UNIVERSIDADE SÃO FRANCISCO

CURSO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE UM PROTÓTIPO DE ROBÔ MÓVEL AUTÔNOMO: HARDWARE E SOFTWARE

Área de Robótica

por

André Luis Gasparoti

Paulo Eduardo Silveira, Mestre
Orientador

Itatiba (SP), dezembro de 2009

UNIVERSIDADE SÃO FRANCISCO

CURSO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE UM PROTÓTIPO DE ROBÔ MÓVEL AUTÔNOMO: HARDWARE E SOFTWARE

André Luis Gasparoti

Monografia apresentada à Banca Examinadora
do Trabalho de Conclusão do Curso de
Engenharia Elétrica para análise e aprovação.
Orientador: Paulo Eduardo Silveira, Mestre

Itatiba (SP), dezembro de 2009

AGRADECIMENTOS

Meus sinceros agradecimentos a todos que de alguma forma auxiliaram-me na realização deste trabalho, em especial:

A Deus, pelas oportunidades e bênçãos que surgiram em meu caminho.

À minha família, pela base sólida que me deu forças para encarar a vida de frente. À minha mãe Edna, meu pai Orrivail e minhas irmãs Mariana e Mila, pelas noites em claro, pelo amor, amparo, apoio, sustento e compreensão que proporcionaram ao longo destes duros anos.

À minha amada Suzana por acrescentar beleza e razão aos meus dias, pelo carinho, amor e compreensão, principalmente nos momentos difíceis desta jornada.

Aos meus amigos e companheiros de estrada: André Cristiano, Cícero, Davilson e Júlio, por toda a ajuda, amizade e companheirismo que dispuseram durante estes anos escolares em que estivemos juntos.

Ao meu orientador Paulo Silveira, por não desistir e acreditar em mim. Por proporcionar e compartilhar de seus conhecimentos, que não só me auxiliaram neste trabalho, mas que também serão de grande utilidade em meu futuro profissional.

Em especial ao professor Ely Paiva, no qual dedico este trabalho, pois através dele tive a oportunidade do contato com o saber e desenvolver tendências pela área de robótica. Durante os anos de curso, fora nosso mestre, tutor, amigo, proporcionando conhecimento ímpar. Agradeço pela ajuda, motivação, pela confiança que depositou em minha pessoa ao ofertar de seu tempo, conhecimento e recursos para que pudesse desenvolver-me durante os anos que tive o privilégio ser seu aluno.

"Escolhas um trabalho de que gostes e não terás que trabalhar um único dia em tua vida"

Confúcio

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

GASPAROTI, André L. **Protótipo de um Robô Móvel Autônomo: hardware e software.** Itatiba, 2009. no f. Trabalho de Conclusão de Curso, Universidade São Francisco, Itatiba, 2009.

Este trabalho tem por objetivo desenvolver um protótipo de um robô móvel autônomo que interaja com o meio através da leitura de uma linha térrea, dotado de sensores infravermelhos (seguidor de linha) e dispositivos mecânicos de proteção contra obstáculos. Com este intuito, fora implementado um circuito de controle para motores CC, por pwm, e utilizado um microcontrolador PIC16F877A para interagir com os periféricos e armazenar o programa desenvolvido em linguagem C. Inicialmente, é feita uma revisão bibliográfica sobre dispositivos básicos que compõe um sistema robótico: periféricos e controles. Em seguida, o sistema de controle, os sensores e a programação propostos são descritos e comentados os resultados obtidos em simulador e *protoboard*.Por fim, são expostos o *layout* da placa em circuito impresso e o robô montado.

Palavras-chave: Robôs Móveis, Sistemas de Controles, Microcontrolador.

ABSTRACT

The objective to this work is a develop a prototype of Autonomous Mobile Robots, that interact with the means through of read the ground line, gifted infrared sensor (Line Follower) and mechanic device to protection into hurdle. With this motif, it was implanted the controls of circuit to motors CC, with pwm, and used the microcontroller PIC1FF877A for interactive with peripheral and hold the software developed into language C. At first, it's make the bibliography revision about basic devises that compose in system of robot: peripheral and controls. In a row, the system of control, the sensors and program proposed were explained and commented the effect gotten in the software simulator and protoboard. Eventually, it was exposed the layout and robot assembled.

***Keywords**: Mobile Robots, System of Control, Microcontroller.*

1. INTRODUÇÃO

A robótica tem crescido muito nos últimos anos. Alimentados por histórias de ficção científicas, no qual o robô está presente no cotidiano para desempenhar diversos trabalhos, na atualidade, inúmeros são os que têm tido por objetivo desenvolver autômatos para diversos fins: em linhas de montagens industriais, equipamentos hospitalares (cirúrgicos), robôs bracejadores para inspeção em linhas de transmissão elétrica, dutos, exploração oceânica, extração de petróleo, exploração espacial, enfim, em todo lugar onde o ser humano estaria exposto a situações perigosas que o colocaria em risco ou a algum dano físico.

Com o interesse em estudar os componentes que compõe um circuito eletrônico de um robô e lógicas de programação para desempenhar funções autônomas, o intuito deste presente trabalho é utilizar um protótipo de um robô móvel e aplicar algumas funções lógicas, utilizando uma linguagem de programação em alto nível a fim de executar aplicações dando uma funcionalidade e assim automatizá-lo.

Para isto é necessário estudar os tipos de sensores que atuarão no robô, o conceito de motores (maior torque ou velocidade, potência consumida), fontes de energia (pilhas ou baterias), o estudo do microcontrolador a ser utilizado (seu hardware e firmware, capacidade de memória) e os demais componentes que constituem o esquemático eletrônico: cristal, filtros, e demais circuitos que proporcionem o bom funcionamento do protótipo.

Outro tópico é a linguagem de programação a ser empregada (rotinas e algoritmos de programação), qual software de compilação e gravação compatíveis com o microcontrolador.

A tecnologia está em todos os seguimentos e também em constante atualização. Através do estudo da robótica, do universo que o compõe, e do posicionamento para solução de problemas é que, encontrar-se-á ferramentas para abranger os conhecimentos e romper as barreiras da evolução.

1.1. OBJETIVOS

O trabalho proposto é do desenvolvimento de um protótipo de um robô móvel autônomo, tendo por interface e atuação ao microcontrolador: sensores analógicos e digitais, sistema PWM para controle de velocidade dos motores, sistema de inversão de rotação, e demais sub-sistemas a fim de tornar possível o seu funcionamento.

Aplicar-se-á, através de uma linguagem de programação de alto nível, algumas rotinas para locomoção e lógicas de supervisionamento.

2. REVISÃO BIBLIOGRÁFICA

2.1. ROBÓTICA

Robôs são máquinas, controladas ou autônomas, que desempenham funções como se mover, manipular objetos e interagir com o ambiente. Segundo Fernando Pazos (PAZOS, 2002), "são capazes de executar, com uma velocidade sobre-humana, determinadas rotinas repetidamente e com perfeita precisão. Com estes quesitos, o robô substituiu o homem em diversas áreas onde se empregava o trabalho manual e exigia demasiadamente do trabalhador".

Em 1770 foi inventado o primeiro dispositivo mecânico, controlado por mecanismo de relógio, movimentava peças, cordas e sinos. Henri Maillardet construiu, em 1805, um boneco capaz de escrever e desenhar, levando 5 minutos para executar uma tarefa e tinha vários itens em sua memória mecânica, com a possibilidade de seleciona-las. Estas criações mecânicas devem ser analisadas como forma de explorar a capacidade humana, a seu tempo, e o desejo aprimorar as técnicas de trabalho. Outras máquinas foram criadas: fiandeira de fusos múltiplos, tear mecânico entre outras (PAZOS, 2002).

Para Marcelo (ZANELATTO, 2004), duas tecnologias podem ser denominadas antecessoras da robótica: o telecomando e o comando numérico. O telecomando se trata do uso de manipuladores remotos controlados pelo ser humano. Manipuladores são dispositivos eletromecânicos que produzem movimentos indicados pelo operador através de um dispositivo adequado, como por exemplo, um joystick. O comando numérico, desenvolvido no final da década de 40 e início de 50, fora utilizado para controlar ações de uma máquina operatriz e programado por meio numérico através de um teclado ou leitura do cartão perfurado. Com esta combinação, a base para demais invenções estava feita. Diversas invenções surgiram, em 1946, George C. Devol desenvolveu um dispositivo que registrava sinais elétricos magneticamente e reproduzia-os para controlar a máquina, e em 1954, por Cyril Walter Kenward, o primeiro a patentear um dispositivo robótico.

Entretanto, a idéia de que temos hoje de robô moderno, surgiu em 1962, por Joseph Engelberger que, juntamente com Devol, desenvolveu o primeiro protótipo, denominado *Unimate*, utilizado na Ford Motor Company, para descarregamento de uma máquina de fundição de pressão. Após este invento, desenvolveram o *PUMA*, um robô com braços articulado, baseado em estudos de automação e montagem realizados pela General Motors (PAZOS, 2002).

Na década de 70, com o aumento da velocidade, os controles passaram a ser feitos por controladores digitais, como o CLP (Controlador Lógico Programável). E diversas outras aplicações, no campo da robótica foram desenvolvidas, para as mais diversas aplicações, desde robôs móveis autônomos, até humanóides com capacidade de interagirem com o ambiente e criar ações de acordo com funções pré-programadas (ZANELATTO, 2004).

2.2. TIPOS DE ROBÔS

Pelo fato das diferentes funções que os robôs podem exercer, é possível classifica-los em:

- Robôs Inteligentes: são manipulados por sistemas multifuncionais controlados ou não por computadores, capazes de interagir com o ambiente, tomando decisões em tempo real devido aos sensores que os integram;
- Robôs controlados por computadores: semelhantes aos robôs inteligentes, não são dotados de sensores, assim, inexiste a autonomia devido à falta de estímulos externos que não sejam dadas pelo computador;
- Robôs de aprendizagem: limitam-se a repetir seqüência de movimentos, realizados com intervenção de um operador ou memorizadas em seu hardware;
- Manipuladores: sistemas mecânicos multifuncionais, cujo sistema permite governar seus membros de forma manual ou seqüenciável, através de um ciclo de trabalho.

Também é possível classificar os robôs partindo do ponto de visto do controle de seus movimentos:

- Sem controle-servo: o programa que controla os movimentos dos componentes do robô se dá em um posicionamento ponto-a-ponto no espaço;
- Com controle-servo: permite o movimento do robô em função de seus eixos, realizados de forma trajetória contínua ou ponto-a-ponto, tanto o corpo quanto um de seus membros.

2.3. ROBÓTICA MÓVEL AUTÔNOMA

Nas últimas décadas, o desenvolvimento e tecnologias computacionais, eletrônicas e mecânicas, controles sofisticados, incluindo algoritmos complexos, possibilitaram o surgimento de

plataformas robóticas inteligentes, com capacidade de se locomoverem, desviarem de obstáculos, supervisionarem territórios, coletarem objetos, além analises de solos e demais aplicações (CRAIG, 2005).

Os Robôs Móveis Autônomos (RMA) carecem de alguns dispositivos para que seja dada a autonomia e a implementação das funções estabelecidas. Sensores, motores para locomoção ou execução de algum artífice devem estar ligados a algum controle ou dispositivo que ofereça a possibilidade de incrementar lógicas ao robô. Duas linhas de pesquisa voltadas ao controle para robôs: o sistema planificado e reativo.

O "sistema planificado" (ZANELATTO, 2004) é baseado nas ações do robô em um modelo gravado na memória. No sistema reativo, não há modelos predefinidos, o robô interage com o meio através dos sensores e atuadores que o compõe, e através destes estímulos, decidir sobre as ações a realizar. Há vantagens e desvantagens em ambas filosofias. No sistema planificado há maior facilidade em implementação, devido ao conhecimento do território, entretanto torna-se limitado para aquele ambiente para o qual fora programado. Para um sistema reativo, há complexidade em programar, além da eletrônica elaborada, porém existe maior liberdade e autonomia em diversos ambientes.

Os robôs autônomos são divididos e três áreas de aplicação:

- Industriais: plataformas móveis no qual executam tarefas pesadas como o carregamento de materiais pesados. São limitados com relação à autonomia e, a princípio, o método de navegação era dado por uma linha no chão, o que requeria altos investimentos nas mudanças de infra-estrutura na planta das empresas. Exemplos típicos: carregadores de materiais em fábricas e depósitos, no qual são orientados por linhas no chão;
- Serviços: executam tarefas de serviços para diversas aplicações, tais como transporte, manipulação, exploração, vigilância, entre outros no qual exijam um mínimo de conhecimento prévio do ambiente estruturado. Uma aplicação de serviço é a de limpeza de metrôs em determinados países. Outras finalidades são os robôs destinados à pesquisa e desenvolvimento de tecnologias;
- Campo: realiza tarefas em ambientes desestruturados, e em grande parte perigosos. O desenvolvimento para esta modalidade tem crescido muito com as novas tecnologias e

investimentos para aplicações como explorações espaciais, marítimas, em vulcões, dutos de gás, limpeza de acidentes nucleares, navegação em auto-estradas, entre outros.

2.4. CONTROLE DO SISTEMA ROBÓTICO

Um sistema robótico pode ser dividido em sub-sistemas eletrônicos e mecânicos capacitando o robô em sua autonomia. Estes subsistemas estão interligados a uma planta, cuja finalidade é receber os estímulos dos sensores e enviar aos atuadores e controles, seguindo a uma lógica gravada em sua memória. Conforme mencionado, quando o robô não é controlado por um computador, necessita-se de que um ou mais dispositivos aloquem em memórias a estratégia de controle ou lógica de programação. A filosofia irá determinar a aplicação e o grau de liberdade com que o RMA atuará em um ambiente estruturado ou não. Na Figura 1, um diagrama de blocos está representando um sistema robótico completo, contendo os vários tipos possíveis de interfaces.

Os sensores têm grande importância, devido ao fato de serem a única forma de receber os estímulos do ambiente. No bloco de controle encerram os *drivers* que proporcionarão ao robô os meios de locomoção. Pode ser motor CC, servo-motor ou motor de passo, o dispositivo irá determinar o tipo de controle. O bloco de atuação englobará tanto atuadores mecânicos analógicos quanto digitais: Led's, displays, pistão, braços robóticos, garras, CI's, entre outros. Com a determinação das cargas, deverá ser estipulada a forma de alimentação, considerando: tensão e corrente fornecida, e tempo de autonomia do robô.

A seguir, serão estudados separadamente os blocos principais que compõe a plataforma robótica.

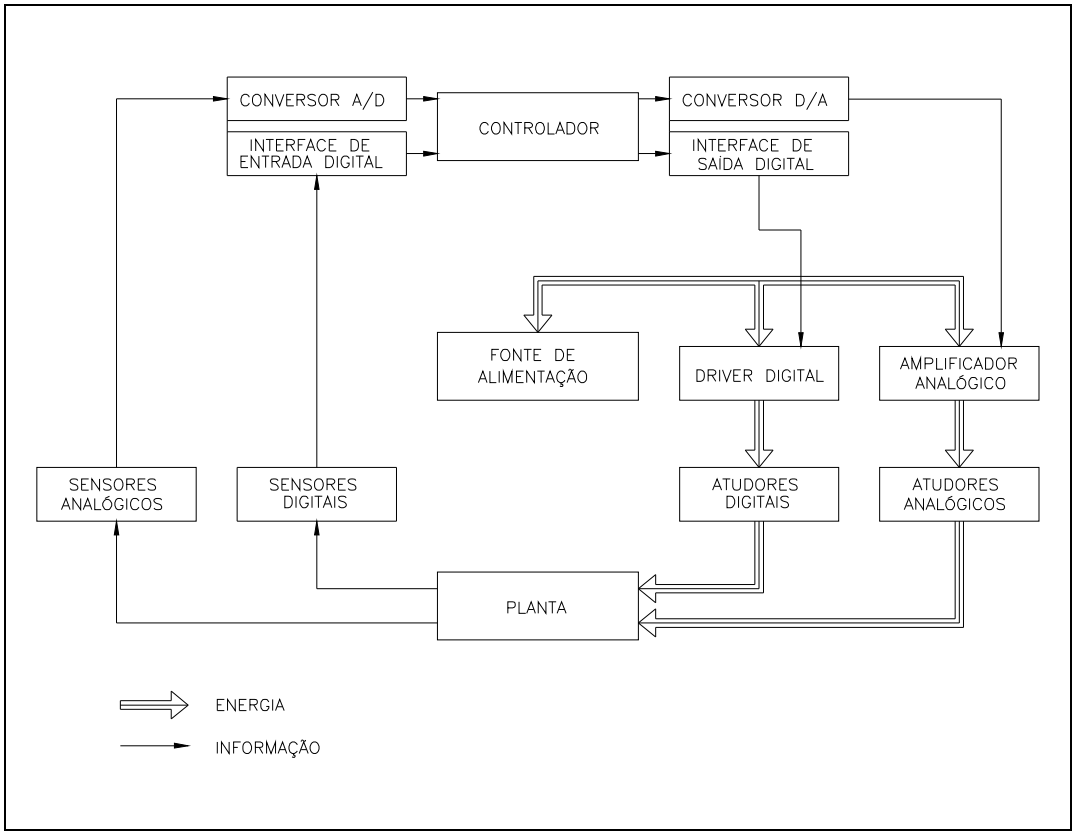


Figura 1. Diagrama de blocos de um sistema RMA.
Fonte: Adaptado de PAZOS (2002)

2.5. SENSORES

As funções dos sensores são as mesmas desempenhadas pelos sentidos humanos, captar através de estímulos externos o que se encontra no ambiente. “Sensor, ou transdutor, é um dispositivo que entrega um sinal elétrico proporcional a uma grandeza física mensurada” (PAZOS, 2002). Para todo um sistema em malha fechada, há a necessidade de algum sensor, exceto quando controlador e a planta trabalhem com as mesmas grandezas físicas, por exemplo, um sistema mecânico com regulagem mecânica da velocidade, ou um amplificado operacional realimentado.

Os sensores podem ser analógicos ou digitais. O transdutor analógico toma qualquer valor ao longo do tempo. Exemplos de grandezas analógicas: pressão, temperatura, umidade, velocidade, distância, luminosidade, altitude, torque, entre outras. Sensores digitais assumem apenas dois valores ao longo do tempo: 0 e 1. Estes valores não necessariamente precisam ser grandezas físicas. Exemplo: presença de algum objeto em determinado local.

As características comuns dos transdutores são: faixa (ou range), resolução, sensibilidade, linearidade, histerese, exatidão ou erro, relação sinal/ruído, entre outras mais específicas para determinados tipos de sensores (WERNECK, 1996).

- Faixa: todo nível de amplitude (escala) da grandeza medida, supondo que o sensor opera dentro de uma precisão especificada;
- Resolução: menor incremento da grandeza medida que provoca uma mudança no sinal de saída do sensor;
- Sensibilidade: relação de saída e a grandeza medida, isto é, a função de transferência do sensor;
- Linearidade: quando há variações iguais medidas e se obtêm variações iguais na saída, define-se que sensor possui certa linearidade;
- Histerese: fenômeno que ocorre quando um sinal de entrada é maior que o normal e o sinal de saída decresce até um valor de entrada normal. Há diversos fatores para produzir-se histerese no sistema. Como por exemplo, quando um sensor de posição faz a leitura enquanto as engrenagens de locomoção possuem folgas entre os dentes;
- Exatidão ou Erro: diferença entre o valor real do sinal de saída entregue pelo transdutor e o sinal ideal que deveria ser fornecido;
- Relação Sinal/Ruído: relação entre a potência do sinal de saída e a potência do ruído gerado, ou seja, e o sinal de entrada for nulo e na saída houver algum sinal, este será denominado ruído.

Os RMA's utilizam basicamente dois tipos de sensores: de posicionamento e de colisão.

2.5.1. Sensores de Posicionamento

Responsáveis por fornecer ao robô a localização, estes sensores medem de forma relativa ou absoluta a posição atual do robô e os envia para que possa atuar no ambiente. São eles: GPS, odometria, bússolas e Beacon, alguns dos sensores mais utilizados para este fim.

- GPS: através de um sistema de rede de satélite, é capaz de determinar longitude, latitude e altitude do robô. Oferece um valor absoluto, embora a desvantagem de ser ineficiente

em lugares urbanos e de precisão baixa para uso civil. Entretanto, é amplamente utilizado para RMA's de campo;

- Odometria: mede a distância percorrida pelo robô, calculando, assim, a posição em relação ao ambiente;
- Bússola: informa o ângulo no qual o robô se encontra em determinado momento. Uma vantagem é a da possibilidade de retornar a um valor absoluto, sem que haja a necessidade de depender de um estado anterior do mesmo;
- Beacon: dispositivos instalados em pontos estratégicos com a finalidade de emitir sinais luminosos ou via rádio e a partir destes sinais fazer com que o robô seja capaz de calcular a posição.

2.5.2. Sensores de colisão

Utilizados para desvio de obstáculos, são responsáveis também por supervisionar o ambiente, criando um mapa em sua memória, dependendo de sua execução. Alguns deles: infravermelho, ultra-sônico, laser, câmera de vídeo, contato (mecânico).

