unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

EDUARDO GOMES LAMAS OTERO

DESENVOLVIMENTO DE SOFTWARE PARA EXIBIÇÃO DE EXAMES DE ELETROCARDIOGRAMA

Guaratinguetá
2012

EDUARDO GOMES LAMAS OTERO

DESENVOLVIMENTO DE SOFTWARE PARA EXIBIÇÃO DE EXAMES DE ELETROCARDIOGRAMA

Trabalho de graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador: Prof. Dr. Samuel Euzédice de Lucena

Guaratinguetá
2012

Otero, Eduardo Gomes Lamas
O874d Desenvolvimento de software para exibição de exames de eletrocardiograma / Eduardo Gomes Lamas Otero – Guaratinguetá : [s.n], 2012.
55 f : il.
Bibliografia: f. 49

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2012.
Orientador: Prof. Dr. Samuel Euzédice de Lucena

1. Processamento de sinais I. Título

CDU 621.381

unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

DESENVOLVIMENTO DE SOFTWARE PARA EXIBIÇÃO DE EXAMES DE ELETROCARDIOGRAMA

EDUARDO GOMES LAMAS OTERO

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE **"GRADUAÇÃO EM ENGENHARIA ELÉTRICA"**

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Prof. Dr. SAMUEL EUZÉDICE DE LUCENA
Coordenador

BANCA EXAMINADORA:

Prof. Dr. SAMUEL EUZÉDICE DE LUCENA
Orientador/UNESP-FEG

Prof. Dr. DANIEL JULIEN B. DA S. SAMPAIO
UNESP-FEG

Prof. M.Sc. FERNANDO RIBEIRO FILADELFO
UNESP-FEG

Novembro de 2012

à minha família, amigos e professores. Em especial àqueles que fazem parte de dois ou mais destes grupos ao mesmo tempo.

OTERO, E.G.L. **Desenvolvimento de software para exibição de exames de eletrocardiograma**, 2012. 55 pg. Trabalho de Graduação (Graduação em Engenharia Elétrica) – Faculdade de Engenharia do campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2012.

RESUMO

Este trabalho trata do desenvolvimento de um software, em linguagem C++, para leitura e representação gráfica de arquivos contendo informações referentes a exames de Eletrocardiograma (ECG). A informação de interesse está gravada em um sinal modulado em FM e comprimido por um algoritmo comum em telecomunicações (o ADPCM). O sinal de ECG tem essas características por ter sido gerado por um equipamento experimental, fruto de pesquisa realizada pelo docente orientador deste trabalho.

No desenvolvimento são abordadas técnicas de demodulação FM em tempo discreto, filtros discretos e processamento digital de sinais de um modo geral.

Também são abordados, de forma mais breve, tópicos necessários para o reconhecimento básico de um ECG e suas características.

O desenvolvimento do software se dá em paralelo com simulações no MATLAB, com a finalidade de ilustrar as ferramentas utilizadas e os passos necessários para a recuperação do sinal de ECG.

Por fim é feita uma avaliação geral do software desenvolvido e futuros desenvolvimentos são apontados para melhorar o desempenho e corrigir defeitos do programa.

PALAVRAS-CHAVE: Processamento de sinais, modulação FM, RIFF, formato WAV, formato WAVE, filtros digitais.

OTERO, E.G.L. **Development of Software for Reading Electrocardiogram Exams**, 2012. 55 pg. Graduation Work (Electrical Engineering) – Faculdade de Engenharia do campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2012.

ABSTRACT

The main objective is to create a software, using C++ language, for reading and exhibiting in a graphic an Electrocardiogram (ECG) wave. The data is recorded as a FM modulated signal and compressed using Adaptive Differential Pulse Code Modulation (ADPCM).

The signal have this characteristics because it was acquired using an experimental equipment, this equipment is the result of research made by the professor who supervised this work.

FM demodulation techniques in discrete time, discrete filters and digital signal processing are some of the topics that can be found in this essay.

Some concepts about the human heart and about ECG waves are also briefly introduced. These concepts are necessary for understanding the final evaluation of the software performance.

The development is partly made using MATLAB. Most of the functions that are used on the software are first tested and designed in MATLAB environment.

In the end, an evaluation is done comparing the results that are expected with the ones that MATLAB presents and the ones that the developed software presents.

.

KEYWORDS: Digital Signal Processing, FM demodulation, RIFF, WAV, WAVE, Digital Filters.

SUMÁRIO

1. INTRODUÇÃO

Neste trabalho são estudados conceitos de processamento digital de sinais, modulação e demodulação FM e também conceitos relacionados ao exame de eletrocardiograma (ECG). Este último, porém, de maneira bastante superficial.

Segundo Lucena (2008) é possível a utilização de aparelhos portáteis tocadores de MP3 para fins de registro de dados. A primeira dificuldade que se encontra para tal uso é o algoritmo interno do tocador de MP3, que não permite a gravação direta dos dados, uma vez que o mesmo apenas permite gravação de sinais de voz.

Para contornar essa situação, o sinal que se quer registrar pode ser, por exemplo, modulado em FM e depois gravado normalmente pelo tocador de MP3, pois o algoritmo do aparelho interpreta o sinal FM como áudio convencional e realiza a gravação normalmente.

A fim de dar continuidade a essa proposta, foi desenvolvido um software para demodular tal sinal FM e exibir de forma gráfica os dados registrados.

O sinal utilizado como estudo de caso consiste em um exame de ECG de um voluntário. Esse sinal foi registrado utilizando o procedimento citado anteriormente e o objetivo final é recuperar os dados e traçar o ECG do voluntário em uma interface gráfica.

Inicialmente, é feita uma revisão dos tópicos necessários para o desenvolvimento do software. Em uma segunda etapa, simulações utilizando MATLAB são realizadas para aplicar a teoria apresentada e finalmente demodular o sinal que contém o ECG.

O desenvolvimento do software se dá em paralelo com as simulações no MATLAB, implementando por etapas o algoritmo que é utilizado para realizar a demodulação.

Por fim, os resultados obtidos são analisados e propostas de possíveis melhorias são feitas.

2. O CORAÇÃO E O ELETROCARDIOGRAMA

O coração é um dos órgãos mais importantes do corpo humano sendo responsável pelo bombeamento do sangue. No coração podemos observar a existência de quatro câmaras, duas superiores (átrio direito e átrio esquerdo) e duas inferiores (ventrículo direito e ventrículo esquerdo).

O sangue que vem dos pulmões, rico em oxigênio, circula no interior das câmaras do lado esquerdo enquanto o sangue que vêm do resto do corpo circula nas câmaras do lado direito.

A figura (1) ilustra as quatro câmaras e o fluxo do sangue dentro do coração. O sangue vem do corpo pelas veias cava superior (1) e inferior (2) e flui para o átrio direito (3). Do átrio direito, o sangue é bombeado para o ventrículo direito (4). A partir do ventrículo direito o sangue é bombeado para os pulmões, onde ocorre a oxigenação. Dos pulmões o sangue segue para o átrio esquerdo (5) e daí para o ventrículo esquerdo (6). O ventrículo esquerdo é responsável por bombear o sangue para todo o corpo e por isso é mais forte do que as outras câmaras, possuindo paredes de maior espessura (anotações de aula)[1].

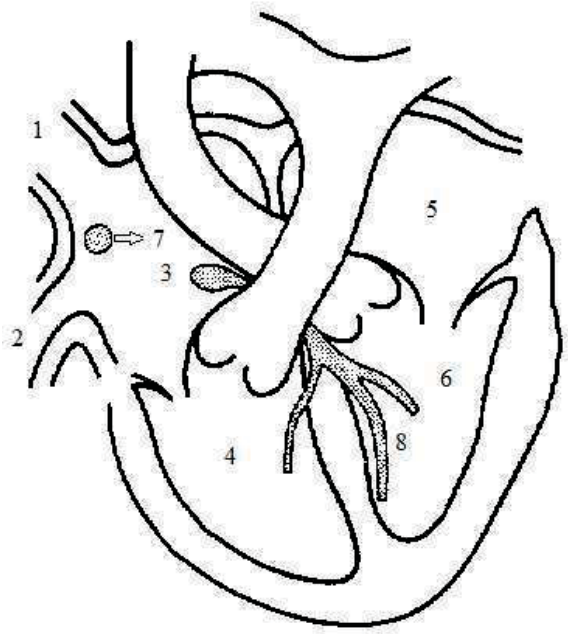

Figura 1 – Anatomia básica do coração humano: (1) Veia cava superior, (2) Veia cava inferior, (3) Átrio direito, (4) Ventrículo direito, (5) Átrio esquerdo, (6) Ventrículo esquerdo, (7) Nódulo sinoatrial, (8) Nódulo Atrioventricular (figura adaptada de Höpfel, 2011 – anotações de aula).

[1] Anotações de aula da disciplina Medical Sensors, 2011 (prof. Dr. Dieter Höpfel, Hochschule Karlsruhe)

Os nódulos sinoatrial e atrioventricular são parte do sistema que gera os pulsos elétricos para o funcionamento do coração. Por meio de um complexo arranjo, células se polarizam gerando estímulos automáticos e sincronizados que controlam os batimentos.

Essa polarização gera um campo elétrico que varia de acordo com as fases do funcionamento do coração. O eletrocardiograma é o exame que permite a análise de algumas características do funcionamento do coração por meio das informações extraídas do comportamento deste campo elétrico.

O coração possui funções bastante especificas como segue:

Automaticidade

Habilidade do coração de gerar impulsos requeridos para excitação cardíaca. Normalmente o nódulo sinoatrial, que fica no átrio direito possui a maior automaticidade e serve como uma fonte de impulsos.

Condutibilidade

A habilidade do coração de transmitir pulsos do lugar onde esses são gerados até o miocárdio contráctil. Normalmente, o impulso se propaga do nódulo sinoatrial para o miocárdio atrial e ventricular.

Excitabilidade

Habilidade do coração de se excitar em resposta a impulsos.

Contratilidade

Capacidade do coração de se contrair em resposta a impulsos e de bombear sangue para os pulmões e para o corpo.

Tonicidade

Habilidade do coração de manter sua forma durante a diástole.

Refratariedade

A inabilidade das células do miocárdio excitado de serem reativadas por um impulso adicional. Assim, a excitabilidade do sistema de condução do coração e do miocárdio contrátil varia em diferentes períodos do ciclo cardíaco, com as células cardíacas sendo refratárias à estimulação durante a sístole e tornando-se capaz de ficarem excitadas novamente durante a diástole.

Condução aberrante

Condução anormal do impulso através do átrio ou ventrículos. Caso em que o impulso chega ao ventrículo ou, mais raramente, aos átrios, quando um ou vários feixes de seu sistema de condução estão ainda em estado de retratilidade, o que resulta numa condução irregular pelas câmaras do coração.

O exame de eletrocardiograma permite a investigação de funções como automaticidade, condutividade, excitabilidade, retratilidade e condução aberrante, mas fornece apenas dados indiretos sobre a contratilidade e nenhuma evidência sobre a tonicidade. (ORLOV, 1988)

Embora existam diversas variações, a forma de onda mais conhecida do ECG é composta de segmentos denominados P, Q, R, S, T e U. A figura 2 ilustra de forma básica uma onda típica de um ECG.

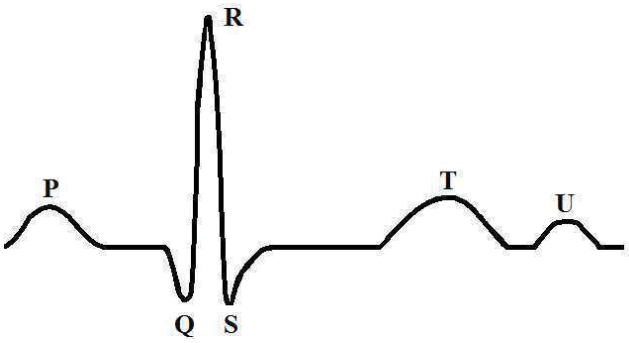

Figura 2: Onda típica de ECG (adaptado de ORLOV, 1988).

A onda denominada P ocorre na despolarização atrial, quando a excitação elétrica se espalha pelos dois átrios. A linha PQ indica uma etapa de transição da excitação indo do átrio para o ventrículo.

O trecho QRS é denominado complexo QRS e indica o espalhamento da excitação em ambos os ventrículos. A linha ST (entre o fim da onda S e o início da onda T) representa outra transição da excitação. A onda T indica a repolarização dos ventrículos.

A onda U não é considerada significante e em alguns casos é bastante reduzida ou ainda ausente (HÖPFEL, 2011).

A descrição da onda típica de um ECG é importante para a análise final, depois da recuperação do sinal. Isso porque é necessário saber se a informação foi realmente recuperada com qualidade ou se existem deficiências a serem sanadas.

3. O FORMATO WAV

O formato WAV é um subconjunto da especificação RIFF (Resource Interchange File Format) da Microsoft. Os arquivos da especificação RIFF são compostos de várias seções discretas chamadas de Chunks.

Um arquivo RIFF consiste em um chunk RIFF que pode conter outros chunks no seu interior. O chunk RIFF tem a seguinte forma:

"RIFF", fileSize, fileType, data

onde RIFF consiste nos caracteres R, I, F e F em ASCII, fileSize é um valor de 4 bytes que identifica o tamanho de data. O valor de fileSize diz o tamanho do arquivo em bytes, excluindo os bytes ocupados por RIFF e pelo próprio fileSize. Por último, data consiste em outros chunks quaisquer.

Os outros chunks contidos em RIFF tem a seguinte forma:

chunkID, chunkSize, data

onde chunkID é um FOURCC (Four-character code) que identifica o tipo de dado contido no chunk. O trecho chunkSize é um valor de 4 bytes que dá o tamanho do trecho data, não incluindo os bytes ocupados por chunkID ou pelo próprio chunkSize (MICROSOFT – 2012).

Por ser um padrão bastante difundido é possível encontrar diversas variações no formato básico de um arquivo WAV. De forma geral podemos descrever um arquivo WAV por sua forma canônica como na tabela 1.

A descrição da forma canônica é uma tradução livre de CCRMA (2003). Essa descrição se baseia principalmente no projeto SoX (vide referências - SOX).

Endian	Offset (bytes)	nome do campo	Tamanho do campo
big	0	ChunkID	4
little	4	ChunkSize	4
big	8	Format	4
big	12	Subchunk1ID	4
little	16	Subchunk1Size	4
little	20	AudioFormat	2
little	22	NumChannels	2
little	24	SampleRate	4
little	28	ByteRate	4
little	32	BlockAlign	2
little	34	BitsPerSample	2
big	36	Subchunk2ID	4
little	40	Subchunk2Size	4
little	44	data	Depende de data

Tabela 1: Forma canônica de um arquivo tipo WAV

Os três primeiros campos formam a descrição básica do arquivo RIFF, contendo nome do chunk (RIFF=0x52494646), tamanho (descreve o tamanho do resto do arquivo) e Format (WAVE=0x57415645).

O formato WAV consiste de dois subchunks, fmt e data. Os oito próximos campos da tabela definem fmt. Como segue:

Subchunk1ID: deve conter as letras ‘fmt’ (0x666d7420)

Subchunk1Size: tamanho do resto do chunk (soma dos tamanhos dos outros chunks seguintes, para PCM deve ser 16).

AudioFormat: Para PCM o valor é 1. Se o valor for diferente, indica alguma forma de compressão.

NumChannels: Número de canais, literalmente. Um para mono, dois para estéreo e assim sucessivamente.

SampleRate: taxa de amostragem.

ByteRate: taxa de amostragem multiplicada pelo número de canais e pelo número de bits por amostra dividido por oito.

BlockAlign: número de canais multiplicado por bits por amostra dividido por oito.

BitsPerSample: número de bits por amostra.

