CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

RAFAEL DE MELLO ALFARONE

PERSIANA RESIDENCIAL AUTOMATIZADA UTILIZANDO SENSOR DE LUMINOSIDADE

Orientador: Thiago de Miranda Leão Toribio

BRASÍLIA / DF

2º SEMESTRE DE 2010

RAFAEL DE MELLO ALFARONE

PERSIANA RESIDENCIAL AUTOMATIZADA UTILIZANDO SENSOR DE LUMINOSIDADE

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Thiago de Miranda Leão Toribio

BRASÍLIA / DF

2º SEMESTRE DE 2010

RAFAEL DE MELLO ALFARONE

PERSIANA RESIDENCIAL AUTOMATIZADA UTILIZANDO SENSOR DE LUMINOSIDADE

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Thiago de Miranda Leão Toribio

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandez

Coordenador do Curso

Banca Examinadora:

_____

Thiago de Miranda Leão Toribio, Doutor em Fisica Teórica.

Orientador

_____

Prof. Maria Marony, Mestre em Engenharia Elétrica.

UniCEUB – Centro Universitário de Brasília

_____

Prof. Marco Antônio, Mestre em Ciência da Computação.

UniCEUB – Centro Universitário de Brasília

DEDICATÓRIA

Dedico este projeto e monografia, a meus pais, minhas irmãs, todos os meus familiares, principalmente a minha esposa e ao meu filho que estiveram sempre me apoiando e dando forças para que o meu sonho fosse realizado.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ele me abençoar com a sabedoria e entendimento, a meus pais pelos esforços incansáveis para formação de meu caráter, a minha esposa, pelo apoio e tempo gastos para me ajudar, meu filho, que teve que brincar sem o pai para que eu pudesse realizar esse trabalho, às minhas irmãs, madrasta Monica, ao Guillermo, todos meus familiares, ao José Carlos do Laboratório de Eletrônica, Gabriel Santos, Gustavo Suzukawa, Alcides Rafael, Jean Matheus, enfim, a todos que estiveram presentes neste momento tão importante da minha vida e em especial aos meus professores que partilharam comigo seu conhecimento, conhecimento esse que me acompanhará pelo resto da vida.

O meu muito obrigado!

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

LDR - *Light Dependent Resistor*

A/D – Analógico / Digital

CI – Circuito integrado

LCD – *Liquid Cristal Display*

LED – *Light Emiting Diode*

CdS - Sulfeto de cádmio

RESUMO

Neste projeto é apresentada uma proposta de automação de persianas residenciais controlada por luminosidade. A maneira escolhida para essa automação foi o controle de um motor de passo por meio de um microcontrolador, que recebe informação de um sensor de luminosidade, o sensor utilizado é um LDR (Light Dependent Resistor). O projeto tem a função de abrir a persiana com a ausência de luz, fechar a persiana com a presença de luz e regular a angulação de acordo com a incidência da luz. Também foi desenvolvida uma maneira de acionar a abertura e o fechamento da persiana por meio de botões. Para o funcionamento do sistema é utilizado um motor de passo do tipo unipolar de imã permanente que está controlado por um microcontrolador modelo PIC16F876A, que também possui a função de receber informações do usuário. É utilizado um kit de desenvolvimento da ACEPIC, desse kit são utilizados Display de LCD, o clock externo e a gravadora.

Palavras Chave: Motor de passo, microcontrolador, sensor de luminosidade, automação, LDR.

ABSTRACT

In this project is presented a proposal for residential shutters automation controlled by light. The way chosen for this automation is the control of a step motor by means of a microcontroller, which receives the information of a light sensor, the sensor used is an LDR (Light Dependent Resistor). The project has the function to open the blind with the light absence, close the blind with the light presence and set the angulations according to the incidence of light. Also a way was developed to set in motion the opening and the closing of the blind by the way of buttons. For the operation of the system a step motor of the single polar type with permanent magnet will be used, which will be controlled by a microcontroller, model PIC16F876A, which also will have the function to receive the information from the user. It uses a development kit ACEPIC, are used LCD Display, the external clock and record label of this kit.

Keywords: step motor, microcontroller, automation, light sensor, LDR.

CAPÍTULO 1 - INTRODUÇÃO

1.1 - Apresentação do Problema

Uma persiana funciona com uma mola que é controlada por uma lingueta e uma pequena alavanca de retenção. A Persiana automatizada vai fazer com que os usuários não tenham que se preocupar com a incidência de sol em seus pertences, pois caso haja necessidade, o usuário pode acionar essa funcionalidade e caso não tenha problema com a incidência de sol ele pode desativar.

Este projeto visa minimizar os problemas em residências, como a incidência de sol em móveis, que podem ficar com a cor queimada, em objetos eletrônicos deste local, como computadores, monitores e televisões de LCD (*Liquid Cristal Display*), que liberam muito calor e podem aquecer mais que o normal ou queimar com a incidência direta do sol, também em animais de estimação que ficam presos neste local, podem até morrer com o calor do sol.

Sem a persiana o proprietário poderá sofrer prejuízos como o de uma televisão de LCD queimada, tendo que pagar uma bagatela de mais de mil reais para adquirir uma nova, pois muitas vezes o conserto da televisão sai muito caro.

Neste trabalho é utilizado como sensor de luminosidade o LDR (*Light Dependent Resistor*), desta forma o microcontrolador utilizado rotaciona a persiana de acordo com a informação obtida do LDR. O projeto conta também com controles para abrir e fechar a persiana de acordo com a necessidade do usuário.

Possivelmente, em um futuro muito próximo, os sistemas automatizados estarão presentes em muitas residências, assim como há em outros países (Cidades Inteligente *SmartCities*), trazendo conforto, economia e segurança para as pessoas. Como no exemplo a seguir. [ROSARIO]

Os 215 mil habitantes do município inglês de Southampton, Inglaterra, já começaram a usufruir, em suas tarefas cotidianas, dos benefícios da Era da Informação. Como parte do projeto *SmartCities*, patrocinado pela Comunidade Europeia e coordenado pela Schlumberger, empresa líder do mercado mundial de *smart cards*, os cidadãos daquela cidade terão acesso a

transporte, lazer, educação e serviços sem a necessidade de carregar dinheiro. Esta mudança proporcionará maior eficiência para a prefeitura local. O usuário terá a possibilidade de adicionar e remover aplicações dos *smart cards*, usando tanto um terminal de acesso público, um telefone móvel ou um PC conectado à internet, essenciais para a viabilização do projeto. [ROSARIO]

Como este projeto depende das condições climáticas, é muito importante ter componentes que satisfaçam o projeto, sendo assim, para o bom funcionamento deste é necessário um bom sensor de luminosidade (LDR).

A dúvida é: Seria possível implementar uma persiana automatizada ligada a um sensor de luminosidade (LDR) com valor reduzido quando comparado com as presentes no mercado?

1.2 - Motivação

A principal motivação para realização deste projeto é utilizar conhecimentos adquiridos no curso de Engenharia de Computação que são ligados à área de automação. Várias disciplinas como “Microprocessadores e Microcontroladores”, “Linguagens e Técnicas de Programação”, “Circuitos e Máquinas Elétricas”, ”Arquitetura de Computadores”, ”Instalações Elétricas”, dentre outras do curso de Engenharia da Computação, fornecem uma real percepção de como são projetados os sistemas de automação de uma forma simplificada, sendo também estudado ao longo do curso o funcionamento dos microcontroladores, componente utilizado no projeto.

A construção de um sistema de automação residencial de baixo custo foi outra grande motivação para a realização deste projeto, pois muitas pessoas têm problemas com a incidência de sol em seus pertences e não se lembram de fechar a cortina ou persiana. Um sistema barato de persiana automatizada pode ser a melhor e mais acessível solução.

1.3 - Objetivos do Trabalho

O objetivo deste projeto é especificar, desenvolver e implementar um sistema automatizado de persianas residenciais, de modo a controlar a incidência de Sol, sendo que a persiana efetua um giro de acordo com a intensidade da luz, quando sem incidência de luz a persiana fica em um ângulo que permita entrada máxima de luz e caso haja a incidência de luz o ângulo da persiana deve ser calculado.

Para que seja realizado o objetivo proposto, é necessário utilizar um microcontrolador, que aciona um motor de passo, fazendo com que a persiana abra ou feche de acordo com a luz atuante em um sensor de luminosidade (LDR).

Para uma maior interação e comodidade do usuário com o sistema, o projeto conta com botões, a fim de ter um controle de abertura e fechamento da persiana. Esse controle de angulação pelos botões é realizado da seguinte forma: Se o usuário apertar um botão a persiana gira no sentido horário e quando aperta o outro a persiana gira no sentido anti-horário.

Para interpretar os comandos de abertura e fechamento das janelas e transformá-los em informações capazes de movimentar o motor de passo, é utilizado um microcontrolador da família PIC. Este microcontrolador também é responsável por enviar a informação correta que deve ser mostrada no display, que tem a função de mostrar em sua tela se a persiana está aberta ou fechada.

1.4 - Justificativa e Importância do Trabalho

Este projeto enquadra-se dentro das matérias estudadas no curso de Engenharia da Computação, algumas disciplinas ligadas diretamente ao tema proposto são: Microcontroladores e Microprocessadores, Circuitos e Máquinas Elétricas, Linguagens e Técnicas de Programação, Arquitetura de Computadores e Instalações Elétricas.

O tema proposto é muito atual, sendo que as pesquisas na área de automação estão em ascensão, não somente automação como a integração de todos os sistemas domésticos como foi dito por José Roberto Muratori, engenheiro, membro, fundador e conselheiro da AURESIDE – Associação Brasileira de Automação Residencial: “automação é apenas parte de um processo maior, a integração de todos os sistemas domésticos, o que inclui a instalação elétrica, a segurança do imóvel, a distribuição de áudio e vídeo, toda rede de comunicações e as utilidades”. Este conceito, assim apresentado, é tão novo e desconhecido que pode ser encarado como um grande desafio para os profissionais a sua correta interpretação e implantação. [AUTOMATIZAR]

O avanço da tecnologia, e facilidades à mão de todos, fazem com que a concorrência nesta área da Engenharia seja cada vez maior, onde não só as grandes empresas têm a possibilidade de elaborar projetos de automação e sim qualquer pessoa que tenha o

conhecimento das tecnologias já existentes no mercado, tornando assim possível uma queda no preço de sistemas similares.

1.5 - Escopo do Trabalho

O projeto utiliza um microcontrolador, que aciona um motor de passo, fazendo com que a persiana abra ou feche, girando de acordo com a luminosidade atuante no sensor (LDR) e o status aparece no display. O projeto conta também com botões, a fim de ter um controle de abertura e fechamento da persiana.

O projeto não vai fazer com que a persiana abra ou feche em caso de chuva e nem em caso de vento.

1.6 - Resultados Esperados

O LDR (sensor) envia para o microcontrolador o sinal para abrir ou para fechar e o nível de intensidade da luz para que o ângulo de abertura seja calculado, recebendo esse sinal, o microcontrolador verifica se a persiana está aberta ou fechada e se o ângulo é o mesmo a ser utilizado, no caso do sinal ser diferente do que estava antes, o microcontrolador envia um sinal para o motor de passo fazendo com que ele feche ou abra a persiana alterando o ângulo, com a persiana fechada é preciso girar a persiana até que fique com o ângulo calculado. A posição em que se encontra a persiana aparece no display. Caso ocorra um erro, também é mostrado no display, o microcontrolador solicita um novo sinal para o sensor e o sinal será reenviado para que o erro seja corrigido. A persiana deve estar de acordo com o que estiver no display. Topologia ilustrativa na Figura 1.1.

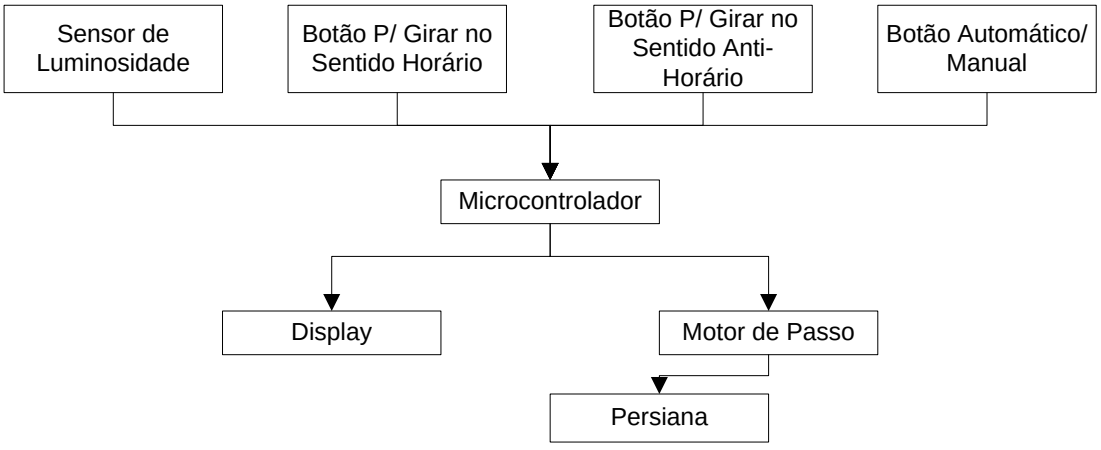


Figura 1.1 – TOPOLOGIA

[AUTOR]

1.7 - Estrutura do Trabalho

A estrutura desta monografia consiste em cinco capítulos que trata os assuntos descritos abaixo:

Capítulo 2 – Este Capítulo apresenta uma descrição profunda e detalhada do problema que pretende resolver, bem como os fatores, parâmetros e ambientes associados, além de um levantamento de como o referido problema vem sendo resolvido pelos profissionais da área envolvida, os limites de resolução dos métodos atuais, por que vale a pena resolver ou aprofundar o estudo deste problema.

Capítulo 3 – É o capítulo de referencial tecnológico, onde a teoria e a tecnologia utilizadas neste projeto são demonstradas, abordando uma teoria acerca dos dispositivos utilizados no desenvolvimento do projeto, visando auxiliar e levantar um conhecimento já concretizado para uma melhor compreensão do projeto.

Capítulo 4 – É o capítulo demonstrativo da implementação de software e hardware. Aqui é descrita a conexão entre os componentes do hardware, programação efetuada no microcontrolador e utilização de alguns componentes eletrônicos necessários para o funcionamento dos componentes principais ou somente para regulagem.

Capítulo 5 – É o capítulo onde são expostos os resultados através de figuras, textos com o funcionamento do protótipo, custos e onde é feita uma avaliação global do projeto.

Capítulo 6 – É o capítulo onde é deixada a impressão causada pelo projeto, por meio de descrição das dificuldades encontradas. Onde as considerações finais serão expostas e descritas às sugestões para trabalhos futuros.

CAPÍTULO 2 – APRESENTAÇÃO DO PROBLEMA

2.1 - Conceito de Automação Residencial

A Automação Residencial aplica técnicas de automação para que o conforto e a segurança das residências e conjuntos habitacionais possam melhorar, como: Controle de acesso por biometria, porteiro e portões eletrônicos, circuitos Fechados de Televisão (CFTV), controle de luminosidade de ambientes, controle de umidade, temperatura e ar condicionado (HVAC), etc. [LUZ, 2009].

Como qualquer novidade, a Automação Residencial inicialmente é entendida somente como um sinal de status, modernidade, comodidade e conveniência pelo cliente. Quando o cliente se intera mais e recebe mais informações sobre o produto, ele percebe que além das características citadas anteriormente possui também um fator de economia e percebe que o produto se torna uma necessidade vital. [LUZ, 2009]

Em um País com milhões de pessoas morando em favelas, há sentido em desenvolver automação residencial? Em um país em que muitas regiões não possuem nem saneamento básico é complicado falar de automação. Apesar da dura realidade do país, o custo dessa tecnologia está cada vez mais acessível e o público que possui condições para adquirir esse produto vem crescendo. [ROSÁRIO]

Boa parte dos produtos ainda procura dar conforto e até status, mas muitas vezes trata-se de investimento no conforto e melhoria de vida, além de contemplara aspectos relativos à segurança residencial e predial, e no caso que já atinge grande parte da população, no projeto de controle de instalações prediais de grande porte, como hospitais públicos e postos de saúde, prefeituras, distritos residenciais, escolas, prédios públicos e condomínios residenciais. [ROSÁRIO]

2.2 - Visão do projeto

A Persiana automatizada vai fazer com que os usuários não tenham que se preocupar com a incidência de sol em seus pertences, pois caso haja necessidade o usuário pode acionar essa funcionalidade e caso não tenha problema com a incidência de sol ele pode desativar.

A necessidade de manutenção diminui e a vida útil da persiana aumenta, por não ter que controlá-la por meios convencionais e sim automaticamente com o sensor de luz ou manualmente por meio de botões, acabando com a má utilização da ferramenta através do manuseio da corda.

Com a finalidade de acabar com o problema da incidência solar em ambientes fechados, problema esse que afeta sofás e objetos de couro, que podem ficar com a cor queimada e manchas irreversíveis; objetos eletrônicos, como computadores, monitores e televisões de LCD, que liberam muito calor, podendo aquecer mais que o normal e no caso das televisões e monitores de LCD, podem queimar com a incidência de luz solar; animais de estimação que ficam presos nestes locais, podendo até chegar a óbito com o calor do sol; e pacientes de hospitais que não podem levantar para fechar à persiana e devem esperar a chegada de um (a) enfermeiro (a).

Um caso comum é a pessoa sair para trabalhar enquanto o sol ainda não apareceu e esquecer a cortina, persiana ou blackout abertos. Isso pode causar um prejuízo grande para o proprietário dos objetos que existem neste ambiente. Por exemplo: A televisão de LCD não pode receber a luz do sol, pois pode estragar o líquido polarizador, que pode ocorrer em 1 dia ou em 10 anos. O sofá, objetos de couro, móveis, tapetes e enfeites, de uma sala de visitas, que sofrem com essa incidência de sol, varias vezes e por tempos prolongados, ficam com um aspecto sujo (cores queimadas e/ou manchas irreversíveis) causando constrangimento ao proprietário quando recebe visitas. [MARTINS]

Já no caso das televisões de LCD e Plasma, o proprietário pode sofrer um grande prejuízo podendo chegar a mais de 10 mil reais, que é o valor das televisões deste tipo, com muitas funções e tamanho extraordinário.

A exposição ao sol é muito comum na maioria das residências, tanto em televisões como em computadores, e com isso o número de painéis LCD danificados vem aumentando, de acordo com técnico em reparos de televisões.

Este trabalho utiliza sensor de luminosidade (LDR), como na figura 2.1, sendo assim, a persiana rotaciona dependendo da incidência de luz. O projeto conta também, com controles para girar no sentido horário ou anti-horário a persiana de acordo com a necessidade do usuário.

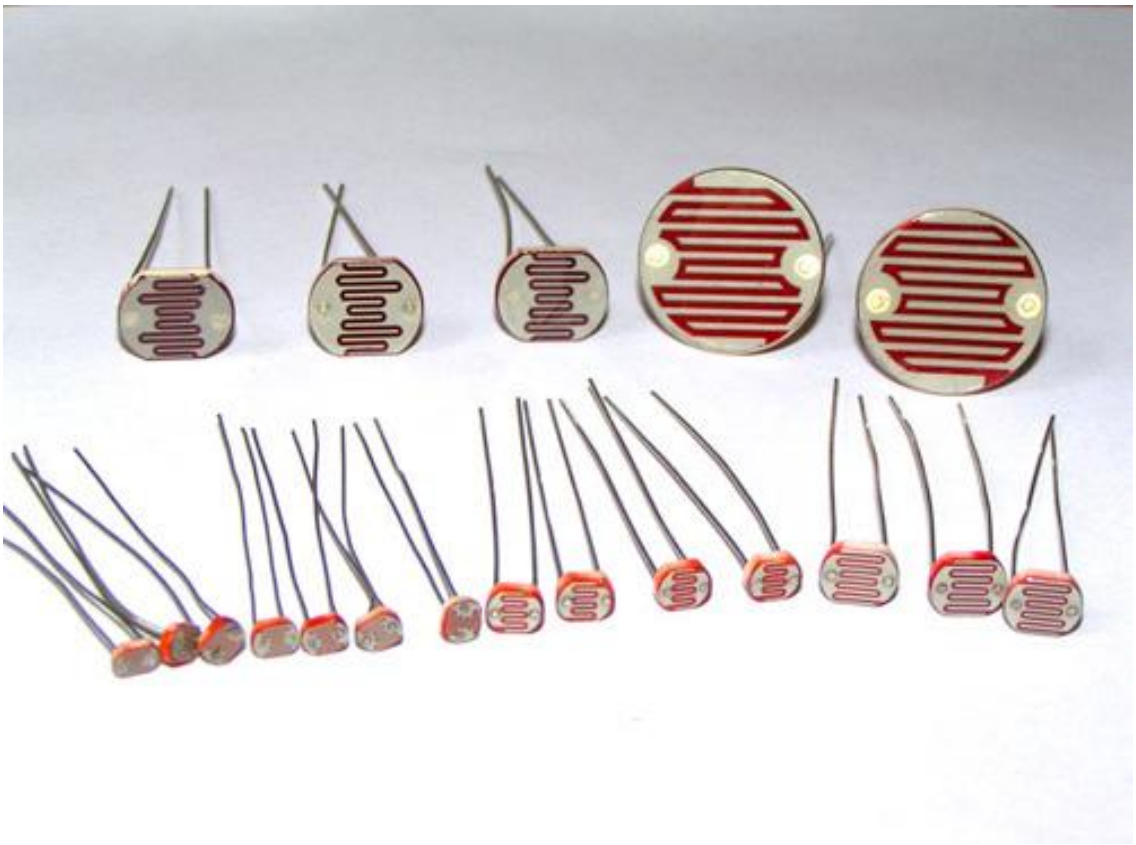


Figura 2.1 – LDR

(http://www.tps.ac.th/~panya/class/electro-magnetic1/ohms-law/index.htm)

É possível que em pouco tempo, os sistemas automatizados estejam presentes em muitas residências, assim como há em outros países, trazendo conforto, economia e segurança para as pessoas.

Como este projeto depende do grau de incidência da luz solar, é necessário um bom sensor de luminosidade (LDR).

O uso das persianas, por sua vez, reduz bastante à possibilidade de uso da iluminação solar natural, impossibilitando um potencial de economia de energia. Mas para compensar há a economia por meio da diminuição do uso de ar-condicionado dentro desse ambiente que está sendo protegido do calor externo.

2.3 - Características das Persianas

Uma persiana funciona com uma mola que é controlada por uma lingueta e uma pequena alavanca de retenção. Elas podem ser consideradas como um tipo de cortina. Possuem função térmica, podem bloquear a entrada indesejada de calor nas épocas de clima mais quente no ano e manter o calor em épocas de clima mais frio. Em ambos os casos, dependendo da cor e do formato, reduzem a luz em graus variados.

O controle da quantidade de luz natural é uma característica especial das persianas que concede muitas vantagens para o usuário. Ela não permite a entrada dos raios solares, o que protege o mobiliário e os valores existentes dentro do ambiente, mas deixam que a claridade chegue, tornando o espaço confortável e aconchegante, contribuindo com o bem estar e a saúde das pessoas presentes no ambiente. Além de criar esses efeitos e sensações, as persianas proporcionam mais privacidade ao espaço onde se encontram, permitem a criação dos mais diversos ambientes: românticos, modernos, aconchegantes ou práticos.

2.4 – Preço comercial

Algumas empresas como a GN TROPICAL - Persianas Externas, Motores e Componentes oferecem este tipo de serviço, porém os preços não são tão acessíveis. Conforme conversa com uma atendente desta empresa, para uma persiana com 2m por 1,5m com motor, mecanismos e o modelo mais barato sairiam por R$ 2.208,20 como na figura 2.2, já modelos mais procurados chegando a mais de quatro mil reais. [TROPICAL, 2010]

Quantidade	Unidade	Código	Produto	Unitário	Total
2,15	M2	200226	Esteira em PVC - Passo 57 - BCO	R$ 180,00	R$ 387,00
1	PC	SU107	Perfil 'U' com 1 Encaixe de Feltro - FOS	R$ 160,00	R$ 160,00
1	UN	KIT003	ACESSORIOS DE INST.P/ PERFIL JANELA	R$ 5,20	R$ 5,20
1	ML	015	Eixo Octogonal em Alumínio 60 mm	R$ 160,00	R$ 160,00
2	PC	100333	MANCAL COM ROLAMENTO 10MM	R$ 18,00	R$ 36,00
1	PC	100109	Ponteira Octagonal para Eixo de Alumínio	R$ 10,00	R$ 10,00
1	KIT	M45-168	Motor20NM-110v/60hz-botoeira 52 kg Reven	R$ 1.450,00	R$ 1.450,00

Total: **R$ 2.208,20**

Figura 2.2 – Orçamento nº 1087 - 25/08/2010

[TROPICAL, 2010]

A persiana descrita acima, apesar da botoeira, não possui o sensor e por tal motivo não saiu por um valor mais elevado, pois para colocar o sensor teria o custo do microcontrolador, circuito, placa e do próprio sensor, além da mão de obra para criação desse sistema e o lucro da empresa.

Com esse custo a mais, ainda não seria possível fazer uma persiana giratória para medir a quantidade de luminosidade e regular o ângulo de abertura, pois o custo descrito no orçamento acima é de uma persiana do tipo esteira, uma persiana que possua a propriedade do giro sairia mais caro. Uma persiana com essas características somente seria acessível por famílias de classes superiores a classe média.

CAPÍTULO 3 – REFERENCIAL TECNOLÓGICO

Este capítulo tem como objetivo expor um pequeno referencial teórico para o projeto final. As principais teorias relevantes são expostas aqui.

3.1 - Motor de Passo

O motor de passo é um tipo de motor elétrico, isto é, ele é uma máquina que transforma energia elétrica em energia mecânica auxiliando em algumas atividades que não seriam possíveis sem ele.

