ESCOLA DE EDUCAÇÃO PROFISSIONAL SENAI PORTO ALEGRE
CURSO TÉCNICO EM ELETRÔNICA

MARCOS VINICIUS DA SILVA ROGOWSKI
YURI ANDRADE RODRIGUES

Display de Varrimento Mecânico
"The Propeller Clock"

Monografia apresentada como requisito parcial para a obtenção da habilitação plena em Técnico em Eletrônica.

Prof. Flavio Rocha de Avila
Orientador

Porto Alegre, Julho de 2009.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

ROGOWSKI, Marcos Vinicius da Silva; RODRIGUES, Yuri Andrade.

Display de Varrimento Mecânico: the propeller clock / Marcos Vinicius da Silva Rogowski; Yuri Andrade Rodrigues – Porto Alegre: SENAI / Escola de Educação Profissional de Porto Alegre / Curso Técnico em Eletrônica, 2009.

44 f.: il.

Monografia (curso técnico) – Serviço Nacional de Aprendizagem Industrial. Escola de Educação Profissional Porto Alegre. Curso Técnico em Eletrônica Industrial. Porto Alegre, 2005. Orientador: Avila, Flávio Rocha de.

1. Display de varrimento mecânico. I. Avila, Flávio Rocha de.

ESCOLA DE EDUCAÇÃO PROFISSIONAL SENAI PORTO ALEGRE
Diretor: Prof. Msc. Clóvis Leopoldo Reichert
Coordenador: Alexandre Gaspary Haupt
Orientadora Pedagógica: Dione Danesi Gallo de Araújo
Psicóloga: Marília Marques
Bibliotecária: Patrícia Redel Nunes Teixeira

AGRADECIMENTOS

Primeiramente, dedicamos este trabalho aos nossos pais, pois sem eles nada disso seria possível. Agradecemos ao nosso professor e orientador, Flavio Rocha de Avila, por toda paciência, ajuda e todo o conhecimento compartilhado conosco. E agradecemos também ao companheirismo, amizade e pelos momentos de descontração compartilhados com todos os nossos colegas e professores durante este período de formação profissional, momentos especiais que nos motivaram a seguir em frente, e que certamente guardaremos e lembrarem com bastante orgulho pelo resto de nossas vidas.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

RT	Recursos Técnicos
RH	Recursos Humanos
LED	*Light Emitting Diode* (Diodo Emissor de Luz)
IC	Circuito Integrado
AC	*Alternate Current* (Corrente alternada)
DC	*Direct Current* (Corrente direta / contínua)
RTC	*Real Time Clock* (Relógio em Tempo Real)
PCI	Placa de Circuito Impresso
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado).
FPS	*Frames per Second*
VCR	*Video Cassete Recorder*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este é um dispositivo diferenciado, pois, diferentemente dos relógios convencionais, neste não existem ponteiros reais nem painéis estáticos.

Como o próprio nome deixa subentendido, "*The Propeller Clock*" é um relógio constituído de uma hélice acoplada ao eixo de um motor. A interface deste é efetivada através de uma barra de *LEDs* que se encontra acoplada a esta hélice.

Utilizando os princípios de varrimento mecânico, a barra de *LEDs* é previamente programada para que, em determinados momentos já pré-estabelecidos, pisquem, provocando aos observadores a ilusão de imagens serem exibidas no ar. Estes são conhecidos também como *Displays* Aéreos.

Especificamente o "*The Propeller Clock*", consiste em um *RTC* – implementado por *software* – que exibe, no ar, a imagem de um pseudo-relógio analógico.

Palavras-Chave: Relógio; hélice; imagens no ar; ilusão de óptica; varrimento mecânico.

Mechanically Scanned Display

ABSTRACT

This is a differential device, therefore, unlike the conventional clocks, it doesn't have real pointers or static panels.

As the name leaves understood, "The Propeller Clock" is a clock consisting of a propeller attached to the axis of an engine. The interface of this is done through a LED bar which is attached to the propeller.

Using the principles of mechanical scanning, the bar of LEDs is already programmed for, in the pre-set times, flashes, causing to the observer an illusion of images displayed in the air. These are also known as Air Displays.

Specifically "The Propeller Clock", consists in a RTC - implemented in software - that displays in the air, pseudo-image of an analog clock.

Keywords: Air display; propeller; mechanical scanning; illusion; images in the air.

1 INTRODUÇÃO

Como o próprio nome sugere, "*The Propeller Clock*" é um relógio constituído de uma hélice acoplada a um motor. Nesta hélice existe uma barra de *LEDs* que será responsável pela interface do dispositivo. Para obter essa interface, utilizam-se os conceitos de "*Air Displays"*.

Os *displays* aéreos são dispositivos que visam efetuar uma interface sem utilizarem periféricos. Acoplados à hélice, existem *LEDs* tendo seu acionamento controlado enquanto o motor gira, causando a ilusão de que mensagens estão sendo escritas no ar.

1.1 Objetivos Gerais

Este projeto tem como objetivo principal demonstrar a capacidade do grupo em integrar todos os conceitos e conhecimentos adquiridos no decorrer do curso e obter a habilitação plena em Técnico em Eletrônica.

1.2 Objetivos Específicos

Este visa pôr em prática todos os conhecimentos adquiridos durante o Curso Técnico em Eletrônica, integrando os conceitos absorvidos nas unidades curriculares. Estas que por sua vez são: Análise de Circuitos Elétricos, Processos de Qualidade, Manutenção de Circuitos Eletrônicos Analógicos, Expressões Gráficas, Manutenção Eletrônica, Sistemas de Controle, Automação Industrial e Integração Eletrônica.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

Após a escolha do projeto, foram feitas, pelo grupo, pesquisas buscando qualquer tipo de informações conceituais e práticas que apontem para o dispositivo alvo, para que essas possam ser utilizadas como referência ou contribuir para a fundamentação do mesmo. A partir desta, foi tomada ciência de todos os fenômenos envolvidos e a partir destes puderem ser tomadas decisões sobre a utilização de atuadores, acionamento das cargas, topologia de controle e dimensões mecânicas.

2.2 Descrição

2.2.1 Ilusão de Óptica

O termo Ilusão de Óptica aplica-se a todos os fenômenos que tem a capacidade de distorcer a resposta do sistema visual humano, sendo capaz de induzir o ser a ver algo que não está fisicamente presente, ou simplesmente alterar a forma de percepção visual. Estas podem assumir caráter fisiológico ou até mesmo cognitivo, surgindo naturalmente ou sendo induzidas.

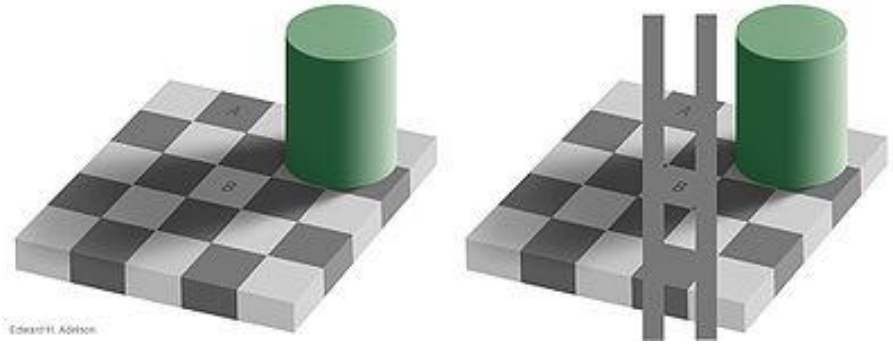
As cores e as formas usuais sobre o que pode ser visto, surge instantaneamente nos circuitos neurais dos seres humanos e influenciam na representação de alguma cena. As propriedades percebidas dos objetos, tais como o brilho, tamanho angular, e cor, são determinadas inconscientemente e não são propriedades físicas reais. A interpretação do que o ser humano vê no mundo exterior é uma tarefa muito complexa.

