PONTIFÍCIA UNIVERSIDADE CATÓLICA

Maurício Mendez Ribeiro

Pressão Positiva Contínua nas Vias Aéreas

- CPAP -

Professor orientador: **Prof. João Antônio Palma Setti**

_____
Prof. João Antônio Palma Setti

Julho 2008

PONTIFÍCIA UNIVERSIDADE CATÓLICA

Maurício Mendez Ribeiro

Projeto Final

Pressão Positiva Contínua nas Vias Aéreas

- CPAP -

Projeto Final apresentado ao Curso de Engenharia de Computação, ao Professor Luiz Lima, da Pontifícia Universidade Católica do Paraná, sob orientação do Professor João Antônio Palma Setti.

LISTA DE FIGURAS

LISTA DE TABELAS

Histórico da Revisão

Data	Versão	Descrição	Autor
19/08/2007	0.1	Proposta de Projeto	Maurício M. Ribeiro
24/09/2007	0.3	Planejamento de Projeto	Maurício M. Ribeiro
25/09/2007	0.5	Planejamento de Projeto	Maurício M. Ribeiro
03/10/2007	1.0	Planejamento de Projeto	Maurício M. Ribeiro
10/11/2007	1.1	Projeto Físico	Maurício M. Ribeiro
15/11/2007	1.3	Projeto Físico	Maurício M. Ribeiro
23/11/2007	1.5	Projeto Físico	Maurício M. Ribeiro
24/11/2007	1.7	Revisão Projeto Físico	Maurício M. Ribeiro
27/11/2007	2.0	Entrega do Projeto Físico	Maurício M. Ribeiro
02/03/2008	2.1	Montagem do Hardware MSP	Maurício M. Ribeiro
15/03/2008	2.2	Montagem do Hardware Interface	Maurício M. Ribeiro
21/04/2008	2.6	Alterações e Relatório do Protótipo	Maurício M. Ribeiro
23/04/2008	3.0	Projeto Físico Revisado	Maurício M. Ribeiro
30/05/208	3.2	Alterações no Hardware	Maurício M. Ribeiro
07/06/2008	3.4	Testes do hardware	Maurício M. Ribeiro
20/06/2008	3.6	Teste do Controlador	Maurício M. Ribeiro
26/06/2008	3.8	Ajustes do Hardware	Maurício M. Ribeiro
28/06/2008	3.9	Etapa de Testes	Maurício M. Ribeiro
29/06/2008	4.0	Relatório Final	Maurício M. Ribeiro

1. Resumo

A síndrome da apnéia e o ronco causam aos pacientes transtornos sociais e psicológicos como sonolência diurna excessiva, transtornos da conduta e da personalidade podendo haver conseqüências físicas como arritmias cardíacas, hipertensão e isquemia miocárdica [2,12].

O CPAP é utilizado como terapia para estes casos, no entanto os equipamentos disponíveis atualmente no mercado nacional são importados e de alto custo, logo nem todos os pacientes tem acesso.

Os equipamentos importados comercializados no Brasil atualmente são de diversas marcas como Respironics, Breas, Sullivan, Taema entre outros.

Os produtos relacionados a terapia do sono que utilizam o conceito de pressão positiva nas vias aéreas, possuem características básicas conforme configuração prescrita pelo médico, produtos mais sofisticados permitem ao paciente na expiração uma diferença de fluxo permitindo uma sensação de alívio, outros ainda permitem a configuração de pressão expiratória e inspiratória.

Esses equipamentos são comercializados novos e usados, o preço varia conforme a moeda americana, moeda européia, características e acessórios. Para a utilização de um equipamento de CPAP são necessários acessórios de qualidade pois os acessórios estão diretamente relacionados com a qualidade do sono e da utilização diária do equipamento, caso os acessórios sejam de qualidade inferior podem vir a causar incomodo ao paciente durante o sono, provocando a não utilização do equipamento mesmo de forma involuntária.

Desenvolvendo um CPAP no mercado nacional é possível diminuir o custo e proporcionar um melhor acesso aos usuários finais.

No desenvolvimento do CPAP será utilizada tecnologia disponível no Brasil, turbina de baixo ruído e técnicas de controle, implementadas na linguagem C.

2. Introdução

A síndrome da apnéia e o ronco têm sido muito discutidos atualmente. Este problema, além dos transtornos sociais e psicológicos, trás conseqüências

físicas para o paciente (arritmias cardíacas, hipertensão e acidente vascular cerebral) [2].

A apnéia do sono é a obstrução das vias aéreas por tecidos da orofaringe por alguns momentos durante a noite, impedindo a respiração por alguns segundos. O ronco ocorre devido à vibração dos tecidos da orofaringe após a passagem do ar durante as incursões respiratórias [2,8,12] .

O CPAP pode ser utilizado para o tratamento desses problemas, pela facilidade de adaptação e a eficácia de aparelhos já desenvolvidos. No entanto, estes equipamentos são importados possuem um alto custo e não existe similar fabricado no Brasil [2].

Através do CPAP existe um fluxo permanente de ar, que mantém as vias aéreas respiratórias abertas proporcionando assim uma melhor distribuição do gás nas unidades alveolares [2].

O objetivo é desenvolver um equipamento de ventilação, baseado no conceito de pressão positiva nas vias aéreas (CPAP) para o tratamento da *apnéia obstrutiva do sono* (AOS) nos adultos.

Basicamente o equipamento receberá um determinada configuração, conforme prescrita pelo médico, em modo *"config"*, a configuração ficará armazenada na memória. Ao ligar o equipamento em modo de usuário, o microcontrolador buscará a informação pré-configurada e a utilizará para acionamento da turbina. O sensor de fluxo fará a leitura do fluxo de saída, retornando essas informações ao microcontrolador, o qual deverá verificar se é necessário a correção das informações enviadas a turbina para correção do fluxo, não permitindo variações maiores que +/-1 cmH_20 na saída.

Este documento é composto pelo detalhamento do projeto, estado da arte em que são as soluções atuais, trabalho a ser desenvolvido durante a execução do projeto, descrição das tecnologias utilizadas, procedimentos de testes e validação, cronograma e conclusão.

3. Detalhamento do Projeto

A apnéia obstrutiva é um bloqueio repetido das vias aéreas durante o sono. A obstrução pode ocorrer no nariz, no palato mole ou na base da língua.

Esta obstrução leva a ruptura ou fragmentação do sono e a queda da oxigenação do sangue[2].

A apnéia não tratada aumenta a incidência de problemas cardíacos que incluem aumento de pressão arterial, arritmia cardíaca e infarto do miocárdio, pode ainda apresentar apoplexias, o ato de dirigir automóvel sonolento aumenta grandemente o risco de sofrer acidentes automobilísticos [2,12].

O conceito de pressão positiva nas vias aéreas consiste em evitar a completa eliminação do gás inspirado, mantendo por conseqüência direta maior estabilidade alveolar. O aumento da capacidade residual funcional faz com que ocorra aumento da pressão intra-alveolar ao final da expiração, permitindo, assim, melhora nas trocas gasosas. A aplicação do CPAP mantém fluxo aéreo permanente, proporcionando uma melhor distribuição do gás nas unidades alveolares [8].

3.1 Estado da Arte

As terapias médicas adicionais para diminuir a severidade da apnéia do sono são diversas, dentre elas destacam-se: perder peso, suspender uso de sedativo e álcool, mudar a posição do corpo ao dormir (decúbito dorsal, decúbito ventral, decúbito lateral direito e esquerdo) [2].

A cirurgia é indicada para pacientes que apresentam anormalidades anatômicas, pacientes que possuem saúde suficiente para se submeter a um procedimento cirúrgico ou para pacientes que não obtiveram sucesso no tratamento alternativo.

O CPAP é um tipo de terapia alternativa que proporciona ao paciente o conceito de pressão positiva nas vias aéreas, desta forma o paciente não necessita submeter-se a procedimentos cirúrgicos, apenas deve seguir uma dieta e utilizar o equipamento de forma correta, conforme prescrito por seu médico.

Não há equipamento de CPAP produzido no Brasil devido à arquitetura e tecnologia utilizada em seu projeto. As tecnologias empregadas nestes equipamentos são diversas e a manutenção de alto custo.

Dentre os equipamentos comercializados no Brasil destacam-se:

- **CPAP Smart – Virtuoso LX marca Respironics**

  Fornece pressão positiva contínua nas vias respiratórias (CPAP) apenas para o tratamento da Apnéia Obstrutiva do Sono (AOS) nos adultos. O Sistema fornece diversas opções de tratamento para que o paciente possa se adaptar às suas necessidades.

  CPAP – Pressão Positiva Contínua nas Vias Respiratórias (PPCVR)
  Auto-CPAP – o sistema controla a respiração durante o sono e ajusta a pressão automaticamente para adaptar as necessidades do paciente seu modelo é apresentado na figura 1 [10].

Figura 1- Virtuoso LX

- **REMstar | pro with C-FLEX marca Respironics**

  O sistema REMstar Pro com C-Flex da Respironics é um dispositivo de emissão de pressão positiva continua (CPAP) concebido apenas para o tratamento da apnéia obstrutiva do sono em pacientes adultos, seu modelo é apresentado na figura 2 [11].

Figura 2 - REMstar pro

- **CPAP TAEMA**

Sistema para tratamento do sono e apnéia, seu modelo é apresentado na figura 3.

Figura 3 - CPAP Taema

- **CPAP Sullivan** S6 II Lightweight

Desenvolvido com tecnologia de alta absorção de ruídos e motor com microprocessador controlado que mantém a pressão constante para maior conforto na terapia.

Os botões frontais com backlight facilitam a verificação e o ajuste dos parâmetros mesmo durante a noite, seu modelo é apresentado na figura 4.

Figura 4 - CPAP Sullivan

Contra-Indicações

Estudos efetuados mostram que o uso do tratamento de pressão de netilçação positiva é contra indicado em casos de:

1. Doença Pulmonar Enfisematosa;
2. Baixa Pressão Arterial Patológica;
3. Pneumotórax;

O uso do tratamento de pressão de ventilação positiva pode ser temporariamente contra-indicado se apresentar sinais de sinus ou de infecção no ouvido médio. Se houver duvidas durante o tratamento é necessário contatar o médico responsável [10] .

O custo dos equipamentos novos comercializados no Brasil são em torno de quatro mil reais, os valores dependem das características e particularidades de cada equipamento, no entanto além de comprar o CPAP, para a utilização deve-se comprar os acessórios que em muitos casos são vendidos separadamente pois depende do cliente escolher qual acessório lhe proporciona menor incomodo.

