RINALDO BRANDOLIM

SISTEMA EMBARCADO DE ULTRA-SOM COM DOPPLER PULSADO: DESENVOLVIMENTO DAS INTERFACES

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo
Curso de Engenharia de Computação com
ênfase em Sistemas Embarcados

ORIENTADOR: Professor Doutor Carlos Dias Maciel

São Carlos
2007

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Resumo do Projeto de Formatura apresentado à EESC-USP como parte dos requisitos necessários para a obtenção da conclusão do curso de Engenharia de Computação.

SISTEMA EMBARCADO DE ULTRA-SOM COM DOPPLER PULSADO: DESENVOLVIMENTO DAS INTERFACES

Rinaldo Brandolim

06 / 2007

Orientador: Prof. Dr. Carlos Dias Maciel

Área de Concentração: Engenharia Biomédica

Palavras-chave: Ultra-som, Efeito Doppler, Doppler Pulsado, Comunicação Serial, FPGA, VHDL, Java.

RESUMO

No processo de hemodiálise, o sangue é retirado, filtrado e devolvido ao paciente. Para sua realização, é necessária a confecção de um acesso vascular para permitir a saída e o retorno do sangue. A manutenção de uma boa adequabilidade do processo de hemodiálise em pacientes que sofrem de insuficiência renal depende diretamente do funcionamento correto do acesso vascular. O estudo das aplicações do ultra-som, principalmente na medicina, apresenta grandes desafios para a engenharia e é uma área ainda em aberto, permitindo que novas descobertas e avanços tecnológicos sejam alcançados. Neste âmbito, este projeto tem por objetivo construir um sistema capaz de medir a velocidade do fluxo sangüíneo no acesso vascular, utilizando os princípios de ultra-som e do efeito Doppler, de um modo mais acessível, uma vez que as técnicas atuais de medição normalmente possuem custo elevado ou são pouco práticas. A medição do fluxo sanguíneo é utilizada como métrica na análise do acesso, permitindo a identificação prévia de falhas no mesmo, as quais são as principais causas de morbidade e hospitalização dos pacientes em hemodiálise [1].

Abstract of Graduation Project presented to EESC-USP as a partial fulfillment of the requirements to conclude the Computer Engineering course.

EMBEDDED SYSTEM WITH PULSED DOPPLER ULTRASOUND: INTERFACES BUILDING

Rinaldo Brandolim

06 / 2007

Advisor: Dr. Carlos Dias Maciel

Concentration Area: Biomedical Engineering

Keywords: Ultrasound, Doppler Effect, Pulsed Doppler, Serial Communication, FPGA, VHDL, Java.

ABSTRACT

During hemodialysis process, the blood is removed, filtered and returned to the patient. This requires the creation of a vascular access to allow the exit and entrance of blood. The correct work of the hemodialysis process in patients suffering from renal insufficiency depends directly on the correct work of the vascular access. Studies of the ultrasound applications, mainly in medicine, have presented great challenges to engineering and it is still an open ground that allows new discoveries and technological advances. In this context, the main purpose of this project is to build a system to measure the blood flow velocity in the vascular access in a more accessible way, once that actual techniques usually are expansive or not so practical. This measure can be used to evaluate the access, allowing the previous identification of access failures, which are the main patient morbidity and hospitalization cause in hemodialysis.

1. INTRODUÇÃO

1.1. ORGANIZAÇÃO DA MONOGRAFIA

Neste primeiro capítulo introdutório são apresentadas as motivações para o desenvolvimento do trabalho, o seu objetivo e os materiais e métodos envolvidos na sua construção. O segundo capítulo apresenta uma explicação dos conceitos teóricos envolvidos no projeto. No terceiro capítulo são descritos o estado atual do projeto, os métodos e implementações envolvidos e as atividades realizadas durante o desenvolvimento. Já no quarto capítulo são apresentados os resultados obtidos durante as simulações e testes efetuados com o intuito de validar o trabalho, além das dificuldades e limitações encontradas no decorrer do mesmo. No quinto capítulo é feita uma conclusão, apresentadas as considerações finais, bem como os possíveis trabalhos futuros. Finalmente, no sexto e último capítulo são apresentadas as obras bibliográficas consultadas para a realização do projeto.

1.2. MOTIVAÇÃO

Hemodiálise é o processo artificial de remoção, por filtração, de substâncias tóxicas (como, por exemplo, a uréia) e do excesso de líquido do sangue. Este procedimento é realizado em pacientes que sofrem de insuficiência renal crônica ou aguda, situações estas em que a eliminação de tais substâncias é comprometida devido à falência dos mecanismos excretores renais.

Para a realização desta filtragem é necessária a confecção de um acesso vascular que permita a retirada e o retorno do sangue para o paciente por meio de agulhas de grande calibre. O acesso vascular mais comum é a Fístula Artério-Venosa (FAV), na qual, através de procedimento cirúrgico, uma veia é interconectada a uma artéria, usualmente no pulso. Este procedimento faz com que o fluxo de sangue através da veia aumente, ocorrendo a dilatação e o espessamento de suas paredes tornando-a mais saliente, facilitando a introdução de agulhas para a retirada e retorno do sangue [2]. Além da FAV há ainda o Cateter e o Enxerto Artério-Venoso com material sintético (Prótese).

Falhas no acesso vascular tem sido uma causa freqüente de morbidade e hospitalização dos pacientes de hemodiálise. Pesquisas indicam que a diminuição do fluxo sangüíneo no acesso pode servir como indicativo da ocorrência de falha. A monitoração adequada do acesso permite a identificação prévia dessas falhas, possibilitando a intervenção corretiva em um momento oportuno, aumentando assim, o tempo de vida do acesso e diminuindo as taxas de morbidade e hospitalização.

Normalmente os métodos para avaliação de acessos, tais como o ultra-som duplex e a técnica de diluição de indicador, são caros e demandam grande tempo para sua realização, dificultando a aplicação dos mesmos. No Brasil, em geral não há um programa específico para avaliação de

acessos vasculares, sendo estes analisados através de exames clínicos ou indiretamente através dos alarmes das máquinas de hemodiálise e da análise laboratorial da dose de hemodiálise oferecida ao paciente [2]. No entanto estes métodos não são eficazes, detectando o problema em uma fase já avançada, o que geralmente exige reparo cirúrgico ou um novo acesso, além de diminuir a qualidade de vida do paciente.

Neste contexto, vários estudos têm sido realizados a fim de se desenvolver metodologias alternativas para avaliação de acessos vasculares que sejam mais acessíveis, eficazes e de fácil aplicação clínica. Uma dessas alternativas é o ultra-som Doppler pulsado, que vem sendo utilizado há alguns anos na medicina, e que apresenta ótima resolução espacial e boa sensibilidade para a medição da velocidade do fluxo, além de reduzir custos.

O ultra-som na medicina enfrentou um rápido crescimento, consolidando-se como uma área que provê importantes desafios e problemas de engenharia. É um campo aberto para muitas técnicas desenvolvidas para outras aplicações e serve de inspiração para o desenvolvimento e avanço tecnológico.

1.3. OBJETIVO

O projeto foi desenvolvido como Trabalho de Conclusão de Curso e tem por objetivo desenvolver um sistema de ultra-som Doppler pulsado para permitir a avaliação do acesso vascular de maneira mais acessível e prática, através da análise da velocidade de deslocamento do fluxo. O projeto como um todo foi dividido em duas partes, sendo desenvolvido em parceria com outro estudante de graduação do curso de Engenharia de Computação, Alessandro Tadashi Miyasaka. Mais especificamente, a parte que corresponde ao vigente documento é responsável por, após o processamento do sinal de ultra-som refletido realizado pela outra parte do projeto, enviar os dados necessários, via comunicação serial, da FPGA para o PC, onde será calculada a freqüência da onda refletida, a qual será utilizada para obter a velocidade de fluxo. Por fim, estes valores calculados são exibidos na tela do PC de forma clara para o usuário.

1.4. MATERIAIS E MÉTODOS

Boa parte do projeto como um todo foi desenvolvido em VHDL utilizando o “kit” de desenvolvimento Spartan-3 da Xilinx, o qual pode ser visto na Figura 1.

Figura 1 - “Kit” de desenvolvimento Spartan-3 da Xilinx [3]

O código responsável por receber os dados referentes ao sinal refletido e enviá-los para o PC via comunicação serial foi desenvolvido em VHDL. Depois de enviadas, estas informações são coletadas na porta serial do PC, analisadas e exibidas por um programa feito em Java utilizando-se o ambiente de desenvolvimento Netbeans (Netbeans IDE 5.5.1).

Além dos materiais acima descritos, utilizou-se ainda um transdutor reto de freqüência 2.25 MHz, o equipamento *Ultrasonic Pulser/Receiver* Modelo 5800 da Panametrics conectado ao transdutor, um osciloscópio para analisar os sinais, um “protoboard” para conectar os elementos nas entradas e saídas do “kit” e, para realizar os ensaios finais, também foi usada uma bomba peristáltica.

1.4.1. “KIT” DE DESENVOLVIMENTO SPARTAN-3 XILINX

Este é um “kit” bastante interessante, pois possui várias funcionalidades, apresenta um alto desempenho e custo relativamente baixo. Algumas de suas principais características são [3]:

- Clock interno de 50 MHz
- 4 display de sete segmentos
- Porta serial RS-232 de 9 pinos (DB9)
- 3 portas de expansão de 40 pinos cada para conexão
- 216 kbits de memória RAM, 2 Mbit de Flash e 1Mbyte de SRAM
- 4 botões e 8 chaves

O diagrama de blocos do “kit” pode ser visto na Figura 2.

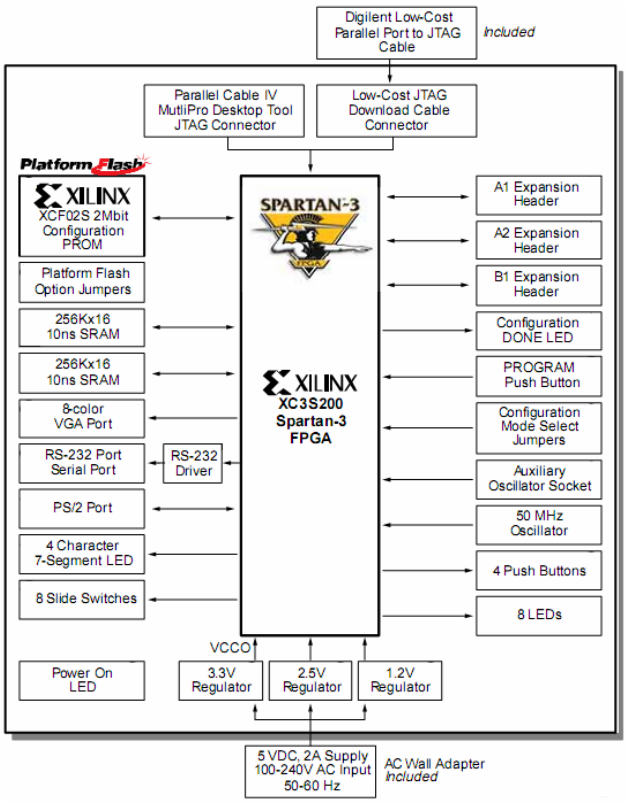

Figura 2 - Diagrama de blocos do “kit” de desenvolvimento Spartan-3 da Xilinx [3]

1.4.2. PANAMETRICS ULTRASONIC PULSER/RECEIVER MODEL 5800

Este equipamento é usado para transmitir e receber o ultra-som. Ele pode ser utilizado como um instrumento de testes com ultra-som em uma grande variedade de metais, plásticos e compostos biomédicos, podendo controlar, de maneira eficiente os sinais de transmissão e de recepção com baixas taxas de ruído e interferência. O equipamento pode ser visto na Figura 3.

Figura 3 - Panametrics Ultrasonic Pulser/Receiver Model 5800

2. FUNDAMENTOS TEÓRICOS

2.1. HEMODIÁLISE

Os pacientes que por algum motivo tiveram a falência de seus excretores renais, sofrendo de insuficiência renal, tem dois métodos de tratamento que substituem as funções do rim: a diálise e o transplante renal.

A diálise é um processo artificial que serve para retirar, por filtração, as substâncias indesejáveis acumuladas pela insuficiência renal. Isto pode ser feito usando a membrana filtrante do rim artificial e/ou a membrana peritoneal. Existem dois tipos de diálise: peritoneal e hemodiálise [4].

A hemodiálise é o procedimento mais freqüente de tratamento para pacientes com insuficiência renal crônica [2]. Nela, o sangue é retirado, filtrado e devolvido ao paciente. Para sua realização é necessária a confecção de um acesso vascular para permitir a saída e o retorno do sangue.

Durante o processo, o sangue é removido do paciente e enviado para a máquina de diálise onde é exposto à solução de diálise (conhecida como dialisato) através de uma membrana semipermeável, permitindo assim, a troca de substâncias entre o sangue e a solução [5] e a remoção de substâncias tóxicas e do excesso de líquido acumulado no sangue e tecidos do corpo, em conseqüência da falência renal. Só então, o sangue é devolvido ao paciente. A Figura 4 apresenta um esquema representativo do circuito de hemodiálise.

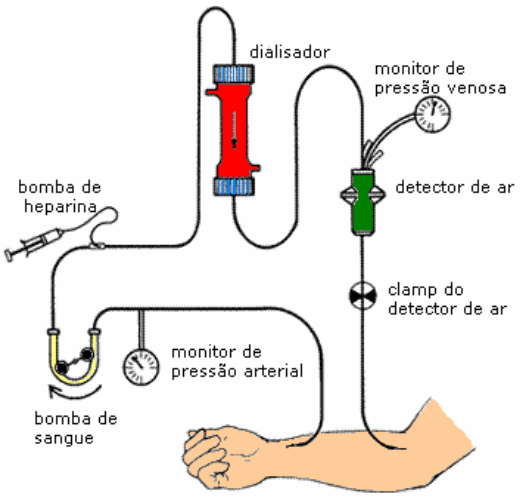

Figura 4 - Esquema representativo do circuito de hemodiálise

2.2. AVALIAÇÃO DE ACESSOS VASCULARES

A manutenção da qualidade do processo de hemodiálise em pacientes que sofrem de insuficiência renal depende diretamente do funcionamento correto do acesso vascular. Um acesso operando corretamente é aquele que proporciona um bom fluxo sangüíneo, apresenta um tempo de utilização adequado e com um baixo índice de complicações.

