

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática
Curso de Engenharia de Produção

Desenvolvimento de um Protótipo de um Oxímetro de Pulso

Fernando Vinícius Gonçalves Magro

TCC-EP-32-2008

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática
Curso de Engenharia de Produção

Desenvolvimento de um Protótipo de um Oxímetro de Pulso

Fernando Vinícius Gonçalves Magro

TCC-EP-32-2008

Trabalho de Conclusão de Curso apresentada como requisito de avaliação no curso de graduação em Engenharia de Produção na Universidade Estadual de Maringá – UEM.

Orientador(a): Prof. Dr. Donizete Carlos Bruzarsco

**Maringá - Paraná
2008**

Fernando Vinícius Gonçalves Magro

Desenvolvimento de um Protótipo de um Oxímetro de Pulso

Este exemplar corresponde à redação final do Trabalho de Conclusão de Curso aprovado como requisito parcial para obtenção do grau de Bacharel em Engenharia de Produção da Universidade Estadual de Maringá, pela comissão formada pelos professores:

Orientador(a): Prof^(a). Prof. Dr. Donizete Carlos Bruzarosco
Departamento de Informática, CTC

Prof^(a). Márcia Marcondes Altimari Samed
Departamento de Informática, CTC

Maringá, outubro de 2008

RESUMO

Este trabalho apresenta uma implementação de um protótipo oxímetro de pulso, equipamento que faz a medição da saturação de oxigênio no sangue arterial. Descreve também um sistema para *desktop*, capaz de processar as informações do equipamento num microcomputador pessoal. O trabalho mostra os conceitos envolvidos, o desenvolvimento do *hardware* e do *software*, incluindo o diagrama de blocos e esquema elétrico dos circuitos eletrônicos. Também é abordado o desenvolvimento do *software* segundo os métodos propostos pela Engenharia de Software, utilizando-se os diagramas da UML.

Palavras-chave: Oximetria, Equipamentos de Monitorização, UM,. Engenharia de Software, Hardware.

SUMÁRIO

LISTA DE ILUSTRAÇÕES.....	VI
LISTA DE QUADROS.....	VIII
LISTA DE ABREVIATURAS E SIGLAS	IX
LISTA DE SÍMBOLOS	X
1 INTRODUÇÃO	1
1.1 JUSTIFICATIVA	2
1.2 DEFINIÇÃO E DELIMITAÇÃO DO PROBLEMA	2
1.3 OBJETIVOS	3
1.3.1 <i>Objetivo geral</i>	3
1.3.2 <i>Objetivos específicos</i>	3
1.4 METODOLOGIA.....	3
1.4.1 <i>Revisão de Literatura</i>	4
1.4.2 <i>Projeto de hardware</i>	4
1.4.3 <i>Projeto de software</i>	4
1.4.4 <i>Construção do protótipo</i>	5
1.4.5 <i>Validação</i>	5
2 REVISÃO DA LITERATURA.....	6
2.1 OXÍMETRO DE PULSO	6
2.2 CONCEITOS DE <i>HARDWARE</i>	7
2.3 CONCEITOS DE <i>SOFTWARE</i>	9
2.3.1 <i>UML</i>	10
2.3.2 <i>Processo Unificado</i>	12
2.3.3 <i>Sistema Embarcado</i>	13
3 DESENVOLVIMENTO.....	15
3.1 PROJETO DE <i>HARDWARE</i>	15
3.1.1 <i>Fonte de Alimentação</i>	16
3.1.2 <i>Placa Mãe</i>	21
3.1.3 <i>Probe (sensor)</i>	23
3.1.4 <i>Módulo de Oximetria</i>	24
3.1.5 <i>Display gráfico</i>	26
3.1.6 <i>Teclado</i>	27
3.2 PROJETO DE <i>SOFTWARE</i>	28
3.2.1 <i>Módulo do Oxímetro</i>	28
3.2.2 <i>Módulo para Desktop</i>	30
4 CONCLUSÃO	38
5 REFERÊNCIAS	40
ANEXO A – DOCUMENTAÇÃO DO MÓDULO DE OXIMETRIA SPOX-410.....	42
ANEXO B – TELAS DO <i>SOFTWARE</i> PARA DESKTOP	43
ANEXO C – CÓDIGO FONTE DO SISTEMA EMBARCADO	46
GLOSSÁRIO	54

LISTA DE ILUSTRAÇÕES

FIGURA 1 - DIAGRAMAS DA UML	11
FIGURA 2 – <i>WORKFLOWS</i> DO PROCESSO UNIFICADO	13
FIGURA 3 - DIAGRAMA DE BLOCOS DO PROTÓTIPO	15
FIGURA 4 - DIAGRAMA DE BLOCOS DA FONTE DE ALIMENTAÇÃO	16
FIGURA 5 - ESQUEMA ELÉTRICO COMPLETO DA FONTE DE ALIMENTAÇÃO	17
FIGURA 6 – CIRCUITO RETIFICADOR	17
FIGURA 7 – ETAPAS DO CIRCUITO RETIFICADOR.....	18
FIGURA 8 - CIRCUITO CARREGADOR DE BATERIAS	19
FIGURA 9 - CIRCUITO REGULADOR DE TENSÃO.....	20
FIGURA 10 - ESQUEMA ELÉTRICO DA PLACA MÃE.....	21
FIGURA 11 – DIVISOR DE TENSÃO E AMPLIFICADOR OPERACIONAL DE GANHO UNITÁRIO	22
FIGURA 12 - <i>PROBE</i> , SENSOR DE OXIMETRIA.....	23
FIGURA 13 - ELEMENTOS DO SENSOR	24
FIGURA 14 – MÓDULO DE OXIMETRIA MODELO SPOX-410.....	25
FIGURA 15 – CONECTOR DB9	25
FIGURA 16 – CONEXÃO COM O COMPUTADOR	25
FIGURA 17 – FORMATO DE UMA TRANSMISSÃO ASSÍNCRONA 8N1	26
FIGURA 18 - <i>DISPLAY</i> TOSHIBA T6963C, 240X64 PIXELS.....	26
FIGURA 19 – TECLADO DO PROTÓTIPO	27
FIGURA 20 – ESQUEMA ELÉTRICO DO TECLADO	27
FIGURA 21 – DIAGRAMA DE COMPONENTE	28
FIGURA 22 - DIAGRAMA DE ATIVIDADE DO <i>SOFTWARE</i> EMBARCADO	29
FIGURA 23 - DIAGRAMA DE CASOS DE USO DO <i>SOFTWARE</i> PARA <i>DESKTOP</i>	31
FIGURA 24 – DIAGRAMA DE CLASSES	33
FIGURA 25 – DIAGRAMA DE SEQÜÊNCIA: <i>CADASTRAR PACIENTE</i>	34
FIGURA 26 – DIAGRAMA DE SEQÜÊNCIA: <i>CRIAR EXAME</i>	35
FIGURA 27 – DIAGRAMA DE SEQÜÊNCIA: <i>OBTER LEITURA INSTANTÂNEA</i>	35
FIGURA 28 – DIAGRAMA DE SEQÜÊNCIA: <i>INICIAR MONITORAÇÃO CONTÍNUA</i>	36
FIGURA 29 - DIAGRAMA DE SEQÜÊNCIA: <i>SALVAR ESTADO PACIENTE</i>	36
FIGURA 30 - DIAGRAMA DE SEQÜÊNCIA: <i>EMITIR RELATÓRIO</i>	37

LISTA DE TABELAS

TABELA 1 : CUSTO DOS PRINCIPAIS COMPONENTES UTILIZADOS	38
TABELA 2: COMPARAÇÃO ENTRE O PROTÓTIPO E MODELOS NACIONAIS / IMPORTADOS	39

LISTA DE QUADROS

QUADRO 1 - CONEXÕES ENTRE O MICROCONTROLADOR E O <i>DISPLAY</i>	22
---	----

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ANSI	Instituto Nacional Americano de Padronização
CAD	<i>Computer-Aided Design</i> – Desenho Auxiliado por Computador
CPU	<i>Central Processing Unit</i> – Unidade Central de Processamento
INMETRO	Instituto Nacional de Metrologia, Normalização e Qualidade Industrial
LCD	<i>Liquid Crystal Display</i> – Pannel de Cristal Líquido
LED	<i>Light Emitting Diode</i> – Diodo Emissor de Luz
OEM	<i>Original Equipment Manufacturer</i> – Fabricante de Equipamento Original
PC	<i>Personal Computer</i> – Computador Pessoal
PCI	Placa de Circuito Impresso
RAM	<i>Random Access Memory</i> – Memória de Acesso Randômico
ROM	<i>Read Only Memory</i> – Memória Somente Leitura
UML	<i>Unified Modeling Language</i> – Linguagem de Modelagem Unificada
UTI	Unidade de Terapia Intensiva
VAC	Tensão Alternada
VDC	Tensão Contínua
VPC	Tensão Por Célula
USB	<i>Universal Serial Bus</i>
USART	Transmissor/Receptor Universal Síncrono e Assíncrono

LISTA DE SÍMBOLOS

μ	Micro - Prefixo numérico representando 10^{-6}
k	Quilo - Prefixo numérico representando 10^3
p	Pico - Prefixo numérico representando 10^{-12}
F	Farads – Unidade de Capacitância
Ω	<i>Ohm</i> – Unidade de Resistência
b	<i>bit</i> – Unidade de Informação
V	Volts – Unidade de Tensão Elétrica

1 INTRODUÇÃO

Durante um atendimento médico, os procedimentos a serem efetuados dependem de uma avaliação do estado do paciente; para essa tarefa são empregados, juntamente com o conhecimento médico, vários dispositivos e equipamentos, onde tal avaliação pode conduzir o paciente a uma simples medicação ou em casos mais graves até uma intervenção cirúrgica (WEBSTER, 2006).

A maioria dos equipamentos de monitoramento tem aplicação no pré-operatório, na sala de operações e no pós-operatório. Alguns sistemas têm mais utilidades antes e logo após um procedimento, como é o caso da oximetria de pulso. A oximetria consiste numa medição contínua e indireta da oxigenação dos tecidos através de um método não-invasivo, que se relaciona com o conteúdo de oxigênio no sangue, de enorme importância nas práticas cirúrgicas e anestésicas moderna. O oxímetro tem grande importância na recuperação do impacto anestésico-cirúrgico, onde é comum o desenvolvimento de hipoxemia (baixa concentração de oxigênio no sangue) nesse período pós-operatório imediato (MARCONDES *et al.*, 2006).

Atualmente, no mercado brasileiro de equipamentos eletromédicos, é possível encontrar diversas marcas de oxímetro de pulso, tais como Ohmeda, Emai, Dixtal, dentre outros, que oferecem uma vasta gama de recursos, como indicadores do sinal captado, alarmes sonoros e visuais, curva pletismográfica, baterias de longa duração, enfim, possuem diversas características que os tornam mais versáteis e indispensáveis no ambiente médico (FERNANDES; OJEDA; LUCATELLI, 2001). Entretanto, uma prática que ganha destaque é o uso de microcomputadores para auxiliar o diagnóstico de pacientes, e verificou-se que os equipamentos eletromédicos nacionais não oferecem conectividade aos PCs de forma fácil e rápida, gerando relatórios, provendo gráficos e informações em tempo real e, sobretudo de baixo custo.

Em alguns exames, a monitoração contínua e ininterrupta dos níveis de oximetria é de importância fundamental para o diagnóstico médico, como por exemplo, em pacientes submetidos à polissonografia - um exame que busca identificar problemas de apnéia,

hipopneia e outras doenças relacionadas ao sono. Contudo, a polissonografia realizada num laboratório de sono é dispendiosa, envolve recursos humanos e técnicos consideráveis, que não se encontram facilmente disponíveis (VENTURA; OLIVEIRA; DIAS, 2007). Portanto, identificou-se a necessidade de unir o oxímetro ao microcomputador, através de um *software* que possibilite a monitoração dos sinais através de uma interface mais completa. Evidentemente isso resultaria em inúmeros benefícios: o médico poderia monitorar os níveis de oxigenação do paciente da sua sala, da sua casa ou de qualquer local remoto através da Internet, e até emitir relatórios computadorizados economizando recursos.

Diante do cenário apresentado, propõe-se o desenvolvimento de um protótipo de oxímetro de pulso que fornecerá valores em uma forma numérica através de sinais digitais, os quais podem ser interpretados e tratados por microcontroladores ou computadores, a um custo menor em relação aos oferecidos pelo mercado.

1.1 Justificativa

A implementação do oxímetro de pulso se justifica pela necessidade de desenvolver um *hardware* e *software* destinado à área médica, que possua conectividade aos microcomputadores auxiliando o diagnóstico de pacientes. Diante desse fato, as contribuições serão a redução dos custos, um melhor aproveitamento do potencial das informações e agilidade na análise dessas informações.

1.2 Definição e delimitação do problema

Este trabalho definirá os conceitos relacionados à área médica e de software envolvidos com o tema, apresentará o desenvolvimento do projeto de *hardware* e *software* do protótipo do oxímetro, bem como a sua implementação. Alguns termos da área médica utilizados:

- a) Oximetria de pulso: pode ser definida como uma medida que busca, em primeira instância, garantir níveis adequados de oxigênio no sangue arterial para evitar a hipoxemia. (NUNES; TERZI, 1999).

- b) Hipoxemia: entende-se como uma deficiência de concentração de oxigênio no sangue arterial. Logo, o oxímetro de pulso permite um monitoramento do estado do paciente de uma forma contínua e não-invasiva, através da saturação parcial de oxigênio.
- c) Método não-invasivo: é caracterizado pela ausência de invasores no corpo do paciente, ou seja, não há necessidade de inserir instrumentos furando ou fazendo incisões em qualquer parte do corpo.

O escopo deste trabalho delimita-se em apresentar uma forma de implementação de um oxímetro de pulso, descrevendo todos os recursos, ferramentas e métodos necessários para que seja construído um protótipo funcional, pretendendo atender a uma necessidade - a conectividade com os microcomputadores pessoais. Para isso, será apresentado todos os passos e processos necessários, fazendo-se com que se unam todos os componentes necessários. Grande parte das figuras e diagramas utilizados neste trabalho é de própria autoria.

1.3 Objetivos

1.3.1 Objetivo geral

O objetivo desse trabalho é desenvolver um protótipo de oxímetro de pulso, oferecendo conectividade ao microcomputador e permitindo aos usuários terem uma melhor interpretação e manipulação dos dados obtidos.

1.3.2 Objetivos específicos

Este trabalho tem como objetivos específicos:

- a) Definir os requisitos;
- b) Projetar o *hardware*;
- c) Projetar o *software*;
- d) Construir o protótipo.

1.4 Metodologia

A metodologia para o desenvolvimento desse trabalho é composta das seguintes fases:

1.4.1 Revisão de Literatura

A revisão de literatura será realizada por meio de pesquisa bibliográfica, ou seja, utilizaram-se bases de dados de teses e artigos, livros e periódicos disponíveis na Internet como fontes de pesquisa.

1.4.2 Projeto de *hardware*

O oxímetro fará a leitura dos sinais biométricos do paciente através de um módulo de oximetria, dotado de um sensor. Será tarefa do *hardware* e do *software* proposto por esse trabalho ler e interpretar os valores obtidos e exibi-los num display, assim como transmiti-los a um microcomputador através de uma interface serial. Nesta etapa, será desenvolvido um projeto de *hardware*, mostrando seus principais componentes e seu inter-relacionamento.

O projeto do *hardware* do oxímetro será desenvolvido de acordo com os requisitos de *software*, pois será a base onde ele será executado. Inicialmente, deve-se fazer uma seleção de componentes eletrônicos, levando em consideração a função e o custo de cada um. Essa seleção de componentes será realizada de acordo com necessidades específicas, tais como amplificadores operacionais para realizar ganhos de tensão, circuitos integrados diversos, *display*, teclado, dentre outros. Para realizar a interligação entre esses componentes, constituindo um circuito eletrônico, será necessário realizar consultas aos *datasheets* dos fabricantes, os quais nos darão informações e especificações sobre os componentes. Tais *datasheets* estão disponíveis no próprio site de cada fabricante. Com essas informações em mãos, o próximo passo será desenvolver um diagrama de blocos e um Esquema Elétrico utilizando CAD (Desenho Auxiliado por Computador) e efetuando simulações via software para comprovar o funcionamento.

1.4.3 Projeto de *software*

Em relação ao *software*, será dividido em dois: um módulo que executará no próprio oxímetro, formando um sistema embarcado, e o outro que será utilizado num microcomputador, para um melhor tratamento dos dados. O usuário poderá interagir com os dois *softwares*, uma vez que ambos irão apresentar uma interface, seja num pequeno *display* disponível no protótipo, quanto na tela do computador, onde terá um acesso mais completo aos dados e permitindo uma melhor análise dos mesmos. Uma vez levantado todos os

requisitos, inicia-se a fase de implementação. No caso do *software* embarcado, que será programado no microcontrolador, deverá ser utilizada uma linguagem própria para esse dispositivo, tal como o ANSI C, através do ambiente integrado *mikroC 7*, desenvolvida pela *mikroElektronica*.

O *software* a ser executado no computador, poderá ser implementado em qualquer outra linguagem de alto nível, como *Java*.

1.4.4 Construção do protótipo

O *software* será desenvolvido utilizando-se um microcomputador normal, no compilador *mikroC*. A construção do protótipo poderá ser feita sob um *protoboard*, onde dispomos de uma matriz de pontos que facilitam a interligação entre os componentes eletrônicos, formando assim uma versão funcional do hardware. O passo seguinte será alimentar o circuito fornecendo-o a tensão previamente calculada, e carregar o *software* na memória do microcontrolador, utilizando um PC. Com o circuito em funcionamento, faz-se necessário realizar medições e ajustes documentando-se todas as alterações.

1.4.5 Validação

A validação dos dados exibidos no *display*, seja ele a saturação de oxigênio ou a frequência cardíaca do paciente, pode ser comprovada através de um simulador, devidamente calibrado e aferido pelo INMETRO. Para realizar essa tarefa, o simulador toma o lugar do paciente, enviando os sinais biométricos ao protótipo. Será utilizado o simulador da marca *R&D Mediq*, modelo *HS20*, que é capaz de simular valores de SpO_2 de 0% a 99% e batimentos cardíacos de 30 a 240 BPM.

