UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM SOFTWARE PARA O RECONHECIMENTO DE NOTAS MUSICAIS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ADRIANO LUIZ MORETTI

BLUMENAU, JUNHO/2003

2003/1-1

PROTÓTIPO DE UM SOFTWARE PARA O RECONHECIMENTO DE NOTAS MUSICAIS

ADRIANO LUIZ MORETTI

ESTE TRABALHO DE CONCLUSÃO DE CURSO FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Paulo César Rodacki Gomes — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Paulo César Rodacki Gomes

_____

Prof. Miguel A. Wisintainer

_____

Prof. Cláudio Loesch

AGRADECIMENTOS

Gostaria de agradecer aos meus pais Luiz Conrado Moretti e Ana Fagundes Moretti pelo carinho que sempre tiveram por mim pelo incentivo ao estudo, vocês foram fundamentais para a realização deste trabalho. Gostaria também de agradecer aos meus cunhados Carlos Alberto Marcos e Luiz Alberto Bertotti. Não poderia esquecer de citar minha sogra Edviges Matilde Valle Raimundo juntamente com os meus sobrinhos Carlos Gabriel, Giovanna e Luiz Eduardo que são pessoas muito especiais para mim.

Gostaria de fazer um agradecimento todo especial a minha esposa, amiga e companheira Simone Raimundo Moretti que, com paciência, me auxiliou e acompanhou nas horas mais difíceis.

Agradeço também ao Prof. Paulo César Rodacki Gomes, me orientou com sabedoria neste trabalho de conclusão de curso e ao Prof. Cláudio Loesch pelo auxílio prestado para a realização deste trabalho. Agradeço também a todos os demais professores e funcionários que prestaram auxílio direto ou indireto para a realização deste trabalho.

Eu gostaria também de agradecer os colegas, amigos e músicos que passaram comigo esses anos em sala de aula e pelo incentivo e apoio prestado. Sandro Ferrari, Charles Janesch, Jair Paulo, Arquelau Pasta, Edson Vander, Edson Braz, Marilan Tagliari, Silos Pandini, Roberto Velloso, Rulye Oliveira, obrigado por estarem ao meu lado.

RESUMO

O presente trabalho trata do reconhecimento de notas musicais tocadas por acordeão. O reconhecimento parte de um arquivo *wave*, que deve conter notas musicais pré-gravadas. Em seguida é feito o processamento deste arquivo utilizando a transformada rápida de Fourier para encontrar a freqüência fundamental do som gravado no arquivo *wave*. Como resultado final, a nota reconhecida é representada em um teclado de acordeão no protótipo desenvolvido para validar este estudo. A estratégia de reconhecimento das notas musicais foi a utilização de um algoritmo que calcula a média ponderada dos valores de freqüência encontrados para obter uma resposta mais precisa.

ABSTRACT

The present work deals with musical note recognition for accordion. The note recognition starts from wave files containing pre-recorded musical notes. The process uses the Fast Fourier Transformation in order to find the pre-recorded sound's fundamental frequency. As a final result, the corresponding musical note is presented in a virtual accordion keyboard, which is part of the software prototype developed to validate the proposal. The musical note recognition method calculates the weighted media of frequency values as a mean to obtain more accurate results.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAS DE ABREVIATURAS

ADC *Analogic-Digital Converter* - Conversor Analógico Digital

CASE *Computer Aided Software Engineering* – Engenharia de Software Auxiliada por Computador

DAC *Digital-Analogic Converter* - Conversor Digital Analógico

FFT *Fast Fourier Transformation* – Transformada Rápida de Fourier

FT *Fourier Transformation* **–** Transformada de Fourier

OO Orientação a objetos

PCM *Pulse Code Modulation* - Modulação em Código de Pulsação

RIFF *Resource Interchange File Format* - Formato de Arquivo com Recurso de troca

UML *Unified Modeling Language* – Linguagem de Modelagem Unificada

SUMÁRIO

1 INTRODUÇÃO

A música é uma arte que está presente no mundo inteiro, entre todas as gerações e culturas distintas, e essa arte já existe desde os primórdios da existência humana. Segundo Ribeiro (1965), a música é a arte de combinar sons, de modo a produzirem sensações agradáveis ao ouvido e despertarem na alma emoções especiais. Com o passar dos anos, a música vem sofrendo uma grande evolução nas suas teorias e técnicas. Músicos que estudam técnicas avançadas e com variações muito rápidas, recheiam novas músicas com notas e acordes, dentro de um pequeno intervalo de tempo no compasso musical.

Como um dom, vários músicos têm uma percepção auditiva muito boa, e com isso conseguem capturar ou identificar as notas musicais com seus próprios ouvidos. Porém, muitos desses músicos não têm uma percepção musical boa, ou seja, não conseguem capturar ou identificar intuitivamente com a audição as notas musicais. Então acabam tocando a música incorretamente ou até mesmo deixam de tocar, por terem alguma dúvida em relação a alguma nota musical. Para que um músico tenha total domínio sobre seu instrumento, ele precisa dominar, tanto a teoria quanto a prática, e isso inclui um ouvido bem treinado. Sem uma boa percepção musical, dificilmente chega-se a uma boa proficiência em qualquer instrumento musical. Em virtude desse problema, desenvolveu-se um protótipo para o reconhecimento auditivo de notas musicais monofônicas, para auxiliar os músicos que possuem pouca percepção musical.

Para realizar o reconhecimento auditivo de notas musicais utilizam-se arquivos do tipo *wave*, onde foram gravadas as notas musicais e a transformada de Fourier. O formato do arquivo *wave* foi desenvolvido pela *Microsoft Corporation*, para fins de armazenar dados de áudio digital, pois com a sua estrutura, tornou-se fácil trabalhar os dados que este arquivo compõe. Mas para processar esses dados utiliza-se a transformada de Fourier.

Os estudos que geraram a transformada de Fourier tiveram início no século XVIII, quando o físico Joseph Sauvier e o músico Jean Philipe Rameau notaram que qualquer nota musical poderia ser analisada segundo uma freqüência fundamental e outras, múltiplas dessa, chamadas harmônicas. Após isso, muitos foram os estudiosos que trabalharam sobre tais observações, dentre eles Bernoulli e Euler, mas só em 1807, após Newton (e outros) terem desenvolvido o Cálculo Matemático, é que Jean-Baptiste Joseph Fourier estabeleceu uma

prova, através da qual, mostrou-se o quão geral era o fenômeno observado por Sauvier e Rameau. Fourier demonstrou que qualquer função *f(x)*, real e contínua, pode ser expressa como uma série de somas de termos. Utilizando os algoritmos da Transformada de Fourier, pode-se desenvolver um software para o reconhecimento de notas musicais conforme descrito a seguir (SANTOS, 2001).

Segundo Matlab (1997), a transformada de Fourier e a sua inversa são muito usadas em análise de circuitos para determinar as características de um sistema em ambos os domínios de tempo e de freqüência. Aplicando a transformada rápida de Fourier em um sinal de áudio correspondente a uma nota musical, composto por conjunto de valores ao longo do tempo, obtém-se amostras de freqüências provenientes das amostras de amplitude. Sabendo – se a freqüência fundamental da amostra, pode-se identificar a nota musical correspondente.

1.1 OBJETIVOS

Este trabalho de conclusão de curso tem como objetivo desenvolver um protótipo de software, para auxiliar os músicos no reconhecimento das notas musicais.

Os objetivos específicos do trabalho são:

a) ler um arquivo do tipo *wave*;
b) utilizar a Transformada Rápida de Fourier para transformar o sinal de áudio no domínio da amplitude para o domínio da freqüência;
c) identificar as notas fundamentais no espectro de freqüências;
d) gerar baterias de testes de reconhecimento de notas musicais.

1.2 ESTRUTURA DO TRABALHO

A seguir, apresentar-se-á uma breve descrição de cada capítulo do trabalho.

O primeiro capítulo apresentou uma introdução ao conteúdo deste trabalho, bem como a justificativa para realização do mesmo e seus objetivos.

O segundo capítulo apresenta uma breve revisão bibliográfica sobre música, a sua percepção e seus conceitos. Este capítulo também aborda aspectos relativos à tecnologia para armazenamento e processamento de sons no computador, tecnologia MIDI e som digital.

O terceiro capítulo apresenta um estudo sobre a transformada de Fourier e seus conceitos agregados.

O quarto capítulo apresenta a descrição do protótipo de Reconhecimento de Notas Musicais e detalhes de sua implementação.

O quinto capítulo apresenta as conclusões obtidas através da realização deste trabalho e também algumas sugestões para futuros trabalhos.

2 TEORIA DA MÚSICA

Hoje no mundo inteiro, a música chega a ter um grande espaço cultural na vida de muitas pessoas. O músico por sua vez, usa a música para expressar como é a sua vida ou como a vida é. Seja ela uma música alegre, ou uma música triste na qual, o músico talvez mostra a sua dor ou um sentimento que teve alguma vez no passado. Utilizando melodias e harmonias, o músico cria sons musicais combinados e sons musicais simultâneos.

Quando se trata de som musical, o efeito audível é definido de um ou vários movimentos de corpos vibratórios, que produzem o som. Tendo fontes sonoras pode-se produzir o som. Como exemplo: instrumentos de corda (como o violão, cavaquinho...), ou até instrumentos de sopro como a flauta, o acordeão entre outros. Outro tipo de instrumento a ser citado é a bateria, que na verdade é tido como um instrumento de membrana ou pele.

A vibração produzida tem diferentes características que definem as propriedades físicas do som. São chamadas de: altura, intensidade e timbre. A altura é a propriedade do som de ser grave, médio ou agudo. A intensidade é a propriedade do som de ser fraco, moderado ou forte e, é caracterizado pela amplitude da vibração do corpo vibratório. O timbre nada mais é do que a qualidade do som. É através dele que se diferencia e se reconhece o instrumento que produz o som.

2.1 NOTAS MUSICAIS

Embora seja infinito o número de sons empregados na música, para se representar a notação musical, pode-se utilizar apenas sete notas naturais. O Quadro 1 apresenta as sete notas:

Quadro 1 - As sete notas da notação musical

Dó	Ré	Mi	Fá	Sol	Lá	Si

Mas na verdade a seqüência das notas musicais também possui notas de meio tom, totalizando doze notas musicais. Além das notas naturais existem os sustenidos que são representados pelo símbolo #, e bemóis que são representados pelo símbolo *b.* No Quadro 2 é apresentada a seqüência correta na utilização das notas que, por sua vez, utiliza a

nomenclatura das notas sustenidos, ou seja, a interpretação de uma escala cromática ascendente:

Quadro 2 - Escala cromática ascendente

Dó	Dó#	Ré	Ré#	Mi	Fá	Fá#	Sol	Sol#	Lá	Lá#	Si	Dó

Já no Quadro 3 são representadas as notas que utilizam as nomenclaturas das notas bemóis:

Quadro 3 - Escala cromática descendente

Dó	Si	Si*b*	Lá	Lá*b*	Sol	Sol*b*	Fá	Mi	Mi*b*	Ré	Ré*b*	Dó

Este tipo de notação é utilizado no Brasil, mas existe uma notação, utilizada na Alemanha, Inglaterra, Estados Unidos e alguns outros países do mundo, que é a dos monossílabos, partindo-se do Dó. O Quadro 4 mostra esta notação:

Quadro 4 - Notação dos monossílabos

C	D	E	F	G	A	B

Já o Quadro 5 mostra a notação dos monossílabos de uma escala cromática:

Quadro 5 - Escala cromática

C	C#	D	D#	E	F	F#	G	G#	A	A#	B

Nas próximas seções serão apresentados conceitos básicos de teoria musical, tais como intervalos, escalas e acordes e conceitos de acústica.

2.2 INTERVALOS

Em teoria musical, o conceito de intervalo significa a distância entre dois sons, ou duas notas. Há duas espécies de intervalos, os intervalos simples, que são intervalos dentro de uma oitava[1] e os intervalos compostos, que são intervalos que vão além de uma oitava. Os intervalos podem ser caracterizados por maiores, menores, aumentados, diminutos e justos.

[1] Oitava é o espaço musical que compreende duas notas de mesmo nome, mas de alturas diferentes. Por exemplo, uma oitava é o espaço entre uma nota dó a próxima nota dó na escala, mais aguda que a primeira.

Dentro de uma escala de Dó maior, pode-se definir os intervalos de uma nota para a outra. Por exemplo: O intervalo da nota Dó para a nota Ré é de um tom; a de Ré para a nota Mi é de um tom; já o intervalo da nota Mi para a nota Fá é de um semitom; o intervalo da nota Fá para a nota Sol é de um tom; a de Sol para a nota Lá é de um tom; a de Lá para a nota Si e de um tom; e a nota Si para a nota Dó é de um semitom.

2.2.1 ESCALAS E ACORDES

As escalas são determinadas conjuntos de notas musicais, iniciando por uma nota, que recebe os adjetivos de "tônica" ou "fundamental", e mantendo intervalos bem definidos entre as notas do conjunto. Normalmente as escalas são compostas por conjuntos de sete notas. Existem várias escalas de notas musicais, podendo-se citar como exemplo a escala de Ré maior (sendo Ré a nota fundamental), a escala de Lá menor, a escala cromática comentada anteriormente nesta sessão, entre outras. Exemplificando melhor, no Quadro 6, tem-se a seqüência da escala de Ré maior ascendente:

Quadro 6 - Escala ascendente de Ré maior

Ré	Mi	Fa#	Sol	Lá	Si	Do#	Ré

Já no Quadro 7 tem-se a escala de Ré maior descendente:

Quadro 7 - Escala descendente de Ré maior

Ré	Ré*b*	Si	Lá	Sol	Sol*b*	Mi	Ré

Como citado anteriormente, tem-se a escala de Lá menor. No quadro 8 mostra-se a seqüência da escala de Lá menor harmônica:

Quadro 8 - Escala de lá menor harmônica

Lá	Si	Dó	Ré	Mi	Fá	Sol#	Lá

Além de existir a escala de Lá menor harmônica, tem-se a escala de Lá menor natural. No Quadro 9 mostra-se a seqüência da escala de Lá menor natural:

Quadro 9 - Escala de lá menor natural

Lá	Si	Dó	Ré	Mi	Fá	Sol	Ré

Segundo Chediak (1986), o acorde é um conjunto de três ou mais notas musicais diferentes. Quando formado por três sons é chamado de tríade, por quatro sons de tétrade e por mais de quatro sons, de tétrade com nota acrescentada. Assim como nas escalas, os acordes também possuem uma nota tônica, que é aquela nota cujo som é mais representativo dentro do conjunto de notas que formam o acorde.

Os acordes podem ser reproduzidos de forma melódica ou harmônica. A reprodução melódica é a reprodução de um acorde de forma arpegiada, ou seja, nota após nota. A reprodução harmônica é caracterizada pelo toque dos intervalos pertencentes ao acorde simultaneamente, de forma a soarem juntos, ao mesmo tempo, como um som só.

2.3 PROPRIEDADES FÍSICAS DO SOM

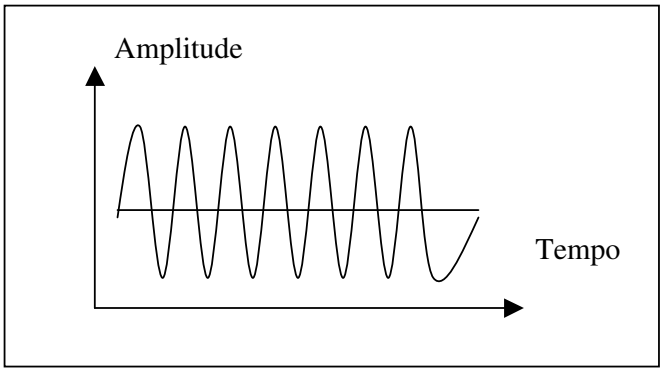
A audição resulta na percepção de flutuações periódicas da pressão de um meio, que é o ar que o ser humano respira. Contudo, o ouvido do ser humano não percebe esta pressão, mas recebe as vibrações que o ar transmite, levando em consideração uma faixa de parâmetros, ou seja, valores que esta pressão gera para a percepção de um ser humano.