Estudos mostram que existem mais de trinta áreas diferentes de cérebro humano voltadas a reproduzir e processar as imagens do mundo exterior. Algumas áreas parecem corresponder ao movimento, outras à cor, outras à profundidade e mesmo à direção de um contorno. Existe também uma tendência do cérebro humano a simplificar as coisas em relação ao que elas realmente são. E é essa simplificação, que permite ao ser humano uma apreensão mais rápida - ainda que imperfeita - da realidade exterior, o que dá origem às ilusões de óptica.

2.2.1.1 *Ilusão de Luminosidade*

A luminosidade é uma variável subjetiva que não corresponde de um modo preciso a uma quantidade física. É uma estimativa da proporção de luz incidente que é refletida por uma superfície, feita pelo sistema visual.

Figura 1 – Ilusão de Óptica



Como ilustra a Figura 1, percebe-se o quadrado A como sendo mais escuro do que o quadrado B. No entanto, como se vê pela figura da direita, em que simplesmente se adicionou duas barras com a mesma cor de A, ambos têm exatamente a mesma cor - têm a mesma quantidade de luz visível que chega ao olho vindo da superfície.

O que se passa é que o sistema visual não se limita a medir a quantidade de luz que chega ao olho, que é influenciada pelas sombras. Parece ter em conta o contraste local e saber que as mudanças de luz na transição entre superfícies de cores diferentes são geralmente mais abruptas do que as causadas por sombras. O sistema visual, sabiamente utiliza apenas a informação sobre as transições mais abruptas para construir uma imagem. E por isso estima a cor dos objetos sem se deixar enganar pelas sombras de um objeto visível. (WIKIPEDIA, 2009).

2.2.2 Cadência

Cadência, ou *Frame Rate,* é a medida da freqüência em que um dispositivo de processamento de imagens produz consecutivos quadros – ou *Frames*. O termo se aplica igualmente para gráficos de computador, vídeo câmeras, sistemas de captura de movimento e retro-projetores, sendo este o conceito dos projetores de cinema.

A unidade de medida de cadência é em quadros por segundo – *FPS*.

Durante um instante de tempo, seja qual for o dispositivo de saída das imagens, este apresenta uma série de imagens estáticas, que devido ao curto intervalo entra as imagens, causa a ilusão de movimento da imagem.

A visão humana é capaz de captar até vinte quadros por segundo, aproximadamente. Acima disto, o olho humano não é capaz de distinguir se a imagem é realmente estática, ou se ela realmente está em movimento. (WIKIPEDIA. 2009).

2.3 Estado da Arte

A idéia de desenvolver um relógio utilizando os conceitos de escrita por varrimento mecânico foi originada por Bob Blick. Foi em 1995 que Blick desenvolveu sem primeiro relógio, utilizando um *array* de sete *LEDs* acoplados a um motor de *VCR*. Blick batizou sua criação de "*The Propeller Clock*".

Figura 2 – "*The Propeller Clock*": Em funcionamento.

Figura 3 – "*The Propeller Clock*": Vista lateral.

Depois de Blick, muitos outros implementaram outros relógios. Muitas vezes utilizando topologias distintas, mas sempre referenciando a criação de Blick.

Atualmente é possível observar muitas ramificações deste dispositivo. Em sua maioria, são aplicações para entretenimento, mas utilizadas de maneiras bastante diversificadas.

Figura 4 – Desenho Utilizando Varrimento Mecânico.

Figura 5 – Desenhos Utilizando Varrimento Mecânico.

3 METODOLOGIA

3.1 Introdução

Para facilitar o desenvolvimento do projeto e garantir a fidelidade nos detalhes do mesmo, foi desenvolvido um esquema organizacional de trabalho, abordando todas as características do projeto, desde suas definições à sua integração.

Buscando facilitar o gerenciamento e a execução do projeto, o mesmo foi dividido em cinco etapas, cada qual com suas atividades subordinadas.

3.2 Descrição das Etapas

3.2.1 Etapa A → Fundamentação teórica

Esta é a principal etapa do projeto, pois ela consiste em ditar suas características gerais, sendo premissa fundamental para o desenvolvimento do mesmo.

1.Definições do Projeto: → ***RH***: *Marcos Vinicius; Yuri*

1.1 Escolha do projeto.

1.2 Definições de funcionamento.

Meta → Definir projeto.

Indicador de Desempenho → Projeto escolhido.

2.Pesquisa de Referências:

2.1 Busca por *Samples*.

2.1.1 Buscar *Samples* de *Hardware*. → ***RH***: *Yuri*

2.1.2 Buscar *Samples* de *Firmware*. → ***RH:*** *Marcos Vinicius*

Meta → Buscar projetos exemplos.

Indicador de Desempenho → Exemplos de aplicações encontrados.

RT → Livros; Internet.

3.Fundamentação Teórica: → ***RH***: *Marcos Vinicius*

3.1 Descrição teórica do projeto.

3.2 Referências Bibliográficas.

Meta → Definição teórica do projeto.
Indicador de Desempenho → Projeto fundamentado.
RT → Livros; Internet.

3.2.2 Etapa B → Definições de *Hardware*

Nesta etapa será definida a plataforma de *hardware*. Desde sua parte mecânica e sua parte eletrônica.

1.Definição de Plataforma de *Hardware*: → ***RH***: Marcos Vinicius; Yuri
1.1 Parte mecânica.
1.1.1 Definir alocação do *hardware* eletrônico.
1.1.2 Definir dimensões.
1.2 Parte eletrônica.
1.2.1 Definição esquema elétrico.
1.2.1.1 Desenvolver esquema elétrico. → ***RH***: Yuri
1.2.1.2 Depurar esquema elétrico. → ***RH***: Marcos Vinicius
Meta → Projetar uma plataforma de hardware.
Indicador de Desempenho → Plataforma de hardware definida.

2.Montagem de *Hardware*: → ***RH***: Yuri
2.1 Parte mecânica.
2.1.1 Adquirir componentes.
2.1.2 Montar plataforma.
2.1.3 Depurar plataforma. → ***RH***: Marcos Vinicius
2.2 Parte eletrônica.
2.2.1 Adquirir componentes.
2.2.2 Implementação de esquema elétrico.
2.2.2.1 Montagem em placa de protótipos.
2.2.2.1.1 Montar em *Protoboard*.
2.2.2.1.2 Depurar montagem. → ***RH***: Marcos Vinicius
2.2.2.2 Montagem em placa dedicada.
2.2.2.2.1 Desenhar *layout* da PCI.
2.2.2.2.2 Confeccionar PCI.
2.2.2.2.3 Depurar PCI. → ***RH***: Marcos Vinicius
Meta → Montar as plataformas mecânicas e eletrônicas.
Indicador de Desempenho → Plataforma montadas.
RT → Componentes eletrônicos passivos e discretos; micro controlador; *protoboard;* placa de fibra de vidro.

3.Integração de Plataformas. → ***RH***: Yuri
3.1 Integrar plataforma mecânica e eletrônica.
3.2 Depurar integração. → ***RH***: Marcos Vinicius

Meta → Integrar plataformas.
Indicador de Desempenho → Plataformas unificadas e funcionais.

3.2.3 Etapa C → Definições de *Firmware*

Está é a etapa que será responsável pelo desenvolvimento do controle do sistema, este que será controlado através de um *software* dedicado. Sendo assim, esta etapa se torna a mais importante do projeto.

1.Definições de Controle. → ***RH***: Marcos Vinicius; Yuri
1.1 Topologia de controle.
1.1.1 Desenvolver fluxo de programa. → ***RH***: Marcos Vinicius
1.1.2 Depurar Fluxograma. → ***RH***: Yuri
Meta → Definir topologia de controle.
Indicador de Desempenho → Fluxo de programa efetuado.

2.Desenvolvimento de *Firmware*. → ***RH***: Marcos Vinicius; Yuri
2.1 Definição de ferramentas.
2.1.1 Definir Linguagem de Programação.
2.1.1.1 Definir técnica de programação.
2.2 Desenvolvimento do programa. → ***RH***: Marcos Vinicius
2.2.1 Transcrever fluxograma.
2.3 Depuração do programa. → ***RH***: Yuri
Meta → Desenvolver programa para controle do sistema.
Indicador de Desempenho → Programa concluído e funcional.
RT → Compilador.