Os três últimos campos da tabela anterior definem o chunk data. Neste chunk os dados do arquivo WAV são gravados de acordo com as especificações definidas nos outros chunks. É possível definir os campos restantes como:

Subchunk2ID: deve conter as letras ‘data’ (0x64617461)

Subchunk2Size: número de bytes no trecho data. Pode ser descrito como o número de amostras multiplicado pelo número de canais e pelo número de bits por amostra dividido por 8 (sendo assim, dá o total de bytes no trecho data).

Data: Contêm os dados do arquivo no formato descrito pelos chunks anteriores.

No capítulo 5. Desenvolvimento, item 5.1. Arquivos WAV, são dados exemplos de tratamento de arquivos formato WAV, necessários para o entendimento dos algoritmos desenvolvidos. Esse tratamento não é abordado aqui por conveniência e para evitar redundâncias.

3.1. Compressão ADPCM

Arquivos WAV no padrão PCM podem ocupar bastante espaço dentro da memória, pois contém a informação completa, ponto a ponto, do áudio amostrado (dentro dos limites impostos pelo processo de amostragem). Também em sistemas de comunicação, a transmissão de dados PCM é custosa e é bastante comum e difundida a transmissão utilizando dados comprimidos.

ADPCM (Adaptive Differential Pulse Code Modulation) é um algoritmo de compressão com o qual é possível alcançar altas taxas de compressão com relação ao PCM. Esse tipo de compressão remove redundâncias desnecessárias para a transmissão dos dados, tornando a comunicação mais rápida ou diminuindo a banda ocupada (GIBSON, 1980).

Ao invés de gravar o dado como um todo se utiliza uma técnica para prever o valor da próxima amostra baseado nas amostras anteriores e computa-se apenas a diferença entre a amostra real e a amostra prevista.

A diferença entre o valor previsto e o valor real ocupa uma quantidade menor de memória, se a predição for adequada, e assim é realizada a compressão do arquivo.

A União Internacional de Telecomunicações (ITU – International Telecommunication Union) por meio do CCITT (International Telegraph and Telephone Consultative Committee) estabelece o padrão para compressão ADPCM na recomendação G.726.

Na recomendação G.726 são estabelecidos uma série de parâmetros para padronizar a compressão utilizando ADPCM. A seguir, a compressão ADPCM será explorada, de acordo com essa recomendação.

Os tópicos seguintes abordam o codificador e o decodificador ADPCM de forma sucinta, para maiores informações consultar a recomendação G. 726. As informações seguintes foram traduzidas e adaptadas diretamente da recomendação G. 726.

3.1.1. O Codificador ADPCM

O diagrama de blocos simplificado do codificador ADPCM pode ser visto na Figura 3:

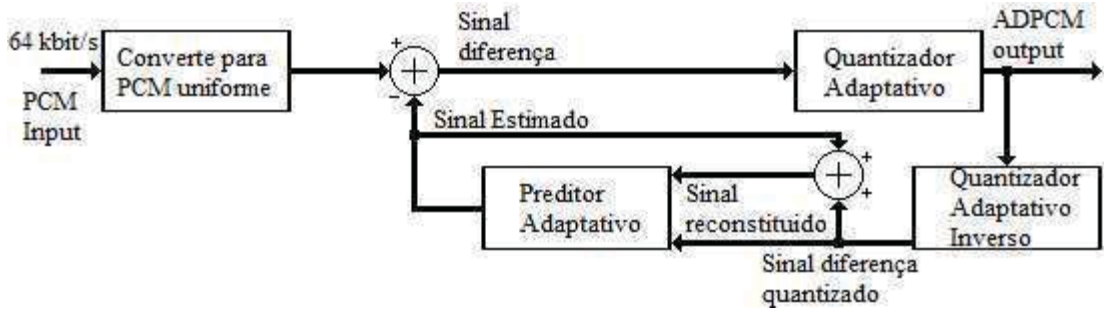

Figura 3 – Diagrama de blocos simplificado do codificador ADPCM (adaptado de G. 726 – ITU).

Após a conversão para PCM uniforme, subtrai-se o sinal de entrada do Sinal Estimado (gerado com base nas amostras anteriores). Um Quantizador Adaptativo de 31, 15, 7 ou 4 níveis é utilizado para quantificar essa diferença em 5, 4, 3 ou 2 dígitos binários respectivamente. O Sinal Estimado é então somado ao Sinal Diferença Quantizado para produzir a versão reconstituída do sinal de entrada. Tanto o sinal reconstituído quanto a diferença quantizada são inseridos no Preditor Adaptativo que produz a estimativa do sinal de entrada, fechando o ciclo.

3.1.2. O Decodificador ADPCM

O diagrama de blocos simplificado do decodificador ADPCM pode ser visto na Figura 4:

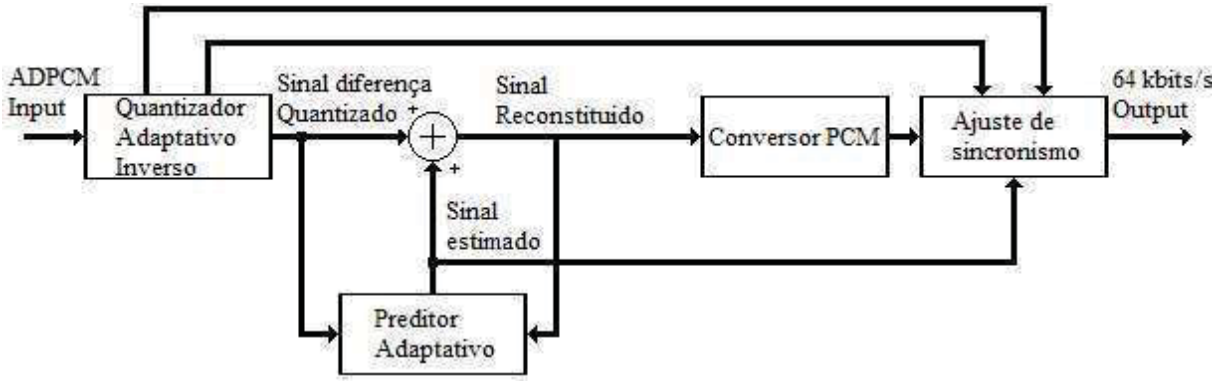

Figura 4 – Diagrama de blocos simplificado do decodificador ADPCM (adaptado de G. 726 – ITU).

O decodificador possui uma estrutura de realimentação idêntica à do codificador, um conversor de PCM uniforme para lei A ou lei μ e um bloco de sincronismo.

4. MODULAÇÃO ANGULAR

O princípio fundamental da modulação angular está em modificar a fase ou a frequência de uma onda portadora de acordo com as variações em um sinal modulante.

A modulação angular oferece, a troco de uma maior faixa de espectro necessária, além de maior complexidade dos sinais e dos circuitos envolvidos, uma melhor relação sinal ruído (com relação à modulação em amplitude).

Neste item é feita uma breve introdução sobre modulação angular, com ênfase em modulação FM e algumas informações sobre métodos de demodulação FM.

A análise detalhada da modulação angular como um todo é uma questão ampla e foge do escopo deste trabalho. Para maiores detalhes sobre circuitos moduladores e demoduladores, transmissão FM e tópicos específicos sobre telecomunicações, as referências citadas são uma boa fonte de informações (consultar Haykin, 2001).

4.1. Conceitos iniciais

Esta seção foi baseada no capítulo 2 de Communication Systems, quarta edição, Haykin (2001).

Seja a função $s(t)$:

$$s(t) = A_c * \cos[\theta_i(t)] \quad (1)$$

o sinal em função do tempo da onda modulada por $\theta_i(t)$ e supondo que $\theta_i(t)$ varia linearmente em um pequeno intervalo de tempo, a frequência média, em Hz, no intervalo que vai de t até $t + \Delta t$ será:

$$f_{média}(t) = \frac{\theta_i(t+\Delta t) - \theta_i(t)}{2*\pi*\Delta t} \quad (2)$$

fazendo Δt tender a zero, é possível escrever a frequência instantânea:

$$f_i = \lim_{\Delta t \to 0} \frac{\theta_i(t+\Delta t) - \theta_i(t)}{2*\pi*\Delta t} \quad (3)$$

O que leva a:

$$f_i = \frac{1}{2*\pi} * \frac{d\theta_i(t)}{dt} \quad (4)$$

que é a frequência instantânea do sinal $s(t)$. No caso de o sinal possuir uma frequência constante tem-se:

$$\theta_i(t) = 2 * \pi * f_c * t + k \quad (5)$$

onde k é o valor de $\theta_i(t)$ quando o tempo é igual a zero.

A ideia central da modulação angular é variar o ângulo $\theta_i(t)$ de forma proporcional à variação do sinal que se quer transmitir. As formas mais comuns de se variar $\theta_i(t)$ são conhecidas por modulação em fase (PM) e modulação em frequência (FM).

Na modulação em fase, a fase do sinal da portadora é modulada de acordo com as informações contidas no sinal modulante. Pode-se escrever o ângulo instantâneo como:

$$\theta_i(t) = 2 * \pi * f_c * t + k_p * m(t) \quad (6)$$

onde f_c é a frequência da onda portadora, k_p é a sensibilidade à fase do modulador e $m(t)$ é o sinal modulante. Assim sendo, o sinal modulado em função do tempo é dado pela equação (7).

$$s(t) = A_c * \cos[2 * \pi * f_c * t + k_p * m(t)] \quad (7)$$

Na modulação FM a frequência do sinal modulado varia de acordo com o sinal modulante, é possível então escrever:

$$f_i(t) = f_c + k_f * m(t) \quad (8)$$

onde k_f é a sensibilidade à frequência do modulador e $m(t)$ é o sinal modulante. A partir da equação (4), que relaciona a frequência instantânea ao ângulo instantâneo, pode-se escrever:

$$\frac{d\theta_i(t)}{dt} = 2 * \pi * (f_c + k_f * m(t)) \quad (9)$$

que leva a:

$$\theta_i(t) = 2 * \pi * f_c * t + 2 * \pi * k_f \int_0^t m(\tau)\, d\tau \quad (10)$$

assumindo que em $t = 0$ o ângulo seja igual a zero.

Portanto, o sinal modulado em função do tempo poderá ser descrito como na equação (11).

$$s(t) = A_c * \cos[2 * \pi * f_c * t + 2 * \pi * k_f \int_0^t m(\tau)\, d\tau] \quad (11)$$

Observando a figura (5) a seguir, é possível fazer uma comparação entre PM e FM.

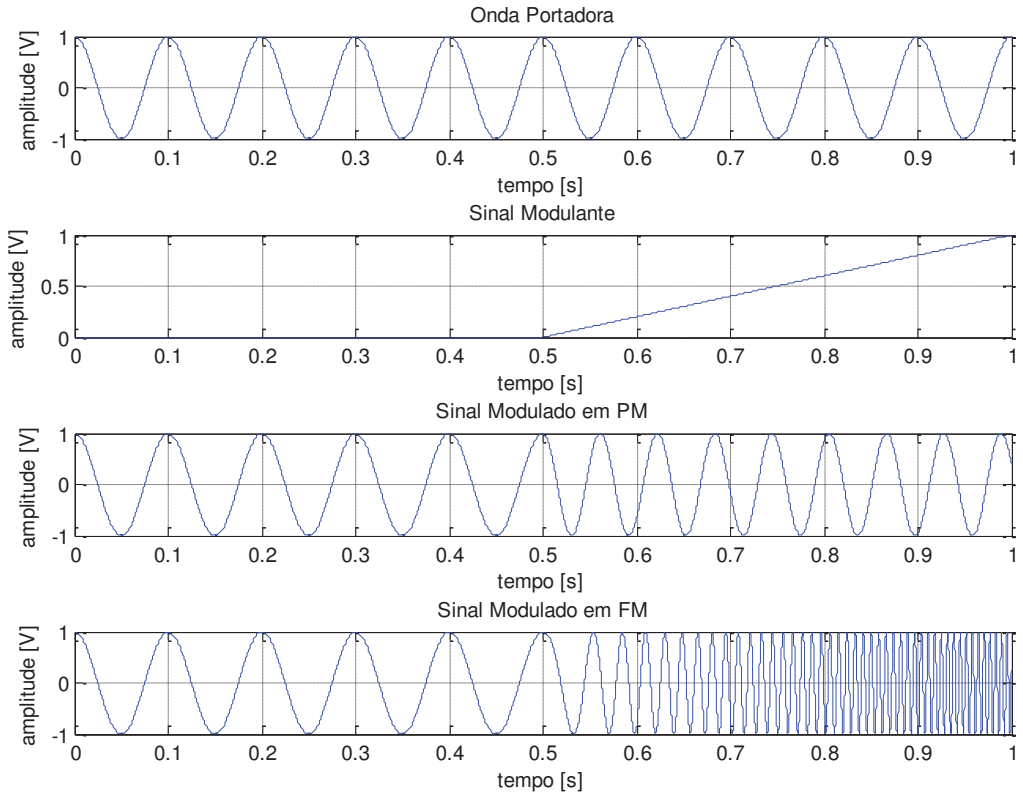

Figura 5 – Comparativa entre modulação PM e FM (O autor).

Nota-se claramente que na modulação FM a frequência aumenta conforme o sinal modulante em rampa cresce. É possível notar também que a frequência do sinal PM é proporcional à taxa de variação do sinal modulante. As equações (10) e (11) deixam clara esta relação.

Portanto, um sinal FM pode ser gerado integrando-se $m(t)$ e depois modulando-se em fase e um sinal PM pode ser gerado derivando $m(t)$ e realizando uma modulação em frequência.

4.2. Demodulação FM em tempo discreto

A demodulação FM consiste na extração das informações contidas no sinal modulado $s(t)$. Para extrair estas informações, deve-se determinar a frequência instantânea do sinal.

Diversas são as abordagens possíveis tanto para sinais contínuos quanto para sinais discretos. A seguir temos uma breve introdução a quatro métodos de detecção FM bastante difundidos.

4.2.1. Demodulação FM por diferenciação

A derivada de um sinal resulta na taxa de variação do mesmo. A derivada de um sinal senoidal de frequência constante nada mais é do que um sinal cossenoidal (que é apenas um seno deslocado). Se a frequência desse sinal for modulada (por uma rampa, por exemplo), a taxa de variação do sinal também será alterada de forma proporcional.

Assim pode ser estabelecida uma relação entre a taxa de variação do sinal no tempo com a variação da frequência deste sinal. A figura 6 ilustra essa relação.

