de endereçamento de 7 a 10 bits;
- Velocidade de 100 a 400 Kbps;

A operação da interface *USART* no modo I<sup>2</sup>C é selecionada pelos *bits* UxCTL:SYNC = 1 e UxCTL:I2C = 1. O módulo do MSP430F5529 pode funcionar em quatro modos de comunicação: transmissor mestre, receptor mestre, transmissor escravo e receptor escravo. No caso do projeto, precisamos apenas avaliar o desempenho do modo transmissor mestre pois apenas enviaremos os dados ao MCP4725.

A seleção entre a operação no modo mestre é feita pelo *bit* U0CTL:MST: que em nível "1" teremos o modo ativado. A seleção entre a transmissão é feita pelo *bit* I2CTCTL:TRX, a qual depende do modo de operação da I<sup>2</sup>C: Para o modo mestre e modo de transmissão teremos TRX=1.

Além disso, quando o módulo funciona como transmissor, existem ainda duas possibilidades de operação: a contagem automática do número de *bytes* transmitidos (*bit* I2CTCTL:I2CRM=0) ou controle manual dessa operação (*bit* I2CTCTL:I2CRM = 1). O controle automático do número de bytes transmitidos utiliza um registrador para armazenar a quantidade total de bytes a ser transmitida (excluindo-se o primeiro de controle). Um contador interno é inicializado com o valor desse registrador e decrementado a cada *byte* transmitido. Quando o contador chega a zero, a transmissão pode ser automaticamente finalizada por uma condição de parada.

Assim, os passos para realizar uma transmissão utilizando o modo mestre com contagem automática

## 4 SIMULAÇÕES E RESULTADOS

Neste capítulo, serão apresentados os resultados experimentais de todo o processo de obtenção do efeito de *Reverb Shimmer*, bem como alguns resultados de testes de pequenos algoritmos implementados no microcontrolador MSP430F5529 com seu conversor Analógico Digital de 12 bits e a comunicação I<sup>2</sup>C entre o  $\mu C$  e o dispositivo MCP 4725.

### 4.1 DESCRIÇÃO DOS EXPERIMENTOS

Os experimentos ora mostrados foram realizados de forma a demonstrar toda a abordagem de aprendizagem durante o projeto. Conforme será explicado na conclusão do trabalho, posto que foi identificado, através de experimento e cálculos realizados na implementação as limitações do hardware proposto para a execução do código do efeito digital em si.

Os resultados das simulações serão divididos da seguinte ordem:

1. Comparativo da quantização de um sinal de áudio com resoluções de 16 e 12 bits;
2. Comparativo da quantização de um sinal de áudio com resoluções de 12, 10 e 8 bits;
- 3.

Para os experimentos acima mencionados serão utilizados os seguintes dados mostrado na tabela 4.1:

Fonte de Áudio	'guitar-clean16.wav' - som de guitarra limpa
Taxa de Amostragem	44100 Hz
Número de Canais	1 canal mono

Tabela 4.1: Dados relativos as simulações no computador

Esses resultados das figuras são apenas a título de informações preliminares na escolha da resolução adequada para que o sinal seja tratado posteriormente no microcontrolador. Nesse caso, posto que o conversor em questão seja de 12 (doze) *bits* percebe-se que não há uma perda significativa na resolução do sinal, bem como audivelmente não se percebeu grande perda de qualidade em relação ao áudio original em 16 *bits*.

Essa resposta do filtro passa baixa (figura ??) com frequência de corte igual a 2kHz. Esse filtro foi utilizado o número de 51 *Taps*. Nota-se que a atenuação começa a partir da frequência  $f_{[\text{rad/samples}]} = f_{[\text{cycles/sec}]} \cdot \frac{\text{sec}}{\text{samples}} \cdot \frac{\text{rad}}{\text{cycle}} \Rightarrow f_{[\text{rad/samples}]} = f_{[\text{cycles/sec}]} \cdot \frac{2\pi}{f_s} \Rightarrow f = 44100 \cdot 0.2/2\pi = 1403.74\text{Hz}$ .