3.2.4 Etapa D → Integração de *Hardware* e *Software*

Esta etapa caracteriza-se pela integração das duas partes do sistema: a parte do hardware e a parte do software. Nela serão efetuados os testes finais e a funcionalidade do sistema.

1.Definição de ferramentas. → ***RH***: Marcos Vinicius; Yuri
1.1 Definir *software* para gravação.
1.2 Definir *hardware* para gravação.
1.3 Definir plataforma de *debug*.
Meta → Definir plataforma para *debug* e gravação.
Indicador de Desempenho → Plataforma para *debug* e gravação escolhida.

2.*Debug* do programa. → ***RH***: Marcos Vinicius

2.1 Gravar programa.
2.2 Depurar programa.
Meta → Testar e retificar programa.
Indicador de Desempenho → Programa funcional.
RT → Estação de Desenvolvimento CUSCOPIC.

3.Integração *Hardware* e *Software*. → ***RH***: Marcos Vinicius
3.1 Integrar *Hardware* e *Software*.
3.2 Depurar integração. → ***RH***: Yuri
Meta → Integrar *Hardware* e *Software*.

3.3 Cronograma

Para desenvolvimento do projeto, utilizou-se este cronograma como referência de tempo para execução das atividades supra referidas.

Tabela 3.1 - Cronograma

Etapa	Título	Atividade	Prazo
1	Proposta de TCC	Definição do tema; Cronograma; Apresentação da Proposta	20/03
2	Trabalho Teórico	Pesquisa Teórica	09/04
3	Projeto	Elaboração do Projeto	24/04
4	Execução	Executar Projeto	15/05
5	Finalização	Resultados Obtidos	29/05
6	Conclusão	Elaboração da Conclusão	15/06
7	Entrega	Entrega TCC	19/06
8	Seminário	Seminário de Andamento	22/06; 29/06
9	Apresentação	Apresentação Final	07/07; 09/07

4 DESENVOLVIMENTO

4.1 Mecânica

4.1.1 Introdução

Integrando todos os conceitos e valores adquiridos no Curso Técnico em Eletrônica, optou-se pela utilização de materiais alternativos como base de implementação do projeto. Entenda-se por material alternativo, todo material que atualmente esteja inutilizado e, sendo verídica a possibilidade de reciclagem, seja efetivada sua reutilização.

Esta é uma alternativa de baixo custo e, principalmente, ecologicamente correta.

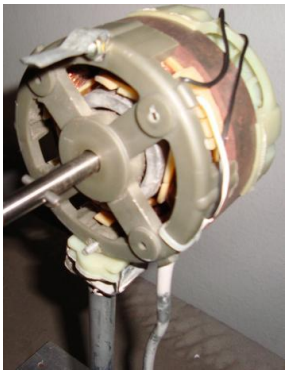
4.1.2 Descrição das Partes

4.1.2.1 *Atuador*

Para efetuar a rotação de nosso dispositivo, é necessária a utilização de um motor como atuador. Este que foi retirado de um antigo ventilador que se encontrava fora de uso.

O atuador alvo – cujas especificações técnicas estão apresentadas na Tabela XX – opera com corrente alternada, facilitando o seu acionamento.

Figura 6 – Motor Atuador



4.1.2.2 *Base*

Primeiramente, tentou-se utilizar a base original do ventilador. Logo no primeiro teste, verificou-se que mesma não estava adequadamente dimensionada para a aplicação alvo, fazendo com que fosse necessária a confecção de uma nova base.

A base original do atuador mostrou-se extremamente leve, fazendo com que houvesse um grande deslocamento do dispositivo. Para retificação deste problema, confeccionou-se uma base firme e de massa consideravelmente maior que a original, fixando uma haste a uma chapa, ambas de aço, e o motor em extremidade superior.

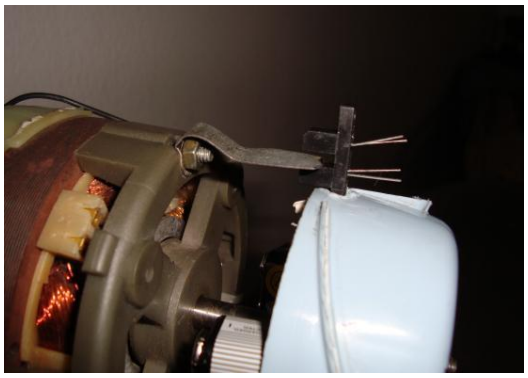
Figura 7 – Base Mecânica

4.1.2.3 *Sensor de referência*

Para obtenção de uma referência externa para o *firmware*, foi utilizada uma chave óptica (ITR8102) no centro geométrico da estrutura. Esta chave é constituída de um IR LED em frente a um *phototransistor*, ambos dentro de um encapsulamento plástico para

que sejam evitadas interferências externas. Conforme ilustra a Figura XX, existe uma haste metálica fia na estrutura responsável por interromper o feixe de luz emitido pelo sensor.

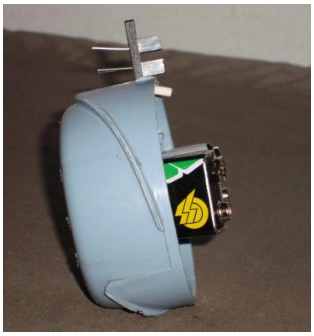
Figura 8 – Chave Óptica



4.1.2.4 *Instalação das baterias de alimentação*

Para a alimentação do hardware, foram utilizadas duas baterias 9v. Para que as mesmas pudessem ser acopladas à estrutura mecânica sem influenciar no balanceamento da "hélice", elas foram alocadas, simetricamente, uma em cada extremidade da mesma estrutura de plástico na qual o sensor é acoplado – conforme ilustra a Figura XX. Dessa forma, mantém-se o balanceamento da estrutura na hora da rotação.

Figura 9 – Instalação de Baterias



4.1.2.5 *Eixo*

Para fixação do *Hardware* no eixo do motor, foi utilizado o "miolo" da antiga hélice do mesmo. É neste mesmo "miolo" que são acoplados o sensor e as baterias.

Figura 10 – Fixação Inferior

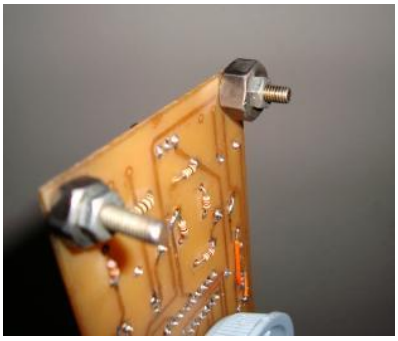
4.1.2.6 *Geometria e Balanceamento*

Após completar a montagem e instalação dos periféricos, foi efetuado o teste de balanceamento do mecanismo, para possibilitar que sejam efetuados os requeridos ajustes.

É extremamente necessário que uma minuciosa calibração seja efetuada, pois qualquer desvio no balanceamento do mecanismo gera uma considerável distorção na imagem.

Para efetuar esta correção, foram utilizados pequenos elementos roscados (parafusos e porcas) nas extremidades da nossa base de *hardware*, como ilustrado na Figura XX.