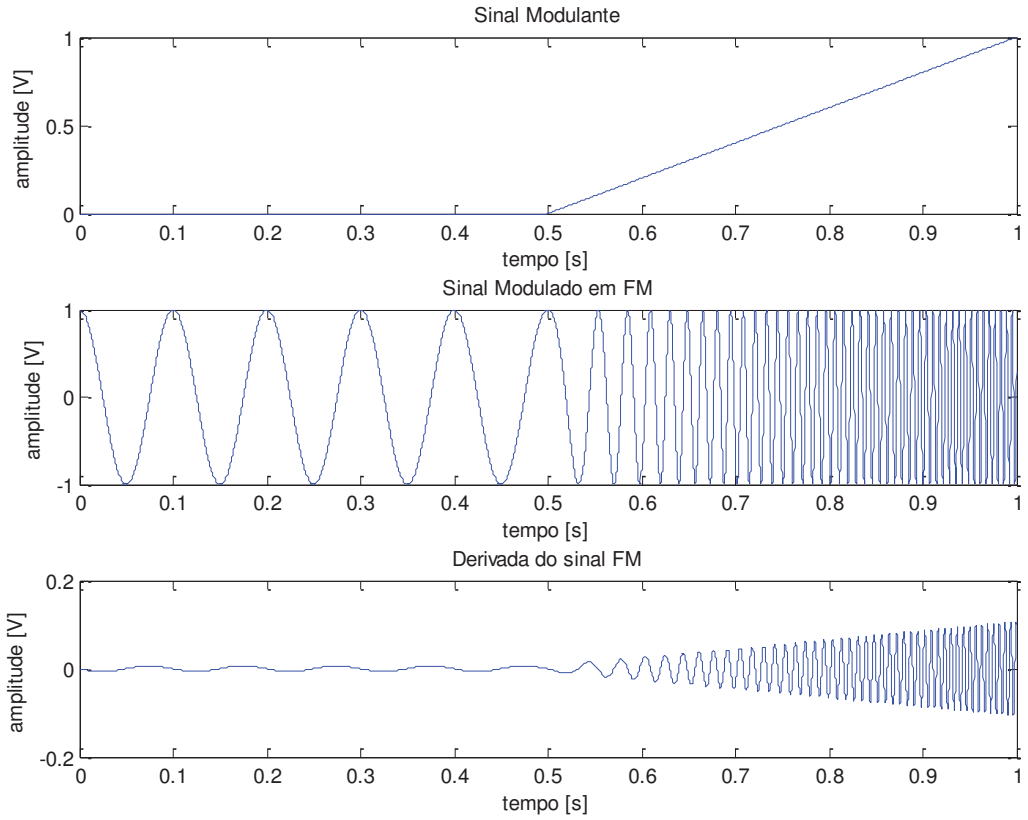

Figura 6: Relação entre a derivada de um sinal FM e o sinal modulante original (O autor).

Derivando equação (11), que representa o sinal modulado em FM em função do tempo, temos:

$$\frac{d[s(t)]}{dt} = \frac{d\{A_c * \cos\left[2 * \pi * f_c * t + 2 * \pi * k_f \int_0^t m(\tau)\, d\tau\right]\}}{dt}$$

Que resulta em:

$$\frac{d[s(t)]}{dt} = A_c\{-\text{sen}\left[2 * \pi * f_c * t + 2 * \pi * k_f \int_0^t m(\tau)\, d\tau\right] * [2\pi f_c + 2\pi k_f m(t)]\} \quad (12)$$

A equação 12 representa um sinal modulado tanto em amplitude quanto em frequência (VASAVADA, 1996). Essa modulação em frequência e em amplitude pode ser observada no nas formas de onda mostradas na da figura 6.

Por fim, aplicando-se um detector de envoltória, é possível recuperar o sinal modulante.

4.2.2. Demodulação FM por quadratura

Detecção por quadratura é um dos métodos mais populares de demodulação FM (VASAVADA, 1996). O diagrama de blocos de um demodulador FM por quadratura é exibido na figura 7.

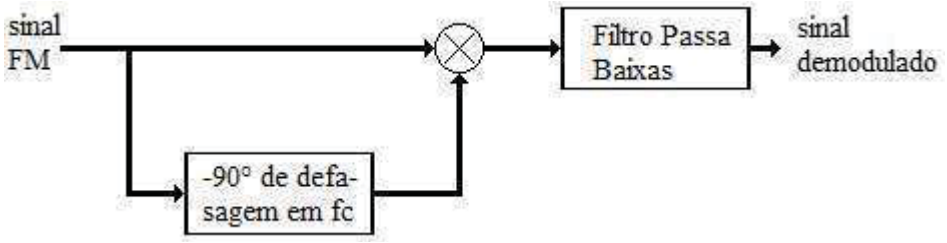

Figura 7 – Diagrama de blocos de demodulação por quadratura.

O bloco de defasamento realiza uma defasagem proporcional à frequência instantânea do sinal FM. O sinal defasado é então multiplicado pelo sinal FM e o resultado da multiplicação passa por um filtro passa baixa, recuperando a informação.

4.2.3. Demodulação FM por detector de inclinação

Utiliza se um circuito com resposta em rampa para transformar variações de frequência em variações de amplitude. É o princípio utilizado no detector de Foster-Seeley (GOMES, 2002).

Considerando um filtro passa faixa com características semelhantes ao ilustrado na figura 8, e um sinal FM com frequência variando entre 800 e 1600 Hz, é possível converter as variações de frequência em variações de amplitude.

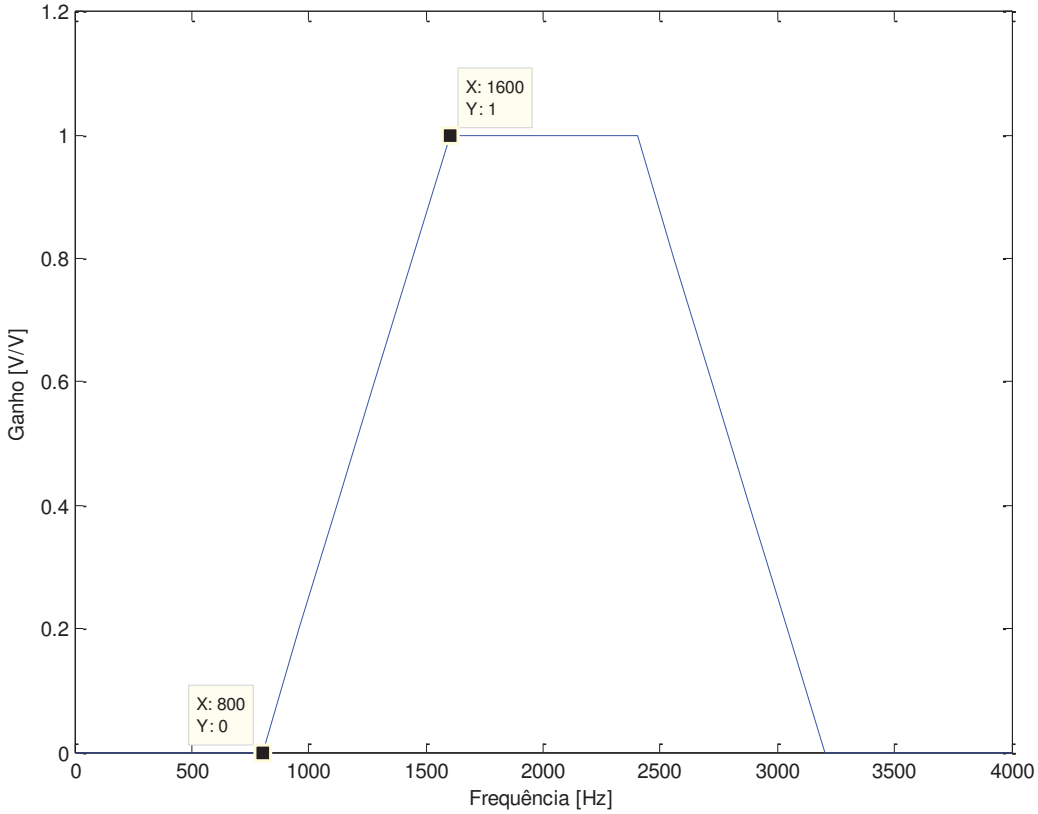

Figura 8 – Filtro passa faixa ideal para aplicação como detector de inclinação (O autor).

Após a aplicação do filtro, tem-se um sinal modulado em frequência e amplitude. A informação final pode ser recuperada com o uso de um detector de envoltória.

4.2.4. Demodulação FM por detecção de cruzamento de zero

É evidente a relação entre a frequência do sinal e o número de vezes que o cruzamento de zero ocorre. Para frequências de portadora relativamente altas, é possível utilizar a aproximação de que a frequência instantânea varia pouco a cada período.

Partindo dessa aproximação, mede-se a frequência instantânea do sinal a cada período computando os cruzamentos com zero.

5. DESENVOLVIMENTO

O desenvolvimento e os estudos relativos ao software para exibir o ECG foi dividido em três etapas.

Primeiramente, foi desenvolvido um software capaz de processar arquivos WAV de forma geral, lidando com todos os cabeçalhos e extraindo os dados do arquivo para posterior descompressão.

Após a constatação do funcionamento do software, partiu-se para a etapa de descompressão. Um algoritmo modificado de um decodificador ADPCM foi implementado para realizar a recuperação dos dados comprimidos, permitindo posterior filtragem e processamento.

Finalmente, são desenvolvidos os filtros e o processo de demodulação FM. Ao final é feita uma análise dos resultados obtidos tanto no ambiente MATLAB quando em C++.

5.1. Arquivos WAV

Neste tópico são apresentados os resultados obtidos quanto à abertura e representação gráfica de arquivos tipo WAV.

A abertura e processamento de um arquivo WAV sem compressão é relativamente simples. A leitura do arquivo é feita caractere por caractere, salvando dados relevantes ao processamento em vetores com tamanhos pré-determinados.

Após a aquisição dos dados relevantes ao processamento, o subchunk contendo os dados é armazenado em um vetor cuja dimensão é determinada pelos cabeçalhos anteriores.

As informações contidas nos cabeçalhos são utilizadas para determinar o procedimento de processamento dos dados. Tomando como exemplo um arquivo WAV com taxa de amostragem de 8000 bits por segundo e 8 bits por amostra, tem-se:

Carácter em hexadecimal	**ASCII ou convertido para decimal**	**Descrição**
0x52 0x49 0x46 0x46	R I F F	Indica tipo RIFF
0x3C 0x1F 0x00 0x00	78*1 + 31*256 + 0 + 0 = 8014	Tamanho total em bytes
0x57 0x41 0x56 0x45	W A V E	Indica arquivo WAVE
0x66 0x6D 0x74 0x20	f m t space	Início do trecho format
0x10 0x00 0x00 0x00	16	Tamanho do trecho format
0x01 0x00	PCM	Indica PCM sem compressão
0x01 0x00	Nro de canais	Número de canais
0x40 0x1F 0x00 0x00	64*1 + 31*256 + 0 + 0 = 8000	Taxa de amostragem
0x40 0x1F 0x00 0x00	64*1 + 31*256 + 0 + 0 = 8000	Byte rate
0x01 0x00	Block align = 1	= n.° de canais*bits por amostra/8
0x08 0x00	Bits per sample	Número de bits por amostra
0x64 0x61 0x74 0x61	d a t a	Indica início do trecho data.

Tabela 2: Exemplo de cabeçalho de arquivo WAV.

Após a leitura e armazenamento dos dados de cabeçalho, as amostras podem ser decodificadas. Para o exemplo anterior, a decodificação de um trecho de amostras tem a seguinte forma:

Hexadecimal	Decimal	Normalizado
0x56	86	(86-128)/127= **-0,331**
0x57	87	(87-128)/127= **-0,323**
0x58	88	(88-128)/127= **-0,314**
0x5A	89	(89-128)/127= **-0,307**

Tabela 3: Exemplo de dados contidos em um arquivo WAV.

O inverso da taxa de amostragem nos dá o valor do intervalo de tempo entre cada uma das amostras em segundos. Assim é possível traçar o gráfico equivalente aos dados armazenados no arquivo em questão.

A fim de testar a eficiência do programa desenvolvido, formas de onda foram geradas por um software comercial (GoldWave, versão de teste) e então processadas pelo programa desenvolvido neste trabalho. A partir de uma inspeção visual é possível determinar se o programa atende o objetivo de ilustrar graficamente as formas de onda sem distorções significativas.

A seguir são ilustrados os resultados obtidos com o processamento dos arquivos sintetizados. Os arquivos possuem taxa de amostragem de 8000 bits por segundo, o tamanho de cada amostra é de 8 bits e a duração total do arquivo é de um segundo. Todos os arquivos ilustrados possuem apenas um canal (mono).

A figura 9 ilustra graficamente um arquivo tipo WAV contendo um impulso unitário. Visualmente, o gráfico não apresenta nenhum tipo de distorção.

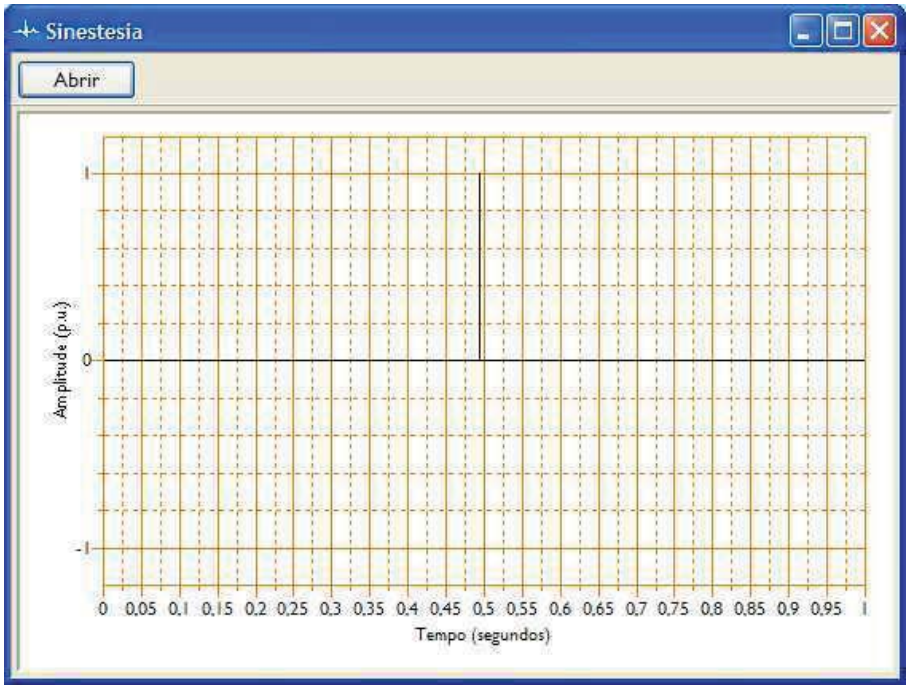

Figura 9 – Representação gráfica de um impulso unitário no software desenvolvido (O autor).

A figura 10 apresenta a resposta do software quando se plota um degrau unitário de 0,5 segundos de duração. Mais uma vez, nota-se que não ocorrem distorções visualmente perceptíveis.

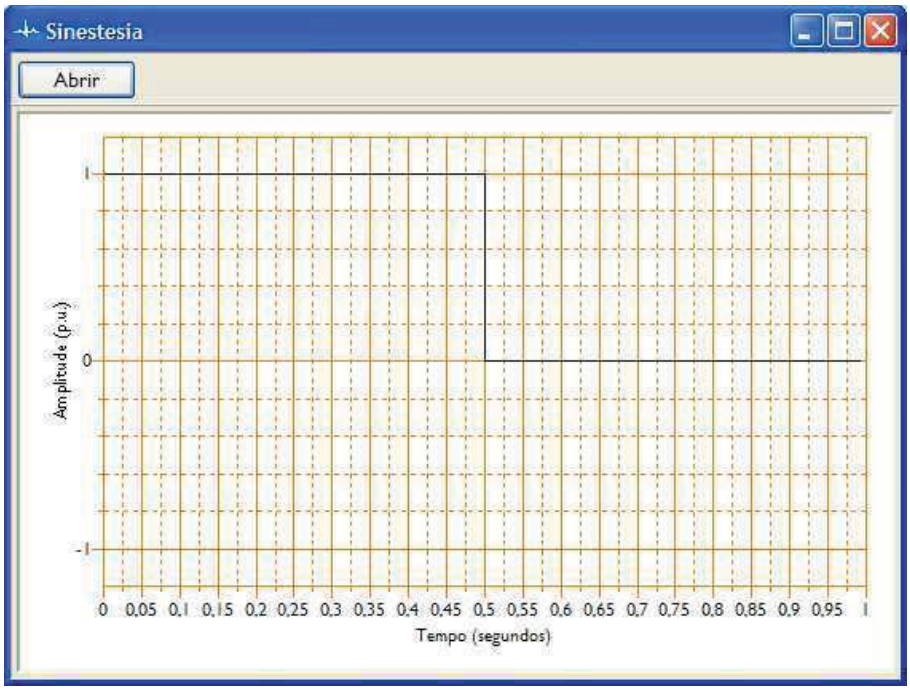

Figura 10 – Representação gráfica de um degrau unitário no software desenvolvido (O autor).