Equipamentos usados são vendidos por um preço mínimo de oitocentos reais R$800,00 mais o valor referente aos acessórios.

3.2 Trabalho Desenvolvido

A aplicação de um fluxo aéreo permanente, mantém as vias aéreas respiratórias abertas para proporcionar uma melhor distribuição do gás nas unidades alveolares, consiste no desenvolvimento de um equipamento que gere um determinado fluxo de ar pré-programado com baixa oscilação em sua saída[2].

Para se obter um fluxo de ar pré-programado “constante”, será efetuada a leitura do fluxo na saída do equipamento. Os desvios do fluxo em relação ao programado será corrigido através das técnicas empregadas no controle de erro no microcontrolador.

O funcionamento do equipamento dependerá da ultima configuração válida, a qual será armazenada em sua memória. Ao ligar o equipamento, inicializará o microcontrolador o qual deve fornecer o acionamento da turbina e manter seu funcionamento para que a turbina gere o fluxo pré-configurado, o sensor localizado próximo a saída do fluxo tem como função medir o fluxo de saída para que o controlador compare os valores adquiridos na saída com o pré-configurado e caso necessário corrigir o fluxo gerado pela turbina.

A configuração do equipamento poderá ser efetuada, desde que insira um código para entrar no modo de configuração, no entanto a alteração de valores de fluxo e rampa, devem ser respeitadas conforme a solicitação do médico.

4. O Projeto

4.1 Hardware e Software

O projeto do hardware é baseado no microcontrolador MSP430F449, sensor de pressão/fluxo MPXV5004G, display LCD 16x2 para visualização das informações, teclado e turbina.

O sistema desenvolvido tem como objetivo o controle da pressão/fluxo constante na saída do equipamento se baseando nas informações adquiridas pelo sensor. A pressão mínima fornecida é de 4cm H_2O e a máxima de 16cm H_2O, com uma variação de +/- 1cm H_2O.

O ambiente de desenvolvimento utilizado é o IAR Embedded Workbench KickStart for MSP430 V3, desenvolvido pela Iar Systems.

O projeto está sendo desenvolvido na linguagem C, no entanto para obter maior velocidade também é utilizado a linguagem Assembly.

O sistema basicamente é constituído de sete blocos, conforme a figura 5, com suas funções descritas na tabela 1.

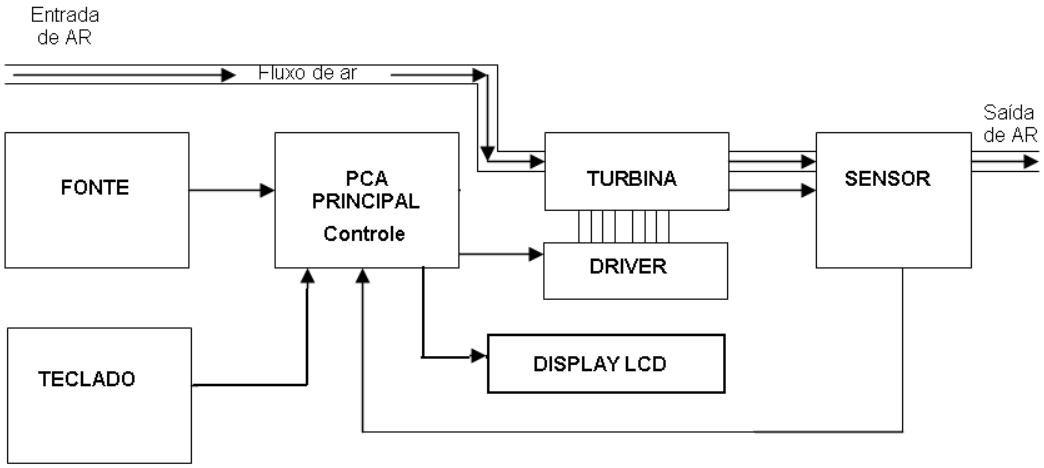

Figura 5 - Diagrama em Bloco

BLOCO	FUNÇÃO	IMPLEMENTAÇÃO
FONTE	Alimentação dos circuitos.	Utilização de uma fonte que

		satisfaça as especificações de todos os fabricantes dos componentes utilizados.
PCA PRINCIPAL CONTROLE	Controle da turbina, interface com o usuário e processamento de sinais. Verifica o fluxo de saída, compara com o fluxo pré-configurado e corrige a turbina.	Desenvolvimento do Hardware para utilização do microcontrolador MSP430. Desenvolvimento em Linguagem C
TURBINA	Gerar maior fluxo de ar na saída em relação à entrada.	Utilização do Motor Brushless DC, sendo necessário o projeto de acionamento do motor conforme especificações do fabricante.
SENSOR	Adquirir o fluxo gerado pela turbina.	Na placa, configurado conforme especificações do fabricante.
Display LCD	Interface com o usuário	Utilização da porta de interface com o MSP430.
Teclado	Interface do usuário com o equipamento.	Utilização da porta de interface com o MSP430.
Driver	Placa de potência para acionar a turbina	Configuração de circuito integrado conforme especificações do datasheet ou desenvolvimento da etapa de potencia para acionamento da turbina por PWM.
DAC	Conversor Digital /Analógico	Consiste em converter um sinal digital em um valor analógico.

Tabela 1 - Descrição do Diagrama em Bloco

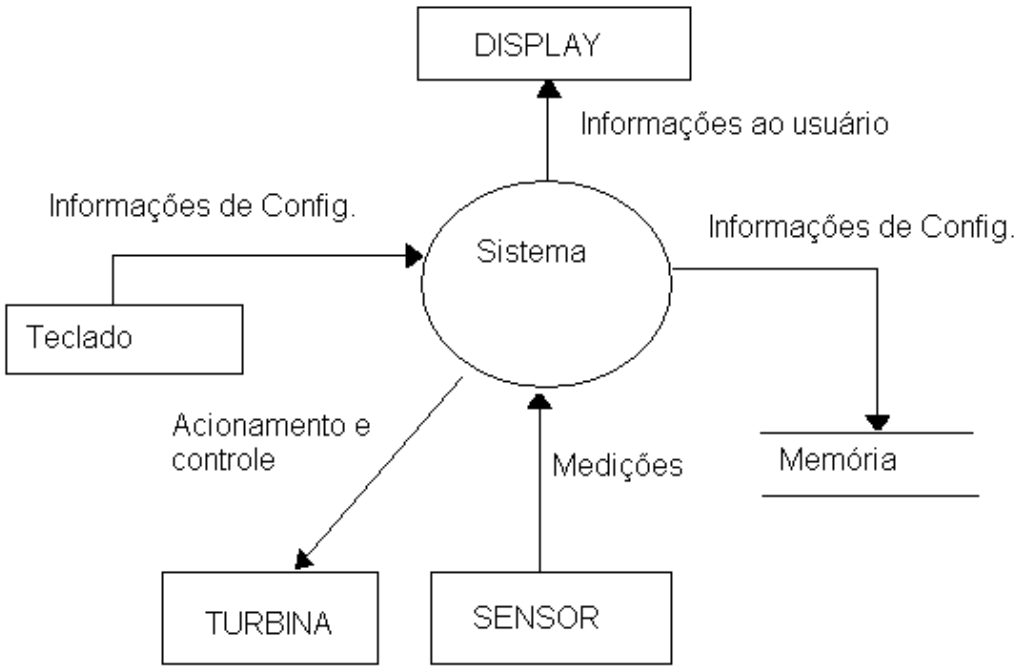

Figura 6 - DFD do Sistema

A figura 6, representa o diagrama de fluxo de dados aplicado ao projeto.

5 Tecnologias Utilizadas

O microcontrolador MSP430, apresenta baixo consumo, baixa tensão de operação, alto desempenho, facilidade de gravação e depuração, além de um design simples e ao mesmo tempo poderoso[7,9].

No microncontrolador será desenvolvido o software de controle da turbina, assim como o tratamento dos sinais adquiridos pelo sensor, além de fornecer ao usuário as informações de configuração do equipamento, pré-configurado via teclado pelo usuário.

O desenvolvimento do controle de fluxo no projeto será realizado em linguagem C, devido à rapidez e a facilidade em operar o microcontrolador.

A turbina utilizada no desenvolvimento do projeto envolve alta tecnologia a partir de seu funcionamento sem escovas, é bastante silenciosa e ainda de longa durabilidade dependendo dos rolamentos instalados.

a. MSP430 PUCPR Classroom Kit V2.0 - PrototypeSP430F149

MSP430 é um microcontrolador de 16 bits, possui arquitetura RISC combina um conjunto reduzido de 27 instruções e 24 emuladas utilizando uma arquitetura de barramento clássica Von Neumann, permitindo que a CPU possua um espaço único de endereçamento de memória[7.9,].

Dispositivo dotado de memória de programa PROM, ROM ou FLASH (MSP430F1XX). São dispositivos de uso geral, dotados de uma grande variedade de periféricos[9].

As características da arquitetura MSP430 são:

- Baixo Consumo;
- Baixa Tensão de Operação;
- Alta Performance;
- Conjunto de instruções Ortogonais;
- Número Reduzido de instruções;
- Grande quantidade de periféricos;
- Facilidade de gravação e de depuração;

- Diversos encapsulamentos;

Na figura 7, temos o kit v2.0 do MSP430, desenvolvido pela PUCPR(Pontifícia Universidade Católica do Paraná)[7].

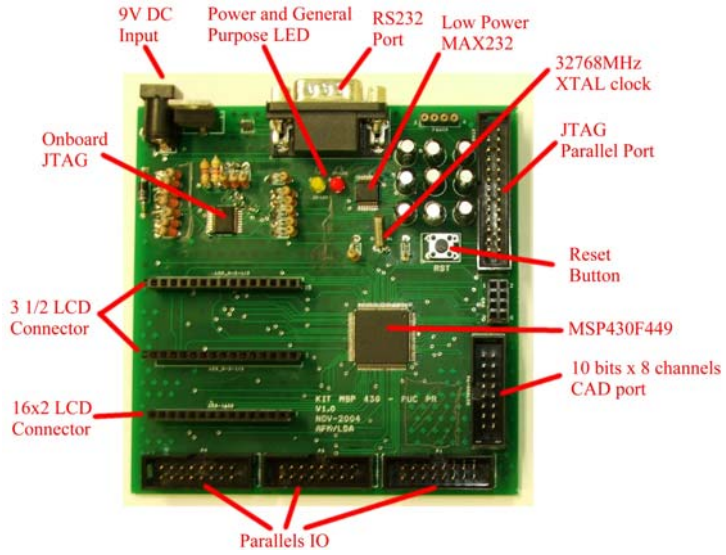

Figura 7 - Kit de desenvolvimento MSP430F149

b. IAR Embedded Workbench KickStart for MSP430 V3

O ambiente Embedded Worbench é um IDE(*Integrated Development Environment* – Ambiente Integrado de Desenvolvimento) composto de um editor de arquivos, montador Assembly, compilador C e Embedded C++, ligador, simulador e emulador. Logo para o projeto necessita apenas de uma ferramenta de software para desenvolvimento do processo, mesmo havendo outras ferramentas disponíveis como MSPGCC, *Code Composer essentials, Quadravox, Imagecrafte Crossworks(Rowley)*[9].