Em função dos equipamentos de captação, o seu funcionamento para um sistema eletrônico de qualquer natureza, tem-se que as vibrações sonoras são convertidas em sinais elétricos por transdutores. Os transdutores de sinais elétricos para sinais acústicos chamam-se de auto-falantes e os transdutores de sinais acústicos para sinais elétricos chamam-se microfones.

Para que se detalhe as propriedades do som, os complexos harmônicos e as filtragens, é necessário que se conheça as propriedades dos sinais periódicos. Conforme Paula Filho (2000), a onda senoidal é a forma mais simples de se detalhar, graças a um fato matemático fundamental.

Na Figura 1 tem-se um exemplo de uma onda senoidal, sendo que esta ondulação, ou comprimento da onda que existe entre o eixo da amplitude e do tempo é a freqüência:

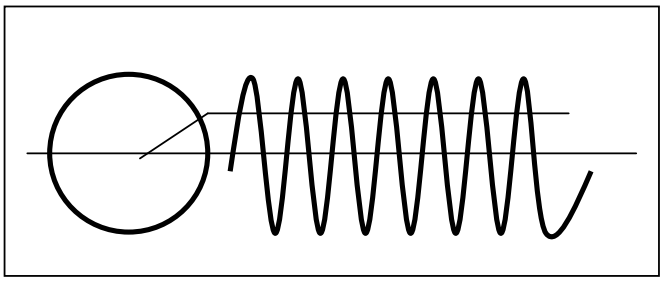
Figura 1– Onda senoidal



Fonte: Paula Filho (2000)

Conforme Paula Filho (2000), "a senóide é gerada quando um ponto percorre uma circunferência com movimento angular uniforme; a coordenada *y* do ponto (sua projeção no eixo vertical) descreve uma senóide. Pode-se mostrar também que a coordenada *x* gera uma cossenóide que é uma senóide deslocada a noventa graus". Na Figura 2 tem-se a geração da senóide com a sua circunferência e o comprimento da onda:

Figura 2– Geração da senóide



Fonte: Paula Filho (2000)

2.4 PARÂMETROS PERCEPTUAIS DO SOM

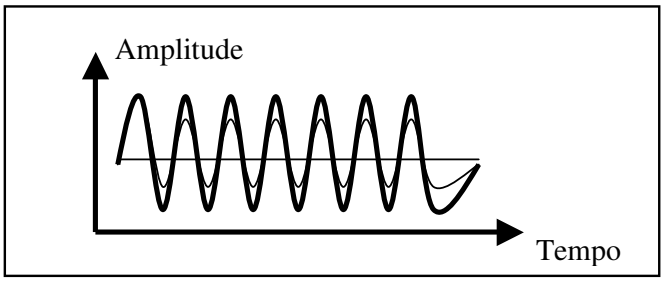
Na onda senoidal observam-se as propriedades perceptuais básicas do som:

- intensidade: qualidade que distingue sons fortes de sons fracos referente a volume;
- altura: qualidade que distingue sons graves de agudos;
- timbre: qualidade que distingue sons da mesma intensidade e altura, quando emitidos por instrumentos diferentes.

2.4.1 INTENSIDADE

A intensidade representa a potência acústica entregue pelo sinal, sendo que a potência é uma quantidade de energia por unidade do tempo, medida em *watts*. A amplitude de uma senóide determina uma potência sonora, ou seja, a potência é proporcional ao quadrado da amplitude. Conforme Paula Filho (2000), a Figura 3 representa dois sons com amplitudes diferentes.

Figura 3 – Diferentes amplitudes



Fonte: Paula Filho (2000)

Ainda conforme Paula Filho (2000), a percepção do ouvido é logarítmica, ou seja, para dobrar a intensidade de um som é preciso dez vezes mais potência acústica, sendo que a sensação da intensidade é bastante complexa, considerando todos os tipos de percepção musical.

A potência é medida em decibéis que é representado pela sigla *dB*. A intensidade de um sinal sonoro em decibéis é igual a dez vezes a potência de um som de referência sendo, que a referência pode ser a maior amplitude produzida por um dispositivo de som ou menor a intensidade audível. Considera-se que a intensidade audível tem 0 *dB* e a potência audível chega a 120 *dB*.

Tem-se os valores em decibéis para cada tipo de som. Conforme Paula Filho (2000).

Quadro 10 – Potência de sons típicos

Exemplo de sons	Nível (dB)
Limiar da audibilidade	0
Estúdio acústico	20
Sala de estar	40
Conservação normal, a 1 m	60
Rua de cidade	80
Grito a 1,5 m	100
Decolagem de jato	120

Fonte: Paula Filho (2000)

Pode-se considerar que a potência igual ou superior a 100 *dB* é uma poluição sonora, já uma música tem 96 *dB* de potência, e o som dos telefones tem em média uma potência de 48 *dB*.

2.4.2 ALTURA

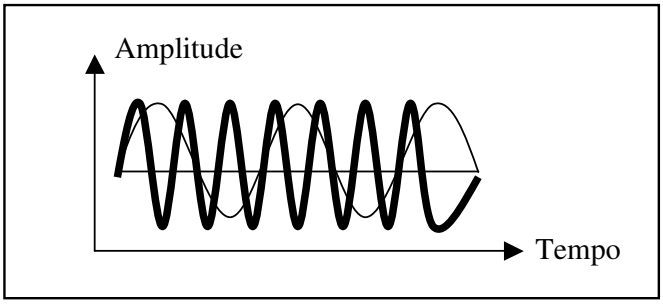
De acordo com Paula Filho (2000), "o segundo parâmetro que define a senóide é a sua freqüência, medida em ciclos por segundo, ou *hertz* (Hz). A freqüência corresponde ao número de ciclos por segundo que o ponto gerador da senóide percorre. Freqüências mais altas correspondem a senóides mais curtas horizontalmente, ou seja, de menor comprimento da onda".

Altura ou *pitch*, é a freqüência que corresponde do sinal senoidal, sendo que *pitch* de freqüências menores são mais graves e *pitch* com freqüências maiores são mais agudos. O volume significa intensidade sonora, não se pode associar altura com o volume. O ouvido humano tem um limite na percepção de altura. Conforme Paula Filho (2000), esse limite é inferior a 16 *Hz* e superior de 15 *kHz* a 20 *kHz*.

Cada nota musical tem a sua freqüência e costuma-se afinar a palheta (correspondente e uma tecla) do Lá central do teclado do acordeão em 440.00 *Hz*. Já um Lá em uma oitava abaixo (mais grave) tem a sua freqüência de 220.00 *Hz*, e um Lá em uma oitava acima (mais agudo) tem 880.00 *Hz*. Nota-se que de uma oitava para a outra a freqüência sempre vai aumentando em dobro ou diminuindo pela metade. No final deste trabalho, o anexo 1 contém uma tabela com as freqüências de todas as notas de um teclado de acordeão.

Na Figura 4 tem-se a representação de dois sinais de diferentes freqüências. O sinal representado pelo traço mais cheio tem a metade da freqüência do outro.

Figura 4 – Diferentes freqüências

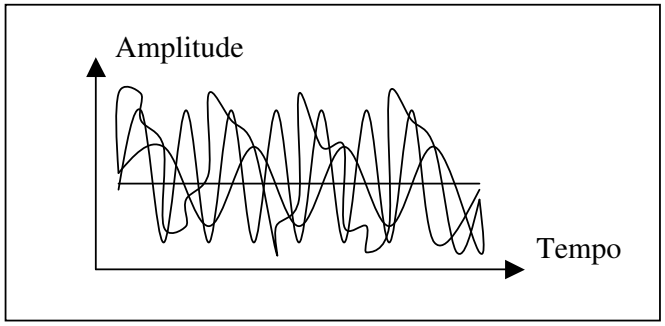


2.4.3 TIMBRE

O timbre é uma percepção auditiva muito importante para o ouvido do ser humano. O timbre diferencia um instrumento musical de outro, embora um Lá tocado em um teclado de acordeão tenha a mesma freqüência tocada por uma nota Lá no violão, por meio do reconhecimento do timbre, o ouvido humano pode diferenciar o som dos dois instrumentos. Matematicamente suas freqüências são iguais, mas pela percepção sabe-se que são timbres diferentes. A percepção do timbre trabalha com o aspecto do domínio do tempo e com aspecto do domínio da freqüência.

Na Figura 5, mostra-se como uma onda complexa pode ser formada pela composição de vibrações mais simples. Cada onda tem amplitude, freqüência e fases diferentes, sendo que, se a amplitude, a freqüência, e as fases soarem diferente, o ouvido humano perceberá um timbre diferente:

Figura 5 – Onda composta



Fonte: Adaptação de Paula Filho (2000)

A fase, que é agregada ao timbre, é considerada o início de uma senóide. Com tudo, não é tão importante para o som em termos de percepção, mais importante são os resultados intermediários do processamento digital do som.

2.5 ÁUDIO DIGITAL

Nesta sessão serão abordados o processo de digitalização do som e suas etapas para a realização do processo da conversão digital e o processamento digital do som onde, se envolve os processo de tempo e freqüência.

2.5.1 DIGITALIZAÇÃO DO SOM

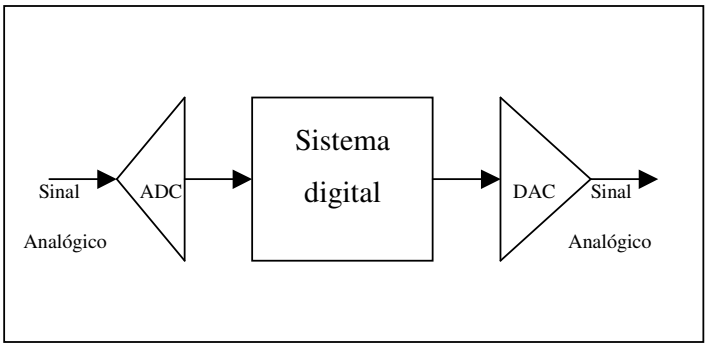
O sinal sonoro, que é um sinal de pressão mecânica, pode ser representado por dispositivos e sinais analógicos, que emitem sinais elétricos ou magnéticos. Para citar alguns exemplos de dispositivos analógicos, tem-se os amplificadores e os gravadores de fita cassete. Citando também os dispositivos e sistemas eletrônicos digitais, tem-se os equipamentos a disco *laser*, redes telefônicas digitais e outros.

Sendo que se pode gravar uma ou várias cópias em um sistema digital. A qualidade da última gravação é tão boa quanto à da primeira cópia. Leva-se em consideração que os sistemas digitais não são suscetíveis aos ruídos.

Na Figura 6, tem-se um exemplo de como é obtido um sinal digital. Para isso, tem-se que converter o sinal elétrico analógico, adquirido de um microfone ou equipamento analógico de reprodução. Em outras palavras, o sinal entra como sinal analógico passando

pelo conversor analógico digital (ADC), é convertido para sinal digital e depois tem-se a sua saída passando pelo conversor digital analógico (DAC), retornando novamente sinal analógico.

Figura 6 – Digitalização do som



Fonte: Paula Filho (2000)

Para detalhar o processo de digitalização do som, são respeitadas as seguintes etapas:

1. **filtragem:** neste processo faz-se limitação de uma faixa de freqüência que existe no sinal. As freqüências acima do corte da filtragem são eliminadas para evitar a ocorrência de pseudonímia. Também conhecida como teorema de *nyquist*, a pseudonímia faz com que um sinal periódico possa ser reconstruído se a taxa de amostragem obtiver um valor mínimo. Este valor deve ser superior ao dobro da freqüência, isto é, a freqüência mais alta no sinal digital. Resume-se que a pseudonímia é chamado de dobramento ou *folding*.

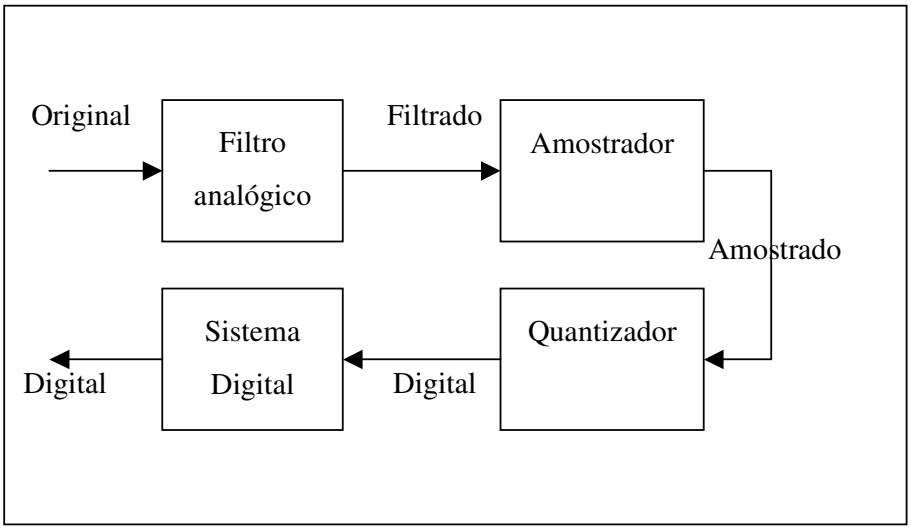
2. **amostragem:** é gerada por seqüências de pulsos, e cada pulsação contém uma amostra de som. A amostragem neste processo faz com que o sinal analógico passe por um circuito amostrador, e deste circuito distribua taxas de amostragens do som a ser digitalizado.

3. **quantização:** utiliza-se neste processo o conversor analógico digital (ADC), onde os pulsos gerados na amostragem passam a ser convertidos em números.

4. por último, faz-se a gravação dos arquivos.

Na Figura 7, o processo de digitalização do som está representado esquematicamente:

Figura 7 – Processo de digitalização do som



Fonte: Paula Filho (2000)

2.5.2 OPERAÇÕES DE PROCESSAMENTO DIGITAL DE SOM

Tem-se dois grupos de operações para o processamento digital do som. O processamento no domínio do tempo e o processamento no domínio da freqüência.

Para o processamento no domínio do tempo a representação digital do som presta-se com muita facilidade à operação de edição. No processamento no domínio do tempo pode-se simular um estúdio de som, com operações de cortar, colar, atenuar, realçar e misturar segmentos de sinais de áudio. O som digital pode ser armazenado, recuperado e transmitido como qualquer arquivo, e combinado com outros tipos de mídia.

Já o processamento no domínio da freqüência, abrange operações que não dependem do processamento de amostras isoladas, mas requerem o tratamento de seqüência de amostra, como um todo. Tem como suas aplicações: a filtragem digital, que pode ser usada para recuperação gravações; ajuste de duração e altura de amostras de som, usado para a correção de gravação em produção de efeitos especiais de áudio; várias técnicas de síntese musical; identificação e reconhecimento.

Relevantemente, cita-se a transformada de Fourier para o processamento digital do som, tema abordado no próximo capítulo.

2.6 SISTEMAS MIDI

Nesta sessão será descrito como é o sistema MIDI, descrevendo o seu protocolo, tecnologia, seus arquivos e um comparativo entre MIDI e áudio digital.

2.6.1 PROTOCOLO MIDI

Os instrumentos musicais eletrônicos como teclados, sintetizadores e outros, utilizam o protocolo MIDI para a sua comunicação. Este protocolo pode ser do tipo de uma rede local, onde é feita a comunicação de um ser humano com vários instrumentos, ou seja, o ser humano controla a regência de um conjunto de instrumentos.

Dentro de um computador, o protocolo MIDI faz a representação das músicas tocadas por ele. Este protocolo define o formato .MID, que são os arquivos que vão reproduzir a música através de um sintetizador MIDI.