A figura 11, por sua vez, ilustra graficamente um pulso triangular com pico localizado em 0,5 segundos. Algumas pequenas deformidades podem ser notadas tanto na rampa de subida quanto na rampa de descida, isso se dá por conta da baixa resolução do arquivo (taxa de amostragem e número de bits por amostra reduzidos).

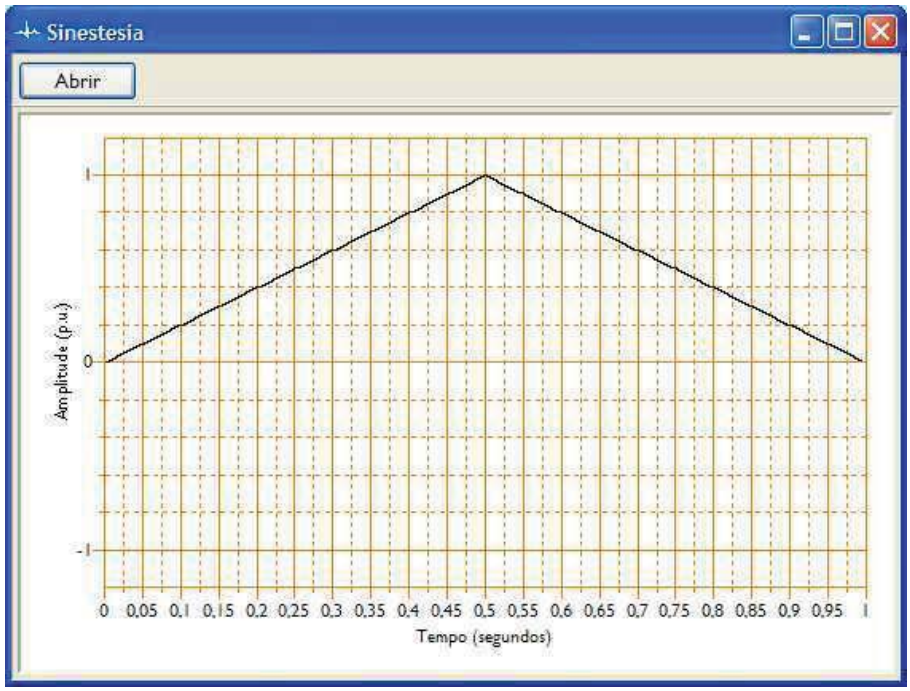

Figura 11 – Representação gráfica de um pulso triangular no software desenvolvido (O autor).

A figura 12 mostra o resultado obtido com a execução de um arquivo contendo uma senóide com frequência de 1 Hz. Mais uma vez podemos observar alguns degraus na forma de onda decorrentes da baixa resolução do arquivo gerado.

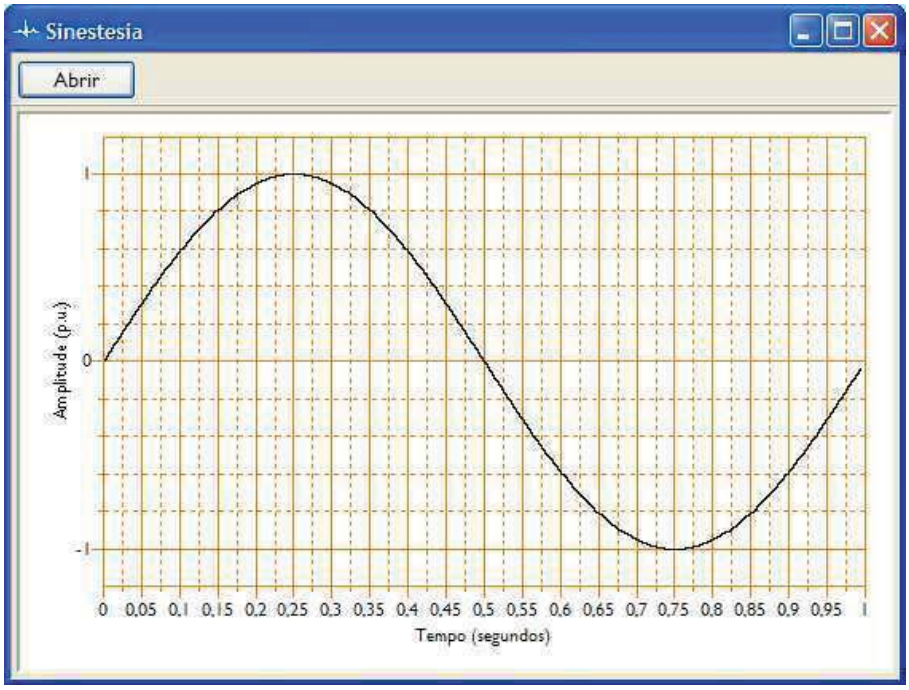

Figura 12 – Representação gráfica de uma senóide no software desenvolvido (O autor).

A fim de ilustrar um sinal mais complexo um trecho de uma música foi processado no MATLAB e o resultado obtido pode ser comparado com o mesmo trecho sendo representado pelo software desenvolvido. O trecho tem 0,1 segundo de duração e o arquivo é amostrado a uma taxa de 8000 bits por segundo com amostras de 16 bits.

As figuras 13 e 14 ilustram essa comparação.

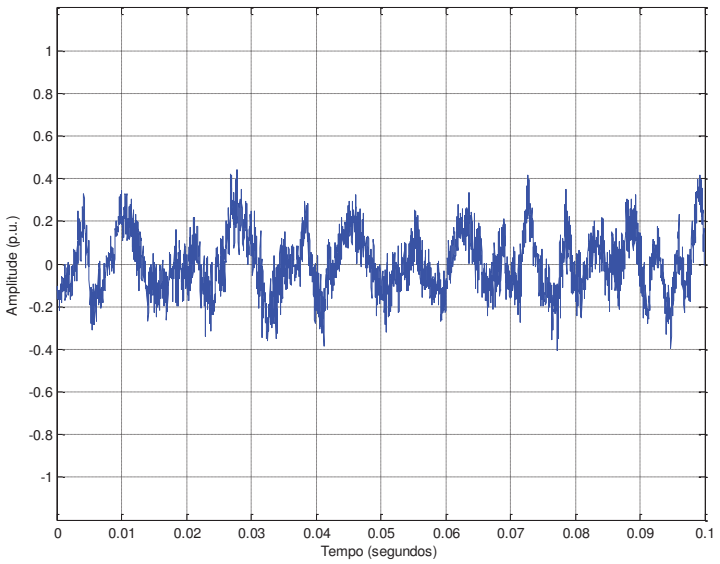

Figura 13: Trecho de música representado graficamente no MATLAB (O autor).

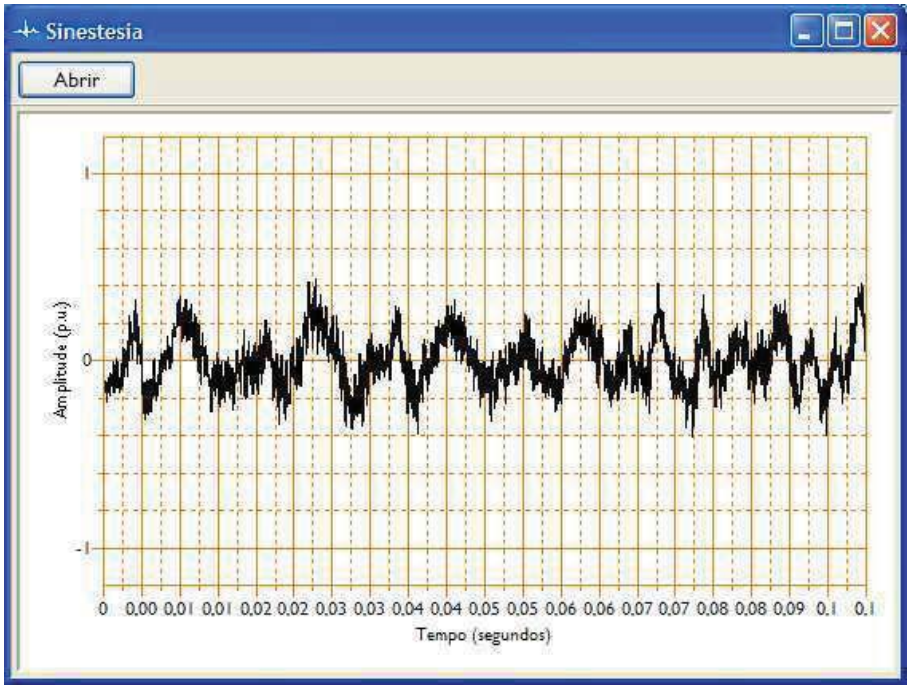

Figura 14: Trecho de música representado graficamente no software desenvolvido (O autor).

A comparação visual entre as figuras 13 e 14 indica que o programa desenvolvido é capaz de plotar ondas mais complexas de forma satisfatória.

De maneira geral, é possível concluir que o programa desenvolvido cumpre os objetivos desta seção, sendo capaz de representar um arquivo WAV de forma gráfica.

5.2. Descompressão de arquivos em ADPCM

Neste tópico são apresentados os resultados obtidos com o software desenvolvido quanto à descompressão de arquivos ADPCM e posterior representação gráfica dos mesmos.

Os aparelhos tocadores de MP3 em geral gravam áudio em WAV com dados comprimidos pelo algoritmo ADPCM. Para recuperar os dados no estudo de caso deste trabalho se torna então necessário decodificar os dados contidos em um arquivo comprimido em ADPCM.

Foi desenvolvido um algoritmo baseado em um artigo da empresa Dialogic (DIALOGIC, 2007). Algumas alterações foram realizadas no algoritmo original uma vez que o poder de processamento do PC permite que algumas simplificações sejam deixadas de lado.

Arquivos em IMA-ADPCM foram gerados em software comercial a fim de testar o programa desenvolvido quanto à decodificação de arquivos compactados com ADPCM. A figura 15 ilustra uma senóide de 0,1 Hz gerada a partir de um arquivo WAV/ IMA - ADPCM.

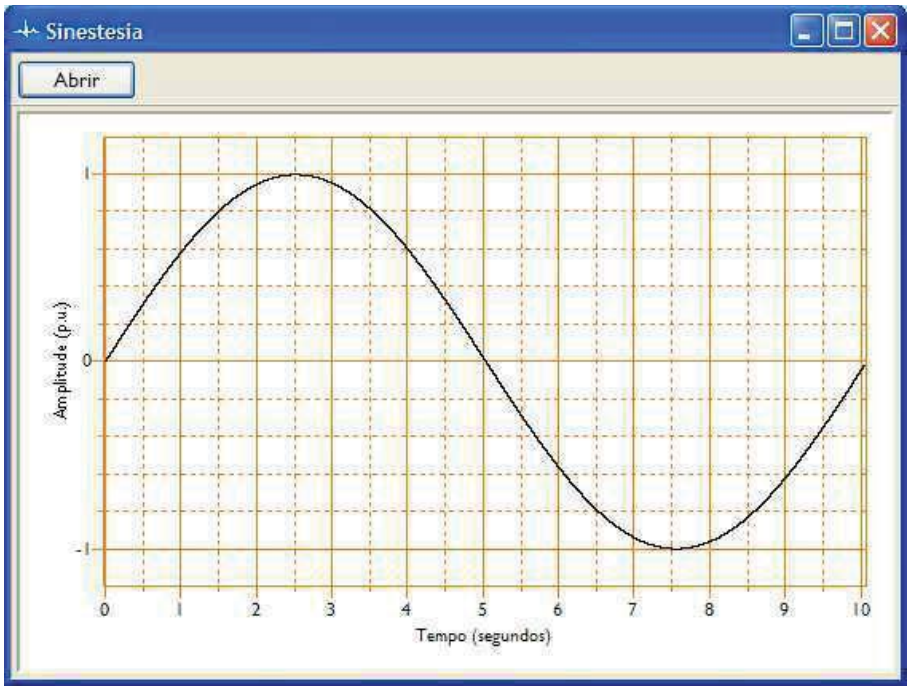

Figura 15: Senóide reconstituída a partir de arquivo compactado utilizando ADPCM, f= 0,1 Hz (O autor).

Para f= 0,1 Hz o sinal reconstituído tem boa qualidade. Visualmente não é possível apontar distorções significativas.

A figura 16 ilustra a continuidade do experimento, com um aumento de frequência para 0,2 Hz.

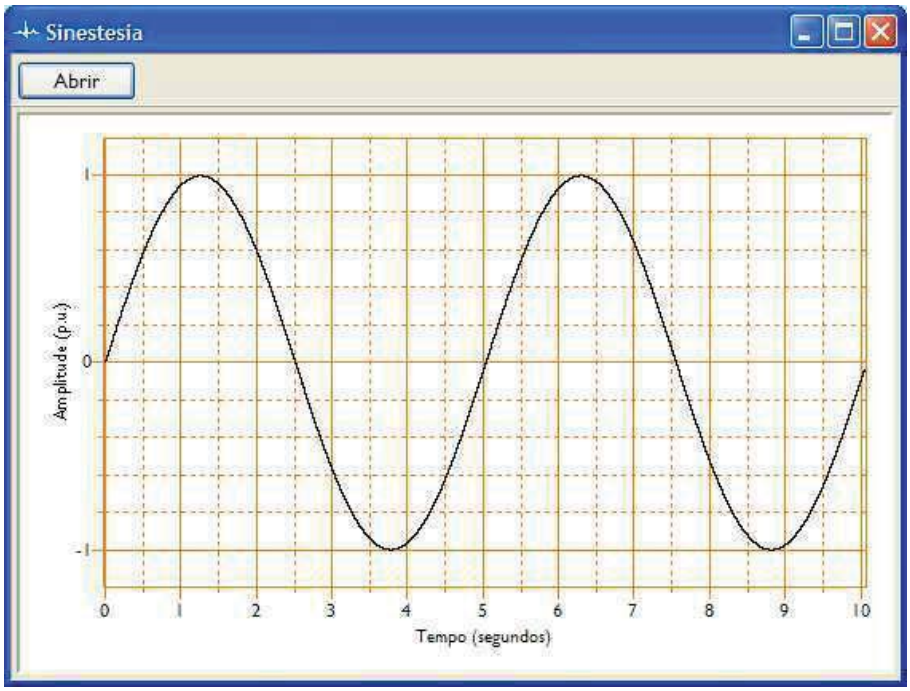

Figura 16: Senóide reconstituída a partir de arquivo compactado utilizando ADPCM, f= 0,2 Hz (O autor).

Na figura 17 é possível observar o efeito do aumento da frequência para f= 0,4 Hz.

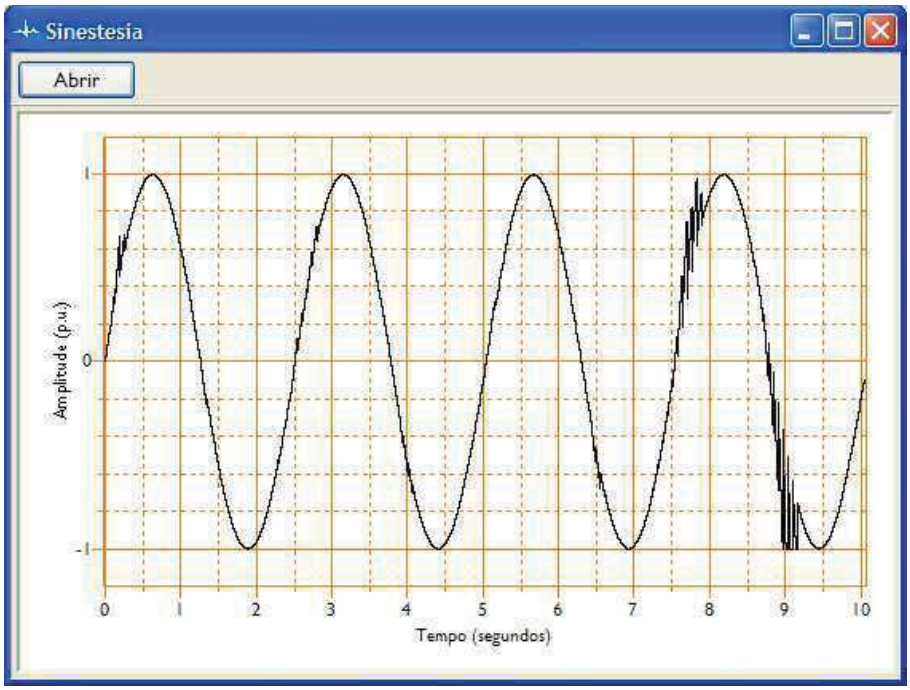

Figura 17: Senóide reconstituída a partir de arquivo compactado utilizando ADPCM, f= 0,4 Hz (O autor).