O ambiente é baseado no conceito de *workspaces* (espaços de trabalho) que são módulos que podem agregar um ou mais projetos, logo em um projeto pode haver mais de um código fonte distribuído em arquivos, para gerar um arquivo binário, utilizável na simulação e programação do microcontrolador[9].

Este ambiente proporciona ao programador a utilização de linguagem Assembly ou linguagem C. Na figura 8 tem-se a janela inicial do Embedded Workbench.

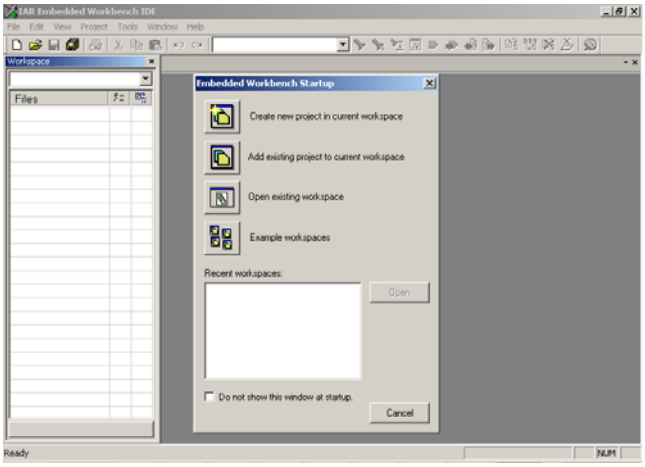

Figura 8 - Workspace Embedded Workbench

c. Dispaly LCD 16x2

Os módulos LCD são interfaces de saída muito útil em sistemas microprocessados. Estes módulos podem ser gráficos e a caracter. Os módulos LCD gráficos são encontrados com resoluções de 122x32, 128x64, 240x64 e 240x128 dos pixel, e geralmente estão disponíveis com 20 pinos para conexão[7].

Os módulos podem ser encontrados com *LED backlight* (com uma iluminação de fundo) para facilitar as leituras durante a noite, permite ainda ajuste de intensidade de luz e contraste[7].

Figura 9 - LCD no MSP430

Estes módulos utilizam um controlador próprio, permitindo sua interligação com outras placas através de seus pinos, onde deve ser alimentado o módulo e interligado o barramento de dados e controle do módulo com a placa do usuário. Naturalmente que além de alimentar e conectar os pinos do módulo com a placa do usuário deverá haver um protocolo de comunicação entre as partes, que envolve o envio de bytes de instruções e bytes de dados pelo sistema do usuário[7].

Na figura 9, está o display LCD junto ao módulo do MSP430.

d. Sensor de Fluxo ou Pressão

Figura 10 - Sensores de Fluxo

O sensor piezo resistivo é um transdutor monolítico de silício de pressão, esses sensores são utilizados para uma vasta gama de aplicações, mas sobretudo as que empregam um microcontrolador ou microprocessador com A/D insumos[1].

Figura 11 - Sensor

O sensor combina uma estirpe altamente sensíveis implantados com bitola fina, filme metalizado, bipolares e fornece uma descrição exata, o elevado nível de saída sinal analógico é proporcional à pressão aplicada. O sensor pode ser visualizado nas figuras 10 e 11[1].

e. Motor BRUSHLESS DC

O motor de corrente contínua sem escovas ou BLDC (Brushless DC) oferece diversas vantagens sobre os motores de corrente contínua com escovas, dentre as quais destacam-se: a confiabilidade mais elevada, o ruído reduzido, a vida útil mais longa (devido a ausência de desgaste da escova), a eliminação da ionização do comutador e a redução total de interferência eletromagnética .

A desvantagem principal do motor sem escovas é o custo mais elevado, a qual se deve a dois fatores: primeiramente, estes motores requerem dispositivos MOSFET de alta potência na fabricação do controlador eletrônico de velocidade. Os motores de BLDC necessitam de um circuito integrado, chamado de controlador eletrônico de velocidade para oferecer o mesmo tipo de controle variável. O motores BLDC são considerados mais eficientes do que os motores de corrente contínua escovados. Isso significa que para a mesma potência de entrada, os motores de BLDC converterão mais energia elétrica em energia mecânica do que um motor de corrente contínua escovado. A eficiência é maior na região de "baixa-carga" e "à vazio" na curva característica do motor. Sob cargas mecânicas elevadas, os motores de BLDC e os motores escovados de alta qualidade são equivalentes em eficiência.

O motor sem escova é um imã hibrido permanente. A figura 12 é uma ilustração simplificada de como funciona o motor. O fluxo de corrente é causado pela condução na armadura, produzindo o torque. A lei física que expressa o funcionamento do motor é[13]:

$$F = K.B.l.i \quad (1)$$

Onde:

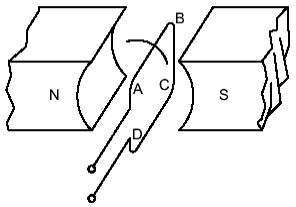

Figura 12 - Motor simplificado

F = Força
K = Uma Constante
B = Densidade do fluxo de ar
l = Comprimento do condutor
i = corrente no condutor

Se mais de um condutor está a utilizar a mesma corrente, então:

$$F = K.B.l.i.z \quad (2)$$

Onde z=numero de condutores em serie.
O torque é definido por;

$$T = F.R$$

R = Raio

Logo, $T = K.B.l.i.z$ (3)

Devido a suas características e modos de funcionamento proporciona a projetos em que são utilizados vida longa e principalmente a não emissão de ruídos.

O motor sem escova foi escolhido devido as suas características que são envolvidas diretamente no projeto, pois ao utilizar um motor comum é necessário a implantação de filtros que anulem estes ruídos. Para haver êxito na

conclusão do projeto o CPAP, deve ser silencioso para que não acorde o usuário ou ainda seu companheiro [3,6,13].

f. Fonte DC

A fonte é constituída por circuitos de proteção estabilizada em 24V basicamente por fusíveis para proteção dos circuitos, transformador, ponte retificadora e filtros.

6 Procedimentos de Teste e Validação do Projeto

O procedimento de teste do equipamento a ser desenvolvido apresentará as informações ao usuário de forma correta, avaliando o que é mostrado em seu display e comparado com o projeto inicial.

O manômetro digital irá testar o fluxo de saída assim é possível verificar se o sistema de controle funciona de acordo com a configuração definida pelo usuário.

Calibração dos sensores será conduzida conforme especificações do fabricante.

O software desenvolvido será testado e corrigido através do ambiente de desenvolvimento IAR Embedded Workbench KickStart for MSP430 V3, desenvolvido pela Iar Systems[9].

a. Testes em caixa Branca

Os testes em caixa branca foram realizados através de simulações e emulações através do ambiente de desenvolvimento Embedded Workbench, onde foram adicionadas a área de trabalho informações de fluxo desejado, sob estas informações o sistema corrigir o fluxo na saída. Também foi possível testar a interface de teclado e display.

b. Testes em caixa Preta

Os testes em caixa preta ocorreram manualmente com teste de mesa sob o código fonte, as respostas do microprocessador foram analisadas quando

havia a necessária de correção de um determinado fluxo, como modificando a configuração do equipamento em que o mesmo deve responder próximo do tempo real e alterando para o modo de rampa os resultados obtidos durante os testes foram avaliados, permitindo alterações no código fonte para melhor desempenho do equipamento.

7 Desenvolvimento do Projeto

a. Sensor de Pressão

O elemento sensor de pressão utilizado é um transdutor diferencial modelo MPXV5004G, fabricado pela empresa Motorola, Inc. De acordo com o fabricante (Freescale Semiconductor), este sensor apresenta compensação interna de temperatura e pode ser utilizado tanto para medição de pressão em determinado ponto, como o diferencial de pressão entre dois pontos distintos. O referido modelo de sensor é produzido para atender uma faixa de pressão que varia de 0 a 3,92 KPa (0 a 40 cm H_2O), apresentando erro máximo de 1,5 % para temperaturas entre 0 a 60°C. Quando alimentados por uma tensão estabilizada de 5 V, emite sinais analógicos que variam de 1,0 V a 4,9 V , os quais podem ser transformados em leituras de pressão, segundo o fabricante (Freescale) pela seguinte equação:

$$P = (Vout - 1) +/- 1{,}5\% \text{ , para } Vcc = 5V$$

em que P(KPa) é a diferença de pressão observada nas entradas do transdutor e Vout a diferença de potencial elétrico entre os pinos de saída e o referencial terra (GND).[15]

As saídas do transdutor de pressão foram ligadas a porta P6 do microcontrolador MSP430F449IFZ, a qual oferece a função de um conversor A/D 12 bits e ao referencial terra do periférico. O fabricante disponibiliza outros modelos, com as mesmas características, que podem ser utilizados para atender a outras faixas de pressão.[15]

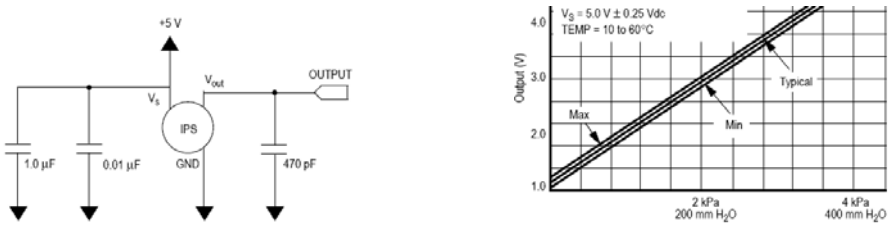

Figura 13 - Montagem Sensor

Figura 14 - Gráfico Sensor

b. Interface com usuário (Teclado, Indicações Visuais, Display)

O Teclado utilizado no projeto é constituído de 4 Botões, em uma menbrana, a porta de comunicação utilizada para a conexão da “Interface Teclado” é a porta P3.