2 REVISÃO DA LITERATURA

2.1 Oxímetro de Pulso

Os oxímetros de pulso são largamente utilizados na prática clínica, principalmente nas Unidades de Terapia Intensiva (UTI) para monitorar a saturação de oxigênio, detectando e prevenindo a hipoxemia. Monitorar a saturação de oxigênio durante a anestesia é um procedimento padrão, que é sempre feito utilizando-se oxímetro de pulso. Esses oxímetros também são úteis durante procedimentos de broncoscopia, endoscopia, cateterização cardíaca, testes físicos e estudos do sono (WEBSTER, 2006, p.472).

Esse equipamento é um monitor que realiza medidas da saturação da hemoglobina do sangue arterial (SpO₂) pela transmissão de dois comprimentos de onda de um sinal luminoso, geralmente 660nm (vermelho) e 940nm (infravermelho), que atravessa ou é refletida pelos tecidos humanos (ABNT, 1997). O princípio de funcionamento do oxímetro de pulso é baseado na lei Beer-Lambert que relaciona a concentração de um soluto com a intensidade de luz transmitida através de uma solução (KNOBEL, 1998). Sendo assim, o oxímetro consegue diferenciar a absorção de energia luminosa devido ao sangue arterial relacionando a um valor de SpO₂ (FERNANDES; OJEDA; LUCATELLI, 2001). O equipamento contém um *probe* (compartimento que contém o sensor) no corpo do paciente e este sensor está ligado a uma unidade computadorizada que informa os valores obtidos (FEARNLEY, 1995).

Alguns oxímetros encontrados no mercado, tais como o Dixtal modelo DX-2515 (HOLZHACKER, 2001), são apropriados para utilização com pacientes adultos, pediátricos e neonatais. Este modelo proporciona uma mensuração fidedigna contínua, apresentação dos valores e alarmes para a SpO₂ e pulso cardíaco, podendo funcionar através de uma tomada de energia elétrica ou de sua bateria interna recarregável. A saturação de oxigênio e o pulso cardíaco são atualizados uma vez a cada segundo e imediatamente exibidas num *display LCD*. Este equipamento ainda emitirá aviso sonoro quando algum nível fisiológico crítico for detectado (HOLZHACKER, 2001).

2.2 Conceitos de *Hardware*

O desenvolvimento do *hardware* para o oxímetro proposto nesse trabalho envolverá os seguintes conceitos:

- a) Circuitos integrados: são circuitos cujos componentes e conexões são feitas em áreas distintas de um único *chip*, construído de um material condutor, como o silício (GIBILISCO, 2001, p.371).
- b) Microcontroladores: são circuitos integrados de baixo custo, rotulados como *Single-chip computer*. Isso significa que, mesmo encapsulado num simples *chip* de silício, ele tem as mesmas características de um computador pessoal. O microcontrolador é capaz de armazenar e rodar programas – sua mais importante característica. Ele contém uma CPU (Unidade Central de Processamento) , memória RAM (Memória de Acesso Randômico) e ROM (Memória Somente Leitura), linhas de entrada e saída, contadores, e em alguns modelos incluem até conversores de sinais analógicos para digital e vice-versa (IOVINE, 2000, p.1).
- c) Comunicação Serial: é a transmissão de informações ao longo de um caminho, onde os dados são enviados um após o outro (GIBILISCO, 2001, p.620).

O projeto de hardware e seu desenvolvimento consistem das seguintes etapas (IOVINE, 2000):

- a) Definir os requisitos;
- b) Coletar informações dos componentes em questão;
- c) Adquirir componentes que atendam aos requisitos;
- d) Criar um Diagrama de Blocos preliminar;
- e) Desenhar um Esquema Elétrico usando CAD (desenho auxiliado por computador);
- f) Efetuar simulações utilizando *softwares* específicos ou montando sobre um *protoboard*;
- g) Documentar o projeto e gerar a lista de materiais;
- h) Iniciar o desenho da Placa de Circuito Impresso (PCI);
- i) Revisar e verificar o circuito em operação utilizando-se de osciloscópios, multímetros e outras ferramentas;
- j) Atualizar e completar a documentação incluindo as mudanças no projeto.

A ordem das tarefas acima é independente, e algumas delas podem ser feitas em paralelo. O projeto do *hardware* normalmente é feito em paralelo com o *software*, garantindo que ele seja totalmente compatível com a versão em desenvolvimento (ARNOLD, 2000, p.21). Por isso, se torna extremamente útil a utilização de ferramentas de simulação computadorizadas, como é o caso do software *Proteus VSM*, desenvolvido pela *LabCenter* (HILLS, 1999). Simuladores como esse buscam testar a CPU a ser utilizada, puramente via *software*, possibilitando analisar o comportamento do circuito eletrônico de diversos pontos. A grande vantagem desse simulador é a capacidade de exercitar o *software* interagindo com o *hardware* (HILLS, 1999).

Segundo Velloso (2006, p.31), nas atividades que envolvem projetos eletrônicos, o uso da expressão gráfica de esquemas é muito importante para a simplificação da complexidade de um circuito, que utiliza numerosos componentes e dispositivos. O desenvolvimento de um projeto eletrônico deve conter desenhos de diagramas eletrônicos, em forma Diagrama de Blocos, com a descrição funcional detalhada dos componentes e com o desenho do Esquema Eletrônico completo.

O Diagrama de Blocos é uma representação do circuito desenhada por linhas simples e figuras geométricas, e nela estão contidas as informações funcionais básicas, interligadas por setas que indicam o curso do sinal através do sistema ou do circuito elétrico do dispositivo. Os Esquemas Elétricos indicam as ligações básicas necessárias à compreensão do funcionamento de um circuito. Os componentes devem indicar suas numerações e valores (resistência, capacitância, tipo de componente). Para representar os componentes, utiliza-se de símbolos gráficos estabelecidos por diversas normas da ABNT (SEDRA, 2004), dentre elas:

- NBR 5448: Símbolos Gráficos de Resistores;
- NBR 5449: Símbolos Gráficos de Capacitores;
- NBR 5446: Símbolos Gráficos de Relacionamento usados na Confecção de Esquemas;
- NBR 5280: Símbolos Literais de Identificação de Elementos de Circuito;
- NBR 5264: Símbolos Gráficos de Dispositivos de Conexão.

As ligações entre os componentes são esquematizadas através de linhas que representam fios idéias, ou seja, sem nenhuma resistência. A esquematização de circuitos integrados se dá através de suas portas lógicas, por exemplo, com o desenho da porta e a indicação do número do terminal (pino) ao qual corresponde. Nos outros circuitos integrados complexos, como é o

caso do microcontrolador, a notação usada é um retângulo, o qual representa a função de seus terminais e a indicação numérica do mesmo, além do número que ele ocupa no projeto.

2.3 Conceitos de *Software*

A Engenharia de *Software* é a aplicação de uma abordagem sistemática, disciplinada e quantificável, para o desenvolvimento, operação e manutenção do *software*; isto é, a aplicação da engenharia ao *software*. Ela cria e utiliza-se de sólidos princípios da engenharia com o intuito de obter softwares econômicos que sejam confiáveis e que trabalham eficientemente em máquinas reais (PRESSMAN, 2006). Seus métodos proporcionam representações gráficas, introduzem e asseguram qualidade ao *software*, fornecem apoio automatizado ou semi-automatizado através de ferramentas e também nos proporcionam meios para avaliar o progresso.

Segundo Pressman (2006), a engenharia de *software* é uma tecnologia em camadas, as quais são as seguintes: foco na qualidade, processo, métodos e ferramentas. A engenharia de *software* deve se apoiar num compromisso organizacional com a qualidade, a qual é considerada a camada base. A camada de processo é considerada o alicerce, a qual age como um adesivo que mantém unidas as camadas de tecnologia. Ela define um arcabouço que deve ser estabelecido para a efetiva utilização da tecnologia de *software* (PRESSMAN, 2006). Os métodos fornecem a técnica de “como fazer” os *softwares* e abrangem um conjunto de tarefas que incluem comunicação, análise de requisitos, projeto, implementação, testes e manutenção. As ferramentas fornecem o apoio automatizado ou semi-automatizado para o processo e para os métodos.

Um arcabouço de processo genérico é definido por Pressman (2006), o qual é aplicável a grande maioria dos projetos de *software*. Ele é composto pelas seguintes atividades:

- Comunicação: envolve alta comunicação e colaboração com o cliente (e demais interessados) e abrange o levantamento de requisitos e outras atividades relacionadas;
- Planejamento: define um plano para o trabalho de engenharia de *software* a ser desenvolvido. Descreve as tarefas técnicas a ser conduzida, a previsão de riscos, a necessidade de recursos, os resultados a serem produzidos e um cronograma de trabalho;

- Modelagem: inclui a geração de modelos que possibilitam ao desenvolvedor e ao cliente, compreender melhor os requisitos do *software* e o projeto que vai atender a esses requisitos;
- Construção: refere-se à geração de código (manual ou automático), bem como aos testes necessários para revelar erros no código;
- Implantação: O software parcial ou completo é entregue ao cliente, que o avalia e fornece *feedback* ao desenvolvedor;

Modelos prescritivos de processo foram originalmente propostos para colocar ordem no caos do desenvolvimento de *software*. Esses modelos definem um conjunto distinto de atividades, ações, tarefas, marcos e produtos de trabalho que são indispensáveis para se fazer engenharia de *software* com alta qualidade. Existem vários modelos prescritivos tais como: cascata, incrementais, RAD, evolucionários e especializados.

2.3.1 UML

A *Unified Modeling Language* (UML) é uma linguagem visual para modelar sistemas. Isso quer dizer que ela é constituída de elementos gráficos que permitem representar os conceitos do software a ser desenvolvido. Através dos elementos gráficos definidos nessa linguagem, podem-se construir diagramas que representam diversas perspectivas de um sistema (BEZERRA, 2002).

Segundo Bezerra (2002), cada elemento gráfico possui uma *sintaxe* (isto é, uma forma pré-determinada de se desenhar um elemento) e uma *semântica* que definem o que significa o elemento e para que ele deva ser utilizado.

Um processo de desenvolvimento que utilize a UML como linguagem de suporte à modelagem envolve a criação de diversos documentos. Na terminologia da UML, esses documentos são denominados *artefatos de software*. São os artefatos que compõe as visões do sistema.

Esses artefatos gráficos produzidos durante o desenvolvimento de um sistema de software são definidos através da utilização dos diagramas da UML. Os diagramas da UML são listados na Figura 1.

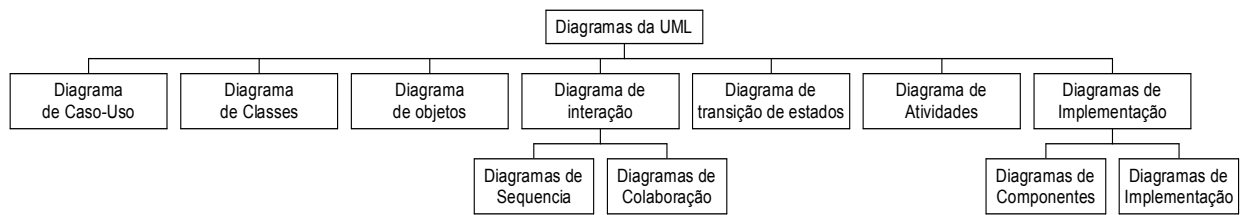


Figura 1 - Diagramas da UML

No decorrer deste trabalho, serão usados os seguintes diagramas:

- Diagrama de Componentes: ilustra os vários componentes de um *software* e suas dependências. Uma dependência liga um componente a outro que lhe fornece algum tipo de serviço (BEZERRA, 2002, p.249). Um componente é uma unidade de *software* que pode ser utilizada na construção de vários sistemas e que pode ser substituída por outra unidade que tenha a mesma funcionalidade. Os componentes podem prover acesso aos seus serviços através de uma interface. Além de oferecer serviços a outros componentes do sistema, ele também pode requisitar serviços.
- Diagrama de Atividade: representam-se os estados de uma atividade específica. Na verdade, o diagrama de atividade pode ser visto como uma extensão dos fluxogramas, porém com uma notação ligeiramente diferente (BEZERRA, 2002, p.228). Um diagrama de atividade exhibe os passos de uma computação. Cada estado corresponde a um dos passos da computação, onde o sistema está realizando algo.
- Diagrama de Casos de Uso: corresponde a uma visão externa do sistema e representam graficamente os atores, casos de uso e relacionamento entre esses elementos.
- Diagrama de Classes: utilizado na construção de um modelo de classes desde o nível de análise até o nível de especificação detalhada. De todos os diagramas da *UML*, esse é o mais rico em termo de notação (BEZERRA, 2002, p.97).
- Diagrama de Seqüência: é um tipo de diagrama de interação, ou seja, é utilizado para modelar a lógica de um cenário de Casos de Uso. Ele também é utilizado para modelar a troca de mensagens entre objetos, dando ênfase na ordem temporal das mensagens trocadas entre eles (BEZERRA, 2002, p.97). Um conceito importante ilustrado por esse diagrama são os Objetos de Fronteira, responsáveis por apresentar os resultados de uma interação dos objetos internos em algo que possa ser entendido pelo ator. Segundo Bezerra, esses Objetos de Fronteira existem para que o sistema possa se

comunicar com o mundo exterior, podendo ser uma interface gráfica com o usuário, por exemplo.

2.3.2 Processo Unificado

Um dos processos derivados de modelos prescritivos apresentados é o Processo Unificado. Ele faz uso da Linguagem de Modelagem Unificada (*Unified Modeling Language – UML*), é dirigido por Casos de Uso e também é centrado na arquitetura do *software*. Outra característica interessante do Processo Unificado é que ele constitui-se de um processo incremental, baseado em componentes. Possui uma infra-estrutura genérica de processos que pode ser especializada para uma ampla classe de sistemas de *software*, de diferentes tamanhos e diferentes áreas (FALBO, 2000), possibilitando o desenvolvimento de forma incremental e iterativa (JACOBSON *et al.*, 1998), composta de quatro fases (SANTANDER; VASCONCELOS, 2000):

- a) Concepção: nesta fase objetiva-se definir os casos de uso mais críticos, as funções chave do sistema e delimita-se o escopo do produto a ser desenvolvido;
- b) Elaboração: define-se a descrição arquitetural do *software*. Procura-se também definir a maioria dos casos de uso, capturando os requisitos do *software*;
- c) Construção: foca-se o projeto e a implementação, visando dar capacidade operacional ao *software*;
- d) Transição: envolve a realização de testes com o usuário, corrigir defeitos encontrados e realizar treinamento.

Cada uma destas fases é geralmente composta por cinco *workflows*, denominados de *Requisitos*, *Análise*, *Projeto*, *Implementação* e *Teste*, como mostra a Figura 2.

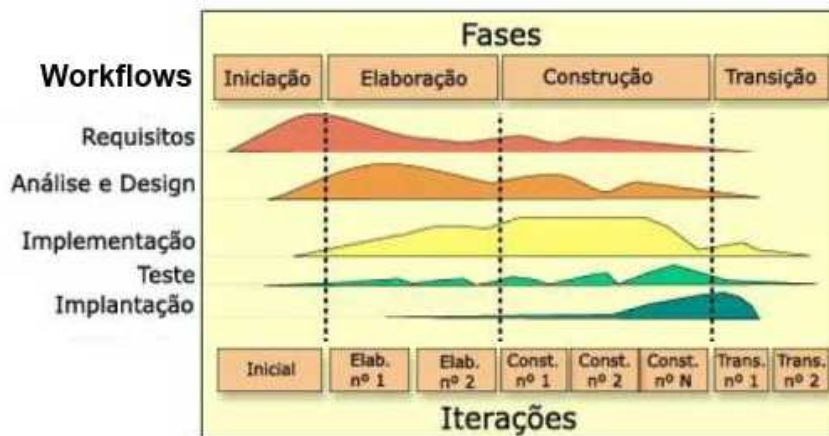


Figura 2 – Workflows do Processo Unificado

Fonte: O Processo Unificado

Dependendo da fase, maior ou menor nível de detalhes pode ser desenvolvido em um *workflow* específico. Na fase de concepção, por exemplo, a maior concentração de esforços estará nos *workflows* iniciais de requisitos e análise. Os outros *workflows* de Projeto, Implementação e Teste podem conter menos informações e menos detalhes.

O conceito de processo iterativo vem da propriedade de que cada fase pode conter uma ou mais iterações de desenvolvimento, sendo que cada iteração realiza um ciclo completo nos *workflows*. Depois de concluída uma iteração, ocorre uma evolução, a qual incrementa funcionalidades e/ou melhorias ao sistema (SANTANDER; VASCONCELOS, 2000).

2.3.3 Sistema Embarcado

Um sistema embarcado, ou sistema embutido, é um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. Diferente de computadores de propósito geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas pré-definidas, geralmente com requisitos específicos. Já que o sistema é dedicado à tarefas específicas, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto (MARWEDEL, 2003). O protótipo do oxímetro, que engloba tanto o *hardware* como o *software*, poderá ser classificado como um sistema embarcado, uma vez que possui as seguintes características :

- São projetados para realizar uma função ou uma gama de funções, e não para ser reprogramado por usuários finais;

- b) Interagem com o ambiente em que se encontram, coletando dados de sensores e atuando utilizando atuadores, tais como indicadores sonoros ou sinais luminosos;
- c) Devem ser confiáveis, estáveis e possuir características de recuperação em caso de falhas e oferecer segurança, tanto em relação ao ambiente em que está inserido quanto às informações neles contidas;
- d) Possuem métricas de eficiência além das já conhecidas para projetistas de sistemas *desktops* ou servidores: consumo de energia, tamanho do código, execução eficiente, peso e custo.

3 DESENVOLVIMENTO

3.1 Projeto de *Hardware*

De acordo com o que já foi exposto anteriormente, o *software* do protótipo necessitará de uma plataforma especialmente desenvolvida, ou seja, faz-se necessário criar um circuito eletrônico que contenha todas as partes necessárias para o funcionamento do sistema. A Figura 3 representa todo o equipamento através de um Diagrama de Blocos.