Distorções significativas podem ser observadas na figura 17. Perto dos nove segundos, há saturação do sinal em alguns pontos.

As distorções ocorrem de forma não periódica. Entre 4,0s e 6,5s há um trecho quase livre de distorções enquanto no último período é clara a presença de distorções.

As figuras 18 e 19 ilustram o resultado obtido com uma frequência de 1,0 Hz. A figura 19 nada mais é do que o detalhe da figura 18 ao limitar a plotagem ao tempo de 2 segundos.

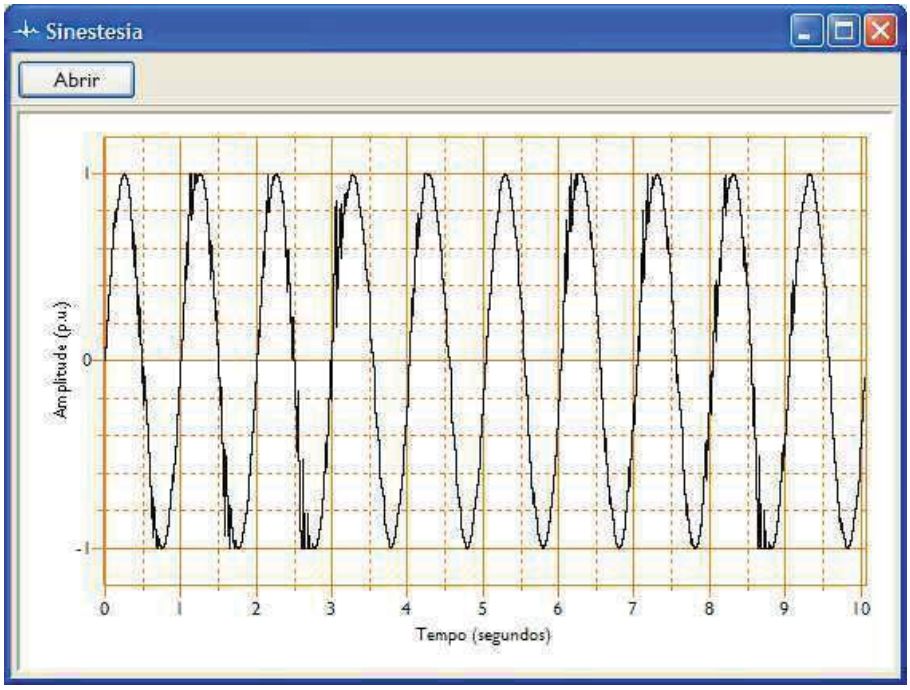

Figura 18: Senóide reconstituída a partir de arquivo compactado utilizando ADPCM, f= 1,0 Hz (O autor).

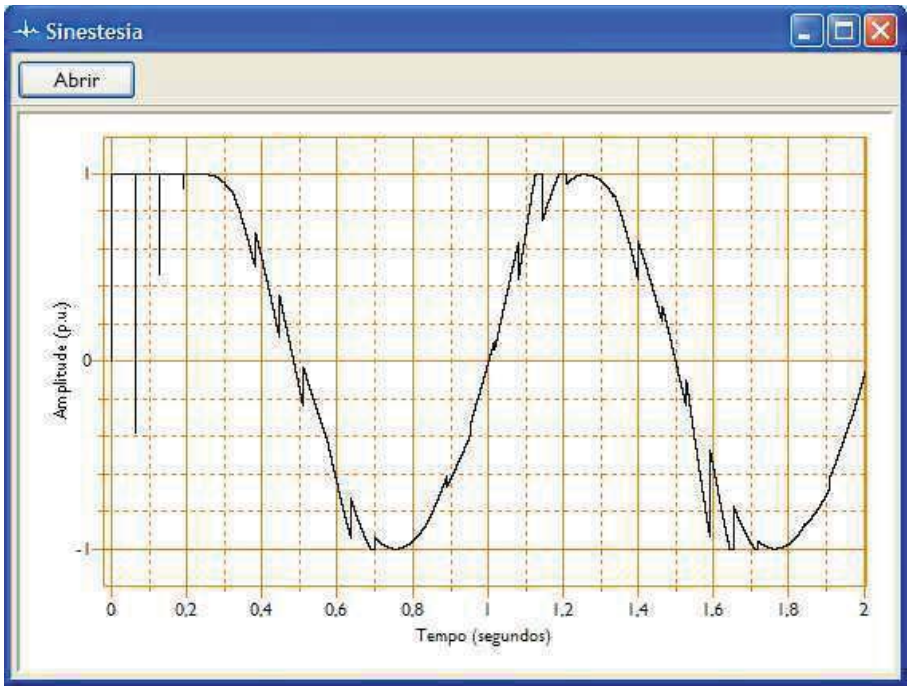

Figura 19: Senóide reconstituída a partir de arquivo compactado utilizando ADPCM (f= 1,0 Hz), execução limitada a 2 segundos (O autor).

Nota-se que em 1 Hz as distorções ocorrem novamente de forma não periódica.

A figura 19 ilustra o mesmo sinal sendo descompactado com uma limitação no tempo de execução. As distorções ocorrem de maneira diferente da figura 18, nota-se que o algoritmo de descompressão já inicia a execução saturado e demora cerca de 0,2s para sair da saturação.

A cada 256 amostras, o arquivo possuí 2 bytes contendo uma amostra de 16 bits sem compressão. Essa amostra sem compressão serve como guia para o algoritmo de descompressão voltar às condições normais caso haja saturação ou algum problema no preditor adaptativo.

A inserção de um break point a cada amostra sem compressão indica que todos os ‘dentes’ ocorrem nos pontos de guia. Portanto, é possível concluir que o preditor adaptativo responde frequentemente muito rápido ou muito devagar às mudanças no sinal a partir de frequências em torno de 0,4 Hz.

Como o algoritmo ADPCM tem como principal aplicação a compressão de sinais de voz para transmissão digital, outros testes foram realizados aplicando-se sinais com frequências dentro da faixa de voz. A figura 20 ilustra o resultado obtido para uma senóide com 300 Hz de frequência.

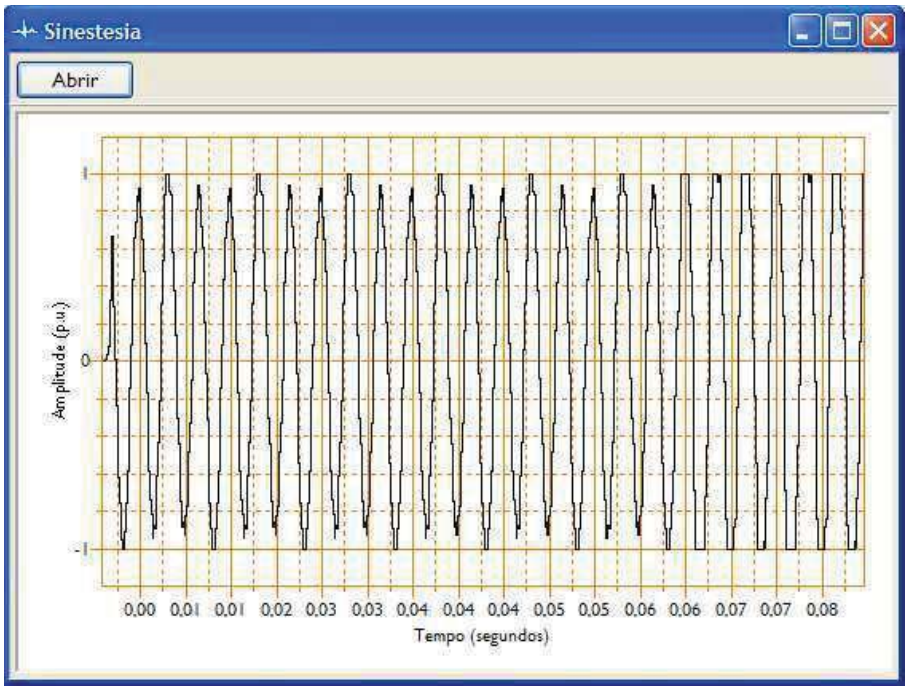

Figura 20: Senóide reconstituída a partir de arquivo compactado utilizando ADPCM, f= 300 Hz (O autor).

Nota-se uma resposta razoável para a frequência de 300 Hz. Porém há uma tendência de saturação do decodificador a partir de t= 0,07s e uma tendência de atenuação antes de t= 0,06s. A fim de estabelecer a utilidade do algoritmo para o estudo de caso do sinal ECG modulado em FM, um teste foi realizado para plotar o sinal ainda modulado em FM. O resultado, ilustrado na figura 21, tem frequência na faixa dos 1500 Hz.

Figura 21: Senóide reconstituída a partir de arquivo compactado utilizando ADPCM, frequência da ordem de 1500 Hz (O autor).

A faixa de voz vai de 300 a 3000 Hz, logo o sinal modulado em FM está aproximadamente no meio da faixa na qual o algoritmo deveria ter um bom desempenho.

Após os experimentos realizados, diversas mudanças no algoritmo foram propostas e executadas. Infelizmente, os resultados não foram satisfatórios para a faixa de frequência de interesse.

O uso da técnica de compressão ADPCM implica na utilização de algoritmos de descompressão e compressão com preditores iguais. Acredita-se que o preditor proposto no software desenvolvido possua uma resposta bastante diferente do tocador de MP3 portátil.

Devido ao desempenho insatisfatório, a solução adotada foi realizar a decodificação em software externo. Ao realizar tal procedimento, o próprio software comercial exibiu mensagens de erro informando que o arquivo poderia estar corrompido e que pode ter ocorrido erro no cálculo da duração total do mesmo.

A fim de suplantar essa dificuldade, diversas alternativas foram estudadas e a que parece mais promissora é a utilização de bibliotecas externas para realizar a decodificação sem a necessidade da execução de outro software de tratamento de arquivos WAV.

O projeto SoX (Sound Exchange) fornece uma série de ferramentas para tratamento e processamento de áudio inclusive no formato ADPCM. SoX é gratuito e seu código fonte é aberto, a utilização deste recurso seria provavelmente uma boa solução para o problema da descompressão ADPCM.

A utilização das bibliotecas do projeto SoX foi estudada mas não houve tempo hábil para implementação eficiente dos recursos desta ferramenta. A partir deste ponto todos os arquivos utilizados foram devidamente decodificados para PCM utilizando GoldWave (versão de teste).

5.3. DEMODULAÇÃO FM EM TEMPO DISCRETO

Neste tópico são apresentados os resultados obtidos com o algoritmo de demodulação FM em tempo discreto.

O algoritmo de demodulação FM em tempo discreto foi desenvolvido com base no conceito de detector de inclinação. Segundo Gomes (2002), um dos métodos mais simples para recuperar a informação contida em um sinal FM é o aproveitamento da inclinação praticamente linear da região não ressonante de um circuito sintonizado. Após a filtragem, a parte negativa do sinal é removida por um ceifador e o sinal resultante passa por um detector de envoltória. O processo é ilustrado na figura 22.

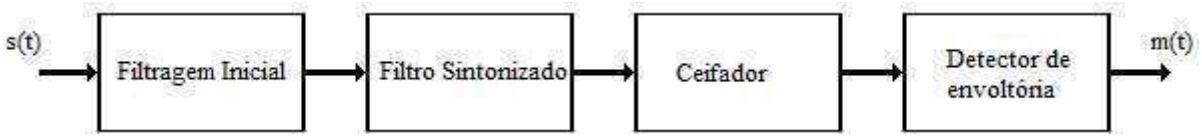

Figura 22: Processo de demodulação FM (O autor)

O bloco de filtragem inicial deve eliminar ruídos de alta ou baixa frequência fora da faixa de interesse. Experimentos com diversos filtros indicam que a principal interferência no sinal do caso de estudo (contendo informação relativa a um exame de ECG) se dá em altas frequências, portanto foi desenvolvido um filtro passa baixa para eliminar esses ruídos.

O filtro sintonizado é um passa faixa, projetado de tal forma que o sinal FM se encontre em uma região aproximadamente linear do filtro. O resultado dessa segunda filtragem é um sinal modulado em amplitude e em frequência.

O ceifador faz o papel de um diodo, excluindo a parte negativa do sinal, condicionando o mesmo para o processamento no detector de envoltória.

O detector de envoltória é um filtro passa baixa com frequência tal que detecte a envoltória do sinal de entrada, portanto em sua saída tem-se o sinal recuperado.

Foram desenvolvidas funções para cada bloco do processo. O conjunto foi então avaliado no MATLAB e em seguida implementado em C++. A seguir, são exploradas as funções de uma forma individual e no final do capítulo é feita a avaliação e comparação entre o desempenho obtido no MATLAB e no software desenvolvido.

A teoria necessária para criação dos filtros foi baseada em SMITH, 1998. As funções utilizadas também tem como base a mesma fonte.

5.3.1. Filtragem inicial

Seja o sinal FM com uma portadora de 1400 Hz e desvio de frequência de 200 Hz (características do sinal FM com o ECG a ser recuperado). Considerando que o sinal pode conter ruídos consideráveis fora da faixa especificada (especialmente devido ao processo de decodificação ADPCM - PCM), foi projetado um filtro passa baixa com frequência de corte de 1800 Hz.

O filtro é um FIR que utiliza janela de Hamming. A figura 23 ilustra o comportamento do filtro criado.

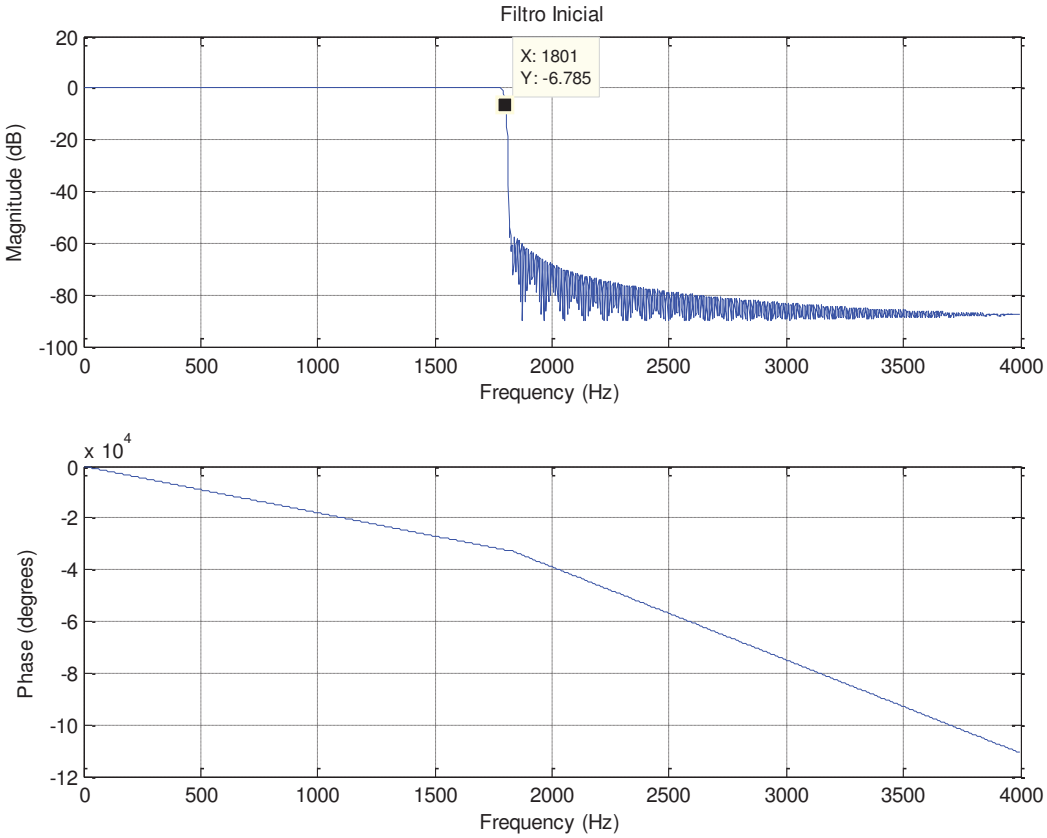

Figura 23: Filtro Inicial, diagramas de magnitude e de fase (O autor).