A interface foi projetada totalmente pelos sinais fornecidos pela placa da CPU, baseado em uma matriz 4 x 1, onde tem-se quatro linhas e uma coluna. A leitura do teclado é feita colocando um nível lógico “0” em cada sinal de ativação da coluna e monitorando os sinais das linhas. Caso uma tecla seja pressionada, a respectiva linha terá um nível lógico “0”; caso contrário , a linha apresenta um nível “1”. Essa operação é repetida para cada linha.

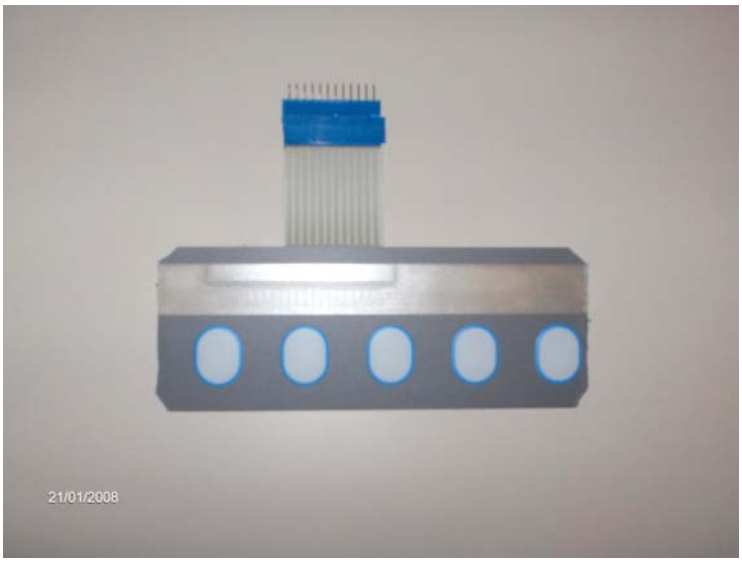

Figura 15 - Teclado Menbrana

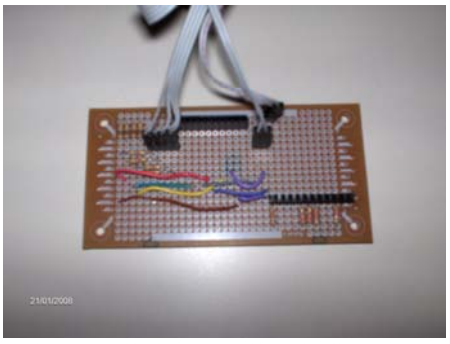

Figura 16 - Placa de Interface do Teclado

c. Driver Controller para Brushless DC Motor

O MC33035 é um componente controlador de motor brushless DC possui alto desempenho possui todas as funções ativas necessárias para implementar um circuito com duas, três ou quatro fases.

Este dispositivo consiste de um rotor posição decodificador de sequência adequada para comutação, através da referência emitida pelo sensor se tem o fornecimento de energia, freqüência “sawtooth” ao oscilador programável, na saída possui três coletores aberto aos condutores do motor, e três altas correntes totem poll, ideal para o acionamento dos MOSFETs de potência.

O dispositivo possui recursos de proteção por subtensão, lock-out, ciclo-a-ciclo limita a um tempo selecionável, propriedade interna térmica programada, e uma única saída que pode ser, interligados a um microprocessador . O sistema controlado típico possui funções que incluem loop velocidade, para frente ou direção inversa e freio “sistema dinâmico de travagem”. O MC33035 é projetado para operar com sensor elétrico de fase de 60 ° / 300 ° ou 120 ° / 240 °, e também pode controlar de forma eficaz motores DC.

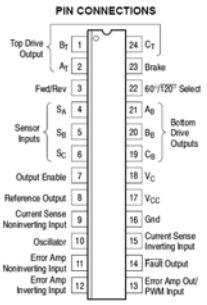

Figura 17 - MC33035

d. MSP430F449IFZ

O microcontrolador MSP430 permite a realização de operações envolvendo operandos de 8 ou 16 bits, além de diversas funções já inclusas no micorcontrolador. A leitura do sensor de pressão é efetuada diretamente pelo MSP, utilizando o conversor A/D 12 bits, permite obter a melhor relação entre o consumo de corrente e velocidade de conversão, referência interna de 1,5 ou 2,5

V provenientes do ADC12 ou externa; auto-calibração para a correção da tensão de offset de saída.[7]

Figura 18 - Montagem da Placa MSP430

e. Conversor A/D

A série do MSP430F449IFZ não possui decodificador analógico digital, então para o controle efetivo do motor brushless foram criados dois projetos de decodificação do sinal digital em sinal analógico.

Um conversor digital-analógico é um dispositivo onde um sinal digital é conectado a sua entrada e este o converte para uma tensão ou corrente analógica proporcional. No entanto, tecnicamente, a saída de um conversor D/A não é uma quantidade analógica porque pode assumir apenas valores específicos. O número de valores possíveis diferentes na saída pode ser aumentada e a diferença entre valores sucessivos diminuída, aumentando-se, apenas, o número de bits da entrada. Isso permite produzir uma saída que é bastante parecida com uma quantidade analógica que varia continuamente dentro de uma faixa de valores.

Uma característica bastante importante em um conversor D/A é a resolução. A resolução de um conversor D/A é definida como a menor alteração que pode ocorrer na saída analógica como resultado de uma mudança na entrada digital. A resolução é sempre igual ao peso do LSB e também é chamada de tamanho do degrau, pois a tensão de saída muda conforme o valor digital de entrada é alterado de um degrau para o próximo.

É chamado de saída de fundo de escala o máximo valor que o D/A converte em sua saída.

Como a resolução é o fator de proporcionalidade na relação de entrada e saída de um conversor D/A, tem-se:

Saída analógica = K x entrada digital

$$resolução = K = \frac{A_{fS}}{(2^n - 1)}$$

Equação 1

Onde A_{FS} é a saída de fundo de escala e n é o número de bits.

i. Conversor TTL74LS145

O componente 74LS145 é um conversor D/A BCD decimal, converte valores digitais de “0h a Ah” em valores decimais, utilizando este dispositivo mais um malha de resistores era totalmente viável a aplicação deste circuito no projeto.

ii. DAC0800

O Dispositivo DAC0800 é um conversor D/A de 8 bits, possui alta qualidade na conversão de valores no entanto necessita de alimentação negativa para o seu funcionamento logo não será uma forma de implementação viável no projeto uma vez que a fonte até então utilizada possui 24V.

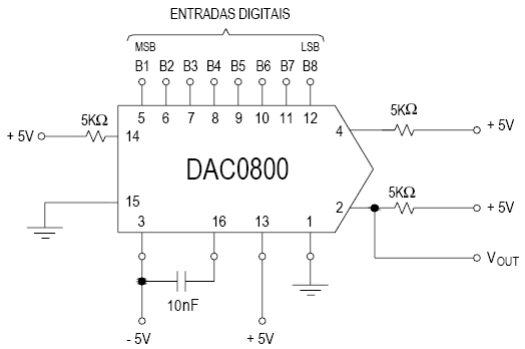

Figura 19- Configuração do DAC0800

f. Circuitos de Proteção

O motor brushless DC consome alta corrente, afim de proteger a porta de acionamento do mesmo foi desenvolvido um sistema de proteção utilizando componentes opto acompladores 4N25, isolando os circuitos no entanto não fornecendo falhas durante a comunicação.

Devido a falhas e valores de tensão flutuantes os componentes 4N25 foram subtituídos por buffers TTL 74LS04N portas lógicas inversoras, com isso obteve-se a proteção do microprocessador e a eliminação de valores flutuantes.

8 Implementação do Projeto

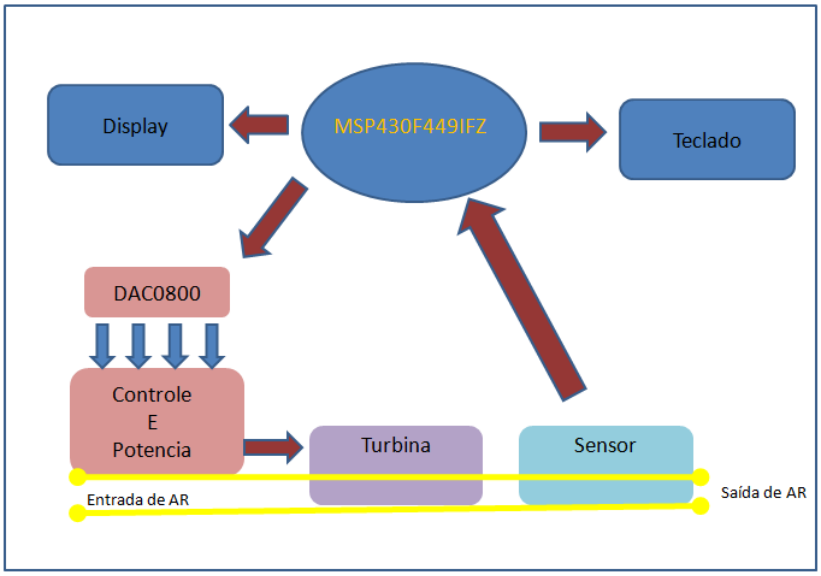

Figura 20 - Daiagrama em Bloco 2

a. Implementação do Hardware

i. Sensor

O Circuito desenvolvido para o sensor é constituído de três capacitores especificados pelo fabricante, deste modo os valores esperados de pressão são definidos pela fórmula abaixo:

$$P = (Vout - 1) +/- 1{,}5\% \text{ , para } Vcc = 5V$$

Equação 2 - Fórmula do Sensor

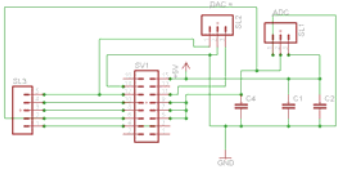

Figura 21 - Circuito do Sensor

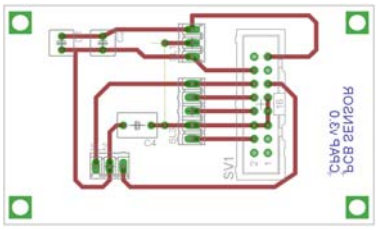

Figura 22 - PCB Sensor

ii. Placa de controle

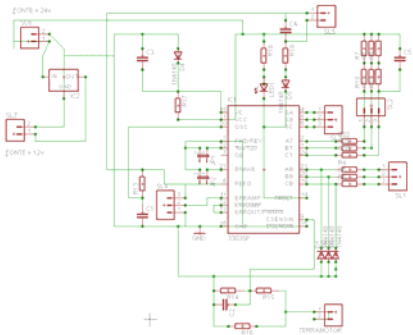

Figura 23 - Circuito de Controle

A placa de controle foi projetada com “clock” operando em 22KHz aproximadamente, sendo definida por C1 e R13, as entradas 4,5 e 6 do MC33035 são os sinais dos sensores Hall do motor. O pino 3 define o sentido de rotação horário ou anti-horário conforme a hélice adaptada no motor Brushless o pino 3 foi conectado ao GND, assim como os pinos 22 e 23, representam respectivamente o ângulo de rotação 60° graus e o freio.