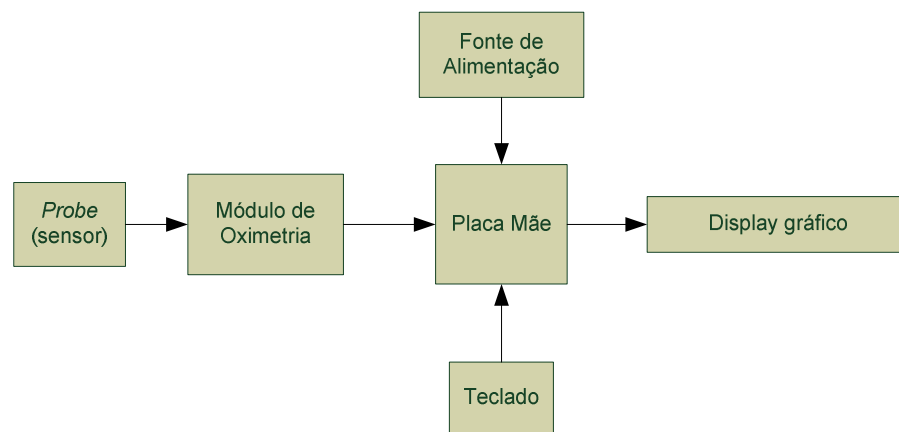


Figura 3 - Diagrama de Blocos do protótipo

Através desse diagrama, representa-se a constituição do equipamento, com as devidas relações de funcionamento que existem entre os diversos constituintes, a saber:

- Fonte de Alimentação: exigida por quase todos os circuitos eletrônicos, é responsável por abaixar a tensão fornecida pela rede elétrica para níveis requeridos pelo circuito a ser alimentado. Por sua vez, também é composta pelos seguintes circuitos:
 - Circuito Retificador: transforma a corrente alternada para corrente contínua;
 - Circuito Carregador de Baterias: responsável por manter a bateria sempre carregada e pronta para assumir caso haja falhas ou ausência de rede elétrica;
 - Circuito Regulador de Tensão: usado para manter as tensões fornecidas em valores constantes, ou seja, serve como uma linha de defesa contra picos de tensão e instabilidade da rede elétrica.
- Placa Mãe: responsável por unir os periféricos, tais como o sensor, módulo de oximetria, *display* gráfico, fonte e teclado. Sua função é criar meios para que o microcontrolador possa comunicar-se com todos os componentes do protótipo, com a

maior velocidade e confiabilidade possível. Destaca-se também por fornecer a alimentação necessária a todos os outros componentes do sistema;

- *Probe* (sensor): é a unidade que vai junto ao corpo do paciente, emitindo radiações visíveis e infravermelhas. Sua função é informar ao Módulo de Oximetria a quantidade de luz absorvida proporcional à saturação de oxigênio;
- Módulo de Oximetria: é uma unidade computadorizada, composta por processadores digitais que calculam tanto a saturação de oxigênio no sangue quanto à frequência cardíaca, através de equações matemáticas. Os resultados são enviados para a Placa Mãe no formato serial, ou seja, um *bit* após o outro;
- *Display* gráfico: unidade dotada de um Pannel de Cristal Líquido (LCD), composto por *pixels* monocromáticos, retro-iluminados. Possui um controlador interno responsável por interpretar os dados da Placa Mãe. Através desse *display*, é possível mostrar ao usuário os valores medidos e a curva pletismográfica, bem como outros sinais visuais;
- Teclado: é um conjunto de teclas que permitem a seleção dos parâmetros de alarme, modos de operação, contraste do *display* e qualquer outra função que necessite intervenção do usuário. É composto por teclas tácteis, com contatos do tipo “normalmente abertos”. É função do *software* determinar se um botão está pressionado ou não;

3.1.1 Fonte de Alimentação

A Fonte de Alimentação do protótipo é responsável por reduzir a tensão de 110VAC disponível na rede elétrica para os níveis requeridos pelos outros componentes (ARNOLD, 2000). As Figuras 4 e 5 representam, respectivamente, o Diagrama de Blocos e o Esquema Elétrico da fonte, composta pelos circuitos Retificador, Regulador de Tensão e Carregador de Baterias.

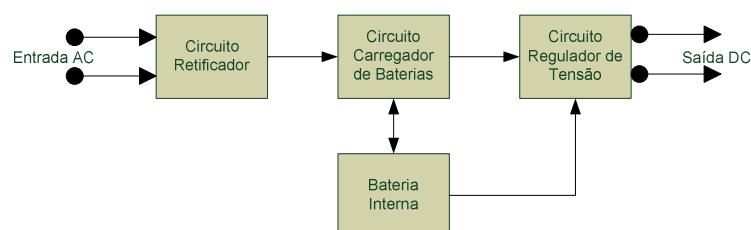


Figura 4 - Diagrama de Blocos da Fonte de Alimentação

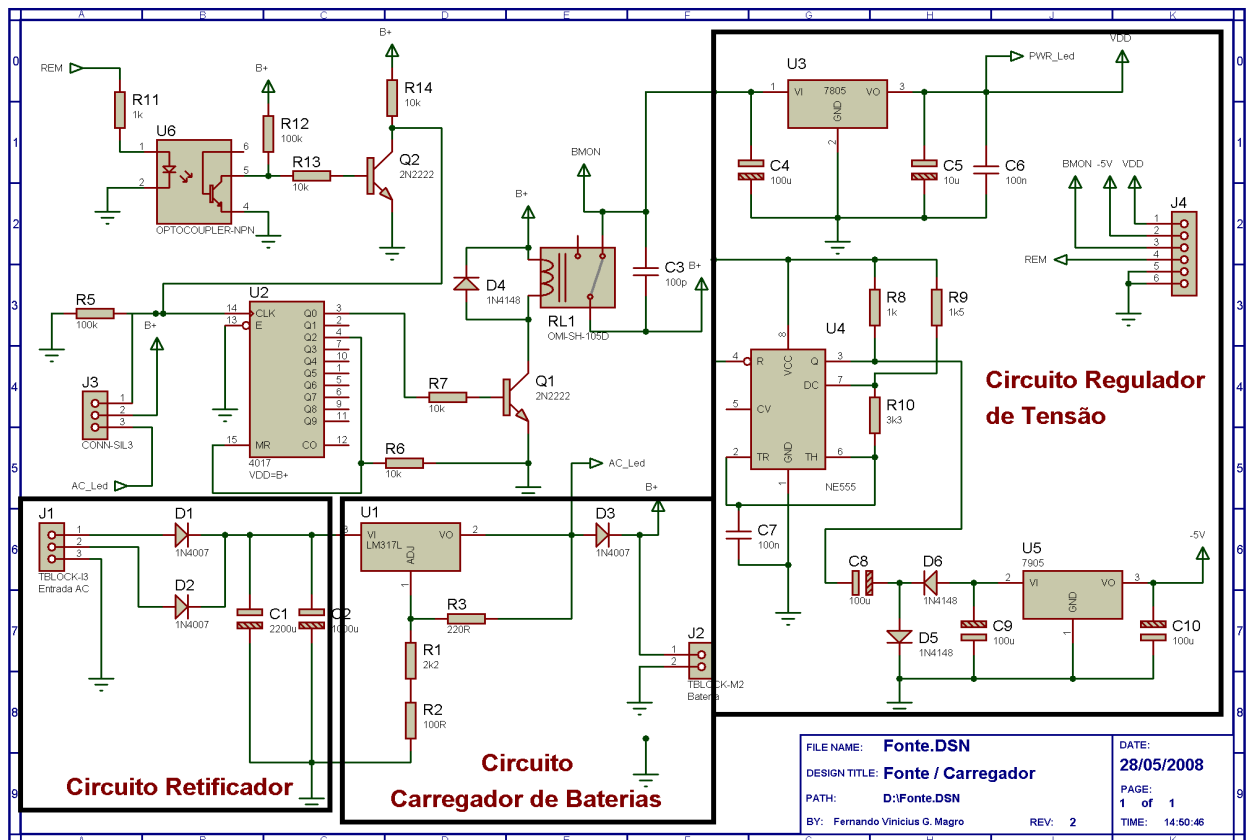


Figura 5 - Esquema Elétrico Completo da Fonte de Alimentação

Todos os outros diagramas presente nesse trabalho foram desenhados com o auxílio da ferramenta *ISIS Professional*, comercializada junto com pacote *Proteus*, desenvolvido pela *Labcenter* (HILLS, 1999).

3.1.1.1 Circuito Retificador

O circuito Retificador, ilustrado na Figura 6, é responsável por converter a corrente alternada, disponível na rede elétrica, para corrente contínua, uma vez que todos os outros componentes do protótipo precisam ser alimentados dessa forma. Esse tipo de circuito é conhecido como “Retificador de Onda Completa”.

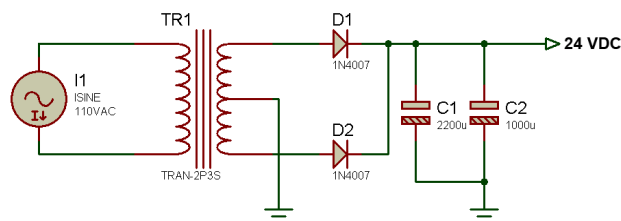


Figura 6 – Circuito Retificador

O transformador TR1 é do tipo 24+24, responsável por abaixar a tensão de 110VAC para 24VAC, fornecendo uma corrente máxima de 1A. Os diodos D1 e D2 são do tipo 1N4007, retificadores de uso geral, com uma baixa queda de tensão e resistente a surtos de corrente. De acordo com Mims (1986), os diodos são componentes que conduzem eletricidade somente em uma direção, enquanto bloqueia o fluxo de corrente numa direção oposta. É importante notar que a corrente de saída do transformador ainda é alternada, porém a tensão é de 24VAC oscilando numa frequência de 60 Hz. Dessa forma, os diodos D1 e D2 irão transformar essa corrente em contínua, mantendo a amplitude (tensão). Esse fenômeno é ilustrado na Figura 7.

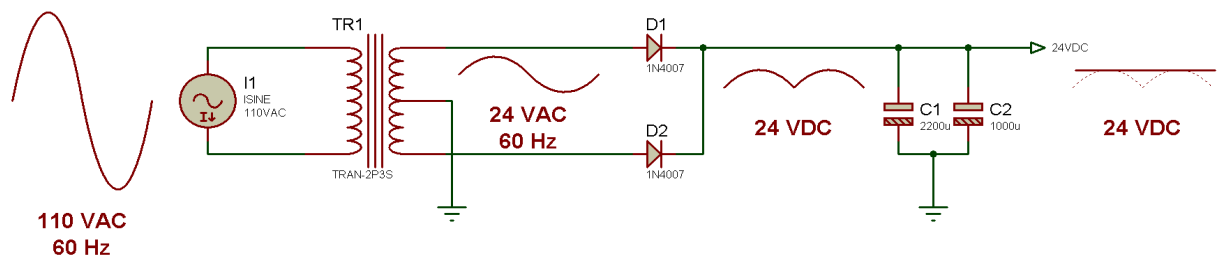


Figura 7 – Etapas do Circuito Retificador

Os capacitores C1 e C2 são do tipo eletrolítico, de 2200 μ F e 1000 μ F respectivamente, agindo de modo a eliminar o *ripple*. De acordo com a Figura 7, o *ripple*, que é a componente alternada (VAC) que incide sobre a corrente contínua (VDC), foi eliminado ou significativamente reduzido utilizando-se esses capacitores. Sendo assim, temos 24 VDC na saída desse Circuito Retificador.

3.1.1.2 Circuito Carregador de Baterias

O protótipo do oxímetro deverá oferecer uma segunda fonte de energia; caso a rede elétrica apresente falhas, ou interrupções no fornecimento, uma bateria interna deverá assumir. A alimentação interna também traz mobilidade ao equipamento. Neste caso, a bateria deverá ser capaz de alimentar o protótipo por no mínimo duas horas, tempo necessário para transporte de pacientes críticos entre hospitais, auxiliando o processo de oxigenoterapia com o intuito de diminuir a mortalidade e a incidência de hipoxemia (MARCONDES, 2006).

Entretanto, para manter a bateria num nível confiável, sempre pronta para assumir a alimentação em qualquer momento, o Circuito Carregador de Baterias tem a função de

fornecer uma tensão constante à bateria, chamada de Tensão de Flutuação. A Figura 8 mostra a representação esquemática do circuito.

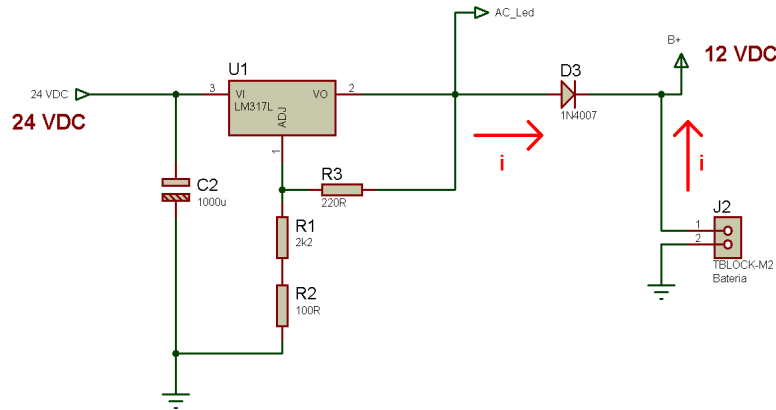


Figura 8 - Circuito Carregador de Baterias

De acordo com o esquemático acima, o circuito integrado U1 é do tipo LM317L, um Regulador de Tensão Variável. Sua função é reduzir a tensão de 24 VDC na sua entrada, dada pelo Circuito Retificador, para 13,6 VDC – a Tensão de Flutuação da bateria. Segundo Emadi (2005, p.17-18), a Tensão de Flutuação (Tf) para um bateria selada é de 2,27 Volts por célula (V_{pc}). De acordo com as especificações da bateria selada utilizada no protótipo, fabricada pela Rontek, ela possui 6 células e tem uma tensão de 12V entre seus pólos. Logo, a Tensão de Flutuação pode ser comprovada multiplicando-se a tensão necessária por célula pela sua quantidade:

$$Tf = 2,27V \times 6 = 13,62V \quad (1)$$

Para fazer com que a saída de U1 esteja regulada a 13,6V (Tf), se faz necessário o uso de resistores externos, identificados por R1, R2 e R3, como visto em seu *datasheet* National (1982). É importante observar a função do diodo D3, do tipo 1N4007; ele é responsável por direcionar o fluxo da corrente, denotada por “i” na Figura 8. O conector J2 vai diretamente à bateria, cujo pino 1 vai ao pólo positivo e o outro ao negativo.

Dessa forma, teremos 13,6V regulados no final do circuito quando alimentado pela rede elétrica. Entretanto, com falhas ou ausência da rede, teremos 12V de saída pois a bateria assumirá a alimentação. Essa variação devido ao tipo de fonte, externa ou interna

respectivamente, não irá exercer influência na próxima etapa da fonte, que será dedicada à regulação.

3.1.1.3 Circuito Regulador de Tensão

Este circuito é alimentado pela saída do Carregador de Baterias ou pela própria bateria interna, na ausência de energia elétrica. A Figura 9 ilustra o circuito e seus componentes.

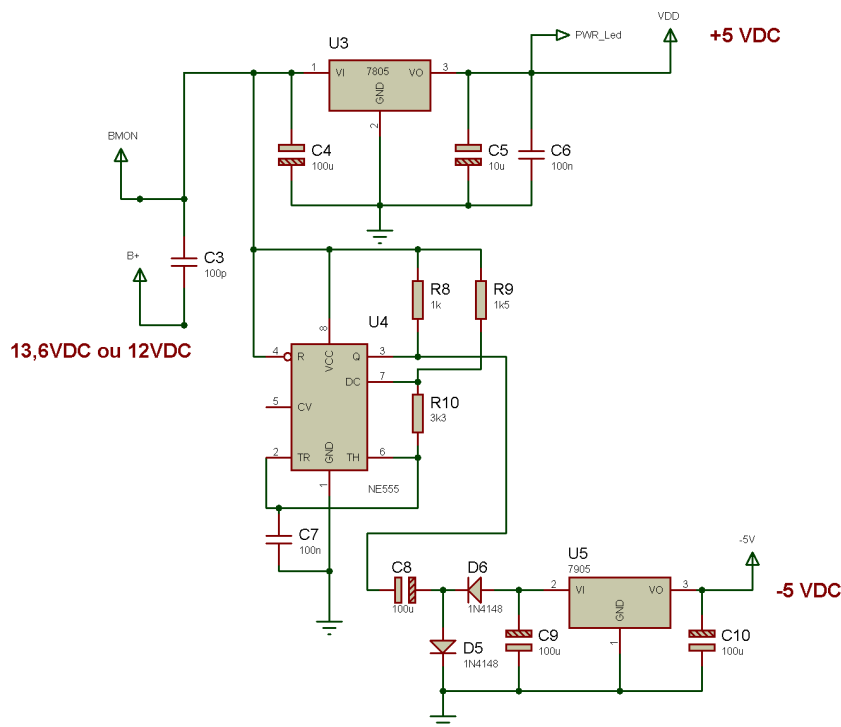


Figura 9 - Circuito Regulador de Tensão

Entretanto, as tensões fornecidas pela fonte ou pelo Carregador de Baterias ainda não apresentam níveis que poderão ser utilizados na alimentação dos demais componentes do protótipo. Tais componentes necessitam ser alimentados por tensões mais baixas, devidamente reguladas, ou seja, independente de alterações devido picos de consumo e imune às oscilações. A função desse Circuito Regulador de Tensão é converter a tensão aplicada na sua entrada para um valor fixo, geralmente menor (MIMS, 1986).

Em relação ao esquema da Figura 9, o componente U3 é um regulador de tensão da família 7805. Ele admite uma entrada de até 35 Volts, enquanto sua saída sempre será regulada a 5 Volts, e esta diferença será dissipada na forma de calor. Todos os capacitores da Figura 9 têm a finalidade de fornecer rapidamente picos de corrente elevada que os *chips* exigem

substancialmente. Eles são chamados de Capacitores de Desacoplamento. Os outros circuitos integrados, U4 e U5, são utilizados para gerar uma tensão negativa de amplitude -5 Volts, utilizada pelo amplificador operacional da Placa Mãe.

3.1.2 Placa Mãe

A Placa Mãe do protótipo será aquela em que estará presente o microcontrolador, componente principal de equipamento. A Figura 10 mostra o Esquema Elétrico da Placa Mãe.