Figura 11 – Balanceamento



4.1.3 Lista de Peças

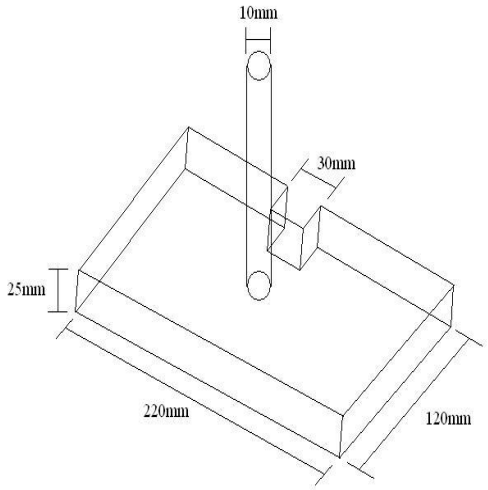
Tabela 4.1 – Lista de Peças Mecânicas

Peça	Quantidade	Descrição	Função	Valor
Motor AC	1 unidade	Tensão: 127 v Corrente: 600 mA Potencia: 75 W	Rotacionar Hardware	Reaproveitado
Tampa	1 unidade	Diâmetro: 50 mm	Suporte da placa com o eixo	Reaproveitado
Suporte das baterias	1 unidade	Espessura: 30 mm Diâmetro: 75 mm	Suporte das baterias no eixo	Reaproveitado
Cilindro	1 unidade	Diâmetro: 10 mm Comprimento: 200 mm	Segurar o motor fixando-o na base	R$ 5,00
Base de Aço	1 unidade	Largura: 220 mm Comprimento: 120 mm Altura: 25 mm Massa: 4,5 kg	Esta base serve como peso para o projeto não se deslocas quando estiver em funcionamento.	R$ 10,00
			Total:	R$ 15,00

4.1.4 Desenho Mecânico

A figura a seguir ilustra a base de sustentação do motor atuador.

Figura 12 – Dimensões Mecânicas Suporte



4.2 Eletrônica

Devido às propriedades do projeto, não foi possível utilizar uma plataforma de *hardware* pronta, fazendo com que fosse necessário o desenvolvimento de uma plataforma de *hardware* específica.

4.2.1 Hardware Desenvolvido

4.2.1.1 *Introdução*

Sem a ciência de alguma plataforma de hardware pronta para desenvolvimento do projeto, optou-se por projetar uma plataforma de *hardware* específica, projetando desde seu esquema elétrico à sua placa manufaturada.

4.2.1.2 *Descrição*

4.2.1.2.1. *Placa de Circuito Impresso*

Após a confecção do esquema elétrico (vide anexo), através do *software* para desenvolvimento de PCI, *Cadsoft Eagle 5.3.0*, foi efetuado o roteamento da PCI utilizando uma placa de face dupla.

Depois de impressas as trilhas em folha tipo Ofício A4, as mesmas foram recortadas nas devidas proporções – 200x60mm – as mesmas foram coladas à placa e levadas à prensa, fazendo com que após dois minutos, a placa ficasse com os desenhos das trilhas em suas respectivas faces. Após, a placa é imersa em solução de Percloreto de Ferro ($FeCl_3$), para efetuar a corrosão do cobre nas partes onde não existem trilhas desenhadas. Após a corrosão, é efetuada a furação e corte da placa, ficando em condições de soldagem.

4.2.1.2.2. *Acionamento de Motores*

Foi utilizado apenas um motor para a confecção deste projeto, este que é responsável pelo movimento do *display*.

Seu acionamento é feito diretamente pela rede elétrica – atendendo às suas especificações – e não é efetuado nenhum tipo de controle sobre o atuador. Todo o controle do projeto é através de *software*.

4.2.1.2.3. *Acionamento de Cargas*

As cargas a serem acionadas pelo nosso circuito de micro-controlado – utilizando PIC16F628A – são 12 (doze) *LEDs* de 5mm, vermelhos, de alto brilho, que se encontram alinhados.

Para o acionamento das cargas foram utilizados dois *arrays* de *transistores* (ULN2003). Estes foram utilizados com o intuito de drenar pouca corrente dos pinos do micro-controlador, evitando assim uma possível reinicialização do dispositivo.

Por motivos de segurança, foram utilizadas duas baterias como fontes de alimentação.

4.2.1.2.4. *Sensores*

Para obter um ponto de referência para o controle do dispositivo, é necessário um componente para o sensoriamento do motor. Primeiramente, foi implementado um sensor óptico de reflexão. O sensor era constituído de um IR LED e um *Transistor* fotossensível, acoplados no eixo do motor. Em determinada parte do corpo do motor, havia um espelho que refletia o sinal do LED e saturava o *phototransistor*. Após alguns testes, foi verificado que este sensor era suscetível a diversas interferências externas, não existindo fidelidade na resposta de sinal do mesmo. Para solucionar este problema, foi utilizada uma chave-óptica, cuja função é a igual a do sensor óptico por reflexão, e seu encapsulamento (Figura 4.1.1.3.1) prevê interferências externas, fazendo com que a mesma não seja suscetível a interferências. Seu código é ITR8102 (Figura 4.1.1.3.2). Sua instalação foi efetuada no eixo do motor através de um suporte plástico, tendo uma haste (Figura XX) - fixada no corpo do motor - gerando sua interrupção, conforme ilustra a Figura XX.

Figura 13 – Dimensões Mecânicas do Sensor infravermelho

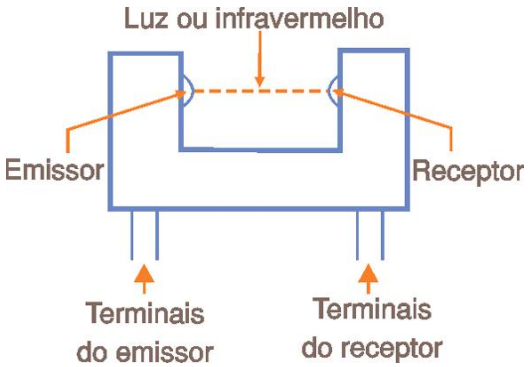
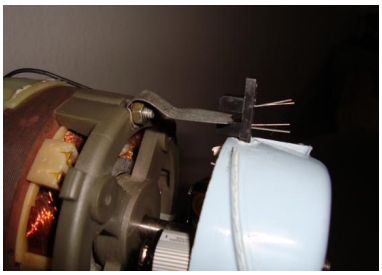


Figura 14 – Diagrama Elétrico Sensor Óptico

Figura 15 – Haste de Interrupção do Sensor

Figura 16 – Interrupção do Sensor



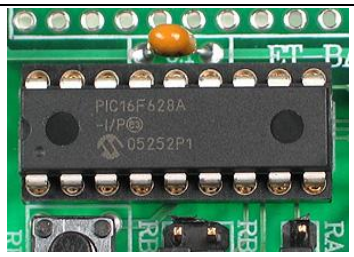
4.2.1.3 *Gravação In-Circuit*

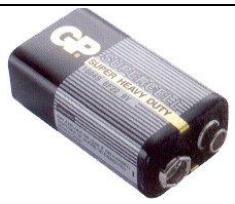
Para efetuar a gravação do *firmware*, e manter a integridade mecânica do micro-controlador, foi projetado um conector de cinco pinos – ligados aos respectivos pins de gravação do micro-controlador - para gravação *in-circuit*. Para efetuar a gravação, basta conectar o respectivo cabo e baixar o programa compilado.

Figura 17 – Conector para Cabo de Gravação

4.2.1.4 *Lista de Peças*

Tabela 4.2 – Lista de Peças de *Hardware*

Item	**Qde**	**Imagem**
Micro-controlador PIC16F628A	1	
Soquetes torneados 16 pinos	2	
Soquetes torneados 18 pinos	1	
Chaves táteis (PCB *Push-Buttons*)	3	
LED *Ultra Bright* 5mm Vermelhos	12	
Resistor 1kΩ - 1/8W	5	
Resistor 330Ω - 1/8W	13	
Resistor 10kΩ - 1/8W	1	
Resistor 47kΩ - 1/8W	1	
Resistor 100Ω - 1/8W	1	

CI ULN2003	2	
ITR8102	1	
Transistor NPN BC548	1	
Capacitor Cerâmico 100 nF	1	
Capacitor Cerâmico 22 pF	1	
Bateria 9v	2	
Conector para Baterias	2	
Barra de 5 Pinos Torneados	1	

4.2.1.5 *Desenho Eletrônico*

Apresentam-se aqui todos os desenhos referentes à documentação da PCI.