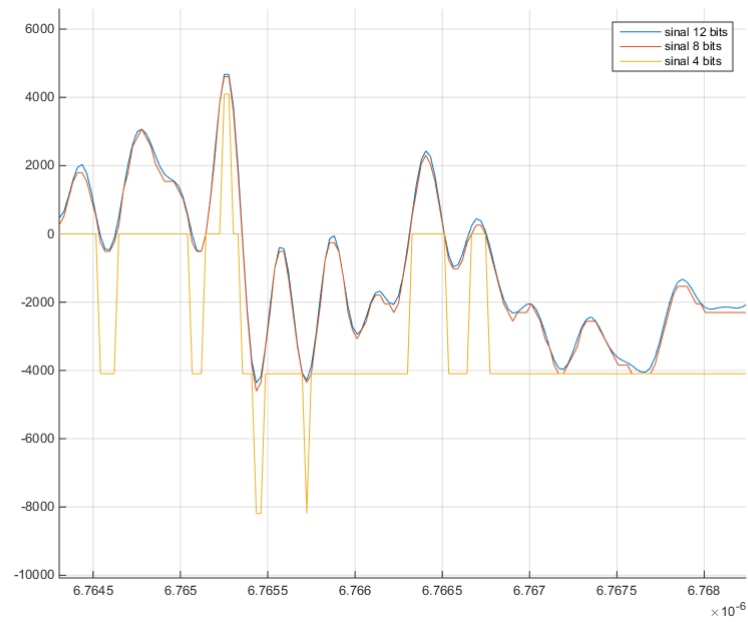


Figura 4.1: *Quantização do sinal de áudio em 12, 8 e 4 bits PCM*

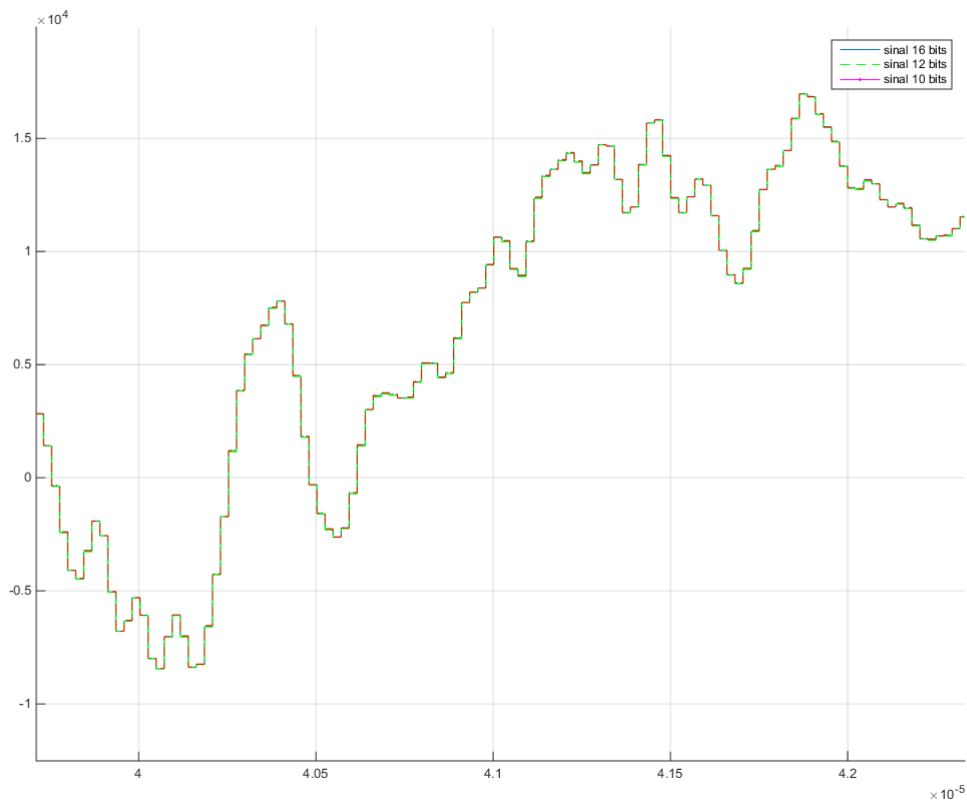


Figura 4.2: *Quantização do sinal de áudio em 10, 12 e 16 bits PCM*

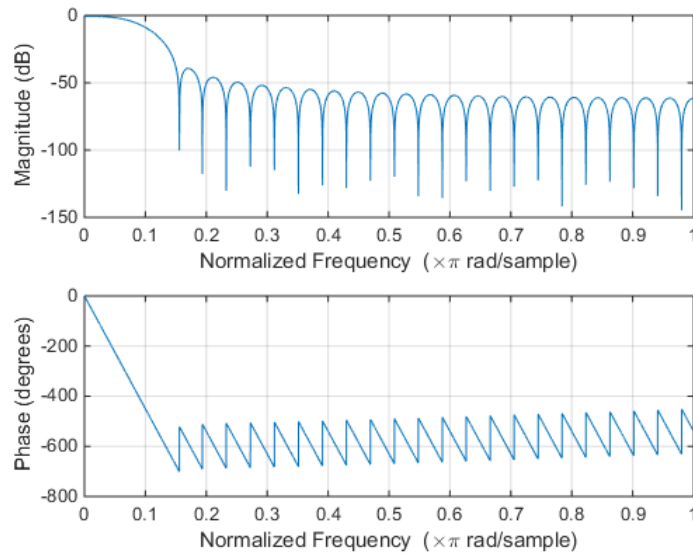


Figura 4.3: Resposta do filtro FIR-Passa-Baixa com janela Blackman

A seguir temos nas figuras 4.4(a) e 4.4(b) os áudios para serem utilizados no processo de convolução com o sinal original, originando o efeito de reverb em convolução.