Os motores de passo são projetados a fim de se ter um maior controle sobre a rotação do motor através de passos realizados pelo motor em cada volta que ele realiza. Uma importante característica desses motores, é sua compatibilidade com sistemas digitais. [TORO, 1994]

São comumente utilizados em sistemas de controle digital, onde ele recebe comandos na forma de uma sequencia de pulsos para girar ou mover um objeto por uma distancia precisa. Nas impressoras existem motores de passo para alinhamento do papel e do cabeçote de impressão, máquinas cirúrgicas que devem controlar distâncias também utilizam motores de passo para ter uma precisão cirúrgica, com essas máquinas os médicos podem operar pacientes a distancia, diminuindo um pouco do risco de infecção, já que as incisões são calculadas para serem o menor possível e são realizadas pelo motor de passo com precisão. [TORO, 1994]

Pela sua característica de transformar as informações digitais recebidas pelos computadores e microcontroladores em força mecânica, é bastante utilizado em aplicações como as citadas. [TORO, 1994]

O motor de passo é composto de um motor e um estator, onde o primeiro é a parte interna e que gira do motor, e a segunda é a parte externa e estática do motor, onde são enroladas as quatro bobinas, pequenos cilindros de cobre, que geram um campo magnético girante. No acoplamento entre o estator e o rotor, deve restar um pequeno espaço vazio para ocorrer o movimento de rotação e deve ser feito através de um sistema de engrenagens, de modo a aumentar a precisão do motor diminuindo a angulação dos passos. [MATOS, 2009]

Na figura 3.1 são apresentadas as partes do motor.

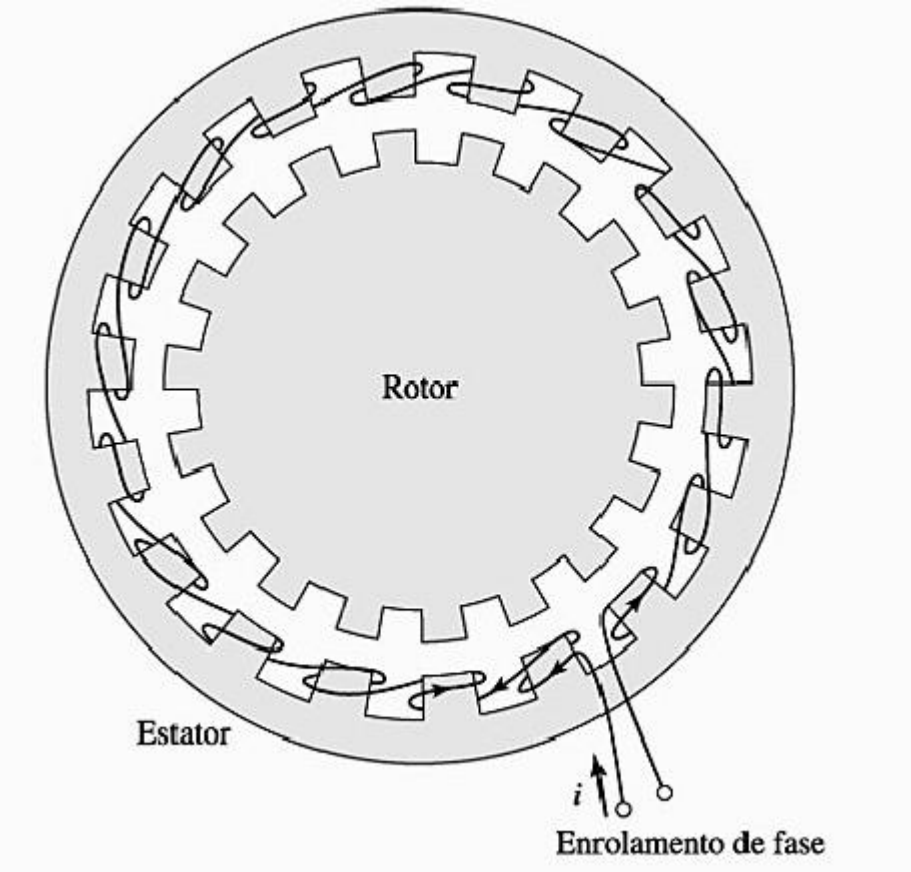


Figura 3.1 – Motor de passo: Rotor, Estator e Enrolamento de fase.

[GAMEIRO]

O motor de passo permite o controle das polaridades de cada bobina de acordo com a necessidade e com isso o motor poderá girar de diversas formas possíveis, isso permite um controle mais preciso dos movimentos do motor.

3.1.1 - Tipos Básicos

Os motores de passo são classificados de acordo com as características de sua construção, eles são divididos em três tipos principais, o de relutância variável, o de Imã Permanente, e o Híbrido. [FITZGERALD, 2006]

O motor de passo de relutância variável tem como característica um estator e um rotor de polos salientes, sendo um estator laminado e um rotor com várias polaridades. Cada uma

das fases é colocada em dois polos opostos do rotor ligados geralmente em série. Esse tipo de motor possui em sua maioria passos de 5 a 15 graus. [GAMEIRO]

Na figura 3.2 é representado um motor de passo de relutância variável.

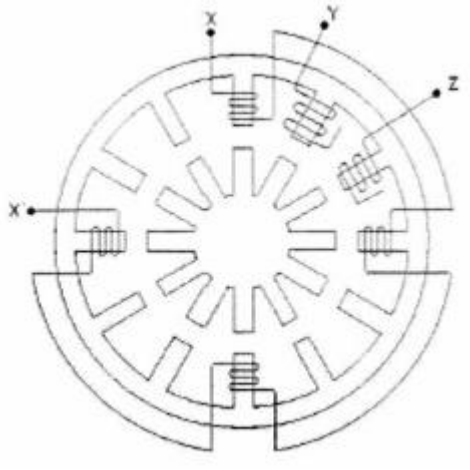


Figura 3.2 – Motor de passo de Relutância Variável

[FITZGERALD, 2006]

Quando a fase X for energizada, os quatro dentes do estator, que estão com a fiação da fase X, se alinharão com os quatro dentes do rotor por meio de atração magnética, neste ponto o motor estará parado, para que ele dê o primeiro passo a fase X deverá ser desligada e a fase Y deverá ser ligada, assim os quatro dentes do rotor se alinharão com os dentes do estator e o motor terá dado um passo no sentido horário, no caso da Figura 4 o passo será de 15 graus. Para que o motor continue girando, é só desligar a fase Y e energizar a fase Z, depois energizar novamente a fase X, e assim sucessivamente. [FITZGERALD, 2006]

Outro tipo de motor de passo é o de Ímã Permanente, apresentado na Figura 3.3 a seguir. Esse motor se caracteriza por possuir um rotor sem dente e magnetizado perpendicularmente ao eixo. São construídos com materiais alcalinos ou ferrosos, e devido à sua magnetização perpendicular possuem torque estático não nulo. [MATOS, 2009]

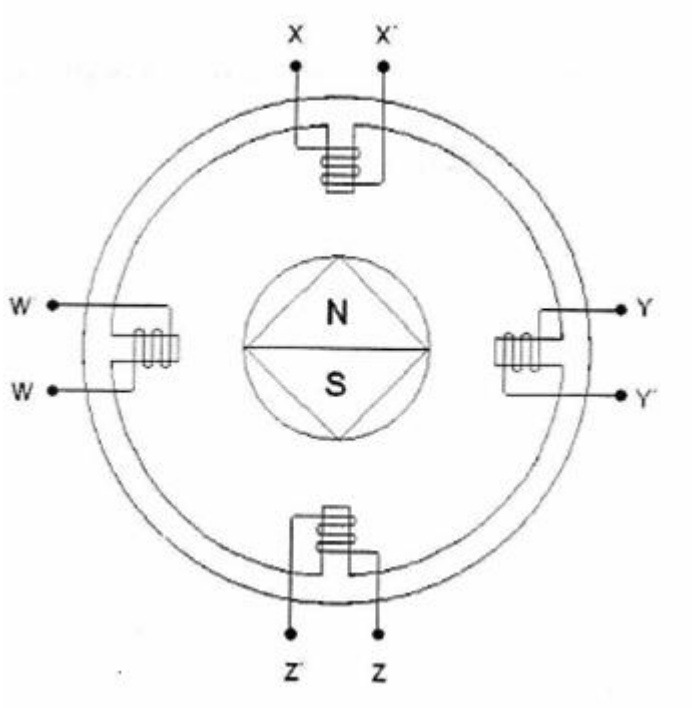


Figura 3.3 – Motor de passo de Ímã Permanente

[FITZGERALD, 2006]

Quando energizamos a fase X o rotor aponta para cima, quando energizamos a fase Y, o rotor apontará para a direita, fazendo com que o motor gire em um passo de 45 graus no sentido horário. Para o motor continuar a girar no sentido horário, é necessário energizar as fases XYZW sequencialmente. Os motores de Ímã Permanente possuem na maioria das vezes taxas de passo relativamente baixas, que ficam na ordem de 45 ou 90 graus, em contrapartida oferecem um torque mais elevado. [FITZGERALD, 2006]

O ultimo tipo de motor de passo é o Híbrido, as características desse motor consistem em mesclar as características dos outros 2 tipos de motores de passo, utilizando-se somente das características desejáveis. Possui um alto torque e pode operar em altas velocidades de passo como o motor de passo que utiliza imã permanente e possui ângulos de passo pequenos de 0,9 a 5 graus como o motor de passo que utiliza relutância variável. Na maioria dos casos, são formados por polos conseguidos através de dois enrolamentos, conforme a figura 3.4, desta forma é possível à utilização de apenas uma fonte de alimentação. [MATOS, 2009]

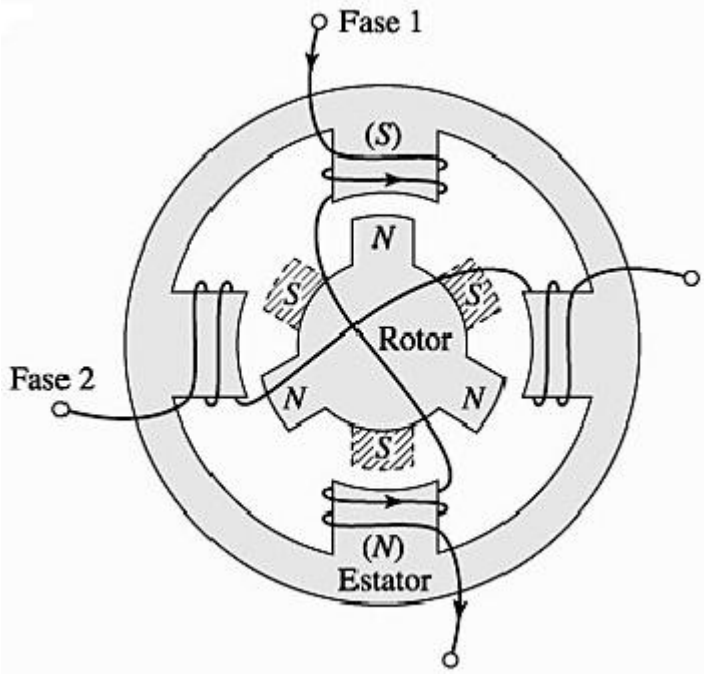


Figura 3.4 – Motor de passo Híbrido

[FITZGERALD, 2006]

A Figura 3.4 representa um motor de passo híbrido onde a fase 1 está energizada, de tal forma que a parte superior do estator seja um polo sul e a parte inferior seja o polo norte, esta configuração faz com que o rotor, constituído de Ímã Permanente, alinhe-se com o estator conforme a Figura. Para que o motor dê passos, e consequentemente gire, deve-se retirar a energia da fase 1 e energizar a fase 2, de tal forma que ao lado esquerdo do estator seja o polo sul e o lado direito seja o polo norte, assim o motor girará 30 graus no sentido horário, é importante observar que se as polaridades do estator fossem trocadas o motor giraria no sentido anti-horário. Para continuar a girar, deve-se desligar a fase 2 e energizar a fase 1, porém os polos das fases devem estar invertidos em relação à situação inicial, e assim sequencialmente. [FITZGERALD, 2006]

3.1.2 - Modos de Funcionamento

Os motores de passo também podem ser usados de três modos distintos, essa variação está relacionada ao número de passos por revolução que o motor faz, e ao modo de energização utilizado. [BRITES, 2008]

O primeiro modo é o passo normal (*Full-Step*), nesta operação é utilizado o passo normal do motor, ou seja, o passo completo. A utilização deste modo pode ser feita de duas

formas, na primeira forma o motor funciona com uma única fase energizada por vez. Esta forma é indicada quando não é importante nem o torque nem a velocidade, é requerido uma menor potência. A segunda forma que o passo normal acontece é quando energizamos duas fases por vez, desta forma diminuímos o problema com o torque e com a velocidade, porém é necessário o dobro da potência para o funcionamento do motor, em relação à utilização de apenas uma fase. [MATOS, 2009]

O segundo modo de energização de um motor de passo é o Meio-Passo (*Half-Step*), este modo consiste na alternância da energização de uma fase e energização de duas fases ao mesmo tempo, resultando em passos com a metade do tamanho de um passo normal, aumentando assim a precisão do motor. O torque do motor, neste caso, varia de acordo com o número de fases energizadas. Com essa disposição, este motor se encontra livre de ressonância, podendo utilizar uma grande faixa de velocidade e de carga. [MATOS, 2009]

O terceiro e último modo é o Micro-Passo (*Micro-Step*), neste modo o passo natural de um motor pode ser dividido em outros vários passos menores. Os motores de Micro-Passos são produzidos adicionando nas duas bobinas correntes de acordo com o seno e o cosseno. Este modo proporciona uma resolução maior, e é utilizado quando há necessidade de movimentos sem trancos e de muita precisão. [MATOS, 2009]

3.1.3 - Formas de Operação

Os motores de passo podem ser classificados como bipolar ou unipolar, dependendo do número de enrolamentos por fase do motor. [BRITES, 2008]

O motor de passo unipolar, apresentado na Figura 3.5 a seguir, tem dois enrolamentos por fase, sendo um para cada sentido da corrente. Desta forma sua configuração torna-se mais simples de trabalhar já que cada bobina recebe o sinal independente, sem ter que se preocupar com sentido da corrente.

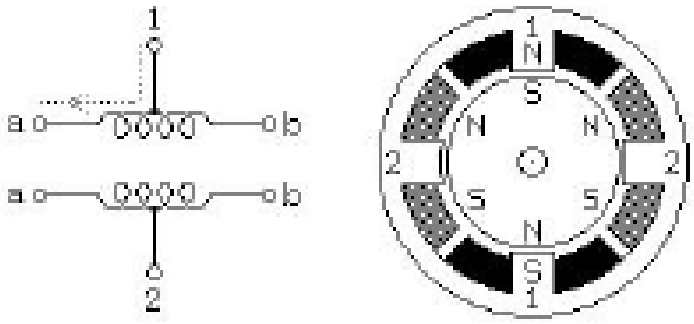


Figura 3.5 – Motor de passo Unipolar

[BRITES, 2008]

O motor de passo bipolar, apresentado na Figura 3.6 a seguir, tem um único enrolamento por fase. Sendo assim, a corrente de um enrolamento necessita ser invertida para que inverta o polo magnético, com isso seu circuito é um pouco mais complicado que o circuito do motor de passo unipolar.

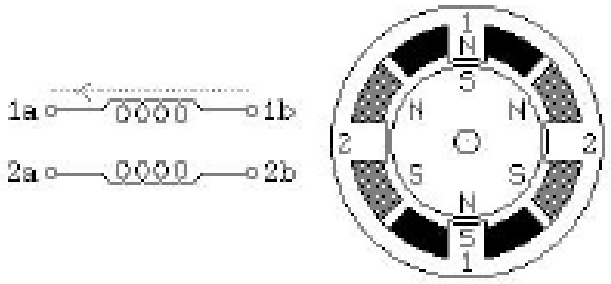


Figura 3.6 – Motor de passo Bipolar

[BRITES, 2008]

3.1.4 - Motor Utilizado no Projeto

O motor utilizado no projeto é o motor de passo modelo SM7.5-A12P-NP, consiste em um motor Unipolar, de Ímã Permanente, com passo de 7,5 graus, utiliza uma tensão nominal de 12 V e com modo Passo Normal ou *Full-Step*.Segue figura 3.7, com 2 fotos de ângulos diferentes do motor utilizado.

O motor utilizado é unipolar, pois é mais simples de configurar; de Ímã Permanente, pois possui o torque estático não nulo; a tensão de 12 V para uma maior potência; e de Passo Normal (*Full-Step*) por não necessitar de uma precisão maior.

Figura 3.7 – Motor Utilizado no Projeto

[AUTOR]

3.2 - Sensores Óticos

"São componentes eletrônicos de sinalização e comando que executam detecção de qualquer material sem que haja contato mecânico entre eles." [THOMAZINI, 2005]

Esse tipo de sensor toma como base um emissor, que envia uma luminosidade e um receptor que recebe essa luminosidade e se for forte o bastante para que o seu estado seja alterado, acontece à alteração dessa saída.

3.2.1 - LDR (Light Dependent Resistor)

LDR significa Resistor Dependente de Luz ou fotorresistor. Esse fotorresistor é usado como sensor de luz. [THOMAZINI, 2005]

Com a incidência da luminosidade, algumas substâncias têm a sua resistência alterada dependendo da quantidade de luminosidade e com isso ocorre a liberação de portadores de carga que ajudam a condução da corrente elétrica como na figura 3.8. [THOMAZINI, 2005]

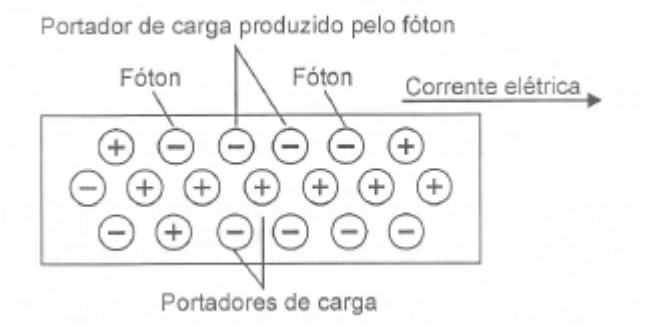


Figura 3.8 – Representação de portadores de carga que reduzem a resistência elétrica de determinados materiais.

[THOMAZINI, 2005]

Nos detectores de fótons, os fótons individuais incidentes interagem com os elétrons dentro do material. Na absorção de um fóton liberta um elétron do material, ou seja, a absorção de fótons aumenta o número de portadores de carga do material ou muda a sua mobilidade. Como os portadores de carga ficam soltos, a resistência do material diminui.

Os LDRs são formados de sulfeto de cádmio, que tem como fórmula química o CdS. Essas células de CdS são também chamadas de fotocélulas de sulfeto de cádmio. Os LDRs apresentam uma resistência elevada no escuro, chegando a milhões de ohms e apresentam uma resistência bem menor quando na claridade, chegando a algumas centenas de milhares de ohms. Esse comportamento é demonstrado na figura 3.9. [THOMAZINI, 2005]

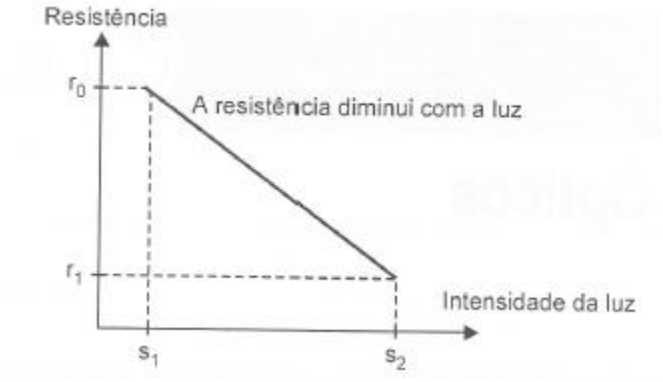


Figura 3.9 – A variação de resistência com a luz.

[THOMAZINI, 2005]

A superfície do LDR é composta de sulfato de cádmio, pequenas trilhas do material condutor, eventualmente ouro, que se entrelaçam com o material condutor de modo a

aumentar a superfície de contato e com isso chegar a uma maior capacidade de corrente e a uma maior sensibilidade. [THOMAZINI, 2005]

A luminosidade chega à superfície do LDR por uma janela de plástico que o envolve. Para ter acesso ao sensor, têm-se dois terminais ligados ao LDR, esses terminais são ligados ao circuito que o utilizará. Na figura 3.10 segue um LDR com as indicações desses terminais e dos símbolos utilizados. [THOMAZINI, 2005]

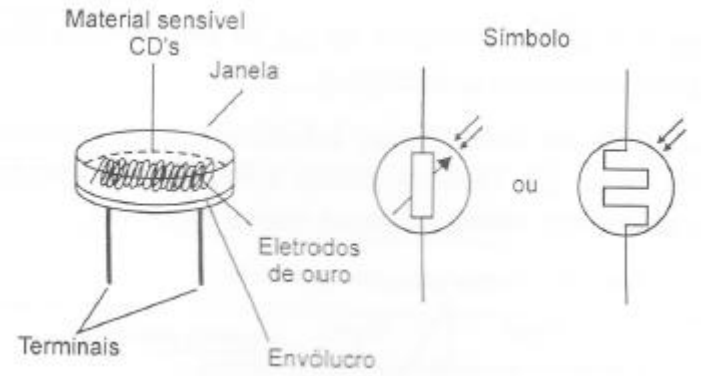


Figura 3.10 – LDR, aspecto e símbolo.

[THOMAZINI, 2005]

A corrente que passa pelos LDRs pode passar em qualquer um dos sentidos e mesmo assim ainda tem a mesma resistência. Por isso são chamados de componentes bipolarizados. [THOMAZINI, 2005]

Existem LDRs com tamanhos e formatos diferentes como os que aparecem na figura 3.11.

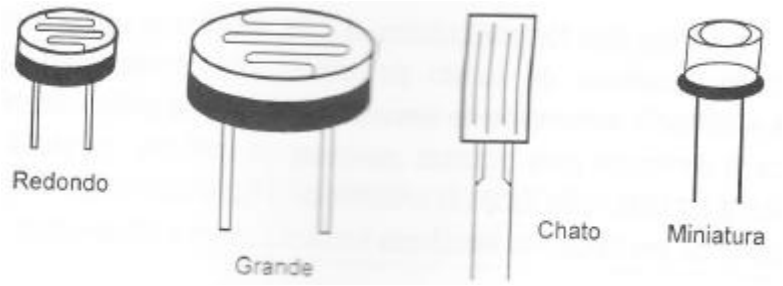


Figura 3.11 – Alguns tipos comuns de LDRs encontrados no comércio.

[THOMAZINI, 2005]

O LDR com uma superfície maior tem mais sensibilidade e também uma maior capacidade de liberar calor, por isso quanto maior a superfície dele, mais intensa a corrente controlada por ele. [THOMAZINI, 2005]

Um exemplo é o LDR grande que pode controlar alguns dispositivos como relés sensíveis e até mesmo lâmpadas de baixa potência. Já os LDRs pequenos devem utilizar circuitos amplificadores, para que a sua sensibilidade seja aumentada. “Por exemplo, um LDR típico de um cm. A resistência máxima, no escuro, desse componente deve ficar entre 1MΩ e 10MΩ, dependendo do tipo, e a resistência com iluminação máxima (ambiente) deve ficar entre 75Ω e 500Ω tipicamente.”. “Para a verificação destas características pode ser feito um teste utilizando um multímetro. Com o LDR iluminando temos a resistência mínima, e cobrindo o LDR de modo que nenhuma luz o atinja, temos a resistência máxima.”. [THOMAZINI, 2005]

O tempo de resposta de um fotorresistor é representado como o tempo necessário para a condutância subir 63% do valor de pico após a célula ter sido iluminada (tempo de subida), e o tempo necessário para a condutância descer a 37% do valor de pico após ter sido removida a luz (tempo de descida). “O tempo de resposta depende do nível de iluminação, da resistência de carga, da temperatura ambiente e das “condições pré-históricas”. O tempo de subida diminui conforme a resistência de carga é aumentada, no entanto o tempo de descida aumenta.”. [THOMAZINI, 2005]

Normalmente, quando um fotorresistor é mantido no escuro por certo período de tempo antes do uso, sua condutância será maior se for comparado com um fotorresistor mantido num certo nível de luz. Essa diferença é chamada de “efeito pré-histórico”. A extensão do efeito é maior para CdS do que para CdSe. Esse efeito não é significativo para aplicações gerais, entretanto, quando o fotorresistor é utilizado em aplicações em que a iluminação é menor do que um lux, o efeito deve ser levado em consideração. [THOMAZINI, 2005]

O LDR é um dispositivo lento. Enquanto outros tipos de sensores como os fotodiodos e os fototransistores podem perceber variações muito rápidas de luz, em frequências que chegam a dezenas ou mesmo centenas de megahertz, o LDR tem um “tempo de recuperação” muito longo. Estando totalmente iluminado e sendo a luz cortada, ocorre um determinado intervalo de tempo para que a resistência, inicialmente no valor mínimo, volte ao valor

Maximo. Este comportamento é demonstrado na figura 3.12, que apresenta a faixa de operação do LDR. [THOMAZINI, 2005]

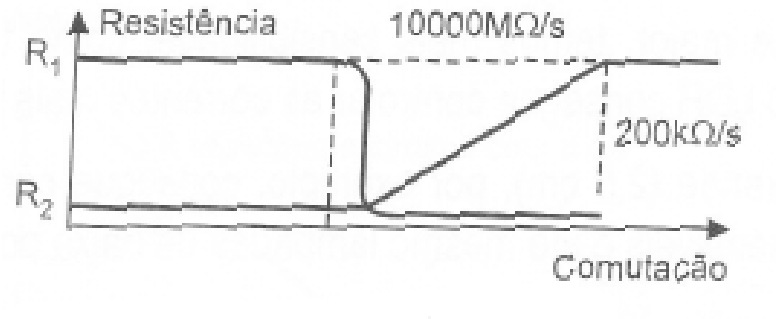


Figura 3.12 – Faixa de operação do LDR. R1 se refere à iluminação e R2 ao escurecimento sobre um LDR.