Figura 18 – Roteamento PCI

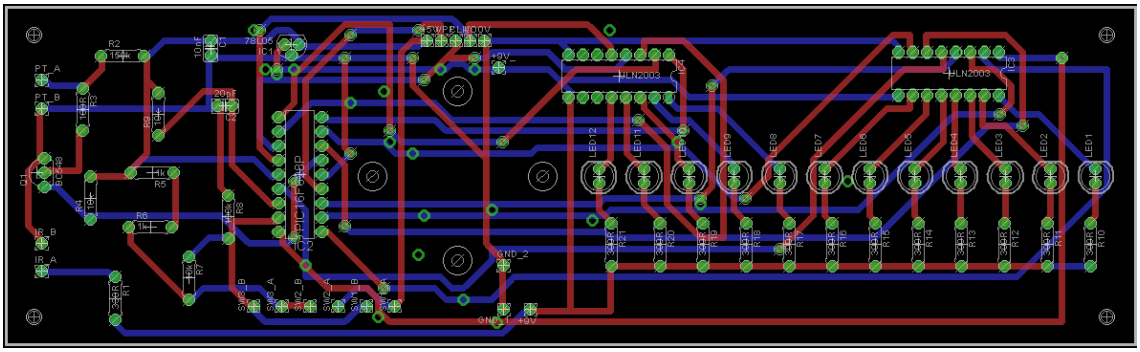


Figura 19 – PCI *Bottom Layer*

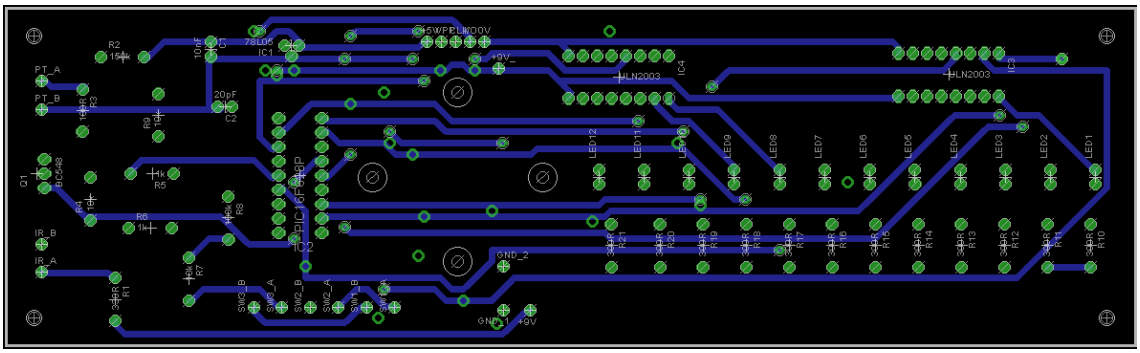


Figura 20 – PCI *Top Layer*

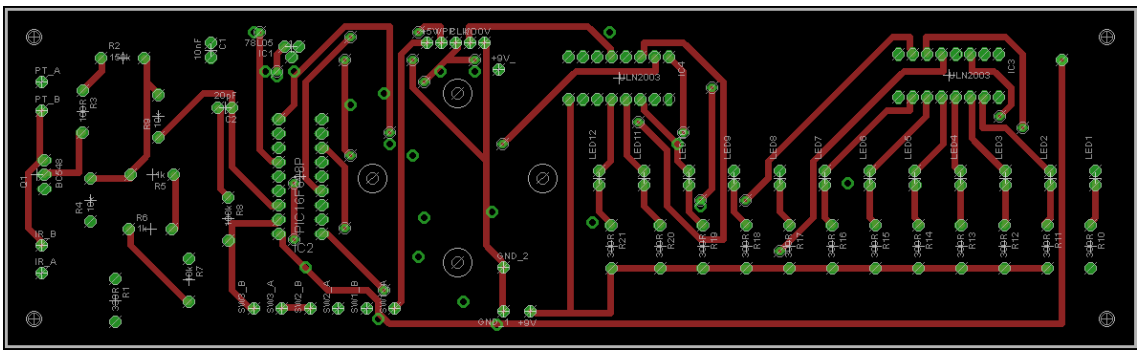
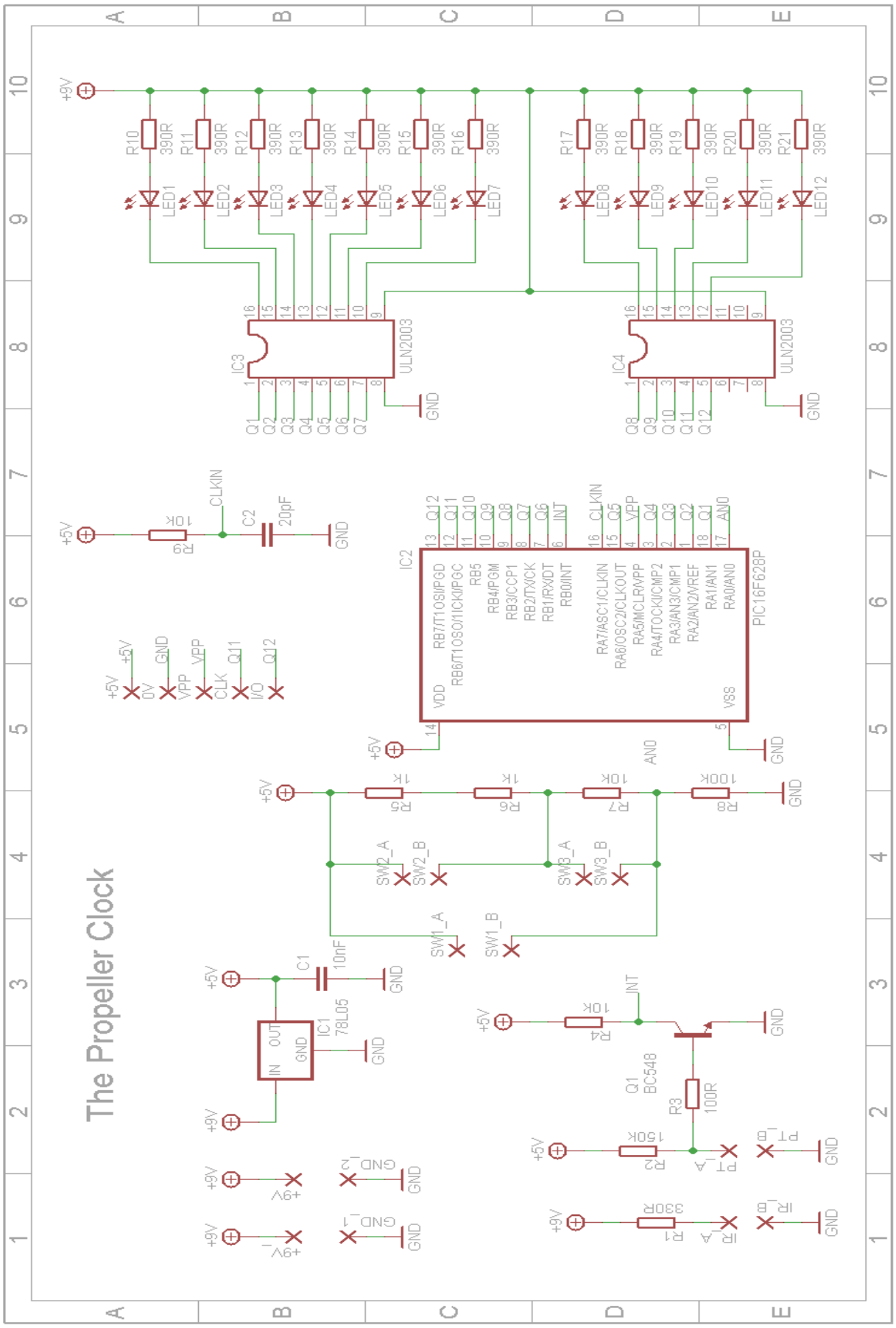


Figura 21 – Diagrama Elétrico *"The Propeller Clock"*



4.3 *Firmware*

4.3.1 Introdução

Comunicação é uma palavra de forte impacto global. A comunicação é algo imprescindível para a convivência em um meio. Mas para haver comunicação é extremamente necessário que exista uma sinergia entre as partes. É necessário estabelecer um protocolo para que ambas as partes se comuniquem; é necessário que ambas falem a mesma língua. No mundo dos micro-controladores isto não é diferente.