- Infravermelho: pelo baixo custo e simplicidade na implementação, este sensor é amplamente utilizado, embora ao raio de ação reduzido. Através de um diodo emissor infravermelho, a luz é enviada e parte dela refletida, sendo captada por um receptor ótico, um fototransistor. A luz faz com que na base do fototransistor haja uma recombinação fazendo-o saturar e forçando a uma saída lógica zero. Quando um objeto obstrui o caminho da luz, a tensão passa a ser nível lógico 1 de saída;
- Ultra-sônico: ao contrário do infravermelho, o sonar tem um raio de atuação muito largo, sendo assim uma desvantagem. Como vantagem, o sonar é de baixo custo e de fácil operação, sem a necessidade de muitos recursos computacionais. O seu funcionamento se dá através da emissão de ondas de som em alta freqüência e ao atingir um objeto, a onda refletida é captada pelo transdutor. A distância será calculada dividindo o tempo percorrido por 2 e multiplicando pela velocidade do som;
- Laser: há vários tipos de tecnologias que envolvem os sensores a laser. Um caso típico é o feixe emitido e captado por um fotosensor, no qual o funcionamento se assemelha ao infravermelho. Outra tecnologia empregada o controle motorizado de espelhos. O

espelho é controlado para encontrar o melhor ângulo para reflexão do feixe de luz. O cálculo da distância se dá utilizando o ângulo do espelho por triangulação;

- Câmera de vídeo: para implementar a visão para um RMA, e necessário o processamento das imagens via computador. Quanto maior a resolução, maior será o tempo de processamento, encarecendo o projeto. Para o processamento da imagem pelo computador, existem vários métodos, denominados de métodos de reconhecimento padrão. Cada imagem é analisada, comparando os pixels vizinhos, distinguindo o brilho ou cor e, assim, destacar as bordas das imagens. Após isso, o computador compara com informações armazenadas em seu banco de dados para que possa tomar alguma decisão. Há casos do uso de duas câmeras simultaneamente com finalidade de gerar padrões e 3 dimensões, além do cálculo da distância;
- Contato: são sensores mecânicos (fim de curso), com contato NA/NF (normalmente aberto / normalmente fechado) que atuam ao esbarrar em algum objeto. Com grande capacidade de repetibilidade, são de baixo custo e pode ser implementado em todo tipo de robô móvel como um sensor de emergência, caso algum outro sensor não funcione.

Para uma navegação autônoma, a idéia é aplicar vários tipos de sensores, a fim de atender as necessidades do robô quanto à estrutura do ambiente.

2.6. ATUADORES

"São dispositivos que transformam um determinado tipo de energia em um outro tipo diferente, entregam à planta a excitação necessária para seu funcionamento, na forma de outro tipo de energia adequado" (PAZOS, 2002). Diversas são as classificações de atuadores, porém a mais usual é a de que a distinção dos tipos de atuadores se dá segundo a fonte de energia consumida. Por exemplo, para uma planta que se baseia em movimento são necessários atuadores que forneçam energia mecânica, ou para uma planta de um sistema térmico, precisará de atuadores que forneçam energia térmica necessária.

Os atuadores mais utilizados são: hidráulicos, pneumáticos e elétricos.

2.6.1. Hidráulicos

Os atuadores hidráulicos utilizam como fonte de energia a pressão de um fluído denso e viscoso. São capazes de suportar cargas sem a necessidade de acrescentar mais energia, adequados a

movimentos mais lentos e precisos. A desvantagem é a carência de dispositivos auxiliares: sistemas de refrigeração, eliminadores de ar, filtros de partículas, entre outros.

2.6.2. Pneumáticos

Os motores pneumáticos obtêm movimento de rotação mediante a pressão do ar, dividindo-se em dois tipos: aletas rotativas e pistões axiais.

Os modelos de aletas rotativas apresentam rotor excêntrico no qual estão dispostas aletas longitudinais variáveis. Com o ar comprimido e um compartimento com a carcaça e duas aletas, estas giram até que o compartimento tenha maior volume. Os pistões axiais possuem um giro de acordo com as forças exercidas por vários cilindros que se apóiam sobre um plano inclinado em um tambor.

2.6.3. Elétricos

Motores elétricos são dispositivos que transformam energia elétrica em mecânica. A energia mecânica é desenvolvida através da rotação de um eixo que gira com determinada velocidade e torque. Com a rotação do eixo tem-se o movimento do robô ou de alguma de suas partes.

Existem diversos tipos de motores, segundo a construção do robô e o tipo de energia utilizada:

- Motores de Corrente Alternada
- Motores de Corrente Contínua
- Motores de Passo
- Servomotores

2.6.3.1. Motores de Corrente Alternada

Não muito utilizado em robótica devido à dificuldade em efetuar controle de velocidade e torque, além do volume grande deste tipo de motor.

Seu princípio de funcionamento se baseia no campo girante, que surge quanto uma corrente alternada trifásica é aplicada em seus pólos defasados fisicamente em 120°. Assim, em cada

instante, um par de pólos possui um campo de maior intensidade, ocasionando a associação vetorial deste efeito de campo. Há dois tipos de motores de corrente alternada: síncronos e assíncronos.

Os motores síncronos funcionam com velocidade estável, utilizando-se de um induzido que possui campo constante pré-definido e, assim, aumentar a resposta ao processo criado pelo campo girante. Empregado quando o objetivo é ter velocidade estável indiferente da carga, ou quando requer grande potência à torque constante.

Os motores assíncronos, ou motores de indução são robustos, de baixo custo e de grande simplicidade, adequado para vários tipos de maquinas. Funciona com velocidade constante, com pequenas variações de acordo com a carga mecânica aplicada ao seu eixo. Maior facilidade de controle do que motores síncronos, devido ao emprego de conversores de freqüência.

São subdivididos em:

- Síncronos:
  - Polifásico;
  - Monofásico: com imã permanente, histerese, relutância e de passo;
- Assíncronos:
  - Polifásico: “gaiola de esquilo” ou em curto-circuito, rotor enrolado ou bobinado;
  - Monofásico: “gaiola de esquilo” com fase dividida, capacitor de partida, capacitor permanente, pólos sombreados e dois capacitores; rotor enrolado tipo repulsão ou repulsão de partida.

2.6.3.2. Motores de Corrente Contínua

Motores de corrente contínua são os dispositivos atuadores mais utilizados em robótica móvel para prover a locomoção. Neles há um par de terminais nos quais deverão ser ligados a uma bateria e a polaridade desta fonte é que determinará o sentido de rotação. Possui movimento suave e contínuo, além da possibilidade de se obter torque elevado quando utilizados com ma redução mecânica adequada. A desvantagem deste tipo de atuador se dá na dificuldade do controlador reconhecer a posição do eixo e a velocidade de rotação, pois este último depende de sua carga. Ao

definir um motor CC deverá ser levada em consideração à força com que ele poderá girar (torque) com relação à parte mecânica e a velocidade de rotação angular, ω, que equivale ao ângulo que gira o eixo por unidade de tempo. Através destas grandezas, é possível definir a potência necessária para o motor movimentar o robô.

Em um motor de corrente contínua, há duas partes, uma que é fixa, denominada estator, o qual produz o campo magnético, e a parte rotatória chamada de rotor ou armadura que possui uma bobina por onde passa a corrente elétrica, conforme Figura 2. O enrolamento do campo (estator) está dividido em duas partes ligadas em série, cujos produzem fluxo magnético constante em um sentido (HONDA, 2006).

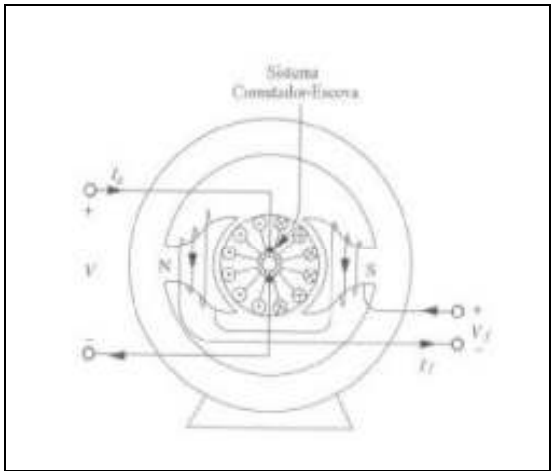


Figura 2: Desenho de um motor de 2 pólos.
Fonte: Adaptado de HONDA (2006)

Na prática, o rotor é composto de várias espiras concêntricas deslocadas em um determinado ângulo entre si e enroladas em um núcleo ferromagnético, sendo os terminais conectados a dois segmentos do comutador. A corrente que circula no rotor é fornecido por uma fonte CC e injetada através das escovas. A função do comutador é inverter periodicamente o sentido da corrente na armadura de tal modo que garanta que o torque tenha sempre o mesmo sentido e impedir com que a armadura fique parada em posição de equilíbrio.

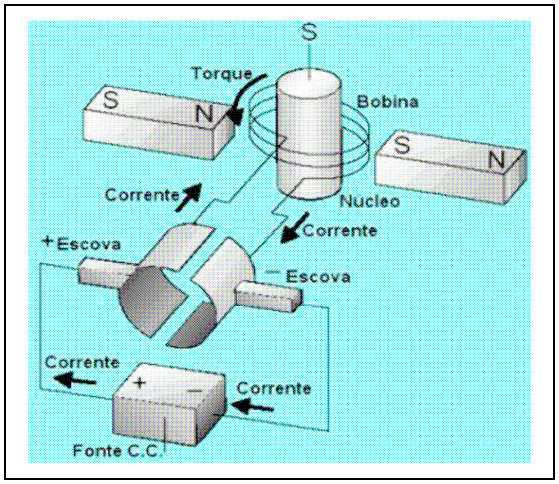


Figura 3: Princípio de funcionamento de um motor CC.
Fonte: Adaptado de HONDA (2006)

O campo gerado pelo estator é produzido através de duas formas: circulação da corrente no bobinado ou com um imã permanente. No primeiro caso, a fonte que alimenta o estator pode ser a mesma que alimenta o rotor, derivando a máquina com características diferentes:

- Motor Série: neste caso, os dois bobinados, do rotor e do estator, estão conectados em série. Com esta ligação, o motor é conhecido como universal, podendo trabalhar tanto co corrente contínua quanto alternada. Com a corrente contínua, o campo magnético gerado pelo estator tem sempre o mesmo sentido. Há a inversão na medida m que o bobinado chaga à vertical. Com a corrente alternada, a mudança de sentido da corrente muda a uma freqüência igual à da tensão de entrada. A característica principal desta conexão é a rotação lenta quando está com carga e o giro rápido quando está a vazio. Ao mesmo tempo em que possui alto torque quando está com a velocidade lenta.
- Motor Paralelo: as bobinas do estator estão em paralelo com as da armadura. A característica principal desta ligação é que a velocidade muda pouca, com ou sem carga. Para ajuste de velocidade é preciso modificar a potência alterando a tensão de entrada.
- Motor Composto: combinação entre motor série com o paralelo, no qual o campo magnético é produzido no estator através de duas bobinas, uma conectada em série e outra em paralelo, ambas com a armadura. Há duas formas de conectar, uma no qual os campos magnéticos se adicionem, outra que se subtraem (mesmo sentido e sentido inverso, respectivamente). No primeiro caso velocidade aumenta com o aumento do

torque. No segundo caso, assume características série ou paralelo, dependendo do tamanho das bobinas.

- Motor com imã permanente: o campo magnético do estator é gerado através de um imã permanente. A vantagem deste tipo de motor CC é o campo magnético constante em um volume reduzido. Existem várias arquiteturas de motores com imã permanente. Uma delas é a que possui o rotor de núcleo de ferro laminado com fendas no qual estão enrolados os condutores. Este tipo possui alta indutância, alta inércia e baixo custo. Outro tipo é o de bobina superficial, em que os condutores da armadura estão colocados em fendas segurados à superfície do rotor, fazendo com que haja uma redução de correntes parasitas induzidas. O movimento é contínuo e suave e possui alta indutância, um custo maior, devido ao imã que deverá ser empregado, e diâmetros maiores. Há também o tipo em que rotor não é ferromagnético, e sim fibra de vidro ou resinas epóxi. Para este tipo, os imãs precisam ser melhores para produzir fluxos magnéticos e o mesmo torque na saída. A inércia é baixa, melhorando a resposta ao controle.

2.6.3.3. Motor de Passo

As saídas destes motores são em forma de incrementos angulares, controlados por impulsos elétricos na alimentação. A vantagem deste tipo de motor é que ele pode ser utilizado em malha aberta, pois o driver de controle conhece a posição exata do eixo. A desvantagem é que não possuem grandes torques, por isso são empregados em serviços leves, para que não percam os passos. A vantagem é o torque de retenção, o que não ocorre em motores CC.

O motor de passo é constituído de um rotor com dois conjuntos de pólos separados, sendo que cada conjunto possui pólos que parecem como dentes de engrenagem, e um estator com vários pólos eletromagnéticos. A armadura possui números ímpares de pólos e o estator números pares, de forma que os pólos não poderão estar alinhados todos de uma vez, propiciando o movimento em passos. Convencionalmente, os ângulos estão entre 1,8° e 30°.

Há duas seqüências utilizadas: meio passo e passo completo. A de meio passo magnetiza uma bobina a cada passo. O passo completo magnetiza duas bobinas (ou conjunto de bobinas) por passo. A velocidade é ajustada através do clock de entrada. Quanto maior a velocidade, menor o torque.

2.6.3.4. Servo-motor

Motor, geralmente de corrente contínua, que utiliza sensores de posição ou velocidade permitindo que se possa controlar estas grandezas físicas. Para isto, há controladores que tornam este tipo de motor mais preciso. O controle faz uso de sinais de referência vindo dos sensores para conhecer a real posição ou aumentar a velocidade, realimentando com estes sinais o servo. Em muitos casos, os servo-motores exigem sinais modulados por largura de pulso (PWM) como referência.

2.7. CONTROLE

Em um RMA, não basta apenas ter o motor para movimentar o robô ou alguma de suas partes. É preciso que haja um controle para fornecer ao dispositivo movimento coordenado e preciso. Como a alimentação, geralmente, é em corrente contínua, os tipos de controles mais utilizados são os que convertem CC para CC. São denominados choppers, ou conversores CC-CC. Eles se assemelham a transformadores em corrente alternada com relação de espiras variável. É possível variar a tensão CC de saída e fornecer um controle de velocidade de acordo com sinais de referências enviadas pelos sensores que atuarão no robô. Há duas formas de trabalho (RASHID, 1999):

- Operação em freqüência constante: consiste em manter a freqüência de operação constante e varia o tempo de condução. Este tipo de controle é denominado Modulação por Largura de Pulsos, ou PWM (*Pulse Width Modulation*);
- Operação em freqüência variável: a freqüência, neste caso, é quem varia, enquanto o tempo de condução, t_1, e de bloqueio, t_2, são mantidos. Denominado Modulação em Freqüência (*Frequency Modulation*), a freqüência precisa ser variada amplamente para se obter uma faixa de tensão de saída completa, gerando harmônicas e encarecendo o projeto por exigir diversos filtros.

O circuito chopper pode ter dois princípios de funcionamento, dependendo do seu circuito: abaixador e elevador. No circuito abaixador há dois modos: primeiro a corrente flui da fonte para a carga, depois, quando é desligada, a corrente continua a fluir pelo diodo de comutação (Figura 4.a). No circuito elevador, quando a chave estiver fechada, a corrente no indutor irá armazenar energia. Se a chave for aberta, o indutor irá transferir a energia à caga através do diodo (Figura 4.b).

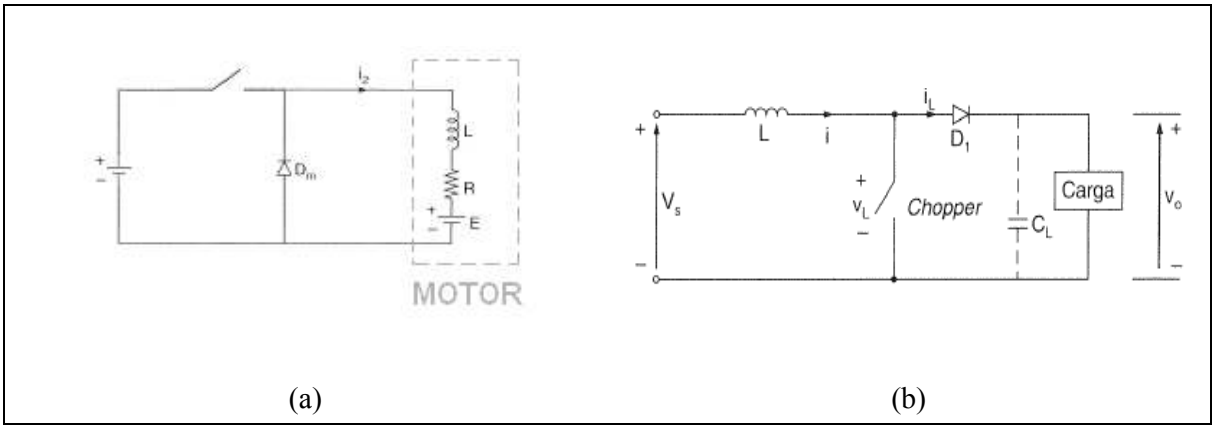
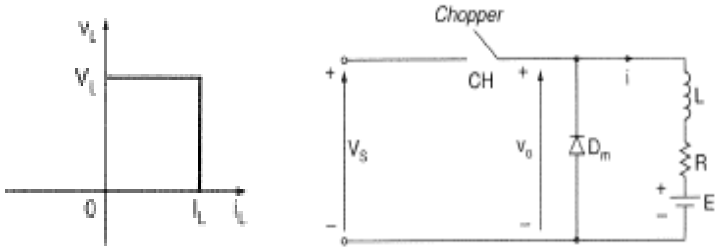


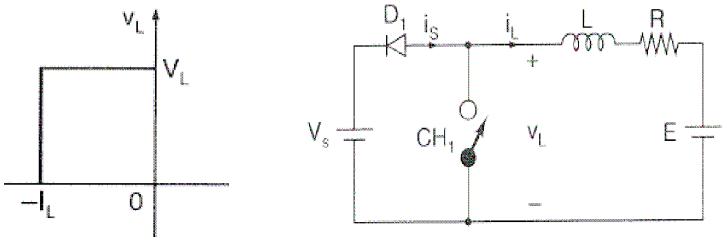
Figura 4. Princípio de funcionamento: (a) circuito abaixador; (b) circuito elevador.
Fonte: Adaptado de RASHID (1999)

Os conversores CC-CC (Figura 5) são classificados em cinco tipos: Classe A, B,C, D e E.

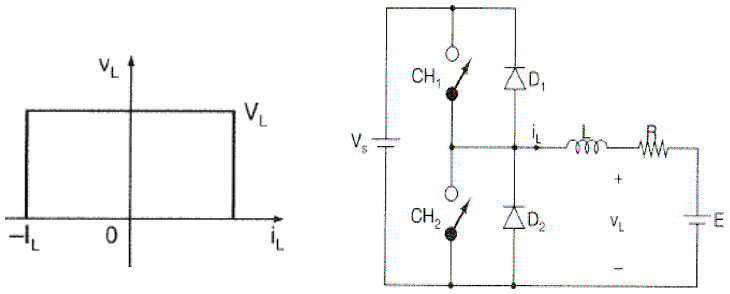
- Classe A: o sentido da corrente é para dentro da carga, sendo assim, a ela e a tensão são positivas. Circuito conhecido como *chopper* de um quadrante, operando como um retificador;
- Classe B: a corrente flui para fora da carga, por isso, a tensão é positiva e a corrente negativa. Também é conhecido como *chopper* de um quadrante, ou que o circuito opera como um inversor;
- Classe C: a corrente na carga é tanto positiva quanto negativa, porém a tensão será sempre positiva. Denominada de *chopper* de dois quadrantes, e ora opera como chopper classe A, ora classe B, dependendo da ligação das chaves no circuito;
- Classe D: a corrente é sempre positiva, embora a tensão seja positiva e negativa, operando tanto como um retificador ou inversor, dependendo da ligação do circuito;
- Classe E: na carga, a corrente e tensão são tanto positivas quanto negativas, denominando esta classe como chopper de quatro quadrantes. É a combinação de dois *choppers* classe C e opera como um inversor monofásico em ponte (ponte H).


Chopper
CH
i
L
R
E
V_S
v_0
D_m
V_L
I_L
i_L
0

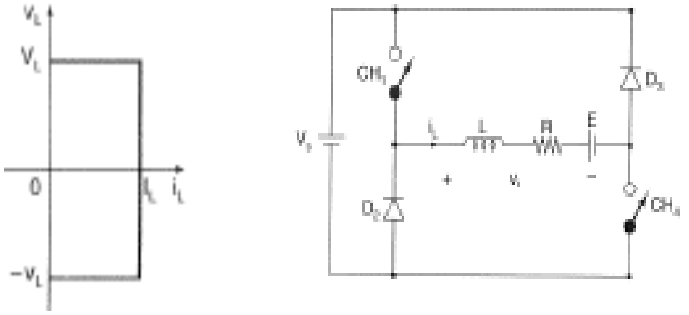
(a)


D_1
i_S
i_L
L
R
V_S
CH_1
v_L
E
V_L
$-I_L$
0

(b)


CH_1
D_1
V_S
i_L
L
R
CH_2
D_2
v_L
E
V_L
$-I_L$
0
i_L

(c)


V_L
0
$-V_L$
i_L
CH_1
D_2
D_1
CH_2
L
R
E

(d)

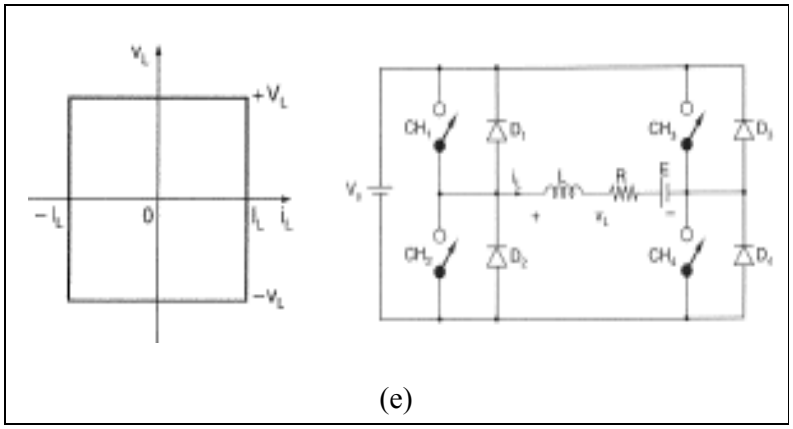


Figura 5: Conversores CC-CC: (a) Classe A; (b) Classe B; (c) Classe C; (d) Classe D; (e) Classe E.
Fonte: Adaptado de RASHID (1999)

Para um RMA, o controle comumente usado é baseado no princípio de funcionamento do conversor CC-CC classe E, acionamento em ponte.