É possível observar, segundo as marcações na figura (23), que o sinal de interesse não deve sofrer atenuações. O propósito desse filtro é apenas atenuar frequências acima da faixa de interesse.

A função utilizada para criar o filtro foi desenvolvida tomando o cuidado de não utilizar recursos avançados do MATLAB, tornando possível a implementação quase direta da mesma em C++. O código pode ser encontrado na integra no Anexo 1.

5.3.2. Filtro Sintonizado

Tomando por base novamente as características do sinal do estudo de caso, ou seja, 1400 Hz com desvio de frequência de 200 Hz, é preciso fazer o projeto de um filtro que possua uma região aproximadamente linear situada entre 1200 e 1600 Hz.

Utilizando desta vez a função desenvolvida para criar filtros passa-faixa, temos:

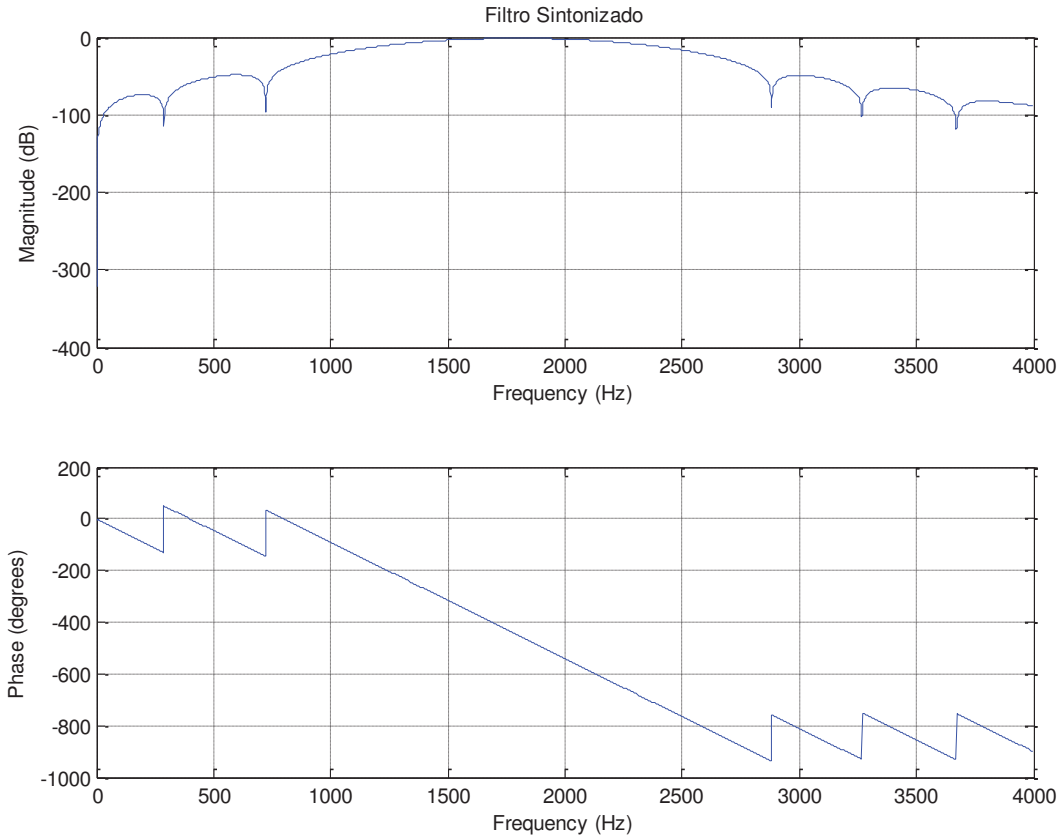

Figura 24: Filtro Sintonizado, diagramas de magnitude e de fase (O autor).

Para implementar esse filtro, primeiro foi desenvolvida uma função capaz de criar filtros passa faixa. Deve-se inserir a frequência de corte superior e inferior desejadas e o tamanho da janela do filtro. Como resultado, a função retorna os valores da janela. O filtro criado é do tipo FIR e utiliza janela de Hamming.

A função desenvolvida pode ser encontrada na íntegra no Anexo II.

O projeto final do filtro se deu após uma série de testes com diversas frequências de corte e diversos tamanhos de janela. A figura 25 ilustra o diagrama de magnitude do passa faixa em escala linear, para que a região de interesse possa ser visualizada.

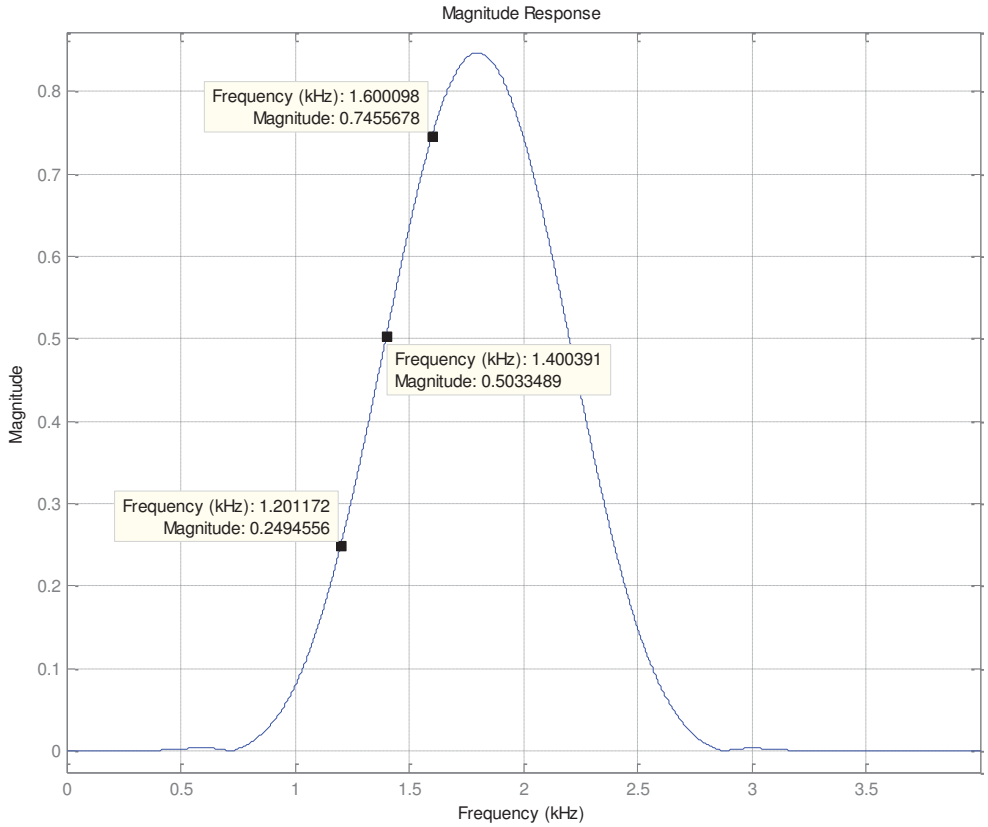

Figura 25: Filtro Sintonizado, diagramas de magnitude com escala linear (O autor).

Frequências acima dos 1800 Hz devem ser atenuadas pelo filtro inicial, não tendo influência significativa sobre o resultado final. O resultado dessa filtragem deve ser um sinal com amplitude proporcional à frequência que pode ser facilmente demodulado por um ceifador seguido de um detector de envoltória.

5.3.3. Ceifador

O ceifador nada mais é do que um laço contendo um comando condicional. Se o sinal for maior ou igual a zero não sofre alterações. Se o sinal for menor que zero, deve ser eliminado.

A função preenche com zeros os pontos do sinal que são negativos. A função é reproduzida na íntegra no anexo III.

5.3.4. Detector de envoltória

O detector de envoltória é um filtro passa baixa com uma frequência ajustada para detectar o contorno do sinal. Diferentes frequências foram testadas e chegou-se ao ajuste de 100 Hz. A figura 26 ilustra as características de fase e magnitude do filtro passa baixa.

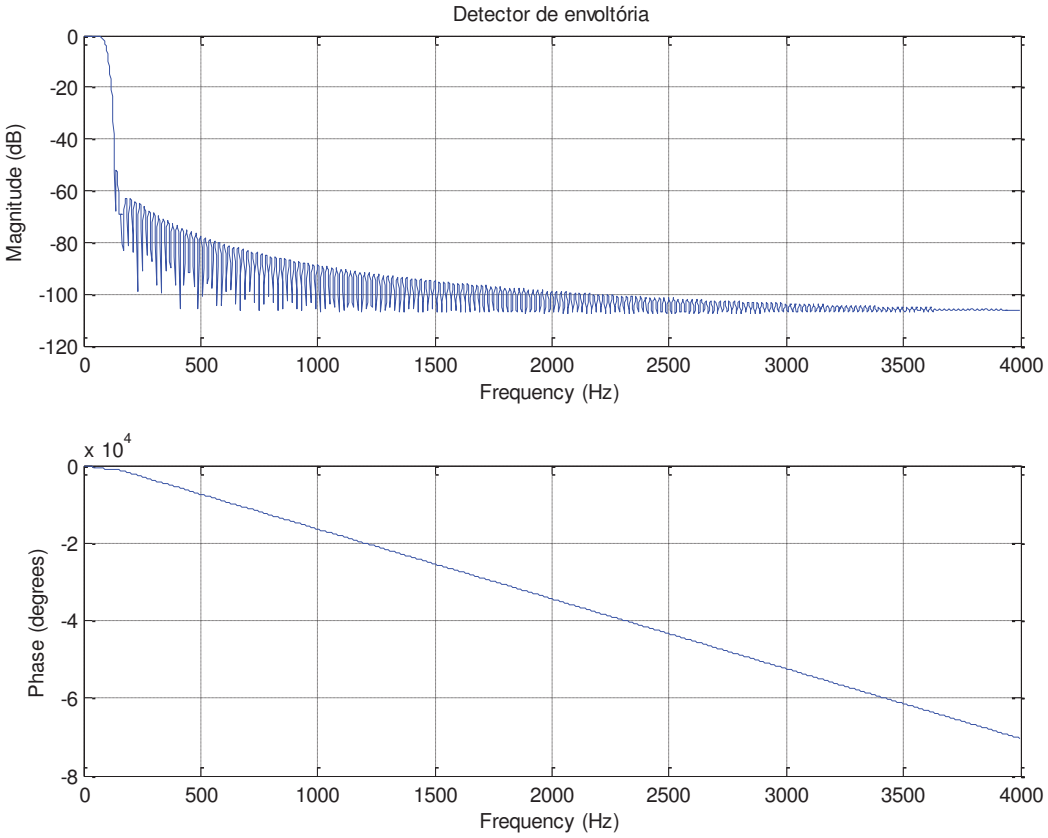

Figura 26: Filtro passa baixa (detector de envoltória), diagramas de magnitude e de fase (O autor).

5.3.5. Aplicação

A filtragem de sinais foi realizada por meio de convolução. A função para realizar a convolução também foi escrita de forma que pudesse ser facilmente reescrita em C++. O código pode ser encontrado na íntegra no Anexo IV.

Para testar o algoritmo de demodulação, primeiro foi desenvolvido um teste com uma senóide de 2 Hz modulando uma portadora de 1400 Hz com desvio de frequência máximo de 200 Hz. A figura 27 ilustra o sinal modulante, o sinal FM e o sinal recuperado utilizando o algoritmo de demodulação FM do próprio MATLAB.

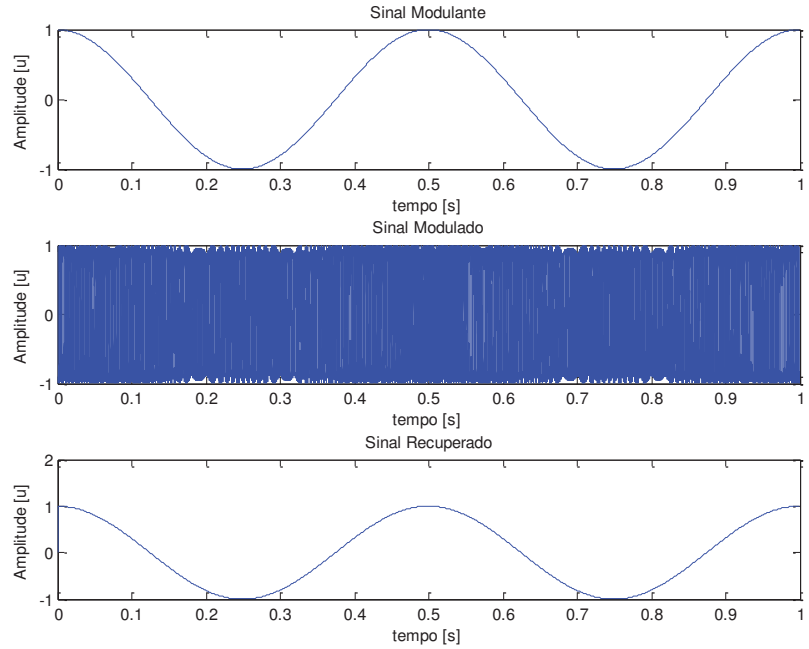

Figura 27: Sinal modulado e recuperado utilizando algoritmo interno do MATLAB (O autor).

A figura 28, por sua vez, ilustra o sinal recuperado utilizando o algoritmo e funções desenvolvidas para posterior implementação em C++.

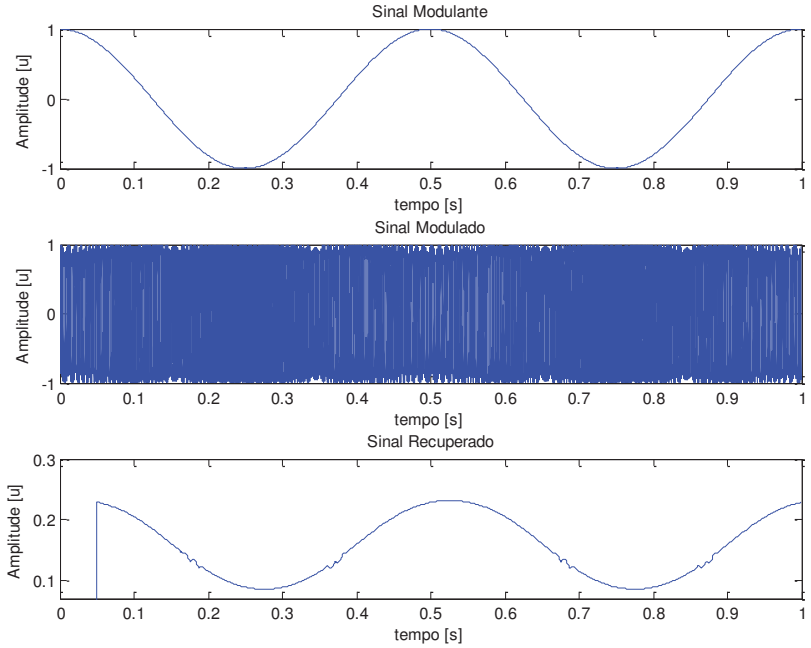

Figura 28: Respectivamente, sinal modulante criado no MATLAB; sinal modulado por função interna ao MATLAB e, mais abaixo, sinal recuperado pelo algoritmo desenvolvido (O autor).