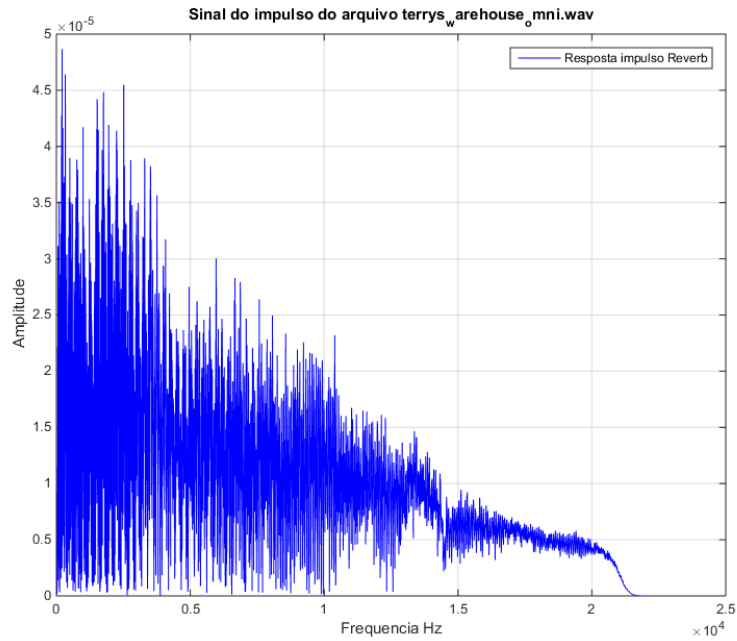
Resultado da resposta em frequência (figura 4.5) da aplicação do filtro FIR-passa-baixa no sinal original de 12 bits:

## 4.2 ANÁLISE DOS RESULTADOS

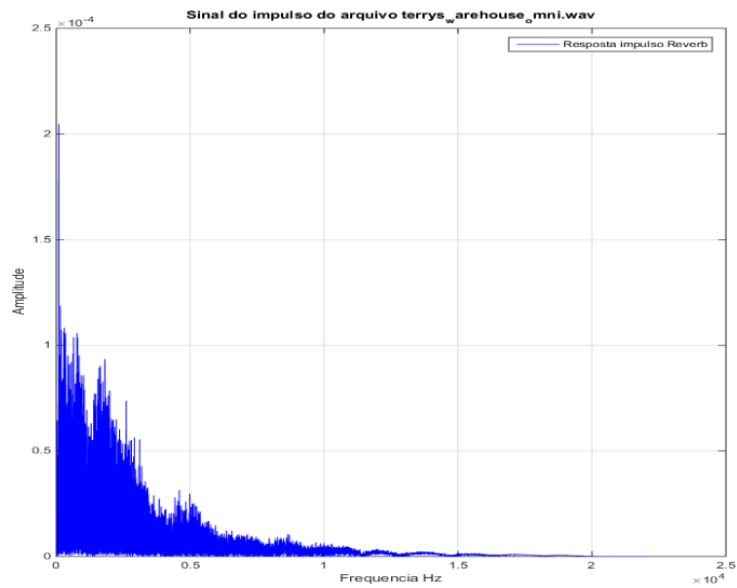
### 4.2.1 Do desempenho do Código MATLAB

Avaliando a figura 4.6 que demonstra o resultado do efeito de deslocamento seletivo em frequência (*pitch-shifter*) aplicado ao sinal filtrado observamos claramente para um deslocamento de 1 oitava (para a nota A3 = 220Hz), percebemos que a resposta em frequência bate exatamente no valor que é o dobro ( $2^{12/12}$ ) - A4 = 440Hz. Concluindo que o algoritmo de fato desloca em saltos de meio tom a frequência do sinal, em outras palavras, consegue alterar o *pitch* do sinal sem alterar sua duração temporal.

Não obstante, analisando o mesmo efeito só que agora com diversas notas tocadas (figura 4.7) ao mesmo tempo (harmonia) percebemos que o efeito tem certas diferenças em termos de amostragem das frequências que foram deslocadas do sinal. Isso é facilmente explicado e já mencionado no parágrafo 3.3.1.2 da seção que fala sobre *phase-vocoder*, que diz que a energia das frequências ora existentes entre dois segmentos de frequências (*bins*) consecutivos, terá seu valor distribuído entre esses dois segmentos.



(a) Espectro de frequência de uma resposta impulsional de um som oriundo do Palácio de *Falkland Palace in Fife*.



(b) Espectro de frequência de uma resposta impulsional de um som oriundo de uma *empty warehouse* (armazem vazio).

Figura 4.4: Respostas Impulsionais de Ambiente: Exemplos retirado do website: *OpenAir*.