[THOMAZINI, 2005]

A baixa taxa de comutação do LDR impede que ele seja usado em sensores do tipo leitor de cartões perfurados, códigos de barras ou sistemas de alarmes modulados. No entanto, em aplicações mais simples, cujos tempos necessários para a atuação sejam maiores, como alarmes, brinquedos, sensores de luz ambiente, detectores de níveis de iluminação, fotômetros, ele é muito útil. [THOMAZINI, 2005]

3.3 - Microcontrolador

O microcontrolador é um componente eletrônico que pode ser programado de acordo com a sua utilidade e com a necessidade do programador, e é utilizado no controle de processos lógicos. [SOUZA, 2005]

O controle realizado pelo microcontrolador envolve periféricos como: LEDs (*Light Emiting Diode*), botões, displays de sete segmentos, displays de cristal líquido (LCD), resistências, relês, sensores diversos (pressão, temperatura, luminosidade, etc.) e muitos outros. Esses controles são chamados de lógicos, pois a operação de cada um desses periféricos é baseada nas ações que devem ser tomadas de acordo com o estado de entrada ou de saída desses periféricos. [SOUZA, 2005]

O microcontrolador é programável, pois toda a lógica de operação de que acabamos de falar é estruturada na forma de um programa e gravada dentro do componente. Depois disso, toda vez que o microcontrolador for alimentado, o programa interno será executado. Quanto à "inteligência" do componente, pode-se associá-la à Unidade Lógica Aritmética (ULA), pois é nessa unidade que todas as operações matemáticas e lógicas são executadas. Quanto mais poderosa a ULA do componente, maior sua capacidade de processar informações. [SOUZA, 2005]

Nesta definição, o microcontrolador ganhou ainda o adjetivo "pequeno", pois em uma única pastilha de silício encapsulada (popularmente chamada de CI (Circuito integrado) ou CHIP), temos todos os componentes necessários ao controle de um processo, ou seja, o microcontrolador está provido internamente de memória de programa, memória de dados, portas de entrada e/ou saída paralela, timers, controladores, comunicação serial, PWMs, conversores analógico-digitais, etc. Esta é uma das características fundamentais que diferencia os microcontroladores dos microprocessadores, pois os últimos, apesar de possuírem uma ULA muito mais poderosa, não possuem todos esses recursos em uma única pastilha. [SOUZA, 2005]

Os microcontroladores estão presentes no dia-a-dia de todos, em lugares que muitos nem sabem que se trata de um microcontrolador fazendo as funções básicas de certo objeto comum. Esses microcontroladores já estão presentes há bastante tempo no dia-a-dia e alguns desses exemplos são: Eletrodomésticos, videocassetes, DVDs, celulares, alarmes e até mesmo em brinquedos, entre uma gama imensa de objetos que fazem esse uso.

3.3.1 - A Família PIC

Os microcontroladores PIC apresentam uma estrutura de máquina interna do tipo Havard, enquanto grande parte dos microcontroladores tradicionais apresenta uma arquitetura tipo Von-Neumann. A diferença está na forma como os dados e o programa são processados pelo microcontrolador. Na arquitetura tradicional, tipo Von-Neumann, existe apenas um barramento (bus) interno (geralmente de oito bits), por onde passam as instruções e os dados. Já na arquitetura tipo Havard existem dois barramentos internos, sendo um de dados e outro de instruções. No caso dos microcontroladores PIC, o barramento de dados é sempre de oito bits e o de instruções pode ser de 12, 14 ou 16 bits, dependendo do microcontrolador. Esse tipo de arquitetura permite que, enquanto uma instrução é executada, outra seja "buscada" da memória, o que torna o processamento mais rápido. Além disso, como o barramento de

instruções é maior do que oito bits, o OPCODE da instrução já inclui o dado e o local onde ela vai operar (quando necessário), o que significa que apenas uma posição de memória é utilizada por instrução, economizando assim muita memória de programa. [SOUZA, 2005]

Desta forma, pode-se observar que dentro da palavra do OPCODE, que pode ser de 12, 14 ou 16 bits, não sobra muito espaço para o código da instrução propriamente dito. Por isso, os PICs utilizam uma tecnologia chamada RISC, que significa Reduced Instruction Set Computer (Computador com set de instruções reduzidas). Desta forma, os PICs possuem mais ou menos 35 instruções (o número correto varia de acordo com o microcontrolador), muito menos que os microcontroladores convencionais (CISC) que chegam a possuir mais de cem instruções. Isso torna o aprendizado muito mais fácil e dinâmico, mas, por outro lado, implica no fato de que muitas funções devem ser "construídas", pois não possuem uma instrução direta, exigindo maior habilidade do programador. [SOUZA, 2005]

3.3.2 - Microcontrolador PIC16F876A

As características do microcontrolador usado no projeto são apresentadas na Figura 3.13.

Key Features	PIC16F876A
Operating Frequency	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	8K
Data Memory (bytes)	368
EEPROM Data Memory (bytes)	256
Interrupts	14
I/O Ports	Ports A,B,C
Timers	3
Capture/Compare/PWM modules	2
Serial Communications	MSSP, USART
Parallel Communications	—
10-bit Analog-to-Digital Module	5 input channels
Analog Comparators	2
Instruction Set	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin MLF

Figura 3.13 – Características do microcontrolador

Fonte: Datasheet do Microcontrolador PIC16F876A (em anexo)

Os pinos do microcontrolador são responsáveis pela interação entre o microcontrolador e o resto do circuito através do envio de sinais elétricos. A pinagem do PIC 16F876A é apresentada na Figura 3.14.

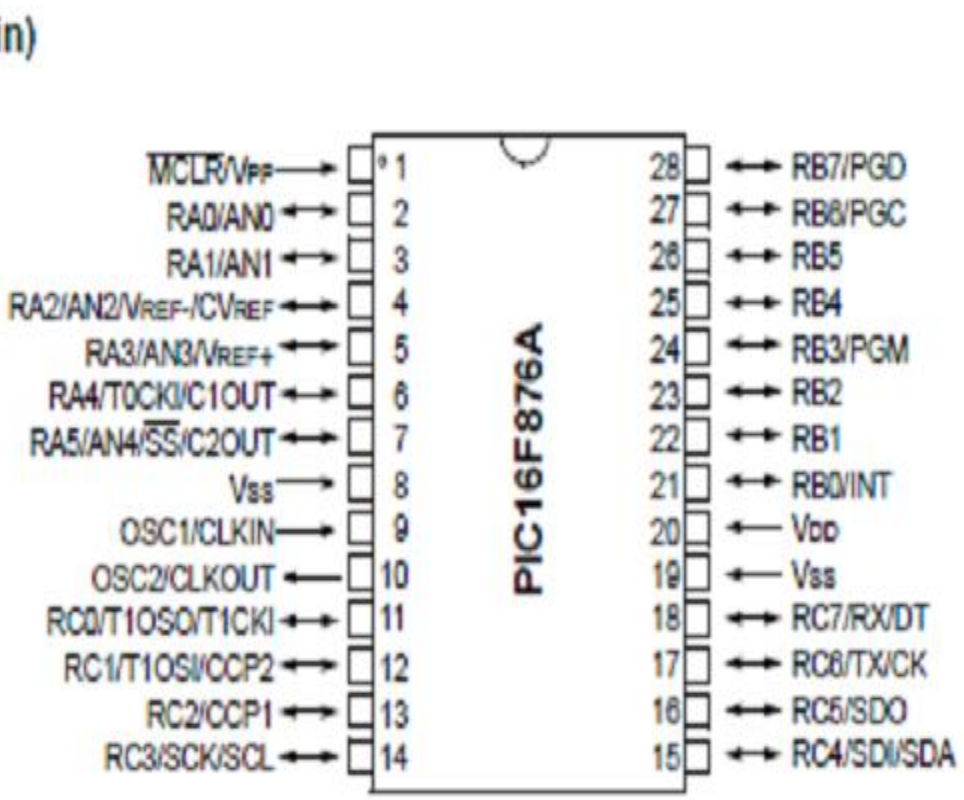


Figura 3.14 – Pinagem do microcontrolador

Fonte: Datasheet do Microcontrolador PIC16F876A (em anexo)

Cada pino descrito anteriormente tem uma função distinta, no datasheet em anexo são descritas as características e as funções de cada pino.

3.3.3 - Interrupção

Este PIC possui diversas interrupções e todas geram um desvio para o vetor de interrupção (0004h), desde que configuradas para isso. São divididas em convencionais e as de periféricos, como a dos conversores A/D (Analógico / Digital). A interrupção A/D ocorre quando uma conversão A/D é completada. [LAVINIA, 2005]

Como os “dados” que são recebidos de fora dos componentes são analógicos e para que o microcontrolador possa entender os dados, eles não podem ser analógicos, esses devem ser transformados em digitais. Para realizar essa transformação, de analógico em digital, utiliza-se o conversor A/D. Quanto maior a quantidade de bits no conversor, maior é a sua

resolução e a sua precisão. O PIC 16F876A possui um conversor interno de dez bits, mas existem outros modelos com oito ou doze bits. [LAVINIA, 2005]

3.4 - Circuitos Integrados e Componentes

Neste tópico, são descritas algumas informações técnicas dos componentes utilizados na implementação do sistema.

3.4.1 - Display

Este projeto usa o Display de Cristal Líquido com dimensões 16x2, e numeração AGM-1602B. O Display é utilizado para interação com o usuário do sistema, nele são escritas as ações efetuadas pelo sistema. Ele possui para escrita, duas colunas com possibilidade de utilização de dezesseis caracteres em cada uma delas. A Figura 3.15 é uma foto do display mencionado.

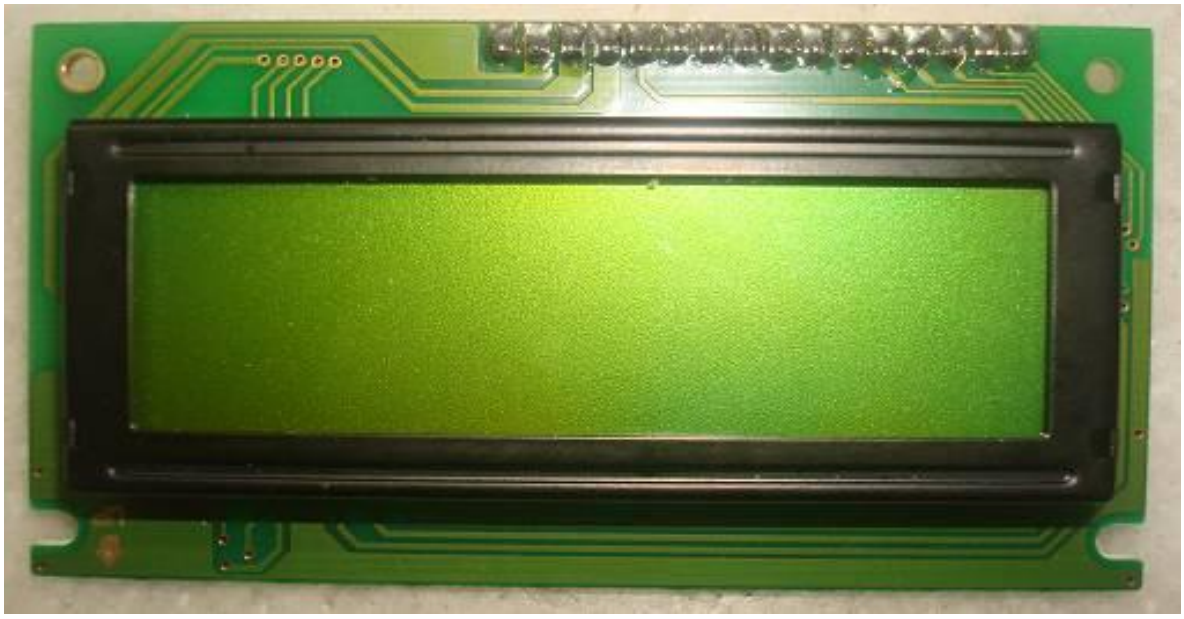


Figura 3.15 – Display

[AUTOR]

Este display possui dezesseis pinos dispostos conforme aparece nas características do AGM-1602B em anexo, sendo que dois deles são para funcionamento do *Black Light* que é o acendimento com luz esverdeado do fundo do display, porém nem todos os displays os possuem.

3.4.2 - CI L298N e Ponte H

Ponte H é um circuito eletrônico que permite que um motor rode tanto no sentido horário quanto no sentido anti-horário. Estes circuitos são encontrados em circuitos prontos, como no CI L298N.

O circuito é construído com quatro chaves que são acionadas de forma alternada. Para cada configuração das chaves o motor gira em um sentido.

A Figura 3.16 ilustra um exemplo de circuito elétrico que é uma ponte H.

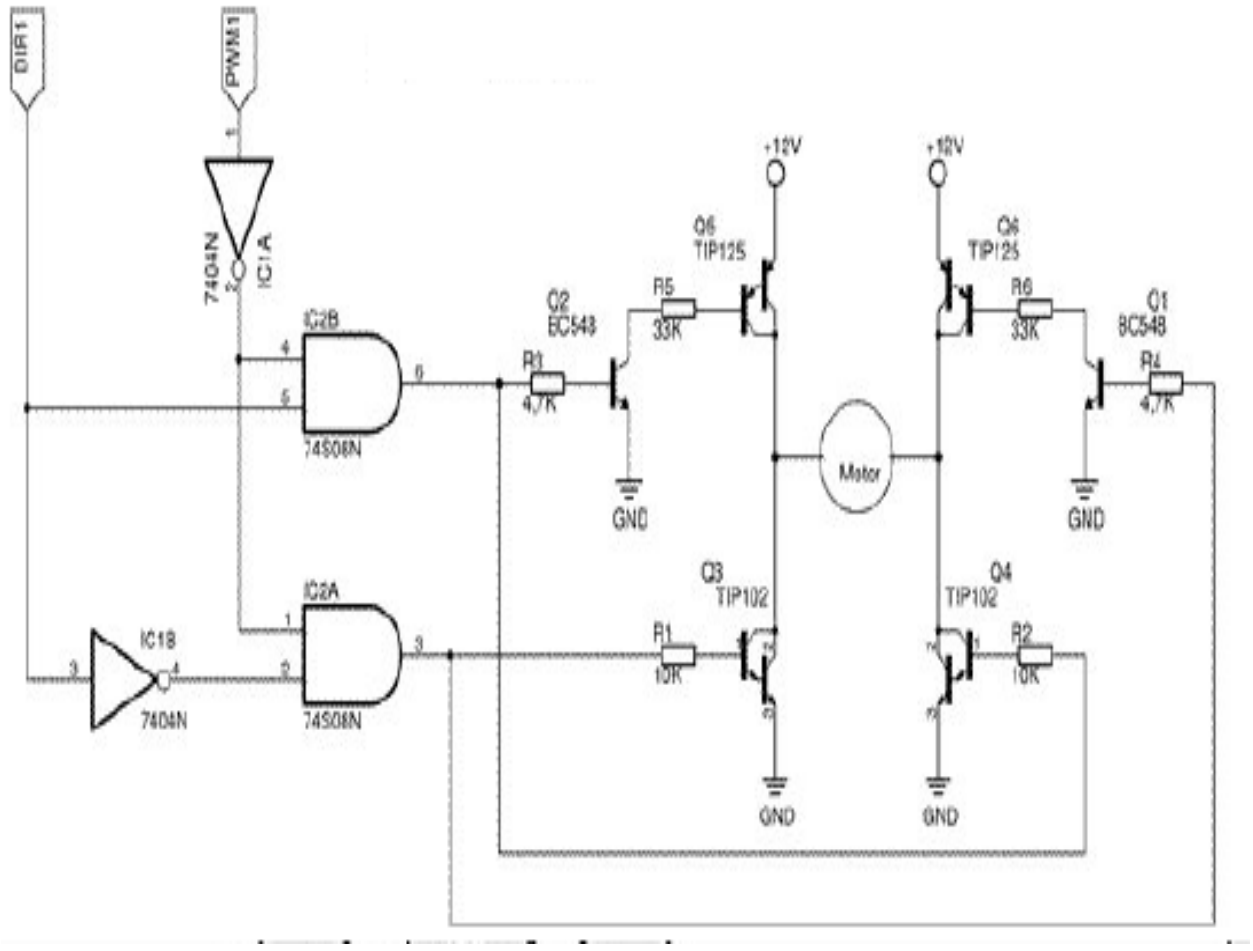


Figura 3.16 – Ponte H

[BRITES, 2008]

Para construção da ponte H pode ser utilizado qualquer tipo de componente que simule uma chave liga-desliga como transistores e relés. [BRITES, 2008]

O L298 é um circuito monolítico integrado de 15 ligações, ele é um excitador de tensão com duas Ponte-H, já comentada anteriormente, que são projetadas para aceitar

padrões lógicos TTL (Transistor-Transistor-Logic), tornando possível controlar cargas indutivas. A figura 3.17 ilustra a pinagem deste CI (Circuito integrado).

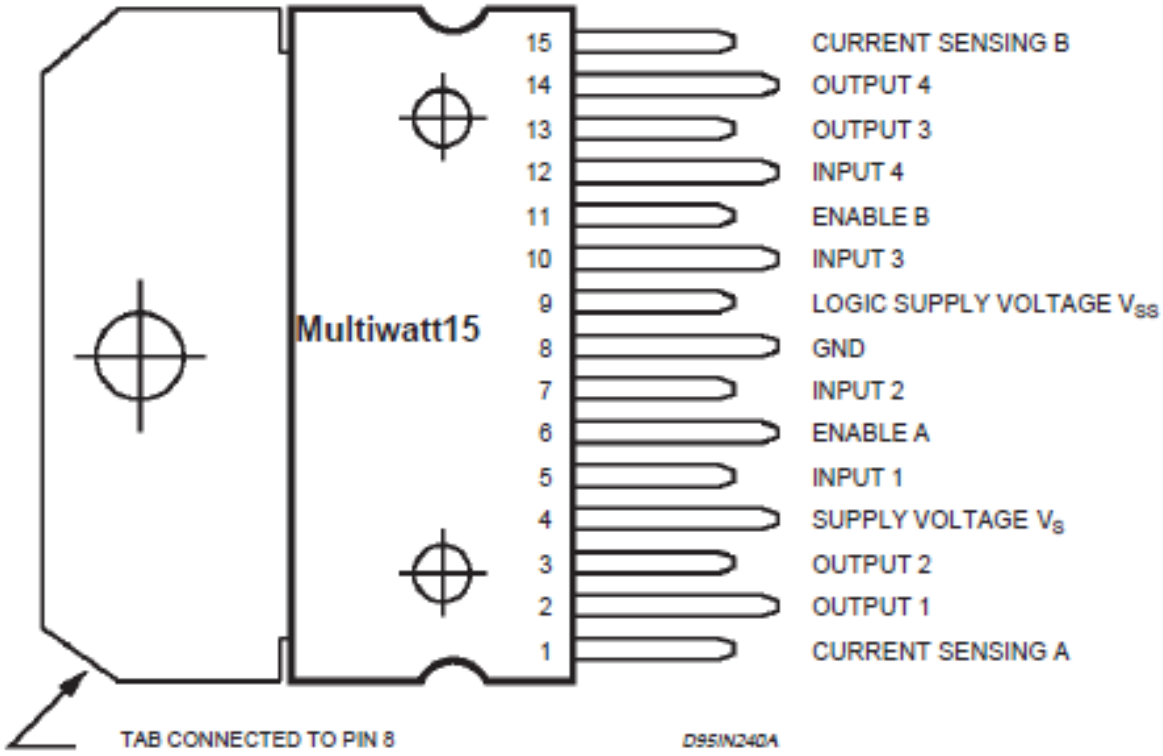


Figura 3.17 – Pinagem do CI L298N

Fonte: Datasheet do CI L298N (em anexo)

Sendo assim, além desse circuito ser composto por duas Ponte-H, recebe os dados direto do microcontrolador em formato TTL, e enviar ao motor de passo a tensão desejada para o seu funcionamento.

O CI L298N pode operar com tensão de saída de até 46 volts, tornando-o útil tanto para os motores de passo unipolar como os bipolares, ele aguenta correntes de até 4 ampéres e tem baixa tensão de saturação, este CI também possui proteção contra temperatura excedente e o valor “0” dele é ativo com tensão de entrada de até 1,5 volts, tornando-se eficiente contra ruídos.

A Figura 3.18 ilustra o diagrama de blocos do CI L298N, onde é possível ver as duas Ponte-H.

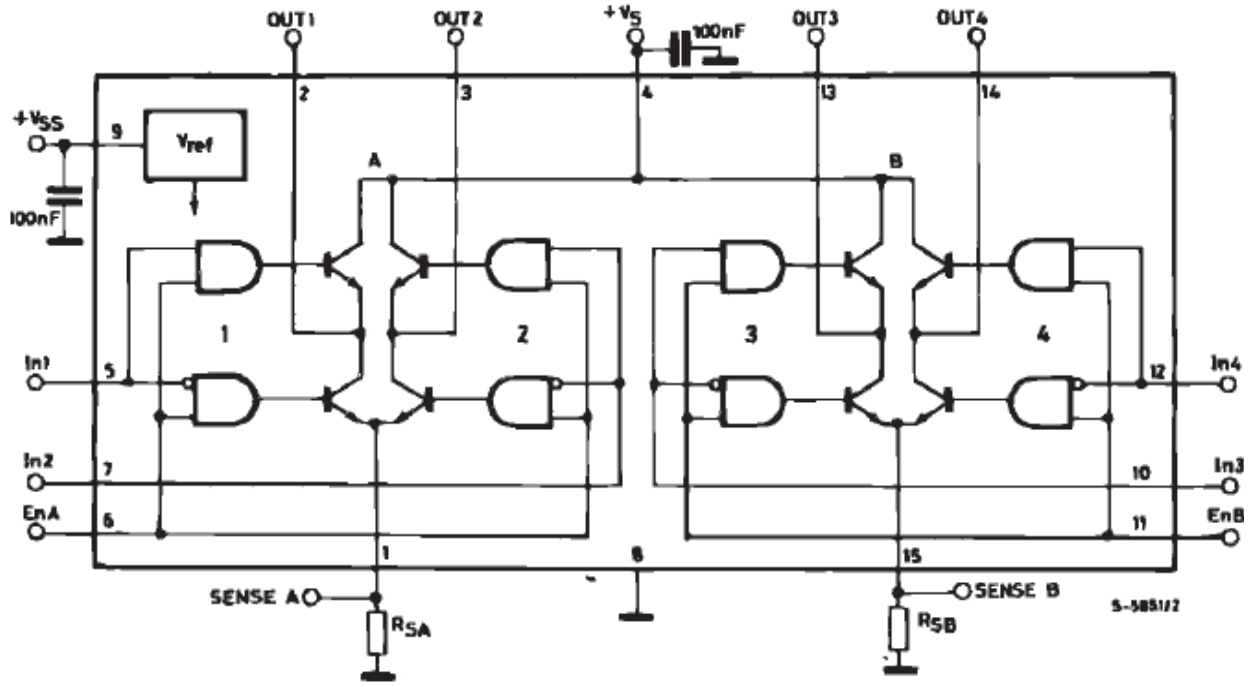


Figura 3.18 – Diagrama de Blocos L298

Fonte: Datasheet do CI L298N (em anexo)

3.5 - Programação

O desenvolvimento de programas para serem utilizados em microcontroladores e microprocessadores complica muito quando a complexidade do sistema a ser desenvolvido aumenta, portanto o programador tem um desgaste proporcional à complexidade. [PEREIRA, 2009].

No início, os dispositivos programáveis eram programados com códigos de máquina. Esses códigos de maquina eram códigos inseridos por dispositivos como teclado, leitora de cartões, fitas perfuradas ou discos magnéticos. As informações desses códigos eram formadas por dígitos binários, para que fossem entendidos e executados pelas maquinas. [PEREIRA, 2009]

A programação em código de máquina era muito mais complexa que a programação que conhecemos hoje em dia, com isso o tempo gasto e o custo para desenvolver uma aplicação era maior. [PEREIRA, 2009]

3.5.1 - Linguagem de Programação Assembly

O surgimento de novas formas de programação veio da necessidade cada vez maior de sistemas complexos, para isso criara a linguagem Assembly. [PEREIRA, 2009]

A linguagem Assembly utiliza mnemônicos para representar os códigos de máquina que eram utilizados antes. Para que as máquinas pudessem entender os comandos a serem executados, esse código era traduzido para a linguagem de máquina por um programa chamado Assembler (montador) e somente depois introduzido na máquina. [PEREIRA, 2009]

Com o Assembler, o trabalho de quem programa os microcontroladores e microprocessadores foi reduzido, pois os comandos deixaram de ser em binário e passaram a ser comandos utilizando letras, números e sinais. Mesmo com essa facilidade o problema não foi totalmente solucionado, pois o Assembler é considerado uma linguagem de baixo nível, ou seja, não possuí nenhuma função diferente das definidas no conjunto de instruções do processador.

3.5.2 - Linguagem de Programação C

A linguagem C foi criada em 1972 por Dennis Ritchie, da Bell *Laboratories*, segundo Fábio Pereira, essa linguagem é considerada uma linguagem de nível intermediário entre o Assembler e as linguagens de alto nível.

"É uma linguagem de programação genérica desenvolvida para ser tão eficiente e rápida quanto à linguagem Assembly e tão estruturada e lógica quanto as linguagens de alto nível (PASCAL, JAVA, etc.)." [PEREIRA, 2009].

A linguagem C foi usada no desenvolvimento de sistemas operacionais como o UNIX, LINUX e até mesmo o WINDOWS.

A Linguagem C é uma linguagem de alto nível, e como as outras linguagens desse nível é dividida em módulos ou estruturas independentes e com certas tarefas. Esses módulos e estruturas são chamados de funções na linguagem C, essas funções facilitam o desenvolvimento dos sistemas. [PEREIRA, 2009].

A programação dos microcontroladores hoje em dia é feita na maioria das vezes em C, pois a maioria deles contam com compiladores de linguagem C.

Quando programados na Linguagem C, os microcontroladores podem realizar muitas funções que para serem implementadas com a linguagem Assembly levariam muito mais

tempo. A programação em C possui uma portabilidade muito maior, tornando a adaptação dos programas para outras funcionalidades mais fácil e rápida.

Segundo Fábio Pereira, a linguagem C é a linguagem de alto nível mais eficiente atualmente disponível. Eficiência de uma linguagem para microcontroladores é a medida do grau de inteligência com que o compilador traduz um programa na linguagem programada para a linguagem de máquina. Quanto menor o tempo e mais rápido o código gerado, maior será a eficiência da linguagem e do compilador utilizados.

Para a realização desse projeto, a linguagem escolhida para a programação do microcontrolador foi a linguagem C, pois é a linguagem de alto nível mais eficiente, tornando-a mais simples para desenvolver a programação necessária e com uma maior capacidade de tradução para a linguagem de máquina.

CAPÍTULO 4 – SISTEMA DE AUTOMAÇÃO DE PERSIANAS

4.1 - Apresentação Geral

Este projeto baseia-se na intenção de trazer comodidade para uma residência, na forma de automatização das persianas. Neste é apresentando o protótipo de uma persiana automatizada utilizando microcontrolador..

O protótipo é composto de um sensor de luminosidade, um botão para girar a persiana no sentido horário e outro no anti-horário, um botão para ativar o modo manual ou automático, um display de LCD, um motor de passo, um circuito integrado L298N e um microcontrolador da família PIC.