4.3.1.1 *Linguagem Assembly*

Esta é a língua primitiva dos micro-controladores. Linguagem *Assembly*, como o próprio nome sugere, é definida como linguagem de montagem. Tem essa definição, pois é uma linguagem de baixo nível, sendo ela a mais próxima da linguagem de máquina.

Para haver comunicação com o homem, esta linguagem utiliza símbolos que correspondem a bits, estes símbolos são chamados de mnemônicos. Cada dispositivo tem seus mnemônicos respectivos à sua arquitetura interna, sendo necessário à pessoa que vai montar o programa conhecer todos os registradores do dispositivo alvo.

4.3.1.1.1. *Assembler*

Assembler ou montador é um programa para micro-computadores responsável pela montagem de um firmware escrito em linguagem de montagem (*Assembly*). Ele simplesmente transforma um arquivo de montagem, escrito através de mnemônicos, em um arquivo de código de máquinas, fazendo assim que seja possível sua gravação no dispositivo.

4.3.1.2 *Linguagem C ANSI*

Esta é uma linguagem de programação compilada, de alto nível, estruturada e padronizada. Desenvolvida nos laboratórios da *AT&T BELL*, esta veio para suprir deficiências da antiga linguagem utilizada, a Linguagem B. Surgiu então sua sucessora, a Linguagem C.

A Linguagem C tornou-se a linguagem de programação mais utilizada do mundo, devido sua forma estruturada e sua versatilidade. Mesmo sendo uma linguagem de alto nível, permitindo um alto nível de abstração, ela permite o acesso de baixo nível aos dispositivos com a inserção de diretivas *Assembly* em seu código fonte.

Após sua difusão mundial, a Linguagem C começou a substituir a Linguagem *BASIC* no quesito de linguagem mais utilizada para programação de

microcomputadores. Com esse grande impacto, o Instituto Norte-Americano de Padrões (ANSI) criou um comitê para estabelecer uma especificação de padronização da linguagem, surgindo a Linguagem C ANSI.

4.3.1.2.1. *Compilador CCS*

Diferente da linguagem em A*ssembly*, as outras linguagens de programação necessitam ser compiladas antes de serem gravadas. Compilar é transformar o código fonte de uma linguagem de alto nível em uma seqüência de diretivas de montagem, ou seja, quando um programa é compilado ele é traduzido à linguagem de máquina utilizando todas as diretivas de montagem que o dispositivo dispõe.

Atualmente, existem compiladores desenvolvidos especificamente para micro-controladores. A *CCS, Inc.*, desenvolveu um compilador de Linguagem C específico para os micro-controladores da família PIC®MCU e dsPIC®DSCs, da Microchip. Este compilador é provido de todos os operadores padrões da Linguagem C ANSI com bibliotecas embutidas específicas para acesso aos registradores internos dos dispositivos micro-controladores da Microchip. (CCS, 2009)

4.3.1.3 *Ambientes de Desenvolvimento Integrado*

Ambientes de Desenvolvimento Integrado, os *IDE*s, são ferramentas utilizadas para editar e gerir projetos de *softwares*.

Para projetos utilizando a família PIC®MCU e dsPIC®DSCs, a Microchip desenvolveu uma *IDE* própria, esta que é disponibilizada gratuitamente, o MPLAB *IDE*. Através deste, é possível gerenciar o desenvolvimento dos projetos de *firmwares* para todos os dispositivos disponibilizados pela Microchip.

O MPLAB *IDE* permite que sejam adicionados ao ambiente integrado, *softwares* de terceiros, para potencializar sua utilização. Existem *plug-ins* que possibilitam a integração dos compiladores CCS ao MPLAB *IDE*, possibilitando a escrita de códigos para programação em Linguagem C. (MICROCHIP, 2009)

4.3.2 ***Firmware*** **Desenvolvido**

4.3.2.1 *Introdução*

Sucintamente, *Firmware* é um *software* embarcado, também definido como um conjunto de instruções operacionais, desenvolvido exclusivamente para um dispositivo alvo, tendo sua gravação efetuada diretamente no mesmo. Este é o resultado da comunicação entre o homem e a máquina.

"*The Propeller Clock"* é um dispositivo controlado por *software*, onde necessita um RTC para obter a base de tempo e assim possibilitar o controle de acionamento dos componentes de sua interface

4.3.2.2 *Descrição*

A lógica de funcionamento do *firmware* pode ser dividida em quatro etapas: a primeira etapa é o pré-set do relógio; a segunda etapa é responsável pelo controle da base de tempo do relógio; a terceira etapa é a responsável por controlar a rotação do motor; e a última etapa, responsável por controlar o acionamento da barra de *LEDs* – levando em consideração as duas primeiras etapas.

4.3.2.2.1. *Ajuste do Relógio*

Primeiramente, é necessário fazer o pré-set do horário. Para isso, utilizam-se os três botões que estão configurados por um divisor de tensão ligado à entrada de sinal analógico do micro-controlador. Cada botão pressionado tem um respectivo valor de tensão analógico, este que é reconhecido e tratado pelo micro-controlador.

Cada botão tem uma função distinta: Switch1 incrementa dez vezes e entra na função; Switch2 incrementa uma vez; Switch3 seleciona a função e seta incremento.

No momento do ajuste do relógio, o motor estará desligado. Sendo assim, utiliza-se a barra de *LEDs* para fazer a IHM. Esta interface é feita através de um código binário, utilizando o *nibble* menos significativo para demonstrar a função atual e o próximo *byte* para mostrar os incrementos. Existem três funções de ajuste: 'Ajuste de Hora', 'Ajuste de Minuto' e 'Finaliza Ajuste'.

4.3.2.2.2. *Controle do Tempo*

Como é proposto em seu próprio título, é necessário criar um relógio para o controle da base de tempo. A idéia inicial para implementação deste relógio foi utilizar uma fonte de *clock* externo com uma freqüência de 1Hz. Pensando nas limitações mecânicas do dispositivo - e nas limitações do próprio micro-controlador - optou-se em implementar um *RTC* por *software*.

Para implementação do *RTC*, foi adaptada uma biblioteca pronta. Esta biblioteca configura um *timer* interno do micro-controlador para atender ma interrupção a cada segundo, fazendo assim com que haja o incremento de segundo. Para incremento do minuto, testa-se a variável que armazena os segundos. Para incremento da hora, testa-se a variável dos minutos.

4.3.2.2.3. *Controle de Rotação do Motor*

O controle de rotação do motor é efetuado através a resposta do Sensor Óptico acoplado no eixo do motor. Este sensor atua sempre que o motor completa um giro de 360º, marcando cada volta.

Nos micro-controladores da família PIC® existe um pino que pode ser configurado para atender uma interrupção externa toda vez que detectar uma troca de nível de sinal. Este pino é utilizado para atender uma interrupção externo sempre que o sensor atuar.

Para calcular a rotação do motor, utiliza-se uma interrupção externa, esta que será atendida sempre que o sensor atuar – detectando uma borda de descida no pino do micro-controlador. Toda vez que esta interrupção externa for atendida, o programa incrementa uma variável, que, a cada segundo, conta quantas vezes foram detectadas estas interrupções, ou seja, guarda o número de voltas que foram dadas pelo motor durante um segundo.

4.3.2.2.4. *Acionamento da Barra de LEDs*

O controle para o acionamento dos *LEDs* é dependente das supra-referidas etapas, pois o acionamento dos mesmos ocorre com base na hora e velocidade atual.

Primeiramente, é necessário salientar o número de posições que cada ponteiro pode assumir. O ponteiro responsável pela exibição das horas pode assumir 12 posições, enquanto os ponteiros responsáveis pela exibição dos minutos e segundos assumem 60 posições cada.