Os grandes fabricantes de produtos compatíveis com o protocolo MIDI são encarregados da padronização internacional do protocolo. São esses fabricantes que formam a associação MIDI.

2.6.2 TIPOS DE INSTRUMENTOS MIDI

Conforme Paula Filho (2000), o instrumento MIDI mais popularmente conhecido é o teclado sintetizador e suas teclas são semelhantes às de um piano. Hoje no mercado existem vários tipos de teclados que variam de 49 notas até 88 notas. Nos teclados mais simples, o músico não tem controle do volume do som produzido, a não ser, através de um controle mestre de painel. Os teclados mais sofisticados produzem notas com intensidade e velocidade, proporcionando ao músico o mesmo tato produzido por um piano acústico.

Existem instrumentos MIDI, chamados módulos, seu funcionamento requer respostas de mensagens MIDI e não possuem teclados locais. Os módulos seqüenciadores funcionam com o armazenamento seqüencial de mensagens MIDI, isto é, possuem memória para este tipo de tecnologia. Neste módulo tem-se a possibilidade de reproduzir músicas pré-gravadas.

Se possuírem transdutores, os instrumentos acústicos também irão funcionar como um controlador MIDI. Instrumentos de sopro como o acordeão e instrumentos de corda como o violão, hoje podem ser fabricados com a tecnologia de instrumentos MIDI, que geram seus sons mecanicamente. Por exemplo, ao tocar uma tecla do acordeão MIDI, ou ao tocar uma corda do violão MIDI, será reproduzida a mensagem MIDI correspondente à nota que foi gerado a partir do instrumento.

2.6.3 A TECNOLOGIA MIDI

Com a tecnologia MIDI, o músico tem em mãos a sincronia com os outros dispositivos MIDI. O músico por sua vez, pode ter dez teclados ligados em sincronia e com apenas um teclado operar os outros nove. Contudo, o que faz tudo isso e é visível, é o *hardware.*

Com o uso do protocolo, os instrumentos eletrônicos musicais passam suas informações ou dados através de *Bytes*.

A tecnologia MIDI possui cinco conectores. O *MIDI IN, MIDI OUT, MIDI THRU, MIDI ON* e o conector *MIDI OFF*. O *MIDI IN* serve para receber os dados de um outro dispositivo, já o conector *MIDI OUT* serve para enviar. O conector *MIDI THRU* serve como um ponteiro, ou seja, usa-se como uma passagem de sinal e não se permite alterar os dados.

Os outros dois conectores fazem o papel real de um instrumento. Um exemplo clássico seria quando um músico pressiona a tecla de qualquer nota do teclado do acordeão, a nota fica soando até o músico tirar o seu dedo da tecla que pressionou. E os conectores que fazem este papel são o *MIDI ON*, quando o músico pressiona uma tecla, e *MIDI OFF* quando o músico solta a tecla do teclado. Estes dois conectores fazem com que o músico controle o tempo de execução de uma música.

Em outras palavras pode-se dizer que a tecnologia MIDI possui uma mensagem para cada ação musical. Existe a mensagem que regula o efeito da vibração da nota, a outra mensagem mantém as notas soando mesmo que as teclas sejam soltas, até desligar o instrumento musical MIDI. A outra, avisa o instrumento para mudar o seu *patch*, que significa mudar o conjunto de timbres.

2.6.4 ARQUIVOS MIDI

O formato *.MID* é bastante compacto em comparação a outros formatos de sinais de áudio. O *patch* é um número que codifica qual o tipo de timbre está sendo gerado, sendo que o resultado da reprodução dos arquivos *MIDI* será um sinal de áudio.

Os arquivos *.MID* são organizados em trilhas, ou seja, segundo Paula Filho (2000), isso vale para os arquivos *MIDI* formato 1, que são os mais comuns. O formato 0 mistura todos os eventos em uma única trilha. Uma variante é o formato *RMI*, que obedece à convenção da *Microsoft* para arquivos de mídia, chamada de *RIFF*, que será abordado ainda neste capítulo.

As trilhas representam as vozes dos instrumentos monofônicos, ou seja, a voz de um único instrumento musical. As trilhas têm a sua organização seqüencial de eventos, sendo que estes eventos correspondem ao tempo de duração do som e as mensagens *MIDI*, entre outros.

2.6.5 MIDI X ÁUDIO

Conforme Paula Filho (2000), aplicativos multimídia, arquivos de áudio como o arquivo *wave*, devem ser usados para a gravação de voz, ruído e efeitos de som. No caso de música, existe a opção de usar os arquivos *MIDI*. Estes apresentam as seguintes vantagens:

- Seu tamanho é muito menor que os arquivos de áudio, ou seja, seus dados são compactados;
- Permite que os parâmetros como durações e volumes sejam trabalhados nos seqüenciadores;
- Através dos mapeadores, permitem trocas na associação entre vozes e timbres;
- São mais aproveitados para a reprodução ou a geração de músicas em aplicações de tempo real, pelo fato de que seu consumo computacional é baixo.

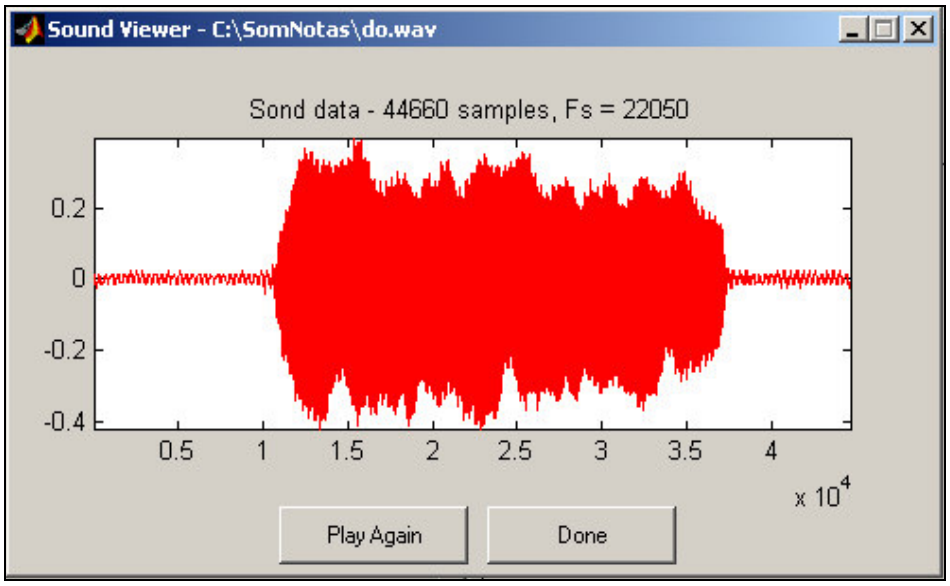
Por outro lado, vários problemas são encontrados nos arquivos *MIDI* como, por exemplo, reproduzem apenas músicas, não sendo recomendado à gravação de voz, ruídos e efeitos de som, e sua qualidade é baixa porque depende de um sintetizador *FM*. Para músicos clássicos, tem-se a dificuldade de representar a música não-convencional, ou seja, fazer um

jogo de foles no acordeão *MIDI* não fica perfeito, pois a placa que este instrumento possui simula um som quase que parecido com um jogo de foles tocado em um acordeão acústico.

2.7 ARQUIVOS WAVE

Segundo Meirelles (2002) o formato de arquivo do tipo *wave* é um dos mais populares para armazenamento de dados de áudio digital. Desenvolvido pela *Microsoft Corporation*, atingiu tal nível de popularidade devido ao grande número de programas escritos para a plataforma *Windows*. Atualmente, quase todo programa que armazena ou lê áudio digital tem suporte para esse tipo de arquivo. Na Figura 8 abaixo, tem-se um gráfico de amplitude x tempo de um arquivo *Wave* monofônico mostrando o seu comportamento, sendo que este arquivo corresponde à nota musical DÓ. Os valores de amplitude se encontram no eixo vertical, e o tempo está no eixo horizontal.

Figura 8 - Plotagem do arquivo do.wav



Fonte: Adaptação de MatLab (1997)

2.7.1 ESTRUTURA DO ARQUIVO

Conforme Meirelles (2002), os arquivos *wave* são estruturados no padrão *RIFF* (*Resource Interchange File Format*), que são comumente usados para arquivos multimídia na plataforma *Windows*. Os dados são organizados em *chunks* (pedaços), nos quais cada *chunk* tem um cabeçalho que descreve o tipo do *chunk* e o seu tamanho. A vantagem desse tipo de

organização é que programas que não reconhecem um certo tipo de *chunk* podem ignorá-lo e continuar o processamento do restante dos *chunks* conhecidos do arquivo.

O *Chunck* é um pedaço do cabeçalho, ou seja, uma estrutura à parte dentro de outra. O arquivo tem uma forma geral composta por um *headers* (cabeçalho) e o corpo (dados). Dentro desse *headers* existem outras formas. Os *chuncks* que possuem seu próprio cabeçalho e corpo também existem de outras formas. Por exemplo, o *chunk.fmt*, que é uma estrutura que tem seus *headers* ou identificador "*fmt"*, e seus dados (*sample rate*, número de canais, etc) que está contida dentro de uma estrutura mais geral. O importante é ressaltar que cada *chunk* começa com o seu identificador.

Há também, em alguns casos, a presença de *sub-chunks*. Os arquivos *RIFF* têm um *chunk* principal, denominado *RIFF chunk*, que define o tipo de dados como, por exemplo, associando-o ao tipo *wave*.

2.7.2 ESTRUTURA DO CABEÇALHO

Ainda conforme Meirelles (2002), para um estudo mais elucidativo do formato *wave*, toma-se como exemplo o arquivo "*chord.wav*". Ele é *PCM stereo* (*pulse code modulation*), é a maneira que o áudio digital é codificado nos arquivos *wave*, e em outras diversas aplicações. Para a transmissão ou recepção, ou gravação ou reprodução, o sinal analógico é digitalizado e codificado de acordo com o meio. Mais detalhes podem ser encontrados em [PULSE-CODE]. Neste caso, o *PCM* tem o tamanho de 97016 bytes, taxa de amostragem de 22.05 kHz e tem 16 bits por amostra. Essas informações podem ser encontradas dentro do *chunk* "fmt ", detalhado na Figura 9 abaixo.

Figura 9 - Chunk "FMT"

Offset	Size	Description	Value
0x00	4	Chunk ID	"fmt " (0x666D7420)
0x04	4	Chunk Data Size	16 + extra format bytes
0x08	2	Compression code	1 - 65,535
0x0a	2	Number of channels	1 - 65,535
0x0c	4	Sample rate	1 - 0xFFFFFFFF
0x10	4	Average bytes per second	1 - 0xFFFFFFFF
0x14	2	Block align	1 - 65,535
0x16	2	Bits per sample	2 - 65,535
0x18	2	Extra format bytes	0 - 65,535
0x1a	Extra format bytes		

Fonte: Meirelles (2000)

A primeira informação encontrada após o identificador é o tamanho do *chunk*, que será 16, caso não existam informações extras. Após isso se tem a informação sobre o código de compressão, atribuído de acordo com a Figura 10 em seguida.

Figura 10- Tabela padrão

Code	Description
0 (0x0000)	Unknown
1 (0x0001)	PCM/uncompressed
2 (0x0002)	Microsoft ADPCM
6 (0x0006)	ITU G.711 a-law
7 (0x0007)	ITU G.711 µ-law
17 (0x0011)	IMA ADPCM
20 (0x0016)	ITU G.723 ADPCM (Yamaha)
49 (0x0031)	GSM 6.10
64 (0x0040)	ITU G.721 ADPCM
80 (0x0050)	MPEG
65,536 (0xFFFF)	Experimental

Fonte: Meirelles (2002)

2.7.3 ORGANIZAÇÃO DOS DADOS DE ÁUDIO

Ainda conforme Meirelles (2002), os dados de áudio em um arquivo *wave* são guardados no *"data" chunk* e ordenados no padrão *little endian,*que quer dizer uma forma de ordenação dos dados com o primeiro bit menos significativo. As ordenações *little endian* são padrões usuais, não sendo, portanto uma exclusividade de arquivos *wave*. As informações encontradas no "*fmt* " *chunk* são essenciais para, por exemplo, fazer-se a leitura das amostras.

Pode-se calcular o número de amostras pela relação *Nsamples = Nbytes/(bitsPerSample/8)*, onde *Nbytes* é o número de *bytes* de dados, ou seja, o tamanho do *"data" chunk*. No caso do arquivo *"chord.wav"*, tem-se 48462 amostras.

2.7.4 NÚMERO DE CANAIS

Ainda conforme Meirelles (2002), o número de canais é bastante importante para entender-se a maneira na qual, estão organizadas as amostras. Um arquivo *stereo* para ser tocado, isto é, mandado para um conversor digital analógico (DAC), precisa que duas amostras sejam enviadas simultaneamente, o que é chamado de *sample frame*. Por tal motivo, as amostras são organizadas canal por canal, sendo que primeiro tem-se a primeira amostra do canal 1, seguida pela primeira amostra do canal 2, depois a segunda amostra do canal 1, seguida da segunda amostra do canal 2 e assim por diante. Essa organização está ilustrada nos quadros abaixo:

Quadro 11 - 8 Bits mono PCM

Amostragem 1	Amostragem 2	Amostragem 3	Amostragem 4
Canal 0	Canal 0	Canal 0	Canal 0

Quadro 12- 8 Bits stereo PCM

Amostragem 1		Amostragem 2	
Canal 0	Canal 1	Canal 0	Canal 0
(Esquerda)	(Direita)	(Esquerda)	(Direita)

Quadro 13 - 16 Bits mono PCM

Amostragem 1		Amostragem 2	
Canal 0	Canal 0	Canal 0	Canal 0
Baixa ordem	Alta ordem	Baixa ordem	Alta ordem
byte	byte	byte	byte

Quadro 14 - 16 Bits stereo PCM

Amostragem 1				Amostragem 2			
Canal 0	Canal 0	Canal 1	Canal 1	Canal 0	Canal 0	Canal 1	Canal 1
(Esquerda)	(Esquerda)	(Direita)	(Direita)	(Esquerda)	(Esquerda)	(Direita)	(Direita)
Baixa ordem	Alta ordem	Baixa ordem	Alta ordem	Baixa ordem	Alta ordem	Baixa ordem	Alta ordem
Byte	byte	byte	byte	byte	byte	Byte	byte

3 TRANSFORMADA DE FOURIER

Conforme Bruns (1995), no inicio do século XVIII, o matemático francês Jean Baptiste Fourier realizou pesquisas para descobrir como o som poderia ser dividido em elementos basicamente idênticos. Para isso, ele desenvolveu um processo matemático que divide um som em um número finito de ondas senóides, o qual é atualmente conhecido como análise de Fourier. Esse processo também pode ser utilizado em ordem contrária, de modo a produzir sons específicos artificialmente.

3.1 A TRANSFORMADA DE FOURIER

A transformada de Fourier (TF) de uma função h(t) dá-se através de:

$$\hat{h}(f) = \Im(h(t)) = \int_{-\infty}^{+\infty} h(t) e^{-i2\pi ft} dt \qquad (1)$$

A integral em (1) pode ser interpretada como uma média ponderada infinita de h(t) usando a função moduladora $e^{-i2\pi ft}$ como fator de ponderação. O operador Fourier $\Im$ transforma uma função h(t) definida no domínio do tempo (ou do espaço, a depender da interpretação física da variável t) para uma função representada por H(f) ou $\hat{h}(f)$ no domínio da freqüência f.

Se $h(t) \in \mathbf{L}^2(\mathbb{R})$ (espaço das funções de quadrado integrável), então sua TF $\hat{h}(f)$ sempre existe. Desta forma, $\Im$ é um operador $\Im : \mathbf{L}^2(\mathbb{R}) \rightarrow \mathbf{L}^2(\mathbb{R})$.