Na operação em sentido direto, a tensão de armadura, eletromotriz, a corrente, o torque e a velocidade do motor estão positivos, ou seja, primeiro quadrante. Quando ocorre uma frenagem no sentido direto, isto é, no segundo quadrante, a tensão no rotor e a eletromotriz estão positivas, entretanto para o torque poder ser negativo, a corrente deverá estar negativa, invertendo o fluxo de energia. No terceiro quadrante, o motor está operando no sentido inverso, assim, todas as grandezas estão negativas. Para ocorrer à frenagem regenerativa, no quarto quadrante, a tensão e a eletromotriz mantêm negativas, porém o torque e a corrente passam a ser positivas, completando o circuito.

2.7.1. Reguladores Chaveados

Os circuitos choppers podem assumir a função de reguladores de tensão CC a fim de converter uma tensão desregulada em uma tensão CC regulada, no qual é obtida através da modulação por largura de pulso durante uma freqüência fixa. Elas são subdivididas em 4 topologias básicas e algumas isoladas. As principais são:

1. *Buck* (abaixador);
2. *Boost* (elevador);
3. *Buck-Boost* (inversor);
4. *Cúk*.

2.7.1.1. Regulador ***Buck***

A tensão média de saída é menor do que a entrada. No circuito *Buck*, geralmente se usa transistores para chavear o sinal. Quando o transistor estiver polarizado, a corrente flui crescente pelo indutor, capacitor e resistor. Em um segundo momento, quando o transistor estiver aberto (desligado), a corrente mantém-se pelo diodo de comutação, indutor, capacitor e carga. A corrente do indutor cai até que o transistor comute novamente. O regulador *Buck* é simples, de eficiência elevada e unidirecional, ou seja, a corrente flui em apenas um sentido.

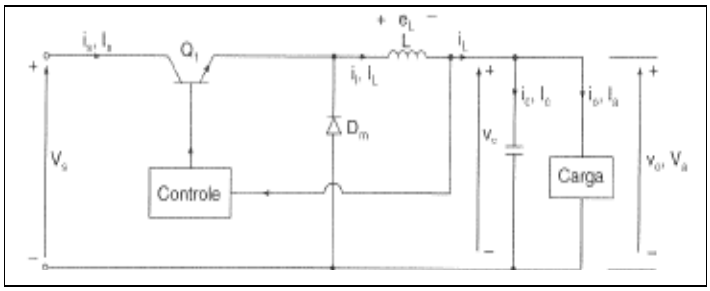


Figura 6:Circuito Regulador Abaixador.
Fonte: Adaptado de RASHID (1999)

2.7.1.2. Regulador ***Boost***

A tensão média de saída é maior do que a entrada. Neste circuito, o transistor, em geral, é de potência. Quando o transistor está polarizado, a corrente flui por ele, e pelo indutor. Ao ser desligado, a corrente que estava fluindo passa a percorrer pelo indutor, capacitor, carga e pelo diodo de comutação. A energia armazenada no indutor passa para a carga. O regulador *Boost* tem elevado sinal na saída e eficiência. No entanto, tem alto pico de corrente fluindo pelo transistor e a tensão de saída pode sofre muita variação exigindo filtros capacitivos e indutivos melhores que o regulador *Buck*.

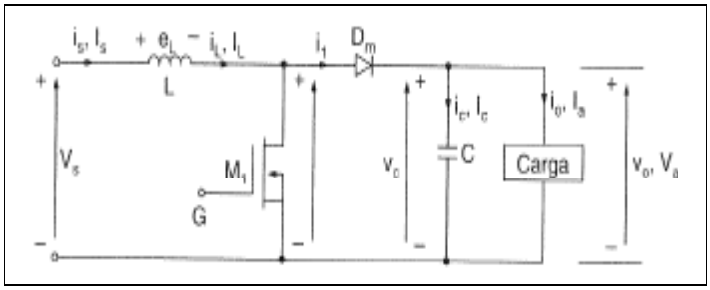


Figura 7: Circuito Regulador Elevador.
Fonte: Adaptado de RASHID (1999)

2.7.1.3. Regulador ***Buck-Boost***

Oferece sinal de saída menor e maior do que a entrada, além da polaridade de saída ser inversa com relação ao sinal de entrada, denominando este tipo de regulador como inversor. Ele opera da seguinte forma: quando o transistor conduz o diodo está reversamente polarizado e a corrente cresce através do indutor e do transistor. Quando o transistor é desligado, a corrente do indutor flui por ele, pelo capacitor e pelo diodo. A energia que estava no indutor segue para carga. Este circuito oferece sinal inverso da entrada sem uso de transformador, a corrente de entrada é descontínua e há uma corrente elevada fluindo pelo transistor.

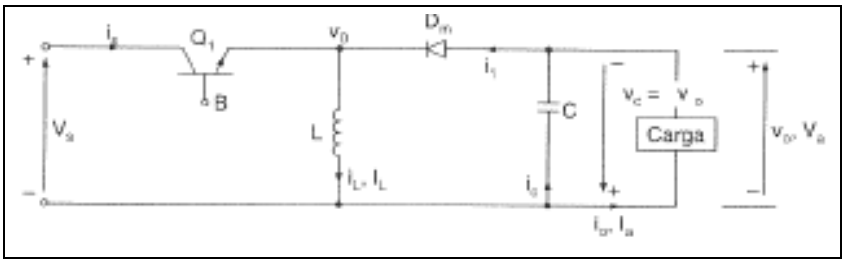


Figura 8: Circuito Regulador Inversor.

Fonte: Adaptado de RASHID (1999)

2.7.1.4. Regulador ***Cúk***

Este circuito é similar a *buck-boost*. Quando a tensão de entrada está ligada, mas o transistor não, o diodo de comutação está diretamente polarizado e o primeiro capacitor é energizado pelo indutor L1, o diodo e pela alimentação de entrada. Com o transistor comutado, corrente no indutor cresce. A tensão no primeiro capacitor polariza o diodo reversamente desligando-o, e descarrega a energia na malha formada pelo primeiro e segundo capacitor (C1 e C2), carga e o segundo indutor (L2). O diodo e o transistor fornecem o chaveamento síncrono e o capacitor C1 faz a transferência de energia da fonte à carga.

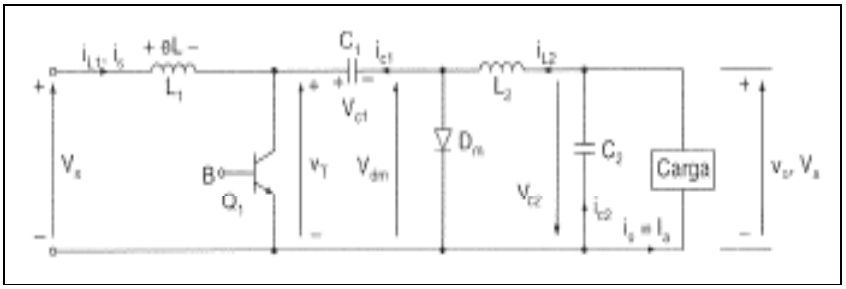


Figura 9: Circuito Regulador *Cúk*.

Fonte: Adaptado de RASHID (1999)

2.7.1.5. Regulador em Ponte

Esta é uma topologia isolada, comumente utilizada em controle de motor CC para robôs móveis. Consiste em um circuito chopper de 4 quadrantes, classe E, (vide Figura 5.e), no qual as chaves para a comutação são transistores (figura). A vantagem do uso deste circuito é a possibilidade de conduzir diretamente e inversamente, produzindo o efeito de rotação direta e inversa do motor (carga).

Quando os transistores Q1 e Q4 (Figura 10) estiverem polarizados, a corrente irá fluir pela carga em um sentido. Ao serem comutados os transistores Q2 e Q3, a corrente flui inversamente. Os diodos de comutação, em paralelo com os transistores, têm a finalidade de drenar a corrente que poderia forçar a passagem através dos transistores.

Para o controle de velocidade e rotação do motor CC, este circuito é alimentado por um sinal modulado, com freqüência constante, variando apenas o tempo do pulso, conhecido como Modulação por Largura de Pulso (PWM).

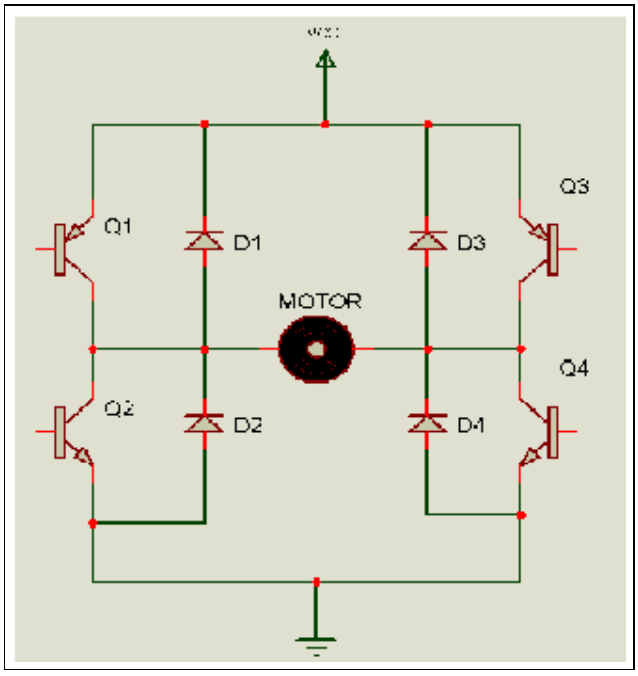


Figura 10: Circuito acionador em Ponte.

2.7.2. Modulação por Largura de Pulso (PWM)

Para compreender o funcionamento deste tipo de modulação, pode-se partir da aplicação em um circuito formado por chave de ação rápida e uma carga a ser controlada.

Quando o interruptor estiver desligado, não haverá corrente na carga e a potência será nula. Ao ser ligada, a carga receberá a corrente total da fonte e a máxima potência. Sendo esta a situação em que está com 100% de potência aplicada, para obter-se 50%, é preciso aumentar a velocidade de chaveamento.

Assim, mantendo a chave 50% aberta (t_1) e 50% fechada (t_2), em um único ciclo (T), tem-se o controle da potência aplicada na carga, como mostra a Figura 11. Assim, a potência média aplicada à carga é de 50%.

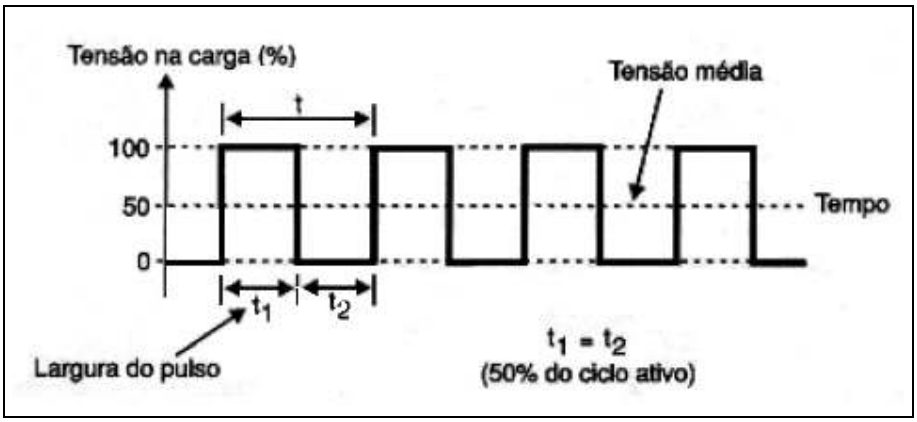


Figura 11: Sinal com modulação PWM.
Fonte: Adaptado RASHID (1999)

Portanto, o controle da chave pode definir a largura de pulso pelo tempo em que ela fica na condição fechada e o intervalo entre pulsos em que fica em aberto. Estes dois tempos definem o período e, conseqüentemente, a freqüência de controle.

A relação entre o tempo em tem-se o pulso e o tempo de um período completo de operação da chave define o ciclo ativo, conforme Figura 12.

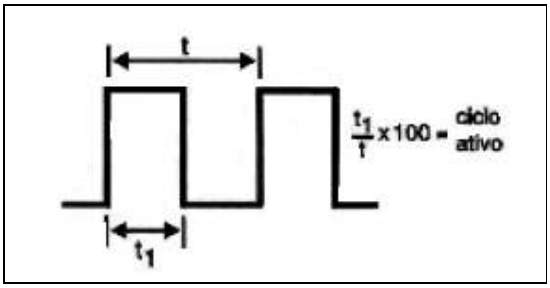


Figura 12: Definição de ciclo ativo.
Fonte: Adaptado RASHID (1999)

É possível variar a largura de pulso e o intervalo de tempo a fim de obter ciclos ativos diferentes, controlando a potência média fornecida à carga, assim como mostra a Figura 13.

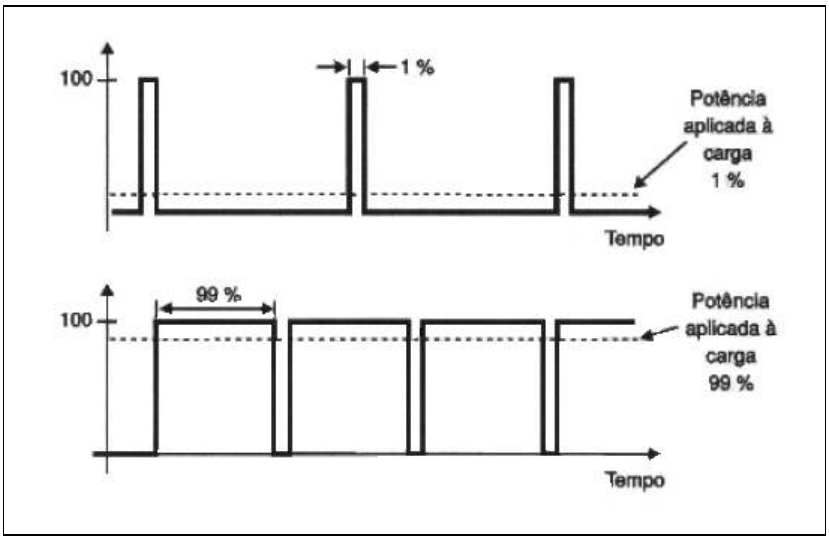


Figura 13: Controle da potência pelo ciclo ativo.
Fonte: Adaptado RASHID (1999)

Há diversas vantagens em utilizar os circuitos de controle por PWM, sendo uma delas a da dissipação nula, pois quando a chave estiver em aberto, não há corrente e, teoricamente, quando estiver fechado, o circuito apresenta uma resistência nula, portanto a tensão nula, bem como a potência. Na teoria o circuito de controle PWM não dissipa potência, contudo, na prática isto não ocorre devido à incapacidade dos dispositivos, utilizado nos circuitos de controle, em abrir e fechar em um tempo muito pequeno. Durante a mudança de estado, há um intervalo de tempo no qual surge uma pequena resistência crescente e decrescente, como mostra a curva de comutação na Figura 14.

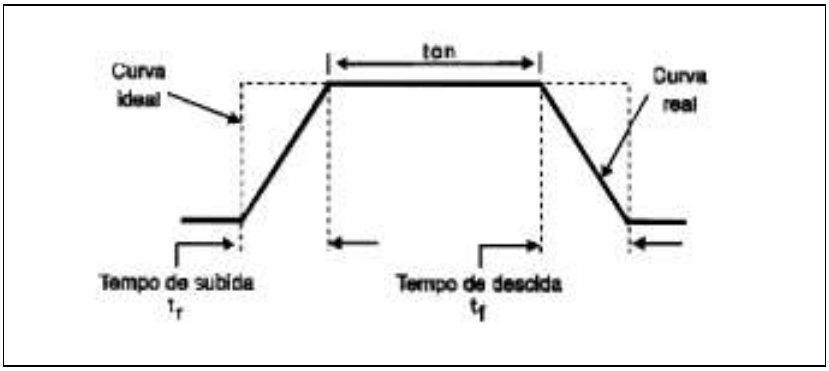


Figura 14: Curva de comutação PWM.
Fonte: Adaptado RASHID (1999)

Os tempos de subida (t_r) e de descida (t_f) geram quantidade considerável de calor. Todavia, o controle por PWM gera menos dissipação de potência do que circuito de controle linear.

Outro fator a ser atentado quando projetado um circuito de controle por PWM são os transistores, nos quais se comportarão como chaves. Tanto transistores de efeito de campo quanto bipolares não se comportam como resistências nulas, assim, há quedas de tensão quando saturados.

2.8. MICROCONTROLADORES

2.8.1. Microprocessador

"Os dispositivos microcontrolados são dispositivos que se originaram dos microprocessadores" (ZANCO, 2006). "O microprocessador, conhecido também como Unidade Central de Processamento, ou CPU (*Central Processing Unit*), trata-se de um dispositivo semicondutor integrado (CI), constituído de milhões de transistores, com os quais implementam variados circuitos lógicos. É através dele que são executados e armazenados dados e instruções" (SOUZA, 2003).

Basicamente, divide-se em três partes: Unidade Lógica Aritmética (ULA), Registradores Internos (RI) e Blocos de Temporização e Controle (UC), onde:

- ULA: responsável pela execução das instruções correspondentes às operações lógicas e aritméticas. Outra característica importante é que, ao executar uma instrução, ela define a condição numérica do resultado por meio de bits sinalizadores, denominados *flags* sinalizadores, possibilitando ao sistema um gerenciamento de determinada situação;
- RI: compostos por *m* registradores de *n* bits interligados em paralelo entre si. Constituídos de flip-flops, realizam operações de leitura e escrita das informações captadas pelos periféricos. São memórias voláteis, portanto necessitam manter energizadas. Vale ressaltar, os RI´s não são armazenadores de dados, mas de dados temporários com os quais utiliza para executar os processamentos dos mesmos;
- UC: controla o fluxo de informações para as unidades de memória e de entrada e saída, por meio da definição dos sinais de leitura, escrita, liberação do barramento de dados, entre outros.

Para a comunicação entre as interfaces existem os três tipos de barramentos (Figura 15):

- Barramento de Endereços: utilizado pela CPU para definir os endereços das posições de memória dos programas, pelos quais busca as instruções a serem executadas, defini os endereços da memória de dados ou informações de entrada e saída. Este barramento é unidirecional;
- Barramento de Dados: responsável pelo tráfego de informações vindos da memória ou dispositivos *I/O's* (entrada/saída).O barramento de dados é bidirecional;
- Barramento de Temporização e Controle: defini os sinais de temporização e controle a fim de gerenciar o tempo e a direção do fluxo de informações nas operações de leitura e escrita. É m barramento unidirecional.

Outros componentes básicos são: circuito de reset, oscilador (defini o *clock* para sincronismo do sistema) e interrupções (sincroniza com eventos externos ou internos).

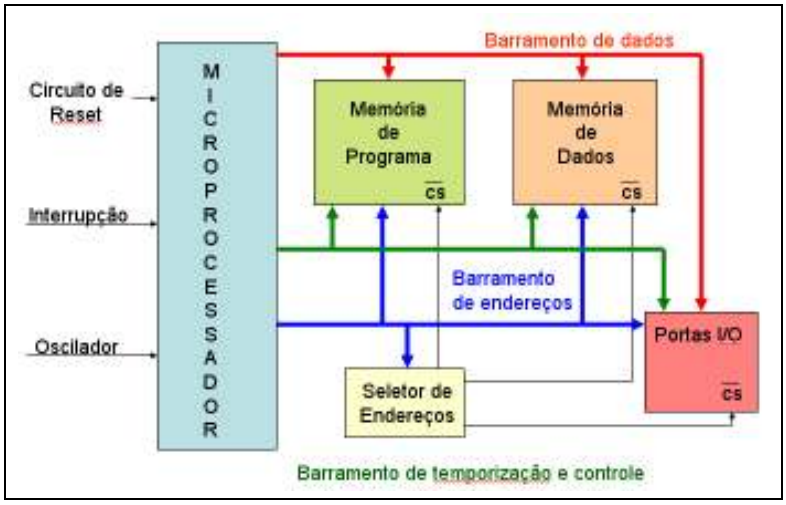


Figura 15: Diagrama de blocos de um sistema genérico de um microprocessador
Fonte: Adaptado de ZANCO (2006).