O protótipo tem a função de girar a persiana de acordo com critérios especificados. Quando o botão manual/automático estiver no modo automático a persiana gira até um ângulo que vai depender da intensidade de luz solar e com isso gira até ficar em 90° permitindo a entrada máxima de luz quando não houver luz solar. Já quando o botão manual/automático estiver no modo manual a persiana gira de acordo com o comando do usuário utilizando os botões de girar no sentido horário e anti-horário. O display de LCD mostra se há a presença de luz ou não e se a persiana esta abrindo, aberta, fechando, fechada, girando no sentido horário ou girando no sentido anti-horário. A mensagem que aparece no display é enviada pelo microcontrolador.

As Figuras 4.1 e 4.2 ilustram o fluxograma referente ao projeto, onde é possível ver os passos que o microcontrolador segue de acordo com as entradas.

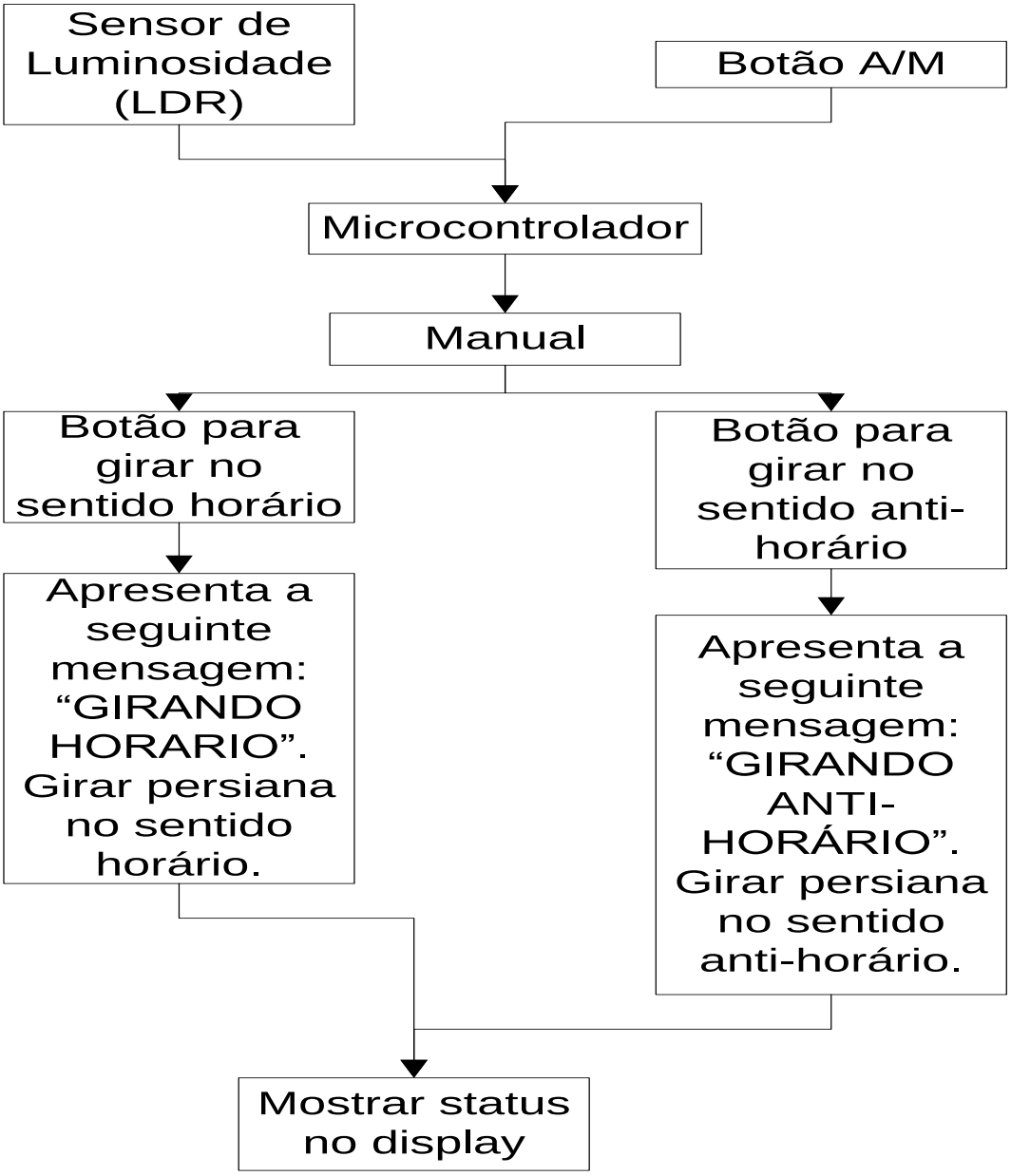


Figura 4.1 – Fluxograma (Modo Manual)

[AUTOR]

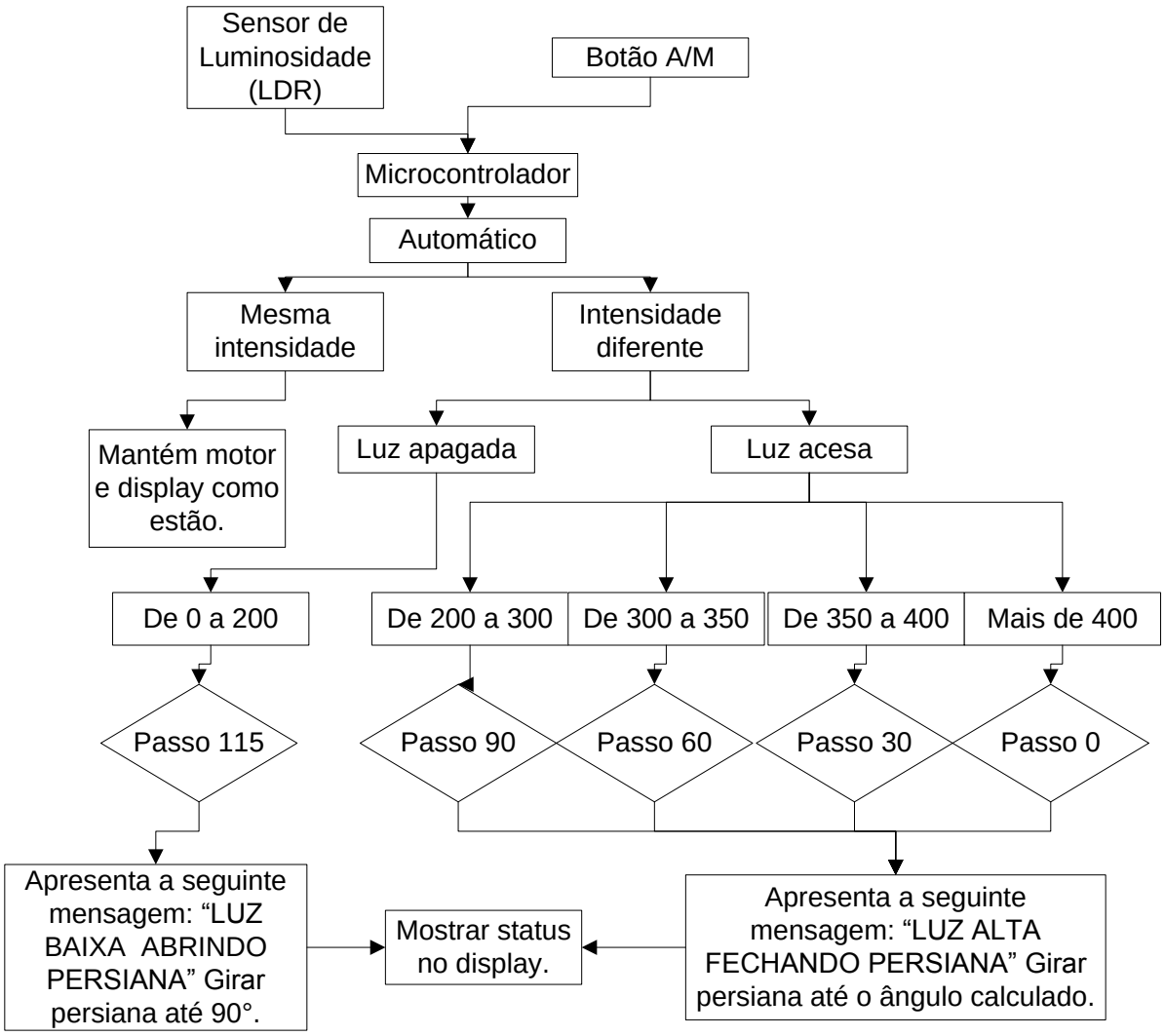


Figura 4.2 – Fluxograma (Modo Automático)

[AUTOR]

4.2 - Apresentação Final do Circuito

Para testar o circuito, foi utilizado o software Proteus 7.2. Esta ferramenta robusta e confiável permite simular vários tipos de circuitos eletrônicos através do computador. Com isso, economiza-se tempo e dinheiro, já que com os circuitos testados no software não há a possibilidade de queimar componentes e é construído com mais praticidade. Com este software é possível que sejam simulados os microcontroladores, ou seja, após a compilação do algoritmo é possível implementar o arquivo ".HEX" no microcontrolador e testar as funcionalidades do circuito por completo.

A Figura 4.3, ilustra as ligações feitas entre os elementos do projeto, utilizando o Proteus7.2.

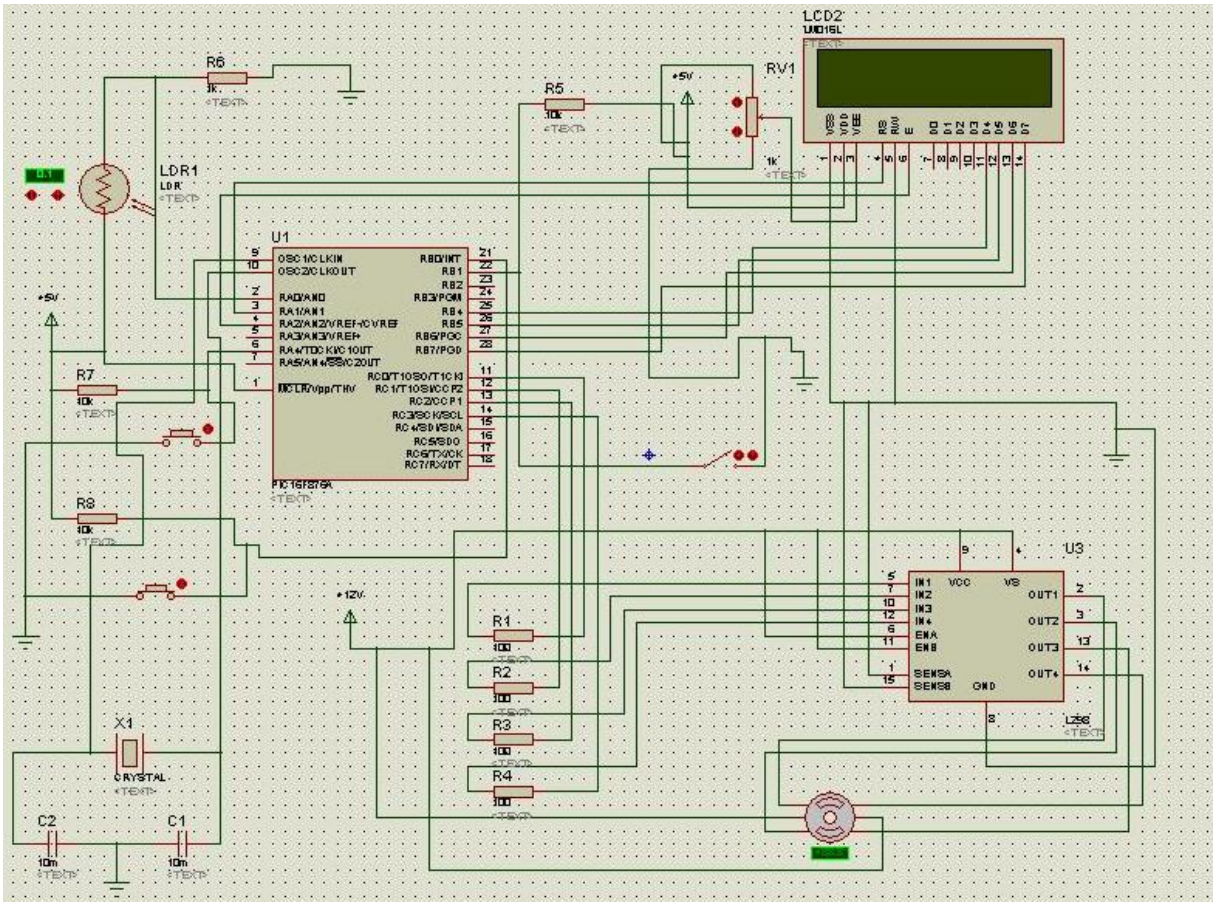


Figura 4.3 – Ligações do projeto

[AUTOR]

Serão apresentados a seguir os elementos do circuito simulado no Proteus. O LDR na figura 4.4, o Microcontrolador na figura 4.5, o botão (sentidos horário e anti-horário) na figura 4.6, o cristal resposnsável pelo clock externo na figura 4.7, o switch (Automático/Manual) na figura 4.8, o CI L298N na figura 4.9, o motor de passo na figura 4.10 e o display de LCD na figura 4.11.

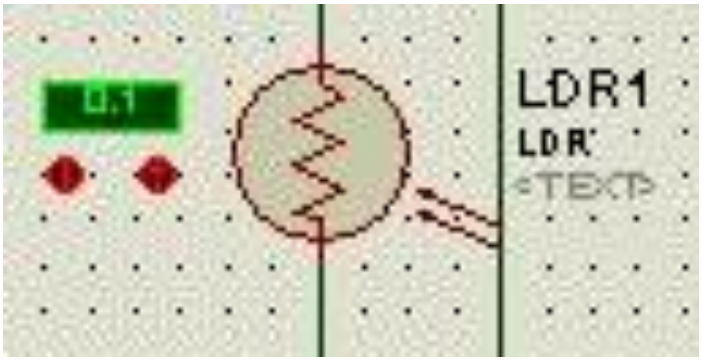


Figura 4.4 – LDR (Proteus)

[AUTOR]

Figura 4.5 – Microcontrolador (Proteus)

[AUTOR]

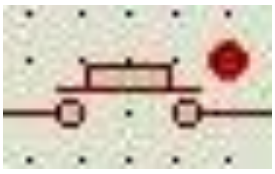


Figura 4.6 – Botão (Proteus)

[AUTOR]

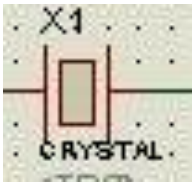


Figura 4.7 – Cristal (Proteus)

[AUTOR]

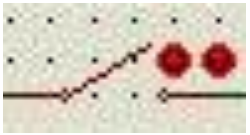


Figura 4.8 – Switch (Proteus)

[AUTOR]

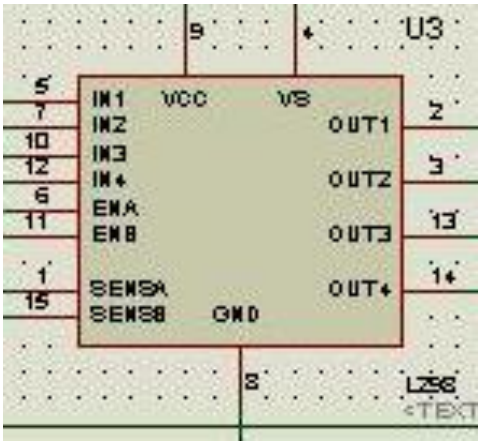


Figura 4.9 – CI L298N (Proteus)

[AUTOR]

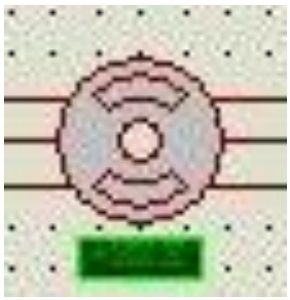


Figura 4.10 – Motor de Passo (Proteus)

[AUTOR]

Figura 4.11 – Display (Proteus)

[AUTOR]

O microcontrolador está ligado aos elementos do circuito, direta ou indiretamente, seguem as descrições das ligações entre o microcontrolador e os componentes que recebem ou enviam sinais ao microcontrolador.

- Pino 1 – Fonte de alimentação 5 volts.
- Pino 2 (RA0/AN0) – Sensor de Luminosidade (LDR).
- Pino 6 (RA4) – Botão para girar a persiana.
- Pino 21 (RB0) – Botão para girar a persiana.
- Pino 22 (RB1) – Chave para controle do modo manual e automático.

As ligações feitas entre o display de cristal líquido e o microcontrolador são as seguintes.

- Pino 1 – Terra.
- Pino 2 – Fonte de alimentação 5 volts.
- Pino 3 – Potenciômetro em série com o terra.
- Pino 4 – Pino 3 (RA1/AN1) do microcontrolador.
- Pino 5 – Terra.

- Pino 6 – Pino 4 (RA2/AN2) do microcontrolador.
- Pino 7 – Terra.
- Pino 8 – Terra.
- Pino 9 – Terra.
- Pino 10 – Terra.
- Pino 11 – Pino 25 (RB4).
- Pino 12 – Pino 26 (RB5).
- Pino 13 – Pino 27 (RB6).
- Pino 14 – Pino 28 (RB7).

Conforme já foi detalhado anteriormente, o circuito integrado L298N, faz a ligação entre o microcontrolador e o motor de passo. Segue a forma em que os pinos do L298N são ligadas ao motor de passo e ao microcontrolador:

- Pino 1 – Terra.
- Pino 2 – Fio azul do motor de passo.
- Pino 3 – Fio branco do motor de passo.
- Pino 4 – Fonte de alimentação de 12 volts.
- Pino 5 – Pino 11 (RC0) do microcontrolador.
- Pino 6 - Fonte de alimentação de 12 volts.
- Pino 7 – Pino 12 (RC1) do microcontrolador.
- Pino 8 – Terra.
- Pino 9 – Fonte de alimentação de 12 volts.

- Pino 10 – Pino 13 (RC2) do microcontrolador.
- Pino 11 – Fonte de alimentação de 12 volts.
- Pino 12 – Pino 14 (RC3) do microcontrolador.
- Pino 13 – Fio vermelho do motor de passo.
- Pino 14 – Fio amarelo do motor de passo.
- Pino 15 – Terra.

Depois de feitas as ligações e feita a gravação do arquivo ".hex" no microcontrolador, o circuito está pronto. Algumas figuras com o funcionamento da simulação no proteus serão explicadas a seguir.

4.3 - Controle do Motor de Passo

O motor de passo utilizado nesse projeto é um motor de passo de Ímã Permanente e Unipolar, tendo assim um ganho de torque comparado ao motor de Relutância Variável, do modelo SM7.5-A12P-NP.

Os motores de passo trabalham através de sinais digitais, que indicam quando uma das quatro bobinas está recebendo energia ou não. A ordem em que as bobinas são energizadas indica se o motor esta em movimento e como ele esta se movendo.

Na Tabela 1 é indicada a ordem que as bobinas são ligadas e desligadas para que o motor possa realizar um passo.

Tabela 1 – Passo do motor

Ordem da programação das bobinas.	B1	B2	B3	B4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Segue abaixo parte da programação utilizada no sistema para girar o motor de passo nos dois sentidos.

```
  delay_ms(50);
output_high(MOT1);
output_low (MOT2);
output_low(MOT3);
output_low(MOT4);
delay_ms(50);
output_low(MOT1);
output_high(MOT2);
output_low (MOT3);
output_low(MOT4);
delay_ms(50);
output_low(MOT1);
output_low(MOT2);
output_high(MOT3);
output_low (MOT4);
delay_ms(50);
output_low (MOT1);
output_low(MOT2);
output_low(MOT3);
output_high(MOT4);
delay_ms(50);
output_low(MOT1);
```

```
output_low(MOT2);
output_low(MOT3);
output_low(MOT4);
  passo++;
```

Para que o motor gire no sentido contrario, a função é essa.

```
delay_ms(50);
output_low(MOT1);
output_low(MOT2);
output_low(MOT3);
output_high(MOT4);
delay_ms(50);
output_low(MOT1);
output_low(MOT2);
output_high(MOT3);
output_low(MOT4);
delay_ms(50);
output_low(MOT1);
output_high(MOT2);
output_low(MOT3);
output_low(MOT4);
delay_ms(50);
output_high(MOT1);
output_low(MOT2);
output_low(MOT3);
```

```
output_low(MOT4);
  delay_ms(50);
output_low(MOT1);
output_low(MOT2);
output_low(MOT3);
output_low(MOT4);
  passo--;
```

Neste programa, “MOT1, MOT2, MOT3 e MOT4” representam as saídas do microcontrolador que enviam sinais digitais para o motor de passo. Cada uma dessas funções tem como finalidade dar um passo no motor, um para girar a persiana no sentido horário e outro no sentido anti-horário.

O microcontrolador é capaz de enviar o sinal na ordem em que o motor necessita receber para realizar o movimento, porém, a tensão não é capaz de fazer com que o motor realize o movimento. No sistema é utilizado o CI L298N com a finalidade de aumentar a tensão recebida do microcontrolador para que seja capaz de realizar o movimento do motor de passo.

4.4 - Controle do Sensor de Luminosidade

O sensor de Luminosidade tem um de seus fios conectado, em paralelo com um resistor aterrado, no pino RA0/AN0 do microcontrolador, que é configurado para receber sinais analógicos, e o outro no polo positivo da fonte de alimentação. Sua função é alternar a tensão recebida pelo microcontrolador quando aumentar ou diminuir a luminosidade recebida, para que o microcontrolador possa realizar as funções que foram programadas para a situação.

A representação bot1 é referente ao botão que alterna manual/automático, enquanto estiver no modo automático, a persiana irá abrir quando estiver escuro e realizar uma angulação quando houver a incidência de luminosidade. A parte do programa responsável por chamar essas funções é:

```
if(!input(bot1)){       // Automático
    if((passovalor==115)&&(passo!=passovalor))
      tempo_escuro_abre_persiana();
                else if((passovalor!=passo)&&(passovalor!=115))
      tempo_claro_fecha_persiana();
    else
      status_lcd();      //Escrever status no LCD
  }
```

Onde “passo” é o passo atual do motor, “passovalor” é o passo calculado pelo microcontrolador, já “tempo_escuro_abre_persiana ()” e “tempo_claro_fecha_persiana ()” são funções para manter a persiana no ângulo desejado.

Para que seja implementado em um ambiente real, deve-se coloca-lo virado para cima e com uma proteção transparente contra a chuva.

4.5 - Controle do Display de LCD

O controle do display de LCD foi realizado com a utilização de uma função (mod_lcd.c), retirada de um exercício do livro do Fabio Pereira. [PEREIRA, 2009]

O programa mod_lcd.c se encontra no anexo desta monografia.

4.6 - Controle do Microcontrolador

4.5.1 - Programação de Controle

A programação do microcontrolador é realizada na linguagem C.

Para facilitar a utilização dos pinos do microcontrolador são nomeados os pinos de acordo com a sua utilidade. Segue trecho do código onde são feitas essas definições nos pinos do PIC.

```
#include <16F876a.h>
#define MOT1 PIN_C1          // Declara porta para controlar 1º passo do motor
#define MOT2 PIN_C2          // Declara porta para controlar 2º passo do motor
```

```
#define MOT3 PIN_C3          // Declara porta para controlar 3º passo do motor
#define MOT4 PIN_C4          // Declara porta para controlar 4º passo do motor
#define bot1 PIN_B1          // Manual / Automático
#define bot2 PIN_A3          // Botão GIRA1 persiana
#define bot3 PIN_A5          // Botão GIRA2 persiana
```

Este Programa possui várias funções, que na linguagem C são partes do código que são divididas para que não sobrecarregue a parte principal e podem ser chamadas a qualquer momento no programa, as funções utilizadas no programa são:

```
void BOT_GIRA_1(void);                  // Função - Abrir persiana claro
void tempo_escuro_abre_persiana (void); // Função - Abrir persiana escuro
void BOT_GIRA_2(void);                  // Função - Fecha Persiana escuro
void tempo_claro_fecha_persiana (void); // Função - Fecha Persiana claro
void status_lcd (void);                 // Função - Escrever status no LCD
void transpassovalor (void);            // Função - transforma valor em passo
long AD (int CANAL);                    // Função - Ler entrada analógica
```

4.5.2 - Inserção do Algoritmo no Microcontrolador

Para copiar o programa na linguagem C para o microcontrolador, para isso *é necessária a compilação através de um* compilador como o PIC C *Compiler aqui usado*. O compilador gera um arquivo ".HEX" que é o tipo de arquivo interpretado pelo microcontrolador PIC.

Na figura 4.12 pode-se observar o programa utilizado para compilar o algoritmo.

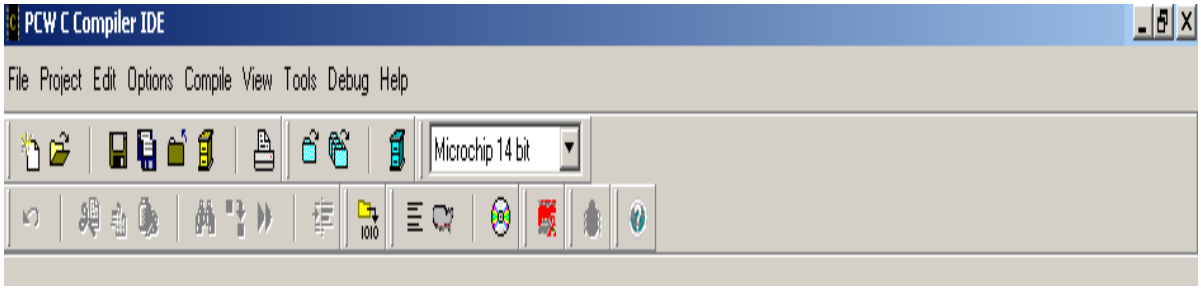


Figura 4.12 – PIC C Compiler

[AUTOR]

Este programa, após receber o programa desenvolvido em C, compila o algoritmo e gera vários tipos de arquivos, conforme apresentado na Figura 4.13 a seguir, porém como dito anteriormente o microcontrolador trabalha com o arquivo de extensão ".HEX", esta extensão é chamada de hexadecimal, pois utiliza números hexadecimais.