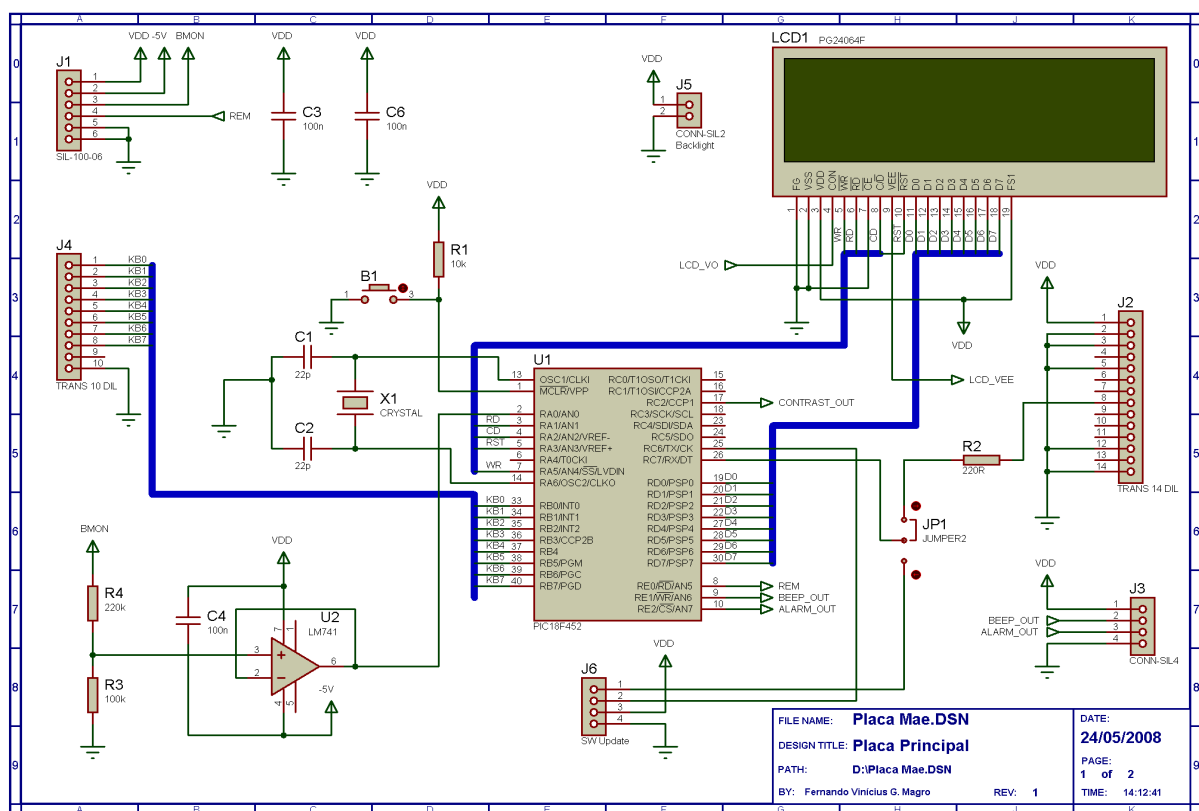


Figura 10 - Esquema Elétrico da Placa Mãe

De acordo com o *datasheet* Microchip (2002), o microcontrolador U1, do tipo PIC18F452 é um dispositivo de 40 pinos, que trabalha numa frequência máxima de operação de 40 Mhz, possui 32kb de memória interna não-volátil e é capaz de rodar programas escrito pelo usuário. É nesse *chip* que o *software* do oxímetro será armazenado.

O *display*, apesar de ser representado no Esquema da Figura 10, é um componente fisicamente separado da Placa Mãe. Ele está nesse diagrama somente para ilustrar as conexões diretas com o microcontrolador. As conexões com o *display* estão representadas no Quadro 1.

Pino do Microcontrolador		Pino do <i>Display</i>	
RA1/AN1	3	RD	6
RA2/AN2	4	CD	8
RA3/AN3	5	RST	10
RA5/AN4	7	WR	5
RD0	19	D0	11
RD1	20	D1	12
RD2	21	D2	13
RD3	22	D3	14
RD4	27	D4	15
RD5	28	D5	16
RD6	29	D6	17
RD7	30	D7	18

Quadro 1 - Conexões entre o microcontrolador e o *display*

Fonte: Manual de Usuário mikroC.

A frequência de operação do sistema, ou seja, a quantidade de instruções de códigos que serão executadas em um segundo são definidas pelo conjunto de componentes X1, C1 e C2. Segundo o *datasheet* Microchip (2002, p.18), os valores de C1 e C2 para um cristal X1 de 10Mhz, poderá ser entre 15pF e 33pF. Logo, foram selecionados capacitores de 22pF por serem comercialmente disponíveis, de fácil obtenção.

Outro componente importante da Placa Mãe, que merece atenção, é o Amplificador Operacional referenciado como U2, do tipo LM741, ilustrado na Figura 11.

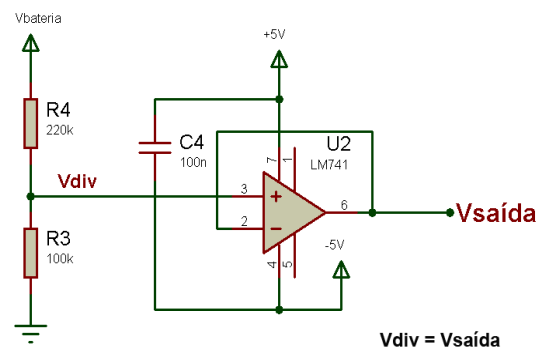


Figura 11 – Divisor de tensão e Amplificador Operacional de Ganho Unitário

O LM741 é um amplificador linear de alto desempenho com uma grande variedade de usos. Neste caso, ele está configurado como um Amplificador de Ganho Unitário (MIMS, 1985). Sua função é reduzir a tensão da bateria, que entra no resistor R4, para um nível entre 0 e 5

Volts – os valores aceitáveis para as entradas analógicas do microcontrolador (MICROCHIP, 2002), a fim de monitorar o nível da bateria via *software*. Os resistores R4 e R3 de 220kΩ e 100kΩ, respectivamente, funcionam como Divisores de Tensão segundo a equação:

$$V_{div} = \frac{V_{bateria} \times R3}{(R3 + R4)} \quad (2)$$

Substituindo os valores de R3 e R4, obtemos a seguinte relação:

$$V_{div} = 0,303 \times V_{bateria} \quad (3)$$

Logo, quando a bateria atingir sua mais alta tensão (carga máxima), de 13,6 Volts, a entrada do microcontrolador irá receber somente 4,12 Volts, garantindo um valor aceitável para a entrada analógica. A função do amplificador, nesse caso, não é amplificar, uma vez que seu ganho é unitário; mas sim evitar que a entrada analógica consuma corrente dos resistores divisores, influenciando os valores medidos. Portanto, a corrente é consumida do amplificador (MIMS, 1985).

Os capacitores C3, C4 e C6 são de Desacoplamento, com a função de garantir uma alimentação apropriada ao circuito. O conector J1 é responsável por receber a alimentação da fonte, o J2 é responsável por comunicar-se com o Módulo de Oximetria e o J4 com o Teclado.

3.1.3 *Probe* (sensor)

A Figura 12 é uma fotografia do sensor utilizado no protótipo, conhecido como *probe*.



Figura 12 - *Probe*, sensor de oximetria

O sensor é o componente do oxímetro responsável por obter os níveis de saturação de oxigênio do sangue do paciente. Possui dois LEDs (Diodo Emissor de Luz) na parte superior, emitindo luz em comprimentos de onda distintos: vermelho e infravermelho. Na parte

inferior, ele possui um Fotodiodo, componente sensível à luz, cuja função é registrar a parcela absorvida pelo sangue. Ele é conectado ao Módulo de Oximetria através de um cabo. A Figura 13 mostra os componentes básicos do sensor.

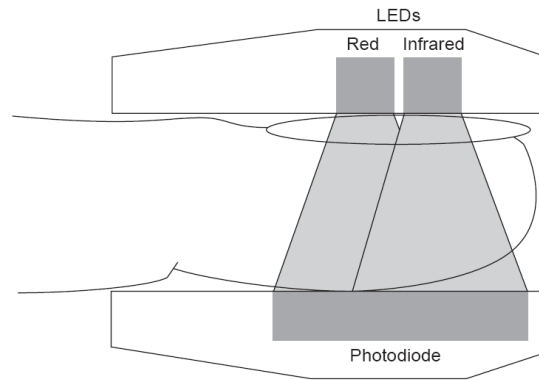


Figura 13 - Elementos do sensor

Fonte: Encyclopedia of medical devices and instrumentation.

Devido à pulsão arterial no local da medição, haverá diferentes níveis de absorção de luz devido à pele, os tecidos, sangue venoso e arterial. Entretanto, a cada batida, o coração contrai e há um acréscimo no volume de sangue arterial no local da medição, resultando em uma maior absorção de luz. Os sinais recebidos no Fotodiodo têm um formato de onda característico (com picos entre cada batimento cardíaco). Por isso, emprega-se o termo “oxímetro de pulso”, uma vez que os picos ocorrem a cada batimento cardíaco (WEBSTER, 2006, v.1, p.473).

3.1.4 Módulo de Oximetria

O Módulo de Oximetria, que recebe os dados do *probe* (sensor), foi adquirido da empresa SystemPartner, com sede em São Caetano do Sul/SP. Não é necessário conhecer detalhes sobre o funcionamento interno desse módulo, tampouco seus esquemas elétricos; somente será necessário saber qual seu protocolo de comunicação e a forma de alimentá-lo. A Figura 14 é uma fotografia do módulo usado no protótipo. Sua escolha foi realizada devido ao fato de ser o um dos pouco fabricados no país, haja vista que os equipamentos que os utilizam são de origem estrangeira.



Figura 14 – Módulo de Oximetria modelo SPOX-410

Os valores medidos pelo módulo são transmitidos no formato serial para o microcontrolador. A interface do protótipo com o computador é realizada através da Porta Serial, com um conector DB9, ilustrado na Figura 15. O cabo está ilustrado na Figura 16.

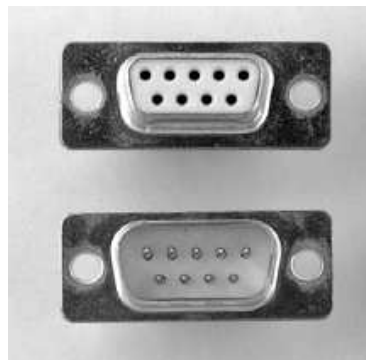


Figura 15 – Conector DB9

Fonte: *Serial Port Complete*.



Figura 16 – Conexão com o computador

Para fazer com que o microcontrolador se comunique com o módulo de oximetria, será necessário o uso de um resistor de 220Ω entre o pino RX e a sua saída. A função desse resistor é limitar a corrente, evitando que se sobrecarregue a saída digital do módulo.

De acordo com o Anexo A, documentação fornecida pelo fabricante do módulo, o formato de comunicação é digital no formato 8N1, numa velocidade de 9600bps. Tal comunicação se dá através de “palavras”, onde cada uma contém um *Start* bit, os bits de dados, um bit opcional de paridade, e um ou mais *Stop* bits. (AXELSON, 2000). No formato 8N1, as palavras são enviadas começando-se por um *Start* bit, seguido de oito bits de dados e um *Stop* bit. A letra “N”, do 8N1, significa que as palavras não contêm *Stop* bit. Esse tipo de transmissão é Assíncrona, ou seja, a velocidade de transmissão é independente das partes; não há um acordo determinando se um bit deverá ser enviado após o outro num certo período de tempo. Nesse caso, o próprio transmissor determina o período de tempo entre os bits. O formato de onda da transmissão esta representado na Figura 17.

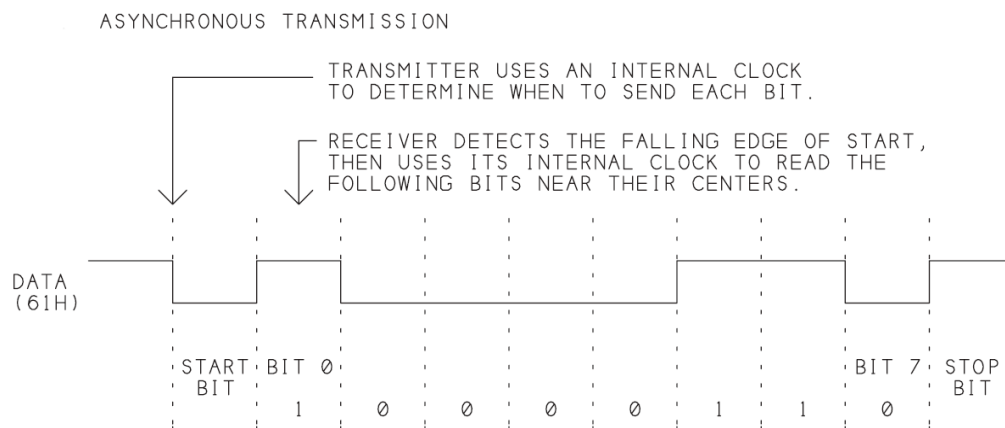


Figura 17 – Formato de uma transmissão assíncrona 8N1

Fonte: *Serial Port Complete*.

3.1.5 *Display* gráfico

O *display* gráfico utilizado no projeto do protótipo possui 15360 pixels na sua área visível, distribuídos em 240 colunas e 64 linhas. É baseado no controlador Toshiba modelo T6963C. Foi escolhido devido à sua fácil conexão e por possuir rotinas pré-escritas na linguagem C. A Figura 18 é uma fotografia do *display* utilizado no protótipo.



Figura 18 - *Display* Toshiba T6963C, 240x64 pixels

3.1.6 Teclado

O teclado é composto por teclas tácteis, com contatos “normalmente abertos”, ou seja, no seu estado normal (não-pressionados) apresentam resistência infinita entre seus pinos. Caso contrário, ao serem pressionados, seus contatos “se fecham” e conduzem corrente elétrica. A Figura 19 é uma fotografia do teclado construído especialmente para o protótipo.

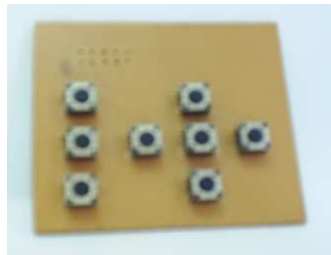


Figura 19 – Teclado do protótipo

Optou-se por arranjá-los em um formato de cruz, representando-se um direcional para facilitar a navegação nos *menus* e permitir acesso direto a funções pré-definidas através dos botões à esquerda. Cabem a essas funções silenciar ou ajustar os níveis de alarme, acessar o *menu* e ajustar o contraste do *display*. O Esquema Elétrico do teclado está na Figura 20.

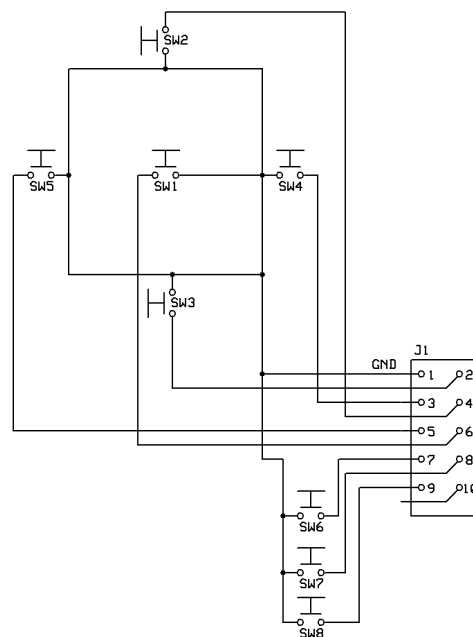


Figura 20 – Esquema Elétrico do Teclado

3.2 Projeto de Software

3.2.1 Módulo do Oxímetro

3.2.1.1 Diagrama de Componente

O diagrama de componentes, ilustrado na Figura 21, mostra os vários componentes de um *software* embarcado e suas dependências.

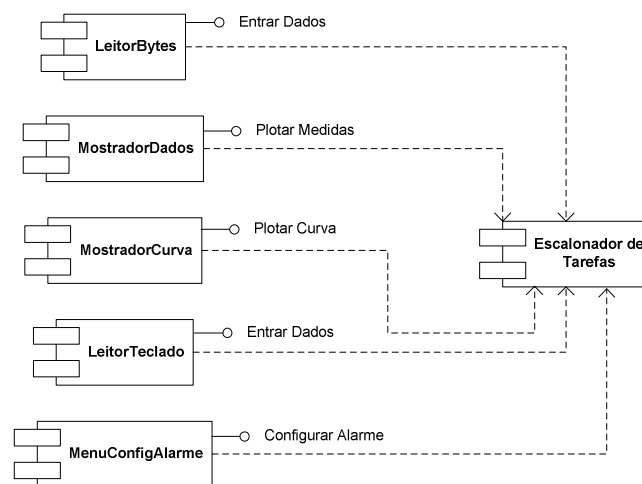


Figura 21 – Diagrama de Componente

Os componentes representados acima oferecem serviços específicos, onde cada um exerce sua própria funcionalidade, a saber:

- Escalonador de Tarefas: componente de *software* responsável por incrementar uma variável numérica em um intervalo de tempo definido. Baseando-se no valor dessa variável, este componente define qual será o próximo a ser executado;
- LeitorBytes: este componente efetua a comunicação serial com o módulo de oximetria utilizando-se das rotinas de comunicação própria linguagem. Fornece para os outros componentes os valores de SpO₂ (saturação de oxigênio) e BPM (batimentos cardíacos) em forma de variáveis numérica;
- MostradorCurva: componente responsável por plotar no *display* os valores de SpO₂ e BPM, no formato de uma curva, sempre da esquerda para à direita;
- MostradorDados: este componente exibe os valores numéricos no *display*, fazendo com que o usuário tenha uma fácil visualização dos parâmetros;

- LeitorTeclado: sua finalidade, quando executado, é ler o estado do teclado e desviar o fluxo de execução para a rotina desejada pelo usuário, representada pelos botões;
- MenuConfigAlarme: responsável por exibir no *display* um menu onde o usuário poderá selecionar quais serão os valores máximos e mínimos de SpO₂ e BPM nos quais o alarme irá atuar.