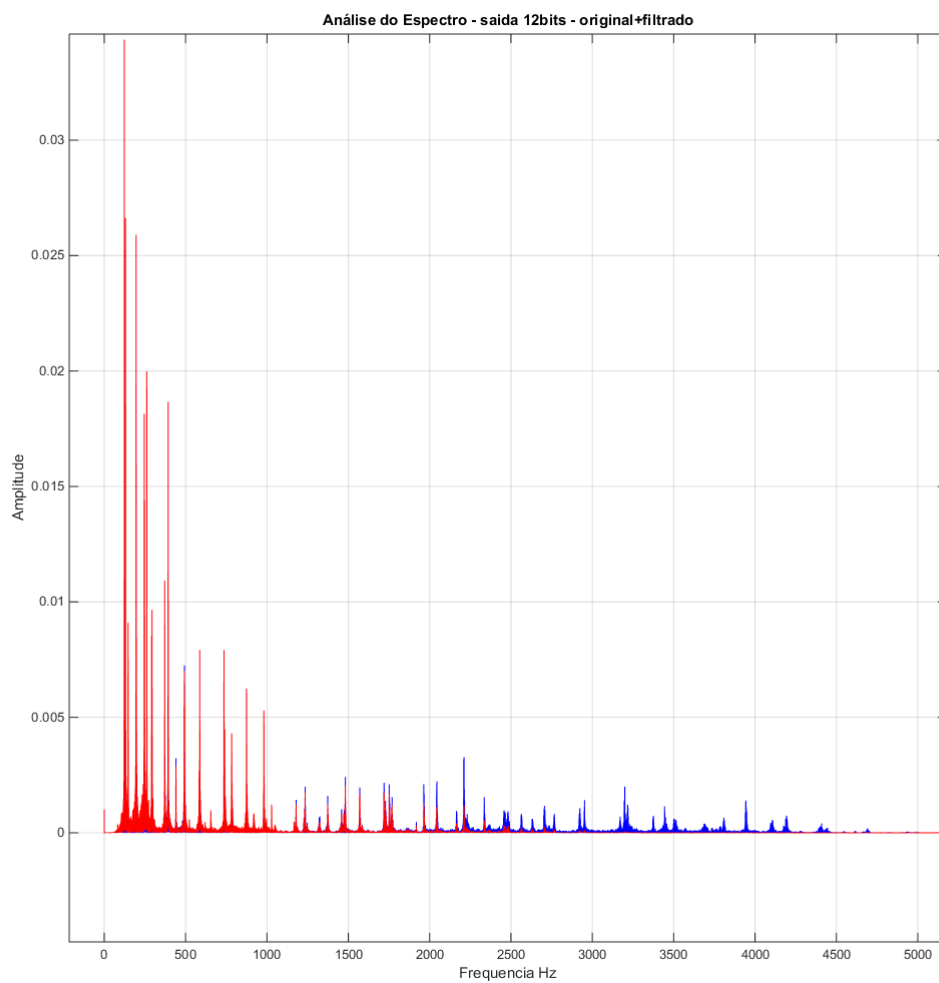


Figura 4.5: Espectro de frequência do sinal original 12 bits e o sinal filtrado pelo filtro casual FIR.

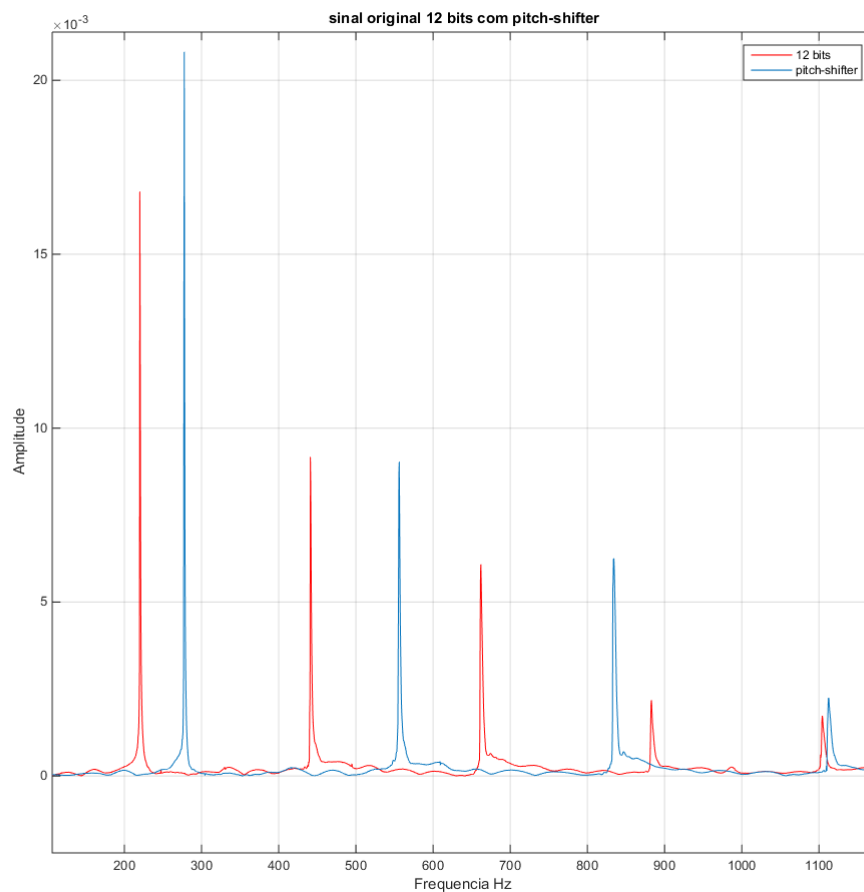


Figura 4.6: Análise espectral de um sinal puro de guitarra tocada na nota Lá -  $A2 = 220\text{Hz}$  e o efeito pitch-shifter utilizado deslocamento de 1 oitava.