Tanto microprocessadores quanto os microcontroladores, possuem estruturas próprias de acordo com a tecnologia (ZANCO, 2006). Existem dois tipos de arquitetura:

- Von Neumann: considerada clássica entre os microprocessadores caracteriza-se pela possibilidade de armazenar, no mesmo espaço de memória programas e dados, manipulando ambos. Para melhorar o fluxo, utiliza-se de um recurso denominado *pipeline*. São três estágios: em um estado normal de operação, enquanto se executa uma instrução, outra está sendo decodificada e, uma terceira sendo buscada na memória;

- Harvard: com arquitetura melhorada, separou-se o barramento de dados de programa do barramento de dados. Faz uso do *pipeline*, porém de cinco fases: IF (busca de instrução na memória), ID (leitura dos registradores e decodificação da instrução), EX (execução da instrução), MEM (acesso a operando à memória) e WB (escrita de um resultado em um registrador);

Há dois tipos de conjuntos de instruções:

- CISC (*Complex Instruction Set Computer*): exige grande número de instruções por ciclo, número de modos de endereçamento elevado, cuja implementação somente é possível através de tecnologia de microprogramação;
- RISC (*Reduced Intruction Set Computer*): o número de instruções e modos de endereçamento são reduzidos. A arquitetura é apropriada para um controlador totalmente *hardwired*.

Na Tabela 1 abaixo, há uma comparação das duas tecnologias:

Tabela 1: Comparativo entre conjuntos de instruções RISC e CISC

RISC	**CISC**
Instruções simples durante 1 ciclo	Instruções complexas durante vários ciclos
Apenas LOAD/STORE referencia a memória	Qualquer instrução pode referenciar a memória
Alto uso de *Pipeline*	Baixo uso de *Pipeline*
Instruções executadas pelo *hardware*	Instruções interpretadas pelo microprograma
Instruções com formato fixo	Instruções de vários formatos
Poucas instruções e modos de endereçamento	Muitas instruções e modos de endereçamento
Múltiplos conjuntos de registradores	Conjunto único de registradores
A complexidade está no compilador	A complexidade está no microprograma

Fonte: Adaptado de SOUZA (2003).

A vantagem do tipo CISC é trabalhar com macros, permitindo o uso de uma única instrução, ao invés de várias instruções simples, ideal para microprocessadores, que utilizam grande conjunto de instruções e circuitos complexos. Porém, como os microcontroladores utilizam-se de poucas instruções, os circuitos são menores e, com conseqüência, menor consumo de energia e maior rapidez, a tendência foi de utilizara arquitetura Harvard com a estrutura RISC. O microcontrolador permite o *pipeline* das instruções e estas são simétricas, ou seja, podem operar em todos os

registradores ou em qualquer modo de endereçamento. Não há combinações especiais de instruções, exceções ou restrições, sendo assim, facilita ao programador, pois precisa lembrar-se de um número reduzido de instruções.

2.8.2. Microcontrolador (baseado no PIC16F877A)

O microcontrolador é um dispositivo semicondutor em forma de circuito integrado, cujo integra todas as partes de um microcomputador: microprocessador (CPU), memórias voláteis (RAM, SRAM, DRAM, Flash), memórias não-voláteis (ROM, PROM, EPROM, EEPROM), e portas entrada e saída (comunicação paralela, serial, conversor A/D e D/A).

Existem vários fabricantes de microcontroladores, cada um com sua especificidade. Portanto este estudo será dirigido ao microcontrolador da série PIC16F877x, fabricado pela empresa Microchip®.

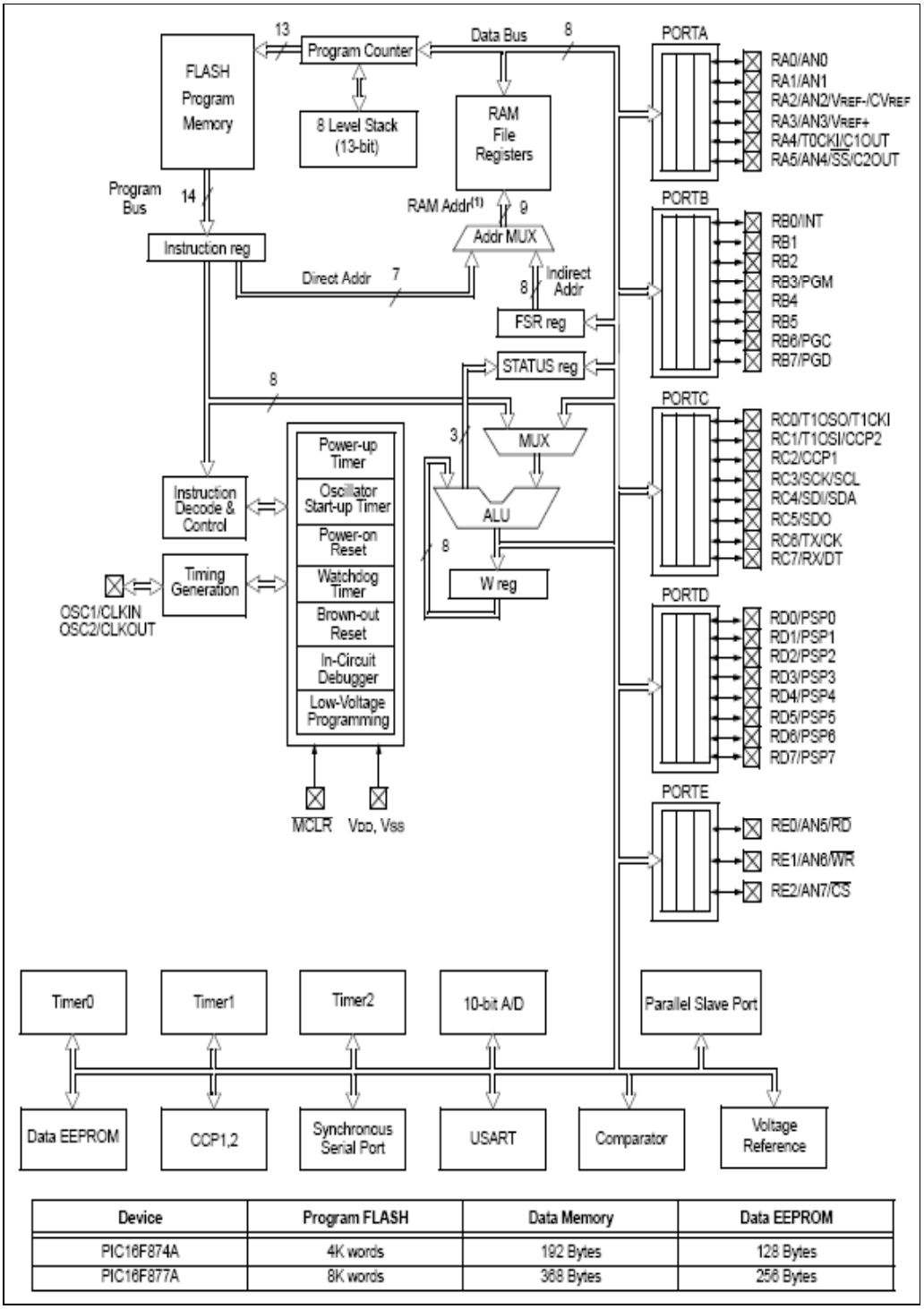
Os microcontroladores da família PIC possuem um barramento de dados de 8 bits e o de instruções pode ser de 12, 14 ou 16 bits. O PIC16F877A possui 14 bits para cada instrução, totalizando em um conjunto de 35 instruções por ciclo de máquina. As características principais deste microcontrolador são dadas na Tabela 2 abaixo:

Tabela 2: Características Principais do PIC 16F877A.

PIC16F877A	
Freqüência de operação	DC-20MHz
RESETS (e *delays*)	POR, BOR, (PWRT, OST)
Memória *flash* (14 bits/instrução)	8k
Memória de Dados (bytes)	368
Memória EEPROM (bytes)	256
Interrupções	15
Porta *I/O*	Port A, B, C, D, E
Timers	3
Captura/Comparação/PWM	2
Comunicação Serial	MSSP, USART
Comunicação Paralela	PSP
Conversor A/D (10bits)	8 canais de entrada
Comparador Analógico	2
Set de Instruções	35
Encapsulamento	40 pinos PDIP, 44 pinos PLCC/QFP

Fonte: Adaptado do *datasheet PIC16F87XA*

Na Figura 16 está o bloco da estrutura interna do PIC 16F877A.



Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Figura 16: Diagrama em blocos da estrutura interna do PIC16F877A

Fonte: Adaptado do *datasheet PIC16F87XA*.

2.8.2.1. Pinos e Registradores

Os pinos do microcontrolador e mapa de registros do PIC estão no Anexo I.

É de suma importância realizar a configuração, durante a programação, das portas de entradas e saídas e dos bits de configurações:

PortA e registrador TRISA

Porta bidirecional de 6 bits de largura. A direção dos pinos é determinada pelo registrador TRISA. Quando validado um bit de TRISA, faz com que o pino correspondente da porta funcione como entrada. Resetando este bit, a porta atuará como saída.

A porta RA4 é multiplexada com a entrada de clock do módulo *Timer* 0. O demais pinos são multiplexados com entradas analógicas e tensão de referência Vref. Para executar a leitura analógica, é preciso configurar os registradores ADCON1 e/ou ADCON2.

Tabela 3: Funções do pino A.

Name	Bit#	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input
RA1/AN1	bit1	TTL	Input/output or analog input
RA2/AN2	bit2	TTL	Input/output or analog input
RA3/AN3/VREF	bit3	TTL	Input/output or analog input or VREF
RA4/T0CKI	bit4	ST	Input/output or external clock input for Timer0 Output is open drain type
RA5/$\overline{SS}$/AN4	bit5	TTL	Input/output or slave select input for synchronous serial port or analog input

Legend: TTL = TTL input, ST = Schmitt Trigger input

Fonte: Adaptado do *datasheet PIC16F87XA*

Tabela 4:Resumo dos registradores da porta A.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

Fonte: Adaptado do *datasheet PIC16F87XA*.

PortB e registrador TRISB

Porta bidirecional, possui 8 bits de largura. A direção é dada pelo registrador TRISB, aplicando a mesma lógica de habilitação e desabilitação que o registrador TRISA.

O pino RB0 pode ser configurado como interruptor a partir do registrador *OPTION*. Os pinos RB3, RB6 e RB7 são multiplexado com a função de programação em baixa tensão. Os pinos da porta B podem ser configurados para funcionar como resistores *pull-up*. Quando configurados como entrada, do RB4 ao RB7, há a possibilidade de gerar interrupções, sendo que os valores dos pinos são comparados com a o valor da última leitura, salvando-os no *latch*. Então, é enviada a diferença para uma porta lógica OU gerando as interrupções.

Tabela 5: Funções do pino B.

Name	Bit#	Buffer	Function
RB0/INT	bit0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM	bit3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB6/PGC	bit6	TTL/ST(2)	Input/output pin (with interrupt on change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit7	TTL/ST(2)	Input/output pin (with interrupt on change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Fonte: Adaptado do *datasheet PIC16F87XA*

Tabela 6: Resumo dos registradores da porta B.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

Fonte: Adaptado do *datasheet PIC16F87XA*.

PortC e registrador TRISC

Assim como o PortB, a porta C possui 8 bits de largura, sendo esta também bidirecional. Para determinar se os pinos são entradas ou saídas, é utilizado o registrador TRISC.

Caso as funções de periféricos estejam habilitada, haverá sobreposições sobre as configurações. Por exemplo, na porta RC2, ter-se-á a porta CCP1 (recurso de envio de sinal PWM).

Tabela 7: Funções do pino C.

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input
RC1/T1OSI/CCP2	bit1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/PWM1 output
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output
RC6/TX/CK	bit6	ST	Input/output port pin or USART Asynchronous Transmit or Synchro-nous Clock
RC7/RX/DT	bit7	ST	Input/output port pin or USART Asynchronous Receive or Synchro-nous Data

Legend: ST = Schmitt Trigger input

Fonte: Adaptado do *datasheet PIC16F87XA*

Tabela 8: Resumo dos registradores da porta C.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged.

Fonte: Adaptado do *datasheet PIC16F87XA*.

PortD e registrador TRISD

Bidirecional com 8 bits de largura, as características de entrada ou saída são determinadas pelo registrador TRISD. Esta porta pode ser configurada como uma porta paralela de 8 bits, cujo bit de controle está no registrador TRISE (PSPMODE).

Tabela 9: Funções do pino D.

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL(1)	Input/output port pin or parallel slave port bit0
RD1/PSP1	bit1	ST/TTL(1)	Input/output port pin or parallel slave port bit1
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or parallel slave port bit2
RD3/PSP3	bit3	ST/TTL(1)	Input/output port pin or parallel slave port bit3
RD4/PSP4	bit4	ST/TTL(1)	Input/output port pin or parallel slave port bit4
RD5/PSP5	bit5	ST/TTL(1)	Input/output port pin or parallel slave port bit5
RD6/PSP6	bit6	ST/TTL(1)	Input/output port pin or parallel slave port bit6
RD7/PSP7	bit7	ST/TTL(1)	Input/output port pin or parallel slave port bit7

Legend: ST = Schmitt Trigger input TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffer when in Parallel Slave Port Mode.

Fonte: Adaptado do *datasheet PIC16F87XA*

Tabela 10: Resumo dos registradores da porta D.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by PORTD.

Fonte: Adaptado do *datasheet PIC16F87XA*

PortE e registrador TRISE

Esta porta possui três pinos, bidirecionais, torna-se controle da porta paralela (PortD) quando o bit4 do TRISE estiver habilitado (PSPMODE). Para isto, os bits 2 a 0 deverão estar configurados como E/S digitais.

Também são multiplexados com entradas analógicas adequando a configuração do registrador ADCON1.

Tabela 11: Bits do registrador TRISE.

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	bit2	bit1	bit0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n= Value at POR reset

Parallel Slave Port Status/Control Bits

bit 7 : **IBF:** Input Buffer Full Status bit
1 = A word has been received and is waiting to be read by the CPU
0 = No word has been received

bit 6: **OBF**: Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5: **IBOV**: Input Buffer Overflow Detect bit (in microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4: **PSPMODE**: Parallel Slave Port Mode Select bit
1 = Parallel slave port mode
0 = General purpose I/O mode

bit 3: **Unimplemented**: Read as '0'

PORTE Data Direction Bits

bit 2: **Bit2**: Direction Control bit for pin RE2/$\overline{CS}$/AN7
1 = Input
0 = Output

bit 1: **Bit1**: Direction Control bit for pin RE1/$\overline{WR}$/AN6
1 = Input
0 = Output

bit 0: **Bit0**: Direction Control bit for pin RE0/$\overline{RD}$/AN5
1 = Input
0 = Output

Fonte: Adaptado do *datasheet PIC16F87XA*

Tabela 12: Funções do pino E.

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit0	ST/TTL(1)	Input/output port pin or read control input in parallel slave port mode or analog input: $\overline{RD}$ 1 = Not a read operation 0 = Read operation. Reads PORTD register (if chip selected)
RE1/$\overline{WR}$/AN6	bit1	ST/TTL(1)	Input/output port pin or write control input in parallel slave port mode or analog input: $\overline{WR}$ 1 = Not a write operation 0 = Write operation. Writes PORTD register (if chip selected)
RE2/$\overline{CS}$/AN7	bit2	ST/TTL(1)	Input/output port pin or chip select control input in parallel slave port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input TTL = TTL input
Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port Mode.

Fonte: Adaptado do *datasheet PIC16F87XA*

Tabela 13: Resumo dos registradores da porta E.

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by PORTE.

Fonte: Adaptado do *datasheet PIC16F87XA*.

Além das configurações básicas, o PIC possui outras configurações essenciais para serem executadas durante a programação:

- *Instruction Register*: registrador de instruções do programa a ser executado;
- OSC1/OSC2: Entradas e saídas do cristal externo, no qual irá determinar a freqüência de trabalho. Antes de gravar o programa, deve-se definir o valor de freqüência que irá trabalhar o PIC;
- *Watchdog Timer*: registro responsável pelo não travamento do microcontrolador. Basicamente, é um contador no qual o programa determina um valor limite e, durante o desenvolvimento do programa, aloca em locais específicos. Quando o registro chegar ao limite, traz o programa para a primeira linha (0000), fazendo com que o programa não trave, mas fique em *loop* infinito;

- *Power-on Reset (POR)*: pulso gerado internamente ao chip quando ocorre uma subida da tensão de alimentação detectada (entre 1,2V a 1,7V);
- *Power-up Time (PWRT)*: fornece um tempo de saída fixo de 72ms na energização após POR e enquanto estiver ativo o PIC é mantido em reset. O atraso gerado permite á fonte de alimentação do PIC atingir nível aceitável de tensão para operação;
- *Oscillator Start-up Timer (OST)*: fornece 1024 ciclos de oscilação de atraso, ao fim do atraso do PWRT, garantido com que o circuito de oscilação a cristal atinja a estabilidade.
- *Brown-out Reset (BOR)*: o bit de configuração *BODEN* habilita o circuito BOR. Se a tensão de alimentação estiver abaixo de VBOR por um tempo TBOR (parâmetros definidos na programação), ele reinicia-se o PIC;
- *Power Control/Status Register (PCON)*: executa o controle de *resets*.
- *Timers*: registros que podem ser utilizados tanto como temporizador, como contador de pulsos;
- Pilha: de 13 bits e 8 níveis de profundidade, cujo corresponde a 8 locais de memória com 13 bits de largura. Utilizado quando ocorre um desvio do programa para um subprograma, então salvo em uma pilha, para retirar da pilha, utiliza-se à instrução RETURN, RETLW ou RETFIE;
- *Master Clear*: reset durante operação normal. Durante a gravação, é inserido a tensão 13,5V e a alimentação do PIC com 4,5V, garantindo com que na interrupção da gravação nem falhas;
- Programação *In-Circuit*.

3. METODOLOGIA

Conforme visto, com o advento da eletrônica e automação, os robôs se tornaram cada vez mais comuns no cotidiano dos seres humanos. Este projeto visa em um protótipo de robô móvel autônomo seguidor de linha, habilitado por sensores infravermelhos. Dispositivos mecânicos auxiliam como proteção, caso o robô esbarre em algum objeto em seu caminho.

Mecanicamente, possui dois motores, cada um conectado a duas engrenagens, que aumentam o torque do mesmo. Duas rodas emborrachadas na parte traseira e uma roda livre na dianteira. A fixação do conjunto será feita na própria placa de circuito impresso.

A programação em linguagem C será compilada e gravada no microcontrolador PIC 16F877A, com o auxílio do software MicroC®e o PICKit. A simulação do circuito e o desenho para confecção da placa de circuito impresso fora feito no software Proteus®.

A primeira instância, serão apresentados os circuitos dos sensores infravermelhos construídos e o driver de controle dos motores. Após, será demonstrado o circuito em si e a programação.

3.1. SENSORES INFRAVERMELHOS

Os sensores infravermelhos foram construídos utilizando alguns dos componentes contidos na Tabela 16. Como o sinal obtido continha muita oscilação, fora acrescentado um capacitor na saída, conforme o circuito montado na Figura 17.

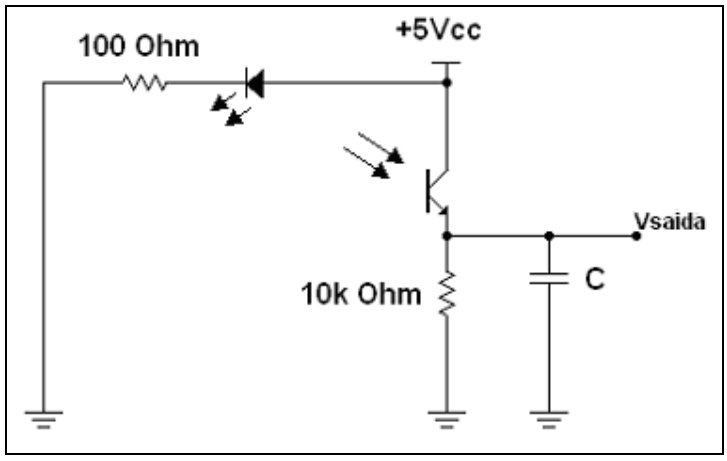


Figura 17: Circuito do sensor infravermelho.

Com o auxílio de uma folha em branco, alimentando o circuito com 5V, fora relacionado à distância com o valor do sinal de saída do sensor. Assim, possibilitou-se determinar uma distância X em que os sensores ficarão do solo e conhecer o valor que fará com que o microcontrolador adote uma instrução com base neste sinal.

Como o robô terá características de um seguidor de linha, seria preciso conhecer os valores na situação em que o sensor esteja focado na linha. Para contrastar com o solo branco, a linha será preta. Os valores de tensão obtidos na faixa preta foram 1,15V a 1,40V.

Entretanto, como a seleção de valores serão feitas via software, contidos na programação a ser gravada no microcontrolador, fora preciso obter os parâmetros em binário. Com o auxílio de uma placa de simulação (Kit Didático para Microcontrolador PIC16F877A – T&S Equipamentos Eletrônicos), fora coletado os valores em bytes, convertidos em decimal e demonstrados na tela em LCD do próprio kit. A faixa de atuação dos sensores obtidos a uma distância de aproximadamente 10mm do solo foram:

- Branco: 600 a 900 bytes;
- Preto: 100 a 300 bytes;

Com isto, os sensores poderiam ser calibrados via software, caso houvesse alguma mudança pequena da distância em relação ao solo.