3.2.1.2 Diagrama de Atividade

Na Figura 22, logo abaixo, está representado o Diagrama de Atividades do *software* embarcado, englobando todos os estados possíveis, isto é, desde a inicialização do protótipo até o seu ciclo normal de funcionamento.

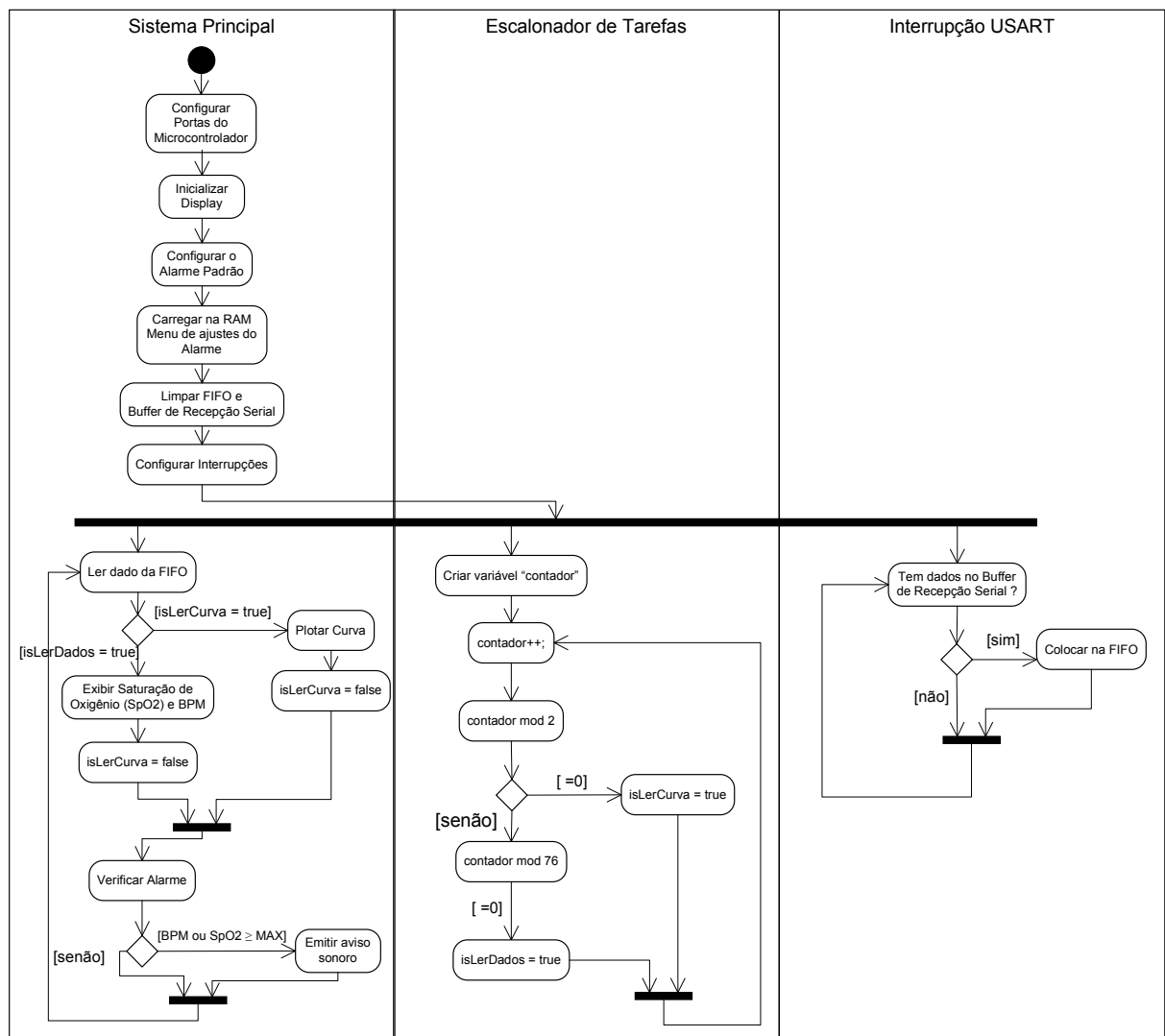


Figura 22 - Diagrama de Atividade do *Software* embarcado

Analisando a Figura 22, verifica-se que o *software* embarcado possui três processos básicos:

- Sistema Principal: responsável por inicializar todo *hardware*, como o *display* e o módulo serial do microcontrolador, exibir os dados das medições na tela e acionar o alarme caso os parâmetros estejam fora da faixa padrão. Segundo Holzhacker (2001), para um adulto, a frequência cardíaca deve-se situar entre 40 a 150 BPM, enquanto a saturação de oxigênio no sangue deve variar entre 85 a 100%.
- Escalonador de Tarefas: seu objetivo é dividir as tarefas - plotar a curva ou exibir os valores no *display*. Ele se faz necessário, pois o microcontrolador é *monotarefa*, ou seja, não é capaz de reproduzir dois processos concorrentes - em estado de paralelismo real (IOVINE, 2000). A implementação adotada aqui faz uso de um recurso de *hardware* chamado *interrupções* para executar processos simultâneos de uma forma limitada, independente do Sistema Principal. Logo, o que este escalonador faz é uma técnica conhecida como *time slicing*, dedicando uma fatia do tempo para cada procedimento do Sistema Principal. Ele utiliza-se das variáveis *booleanas* `isLerDados` e `isLerCurva` para sinalizar ao Sistema Principal qual atividade deverá ser realizada. Esse desvio de fluxo está representado logo abaixo do processo “*Ler Dados da FIFO*”. Após essa tarefa, os valores das variáveis `isLerDados` e `isLerCurva` são obtidos, e dependendo de seu conteúdo, o fluxo é desviado para o processo “*Plotar Curva*” ou “*Exibir Saturação (SpO2) e BPM*”.
- Interrupção USART: é um processo dedicado do microcontrolador, também independente do Sistema Principal, responsável por receber os dados do módulo de oximetria como mostrado anteriormente pela Figura 17.

3.2.2 Módulo para *Desktop*

3.2.2.1 Descrição do Sistema

O módulo para *Desktop* será constituído basicamente de um *software* e um banco de dados, onde o médico poderá acompanhar, através de um computador comum, todas as medições realizadas pelo oxímetro, em tempo real. O *software* deverá ser capaz de exibir na tela os valores medidos de SpO2 (saturação de oxigênio) e BPM (batimentos cardíacos), bem como a

curva pletismográfica. Além da visualização desses parâmetros, o sistema deverá oferecer também uma opção para “Salvar” os dados num arquivo, caso o médico decida gravar o estado do paciente. Cada arquivo deverá ser identificado pelo código e nome do paciente em questão, juntamente com a data e hora em que as medições foram realizadas. O paciente deverá ser previamente cadastrado. Outra funcionalidade importante que o *software* deverá desempenhar é a análise contínua dos dados, uma vez que o oxímetro poderá ser empregado em exames de polissonografia e outros de longa duração, por exemplo. Nesse caso, o volume de dados coletados será considerável, logo, o *software* deverá registrar em um gráfico todas as medições dos parâmetros, e ao fim do exame, exibir os valores médios, automatizando a tarefa de um enfermeiro ou outra pessoa dedicada no registro e acompanhamento desses dados.

Por fim, a emissão de relatórios também deverá fazer parte do sistema. O médico deverá ter acesso a uma área onde será possível filtrar os relatórios segundo o paciente, a data e hora de realização do exame. Esse relatório deverá ser impresso e conter todos os dados daquela medição selecionada. Outro ponto que merece atenção é que o sistema não deve permitir que se exclua um paciente nem um exame, uma vez que deve ser mantido um histórico dos mesmos.

3.2.2.2 Diagrama de Casos de Uso

O Diagrama de Casos de Uso na Figura 23 ilustra a interação entre o usuário e as diversas funcionalidades do sistema. Logo abaixo está descrito sucintamente cada caso de uso.

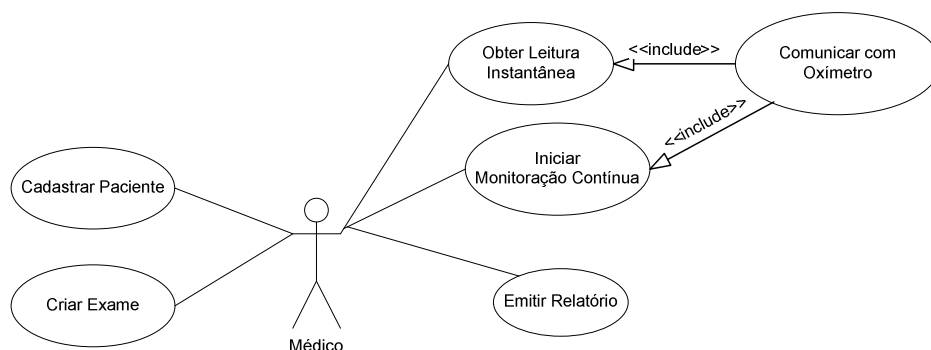


Figura 23 - Diagrama de Casos de Uso do *software* para Desktop

- Cadastrar Paciente: este caso de uso inicia-se quando o médico seleciona no *menu* a opção de “*Cadastrar Paciente*”. Logo em seguida, é exibido um formulário na tela onde deverão ser preenchidos alguns campos como o código e o nome do paciente. O médico terá acesso aos botões “*Salvar*” e “*Cancelar*”.
- Criar Exame: quando selecionado no *menu*, é novamente exibido ao médico um formulário onde ele poderá selecionar pacientes pré-cadastrados no sistema, retornando o código e o nome do mesmo. A data e hora atual do sistema serão registradas num campo desabilitado para edição, marcando assim o momento de criação daquele exame. Deverão ser exibidos na tela alguns botões, como “*Salvar Exame*”, “*Obter Leitura Instantânea*” e “*Iniciar Monitoração Contínua*”.
- Obter Leitura Instantânea: este caso de uso é iniciado quando o médico deseja obter uma leitura instantânea, ou seja, uma única amostra dos parâmetros (BPM e SpO₂) do paciente naquele momento. Os valores medidos deverão ser exibidos na forma numérica.
- Iniciar Monitoração Contínua: este caso de uso permite que o médico obtenha uma monitoração contínua dos parâmetros do paciente, e ao fim desta monitoração dado através de um botão “*Finalizar*”, deverá ser exibido na tela um gráfico contendo os valores em função do tempo, bem como os valores médios calculados e a duração do exame.
- Emitir Relatório: funcionalidade acessada através do *menu* “*Relatório*”, onde o médico terá acesso aos exames já realizados e as leituras e medições relacionadas aos mesmos. Esta tela deverá oferecer botões para “*Imprimir Relatório*” e “*Visualizar Impressão*”.
- Comunicar com Oxímetro: caso de uso responsável por estabelecer a comunicação com o *hardware* e fornecer dados às outras funcionalidades. Através dele que serão obtidas as medições do protótipo, logo, o médico não terá acesso a ele, mas sim os outros casos de uso ilustrados na relação de <<*include*>>.

3.2.2.3 Diagrama de Classes

A Figura 24 representa as classes do sistema para *Desktop* e seus relacionamentos.

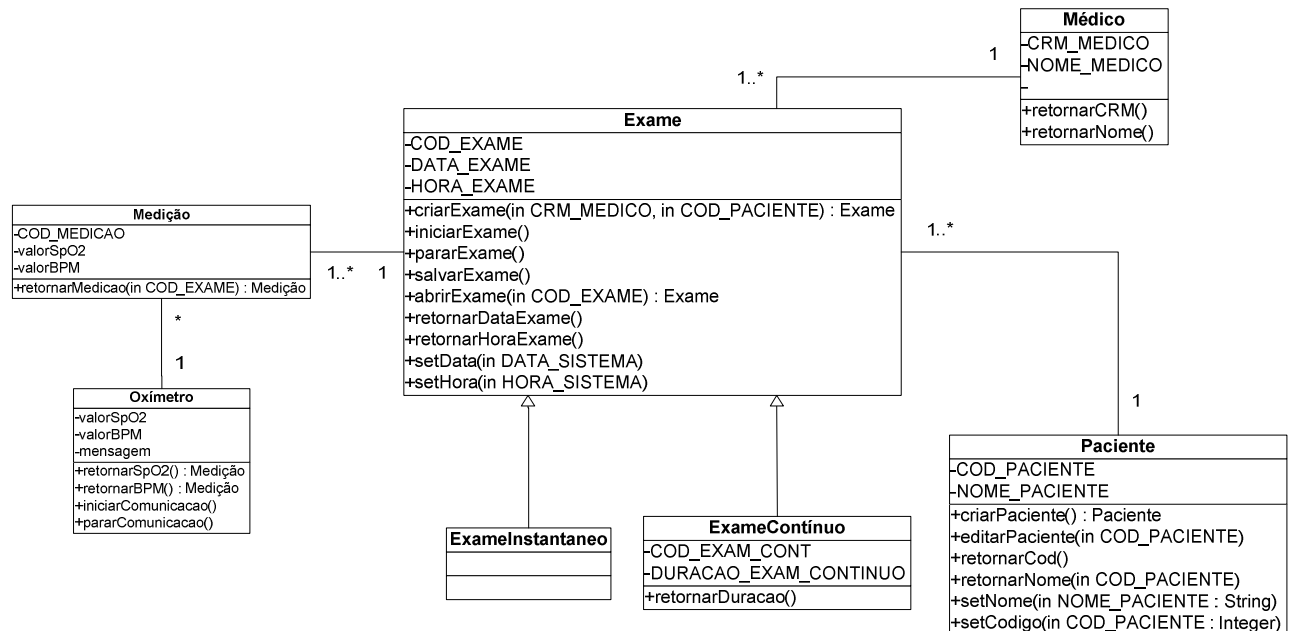


Figura 24 – Diagrama de Classes

Cada classe representada no diagrama acima representa uma entidade no sistema, a saber:

- **Exame:** definida como uma superclasse, ou uma classe-mãe. É dela que partirão os principais relacionamentos. Ela é responsável por persistir a data / hora da realização de cada exame, e possui métodos responsáveis por iniciar ou parar um novo exame, por exemplo. Nunca será criada uma instância dessa classe no sistema, isto é, todos os exames criados pelo usuário serão do tipo *ExameInstantâneo* ou *ExameContínuo*.
- **ExameInstantâneo:** produzirá instâncias especializadas da superclasse *Exame*, ou seja, objetos dessa classe irão herdar todos os atributos e métodos da classe *Exame*, entretanto não irão implementar nenhuma nova característica.
- **ExameContínuo:** outra classe que também irá herdar todas as características da classe *Exame*, porém possui um atributo e um método adicional. O método *retornarDuração* retorna o valor da variável *DURACAO_EXAM_CONTINUO*, que é responsável por armazenar a duração do exame.
- **Paciente:** classe que representa o paciente propriamente dito, que poderá ter um ou mais exames.

- Médico: é a entidade responsável por criar um novo exame, e poderá estar associado a vários exames.
- Medição: cada objeto (instância) do tipo Medição irá conter os valores adquiridos pelo oxímetro, onde cada exame poderá ter várias medições quando for do tipo ExameContínuo, e por outro lado, uma única medição quando for do tipo ExameInstantâneo.
- Oxímetro: é a entidade que representa a unidade física, o protótipo do oxímetro em si. O sistema poderá ter somente uma instância dessa classe durante sua execução, onde os valores adquiridos através da porta serial poderão ser obtidos através de chamadas aos métodos `retornarSpO2` e `retornarBPM`.

3.2.2.4 Diagrama de Seqüência

Nas figuras a seguir estão ilustrados todos os Diagramas de Seqüência para cada Caso de uso apontado na Figura 23. Em relação às Figuras 25 até a Figura 30, temos que os objetos do tipo `telaCadastroPaciente`, `telaCadastroExame` e assim por diante, são os Objetos de Fronteira.

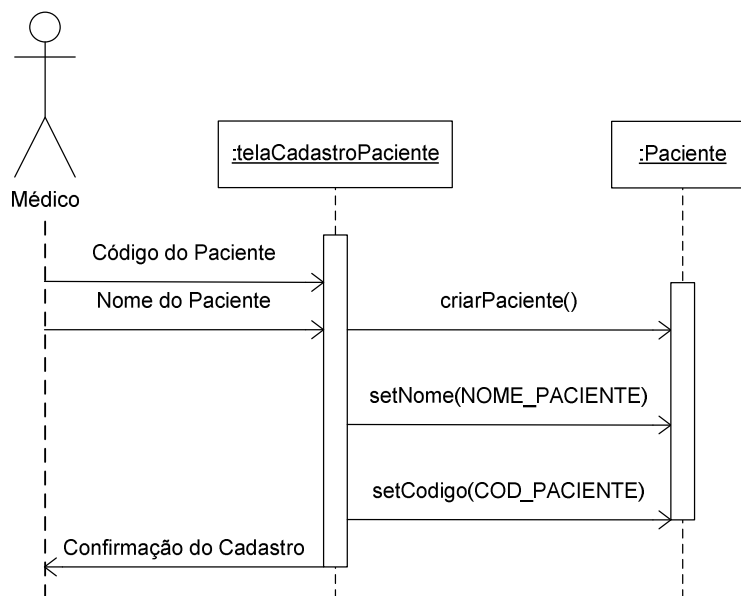


Figura 25 – Diagrama de Seqüência: Cadastrar Paciente

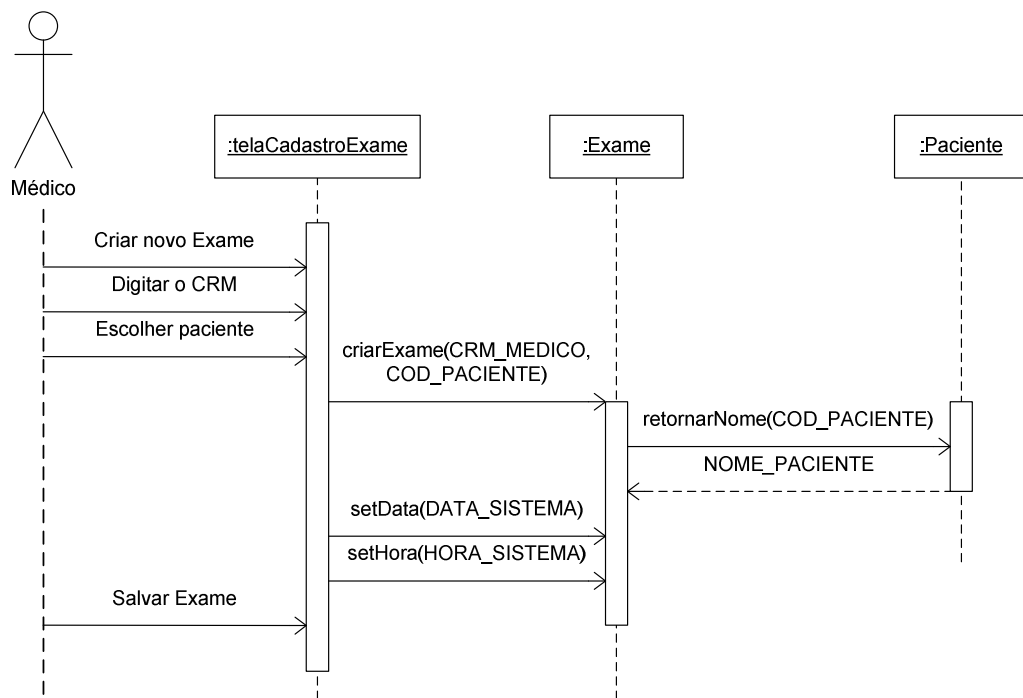


Figura 26 – Diagrama de Seqüência: *Criar Exame*

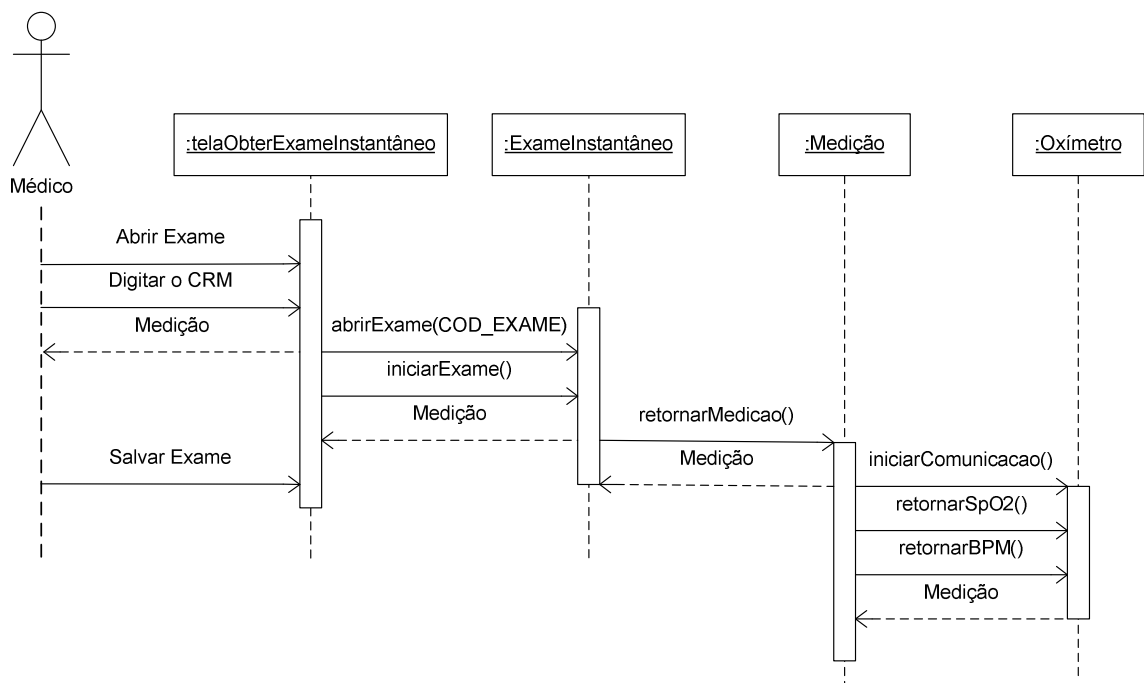


Figura 27 – Diagrama de Seqüência: *Obter Leitura Instantânea*

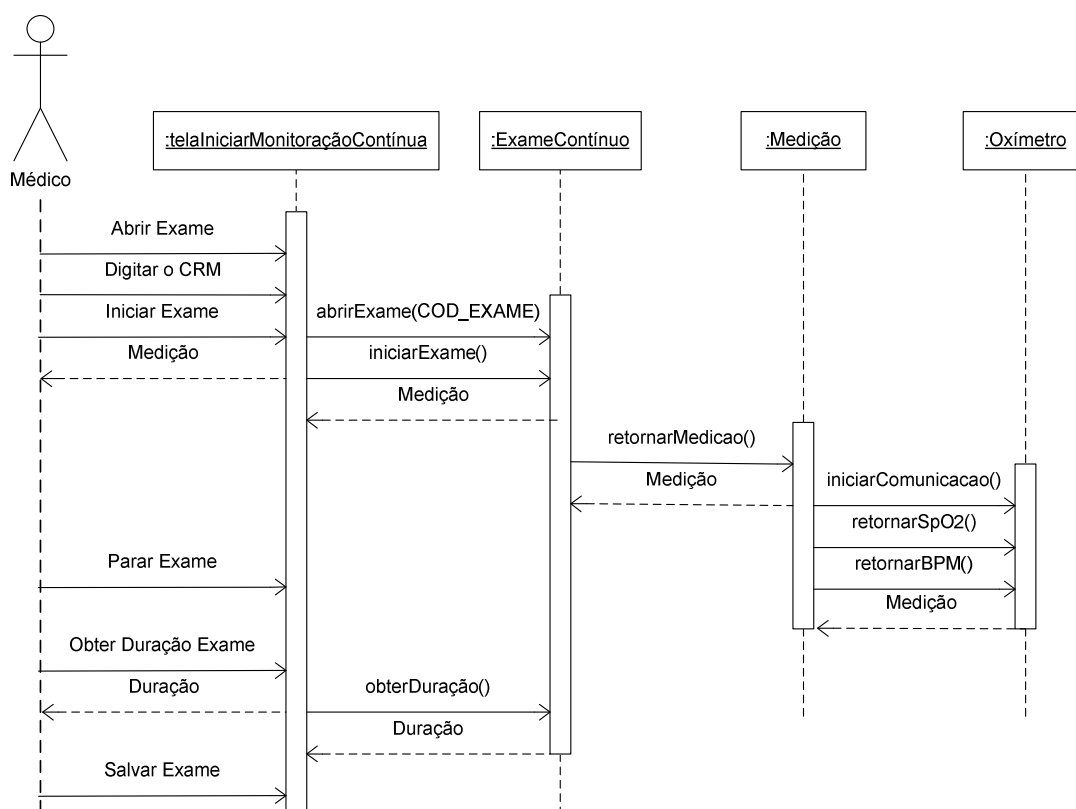


Figura 28 – Diagrama de Sequência: *Iniciar Monitoração Contínua*

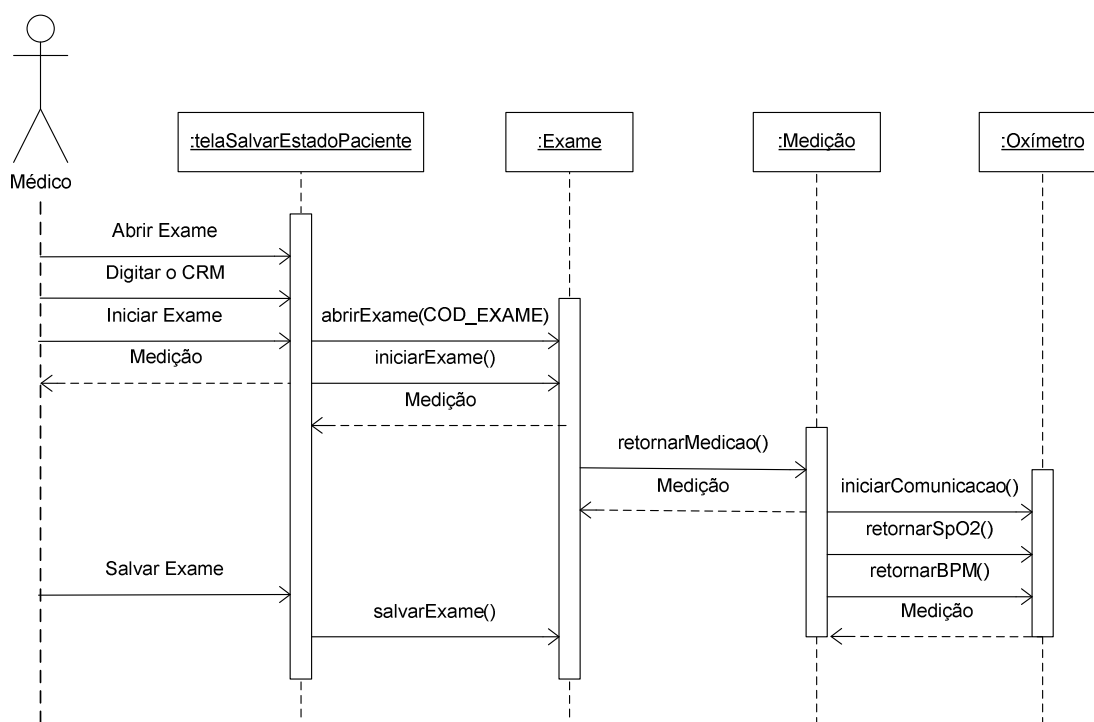


Figura 29 - Diagrama de Sequência: *Salvar Estado Paciente*

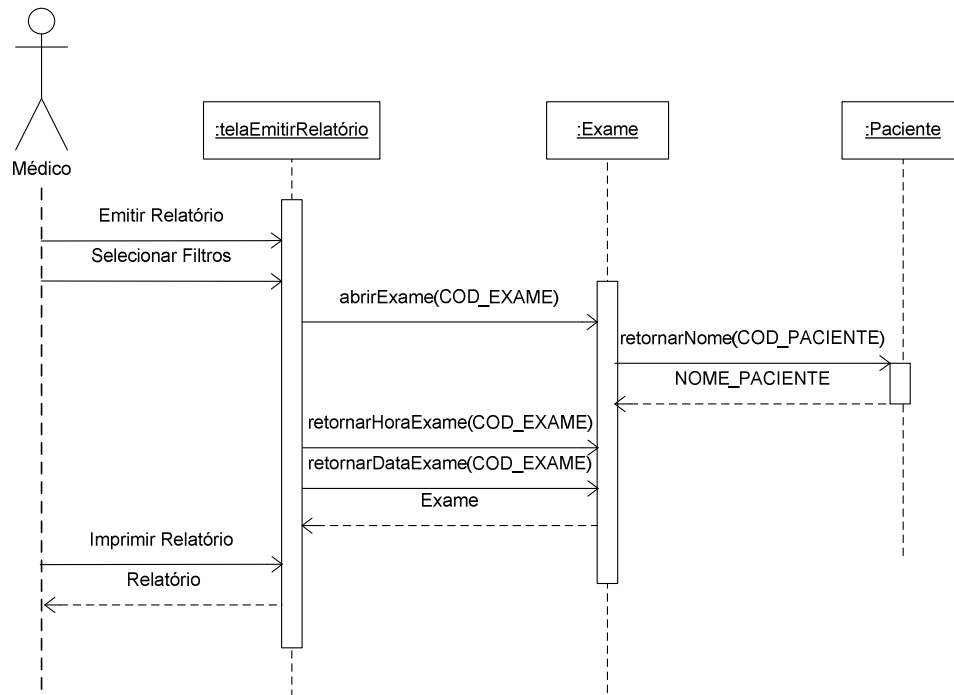


Figura 30 - Diagrama de Seqüência: *Emitir Relatório*

3.2.2.5 Telas do Sistema e Código Fonte

As telas do sistema para *desktop*, foram criadas em Java com o auxílio do *NetBeans IDE*, e estão disponíveis no Anexo B. A linguagem Java foi escolhida devido sua característica peculiar: a portabilidade. Dessa forma, o sistema poderá rodar em qualquer microcomputador, independente da plataforma e do sistema operacional, desde que tenha uma *JVM* (Maquina Virtual Java) instalado. A *JVM* Java está disponível para *download* no site do fabricante *Sun Microsystems* sem custo algum.

O Anexo C disponibiliza todo o código fonte do sistema embarcado, escrito na linguagem *C*. Ao contrário do que se pode encontrar em outros trabalhos envolvendo microcontroladores, essa linguagem foi escolhida em oposição ao *Assembler*, já que esta ultima é uma linguagem de baixo nível, o que aumentaria a complexidade na fase de implementação.

4 CONCLUSÃO

O principal objetivo desse trabalho, que é o desenvolvimento de um oxímetro de pulso capaz de conectar-se a um microcomputador, torna-se viável seguindo-se um desenvolvimento planejado, utilizando-se dos diagramas da UML para modelagem do *software*, e das demais ferramentas citadas no seu decorrer para a criação do *hardware*.

O desenvolvimento da fonte de alimentação, durante o projeto do *hardware*, mostrou-se ser uma etapa complexa, devido à necessidade de incorporar um carregador de baterias no circuito, uma vez que esse tipo de equipamento deverá ser operado também na falta de energia elétrica. Em contrapartida, a comunicação com o módulo de oximetria mostrou-se ser uma tarefa fácil, devido à documentação já existente fornecida pelo fabricante. O tamanho físico reduzido do circuito eletrônico também oferece uma possibilidade de se construir um equipamento facilmente portátil. Outro ponto muito importante é o baixo custo apresentado pelo equipamento, uma vez que os componentes utilizados na sua montagem são encontrados com facilidade no mercado nacional, e em geral, seu custo total é menor do que um equipamento de origem estrangeira, como pode ser visto na Tabela 1.

Tabela 1 : Custo dos principais componentes utilizados

Produto	Preço
PIC18F452 I/P	R\$ 25,99
Módulo Oximetria	R\$ 300,00
Display T6963C	R\$ 90,00
Componentes discretos	R\$ 100,00
CUSTO TOTAL	R\$ 515,99

Fonte: Farnell-Newark Componentes LTDA.

Comparando o custo total do protótipo com o preço final para o consumidor, ao adquirir um equipamento similar, sem conexão USB e conseqüentemente, sem um *software* tão completo como o aqui apresentado, é possível concluir que realmente há uma diferença considerável. Os dados encontram-se na Tabela 2 logo abaixo.

Tabela 2: Comparação entre o protótipo e modelos nacionais / importados

Produto	Tem USB?	Origem	Preço
Emai OXP-10	✗	Nacional	R\$ 3.195,00
Moriya M1000	✗	Nacional	R\$ 3.497,00
Nellcor N-600x	✗	Importado	R\$ 5.597,41
PROTÓTIPO	✓	Nacional	R\$ 515,99

Fonte: Cirúrgica Passos LTDA.

Em relação ao desenvolvimento, foi possível concluir que divisão das atividades do *software* do oxímetro, representadas no Diagrama de Atividades, também apresentou resultados muito satisfatórios. Foi possível implementar um sistema embarcado capaz exibir todos os parâmetros do paciente no *display*.

A principal vantagem desse protótipo, que é oferecer conectividade aos PCs de forma fácil e rápida, mostrou-se ser um diferencial em relação aos já existentes. Essa característica pôde ser apresentada através do *software* para o *desktop*.

Por fim, provou-se ser possível oferecer ao corpo médico uma melhor análise das informações coletadas pelo equipamento, auxiliando a tomada de decisões durante um exame, e até mesmo um melhor acompanhamento dos parâmetros do paciente em um ato cirúrgico.

5 REFERÊNCIAS

ARNOLD, Ken. **Embedded Controller Hardware Design**. Virginia-USA: LLH Technology Publishing, 2000.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9919**: Oxímetro de pulso para uso médico – Prescrições. Rio de Janeiro, 1997.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 14724**: Informação e documentação – Trabalhos acadêmicos – Apresentação. Rio de Janeiro, 2001. 6 p.

AXELSON, Jan. **Serial Port Complete**. Madison, WI: Lakeview Research, 2000.

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistema com UML**. Rio de Janeiro: Elsevier, 2002.

BOEHM, Barry. **A spiral model of software development and enhancement**. IEEE CS PRESS, v. 21, n. 5, p. 61-72, 1988.

EMADI, Ali; NASIRI, Abdolhosein; BEKIAROV, Stoyan B. **Uninterruptible Power Supply and Active Filters**. Chicago: CRC-Press, 2005.

FALBO, Ricardo. A Experiência na Definição de um Processo Padrão Baseado no Processo Unificado. **Anais do II Simpósio Internacional de Melhoria de Processo de Software, SIMPROS'200**. São Paulo: 2000.

FEARNLEY, Dr. SJ, **Pulse Oximetry**. Torquay, Inglaterra: Department of Anaesthetics, Torbay Hospital, 1995.

FERNANDES, Reinaldo; OJEDA, Renato; LUCATELLI, Marcos. Ensaio para Avaliação de Funcionalidade de Oxímetros de Pulso. **II Congresso Latinoamericano de Ingeniería Biomédica**. Cuba: 2001.

GIBILISCO, Stan. **The illustrated dictionary of Electronics**. New York: McGraw-Hill, 2001.

HILLS, Chris. Embedded MCU Debuggers – Third Edition. **Java C & C++ Spring Conference**. Oxford Union, UK: 1999.

HOLZHACKER, Albert. **Oxímetro de Pulso DX-2515: Manual de Operação**, 5 ed. Manaus: Dixtal, 2001.


IOVINE, John. **PIC Microcontroller Project Book**. New York: McGraw-Hill, 2000.

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **The unified software development process**. USA: Addison-Wesley, 1998, 463 p.

- KNOBEL, Elias. **Condutas no paciente Grave**. 2 ed. São Paulo: Atheneu, 1998.
- MARCONDES, Giancarlo et al . Transporte de pacientes sem oxigenoterapia para a sala de recuperação pós-anestésica: repercussões na saturação de oxigênio e fatores de risco associados à hipoxemia. **Rev. Bras. Anesthesiol.** , Campinas, v. 56, n. 4, 2006 .
- MARWEDEL, Peter. **Embedded System Design**. Kluwer Academic Publishers. 1 ed. Dortmund: Kluwer Academic Publisher, 2003.
- MICROCHIP. “**PIC18FXX2 Datasheet**”. Microchip Technology Inc, 2002.
- MIKROELETRONIKA. “**mikroC User’s Manual**”. MikroEletronika Books, 2007.
- MIMS, Forrest M. III, **Engineer’s Mini-Notebook - Op Amp IC Circuits**. Radio Shack, 1985.
- MIMS, Forrest M. III, **Getting Started in Electronics**. Radio Shack, 1986.
- NATIONAL. “**Datasheet of LM117/LM217/LM317 3-Terminal Adjustable Regulator**” National Semiconductor Corporation Linear Databook, 1982, pp. 1-23 a 1-30.
- NUNES, Wilma Aparecida; TERZI, Renato Giuseppe Giovanni. Oximetria de pulso na avaliação do transporte de oxigênio em pacientes críticos. **Rev. Latino-Am. Enfermagem** , Ribeirão Preto, v. 7, n. 2, 1999.
- PRESSMAN, Roger. **Engenharia de Software**. São Paulo: Makron Books, 1995.
- PRESSMAN, Roger. **Engenharia de Software**, 6^a edição. São Paulo: McGraw-Hill, 2006.
- SANTANDER, Victor; VASCONCELOS, Alexandre. Mapeando o Processo Unificado em Relação ao CMM – Nível 2. **XI CITS - Qualidade de Software**. Curitiba: 2000, pp. 120-137.
- SCOTT, Kendall. **O processo Unificado**. Porto Alegre: Bookman, 2003.
- SEDRA, A.S., SMITH, K.C. **Microeletrônica**, 4^a Edição. São Paulo: Makron Books, 2004.
- VELLOSO, Alexandre. **Desenho Básico – Eletrônica**. Rio de Janeiro: Escola Técnica Estadual República, 2006.
- VENTURA, Celestina, OLIVEIRA, Ana Sofia, DIAS, Rita *et al.* Papel da oximetria noturna no rastreamento da síndrome de apneia-hipopneia obstrutiva do sono. **Rev Port Pneumol**, jul. 2007, vol.13, no.4, p.525-551. ISSN 0873-2159.
- WEBSTER, John G. **Encyclopedia of medical devices and instrumentation**, vol.1, 2 ed. New York: John Wiley & Sons, 2006.
- YORDON, Edward. **Análise estruturada moderna**. Tradução: Dalton Conde de Alencar. Rio de Janeiro: Campus, 1990.

ANEXO A – Documentação do Módulo de Oximetria SPOX-410

Documentação incluída ao Módulo de Oximetria, fabricado por SystemPartner. Traz detalhes sobre o protocolo de comunicação e alimentação do módulo.

	SYSTEM PARTNER – ENGENHARIA DA QUALIDADE	MS-PCM-002-0410
	MANUAL DE SERVIÇO	REV 7
	MODULO DE OXIMETRIA PCM-0020410 (SPOX410)	Página 15 de 19

7) PRODUTO OEM0020007:

ALIMENTAÇÃO: 5V (pino 1 de CN3)

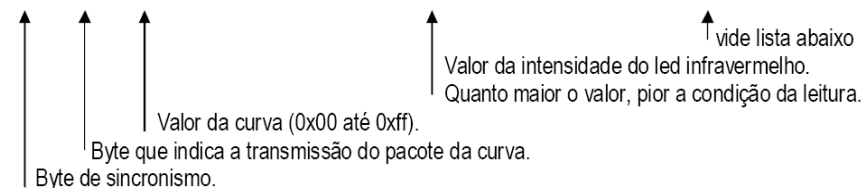
PROTOCOLO DE COMUNICAÇÃO: Baud Rate: 9600 8N1

7.1) Envio das informações quando há pulso e leitura de saturação:

7.1.1) Modo curva:

A curva pletismográfica é enviada a uma taxa de 120 vezes por segundo a 9600 bps.

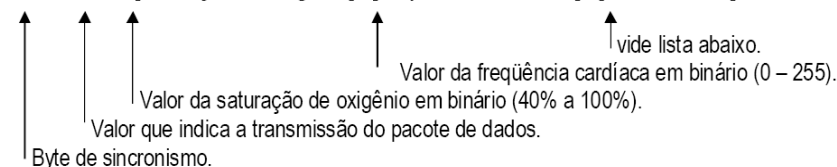
0x10 , 0x7E , [byte da curva pletismográfica], [byte do led infravermelho], [MENSAGENS]



7.1.2) Modo dados:

Quando há leitura de saturação e frequência cardíaca é enviado o seguinte pacote a cada pulso detectado:

0x10 , 0x21 , [saturação de oxigênio] , [frequência cardíaca] , [MENSAGENS]



7.2) Envio das informações quando não há pulso nem leitura da saturação:

7.2.1) Modo curva: Enviado na mesma taxa (120 por segundo) com as seguintes informações

0x10, 0x7E , 0x80, [byte do led infravermelho], [MENSAGENS]

7.2.2) Modo dados: Enviado uma vez a cada 6 segundos quando há a perda do pulso:

0x10, 0x21, 0xff, 0xff, [MENSAGENS]

7.3) MENSAGENS:

0x00	=	Pulso normal.
0x01	=	Sem pulso.
0x02	=	Procurando pulso.
0x04	=	Instalar sensor.
0x08	=	Sensor desconectado.
0x10	=	Movimento detectado.
0x20	=	Reservado.
0x40	=	Baixa perfusão.
0x80	=	Reservado.

ANEXO B – Telas do *software* para desktop

The screenshot shows the 'Exames de Oximetria' application window. The menu bar includes 'Cadastro', 'Exame', 'Relatórios', 'Configurações', and 'Ajuda'. The 'Cadastro' menu is open, showing 'Paciente' (with sub-items 'Novo Paciente' and 'Buscar Paciente') and 'Sair'. A 'Cadastro de Pacientes' dialog box is displayed in the center. It contains the following fields: 'Nome do Paciente' (João da Silva), 'Número do Convênio' (00912-AB01), 'Código' (0021), 'Data de Nascimento' (01/05/1965), 'Dt. Cadastro' (02/04/2008), 'Altura' (1,85m), 'Peso' (86 Kg), and 'Tipo Sanguíneo' (B+). There is an 'Obs.' text area and 'Salvar' and 'Cancelar' buttons at the bottom.

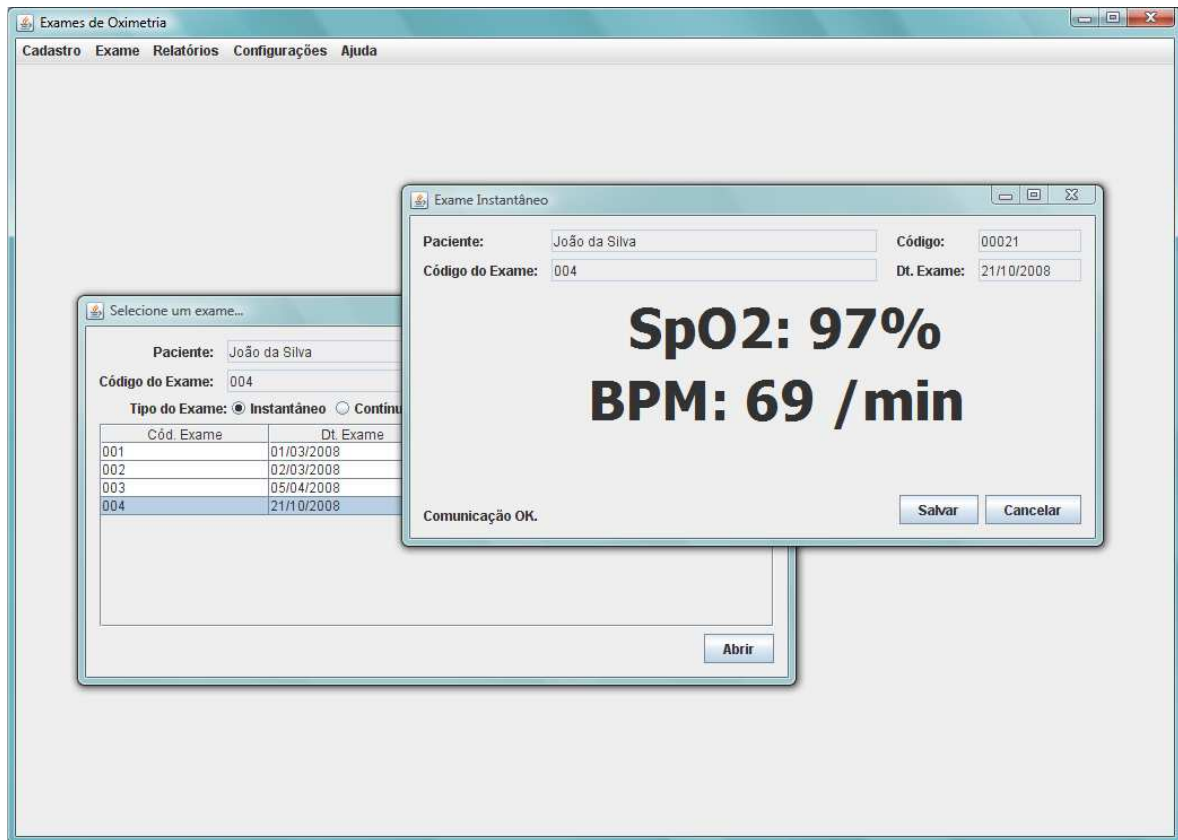
Nome do Paciente:	João da Silva		
Número do Convênio:	00912-AB01	Código:	0021
Data de Nascimento:	01/05/1965	Dt. Cadastro:	02/04/2008
Altura:	1,85m	Peso:	86 Kg
Tipo Sanguíneo:	B+		
Obs.:			

Cadastro de Paciente

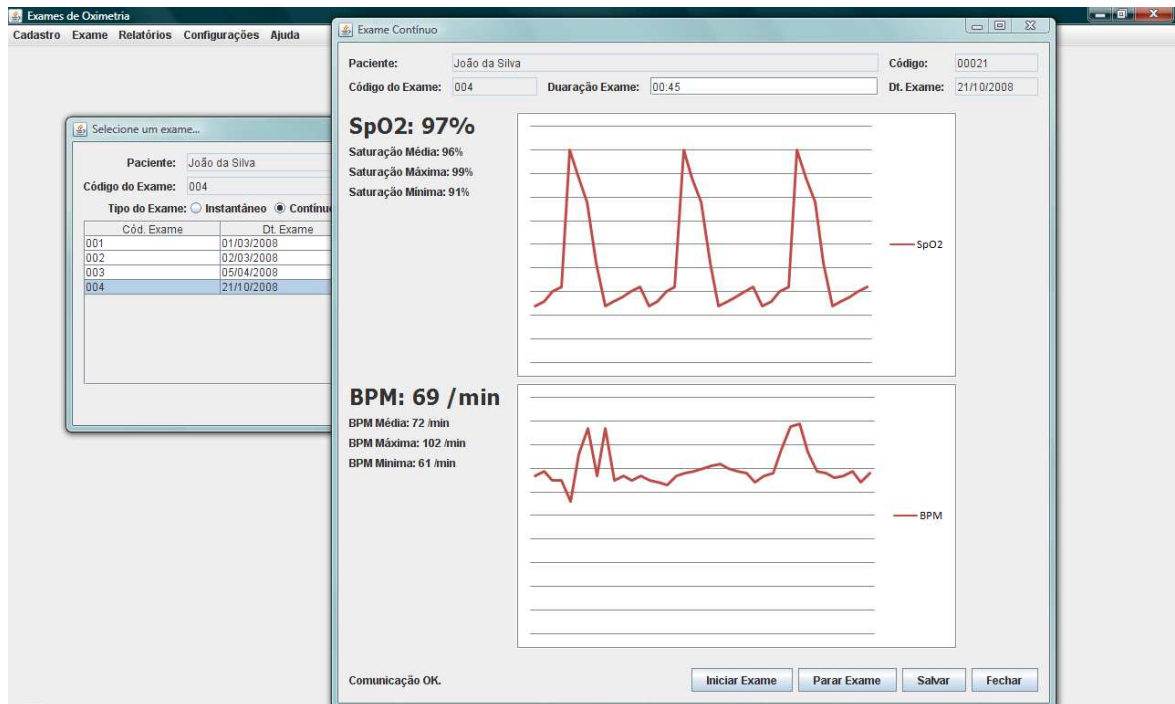
The screenshot shows the 'Exames de Oximetria' application window. The menu bar includes 'Cadastro', 'Exame', 'Relatórios', 'Configurações', and 'Ajuda'. The 'Exame' menu is open, showing 'Novo Exame' and 'Buscar Exame'. A 'Cadastro de Exames' dialog box is displayed in the center. It contains the following fields: 'Paciente' (João da Silva) with a 'Buscar...' button, 'Código do Exame' (004), 'Dt. Exame' (21/10/2008), and 'Tipo do Exame' with radio buttons for 'Instantâneo' and 'Contínuo' (selected). There are 'Salvar' and 'Cancelar' buttons at the bottom.

Paciente:	João da Silva	Buscar...	
Código do Exame:	004	Dt. Exame:	21/10/2008
Tipo do Exame:	<input type="radio"/> Instantâneo <input checked="" type="radio"/> Contínuo		

Cadastro de Exames



Realizando um exame instantâneo



Realizando um exame contínuo

Exames de Oximetria

Cadastro Exame Relatórios Configurações Ajuda

Relatório

Anterior Próxima

Salvar Fechar

Selecione um relatório...

Paciente: João da Silva

Código do Exame: 004

Tipo do Exame: ☒ Instantâneo

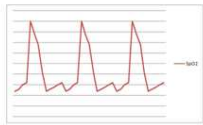
Cód. Exame	DT
001	01/03/2008
002	02/03/2008
003	05/04/2008
004	21/10/2008

Hospital da Cidade
Av. da Saúde, 1122 - Uberlândia - MG
(34) 3822-2111

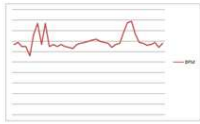
Paciente: 0021-João da Silva
Convênio: 00912-AB01
Data Nasc: 01/05/1965
Altura: 1,85m
Peso: 86 kg

Exame Contínuo de Oximetria

Código: 004
Dt. Realização: 21/10/2008
Médico Responsável: Dr. Edson Junior
Início do Exame: 14:30 PM
Fim do Exame: 22:50 PM
Duração: 8h 20m



Saturação Média:	96%	-
Saturação Máxima:	99%	17:31:55 PM
Saturação Mínima:	91%	21:05:11 PM



BPM Média:	72/min	-
BPM Máxima:	102/min	16:24:21 PM
BPM Mínima:	62/min	22:21:59 PM

Dr. Edson Junior
CRM - 11223344

Relatório de um exame contínuo

ANEXO C – Código Fonte do Sistema Embarcado

```
//*****
//      Oxímetro de Pulso
//      Baseado no módulo de oximetria SystemPartner:
//          SPOX-410
//          PCM-0020410
//          OEM-0020007
//
//      Hardware:
//          PIC18F452 I/P @ 40 Mhz (10 Mhz xtal + PLL )
//          5 Vdc PSU
//          T6963C 240x64 GLCD
//
//      Fernando Vinícius G. Magro 2007/2008
//*****

// SAIDAS
#define ALARM_OUT      PORTE.F2 // pino de saída para alarme disparando
#define BEEP_OUT       PORTE.F1 // saída para o Buzzer – bepp BPM
#define REM_OUT        PORTE.F0 // Sinal para o clock do 4017, desligando a fonte
#define SBUFF_SIZE     128 // potência de 2 para uma indexação rápida
#define BUFOVFL        1

unsigned int contador,contadorAlarmeTemp;
unsigned float tempoEstouro;
unsigned short ByteLido, ByteSync, isLerCurva, isLerDados;
unsigned short ByteCurva, ByteLed, ByteMSG, ByteSpO2, ByteBPM, ultimoValorCurva;
char StrCurva[4], StrLed[4], StrSpO2[4], StrBPM[4], StrMSG[4];
char StrChar[4];
char      msg[7]; // DEBUG
unsigned short xi, yf;
unsigned int tempoF2Press,tempoF3Press,tempoOKPress;
unsigned short dispararAlarme, isAlarmeTemp, isAlarmeGeral, isAlarmeAuto,
               isBeep, tempoAlarmeTemp, isPiscarBPM, isPiscarSpO2;

unsigned short valorContraste, isContrasteAlterado;

#include "include/T6963C.h"
#include "include/graficos.c"
#include "include/rotinasBuffer.c"

void configPortas();
void configInterrupt();
void configPWM();
void incrementarContraste();
void decrementarContraste();
unsigned short lerContraste(); // retorna o armazenado na posição 4 EEPROM
void gravarContrasteEEPROM(unsigned short valorContraste); // armazenado na posição 0 EEPROM
void verificarAlarmes(unsigned short SpO2,unsigned short Bpm); // Retorna 1 se alarme deve
disparar, 0 caso contrário.
void inverterPanel();
void alternarAlarmeTemp();
void alternarAlarmeGeral();
void alternarBeepBPM();
void alternarAlarmeAuto();
void configAlarmePadrao();
void limparCurva();

#include "include/menu.c"

void interrupt() {
    unsigned char RxStatus ;

    // interrupção USART
    if ( PIR1.RCIF ) {
        as.ccbuf[as.endbuf] =RCREG ;
        as.endbuf++ ;
    }
}
```



```

    as.endbuf &= (SBUFF_SIZE-1) ;

    if ( ((as.endbuf+1)&(SBUFF_SIZE-1)) == as.startbuf )
    {
        // Buffer Overflow!...
        as.startbuf = 0 ;
        as.endbuf    = 0 ;
    }
    PIR1.RCIF = 0 ; // limpa a interrupção
}

//interrupção TMRO
if(INTCON.TOIE == 1 && INTCN.TOIF ==1) {
    contador++;

    if (contador%2==0) {
        isLerCurva=1;
    }

    // incrementa contador de tempo do alarme temporarioo
    if (isAlarmeTemp) {
        contadorAlarmeTemp++;
    } else {
        contadorAlarmeTemp=0;
    }

    if (contador%76==0) { // aprox. 1 segundo - atualização dos dados
        // e tempo "piscar" do alarme visual
        isLerDados=1;

        if ( isAlarmeGeral && (!isAlarmeTemp) ) {
            if (dispararAlarme) { // dispara o alarme
                ALARM_OUT=~ALARM_OUT;
            } else {
                ALARM_OUT=0;
            }
        } else {
            ALARM_OUT=0; // desliga o alarme
        }
    }

    INTCN.TOIE = 1;
    INTCN.TOIF =0;
}

// interrupção externa RB0
if (INTCON.INTOIF == 1) {
    isMenu=1;
    INTCN.INTOIF=0;
}

BEEP_OUT=0;
} //interrupt

void main() {

    xi = 61; // posição x0 de inicio do pletismograma

    configPortas();

    // inicializa PWM para o Contraste do LCD
    configPWM();
    valorContraste = lerContraste(); // Lê e aplica o ultimo valor do Contraste
    PWM_Change_Duty(valorContraste);

    T6963C_init(240,64,6,&PORTD,&PORTA,5,1,2,3);

    panel=0;

```

```

T6963C_graphics(1) ;
T6963C_text(1) ;
T6963C_panelFill(0);
T6963C_cursor(0);

//Desenhando a tela principal no panel 0
T6963C_line(60,0,60,64,T6963C_WHITE);
T6963C_write_text("SP02 %",0,0,T6963C_ROM_MODE_XOR);
T6963C_write_text("BPM",0,4,T6963C_ROM_MODE_XOR);
T6963C_box(60,64,240,55,T6963C_WHITE);

// exbindo icones
// offset de 3 pixels entre os ícones
T6963C_sprite(228,0,icone_beep_on,12,12); // beep ligado por padrão
T6963C_sprite(216,0,icone_sino,12,12); // alarme ligado no valor padrão

//Configurações de alarme e valores padrão
configAlarmePadrao();
tempoAlarmeTemp=120; //tempo em que o alarme permanece desabilitado (seg)
dispararAlarme=0;
isAlarmeTemp=0;
isAlarmeGeral=1;
isAlarmeAuto=0;
isBeep=1;

//Carregando o menu na RAM - panel 1
T6963C_setGrPanel(1);
T6963C_setTxtPanel(1);
T6963C_panelFill(0);

//desenhando a janela
T6963C_rectangle(5, 5, 230, 58, T6963C_WHITE);
T6963C_box(10, 59, 235, 63, T6963C_WHITE);
T6963C_box(231, 10, 235, 58, T6963C_WHITE);
T6963C_write_text("Ajustar alarmes",1,1, T6963C_ROM_MODE_XOR);
T6963C_line(5,17,230,17, T6963C_WHITE);

// imprimindo os rótulos
T6963C_write_text("SP02",9,3, T6963C_ROM_MODE_XOR);
T6963C_write_text("Maximo:",8,5, T6963C_ROM_MODE_XOR);
T6963C_write_text("Minimo:",8,6, T6963C_ROM_MODE_XOR);
T6963C_write_text("BPM",26,3, T6963C_ROM_MODE_XOR);
T6963C_write_text("Maximo:",24,5, T6963C_ROM_MODE_XOR);
T6963C_write_text("Minimo:",24,6, T6963C_ROM_MODE_XOR);

// volta a escrever no panel 0
T6963C_setGrPanel(0);
T6963C_setTxtPanel(0);

// limpa o buffer
as.startbuf = 0 ;
as.endbuf = 0 ;

configInterrupt();
Usart_Init( 9600 );

isMenu = 0;

// pré-inicializa as variáveis, evitando primeira leitura incorreta
ByteLido=0;
ByteSync=0;
ByteCurva=0x80;
ByteMSG=0;
ByteSpO2=0xff;
ByteBPM=0xff;
tempoF2Press=0;
tempoF3Press=0;
tempoOKPress=0;

```

```

ultimoValorCurva=34;
contadorAlarmeTemp=0; // contador que armazena o tempo do alarme temporário
isContrasteAlterado=0; // nenhuma alteração inicial do contraste

while (1) {

    ByteSync=lerByte();
    if(ByteSync==0x10) {
        ByteLido=lerByte();
        if(ByteLido==0x7E){
            ByteCurva=lerByte();
            ByteLed=lerByte();
            ByteMSG=lerByte();
        }
        if(ByteLido==0x21){
            BEEP_OUT=(ByteMSG < 0x02) ? 1 : 0; // beep somente se a MSG > 0x02
            ByteSp02=lerByte();
            ByteBPM=lerByte();
            BEEP_OUT=0;
        }
    }

    // faz as conversões necessárias
    ByteToStr(ByteSp02,StrSp02);
    ByteToStr(ByteBPM,StrBPM);
    // imprime a mensagem
    T6963C_write_text( decodificaMSG(ByteMSG) ,11,7,T6963C_ROM_MODE_XOR);

    // BOTAO F2
    if (BT_F2==0) {
        tempoF2Press++;
        if (tempoF2Press==300) { // quando segurar pressionado por muito tempo
        }
    }
    else {
        if ( ((tempoF2Press!=0) && (tempoF2Press<150)) && isAlarmeGeral ) { // somente
alterna alarmTemp se AlarmeGeral on
            alternarAlarmeTemp(); //quando um simples clique
        }
        tempoF2Press=0;
    }

    // BOTAO F3
    if (BT_F3==0) {
        tempoF3Press++;
        // verifica se as 2 teclas estão sendo pressionadas
        // para o ajuste do Contraste
        if (BT_UP==0) {
            incrementarContraste();
            isContrasteAlterado=1;
        } else if (BT_DOWN==0) {
            decrementarContraste();
            isContrasteAlterado=1;
        }
    }
    else {
        if ( (tempoF3Press!=0) && ( ! isContrasteAlterado) ) { // 'release' do botão
            alternarBeepBPM();
        } else if (isContrasteAlterado) {
            gravarContrasteEEPROM(valorContraste);
        }
        tempoF3Press=0;
        isContrasteAlterado=0;
    }

    // BOTAO OK_AUTO
    if (BT_OK_AUTO==0) {
        tempoOKPress++;
        if (tempoOKPress == 1) { // simples clique

```

```

        alternarAlarmeAuto();
    }
} else {
    tempoOKPress=0;
}

if(isLerCurva) {

    isLerCurva=0;
    yf = 18 + ((0.13 * ByteCurva));
    T6963C_line(xi,51,xi,18,T6963C_BLACK); //faz a limpeza
    T6963C_line(xi,ultimoValorCurva,xi,yf,T6963C_WHITE);
    ultimoValorCurva=yf;
    if(xi == 240) {
        xi = 61;
    } else {
        xi++;
    }
}

if(isLerDados) {
    isLerDados=0;
    if (ByteSpO2==0xFF && ByteBPM==0xFF) { // SEM PULSO NORMAL
        dispararAlarme=0; // desliga o alarme caso BPM e SPO2 ==255
        if (isAlarmeAuto) { // desliga o alarme automático ao retirar sensor
            alternarAlarmeAuto();
            isAlarmeAuto=0;
        }
        //limpa a area, caso não haja pulso
        T6963C_sprite(0, 9, vazio , 20, 20) ;
        T6963C_sprite(21, 9, vazio , 20, 20) ;
        T6963C_sprite(41, 9, vazio , 20, 20) ;
        T6963C_sprite(0, 41, vazio , 20, 20) ;
        T6963C_sprite(21, 41, vazio , 20, 20) ;
        T6963C_sprite(41, 41, vazio , 20, 20) ;
    } else {
        exibeNumero(0,9,StrSpO2,isPiscarSpO2);
        exibeNumero(0,41,StrBPM,isPiscarBPM);
        verificarAlarmes(ByteSpO2,ByteBPM);
    }
}

if ( isMenu ) {
    inverterPanel();
    if (panel == 1) { // saindo do menu..
        exhibirMenu();
        inverterPanel();
        xi=61; // redesenha a curva, do ponto inicial, invalidando a já existente
    }
    isMenu = 0;
}

// Verifica se o tempo do Alarme Temporario já estourou..
if (isAlarmeTemp) {
    if (contadorAlarmeTemp > floor(tempoAlarmeTemp/tempoEstouro)) {
        alternarAlarmeTemp();
    }
}

} //while
} // main

void configPortas() {
//    ADCON1 = 0x07; // PORTA digital
    ADCON1 = 0b00001110;
    TRISA = 0b10000000; // PORTA saída - RA0: entrada an. monitor bat
    PORTA = 0;
    TRISB = 1; // PORTB entrada
    TRISC = 1; // necessário para lançar as interrupções USART ao iniciar.
}

```

```

    TRISE = 0; // saida Beeps
    PORTE = 0; // Estado inicial PORTE
}

void configInterrupt() {
//    TOCON = 0x82; // 0b10000010 16bit PS 1:8, estouro: = 0.0524 seg
    TOCON = 0b10000000 ; // 0b10000000 16bit PS 1:2, estouro: = 0.0131 seg
    tempoEstouro = 0.0131;
    PIE1.RCIE = 1 ;
    INTCON.GIE = 1 ;
    INTCON.PEIE = 1 ;
    INTCON.TMR0IE = 1 ; // interrupção TMR0
    INTCON.INT0IE = 1 ; // interrupção RBO
}

void configPWM() {
    TRISC.F2=0;
    PWM_Init(20000); // frequencia
    PWM_Start();
    valorContraste=0;
    PWM_Change_Duty(valorContraste);
}

void incrementarContraste() {
    if (valorContraste < 255) {
        valorContraste++;
        PWM_Change_Duty(valorContraste);
//        gravarContrasteEEPROM(valorContraste);
    }
}

void decrementarContraste() {
    if (valorContraste > 140) {
        valorContraste--;
        PWM_Change_Duty(valorContraste);
//        gravarContrasteEEPROM(valorContraste);
    }
}

unsigned short lerContraste() {
    return Eeprom_Read(4);
}

void gravarContrasteEEPROM(unsigned short valorContraste) {
    Eeprom_Write(4,valorContraste);
    Delay_ms(20);
}

void inverterPanel() {
    panel++; // inverte o panel
    panel &= 1;
    T6963C_setGrPanel(panel);
    T6963C_setTxtPanel(panel);
    T6963C_displayGrPanel(panel);
    T6963C_displayTxtPanel(panel);
}

void verificarAlarmes(unsigned short SpO2,unsigned short Bpm) { // retorna 0 se OK, 1 BPM,
    unsigned short BPMdisparado, SpO2disparado;

    if ( (Bpm > valorAlarme[2]) || (Bpm < valorAlarme[3])) {
        BPMdisparado=1;
        if (isAlarmeGeral) { // pisca somente se o alarmeGeral estiver ativo
            isPiscarBPM=~isPiscarBPM;
        } else {
            isPiscarBPM=0;
        }
    } else {
        BPMdisparado=0;
    }
}

```

```

        isPiscarBPM=0;
    }

    if ( (Sp02 > valorAlarme[0]) || (Sp02 < valorAlarme[1]) ) {
        Sp02disparado=1;
        if (isAlarmeGeral) {
            isPiscarSp02=~isPiscarSp02;
        } else {
            isPiscarSp02=0;
        }
    } else {
        Sp02disparado=0;
        isPiscarSp02=0;
    }

    if ( BPMdisparado || Sp02disparado ) {
        dispararAlarme=1;
    } else {
        dispararAlarme=0;
        isPiscarSp02=0;
        isPiscarBPM=0;
    }
}

void alternarAlarmeGeral() {
    if(isAlarmeGeral) {
        T6963C_sprite(216,0,icone_vazio,12,12);
        isAlarmeGeral=0;
    } else {
        T6963C_sprite(216,0,icone_sino,12,12);
        isAlarmeGeral=1;
    }
}

void alternarAlarmeTemp(){
    if(isAlarmeTemp) {
        T6963C_sprite(204,0,icone_vazio,12,12);
        isAlarmeTemp=0;
    } else {
        T6963C_sprite(204,0,icone_relogio,12,12);
        isAlarmeTemp=1;
    }
}

void alternarBeepBPM() {
    if(isBeep) {
        T6963C_sprite(228,0,icone_beep_off,12,12);
        isBeep=0;
        TRISE.F0 = 1; //torna a porta como ENTRADA, desabilitando o alarme
    } else {
        T6963C_sprite(228,0,icone_beep_on,12,12);
        isBeep=1;
        TRISE.F0 = 0; // torna novamente como saída
    }
}

void alternarAlarmeAuto() {
    unsigned short offsetSp02, offsetBPM;
    offsetSp02 = 5;
    offsetBPM = 5;

    if(isAlarmeAuto) {
        T6963C_sprite(174,0,icone_vazio30,30,12);
        //    configAlarmePadrao(); // volta as configurações padrão de alarme
        isAlarmeAuto=0;
    } else if (ByteSp02 != 0xFF) { // não permite setar alarme automático com ausencia de pulso
        T6963C_sprite(174,0,icone_alarme_auto,30,12);
        isAlarmeAuto=1;
    }
}

```

```

        valorAlarme[0] = (ByteSpO2 < (100-offsetSpO2) ) ? ByteSpO2+offsetSpO2 : 100;    //SpO2Alto
        valorAlarme[1] = ByteSpO2-offsetSpO2;    //SpO2Baixo
        valorAlarme[2] = ByteBPM+offsetBPM;    //BPMAlto
        valorAlarme[3] = ByteBPM-offsetBPM;    //BPMBaixo
    }
}

//Configurações de alarme e valores padrão
void configAlarmePadrao() {
    valorAlarme[0] = 100; // SpO2 alto
    Delay_ms(20);
    valorAlarme[1] = 85; // SpO2 baixo
    Delay_ms(20);
    valorAlarme[2] = 150; // BPM alto
    Delay_ms(20);
    valorAlarme[3] = 40; // BPM baixo
    Delay_ms(20);
}

```

GLOSSÁRIO

Capacitância	É a grandeza elétrica de um capacitor, determinada pela quantidade de energia elétrica que pode ser armazenada.
Capacitor	É um componente que armazena energia num campo elétrico, construídos com placas condutoras separadas por elementos isolantes.
<i>Chip</i>	Dispositivo microeletrônico que consiste de muitos componentes interligados capazes de desempenhar muitas funções. Suas dimensões são extremamente reduzidas. Também conhecido como Circuito Integrado.
Curva Pletismográfica	É um gráfico que ilustra os batimentos cardíacos (sístole e diástole).
<i>Datasheet</i>	Documento relativo a um componente eletrônico, representando uma Folha de Dados com especificações e dados técnicos.
Diodo	É um dispositivo ou componente eletrônico composto de cristal semicondutor de silício ou germânio numa película cristalina cujas faces opostas são dopadas por diferentes gases durante sua formação.
<i>Display</i>	Dispositivo para apresentar informações, de modo visual, cuja entrada é fornecida por sinais elétricos.
<i>Feedback</i>	É o procedimento que consiste no provimento de informação a uma pessoa sobre o desempenho, conduta ou eventualidade executada por ela, e objetiva reorientar e/ou estimular ações determinadas, executadas anteriormente.
Fios Ideais	É uma representação hipotética de um fio que não apresenta resistência elétrica.
Fotodiodo	Componente eletrônico sensível à luz.
<i>Hardware</i>	Conjunto de componentes eletrônicos, circuitos integrados e placas.
Hemoglobina	É uma substância que circula no sangue dentro dos glóbulos vermelhos.
<i>Lei Beer-Lambert</i>	É uma relação empírica que, na Óptica, relaciona a absorção de luz com as propriedades do material atravessado por esta.
Multímetro	Destinado a medir e avaliar grandezas elétricas, é um instrumento que pode ter mostrador analógico (de ponteiro) ou digital.
Onda	Em física, uma onda é uma perturbação oscilante de alguma grandeza física no espaço e periódica no tempo – um pulso energético.
Osciloscópio	É um instrumento de medida eletrônico que cria um gráfico bi-dimensional visível de uma ou mais diferenças de potencial.
Placa de Circuito	Consiste de uma placa de fenolite, fibra de vidro, fibra de poliéster, filmes específicos à base de diversos polímeros, etc., que possuem a superfície coberta numa ou nas duas faces por uma película de cobre, prata, ou ligas à

Impresso	base de ouro, níquel entre outras, nas quais são desenhadas pistas condutoras que representam o circuito onde serão fixados os componentes eletrônicos.
Portas Lógicas	São dispositivos, ou circuitos lógicos, que operam um ou mais sinais lógicos de entrada para produzir uma e somente uma saída, dependente da função implementada no circuito. São geralmente usadas em circuitos eletrônicos, baseadas na Lógica Matemática ou Lógica de Boole.
<i>Probe</i>	Compartimento que abriga o sensor utilizado no Oxímetro de Pulso.
<i>Protoboard</i>	É uma placa que forma uma matriz de contatos, com milhares de furos e conexões condutoras de corrente elétrica, utilizada para montagem de circuitos elétricos experimentais.
Protótipo	É um produto que ainda não foi comercializado, mas está em fase de testes ou de planejamento.
Resistência	É a capacidade de um corpo qualquer se opor à passagem de corrente elétrica pelo mesmo, quando existe uma diferença de potencial aplicada.
Resistor	Dispositivo elétrico muito utilizado em eletrônica, com a finalidade de transformar energia elétrica em energia térmica, a partir do material empregado, que pode ser carbono.
Sensor	Dispositivo que recebe e responde a um estímulo ou um sinal.
<i>Software</i>	Programa de Computador, seqüências de instruções a serem executadas por um computador ou máquina semelhante.
Solução	É uma mistura de compostos que se apresenta de forma homogênea.
Soluto	É a substância de minoritária numa solução ou, em geral, a substância de interesse.
Terminal (pino)	Pequeno pino de metal condutor elétrico, que provê conexão aos circuitos integrados aos outros componentes de um circuito.
Transformador	Dispositivo destinado a transmitir energia elétrica ou potência elétrica de um circuito a outro, transformando tensões, correntes. Operam sobre o princípio da indução eletromagnética.
Unidade Computadorizada	É um dispositivo, uma máquina ou um produto dotado de um microprocessador e outros componentes, capaz de realizar cálculos e executar operações pré-programadas.
Valor de SpO2	É a percentagem de saturação de oxigênio no sangue.
<i>Workflow</i>	É o Fluxo de Trabalho, ou seja, a seqüência de passos necessários para que se possa atingir a automação de um processo.

**Universidade Estadual de Maringá
Departamento de Informática
Curso de Engenharia de Produção
Av. Colombo 5790, Maringá-PR
CEP 87020-900
Tel: (044) 3261-4196 / Fax: (044) 3261-5874**