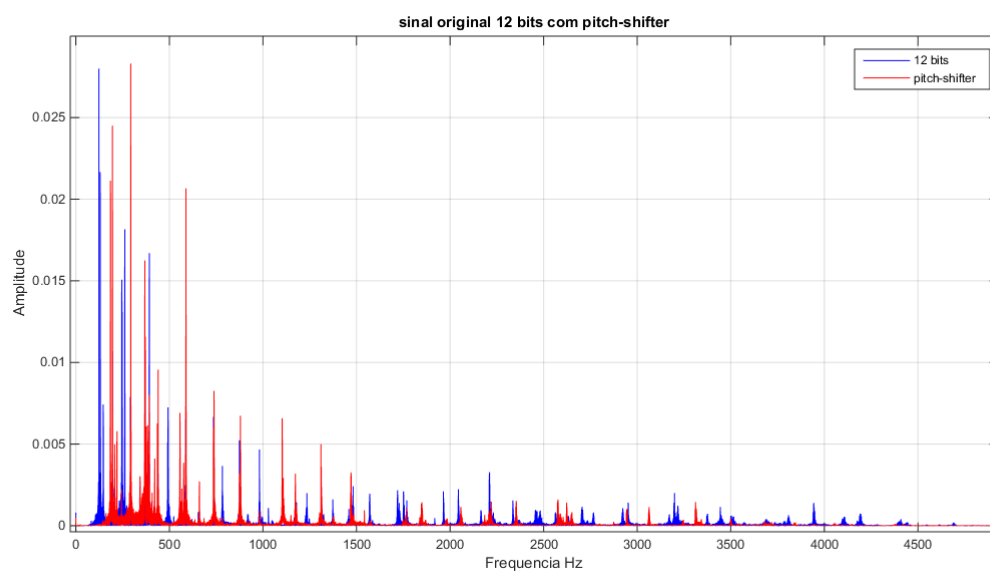


Figura 4.7: Análise espectral de um áudio de guitarra tocando um acorde dedilhado. Pitch-shifter utilizando deslocamento de 1 oitava.

# 5 CONCLUSÕES

## 5.1 REALIZAÇÃO DO PROJETO

O projeto foi executado dentro das limitações impostas de hardware do uC MSP430F5529LP no quesito de seu desempenho de conversão digital analógica, bem como suas interrupções, período de amostragem bem como sua integração com o conversor Digital-Analógico MCP4725 através da interface  $I^2C$ .

Não foram avaliados, no entanto, o desempenho do dispositivo na aplicação dos filtros digitais e o modelo ora projetado no *software* Matlab que demonstrou o efeito de *reverb-shimmer* em si. Muito embora, essa parte do projeto estivesse dentro do escopo inicial do trabalho, a sua realização se daria com melhores resultados com a utilização de um DSP específico para esta aplicação, nesse caso a utilização do TMS320F2837xS Delfino<sup>TM</sup>.

Foram constatados que o efeito *reverb-shimmer* possui uma grande complexidade em termos de estabilidade, pois, conforme observado nas simulações, a malha de realimentação com o bloco de *delay's* aleatórios podem deixar o sistema instável e com isso efeitos indesejados no resultado final.

Dentro dessa realidade podemos dizer que o objetivo do trabalho foi alcançado com respeito ao entendimento claro do modelo matemático e suas implicações na escolha de se projetar um efeito seletivo em frequência, entender suas limitações dentro do contexto de filtros digitais, seu critério de estabilidade e consubstanciar elementos necessários para que seja aplicado dentro de um contexto de projeto de *hardware*.

## 5.2 AUTO-CRÍTICA

Considerando a seção 5.1, vale destacar, por ora, que a realização em hardware foi a maior dificuldade encontrada dentro do projeto por conta das escolhas dos parâmetros limitantes, tais como:

1. Resolução da conversão - 12 bits;
2. Período de Amostragem do *Timer*;
3. Imprecisão nas amostras coletadas;
4. Impossibilidade de trabalhar com ponto flutuante.

Diante disso foi necessário concluir o trabalho apenas entendendo as limitações de hardware e escolhendo seus parâmetros baseado nos conceitos de processamento digitais de sinais e não apenas em testes cegos de desempenho. Por outro lado, não se avaliou o desempenho com os códigos ora projetados no MATLAB para o uC em questão, mesmo este sendo o modelo mais modesto.

### **5.3 MODELOS PARA TRABALHOS FUTUROS**

# REFERÊNCIAS BIBLIOGRÁFICAS

ALBAR, N. *Tutorial: Efeito de Guitarras*. 2007. Cifraclub. Disponível em: <<https://www.cifraclub.com.br/aprenda/guitarra/tutoriais/81/>>.

BRAGA, N. C. *Curso de Eletrônica - Eletrônica Digital - 2*. [S.l.]: Instituto Newton C. Braga, 2012.

CHAKRABORTY, S. Advantages of blackman window over hamming window method for designing fir filter. *International Journal of Computer Science & Engineering Technology (IJCSET)*, 2013. ISSN 2229-3345.

DAVIES, J. H. *MSP430 Microcontroller Basics*. NEWNES, 2008. ISBN 0750682760. Disponível em: <[http://www.ebook.de/de/product/7025920/john\\_h\\_davies\\_msp430\\_microcontroller\\_basics.html](http://www.ebook.de/de/product/7025920/john_h_davies_msp430_microcontroller_basics.html)>.