3.2. CONTROLE

Para executar o controle dos motores, fora projetado um conversor CC-CC, classe E, também conhecido por "ponte H". O circuito em ponte, conforme visto (vide Figura 10), possibilita com que os motores executem rotação direta e inversa, aumentando o mobilidade do autômato.

Para obter o controle de velocidade, fora previsto o uso do PWM vindo do microcontrolador, dos pinos 16 e 17 (CCP1 e CCP2), no qual a largura de pulso será alterada via software.

No entanto, apenas os transistores como chaves não são suficientes para executar o controle, há necessidade de algum componente na base para habilitar o chaveamento de acordo com o sinal PWM vindo do PIC. Para isto, o circuito empregado conta com uma porta lógica E em cada base dos transistores deste conversor, conforme Figura 18.

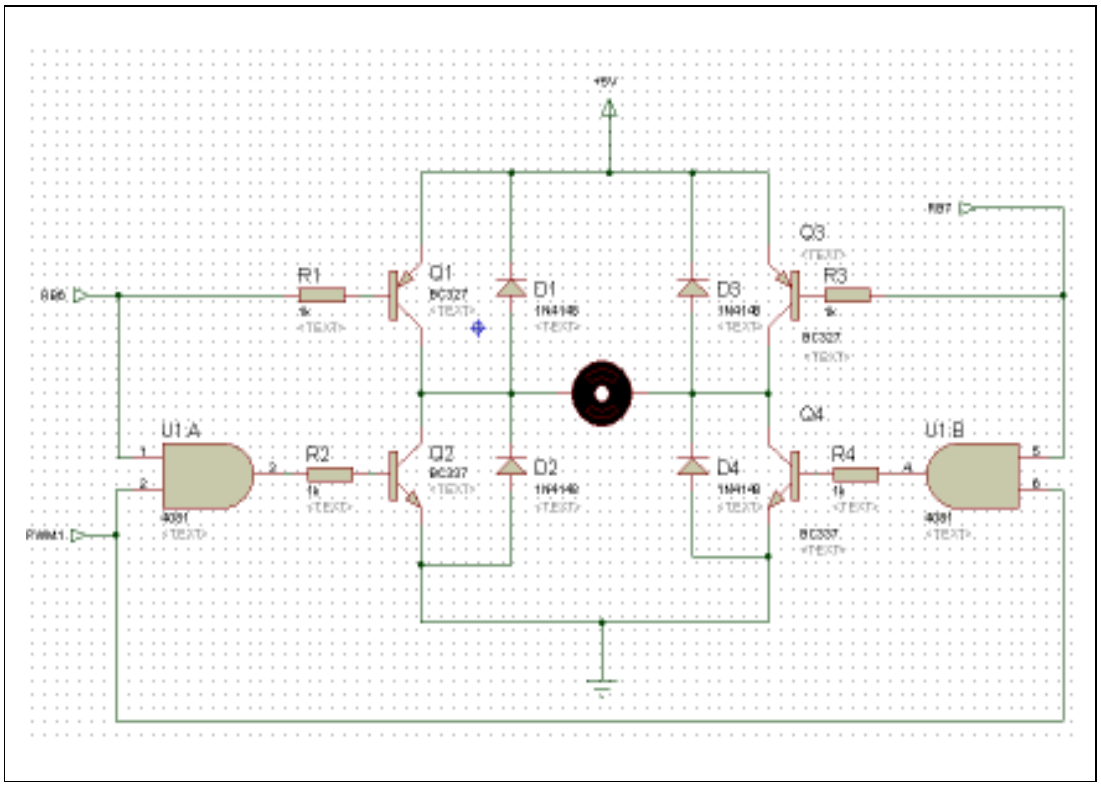


Figura 18: Circuito ponte H.

Quando o pulso do PWM e a porta RB6 estiverem em nível lógico 1 (5V), e a porta RB7 do microcontrolador estiver em nível lógico 0 (0V), os transistores Q2 e Q3 irão conduzir, produzindo a rotação em um sentido. Quando o pulso do PWM estiver em nível lógico 0 (0V), mantendo a mesma situação anterior nos pinos RB6 e RB7, a porta lógica E (U1A) terá 0 na saída, interrompendo a condução no motor. Variando a largura de pulso, haverá aumento ou diminuição na permanência das chaves (transistores) comutadas, variando, assim, a velocidade no motor. Para haver a mudança na rotação do motor, basta inverter os níveis lógicos nas portas RB6 e RB7 do microcontrolador.

3.3. CIRCUITO COMPLETO

Os materiais utilizados na montagem do circuito do RMA estão contidos na Tabela 14:

Tabela 14: Materiais utilizados na construção do robô.

MATERIAIS UTILIZADOS	
COMPONENTE	**QUANTIDADE**
Microcontrolador PIC16F877A	1
CD4081BC - 4 portas lógicas E	1
Cristal 20MHz	1
Capacitor Cerâmico 33pF	2
Capacitor Poliéster 1uF	3
Resistor 100Ω	9
Resistor 470Ω	10
Resistor 10KΩ	10
Micro-chave (Fim de Curso)	4
Chave	1
Botão	1
LED vermelho	2
Fototransistor infravermelho	3
LED emissor infravermelho	3
Transistor PNP BC327	4
Transistor NPN C337	4
Diodo 1N4148	8
Borne duplo	1
Motor	2

O microcontrolador utiliza um oscilador externo (cristal de 20MHz) cujo determina a freqüência de trabalho do mesmo.

No circuito projetado, estão dispostas as micro-chaves com finalidade de proteção. Sendo elas constituídas de contatos NA (normalmente aberto), quando acionadas, habilitam uma função, configurada via software, que fará com que o robô pare, retorne durante certo tempo e continue em outra direção. As portas utilizadas no microcontrolador são RE0, RE1, RE2 e RB0.

Quanto aos leds vermelhos, um indica que o robô está energizado. Outro será utilizado para sinalizar a operação de inversão de rotação.

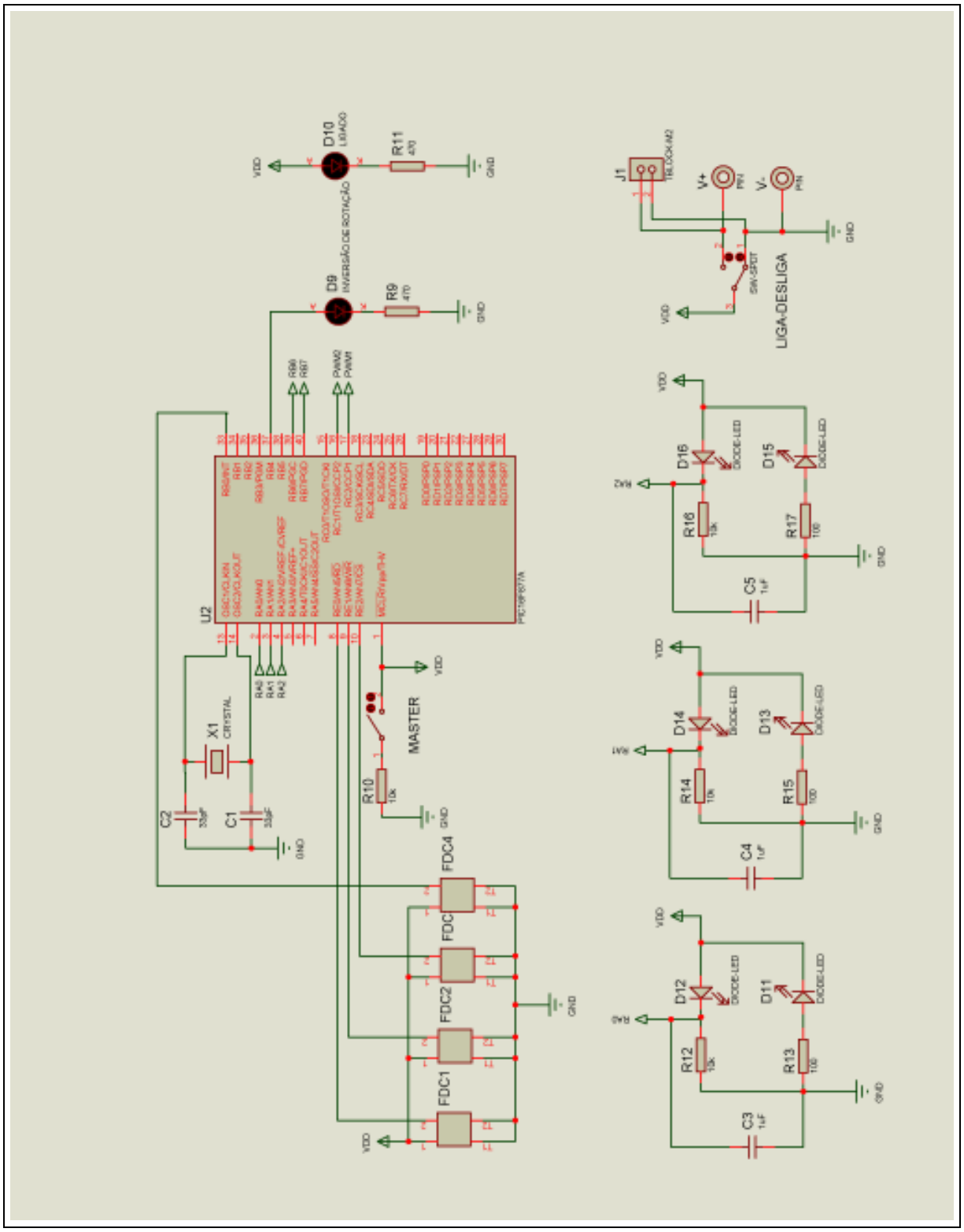
A alimentação e as saídas de sinais dos sensores infravermelhos, no qual estão sendo utilizadas as portas RA0, RA1 e RA2, foram disponibilizadas em pinos (ilhas). Um botão fora colocado na porta correspondente ao *master-clear*. Ela tem a função de reiniciar o programa gravado no PIC quando aplicado um nível lógico 0.

Embora a alimentação seja feita por baterias (6V), fora acrescentado um borne duplo a fim de permitir alimentação por outra fonte contínua de energia.

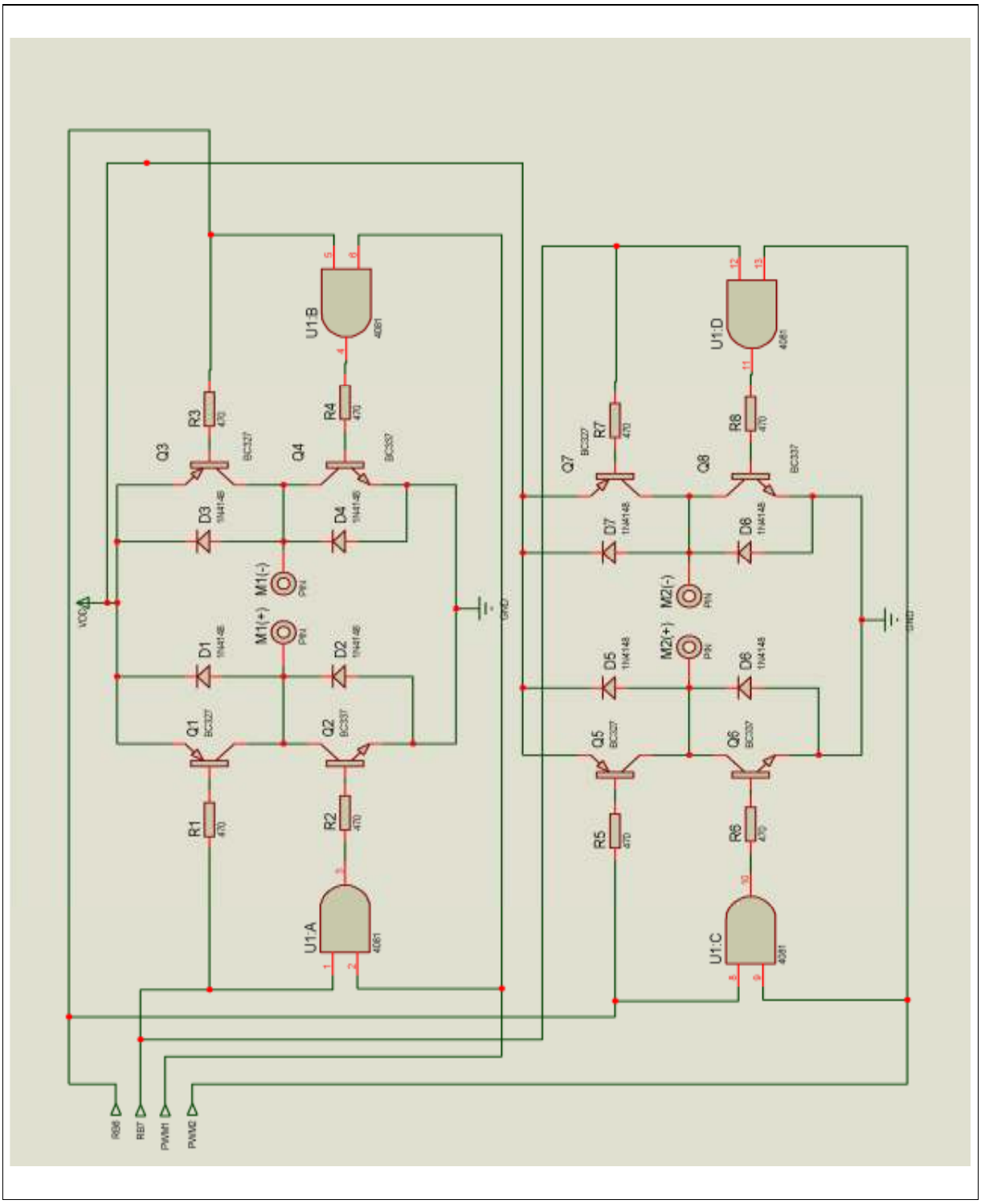
A relação dos pinos do microcontrolador utilizados no circuito estão dispostos na Tabela 15 e o circuito completo do robô, com os *drivers* de controle, foram disponibilizados na Figura 19.

Tabela 15: Relação de pinos utilizados do microcontrolador.

RELAÇÃO DE PINOS DO MICROCONTROLADOR UTILIZADOS	
PINOS	**FUNÇÃO**
RAO/AN0	Entrada de sinal analógico (sensor 1)
RA1/AN1	Entrada de sinal analógico (sensor 2)
RA2/AN2	Entrada de sinal analógico (sensor 3)
RB0	Entrada de sinal digital (FDC 4)
RB4	Saída de sinal (inversão de rotação)
RB6	Saída de sinal digital (habilitação de controle de motor)
RB7	Saída de sinal digital (habilitação de controle de motor)
RC1/CCP2	Saída de sinal (PWM 1)
RC2/CCP1	Saída de sinal (PWM 2)
MCLR	RESET


D10
LED-AZUL
R11
470
VDD
GND
INVERSÃO DE ROTAÇÃO
D9
R9
470
J1
V+
V-
PIN
SW-SPDT
LIGA-DESLIGA
PWM2
PWM1
U2
X1
CRYSTAL
C2
33pF
C1
33pF
MASTER
R10
10k
FDC4
FDC3
FDC2
FDC1
D16
D15
D14
D13
D12
D11
DIODE-LED
R16
R17
R14
R15
R12
R13
100
C5
C4
C3
1uF

(a)



(b)

Figura 19: Circuito Completo: (a) circuito de controle e de sensores; (b) circuito acionador em ponte.

3.4. PROGRAMAÇÃO

O software fora construído a partir da lógica de programação, contendo a rotinas a serem executadas. O diagrama pode ser visualizado no fluxograma da Figura 20 . Como os sensores mecânicos atuarão como proteção, o robô seguirá em frente somente se os mesmos estiverem desativados. Verificado a situação dos fins de curso, o programa fará a leitura dos sensores infravermelhos. Há cinco situações em que os sensores infravermelhos determinarão as ações do robô:

- Os três sensores desativados (S1, S2, S3): o robô não identificou a faixa, por isso ficará seguindo sem rumo até encontra-la;
- Sensor localizado no centro ativado (S2): andará em frente, seguindo a linha;
- Sensor localizado a direita habilitado (S3): virará à direita;
- Sensor localizado à esquerda habilitado (S1): virará à esquerda;
- Todos os sensores ativados (S1, S2, S3): o robô irá parar.

A partir destes algoritmos, fora estruturado o programa utilizando a linguagem C, através do software MicroC, conforme a Figura 21. No início do projeto, deve-se configurar o tipo de microcontrolador a ser empregado, a freqüência de clock e os bits de configurações do PIC.

Ao iniciar o programa, as portas a serem utilizadas do PIC (TRISA, TRISB, TRISD e TRISE) foram habilitadas como entrada ou saída de sinal. Outro ponto importante fora definir as portas RA0, RA1 e RA2 como analógicas. Para isto seguiu-se a configuração descrita a tabela contida no Anexo I (adcon1=0b00000010).

Seguindo a lógica, enquanto as chaves estivessem desabilitadas, os sensores infravermelhos fariam com que o RMA andasse e seguisse a linha no chão. Caso uma das chaves fosse ativada, o robô iria voltar durante um certo tempo (incremento de “a” até 20) iria para frente, em direção oposta do fim de curso atuado.

A todo instante a velocidade é habilitada em 100%, exceto nas mudanças de direção. Para virar, o robô diminui a velocidade de uma das rodas e mantém o máximo de PWM na outra. A freqüência de trabalho do programa é de 20kHz. A faixa de trabalho do sensor varia de menor ou igual a 300 para a linha e maior ou igual a 600, para o chão.