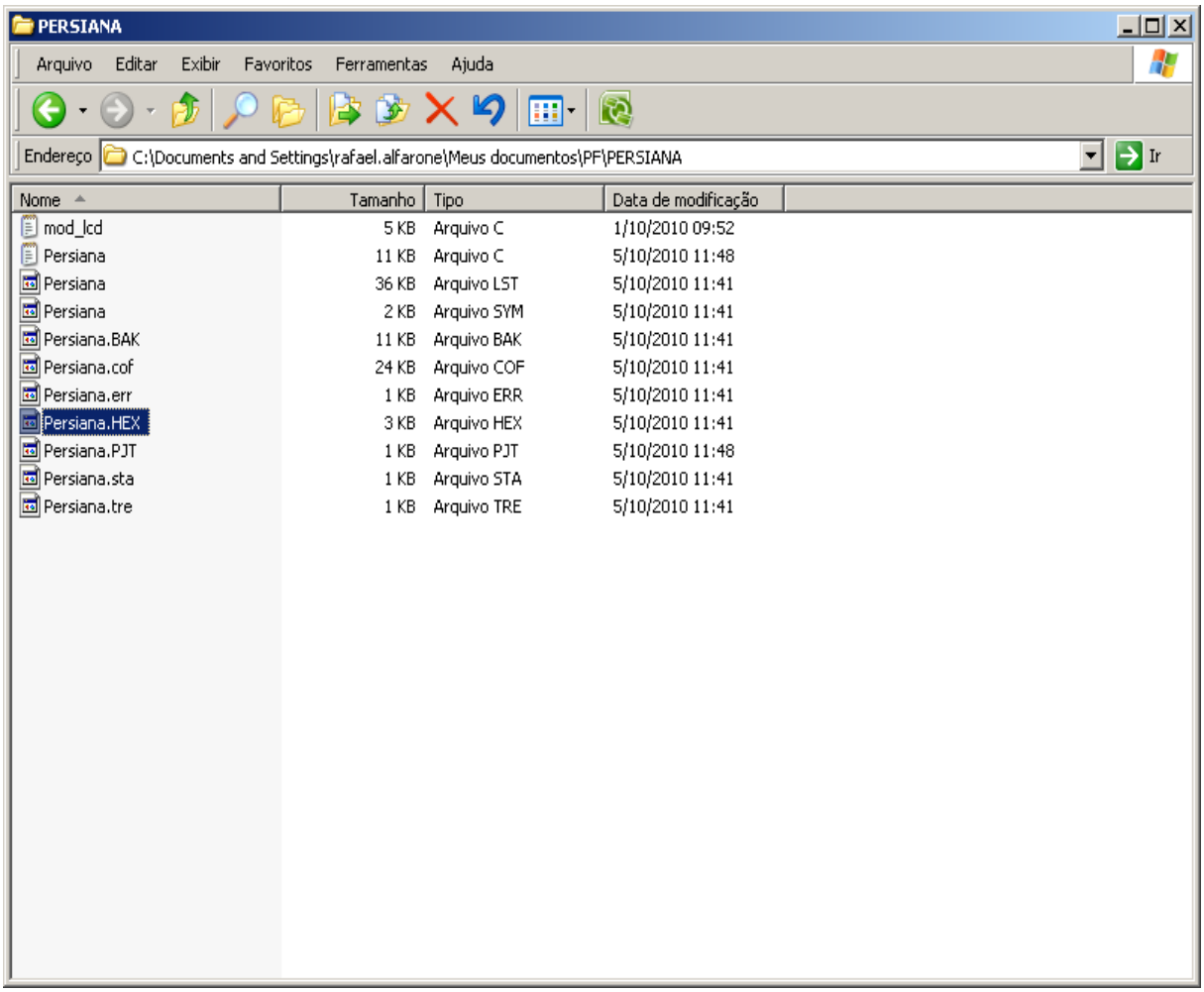


Figura 4.13 – Extensões geradas pelo PIC C Compiler

[AUTOR]

Após a gravação do arquivo hexadecimal no microcontrolador através da gravadora integrada no kit, ele esta pronto para ser utilizado.

4.7 - Apresentação do Protótipo

O protótipo começou a ser implementado em um kit com microcontrolador, gravadora e Display de LCD a serem utilizados no projeto. A Figura 4.14 ilustra esse kit.

Figura 4.14 – Foto do Kit

[ACEPIC]

O protótipo foi feito inicialmente sem o sensor e sem os botões, somente com o motor e o Kit, ligados a um protoboard que possui o CI L298N e o circuito necessário para que o motor realizasse a sua função. Como apresentado na figura 4.15.

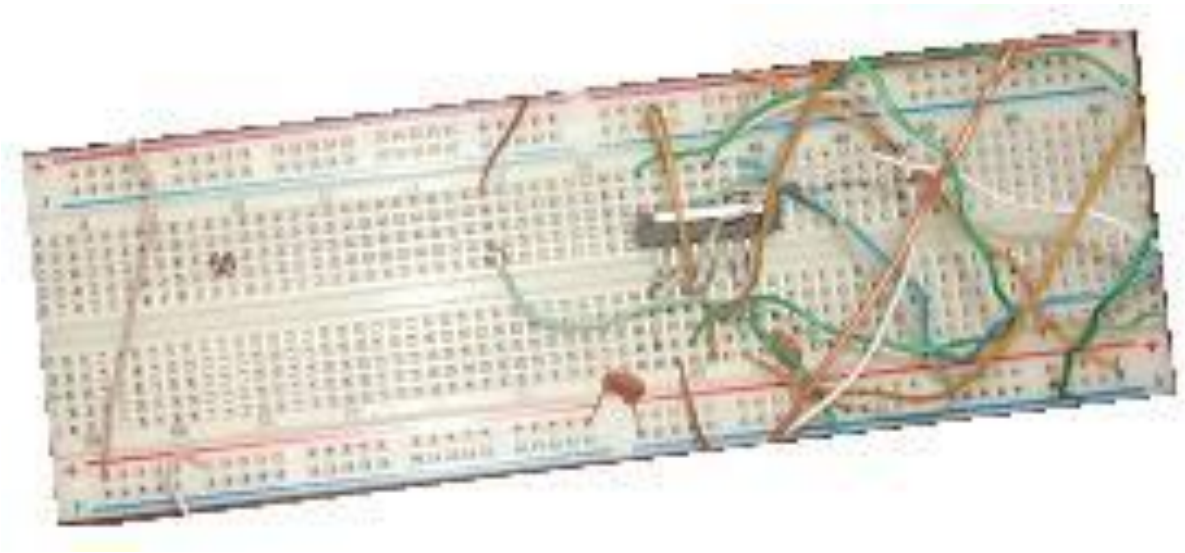


Figura 4.15 – Foto do Protótipo (1)

[AUTOR]

Com o circuito funcionando já na placa de circuito impresso, a maquete começou a ser preparada, para isso foi necessária a utilização de uma engrenagem (ver na Figura 4.16), para o motor de passo e uma persiana com o tamanho 1,50 m x 0,40 m.

Pode-se observar na figura 4.17 uma visão geral da maquete.

Figura 4.16 – Foto da Engrenagem Utilizada

[AUTOR]

Figura 4.17 – Foto do Protótipo (2)

[AUTOR]

A ligação física entre o motor e a persiana é feita conforme pode-se verificar na figura 4.18.

Figura 4.18 – ligação física entre o motor e a persiana

[AUTOR]

CAPÍTULO 5 – RESULTADOS OBTIDOS

5.1 - Apresentação da área de Aplicação do modelo

Este projeto tem uma vasta área em que pode ser utilizado, como empresas que necessitem da proteção contra o sol no seu interior, e é bem possível que em pouco tempo várias residências estejam automatizadas, posto que já existe uma tendência de automatizar as residências em construção.

5.2 - Descrição da Aplicação do Modelo

O projeto começou pelo desenvolvimento de um programa e pela simulação do circuito desenvolvido, para a realização desses passos foram utilizados os programas PIC C Compiler e Proteus 7.2. Após a simulação do circuito, o programa e o circuito já estavam prontos para ser utilizados no protótipo.

O protótipo foi desenvolvido começando pela gravação do programa no microcontrolador. Com o microcontrolador pronto foi feito o circuito onde o motor, o sensor e os botões atuam, a partir daí começaram os testes e a calibração do motor, que, para isso, alterou-se a programação do microcontrolador, após esses passos o protótipo estava pronto.

Para definir-se o número de passos necessários ao motor de passo para que este chegasse ao giro médio e ao giro máximo, foram feitos giros manuais na engrenagem até que a persiana chegasse ao ponto de maior entrada de luminosidade, que foi adotado 90°, e ao ponto máximo possível. Esse número de voltas foi multiplicado pelo número de passos por volta.

5.3 – Testes

O sensor foi testado com valores escolhidos pelo autor e alterados até que atendessem ao propósito do projeto, sendo divido em 5 intervalos diferentes baseados na tensão recebida pelo sensor.

Na função a seguir são apresentadas as variações adotadas pelo autor e o número de passos adotados para cada intensidade de luz, onde quanto maior o valor enviado pelo LDR

para o microcontrolador, maior é a intensidade e 115 é o número de passos do motor para que a persiana chegue ao ângulo médio.

```
if (valor>=0&&valor<200)
        passovalor=115;
if (valor>=200&&valor<300)
        passovalor=90;
if (valor>=300&&valor<350)
        passovalor=60;
if (valor>=350&&valor<400)
        passovalor=30;
if (valor>=400)
        passovalor=0;
```

Para que fosse montada a placa de circuito impresso, foram realizados testes no Proteus, para que não fossem necessárias trocas de componentes e de placas posteriormente. Os testes no Proteus são ilustrados a seguir.

A figura 5.1, ilustra a situação em que o sistema está funcionando em modo automático, com o sensor de luminosidade (LDR) estando com um valor baixo de lux, ou seja, não esta recebendo luz o bastante para que a persiana diminua o seu ângulo diminuindo a incidência de luz ou altere o seu ângulo para que admita a entrada máxima de luz. Nesta situação o display mostra na primeira linha “LUZ BAIXA”, que significa que a luminosidade esta baixa e na segunda linha do display está à informação “ABRINDO PERSIANA”, esta condição ocorre enquanto a persiana não atingir o ângulo que permite a incidência máxima de luminosidade.

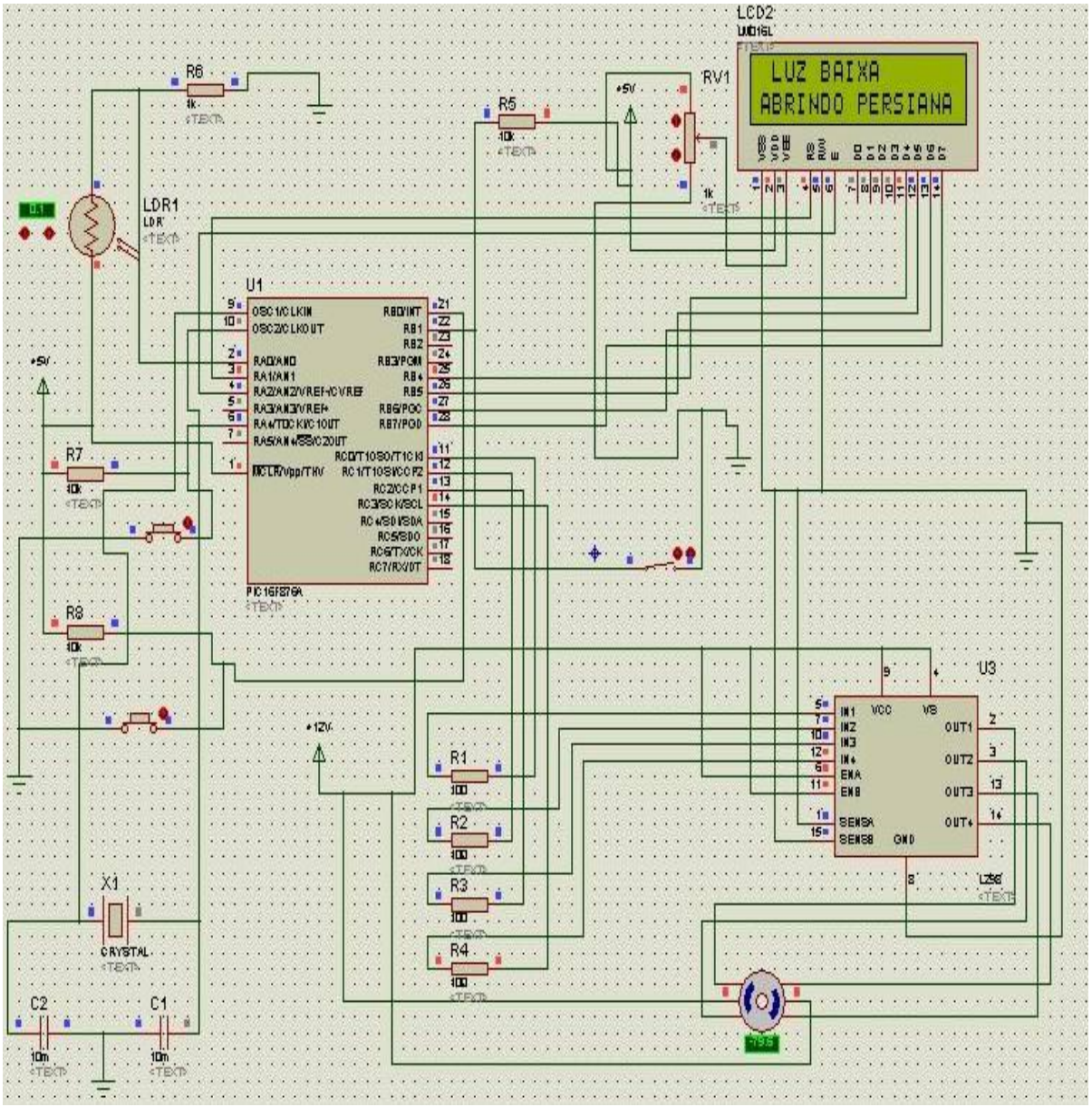


Figura 5.1 – LUZ BAIXA – ABRINDO PERSIANA (AUTO)

[AUTOR]

A figura 5.2, ilustra a situação que vem após a situação descrita na figura 5.1, neste caso a persiana atinge o ângulo que permite a incidência máxima de luz, logo a persiana encontra-se aberta, assim sendo o motor de passo para de girar a persiana e o sistema fica na espera de outra operação. Nesta situação o display mostra na primeira linha “LUZ BAIXA” e na segunda linha “PERSIANA ABERTA”.

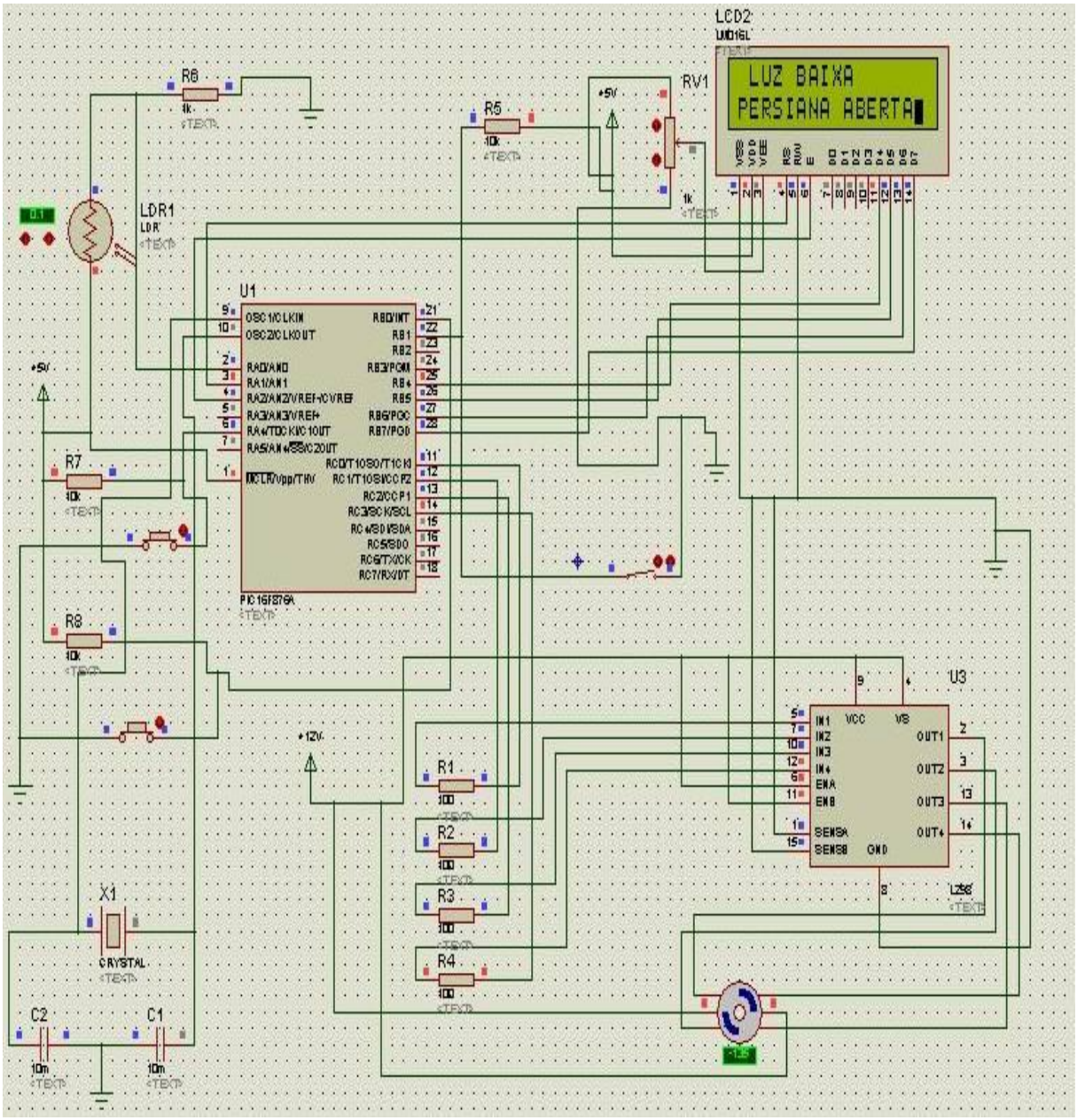


Figura 5.2 – LUZ BAIXA – PERSIANA ABERTA (AUTO)

[AUTOR]

A figura 5.3, ilustra a situação contraria à mostrada na figura 5.1, neste caso o sistema continua operando em modo automático, porém o sensor de luminosidade (LDR) foi ativado e o ângulo para essa quantidade de lux recebido é calculado, com isso é enviado um sinal com o número de passos que o motor deve girar, girando a persiana até o ângulo calculado. Nesta situação o display mostra na primeira linha “LUZ ALTA”, que significa que a luminosidade esta alta e na segunda linha do display está à informação “FECHANDOPERSIANA”, esta condição ocorre enquanto a persiana não atingir o ângulo calculado pelo programa.

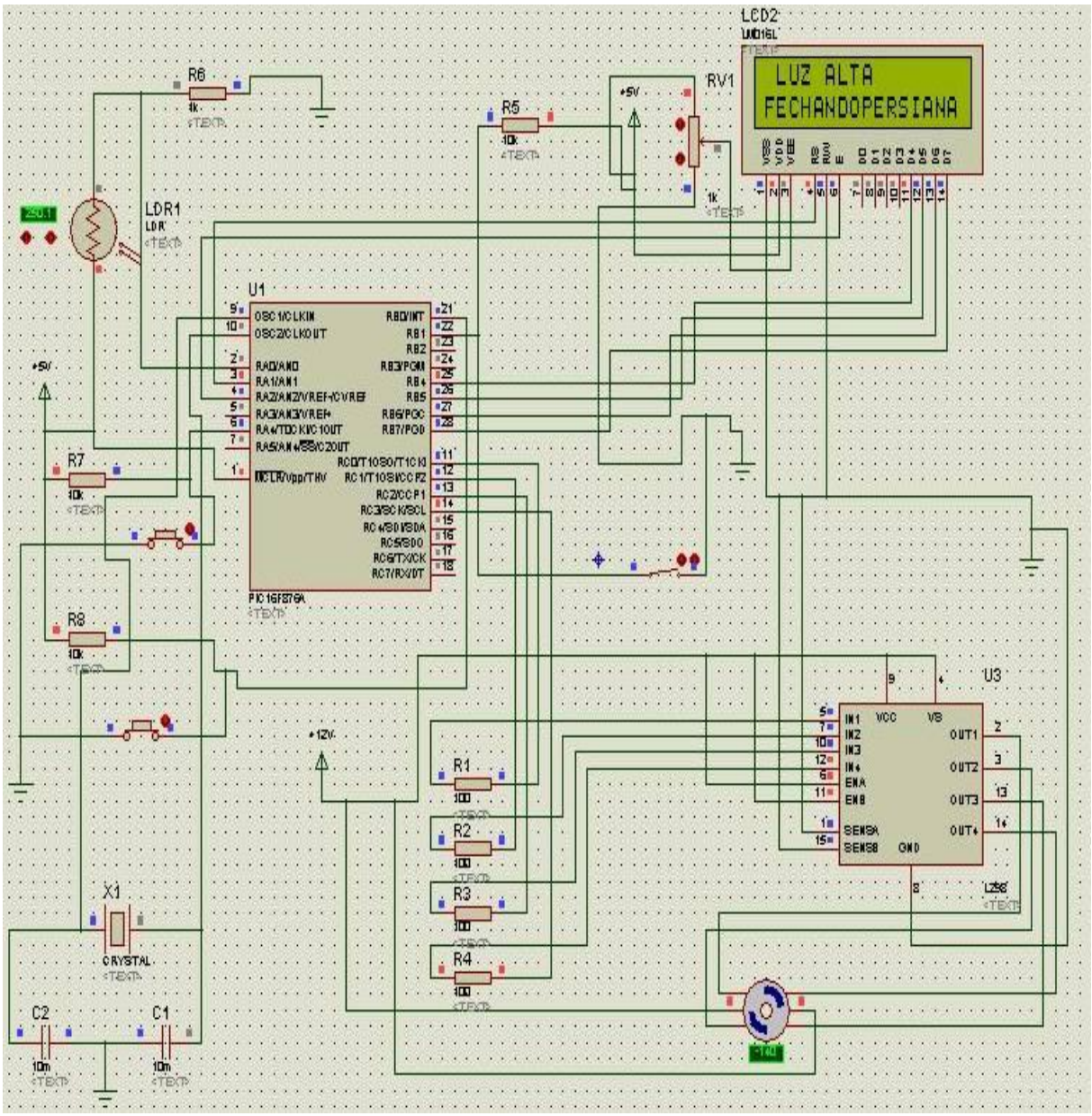


Figura 5.3 – LUZ ALTA – FECHANDOPERSIANA (AUTO)

[AUTOR]

A figura 5.4, ilustra a situação que vem após a situação descrita na figura 5.3, neste caso a persiana atinge o ângulo calculado, logo a persiana encontra-se fechada, assim sendo o motor de passo para de girar a persiana e o sistema fica na espera de outra operação. Nesta situação o display mostra na primeira linha “LUZ ALTA” e na segunda linha “PERSIANA FECHADA”.

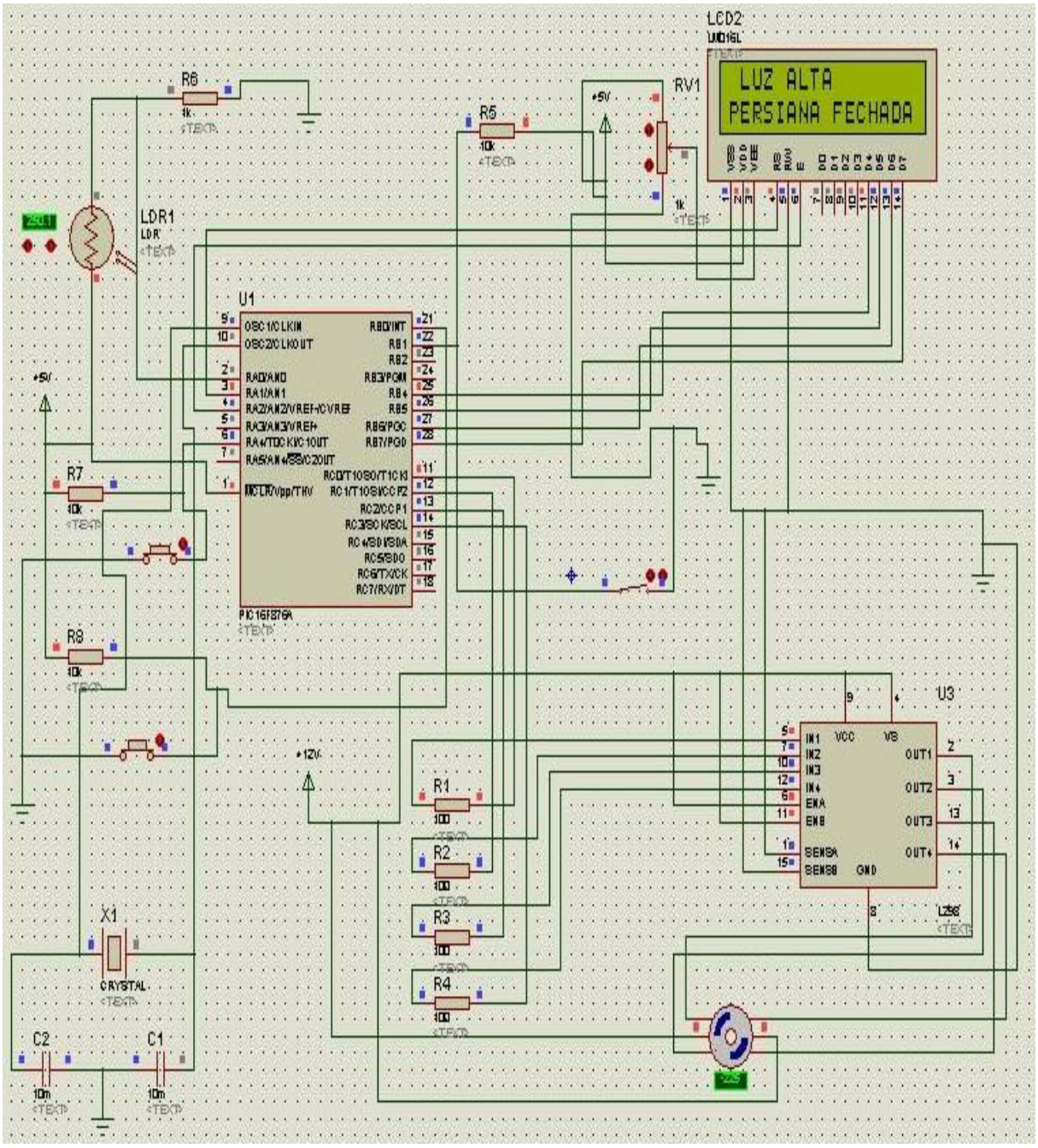


Figura 5.4 – LUZ ALTA –PERSIANA FECHADA (AUTO)

[AUTOR]

É importante salientar que enquanto o sistema estiver em modo automático, os botões de abrir e fechar a persiana não tem funcionamento.