DEMERS FRANCOIS GRONDIN, A. V. L. *Guitar Pitch Shifter*. 2009. Disponível em: <<http://www.guitarpitchshifter.com>>.

HAYKIN, S.; VEEN, B. V. *Sinais E Sistemas*. Bookman, 2001. ISBN 9788573077414. Disponível em: <<https://books.google.com.br/books?id=tdNYclZwaYIC>>.

III, J. O. S. *Physical Audio Signal Processing for Virtual Musical Instruments and Digital Audio Effects*. [S.l.]: W3K Publishing, 2010. ISBN 0974560723.

KUNDERT, A. *WinFilter: The easiest way to design a Digital Filter*. 2004. Disponível em: <<http://http://www.winfilter.20m.com/>>.

LAROCHE, M. D. J. New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1999.

LYNN, P. A.; FUERST, W. *Introductory Digital Signal Processing with Computer Applications with 3.5 Disk*. 2nd. ed. New York, NY, USA: John Wiley & Sons, Inc., 1998. ISBN 0471976318.

OPPENHEIN, R. W. S. A. V. *Discrete-Time Signal Processing*. [S.l.]: Prentice-Hall, 1998.

PORTNOFF, M. R. Implementation of the digital phase vocoder using the fast fourier transform. *IEEE TRANSACTIONS ON ACUSTICS, SPEECH, AND SIGNAL PROCESSING*, 1976.

ROBERTS, C. T. M. R. A. *Digital Signal Processing*. [S.l.]: Addison-Wesley Company, 1987.

SETHARES, W. A. *A Phase Vocoder in Matlab*. 1999. Disponível em: <<http://sethares.engr.wisc.edu/vocoders/phasevocoder.html>>.

## APÊNDICES

## I.1 CÓDIGOS EM MATLAB RELACIONADOS AO MÉTODO PROPOSTO

### Código MATLAB responsável pela realização do efeito shimmer numa amostra de Áudio

Listing 1: Code: "audio.m":Código Matlab para a realização da análise espectral do efeito shimmer em um sinal de áudio de 12bits/44100Hz

```
1      clc;
2      clear all;
3      info = audioinfo('guitar.wav')
4      [y,Fs] = audioread('guitar.wav','native');
5      voiceL = y(1:1000000,1);
6      %voiceR = y(:,2);
7
8      for j = 1:15
9          voiceL = [voiceL, bitshift(voiceL(:,j),-1)]; % shift once to ...
              the right
10         % voiceR = [voiceR, bitshift(voiceR(:,j),-1)];
11     end
12
13     voice = [voiceL];%todas as taxas de bits por amostras de ...
              16bits ate 1
14     %voice = voice/max(max(abs(voice)));
15
16     output12bits = 2\^4.*voice(:,5);%coluna 5 da esquerda 12bits e
17     %coluna 20 do audio direito tbm
18     %12bits
19     output10bits = 2\^6.*voice(:,7);
20     output8bits = 2\^8.*voice(:,9);
21     output4bits = 2\^12.*voice(:,13);
22
23     audiowrite('guitar12bits.wav',output12bits,info.SampleRate);
24     audiowrite('guitar10bits.wav',output10bits,info.SampleRate);
25     audiowrite('guitar8bits.wav',output8bits,info.SampleRate);
26     audiowrite('guitar4bits.wav',output4bits,info.SampleRate);
27
28     info = audioinfo('guitar12bits.wav')
29     info = audioinfo('guitar10bits.wav')
30     info = audioinfo('guitar8bits.wav')
31     info = audioinfo('guitar4bits.wav')
32
33
34     %Amostrando os sinais no tempo
35     figure(1)
36     hold on
```



```

37     t = linspace(0,1/info.SampleRate,length(voice));
38     plot(t,voice(:,1));%16 bits
39     %plot(t,output12bits);%12 bits
40     %plot(t,output10bits);%10 bits
41     plot(t,output8bits);%8 bits
42     %plot(t,output4bits);%4 bits
43     legend('sinal 16 bits','sinal 8 bits');
44     hold off;
45
46
47     % comparacao valores maximos
48     Max12bits = max(2\^4.*voice(:,5))
49     Max16bits = max(voice(:,1))
50
51     %amostrando o sinal no dominio da freq
52     output12bits = double(output12bits);
53     Y1 =      fft(output12bits);
54     N = Fs;
55     transform = fft(output12bits,N)/N;
56     magtransform = abs(transform)/abs(max(abs(transform)));
57     num_bits = length(magtransform);
58     plot([0:1/(num_bits/2-1):1],magtransform(1:num_bits/2))
59     % faxis = linspace(-Fs/2,Fs/2,N);
60     % figure()
61     % plot(faxis,magtransform);
62     xlabel('frequency(Hz)')
63
64     %projetar o filtro
65     [b,a] = butter(8,0.1,'low');
66     H = freqz(b,a,floor(num_bits/2));
67     hold on
68     figure()
69     plot([0:1/(num_bits/2 -1):1], abs(H), 'r');
70     hold off
71     figure()
72     output12bits_filtrado = filter(b,a,output12bits);
73     plot(output12bits, 'b')
74     hold on
75     plot(output12bits_filtrado, 'r')
76     %normalizando
77     output12bits_filtrado = ...
        output12bits_filtrado/max(abs(output12bits_filtrado));
78     legend('audio 12 bits','audio 12bits filtrado');
79     audiowrite('guitar12bitsfiltrado.wav',output12bits_filtrado,info.SampleRate);
80
81     hold off
82     %oitavador