Para saber a posição exata de cada ponteiro, é necessário que exista um ponto de referência fixo. Para isto, o sensor óptico foi estrategicamente acoplado ao eixo do motor, servindo também de referência ao software e fazendo com que, no momento em que o sensor é acionado, o dispositivo esteja na posição ilustrada pela Figura xx. É este ponto de referencia que possibilita estimar o tempo necessário para atingir cada passo dos ponteiros. Partindo destas premissas, é possível controlar devidamente o dispositivo.

Feita a leitura do horário atual, calcula-se a velocidade de rotação do motor para corrigir o tempo de passo dos ponteiros. Com base no horário obtido, sabem-se quantos passos é necessário dar para o ponteiro chegar à respectiva posição. Após saber o número de passos necessário, configura-se um *timer* interno do micro-controlador para aguardar o tempo correto para poder efetivar acionamento da barra de *LEDs* no ponto correspondente. Este procedimento é efetuado para os três ponteiros.

4.3.2.3 *Código Fonte (*.c)*

```
////////////////////////////////////////////
//           The Propeller Clock          //
//       Version 1.0 <> June, 2009        //
// Developed by Marcos Vinicius Rogowski  //
////////////////////////////////////////////

#include <16f628a.h>
#device ADC = 8
#fuses INTRC_IO, PUT, NOWDT, NOMCLR, NOBROWNOUT
#use delay (clock=3932160)
#include <rtc_int.h>

// Protótipos de Funções
//void adj_relogio(void);
//long get_speed(void);
//void get_time(void);
//void show_time(void);

// Programa Pricipal
void main(void)
{
   // Inicialização das Interrupções
   ext_int_edge(H_TO_L);
   setup_timer_2(T2_DIV_BY_16, 250, 10);
   set_timer2(0);
   enable_interrupts(INT_TIMER2);
   enable_interrupts(INT_EXT);
   enable_interrupts(GLOBAL);
   reset_relogio();

while (1)
{
output_high(LED1);
}
}
```

4.3.2.4 *Bibliotecas (*.h)*

```
/*****************************************************
 **          Propeller Clock's Library               **
 *****************************************************
 ** Esta é uma adaptação da biblioteca rtc_sw_3int0.h **
 ** Biblioteca adaptada para PIC16F628A               **
 ** Adaptada por Marcos Vinicius Rogowski             **
 *****************************************************/

/*
Inicialização:

void main(void)
{
   ext_int_edge(H_TO_L);
   setup_timer_2(T2_DIV_BY_16, 250, 10);
   set_timer2(0);
   enable_interrupts(INT_EXT);
   enable_interrupts(INT_TIMER2);
   enable_interrupts(GLOBAL);
   reset_relogio();
}
*/

#use fixed_io (a_outputs = pin_a1, pin_a2, pin_a3, pin_a4, pin_a6)
#use fixed_io (b_outputs = pin_b1, pin_b2, pin_b3, pin_b4, pin_b5,
pin_b6, pin_b7)

#define LED1 pin_a1
#define LED2 pin_a2
#define LED3 pin_a3
#define LED4 pin_a4
#define LED5 pin_a6
#define LED6 pin_b1
#define LED7 pin_b2
#define LED8 pin b3
#define LED9 pin_b4
```

```
   #define LED10 pin_b5
   #define LED11 pin_b6
   #define LED12 pin_b7


   #define CLOCK 3932160 // Frequência do clock [Hz] do sistema de
hardware;
   #define INTS_PER_SECOND CLOCK/160000 // calcula a constante de tempo
para o rtc em função do clock do sistema (CLOCK/(4*16*250*10));


   int8 int_count; // Number of interrupts left before a second has
elapsed
   static short pisca = 0; // variavel que fica 250ms em "1" e 750ms em
"0";


   struct relogio_str
   {
      int  hora;
      int  minuto;
      int  segundo;
   } relogio, novo;


   void reset_relogio(void)
   {
      relogio.hora=00;
      relogio.minuto=00;
      relogio.segundo=00;
   }


   void inc_hora (void){
      relogio.hora++;
      if (relogio.hora > 23)
      {
         relogio.hora = 0;
      }
   }


   void inc_minuto (void){
      relogio.minuto++;
      if (relogio.minuto > 59)
      {
         relogio.minuto = 0;
```

```
      inc_hora();
   }
}

void inc_segundo (void){
   relogio.segundo++;

if (relogio.segundo > 59)
    {
relogio.segundo = 0;
      inc_minuto();
}
}

void pont_seg(void)
{
output_a(0xFF);
output_b(0xFF);
delay_us(500);
output_a(0x00);
output_b(0x00);;
delay_us(500);
}

/* Código de Ajuste do Relógio
short ajuste(struct relogio_str *val_novo){
   short resultado = 1;
   // falta implementar o código pra o ajuste;
   return(resultado);
} */

long PPS, CONTA = 0; // PPS = Pulsos Por Segundo

#INT_TIMER2 // Gera um interrupção a cada 1s.
void clock_isr() {
   if (int_count == INTS_PER_SECOND/4) pisca = ~pisca;
   if(--int_count==0) {
      pisca = ~pisca;
      int_count = INTS_PER_SECOND;
```

```
        PPS = CONTA; // Quarda na variavel PPS o numero de pulsos
ocorridos durante o último segundo.
        CONTA = 0;
        inc_segundo();
    pont_seg();
      }
  }

  #INT_EXT
  void funcao_intermitente() // Interrupção Externa.
  {
  CONTA ++; // Incrementa 1 a cada passagem pelo sensor.
  }


  short set_segundo (struct relogio_str *data, int segundo){
     short resultado = 1;
     if ((segundo >= 0) && (segundo < 60)) data->segundo = segundo;
     else resultado = 0;
     return(resultado);
  }

  short set_minuto (struct relogio_str *data, int minuto){
     short resultado = 1;
     if ((minuto >= 0) && (minuto < 60)) data->minuto = minuto;
     else resultado = 0;
     return(resultado);
  }

  short set_hora (struct relogio_str *data, int hora){
     short resultado = 1;
     if ((hora >= 0) && (hora < 24)) data->hora = hora;
     else resultado = 0;
     return(resultado);
  }
```

4.3.2.5 *Executável (*.hex)*

Após a confecção de um programa, utilizando uma linguagem de programação que necessita ser compilada, no caso, utilizando Linguagem C, tem-se como resultado desta compilação um arquivo contendo um código hexadecimal. Este é o código de máquina correspondente ao código fonte do programa desenvolvido. É a partir deste arquivo que é efetuada a gravação no dispositivo.

Para efetuar a gravação tradicional, é necessária a utilização de um micro-computador com uma porta de dados paralela disponível, pois é a partir dela que o programa é descarregado no micro-controlador.

Para descarregar o programa gerado, necessita-se também a utilização de *softwares* de gravação. Existe uma diversidade considerável de *softwares* de gravação. Basta verificar a compatibilidade do mesmo com seu dispositivo de gravação e dispositivo a ser gravado. Neste projeto, optou-se pelo gravador desenvolvido pela *microEngineering Labs, Inc.*: *meLabs Programmer Beta*.

Segue abaixo o respectivo código hexadecimal dôo dispositivo alvo.

```
:1000000000308A0088280000FF00030E8301A10051
:100010007F08A0000A08A8008A01A00E0408A20018
:100020007708A3007808A4007908A5007A08A6003C
:100030007B08A700831383120B1E20288B187E28B1
:100040008C308400801C26288C18392822088400D3
:100050002308F7002408F8002508F9002608FA000C
:100060002708FB0028088A00210E8300FF0E7F0E60
:1000700009002908063C031D42282A1C41282A1091
:1000800042282A14A90B7B282A1C48282A10492810
:100090002A141830A9003408B2003308B100B401A2
:1000A000B301AD0A2D083B3C03186328AD01AC0A2F
:1000B0002C083B3C03186328AC01AB0A2B08173C07
:1000C00003186328AB01A1306500FF3085000130C3
:1000D0006600FF308600A330F700F70B6D280000A4
:1000E000A13065008501013066008601A330F7006C
:1000F000F70B782800008C108A112628B30A031900
:10010000B40A8B108A112628AB01AC01AD01A728D7
:1001100840183131F30830583168E1507308312E5
:10012000F002A10B301B401831601134830F80070
:1001300006388312920FA30831692008312910 1DE
:1001400083168C1483120B16C0308B048428A130C4
:0801500065008514A728630077
:02400E00103F61
:00000001FF
;PIC16F628A
```

CONSIDERAÇÕES FINAIS

A implementação deste projeto possibilitou ao grupo evidenciar diversos conceitos que, até o momento, eram relativamente superficiais. Conceitos que não são perceptíveis nos dispositivos em sua individualidade, somente a integração dos mesmos torna possível a percepção.

Com base nas pesquisas teóricas e referências de projetos semelhantes já implementados, foram tomadas as decisões referente às características do projeto em si, desde sua etapa bruta – mecânica – à sua etapa mais sofisticada – *firmware*.

Primeiramente foi necessário desenvolver uma base de *hardware* mecânico para o dispositivo. Pensando em soluções "limpas" para o desenvolvimento do mesmo, procurou-se optar, sempre que possível, pela reutilização de materiais, sempre contribuindo com a integridade de nosso meio-ambiente. Para esta base de *hardware* mecânico é necessário que exista um atuador para efetuar a rotação do dispositivo. Como a idéia de controle do dispositivo era fazer o mesmo através de *software*, tornou muito mais fácil a reutilização de materiais, pois qualquer tipo de motor serviria como atuador. Partindo destas premissas, foi efetuada a reutilização de um pequeno ventilador que se encontrava fora de uso. Este utilizava um motor AC.

Após ter definido a base de *hardware* mecânico, foi iniciada a confecção do *hardware* eletrônico. Como esta aplicação é extremamente específica, foi necessário desenvolver uma base de *hardware* especificamente para este projeto.

Depois de projetado o esquema elétrico, optou-se por efetuar a montagem em uma placa manufaturada. Foi neste ponto onde foi encontrada a primeira e maior dificuldade.

Primeiramente, é necessário efetuar o desenho do *layout* da placa com base no esquema elétrico projetado. Entre erros de *layout* e extrema dificuldade em efetuar o roteamento de uma placa em face simples, optou-se em desenvolver uma PCI com face dupla. No quesito roteamento da placa, resultou em uma ótima solução. O problema surgiu quando foi necessário levar a placa à prensa, pois era necessário haver um alinhamento perfeito entre as duas faces, caso contrário, resultava em erros de *layout*.

A confecção da placa de circuito impresso resultou em um extremo atraso no cronograma, que foi obrigatoriamente reconsiderado.

Após a confecção da placa, foi efetuada a montagem da mesma. Na execução dos testes, foi verificado que havia alguns erros na placa. Foram verificados alguns curtos-circuitos que foram retificados sem maiores problemas. Outro problema ocorreu na configuração dos botões, estes que forçaram uma pequena alteração no esquema

elétrico, pois estavam com níveis de tensão muito próximos, dificultando assim sua distinção.

Depois de confecção, montagem e teste da placa, foi efetuada a integração do *hardware* mecânico com o *hardware* eletrônico. Neste ponto foi encontrada a segunda dificuldade. A placa foi acoplada ao eixo do motor do ventilador, de maneira com que ficasse firme. Quando foi ligado o ventilador, o mesmo não suportava a inércia da placa e acabava se deslocando devido a sua pouca massa.

Após o fracasso do primeiro teste de integração, o motor do ventilador foi retirado e foi projetada uma nova base para o mesmo. Novamente reutilizando materiais, utilizou-se uma chapa de aço juntamente com uma haste para fazer um novo suporte para o motor. A haste metálica foi soldada no centro da chapa de aço, esta que tem massa igual a 4,5kg.

Com a nova base mecânica, foi efetuada novamente a integração das duas bases de *hardware*.

No segundo teste de integração, verificou-se que o problema não estava somente na pouca massa do ventilador. Mesmo com uma base de massa considerável, a pseudo-hélice não girava corretamente, pois era necessário efetuar o balanceamento da mesma.

Para efetuar o balanceamento, utilizaram-se parafusos e porcas na extremidade mais "leve" da placa. Após alguns testes, foi encontrado o ponto correto de balanceamento das massas. Como resultado, a pseudo-hélice gira com velocidade e equilíbrio.

Tendo obtido sucesso na integração de *hardwares*, foi efetuada a instalação do sensor óptico. Efetuando os testes, constatou-se que o mesmo não estava funcionando corretamente, pois a chave óptica não foi ligada conforme o esquema elétrico. Após a retificação, a mesma funcionou perfeitamente.

Com toda a integração de *hardware* concluída, foi efetuada a integração do *firmware*. Primeiramente, foi gravado um programa teste. Ao executar o mesmo, verificou-se que um dos *LEDs* não estava sendo acionado. Efetuando os testes, verificou-se que o pino RA4 não estava efetuando a mudança de estado – o mesmo permanecia em nível baixo. Depurando o programa, verificou-se que o mesmo não continha erros. Verificando o *datasheet* do micro-controlador, verificou-se que havia um erro de projeto. O pino RA4 é um coletor aberto. Para resolver o problema, é necessário inserir um resistor de *pull-up* externo, pois o pino RA4 não dispõe de *pull-up* interno. Para facilitar a manutenção, verificou-se que o *LED* controlado pelo pino RA4 tem seu estado sempre semelhante ao *LED* controlado pelo pino RA5. Logo, desligou-se o *LED* do pino RA4 o mesmo fora ligado junto ao pino RA5. Efetuando o teste novamente, verificou-se o perfeito funcionamento do dispositivo.

Após os testes, foi gravado programa principal. Programa funcionando perfeitamente.

Através deste, foi apresentado, na prática, o conceito da exibição de imagens no ar através do varrimento mecânico. Existem diversos projetos semelhantes que, em sua maioria, são de cunho didático ou simplesmente de entretenimento.

Desenvolvendo este, foi bastante perceptível que este tipo de dispositivo tem muito potencial a ser explorado. É possível expandir suas aplicações, não limitando-se as já conhecidas, basta usar a imaginação.

REFERÊNCIAS BILIOGRÁFICAS

MANZANO, J. A. N. G. **Algoritmos: lógica de desenvolvimento de programação de computadores.** 20ª Ed. São Paulo: Érica, 2007.

MANZANO, J. A. N. G. **Estudo Dirigido de Linguagem C**. 10ª Ed. São Paulo: Érica, 2007.

PEREIRA, F. **PIC Programação em C**. 3ª Ed. São Paulo: Érica, 2003.

PATRÃO, B. N. F. **Desenho de caracteres por varrimento mecânico e suas aplicações.** 2003/2004. Trabalho Individual (Faculdade de Ciências e Tecnologia) – Departamento de Engenharia Informática, Universidade de Coimbra, Portugal.

GREGORY, R. L. ***Knowledge in perception and illusion*.** – *Department of Psychology, University of Bristol, 8 Woodland Road, Bristol BS8 ITA UK*.

WIKIPEDIA. **Ilusão de Óptica.** Disponível em: <http://pt.wikipedia.org/wiki/Ilusão_de_óptica>. Acesso em: Junho, 2009.

WIKIPEDIA. **Linguagens de montagem.** Disponível em: <http://pt.wikipedia.org/wiki/Linguagem_de_montagem>. Acesso em: Junho, 2009.

BLICK, B. ***"Propeller Clock" Mechanically Scanned LED Clock.*** Disponível em: <http://www.bobblick.com/techref/projects/propclock/propclock.html>. Acesso em: Junho, 2009.

INC., *CCS*. **CCS *Compilers.*** Disponível em: <http://www.ccsinfo.com/>. Acesso em: Junho, 2009.

TECHNOLOGY, Microchip. **MPLAB IDE.** Disponível em: <http://www.microchip.com/>. Acesso em: Junho, 2009.