As próximas descrições de situações são feitas quando o sistema esta operando no modo manual. Neste modo o usuário tem um controle sobre a persiana, podendo abrir e fechar a persiana utilizando dois botões, um para girar no sentido horário e outro para girar no sentido anti-horário.

A figura 5.5, ilustra uma situação em que o sistema está funcionando em modo manual, com o sensor de luminosidade (LDR) estando com um valor baixo de lux e com a persiana aberta. Nesta situação o display mostra na primeira linha “LUZ BAIXA”, que significa que a luminosidade esta baixa e na segunda linha do display está à informação “PERSIANA ABERTA”, ou seja, a persiana está com o ângulo que admite a entrada máxima de luz.

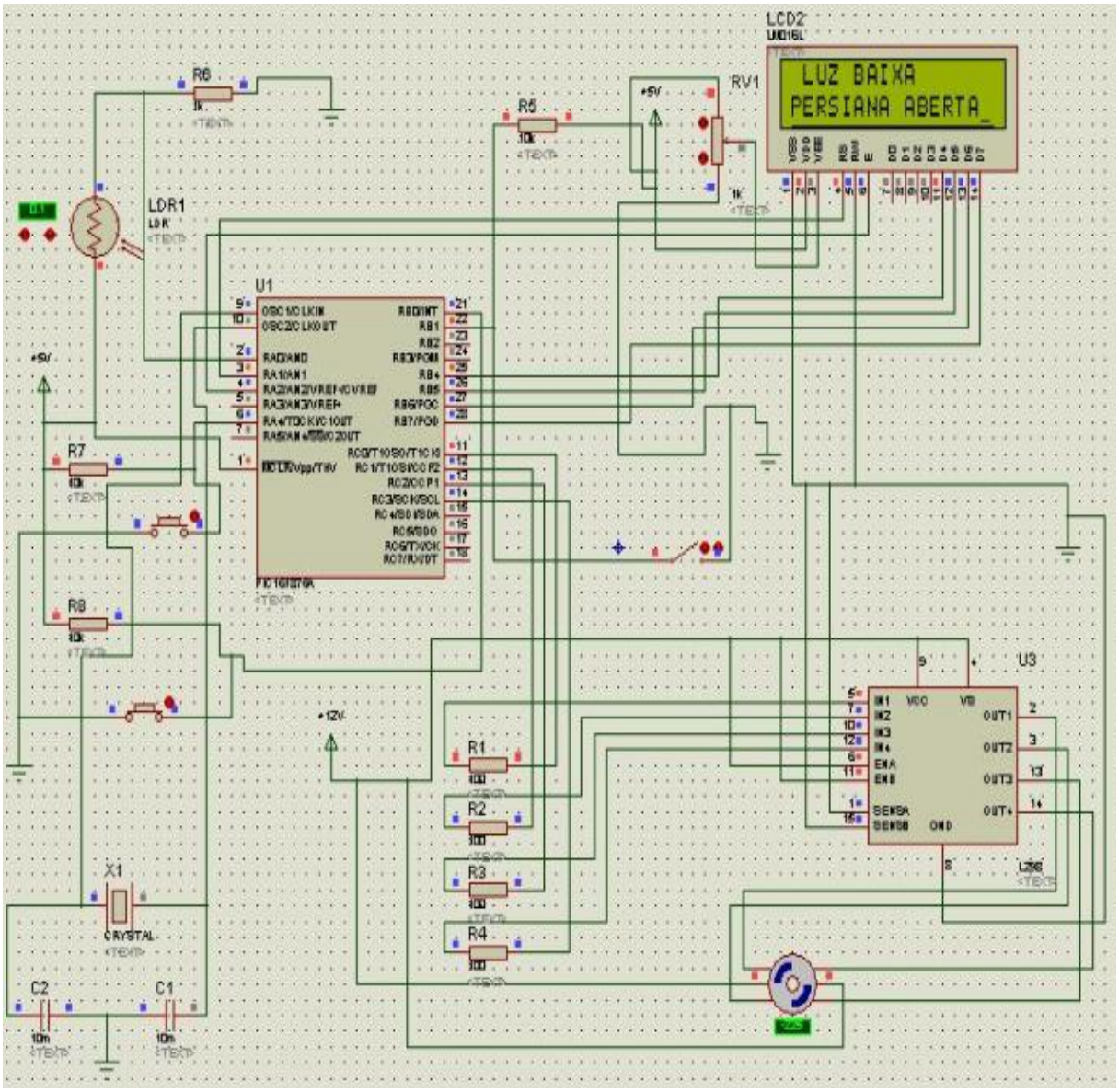


Figura 5.5 – LUZ BAIXA – PERSIANA ABERTA (MANUAL)

[AUTOR]

A figura 5.6, ilustra uma situação em que o sistema está funcionando em modo manual, com o sensor de luminosidade (LDR) estando com um valor alto de lux e com a persiana aberta. Nesta situação o display mostra na primeira linha “LUZ ALTA”, que

significa que a luminosidade esta alta e na segunda linha do display está à informação “PERSIANA ABERTA”, ou seja, a persiana está com o ângulo que admite a entrada máxima de luz.

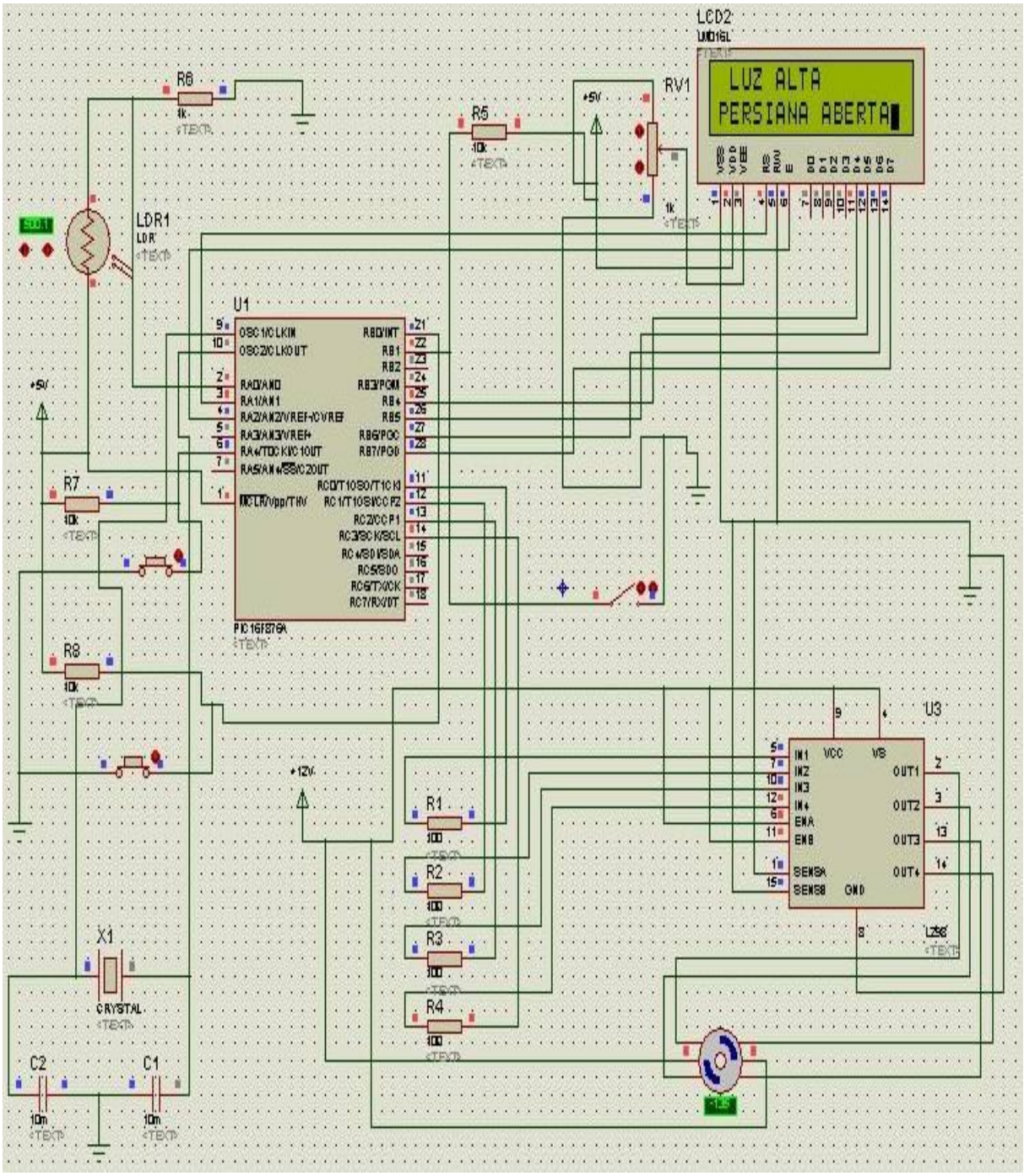


Figura 5.6 – LUZ ALTA –PERSIANA ABERTA (MANUAL)

[AUTOR]

A figura 5.7, ilustra uma situação em que o sistema está funcionando em modo manual, com o sensor de luminosidade (LDR) estando com um valor baixo de lux e com a

persiana fechada. Nesta situação o display mostra na primeira linha “LUZ BAIXA”, que significa que a luminosidade esta baixa e na segunda linha do display está à informação “PERSIANA FECHADA”, ou seja, a persiana está com um ângulo diferente do que admite a entrada máxima de luz.

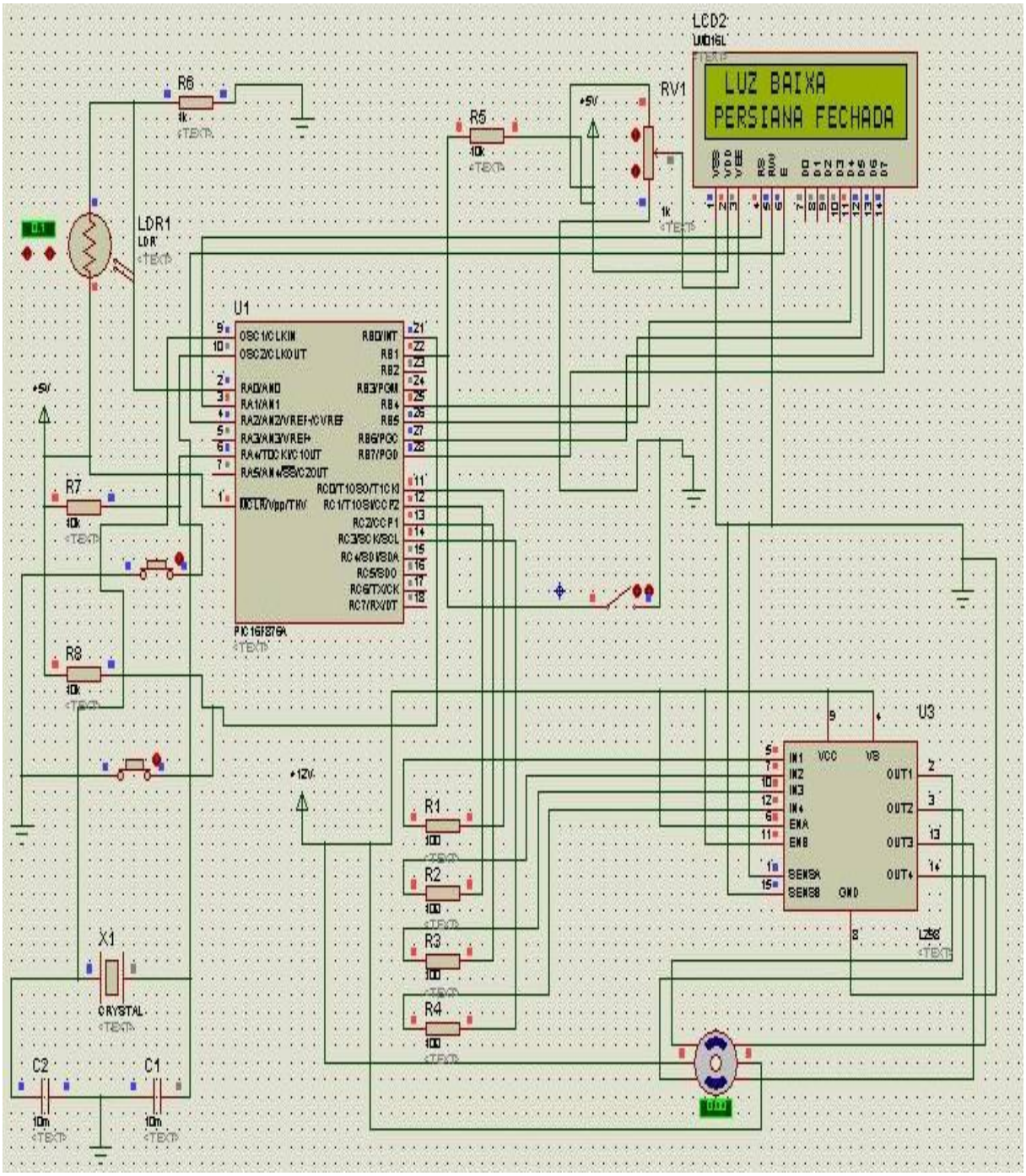


Figura 5.7 – LUZ BAIXA –PERSIANA FECHADA (MANUAL)

[AUTOR]

A figura 5.8, ilustra uma situação em que o sistema está funcionando em modo manual, com o sensor de luminosidade (LDR) estando com um valor alto de lux e com a persiana fechada. Nesta situação o display mostra na primeira linha “LUZ ALTA”, que significa que a luminosidade esta baixa e na segunda linha do display está à informação “PERSIANA FECHADA”, ou seja, a persiana está com um ângulo diferente do que admite a entrada máxima de luz.

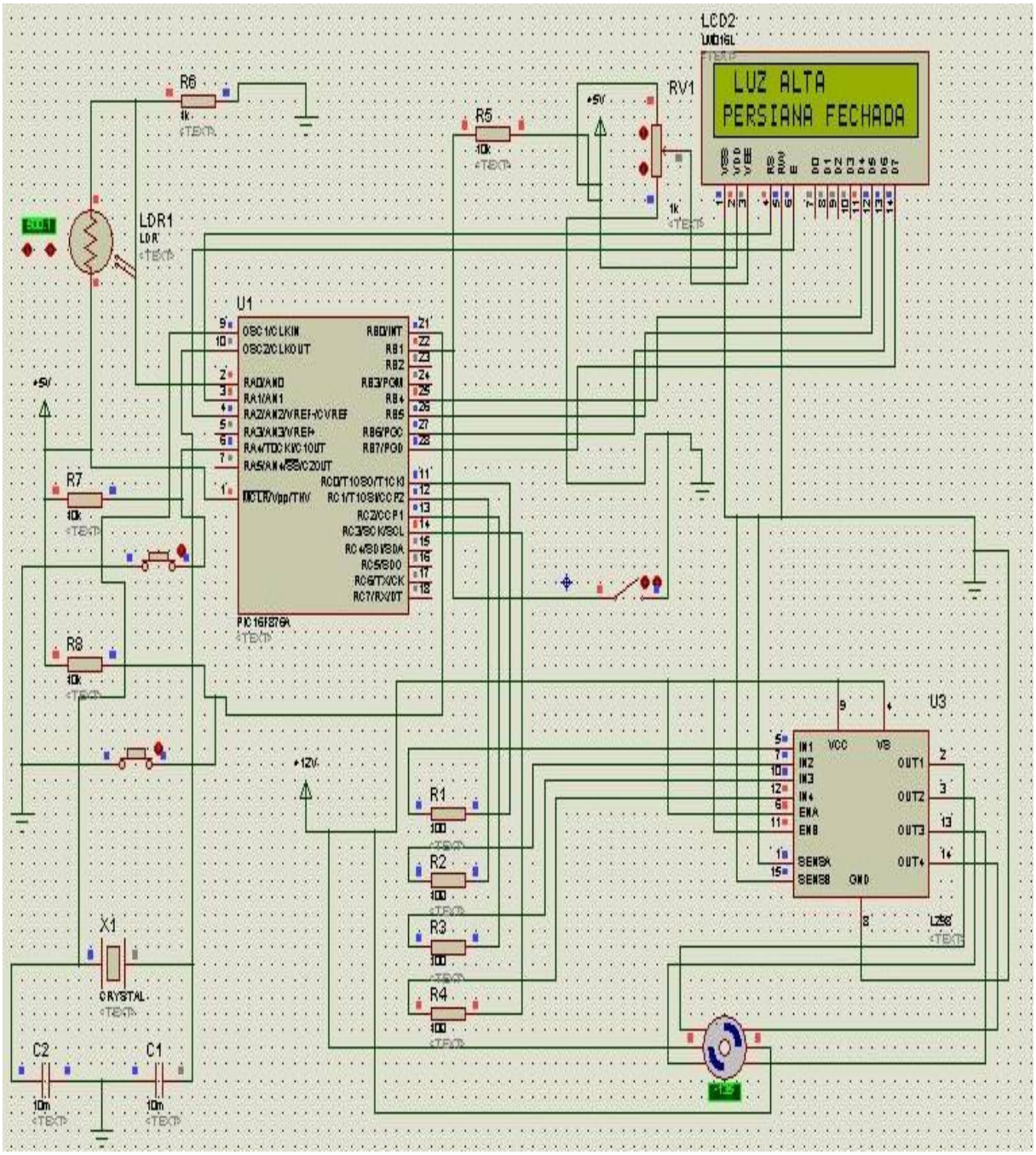


Figura 5.8 – LUZ ALTA –PERSIANA FECHADA (MANUAL)

[AUTOR]

A figura 5.9, ilustra uma situação em que o sistema está funcionando em modo manual e com o botão GIRA 1 funcionando. Nesta situação o display mostra na primeira linha “GIRANDO”, e na segunda linha “HORÁRIO”, ou seja, o motor de passo gira no sentido horário.

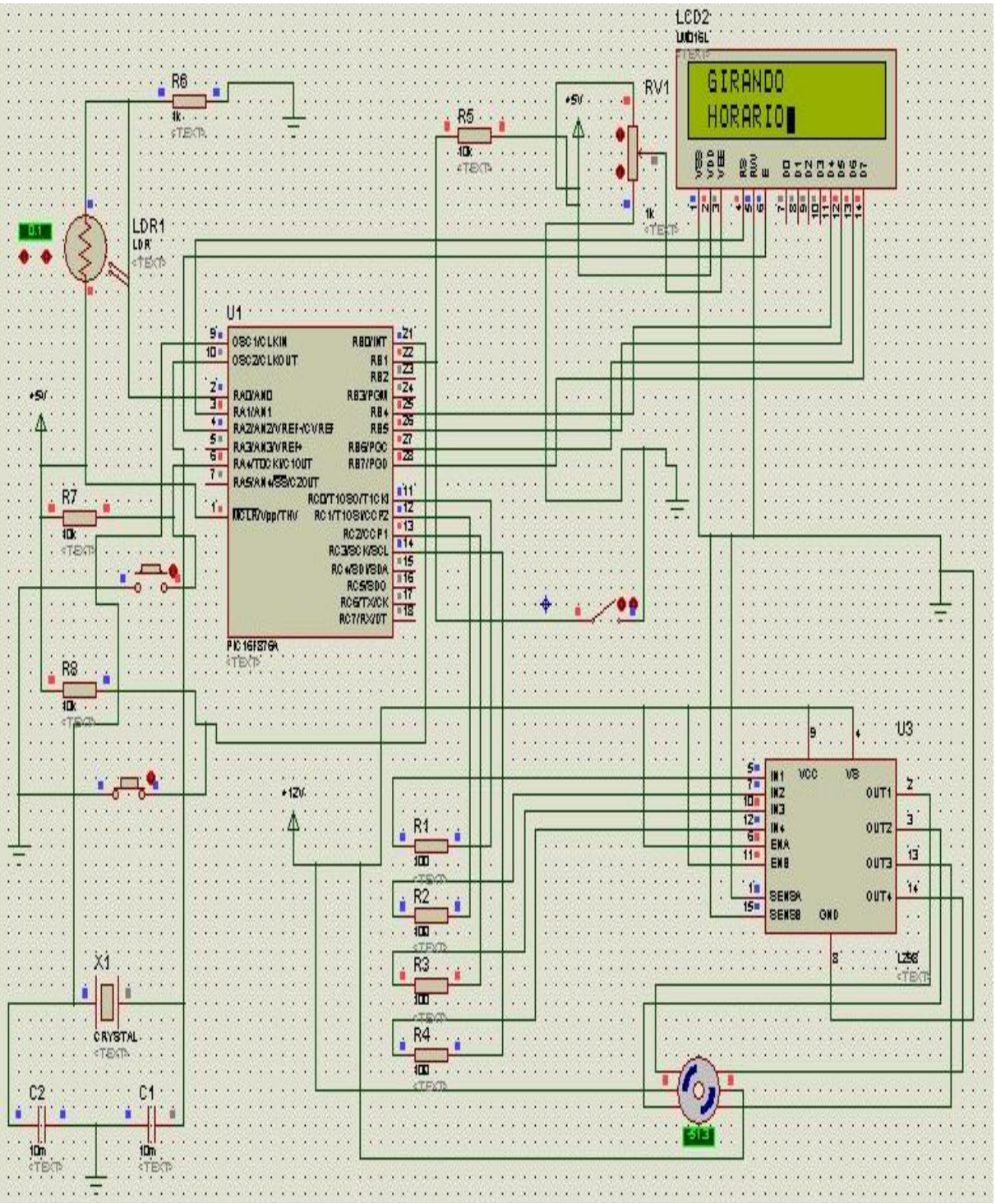


Figura 5.9 – GIRANDO HORÁRIO (MANUAL)

[AUTOR]

A figura 5.10, ilustra uma situação em que o sistema está funcionando em modo manual e com o botão GIRA 2 funcionando. Nesta situação o display mostra na primeira linha “GIRANDO”, e na segunda linha “ANTI-HORÁRIO”, ou seja, o motor de passo gira no sentido anti-horário.

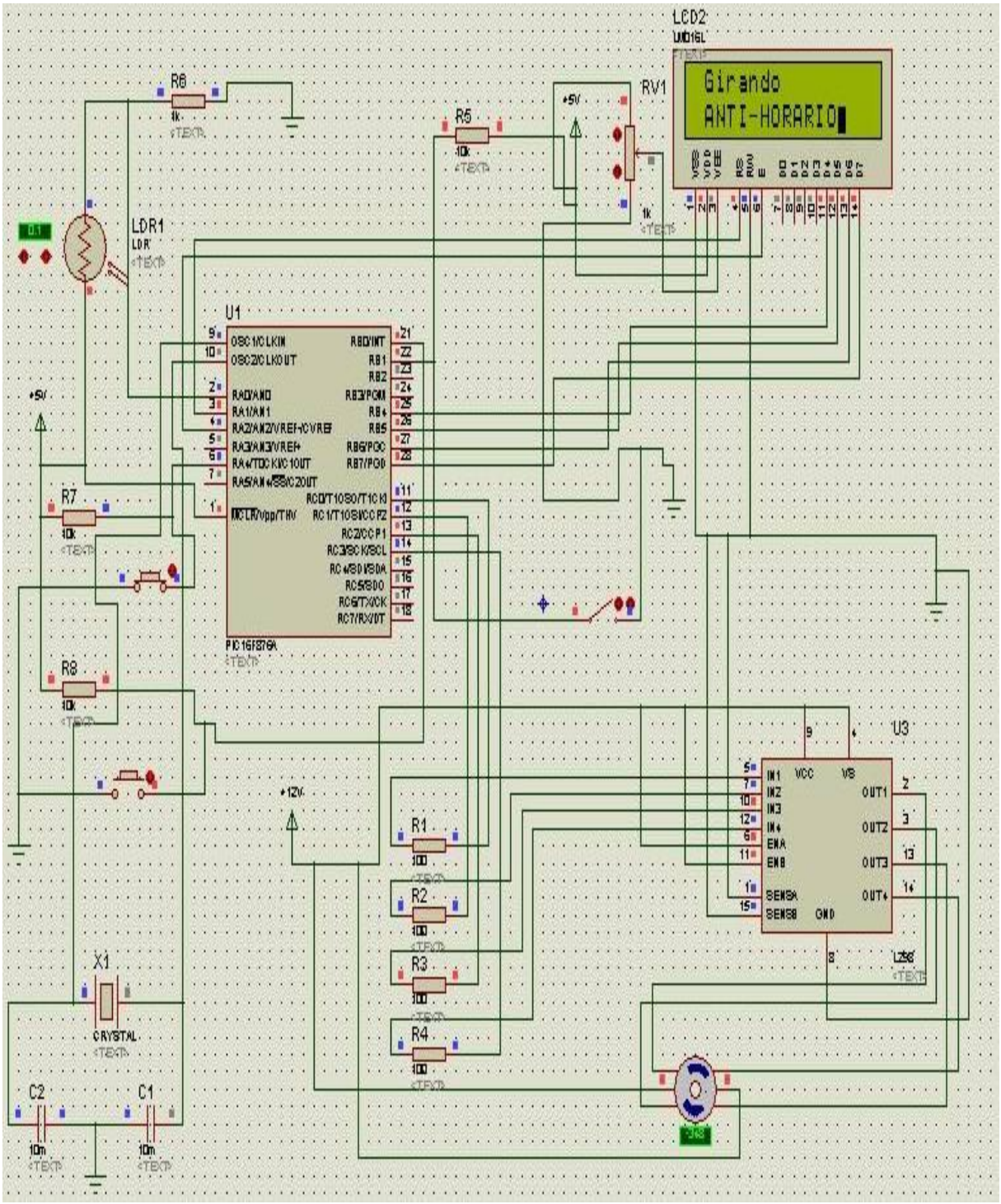


Figura 5.10 – GIRANDO ANTI-HORÁRIO (MANUAL)

[AUTOR]

Os testes do protótipo pronto foram realizados para o modo manual e automático, primeiramente verificando se no modo automático era realizado o giro até o número de passos calculado e se o display apresentava corretamente as informações. Já no modo manual foram realizados testes para ver se a persiana girava para os dois lados dependendo do botão pressionado, parava quando o botão deixava de ser pressionado e se parava quando chegava ao fim.