```

```

83     guitar_oitavado = pitchShift(output12bits,1024,256,2);
84     guitar_oitavado_filter = filter(b,a,guitar_oitavado);
85
86     %pequeno delay
87     leftout=output12bits; % set up a new array, same size as old one
88
89     N=100; % delay amount N/44100 seconds
90
91     for n=N+1:length(guitar_oitavado_filter)
92
93         leftout(n)=output12bits(n)' + guitar_oitavado_filter(n-N); % ...
            approximately 1/4 second echo
94     end
95     output12bits = double(output12bits);
96     Y1 =      fft(output12bits);
97     N = Fs;
98     transform = fft(output12bits,N)/N;
99     magtransform1 = abs(transform)/abs(max(abs(transform)));
100    num_bits = length(magtransform1);
101    plot([0:1/(num_bits/2-1):1],magtransform1(1:num_bits/2))
102    % faxis = linspace(-Fs/2,Fs/2,N);
103    % figure()
104    % plot(faxis,magtransform);
105    hold on
106    leftout = double(leftout);
107    Y1 =      fft(leftout);
108    N = Fs;
109    transform = fft(leftout,N)/N;
110    magtransform2 = abs(transform)/abs(max(abs(transform)));
111    num_bits = length(magtransform2);
112    plot([0:1/(num_bits/2-1):1],magtransform2(1:num_bits/2))
113    % faxis = linspace(-Fs/2,Fs/2,N);
114    % figure()
115    % plot(faxis,magtransform);
116    xlabel('frequency (Hz) ')
117    figure()
118    plot(leftout)
119    hold on
120    plot(output12bits)
121    legend('sinal de 12 bits','sinal com pitchshift+delay')
122    leftout = leftout/max(abs(leftout));
123    output12bits = output12bits/max(abs(output12bits));
124
125    audiowrite('shimmerA.wav',leftout,44100)

```

**Cálculo do Conversor DAC de 12 bits. (amostragem Linear) Code: "mcp4725.c":**

```

1      #include <msp430.h>
2      #include <stdint.h>
3      #include "mcp4725.h"
4      #include "lib/lcd/lcd.h"
5      #include "lib/dma/dma.h"
6      #include "lib/port/port.h"
7      #include "lib/clock/clock.h"
8      #include "lib/adc12/adc12.h"
9      #include "lib/timers/timer.h"
10     #include "lib/serial/serial.h"
11
12     #define MCP4725 0x62
13     #define mcpON      0
14     #define mcpOFF1K   1
15     #define mcpOFF100K 2
16     #define mcpOFF500K 3
17
18     uint16_t adcResult;
19     uint16_t data = 0, write;
20
21     int main(void) {
22         watchDogStop();
23         portInit();
24
25         lcdInit();
26         lcdClear();
27
28         clockInit();
29         clockSetDCO(1000000);
30         clockSelect(DCO, SMCLK);
31         clockSelect(DCO, MCLK);
32
33         adc12Init();
34         portRoute2Perif(P6, 0);
35
36         //timerSetup(B0, ACLK, UP, 3276, 1000);
37         TB0CTL = TBSSEL__ACLK |           // Select ACLK as ...
               clock source
38         MC__UP      |           // Setup but do not count
39         TBCLR;       // Clear timer
40
41         TB0CCR0 = 327;           // Convert every 100ms
42         TB0CCR1 = 100;          // This can be anything
43         TB0CCTL0 = CCIE;
44         TB0CCTL1 = OUTMOD_3;     // Set/reset
45
46         dmaEnable(0);

```

```

47     dmaTrgr(0,DMA_ADC12IFGx);
48     dmaAddr(0,&ADC12MEM0,DMA_FIXED,&adcResult,DMA_FIXED);
49     dmaMode(0,DMA_RPT_SINGLE_TRANSFER);
50     dmaSize(0,1);
51
52     serialInit(I2C);
53
54     __enable_interrupt();
55
56     //mcpWrite(0x2FF);
57
58     volatile uint16_t recData[4096];
59     uint16_t index = 4096;
60
61     while(index) {
62         while(!write);
63         recData[--index] = adcResult;
64         mcpWrite(index);
65         write = 0;
66     }
67     while(1);
68 }
69
70 void mcpWrite(uint16_t data) {
71     uint8_t vector[2];
72     vector[0] = (data >> 8) & 0x0F;
73     vector[1] = (data      ) & 0xFF;
74     serialI2CSendData(MCP4725, vector, 2);
75 }
76
77 #pragma vector=TIMER0_B0_VECTOR
78 __interrupt void isr_tb0_ccr0 () {
79     write = 1;
80     adcDisable();
81     adcEnable();
82 }

```

**Código Auxiliar responsável pelo Efeito Shimmer. Há duas funções aninhadas necessárias que serão inseridas no final do código.**