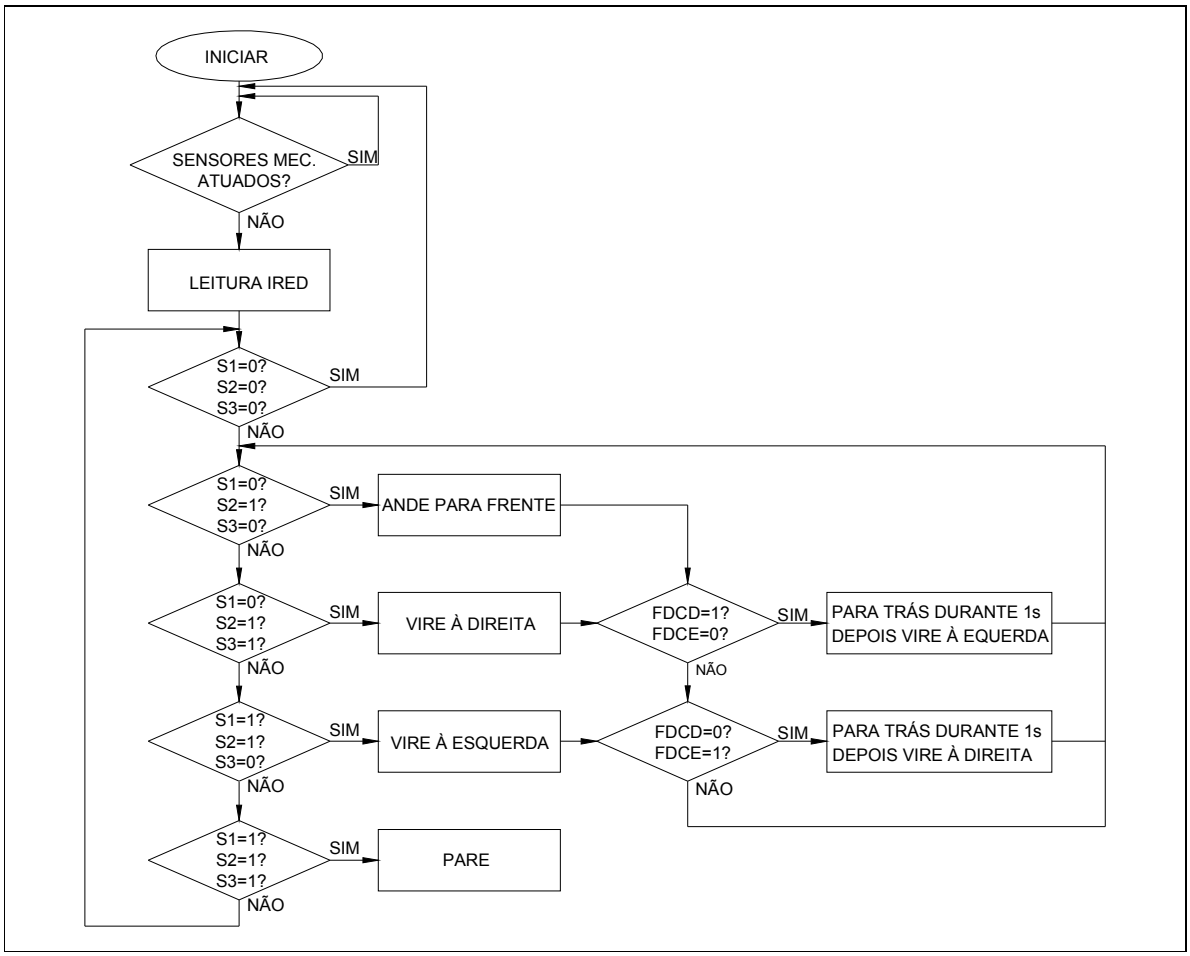


Figura 20: Fluxograma da lógica de programação.

```
void main()
{
int S1,S2,S3,a;              //VARIÁVEIS QUE RECEBEM A LEITURA IRED

trisa=255;                   //CONFIGURA PORTA COMO ENTRADA ANALÓGICA
trisa.f0=1;                  //RA0 – ENTRADA ANALÓGICA 0 = AN0
trisa.f1=1;                  //RA1 – ENTRADA ANALÓGICA 1 = AN1
trisa.f3=1;                  //RA3 – ENTRADA ANALÓGICA 3 = AN3
trisb=0b00000001;            //CONFIGURA RB0 COMO ENTRADA DIGITAL E DEMAIS
PORTAS COMO SAÍDA
trisd=0;                     //CONFIGURA PORTD COMO SAÍDA
trise=255;                   //CONIGURA PORTE COMO ENTRADA DIGITAL
adcon1=0b00000010;           //PROGRAMA AN0, AN1,AN3
a=0;
```

```
while(porte.f0==0||porte.f1==0&&porte.f2==0||portb.f0==0)       //FDC DESACIONADO
{
portb.f6==1;
portb.f7==0;
S1=adc_read(0);                 //SENSOR S1 SERÁ LIDO PELO CANAL 0
S2=adc_read(1);                 //SENSOR S2 SERÁ LIDO PELO CANAL 1
S3=adc_read(2);                 //SENSOR S3 SERÁ LIDO PELO CANAL 3
if(S1>=600&&S2>=600&&S3>=600)       //LENDO PARTE BRANCA
{
portb.f4==1;                    //ACIONA LED DO
while(a<20)
{
a++;
pwm1_init(20000);               //INICIALIZA PWM, CANAL CCP1, COM 20Khz
pwm1_change_duty(255);          //INICIALIZA DUTY COM 100%
pwm1_start();                   //INICIALIZA PWM

pwm2_init(20000);               //INICIALIZA PWM, CANAL CCP2, COM 20Khz
pwm2_change_duty(255);          //INICIALIZA DUTY COM 100%
pwm2_start();                   //INICIALIZA PWM
}
a==0;
pwm1_stop();
pwm2_stop();
portb.f4==0;                    //APAGA LED D0
}
if(S1>600&&S2<=300&&S3>=600)        //SEGUE EM FRENTE
{
pwm1_init(20000);               //INICIALIZA PWM, CANAL CCP1, COM 15Khz
pwm1_change_duty(255);          //INICIALIZA DUTY COM 100%
pwm1_start();                   //INICIALIZA PWM

pwm2_init(20000);               //INICIALIZA PWM, CANAL CCP2, COM 15Khz
pwm2_change_duty(255);          //INICIALIZA DUTY COM 100%
pwm2_start();                   //INICIALIZA PWM
}
if(S1>=600&&S2>=600&&S3<=300)       //VIRA A DIREITA
{
pwm1_change_duty(255);          //INICIALIZA DUTY COM 100%
pwm1_start();                   //INICIALIZA PWM

pwm2_change_duty(0);            //INICIALIZA DUTY COM 0%
pwm2_start();                   //INICIALIZA PWM
}
if(S1<=300&&S2>=600&&S3>=600)       //VIRA A ESQUERDA
{
```

```
pwm1_change_duty(0);        //INICIALIZA DUTY COM 0%
pwm1_start();               //INICIALIZA PWM

pwm2_change_duty(255);      //INICIALIZA DUTY COM 100%
pwm2_start();               //INICIALIZA PWM
}
if(porte.f0==1||porte.f1==1&&porte.f2==0||portb.f0==0)   //ACIONADO FIM DE CURSO
ESQUERDA
{
while(a<20)
{                           //RETORNA
a++;
portb.f6==0;
portb.f7==1;

pwm1_change_duty(255);      //INCALIZA DUTY COM 100%
pwm1_start();               //INICIALIZA PWM

pwm2_change_duty(255);      //INICIALIZA DUTY COM 100%
pwm2_start();               //INICIALIZA PWM
}
a=0;
portb.f6=1;
portb.f7=0;
                            //VIRA A DIREITA
pwm1_change_duty(255);      //INICIALIZA DUTY COM 100%
pwm1_start();               //INICIALIZA PWM

pwm2_change_duty(0);        //INICIALIZA DUTY COM 0%
pwm2_start();               //INICIALIZA PWM
}
if(porte.f0==0||porte.f1==0&&porte.f2==1||portb.f0==1)   //ACIONADO FIM DE CURSO
DIREITA
{
while(a<20)
{                           //RETORNA
a++;
portb.f6=0;
portb.f7=1;

pwm1_change_duty(255);      //INICIALIZA DUTY COM 100%
pwm1_start();               //INICIALIZA PWM

pwm2_change_duty(255);      //INICIALIZA DUTY COM 100%
pwm2_start();               //INICIALIZA PWM
}
```

```
a=0;
portb.f6==1;
portb.f7==0;
                              //VIRA A ESQUERDA
pwm1_change_duty(0);          //INICIALIZA DUTY COM 100%
pwm1_start();                 //INICIALIZA PWM

pwm2_change_duty(255);        //INICIALIZA DUTY COM 0%
pwm2_start();                 //INICIALIZA PWM
}
}
}
```

Figura 21: Programação em linguagem C.

3.5. CONFECÇÃO DA PLACA

Com o auxílio do software "Proteus®", fora possível executar, além da simulação do circuito, o desenho da placa de circuito impresso. O layout da placa está disponível no Apêndice A.

4. RESULTADOS E DISCUSSÕES

Após a montagem dos sensores infravermelhos, fora coletado alguns valores em determinadas situações para análise do funcionamento (Tabela 16) e, assim, levantar a curva de atuação do sensor, conforme a Figura 22.

Tabela 16: Dados obtidos do sensor infravermelho.

CURVA DE OPERAÇÃO DO SENSOR INFRAVERMELHO			
DISTÂNCIA (mm)	Vout 1	Vout 2	Vout (MÉDIA)
250	0,352	0,354	0
240	0,35	0,377	0
230	0,355	0,398	0,3765
220	0,376	0,41	0,393
210	0,384	0,423	0,4035
200	0,409	0,441	0,425
190	0,421	0,458	0,4395
180	0,45	0,487	0,4685
170	0,461	0,508	0,4845
160	0,49	0,534	0,512
150	0,52	0,568	0,544
140	0,54	0,586	0,563
130	0,59	0,615	0,6025
120	0,641	0,693	0,667
110	0,7	0,75	0,725
100	0,795	0,83	0,8125
90	0,913	0,957	0,935
80	1,06	1,07	1,065
70	1,25	1,22	1,235
60	1,55	1,55	1,55
50	1,92	2	1,96
40	2,57	2,73	2,65
30	3,6	3,75	3,675
20	4,93	4,95	4,94
10	4,99	4,99	0
0	4,99	4,99	0

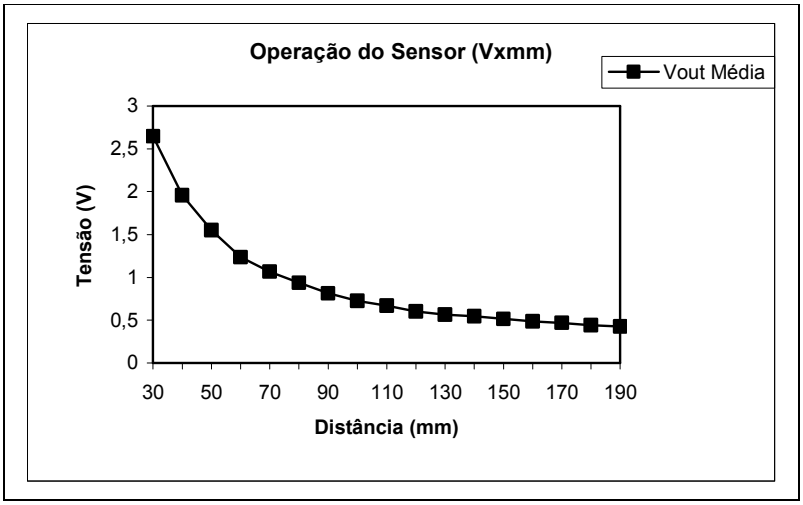


Figura 22: Curva de atuação do sensor infravermelho.

Para fins de teste, no circuito demonstrado na (b)

Figura 19, fora retirado dois dos dispositivos mecânicos, o que não acarretaria em problemas futuros, e os sensores infravermelhos foram substituídos por potenciômetros, para proporcionar a variação de sinais na porta do microcontrolador.

Depois de compilado o programa, ou seja, criado um arquivo de extensão *.hex*, utilizando o software Proteus®, fora transposto este arquivo no PIC (virtual), além de configurado as demais propriedades, conforme Figura 23.

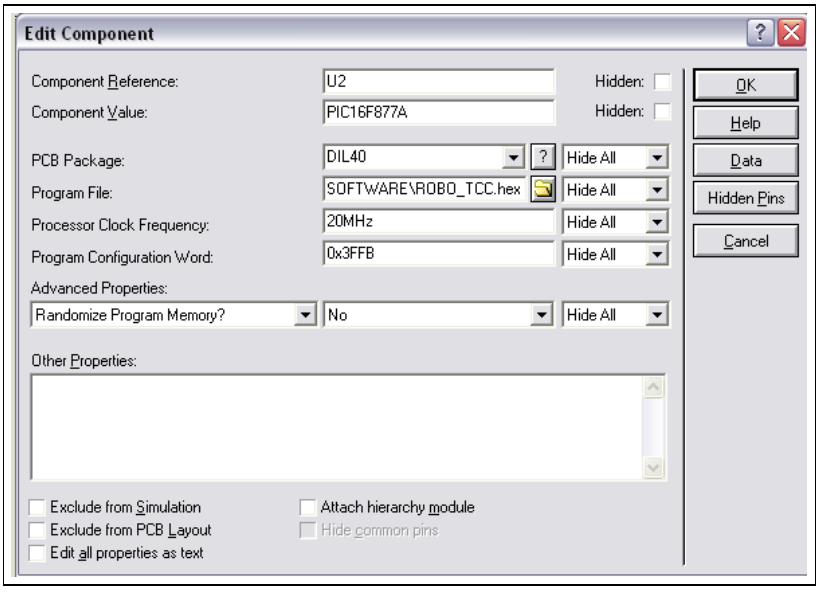


Figura 23: Editor do microcontrolador no simulador.

A freqüência de clock determinada fora 20MHz, conforme configurado no programa compilado.

Com auxílio de um osciloscópio virtual, fora medido quatro pontos no circuito: as saídas PWM das portas CCP1 e CCP2 do microcontrolador e os sinais que chegam em cada motor. Na Figura 24, a ordem dos sinais medidos são: PWM1 (amarelo), PWM2 (azul), Motor 1 (direita - rosa) e Motor 2 (esquerda - verde).

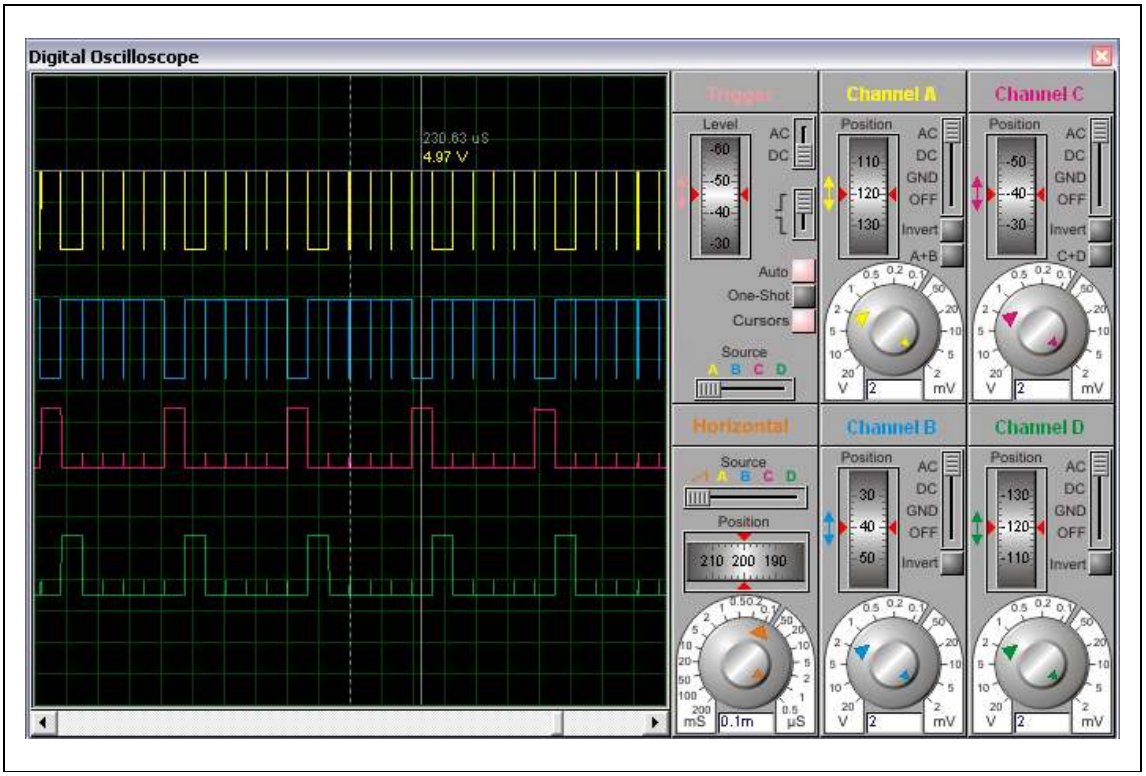


Figura 24: Sinal obtido com detecção de faixa central.

O primeiro sinal obtido, fora da situação em que o sensor localizado no centro do robô detecta a linha escura no solo, enquanto os demais lêem apenas o solo branco. Nota-se na Figura 24 que o valor de PWM é máximo (4,97V), conforme programado em linguagem C (valor 255).

Ao ser detectado a faixa pelo sensor localizado à direita do robô, os valores de PWM se alteram. Como fora determinado no software, o PWM1 mantém-se no valor máximo (5,05V) e o PWM2 vai a zero (509,33mV), conforme a Figura 25. Observa-se que o sinal no motor 2 é baixo (1,27V) enquanto o motor 1 continua com máxima potência (4,98V).

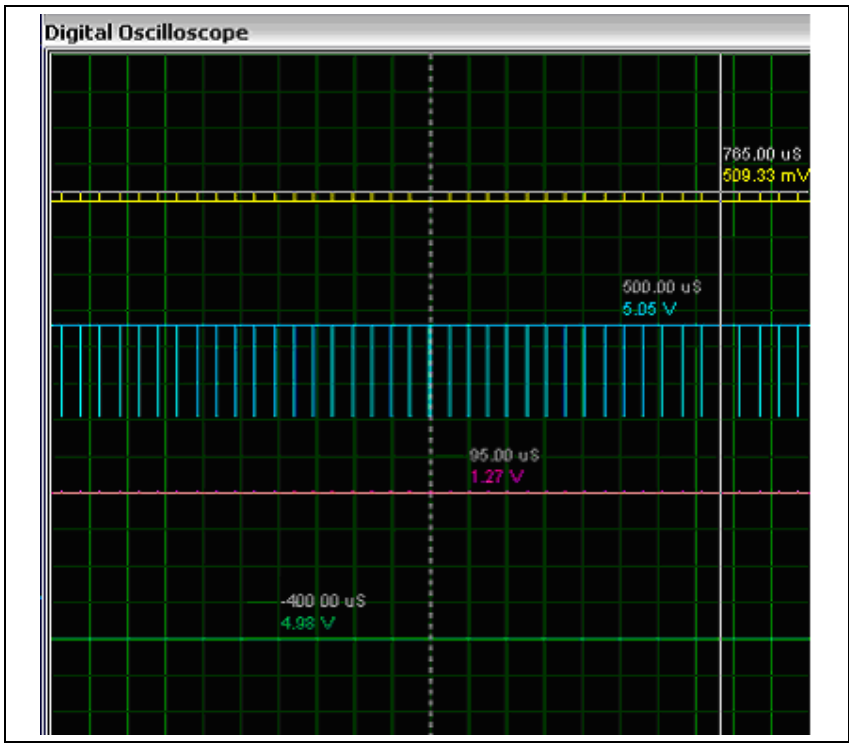


Figura 25: Sinal obtido com detecção de faixa à direita.

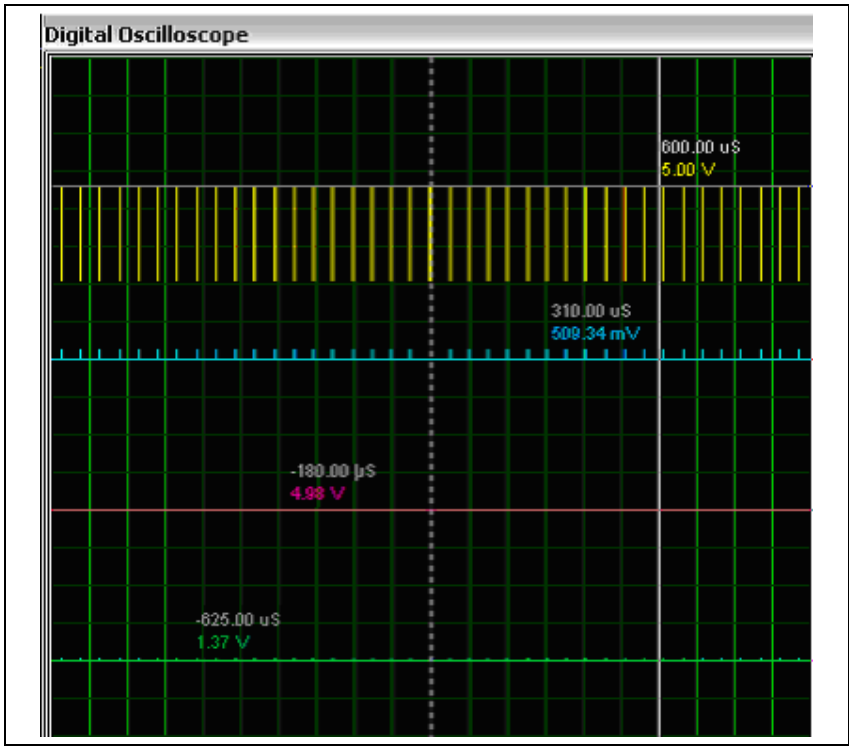


Figura 26: Sinal obtido com detecção de faixa à esquerda.

Do mesmo modo se dá para a situação em que o sensor detecta a linha, porém ocorre o contrário: o PWM2 mantém-se com valor máximo (5,00V), o PWM1 obtém o valor zero (509,34mV), e os sinais nos motores acompanham os respectivos sinais de pwm.

Para os dispositivos mecânicos, quando acionados, inverte-se a polarização dos motores, acionados pela porta RB7 e anulando a RB6, durante um certo tempo. Após este intervalo, o sentido polar dos motores volta ao estado inicial e, se caso tenha sido acionado o fim de curso direito, o PWM1 mantém-se em valor máximo (5,00V) enquanto o PWM2 é travado (307,34mV). Caso o fim de curso atuado tenha sido o esquerdo ocorre o contrário.

Para melhor visualização, na Figura 27, no momento em que o PWM está em sentido inverso, através do programa, está com 50% da potência (valor 127), e depois do intervalo retorna aos valores máximos, de acordo com o descrito acima (255).

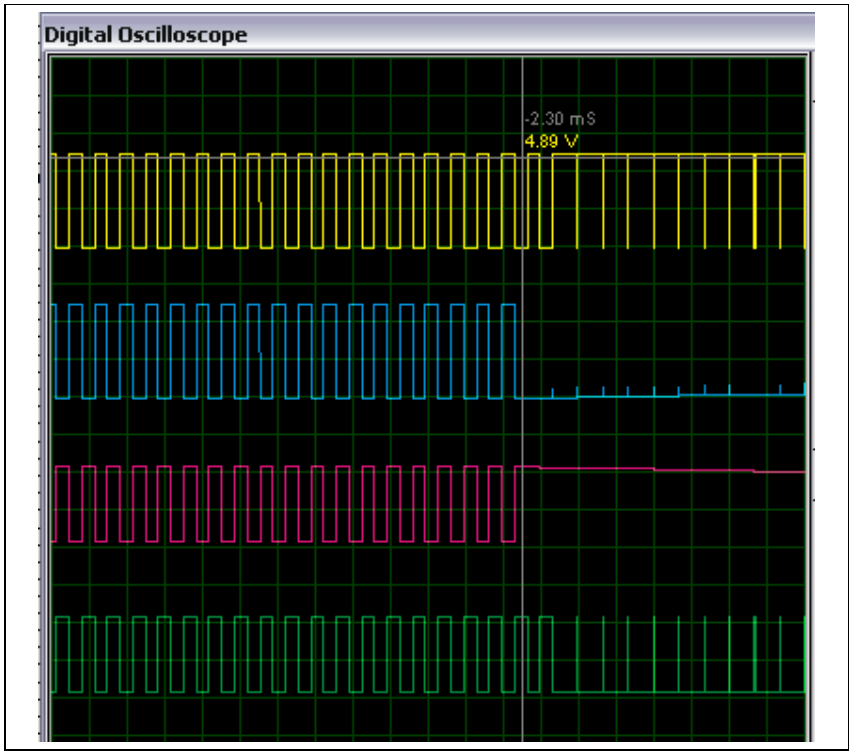


Figura 27: Dispositivo atuando para prover inversão de rotação nos motores.

Além da simulação, para comprovar a veracidade dos dados apresentados, foram feitos testes no *Protoboard*, com o auxílio do Kit Didático para Microcontrolador PIC16F877A (T&S Equipamentos Eletrônicos).