Para a realização desse projeto, várias dificuldades tiveram que ser superadas. Primeiramente a aquisição de um motor de passo que atendesse as especificações do projeto. Não foi possível no comércio local e a aquisição teve que ser realizada através da internet.

Após a compra do motor, a segunda dificuldade foi encontrar um CI que fizesse o motor girar nos dois sentidos. Então depois de várias tentativas com outros CIs como o ULN2003 foi utilizado com êxito o CI L298N.

Outra dificuldade encontrada foi com a construção do circuito para gravação, que foi resolvido com a compra de um kit PIC, aumentando o custo do projeto em R$ 189,90.

A dificuldade que demandou mais tempo para o projeto foi a de resolver como funciona a transformação A/D do LDR, pois com várias soluções encontradas em livros e na internet os erros continuaram acontecendo. Para a resolução desse problema, foi utilizada uma função para controlar essa transformação.

5.4 - Resultados da Aplicação do Modelo

A aplicação protege os objetos dispostos internamente no ambiente que possui essa solução.

A persiana funciona no modo automático como apresentado nas figuras a seguir, abrindo e fechando de acordo com a necessidade e apresentando cada movimento no display. Vídeo em CD anexo com a persiana realizando os movimentos.

Figura 5.11 – Foto do Display apresentando a mensagem Abrindo Persiana

[AUTOR]

Figura 5.12 – Foto do Display apresentando a mensagem Fechando Persiana

[AUTOR]

5.5 - Custos do modelo proposto

Para realizar o projeto, foram gastos R$ 189,90 com o Kit de desenvolvimento ACEPIC 18_28 (PIC16F876A), R$ 2,50 com o LDR, R$ 53,00 com o motor de passo, R$ 66,00 com a persiana (1,50 m x 0,40 m), R$ 5,00 com o CI L298N, R$ 4,00 com os resistores, R$ 2,00 com os capacitores, R$ 1,00 com a placa de circuito impresso e R$ 9,00 com os botões, totalizando R$ 332,40.

Uma boa forma para reduzir esse custo seria fazendo um circuito para gravação e para o LCD, pois o Kit é caro.

5.6 - Avaliação Global do Modelo

O sistema se mostrou capaz de gerar um produto comercial para ser utilizado em residências, empresas, hospitais e outros.

O projeto é útil para muitas pessoas, estas devem economizar com o reparo ou aquisição de objetos que possam ser danificados pela luz solar, por isso é bem possível que com o seu baixo custo, esse projeto possa ser utilizado em muitos lugares em pouco tempo.

CAPÍTULO 6 – CONCLUSÃO

Este projeto é voltado para a área de automação residencial, sendo que a área desenvolvida foi no controle das persianas perante a luminosidade. Para a realização desse projeto, foi necessário um sistema composto de software e hardware.

Para a implementação do projeto, foi utilizado como elemento principal, um microcontrolador da família PIC responsável por realizar o controle das características de cada componente, recebendo informações do sensor (LDR) e dos botões, organizando o que deveria ser feito, enviando comandos para que o motor pudesse realizar o movimento e escrevendo as informações necessárias no display de LCD. Tendo também um circuito integrado L298N, que possibilita que o motor de passo seja capaz de girar tanto no sentido horário quanto no sentido anti-horário.

O sistema se mostrou satisfatório no intuito de automatizar uma persiana, sendo que ainda permite trabalhar em modo manual e modo automático, ou seja, o usuário pode ter controle sobre a persiana, abrindo ou fechando quando quiser.

O sistema de uma forma geral se mostrou muito útil, podendo ser utilizado em locais residenciais, em locais comerciais, em hospitais e também em outros locais, trazendo comodidade, conforto e segurança para as pessoas ou as empresas.

6.1 - Sugestões para Trabalhos Futuros

Como sugestão para outros projetos nesta área, a inserção de mais LDRs para controlar também o lado em que há a maior incidência de luz e realizar o giro com mais precisão, pois saberíamos para que lado girar a persiana para que fosse possível evitar uma maior intensidade de luz. Seria um melhoramento relevante.

Outra boa sugestão seria a automação da residência por completo, tendo uma central única aonde seria possível ver e controlar a situação de cada ponto da casa de acordo com a movimentação. Essa automação poderia realizar tarefas on-line também, sendo controlado do lugar que o usuário estiver.

Outra sugestão seria a utilização de sensores de vento e chuva em uma integração realizada entre uma persiana e uma janela automatizadas, para que a persiana não estragasse

com o vento ou com a chuva que entra pela janela, diminuindo gastos com a manutenção da persiana.

Uma sugestão para que o projeto seja aprimorado, seria também a utilização de um controle remoto para que o utilizador pudesse controlar a alternação entre os modos manual e automático e realizar os movimentos de girar nos dois sentidos quando no modo manual.

REFERÊNCIAS BIBLIOGRAFICAS

FITZGERALD, A. E.; Charles Kingsley Jr.; Stephen D. Umans. **Máquinas Elétricas.** 6ª ed. Porto Alegre: Bookman, 2006.

LAVINIA, Nicolás César; David José de Souza; **Conectando o PIC – Recursos Avançados.** 2ª Ed. São Paulo: Editora Érica Ltda., 2005.

MATOS, Bruno Moreira; **Projeto Final - Janela Residencial Automatizada.** Brasília/DF (2º semestre de 2009).

PEREIRA, Fábio; **Microcontroladores PIC – Programação em C.** 7ª Ed. São Paulo: Editora Érica Ltda., 2009.

SOUZA, David José de; **Desbravando o PIC.** 8ª Ed. São Paulo: Editora Érica Ltda., 2005.

THOMAZINI, Daniel; Pedro Urbano Braga de Albuquerque; **Sensores Industriais – Fundamentos e Aplicações.** 1ª Ed. São Paulo: Editora Érica Ltda., 2005.

TORO, Vincent Del. **Fundamentos de Máquinas Elétricas.** Rio de Janeiro: LTC, 1994.

ACEPIC, Tecnologia e Treinamento. Disponível em (Novembro 2010):

http://www.acepiccamp.com.br

AUTOMATIZAR, Revista; Estamos preparados para a automação residencial? Disponível em (setembro 2010):

http://www.revistaautomatizar.com.br/PDF/Automatizar_02/34.pdf

BRITES, Felipe Gonçalves; Projeto Acadêmico pela Universidade Federal Fluminense. **Motor de Passo**; Julho / 2008. Disponível em (agosto 2010): http://www.telecom.uff.br/pet/petws/downloads/tutoriais/stepmotor/stepmotor2k81119.pdf

GAMEIRO, Natália S.; A. J. Marques Cardoso; **Modelação e Simulação do Motor de Relutância Variável Comutado.** Disponível em (Setembro 2010): http://www.deetc.isel.ipl.pt/jetc05/CCTE02/papers/finais/fortes/22.pdf

LUZ, Gabriela Yoshitani da; Maria Eduarda Scarsanella Miranda; Mariana Pereira Clemes; Marilia Ferrari; Artigo Acadêmico; Universidade Federal de Santa Catarina; **AUTOMAÇÃO RESIDENCIAL.** Julho/2009. Disponível em (agosto 2010):

http://www.arq.ufsc.br/arq5661/trabalhos_2009-1/automacao_residencial/automacao_residencial.pdf

MARTINS, Marcelo. **Samsung LED TV ou a grande diferença entre LCD/LED TV e Plasma/OLED**. Disponível em (setembro 2010): http://www.eletronicosforum.com/artigos_Samsung_LED_TV_diferenca_LCD_LED_Plasma_OLED.htm

ROSARIO, João Mauricio; Fórum Permanente Conhecimento & Tecnologia da Informação - AUTOMAÇÃO E TRABALHO (Sociedade Automatizada). Disponível em (setembro 2010):

www.cori.unicamp.br/foruns/tecno/evento5/rosario.ppt

TROPICAL, GN – Persianas externas, motores e componentes. Disponível em (Setembro 2010): http://persianasgntropical.weebly.com

APÊNDICE

Programa em C.

```
#include <16F876a.h>
#device adc=10
#use delay (clock=8000000)
#include <mod_lcd.c>
#fuses HS,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,NOCPD,NOWRT

#define MOT1 PIN_C1              // Declara porta para controlar 1º passo do motor
#define MOT2 PIN_C2              // Declara porta para controlar 2º passo do motor
#define MOT3 PIN_C3              // Declara porta para controlar 3º passo do motor
#define MOT4 PIN_C4              // Declara porta para controlar 4º passo do motor
#define bot1 PIN_B1              // Manual / Automático
#define bot2 PIN_A3              // Botão GIRA1 persiana
#define bot3 PIN_A5              // Botão GIRA2 persiana

void BOT_GIRA_1(void);                  // Função - Gira persiana sentido 1
void tempo_escuro_abre_persiana(void);  // Função - Abrir persiana escuro (AUTO)
void BOT_GIRA_2(void);                  // Função - Gira persiana sentido 2
void tempo_claro_fecha_persiana(void);  // Função - Fecha Persiana claro (AUTO)
void status_lcd(void);                  // Função - Escrever status no LCD
void transpassovalor(void);             // Função - transforma valor em passo
long AD(int CANAL);                     // Função - Ler entrada analógica
```

```
int passo=0,passovalor=0;
long valor;

//*****************************************************************************




long AD(int CANAL) // declara função usada para ler entrada analógica
{
    long AUXILIAR; //Declara uma variável de 16 bits
    enable_interrupts(GLOBAL); //Habilita uso de interrupção para conversão AD
    setup_adc_ports(ALL_ANALOG); //Habilita todas as analógicas
    setup_adc(ADC_CLOCK_INTERNAL);//Configuração do clock do conversor AD
    set_adc_channel(canal); //Congiguração do canal do conversor AD
    delay_ms(1000); //Tempo para selecionar canal lido
    AUXILIAR = read_adc(); //Faz a leitura e armazena na variável AUXILIAR
    setup_adc_ports(NO_ANALOGS); //Desativa entradas analógicas
    return(AUXILIAR); //Retorna valor analógico lido
}

//*****************************************************************************

Void main(){
        lcd_ini();                      // Inicia LCD
        for ( ;; ){
```

```
                valor = AD(0);
                transpassovalor();
                if(!input(bot1)){        // Modo automático
                        if((passovalor==115)&&(passo!=passovalor))
                                tempo_escuro_abre_persiana();
                        else if((passovalor!=passo)&&(passovalor!=115))
                                tempo_claro_fecha_persiana();
                        else
                                status_lcd();      //Escrever status no LCD
                }
                if(input(bot1)){        // Modo manual
                        if(!input(bot2) && passo < 215)
                                BOT_GIRA_1();
                        else if(!input(bot3) && passo > 0)
                                BOT_GIRA_2();
                        else
                                status_lcd();      //Escrever status no LCD
                }
        }
}

//*************************************************************************

void tempo_escuro_abre_persiana(void){              // Função - Abrir persiana quando escuro
        if(passo!=115)
                printf(lcd_escreve("\f LUZ BAIXA \nABRINDO PERSIANA"));
```

```
while(passo!=115 && !input(bot1)){          // Automático
        if(passo<115){
                delay_ms(50);
                output_high(MOT1);
                output_low(MOT2);
                output_low(MOT3);
                output_low(MOT4);
                delay_ms(50);
                output_low(MOT1);
                output_high(MOT2);
                output_low(MOT3);
                output_low(MOT4);
                delay_ms(50);
                output_low(MOT1);
                output_low(MOT2);
                output_high(MOT3);
                output_low(MOT4);
                delay_ms(50);
                output_low(MOT1);
                output_low(MOT2);
                output_low(MOT3);
                output_high(MOT4);
                delay_ms(50);
                output_low(MOT1);
                output_low(MOT2);
                output_low(MOT3);
```

```
            output_low(MOT4);
            passo++;
    }
    if(passo>115){
            delay_ms(50);
            output_low(MOT1);
            output_low(MOT2);
            output_low(MOT3);
            output_high(MOT4);
            delay_ms(50);
            output_low(MOT1);
            output_low(MOT2);
            output_high(MOT3);
            output_low(MOT4);
            delay_ms(50);
            output_low(MOT1);
            output_high(MOT2);
            output_low(MOT3);
            output_low(MOT4);
            delay_ms(50);
            output_high(MOT1);
            output_low(MOT2);
            output_low(MOT3);
            output_low(MOT4);
            delay_ms(50);
            output_low(MOT1);
```

```
                output_low(MOT2);
                output_low(MOT3);
                output_low(MOT4);
                passo--;
            }
        }
        status_lcd();           //Escrever status no LCD
}

//*****************************************************************************

void tempo_claro_fecha_persiana(void){              // Função - Fechar persiana quando claro
        if(passo!=passovalor)
            printf(lcd_escreve("\f LUZ ALTA \nFECHANDOPERSIANA"));
        while(passo>passovalor && !input(bot1)){
            delay_ms(50);
            output_low(MOT1);
            output_low(MOT2);
            output_low(MOT3);
            output_high(MOT4);
            delay_ms(50);
            output_low(MOT1);
            output_low(MOT2);
            output_high(MOT3);
            output_low(MOT4);
            delay_ms(50);
```

```
        output_low(MOT1);
        output_high(MOT2);
        output_low(MOT3);
        output_low(MOT4);
        delay_ms(50);
        output_high(MOT1);
        output_low(MOT2);
        output_low(MOT3);
        output_low(MOT4);
        delay_ms(50);
        output_low(MOT1);
        output_low(MOT2);
        output_low(MOT3);
        output_low(MOT4);
        passo--;
}
while(passo<passovalor && !input(bot1)){
        delay_ms(50);
        output_high(MOT1);
        output_low(MOT2);
        output_low(MOT3);
        output_low(MOT4);
        delay_ms(50);
        output_low(MOT1);
        output_high(MOT2);
        output_low(MOT3);
```

```
            output_low(MOT4);
            delay_ms(50);
            output_low(MOT1);
            output_low(MOT2);
            output_high(MOT3);
            output_low(MOT4);
            delay_ms(50);
            output_low(MOT1);
            output_low(MOT2);
            output_low(MOT3);
            output_high(MOT4);
            delay_ms(50);
            output_low(MOT1);
            output_low(MOT2);
            output_low(MOT3);
            output_low(MOT4);
            passo++;
        }
        status_lcd();           //Escrever status no LCD
}

//*************************************************************************

void BOT_GIRA_1(void){                          // Função - Gira persiana sentido 1
        printf(lcd_escreve("\f GIRANDO \n ANTI-HORARIO"));
        while (input(bot1) && !input(bot2) && passo < 215){
```

```
delay_ms(50);
output_high(MOT1);
output_low(MOT2);
output_low(MOT3);
output_low(MOT4);
delay_ms(50);
output_low(MOT1);
output_high(MOT2);
output_low(MOT3);
output_low(MOT4);
delay_ms(50);
output_low(MOT1);
output_low(MOT2);
output_high(MOT3);
output_low(MOT4);
delay_ms(50);
output_low(MOT1);
output_low(MOT2);
output_low(MOT3);
output_high(MOT4);
delay_ms(50);
output_low(MOT1);
output_low(MOT2);
output_low(MOT3);
output_low(MOT4);
passo++;
```

```
        }
        status_lcd();           //Escrever status no LCD
}

//*****************************************************************************

void BOT_GIRA_2(void){                          // // Função - Gira persiana sentido 2
        printf(lcd_escreve("\f GIRANDO  \n HORARIO"));
        while (input(bot1) && !input(bot3) && passo > 0){
                delay_ms(50);
                output_low(MOT1);
                output_low(MOT2);
                output_low(MOT3);
                output_high(MOT4);
                delay_ms(50);
                output_low(MOT1);
                output_low(MOT2);
                output_high(MOT3);
                output_low(MOT4);
                delay_ms(50);
                output_low(MOT1);
                output_high(MOT2);
                output_low(MOT3);
                output_low(MOT4);
                delay_ms(50);
                output_high(MOT1);
```

```
                output_low(MOT2);
                output_low(MOT3);
                output_low(MOT4);
                delay_ms(50);
                output_low(MOT1);
                output_low(MOT2);
                output_low(MOT3);
                output_low(MOT4);
                passo--;
        }
        status_lcd();           //Escrever status no LCD
}
//************************************************************************

void status_lcd(void){                          // Função - Escrever status no LCD
        if(passovalor==115&&passo!=115)
                printf(lcd_escreve("\f LUZ BAIXA \nPERSIANA FECHADA"));
        if(passovalor==115&&passo==115)
                printf(lcd_escreve("\f LUZ BAIXA \nPERSIANA ABERTA"));
        if(passovalor!=115&&passo!=115)
                printf(lcd_escreve("\f LUZ ALTA \nPERSIANA FECHADA"));
        if(passovalor!=115&&passo==115)
                printf(lcd_escreve("\f LUZ ALTA \nPERSIANA ABERTA"));
}

//************************************************************************
```

```
void transpassovalor(void){    // Função - transforma valor em passo
        if (valor>=0&&valor<200)
                passovalor=115;
        if (valor>=200&&valor<300)
                passovalor=90;
        if (valor>=300&&valor<350)
                passovalor=60;
        if (valor>=350&&valor<400)
                passovalor=30;
        if (valor>=400)
                passovalor=0;
}
```

ANEXOS

A - Microcontrolador

MICROCHIP

PIC16F87XA
Data Sheet

28/40-pin Enhanced FLASH Microcontrollers

MICROCHIP

PIC16F87XA

28/40-Pin Enhanced FLASH Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High Performance RISC CPU:

- Only 35 single word instructions to learn
- All single cycle instructions except for program branches, which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory, Up to 368 x 8 bytes of Data Memory (RAM), Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8 channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low power, high speed FLASH/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

PIC16F873A/876A devices are available only in 28-pin packages, while PIC16F874A/877A devices are available in 40-pin and 44-pin packages. All devices in the PIC16F87XA family share common architecture, with the following differences:

- the PIC16F873A and PIC16F876A have one-half of the total on-chip memory of the PIC16F874A and PIC16F877A
- the 28-pin devices have three I/O ports, while the 40/44-pin devices have five
- the 28-pin devices have 14 interrupts, while the 40/44-pin devices have 15
- the 28-pin devices have five A/D input channels, while the 40/44-pin devices have eight
- the Parallel Slave Port is implemented only on the 40/44-pin devices

The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

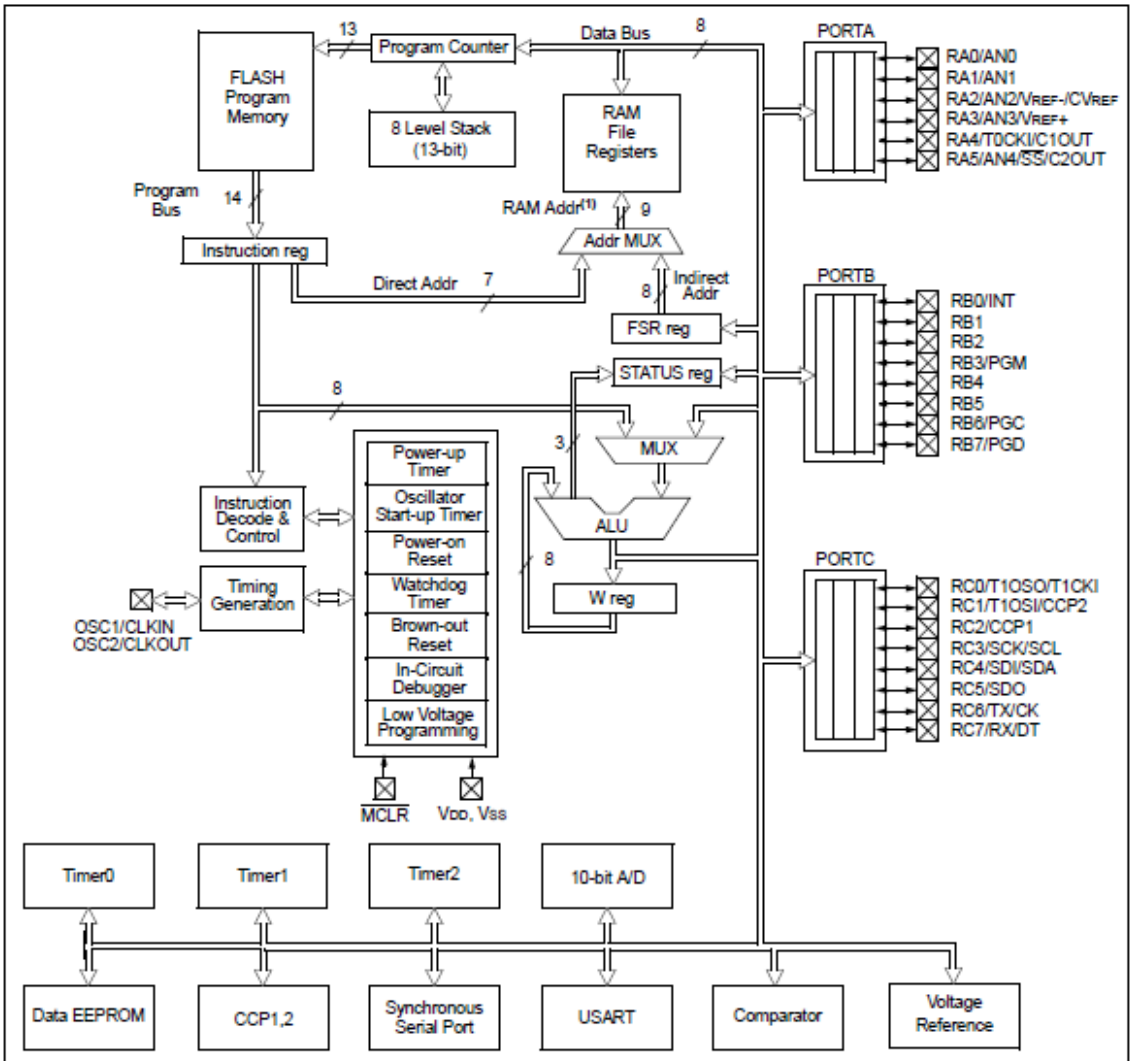
Additional information may be found in the PICmicro™ Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

TABLE 1-1: PIC16F87XA DEVICE FEATURES

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin MLF	40-pin PDIP 44-pin PLCC 44-pin QFP	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin MLF	40-pin PDIP 44-pin PLCC 44-pin QFP

PIC16F87XA

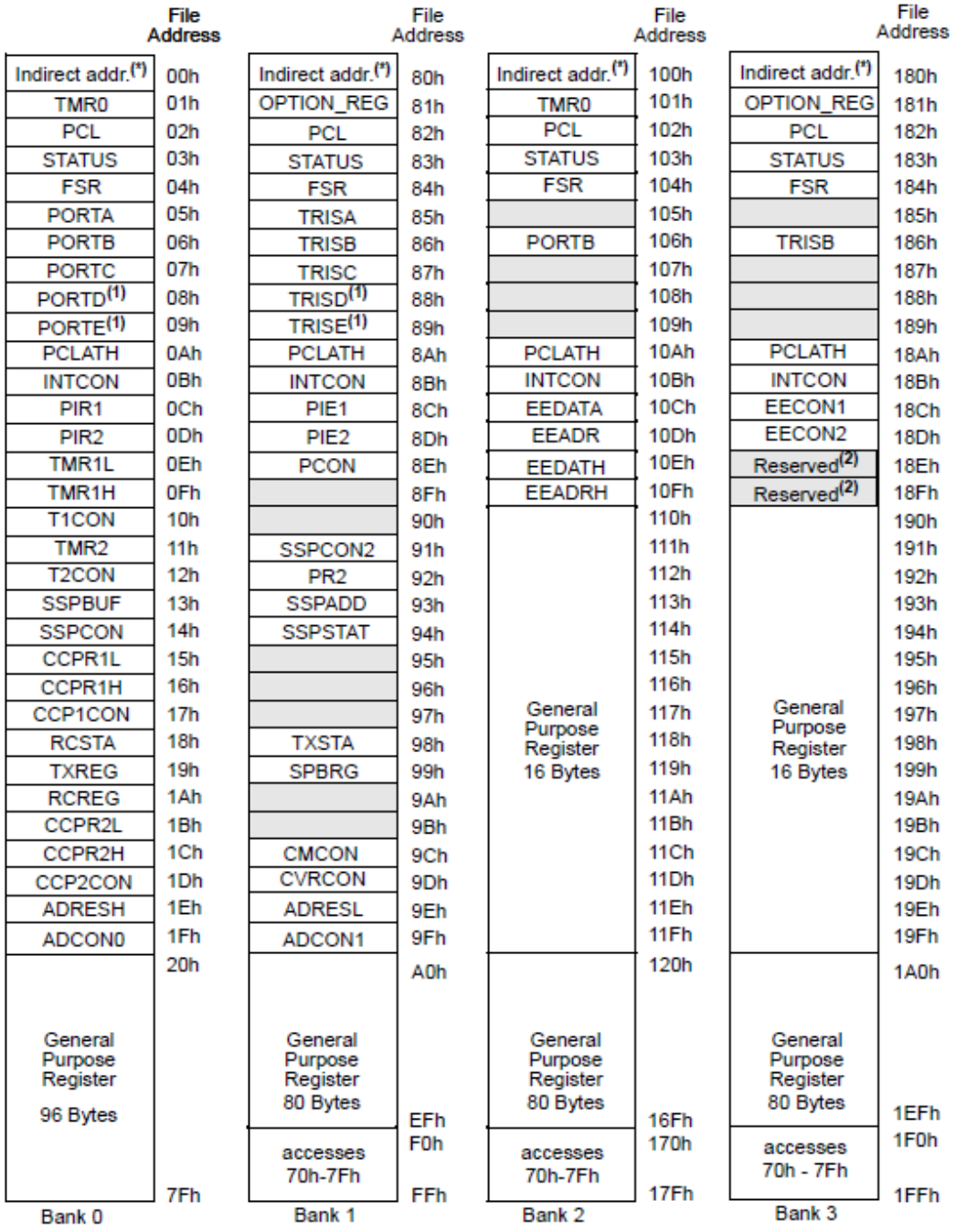
FIGURE 1-1: PIC16F873A/876A BLOCK DIAGRAM



Device	Program FLASH	Data Memory	Data EEPROM
PIC16F873A	4K words	192 Bytes	128 Bytes
PIC16F876A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the STATUS register.

FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP



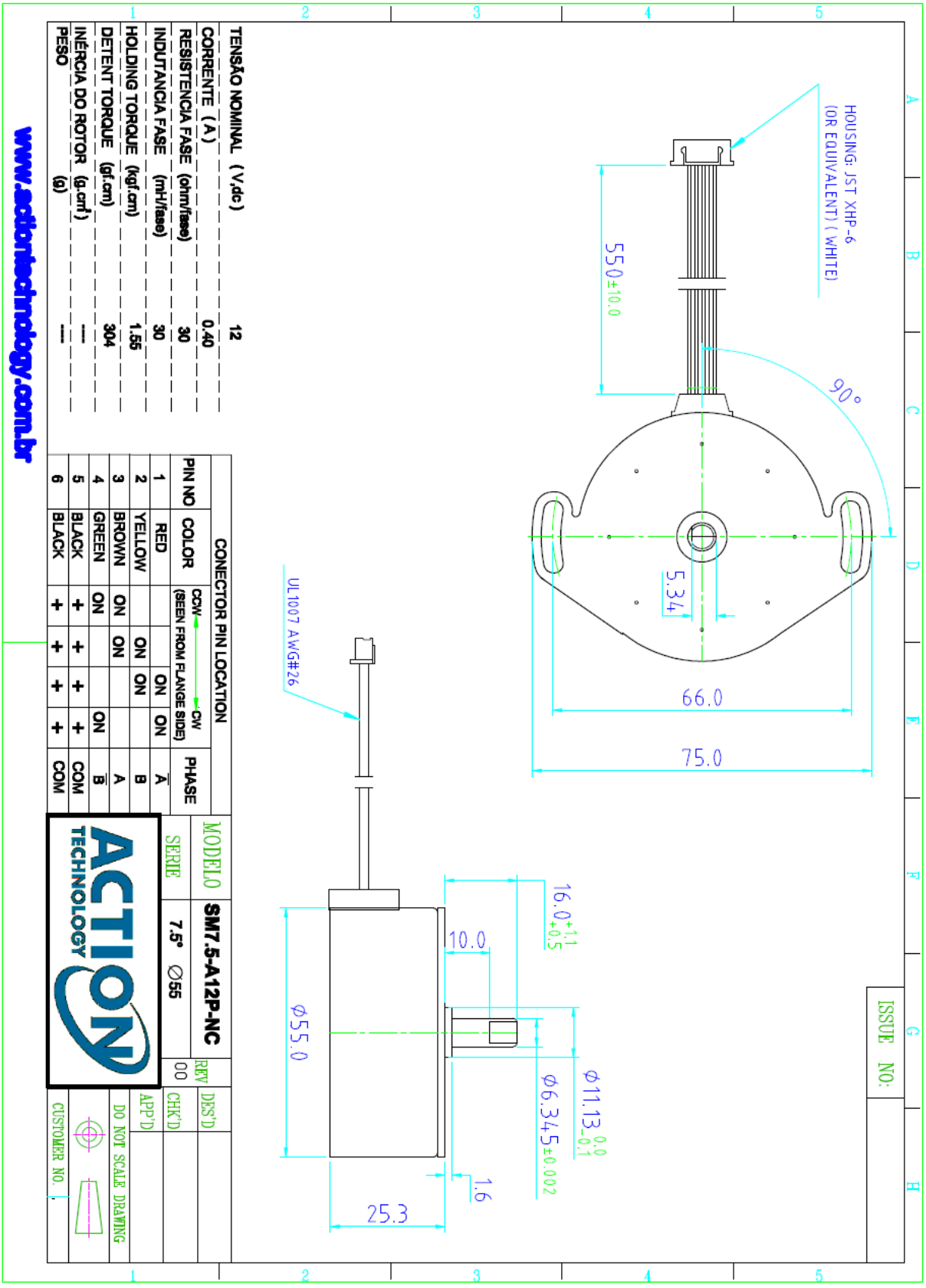
Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.

2: These registers are reserved, maintain these registers clear.

B – Motor de Passo

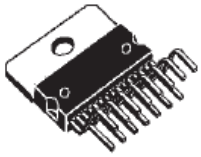


C – Datasheet CI L298N

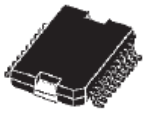
L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15



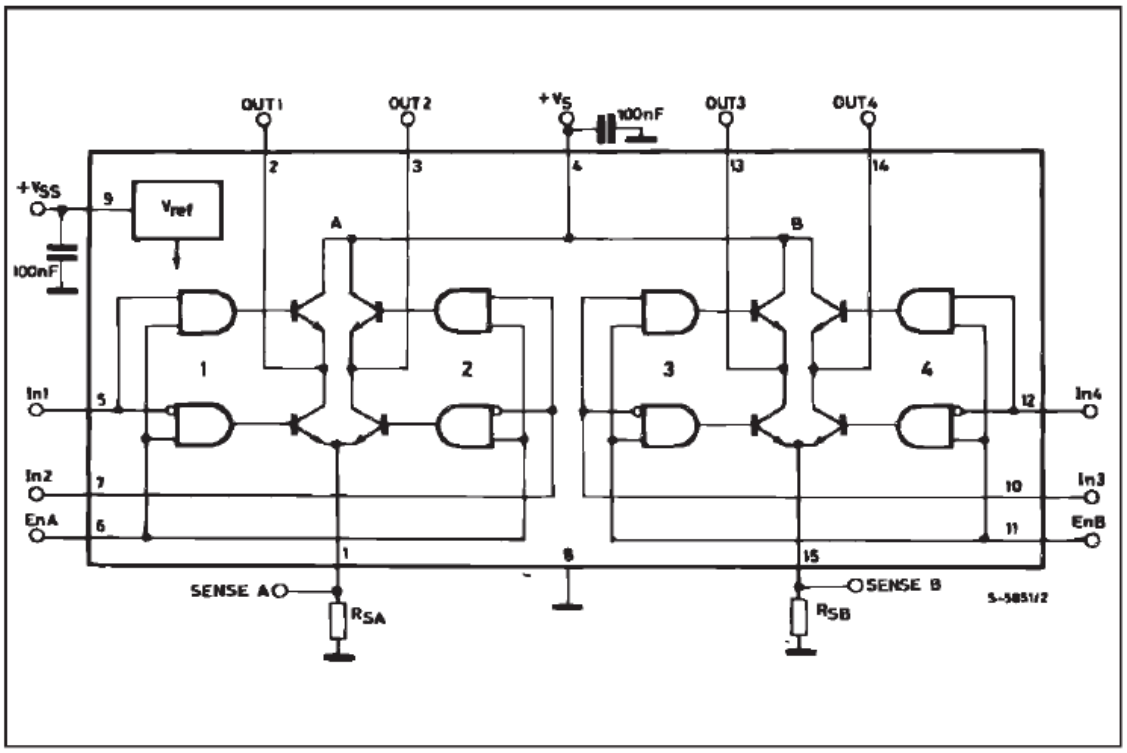
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

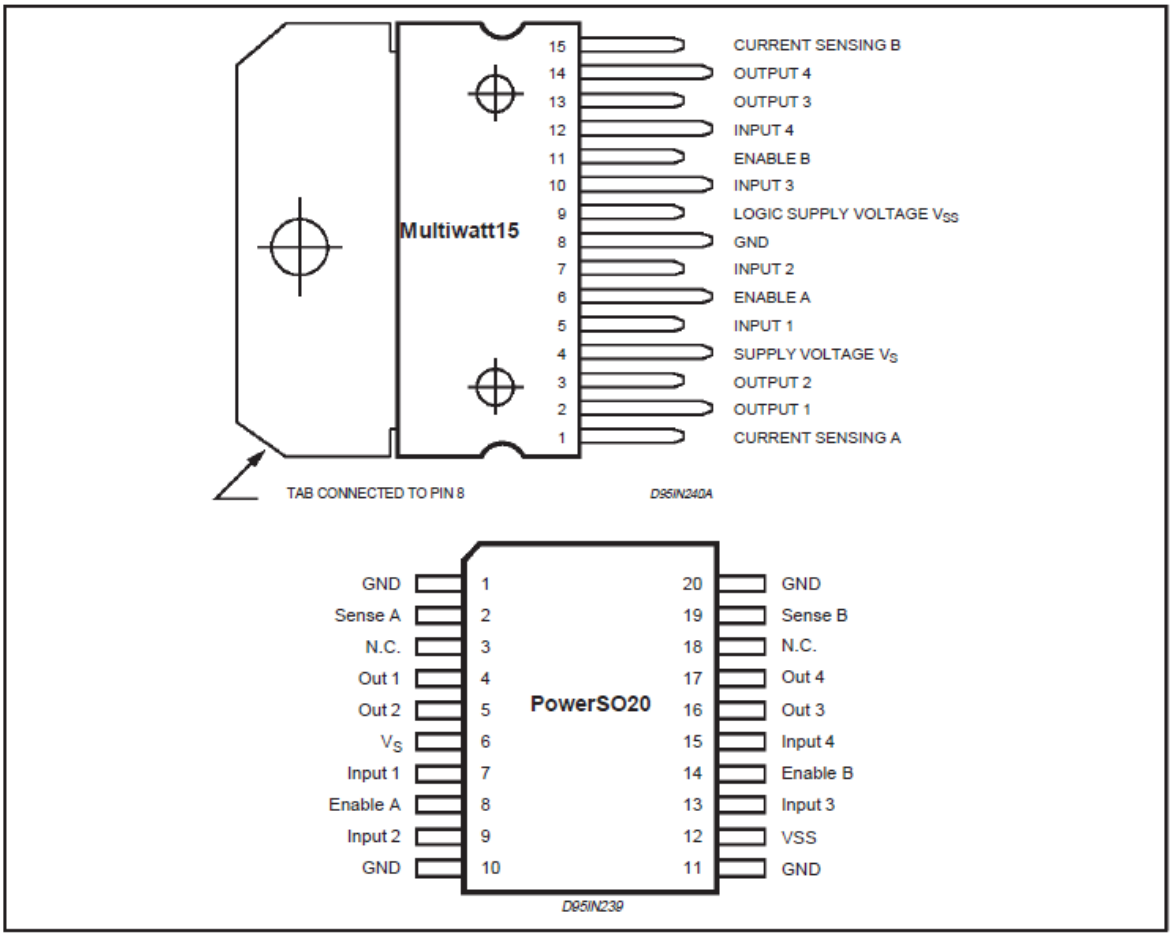
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition		V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)			4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0	V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L	V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0	V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L	V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)			–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)			2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L				–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V			30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)			–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)			2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L				–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V			30	100	µA
$V_{CEsat(H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A		0.95	1.35 2	1.7 2.7	V V
$V_{CEsat(L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)		0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)		1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)			–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ –0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

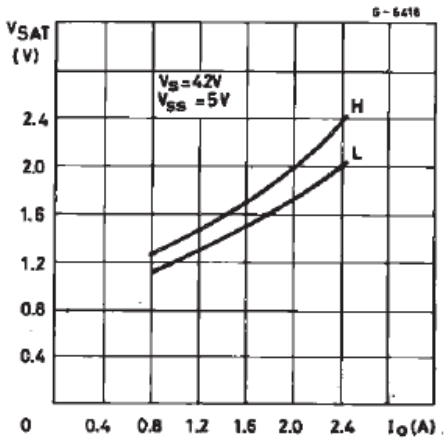
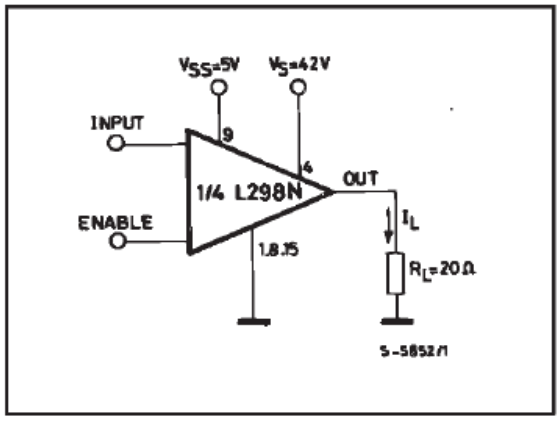


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

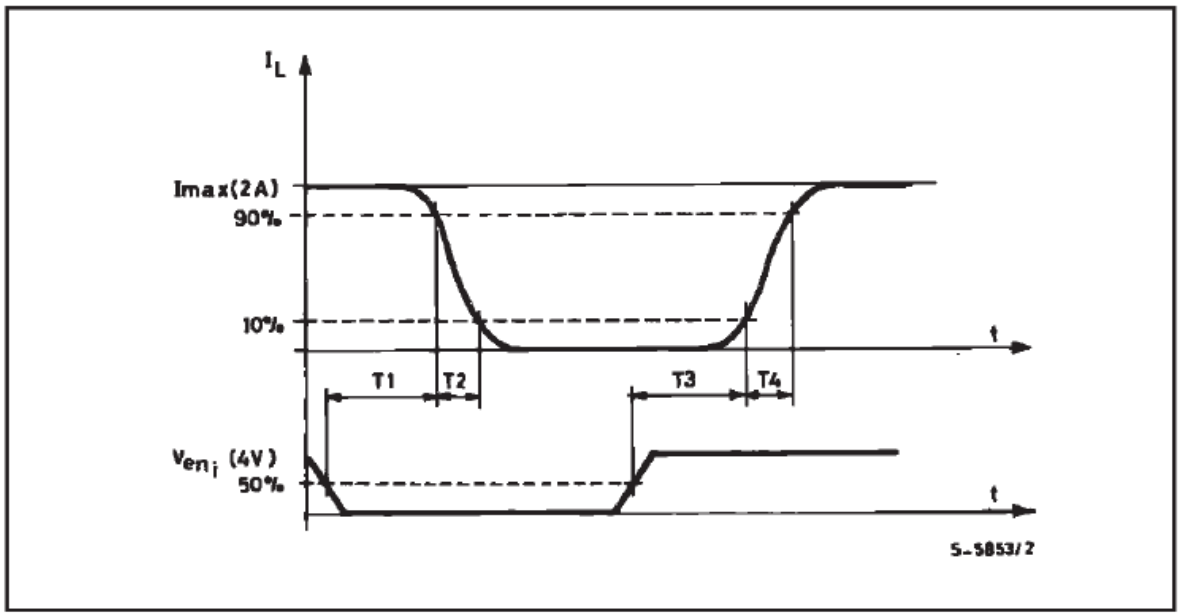
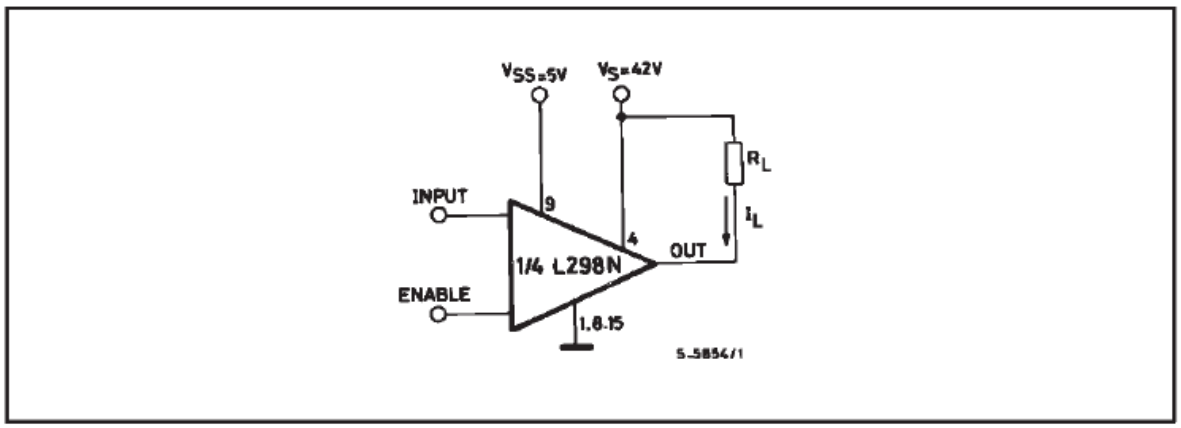


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

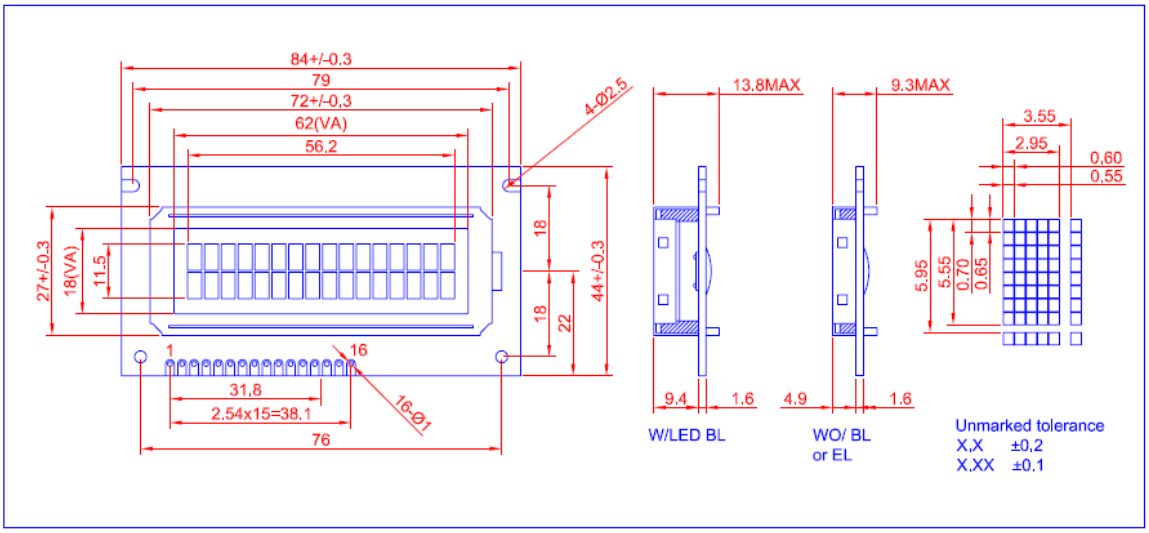
D – Características do AGM-1602B

AGTechnologies
agte@agte.com.br

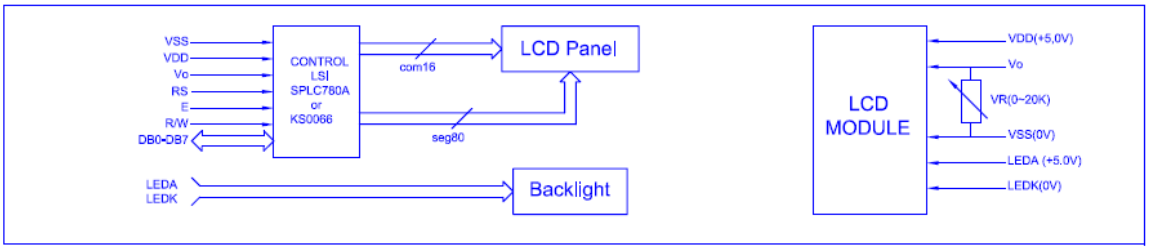
AGM-1602B SERIES

16x2 CHARACTERS
1/16 DUTY, 1/5 BIAS

1.0 DIMENSIONAL DRAWING



2.0 BLOCK DIAGRAM & POWER SUPPLY



3.0 MECHANICAL SPECIFICATIONS & FEATURE

Item	Nominal dimensions (mm)	FEATURE	
		LCD Type	TN, STN, FSTN
Module Size (W*H*T)	84.0x44.0x13.8(9.3)	LCD Colour	Yellow-Green, Gray, Blue
View Area (W*H)	62.0x18	View Angle	6 O'clock, 12 O'clock
Character Pitch (W*H)	3.55x5.95	Display Type	Positive, Negative Type
Character Size (W*H)	2.95x5.55	Rear polarizer	Transflect., Transmis., Reflect.,
Character Font	5x8	Operating Temperature	0°C ~ 50°C, -20°C ~ 70°C
Dot Pitch (W*H)	0.60x0.70	Storage Temperature	-20°C ~ 70°C, -30°C ~ 80°C
Dot Size (W*H)	0.55x0.65	Backlight	LED ARRAY, SIDE LED, EL

4.0 ELECTRICAL CHARACTERISTICS

Item	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Operating Voltage	Vdd	Ta=25°C	—	5.0	—	V
Operating Voltage for LCD	Vlcd	Ta=25°C	—	4.5	—	V
Supply Current	Idd	Ta=25°C, Vdd=5.0V	—	2.0	3.0	mA
Supply Current for Backlight	If	Ta=25°C, Vf=4.0V	—	110	—	mA

5.0 INTERFACE PIN CONNECTIONS

Pin No	Symbol	Level	Description
1~8	DB7-DB0	H/L	Data Bus Line
9	E	H/H -> L	Enable signal for chip
10	R/W	H/L	Read/write selection (H:Read;L:Write)
11	RS	H/L	Register selection (H:Data registor, L:Instruction registor)
12	Vo	–	Power supply for LCD
13	VDD	–	Power supply for Logic(+5.0V)
14	VSS	–	GND
15	LEDA	–	Voltage supply for Backlight (+)
16	LEDK	–	Voltage supply for Backlight (-)

E – Programa mod_lcd.c [PEREIRA, 2009]

```
// As definições a seguir são utilizadas para acesso aos pinos do display
// caso o pino RW não seja utilizado, comente a definição lcd_rw
//#include <16f876a.h>
//#use delay(clock=8000000)
#ifndef lcd_enable
        #define lcd_enable      pin_a2      // pino enable do LCD
        #define lcd_rs          pin_a1      // pino rs do LCD
        //#define lcd_rw        pin_e2      // pino rw do LCD
        #define lcd_b4                  pin_b4      // pino de dados d4 do LCD
        #define lcd_b5                  pin_b5      // pino de dados d5 do LCD
        #define lcd_b6                  pin_b6      // pino de dados d6 do LCD
        #define lcd_b7                  pin_b7      // pino de dados d7 do LCD
#endif

#define lcd_type 2          // 0=5x7, 1=5x10, 2=2 linhas
#define lcd_seg_lin 0x40    // Endereço da segunda linha na RAM do LCD

// a constante abaixo define a seqüência de inicialização do módulo LCD
byte CONST INI_LCD[4] = {0x20 | (lcd_type << 2), 0xf, 1, 6};

byte lcd_le_byte()
// lê um byte do LCD (somente com pino RW)
{
```

```
        byte dado;
        // configura os pinos de dados como entradas
        input(lcd_b4);
        input(lcd_b5);
        input(lcd_b6);
        input(lcd_b7);
        // se o pino rw for utilizado, coloca em 1
        #ifdef lcd_rw
                output_high(lcd_rw);
        #endif
output_high(lcd_enable); // habilita display
        dado = 0;       // zera a variável de leitura
        // lê os quatro bits mais significativos
        if (input(lcd_b7)) bit_set(dado,7);
        if (input(lcd_b6)) bit_set(dado,6);
        if (input(lcd_b5)) bit_set(dado,5);
        if (input(lcd_b4)) bit_set(dado,4);
        // dá um pulso na linha enable
        output_low(lcd_enable);
        output_high(lcd_enable);
        // lê os quatro bits menos significativos
        if (input(lcd_b7)) bit_set(dado,3);
        if (input(lcd_b6)) bit_set(dado,2);
        if (input(lcd_b5)) bit_set(dado,1);
        if (input(lcd_b4)) bit_set(dado,0);
        output_low(lcd_enable);       // desabilita o display
```

```
        return dado;    // retorna o byte lido
}

void lcd_envia_nibble( byte dado )
// envia um dado de quatro bits para o display
{
        // coloca os quatro bits nas saidas
        output_bit(lcd_b4,bit_test(dado,0));
        output_bit(lcd_b5,bit_test(dado,1));
        output_bit(lcd_b6,bit_test(dado,2));
        output_bit(lcd_b7,bit_test(dado,3));
        // dá um pulso na linha enable
        output_high(lcd_enable);
        output_low(lcd_enable);
}


void lcd_envia_byte( boolean endereco, byte dado )
{
        // coloca a linha rs em 0
        output_low(lcd_rs);
        // aguarda o display ficar desocupado
        //while ( bit_test(lcd_le_byte(),7) ) ;
        // configura a linha rs dependendo do modo selecionado
        output_bit(lcd_rs,endereco);
  delay_us(100);        // aguarda 100 us
```

```
        // caso a linha rw esteja definida, coloca em 0
        #ifdef lcd_rw
                output_low(lcd_rw);
        #endif
        // desativa linha enable
        output_low(lcd_enable);
        // envia a primeira parte do byte
        lcd_envia_nibble(dado >> 4);
        // envia a segunda parte do byte
        lcd_envia_nibble(dado & 0x0f);
}



void lcd_ini()
// rotina de inicialização do display
{
        byte conta;
        output_low(lcd_b4);
        output_low(lcd_b5);
        output_low(lcd_b6);
        output_low(lcd_b7);
        output_low(lcd_rs);
        #ifdef lcd_rw
                output_high(lcd_rw);
        #endif
        output_low(lcd_enable);
```

```
        delay_ms(15);
        // envia uma seqüência de 3 vezes 0x03
        // e depois 0x02 para configurar o módulo
        // para modo de 4 bits
        for(conta=1;conta<=3;++conta)
        {
                lcd_envia_nibble(3);
                delay_ms(5);
        }
        lcd_envia_nibble(2);
        // envia string de inicialização do display
        for(conta=0;conta<=3;++conta) lcd_envia_byte(0,INI_LCD[conta]);
}

void lcd_pos_xy( byte x, byte y)
{
  byte endereco;
  if(y!=1)
        endereco = lcd_seg_lin;
  else
        endereco = 0;
  endereco += x-1;
  lcd_envia_byte(0,0x80|endereco);
}

void lcd_escreve( char c)
```

```
// envia caractere para o display
{
  switch (c)
        {
    case '\f'     :       lcd_envia_byte(0,1);
                                delay_ms(2);
                                break;
    case '\n'     :
          case '\r'       :       lcd_pos_xy(1,2);
                                break;
    case '\b'     :       lcd_envia_byte(0,0x10);
                                break;
    default       :       lcd_envia_byte(1,c);
                                break;
  }
}

char lcd_le( byte x, byte y)
// le caractere do display
{
        char valor;
        // seleciona a posição do caractere
        lcd_pos_xy(x,y);
        // ativa rs
        output_high(lcd_rs);
        // lê o caractere
```

```
        valor = lcd_le_byte();
        // desativa rs
        output_low(lcd_rs);
        // retorna o valor do caractere
        return valor;
}
```