Listing 2: Code: "pitchshifting.m": Código Matlab da função auxiliar responsável pelo pitch-shifter

```

1
2
3     function [outputVector] = pitchShift(inputVector, windowSize, ...
4         hopSize, step)

```

```

5      %% Parameters
6
7      % Window size
8      winSize = windowSize;
9      % Space between windows
10     hop = hopSize;
11     % Pitch scaling factor
12     alpha = 2^(step/12);
13
14     % Intermediate constants
15     hopOut = round(alpha*hop);
16
17     % Hanning window for overlap-add
18     wn = hann(winSize*2+1);
19     wn = wn(2:2:end);
20
21     %% Source file
22
23     x = inputVector;
24
25     % Rotate if needed
26     if size(x,1) < size(x,2)
27
28         x = transpose(x);
29
30     end
31
32     x = [zeros(hop*3,1) ; x];
33
34     %% Initialization
35
36     % Create a frame matrix for the current input
37     [y,numberFramesInput] = createFrames(x,hop,winSize);
38
39     % Create a frame matrix to receive processed frames
40     numberFramesOutput = numberFramesInput;
41     outputy = zeros(numberFramesOutput,winSize);
42
43     % Initialize cumulative phase
44     phaseCumulative = 0;
45
46     % Initialize previous frame phase
47     previousPhase = 0;
48
49     for index=1:numberFramesInput
50
51     %% Analysis

```

```

52
53 % Get current frame to be processed
54 currentFrame = y(index,:);
55
56 % Window the frame
57 currentFrameWindowed = currentFrame .* wn' / ...
    sqrt(((winSize/hop)/2));
58
59 % Get the FFT
60 currentFrameWindowedFFT = fft(currentFrameWindowed);
61
62 % Get the magnitude
63 magFrame = abs(currentFrameWindowedFFT);
64
65 % Get the angle
66 phaseFrame = angle(currentFrameWindowedFFT);
67
68 %% Processing
69
70 % Get the phase difference
71 ΔPhi = phaseFrame - previousPhase;
72 previousPhase = phaseFrame;
73
74 % Remove the expected phase difference
75 ΔPhiPrime = ΔPhi - hop * 2*pi*(0:(winSize-1))/winSize;
76
77 % Map to -pi/pi range
78 ΔPhiPrimeMod = mod(ΔPhiPrime+pi, 2*pi) - pi;
79
80 % Get the true frequency
81 trueFreq = 2*pi*(0:(winSize-1))/winSize + ΔPhiPrimeMod/hop;
82
83 % Get the final phase
84 phaseCumulative = phaseCumulative + hopOut * trueFreq;
85
86 % Remove the 60 Hz noise. This is not done for now but could be
87 % achieved by setting some bins to zero.
88
89 %% Synthesis
90
91 % Get the magnitude
92 outputMag = magFrame;
93
94 % Produce output frame
95 outputFrame = real(ifft(outputMag .* exp(j*phaseCumulative)));
96
97 % Save frame that has been processed

```

```

98     outputy(index,:) = outputFrame .* wn' / ...
          sqrt(((winSize/hopOut)/2));
99
100    end
101
102    %% Finalize
103
104    % Overlap add in a vector
105    outputTimeStretched = fusionFrames(outputy,hopOut);
106
107    % Resample with linear interpolation
108    outputTime = interp1((0:(length(outputTimeStretched)-1)),
109    outputTimeStretched,
110    (0:alpha:(length(outputTimeStretched)-1)),'linear');
111
112    % Return the result
113    outputVector = outputTime;
114
115    return
116
117    function [vectorFrames,numberSlices] = ...
          createFrames(x,hop>windowSize)
118
119    % Find the max number of slices that can be obtained
120    numberSlices = floor((length(x)-windowSize)/hop);
121
122    % Truncate if needed to get only a integer number of hop
123    x = x(1:(numberSlices*hop>windowSize));
124
125    % Create a matrix with time slices
126    vectorFrames = zeros(floor(length(x)/hop),windowSize);
127
128    % Fill the matrix
129    for index = 1:numberSlices
130
131        indexTimeStart = (index-1)*hop + 1;
132        indexTimeEnd = (index-1)*hop + windowSize;
133
134        vectorFrames(index,:) = x(indexTimeStart: indexTimeEnd);
135
136    end
137
138    return
139
140
141    function vectorTime = fusionFrames(framesMatrix, hop)
142

```

```

143     sizeMatrix = size(framesMatrix);
144
145     % Get the number of frames
146     numberFrames = sizeMatrix(1);
147
148     % Get the size of each frame
149     sizeFrames = sizeMatrix(2);
150
151     % Define an empty vector to receive result
152     vectorTime = zeros(numberFrames*hop-hop+sizeFrames,1);
153
154     timeIndex = 1;
155
156     % Loop for each frame and overlap-add
157     for index=1:numberFrames
158
159         vectorTime(timeIndex:timeIndex+sizeFrames-1) =
160         vectorTime(timeIndex:timeIndex+sizeFrames-1) + ...
            framesMatrix(index,:);
161
162         timeIndex = timeIndex + hop;
163
164     end
165
166     return

```