Em suma, os valores medidos no osciloscópio e multímetro condisseram com os valores simulados com o software, não tendo a necessidade de se relatar mais.

Após os testes e plotagem do circuito em uma placa de circuito impresso, fora montado o protótipo obtendo os resultados conforme dispostos nas figuras abaixo:

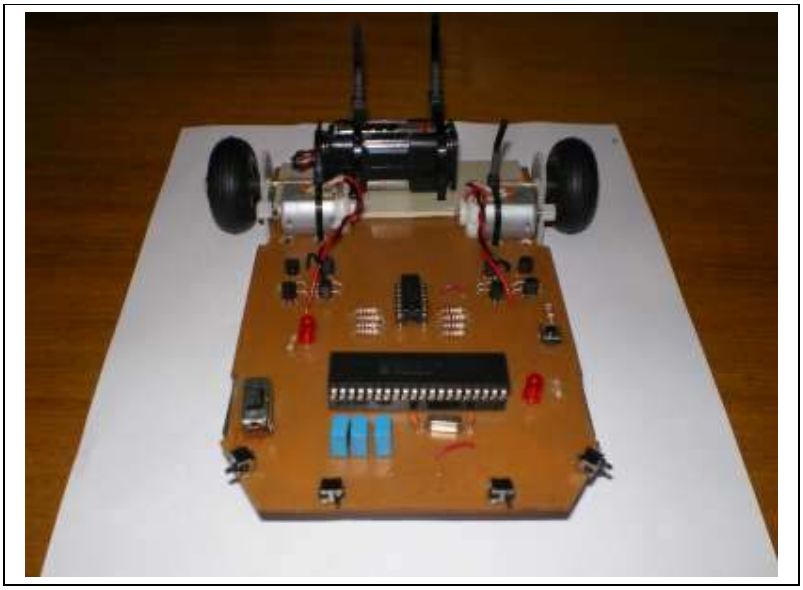


Figura 28: Vista frontal do robô.

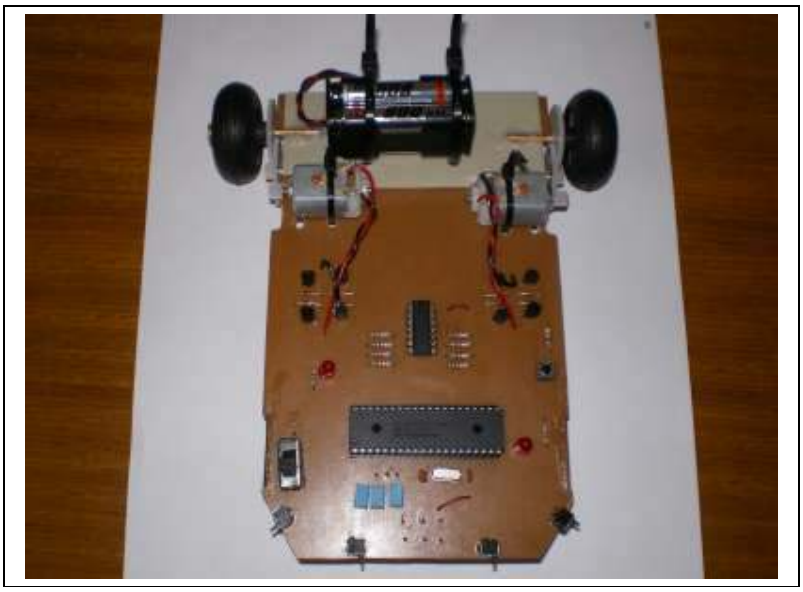


Figura 29: Vista superior do robô.

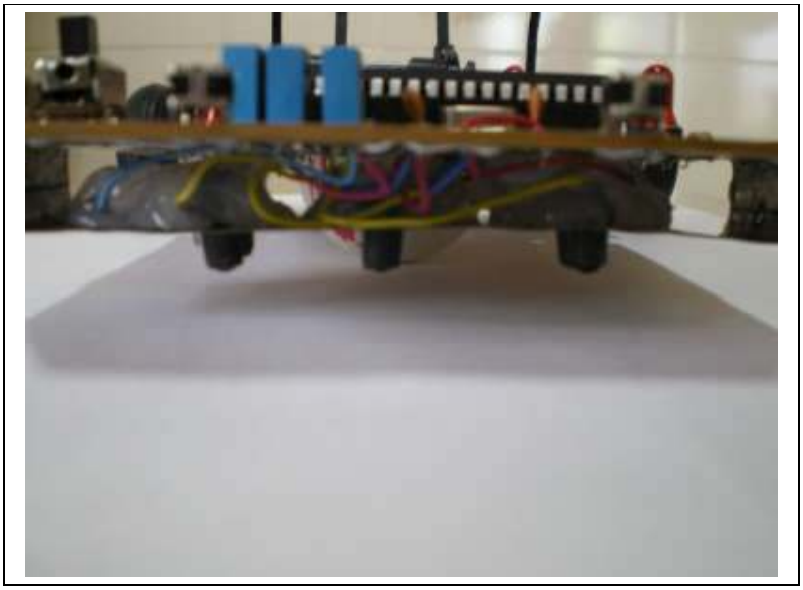


Figura 30: Fixação dos sensores infravermelhos.

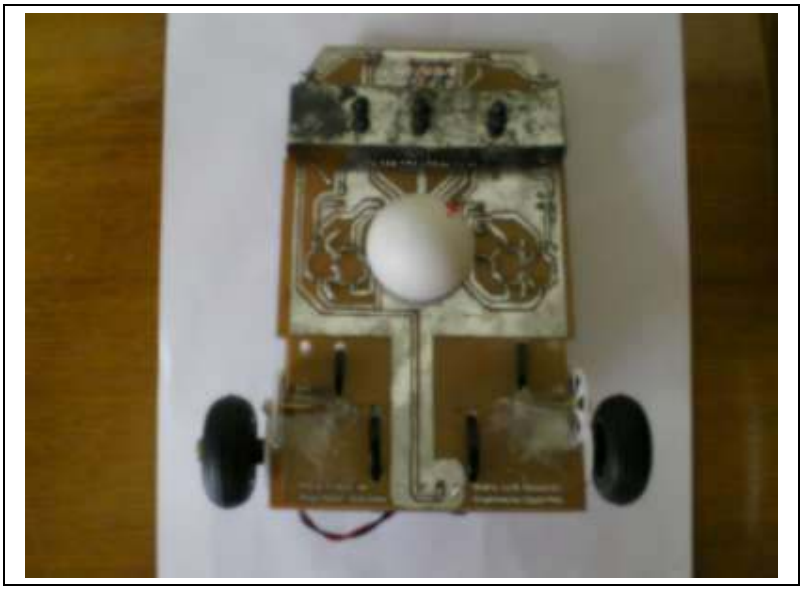


Figura 31: Vista inferior do robô.

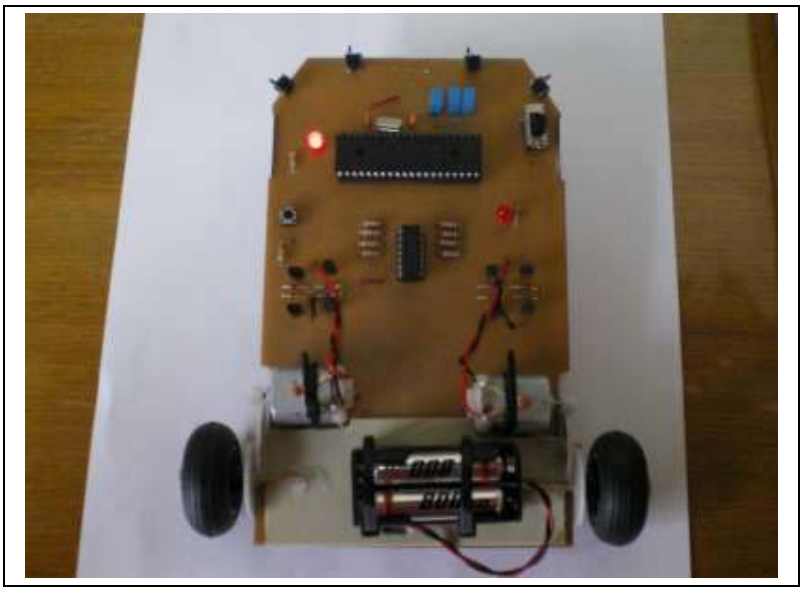


Figura 32: Vista superior do robô ligado.

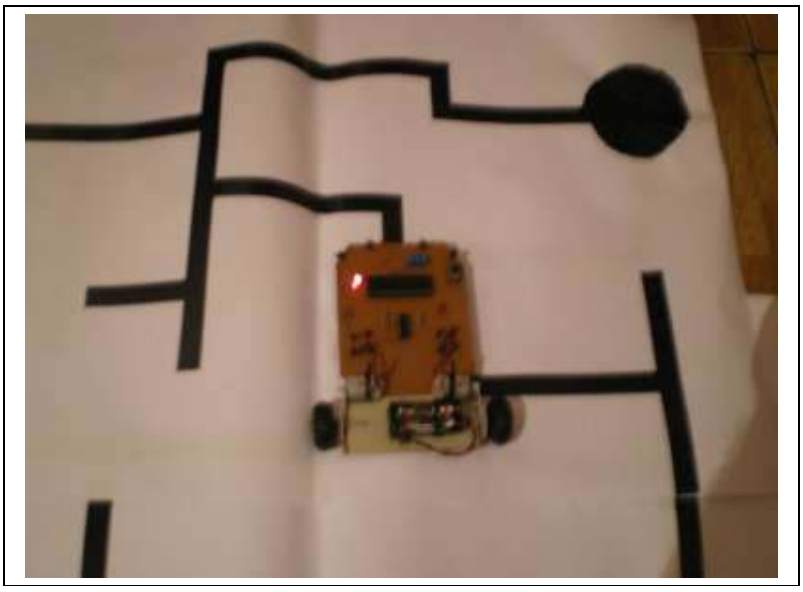


Figura 33: Vista superior do circuito de trajeto do robô.

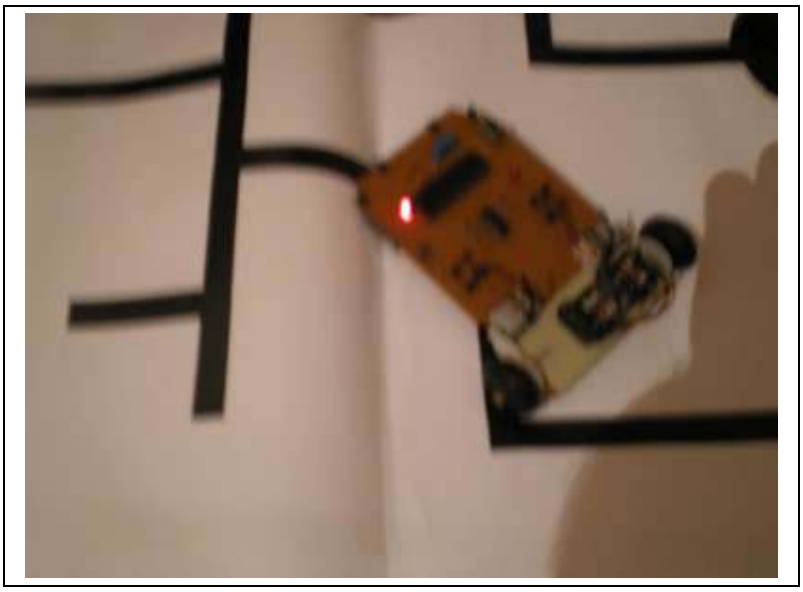


Figura 34: Tentativa de simulação.

5. CONCLUSÃO

Através deste presente trabalho, fora possível desbravar o amplo universo da robótica, através das informações básicas para sua projeção e construção.

O protótipo, não obstante a simplicidade de sua constituição, dispensou um tempo considerável para a elaboração, estudo dos subconjuntos que o compõe, definição de hardware e configuração das rotinas durante a programação.

O circuito apresentou resultados positivos quanto à simulação e testes com *Protoboard*. O microcontrolador PIC16F877A provou-se suficiente para atender as demandas exigidas pela lógica de atuação do RMA, tendo capacidade de maiores implementações com periféricos e programação. O *driver* da ponte H respondeu bem aos comandos de funcionamento direto e inverso de rotação, mesmo com o uso de componentes analógicos e digitais em um circuito de potência. Os sensores atuaram no circuito conforme medidos isoladamente no kit didático, havendo pequenas variações de tensão quanto ao testado, contudo não comprometendo o seu emprego no robô, devido ao fato da parametrização ter sido feita via software e estipulado uma faixa de atuação, e não pontos fixos. Além da utilização como sensores de leitura de faixa, pelos estudos, é possível a aplicação como sensores de distância, substituindo os sensores mecânicos perfeitamente.

O software sofreu diversas modificações, até chegar ao resultado em que fora apresentado. Para melhor entendimento, lógicas simples de atuação com o circuito com menos recursos foram necessários. Para maiores aplicações, a programação necessitará ser melhor desenvolvida, pricipalmente se continuar optando pelo uso da linguagem C.

O custo deste protótipo não fora muito elevado, principalmente por utilizar-se de algumas partes mecânicas de um robô comercial e os sensores terem sidos fabricados com dispositivos analógicos simples, pois se tivesse optado por comprar um sensor infavermelho e desenvolver os mecanismos do robô, este seria inviavel para a finalidade deste trabalho. Os gastos totais giraram em torno de R$100,00.

As dificuldades encontradas se resumem em dois pontos, dentre todas as etapas: a programação e o protótipo final. Com o acrescento de alguns dispositivos, algumas configurações de sub-rotinas e *flags* não foram atendidas. Com isto, o processo de desenvolvimento tornou-se lento, até que fosse encontrada a solução. Quanto ao protótipo, houve dificuldades no

funcionamento correto do robô. Enquanto o circuito testado em *protoboard* e com o uso do kit e desenvolvimento para PIC16F877A obteve desempenho satisfatório, de acordo com o que fora programado, na placa de circuito impresso os motores e a função de proteção através dos dispositivos mecânicos funcionaram, contudo, a aplicação que seria determinada pela leitura dos sensores não apresentou resultados.

Portanto, através da simulação do circuito com recursos virtuais (software) e auxílio do kit didático para microcontroladores, o circuito funcionou corretamente, podendo, assim, validar o circuito e a programação, não obstante aos poblemas gerados no protótipo final. Para isto, seria necessário um tempo maior para estudo do caso quanto a possíveis falhas: trilhas corrompidas, mal-contatos, curto-circuito entre trilhas, até mesmo a hipótese de substituição de alguns componentes que possam ter sofridos algum dano durante os testes.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] WERNECK, Marcelo M. **Transdutores e Interfaces.** Rio de Janeiro: LTC, 1996

[2] PAZOS, Fernando. **Automação de Sistemas e Robótica.** Rio de Janeiro: Axcel Books, 2002.

[3] OGATA, Katsuhiko. **Engenharia de Controle Moderno**. Tradução: Paulo Álvaro Maya. São Paulo: Prentice Hall, 2003. 4 ed.

[4] GONÇALVES, Paulo César. **Protótipo de um Robô Móvel de Baixo Custo para Uso Educacional**. Trabalho de Pós-Graduação; Maringá, 2007.

[5] CRAIG, John J. **Introduction to robotics: mechanics and control**. New Jersey: Prentice Hall, 2005, 3 ed.

[6] ZANCO, Wagner Silva da. **Microcontroladores PIC: técnicas de software e hardware para projetos de circuitos eletrônicos com base no PIC16F877A**. São Paulo: Érica, 2006.

[7] SOUZA, David José de; LAVINIA, Nicolas C. **Conectando o PIC 16F877A: recursos avançados**. São Paulo: Érica, 2003.

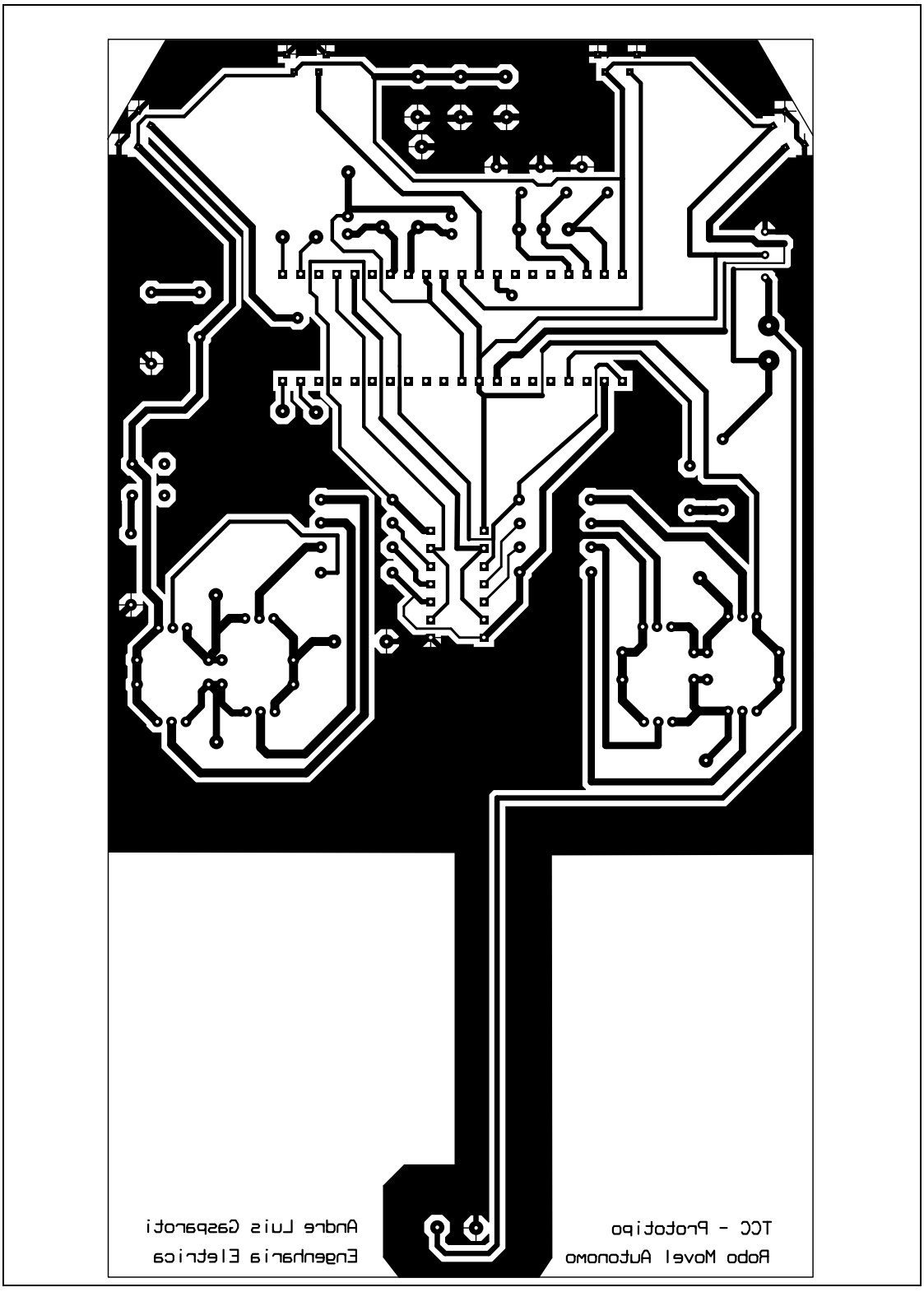
[8] ZANELATTO, Marcelo S. **Robótica Educacional nos Cursos de Ciência da Computação**. Trabalho de Conclusão de Curso; Londrina, 2004.

[9] HONDA, Flávio. **Motores de Corrente Contínua.** Unidade de Automação e Controle - Acionamentos e Motores Elétricos Siemens; Publicação Técnica, 2006.