A importância fundamental de ter-se uma TF definida como operador em $\mathbf{L}^2(\mathbb{R})$ deve-se ao fato deste espaço ser de Hilbert e assim ter um processo de reconstrução garantido. A inversa da TF é $h(t) = \Im^{-1}(\hat{h}(f)) = \int_{-\infty}^{+\infty} \hat{h}(f) e^{i2\pi ft} df$.

Devido à fórmula de Euler, $e^{i\theta} = \cos\theta + i\,\mathrm{sen}\,\theta$, pode-se também escrever a TF de h(t) através de:

$$\hat{h}(f) = \underbrace{\int_{-\infty}^{+\infty} h(t)\cos(2\pi ft)dt}_{\text{parte real}} - i\underbrace{\int_{-\infty}^{+\infty} h(t)\,\mathrm{sen}(2\pi ft)dt}_{\text{parte imaginária}}$$

na qual destacam-se suas partes real $\mathrm{Re}\left(\hat{h}(f)\right)$ e imaginária $\mathrm{Im}\left(\hat{h}(f)\right)$.

Define-se a *amplitude ou espectro de Fourier* de f(t)

$$\left|\hat{h}(f)\right| = \sqrt{\mathrm{Re}\left(\hat{h}(f)\right)^2 + \mathrm{Im}\left(\hat{h}(f)\right)^2}$$

A TF possui diversas propriedades importantes, dentre as quais encontram-se:

Linearidade: $\Im(a.g(t) + b.h(t)) = a\Im\left(g(t)\right) + b\Im\left(h(t)\right) = a.\hat{g}(f) + b.\hat{h}(f)$

Escalonamento: $\Im\left(h(kt)\right) = \frac{1}{|k|}\hat{h}\left(\frac{f}{k}\right)$

Deslocamento no Tempo: $\Im\left(h(t - t_0)\right) = e^{-i2\pi f t_0}\hat{h}(f)$

Deslocamento na Freqüência: $\Im\left(e^{i2\pi f_0 t}h(t)\right) = \hat{h}(f - f_0)$

Funções Pares e Ímpares:

- se h(t) é par, então $\hat{h}(f)$ é a função real par $\hat{h}(f) = \int_{-\infty}^{+\infty} h(t)\cos(2\pi f t)dt$;

- se h(t) é ímpar, então $\hat{h}(f)$ é a função imaginária ímpar $\hat{h}(f) = -i\int_{-\infty}^{+\infty} h(t)\mathrm{sen}(2\pi f t)dt$.

A TF definida em (1) não é adequada para o tratamento de sinais digitais. Um sinal digital unidimensional é representado por um vetor (x_k), para k = 0,1,2,...,N-1, de N elementos. Ele é obtido por amostragem de um sinal analógico h(t), de forma que $x_k = h(k.T)$ são amostras igualmente espaçadas no tempo, tomadas em t = 0, T, 2T, 3T,... O valor constante T > 0 é chamado de *período amostral* e sua inversa $f_a = 1/T$ é a *freqüência amostral*, ou número de amostras por segundo. Para sons, valores típicos de f_a são 11.025 Hz, 22.050 Hz e 44.100 Hz.

Para sinais digitais (x_k) emprega-se a TF discreta

$$\tilde{x}_n = \sum_{k=0}^{N-1} x_k e^{-i2\pi nk/N} \text{ para } n = 0, 1, 2, ..., N\text{-}1 \qquad (2)$$

onde, para cada valor de n, $\tilde{x}_n$ fornece a amplitude das freqüências componentes, determinadas por

$$f = nf_a/N. \quad (3)$$

A transformada discreta de Fourier também é irreversível e possui as mesmas propriedades da TF contínua. Além destas, uma importante propriedade diz respeito aos valores simétricos nas posições n e N-n da TF discreta:

$$\tilde{x}_{N-n} = \overline{\tilde{x}_n} \text{ para } 1 \le n \le N/2.$$

Em outras palavras, consegue-se representar freqüências distintas no vetor até a posição N/2 do vetor $\tilde{\mathbf{x}}$. Assim, a freqüência máxima representável (freqüência de corte) é

$$f_c = \frac{N/2}{NT} = \frac{1}{2T} = \frac{1}{2} f_a .$$

Freqüências presentes no sinal, superiores a este valor, ocasionam um fenômeno denominado *aliasing* e não podem ser determinadas por (2). Deste modo, ao constituir uma amostragem, é necessário garantir que a freqüência amostral seja no mínimo o dobro da freqüência máxima presente no sinal e que se deseja identificar através da TF discreta.

3.2 A TRANSFORMADA RÁPIDA DE FOURIER (FFT)

A FFT não é uma nova transformada. É uma coleção de algoritmos de grande eficiência computacional para cálculos da transformada discreta de Fourier. Os algoritmos da FFT exploram propriedades, periodicidades e simetrias da transformada discreta de Fourier para agilizar o cálculo da transformada e da antitransformada. Esses algoritmos começaram a surgir em 1965 com o trabalho de Tucker (1992) e permitiram o inicio da aplicação do processamento digital em tempo real, particularmente para a voz.

Segundo Tucker (1992) a transformada rápida de Fourier é uma forma de processamento de sinais largamente aplicada. A FFT pode ser utilizada, por exemplo, em aplicações de comunicação, radares, sonares, processamento de sinais, processamento da fala, área de engenharia biomédica, simulações, síntese musical, entre outros.

3.2.1 ESFORÇO COMPUTACIONAL PARA A TDF

Para se obter os valores, são necessárias multiplicações e somas de números complexos. Dada a similaridade entre as duas expressões, podemos analisar uma delas. Para obter-se X(k) para um determinado valor de k, são necessárias n multiplicações complexas e (n – 1) somas complexas. Portanto, para os n valores, 0 <= n <= N –1, são necessárias N^2 multiplicações complexas e N(N-1) = N^2 de somas complexas. Assim a complexidade computacional da TDF é do tipo O(N^2) tanto para as multiplicações como para a soma complexa.

3.2 FUNCIONAMENTO DA FFT

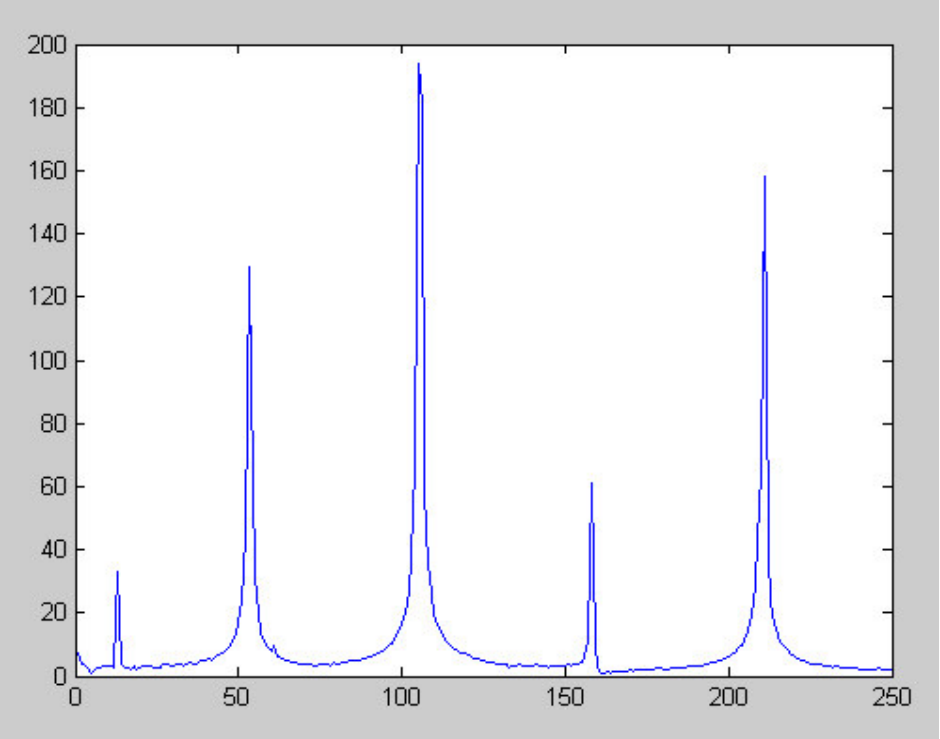
Segundo Cody (1992) uma função no domínio do tempo é traduzida pela FFT em uma função no domínio da freqüência, onde a função pode ser analisada pelas freqüências que nelas são encontradas.

A essência da FFT de uma onda é a decomposição ou separação da onda em uma soma de senóides de freqüências diferentes. A representação gráfica da FFT é um diagrama que mostra a amplitude e a freqüência de cada uma das diferentes senóides.

Mesmo que o número de freqüências diferentes existentes no som resulte em uma grande quantidade de dados que precisam ser processados, é valido utilizar-se desse processo matemático para otimizar os sinais obtidos, porque existe uma diminuição considerável na quantidade de sinais existentes sem que exista perda das características principais da onda.

No uso da FFT, devem ser tomados certos cuidados. A matriz utilizada como entrada para a função deve ser composta por um número de elementos, potência de dois, como (4, 8, 16, 32,...). Como resultado a FFT retorna uma outra matriz contendo (2^{n-1} +1) elementos. Por exemplo, quando se utiliza como entrada um vetor de 256 elementos, obtém-se um outro vetor com 129 elementos. A Figura 11 mostra uma aplicação da FFT sobre a forma da onda na nota Dó. Os picos salientam as freqüências componentes de 525 *Hz* e seus harmônicos.

Figura 11 - Pré-processamento da nota DÓ



Fonte: Adaptação de MatLab (1997)

Conforme a equação (3), para a nota Dó, observa-se que a maior freqüência no maior pico da figura, corresponde aproximadamente na posição 105. assim, f = 105 * 22050 / 4410, onde f = 525.

4 DESENVOLVIMENTO DO PROTÓTIPO

O presente trabalho resultou na criação de um protótipo de *software* que possibilita o reconhecimento de notas musicais, aplicando-se algumas das técnicas e teorias anteriormente vistas.

Nesta seção será descrita a especificação utilizada para o desenvolvimento do protótipo, bem como os requisitos e técnicas para a implementação do mesmo, além de um exemplo de sua utilização.

4.1 REQUISITOS PRINCIPAIS DO PROBLEMA

O protótipo, que foi desenvolvido para o reconhecimento de notas musicais, tem por objetivo reconhecer nota musical monofônica de um teclado de acordeão, sendo um requisito básico o reconhecimento de quarenta e uma notas musicais, desde a nota Fá mais grave do acordeão até a sua nota Lá mais aguda.

O *Kaco Acordes*, nome do protótipo, faz o reconhecimento de notas previamente armazenadas em arquivos do tipo *wave*. Para obter esses arquivos, o músico deve gravar as notas do teclado de um acordeão em seu PC e salvá-las em formato *wave*. Para tanto, será necessário um PC com um recurso computacional com multimídia. Mais especificamente, uma placa de som para fins de poder gravar qualquer nota do teclado do acordeão e armazenar no PC.

O requisito de gravação dos arquivos *.WAV* é de que a sua freqüência amostral deve ser de 22050 Hz, sendo do tipo mono e com 16 *bits* de resolução sonora. E com este, o músico pode abrir o arquivo no *Kaco Acordes* para realizar o reconhecimento da mesma, e visualizá-la no teclado que o protótipo deve possuir.

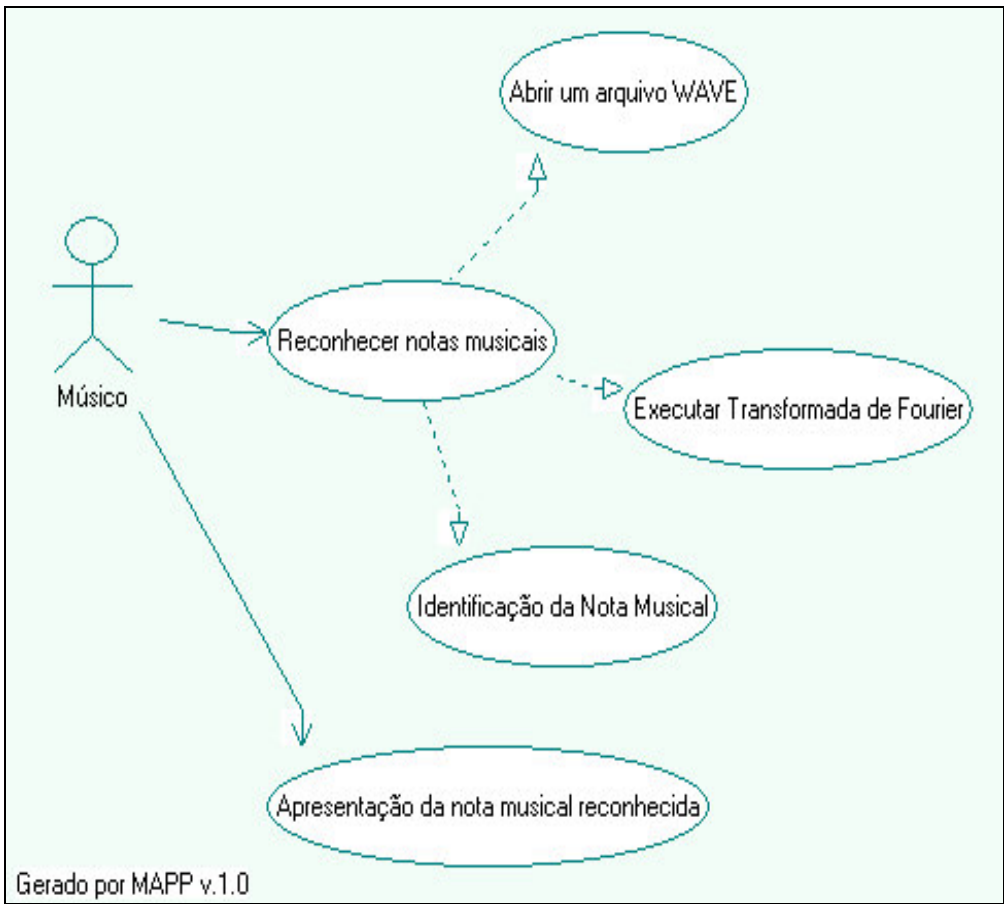
4.2 ESPECIFICAÇÃO DO PROTÓTIPO

Para a especificação do protótipo, utilizou-se a ferramenta CASE MAPP 1.0, Tagliari (2002), e com ela foi gerado o diagrama de casos de uso, diagrama de classes e diagrama de seqüências, através da UML e OO (orientação a objetos). Estes diagramas serão apresentados a seguir.

4.2.1 DIAGRAMA DE CASOS DE USO

A Figura 12 mostra o diagrama de casos de uso do protótipo, tendo como suas ações abrir um arquivo *wave*, reconhecer notas musicais, executar a transformada de Fourier, identificar da nota musical e a apresentar da nota musical reconhecida.