Falhas em acessos vasculares tem sido a principal causa de morbidade e hospitalização de pacientes de hemodiálise. Programas de avaliação apropriados permitem que as falhas sejam detectadas previamente, permitindo que a intervenção seja feita em um momento oportuno, aumentando o tempo de vida do acesso e diminuindo as taxas de morbidade e hospitalização dos pacientes de hemodiálise.

No Brasil, em geral os acessos vasculares são analisados através de exames clínicos ou indiretamente através dos alarmes das máquinas de hemodiálise e da análise laboratorial da dose de hemodiálise oferecida ao paciente. Estes métodos garantem a dose oferecida ao paciente, mas não fornecem informações que possam ser úteis para evitar falhas nos acessos [2].

Pesquisas indicam que a diminuição do fluxo sangüíneo no acesso pode ser utilizada como fator de predição de falha no acesso. Os métodos mais difundidos para medir o fluxo no acesso são os exames com o ultra-som duplex e as técnicas de diluição. O primeiro é um método caro, trabalhoso e sujeito a erros, pois depende do ângulo do transdutor, da medida do diâmetro do acesso e da velocidade média do fluxo. Já as técnicas de diluição são demoradas, devido à reversão do fluxo da máquina de hemodiálise, coleta de sangue e exame de laboratório.

Para contornar tais dificuldades diversos métodos alternativos vêm sendo estudados e propostos, entre eles o Método de Variação de Fluxo que utiliza um sistema Doppler direcional de ondas contínuas [2] e métodos que utilizam o ultra-som Doppler pulsado.

2.3. ULTRA-SOM

A faixa audível para um ser humano normal encontra-se no intervalo de freqüência que varia de 20 Hz a 20000 Hz (20 kHz) aproximadamente. Abaixo deste intervalo, o sinal é denominado infra-som e acima dele, ultra-som, conforme mostrado na Figura 5.

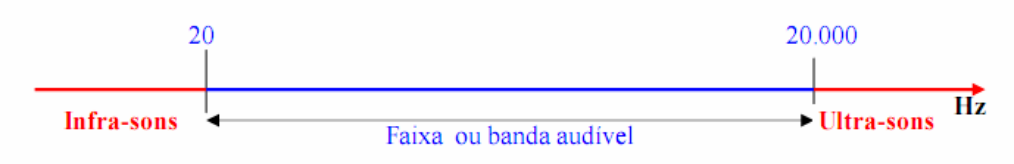

Figura 5 - Faixa de freqüência audível pelo ser humano

O som é uma onda mecânica, ou seja, ondas produzidas por uma perturbação em um determinado meio material – que tenha massa e elasticidade, como os sólidos, líquidos ou gases – e, conseqüentemente, não se propaga no vácuo. A velocidade de propagação depende basicamente de duas características do meio que o transporta: elasticidade e densidade. A elasticidade refere-se ao fato de que toda vez que parte do meio é deslocada de sua posição de equilíbrio ou repouso (deformado) por um agente externo, surge uma força que tende a trazer esta parte de volta para a posição inicial. Já a densidade refere-se à quantidade de massa existente em uma porção unitária do meio. Observa-se que, quanto maior a densidade, isto é, quanto mais próximas as moléculas encontrarem-se umas das outras, maior será a velocidade de propagação. De forma geral, a velocidade nos tecidos moles é de aproximadamente 1540 m/s [2]. A Tabela 1 mostra a velocidade de propagação do som em alguns materiais, bem como a densidade e a impedância acústica dos mesmos.

Tabela 1 - Velocidade do som, densidade e impedância acústica de alguns materiais

Material	Densidade ρ (kg/m^3)	Velocidade v (m/s)	Impedância acústica Z (kg/m^{2}s)
Ar	1,29	3,31 x 10^2	430
Água	1,00 x 10^3	l4,8 x 10^2	1,48 x 10^6
Cérebro	1,02 x 10^3	15,3 x 10^2	1,56 x 10^6
Músculo	1,04 x 10^3	15,8 x 10^2	1,64 x 10^6
Gordura	0,92 x 10^3	14,5 x 10^2	1,33 x 10^6
Osso	1,9 x 10^3	40,4 x 10^2	7,68 x 10^6

2.3.1. IMPEDÂNCIA ACÚSTICA E REFLEXÃO

A impedância acústica de um meio está relacionada com a resistência ou dificuldade que ele impõe a passagem do som, e pode ser calculada como o produto entre a densidade do material (ρ) pela velocidade (v) do som no mesmo, conforme pode ser visto na Equação (2.1) abaixo.

$$Z = \rho \cdot v \qquad (2.1)$$

Quando a onda sonora atravessa uma interface entre dois meios com a mesma impedância acústica, não ocorre reflexão e a onda é toda transmitida para o segundo meio. A quantidade de reflexão que ocorrerá na interface é diretamente proporcional à diferença de impedância entre dois meios, ou seja, quanto maior a diferença, maior será a intensidade de reflexão. E, quanto maior a reflexão do feixe sonoro, maior será a intensidade do eco recebido e, portanto menor a transmissão do som de um meio para o outro.

2.3.2. ATENUAÇÃO DO ULTRA-SOM

Quando uma onda ultra-sônica se propaga através de um material sua intensidade é reduzida devido ao mecanismo da atenuação. A atenuação pode ser provocada por vários fatores, tais como, absorção, espalhamento, reflexão e difração. O processo de absorção envolve a conversão da energia ultra-sônica em uma outra forma de energia (térmica, por exemplo), e depende da natureza do meio de propagação. Os demais processos (reflexão, refração, difração e espalhamento) dependem das características físicas e geométricas do meio e, envolvem mecanismos que fazem com que algumas partes da onda passem a se propagar em direções diferentes da original, enfraquecendo progressivamente a parte da onda que continua na direção original, provocando, assim, a atenuação. A reflexão e a refração ocorrem nas interfaces entre regiões de diferentes impedâncias acústicas. Já a difração ocorre quando barreiras de tamanho finito são interpostas no caminho do feixe. E as perdas por espalhamento são características da estrutura do material, relacionadas ao descasamento da impedância acústica e ao fato do espalhador ter tamanho da mesma ordem de grandeza do comprimento de onda.

A atenuação é normalmente medida em decibéis por unidade de comprimento e é diretamente proporcional à freqüência do transdutor, ou seja, quanto maior a freqüência maior será a atenuação do feixe sonoro. A Tabela 2 mostra a atenuação do ultra-som a 1 MHz em alguns tecidos do corpo humano.

Tabela 2 - Atenuação do ultra-som

Tecido	Atenuação (dB/cm) a 1MHz
Gordura	0,63
Músculo	1,3
Osso	10

2.3.3. ULTRA-SOM NA MEDICINA

A história do ultra-som remonta ao final do século XVIII através de descobertas realizadas em morcegos. Durante muito tempo, o estudo do ultra-som foi impulsionado por objetivos militares e industriais, sendo que as pesquisas sobre aplicações médicas ocorreram com maior intensidade somente após a segunda guerra mundial. Desde então, as técnicas ultra-sônicas têm se tornado cada vez mais importantes na medicina, tanto como ferramenta para diagnósticos, quanto como modalidade terapêutica, possibilitando atualmente, por exemplo, estudar a válvula do coração e observar um feto.

As características não invasivas do ultra-som e sua capacidade de distinguir interfaces entre tecidos com diferentes impedâncias acústicas é que o torna bastante atrativo como um procedimento de diagnóstico. Em contrapartida, os raios-X, por exemplo, só respondem a diferenças de pesos atômicos podendo necessitar que um meio mais denso seja injetado para servir como intermediador e permitir a visualização de certos tecidos.

O uso do ultra-som para diagnósticos baseia-se na reflexão das ondas ultra-sônicas pelos elementos a serem analisados, conforme esquema da Figura 6.

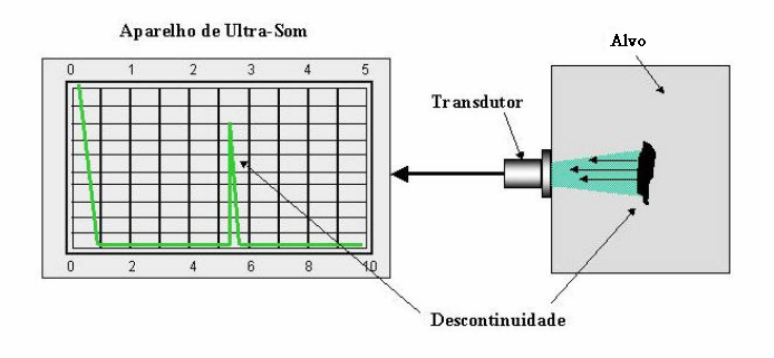

Figura 6 – Reflexão das ondas de ultra-som.

A faixa de freqüência de 1 a 20 MHz cobre a grande maioria das aplicações médicas existentes, sendo que as mais utilizadas para diagnóstico médico encontram-se no intervalo de 2 a 10 MHz. Dentro desta faixa, para se avaliar vasos sangüíneos logo abaixo da pele, utilizam-se freqüências maiores e para vasos mais profundos, freqüências menores. Isto se deve a relação diretamente proporcional entre a freqüência e a atenuação (quanto maior a freqüência maior a atenuação).

O sangue é constituído de plasma, eritrócitos (células vermelhas), leucócitos (células brancas) e plaquetas (pequenas partículas vitais para o processo de coagulação). Devido ao seu tamanho e número, os principais componentes do sangue responsáveis pelo espalhamento do ultra-som emitido por um transdutor são os eritrócitos [2]. Como o eritrócito é bem menor que o comprimento de onda do ultra-som, o espalhamento é independente do seu formato. Depende apenas do volume e da diferença de impedância em relação ao meio que o envolve. O espalhamento é proporcional à freqüência do ultra-som elevada a quarta e ocorre quando o eritrócito absorve energia do feixe de ultra-som, vibra e re-irradia ultra-som em todas as direções. Parte deste atinge o transdutor receptor e corresponde ao sinal de interesse.

2.3.4. GERAÇÃO DO ULTRA-SOM

As ondas de ultra-som são geradas por transdutores ultra-sônicos (ou simplesmente, transdutores). Um transdutor, de forma geral, é um dispositivo que converte um tipo de energia em outro, portanto, o transdutor ultra-sônico converte energia elétrica em energia mecânica e vice-versa. Esses transdutores são construídos a partir de materiais que apresentam efeitos piezelétricos [6].

O efeito piezelétrico foi descoberto por Pierre e Jacques Curie em 1880 e consiste na variação das dimensões físicas (causadas por pressões acústicas) de certos cristais, tais como o quartzo, turmalina e cristais naturais. Essas variações nas dimensões de materiais piezelétricos provocam o aparecimento de campos elétricos neles. Tal fenômeno é obtido aplicando-se eletrodos no cristal piezelétrico com tensão elétrica alternada da ordem de centenas de Volts, de maneira que o mesmo se contrai e se estende ciclicamente. Se tentarmos impedir esse movimento a placa transmite esforços de compressão as zonas adjacentes, emitindo uma onda longitudinal, cuja forma depende da freqüência de excitação e das dimensões do cristal.

É possível também realizar o efeito contrário nesses materiais, causando o que é chamado de efeito magnetoestritivo. Ao se colocar um material piezelétrico sob um campo elétrico, as cargas elétricas da rede cristalina interagem com o mesmo e produzem tensões mecânicas. Assim, o cristal piezelétrico pode transformar a energia elétrica alternada em oscilação mecânica e vice-versa. Isto permite que o mesmo transdutor que emite o sinal ultra-sônico possa funcionar também como detector. Esses fenômenos podem ser vistos na Figura 7.

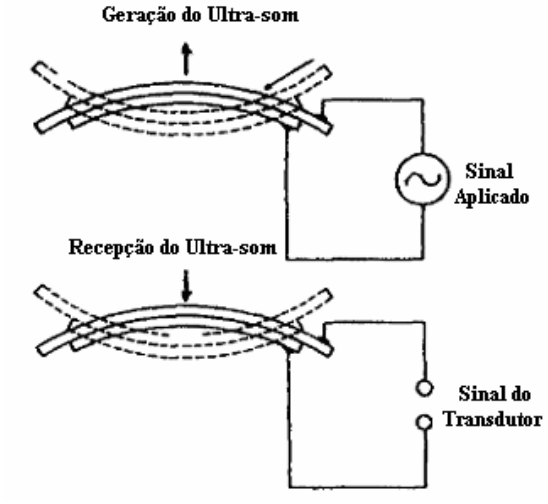

Figura 7 - Efeito Piezelétrico e geração do ultra-som

Existem basicamente três tipos usuais de transdutores: o reto ou normal, o angular e o duplo-cristal [7].

a. TRANSDUTORES RETOS OU NORMAIS

São aqueles que possuem monocristal gerador de ondas longitudinais normal à superfície de acoplamento. Esses transdutores são construídos a partir de um cristal piezelétrico que é colado em um bloco rígido denominado amortecedor que serve de apoio para o cristal e absorve as ondas emitidas pela face colada a ele [7]. A Figura 8 ilustra um transdutor reto ou normal.

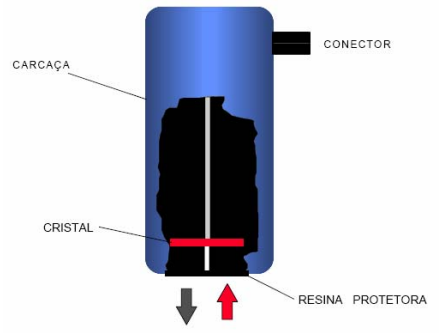

Figura 8 - Transdutor reto ou normal

Em geral os transdutores normais são circulares, com diâmetros de 5 a 24 mm, com freqüência de 0,5; 1; 2; 2,5; 5 e 6 MHz. Outros diâmetros e freqüências existem, porém para aplicações especiais [7].

b. TRANSDUTORES ANGULARES

Diferentemente dos transdutores retos ou normais, o cristal dos transdutores angulares forma um determinado ângulo com a superfície do material. Este ângulo é obtido inserindo uma cunha de plástico, que também funciona como amortecedor para o cristal piezelétrico após a emissão dos impulsos, entre o cristal piezelétrico e a superfície. A cunha de plástico pode ser fixa (englobada pela carcaça) ou intercambiável, sendo esta última uma solução mais econômica, visto que normalmente opera-se com diferentes ângulos (35, 45, 60, 70 e 80 graus) dependendo da mudança de velocidade no meio [7].

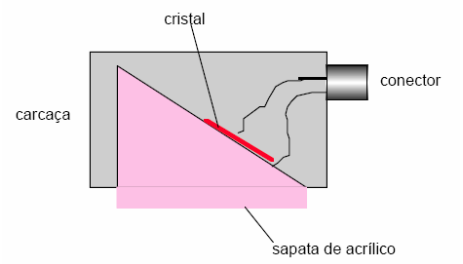

Figura 9 - Transdutor angular

Seu cristal piezelétrico possui dimensões que podem variar entre 8 x 9 mm até 15 x 20 mm e somente recebe ondas ou impulsos ultra-sônicos que penetram na cunha em uma direção paralela à de emissão, em sentido contrário [7].

c. TRANSDUTORES DUPLO-CRISTAL OU SE

Certas situações, tais como quando se trata de inspecionar ou medir materiais de reduzida espessura, ou quando se deseja detectar descontinuidades logo abaixo da superfície do material, não podem ser resolvidas com a utilização de transdutores retos ou normais nem com os angulares. Isto ocorre, pois o cristal piezelétrico recebe o eco num espaço de tempo muito curto após a emissão, não tendo suas vibrações amortecidas suficientemente. Nestes casos, somente um transdutor que separa a emissão da recepção pode ajudar. Para tanto, desenvolveu-se o transdutor de duplo-cristal, no qual dois cristais são incorporados na mesma carcaça, separados por um material acústico isolante e levemente inclinado em relação à superfície de contato, sendo que um deles funciona como emissor e o outro como receptor.

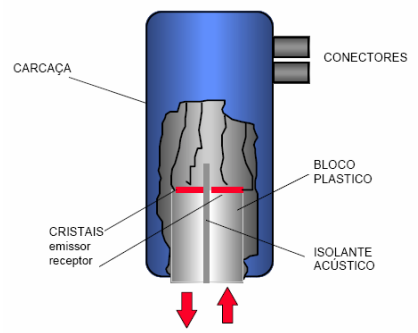

Figura 10 - Transdutor duplo

Os transdutores duplos possuem sempre uma faixa de inspeção ótima que deve ser observada. Fora desta zona a sensibilidade se reduz não podendo, assim, serem utilizados para qualquer distância (profundidade). Em certos casos estes transdutores duplos são utilizados com “focalização”, isto é, o feixe é concentrado em uma determinada zona do material para a qual se deseja máxima sensibilidade [7].

2.4. SISTEMAS DOPPLER

O conhecimento dos fenômenos ondulatórios que ocorrem na natureza (som, luz, etc.) permitiu o desenvolvimento de uma vasta gama de tecnologias aplicáveis a quase todas as atividades humanas. Em particular, a descoberta do Efeito Doppler permitiu um avanço bastante significativo no desenvolvimento de novas técnicas de medição com ultra-som [8].

Efeito Doppler é a mudança na freqüência aparente de uma onda quando esta é emitida ou refletida por um objeto que se encontra em movimento em relação ao observador. No caso de o objeto estar se aproximando do observador, as ondas o alcançarão mais rapidamente aumentando assim a freqüência recebida. Caso contrário, se o objeto estiver se afastando do observador a freqüência recebida diminui. Este fenômeno foi descrito teoricamente pela primeira vez em 1842 pelo físico austríaco Johann Christian Andreas Doppler e comprovado pelo cientista alemão Christoph B. Ballot em 1845 em um experimento com ondas sonoras [5]. Este mesmo fenômeno foi também observado em ondas eletromagnéticas em 1848 pelo francês Hippolyte Fizeau, sendo por isso, também denominado Efeito Doppler-Fizeau.

Um exemplo deste efeito pode ser visto na Figura 11, na qual a fonte de ondas se locomove para a esquerda. É possível observar que a freqüência é menor no lado direito da imagem (do qual a fonte está se afastando) e maior no lado esquerdo (do qual a fonte está se aproximando).

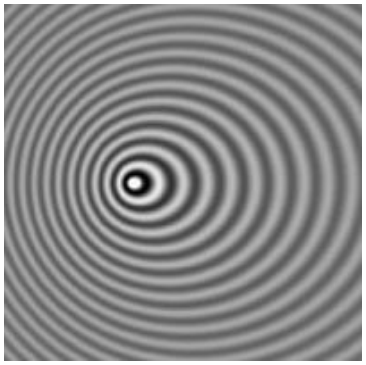

Figura 11 - Exemplo da ocorrência de Efeito Doppler [9]

Se todas as velocidades encontram-se sobre um mesmo eixo, a freqüência recebida f_r é dada pela Equação 2.2 abaixo [10]:

$$f_r = \left(\frac{c - v_r}{c - v_s} \right) f \quad (2.2)$$

Onde v_r é a velocidade do receptor e v_s é a velocidade da fonte, ambas sendo tomadas na mesma direção que a propagação da onda; e f é a freqüência da fonte.

A equação acima pode ser modificada a fim de se obter o valor da freqüência de deslocamento de Doppler, $f_D = (fr - f)$:

$$f_D = \left(\frac{c - v_r}{c - v_s} - 1 \right) f \qquad (2.3)$$

Há várias aplicações em diversas áreas envolvendo os conceitos do Efeito Doppler. De maneira geral, ele é muito utilizado para realizar medições de velocidade. Em astronomia, é possível medir a velocidade relativa das estrelas e outros objetos celestes luminosos em relação à Terra e determinar se eles estão se aproximando ou se afastando. É também utilizado em radares para determinar a velocidade de determinados objetos através da reflexão das ondas emitidas pelo próprio equipamento de medição [9].

Na medicina, o aproveitamento do Efeito Doppler em ondas ultra-sônicas propagando-se através dos tecidos biológicos é a base de uma série de técnicas para estudo, de forma não invasiva, das estruturas móveis do interior do corpo humano. O efeito é utilizado em imagens médicas e na análise do fluxo sangüíneo. Um ecocardiograma utiliza o Efeito Doppler para medir a direção e a velocidade do fluxo sangüíneo ou do tecido cardíaco. Para a medida de velocidade do sangue, o ultra-som é transmitido em direção a uma veia e o som que é refletido pelos eritrócitos (células vermelhas) é detectado. Neste caso, o deslocamento (f_d) – diferença entre a freqüência transmitida (f_t) e a recebida (f_r) – é dependente da velocidade do sangue (v) e do ângulo entre o feixe de som e a direção do fluxo sangüíneo (θ) [11] denominado ângulo Doppler, conforme expresso na Equação 2.4:

$$f_d = f_t - f_r = \frac{2 v f_t \cos\theta}{c} \qquad (2.4)$$

Onde:

v = velocidade da partícula que interage com a onda de ultra-som incidente.

θ = ângulo entre a direção de propagação do ultra-som e a direção de movimento da partícula (ângulo Doppler).

c = velocidade do som no meio.

A equação acima pode ainda ser manipulada para encontrar o valor da velocidade do sangue (v). O ângulo Doppler (θ) normalmente é estimado pelo próprio operador do transdutor, não tendo como ser medido automaticamente. A faixa de valores recomendada é que este esteja entre 0 e 60º (devido à função co-seno) a fim de se obter um resultado mais preciso.

A medição do fluxo sangüíneo permite identificar falhas em acessos vasculares e válvulas cardíacas, sendo um eficiente meio para diagnosticar problemas vasculares como a estenose [9]. Há

duas técnicas principais para se medir a velocidade do fluxo sangüíneo através de um ultra-som Doppler [12]: Doppler de onda contínua e Doppler pulsado.

2.4.1. DOPPLER CONTÍNUO

Como o nome sugere, sistemas Doppler de onda contínua realizam a transmissão e recepção contínua de ultra-som. Os sinais Doppler são obtidos a partir de todas as veias no caminho do feixe de ultra-som (até que o feixe se torne suficientemente atenuado com a profundidade), conforme mostrado na Figura 12. Esta pode ser considerada uma desvantagem dos sistemas Doppler de onda contínua, uma vez que pode haver diferentes fluxos na direção das ondas emitidas pelo transdutor, não tendo como selecionar uma profundidade pré-determinada para analisar o fluxo em um ponto em particular, sendo também impossível obter a distância entre o transdutor e o objeto em movimento. Sendo assim, o Doppler de onda contínua não permite determinar a localização específica das velocidades dentro do feixe e não é utilizado para produzir imagens de fluxo coloridas.

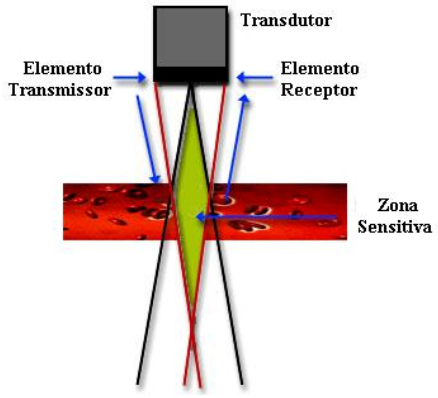

Figura 12 - Doppler de onda contínua

Um exemplo do problema da interferência dos sinais refletidos por várias fontes no caminho do feixe de onda contínua é mostrado na Figura 13.

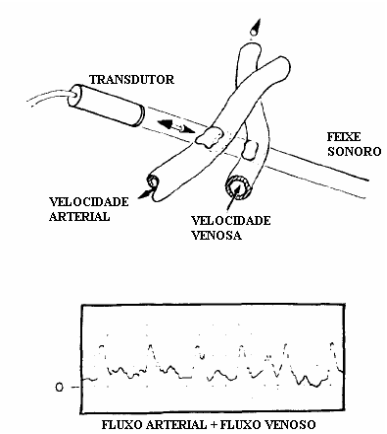

Figura 13 - Equipamentos Doppler de onda contínua irão detectar o fluxo sangüíneo de todas as veias no caminho do feixe sonoro

A principal vantagem da utilização do sistema Doppler de onda contínua é a possibilidade de medir, de forma simples, a direção e a velocidade dos vasos sangüíneos mais superficiais.

Uma propriedade fundamental de todos os métodos para medir a velocidade e a posição ao mesmo tempo é que estas são relacionadas. Em termos físicos é impossível determinar a velocidade de um alvo cuja posição seja precisamente conhecida, uma vez que algum movimento é necessário para permitir que a velocidade seja medida.

2.4.2. DOPPLER PULSADO

Diversas técnicas foram desenvolvidas há alguns anos para o caso de radares, nas quais o sinal transmitido era codificado no tempo de modo que a informação de alcance pudesse ser obtida a partir de atrasos de tempo, enquanto que a informação de velocidade era obtida, normalmente, na forma de deslocamentos da freqüência de Doppler [10]. Este é o princípio básico do Doppler pulsado. Técnicas análogas foram aplicadas na Medicina. Elas permitem analisar os movimentos de vários alvos simultaneamente, e as características dos fluxos sangüíneos podem ser traçadas em um gráfico.

Aparentemente, Baker e Watkins (1967) [10] foram os primeiros a perceber a possibilidade de se obter sinais Doppler de uma área específica fazendo com que um oscilador transmitisse pulsos, permitindo a obtenção de informações a respeito da velocidade dos alvos dentro de uma determinada área comparando-se a fase dos sinais refletidos (eco) com a do sinal transmitido pelo oscilador. Além disso, o tempo decorrido entre a transmissão do pulso e a chegada do sinal refletido determina a profundidade da medida de velocidade, ou seja, a posição da amostra. Os sinais refletidos atingem o receptor em tempos distintos de acordo com a distância entre a

estrutura refletora e o receptor e, como a velocidade do ultra-som é constante nos tecidos do corpo humano, é possível analisar somente um determinado intervalo de tempo, criando-se uma espécie de “janela”, dentro da qual os ecos recebidos podem ser analisados sem que ocorra a interferência de informações irrelevantes. A Figura 14 ilustra um sistema Doppler pulsado.

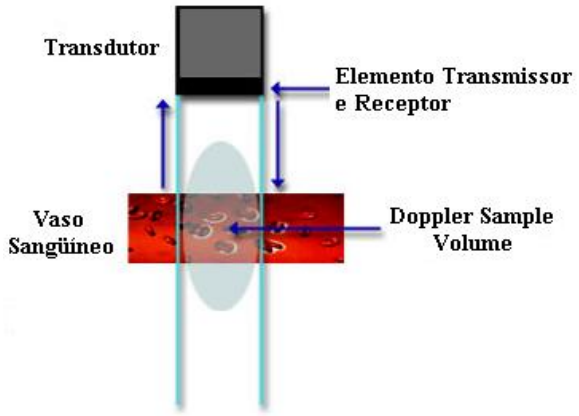

Figura 14 - Doppler de onda pulsada

A princípio não há limite superior para o deslocamento de freqüência de Doppler que pode ser medido por este método. Na prática, entretanto, o limite superior de freqüência (o qual está relacionado com a máxima velocidade que pode ser medida – Limite de Nyquist) que pode ser detectado sem ambigüidade depende da taxa de amostragem. A taxa de amostragem máxima é limitada pelo tempo de transito ultra-sônico para o alvo de interesse, e pelo tempo de declínio de reverberação. Se um determinado sinal possui um espectro de freqüência que se estende desde zero até uma freqüência máxima (f_{max}), é possível transmitir toda a informação no sinal a uma taxa de $2f_{max}$ (ou mais) amostras da amplitude do sinal igualmente espaçadas por segundo. Na prática, este máximo teórico não pode ser alcançado, sendo limitado pela dificuldade de se projetar filtros capazes de rejeitar o sinal na freqüência de amostragem.

Por exemplo, considere os fatores envolvidos na medida do fluxo sangüíneo em uma veia utilizando Doppler pulsado. Uma freqüência de 2 MHz pode ser escolhida, limitando assim a taxa de repetição a aproximadamente 1000 Hz, pelo compromisso entre penetração e reverberação. A teoria da amostragem limita a freqüência de deslocamento de Doppler máxima a 500 Hz; na prática, o limite superior dificilmente será superior a 400 Hz, o que corresponde a uma velocidade máxima do alvo de 150 mm s^{-1} (infelizmente, esta taxa é muito baixa para permitir que movimentos rápidos no coração sejam estudados: a velocidade na válvula mitral, por exemplo, pode atingir 500 mm s^{-1}). Para qualquer freqüência pode ser mostrado que o produto da máxima velocidade e o alcance máximo é igual a uma constante. Assim,

considerações similares podem ser aplicadas ao fluxo sangüíneo próximo ao transdutor: nesta situação, freqüências de 4-35 MHz têm sido utilizadas, escolhidas de modo a alcançar a melhor resolução possível.

Medidas de velocidade e alcance (posição) estão interligadas, portanto melhorar a resolução de alcance implica em diminuir a resolução referente à velocidade. Assim, se o tempo analisando o alvo for infinitamente pequeno, uma resolução de alcance perfeita pode ser alcançada; mas se o alvo não se move em um período de tempo infinitamente pequeno, e então nenhuma informação a respeito da velocidade pode ser obtida.

2.5. COMUNICAÇÃO SERIAL

Comunicação serial refere-se ao processo de enviar dados, um bit de cada vez, seqüencialmente, em um determinado canal de comunicação ou barramento. É utilizada nas comunicações de longa distância e na maioria das redes de comunicação, situações nas quais o custo dos cabos e a dificuldade de sincronização tornam a comunicação paralela (processo em que vários bits são enviados ao mesmo tempo, paralelamente, sobre múltiplos canais) impraticável. Os barramentos seriais dos computadores tornaram-se cada vez mais comuns a medida que avanços tecnológicos possibilitavam que os dados fossem transmitidos em uma velocidade maior.

Na comunicação serial os seguintes quatro parâmetros devem ser especificados:

- A taxa de transmissão (B*aud Rate*).
- O número de bits da informação sendo transmitida.
- Informação a respeito do bit de paridade (se está sendo utilizado ou não).
- O número de *stop bits*.

Cada pacote de informação transmitido é constituído por um único *start bit*, seguido pelos bits de dados, o bit de paridade (opcional) e o(s) *stop bit*(s).

A taxa de transferência (*baud rate*) refere-se à velocidade com que os dados são enviados através de um canal e é medida em transições elétricas por segundo. A norma EIA232 (RS-232) utiliza somente dois níveis de voltagem (*mark* e *space*) e, portanto, ocorre uma transição de sinal por bit, resultando que a taxa de transferência e a taxa de bits (*bit rate*) são iguais. Nesse caso, por exemplo, uma taxa de 9600 *bauds* corresponde a uma transferência de 9600 dados (bits) por segundo, ou um período de aproximadamente, 104 ms (1/9600 s). Este exemplo é ilustrado na Figura 15.

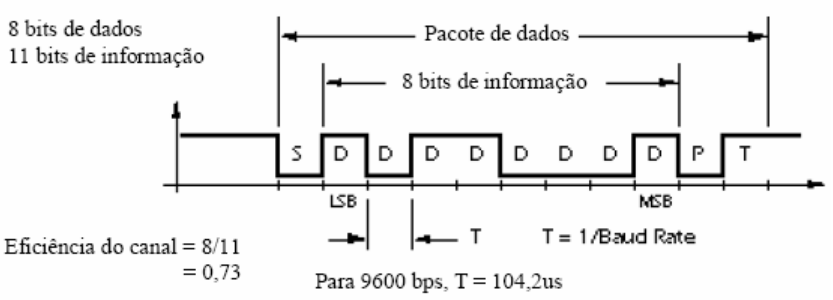

Figura 15 - Exemplo de transmissão serial

A taxa de transmissão máxima permissível de uma mensagem é diretamente proporcional a potência do sinal, e inversamente proporcional ao ruído. A função de qualquer sistema de comunicação é fornecer a maior taxa de transmissão possível, com a menor potência e com o menor ruído possível [13].

2.5.1. TRANSMISSÃO SÍNCRONA E ASSÍNCRONA

Geralmente os dados serializados são enviados em pacotes, conforme descrito anteriormente. Os pacotes são enviados um após o outro, seguido de uma pausa entre eles. Normalmente esta pausa não possui um tamanho fixo. O circuito receptor dos dados deve saber o momento correto para ler os bits individuais do canal, saber exatamente quando um pacote começa e quanto tempo decorre entre os bits. Quando essa temporização é conhecida, o receptor é dito estar sincronizado com o transmissor, e a transferência correta dos dados torna-se possível. Falhas na sincronização causarão a corrupção ou perda dos dados.

Duas técnicas básicas são empregadas para garantir a sincronização entre o transmissor e o receptor [13]. Uma delas consiste em utilizar canais separados para transmitir dados e informação de tempo (sincronização). Neste último são transmitidos pulsos de *clock* para o receptor. Quando um pulso de *clock* é recebido, o receptor lê o canal de dados e armazena o valor do bit encontrado naquele momento. O canal de dados não é lido novamente até que o próximo pulso de *clock* chegue. Como o transmissor é responsável pelos pulsos de dados e de temporização, o receptor irá ler o canal de dados apenas quando comandado pelo transmissor e, portanto a sincronização é garantida. Este mecanismo pode ser visto na Figura 16.

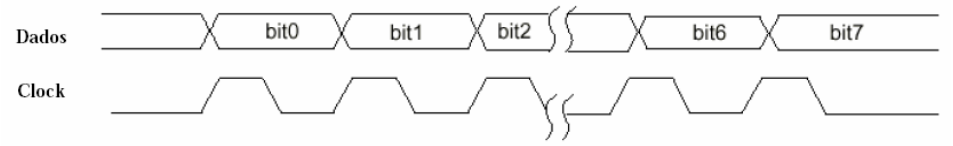

Figura 16 - Transmissão serial síncrona

Uma outra técnica empregada para garantir a sincronização consiste em compor o sinal de *clock* e de dados em um único canal. Isso é usual quando transmissões síncronas são enviadas através de um modem. Dois métodos nos quais os sinais de dados contêm informação de tempo são: codificação NRZ (*Non-Return-to-Zero*) e a codificação Manchester [13].

Já em sistemas assíncronos, a informação trafega por um canal único. O transmissor e o receptor devem ser configurados antecipadamente para que a comunicação se estabeleça a contento. Um oscilador preciso no receptor irá gerar um sinal de *clock* interno que é igual (ou muito próximo) ao do transmissor. Na transmissão assíncrona é necessária a utilização de um protocolo de comunicação para que esta se proceda de maneira correta entre o transmissor e o receptor. No protocolo serial mais comum, os dados são enviados em pequenos pacotes de 10 ou 11 bits, dos quais 8 constituem a mensagem. Quando o canal está em repouso, o sinal correspondente no canal tem um nível lógico ‘1’. Um pacote de dados sempre começa com um nível lógico ‘0’ (*start bit*) para sinalizar ao receptor que a transmissão foi iniciada. O *start bit* faz com que seja inicializado um temporizador interno no receptor avisando que a transmissão começou e que serão necessários pulsos de *clock*. Seguido do *start bit*, 8 bits de dados da mensagem são enviados na taxa de transmissão especificada. O pacote é concluído com os bits de paridade (opcional) e de parada (*stop bit*). A Figura 17 ilustra a comunicação assíncrona.

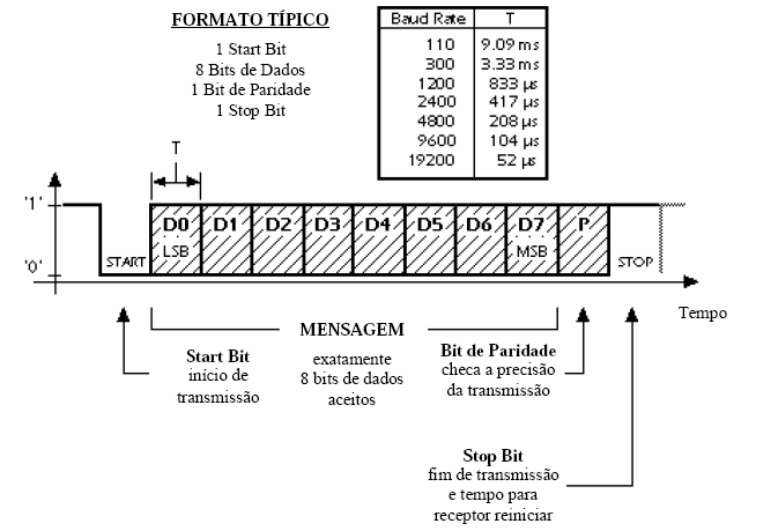

Figura 17 - Transmissão serial assíncrona

O comprimento do pacote de dados é pequeno em sistemas assíncronos para minimizar o risco do oscilador do transmissor e do receptor se destoar. A cada novo pacote enviado, o *start bit* reinicia a sincronização, portanto a pausa entre pacotes pode ser longa e variável.

2.5.2. RS-232

RS-232 (*Recommended Standard* 232) é um padrão para comunicação de dados serial entre equipamentos desenvolvido na década de 60 por um comitê hoje denominado *Electronic Industries Alliance* (EIA). O padrão define, dentre outros aspectos:

- As características elétricas do sinal, tais como níveis de tensão, sinalização de taxa, temporização, capacitância, comprimento do cabo.
- Características mecânicas da interface, conectores e identificação dos pinos.
- Função de cada circuito na interface do conector.
- Subconjunto de interfaces padrão para certas aplicações de telecomunicações.

No entanto, o padrão não especifica o tipo de codificação dos caracteres (por exemplo, ASCII ou EBCDIC), protocolos para detecção de erros ou algoritmos para compressão de dados.

Recebeu várias denominações durante sua história, tais como EIA RS 232, EIA 232 e mais recentemente TIA 232. O padrão continua a ser revisado e atualizado pela EIA e desde 1988 pela *Telecommunications Industry Association* (TIA) também. A revisão C foi publicada em 1969 e a D em 1986. A revisão atual, TIA-232-F (*Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*) foi liberada em 1997.

Desde a revisão C, algumas linhas foram renomeadas, outras foram adicionadas e algumas modificações na temporização foram realizadas com o intuito de melhorar a harmonização com o padrão V.24 da CCITT. No entanto, os equipamentos construídos seguindo as novas revisões são compatíveis com as versões antigas [9].

Se a norma EIA 232 completa for implementada, o equipamento que faz o processamento dos sinais é chamado DTE (*Data Terminal Equipment* – usualmente um computador ou terminal), tem um conector DB25 macho, e utiliza 22 dos 25 pinos disponíveis para sinais ou terra. O equipamento que faz a conexão (normalmente uma interface com a linha telefônica) é denominado de DCE (*Data Circuit-terminating Equipment* – usualmente um modem), tem um conector DB25 fêmea, e utiliza os mesmos 22 pinos disponíveis para sinais e terra. Um cabo de conexão entre dispositivos DTE e DCE contém ligações em paralelo, não necessitando mudanças na conexão de pinos. Se todos os dispositivos seguissem essa norma, todos os cabos seriam idênticos, e não haveria chances de haver conexões incorretas [13]. Na prática não se utiliza a implementação completa, usando-se apenas o que for necessário para a aplicação em questão. Normalmente, ao invés do conector DB25 utiliza-se um DB9, cuja pinagem pode ser conferida na Figura 18, na qual os sinais mais comuns são mostrados em negrito.

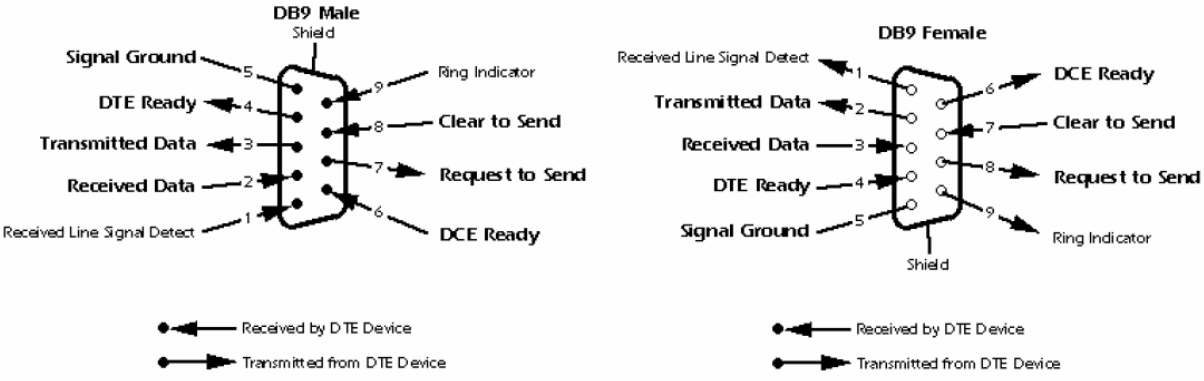

Figura 18 - Conector DB9

A Figura 19 ilustra a convenção mais comum utilizada na conexão dos principais sinais.

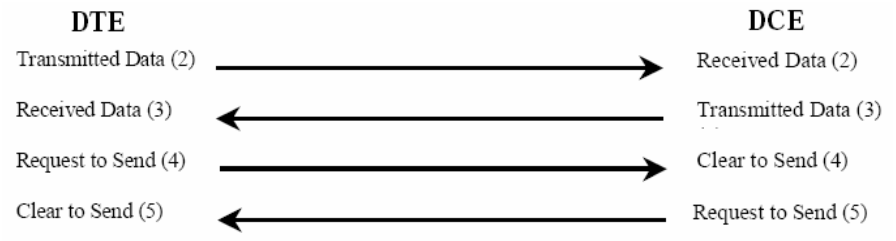

Figura 19 - Convenção na conexão dos sinais mais comuns entre o DTE e o DCE

Para representação elétrica do sinal é utilizada a lógica negativa, conforme pode ser vista na Figura 20.

STATUS		
nível lógico	1	0
nível de tensão	- 25V a -3V	+ 3 V a + 25V
função	OFF	ON
condição do sinal	MARK	SPACE

Figura 20 - Características elétricas RS-232

A faixa de tensões entre –3 volts e +3 volts é considerada uma região de transição para o qual o estado do sinal é indefinido.

2.6. FPGA

FPGA (*Field-Programmable Gate Array*) é um dispositivo semicondutor, largamente empregado para processamento de informações digitais, que contém interconexões e componentes lógicos que podem ser programados de acordo com as necessidades da aplicação sendo desenvolvida. Possui suas origens nos CPLDs (*Complex Programmable Logic Devices*) e foi criado pela Xilinx Inc. na década de 80 [9].

Os componentes lógicos podem ser programados para implementar a funcionalidade das portas lógicas básicas (*AND, OR, XOR, NOT*) ou funções combinacionais mais complexas como decodificadores ou funções matemáticas. Na maioria dos FPGAs, esses componentes lógicos (ou blocos lógicos) também incluem elementos de memória, que podem ser simples flip-flops ou blocos mais completos.

A vasta hierarquia de interconexões programáveis permite que os blocos lógicos de um FPGA sejam conectados de acordo com as necessidades, aumentando a flexibilidade do desenvolvimento.

Além da alta flexibilidade, os projetos em FPGAs possuem outras vantagens, tais como bom *time-to-market*, capacidade de reprogramação em campo (*in field*) para correção de possíveis erros, facilidade de desenvolvimento. Algumas desvantagens que poderiam ser citadas são o elevado custo inicial (normalmente FPGAs são relativamente caras) e o consumo um pouco elevado, apesar de isso estar sendo melhorado.

O FPGA é composto basicamente por três tipos de componentes: blocos de entrada e saída (IOB), blocos lógicos configuráveis (CLB) e chaves de interconexão (*Switch Matrix*). Os blocos lógicos formam uma matriz bidimensional, e as chaves de interconexão são dispostas em formas de trilhas verticais e horizontais entre as linhas e as colunas dos blocos lógicos. Os canais de roteamento possuem chaves de interligação programáveis que permitem conectar os blocos lógicos de maneira conveniente, em função das necessidades de cada projeto [5].

No interior de cada bloco lógico do FPGA existem vários modos possíveis para implementação de funções lógicas. O mais utilizado pelos fabricantes de FPGA é o bloco de memória LUT (*Look-Up Table*). Esse tipo de bloco lógico contém células de armazenamento que são utilizadas para implementar pequenas funções lógicas. Cada célula é capaz de armazenar um único valor lógico (zero ou um).

Para programar o FPGA e definir um comportamento ao mesmo utiliza-se uma linguagem de descrição de hardware (HDL – *Hardware Description Language*) ou esquemático. As linguagens de descrição de hardware mais comuns são o VHDL e o Verilog. A fim de diminuir a complexidade envolvida no projeto utilizando-se HDL, tem-se trabalhado para aumentar o nível de abstração do projeto. Nesse contexto, foi desenvolvido o *SystemC*, uma linguagem para descrição de sistemas

que permite combinar linguagem de alto nível com modelos concorrentes e permitir ciclos de projeto mais rápido para FPGA do que quando se utiliza HDL.

2.7. VHDL

VHDL, ou VHSIC HDL (*Very-High-Speed Integrated Circuit Hardware Description Language*), como o próprio nome diz, é uma linguagem de descrição de hardware usada para facilitar o projeto de circuitos digitais em FPGAs e ASICs (*Application-Specific Integrated Circuits*). Foi desenvolvida originalmente sob o comando do departamento de defesa norte americano, em meados da década de 80, para documentar o comportamento de ASICs. Isto quer dizer que a linguagem VHDL foi desenvolvida para substituir os complexos manuais que descreviam o funcionamento dos ASICs [5]. Procurava-se uma linguagem para descrever hardware que fosse legível tanto por máquinas quanto por humanos e estritamente forçasse o desenvolvedor a escrever código estruturado e compreensível, de forma que o próprio código-fonte servisse como uma espécie de documento de especificação.

O desenvolvimento da VHDL serviu inicialmente aos propósitos de documentação do projeto VHSIC. Entretanto, nesta época buscava-se uma linguagem que facilitasse o projeto de um circuito, ou seja, desenvolver o circuito a partir de uma descrição textual ou um algoritmo, sem a necessidade de especificar explicitamente as ligações entre componentes. VHDL presta-se adequadamente a tais propósitos, podendo ser utilizada para as tarefas de documentação, descrição, síntese, simulação, teste, verificação formal e ainda compilação de software, em alguns casos [5].

Após seu sucesso inicial, sua definição foi posta em domínio público, o que a levou padronização pelo IEEE (*Institute of Electrical and Electronic Engineers*) em 1987. O fato de ser padronizada e de domínio público ampliou ainda mais a sua utilização, novas alterações foram propostas e a linguagem sofreu uma revisão e um novo padrão mais atualizado foi lançado em 1993. Atualmente ela continua a ser revisada e extensões têm sido adicionadas visando tornar a escrita e o gerenciamento de código mais simples.

VHDL é uma linguagem de propósito geral, embora seja necessária a utilização de um simulador para rodar o código. A grande maioria dos simuladores é paga, apesar de alguns fabricantes fornecerem versões gratuitas, mas geralmente limitadas, de seu software. Ainda hoje existem poucos simuladores *open source*. Além do simulador, é altamente recomendada a utilização de uma ferramenta de síntese, mesmo que não se planeje testar o código em um hardware real. A razão para isso é que se pode visualizar a representação gráfica do código após a análise.

O ponto chave da linguagem é que quando utilizada para projetar sistemas ela permite que o comportamento do mesmo seja descrito (modelado) e verificado (simulado) antes que as

ferramentas de síntese traduzam o projeto para o hardware real (portas e fios). Outro benefício é que VHDL permite a descrição de sistemas concorrentes (cada qual com seu próprio comportamento, trabalhando ao mesmo tempo).

2.8. JAVA

Java é uma linguagem de programação orientada a objetos desenvolvida pela *Sun Microsystems* no início dos anos 90.

As aplicações Java são compiladas para *bytecodes* que são executados por uma máquina virtual (programa escrito em código nativo que interpreta *bytecodes* Java genéricos), embora a compilação para código de máquina nativo atualmente seja possível e mais rápida. Em tempo de execução, os *bytecodes* são interpretados ou compilados para código nativo para que possam ser executados, embora a execução direta do *bytecode* por um processador Java também já seja possível. É justamente o fato de Java utilizar uma representação intermediária de código (*bytecode*) que é interpretada pelas máquinas virtuais Java (JVMs) que faz com que os programas desenvolvidos nessa linguagem possam ser executados em qualquer sistema que possua uma máquina virtual.

A linguagem Java ainda é proprietária e é controlada pela *Java Community Process*. Desde seu lançamento em 1995, a linguagem passou por diversas modificações e atualizações, sendo que sua versão atual foi liberada em 2006, já tendo uma nova publicação prevista para 2008.

Possui várias características da sintaxe do C e C++, mas apresenta um modelo de objeto mais simples e algumas outras facilidades. Java foi projetada tendo em vista os seguintes objetivos:

- Deveria utilizar a metodologia de programação orientada a objetos.
- Deveria permitir que o mesmo programa pudesse ser executado em diferentes sistemas operacionais (portabilidade).
- Deveria fornecer recursos e suporte para redes de computadores.
- Deveria ser projetada para executar código de fontes remotas de forma segura.
- Deveria ser fácil de usar, e selecionar o que fosse considerado bom das outras linguagens de programação.

Além dos aspectos acima citados, a linguagem possui outras vantagens, tais como a facilidade de internacionalização, uma vez que ela suporta nativamente caracteres Unicode; possui um vasto conjunto de bibliotecas e APIs (*Application Programming Interface* - Interface de Programação de Aplicativos) facilitando a programação; facilidades para criação de programas distribuídos e multitarefa; desalocação de memória automática por processo de coletor de lixo (*garbage collector*) que previne os vazamentos de memória (*memory leaks*), dentre outras.

Uma API bastante útil que foi utilizada neste projeto é a *Java Communication* (também conhecida como JavaComm ou javax.comm), uma extensão do Java que permite desenvolver aplicações de comunicação serial (permitindo o acesso às portas seriais do computador) para tecnologias tais como *Smart Cards*, sistemas embarcados, robótica, etc. Infelizmente esta API não faz parte da versão Java 2 padrão, tendo que ser baixada separadamente.

As principais críticas quanto a linguagem giram em torno do seu desempenho, dita ser significativamente mais lenta e consumir mais memória do que as linguagens compiladas nativamente como o C ou C++. Entretanto, a velocidade de execução de programas Java tem melhorado com a introdução da compilação *Just-In Time* (JIT), que faz com que a máquina virtual analise o código e realize otimizações em certas partes mais críticas, e de otimizações na Máquina Virtual Java feitas ao longo do tempo. As primeiras implementações da linguagem eram interpretadas pela máquina virtual para alcançar os requisitos de portabilidade. Essas implementações produziam programas que rodavam mais lentos que os compilados para executáveis nativos, uma vez que o processo de interpretação dos *bytecodes* pela JVM é um processo muito custoso, o que fez com que Java sofresse com a fama de ter um pobre desempenho. Já as implementações mais recentes da JVM produzem programas significativamente mais rápidos que os anteriores.

Há ainda outros aspectos considerados negativos, mas mesmo assim, a plataforma Java foi adotada mais rapidamente do que qualquer outra linguagem de programação na história da computação. Em 2003 Java atingiu a marca de 4 milhões de desenvolvedores em todo mundo [5]. Java continuou crescendo e hoje é uma referência no mercado de desenvolvimento de software. Ela tornou-se popular pelo seu uso na Internet e hoje possui seu ambiente de execução presente em *web browsers*, *mainframes*, SOs, celulares, *palmtops*, cartões inteligentes, entre outros.

3. MÉTODOS E IMPLEMENTAÇÕES

3.1. PROJETO

O projeto abordado neste documento foi elaborado como Trabalho de Conclusão de Curso e tem por objetivo final desenvolver um sistema de ultra-som Doppler pulsado para permitir a avaliação do acesso vascular de maneira mais acessível e prática, através da análise da velocidade de deslocamento do fluxo. Um esquema geral do projeto pode ser conferido na Figura 21.

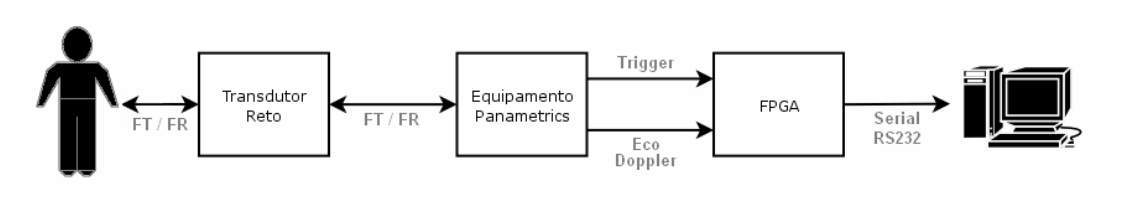

Figura 21 - Esquema geral do projeto

O transdutor de face plana (reto) de freqüência 2.25 MHz foi conectado ao equipamento da *Panametrics*, o qual acionava o transdutor para que este emitisse o sinal de ultra-som. O próprio transdutor captava o sinal refletido pelo acesso vascular e o enviava de volta para o equipamento da *Panametrics*. Este então enviava um sinal de disparo (*trigger*) e o sinal refletido (Eco Doppler) para a FPGA. Para analisar esses sinais, um osciloscópio foi conectado ao equipamento da *Panametrics*. A FPGA, ao receber o sinal de *trigger*, esperava um tempo (de 30 a 40 μs), a fim de pegar a parte de interesse do sinal refletido (onde ocorre a variação do sinal devido ao deslocamento) e então se iniciava a análise do mesmo. Tal análise consistia na contagem do número de pulsos de *clock* contidos nos três pulsos principais do sinal refletido. Esta informação de contagem era então utilizada para obter uma estimativa da freqüência do sinal refletido e exibir seus quatro dígitos principais no *display* de sete segmentos da FPGA. Esta parte descrita anteriormente é referente ao projeto do Alessandro. Neste projeto dever-se-ia, então, enviar essa contagem do número de pulsos, via comunicação serial para o PC, que através de um programa feito em Java, a receberia e calcularia a freqüência do sinal refletido de uma forma mais precisa. A freqüência refletida obtida era utilizada para calcular a velocidade do fluxo, através da Equação (2.4). Por fim, a velocidade obtida era exibida na tela do computador para que pudessem ser feitas as devidas análises.

3.2. EQUIPE

A parte do projeto desenvolvido na FPGA possui os seguintes blocos funcionais: Inicializador, Contador, Display e Serial, cuja ligação entre eles pode ser conferida na Figura 22.

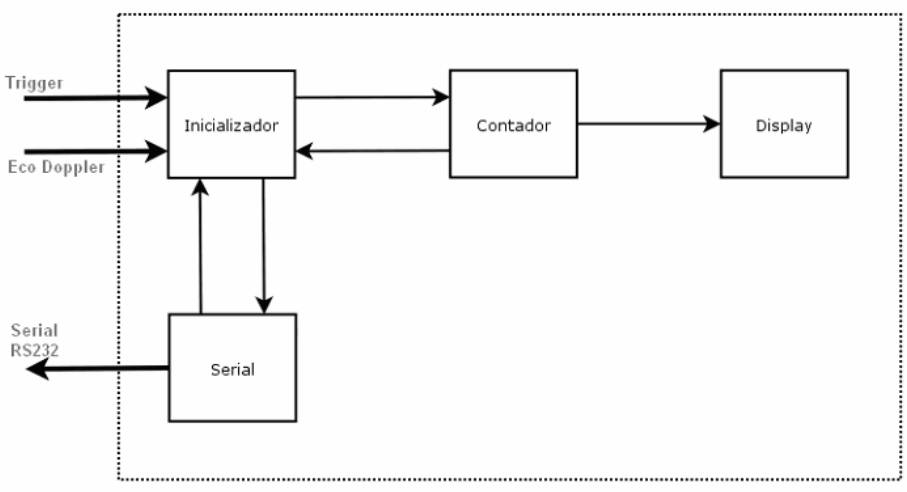

Figura 22 - Blocos funcionais do projeto desenvolvido na FPGA

A parte do projeto desenvolvida pelo Alessandro compreende aos módulos Inicializador, Contador e do Display, enquanto que o módulo da Serial (que tem como função transmitir a contagem de pulsos recebida do Inicializador para o PC via comunicação serial) e o programa desenvolvido em Java (responsável por ler os valores recebidos na porta serial e fazer os devidos cálculos com os mesmo, ou seja, obter a freqüência do sinal refletido, calcular o desvio de freqüência correspondente ao desvio Doppler e calcular a velocidade de deslocamento do fluxo sangüíneo), foram desenvolvidos por mim e estão sendo abordados nesta monografia.

3.3. DESCRIÇÃO DAS ATIVIDADES REALIZADAS

O primeiro passo foi desenvolver o bloco da Serial, ou seja, escrever o código VHDL responsável por receber o valor da contagem dos pulsos de *clock* presentes em três pulsos do sinal refletido calculado pelo bloco Contador e enviado para a Serial pelo bloco Inicializador e então enviar esta contagem para o PC via comunicação serial.

A conexão entre os módulos Serial e Inicializador pode ser conferida na Figura 23.

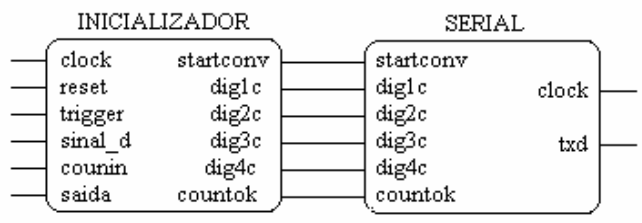

Figura 23 - Conexão entre os Módulos Inicializador e Serial

A comunicação entre os dois blocos é feita da seguinte forma: o Inicializador envia os dígitos referentes ao valor da contagem separadamente, isto é, unidade (*dig1c*), dezena (*dig2c*), centena (*dig3c*) e milhar (*dig4c*), cada um contendo um byte (oito bits). Além dos dígitos, é enviado também um sinal responsável pela inicialização da Serial (*startconv*), informando que os valores já podem ser transmitidos. Após o término da transmissão, o bloco Serial envia um sinal para o Inicilizador (*countok*) informando que os valores já foram transmitidos (término da transmissão).

Para realizar a transmissão, foi implementado o protocolo RS-232 contendo o *start bit*, oito bits de informação e o *stop bit*, conforme pode ser visto na Figura 24.

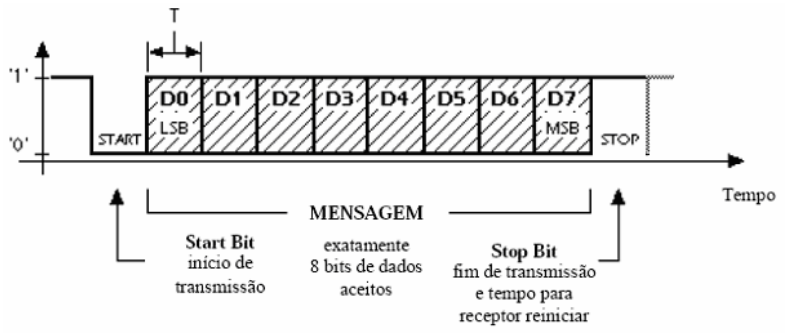

Figura 24 - Formato do pacote utilizado na comunicação serial

3.3.1. MÓDULO SERIAL

Este módulo foi implementado utilizando-se o conceito de Máquinas de Estado Finitas (*Finite State Machines* – FSM). Uma FSM é uma representação do modelo comportamental de um sistema, composta por um número finito de estados, transições entre esses estados e ações. Um estado armazena informações sobre o passado, isto é, reflete as mudanças da entrada desde o início do sistema até o momento atual. Uma transição indica uma mudança de estado e é descrita por uma condição que precisa ser cumprida para que seja habilitada. Uma ação é uma descrição de certa atividade que será realizada em um determinado momento. Máquinas de Estado Finitas é uma forma muito poderosa de representação do comportamento de uma entidade, permitindo a fácil compreensão do que está sendo representado. Além disso, FSM são

relativamente simples de serem implementadas em VHDL. A Figura 25 ilustra a Máquina de Estado Finito do módulo Serial.

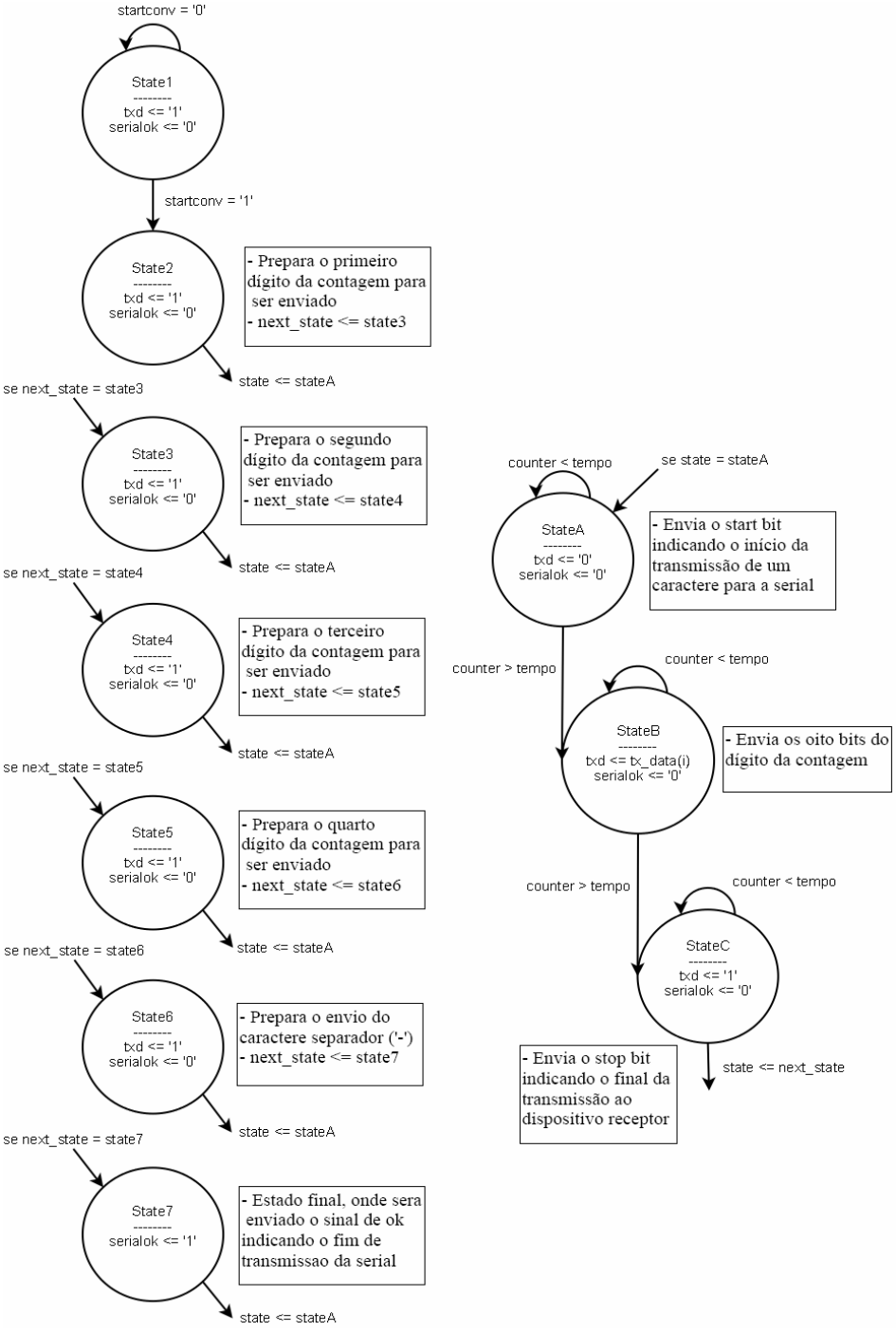

Figura 25 - FSM do Módulo Serial

O Estado 1 da FSM espera pela subida do sinal de inicialização (`startconv`) proveniente do bloco Inicializador. Enquanto isso a linha da comunicação serial é mantida em nível lógico alto (`txd <= 1`), indicando estar ociosa. Ao receber o sinal do módulo Inicializador, passa-se para o Estado 2.

No Estado 2, o primeiro dígito do valor da contagem é preparado para ser enviado para a serial. Após as devidas conversões, o valor do dígito é armazenado em um vetor de oito posições (`tx_data`), guarda-se para qual estado deverá retornar após o envio deste primeiro dígito da contagem (`next_state <= state3`) e vai-se para o Estado A. Neste estado, a transmissão serial é inicializada, ou seja, é enviado o *start bit* (`txd <= 0`) e passa-se para o Estado B. No Estado B são enviados os oito bits de informação, que é o valor contido em `tx_data` (`txd <= tx_data(i)`). Depois de enviado os oito bits do dígito corrente, passa-se para o Estado C que é onde será enviado o *stop bit* (`txd <= 1`), voltando-se a linha para o nível lógico alto. Enviado o *stop bit*, deve-se retornar para o estado anterior ao início da transmissão, que é o valor armazenado no sinal `next_state` (`state <= next_state`).

No Estado 3, o segundo dígito do valor da contagem é preparado para ser enviado para a serial. Após as devidas conversões, o valor do dígito é armazenado em `tx_data`, armazena-se o próximo estado de retorno após a transmissão (`next_state <= state4`) e vai-se para o Estado A, para iniciar a transmissão do segundo dígito.

No Estado 4, o terceiro dígito do valor da contagem é preparado para ser enviado para a serial. Após as devidas conversões, o valor do dígito é armazenado em `tx_data`, armazena-se o próximo estado após a transmissão (`next_state <= state5`) e vai-se para o Estado A, para iniciar a transmissão do terceiro dígito.

No Estado 5, o segundo dígito do valor da contagem é preparado para ser enviado para a serial. Após as devidas conversões, o valor do dígito é armazenado em `tx_data`, armazena-se o próximo estado após a transmissão (`next_state <= state6`) e vai-se para o Estado A, para iniciar a transmissão do quarto dígito.

No Estado 6 é armazenado em `tx_data` o valor correspondente ao traço (‘-’ = "00101101"), que é utilizado como caractere separador entre valores consecutivos das contagens. Então, analogamente aos demais estados, armazena-se para qual estado deverá retornar após a transmissão (`next_state <= state7`) e vai para o Estado A.

Por fim, no Estado 7 envia-se um sinal para o Inicializador indicando o fim da transmissão (`serialok <= 1`) e retorna-se para o Estado 1 recomeçando todo o processo.

É como se fossem duas máquinas de estado conectadas, sendo uma responsável pela preparação dos caracteres para serem enviados pela serial e a outra responsável pelo envio dos caracteres.

3.3.2. LENDO VALORES DA PORTA SERIAL EM JAVA

Depois de desenvolvida a parte responsável pelo envio da contagem de pulsos da FPGA para o computador, o próximo passo seria desenvolver um programa para ler esses dados da porta serial, para que posteriormente estes pudessem ser analisados e utilizados para a realização dos cálculos almejados.

Para coletar os dados da porta serial, optou-se por utilizar a linguagem Java, devido principalmente, as facilidades por esta oferecidas, uma vez que já existe uma API da *Sun*, a *Java Communication*, que lida com a parte mais baixo nível (como, por exemplo, reconhecer o *start bit*, os dados e o *stop bit*, sincronização) da comunicação serial RS-232, facilitando o desenvolvimento pelo programador. Esta API não faz parte da versão Java 2 padrão, tendo que ser "baixada" separadamente.

Para instalar a API, após tê-la "baixado" e descompactado, devem-se seguir os seguintes passos:

- Copiar o arquivo `win32com.dll` para o diretório `C:\Arquivos de Programas\Java\JavaSDK\bin` (isto é, o diretório onde o J2SDK foi instalado no PC).
- Copiar o arquivo `comm.jar` para o diretório `C:\Arquivos de Programas\Java\JavaSDK\lib`.
- Copiar o arquivo `javax.comm.properties` para o diretório `C:\Arquivos de Programas\Java\JavaSDK\lib`.
- Em seguida configure o CLASSPATH para que ele reconheça o arquivo `comm.jar`.

Antes de iniciar a comunicação com a porta serial, é necessário reconhecer as portas existentes na estação de trabalho. A API de comunicação possui o método `getPortIdentifiers()` integrante da classe `CommPortIdentifier` que retorna em uma estrutura `Enumeration`, as portas disponíveis. A classe `CommPortIdentifier` pode ser instanciada para representar uma porta. Para isso, precisa-se varrer a estrutura retornada pelo método e instanciar cada porta através de uma conversão (*casting*) simples:

```
Enumeration listadePortas;
listadePortas = CommPortIdentifier.getPortIdentifiers();
```

```
String[] portas;
int i = 0;
portas = new String[10];
    while (listadePortas.hasMoreElements()) {
        CommPortIdentifier ips =
        (CommPortIdentifier)listadePortas.nextElement();
        portas[i] = ips.getName();
        i++;
}
```

O método `hasMoreElements()` retorna o próximo elemento da estrutura `listaDePortas`, mas o *loop* `while` garante que todos os elementos sejam passados ao *Array* `portas` através do método `getName()`.

Depois de reconhecidas e armazenadas as portas, deve-se abri-las para que se possa realizar a comunicação. O método `getPortIdentifier(String porta)` da classe `CommPortIdentifier` retorna um identificador da porta escolhida, sendo necessário instanciar um objeto para receber esse identificador:

```
CommPortIdentifier cp = CommPortIdentifier.getPortIdentifier(“COM2”);
```

Em seguida deve-se criar uma instância da classe `SerialPort` utilizando o identificador obtido no passo anterior. Para isso, uma conversão deverá ser feita. A porta só pode ser instanciada através de um “*casting*”. Ao mesmo tempo, a porta já pode ser aberta para comunicação:

```
SerialPort porta = (SerialPort)cp.open("Comunicacao",timeout);
```

O método `open()` tem como parâmetros o nome da aplicação (`String`) que está realizando a chamada (normalmente é utilizado o próprio nome da classe para evitar conflitos), que se tornará o dono da porta, e o valor desejado para o *timeout*, isto é, o tempo em milisegundos para bloquear a espera pela abertura da porta.

Neste ponto podem-se configurar os parâmetros da comunicação serial. Isto é feito utilizando o método `setSerialPortParams`:

```
porta.setSerialPortParams(baudrate, porta.DATABITS_8, porta.STOPBITS_1,
   porta.PARITY_NONE);
```

Todos os parâmetros passados acima são do tipo `int`. Através do comando acima é definida a taxa de transmissão em que será realizada a comunicação (`baudrate`), que serão utilizados oito bits de dados, um *stop bit* e nenhuma paridade.

Em seguida, é necessário atribuir fluxos de entrada e saída para a porta. Há várias formas diferentes para realizar tal operação. Uma maneira de se fazer é utilizar as classes abstratas `InputStream` e `OutputStream`, já que a classe `SerialPort` implementa os métodos de entrada e saída dessas classes para comunicação serial. No projeto foi utilizado um `BufferedReader` para receber o fluxo de entrada, ou seja, para ler os dados da porta serial, conforme pode ser visto abaixo:

```
BufferedReader is = new BufferedReader(new
                    InputStreamReader(porta.getInputStream() ) );
```

Não se definiu um fluxo de saída, uma vez que no projeto não se escreve na porta serial.

Os passos para se ler dados da porta serial são:

- Criar um fluxo de entrada.
- Adicionar um gerenciador de eventos para dados na porta serial.
- Instanciar uma *Thread* para aguardar os eventos.
- Tratar o evento e receber os dados.

O primeiro passo já foi detalhado. Para adicionar um gerenciador de eventos para a porta serial basta fazer:

```
porta.addEventListener(this);
```

Este gerenciador faz com que cada vez que ocorrer um evento na porta serial, o método `serialEvent(SerialPortEvent ev)` da classe `SerialPortEventListener` seja chamado. É neste método que o evento deverá ser tratado. A próxima etapa é notificar o objeto `porta` de que podem existir dados para serem lidos:

```
porta.notifyOnDataAvailable(true);
```

Por fim, deve-se tratar o evento, ou seja, ler os dados da serial e armazená-los de alguma forma.

```
public void serialEvent(SerialPortEvent ev) {
    switch (ev.getEventType()) {
        case SerialPortEvent.DATA_AVAILABLE:
            try {
                int c;
                BufferedOutputStream out = new BufferedOutputStream(new
                    FileOutputStream("out.txt"));
                int iteracoes = 0;
                while((iteracoes < 101) && ((c = is.read()) != -1)) {
                    out.write(c);
```

```
                    out.flush();
                    System.out.print((char)c);
                    iteracoes++;
                }
                System.exit(0);
            } catch (Exception e) {
                System.out.println("Erro durante a leitura: " + e );
            }
            break;
    }
```

Foi utilizado um `BufferedOutputStream` para salvar os dados lidos em um arquivo de nome “out.txt”. O comando `c = is.read()` lê o primeiro caractere da informação presente na porta serial e o armazena em `c`. O comando `out.write(c)` escreve o caractere lido no arquivo. Além disso, foi utilizado também um inteiro `iteracoes`, que representa o número de caracteres a serem lidos. Este valor poderá ser informado pelo usuário. Assim a condição: `while((iteracoes < 101) && ((c = is.read()) != -1))` significa, enquanto não leu o número de caracteres desejado e houver caracteres para serem lidos.

3.3.3. CÁLCULOS EM JAVA

Após salvar os valores da contagem de pulsos enviados pelo módulo serial e, lidos através da porta serial do computador, em um arquivo através do programa em Java acima descrito, estes valores devem ser lidos do arquivo para que se possa realizar os cálculos desejados, isto é, calcular o valor da freqüência do sinal refletido e, posteriormente utilizá-lo na obtenção do valor da velocidade do fluxo.

Primeiramente deve-se abrir o arquivo para que este possa ser lido. Isto pode ser feito da seguinte maneira:

```
BufferedReader in = new BufferedReader(new FileReader(“out.txt”));
```

Após aberto, os valores das contagens de pulsos de *clock* podem ser lidos do arquivo. Seus valores foram então armazenados em uma matriz `pulsos`. As contagens foram armazenadas no arquivo seqüencialmente, contendo apenas um traço (‘-’) para separá-las, da seguinte forma: 0060-0061-0059-... Assim, cada linha da matriz `pulsos` conterá um valor da contagem, e quatro colunas, correspondendo aos quatro dígitos. Durante o processo, o valor do traço é descartado, pois este não terá mais utilidade, servindo apenas para separação durante a leitura e armazenagem dos valores.

```
int caractere;
char c;
try {
    while (((caractere = in.read()) != -1) && ((char)caractere != '-'));
    for (int j = 0; j < 19; j++) {
        for (int i = 0; i < 4; i++) {
            c = (char)in.read();
            caractere = (int)c;
            pulsos[j][i] = caractere;
        }
        lido = in.read(); // descarta o -
    }
} catch (Exception e) {
    System.out.println("Erro ao tratar o arquivo!");
}
```

Inicialmente há um *loop* que descarta os primeiros caracteres até encontrar um traço (‘-’), desconsiderando o primeiro valor da contagem lido do arquivo. Isto é feito, pois, devido a problemas de sincronização, este primeiro valor pode não fazer sentido, sendo inválido. Então há dois laços seguidos cujo objetivo é percorrer a matriz `pulsos` armazenando os valores nas suas devidas posições (linhas e colunas).

Um aspecto a ser observado é que no arquivo foram armazenados caracteres (`char`), entretanto, o método `read()` retorna o valor do inteiro correspondente ao caractere lido e não seu valor correto, sendo necessário uma conversão, passando o valor lido para o tipo `char` e posteriormente pegando seu valor inteiro (`int`) correto. Esta conversão é feita pelas linhas:

```
c = (char)in.read();
caractere = (int)c;
```

Depois de armazenados na matriz, os valores precisam ser juntados, ou seja, os dígitos (unidade, dezena, centena e milhar) devem ser compostos a fim de formar o número correto. Isto é feito simplesmente multiplicando-se o dígito correspondente ao milhar por mil, o da centena por cem, o da dezena por dez e somando-se os valores obtidos juntamente com o valor da unidade. Obtidos os números, é então calculada a média de pulsos contados, isto é, somam-se todos os valores da contagem e divide pelo número de valores armazenados, o qual depende do número de iterações realizadas. Obtido a média de pulsos, a freqüência do sinal refletido pode ser obtida através da Equação (3.1) abaixo:

$$f_r = \frac{1}{\left(\frac{media}{3}\right) \times 20ns} \qquad (3.1)$$

Na equação acima, a média de pulsos é dividida por 3, pois são medidas as quantidades de pulsos de *clock* contida em três pulsos do sinal de ultra-som refletido, a fim de aumentar a precisão. Além disso, este termo é multiplicado por 20ns que corresponde ao período do sinal de *clock* da FPGA (50 MHz).

Por fim, depois de calculada a freqüência refletida (f_r), pode-se obter a velocidade de fluxo da partícula que interage com a onda de ultra-som incidente a partir da Equação (3.2).

$$v = \left(\frac{c}{2 f_t \cos\theta}\right)\left(f_r - f_t\right) \qquad (3.2)$$

Onde:

c = velocidade do som no meio.

f_t = freqüência transmitida.

θ = ângulo entre a direção de propagação do ultra-som e a direção de movimento da partícula (ângulo Doppler).

f_r = freqüência refletida.

A velocidade (c) do som nos tecidos do corpo humano é de 1540 m/s, a freqüência transmitida (f_t) e o ângulo Doppler (θ) podem ser informados pelo usuário.

4. RESULTADOS

Inicialmente, foi simulado o comportamento do módulo Serial feito em VHDL no kit de desenvolvimento. Os testes foram feitos, primeiramente, utilizando o simulador ModelSim XE III 6.2c, que acompanha o kit de Desenvolvimento da Xilinx. A partir desse simulador, foi possível verificar o comportamento funcional do módulo, analisando se os sinais estavam sendo corretamente processados e estavam de acordo com o propósito que lhe cabia. Depois de passada a fase de simulação do módulo isoladamente, optou-se por testá-lo juntamente com os demais módulos (Inicializador, Contador e Display) desenvolvidos pelo Alessandro. A Figura 26 ilustra o resultado completo da simulação.

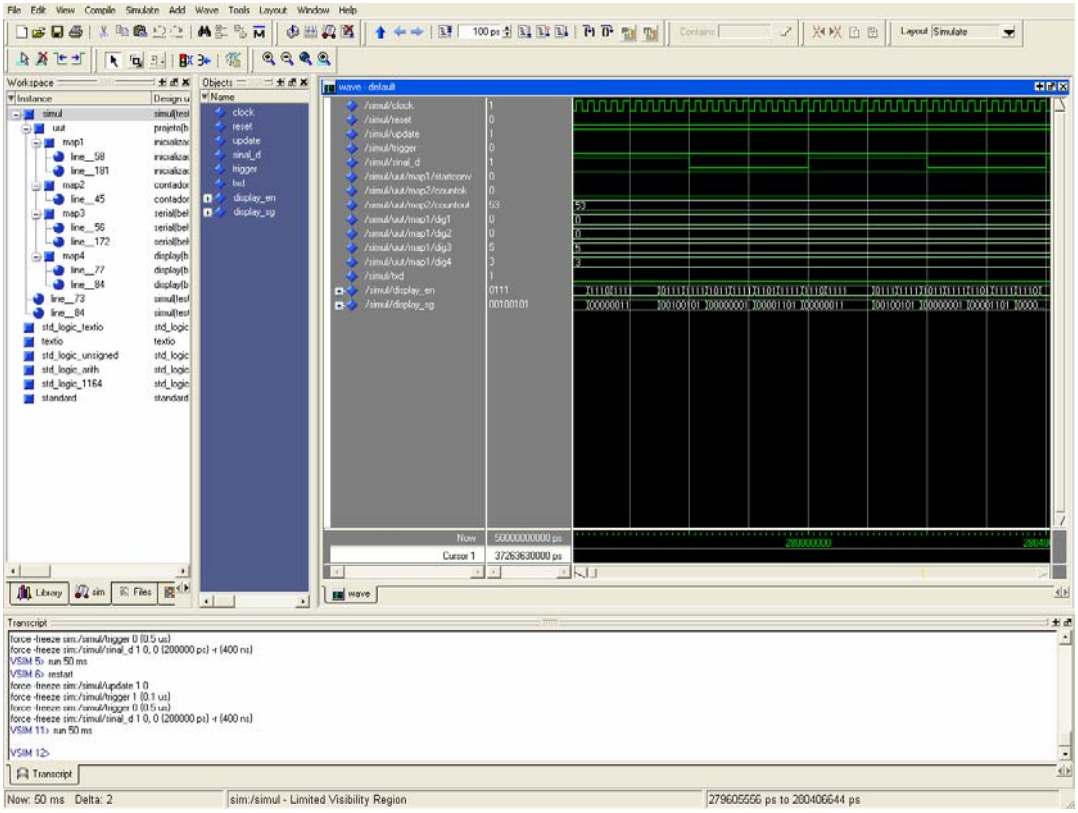

Figura 26 - Resultado da simulação dos módulos desenvolvidos na FPGA

Terminada a fase de simulação, pôde-se concluir que na teoria tudo estava funcionando corretamente. O próximo passo agora era partir para a prática, ou seja, gravar o projeto na FPGA e testá-lo experimentalmente, uma vez que nem sempre o funcionamento correto na simulação implica na mesma situação para o caso real. A fim de efetuar a fase inicial de testes experimentais, realizou-se a seguinte montagem, exibida na Figura 27:

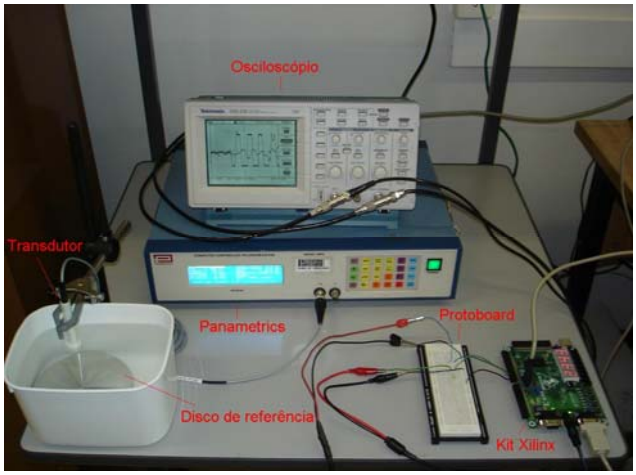

Figura 27 - Montagem realizada para os primeiros testes experimentais

O transdutor conectado ao equipamento da Panametrics teve sua ponta colocada sob a água para que o ultra-som pudesse se propagar. Este emitia e captava os sinais ultra-sônicos refletidos por uma placa de metal (disco de referência). O sinal refletido era então transmitido para o equipamento da Panametrics que o repassava para a FPGA juntamente com o *trigger* responsável por disparar todo o processo. O osciloscópio foi utilizado para a visualização do sinal refletido.