Os pinos 1, 2 e 24 são ligados na etapa de potência junto aos Gate dos Mosfet´s 9640 , as saídas 19, 20 e 21 possuem diodos de proteção para que não forneçam sinais negativos para o controlador quando entrarem em funcionamento pois através destes pinos é enviada a tensão de acionamento das fases do motor através dos Mosfet 640.

A velocidade é controlada a partir da tensão de referência do pino 8 de 6,3V , ou seja ao aplicar um valor de 1,5v a 6,3V ao pino 11 verifica-se o controle da velocidade.

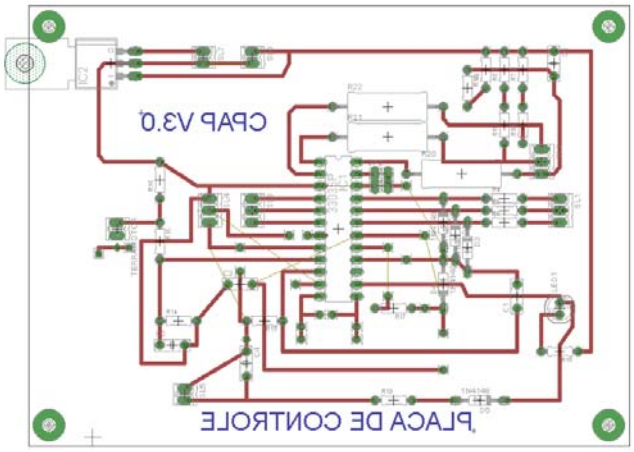

Figura 24 - PCB de Controle

iii. Placa de Potência

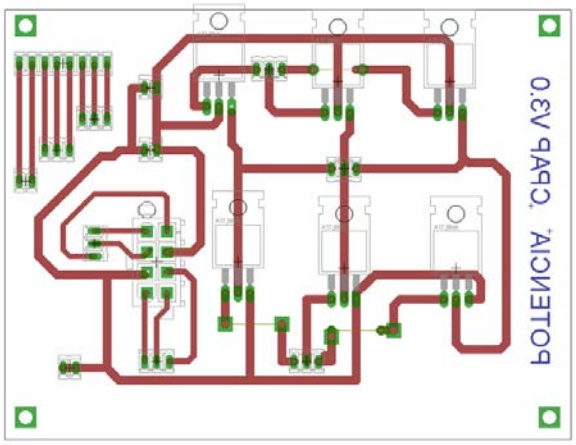

Figura 25 - PCB de Potência

O transistor MOSFET de canal-P IRF9640 na parte superior conduz quando a tensão de porta está próxima de 0V .

Por outro lado, o MOSFET de canal-N IRF640 na parte inferior conduz quando a tensão é de pelo menos 5V acima da tensão do dreno, por isso usamos resistores PULL-UP na sua entrada a fim de garantir essa tensão quando a saída está em tri-state.

Para acionamento de carga indutiva, os transístores MOSFET devem ser dotados (internamente ou externamente) de diodos shotky a fim de desviar destes componentes as correntes em sentido contrário, decorrentes da oposição à corrente aplicada a cada enrolamento.

iv. Codificador e Circuito de Proteção

1. Proteq e TTL 74LS145

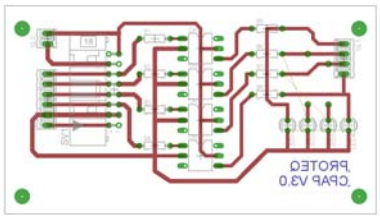

Figura 26 - PCB Proteq

O objetivo do circuito “proteq” é a proteção da porta de saída do microntrolador devido a alta corrente consumida pelo motor, no entanto sua utilização não foi possível devido aos valores de entrada estarem “flutuantes”, logo os sinais após a passagem pela Placa Proteq estavam equivocados.

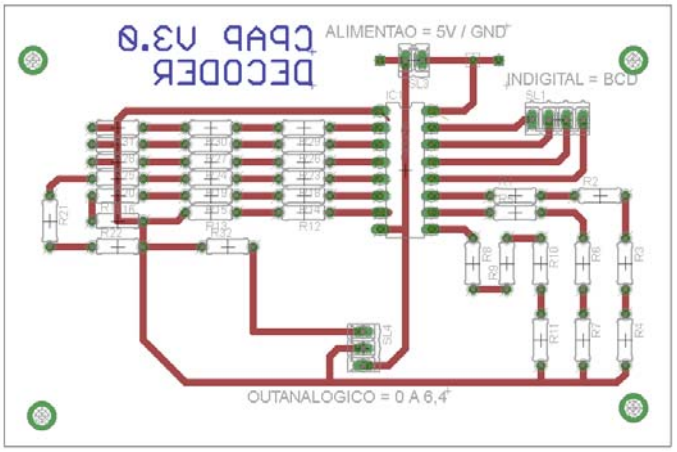

Figura 27 - PCB decoder

O circuito Decoder foi projetado para realizar a conversão de sinais digitais em um sinal analógico através do conversor A/D BCD decimal TTL 74Ls145 acompanhado em cada saída de uma malha de resistores. No entanto os valores convertidos por este circuito não foram aprovados nos testes.

2. Conversor D/A DAC0800

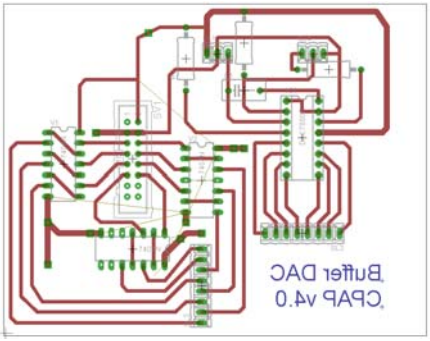

Figura 28 - PCB Conversor DAC 0800

A utilização do DAC0800 foi avaliada após a análise dos resultados apresentados pelas placas “Proteq e Decoder”, devido ao desempenho incompatível com as especificações do projeto.

Utilizando buffers na saída do controlador em que connvertemos as tensões de saída de 3V para 5V informamos ao DAC os valores de entrada, desta forma os valores não são mais flutuantes conforme identificados nos circuitos anteriores.

Como a tensão de referência no DAC é de 6,3 V, e o componente converte valores em 8 bits, para cada bit modificado teremos uma variação na saída de 0,03V aproximadamente a cada bit por exemplo “02h” será igual a 0,06V.

b. Testes do Hardware

Após o projeto de cada módulo, cada módulo foi submetido a uma série de testes, desta forma foi possível definir quais módulos utilizarmos no projeto.

i. Testes das Turbinas

Unipolar ou Bipolar - Normalmente para 2 fases, 1 deles tem 4 fios e o outro tem 5 ou 6 fios. Para saber que tipo de motor estamos trabalhando medimos a resistência entre as fases.

Para o motor ser Unipolar temos 2 valores de resistência e 1 deles é a metade do outro, se for bipolar, apenas um valor de resistência ou circuito aberto.

Através deste teste sabemos que estamos trabalhando com um motor Bipolar.

Teste de operação do motor. Neste teste foram utilizados chaves com possibilidade de VCC(24V) e GND. As chaves foram conectadas nas fases do motor direto, ou então no acionamento dos FETs.

Tendo o cuidado para não curto-circuitar a fonte por fora do motor, para não queimar a fonte, ou obter outros problemas.

Ao variar as posições das chaves, nota-se uma tremida, e uma pequena variação de posição (normalmente 1,8 graus). Ao variar as chaves na sequencia correta de fases (A-B-C) para 3 fases, observa-se um movimento progressivo do motor. Se variar errado, nota-se ele indo e voltando.

Com este teste concluímos se o motor está funcionando ou não e se as fases não estão queimadas.

ii. Teste da Placa de Controle e Potência

O teste do circuito de controle foi efetuado medindo as saídas do CI de controle, verificando a defasagem entre as ondas, com isso podemos verificar se o módulo estava funcionando corretamente.

Com o circuito de potência acoplado ao circuito de controle obtemos sinais próximos ao verificado no datasheet do circuito integrado controlador. Conforme ilustração abaixo:

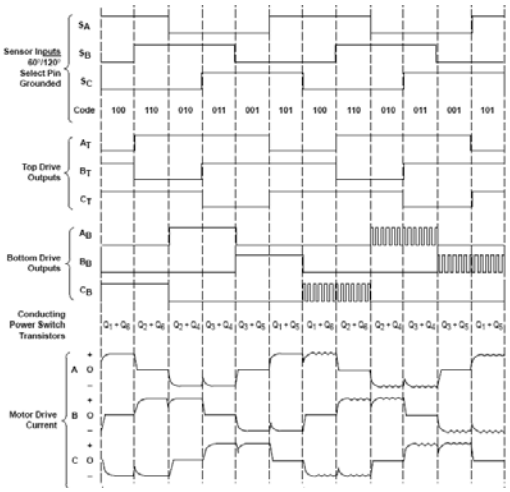

Figura 29 - Formas de Onda

Utilizando um potenciômetro de 10K Ohms nos pinos GND, 11 e o 8, testamos os valores de tensão necessários para o controle da turbina a qual deve gerar o fluxo pré-configurado, os valores adquiridos estão na tabela abaixo.

Tabela 2 - Medidas de Pressão com Trimpot

Vin Pino (11)	Pressão cm H2O
2,98	4
3,19	5,2
3,32	6,0
3,42	6,9
3,43	7,2
3,55	8,1
3,69	9,0
3,80	10,1
4,00	11,2

iii. Teste do módulo Proteq

O módulo proteq foi criado com o intuito de proteger a porta de comunicação do microntrolador, no entanto após a passagem do circuito de proteção, medindo os sinais de saída constatou-se que os sinais ficaram em nível flutuante não sendo

reconhecidos pela porta TTL ou o DAC0800 utilizados para converter o sinal digital em analógico.

iv. Teste da Placa Decoder

Os testes da placa decoder ocorreram através do microcontrolador em que aplicamos um sinal digital na entrada do decoder e em sua saída deveríamos obter 10 valores para o controle efetivo da turbina, no entanto estes 10 valores são insuficientes para o controle total da turbina, analisando a tabela abaixo identificamos que uso deste módulo é inviável devido aos valores apresentados.