Figura 12- Diagrama de caso de uso



Abaixo, segue a explicação detalhada de cada evento do diagrama.

a) Abrir um arquivo Wave: o usuário deve manter em algum diretório de seu PC, arquivos do tipo *wave*, que foram gravadas para processar o reconhecimento da nota musical, e com isto, o usuário abre o arquivo desejado para fazer o reconhecimento;

b) Reconhecer notas musicais: é feita a leitura do arquivo *wave* e organizadas as informações do arquivo (*.data*) em um *array* dinâmico, dividem-se em blocos para a realização do processamento, sendo que cada bloco contém 4096 amostras de informações do arquivo *wave* que, significa um quinto de segundo em tempo,

considerando uma taxa amostral de 22050 Hz;

c) Executar transformada de Fourier: neste evento é processado o bloco de 4096 números de amostras na função transformada de Fourier. A função recebe essas amostras (valores do tipo inteiro) e após o processo, ela retorna o mesmo número de 4096 números de informações do tipo flutuante ou *float,* que representam valores de amplitudes de freqüência, em módulo, sobre os números complexos;

d) Identificação da nota musical: no evento que faz a identificação da nota musical, recebe-se as informações vindas da transformada de Fourier, calcula-se uma média ponderada para cada valor do vetor de freqüência das notas musicais. O cálculo da freqüência é basicamente feito de acordo com a equação (3), porém podem haver imprecisões na localização do pico de máxima amplitude correspondente a freqüência da nota, como pequenos desafinamento do instrumento musical, a nível aceitável. Esta consideração levou a considerar não apenas a amplitude espectral $|\tilde{x}n|$ correspondente a equação *n* no vetor resultante da transformação, mas também no seus dois valores visinhos à esquerda e à direita, por tomar $0.25*|\tilde{x}_{n-2}|+0.50*|\tilde{x}_{n-1}|+|\tilde{x}_n|+0.50*|\tilde{x}_{n+1}|+0.25|\tilde{x}_{n+2}|$. Com as posições calculadas, utiliza-se essa posição no vetor de informações da transformada de Fourier e com isto, será calculada a média ponderada para obter a nota reconhecida;

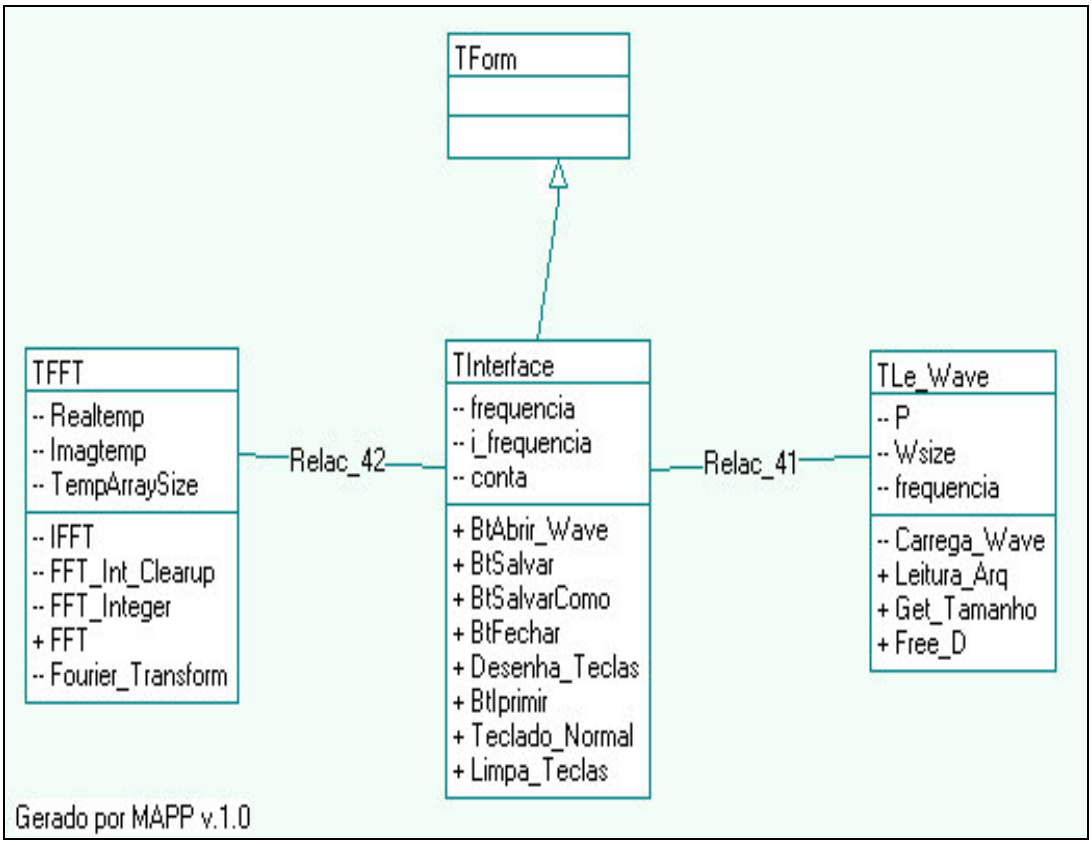
e) Apresentação da nota musical reconhecida: com as notas reconhecidas, este evento se responsabiliza de tocá-las no teclado existente na interface do protótipo. E com isto, o usuário tem a nota no teclado de cor vermelha e ao seu lado apresenta-se o nome da nota, escrito por extenso.

4.2.2 DIAGRAMA DE CLASSES

No desenvolvimento do protótipo foram identificadas quatro classes que são o formulário do protótipo. A classe *Tform* nada mais é do que um componente do *delphi* utilizado para desenvolvimento de aplicativos *Windows* que utilizam janelas de forma padronizada. A classe *Tinterface* fica responsável pelas ações do usuário de abrir, salvar, salvar como, imprimir e sair do protótipo. A classe *Tle_Wave* carrega os dados do arquivo *wave* para um vetor. Esses dados são levados para a classe *TFFT* para ser calculada a transformada de Fourier e depois retorna para a classe *Tinterface* para ser mostrado o resultado obtido.

A Figura 13, apresenta o diagrama destas classes e seus principais atributos e métodos:

Figura 13 – Diagrama de Classe



A classe *Tinterface*, apresentada na figura 14, é responsável por controlar o funcionamento do protótipo como um todo. E esta classe representa toda a estrutura do protótipo. Nesta classe são criadas as classes *Tle_Wave* e *TFFT*, que são responsáveis para que o processamento ocorra.

Ela contém os seguintes atributos:

a) *frequencia*: contém as informações dos códigos de cada nota musical;
b) *i_frequencia*: este atributo faz o controle de descarregar o atributo *freqüência* para fazer a identificação no teclado;
c) *conta*: contém o número de amostras reconhecida pelo processo.

Os métodos da classe são:

a) *BtAbrir_Wave*: abre um arquivo *wave* para ser processado ou um arquivo texto;

b) *BtSalvar*: salva a nota reconhecida em arquivo formato de texto;
c) *BtSalvarComo*: este método é a opção para o usuário salvar a nota reconhecida com outro nome;
d) *BtFechar*: este método se encarrega de fechar o protótipo;
e) *Desenha_teclas*: função que deixa a nota do teclado de cor vermelha, para fins de saber que aquela nota foi reconhecida;
f) *BtImprimir*: este método se encarrega de fazer a impressão da nota reconhecida;
g) *Teclado_Normal*: esta função faz com que o teclado virtual volte a sua cor estabelecida no padrão, depois de cada amostra reconhecida esta função faz este processo;
h) *Limpa_Teclas*: este método garante que o teclado virtual esteja com a sua cor padrão definida e depois fará o processo para o reconhecimento.

A classe *Tle_Wave*, apresentada na Figura 14, se responsabiliza no armazenamento dos dados do arquivo wave. Ela faz todo o controle para mandar em blocos o número de amostras exatas para a classe *TFFT*. Esta classe possui os seguintes atributos:

a) *P*: atributo que serve de ponteiro para indicar todos os dados do arquivo *wave*;
b) *Wsize*: atributo que tem o tamanho dos dados armazenados a partir do arquivo *wave*;
c) *Frequencia*: este atributo é o vetor dinâmico para o armazenamento dos dados do arquivo *wave*.

Os métodos desta classe são listados a seguir:

a) *Carrega_Wave*: esta função separa o cabeçalho e o corpo do arquivo e abstrai somente o que for dados (*.data*);
b) *Leitura_Arquivo*: esta função faz somente a leitura do arquivo controlado pelo atributo *P*;
c) *Get_Tamanho*: função que aloca memória para a realização do processamento;
d) *Free_P*: procedimento que libera memória depois do processamento realizado.

A classe *TFFT*, também apresentada na Figura 14, tem a responsabilidade de fazer todo o processo da transformada rápida de Fourier, e com o seu resultado gerado, volta para a classe *TInterface*. Esta classe possui os seguintes atributos:

a) *Realtemp*: atributo que contém informações dos dados do arquivo *wave* para o

processamento;

b) *TempArraySize*: atributo que faz o controle do *array* para os métodos desta classe.

Existem ainda os métodos da classe *TFFT* que são:

a) *ifft*: este procedimento chama o método *fourier_transform* para obter o cálculo da transformada de Fourier ;

b) *fft_int_clearup*: este procedimento controla a liberação de memória;

c) *fft_integer*: este procedimento controla a alocação de memória;

d) *fft*: este procedimento faz dar o início ao processo;

e) *fourier_transform*: procedimento que calcula a transformada rápida de Fourier, depois retorna o seu resultado para a classe *TInterface*.

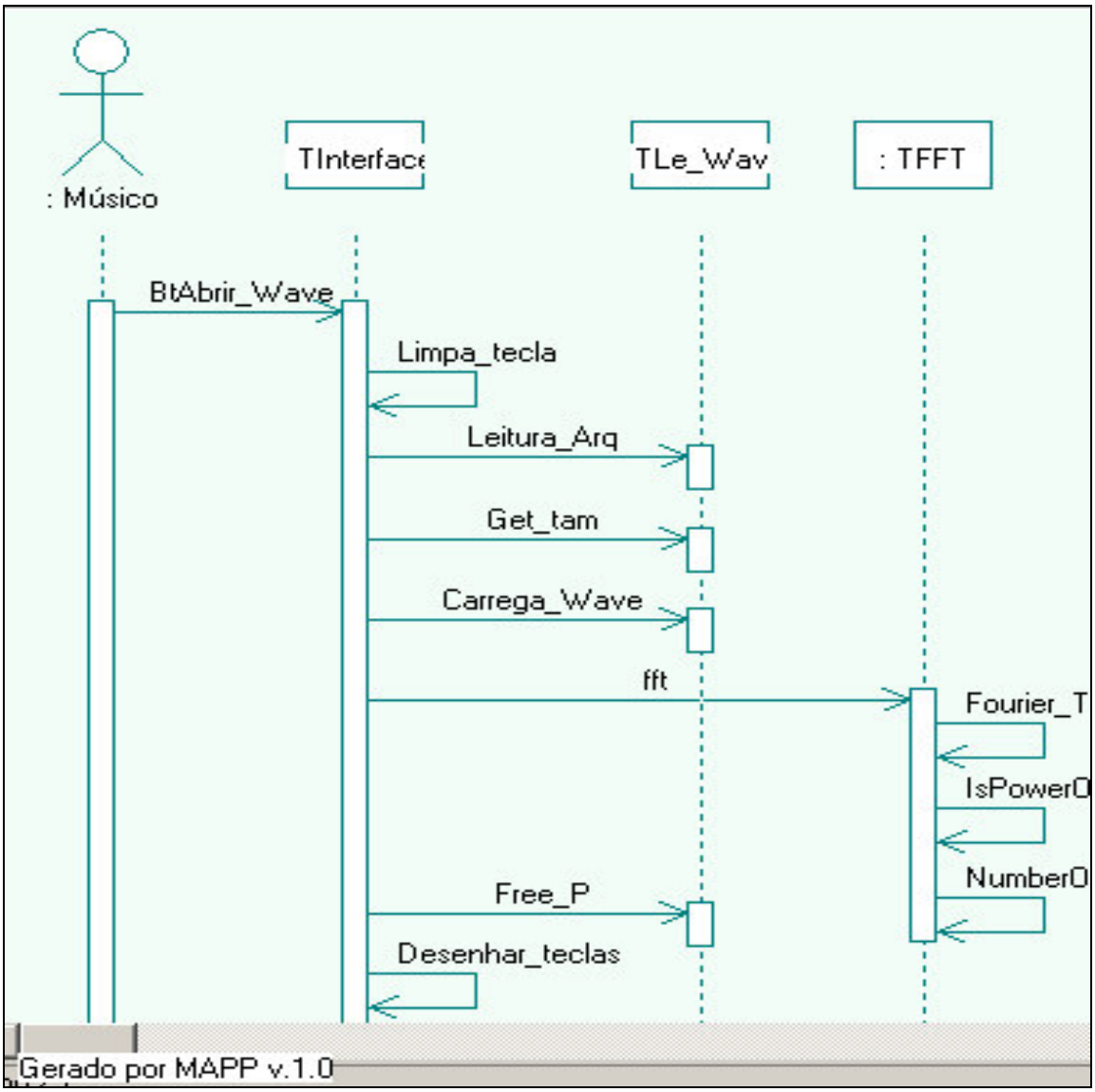
4.2.3 DIAGRAMA DE SEQÜÊNCIA

No diagrama de seqüência nota-se a sua passagem passo a passo no processo do sistema. Descrevendo-se este diagrama, o usuário ou músico abre um arquivo do tipo *wave*, e com o auxilio da classe *Tinterface*, ocorrem os processos que dão início ao processo de reconhecimento de notas musicais.

Conforme comentado anteriormente, primeiramente o protótipo faz uma checagem no teclado para ver se ele está com a sua cor padrão, faz a leitura do arquivo *wave*, depois aloca memória para realizar o processamento, carrega os dados do arquivo *wave* em um vetor dinâmico e chama a transformada rápida de Fourier.

Dando-se continuidade a seqüência do sistema, o processamento entra na classe *TFFT*, processa todos os dados contidos no vetor dinâmico e retorna para a classe *Tinterface*, onde se faz a montagem do teclado virtual e lista as notas dentro de um componente *Tedit.* A Figura 14 mostra o seu comportamento:

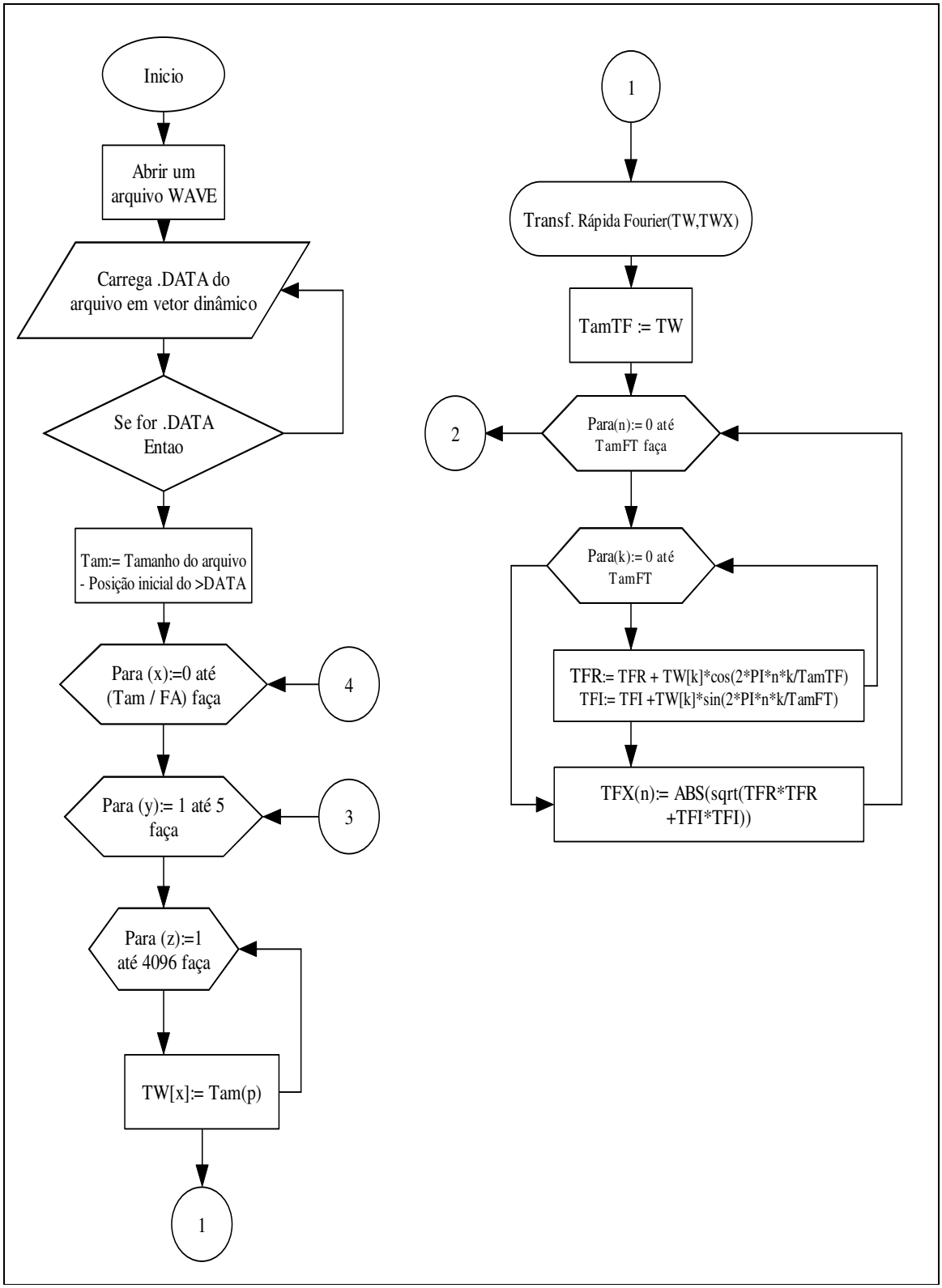
Figura 14 – Diagrama de seqüência

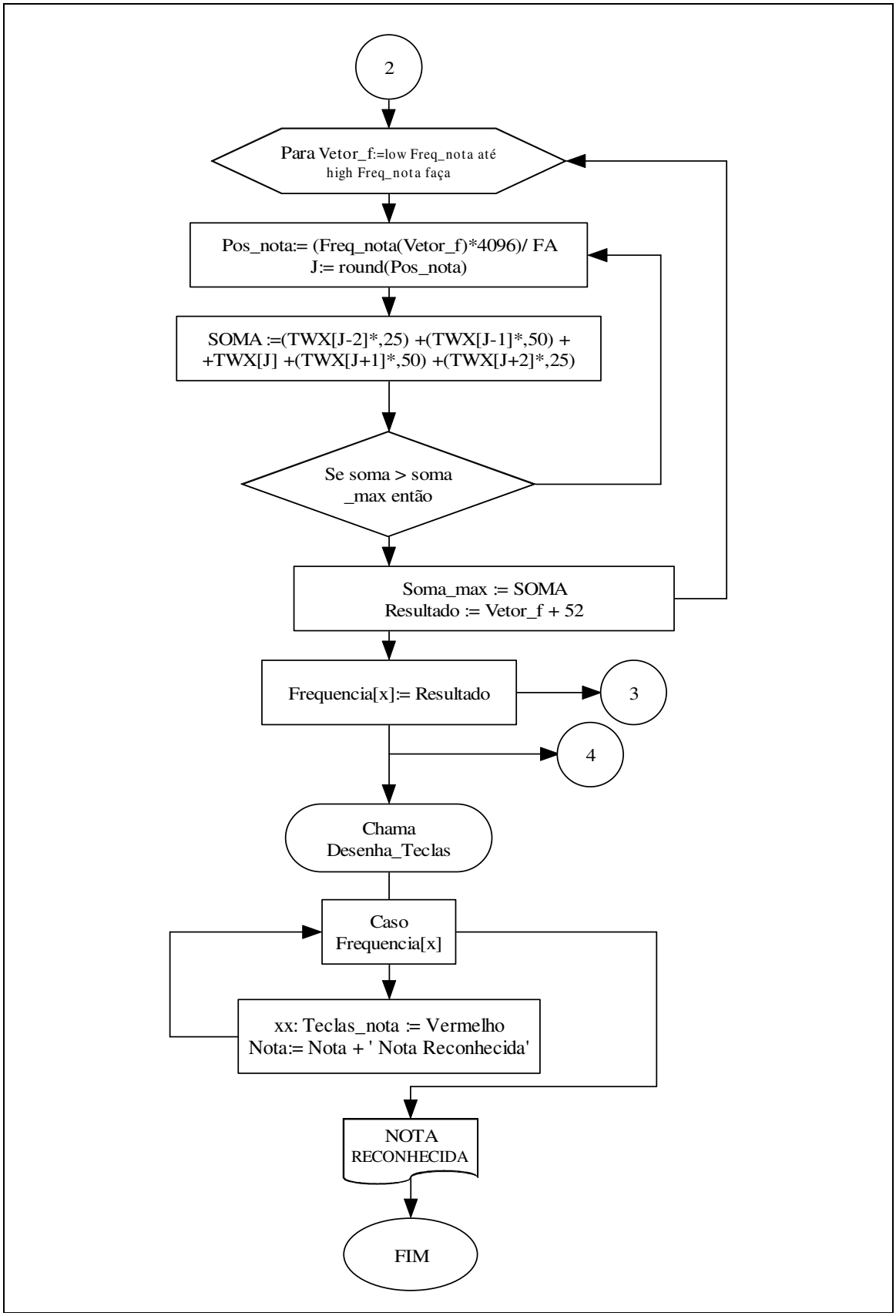


4.2.4 FLUXOGRAMA DO PROTÓTIPO

Para se ter uma outra visualização do protótipo, foi criado um fluxograma para um melhor entendimento do fluxo das informações que trafegam no protótipo. Na Figura 15 tem-se a estruturação do fluxograma do protótipo:

Figura 15 – Fluxograma do protótipo




2
Para Vetor_f:=low Freq_nota até high Freq_nota faça
Pos_nota:= (Freq_nota(Vetor_f)*4096)/ FA
J:= round(Pos_nota)
SOMA :=(TWX[J-2]*,25) +(TWX[J-1]*,50) +
+TWX[J] +(TWX[J+1]*,50) +(TWX[J+2]*,25)
Se soma > soma _max então
Soma_max := SOMA
Resultado := Vetor_f + 52
Frequencia[x]:= Resultado
3
4
Chama Desenha_Teclas
Caso Frequencia[x]
xx: Teclas_nota := Vermelho
Nota:= Nota + ' Nota Reconhecida'
NOTA RECONHECIDA
FIM

4.3 FERRAMENTAS E COMPONENTES UTILIZADOS

A primeira ferramenta utilizada para o desenvolvimento do protótipo foi o *MatLab 6.0*, MatLab (1997), ela possui várias bibliotecas e principalmente comandos, como por exemplo, o comando *fft* que calcula a transformada de Fourier. Devido a sua facilidade para computação simbólica e matemática, o *MatLab 6.0* foi utilizado com o propósito de realização de testes preliminares com transformada de Fourier.

Após os testes preliminares, o protótipo foi desenvolvido no ambiente de desenvolvimento *Delphi 5.0*, linguagem *Object Pascal*, Cantú (2000), e foram empregados conceitos de orientação a objetos. Os componentes utilizados no protótipo e oferecidos pela ferramenta são os *Tpanels* onde foi estruturado o teclado, um *TrichEdit* onde são listadas as notas musicais reconhecidas, um *TmainMenu* onde foi criado o menu. Um *TopenDialog* para abrir o arquivo que irá ser processado, *TsaveDialog* para salvar as notas reconhecidas e o *TprintDialog* para fins de imprimir as notas reconhecidas.

4.3.1 PROCESSO DO DESENVOLVIMENTO DO PROTÓTIPO

O protótipo, antes de seu desenvolvimento, passou por um planejamento para procurar a melhor técnica no desenvolvimento, e com este planejamento, hoje se tem quatro versões do protótipo**.** Com o planejamento concluído, foi visto que a utilização da transformada de Fourier é o método mais eficaz para fazer o reconhecimento das notas musicais.

Como todas as notas musicais possuem a sua própria freqüência, pode-se obter um cálculo que retorne o valor dessa freqüência. Mas, para se chegar a esse cálculo, o protótipo deve ler os dados previamente armazenados em um arquivo *wave* (*.data*) dentro de um vetor dinâmico, ou seja, utiliza-se um vetor dinâmico por não se saber a quantidade de dados que serão armazenados, e depois disso, pode-se trabalhar esses dados em partes iguais. No Quadro 15 apresenta-se o método responsável pelo armazenamento do arquivo.

Quadro 15 – Método CARREGAWAVE

```
// está função faz carregar os dados do arquivo wave
function TLe_Wave.carregawave(n: tfilename): pointer;
var
  cont : longint;
  tam  : longint;
  str  : array[0..3] of char;
  f    : file;
begin
  assignfile(f,n);// comando que abre o arquivo
  reset(f,1);
  for cont := 0 to filesize(f)-1 do begin
      seek(f,cont);
      blockread(f,str,4);
      if str = 'data' then break; //quando for .data pego a
  end;                            // sua posição
     tam := filesize(f)-filepos(f)-1;
       getmem(p,tam);
       wsize := tam;
       blockread(f,p^,tam,cont);
       CloseFile(f);
       result := p;
end;
```

Optou-se por trabalhar em amostras pequenas, sendo cada uma com uma duração de um quinto do segundo. Assim, o músico deve gravar a nota musical com uma freqüência amostral em 22050 *Hz*. Isso significa que, cada 22050 valores de amplitude (representados por números inteiros) do arquivo *wave* correspondem a um segundo de áudio digital, sendo assim, cada blocos analisado tem 4410 números inteiros.

Aplica-se a transformada de Fourier. sobre este bloco encontra-se seu vetor de amplitude amostral. Esses valores que a função da transformada de Fourier retorna, irão servir para começar a calcular o resultado desejado.

No Quadro 16 é apresentado o método que controla a quantidade de amostras para se realizar a transformada de Fourier.

Quadro 16 – Parte do método ABRIR

```
// controla a quantidade de amostra para ser processado na FFT
case AbreWave.FilterIndex of // verifica o tipo do arquivo
    1: begin  // caso o arquivo for wave
       // ponteiro do .data
       p2 := pbyte(Le_Arq.Leitura_arq(AbreWave.filename));
       for y := 0 to (Le_Arq.get_tam div 22050)-1 do
         begin // separa por segundo
         for x:= 0 to 4 do
           begin //separa por um quinto do segundo
           for z := 0 to 4096-1 do
             begin  // carrega um vetor com 4096 de amostra
               TW[z] := p2^;
               ImgIn[z] := 0.0;
               inc(p2); //incrementa o ponteiro
             end;
             // chama a transformada
             ttf.fft(4096, TW, ImgIn, TWX, ImgOut);
             for z := 0 to 4096-1 do// rápida de fourier
                TWX[z] := abs(TWX[z]);
             //TF(TW, TWX);  // chama transformada de fourier

           end;
         end;
         Le_arq.free_P;
         desenhar_teclas;
         end; // caso for arquivo texto entra no case 2
    2:listaNota.Lines.LoadFromFile(AbreWave.Files.Strings[I]);
 end; //case
```

4.3.2 DESCRIÇÃO DAS VERSÕES DO PROTÓTIPO

Conforme comentado anteriormente, desenvolveu-se quatro versões do protótipo. Na primeira versão adotou-se o cálculo da freqüência localizada na posição especifica de maior amplitude do vetor e, na segunda, utilizou-se a média ponderada. Ambas funcionam, mas a versão que utiliza a média ponderada mostrou ser mais eficiente que a outra versão. As outras duas versões utilizaram a transformada de Fourier com um algoritmo simples e a transformada rápida de Fourier, tendo o seu algoritmo mais complexo, porém mais eficiente na hora de seu processo.

A versão que calcula a freqüência, utiliza o retorno da função da transformada de Fourier e procura o maior valor desse vetor. Sendo assim, o vetor que a transformada de Fourier retorna é o que se chama matematicamente de espelhamento, ou seja, valores

duplicados. Como é um espelho, divide-se o vetor pela metade e com esses valores procura-se o maior valor que ele possui. Tendo-se o valor desejado, obtém-se a sua posição *n* no vetor, e com esta variável pode-se calcular a freqüência de acordo com a equação (3). O uso de *fa* = 22050 *hz* e de blocos de tamanho *N* =4410 conduz assim, a $f = \frac{n * 22050}{4410} = 5n$, ou seja, a freqüência corresponde a 5 (cinco) multiplicada pelo número do canal.

Com o valor da freqüência obtida, é feita a comparação da freqüência calculada com a freqüência definida para cada nota musical no padrão MIDI. Caso a freqüência obtida não seja exatamente igual à freqüência padrão MIDI, efetua-se um controle para ver em qual nota musical a freqüência obtida se encaixa, ou seja, com uma tolerância 1% para mais ou para menos dentro de uma freqüência de uma nota musical exata. Por exemplo, se a freqüência calculada for igual a 525.03 Hz, faz-se a seguinte pergunta: "Se (freqüência >= 520) e (freqüência <= 529) então Nota = Dó".

Nos Quadro 17 e mostra-se as condições para se obter o cálculo da freqüência e a nota reconhecida:

Quadro 17 - Verifica a maior posição

```
maior := 1;
for z := 1 to 2205 do //pega-se a metade do vetor da TF
    if TWX[z] > TWX[maior] then //verifica o maior canal
        maior := z;
inc(maior);
setlength(frequencia,length (frequencia) +1);
frequencia[high(frequencia)]:=calcula_midi((maior*22050)/4410);
                            //calcula a frequencia
```

Já com o cálculo da média ponderada, o processo necessita de um vetor com os valores de freqüência de cada nota musical mais um valor de 65 *Hz* que significa zona de silêncio ou ruído. No Quadro 18 tem-se este vetor com os valores de freqüência.

Quadro 18 – Vetor de freqüência musical

```
// valores reais das frequencias das notas musicais
const

   frequ_nota: array[0..41] of real = (65.00,
174.61,185.00,196.00,207.65,220.00,233.08,246.94,261.63,277.18,
293.66,311.13,329.63,349.23,369.99,392.00,415.30,440.00,466.16,
493.88,523.25,554.37,587.33,622.25,659.26,698.46,739.99,783.99,
830.61,880.00,932.33,987.77,1046.50,1108.73,1174.66,1244.51,
1318.51,1396.91,1478.98,1567.98,1661.22,1760.00);
```

Com estes valores disponíveis no vetor de freqüência *frequ_nota*, e os valores que retornam da transformada de Fourier *TWX*, passa-se então a calcular primeiramente uma posição. Esta posição calculada serve de posição no vetor dos valores da transformada de Fourier, e também fornece um valor com o qual é feita a média ponderada. Sendo que, cada posição calculada se refere a cada valor de freqüência contida no vetor *frequ_nota*.

Ainda neste processo, tem-se duas variáveis para fazer o controle. A variável *soma* tem a função de receber o valor da média ponderada, e depois ser comparada com a variável *soma_Max*. Caso a *soma* for maior que a *soma_Max* , a *soma_Max* será atualizada com o valor de *soma,* e a terceira variável *resultado* se atualizará com o código da nota musical reconhecida a cada valor de freqüência calculado. Nota-se que *resultado* recebe a posição do *loop* da função mais 52, isto significa que o resultado é 53 e, corresponde a primeira nota do teclado do acordeão que é a nota Fá.

No Quadro 19 tem-se a notação do código-fonte para o cálculo da média ponderada:

Quadro 19 – Cálculo da média ponderada

```
soma_max:= 0;
  for vetor_f := low(frequ_nota) to high(frequ_nota) do
    begin
      pos_nota := (frequ_nota[vetor_f] * 4096) / 22050;
      j := round(pos_nota);
      soma := (TWX[j-2]*0.25) + (TWX[j-1]*0.5) + TWX[j] +
                (TWX[j+1]*0.5) + (TWX[j+2]*0.25);
      if soma > soma_max then begin
         soma_max := soma;
         resultado := vetor_f + 52;
      end;
    end;
 setlength(frequencia, length(frequencia)+1);
 frequencia[high(frequencia)] := resultado;
```

Na utilização da transformada de Fourier, utilizou-se um algoritmo simples, porém não tão rápido quanto a transformada rápida de Fourier. No Quadro 20 tem-se o algoritmo da transformada de Fourier.

Quadro 20 – Transformada de Fourier

```
procedure Tfrmgeral.TF(var X: array of byte; var TFX: array of
real);
Var
   k,n,tam : integer;
   TFR, TFI : real;
begin
     tam := length(X);
     for n := 0 to tam-1 do begin
          Application.ProcessMessages;
          TFR := 0; TFI := 0;
          for k := 0 to tam-1 do
          begin
             TFR := TFR + X[k]*cos(2*pi*n*k/tam);
             TFI := TFI + X[k]*sin(2*pi*n*k/tam);
          end;
          TFX[n] := abs(sqrt(TFR*TFR + TFI*TFI));
     end;
end;
```

Este algoritmo não é tão rápido pelo fato de não trabalhar com alocação de memória constantemente. Já a transformada rápida de Fourier utiliza este método para fins de processar

o seu cálculo rapidamente. Mas existem regras para o processamento. Como mencionado anteriormente, o número de amostragem que vai para a transformada de Fourier era de 4410, mas a transformada rápida de Fourier aceita o número de amostras sendo múltiplos de dois. Sendo assim, o número de amostra para o processamento das notas musicais é de 4096. Um número de amostra mais aproximado de 4410. Nos Quadro 21 e 22 mostra-se parte do algoritmo que contribui para ser rápido e eficiente.

Quadro 21 – Aloca memória na FFT

```
procedure TTTF.fft_integer(NumSamples: word; var RealIn,
  ImagIn: array of integer; var RealOut, ImagOut: array of
double);
var
    i: word;
begin
    if NumSamples > TempArraySize then begin
       //libera memória caso seja limpa
       fft_integer_cleanup;
       // aloca memória para a proxima amostragem
       GetMem ( RealTemp, NumSamples * sizeof(double) );
       GetMem ( ImagTemp, NumSamples * sizeof(double) );
       // atualiza esta variável com o número de amostra 4096
       TempArraySize := NumSamples;
    end;

    for i := 0 to NumSamples-1 do begin
        RealTemp^[i] := RealIn[i];// numeros de amostra
        ImagTemp^[i] := ImagIn[i];// valores zero para imagem
    end;
    //chama a procedure para o calculo da fft
    FourierTransform (2*PI,NumSamples,RealTemp^, ImagTemp^,
                      RealOut, ImagOut );
end;
```

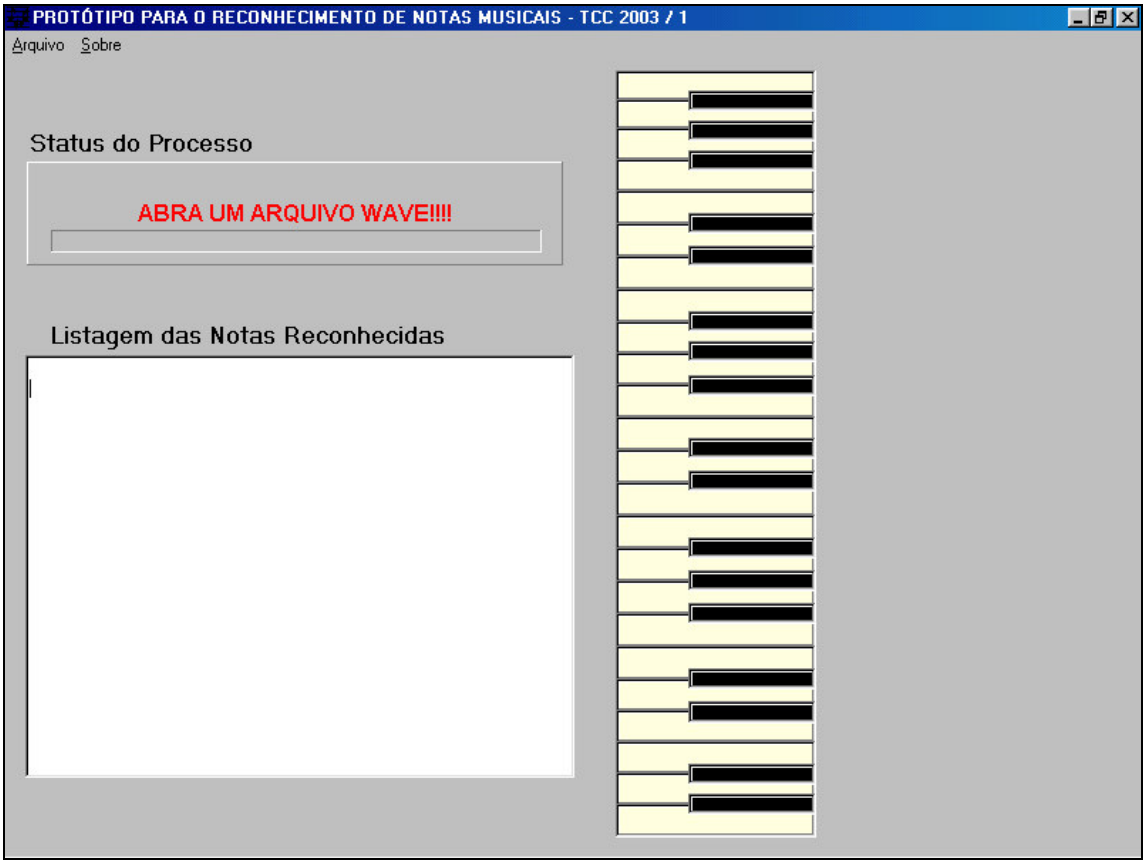
Quadro 22 – Libera memória para FFT

```
procedure TTTF.fft_integer_cleanup;
begin
    if TempArraySize > 0 then begin //verifica as amostra
      if RealTemp <> NIL then begin
        FreeMem ( RealTemp, TempArraySize * sizeof(double) );
        RealTemp := NIL;// foi liberada a memoria da amostra
      end;

      if ImagTemp <> NIL then begin
        FreeMem ( ImagTemp, TempArraySize * sizeof(double) );
        ImagTemp := NIL;
      end;

      TempArraySize := 0;
    end;

end;
```

4.3.3 DEMOSTRAÇÃO DA IMPLEMENTAÇÃO

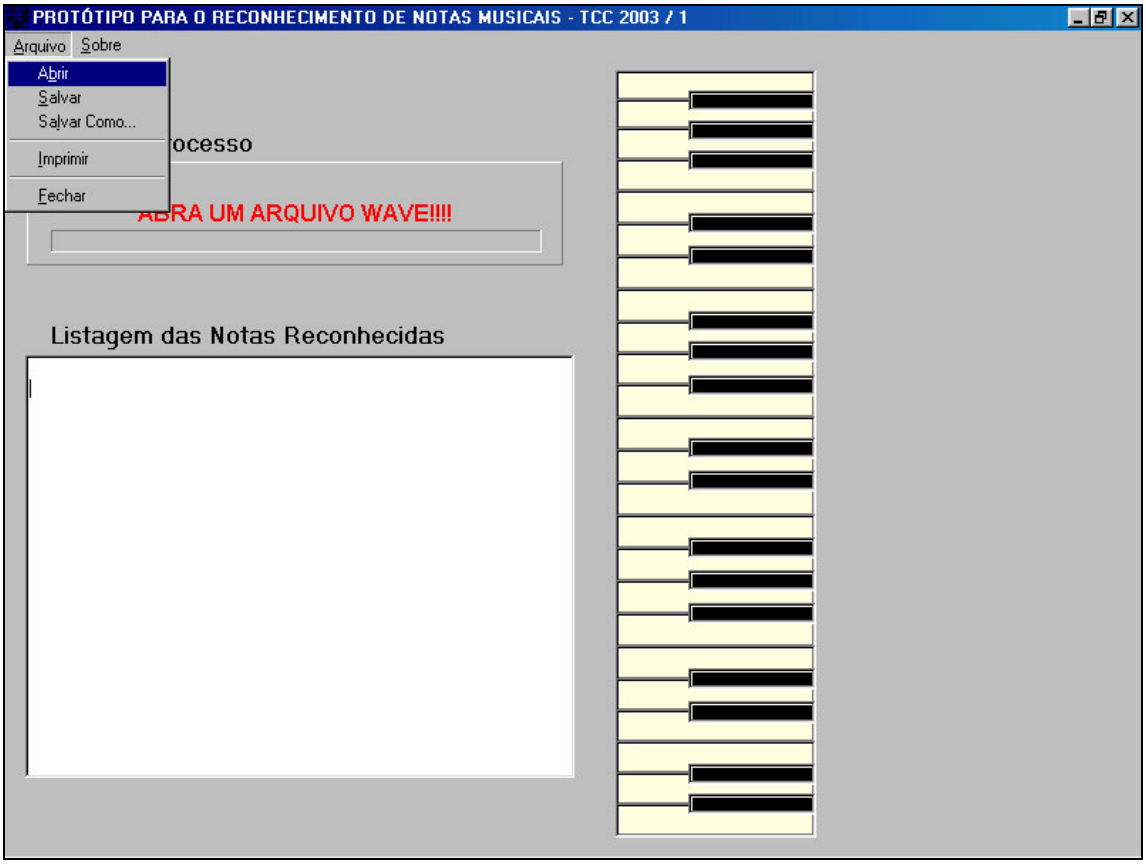
A seguir serão apresentadas, as telas do protótipo e as suas formas de utilização. Na Figura 16 tem-se a tela do protótipo.

Figura 16 – Tela do Protótipo



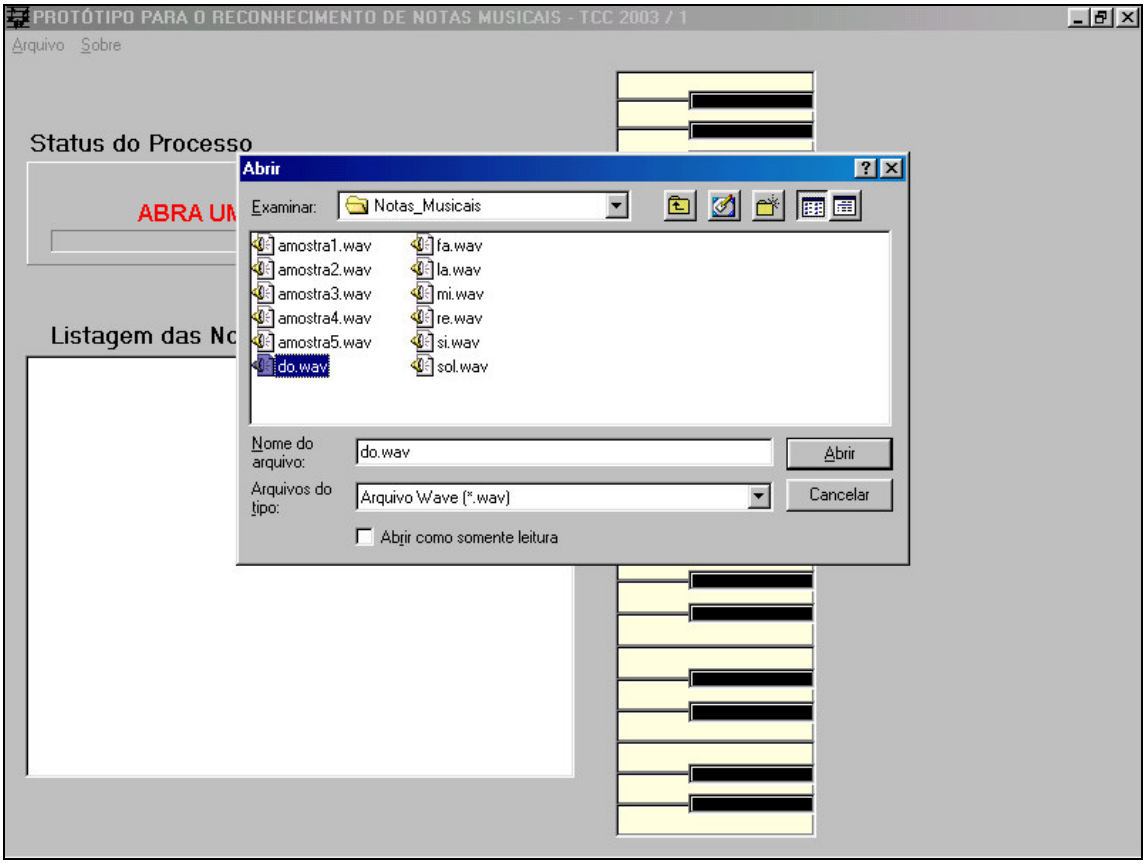
Com o protótipo aberto, o músico pode abrir um arquivo *wave* que está contido em alguns de seus diretórios. Na Figura 17 mostra-se este procedimento.

Figura 17 – Tela menu do protótipo



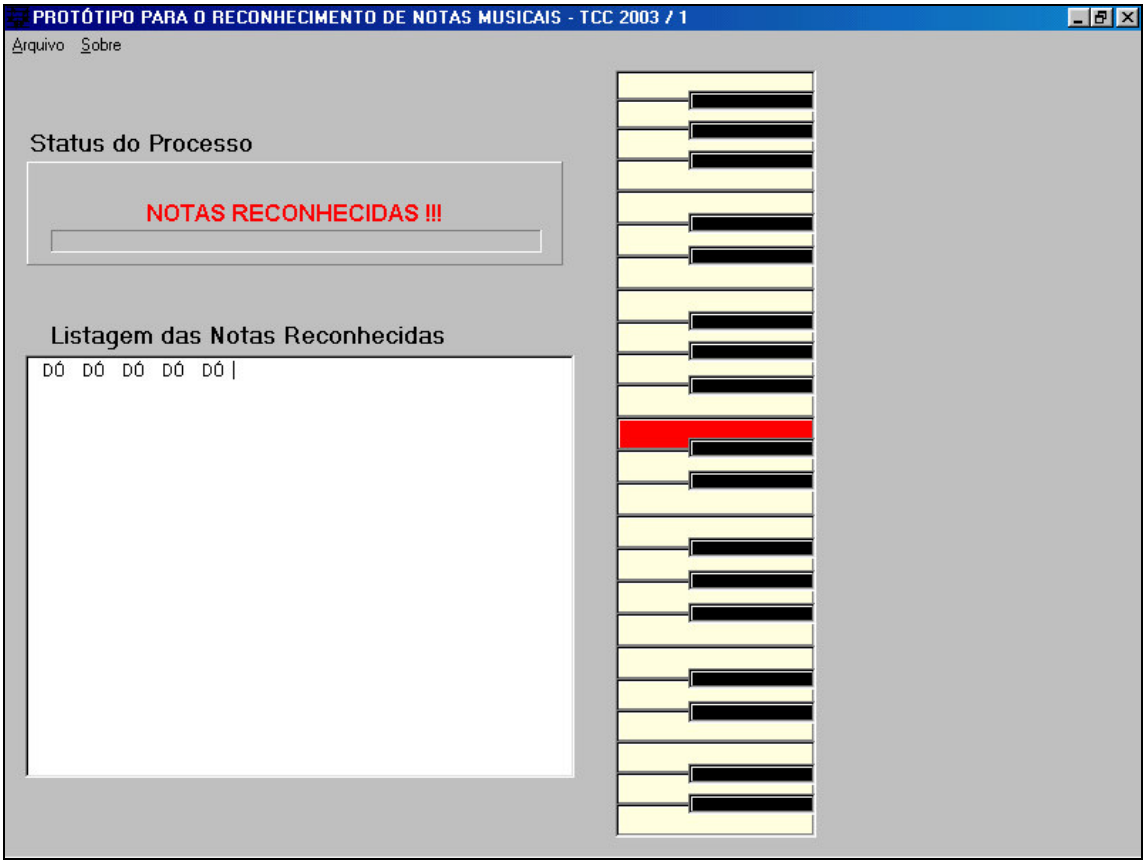
Quando o músico optar por abrir um arquivo *wave*, aparecerão as opções de diretório e ele então seleciona o arquivo desejado. Neste caso, o músico quer reconhecer o arquivo "do.wav" que está no diretório C:\Notas_Musicais. A Figura 18 mostra este procedimento.

Figura 18 – Tela Abrir arquivo Dó.WAVE



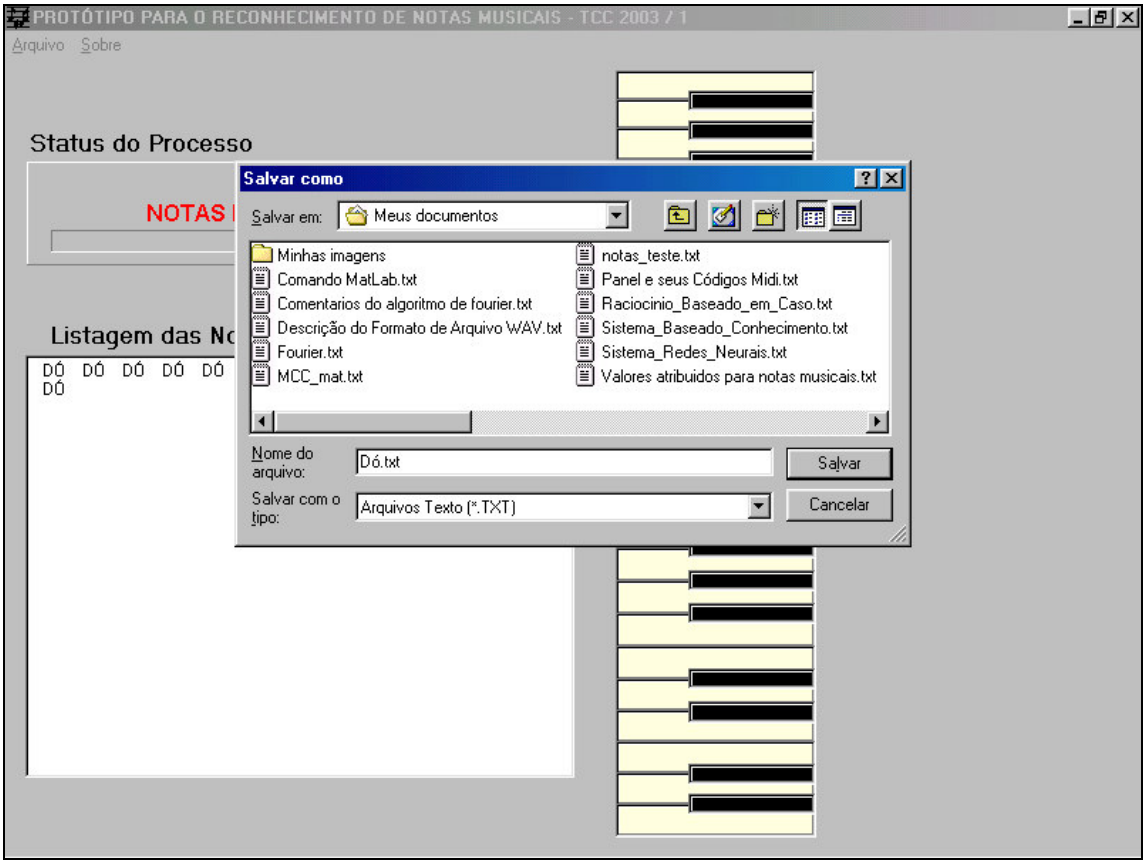
Após selecionado e aberto o arquivo, o protótipo fará o processamento do arquivo do.wav em alguns segundos e mostrará para o músico a nota musical reconhecida através do teclado e da listagem das notas musicais. A Figura 19 mostra o resultado obtido.

Figura 19 – Tela reconhecendo a nota musical Dó



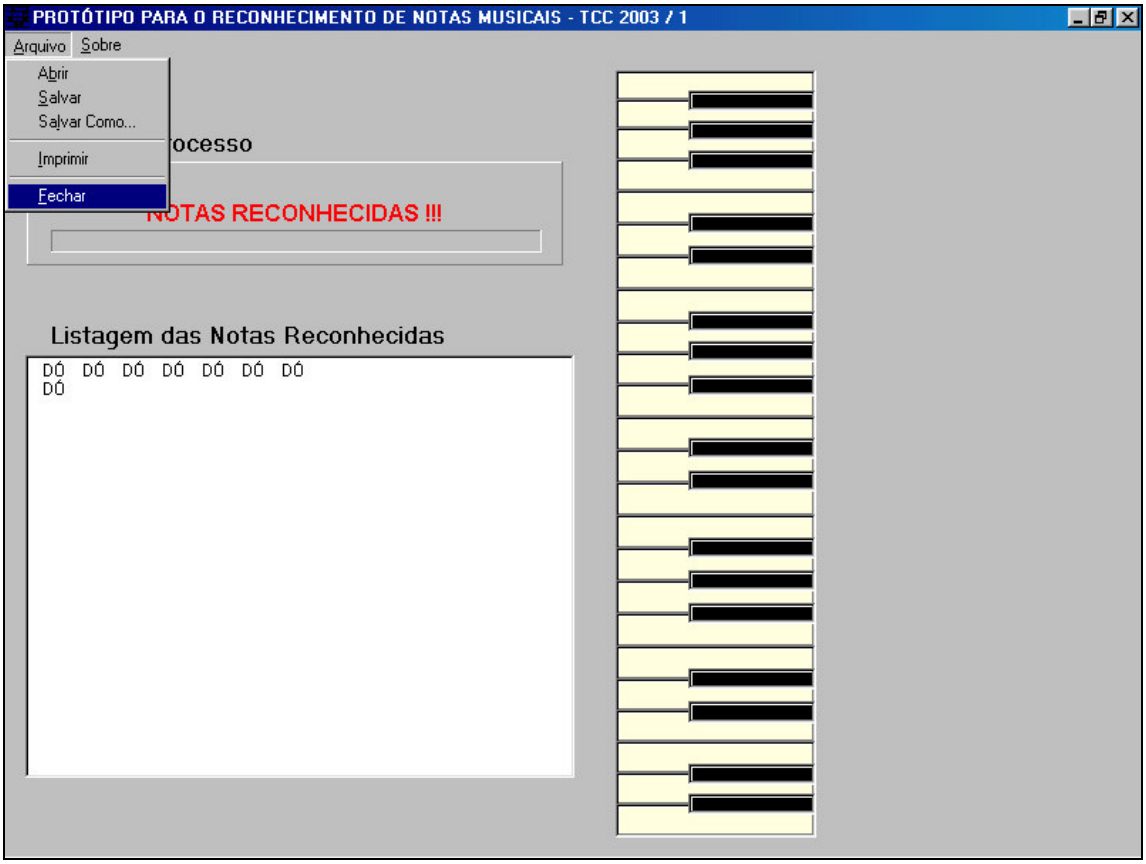
Depois da nota reconhecida, o músico pode salvar em arquivo texto o resultado obtido pelo protótipo. Na Figura 20 tem-se este processo de salvar a nota reconhecida.

Figura 20 – Tela salvando a nota Dó



E na Figura 21 mostra-se o protótipo sendo encerrado.

Figura 21 – Tela sair do protótipo



E com isto, o músico finaliza o uso do protótipo e tira a sua dúvida perante a uma nota musical previamente gravada em um arquivo de áudio digital.

5 CONCLUSÕES

Como vários protótipos são desenvolvidos em diversas universidades, o *Kaco Acordes* teve o seu início de desenvolvimento em setembro de 2002 na Universidade Regional de Blumenau. Tendo em vista a facilidade, utilizou-se algumas ferramentas de apoio para a realização do protótipo, como exemplo, o *MatLab 6.0*.

O *Kaco Acordes* foi criado e desenvolvido para poder fazer o reconhecimento das notas musicais tocadas no acordeão e mostra ao músico qual nota musical foi tocada. O protótipo desenvolvido ainda sofrerá mudanças de melhorias e estudos mais aprofundados e abrangerá a área da computação musical, sendo que protótipo é um protótipo, e não um produto final pronto para ser utilizado profissionalmente.

No decorrer da implementação, encontrou-se um problema para o reconhecimento seqüencial de notas musicais. Em outras palavras, tem-se o processamento de um arquivo *wave* composto de duas ou mais notas musicais gravadas em uma seqüência desejada, como por exemplo: Dó, Ré Mi e Fá. Tendo-se um arquivo com quatro notas musicais seqüenciais, o seu funcionamento dava-se da seguinte forma: a primeira nota era reconhecida com sucesso; na segunda nota, o processo não retornava o resultado esperado e assim respectivamente com as notas musicais seguintes. Pelo que se concluiu, o processo da transformada de Fourier se perde na zona de silêncio, ou seja, no intervalo de uma nota para a outra.

O algoritmo para implementação da transformada de Fourier voltado para a conversão do sinal da amplitude no domínio do tempo, para o domínio da freqüência. Assim, naturalmente esse método foi abordado, pois conforme mencionado na seção 2.4.2, a freqüência de um sinal sonoro corresponde à altura de uma nota musical.

Uma das conclusões evidenciadas neste trabalho é o fato de que a música é uma ciência exata, onde se podem empregar conceitos matemáticos como atributos lógicos, aritméticos (tais como média ponderada), para se obter um valor numérico, cuja interpretação remete à uma nota musical.

5.1 EXTENSÕES

Para complementar este trabalho, sugere-se a implementação de uma interface visual da parte dos baixos de um acordeão na qual se poderia fazer o reconhecimento das notas musicais que são tocadas em seus baixos (por exemplo, o desenho completo de um acordeão, tendo o teclado, o foles e a parte dos baixos). Isto possibilitaria ao músico avaliar quais são as notas do baixo que combinam com as notas tocadas em seu teclado.

Poderia ser implementado um protótipo que, ao invés de reconhecer notas musicais monofônicas, reconhecesse intervalos melódicos e harmônicos tais como terças, quartas, sextas, ou até oitavas. Indo mais além, o autor acredita na relevância de se investigar a possibilidade de reconhecimento de acordes, que são compostos por três ou mais notas musicais.

Outra possível sugestão seria a implementação de uma ferramenta com o intuito de se fazer a calibragem de um instrumento musical. Onde o músico ou um mecânico de instrumentos musicais, checaria usaria o reconhecimento de notas via *software* para avaliar e ajustar a afinação do instrumento. O autor acredita que esta idéia pode ser extendida para uso em plataformas do tipo PDAs, tais como os *Palmtop's*, *PocketPC* e outros.

REFERÊNCIAS BIBLIOGRÁFICAS

BRUNS, Fábio Augusto. **Protótipo para o reconhecimento de palavras através da fala.** 1995. 68 f. Trabalho de conclusão de curso (Bacharel em Ciências da Computação) – Centro de Ciências Exata e Naturais, Universidade Regional de Blumenau, Blumenau.

CANTÚ, Marco. **Dominando o Delphi 5.** a bíblia**.** São Paulo: Makron Books, 2000.

CHEDIAK, Almir. **Harmonia e Improvisação.** Rio de Janeiro: Lumiar Editora. 1986.

CODY, Mac A. The fast wavelet transform. **Dr. Dobb's Journal,** p. 16-28, abr 1992.

MATLAB. **Versão do estudante**. guia do usuário**.** Tutorial escrito por Duane Hanselman e Bruce Littlefield; Tradução Hercules Pereira Neves. São Paulo: Makron Books, 1997.

MEIRELLES, Rodrigo. **Formato Wav**. Microsoft wave file, Rio de Janeiro, [2002?]. Disponível em: <http://www.lps.ufrj.br/~meirelles>. Acesso em 24 mar. 2003.

PAULA FILHO, Wilson de Pádua. **Multimídia**: conceitos e aplicações. Rio de Janeiro: LTC, c2000. xv, 321p.

PULSE-CODE modulation. **Integrated publishing**, Spring, [2002]. Disponível em: <http://www.tpub.com/neets/book12/491.htm>. Acesso em: 21 maio 2003.

RIBEIRO, Wagner. **Elementos da teoria da música.** São Paulo: F.T.D. Ltda, 1965.

SANTOS, Gilmário Barbosa dos. **A transformada de Fourier**, Joinville, [2002?]. Disponível em: <http://www.joinville.udesc.br/processamentodeimagens/fourier.html>. Acesso em: 06 fev. 2003.

TAGLIARI, Marilan R. **Ferramenta de apoio ao mapeamento de especificações estruturada para a especificação orientada a objeto.** 2002. 72 f. Trabalho de conclusão de curso (Bacharel em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

TUCKER, R. **Voice activity detection using a periodicity measure.** In IEEE Proceeding, v. 139, n. 4, ago 1992.

ANEXO 1 – FREQÜÊNCIA E CÓDIGO MIDI PARA CADA NOTA MUSICAL

O Quadro 24 – Freqüência e código MIDI de cada nota musical mostra os valores de freqüências e códigos MIDI para cada nota musical contida em um teclado de acordeão.

Quadro 24 – Freqüência e código MIDI de cada nota musical

Nota	Freqüência (Hz)	Comprimento da onda (cm)	Midi	Nota definida
F_3	174.61	198.	53	FÁ
$F^{\#}_3/G^{b}_3$	185.00	186.	54	FÁ# / SOLb
G_3	196.00	176.	55	SOL
$G^{\#}_3/A^{b}_3$	207.65	166.	56	SOL# / Láb
A_3	220.00	157.	57	LÁ
$A^{\#}_3/B^{b}_3$	233.08	148.	58	LA# / Sib
B_3	246.94	140.	59	SI
C_4	261.63	132.	60	DÓ
$C^{\#}_4/D^{b}_4$	277.18	124.	61	DÓ# / Réb
D_4	293.66	117.	62	RÉ
$D^{\#}_4/E^{b}_4$	311.13	111.	63	RÉ# / Mib
E_4	329.63	105.	64	MI
F_4	349.23	98.8	65	FÁ
$F^{\#}_4/G^{b}_4$	369.99	93.2	66	FÁ# / SOLb
G_4	392.00	88.0	67	SOL
$G^{\#}_4/A^{b}_4$	415.30	83.1	68	SOL# / Láb
A_4	440.00	78.4	69	LÁ
$A^{\#}_4/B^{b}_4$	466.16	74.0	70	LA# / Sib
B_4	493.88	69.9	71	SI

C_5	523.25	65.9	72	DÓ
$C^{\#}_5/D^{b}_5$	554.37	62.2	73	DÓ# / Réb
D_5	587.33	58.7	74	RÉ
$D^{\#}_5/E^{b}_5$	622.25	55.4	75	RÉ# / Mib
E_5	659.26	52.3	76	MI
F_5	698.46	49.4	77	FÁ
$F^{\#}_5/G^{b}_5$	739.99	46.6	78	FÁ# / SOLb
G_5	783.99	44.0	79	SOL
$G^{\#}_5/A^{b}_5$	830.61	41.5	80	SOL# / Láb
A_5	880.00	39.2	81	LÁ
$A^{\#}_5/B^{b}_5$	932.33	37.0	82	LA# / Sib
B_5	987.77	34.9	83	SI
C_6	1046.50	33.0	84	DÓ
$C^{\#}_6/D^{b}_6$	1108.73	31.1	85	DÓ# / Réb
D_6	1174.66	29.4	86	RÉ
$D^{\#}_6/E^{b}_6$	1244.51	27.7	87	RÉ# / Mib
E_6	1318.51	26.2	88	MI
F_6	1396.91	24.7	89	FÁ
$F^{\#}_6/G^{b}_6$	1479.98	23.3	90	FÁ# / SOLb
G_6	1567.98	22.0	91	SOL
$G^{\#}_6/A^{b}_6$	1661.22	20.8	92	SOL# / Láb
A_6	1760.00	19.6	93	LÁ