A FPGA foi conectada à porta serial do PC. Inicialmente, como a parte em Java responsável pela captura dos dados na porta serial do PC ainda não havia sido implementada, foi utilizado o programa *HyperTerminal* para capturar os dados enviados pela serial. A tela do programa pode ser vista na Figura 28. Para executar o programa era necessário, primeiramente informar os parâmetros a serem utilizados durante a conexão. Os parâmetros utilizados foram:

- Porta: COM2
- Baudrate: 2400 bps
- N.º de bits contidos na informação: 8:
- N.º de *stop bits*: 1
- Paridade: Nenhuma

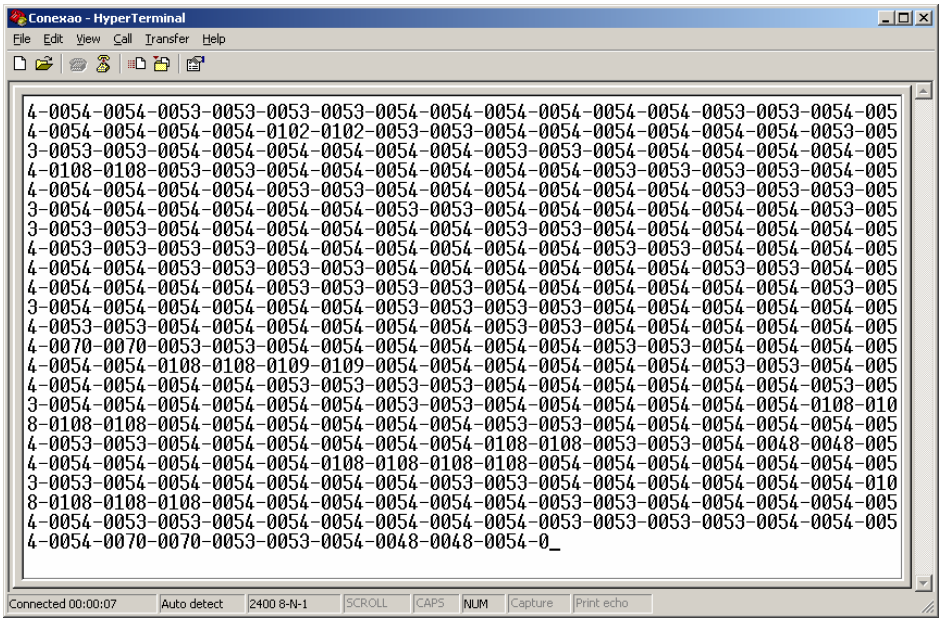

Figura 28 - Tela do programa HyperTerminal - Captura dos valores da contagem enviados pela serial

A partir da figura acima foi possível verificar que o módulo Serial estava funcionando, isto é, os valores da contagem estavam sendo enviados para a porta serial do computador.

O próximo passo então era desenvolver o programa em Java responsável por ler estes valores da porta serial do computador. Como os dados estavam realmente sendo enviados, utilizou-se a mesma

montagem para testar o programa Java. A Figura 29 mostra a tela do programa Netbeans utilizado para desenvolver o programa em Java, na qual pode ser conferido o recebimento dos dados.

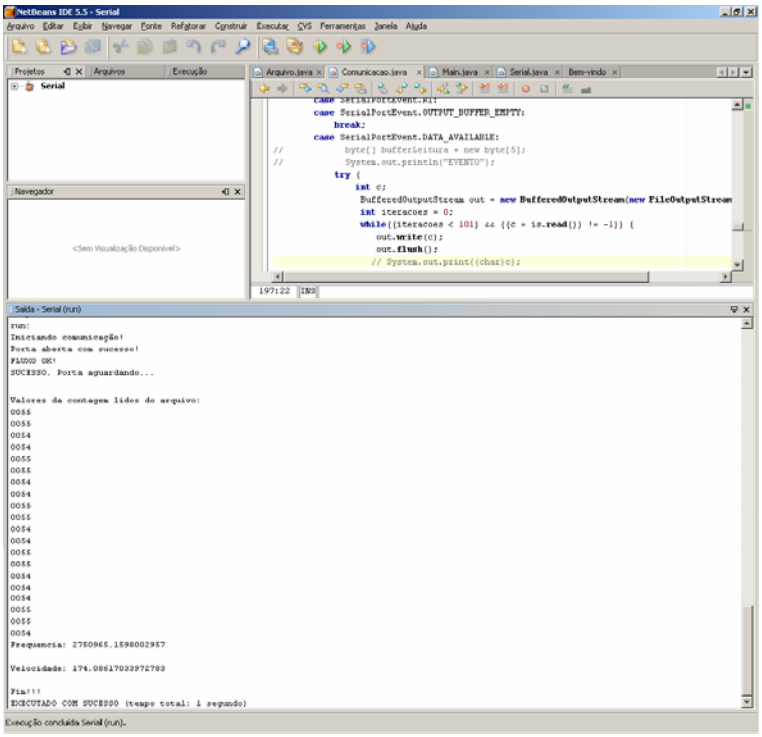

Figura 29 - Tela do programa Netbeans

Testada a parte referente à captura dos dados, criou-se uma interface para que o usuário pudesse interagir com a aplicação, a qual pode ser vista na Figura 30.

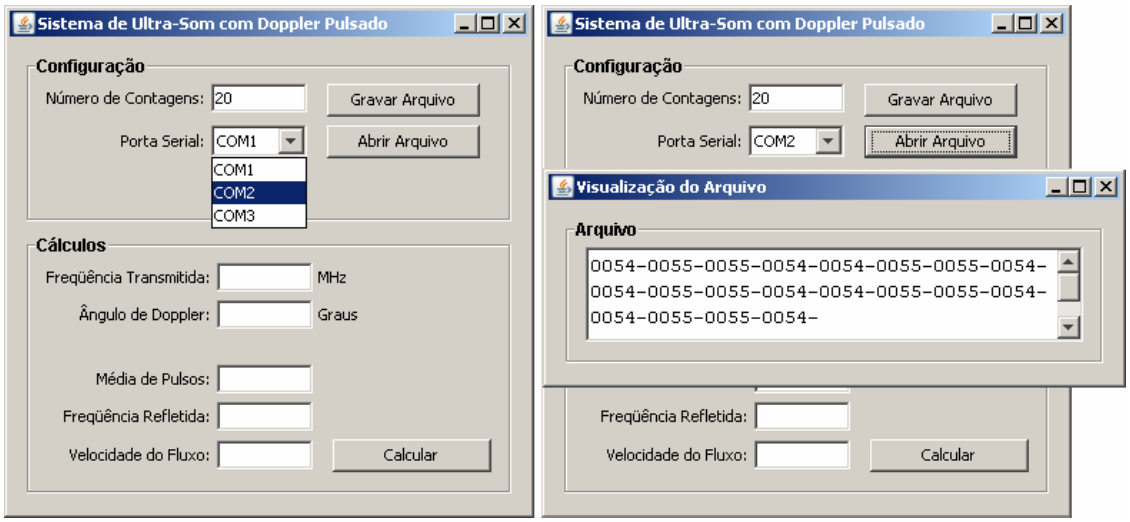

Figura 30 - Interface

Por meio da interface construída, o usuário pode informar o número de valores da contagem do número de pulsos que deveria ser utilizada para obtenção da média da contagem e da freqüência do sinal refletido, selecionar qual porta serial que ele deseja utilizar, visualizar o conteúdo do arquivo gravado contendo os valores das contagens recebidos. Então, os valores da freqüência utilizada na transmissão e o valor do ângulo de Doppler poderiam ser informados ao programa que os utilizaria para o cálculo da freqüência do sinal refletido e da velocidade do fluxo, exibindo os valores obtidos para o usuário.

Como último experimento, era necessário realizar alguns testes agora com a água em movimento. Para isso, utilizou-se uma bomba peristáltica, a qual pode ser vista na Figura 31.

Figura 31 - Bomba Peristáltica

Para conseguir o espalhamento do sinal de ultra-som, foram colocadas micro-partículas de vidro na água. A montagem final pode ser conferida na Figura 32.

Figura 32 - Montagem realizada para testes experimentais finais

4.1. DIFICULDADES E LIMITAÇÕES

A primeira dificuldade enfrentada durante o desenvolvimento do projeto, foi o grande tempo demandado para a compreensão do mesmo e para o estudo dos vários conceitos envolvidos, sendo que muitos desses conceitos não haviam sido abordados durante o decorrer do curso (principalmente no que diz respeito a ultra-som).

Posteriormente, outra complicação enfrentada foi quanto a definição e especificação do projeto. Houve certas dúvidas a respeito do que deveria ser feito realmente que resultaram em algumas mudanças na especificação no princípio do trabalho.

Depois de finalmente definido o que deveria ser realizado, as maiores dificuldades encontradas foram com relação a VHDL e o que era permitido pela FPGA e o que não era. Por exemplo, quando se tentava fazer um *loop* utilizando o comando `while` que iterasse mais que 64 vezes, uma mensagem de erro era obtida. Além disso, foi possível perceber as limitações quanto a efetuação de certas operações matemáticas, principalmente com relação à divisão, a qual só era realizada por múltiplos de 2, fato este que conduziu a idéia de se realizar as contas mais complexas em uma linguagem de alto nível (Java no caso). Além do mais, o laboratório onde o projeto estava sendo desenvolvido, ou mais especificamente, o computador sendo utilizado para o projeto e o kit de desenvolvimento da Xilinx estava sendo compartilhado com um aluno de um outro projeto, o que atrasou um pouco o andamento, principalmente na parte inicial onde havia uma dependência maior do kit, além do fato de a compilação do código VHDL para ser gravado na FPGA ser bastante lento.

Superadas essas dificuldades quanto a VHDL e ao kit, outro aspecto que causou certa complicação foi alguma instabilidade da API de comunicação do Java, às vezes funcionando corretamente, às vezes lançando alguma exceção, além de certa falta de sincronismo que pode ser notada com alguma freqüência, o que dificultou o desenvolvimento da parte implementada em Java.

Algo que também se percebeu foi que, apesar de parecer um tempo bastante longo, quatro meses é um prazo curto para o desenvolvimento de um projeto de engenharia.

5. CONCLUSÃO

O desenvolvimento do projeto das interfaces para o Sistema Embarcado de Ultra-Som com Doppler Pulsado como Projeto de Formatura apresentado à EESC-USP como parte dos requisitos necessários para a obtenção da conclusão do curso de Engenharia de Computação foi muito interessante e proveitoso. Este trabalho envolveu muitos conceitos, alguns dos quais já haviam sido estudados durante do decorrer do curso, como FPGA, VHDL, comunicação serial e Java, mas muitos dos temas envolvidos não haviam ainda sido explorados durante a graduação, tais como os princípios de hemodiálise, avaliação de acessos vasculares, ultra-som, de maneira geral, aqueles conceitos mais relacionados à área de bioengenharia. Assim, através deste projeto foi possível rever alguns assuntos e aprender e assimilar muita informação nova. Além de rever assuntos já abordados, o fato de estes saírem da sala de aula e adentrarem um laboratório para serem aplicados na prática permite uma melhor compreensão e fixação das teorias abordadas em aula, possibilitando entender quando, onde e como estas são aplicadas.

Outro aspecto compensador durante o desenrolar do projeto foi o fato de este ter sido desenvolvido em parceria com outro estudante de graduação do curso de Engenharia de Computação, Alessandro Tadashi Miyasaka. Apesar da divisão, houve uma interação muito boa e produtiva entre ambas as partes, não só pelo fato de cada membro estar ciente do andamento do projeto como um todo, mas também para que a integração final das duas partes pudesse funcionar adequadamente, experiência esta que possibilitou o aprimoramento das habilidades referentes ao trabalho em grupo.

Futuramente, o que poderia ser feito com relação ao trabalho seria aprimorar a parte referente ao cálculo da velocidade realizando mais testes experimentais a fim de validar ainda mais o correto funcionamento do sistema desenvolvido.

6. BIBLIOGRAFIA

[1] NASCIMENTO, Marcelo M. & RIELLA, Miguel C. – Avaliação de acesso vascular em hemodiálise: um estudo multicêntrico. Artigo recebido em 15 de maio de 1998 e aceito para publicação em 16 de novembro de 1999. J. Bras. Nefrol. 1999; 21(1): 22-29.

[2] MOLINA, Paulo S. C. – Contribuição para a avaliação de acessos vasculares em pacientes de hemodiálise com ultra-som doppler de ondas contínuas. Tese submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Doutor em Engenharia Elétrica. Florianópolis – Novembro de 2004.

[3] Xilinx: The Programmable Logic Company – website – http://www.xilinx.com/

[4] ABC da Saúde – website – http://www.abcdasaude.com.br/

[5] Wikipédia, A enciclopédia livre – website – http://pt.wikipedia.org/wiki/

[6] FORP/USP – website – http://www.forp.usp.br/restauradora/us01.htm

[7] ANDREUCCI, Ricardo – Ensaio por Ultra-Som. Aplicação Industrial – Ed. Setembro 2006 – Apoio Abende.

[8] DOPPLER ULTRA-SÔNICO TRANSCRANIANO: ASPECTOS FÍSICOS E TECNOLÓGICOS – website – http://www.spmedica.com.br/tcd_fisica/doppler_uti/doppler.htm

[9] Wikipedia, the free encyclopedia – website – http://en.wikipedia.org/wiki

[10] WELLS, P. N. T. – Biomedical Ultrasonics. Academic Press Limited, 1977.

[11] Radiological Society of North America – website – http://www.rsna.org/index.cfm ; http://ej.rsna.org/ej3/0079-98.fin/doppler.htm

[12] Obstetric ultrasound - a comprehensive guide to ultrasound scans in pregnancy – website – http://www.ob-ultrasound.net/

[13] CANZIAN, Edmur – Minicurso Comunicaçao Serial – RS232 – CNZ Engenharia e Informática Ltda.