Tabela 3 - Valores fornecidos pela DECODER

Sinal Digital	Vout	P(cm H2O)
00h	0,12	0
01h	1,50	0,3
02h	4,95	11,0
03h	1,80	1,5
04h	2,00	2,0
05h	2,23	2,3
06h	2,45	3,0
07h	2,70	3,7
08h	3,00	3,9
09h	3,20	5,0
0Ah	4,95	11,0

v. Teste do Sensor

Conforme a fórmula apresentada na descrição do componente verificamos que a leitura do sensor deve ser feita n vezes e calcular a média dos valores, para que não hajam erros na interpretação de apenas um leitura.

vi. Teste da Placa DAC0800

A placa DAC0800 foi projetada com intuito de corrigir o erro apresentado pela placa DECODER, com isso conseguimos controlar o motor de forma efetiva e ideal para a nossa aplicação, nota-se através deste teste que o circuito de controle é muito sensível pois com uma variação de 10mV temos alteração do fluxo gerado pela turbina conforme a tabela abaixo:

Tabela 4 - Valores fornecidos Pelo DAC0800

Valor	Entrada Digital	Saída Vout	Pressão Cm H2O
0	0x00h	0,0	0,00
153	0x99h	2,98	4,00
157	0x9Dh	3,06	4,50
163	0xA3h	3,18	5,00
167	0xA7h	3,25	5,50
170	0xAAh	3,32	6,00
173	0xADh	3,38	6,60
176	0xB0	3,43	7,10
182	0xB6	3,55	8,00
186	0xBAh	3,63	8,60
189	0xBDh	3,69	9,00
191	0xBFh	3,73	9,40
193	0xC1h	3,77	10,0
197	0xC5h	3,85	10,5
202	0xCAh	3,95	11,0
204	0xCCh	3,99	11,5
255	0xFFh	4,95	12,0

c. Teste do Teclado

Os sistemas de interface foram testados bit a bit, para que caso houvesse erros no projeto fosse corrigidos rapidamente, para isso foi efetuado um código simples em que quando pressionado qualquer botão acendia o Led de teste localizado na porta P2 .

d. Teste do Sensor de Pressão

Os testes realizados no sensor foram efetuados com a variação de uma fonte DC e avaliando os valores obtidos pelo ADC12 no MSP. Com o intuito de eliminar valores incorretos é efetuado a medição de vários valores armazenados em vetor, após a leitura os valores são somados e divididos pelo total de valores fornecendo o valor médio da leitura. Após os testes realizados com a fonte, os testes finais e a calibração do equipamento foram efetuados diretamente no sistema em que utilizando um Manômetro era verificado a pressão , e em seguida colocava-se o sensor e verificava-se a leitura diretamente no display os resultados do teste foram avaliados conforme a tabela abaixo:

Valores de Pressão em cm de H_2O

Tabela 5 - Teste Sensor

Leitura	Pressão Configurada	Pressão Medida Sensor	Pressão Medida Manômetro
1	4,0	3,9	4,1
2	4,0	4,1	4,0
3	4,5	4,4	4,5
4	5,0	5,2	5,1
5	6,0	6,1	6,0
6	7,0	7,3	7,0
7	7,0	6,7	6,9
8.	8,0	8,2	8,0

9	8,5	8,8	8,4
10	9,0	9,1	9,0
11	10,0	9,9	10,1
12	11,0	11,0	10,9
12	12,0	11,7	11,9

Através deste teste verifica-se que os valores lidos pelo sensor estão corretos e dentro da faixa de erro previamente estabelecida de +/- 1 cm H_2O.

e. Teste 24 horas

Como o equipamento é utilizado quando o usuário está dormindo, foram efetuados testes durante 5 noites das 21 horas até 7 horas do dia seguinte, o equipamento funcionou perfeitamente não apresentado qualquer irregularidade.

Quando necessário o módulo de rampa foi acionado, o tempo de rampa foi o correto conforme o configurado.

9 Plano de Testes e Resultados Esperados

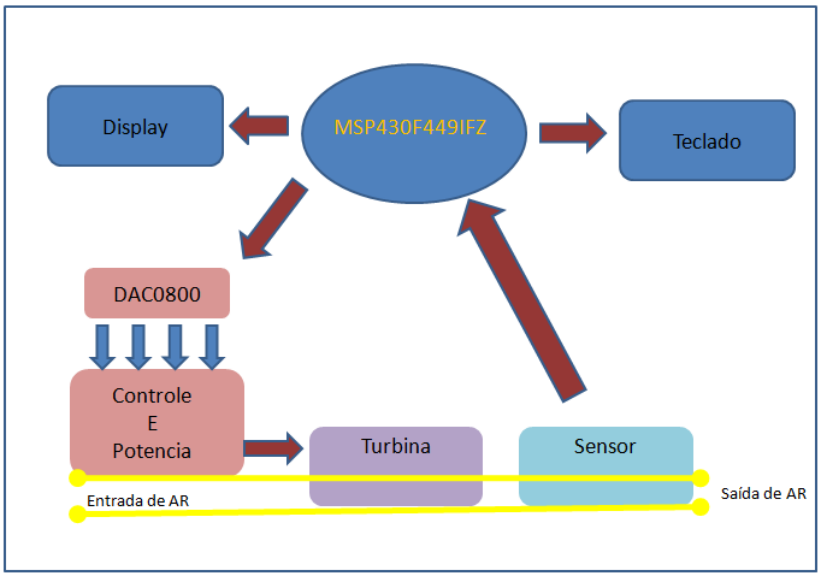

Figura 30 - Diagrama em Bloco

Função Configuração Pressure:

Consiste em efetuar a configuração da Pressão de modo que a mesma deve ser configurada entre 4 e 12 .

Exemplo:

Configurando a Pressão

1. Pressione o Enter, para acessar o menu Pressão. Conforme a tela Abaixo:

2. Para aumentar os valores de Pressão pressione o Botão 2, ou mantenha pressionado até o valor que deseja configurar..
3. Para diminuir o valor pressione o Botão 1, ou mantenha pressionado até o valor que deseja configurar.
4. Para Confirmar a Modificação Pressione Enter.
5. Para Sair da Configuração mantendo a configuração anterior pressione “Sair”.

Resultados Esperados :

Que as variáveis do programa principal recebam as configurações efetuadas no modo config.

Função de Configuração da Rampa:

A função rampa permite 7 tipos de configuração:

00 minutos;

05 minutos;

10 minutos;

15 minutos;

20 minutos;

25 minutos;

30 minutos;

Ao entrar no Menu de configuração da Rampa conforme o procedimento descrito no manual seleciona-se qual o valor de rampa a ser utilizado pelo dispositivo.

Resultado Esperado:

A variável do programa principal deverá receber o tempo selecionado.

Teste do sensor:

Consiste em verificar se a pressão medida está próxima da realidade. Para este teste será utilizado um manômetro para comparação de resultados.

Resultado Esperado:

Os valores adquiridos serão comparados aos valores adquiridos pelo manômetro, desta forma é possível comparar com a realidade.

Teste do Sistema:

Conforme a pré configuração o sistema deverá responder o fluxo pré-configurado, ao acionar o modo rampa o módulo passa a fornecer a pressão mínima em um determinado tempo pré configurado. O sensor mede o fluxo, através desta medição efetua-se o controle do fluxo de saída com uma taxa de amostragem de 500 ms .

10 Conclusão

A obstrução das vias aéreas durante o sono pode causar ao paciente apnéia obstrutiva e ronco, estando sujeito a despertares transitórios repetidos (microdespertares) e fragmentação do sono, dando lugar a manifestações como sonolência diurna excessiva, transtornos da conduta e da personalidade. Estas alterações estão igualmente relacionadas com o surgimento de arritmias cardíacas, hipertensão arterial (com descanso e principalmente sem o descanso noturno) e isquemia miocárdia e morte súbita noturna [2,4,12].

A utilização do CPAP como terapia nestes casos possui extrema eficiência, pois permite ao paciente um fluxo permanente em suas vias respiratórias abertas proporcionando uma melhor distribuição do gás nas unidades alveolares[12].

O objetivo é desenvolver um equipamento de CPAP para o tratamento de AOS possuindo características similares aos importados, no entanto deve possuir baixo custo utilizando tecnologias existentes no mercado nacional, como o microcontrolador MSP430, turbina com funcionamento sem escovas e sensores de pressão.

O projeto de um CPAP nacional é muito importante para pacientes que dependem deste dispositivo para terem um sono bom e tranqüilo, no entanto o projeto de CPAP não é tão simples quanto parece, em primeiro lugar nos deparamos com o motor, por se tratar de um tipo de motor em especial o BLDC, é necessário importar e além disso para efetuar o controle podemos optar por drives de controle o qual foi utilizado neste projeto ou ainda microcontroladores especiais que já possuem a função de controle de motores sem escova.

A utilização destes dispositivos requerem uma série de cuidados durante o manuseio pois realmente são muito sensíveis, o motores ao efetuar qualquer ligação equivocada queimam rapidamente.

Enfim o projeto deste CPAP superou e muito as expectativas de custo, pois a idéia principal era fazer um dispositivo com baixo custo, mas devido as dificuldades encontradas no manuseio destes dispositivos acarretaram num custo elevado.

Apesar do custo este projeto funcionou corretamente, mesmo não fornecendo a pressão máxima de 16 cm de H_20 como desejado, no entanto de acordo com os testes realizados a pressão abaixo do ideal deve ser devido ao motor ser sucateado. Desta forma aceitamos o valor final de 12 cm H_20 como um resultado muito bom.

Pensando em futuro com base neste projeto é totalmente viável lançar este equipamento ao mercado, é possível realizar com a mesma plataforma outras funções além da CPAP, como Auto-CPAP, CPAP C-FLEX, CPAP A-FLEX, BiPAP e ainda utilizando memory card gravar dados referentes ao sono do usuário visando melhores disgnósticos e acompanhamento médico através de software.

11.2 Comentários

As dificuldades na execução do projeto foram muitas, principalmente com a literatura de motores brushless, apesar de terem um funcionamento excelente para

certas aplicações seu uso é complicado e muito sensível um pequeno erro pode comprometer todo o projeto.

A dificuldade em encontrar os componentes no Brasil também devem ser levadas em consideração pois geralmente o projeto atrasa devido ao prazo de entrega dos componentes que muitas vezes são importados.

11 Referências Bibliográficas

1. ALL SENSORS CORP. **MEMS pressure sensor technology, low pressure sensors, pressure transducers**. **Datasheets**. 2005 [online] Disponível: <http://www.allsensors.com/datasheets/commercial_temp/DS-0165_RevB1.pdf > acessado em 15/11/2007.

2. AZEREDO, C. A. C. **Ventilação Mecânica – Invasiva e Não Invasiva**. Rio de Janeiro: Revinter, 1994.

3. FAULHABER; **Minimotor SA. Micromotores – [online]** Switzerland, Disponível: < http://www.minimotor.ch/es/welcome_es.html > Acessado em 24/08/2007.

4. KNOBEL, E. **Condutas no Paciente Grave**. São Paulo: 1998.

5. Marte RP; **MARTE Balanças e Aparelhos de Precisão Ltda - Brasil .BRUSHLESS DC-MICROMOTOR** [online] Brasil, Disponível: <http://www.martebal.com.br/minipro.html > Acessado em 26/10/2007 .

6. Martinez, C.A e Filho, J.A; **MÓDULO DIDÁTICO DE ENSAIOS DE TURBINAS DE AÇÃO. Cobenge.** Belo Horizonte MG 2001.

7. Miguel. A.F. **MICROPROCII. MSP430.** [Online] Disponível:< http://www.engcomp.pucpr.br/afonso/Graduacao/MPII/microprocessadoresII.htm > Acessado em 19/11/2007.

8. OLIVEIRA, R. O. **Pediatria** – Ed Blackbook, 2005

9. PEREIRA. F. **Microcontroladores MSP430**: **Teoria e Prática**. 1° ed Érica, 2005

10. Respironics **Virtuoso LX. Manual de instruções de funcionamento**. 1999 Murrysville, Pensnsylvania USA .

11. Respironics. **REMstar | pro. Manual do utilizador**. 2002 Murrysville, Pensnsylvania USA.

12. RODRIGUES, S. L. **Reabilitação Pulmonar: Conceitos Básicos**. 1º ed. São Paulo: Manole, 2003.

13. Xara Webstyle.; **BEI Kimco Magnetics. Motores** [online]. USA; Kimco Magnetics Division, BEI Technologies, Inc. Disponível: <http://www.beikimco.com > acessado em 24/10/2007

14. DataSheet UDN2936W; **ALLEGRO** 3-PHASE BRUSHLESS DC MOTOR CONTROLLER DRIVE, 1985

15. DataSheet MPXV5004G; **FREESCALE SEMICONDUCTOR**; Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated; 2005

16. DataSheet MC33035-D; **ONSEMI SEMICONDUCTOR;** Brushless DC Motor Controller; 2006

17. DataSheet MC33039-D; **ONSEMI SEMICONDUCTOR;** Brushless DC Motor Controller; 2006

18. DataSheet MC33033-D; **ONSEMI SEMICONDUCTOR;** Brushless DC Motor Controller; 2006

19 . DataSheet TTL 74LS04N; Inversora

20. DataSheet DAC0800; **NATIONAL SEMICONDUCTOR;** Conversor D/A

21. DataSheet TTL 74LS145; Decoders BCD to Decimal.

22. DataSheet ECA-MOTOR ; **EBMPAPST** Disponível:
<http://www.ebmpapst.com > acessado em 02/06/2008

ANEXO A

C:\Users\MAU\Desktop\Projeto_Final_-_CPAP_Código_versao_3\Projeto Final - CPAP Código versao 3\main.cpp
Page 1 of 1

```
#include "msp430x44x.h"
#include "adc.h"
#include "lcd.h"
#include "portas.h"
#include "timer.h"
#include "teclado.h"
#include "menu.h"
#include "aplicativo.h"
int main()
{
	WDTCTL = WDTPW+WDTHOLD;
	Init_Portas();
	Init_ADC();
	Init_Timer();
	Init_LCD();
	Init_Menu();
	Init_App();
	__enable_interrupt();
	while(1)
	{
		Trata_Teclado();
		Trata_Menu();
		Trata_Aplicativo();
		Trata_ADC();
	}
}

#include "lcd.h"
void pause()
{
	for(int cnt=0;cnt<50;cnt++);
}
void snd( unsigned char ch )
{
	LCD_DATA = ch;
	pause();
	LCDDISABLE();
	pause();
	LCDENABLE();
	pause();
	LCDDISABLE();
	pause();
}
/*******************************
* private *
*******************************/
static void sndChar( unsigned char ch )
{
	LCD_TXT();
	snd( ch );
}
static void sndCmd( unsigned char ch )
{
	LCD_CMD();
	snd( ch );
}
/*******************************
* public *
*******************************/
void Init_LCD( )
{
	sndCmd( 0x30);
```

```
        sndCmd( 0x30);
        sndCmd( 0x30);
        sndCmd( 0x38);
        sndCmd( 0x8);
        sndCmd( 0x1);
        sndCmd( 0x6);
        sndCmd( 0xC);
        Clear();
}
void Display16x2()
{
        sndCmd( 0x38 );
}
void Clear()
{
        sndCmd( 0x01 );
        sndCmd( 0x03 );
}
void SetPosition(int linha,int coluna)
{
        if( linha > 1 || coluna > 16)
        return;
        sndCmd( 0x80 | (linha * 0x40)+coluna );


}
void ShowCursor()
{
        sndCmd( 0x0D );
}
void HideCursor()
{
        sndCmd( 0x0C );
}
void Print(char* str)
{
        for( int i = 0; str[i] != 0; i++ )
        {
                sndChar( str[i] );
        }
}
void Sprint(char* str)
{
        while(*str) sndChar(*(str++));
}
```

```
C:\Users\MAU\Desktop\Projeto_Final_-_CPAP_Código_versao_3\Projeto Final - CPAP
Código versao 3\aplicativo.c
Page 1 of 4
#include "msp430x44x.h"
#include "aplicativo.h"
#include "adc.h"
#define TAXA_AMOSTAGEM_CONTROLE_1 20
#define TAXA_AMOSTAGEM_CONTROLE_2 4
const unsigned char buff_press[] =
{0x99 ,0x9D ,0xA3 ,0xA7 ,0xAA ,0xAD, 0xB0
,0xB3 ,0xB6 ,0xBA ,0xBD ,0xBF ,0xC1 ,0xC4
,0xCA ,0xCC ,0xFF
};
volatile unsigned int TempoApp;
ST_APP st_app;
ST_APP_RAMPA st_app_rampa;
ST_APP_PRESSAO st_pressao;
void Init_App()
{
        //Inicio do P4OUT
        P4OUT = Retorna_PressTabela(40); //inicia no mínimo
        st_app.pressao = 40;
        st_app.rampa = 0;
        st_app.app = IDLE;
}
```

```
unsigned char Retorna_Press()
{
	return(st_app.pressao);
}
void Aumenta_Press()
{
	if (st_app.pressao != 120)
	st_app.pressao += 5;
}
void Diminui_Press()
{
	if (st_app.pressao != 40)
	st_app.pressao -= 5;
}
void Seta_Press(unsigned char val)
{
	st_app.pressao = val;
}
unsigned char Retorna_Rampa()
{
	return(st_app.rampa);
}
void Aumenta_Rampa()
{
	if (st_app.rampa != 30)
		st_app.rampa += 5;
}
void Diminui_Rampa()
{
	if (st_app.rampa != 0)
		st_app.rampa -= 5;
}
void Seta_Rampa(unsigned char val)
{
	st_app.rampa = val;
}
void Atua_Pressao()
{
	P4OUT = Retorna_PressTabela(Retorna_Press());
}
void Controle_Pressao(unsigned char val)
{
	P4OUT = val;
}
unsigned char Retorna_PressTabela(unsigned char val)
{
	val = val - 40;
	val = val / 5;
	val = buff_press[val];
	return(val);
}
void Trata_Aplicativo()
{
	unsigned char temppressao;
	if (TempoApp) return;
		switch(st_app.app)
		{
			case IDLE:
				return;
			case RAMPA:
if (st_app_rampa.pressatual == st_app_rampa.presslimite) //acabou , chegou no
destino
{
	st_app.app = IDLE;
}
else
{
	TempoApp = st_app_rampa.reload;
	P4OUT = st_app_rampa.pressatual++;
}
break;
case PRESSAO:
```

```
        TempoApp = TAXA_AMOSTAGEM_CONTROLE_2;
        tempppressao = Converte_Sensor();
        if ((tempppressao < Retorna_Press()) && (st_pressao.pressaofisica <
        Retorna_PressTabela(120)))


        {
                st_pressao.pressaofisica++;
        }
        else if ((tempppressao > Retorna_Press()) && (st_pressao.pressaofisica >
Retorna_PressTabela(40)))
        {
                st_pressao.pressaofisica--;
        }
        P4OUT = st_pressao.pressaofisica;
        break;
        }
}

void Inicia_App(EN_APP tipo)
{
        unsigned char presmin;
        unsigned char presmax;
        unsigned int rampa;
        unsigned int dif;
        st_app.app = tipo;
        switch (st_app.app)
        {
                case RAMPA: //inicia operação com RAMPA
                presmin = Retorna_PressTabela(40); //retorna valor para 40cm
                presmax = Retorna_PressTabela(Retorna_Press());//retorna valor
desejável
                rampa = Retorna_Rampa();
                dif = presmax - presmin;
                if (dif == 0) //já está no limite mínimo
                {
                        st_app.app = IDLE; //retorna para IDLE
                        return;
                }
                rampa = rampa * 600;
                dif = rampa / dif; //dif recebe o tempo de atuação da turbina
                st_app_rampa.reload = dif;
                st_app_rampa.pressatual = presmin;
                st_app_rampa.presslimite = presmax;
                TempoApp = st_app_rampa.reload;
                break;
        case PRESSAO:
                //deixa a função trata_app no pooling cuidar
                st_pressao.pressaofisica = Retorna_PressTabela(Retorna_Press());
                TempoApp = TAXA_AMOSTAGEM_CONTROLE_1;
                Atua_Pressao();
                break;
        }
}
unsigned char App_Idle(void)
{


        if (st_app.app == IDLE)
                return(1);
        else
                return(0);
}
unsigned char Converte_Sensor()
{
        unsigned long temp;
        unsigned int press;
/////////**função não linearizada
/* temp = (unsigned long)Retorna_MediaAD();
temp = temp * 300;
temp = temp / 4096;
if (temp < 100) temp = 0;
```

```
else
temp -= 100;
press = (unsigned int)temp;
return(press);
*/
#define OFFSET 171
       temp = (unsigned long)Retorna_MediaAD();
       temp = temp * 340;
       temp = temp / 4096;
       if (temp < OFFSET) temp = 0;
       else
       temp -= OFFSET;
       press = (unsigned int)temp;
       return(press);
}


#include "msp430x44x.h"
#include "menu.h"
#include "lcd.h"
#include "timer.h"
#include "teclado.h"
#include "aplicativo.h"
#include "misc.h"
#include "mnu_config.h"
#include "mnu_status.h"
ST_MENU st_menu;
char buffmenu[16];
void Init_Menu()
{
       st_menu.passo = 0;
       st_menu.estado = INICIO;
       st_menu.tempomenu = 0;
}
void Trata_Menu()
{
       if (st_menu.tempomenu)


#include "msp430x44x.h"
#include "misc.h"
unsigned char temp;
void Convert_uchar_4digvirg(unsigned char val,char *pont)
{
       temp = val / 100;
       pont[0] = temp + 48;
       val = val - (temp*100);
       temp = val / 10;
       pont[1] = temp + 48;
       val = val- (temp*10);
       pont[2] = ',';
       pont[3] = val + 48;
       pont[4] = 0;
}
void Convert_uchar_2dig(unsigned char val, char *pont)
{
       temp = val / 10;
       pont[0] = temp + 48;
       val = val - (temp*10);
       pont[1] = val + 48;
       pont[2] = 0;
}


#include "msp430x44x.h"
#include "menu.h"
#include "lcd.h"
#include "timer.h"
#include "teclado.h"
#include "aplicativo.h"
#include "misc.h"
#include "mnu_config.h"
```

```
void Menu_Configuracao()
{
	static unsigned char press_temp;
	static unsigned char rampa_temp;
	switch (st_menu.passo)
	{
	case 0: //salva as variáveis
		press_temp = Retorna_Press();
		rampa_temp = Retorna_Rampa();
		st_menu.passo++; //não precisa de break;
	case 1:
		Clear();
		Print(" Pressure: ");
		SetPosition( 1, 0 );
		Print(" cm");
		SetPosition( 1, 1 );
		Convert_uchar_4digvirg(Retorna_Press(),buffmenu);
		Print(buffmenu);
		st_menu.passo++;
	case 2:
		if (Verifica_Teclado() == 0)
			return;
		else
		{
			switch(st_teclado.tecla)
		{
		case TEC1://esquerda
		Diminui_Press();
		// Atua_Pressao();
		st_menu.passo = 1;
		break;
	case TEC2: //direita
		Aumenta_Press();
		// Atua_Pressao();
		st_menu.passo = 1;
		break;
		case TEC3: //enter
		st_menu.passo++;
		break;
	case TEC4: //esc
		Seta_Press(press_temp); //retorna o valor anterior
		st_menu.passo++;
		break;
	}
}
break;
	case 3:
		Clear();
		Print(" Rampa: ");
		SetPosition( 1, 0 );


		Print(" min");
		SetPosition( 1, 1);
		Convert_uchar_2dig(Retorna_Rampa(),buffmenu);
		Print(buffmenu);
		st_menu.passo++;
		break;
	case 4:
		if (Verifica_Teclado() == 0)
			return;
	else
	{
		switch(st_teclado.tecla)
	{
	case TEC1://esquerda
		Diminui_Rampa();
		st_menu.passo = 3;
		break;
	case TEC2: //direita
		Aumenta_Rampa();
		st_menu.passo = 3;
```

```
                break;
        case TEC3: //enter
        Muda_Menu(PRINCIPAL,0);
        break;
        case TEC4: //esc
        Seta_Rampa(rampa_temp); //retorna o valor anterior
        Muda_Menu(PRINCIPAL,0);
        break;
        }
      }
   }
}


#include "msp430x44x.h"
#include "menu.h"
#include "lcd.h"
#include "timer.h"
#include "teclado.h"
#include "aplicativo.h"
#include "misc.h"
#include "mnu_status.h"
#define TEMPO_MOSTRASENSOR 200
void Menu_Status()
{
        unsigned int press;
        switch (st_menu.passo)
        {
        case 0:
                Clear();
                Print("Pressure cm");
                SetPosition( 0, 9 );
                Convert_uchar_4digvirg(Retorna_Press(),buffmenu);
                Print(buffmenu);
                st_menu.passo++;
        case 1:
                if (Verifica_Teclado() == 0)//caso não tenha apertado nada
                {
                        SetPosition(1,0);
                        Print("Sensor: cm");
                        press = Converte_Sensor();
                        SetPosition( 1, 8 );
                        Convert_uchar_4digvirg(press,buffmenu);
                        Print(buffmenu);
                        st_menu.tempomenu = TEMPO_MOSTRASENSOR;
                        break;
                }
                else
                {
                        switch(st_teclado.tecla)
                {
        case TEC1://esquerda
                st_menu.passo++;
                break;
        case TEC2: //direita - muda menu
                st_menu.passo++;
                break;
        case TEC3: //enter
                Inicia_App(PRESSAO);
                break;
        case TEC4: //esc
                Muda_Menu(PRINCIPAL,2);
                break;
      }
    }
break;
        case 2:
                Clear();
                Print("Rampa: min");
                SetPosition(0, 8 );
                Convert_uchar_2dig(Retorna_Rampa(),buffmenu);
```

```
                Print(buffmenu);
                st_menu.passo++;
        case 3:
                if (Verifica_Teclado() == 0)//caso não tenha apertado nada
                {
                        SetPosition(1,0);
                        Print("Sensor: cm");
                        press = Converte_Sensor();
                        SetPosition( 1, 8 );
                        Convert_uchar_4digvirg(press,buffmenu);
                        Print(buffmenu);
                        st_menu.tempomenu = TEMPO_MOSTRASENSOR; //travo o menu por 2
segundos
                        break;
                }
                else
                {
                        switch(st_teclado.tecla)
                        {
                                case TEC1://esquerda
                                        st_menu.passo = 0;
                                        break;
                                case TEC2: //direita - muda menu
                                        st_menu.passo = 0;
                                        break;
                                case TEC3: //enter
                                        Inicia_App(RAMPA);
                                        break;
                                case TEC4: //esc
                                        Muda_Menu(PRINCIPAL,2);
                                        break;
                        }
                }
                break;
        }
}
```

```
#include "msp430x44x.h"
#include "portas.h"
void Init_Portas()
{
        P1DIR = 0x00;
        P2DIR = 0xFF; //configura P2 para modo saida
        P2OUT = 0; // Zera a saida da porta P2
        P3DIR = 0xF0;
        P3OUT = 0x0F;
        P4DIR = 0xFF;
        P5DIR = 0xFF;
        P6SEL |= 0x01; // Enable A/D channel A0
        P6DIR = 0x00;
        //P6SEL = 0x01; //P6SEL = 0x03;
}
```

```
#include "msp430x44x.h"
#include "teclado.h"
#define TEMPORAPIDO1 6
#define TEMPORAPIDO2 2
#define DBTECLADO 100
volatile unsigned char ucDBTeclado;
ST_TECLADO st_teclado;
void Init_Teclado()
{
        P3OUT |= 0x1F;
        ucDBTeclado = DBTECLADO;
}
void Trata_Teclado()
```

```
{
	unsigned char botao;
	static unsigned char ucTempoRapido;
	static unsigned char ucFlagTempoRapido;
	if (ucDBTeclado) return;
		ucDBTeclado = DBTECLADO;
	botao = P3IN & tecmask; //verifica somente as teclas em questão
	if (botao == tecmask) //caso estejam todas soltas
	{
		st_teclado.status = SOLTA;
	}
	else if (st_teclado.status == SOLTA) //caso estivesse solta //caso alguma
tecla esteja pressionada
	{
		ucTempoRapido = 0;
		ucFlagTempoRapido = 0;
		st_teclado.novatecla = SIM; //indica nova tecla pressionada
		st_teclado.status = APERTADA;
		switch((EN_TECLA)botao)
		{
			case TEC1:
			case TEC2:
			case TEC3:
			case TEC4:
				st_teclado.tecla = (EN_TECLA)botao;
				break;
			default: //aberto 2 teclas ao mesmo tempo
//não faz nada
			break;
		}
		asm(" NOP");
	}
	else //a tecla ainda está pressionada
	{
		ucTempoRapido++;
		if (ucFlagTempoRapido == 0) //não foi o tempo rápido
		{
		if (ucTempoRapido == TEMPORAPIDO1)
		{
			ucTempoRapido = 0;
			st_teclado.novatecla = SIM; //indica nova tecla pressionada


			ucFlagTempoRapido = 1;
		}
	}
	else
	{
		if (ucTempoRapido == TEMPORAPIDO2)
		{
			ucTempoRapido = 0;
			st_teclado.novatecla = SIM; //indica nova tecla pressionada
		}
	}
   }
}
unsigned char Verifica_Teclado()
{
	if ((st_teclado.status == APERTADA) && (st_teclado.novatecla == SIM))
	{
		st_teclado.novatecla = NAO;
		return(1);
	}
	return(0);
}


#include "msp430x44x.h"
#include "timer.h"
#include "teclado.h"
#include "menu.h"
#include "aplicativo.h"
```

```
#include "adc.h"
#define PRESCALER_APP 100
unsigned char prescaler_app;
//-------------------------------------------------------------
#pragma vector=TIMERA1_VECTOR //interrupção chamada a cada 400us
__interrupt void isrTimerA(void) //chamada a cada 1ms
{
	TACTL &= ~TAIFG; //Desabilita Interrupção pendente
	if (st_menu.tempomenu) st_menu.tempomenu--;
	if (ucDBTeclado) ucDBTeclado--;
	if (ucTimerAD) ucTimerAD--;
	if (prescaler_app)
{
prescaler_app--;
}
else
{
	prescaler_app = PRESCALER_APP;
	if (TempoApp) TempoApp--;
}
}
void Init_Timer()
{
	TACTL = (TASSEL_1+TAIE); //Aclk (32768Hz) para TimerA, habilita interrupção
	TAR = 0x00; //Zera contador;
	TACCR0 = 0x0021;
	TACTL |= MC_1; //Inicia contagem
}
```