É evidente que a amplitude da onda recuperada deve ser normalizada e que existe um ruído periódico que aparece pela primeira vez um pouco antes dos 0,2 segundos.

Esse ruído pode ser removido facilmente com a implementação de um filtro para suavizar o sinal final. Experimentos com sinais modulantes com frequências maiores exibiram desempenho bastante satisfatório.

A figura 29 ilustra cada um dos sinais obtidos após cada bloco do demodulador.

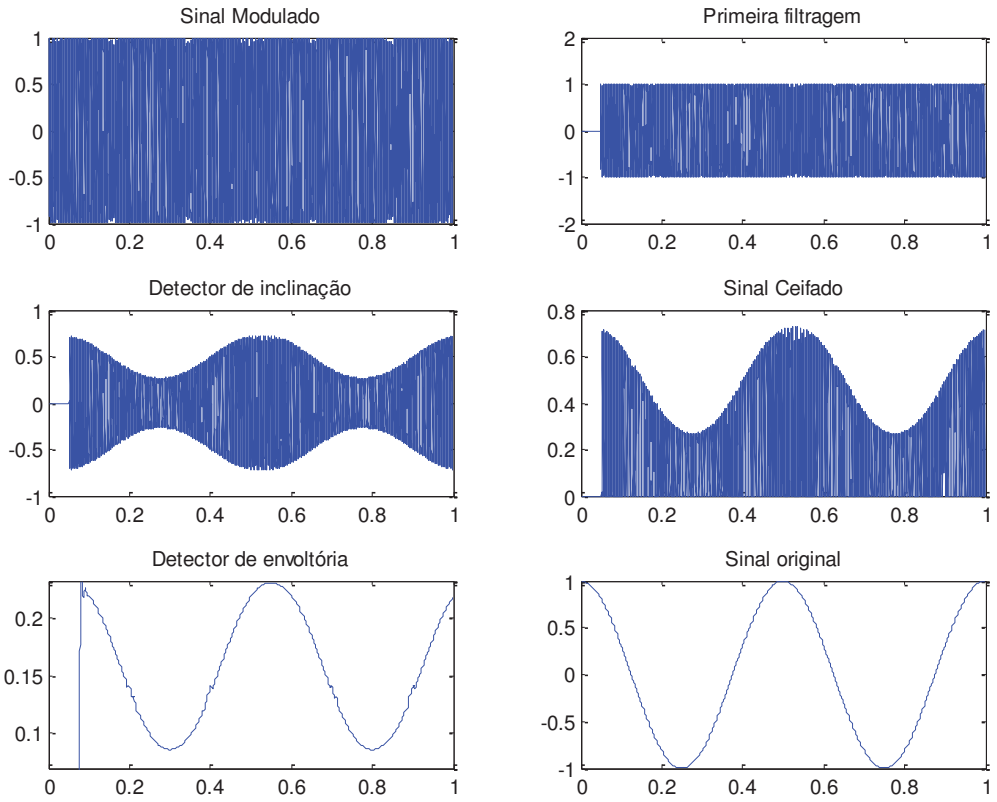

Figura 29: Sinais obtidos após cada um dos blocos do demodulador FM – teste com senóide (O autor).

Uma vez que o desempenho do algoritmo de demodulação FM foi testado com um sinal conhecido, é feito o teste com o sinal contendo o exame de ECG.

A figura 30 ilustra a aplicação do algoritmo de demodulação FM no sinal de teste de ECG. Após o detector de envoltória, foi aplicado ainda mais um filtro passa baixa com frequência de corte de 15 Hz, para suavizar o sinal. A semelhança com um sinal de ECG fica evidente, porém algumas características do complexo PQRST são distorcidas.

A onda P pode ser notada com certa clareza, assim como a onda Q. Vale salientar que o afundamento da onda Q é um pouco maior do que o esperado para um exame típico de ECG. A onda R, por sua vez, também fica evidente.

Esperava-se que a onda S fosse um pouco mais profunda e há um afundamento antes da onda T que também não era esperado. Além disso é possível notar que o sinal tem bastante ruído, apresentando grandes diferenças quando comparado ao sinal ideal da figura 2 (página 12).

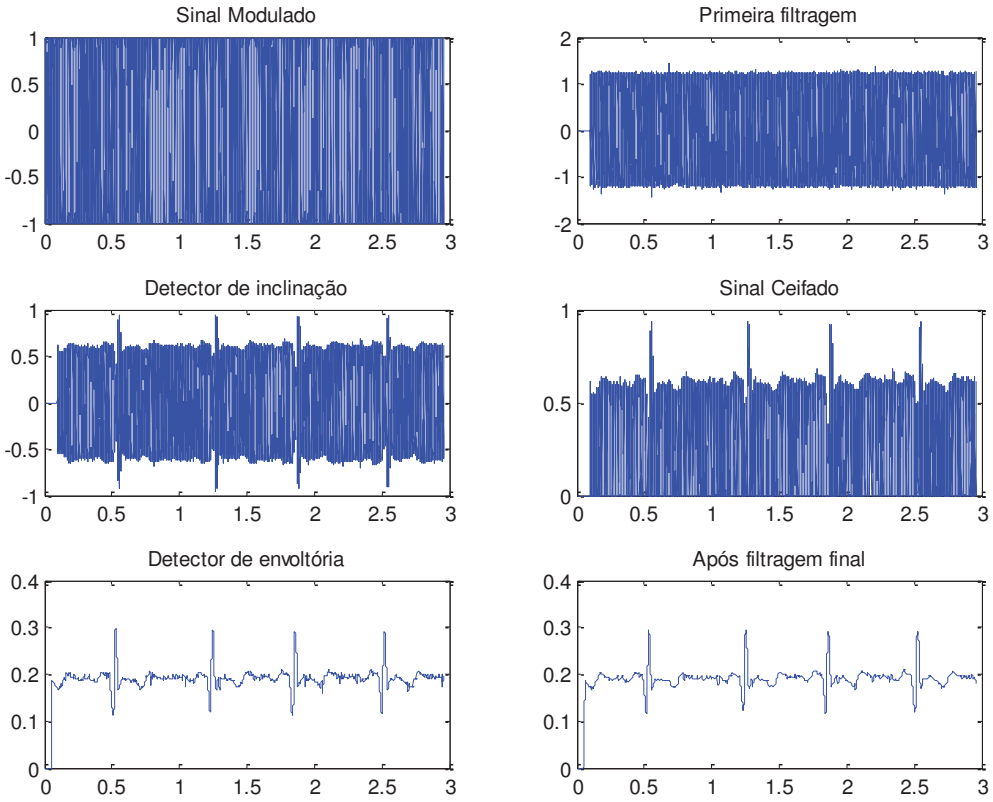

Figura 30: Sinais obtidos após cada um dos blocos do demodulador FM – Estudo de caso, ECG (O autor).

Os resultados obtidos utilizando MATLAB são um bom indicativo do que se pode esperar do software desenvolvido. As figuras de 31 a 36 nas páginas a seguir ilustram os resultados obtidos com o programa escrito em C++.

A cada filtragem executada, o programa plota os resultados possibilitando a observação passo a passo do procedimento.

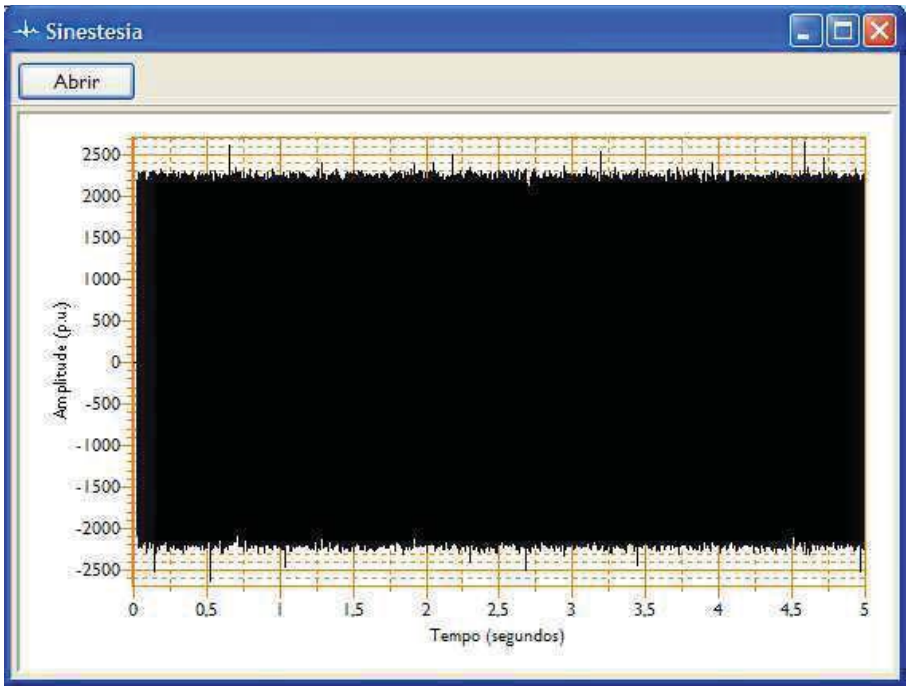

Figura 31: Sinal obtido após filtragem inicial (O autor).

A figura 31 ilustra os 5 primeiros segundos do arquivo após a primeira filtragem, que tem como objetivo remover componentes de frequências acima da faixa de interesse (filtragem inicial, passa baixa com frequência de corte igual a 1800 Hz).

Nota-se que a amplitude varia entre 2500 unidades para mais e para menos. Há grande atenuação do sinal após aplicação dos filtros e por conta disso foi estabelecido um valor alto de amplitude inicial.

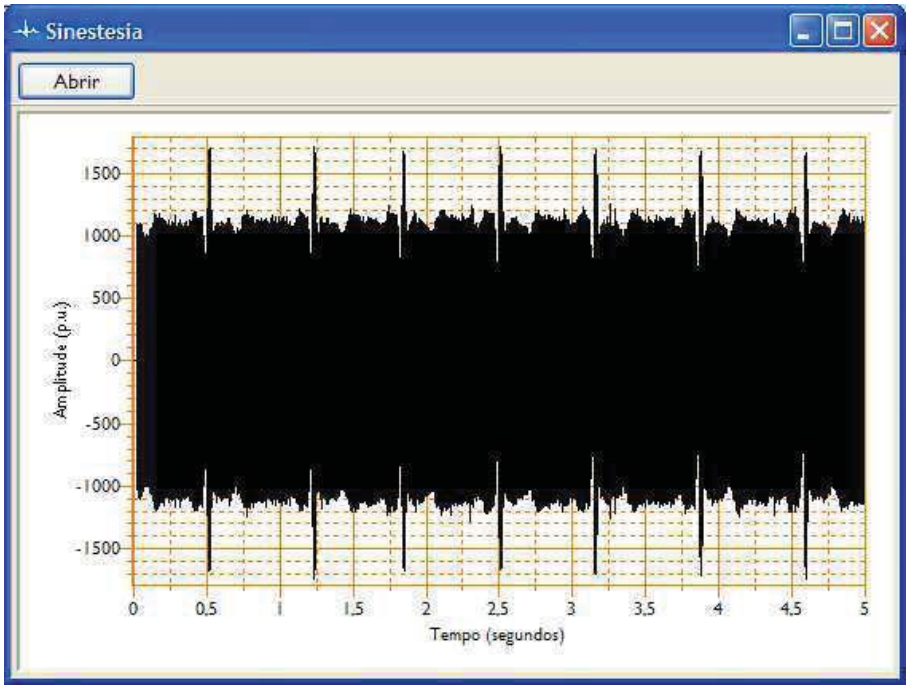

Figura 32: Sinal obtido após detector de inclinação (O autor).

A figura 32, por sua vez, ilustra os 5 primeiros segundos do sinal após a aplicação do filtro passa faixa sintonizado (detector de inclinação). Já é possível notar a semelhança do contorno da onda com a onda ideal do ECG (figura 2, página 12).

A figura 33 ilustra o resultado obtido com a remoção da porção negativa do sinal, remoção necessária para posterior aplicação do detector de envoltória.

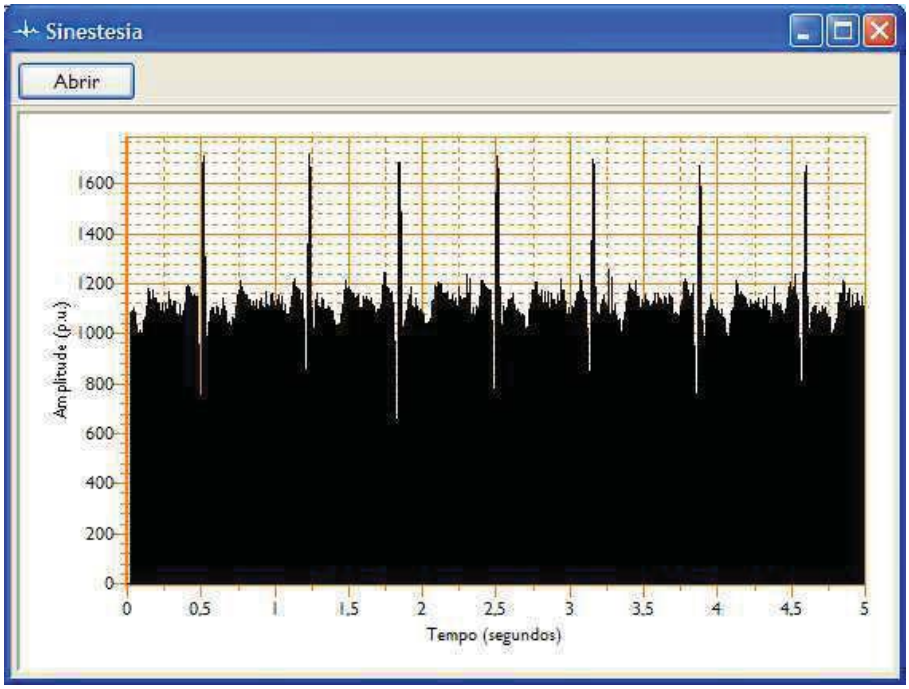

Figura 33: Sinal obtido após ceifador (O autor).

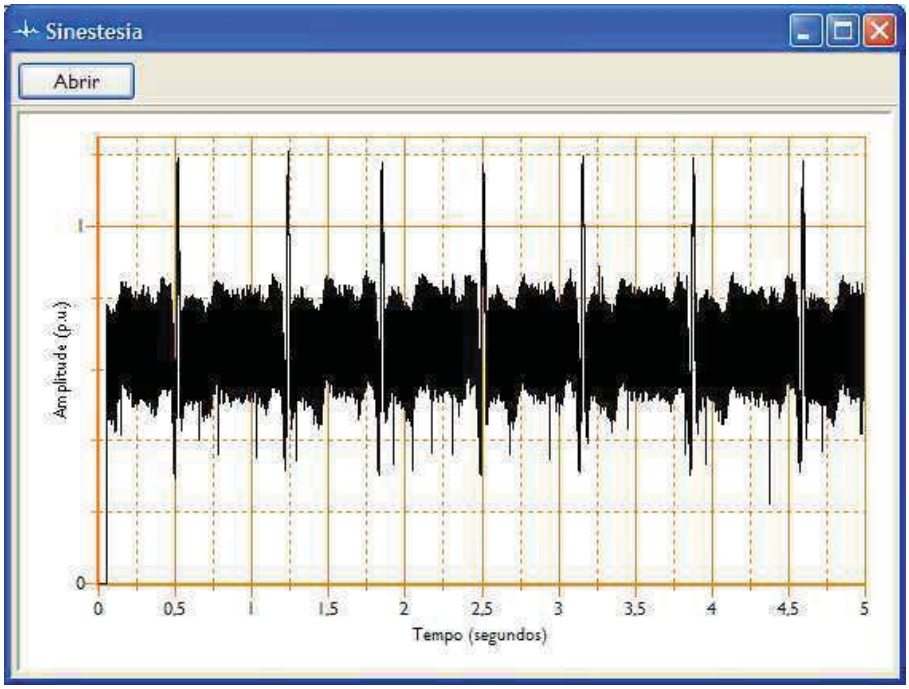

Figura 34: Sinal obtido após aplicação do detector de envoltória (O autor).

O sinal obtido após a aplicação do detector de envoltória apresentou grande ruído (figura 34), especialmente nos trechos localizados entre os complexos QRS do ECG.

Acredita-se que isso tenha ocorrido devido a diferenças no modo como os cálculos são executados no MATLAB e no software desenvolvido. As variáveis utilizadas são do tipo float o que pode ter prejudicado o desempenho final do software.

Para diminuir esse ruído foi utilizado um filtro de média móvel de 20 amostras. O efeito imediato após a aplicação deste filtro pode ser visto na figura 35.

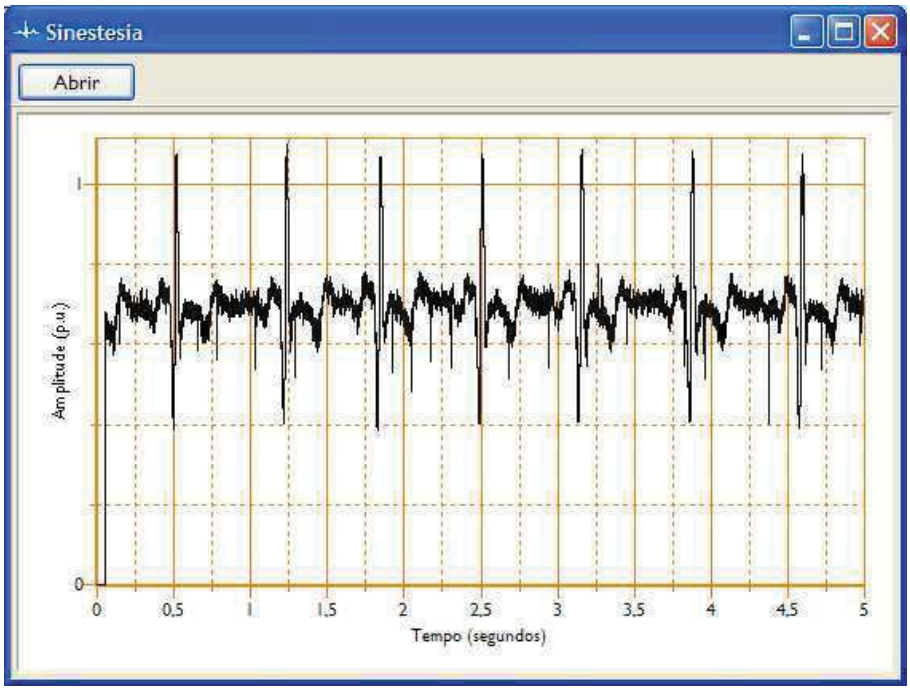

Figura 34: Sinal obtido após aplicação do filtro de média móvel (O autor).

Este sinal já se aproxima bastante do esperado. Colocando a exibição em uma escala conveniente, semelhante à utilizada para avaliações de exames de ECG convencionais, é possível observar o resultado final do processamento no software desenvolvido.

As figuras 36 e 37 ilustram a aparência do sinal de ECG recuperado e exibido em escala conveniente para avaliação.

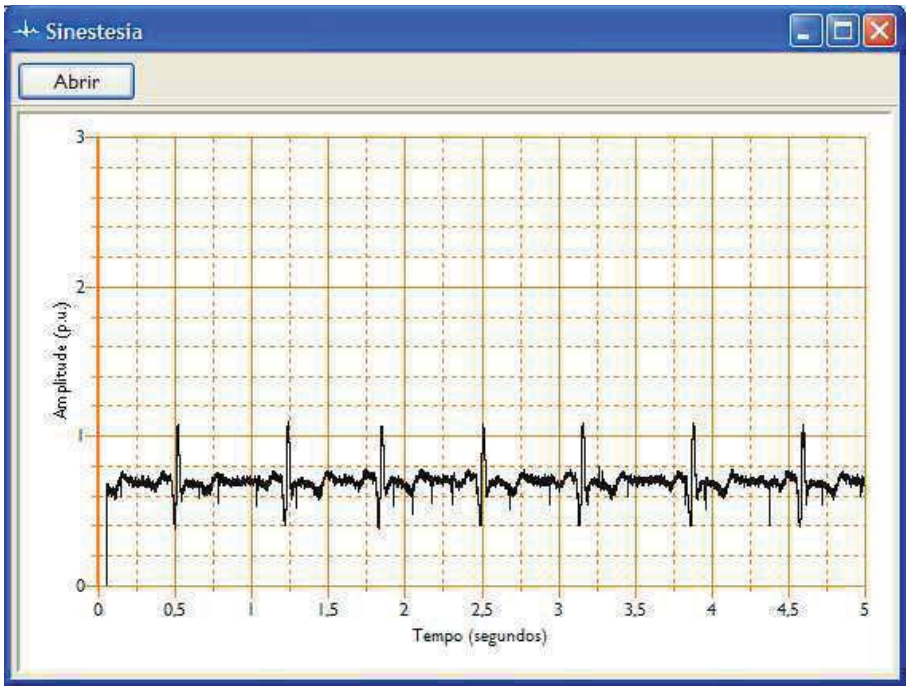

Figura 36: Sinal final, escala adequada para leitura de ECG (O autor).

Pode ser notada ainda a presença de ruído, esse ruído poderia ser removido com a aplicação de um filtro de média móvel utilizando mais amostras. Filtragens adicionais não foram realizadas pois considerou-se que o sinal da figura 36 é suficiente para julgar o desempenho geral do software desenvolvido.

A figura 37 exibe o trecho final da gravação do ECG. Durante toda a extensão do arquivo observou-se qualidade semelhante à ilustrada nas figuras 36 e 37, sem grandes mudanças no comportamento do sinal.

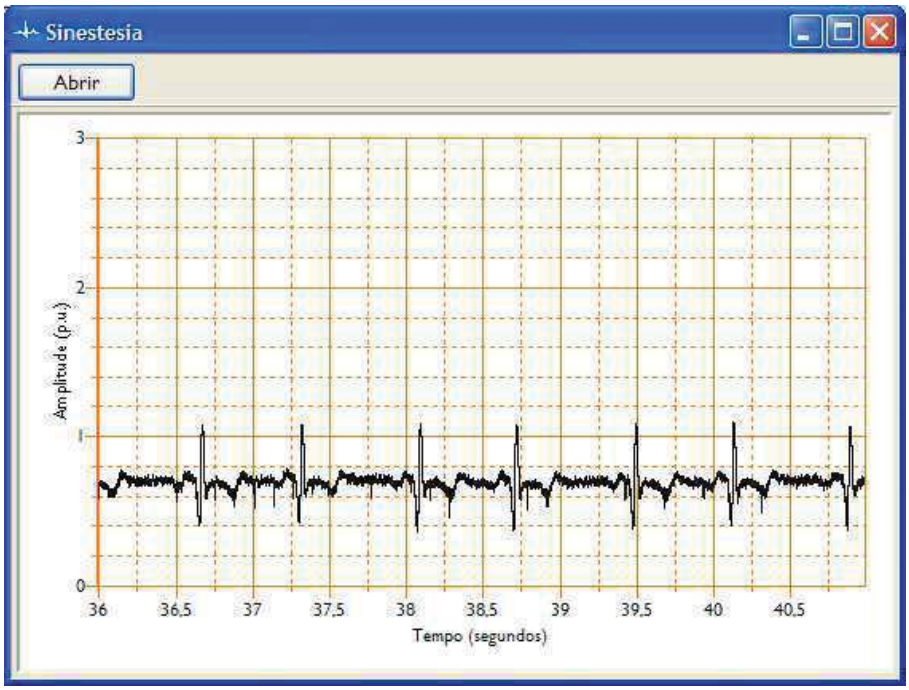

Figura 37: Sinal final, ultimo trecho da gravação original (O autor).

6. CONCLUSÕES

Quanto ao processamento básico de arquivos WAV, o desempenho foi satisfatório e a única melhoria que se pode sugerir é quanto à velocidade. Uma revisão no modo como se processa o arquivo pode sugerir outras abordagens que ocupem menos tempo de processamento (o tempo de processamento não foi levado em conta durante o desenvolvimento do programa).

Quanto à decodificação de arquivos ADPCM, fica apenas a sugestão de se utilizar uma ferramenta externa. Os testes e a busca por uma solução adequada para esse quesito tomaram bastante tempo e criaram diversas versões do software desenvolvido (grande parte delas não funcionais).

No que diz respeito à demodulação, conclui-se que foi atingido um desempenho satisfatório. Melhorias podem ser feitas utilizando variáveis tipo double ao invés do tipo float. Além disso também podem ser feitas melhorias de performance para aumentar a velocidade de execução, apesar de o tempo médio para exibir o ECG não ter passado de cerca de 5 segundos (o que torna o beneficio de se melhorar o desempenho pouco relevante).

De forma geral, a proposta de se utilizar um aparelho portátil tocador MP3 como registrador de dados é considerada válida e o desenvolvimento de software para lidar com os dados registrados é possível.

Finalmente, é sugerida a criação de funções dentro do programa desenvolvido para processar dados referentes ao ECG. Uma função para retornar o número de batimentos por minuto médio, por exemplo, poderia ser implementada sem grandes dificuldades.

O desenvolvimento deste trabalho se mostrou bastante desafiador e o foi possível explorar com mais profundidade uma série de tópicos introduzidos durante o curso de graduação. Outras conclusões diversas foram feitas durante o capítulo de desenvolvimento por se considerar mais prático e coerente que essas informações fossem subsequentes aos tópicos relacionados.

REFERÊNCIAS

CCRMA, disponível em: https://ccrma.stanford.edu/courses/422/projects/WaveFormat/. Acessado em 08/09/2012.

DIALOGIC: Using the ADPCM Algorithm in Dialogic Voice Processing Applications, 2007. Application Note.

HÖPFEL – Anotações de aula e apostila da disciplina Medical Sensors, 2011 (prof. Dr. Dieter Höpfel, Hochschule Karlsruhe).

G 726 ITU. GENERAL ASPECTS OF DIGITAL TRANSMISSION SYSTEMS; TERMINAL EQUIPMENTS. 40, 32, 24, 16 kbit/s ADAPTIVE DIFFERENTIAL PULSE CODE MODULATION (ADPCM), 1990.

GIBSON, Jerry D. Adaptive Prediction in Speech Differential Encoding Systems. PROCEEDINGS OF THE IEEE, VOL. 68, NO. 4, Abril de 1980.

GOMES, Alcides Tadeu 2002. Telecomunicações: Transmissão e Recepção AM-FM : Sistemas Pulsados / Alcides Tadeu Gomes 14ª edição. Érica, 1998. 415 p.

HAYKIN, Simon. Communication Systems / Simon Haykin – 4th ed. John Wiley and Sons, Inc. 2001. 838 p.

LUCENA, Samuel Euzédice de. Registrador de sinais analógicos usando MP3 player portátil com entrada para voz. Tese (livre docência) –Universidade Estadual Paulista "Júlio de Mesquita Filho". Departamento de Engenharia Elétrica, 2008.

MICROSOFT, disponível em: http://msdn.microsoft.com/en-us/library/windows/desktop/ee415713%28v=vs.85%29.aspx. Acessado em 06/09/2012.

ORLOV, V.N. Electrocardiography. Mir Publishers, 1988. 344 p.

SOX, disponível em http://sox.sourceforge.net/. Acessado em 11/11/2012.

SMITH, Steven W. The Scientist and Engineer's Guide to Digital Signal Processing, copyright ©1997-1998 by Steven W. Smith. For more information visit the book's website at: www.DSPguide.com.

VASAVADA, Yash M. PERFORMANCE EVALUATION OF A FREQUENCY MODULATED SPREAD-SPECTRUM SYSTEM. Tese (Mestrado) - Faculty of the Virginia Polytechnic Institute and State University, 1996.

ANEXO I

Função utilizada para criar filtros passa baixa.

```
function[H]=criafiltro(FC,M)
M=M+1;
H=zeros(M,1);
for i = 1:M
    if(i-M == 1)
        H(i) = 2*pi*FC;
    else if (i-M ~= 1)
            H(i)=sin(2*pi*FC*(i-M/2))/(i-M/2)*(0.54-0.46*cos(2*pi*i/M));
        end
    end
end

soma=0;
for i=1:M
    soma=soma+H(i);
end
for i=1:M
    H(i)=H(i)/soma;
end
```

Anexo II

Função utilizada para criar filtros passa faixa.

```
function[H]=passafaixa(FC1,FC2,M)

H1=zeros(M+1,1);  %pre aloca H1 e H2
H2=zeros(M+1,1);

%Cria Máscara para H1:
for i = 1:(M+1)
    if((i-1)-M/2 == 0)
        H1(i) = 2*pi*FC1;
    else if ((i-1)-M/2 ~= 0)
            H1(i)=sin(2*pi*FC1*((i-1)-(M/2)))/((i-1)-M/2)*(0.54-
0.46*cos(2*pi*(i-1)/M));
        end
    end
end

%Normaliza filtro H1:
soma=0;
for i=1:(M+1)
    soma=soma+H1(i);
end
for i=1:(M+1)
    H1(i)=H1(i)/soma;
end

%Cria Máscara para H2:
for i = 1:(M+1)
    if((i-1)-M/2 == 0)
        H2(i) = 2*pi*FC2;
    else if ((i-1)-M/2 ~= 0)
            H2(i)=sin(2*pi*FC2*((i-1)-(M/2)))/((i-1)-M/2)*(0.54-
0.46*cos(2*pi*(i-1)/M));
        end
    end
end

%Normaliza filtro H1:
soma=0;
for i=1:(M+1)
    soma=soma+H2(i);
end
for i=1:(M+1)
    H2(i)=H2(i)/soma;
end

%Transforma H2 em HighPassFilter
for i=1:(M+1)
    H2(i)=-H2(i);
end
H2((M/2)+1)=H2((M/2)+1)+1;

%Soma os dois ''Kernels''
for i=1:(M+1)
    H(i) = H1(i) + H2(i);
```

```
end

%Transforma H em Passa faixas:
for i=1:(M+1)
    H(i)=-H(i);
end
H((M/2)+1)=H((M/2)+1)+1;
```

Anexo III

Função utilizada para ceifar sinal.

```
function[out]=diodo(in)
out=zeros(numel(in),1);

for i=1:numel(in)
    if (in(i) < 0)
        out(i)=0;
    else
        out(i)=in(i);
    end
end
```

Anexo IV

Função utilizada para convoluir sinais.

```
function[y]=convolue (data,H)

for j=(numel(H)+1):numel(data)
      y(j)=0;
      for i=1:numel(H)
      y(j)=y(j)+data(j-i)*H(i);
      end
end
```