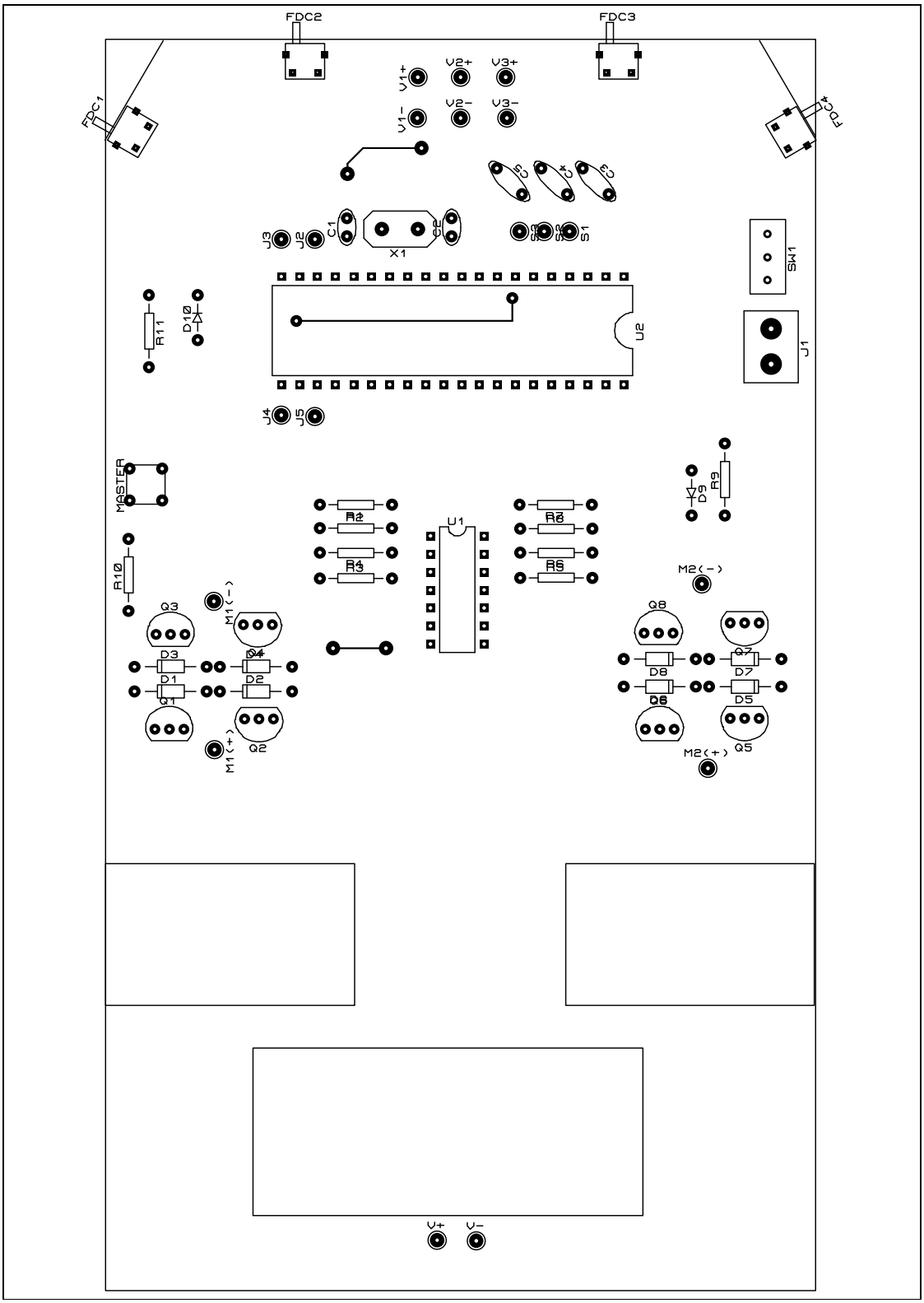
[10] **Datasheet PIC16F87XA**. Disponível em URL: ww1.microchip.com/downloads/em/DeviceDoc/39582b.pdf. Acesso: 06/09/09.

[11] RASHID, Muhammad H. **Eletrônica de Potência: circuito, dispositivos e aplicações**. Tradução: Carlos Alberto Favato. São Paulo: Makron Books, 1999.

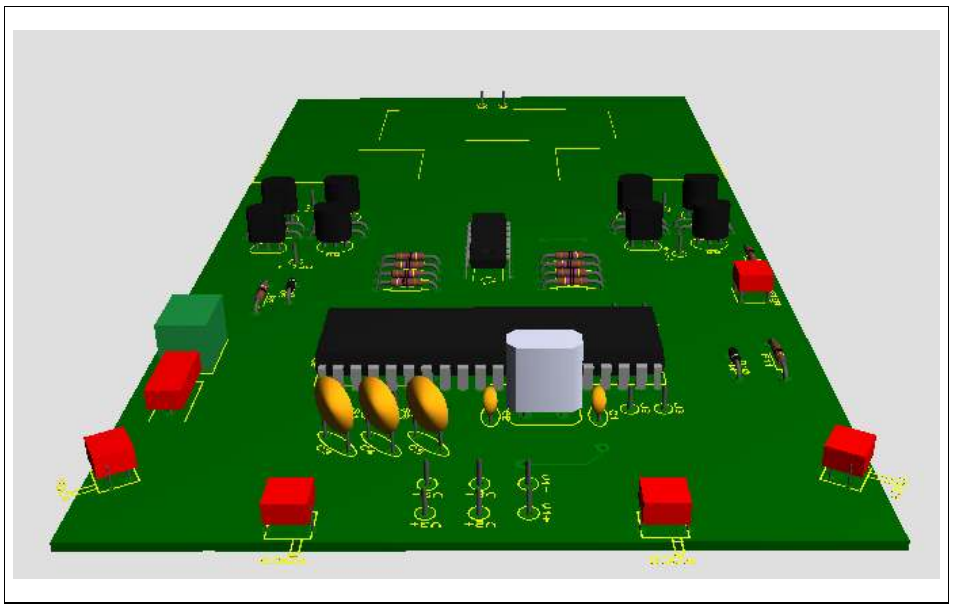
APÊNDICE A – LAYOUT DA PLACA

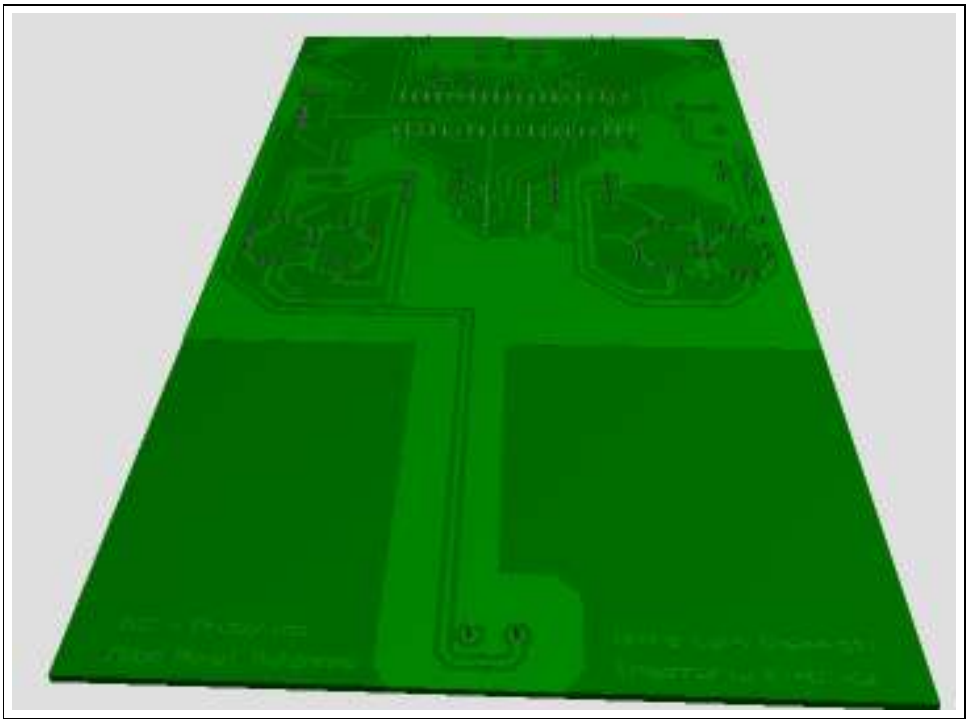


APÊNDICE B – DISPOSIÇÃO DOS COMPONENTES



APÊNDICE C – PLACA EM 3D





ANEXO I – DATASHEET PIC16F877A

PDIP (40 pin)

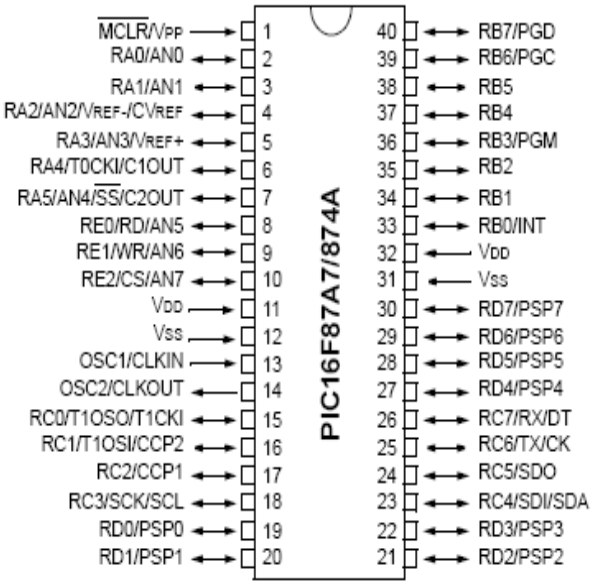
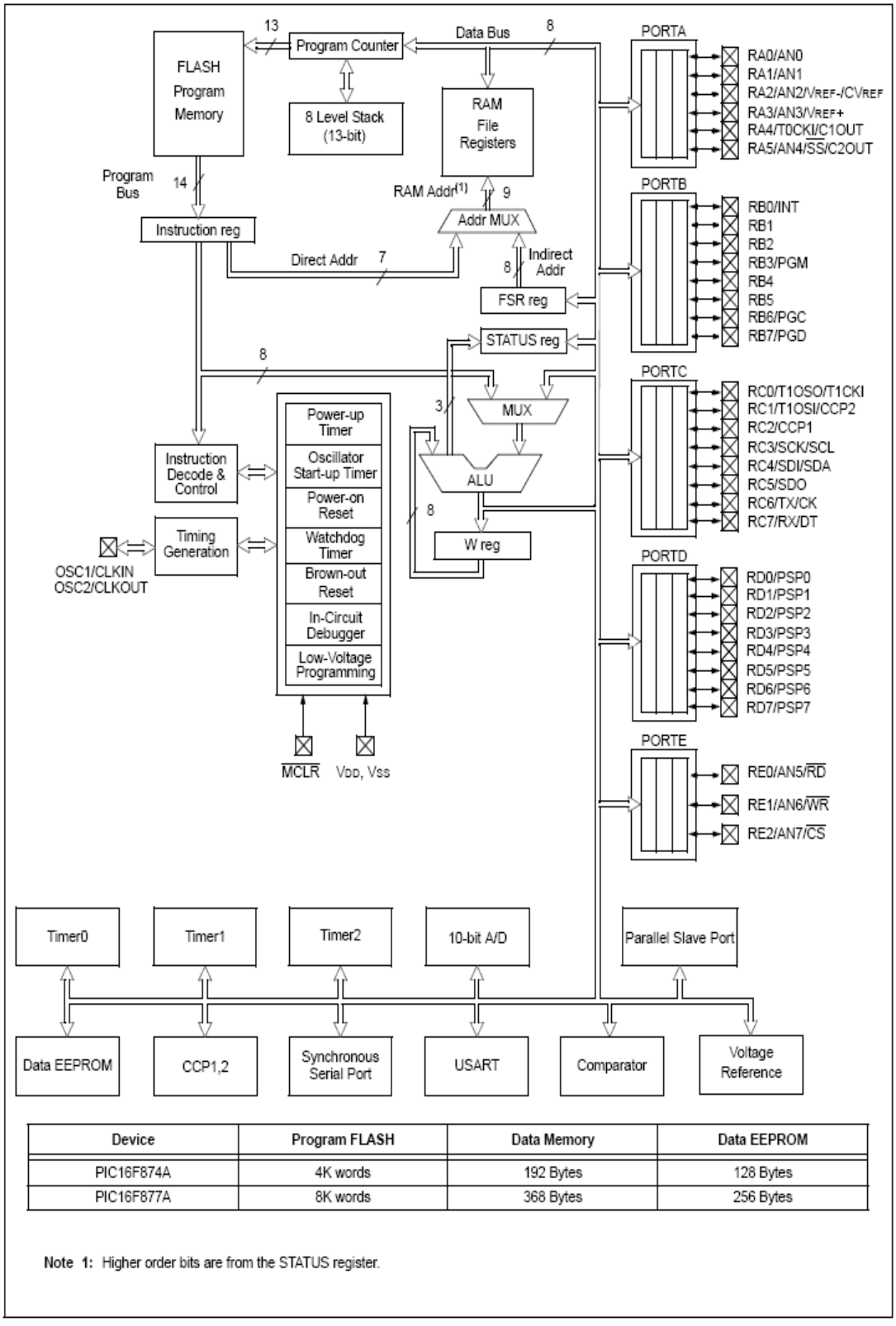


TABLE 1-1: PIC16F87XA DEVICE FEATURES

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

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

FIGURE 1-2: PIC16F874A/877A BLOCK DIAGRAM



Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

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

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI	13	14	30	I	ST/CMOS(4)	Oscillator crystal or external clock input.
OSC1						Oscillator crystal input or external clock source input. ST buffer when configured in RC mode. Otherwise CMOS.
CLKI						External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal or clock output.
OSC2						Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO						In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (input) or programming voltage (output).
$\overline{MCLR}$						Master Clear (Reset) input. This pin is an active low RESET to the device.
VPP						Programming voltage input.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19		TTL	
RA0				I/O		Digital I/O.
AN0				I		Analog input 0.
RA1/AN1	3	4	20		TTL	
RA1				I/O		Digital I/O.
AN1				I		Analog input 1.
RA2/AN2/VREF-/CVREF	4	5	21		TTL	
RA2				I/O		Digital I/O.
AN2				I		Analog input 2.
VREF-				I		A/D reference voltage (Low) input.
CVREF				O		Comparator VREF output.
RA3/AN3/VREF+	5	6	22		TTL	
RA3				I/O		Digital I/O.
AN3				I		Analog input 3.
VREF+				I		A/D reference voltage (High) input.
RA4/T0CKI/C1OUT	6	7	23		ST	
RA4				I/O		Digital I/O – Open drain when configured as output.
T0CKI				I		Timer0 external clock input.
C1OUT				O		Comparator 1 output.
RA5/$\overline{SS}$/AN4/C2OUT	7	8	24		TTL	
RA5				I/O		Digital I/O.
$\overline{SS}$				I		SPI slave select input.
AN4				I		Analog input 4.
C2OUT				O		Comparator 2 output.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8		TTL/ST(1)	
RB0				I/O		Digital I/O.
INT				I		External interrupt.
RB1	34	37	9	I/O	TTL	Digital I/O.
RB2	35	38	10	I/O	TTL	Digital I/O.
RB3/PGM	36	39	11		TTL	
RB3				I/O		Digital I/O.
PGM				I/O		Low voltage ICSP programming enable pin.
RB4	37	41	14	I/O	TTL	Digital I/O.
RB5	38	42	15	I/O	TTL	Digital I/O.
RB6/PGC	39	43	16		TTL/ST(2)	
RB6				I/O		Digital I/O.
PGC				I/O		In-Circuit Debugger and ICSP programming clock.
RB7/PGD	40	44	17		TTL/ST(2)	
RB7				I/O		Digital I/O.
PGD				I/O		In-Circuit Debugger and ICSP programming data.
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32		ST	
RC0				I/O		Digital I/O.
T1OSO				O		Timer1 oscillator output.
T1CKI				I		Timer1 external clock input.
RC1/T1OSI/CCP2	16	18	35		ST	
RC1				I/O		Digital I/O.
T1OSI				I		Timer1 oscillator input.
CCP2				I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	17	19	36		ST	
RC2				I/O		Digital I/O.
CCP1				I/O		Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37		ST	
RC3				I/O		Digital I/O.
SCK				I/O		Synchronous serial clock input/output for SPI mode.
SCL				I/O		Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	23	25	42		ST	
RC4				I/O		Digital I/O.
SDI				I		SPI data in.
SDA				I/O		I^2C data I/O.
RC5/SDO	24	26	43		ST	
RC5				I/O		Digital I/O.
SDO				O		SPI data out.
RC6/TX/CK	25	27	44		ST	
RC6				I/O		Digital I/O.
TX				O		USART asynchronous transmit.
CK				I/O		USART 1 synchronous clock.
RC7/RX/DT	26	29	1		ST	
RC7				I/O		Digital I/O.
RX				I		USART asynchronous receive.
DT				I/O		USART synchronous data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38		ST/TTL(3)	
RD0				I/O		Digital I/O.
PSP0				I/O		Parallel Slave Port data.
RD1/PSP1	20	22	39		ST/TTL(3)	
RD1				I/O		Digital I/O.
PSP1				I/O		Parallel Slave Port data.
RD2/PSP2	21	23	40		ST/TTL(3)	
RD2				I/O		Digital I/O.
PSP2				I/O		Parallel Slave Port data.
RD3/PSP3	22	24	41		ST/TTL(3)	
RD3				I/O		Digital I/O.
PSP3				I/O		Parallel Slave Port data.
RD4/PSP4	27	30	2		ST/TTL(3)	
RD4				I/O		Digital I/O.
PSP4				I/O		Parallel Slave Port data.
RD5/PSP5	28	31	3		ST/TTL(3)	
RD5				I/O		Digital I/O.
PSP5				I/O		Parallel Slave Port data.
RD6/PSP6	29	32	4		ST/TTL(3)	
RD6				I/O		Digital I/O.
PSP6				I/O		Parallel Slave Port data.
RD7/PSP7	30	33	5		ST/TTL(3)	
RD7				I/O		Digital I/O.
PSP7				I/O		Parallel Slave Port data.
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25		ST/TTL(3)	
RE0				I/O		Digital I/O.
$\overline{RD}$				I		Read control for parallel slave port.
AN5				I		Analog input 5.
RE1/$\overline{WR}$/AN6	9	10	26		ST/TTL(3)	
RE1				I/O		Digital I/O.
$\overline{WR}$				I		Write control for parallel slave port.
AN6				I		Analog input 6.
RE2/$\overline{CS}$/AN7	10	11	27		ST/TTL(3)	
RE2				I/O		Digital I/O.
$\overline{CS}$				I		Chip select control for parallel slave port.
AN7				I		Analog input 7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17, 28,40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

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

File Address	Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3
00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h	Indirect addr.(*)
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h		185h	
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h		187h	
08h	PORTD(1)	88h	TRISD(1)	108h		188h	
09h	PORTE(1)	89h	TRISE(1)	109h		189h	
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDATA	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEDATH	18Eh	Reserved(2)
0Fh	TMR1H	8Fh		10Fh	EEADRH	18Fh	Reserved(2)
10h	T1CON	90h		110h		190h	
11h	TMR2	91h	SSPCON2	111h		191h	
12h	T2CON	92h	PR2	112h		192h	
13h	SSPBUF	93h	SSPADD	113h		193h	
14h	SSPCON	94h	SSPSTAT	114h		194h	
15h	CCPR1L	95h		115h		195h	
16h	CCPR1H	96h		116h		196h	
17h	CCP1CON	97h		117h	General Purpose Register 16 Bytes	197h	General Purpose Register 16 Bytes
18h	RCSTA	98h	TXSTA	118h		198h	
19h	TXREG	99h	SPBRG	119h		199h	
1Ah	RCREG	9Ah		11Ah		19Ah	
1Bh	CCPR2L	9Bh		11Bh		19Bh	
1Ch	CCPR2H	9Ch	CMCON	11Ch		19Ch	
1Dh	CCP2CON	9Dh	CVRCON	11Dh		19Dh	
1Eh	ADRESH	9Eh	ADRESL	11Eh		19Eh	
1Fh	ADCON0	9Fh	ADCON1	11Fh		19Fh	
20h	General Purpose Register 96 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h	General Purpose Register 80 Bytes
7Fh		EFh		16Fh		1EFh	
		F0h	accesses 70h-7Fh	170h	accesses 70h-7Fh	1F0h	accesses 70h - 7Fh
		FFh		17Fh		1FFh	

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.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	29, 148
01h	TMR0	Timer0 Module Register								xxxx xxxx	53, 148
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	28, 148
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	20, 148
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	29, 148
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	41, 148
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	43, 148
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	45, 148
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	46, 148
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	47, 148
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	28, 148
0Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	22, 148
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	24, 148
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	26, 148
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	58, 148
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	58, 148
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	55, 148
11h	TMR2	Timer2 Module Register								0000 0000	60, 148
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	59, 148
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	77, 148
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	71, 80, 148
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	61, 148
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	61, 148
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	62, 148
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	110, 148
19h	TXREG	USART Transmit Data Register								0000 0000	116, 148
1Ah	RCREG	USART Receive Data Register								0000 0000	116, 148
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	61, 148
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	61, 148
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	62, 148
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	131, 148
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	125, 148

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	29, 148
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	21, 148
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	28, 148
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	20, 148
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	29, 148
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	41, 148
86h	TRISB	PORTB Data Direction Register								1111 1111	43, 148
87h	TRISC	PORTC Data Direction Register								1111 1111	45, 148
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	46, 148
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	48, 148
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	28, 148
8Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	22, 148
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	23, 149
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	25, 149
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	27, 149
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	81, 149
92h	PR2	Timer2 Period Register								1111 1111	60, 149
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	77, 149
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	77, 149
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	109, 149
99h	SPBRG	Baud Rate Generator Register								0000 0000	111, 149
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	133, 149
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	139, 149
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	131, 149
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000	126, 149

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	29, 148
101h	TMR0	Timer0 Module Register								xxxx xxxx	53, 148
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28, 148
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	20, 148
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	29, 148
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	43, 148
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	28, 148
10Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	22, 148
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	37, 149
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	37, 149
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						--xx xxxx	37, 149
10Fh	EEADRH	—	—	—	—(5)	EEPROM Address Register High Byte				---- xxxx	37, 149
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	29, 148
181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	21, 148
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	28, 148
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	20, 148
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	29, 148
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	43, 148
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	28, 148
18Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	22, 148
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	32, 149
18Dh	EECON2	EEPROM Control Register2 (not a physical register)								---- ----	37, 149
18Eh	—	Reserved maintain clear								0000 0000	—
18Fh	—	Reserved maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

Registrador ADCON1:

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

ADFM: A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

ADCS2: A/D Conversion Clock Select bit (ADCON1 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

Unimplemented: Read as '0'

PCFG3:PCFG0: A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input D = Digital I/O
C/R = # of analog input channels / # of A/D voltage references

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown