6, 3/seis x três)

LEONARDO LOPES CARNELOS

FUTEBOL DE ROBÔS AUTÔNOMOS
PROJETO MECÂNICO E ELETRÔNICO

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obter Graduação em Engenharia Mecânica.

São Paulo

2001

LEONARDO LOPES CARNELOS

FUTEBOL DE ROBÔS AUTÔNOMOS
PROJETO MECÂNICO E ELETRÔNICO

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obter Graduação em Engenharia Mecânica.

Orientador:
Prof. Dr. Marcos Ricardo Pereira Barretto
Departamento de Engenharia Mecânica

São Paulo
2001

Sempre que quisermos descobrir as fronteiras do possível temos que ultrapassar um pouco para o lado do impossível.

Isaac Asimov

Aos meus pais e amigos do C.A.L.C..

Agradecimentos

A minha família, que construiu os fortes alicerces no qual sustento toda minha vida.

Ao orientador deste e outros projetos, mas acima de tudo amigo, Prof. Dr. Marcos Ricardo Pereira Barretto.

Aos companheiros de muitos projetos: Celso Ramos de Souza, Daniel Olioni Andersson, Eduardo Pasianot e Rodrigo de Deus Reinaldo.

Aos demais companheiros de curso por inesquecíveis cinco anos.

Aos professores Ettore, Cabral, Carlos Tu e Celso Furukawa.

Aos técnicos Alceu, César, Fidel, Laércio e Wilson.

E outros muitos pela colaboração com este projeto.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

A automação possibilitou ao homem ampliar seus horizontes a níveis nunca antes imaginados. Os robôs podem executar tarefas difíceis ou perigosas com confiabilidade e precisão altíssimas; porém, necessitam de algoritmos de controle sofisticados, que lhes forneçam as tarefas a serem executadas e tais algoritmos devem prever eventuais ruídos ou falhas no sistema.

Este trabalho se propõe a aplicar a teoria de engenharia mecânica aprendida durante a graduação para o desenvolvimento de tecnologias em vários campos de atuação como o rádio controle, a visão computacional, o *design*, a eletrônica de potência e a mecânica propriamente dita, aplicados em uma competição de futebol de robôs autônomos.

Estes conceitos são os alicerces para soluções em automação de muitos problemas reais, como monitoração de tráfego urbano, monitoração de processos industriais complexos, insalubres ou ainda, a solução de muitos problemas que a sociedade nem desconfia que possam ser automatizados.

1. Introdução

A automação possibilitou ao homem ampliar seus horizontes a níveis nunca antes imaginados. Os robôs podem executar tarefas difíceis ou perigosas com confiabilidade e precisão altíssimas; porém, necessitam de algoritmos de controle sofisticados, que lhes forneçam as tarefas a serem executadas e tais algoritmos devem prever eventuais ruídos ou falhas no sistema.

Este trabalho se propõe a aplicar a teoria de engenharia mecânica aprendida durante a graduação para o desenvolvimento de tecnologias em vários campos de atuação como o rádio controle, a visão computacional, o *design*, a eletrônica de potência e a mecânica propriamente dita, aplicados em uma competição de futebol de robôs autônomos.

Este trabalho é complementar ao trabalho dos outros integrantes de um grupo de pesquisa. O grupo é composto de cinco integrantes: Celso Ramos de Souza, Daniel Olioni Andersson, Eduardo Pasianot, Leonardo Lopes Camelos e Rodrigo de Deus Reinaldo. Este trabalho se propõe a documentar todos os testes, protótipos e resultados referentes ao projeto mecânico e eletrônico dos robôs.

1.1. Motivação

O tema futebol de robôs se trata de uma interação entre diversas áreas da engenharia, e tem como proposta: a cooperação entre robôs, o desenvolvimento de técnicas ligadas à inteligência artificial, a execução de tarefas de percepção visual e o controle em tempo real. Inúmeras aplicações podem ser imaginadas a partir desta idéia, dentre elas destaca-se o controle de linhas de montagem industrial e a monitoração do tráfego urbano.

A *RoboCup* representa pesquisadores de todo o mundo que trabalham com este sistema de controle e promove competições visando intercâmbio e apoio

para tais pesquisas. possui quatro categorias; três delas são disputadas com robôs reais de dimensões diferentes; a quarta categoria é disputada via software, com o auxilio de um simulador.

Um completo histórico sobre futebol de robôs no Brasil pode ser encontrado no site da FIRA, Federação Internacional de Futebol de Robôs Autônomos. As regras variam de acordo com as categorias; as principais, *MiroSot* e *NaroSot* são facilmente encontradas na Internet; além das regras dos jogos são específicos também as dimensões dos jogadores, campo e bola.

2. Definição do problema

Será feito a seguir um aprofundamento nas malhas básicas de controle descritas anteriormente, desmembrando-as e detalhando-as. Primeiramente analisaremos o sistema como um todo e a seguir cada malha individualmente.

2.1. Sistema Futebol de Robô

Está apresentada uma foto ilustrativa de um jogo ocorrido em 1999, pelo campeonato mundial, em Campinas, na categoria da FIRA do projeto.

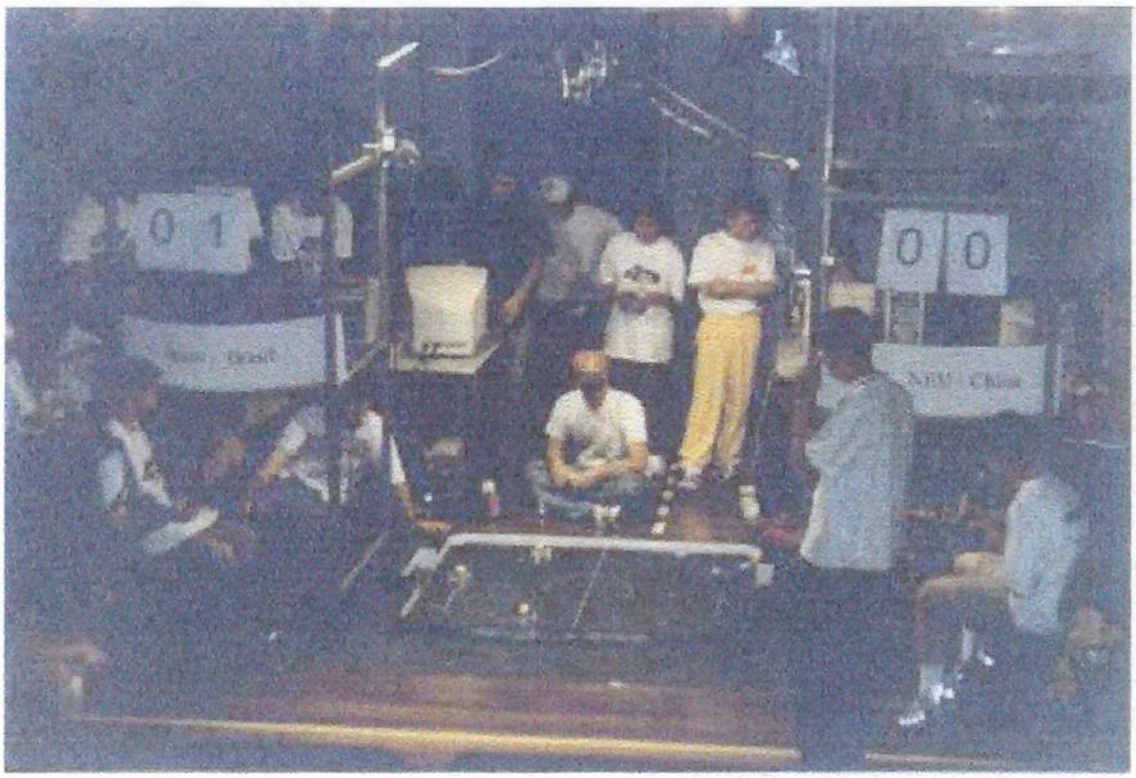


Figura 1 – Foto ilustrativa de um jogo

Tomando o futebol de robôs autônomos como um todo, o grupo identificou as etapas de controle da seguinte maneira:

reconhecimento da imagem – realizado pela câmera de vídeo que capta a imagem do campo fornecendo dados ao computador; este primeiro passo pertence à célula de controle identificada como visão.

processamento da imagem – realizado pelo software, que capta a imagem da câmera, a identifica e a traduz para ser efetuado o controle.

planejamento e controle – também realizado pelo software, nesta etapa encontra-se toda a estratégia do sistema, as tomadas de decisão, o eventual uso de técnicas de memorização como redes neurais, etc.

comandos de transmissão – gerados pelo software, são interpretados pela célula anteriormente intitulada interface computador - robô, ou seja, um transmissor.

atuação no robô – após receber os sinais transmitidos, é efetuado no robô o controle dos motores visando atingir o objetivo pré determinado pela estratégia (geralmente a bola).

A checagem (ou monitoração) dos objetivos é feita pela câmera de vídeo, fechando o sistema.

Abaixo podemos ver um esquema do funcionamento de uma equipe de futebol de robôs autônomos, enfatizando suas malhas.

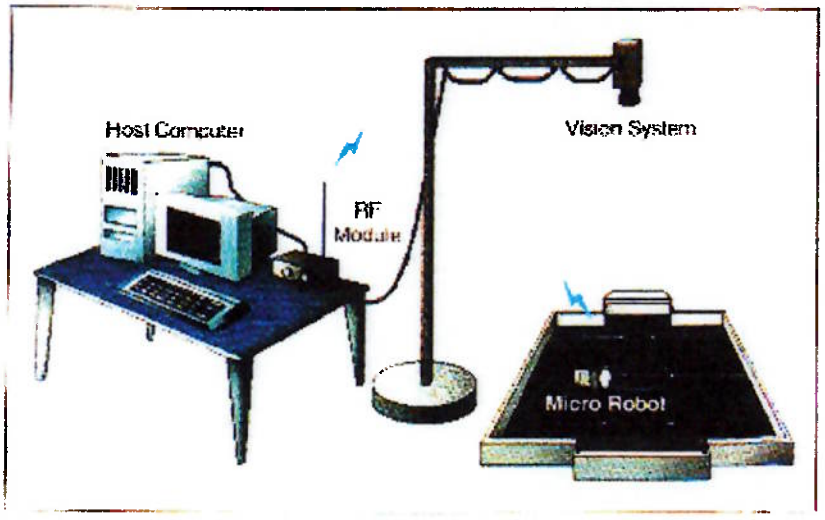


Figura 2 – Esquema de funcionamento

Na próxima figura, temos a divisão do sistema acima detalhado nos moldes clássicos de controle: a unidade central de controle é o software e o objeto a ser controlado (ou planta) é o robô; para efetuar o controle o software dispõe de dispositivos de atuação (bloco de interface ou transmissor) e detecção (visão computacional). Como o usuário não toma decisões no processo, apenas o inicia e finaliza, os dispositivos de comando e monitoração encontram-se "embutidos" no software (unidade central) o que torna o sistema realmente autônomo.

Basicamente, a câmera (sensor) efetua a detecção da posição dos robôs, bola e adversários e passa essas informações ao software, que através de algoritmos de visão computacional (monitoração), passa essa informação através de coordenadas a outro módulo do software de controle, o módulo de inteligência artificial que é quem toma as decisões do sistema (comando), gerando assim, sinais de tensão que passam pela interface computador/transmissor, sendo enviadas via rádio freqüência para os motores elétricos que efetivamente atuam no robô, fazendo com que este se mova de forma ordenada em direção à bola com a intenção de atacar ou defender.

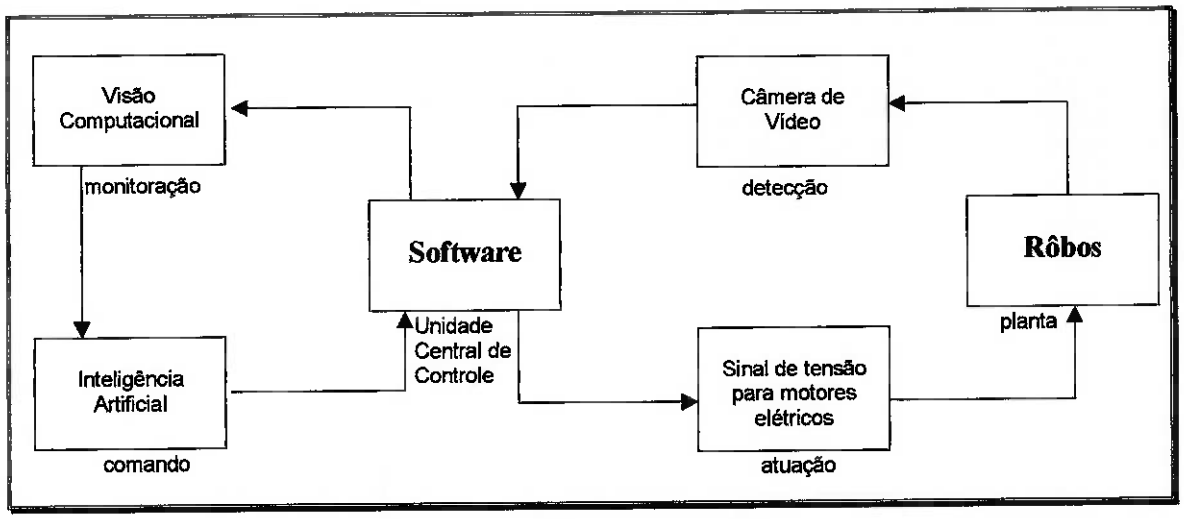


Figura 3 – Módulos de Controle

A partir de agora iremos detalhar melhor cada uma das malhas de controle especificada.

2.2. Controle dos motores

O controle dos motores pode ser realizado de várias formas distintas; uma delas consiste no emprego de servos profissionais associados a sistemas de redução especificamente projetados, o principal problema destes mecanismos é o seu alto custo final, devido principalmente a sua pequena escala de produção.

De forma semelhante, o controle pode ser feito através de *chips* dedicados ao propósito, como o controlador LM629, utilizado pela equipe austríaca presente na 4ª Copa Mundial disputada em 1999 em Campinas e também pela equipe coreana. Este controlador possui, dentre outros dispositivos internos, um gerador de trajetórias, que pode efetuar um controle paralelo; e um gerador de PWM que determina a velocidade a ser aplicada no motor.

Outros integrados podem ser utilizados para os mesmos fins, como o MPC1710A, o L293D, entre outros. Há ainda outra opção para o controle, a geração de um sinal PWM a ser injetado diretamente nos motores; vários e abundantes são os integrados que efetuam tal tarefa, um exemplo é o LM3254.

Um passo a frente estão os coreanos atuais campeões mundiais (equipe RobotIS), que além do controle de velocidade dos motores efetuam um controle de corrente para os mesmos, controlando assim o torque a ser empregado às rodas dos robôs. Este sofisticado controle é comandado por um microcontrolador da família PC, o 80286.

2.3. Interface computador – robô

O controle de servo – mecanismos via conjunto transmissor/receptor de rádio freqüência é uma forma natural e eficaz em termos da relação custo – desempenho para o comando remoto dos robôs. O problema maior seria o enlace computador – transmissor e receptor – controle dos motores; a eliminação de incompatibilidades pode tornar-se complexa.

Esta interface pela porta paralela de um computador pessoal e um transmissor RC Futaba Skysport, com um canal do receptor direcionado para cada motor; ou ainda utilizando a saída serial do computador, como fazem os coreanos da equipe ROBOTIS , enviando as informações em "pacotes" a todos os robôs: cada um dos jogadores identifica se a instrução é endereçada a si e a processa.

Além de utilizar rádios–transmissores prontos, pode-se também projetar e desenvolver um conjunto transmissor/receptor. Estes projetos reduzem o custo final do enlace, embora tornem mais complicada a sua concepção, pois a fabricação de bobinas e os ajustes a serem efetuados requerem tempo e experiência. Porém como será visto mais adiante que o fator monetário da pesquisa foi bastante limitante, o projeto de um conjunto transmissor/receptor se fez necessário.

2.4. Visão computacional

Visão computacional é o meio pelo qual o sistema obtém a localização dos robôs. A distinção entre os robôs de um mesmo time e robôs do time adversários é feita através do emprego de figuras de diferentes cores na parte superior desses.

Devido a grande rapidez com que as posições mudam durante o jogo, e a complexidade que os processos de reconhecimento de imagem envolvem (o que os torna um processo lento), a visão computacional torna-se um ponto de grande relevância; ao ponto de um processo de visão mais rápido tornar os robôs mais competitivos.

A aparelhagem básica para a realização de reconhecimento consiste em uma câmera e uma placa que manda para o computador uma imagem digitalizada. Por exemplo uma câmera JVC3CCD (digital) que possui uma resolução de 320X240 pixels. O sistema utilizado por uma das equipes de 1999 possui uma capacidade de processamento de 30 quadros por segundo, enquanto

que equipes como a atual campeã mundial RobotIS utilizam um processamento de 60 quadros por segundo.

2.5. Software de Controle (Estratégia)

Devido ao fato dos robôs funcionarem de forma autônoma, deve-se procurar formas alternativas de programação que possibilitem ao software a tomada de decisões e a cooperação entre os três jogadores.

Visto que as possibilidades de posicionamento dos robôs jogadores, dos adversários e da bola são infinitos, não se deve restringir o comportamento dos robôs a apenas algumas combinações pré-determinadas de movimentos, esperando que as mesmas supram as necessidades da estratégia.

Para isso pode-se utilizar artifícios como Inteligência Artificial e Redes Neurais Artificiais, ou ainda um banco de dados (memória) com situações já ocorridas e seus desfechos, formando uma árvore de busca binária para tomar decisões (Inteligência Artificial).

Pode-se ainda criar um simulador de estratégias, que preveja o comportamento dos jogadores na teoria, sem a necessidade da utilização do resto do equipamento; isolando este sistema, pode-se detectar e corrigir possíveis imperfeições do mesmo.

2.6. Estrutura montada

Para toda a confecção da equipe, assim como todos os testes realizados que viabilizaram este projeto, foi-se montada toda um infra-estrutura de laboratório, desde o espaço físico ate a instrumentação, que será detalhada nesta seção.

Foram utilizados no projeto dois computadores (PCs), 486 DX4, uma câmera VHS, da Panasonic, uma placa de aquisição de imagens Vídeo Blaster FS200. Outros equipamentos como: multímetros, osciloscópio, *pront-o-boards*,

ferros de solda, alem de uma infinidade de componentes eletrônicos foram utilizados, porém podemos salientar ainda o projeto e confecção de uma fonte reguladora de voltagem de + 5, -5, +12 e –12 volts, feita a partir de uma fonte de computador, além do campo de jogo que será detalhado na seqüência.

Figura 4 – Fonte reguladora de tensão

Como veremos na seqüência do projeto, microcontroladores do tipo PIC, foram utilizados na transmissão via rádio-freqüência, e na geração e interpretação de trajetórias para os robôs, toda a programação foi feita no software MPLab, disponível na Internet, através de um gravador PICSTART Plus.

Como os programas em Assembler passaram por vários refinamentos, os PICs utilizados, foram janelados, ou seja, tinham a possibilidade de serem apagados por luz ultravioleta. Para isso, foi construído um apagador de memórias RAM, utilizando luz ultravioleta.

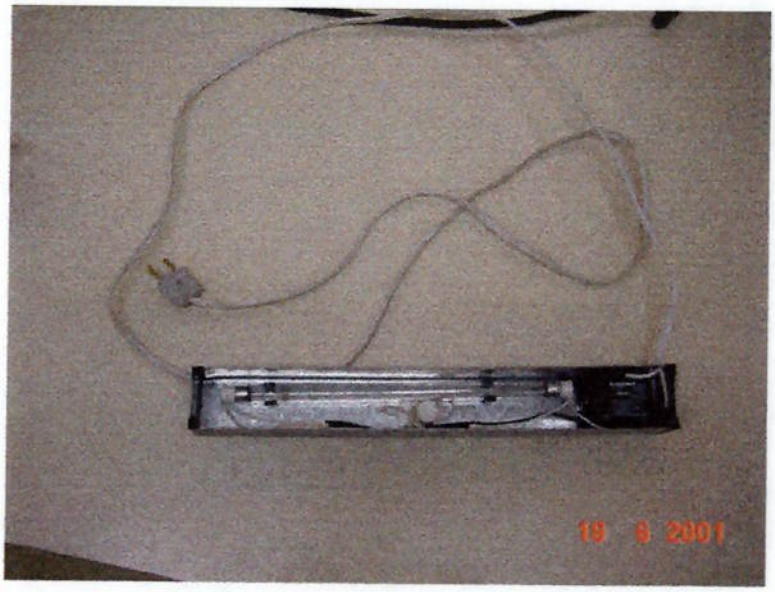


Figura 5 – Apagador de memória RAM

O campo de jogo foi confeccionado em madeira segundo as regras da *Mirosot* (Apêndice 1) com uma chapa de madeira de 10 mm, laterais de MDF de espessura de 25 mm, e altura de 50 mm. A pintura preta, com laterais e linhas brancas, também constam das regras.

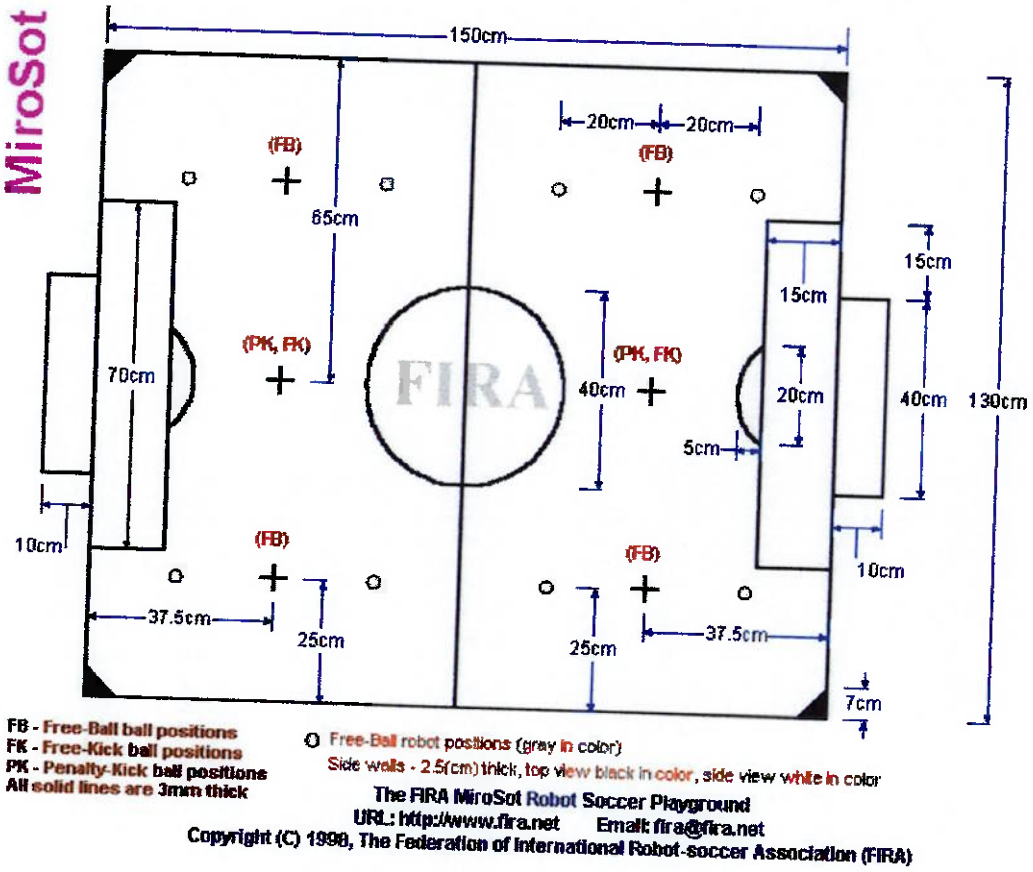


Figura 6 – dimensões do campo

Figura 7 – foto do campo construído

Também foi necessário a construção de estrutura que posicionasse a câmera de vídeo a uma altura de 2,40 m de altura, e esta foi projetada e construída com cantoneiras de 19,3 x 40,9 x 2,9 mm, e parafusos M8 de cabeça sextavada. Alem da câmera, estão presentes na estrutura, presa com mãos francesas de alumínio, duas lâmpadas de 500 W cada uma (holoporte halogênico 500 W IP44) para a iluminação do campo.

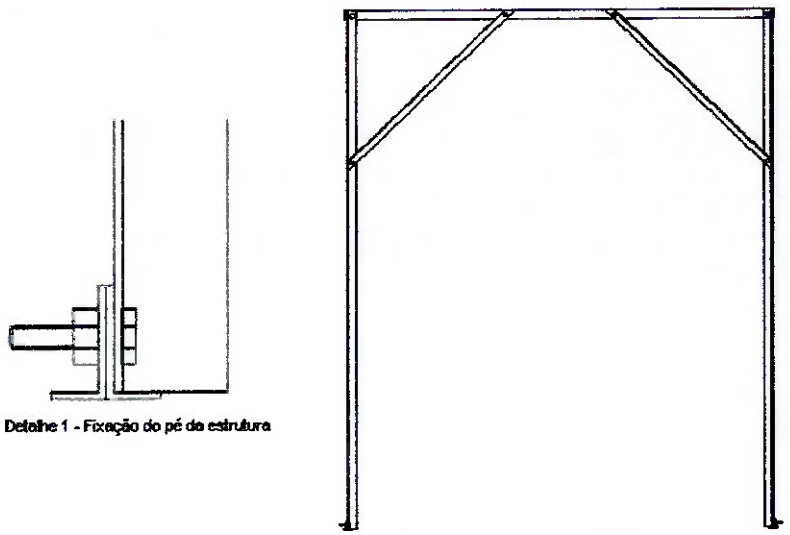


Figura 8 – desenho da estrutura

Figura 9 – foto da estrutura com a câmera

3. Estudo da viabilidade

Nos próximos dois sub itens, são discutidos a viabilidade técnica e financeira do projeto. Vale salientar neste momento que são discutidas apenas a viabilidade do projeto mecânico e eletrônico do projeto. São tomadas como verdade que os outros módulos do projetos são perfeitamente viáveis e já estão funcionando.

Os outros módulos do projeto acima citados são: um sistema de visão computacional que reconhece o campo, jogadores e bola; um software de estratégia e controle que gera trajetórias de defesa e ataque para os robôs, e um sistema de transmissão que envia dados num certo protocolo via rádio freqüência para os robôs.

3.1. Viabilidade técnica

A maior prova da viabilidade técnica do projeto mecânico e eletrônico, é a já existência de equipes de futebol de robôs autônomos que já participam de competições mundiais atualmente, e é exatamente nelas que os integrantes do grupo de trabalho se baseiam para a confecção deste projeto.

Apesar do argumento acima, estão detalhados de uma forma melhor as intenções deste projeto: o maior desafio do projeto mecânico é colocar em um cubo de no máximo 75 mm de lado, um sistema móvel, ágil, rádio controlado e capaz executar trajetórias pré programadas em um PC.

Para isso, as intenções são de usar 2 motores elétricos, acoplados a 2 pequenas rodas, capazes de dar a agilidade requerida destes robôs, no entanto deixando espaço livre para baterias, e circuitos elétricos. Como apenas duas rodas não são suficientes para garantir a planicidade do robôs, estão previstos outros apoios, desprovidos de forcas motoras.

Para isso, sabemos que os motores elétricos encontrados hoje são suficientemente pequenos, para garantir as exigências do projeto, proporcionando ainda a velocidade de resposta e torque também requeridos.

Ainda no corpo mecânico do robô, este deve ser estruturalmente rígido, agüentando o tranco de possíveis colisões com a parede ou adversários.

Em sua estrutura não esta prevista a existência de um mecanismo para o "chute" da bola, pois a existência de solenóide no robô, diminuiria ainda mais o espaço interno do robô, já reduzido, além do aumento do consumo de energia (corrente), necessitando de baterias mais potentes, sem proporcionar um ganho de qualidade muito grande no modelo, com o chute, que pode ser executado, aumentando a rotação de uma das rodas, gerando uma rotação no eixo do robô, que impulsionaria a bola em controle para frente.

Está prevista porém uma ranhura na frente do robô para que a bola se acomode enquanto estiver sob o domínio de tal robô, e que não seja facilmente "roubada" por um adversário.

Dependendo do modelo de robô montado, ou ainda, dos motores elétricos utilizados, estão previstos duas placas eletrônicas, uma de recepção de dados do transmissor, e uma de potência para os motores. A possibilidade de montagem dessas placas em placa universal, ou ate mesmo em circuito impresso, facilmente projetado no Tango 2000, com componentes de micro eletrônica, viabiliza o acomodamento dessas placas num espaço reduzido.

3.2. Viabilidade financeira

Já que a viabilidade técnica não sofre grandes restrições, o grande fator limitante do projeto será a aquisição de equipamentos de ultima geração para a execução deste projeto. Neste projeto iremos priorizar a total redução de custos, o projeto ficando limitado a utilizar alguns equipamentos usados, ou ainda ultrapassados ou que estejam a disposição para utilização.

Para citar um exemplo, a equipe atualmente campeã mundial (RobotIS da China) utiliza motores DC Maxon com blindagem especial contra

ruídos eletromagnéticos que possivelmente poderiam alterar a trajetória correta do robô. Foi feito um orçamento destes motores junto a representante suíça da Maxon Motores, e o valor para a importações de 8 unidades desses motores ficou em por volta de U$1000,00, sendo tatalmente inviável.

Outro exemplo é a existência de placas de processamento de imagem, com o poder de analisar mais de 100 quadros por segundo, um orçamento foi feito junto a Motron, e o valor desta placa estaria em torno de U$3000,00, inviabilizando também sua utilização. A placa a ser utilizada e que está disponível, é ultrapassada e com sorte, conseguirá processar 5 quadros por segundo.

Podemos concluir portanto que este trabalho se propõe a montar uma equipe de futebol de robôs autônomos, não para ser campeã mundial, mas para aplicar e desenvolver todos os conceitos de engenharia envolvidos, devido à sua verba reduzida.

4. Apresentação de possíveis soluções

A partir de agora serão apresentadas algumas possíveis soluções para a confecção de uma equipe de três jogadores (entre eles um jogador diferenciado, o goleiro), serão atreladas a cada solução mecânica, uma solução eletrônica que requer os requisitos de projeto, tanto mecânico quanto de transmissão de dados via radio freqüência. Serão apresentados quatro soluções.

4.1. Solução 1 – Estrutura de Arame

Uma das possíveis soluções seria a utilização de um robô com uma estrutura simples de arame (aço), que daria apenas a sustentação necessária para dois motores elétricos de brinquedo (carrinhos de autorama), a energia elétrica embarcada seriam quatro pilhas AA (6 volts) que serve para alimentar diretamente os motores elétricos, a transmissão é atarvés de um par transmissor/receptor já pronto, também retirado de um brinquedo.

4.2. Solução 2 – Carrinho de Lego

Outra solução é montar um carrinho, utilizando os motores do brinquedo Lego, e suas peças para montar a carenagem, e é controlado através de dois pares de transmissor/receptor de freqüências diferentes, e o sinal de tensão ligado diretamente nos motores.

4.3. Solução 3 – Motores DC

Uma solução é a utilização de dois motores elétricos ligados diretamente a duas rodas de metal, com revestimento de borracha, com uma carenagem de alumínio, controlados por dois sinais de RF de freqüências diferentes, e todo o processamento seria feito no PC.

4.4. Solução 4 – Motores de passo

Pode ser utilizado ainda dois motores de passo, um para cada roda, numa carenagem de metal, só que agora devido a diferente alimentação dos motores, será necessário um circuito de controle e potência dos motores de passo.

5. Detalhamento das Soluções

Neste tópico, pretende-se detalhar e documentar melhor todos as soluções propostas, serão demonstrados todos os desenhos, e circuitos de projeto, para análise de melhor desempenho.

5.1. Solução 1

A solução 1 proposta possui uma estrutura de arame para a sustentação de dois motores elétricos (motor DC), esta estrutura de arame está soldada com estanho e tem o formato de uma gaiola, trazendo o aspecto de carrinho, dando a este a estrutura necessária para se locomover com equilíbrio.

Um conjunto de rodas de borracha, eixo, coroa, pinhão e motores será utilizado para a confecção desta solução todos serão retirados de um brinquedo (carrinho de autorama), não necessitando portanto nenhum projeto de suas partes, apenas a adaptação à gaiola (carenagem) de aço.

O conjunto transmissor/receptor utilizado também foi retirado de um brinquedo, portanto os circuitos tanto de transmissão de dados, como de potência já estavam prontos, não necessitando de maiores projetos, apenas de algumas adaptações. Abaixo segue uma foto do protótipo montado:

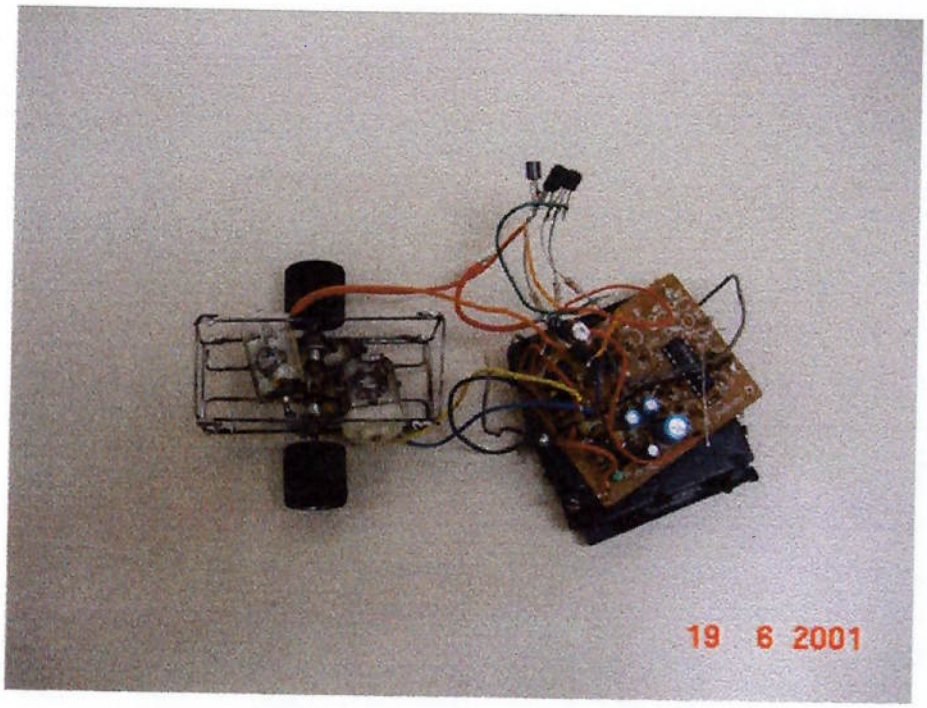


Figura 10 – Foto do protótipo 1

Uma das poucas adaptações necessárias foi na interface computador/transmissor, pois o desejo era que o computador controlasse o movimento do robô. Para isso foi feito uma rotina em linguagem C, que mudasse o sinal de tensão da saída paralela do computador.

No protótipo foram utilizados quatro bits da saída paralela, dois para cada motor, significando cada um, movimento uniforme em cada sentido, como demonstra a tabela abaixo:

Bits de saída	Movimento
1	Motor direito para frente
2	Motor esquerdo para frente
3	Motor direito para trás
4	Motor esquerdo para trás

Tabela 1 – Manobras do protótipo 1

Com apenas estes 4 bits de controle, foram conseguidos as 6 manobras para o robô abaixo descritas.

Palavra de comando	Manobra
00000011 (1 e 2)	Carro para frente
00001010 (2 e 4)	Carro para trás
00001001 (1 e 4)	Curva para esquerda (centro do carro como eixo)
00000110 (2 e 3)	Curva para direita (centro do carro como eixo)
00000001 (1)	Curva para esquerda (roda oposta como eixo)
00000010 (2)	Curva para direita (roda oposta como eixo)

Tabela 2 – Palavras de comando do protótipo 1

A palavra de comando corresponde ao número em binário que corresponde à saída da porta paralela, ou seja, se o desejado é mover o carro para frente, devemos ativar (por em alta), os bits 1 e 2, que corresponde ao número em binário: 00000011, que em decimal vale 3, ou seja, o algoritmo feito em linguagem C, deve mandar a palavra 3 para a saída paralela. Podemos notar que com a resolução de oito bits da saída paralela, temos disponíveis 256 palavras de comando.

Da saída paralela do computador até o transmissor de brinquedo, foi necessário um circuito de potência, chaveamento e isolamento. Como o sinal de tensão do computador é muito baixo, foram utilizados dois transistores, ligados num esquema de amplificação de corrente para na seqüência, o sinal acionar um relê, que além de chavear corretamente o sinal, isola o transmissor da saída do computador, impedindo uma interferência indesejada destes.

Abaixo segue a foto do circuito improvisado, assim como do transmissor, e o protótipo montado.

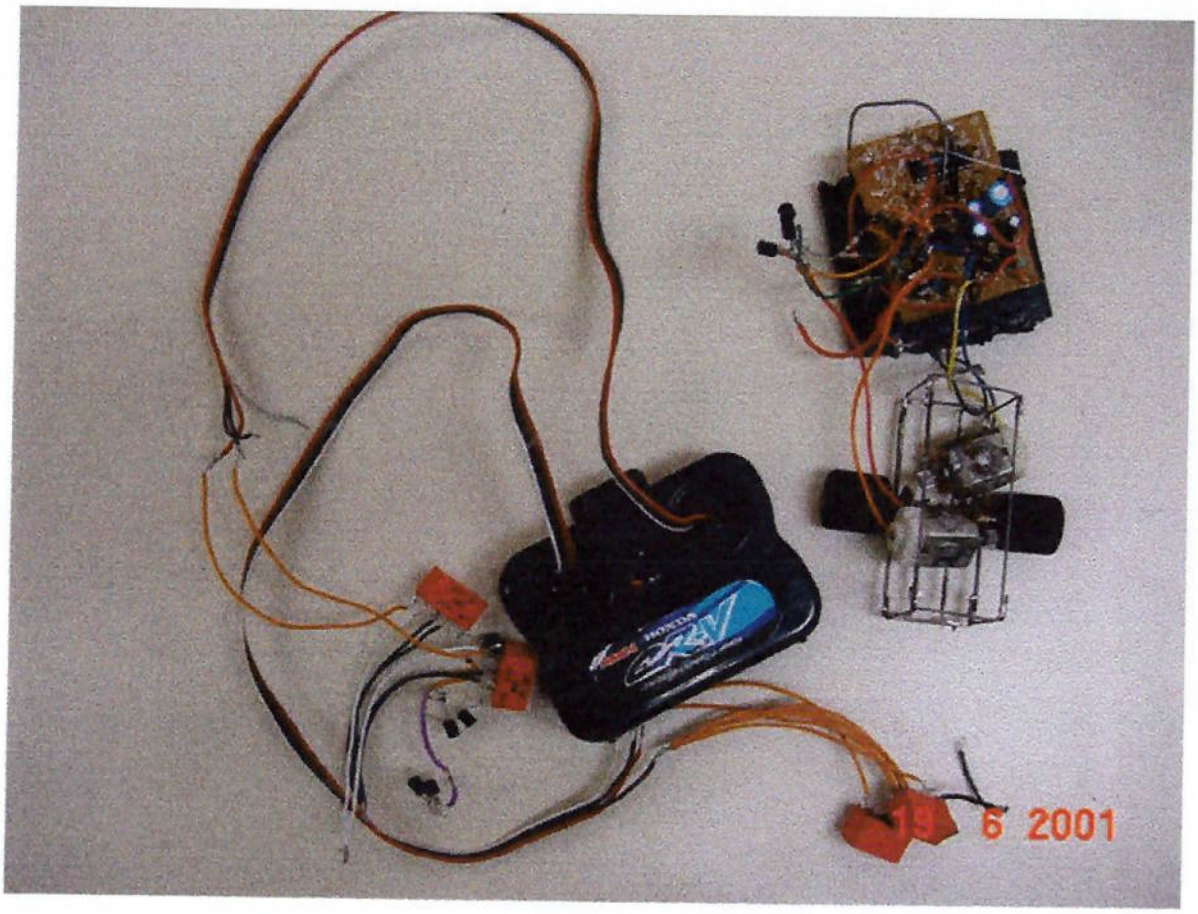


Figura 11 – Foto mostrando o circuito de transmissão

O desempenho deste protótipo foi satisfatório na parte de transmissão, apesar da utilização de um conjunto transmissor/receptor já pronto. As partes mecânicas reagiram de maneira esperada e este protótipo também foi muito útil nos primeiros testes de visão computacional.

5.2. Solução 2

A solução 2 se propôs a testar um transmissor de duas freqüências, utilizados para chavear os dois motores, portanto este foi montado com motores Lego, bem como uma carenagem de peças Lego.

As freqüências utilizadas foram de 453 e 315 MHz, e uma característica destes transmissores era a impossibilidade de transmitir em nível alto por muito

tempo, sendo necessário uma rotina em linguagem C que chaveasse a saída paralela do computador em nível alto.

Com a utilização de motores Lego, não foi necessário um circuito de potência, já que tais motores não consumiam muita corrente das quatro pilhas AA (totalizando 6 volts).

5.3. Solução 3

A solução 3 se utilizou do mesmo tipo de transmissor da solução dois, com o diferencial que esta utilizou, os motores Maxon, com blindagem contra ruídos eletromagnéticos. Para a confecção deste protótipo, foram utilizados perfis em "U" de alumínio, com parafusos M2 para a fixação dos motores, um oposto ao outro.

Devido ao seu tamanho aumentado, não seria possível o engastamento direto das rodas de alumínio confeccionadas diretamente no eixo dos motores, sendo necessária uma redução.

Esta redução foi feita com engrenagens de um brinquedo (Máquina de Costura da Estrela), e reduziam na proporção 3:1. Para não haver maiores vibrações no eixo, foram adaptado à carenagem dois rolamentos que serviram de mancais para o eixo coroa/roda.

Outros rolamentos foram utilizados nas extremidades inferiores, para dar ao carrinho o apoio necessário para sua estabilidade no chão.

Com a utilização destes motores, fez-se necessária a utilização de uma bateria de 9 volts, que alimenta os motores e o circuito de recepção. Foi necessária a utilização de um 7805 para estabilizar a tensão em 5 volts parra o circuito.

O sinal recebido dos rádios, para chavear os motores, teve um tratamento especial: passava por um diodo, para isolar os motores dos rádios, passava por um transistor (BC05) que funcionava como chave, jogando os 9 volts da bateria direto no motor, e passava ainda por um transistor de potência (TIP41), que funcionava como amplificador de corrente.

Uma foto do protótipo montado está apresentada a seguir:

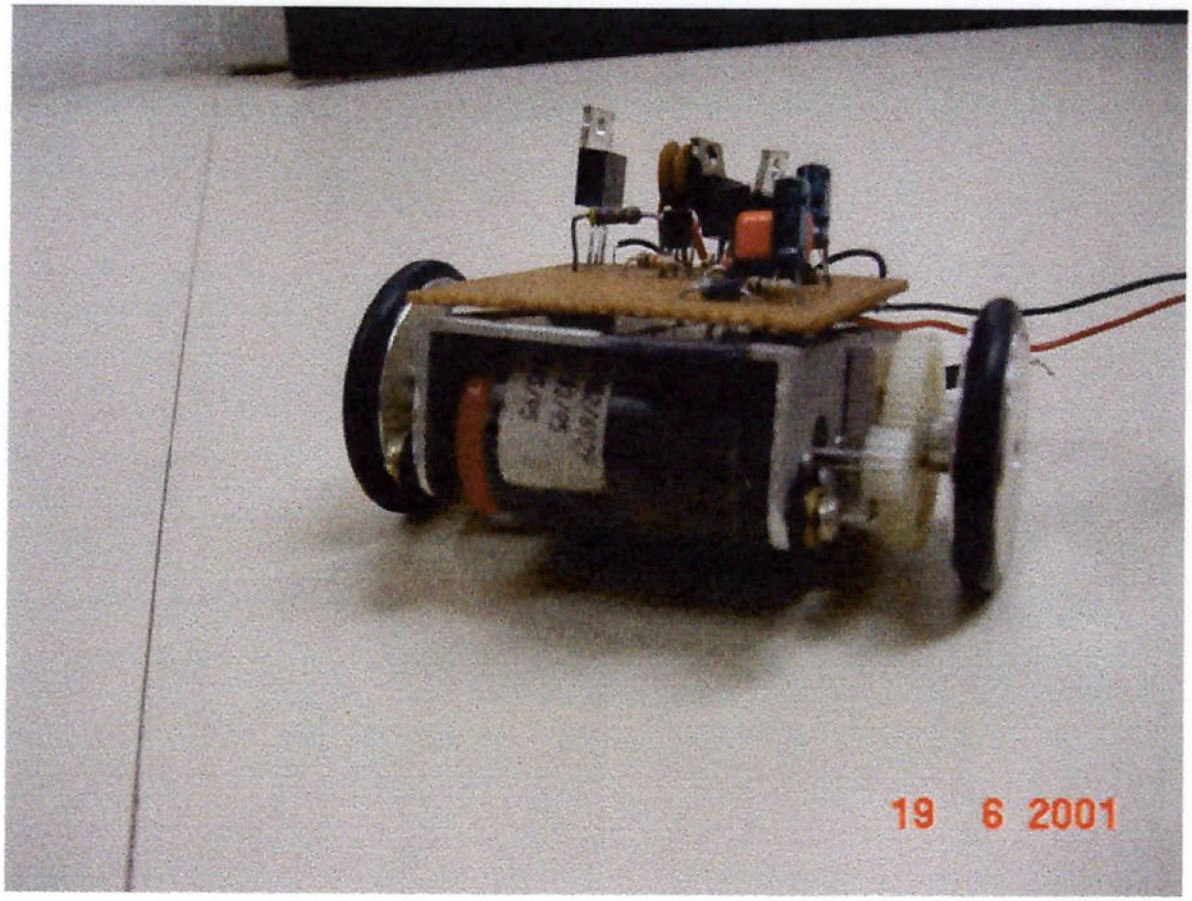


Figura 12 – Foto do protótipo da solução 3

A controlabilidade era comprometida pelos motores usados que fora utilizados, que respondiam diferente a um mesmo nível de tensão, comprometendo trajetórias simples como uma linha reta.

5.4. Solução 4

A solução quatro tem um foco totalmente diferente das demais, com a utilização de um transmissor de uma só freqüência e com a utilização de motores de passo. Os projetos mecânico e eletrônico ficaram bem distintos dos demais.

Esta solução prevê a utilização de um transmissor de um canal, que transmite um pacote de oito bits, que podem ser utilizados para o controle dos motores, que nesta solução são de motores de passo.

No protótipo montado, a estrutura metálica (aço) montada adaptou-se ao formato de quadrado do motor utilizado. A rodas foram torneadas com NYLON, engastadas diretamente no eixo, em sua volta foram utilizados a tira de borracha para maior aderência.

Ainda foram utilizados, parafusos M4, além de rolamentos na sua parte inferior para completar os pontos de contato com o chão, já que só existem duas rodas motrizes.

Com a utilização dos oito bits de controle, foi possível criar um protocolo de controle dos motores de passo. Porém, para o seu correto funcionamento foi necessária a confecção de um circuito de controle e potência dos motores de passo, feitos com os circuitos integrados L297/L93, comerciais, especializados para esta aplicação.

Pronto o circuito, foi detectado que este junto com os motores, consumia por volta de 400 mA, requerendo portanto uma bateria embarcada mais robusta. Esta foi projetada com 4 pilhas de 1,2 volts, totalizando os 4,8 (aproximadamente 5) necessários para o circuito TTL. Em se tratando de uma bateria recarregável de potência, a sua carga tinha que ser lenta, sendo feita a cinco volts e a 120 mA.

Abaixo segue uma foto do protótipo montado:

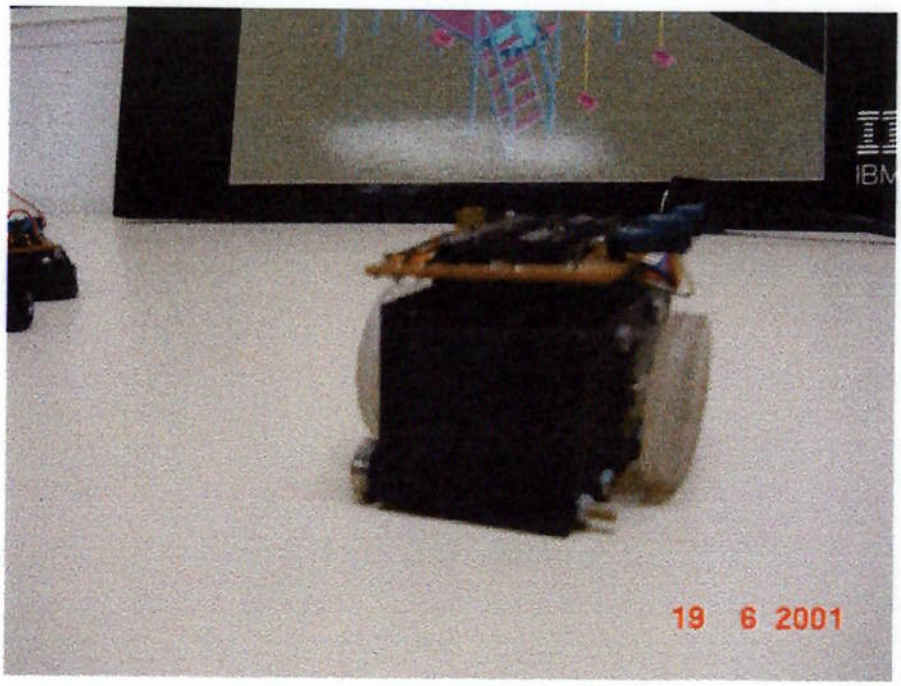


Figura 13 – Foto do protótipo da solução 4

Os problemas observados neste protótipo foram a baixa velocidade de rotação do motor de passo.

6. Escolha da melhor solução

Para a escolha da melhor solução, foi adotado a técnica de matriz de decisão, com os critérios de seleção abaixo descritos, salientando ainda que o desempenho dos protótipos, e o custo de fabricação foram os critérios que receberam o maior peso.

- Custo de fabricação: este critério leva em conta o preço de compra, mesmo que em quantidade, das partes que compõe o robô, ou ainda dos equipamentos necessários adquirir para a confecção de tal; como já foi dito, o custo de fabricação recebeu um dos mais altos pesos.
- Facilidade de fabricação: este critério leva em conta a simplicidade das partes integrantes do robô, ganhando a maior nota o robô que não tiver esquemas de fabricação ou mesmo montagem, considerando a inadequação da oficina disponível para pecas de tamanho reduzido.
- Robustez: foi analisado se a solução tinha robustez suficiente para agüentar o tranco de uma colisão com a parede ou com um adversário, sem sofrer danos, ou ainda no caso do goleiro, ser arrastado para dentro do gol.
- Velocidade: neste critério estão inclusos tanto a velocidade final atingida pelos robôs, como a rapidez com que estes atingem esta velocidade (aceleração)
- Torque: a intenção deste critério é a avaliação do poder de vencer obstáculos, como o goleiro, numa possível disputa de bola.
- Tempo de resposta: este critério pretende avaliar o tempo de resposta, minimizando as defasagens entre a geração da trajetória no computador, passando por todas as interfaces, até a realização desta trajetória pelos robôs
- Controlabilidade: este critério mede a controlabilidade dos robôs, ou seja, a confiabilidade entre a trajetória gerada pelo computador e a mesma realizada pelos robôs, sem a necessidade de efetuar-se maiores ajustes de calibração, ou semelhantes.

- Consumo de potência: este critério avalia a quantidade de energia consumida tanto pelos motores escolhidos, como pelos circuitos que os controlam, limitando assim a potência da bateria embarcada, tendo a possibilidade de diminuir o seu tamanho.
- Aplicabilidade: este critério se fez necessário devido a realização de alguns testes para verificação do perfeito funcionamento principalmente da transmissão, não sendo aplicáveis a uma competição real.
- Adaptação às dimensões: este critério mede a melhor adaptação dos equipamentos necessários ao pequeno espaço interno disponível, medindo portanto a maleabilidade do sistema.
- Manutenabilidade: este critério avalia a facilidade de manutenção dos equipamentos do robô, ou ainda da capacidade de divisão em módulos das partes do robô, que seriam facilmente trocadas no caso de defeito no momento da competição.
- Desempenho do protótipo: este critério recebeu o maior peso devido à possibilidade de fabricação de protótipos para teste.

Definidos os critérios de avaliação, o próximo passo foi montar a matriz de decisão, atribuindo assim, as notas de cada critério para cada solução. As notas atribuídas foram: 10, 7.5, 5, 2.5 e zero, de acordo com o desempenho no critério.

Critérios	Pesos	Solução 1		Solução 2		Solução 3		Solução 4	
Custo de Fabricação	5	10	50	7,5	37,5	5	25	7,5	37,5
Facilidade de Fabricação	3	7,5	22,5	10	30	7,5	22,5	5	15
Robustez	3	0	0	2,5	7,5	7,5	22,5	10	30
Velocidade	3	5	15	5	15	10	30	7,5	22,5
Torque	2	5	10	7,5	15	7,5	15	10	20
Tempo de Resposta	2	5	10	5	10	10	20	7,5	15
Controlabilidade	3	5	15	5	15	2,5	7,5	10	30
Consumo de Energia	3	10	30	5	15	7,5	22,5	2,5	7,5
Aplicabilidade	2	2,5	5	2,5	5	7,5	15	10	20
Desempenho do protótipo	5	2,5	12,5	2,5	12,5	7,5	37,5	10	50
Manutenabilidade	3	2,5	7,5	7,5	22,5	10	30	5	15
Adaptação às dimensões	3	10	30	2,5	7,5	7,5	22,5	5	15
TOTAL	**37**		**207,5**		**192,5**		**270**		**277,5**
Média			5,61		5,20		7,30		7,5

Como podemos observar, a solução 4 foi a que teve melhor desempenho na matriz, obtendo uma média de 7.5, sendo portanto a solução escolhida para a confecção da equipe.

A partir da melhor solução escolhida, serão definidos na seqüência alguns parâmetros do projeto final a ser implementado e estes serão devidamente divididos entre projeto mecânico e projeto eletrônico.

8. Projeto Mecânico

O protótipo da solução 4, foi confeccionado com chapas de alumínio, e tinha as exigências de manter as dimensões previstas nas regras (um cubo de 75 mm), fixar os motores de passo, proporcionar estabilidade para a locomoção dos mesmos e deixar espaço para o circuito eletrônico de controle do robô.

Visando algumas melhorias de robustez, espaço interno e estabilidade, algumas alterações foram feitas.

No protótipo foram utilizados chapa de alumínio 2 mm de espessura pintadas de preto para evitar a oxidação com o tempo. Para a confecção essa chapa foi alterada para aço galvanizado bitola #20 (0,95 mm de espessura), igualmente dobradas e soldadas nas juntas, para diminuir o peso dos robôs, porém mantendo a proteção contra corrosão.

Os motores de passo quadrados foram substituídos por outros redondos e menores, porém o funcionamento permaneceu o mesmo, com quatro bobinas com terra comum. A sua fixação permaneceu a mesma com parafusos M3 na lateral do robô.

As rodas de NYLON com revestimento de borracha permaneceram iguais à do protótipo, com alteração apenas do tamanho do furo do engaste do eixo, que é menor. O engaste foi feito com interferência (com retirada de material).

Os rolamentos de apoio não possuem posição definida, podendo ser fixados numa faixa para melhor regulagem de altura, proporcionando uma maior estabilidade ou robô em movimento.

A disposição das chapas de aço galvanizado prevê a colocação de uma bateria de 9 volts convencional no espaço compreendido entre os motores de passo e a altura do robô prevê o espaço para uma placa eletrônica de controle dos motores de passo. O isolamento entre a carenagem do motor e a placa eletrônica é com uma tira de borracha colada à placa e encaixada no robô.

Com exceção do goleiro, os os outros dois jogadores possuem uma reentrança em suas carenagens para a condução da bola.

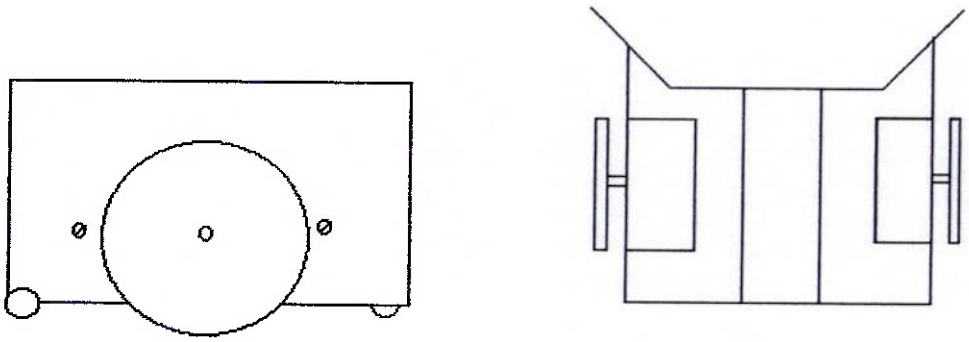


Figura 14 – Desenho dos robôs

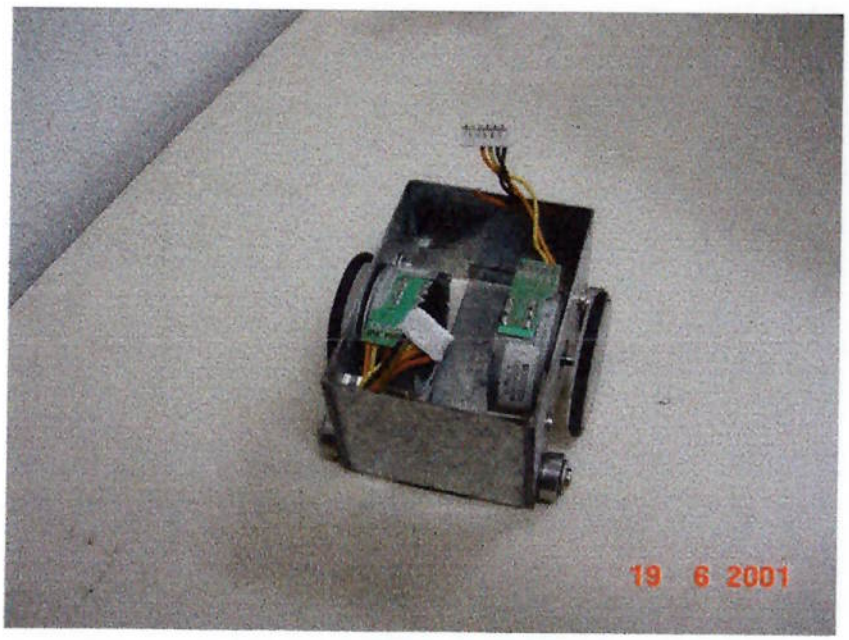


Figura 15 - Foto do robô montado

9. Projeto Eletrônico

O projeto eletrônico está voltado para de alguma forma controlar o robô em questão, tornando real as trajetórias geradas pelo programa de estratégia através da movimentação dos motores de passo.

Para isso um circuito eletrônico foi projetado, contendo o receptor de um canal, e seu microprocessador (o funcionamento do receptor está detalhadamente explicado na trabalho de formatura na referência), e os drivers dos motores de passo.

Para o controle das trajetórias geradas, um protocolo de comunicação foi criado com os 8 bits transmitidos pelo computador e que devem ser identificados pelos robôs. Dois bits para selecionar o robô (robô 1, 2 ou 3), um bit para selecionar a ação do robô (movimentar ou permanecer parado), dois bits para identificar o tipo de manobra a ser efetuada (rotação para direita ou esquerda e translação para frente ou trás) e os últimos três bits para indicar o tempo de efetuação da manobra. Este protocolo será melhor explicado na seqüência.

O sinal ao sair do PIC de recepção, deveria ir direto para um outro PIC que processa este protocolo e gerava as ondas necessárias para os motores de passo realizarem a trajetória. Porém a corrente de saída do PIC não é suficiente para excitar a entrada de outro PIC, sendo necessário a utilização de um *buffer* (74LS244P), para que os PICs se comunicassem.

Geradas as ondas neste segundo microcontrolador, o sinal auxiliado de um oscilador 555, passa pelo driver de controle de motor de passo (L297), e na seqüência pelos drivers de potência de motor de passo (L293B), chegando finalmente aos motores.

A freqüência de oscilação do circuito com o 555 é alterada através de um *trimpot*, alterando portanto a velocidade de rotação dos motores, lembrando que quanto maior a velocidade do motor de passo, menor o seu curso e a sua precisão, pois a taxa de perda de passos aumenta com o aumento da freqüência.

A seguir podemos ver o caminho do sinal descrito num fluxograma, desde o módulo de recepção até os motores:

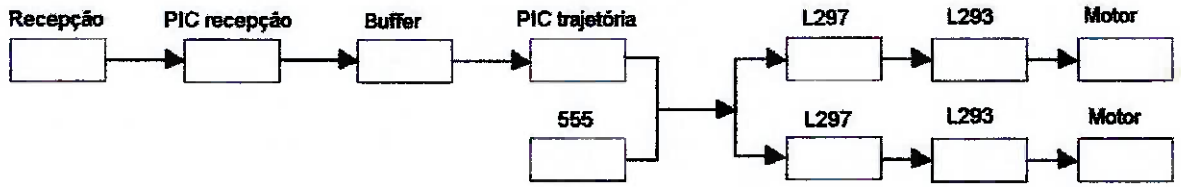


Figura 16 - Fluxograma explicando o caminho do sinal

A placa eletrônica projetada necessitou de uma bateria de potência maior que as convencionais, pois o circuito consume uma corrente de aproximadamente 400 mA, na tensão de 5 volts. Para isso foi produzida uma bateria de 1800 mA.h e 4,8 volts (4 unidades de 1,2 volts em série).

No projeto da placa, para um melhor funcionamento foi decidido separar fisicamente os módulos de controle e potência. Para isso foram confeccionadas duas placas unidas por conectores. Abaixo seguem o projeto das placas e da montagem final.

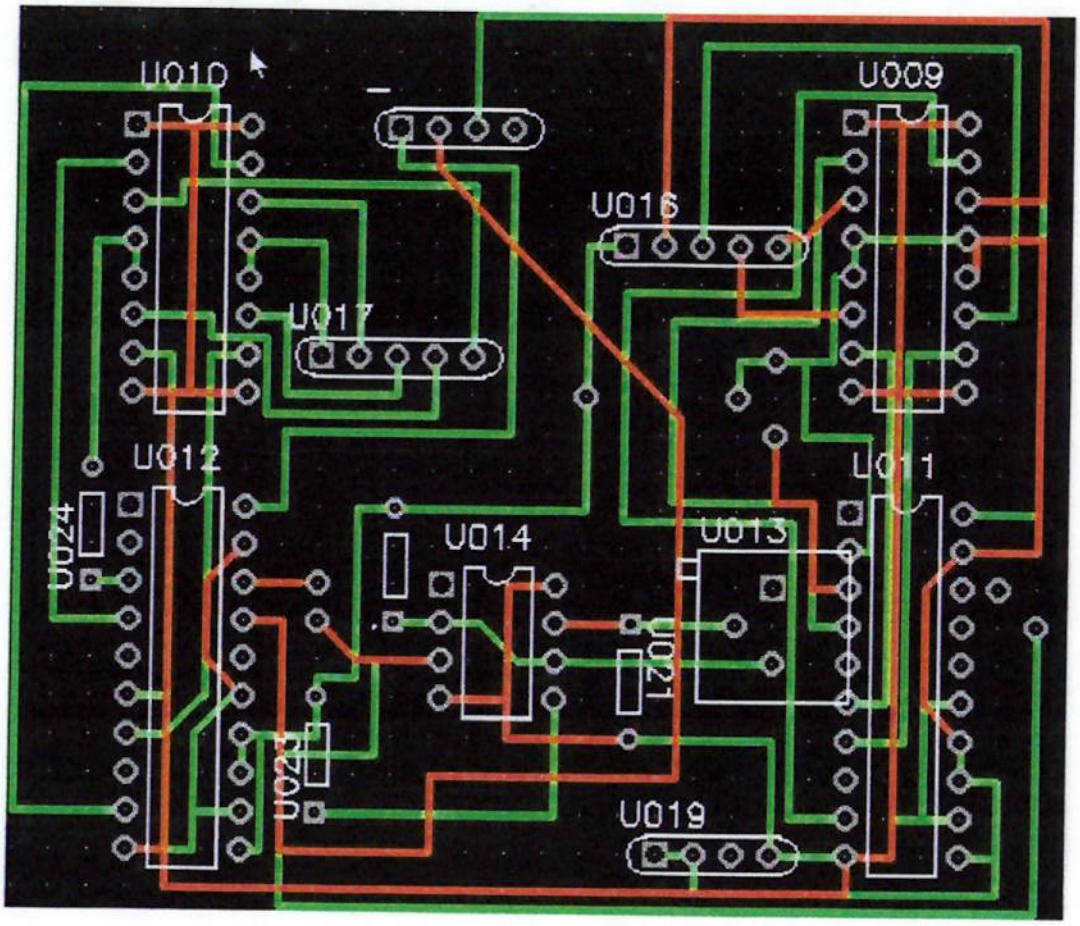


Figura 17 - Layout do circuito 1

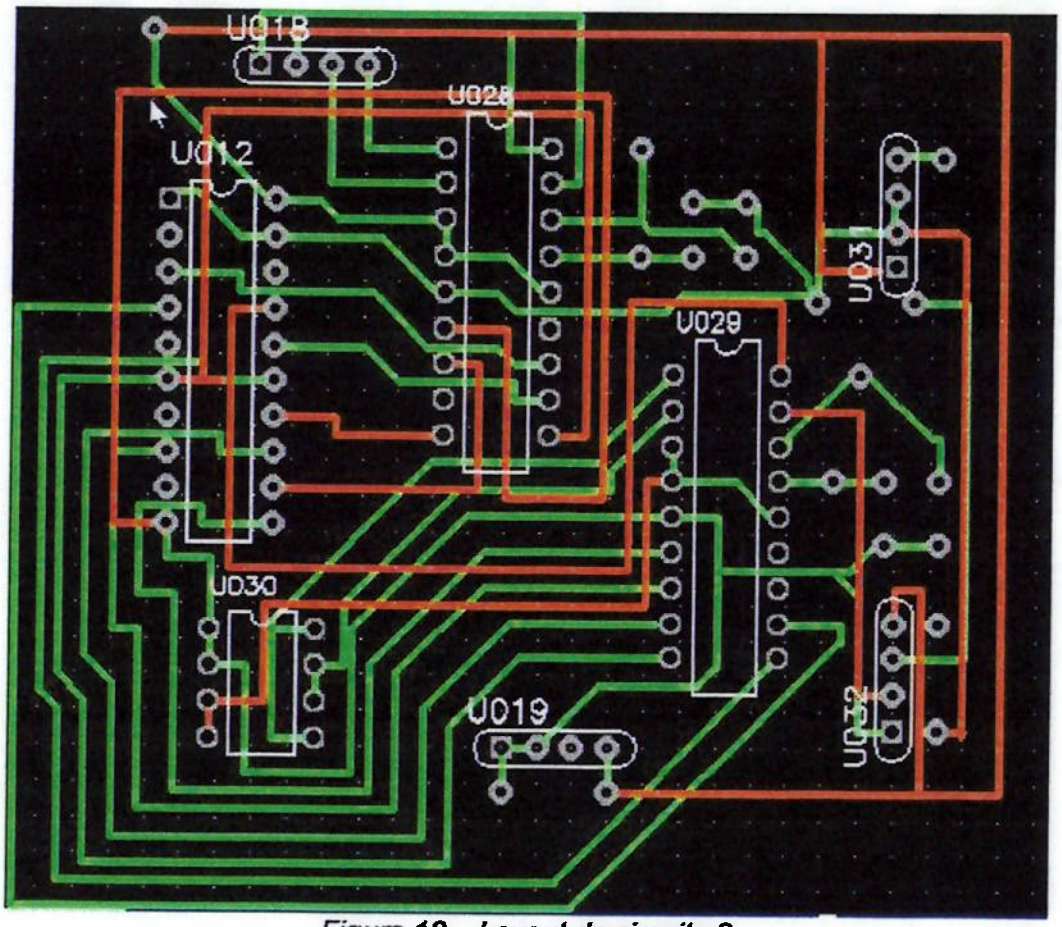


Figura 18 – Layout do circuito 2

O desenho da placa foi feito no software Tango, onde as trilhas avermelhadas são no topo da placa, e as trilhas verdes estão no verso da placa.

A seguir estão listados todos os componentes utilizados na placa:

Quantidade	Componentes
04	Resistor 1 MΩ
01	Resistor 260 Ω
01	*Trimpot* 0 a 10 kΩ
01	Capacitor 470 µF
01	Capacitor 10 pF
01	Capacitor 4,7 µF
01	Capacitor 25 nF
04	Capacitor 22 pF
02	Cristais 3,540 MHz
01	Botão selecionador
04	Conectores 4 pernas
01	*Buffer* – 74LS244P
02	Microcontrolador – PIC 16c54
01	CI – 555
02	Driver controle – L297
02	Driver potência – L293B
01	Receptor

Tabela 3 – Lista de material

A seguir podemos ver três figuras, que mostram o robô montado. Na primeira é possível notar as dimensões do robô, das placas, da bateria, e do espaço interno destinado à bateria do robô. Na segunda, tem-se idéia de como é o robô montado e na terceira destaca-se a posição das placas e a altura final do robô.

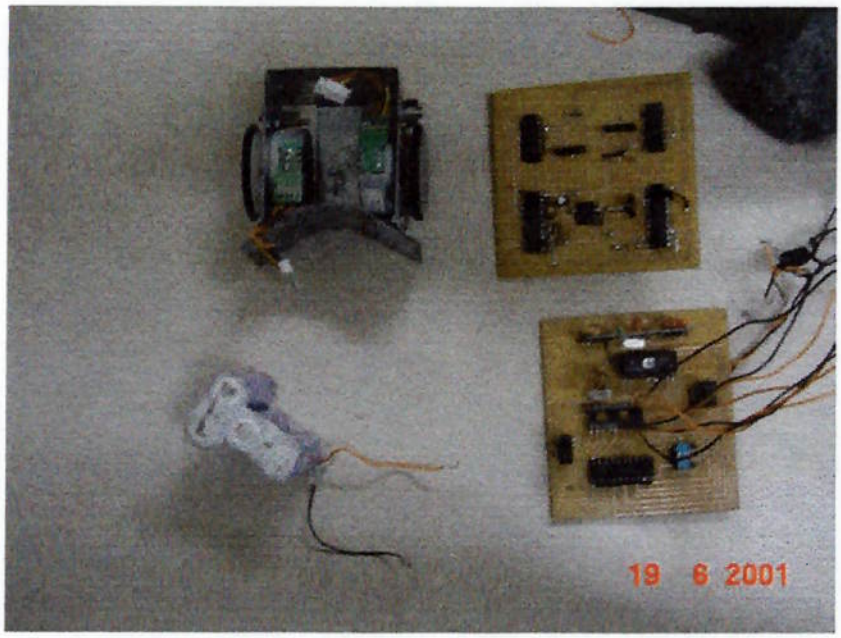


Figura 19 - Foto mostrando robô, bateria e placas

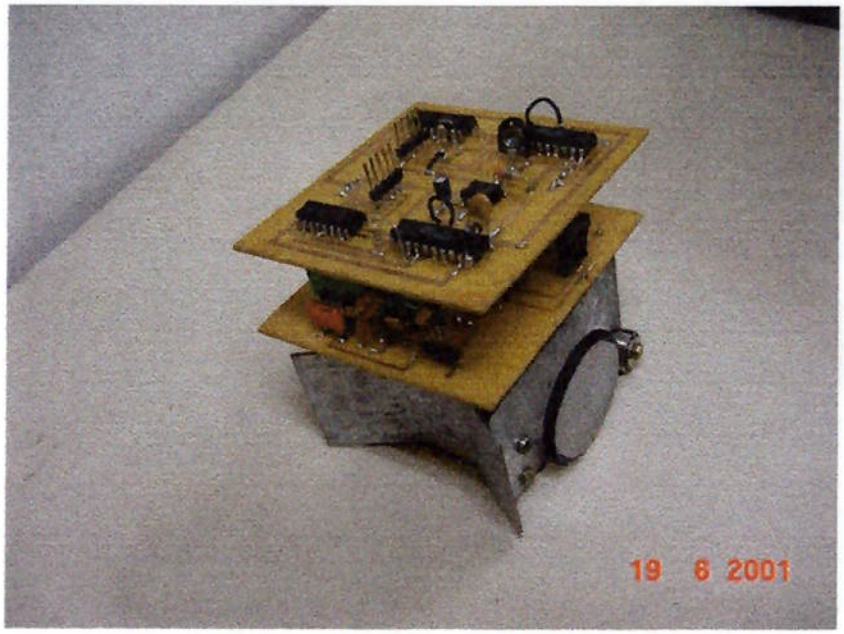


Figura 20 - Foto mostrando a montagem final

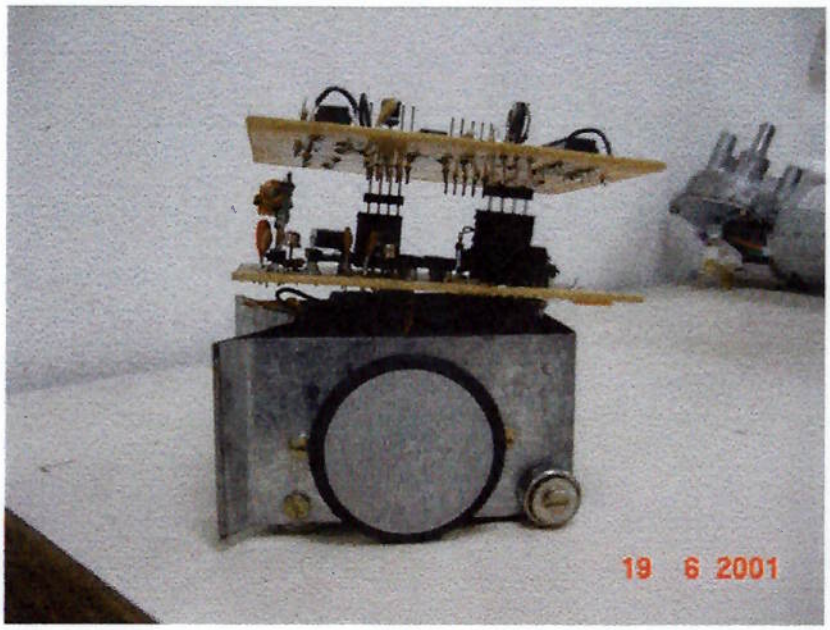


Figura 21 - Foto mostrando a altura excedente do robô montado

Como pode ser visto pelas figuras a confecção das placas não atingiu os padrões de qualidade exigidos, e o robô ficou com uma dimensão maior que os 75 mm de altura previstos pelo regulamento, impossibilitando a participação deste em qualquer competição.

Também foi constatado a imprecisão do circuito oscilatório com o 555, além de problemas de contato com os conectores, tornando o circuito passível de muitas falhas para utilização em qualquer competição.

Por conta dos motivos descritos acima, uma nova versão do circuito foi projetada.

9.1. Projeto Final

Para o projeto final, algumas alterações foram propostas para simplificar a placa eletrônica de controle dos motores de passo e torná-la mais confiável para uma competição. A proposta foi de aproveitar o microcontrolador embarcado, que realiza as tarefas de recepção do sinal para efetuar tarefas de outros circuitos integrados embarcados.

Como o microcontrolador selecionado (PIC 16c54c) possui 4 sinais de entrada (PortA) e 8 sinais de saída (PortB), estes poderiam ser aproveitados para controlar os dois motores de passo por robô, já que cada um precisa de 4 sinais de controle.

Este mesmo já tinha sido selecionado pelo módulo de recepção por possuir 4 entradas e 8 saídas, o mínimo necessário para o controle dos motores.

O programa do microcontrolador foi alterado para que após o módulo de demultiplexação do sinal de recepção, o sinal de cada bit do protocolo criado pela transmissão gerasse um sinal sincronizado dos sinais de tensão para controle dos motores de passo, exatamente a função do L297.

Neste momento enfatiza-se que o protocolo de transmissão e o seu funcionamento, estão detalhadamente explicados no trabalho de formatura complementar a este "Futebol de Robôs Autônomos – Transmissão e Visão Computacional", e que a listagem do programa do microcontrolador que executa as tarefas acima descritas está presente no apêndice 4.

Depois de ter estas ondas geradas e enviadas ao PortB do microcontrolador, o sinal passa pelo CI L293B, que é o driver de potência do motor de passo, proporcionando maior tensão e corrente para o motor que a saída do PIC.

A saída dos L293, o sinal se encaminha diretamente às quatro bobinas dos motores de passo, fazendo estes girar no sentido desejado.

E a partir desta versão do programa, os robôs são selecionados via software, e não mais por um botão na placa, ou seja, para cada robô existe um programa que se difere dos outros apenas na codificação dos bits que selecionam os robôs.

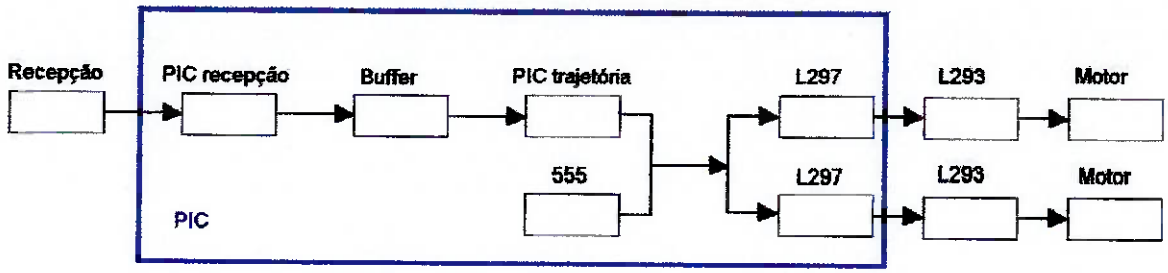


Figura 22 - Fluxograma do novo circuito

Com estas alterações, conseguimos simplificar bastante o circuito de controle dos motores de passo, diminuindo consideravelmente o número de componentes utilizados, porém aumentando a complexidade do programa em Assembler do microcontrolador.

9.2. Controle de trajetória

As trajetórias a serem executadas pelos robôs são geradas num programa de estratégia, e são transmitidas via rádio-freqüência para os robôs, para o controle das trajetórias o código de oito bits gerado pela transmissão foi dividido em várias funções a seguir descritas:

Os dois primeiros bits selecionam o robô (0, 1 ou 2), os dois bits subseqüentes selecionam o tipo de manobra, rotação ou translação ou gatilho de movimento, o bit subseqüente o sentido de movimento, rotação para direita ou esquerda, ou translação para frente ou para trás. Por convenção o robô 0 é o goleiro e os robôs 1 e 2 são defesa e ataque.

Bit	7	6	5	4	3	2	1	0
Dado	D7	D6	D5	D4	D3	D2	D1	D0

Tabela 4 - denominação dos bits

Após a denominação dos bits, segue o código criado para a criação da trajetória:

D7 **D6**	00 – robô 0 01 – robô 1 10 – robô 2 11 – palavra não utilizada
D5 **D4**	00 – rotação 01 – translação 10 – disparo de movimento 11 – palavra não utilizada
D3	0 – anti-horário p/ rotação para trás p/ translação 1 – horário para rotação para frente p/ translação
D2 **D1** **D0**	Magnitude do movimento: 000 0° 0 cm 001 25,7° 5cm 010 51,4° 10 cm 011 77,1° 15 cm 100 102,8° 20 cm 101 128,5° 25 cm 110 154,3° 30 cm 111 180° 35 cm

Tabela 5 – protocolo criado para as trajetórias

Com isso, conseguimos com um código de oito bits, selecionar o robô a ser movimentado, gravar no microcontrolador uma trajetória composta por uma rotação pura e uma translação pura, com suas respectivas magnitudes, geradas pelo programa de estratégia, e um gatilho de movimento.

Para melhor ilustração, neste momento apresentaremos um exemplo. Digamos que em certa parte do programa de estratégia, foi gerado uma trajetória para o robô 1 composta por uma rotação de 25° no sentido horário (CW) e uma translação de 30 centímetros para frente, o procedimento a ser efetuado está descrito a seguir.

Para selecionar o robô, os bits D7 e D6 deveriam ser 00, em seqüência para a manobra de rotação, os bits D5 e D4 deveriam ser 00, para sentido anti-

horário o bit D3 deveria ser 1, e para 25º os bits D2, D1 e D0 deveriam ser 001. Formando a palavra: 00001001, ou seja 9 na base decimal.

Seguindo o mesmo raciocínio para a translação, teríamos a palavra: 00011110, ou seja, 30 em decimal. E para efetuar o gatilho de movimento, encontraríamos a palavra: 00100000, ou seja 32 em decimal.

Portanto, como podemos ver, para efetuar uma manobra com um dos robôs é necessário enviar três palavras de comando na seqüência (9, 30, 32). Vale salientar que quando as manobras desejadas são uma rotação pura, ou uma translação pura, basta gravar no microcontrolador a informação que se deseja efetuar, e disparar o gatilho (D5). E quando informações de mesma natureza (duas ou mais rotações, ou duas ou mais translações), o programa executa a informação mais recente. As palavras não utilizadas não executam função nenhuma no programa, sendo assim, o sistema menos passível de erros de trajetórias a partir de palavras erradas, geradas por ruídos.

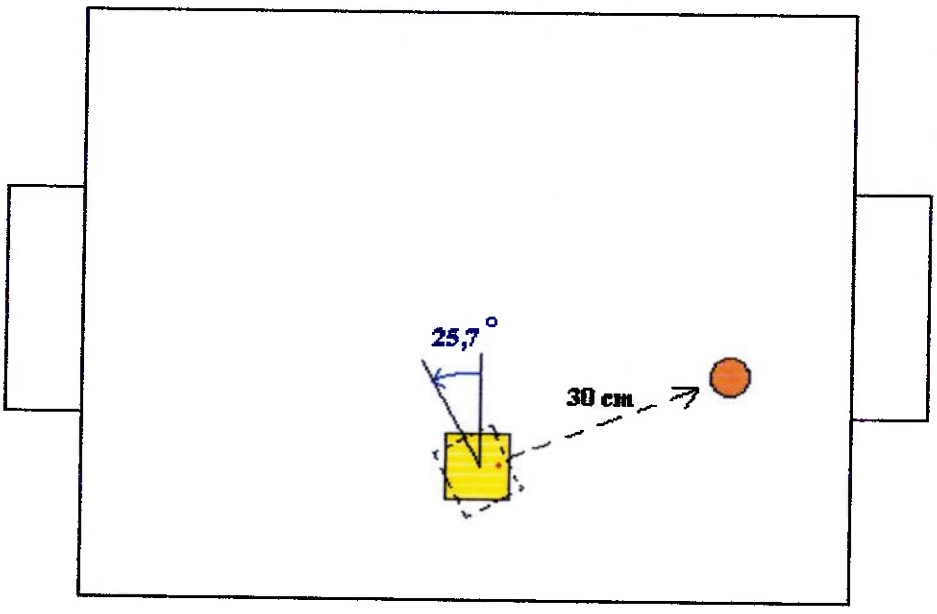


Figura 23 - Ilustração do exemplo de trajetória

É importante salientar ainda que o robô é um sistema de malha aberta, ao enviar os passos que o programa gerou, este confia na precisão da ação ordenada, mesmo que esta não aconteça. Quem acaba fechando a malha de controle do sistema inteiro é a câmera, já que todas as trajetórias são geradas a partir de coordenadas fornecidas pelo programa de visão computacional.

9.3. Detalhamento da placa eletrônica

A partir de agora, estão detalhados todos os componentes presentes na placa eletrônica e sua função, as pinagens e especificações dos componentes estão presentes nos apêndices 2 e 3.

Para um correto funcionamento do receptor, além das ligações de alimentação, terra (0 V) e V_{cc} (5 V), um capacitor de 25 nF é ligado entre o terra e V_{cc}, para garantir o bom funcionamento do rádio durante possíveis variações na alimentação, pois quando estas acontecem, o capacitor que está carregado com V_{cc} é quem fornece energia para a recepção.

Consta no *datasheet* do receptor que existem duas saídas do sinal, uma analógica e uma digital. Como só é necessário a utilização de uma delas no circuito, foi escolhida a melhor, a analógica. Para evitar-se possíveis ruídos, a saída não utilizada (digital) é aterrada através de um resistor alto (1 MΩ), para que este sinal morra e não atrapalhe o circuito.

O último pino do receptor é a entrada do sinal, ou seja, uma antena. Com a colocação de um simples fio rígido, da altura do receptor neste pino, o sinal da saída analógica aumentou aproximadamente 40%, sendo fundamental para o bom funcionamento da placa.

A partir da saída analógica do rádio, devem ser feitas alguns tratamentos com o sinal antes deste chegar ao microcontrolador. Primeiramente, para que não haja correntes inversas do PIC para o rádio, interferindo a recepção, foi colocado um diodo, tendo uma perda de sinal de 0,7 volts.

Mesmo com a colocação do diodo, o sinal (uma onda quadrada no protocolo de transmissão, carregando os dados da trajetória) possuía um sinal DC

de 1,5 volts no mínimo, e o máximo em 4 volts. Esses sinais são reconhecidos pelo PIC como um sinal constante em alta.

Para a resolução deste problema, o sinal começou a passar por um capacitor de 22 μF, para carregar e garantir o nível alto, enquanto que na saída do capacitor, foi ligado um resistor alto (1 MΩ), e daí para o terra para garantir o nível baixo. Com essas ligações, se garantia um sinal de 3,5 volts no máximo e de 0 volts no mínimo, facilmente reconhecido pelo PIC.

Vale salientar que quanto maior o capacitor, mais tempo ele demora para descarregar, portanto, com a utilização de capacitores pequenos (na ordem de pF) neste ponto, o sinal do gatilho tinha nível alto em 3,5 volts. Porém este nível alto ia caindo ao se caminhar para frente na onda devido ao rápido descarregamento do capacitor. Com a utilização de um capacitor maior (da ordem de μF), o capacitor começa a descarregar num tempo maior que a freqüência de transmissão de uma palavra (aproximadamente 50 Hz), não comprometendo o sinal.

O circuito de oscilação do cristal, que gera o *clock* para o microcontrolador, necessita de dois capacitores da ordem de pF ligados ao terra para evitar variações na oscilação. Com isso, o clock do PIC ficou por volta de 3 MHz.

O sinal devidamente tratado entra no PortA do microcontrolador, onde sofre a desmultiplexação dos oito bits nas rotinas de recepção do programa. Isso feito, os sinais agora passam por rotinas de identificação da informação transmitida (robô, gatilho de movimento, rotação, translação, direção e magnitude). O próximo passo são as rotinas de gerar ondas para os motores de passo. Para maiores detalhes de funcionamento, vide apêndice 4, listagem do programa do microcontrolador.

As ondas geradas são levadas ao PortB (0, 1, 2, 3), que irão gerenciar o motor direito, e as outras ondas levadas ao PortB (4, 5, 6, 7), que irão gerenciar o motor esquerdo.

As saídas do PIC vão direto para a entrada dos drivers de potência (L293B), e as saídas correspondentes vão direto para os motores através de conectores.

Com o circuito montado, a corrente máxima medida foi de apenas 180mA (menos de metade do circuito anterior), possibilitando o uso de baterias convencionais, e uma bateria que se encaixava bem dentro dos padrões dos robôs era a bateria 9 volts. Na relação custo benefício, ela possui uma tensão alta para o seu pequeno espaço, além de proporcionar a opção de alimentar os motores com os seus 9 volts, aumentando portanto o torque do robô, extremamente necessário para empurrar a bola, e vencer disputas com um adversário.

Com a utilização da bateria de 9 volts comercial, se fez necessário um regulador de tensão para alimentar os componentes (V_{cc}= 5 V), sendo usado portanto o 7805, que sempre mostrou confiável para esta tarefa. Com este componente montado, o circuito permaneceu funcionando, inclusive sem aquecimentos do 7805.

Com toda parte operacional funcionando, foram acrescentados um botão liga/desliga (*push buttton*), e três Leds. Um Led para indicar o funcionamento da placa (acionado pelo botão), e dois Leds para indicar o tipo de manobra que o robô está efetuando, translação (verde) ou rotação (vermelho). Para conseguir se certificar disto, o programa do PIC eleva o sinal de tensão de dois pinos do PortA ainda não utilizados, sempre que o programa entra nas funções de rotação e translação.

Abaixo segue um esquema do circuito montado:

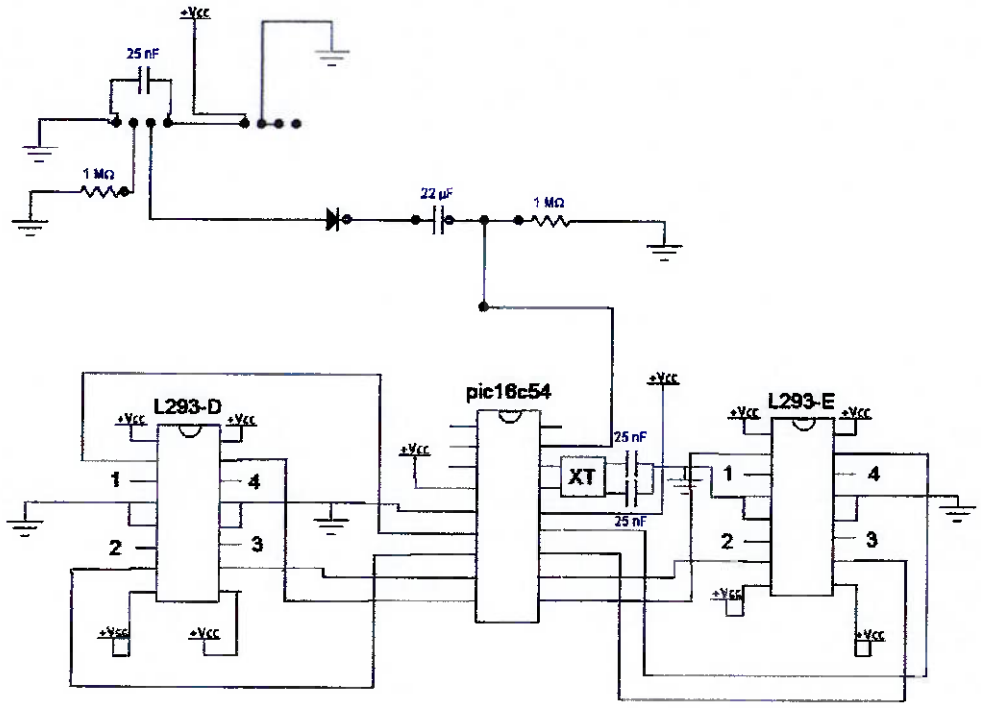


Figura 24 - Circuito esquemático do circuito

Abaixo segue o esquema do circuito impresso da placa, feito no Tango. Nota-se que a pinagem do L293B, obriga o circuito a ser de dupla face, e a preocupação com o tamanho final da placa foi extrema.

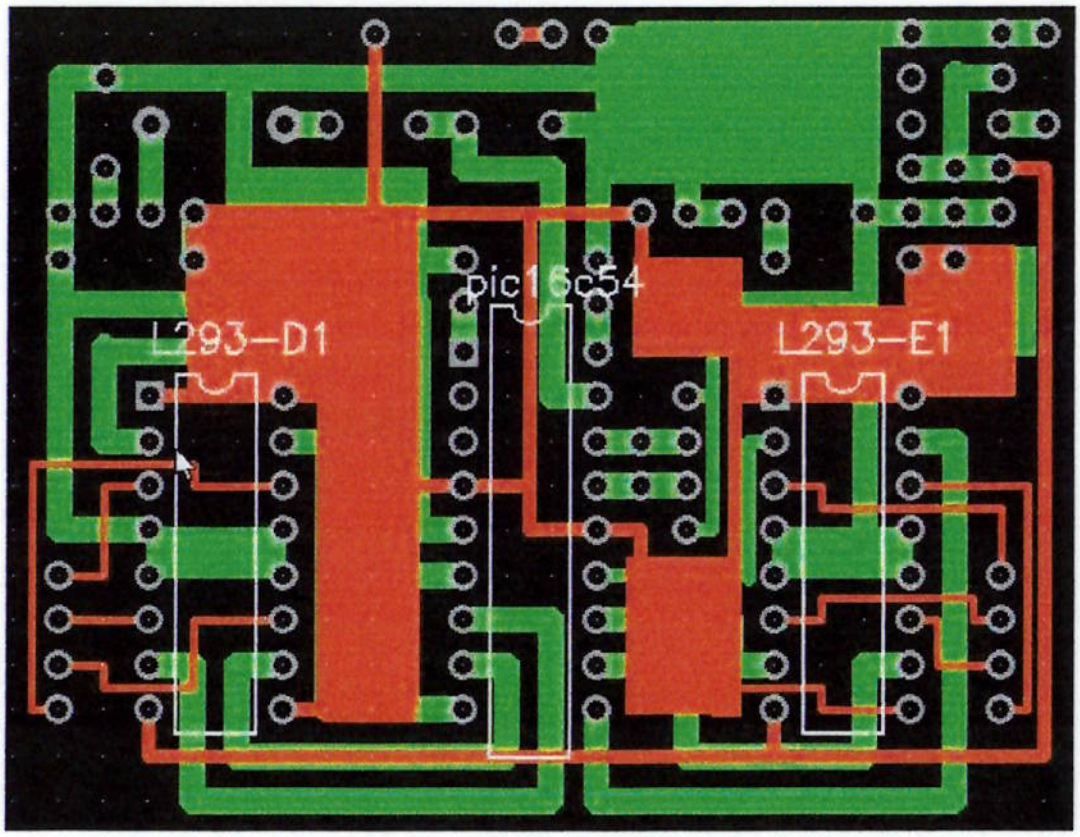


Figura 25 - Projeto do circuito eletrônico dupla face

Vale lembrar também que todos os circuitos integrados são fixados na placa através de soquetes de CI, para fácil remoção na hipótese de falhas. Abaixo segue a nova lista de componentes para a fabricação da placa:

Quantidade	Componentes
01	Receptor
01	Capacitor 25 nF
01	Capacitor 22 μF
02	Resistor 1 MΩ
01	Diodo
02	Capacitor 25 pF
01	Cristal 3,540 MHz
01	Pic 16c54
02	Driver potência – L293B
03	Leds
01	Regulador de 5 volts – 7805

Tabela 6 – Lista de componentes final

Vemos que com as mudanças propostas o tamanho da placa, assim como o número de componentes utilizados reduziu bastante, melhorando a performance dos robôs.

9.4. Acionamento dos motores de passo

Com as alterações da placa eletrônica, esta foi simplificada ao mesmo tempo que o programa do microcontrolador foi sofisticado, nesta nova versão. Os robôs são escolhidos por software, a velocidade dos robôs é alterada mudando uma constante de tempo que gera a freqüência de pulsos do motor de passo também via software, além do tipo de ligação dos motores de passo que também é alterado via software.

Com os robôs montados, foi possível encontrar um ponto ótimo entre a velocidade, alterando a freqüência dos pulsos, e o torque, alterando o tipo de ligação no motor de passo. Vale salientar que esses dados foram adquiridos experimentalmente devido a aquisição de motores usados sem nenhum tipo de manual ou especificação.

A freqüência foi aumentada gradativamente até encontrar-se um ponto que o robô desenvolvia uma boa velocidade para a competição, sem que este perdesse passos durante as manobras programadas, ou seja, perdesse a confiabilidade da malha aberta.

O torque nos motores foi aumentado graças à mudança da utilização de uma bateria de 9 volts convencional, o que possibilitou a utilização desta tensão diretamente no motor de passo, enquanto que ao mesmo tempo o esta tensão é regulada para 5 volts para alimentar o resto do circuito.

Os motores de passo adquiridos possuem 4 bobinas (A, B, C, D), e um terra comum, com estas características, existem três possibilidades de acionamento dos motores: *half-step*, *full-step* de uma fase e *full-step* de duas fases. A seguir estão uma melhor explicação destes acionamentos e os gráficos de controle:

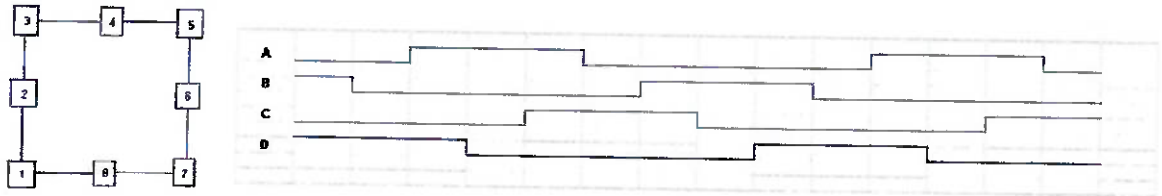


Figura 26 – Acionamento half-step

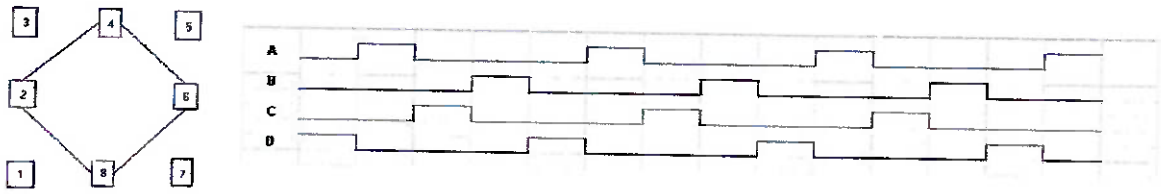


Figura 27 – Acionamento full-step de uma bobina

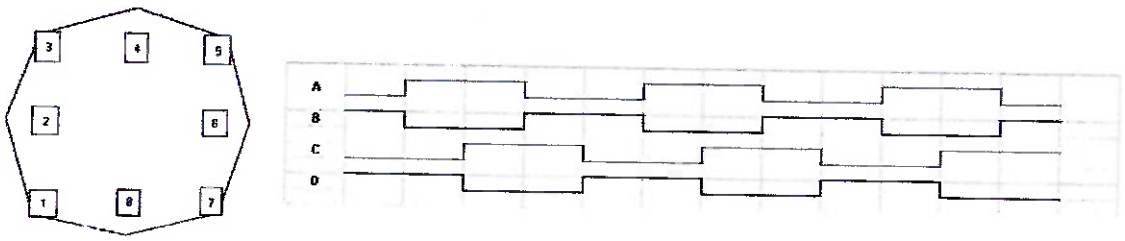


Figura 28 – Acionamento full-step de duas bobinas

Motores de passo com acionamento *half-step*, apesar de possuírem uma resolução melhor (menor rotação por passo), possuem torque variável pelo fato de variarem o número de bobinas ligadas em cada instante (variando uma e duas), prejudicando o desempenho do robô.

O acionamento *full-step* de uma bobina é descartado devido o seu menor torque para um mesmo passo em relação ao acionamento *full-step* de duas bobinas. Estes dois acionamentos possuem torque constante durante o seu funcionamento por não variarem o número de bobinas carregadas por passo, sempre uma no primeiro caso e sempre duas no segundo,

Com o robô montado, o *full-step* de duas bobinas demonstrou ser o melhor acionamento para a aplicação, pois é a montagem que provê maior torque constante, sendo portanto utilizado.

10. Resultados e Conclusões

Os resultados podem ser melhor descritos com a visão do sistema todo funcionando, porém como isto não é possível num relatório, está apresentada uma figura dos robôs em funcionamento:

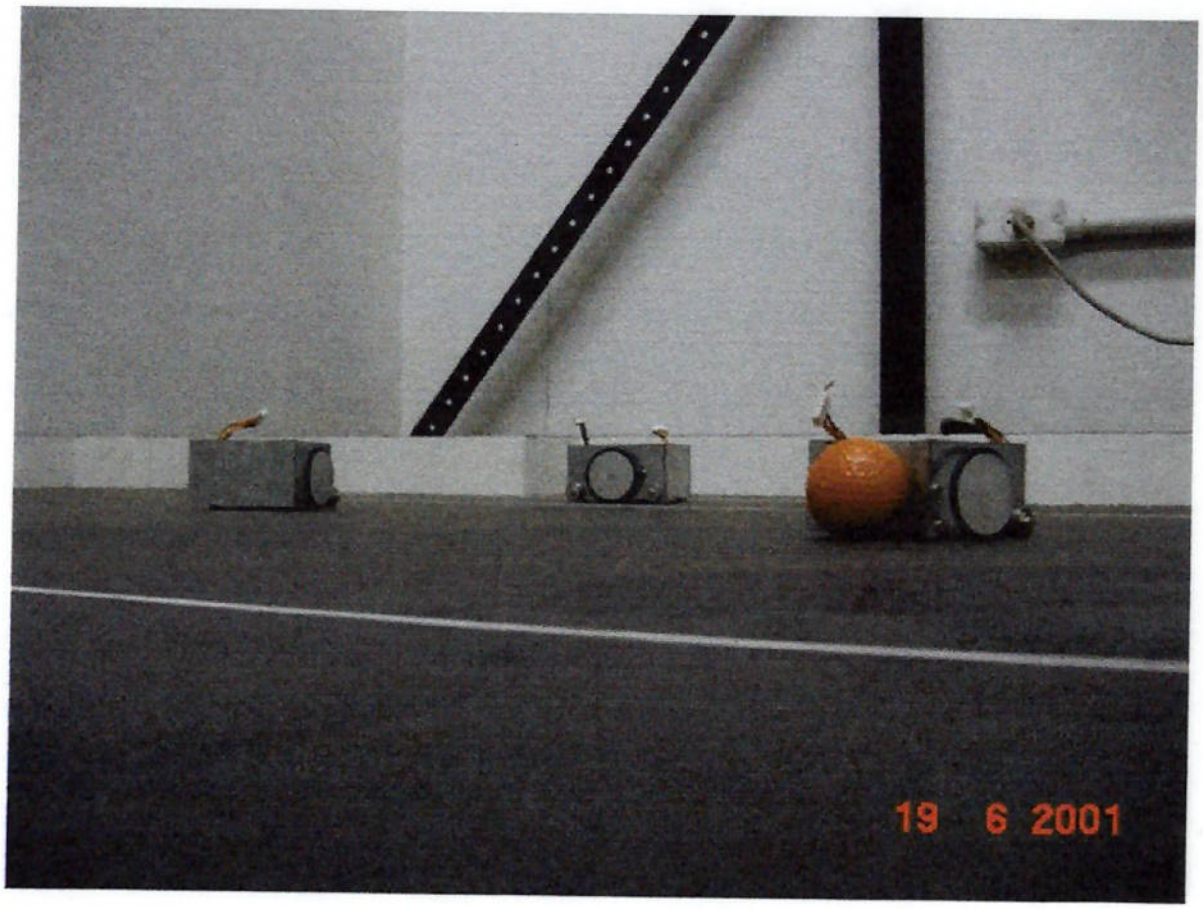


Figura 29 – foto dos robôs em funcionamento

Os projetos mecânico e eletrônico foram um sucesso, assim como o sistema todo integrado, em relação ao que foi proposto: montar uma equipe de futebol de robôs com verba reduzida. Para esta equipe realmente ficar competitiva no cenário internacional seriam necessários alguns investimentos em material especializado e equipamentos de última geração.

Este trabalho, integrado com os outros dois que tratam de transmissão, visão computacional e estratégias, atingiu os objetivos propostos de aplicar os conceitos de engenharia num problema, desenvolvendo um sistema de monitoração por câmera, que alimenta algoritmos de inteligência artificial que tomam decisões corretas para cada situação, e comanda um sistema autônomo a efetuar as decisões tomadas.

Estes conceitos são os alicerces para soluções em automação de muitos problemas reais, como monitoração de tráfego urbano, monitoração de processos industriais complexos, insalubres ou ainda, a solução de muitos problemas que a sociedade nem desconfia que possam ser automatizados.

Referências Bibliográficas

[1] Folha de São Paulo – 22/04/1998, 5º caderno: Informática;

[2] Internet – www.ia.cti.br

[3] IEEE Control Systems – junho/99 – pg 15;

[4] National Semiconductors Datasheet "LM628/LM629 Precision Motion Controller"

[5] National Semiconductors Aplication Notes an 706 – "LM628/LM629 User Guide"

[6] National Semiconductors Aplication Notes an 868 – "Interfacing the HPC and LM629 for Motion Control"

[7] Jones, J.L. & Flinn, A. M. "Mobile Robots: Inspiration to Implementation"

[8] Internet – www.robotis.com

[9] Bishop, Owen " Proyetos de Control Remoto"

[10] Hara, T. ;"Projetos de Eletrônica – Vol 2 – Transmissores de RF"

[11] Motorola Databook

[12] Internet – www.lcmi.ufsc.br/gia/robocup-br/historico.html

[13] Bianchi, R. A. C. ; "Uma arquitetura de controle distribuída para um sistema de visão computacional propositada" – Dissertação de mestrado – 1998

[14] Chagas, G. M. B. & Rillo, A. H.; "Sistema Visput – Visão Computacional Utilizada como realimentação sensorial em jogos de futebol de microrobôs"

[15] Freeman, James, A. & Skapura, D. M. "Neural Networks – Algorithms, Applications, and Programming Techiniques"

[16] Internet – ftp.lac.usp.br

[17] Machine Design – May/1984 – number 12 "1984 Eletrical & Eletronics Reference Issue"

[18] Kenjo, Tak; "Eletric Motors and Their Controls – an Introduction"

[19] Phillips, H. ; "Feedback Control Systems"

[20] Manuais do Matlab vers~ao 4

[21] Taub, H. ; "Circuitos Digitais e Microprocessadores"

[22] Schildt, H. ; "Inteligência Artificial utilizando linguagem C"

[23] Malvino ; "Eletrônica" 4º edição, volumes I e II

[24] Ogata, K. "Engenharia de Controle Moderno" Segunda edição

Apêndice A – Regras do jogo

Law 1. The Field and the Ball

(a) Playground dimensions

A black (non-reflective) wooden rectangular playground 150(cm) x 130(cm) in size with 5(cm) high and 2.5(cm) thick white side-walls will be used. The topsides of the side-walls shall be black in color with the walls painted in white (side view). Solid 7(cm) x 7(cm) isosceles triangles shall be fixed at the four corners of the playground to avoid the ball getting cornered. The surface texture of the board will be that of a ping pong table.

(b) Markings on the playground

The field of play shall be marked as shown in Appendix 1. The center circle will have a radius of 20(cm). The arc, which is part of the goal area, will be 20(cm) along the goal line and 5(cm) perpendicular to it. The major lines/arcs (centerline, goal area borderlines and the center circle) will be white in color and 3(mm) in thick. The free ball (Law 13) robot positions (circles) shall be marked in gray color.

(c) The goal

The goal shall be 40 (cm) wide. Posts and nets shall not be provided at the goal.

(d) The goal line and goal area

The goal line is the line just in front of the goal which is 40(cm) long. The goal areas shall comprise of areas contained by the rectangle (sized 70(cm) x 15(cm) in front of the goal) and the attached arc (20(cm) in parallel to the goal line and 5(cm) perpendicular to it).

(e) The ball

An orange golf ball shall be used as the ball, with 42.7(mm) diameter and 46(g) weight.

(f) The field location

The field shall be indoors.

(g) The lighting condition

The lighting condition in the competition site shall be fixed around 1,000 Lux.

Law 2: The Players

(a) The overall system

A match shall be played by two teams, each consisting of three robots. One of the robots can be the goalkeeper (Law 2.b.2). Three (3) human team members, a "manager", a "coach" and a "trainer" shall only be allowed on stage. One host computer per team, mainly dedicated to vision processing and for location identifying, can be used.

The robots

1. The size of each robot shall be limited to 7.5(cm) x 7.5(cm) x 7.5(cm). The height of the RF communication antenna will not be considered in deciding a robot's size.

(i) The topside of a robot must not be colored in orange. A color patch either blue or yellow, as assigned by the organizers, will identify the robots in a team. All the robots must have (at least) a 3.5(cm) x 3.5(cm) solid region of their team color patch, blue or yellow, visible on their top. A team's identification color will change from game to game, and the team color patch used should be detachable. When assigned with one of the 2-team colors (blue or yellow), the robots must not have any visible patches of those colors used by an opponent team.

Note: The teams are recommended to prepare a minimum of 6 different color patches, other than blue and yellow, for individual robot identification.

(ii) To enable infrared sensing a robot's sides should be colored light, except at regions necessarily used for robot functionality,

such as those for sensors, wheels and the ball catching mechanism. The robots should wear uniforms and the size of which shall be limited to 8(cm) x 8(cm) x 8(cm).

2. A robot within its own goal area (Law 1.d.) shall be considered as the "goalkeeper." The goalkeeper robot shall be allowed to catch or hold the ball only when it is inside its own goal area.

3. Each robot must be fully independent, with powering and motoring mechanisms self-contained. Only wireless communication shall be allowed for all kinds of interactions between the host computer and a robot.

4. The robots are allowed to equip with arms, legs, etc., but they must comply with the size restrictions (Law 2.b.1) even after the appendages fully expanded. None of the robots, except the single designated goalkeeper, shall be allowed to catch or hold the ball such that more than 30 % of the ball is out of view either from the top or from the sides

5. While a match is in progress, at any time the referee whistles the human operator should stop all robots using the communication between the robots and the host computer.

(c) Substitutions

Two substitutes shall be permitted while a game is in progress. At half time, unlimited substitutions can be made. When a substitution is desired while the game is in progress, the concerned team manager should call 'time-out' to notify the referee, and the referee will stop the game at an appropriate moment. The game will restart, with all the robots and the ball placed at the same positions as they were occupying at the time of interrupting the game.

(d) Time-Out

The human operator can call for 'time-out' to notify the referee. Each team will be entitled for two time-outs in a game and each shall be of 2 minutes duration.

Law 3: Transmissible Information

The manager, the coach or the trainer may transmit certain commands directly from the remote host computer to their robots. It is not allowed to transmit commands such as reset signals to stop any/all of the robots or restart signals, without the permission from the referee. Any other information, such as game strategy, can be communicated to robots only when a game is not in progress. The human operator should not directly control the motion of their robots either with a joystick or by keyboard commands under any circumstances. While a game is in progress the host computer can send any information autonomously.

Law 4: The Vision System

In order to identify the robots and the ball on the playground, a vision system can be used. The location of a team's camera or sensor system should be restricted to, over and above their own half of the field including the center line, so that the camera need not has to be moved after the side change at halftime. If both teams wish to keep their cameras over and above the center circle of the playground, they shall be placed side by side, equidistant from the centerline and as close to each other as possible. The location of the overhead camera or sensor system should be at a height of 2(m) or higher.

Law 5: Game Duration

(a) The duration of a game shall be two equal periods of 5 minutes each, with a half time interval for 10 minutes. An official timekeeper will pause the clock during substitutions, while transporting an injured robot from the field, during time-out and during such situations that deem to be right as per the discretion of the timekeeper.

(b) If a team is not ready to resume the game after the half time, additional 5 minutes shall be allowed. Even after the allowed additional time if such

a team is not ready to continue the game, that team will be disqualified from the game.

Law 6: Game Commencement

(a) Before the commencement of a game, either the team color (blue/yellow) or the ball shall be decided by the toss of a coin. The team that wins the toss shall be allowed to choose either their robot's identification color (blue/yellow) or the ball. The team who receives the ball shall be allowed to opt for their carrier frequency band as well.

(b) At the commencement of the game, the attacking team will be allowed to position their robots freely in their own area and within the center circle. Then the defending team can place their robots freely in their own area except within the center circle.

At the beginning of the first and second halves, and after a goal has been scored, the ball should be kept within the center circle and the ball should be kicked or passed towards the team's own side. With a signal from the referee, the game shall be started and all robots may move freely.

(c) At the beginning of the game or after a goal has been scored, the game shall be commenced/continued, with the positions of the robots as described in Law 6.b.

(d) After the half time, the teams have to change their sides.

Law 7: Method of Scoring

(a) The Winner

A goal shall be scored when the whole of the ball passes over the goal line. The winner of a game shall be decided on the basis of the number of goals scored.

(b) The Tiebreaker

In the event of a tie after the second half, the winner will be decided by the sudden death scheme. The game will be continued after a 5 minutes break, for a maximum period of three minutes. The team managing to score the first goal will be declared as the winner. If the tie persists even after the extra 3 minutes game, the winner shall be decided through penalty-kicks. Each team shall take three penalty-kicks, which differs from Law 11 as only a kicker and a goalkeeper shall be allowed on the playground. The goalkeeper should be kept within its goal area and the positions of the kicker and of the ball shall be the same as per the Law 11. After the referee's whistle, the goalkeeper may come out of the goal area. In case of a tie even after the three-time penalty-kicks, additional penalty-kicks shall be allowed one-by-one, until the winner can be decided. All penalty-kicks shall be taken by a single robot and shall commence with the referee's whistle. A penalty-kick will be completed, when any one of the following happens:

1. The goalkeeper catches the ball with its appendages (if any) in the goal area.
2. The ball comes out of goal area.
3. Thirty (30) seconds pass after the referee's whistle.

Law 8: Fouls

A foul will be called for in the following cases.

(a) Colliding with a robot of the opposite team, either intentionally or otherwise: the referee will call such fouls that directly affect the play of the game or that appear to have potential to harm the opponent robot. When a defender robot intentionally pushes an opponent robot, a free kick will be given to the opposite team. It is permitted to push the ball and an opponent player backwards provided the pushing player is always in contact with the ball.

(b) It is permitted to push the goalkeeper robot in the goal area, if the ball is between the pushing robot and the goalkeeper. However pushing the goalkeeper into the goal along with the ball is not allowed. If an attacking robot pushes the goalkeeper along with the ball into the goal or when the opponent robot

pushes the goalkeeper directly then the referee shall call goal kick as goalkeeper charging.

(c) Attacking with more than one robot in the goal area of the opposite team shall be penalized by a goal kick to be taken by the team of the goalkeeper. A robot is considered to be in the goal area if it is more than 50 % inside, as judged by the referee.

(d) Defending with more than one robot in the goal area shall be penalized by a penalty-kick. (A robot is considered to be in the goal area if it is more than 50 % inside, as judged by the referee.) An exception to this is the situation when the additional robot in the goal area is not there for defense or if it does not directly affect the play of the game. The referee shall judge the penalty-kick situation.

(e) It is referred to as handling, as judged by the referee, when a robot other than the goalkeeper catches the ball. It is also considered as handling, if a robot firmly attaches itself to the ball such a way that no other robot is allowed to manipulate the ball.

(f) The goalkeeper robot should kick out the ball from its goal area (Law 1.d.) within 10 seconds. The failure to do so will be penalized by giving a penalty kick to the opposite team.

(g) Giving a goal kick to the team of the goalkeeper will penalize the intentional blocking of a goalkeeper in its goal area.

(h) Only the referee and one of the human members of a team (manager, coach or trainer) shall be allowed to touch the robots. The award of a penalty-kick shall penalize touching the robots without the referee's permission.

Law 9: Play Interruptions

< when: only operator, human a by done be shall robots of relocation and interrupted play The

1. A robot has to be changed.
2. A robot has fallen in such a way as to block the goal.

3. A goal is scored or a foul occurs.
4. Referee calls goal kick (Law 12) or free-ball (Law 13).

Law 10: Free Kick

When a defender robot intentionally pushes an opponent robot, a free kick will be given to the opposite team (Law 8.a.). The ball will be placed at the relevant free kick position (FK) on the playground . The robot taking the kick shall be placed behind the ball. The attacking team can position its robots freely within its own side. The defending robots shall be placed in touch with the goal area on either side of the arc. With the referee's whistle all robots can start moving freely.

Law 11: Penalty-Kick

A penalty-kick will be called under the following situations.

1. Defending with more than one robot in a goal area (Law 8.d.).
2. Failure on the part of a goalkeeper to kick out the ball from its goal area within 10 seconds (Law 8.f.).
3. When any one of the human members touches the robots without the referee's permission, while the game is in progress (Law 8.h.).

When the referee calls a penalty-kick, the ball will be placed at the relevant penalty kick position (PK) on the playground . The robot taking the kick shall be placed behind the ball. While facing a penalty kick one of the sides of the goalkeeper must be in touch with the goal line. The goalkeeper may be oriented in any direction. Other robots shall be placed freely within the other side of the half-line, but the attacking team will get preference in positioning their robots. The game shall restart normally (all robots shall start moving freely) after the referee's whistle. The robot taking the penalty-kick may kick or dribble the ball.

Law 12: Goal Kick

A goal kick will be called under the following situations.

1. When an attacking robot pushes the goalkeeper in its goal area, the referee shall call goal kick as goalkeeper charging (Law 8.b.).
2. Attacking with more than one robot in the goal area of the opposite team shall be penalized by a goal kick to be taken by the opposite team (Law 8.c.).
3. When an opponent robot intentionally blocks the goalkeeper in its goal area (Law 8.g.).
4. When the goalkeeper catches the ball with its appendages (if any) in its own goal area.
5. When a stalemate occurs in the goal area for 10 seconds.

During goal kick only the goalkeeper will be allowed within the goal area and the ball can be placed anywhere within the goal area. Other robots of the team shall be placed outside the goal area during goal kick. The attacking team will get preference in positioning their robots anywhere on the playground, but it must be as per Law 8.c. The defending team can then place its robots within their own side of the playground. The game shall restart with the referee's whistle.

Law 13: Free-Ball

Referee will call a free-ball when a stalemate occurs for 10 seconds outside the goal area. When a free-ball is called within any quarter of the playground, the ball will be placed at the relevant free ball position (FB) . One robot per team will be placed at locations 20(cm) apart from the ball position in the longitudinal direction of the playground. Other robots (of both teams) can be placed freely outside the quarter where the free-ball is being called, but with the rule that, the defending team will get their preference in positioning their robots. The game

shall resume when the referee gives the signal and all robots may then move freely.

Apêndice B – Dados técnicos do microcontrolador

MICROCHIP

PIC16C5X

EPROM/ROM-Based 8-Bit CMOS Microcontroller Series

Devices included in this Data Sheet:

- PIC16C54
- PIC16CR54
- PIC16C55
- PIC16C56
- PIC16CR56
- PIC16C57
- PIC16CR57
- PIC16C58
- PIC16CR58

Note: 16C5X refers to all revisions of the part (i.e., 16C54 refers to 16C54, 16C54A, and 16C54C), unless specifically called out otherwise.

High-Performance RISC CPU:

- Only 33 single word instructions to learn
- All instructions are single cycle (200 ns) except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle

Device	Pins	I/O	EPROM/ ROM	RAM
PIC16C54	18	12	512	25
PIC16C54A	18	12	512	25
PIC16C54C	18	12	512	25
PIC16CR54A	18	12	512	25
PIC16CR54C	18	12	512	25
PIC16C55	28	20	512	24
PIC16C55A	28	20	512	24
PIC16C56	18	12	1K	25
PIC16C56A	18	12	1K	25
PIC16CR56A	18	12	1K	25
PIC16C57	28	20	2K	72
PIC16C57C	28	20	2K	72
PIC16CR57C	28	20	2K	72
PIC16C58B	18	12	2K	73
PIC16CR58B	18	12	2K	73

- 12-bit wide instructions
- 8-bit wide data path
- Seven or eight special function hardware registers
- Two-level deep hardware stack
- Direct, indirect and relative addressing modes for data and instructions

Peripheral Features:

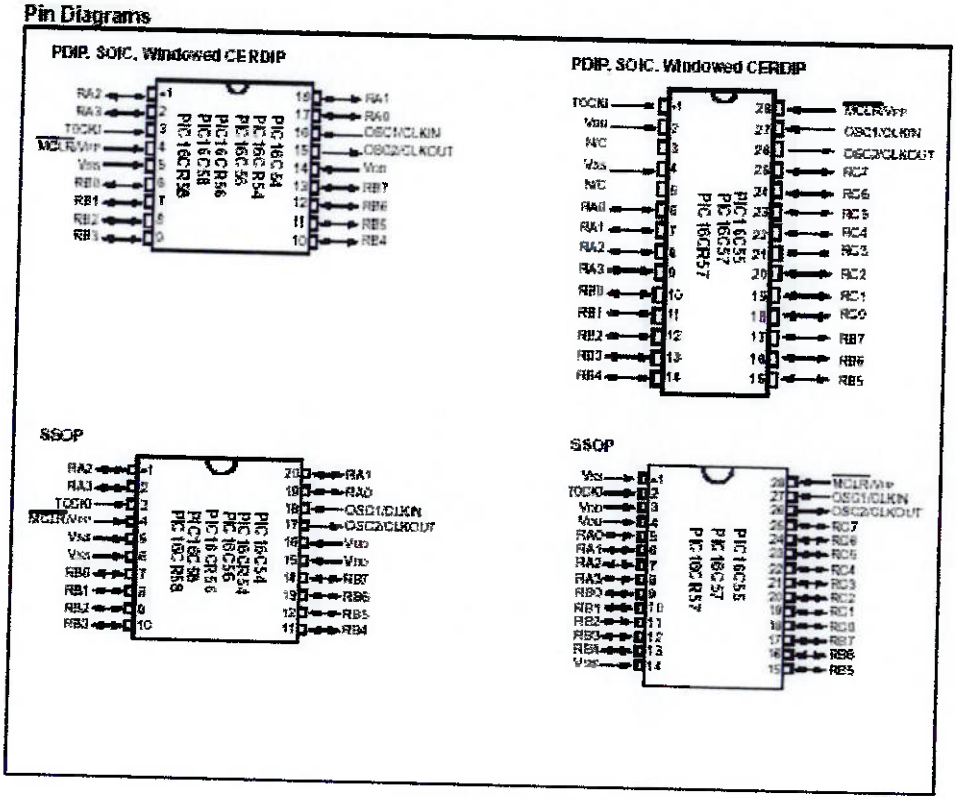
- 8-bit real time clock/counter (TMR0) with 8-bit programmable prescaler
- Power-on Reset (POR)
- Device Reset Timer (DRT)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable Code Protection
- Power saving SLEEP mode
- Selectable oscillator options:
  - RC: Low-cost RC oscillator
  - XT: Standard crystal/resonator
  - HS: High-speed crystal/resonator
  - LP: Power saving, low-frequency crystal

CMOS Technology:

- Low-power, high-speed CMOS EPROM/ROM technology
- Fully static design
- Wide-operating voltage and temperature range:
  - EPROM Commercial/Industrial 2.0V to 6.25V
  - ROM Commercial/Industrial 2.0V to 6.25V
  - EPROM Extended 2.5V to 6.0V
  - ROM Extended 2.5V to 6.0V
- Low-power consumption
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 3V, 32 kHz
  - < 0.6 µA typical standby current (with WDT disabled) @ 3V, 0°C to 70°C

Note: In this document, figure and table titles refer to all varieties of the part number indicated. (i.e., The title "Figure 14-1: Load Conditions - PIC16C54A", also refers to PIC16LC54A and PIC16LV54A parts) unless specifically called out otherwise.

Pin Diagrams



Device Differences

Device	Voltage Range	Oscillator Selection (Program)	Oscillator	Process Technology (Microns)	ROM Equivalent	$\overline{MCLR}$ Filter
PIC16C54	2.5-6.25	Factory	See Note 1	1.2	PIC16CR54A	No
PIC16C54A	2.0-6.25	User	See Note 1	0.9	—	No
PIC16C54C	2.5-5.5	User	See Note 1	0.7	PIC16CR54C	Yes
PIC16C55	2.5-6.25	Factory	See Note 1	1.7	—	No
PIC16C55A	2.5-5.5	User	See Note 1	0.7	—	Yes
PIC16C56	2.5-6.25	Factory	See Note 1	1.7	—	No
PIC16C56A	2.5-5.5	User	See Note 1	0.7	PIC16CR56A	Yes
PIC16C57	2.5-6.25	Factory	See Note 1	1.2	—	No
PIC16C57C	2.5-5.5	User	See Note 1	0.7	PIC16CR57C	Yes
PIC16C58B	2.5-5.5	User	See Note 1	0.7	PIC16CR58B	Yes
PIC16CR54A	2.5-6.25	Factory	See Note 1	1.2	N/A	Yes
PIC16CR54C	2.5-5.5	Factory	See Note 1	0.7	N/A	Yes
PIC16CR56A	2.5-5.5	Factory	See Note 1	0.7	N/A	Yes
PIC16CR57C	2.5-5.5	Factory	See Note 1	0.7	N/A	Yes
PIC16CR58B	2.5-5.5	Factory	See Note 1	0.7	N/A	Yes

Note 1: If you change from this device to another device, please verify oscillator characteristics in your application.

FIGURE 3-1: PIC16C5X SERIES BLOCK DIAGRAM

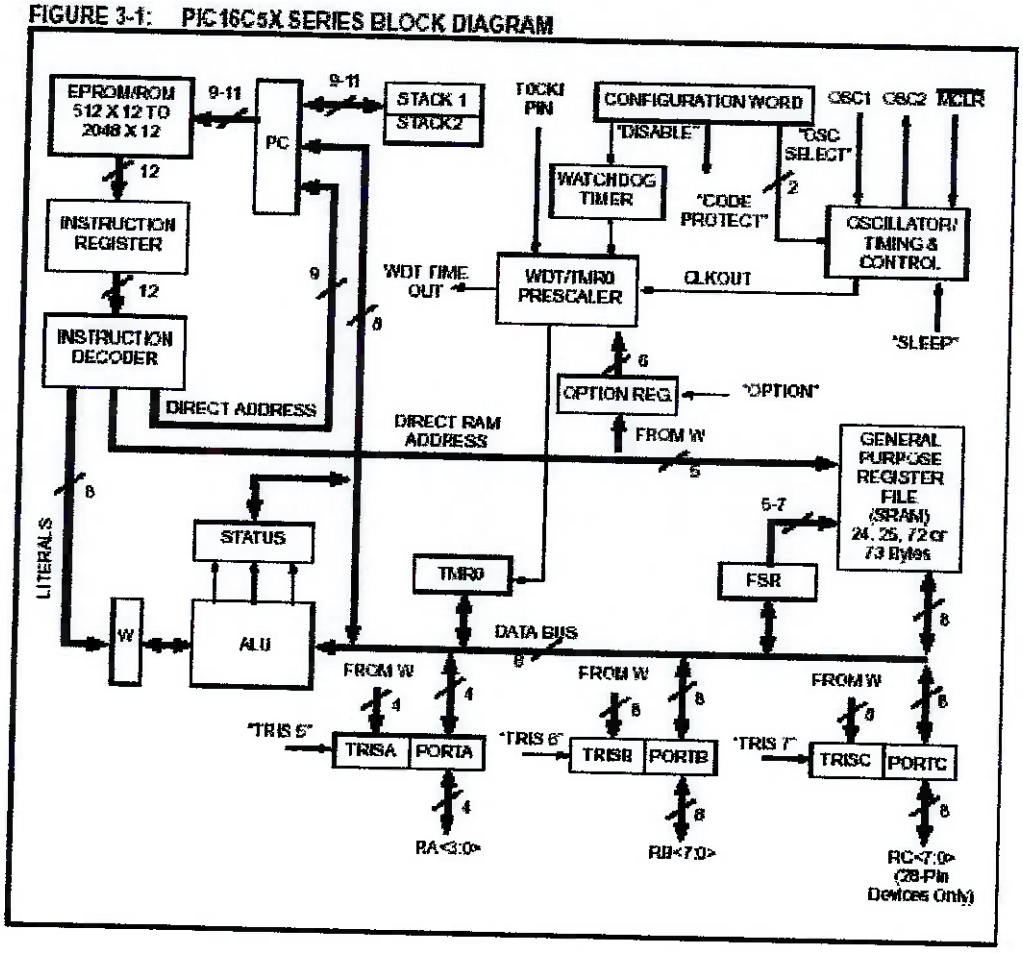


TABLE 3-1: PINOUT DESCRIPTION - PIC16C54s, PIC16CR54, PIC16C56, PIC16CR56, PIC16C58, PIC16CR58

Name	DIP, SOIC No.	SSOP No.	I/O/P Type	Input Levels	Description
RA0	17	19	I/O	TTL	Bi-directional I/O port
RA1	18	20	I/O	TTL	
RA2	1	1	I/O	TTL	
RA3	2	2	I/O	TTL	
RB0	6	7	I/O	TTL	Bi-directional I/O port
RB1	7	8	I/O	TTL	
RB2	8	9	I/O	TTL	
RB3	9	10	I/O	TTL	
RB4	10	11	I/O	TTL	
RB5	11	12	I/O	TTL	
RB6	12	13	I/O	TTL	
RB7	13	14	I/O	TTL	
TOCKI	3	3	I	ST	Clock input to Timer0. Must be tied to V_{SS} or V_{DD} if not in use, to reduce current consumption.
MCLR/V_{PP}	4	4	I	ST	Master clear (RESET) input/programming voltage input. This pin is an active low RESET to the device. Voltage on the MCLR/V_{PP} pin must not exceed V_{DD} to avoid unintended entering of programming mode.
OSC1/CLKIN	16	18	I	ST	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	17	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
V_{DD}	14	15,16	P	—	Positive supply for logic and I/O pins.
V_{SS}	5	5,6	P	—	Ground reference for logic and I/O pins.

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

TABLE 7-3: RESET CONDITIONS FOR SPECIAL REGISTERS

Condition	PCL Addr: 02h	STATUS Addr: 03h
Power-on Reset	1111 1111	0001 1xxx
$\overline{MCLR}$ Reset (normal operation)	1111 1111	000u uuuu(1)
$\overline{MCLR}$ Wake-up (from SLEEP)	1111 1111	0001 0uuu
WDT Reset (normal operation)	1111 1111	0000 uuuu(2)
WDT Wake-up (from SLEEP)	1111 1111	0000 0uuu

Legend: u = unchanged, x = unknown, - = unimplemented read as '0'.
Note 1: $\overline{TO}$ and $\overline{PD}$ bits retain their last value until one of the other RESET conditions occur.
2: The CLRWDT instruction will set the $\overline{TO}$ and $\overline{PD}$ bits.

TABLE 7-4: RESET CONDITIONS FOR ALL REGISTERS

Register	Address	Power-on Reset	$\overline{MCLR}$ or WDT Reset
W	N/A	xxxx xxxx	uuuu uuuu
TRIS	N/A	1111 1111	1111 1111
OPTION	N/A	--11 1111	--11 1111
INDF	00h	xxxx xxxx	uuuu uuuu
TMR0	01h	xxxx xxxx	uuuu uuuu
PCL(1)	02h	1111 1111	1111 1111
STATUS(1)	03h	0001 1xxx	000q quuu
FSR	04h	1xxx xxxx	1uuu uuuu
PORTA	05h	---- xxxx	---- uuuu
PORTB	06h	xxxx xxxx	uuuu uuuu
PORTC(2)	07h	xxxx xxxx	uuuu uuuu
General Purpose Register Files	07-7Fh	xxxx xxxx	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented, read as '0',
q = see tables in Section 7.7 for possible values.
Note 1: See Table 7-3 for RESET value for specific conditions.
2: General purpose register file on PIC16C54/CR54/C56/CR56/C58/CR58.

TABLE 8-2: INSTRUCTION SET SUMMARY

Mnemonic, Operands		Description	Cycles	12-Bit Opcode MSb		LSb	Status Affected	Notes
ADDWF	f,d	Add W and f	1	0001	11df	ffff	C,DC,Z	1,2,4
ANDWF	f,d	AND W with f	1	0001	01df	ffff	Z	2,4
CLRF	f	Clear f	1	0000	011f	ffff	Z	4
CLRW	–	Clear W	1	0000	0100	0000	Z	
COMF	f, d	Complement f	1	0010	01df	ffff	Z	
DECF	f, d	Decrement f	1	0000	11df	ffff	Z	2,4
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	0010	11df	ffff	None	2,4
INCF	f, d	Increment f	1	0010	10df	ffff	Z	2,4
INCFSZ	f, d	Increment f, Skip if 0	1(2)	0011	11df	ffff	None	2,4
IORWF	f, d	Inclusive OR W with f	1	0001	00df	ffff	Z	2,4
MOVF	f, d	Move f	1	0010	00df	ffff	Z	2,4
MOVWF	f	Move W to f	1	0000	001f	ffff	None	1,4
NOP	–	No Operation	1	0000	0000	0000	None	
RLF	f, d	Rotate left f through Carry	1	0011	01df	ffff	C	2,4
RRF	f, d	Rotate right f through Carry	1	0011	00df	ffff	C	2,4
SUBWF	f, d	Subtract W from f	1	0000	10df	ffff	C,DC,Z	1,2,4
SWAPF	f, d	Swap f	1	0011	10df	ffff	None	2,4
XORWF	f, d	Exclusive OR W with f	1	0001	10df	ffff	Z	2,4
BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF	f, b	Bit Clear f	1	0100	bbbf	ffff	None	2,4
BSF	f, b	Bit Set f	1	0101	bbbf	ffff	None	2,4
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	0110	bbbf	ffff	None	
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	0111	bbbf	ffff	None	
LITERAL AND CONTROL OPERATIONS								
ANDLW	k	AND literal with W	1	1110	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	1001	kkkk	kkkk	None	1
CLRWDT	k	Clear Watchdog Timer	1	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
GOTO	k	Unconditional branch	2	101k	kkkk	kkkk	None	
IORLW	k	Inclusive OR Literal with W	1	1101	kkkk	kkkk	Z	
MOVLW	k	Move Literal to W	1	1100	kkkk	kkkk	None	
OPTION	k	Load OPTION register	1	0000	0000	0010	None	
RETLW	k	Return, place Literal in W	2	1000	kkkk	kkkk	None	
SLEEP	–	Go into standby mode	1	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	
TRIS	f	Load TRIS register	1	0000	0000	0fff	None	3
XORLW	k	Exclusive OR Literal to W	1	1111	kkkk	kkkk	Z	

Note 1: The 9th bit of the program counter will be forced to a '0' by any instruction that writes to the PC except for GOTO. (See individual device data sheets, Memory Section/Indirect Data Addressing, INDF and FSR Registers)

2: When an I/O register is modified as a function of itself (e.g. `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

3: The instruction `TRIS f`, where f = 5 or 6 causes the contents of the W register to be written to the tristate latches of PORTA or B respectively. A '1' forces the pin to a hi-impedance state and disables the output buffers.

4: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared (if assigned to TMR0).

ADDWF Add W and f

Syntax: [label] ADDWF f,d
Operands: 0 ≤ f ≤ 31
d ∈ [0,1]
Operation: (W) + (f) → (dest)
Status Affected: C, DC, Z
Encoding: 0010 | 11df | ffff
Description: Add the contents of the W register and register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is '1' the result is stored back in register 'f'.
Words: 1
Cycles: 1
Example:
```
ADDWF   FSR, 0
```
Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0x09
FSR = 0xC2

ANDWF AND W with f

Syntax: [label] ANDWF f,d
Operands: 0 ≤ f ≤ 31
d ∈ [0,1]
Operation: (W) .AND. (f) → (dest)
Status Affected: Z
Encoding: 0001 | 01df | ffff
Description: The contents of the W register are AND'ed with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is '1' the result is stored back in register 'f'.
Words: 1
Cycles: 1
Example:
```
ANDWF   FSR,    1
```
Before Instruction
W = 0x17
TEMP_REG = 0xC2
After Instruction
W = 0x17
TEMP_REG = 0x2

ANDLW And literal with W

Syntax: [label] ANDLW k
Operands: 0 ≤ k ≤ 255
Operation: (W).AND. (k) → (W)
Status Affected: Z
Encoding: 1110 | kkkk | kkkk
Description: The contents of the W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.
Words: 1
Cycles: 1
Example:
```
ANDLW   0x5F
```
Before Instruction
W = 0xA3
After Instruction
W = 0x03

BCF Bit Clear f

Syntax: [label] BCF f,b
Operands: 0 ≤ f ≤ 31
0 ≤ b ≤ 7
Operation: 0 → (f<b>)
Status Affected: None
Encoding: 0100 | bbbf | ffff
Description: Bit 'b' in register 'f' is cleared.
Words: 1
Cycles: 1
Example:
```
BCF     FLAG_REG,  7
```
Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

BSF Bit Set f

Syntax: [label] BSF f,b
Operands: 0 ≤ f ≤ 31
0 ≤ b ≤ 7
Operation: 1 → (f<b>)
Status Affected: None
Encoding: 0101 | bbbf | ffff
Description: Bit 'b' in register 'f' is set.
Words: 1
Cycles: 1
Example:
```
BSF     FLAG_REG,  7
```
Before Instruction
FLAG_REG = 0x0A
After Instruction
FLAG_REG = 0x8A

BTFSC Bit Test f, Skip if Clear

Syntax: [label] BTFSC f,b
Operands: 0 ≤ f ≤ 31
0 ≤ b ≤ 7
Operation: skip if (f<b>) = 0
Status Affected: None
Encoding: 0110 | bbbf | ffff
Description: If bit 'b' in register 'f' is 0 then the next instruction is skipped.
If bit 'b' is 0 then the next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead, making this a 2-cycle instruction.
Words: 1
Cycles: 1(2)
Example:
```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```
Before Instruction
PC = address (HERE)
After Instruction
if FLAG<1> = 0,
PC = address (FALSE)
if FLAG<1> = 1,
PC = address (TRUE)

BTFSS Bit Test f, Skip if Set

Syntax: [label] BTFSS f,b
Operands: 0 ≤ f ≤ 31
0 ≤ b < 7
Operation: skip if (f<b>) = 1
Status Affected: None
Encoding: 0111 | bbbf | ffff
Description: If bit 'b' in register 'f' is '1' then the next instruction is skipped.
If bit 'b' is '1', then the next instruction fetched during the current instruction execution, is discarded and a NOP is executed instead, making this a 2-cycle instruction.
Words: 1
Cycles: 1(2)
Example:
```
HERE    BTFSS   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```
Before Instruction
PC = address (HERE)
After Instruction
if FLAG<1> = 0,
PC = address (FALSE)
if FLAG<1> = 1,
PC = address (TRUE)

CALL Subroutine Call

Syntax: [label] CALL k
Operands: 0 ≤ k ≤ 255
Operation: (PC) + 1→ Top of Stack;
k → PC<7:0>;
(STATUS<6:5>) → PC<10:9>;
0 → PC<8>
Status Affected: None
Encoding: 1001 kkkk kkkk
Description: Subroutine call. First, return address (PC+1) is pushed onto the stack. The eight bit immediate address is loaded into PC bits <7:0>. The upper bits PC<10:9> are loaded from STATUS<6:5>, PC<8> is cleared. CALL is a two-cycle instruction.
Words: 1
Cycles: 2
Example:
```
HERE  CALL  THERE
```
Before Instruction
PC = address (HERE)
After Instruction
PC = address (THERE)
TOS = address (HERE + 1)

CLRF Clear f

Syntax: [label] CLRF f
Operands: 0 ≤ f ≤ 31
Operation: 00h → (f);
1 → Z
Status Affected: Z
Encoding: 0001 011f ffff
Description: The contents of register 'f' are cleared and the Z bit is set.
Words: 1
Cycles: 1
Example:
```
CLRF  FLAG_REG
```
Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00
Z = 1

CLRW Clear W

Syntax: [label] CLRW
Operands: None
Operation: 00h → (W);
1 → Z
Status Affected: Z
Encoding: 0001 0100 0000
Description: The W register is cleared. Zero bit (Z) is set.
Words: 1
Cycles: 1
Example:
```
CLRW
```
Before Instruction
W = 0x5A
After Instruction
W = 0x00
Z = 1

CLRWDT Clear Watchdog Timer

Syntax: [label] CLRWDT
Operands: None
Operation: 00h → WDT;
0 → WDT prescaler (if assigned);
1 → $\overline{TO}$;
1 → $\overline{PD}$
Status Affected: $\overline{TO}$, $\overline{PD}$
Encoding: 0000 0000 0100
Description: The CLRWDT instruction resets the WDT. It also resets the prescaler, if the prescaler is assigned to the WDT and not Timer0. Status bits $\overline{TO}$ and $\overline{PD}$ are set.
Words: 1
Cycles: 1
Example:
```
CLRWDT
```
Before Instruction
WDT counter = ?
After Instruction
WDT counter = 0x00
WDT prescale = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF Complement f

Syntax: [label] COMF f,d
Operands: 0 ≤ f ≤ 31
d ∈ [0,1]
Operation: $(\bar{f})$ → (dest)
Status Affected: Z
Encoding: 0010 01df ffff
Description: The contents of register 'f' are complemented. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words: 1
Cycles: 1
Example:
```
COMF  REG1,0
```
Before Instruction
REG1 = 0x13
After Instruction
REG1 = 0x13
W = 0xEC

DECF Decrement f

Syntax: [label] DECF f,d
Operands: 0 ≤ f ≤ 31
d ∈ [0,1]
Operation: (f) – 1 → (dest)
Status Affected: Z
Encoding: 0000 11df ffff
Description: Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words: 1
Cycles: 1
Example:
```
DECF  CNT,  1
```
Before Instruction
CNT = 0x01
Z = 0
After Instruction
CNT = 0x00
Z = 1

DECFSZ Decrement f, Skip if 0

Syntax: [label] DECFSZ f,d
Operands: 0 ≤ f ≤ 31
d ∈ [0,1]
Operation: (f) – 1 → d; skip if result = 0
Status Affected: None
Encoding: 0010 11df ffff
Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, the next instruction, which is already fetched, is discarded and a NOP is executed instead making it a two-cycle instruction.
Words: 1
Cycles: 1(2)
Example:
```
HERE  DECFSZ  CNT, 1
      GOTO    LOOP
CONTINUE •
         •
         •
```
Before Instruction
PC = address (HERE)
After Instruction
CNT = CNT - 1,
If CNT = 0,
PC = address (CONTINUE),
If CNT ≠ 0,
PC = address (HERE+1)

GOTO Unconditional Branch

Syntax: [label] GOTO k
Operands: 0 ≤ k ≤ 511
Operation: k → PC<8:0>;
STATUS<6:5> → PC<10:9>
Status Affected: None
Encoding: 101k kkkk kkkk
Description: GOTO is an unconditional branch. The 9-bit immediate value is loaded into PC bits <8:0>. The upper bits of PC are loaded from STATUS<6:5>. GOTO is a two-cycle instruction.
Words: 1
Cycles: 2
Example:
```
GOTO THERE
```
After Instruction
PC = address (THERE)

INCF Increment f

Syntax: [label] INCF f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

Operation: (f) + 1 → (dest)

Status Affected: Z

Encoding: 0010 10df ffff

Description: The contents of register f are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register f.

Words: 1

Cycles: 1

Example:
```
INCF  CNT  1
```
Before Instruction
CNT = 0xFF
Z = 0

After Instruction
CNT = 0x00
Z = 1

INCFSZ Increment f, Skip if 0

Syntax: [label] INCFSZ f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

Operation: (f) + 1 → (dest), skip if result = 0

Status Affected: None

Encoding: 0011 11df ffff

Description: The contents of register f are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register f.
If the result is 0, then the next instruction, which is already fetched, is discarded and a nop is executed instead making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Example:
```
HERE    INCFSZ  CNT, 1
        GOTO    LOOP
CONTINUE •
         •
         •
```
Before Instruction
PC = address (HERE)

After Instruction
CNT = CNT + 1
if CNT = 0,
PC = address (CONTINUE)
if CNT ≠ 0,
PC = address (HERE +1)

IORLW Inclusive OR literal with W

Syntax: [label] IORLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) .OR. (k) → (W)

Status Affected: Z

Encoding: 1101 kkkk kkkk

Description: The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example:
```
IORLW  0x35
```
Before Instruction
W = 0x9A

After Instruction
W = 0xBF
Z = 0

IORWF Inclusive OR W with f

Syntax: [label] IORWF f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

Operation: (W).OR. (f) → (dest)

Status Affected: Z

Encoding: 0100 01df ffff

Description: Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Example:
```
IORWF          RESULT, 0
```
Before Instruction
RESULT = 0x13
W = 0x91

After Instruction
RESULT = 0x13
W = 0x93
Z = 0

MOVF Move f

Syntax: [label] MOVF f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

Operation: (f) → (dest)

Status Affected: Z

Encoding: 0010 00df ffff

Description: The contents of register f is moved to destination 'd'. If 'd' is 0, destination is the W register. If 'd' is 1, the destination is file register f itself. 'd' is 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Example:
```
MOVF   FSR   0
```
After Instruction
W = value in FSR register

MOVLW Move Literal to W

Syntax: [label] MOVLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W)

Status Affected: None

Encoding: 1100 kkkk kkkk

Description: The eight bit literal 'k' is loaded into the W register. The don't cares will assemble as 0s.

Words: 1

Cycles: 1

Example:
```
MOVLW  0x5A
```
After Instruction
W = 0x5A

MOVWF Move W to f

Syntax: [label] MOVWF f

Operands: 0 ≤ f ≤ 31

Operation: (W) → (f)

Status Affected: None

Encoding: 0000 001f ffff

Description: Move data from the W register to register 'f'.

Words: 1

Cycles: 1

Example:
```
MOVWF   TEMP_REG
```
Before Instruction
TEMP_REG = 0xFF
W = 0x4F

After Instruction
TEMP_REG = 0x4F
W = 0x4F

NOP No Operation

Syntax: [label] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding: 0000 0000 0000

Description: No operation.

Words: 1

Cycles: 1

Example:
```
NOP
```

OPTION — Load OPTION Register

Syntax:	[label] OPTION
Operands:	None
Operation:	(W) → OPTION
Status Affected:	None
Encoding:	0000 0000 0010
Description:	The content of the W register is loaded into the OPTION register.
Words:	1
Cycles:	1
Example	OPTION

Before Instruction
W = 0x07
After Instruction
OPTION = 0x07

RETLW — Return with Literal in W

Syntax:	[label] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W); TOS → PC
Status Affected:	None
Encoding:	1000 kkkk kkkk
Description:	The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.
Words:	1
Cycles:	2

Example:

```
CALL TABLE ;W contains
           ;table offset
           ;value.
•          ;W now has table
•          ;value.
•
TABLE      ADDWF PC   ;W = offset
           RETLW k1   ;Begin table
           RETLW k2   ;
           •
           •
           •
           RETLW kn   ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of k8

RLF — Rotate Left f through Carry

Syntax:	[label] RLF f,d
Operands:	0 ≤ f ≤ 31 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Encoding:	0011 01df ffff
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words:	1
Cycles:	1

Example:

```
RLF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 1100 1100
C = 1

RRF — Rotate Right f through Carry

Syntax:	[label] RRF f,d
Operands:	0 ≤ f ≤ 31 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Encoding:	0011 00df ffff
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words:	1
Cycles:	1

Example:

```
RRF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 0111 0011
C = 0

SLEEP — **Enter SLEEP Mode**

Syntax: [label] SLEEP

Operands: None

Operation: 00h → WDT;
0 → WDT prescaler;
1 → $\overline{TO}$;
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0011

Description: Time-out status bit ($\overline{TO}$) is set. The power-down status bit ($\overline{PD}$) is cleared. The WDT and its prescaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped. See section on SLEEP for more details.

Words: 1

Cycles: 1

Example:

```
SLEEP
```

SUBWF — **Subtract W from f**

Syntax: [label] SUBWF f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

Operation: (f) – (W) → (dest)

Status Affected: C, DC, Z

Encoding:

0000	10di	ffff

Description: Subtract (2's complement method) the W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example 1:

```
SUBWF   REG1, 1
```

Before Instruction
REG1 = 3
W = 2
C = ?

After Instruction
REG1 = 1
W = 2
C = 1 ; result is positive

Example 2:

Before Instruction
REG1 = 2
W = 2
C = ?

After Instruction
REG1 = 0
W = 2
C = 1 ; result is zero

Example 3:

Before Instruction
REG1 = 1
W = 2
C = ?

After Instruction
REG1 = FF
W = 2
C = 0 ; result is negative

SWAPF Swap Nibbles in f

Syntax: [label] SWAPF f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

Operation: (f<3:0>) → (dest<7:4>);
(f<7:4>) → (dest<3:0>)

Status Affected: None

Encoding:

0011	10df	ffff

Description: The upper and lower nibbles of register f are exchanged. If d is 0 the result is placed in W register. If d is 1 the result is placed in register f.

Words: 1

Cycles: 1

Example:
```
SWAPF   REG1, 0
```
Before Instruction
REG1 = 0xA5
After Instruction
REG1 = 0xA5
W = 0x5A

TRIS Load TRIS Register

Syntax: [label] TRIS f

Operands: f = 5, 6 or 7

Operation: (W) → TRIS register f

Status Affected: None

Encoding:

0000	0000	0fff

Description: TRIS register f (f = 5, 6, or 7) is loaded with the contents of the W register.

Words: 1

Cycles: 1

Example:
```
TRIS    0x5A
```
Before Instruction
W = 0XA5
After Instruction
TRISA = 0XA5

XORLW Exclusive OR literal with W

Syntax: [label] XORLW k

Operands: 0 ≤ k ≤ 255

Operation: (W).XOR.k → (W)

Status Affected: Z

Encoding:

1111	kkkk	kkkk

Description: The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example:
```
XORLW   0xAF
```
Before Instruction
W = 0xB5
After Instruction
W = 0x1A

XORWF Exclusive OR W with f

Syntax: [label] XORWF f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

Operation: (W).XOR.(f) → (dest)

Status Affected: Z

Encoding:

0001	10df	ffff

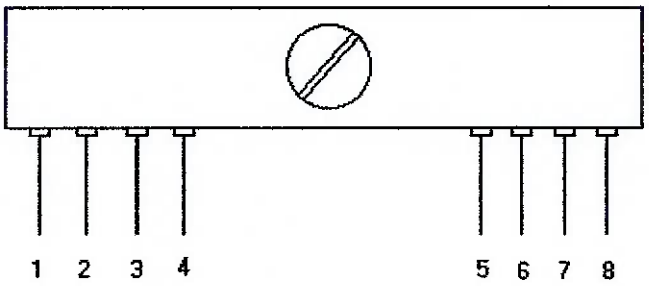
Description: Exclusive OR the contents of the W register with register f. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register f.

Words: 1

Cycles: 1

Example:
```
XORWF   REG,1
```
Before Instruction
REG = 0xAF
W = 0xB5
After Instruction
REG = 0x1A
W = 0xB5

Apêndice C – Informações sobre o receptor



Pino	Função
1	GND – 0 volts
2	Saída digital
3	Saída analógica
4	Vcc – 5 volts
5	Vcc – 5 volts
6	GND – 0 volts
7	GND – 0 volts
8	Entrada - antena

Apêndice D – Informações sobre o L293

SGS-THOMSON MICROELECTRONICS

L293B
L293E

PUSH-PULL FOUR CHANNEL DRIVERS

- OUTPUT CURRENT 1A PER CHANNEL
- PEAK OUTPUT CURRENT 2A PER CHANNEL (non repetitive)
- INHIBIT FACILITY
- HIGH NOISE IMMUNITY
- SEPARATE LOGIC SUPPLY
- OVERTEMPERATURE PROTECTION

DIP16

ORDERING NUMBER : L293B

POWERDIP (16 + 2+ 2)

ORDERING NUMBER : L293E

DESCRIPTION

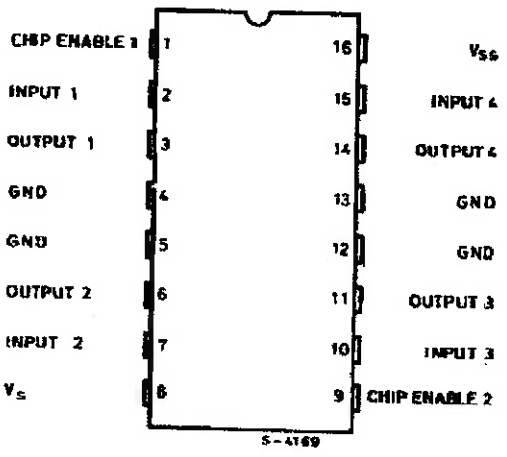
The L293B and L293E are quad push-pull drivers capable of delivering output currents to 1A per channel. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation.

Additionally, the L293E has external connection of sensing resistors, for switchmode control.

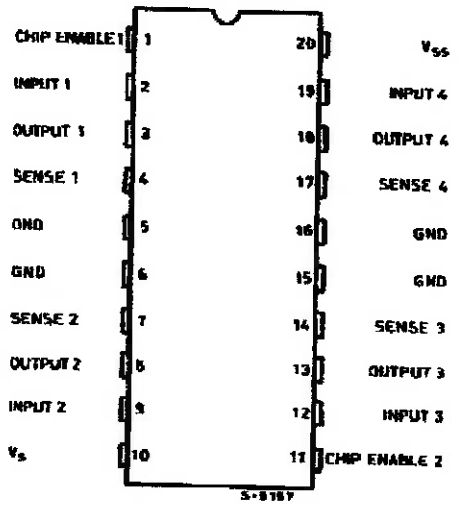
The L293B and L293E are package in 16 and 20-pin plastic DIPs respectively ; both use the four center pins to conduct heat to the printed circuit board.

PIN CONNECTIONS

DIP16 - L293B

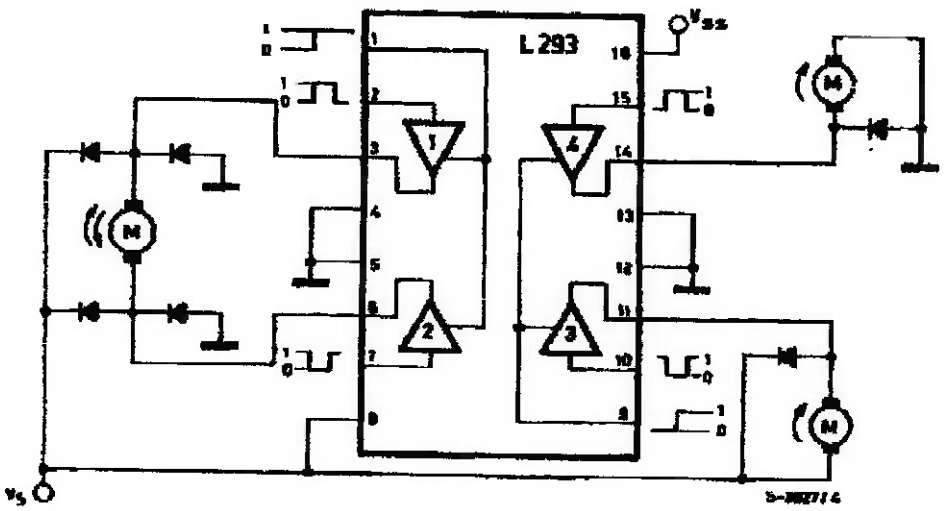


POWERDIP (16+2+2) - L293E

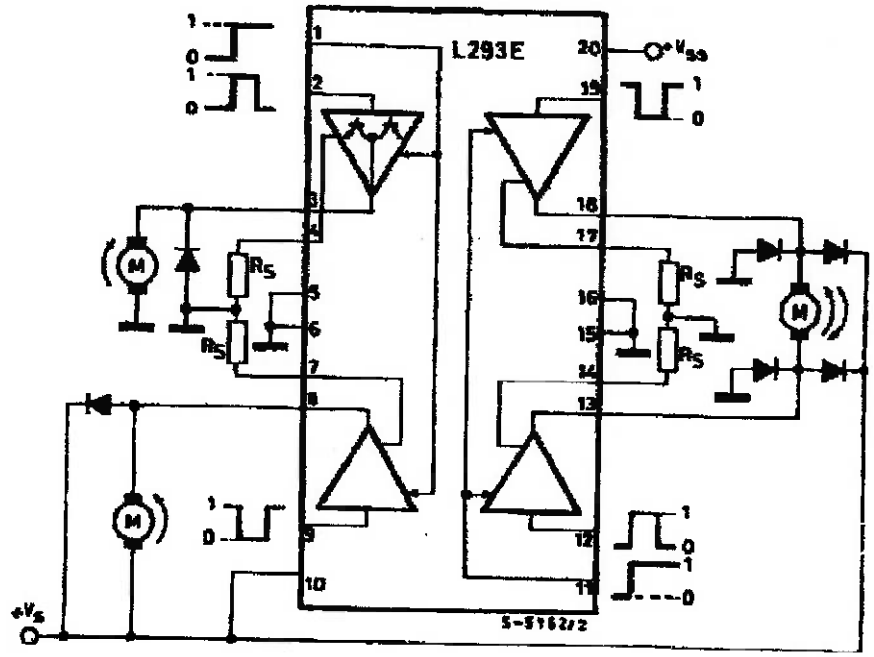


BLOCK DIAGRAMS

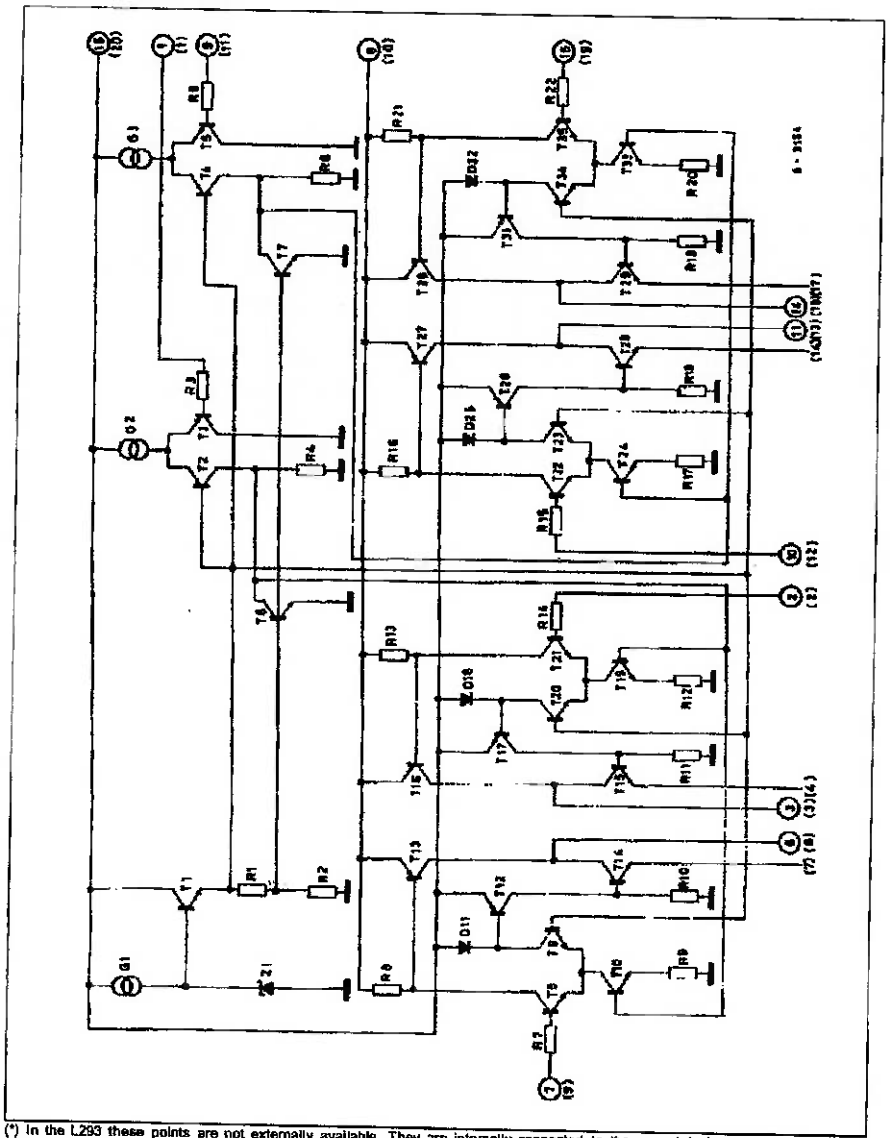
DIP16 - L293B



POWERDIP (16+2+2) - L293E



SCHEMATIC DIAGRAM



(*) In the L293 these points are not externally available. They are internally connected to the ground (substrate).
O Pins of L293 () Pins of L293E.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply Voltage	36	V
V_{ss}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{inh}	Inhibit Voltage	7	V
I_{o4}	Peak Output Current (non repetitive t = 5ms)	2	A
P_{tot}	Total Power Dissipation at $T_{ground-pins}$ = 80°C	5	W
T_{stg}, T_j	Storage and Junction Temperature	-40 to +150	°C

THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	80	°C/W

ELECTRICAL CHARACTERISTICS

For each channel, V_S = 24V, V_{SS} = 5V, T_{amb} = 25°C, unless otherwise specified

Symbol	Parameter	Test Conditions	Min.	TYp.	Max.	Unit
V_s	Supply Voltage		V_{ss}		36	V
V_{ss}	Logic Supply Voltage		4.5		36	V
I_s	Total Quiescent Supply Current	V_i = L I_o = 0 V_{inh} = H V_i = H I_o = 0 V_{inh} = H V_{inh} = L		2 16	6 24 4	mA
I_{ss}	Total Quiescent Logic Supply Current	V_i = L I_o = 0 V_{inh} = H V_i = H I_o = 0 V_{inh} = H V_{inh} = L		44 16 16	60 22 24	mA
V_{iL}	Input Low Voltage		-03.		1.5	V
V_{iH}	Input High Voltage	$V_{SS} \leq 7V$ $V_{ss} > 7V$	2.3 2.3		V_{ss} 7	V
I_{iL}	Low Voltage Input Current	V_i = 1.5V			-10	µA
I_{iH}	High Voltage Input Current	$2.3V \leq V_{IH} \leq V_{ss} - 0.6V$		30	100	µA
V_{inhL}	Inhibit Low Voltage		-0.3		1.5	V
V_{inhH}	Inhibit High Voltage	$V_{SS} \leq 7V$ $V_{ss} > 7V$	2.3 2.3		V_{ss} 7	V
I_{inhL}	Low Voltage Inhibit Current	V_{inhL} = 1.5V		-30	-100	µA
I_{inhH}	High Voltage Inhibit Current	$2.3V \leq V_{inhH} \leq V_{ss} - 0.6V$			±10	µA
V_{CEsatH}	Source Output Saturation Voltage	I_o = -1A		1.4	1.8	V
V_{CEsatL}	Sink Output Saturation Voltage	I_o = 1A		1.2	1.8	V
V_{SENS}	Sensing Voltage (pins 4, 7, 14, 17) (**)				2	V
t_r	Rise Time	0.1 to 0.9 V_o (*)		250		ns
t_f	Fall Time	0.9 to 0.1 V_o (*)		250		ns
t_{on}	Turn-on Delay	0.5 V_i to 0.5 V_o (*)		750		ns
t_{off}	Turn-off Delay	0.5 V_i to 0.5 V_o (*)		200		ns

* See figure 1
** Referred to L293E

TRUTH TABLE

V_i (each channel)	V_o	V_{inh} (**)
H	H	H
L	L	H
H	X (*)	L
L	X (*)	L

(*) High output impedance
(**) Relative to the considerate channel

Figure 1 : Switching Timers

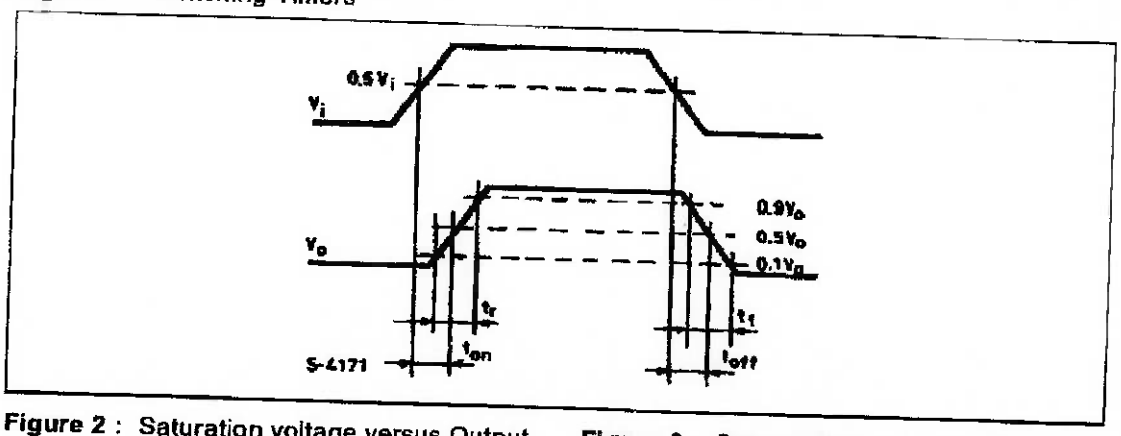


Figure 2 : Saturation voltage versus Output Current

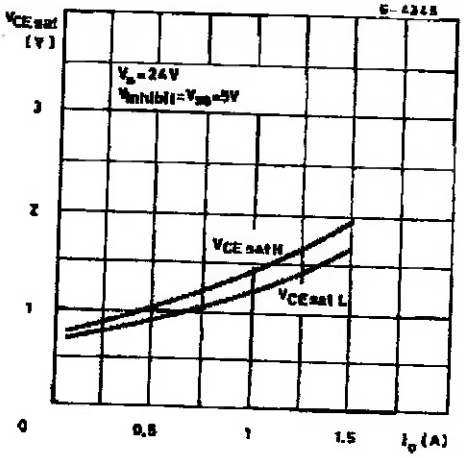


Figure 3 : Source Saturation Voltage versus Ambient Temperature

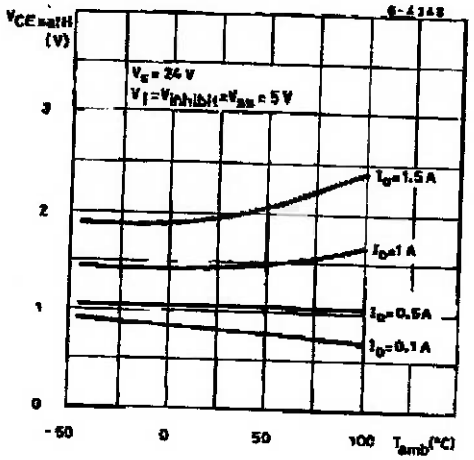


Figure 4 : Sink Saturation Voltage versus Ambient Temperature

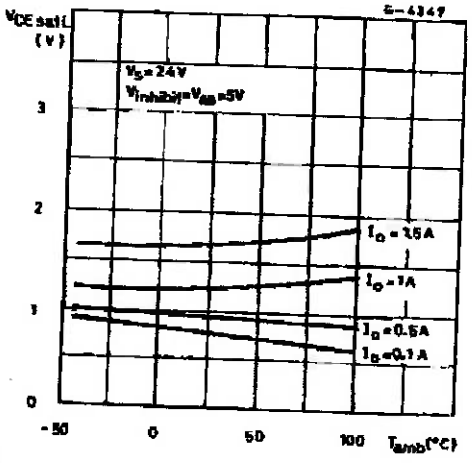
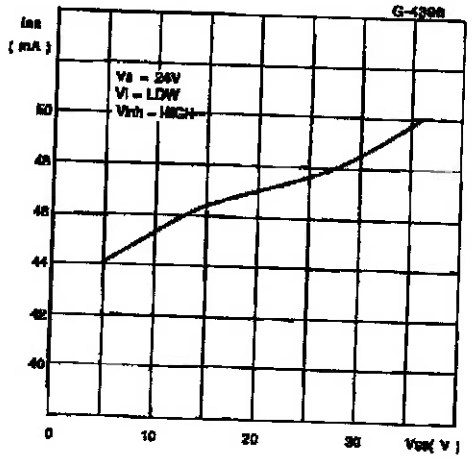


Figure 5 : Quiescent Logic Supply Current versus Logic Supply Voltage



SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

APPLICATIONS OF MONOLITHIC BRIDGE DRIVERS

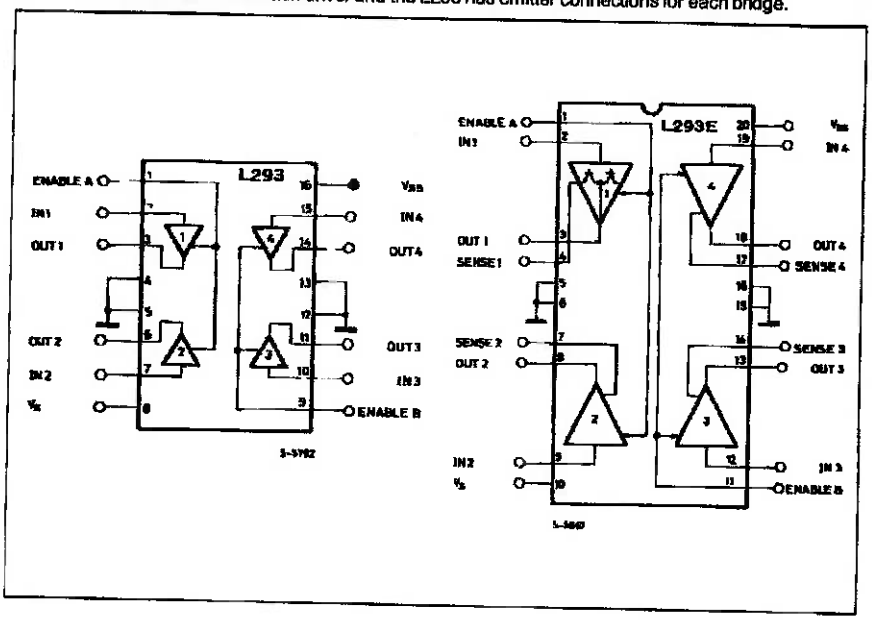
High power monolithic bridge drivers are an attractive replacement for discrete transistors and half bridges in applications such as DC motor and stepper motor driving. This application guide describes three such devices - the L293, L293E and L298 - and presents practical examples of their application.

The L293, L293E and L298 each contain four push-pull power drivers which can be used independently or, more commonly, as two full bridges. Each driver is controlled by a TTL-level logic input and each pair of drivers is equipped with an enable input which controls a whole bridge. All three devices feature a separate logic supply input so that the logic can be run on a lower supply voltage, reducing dissipation. This logic supply is internally regulated.

Additionally, the L293E and L298 are provided with external connections to the lower emitters of each bridge to allow the connection of current sense resistors. The L293E has separate emitter connections for each channel ; the L298 has two, one for each bridge.

Figure 1 shows the internal structure of the L293, L293E and L298. The L293 and L293E are represented as four push pull drivers while the internal schematic is given for the L298. Though they are drawn differently the L293E and L298 are identical in structure ; the L293 differs in that it does not have external emitter connections.

Figure 1 : The L293, L293E and L298 contain four push pull drivers. Each driver is controlled by a logic input and each pair (a bridge) is controlled by an enable input. Additionally, the L293E has external emitter connections for each driver and the L298 has emitter connections for each bridge.



Apêndice E – Listagem do programa do microcontrolador

```
;*************************************************************************
;       Programa Receptor   C.A.L.C.  Robô 00     full step 2 phase on  *
;*************************************************************************

        LIST    p=16C54 ; PIC16C55 is the target processor
        #include "P16C5x.INC" ; Include header file
;
; posições RAM utilizadas no programa
;

tempo   equ     H'08'
dado    equ     H'09'
treto   equ     H'0A'
tgiro   equ     H'0B'
tfire   equ     H'0C'
trat    equ     H'0D'
giro    equ     H'0E'
reto    equ     H'0F'
cte     equ     H'10'
        org     0
        goto    config

;*************************************************************************
; Loop de tempo para recepção                                            *
;*************************************************************************
; Modifica o divisor do timer para mesma base de tempo do receptor      *
; A rotina permanece em loop até o valor do timer ser igual ao valor    *
; da variável tempo, que é a cte de tempo da transmissão via rádio.     *
; Para verificar se o timer é igual a tempo, subtrai-se um do outro     *
; e testa-se o bit 2 do registrador STATUS, afetado em operações        *
; matemáticas                                                            *
;*************************************************************************

loopr   movlw   H'03'
        OPTION
        clrf    TMR0
rl      movf    TMR0,0
        subwf   tempo,0
        btfss   STATUS,2
        goto    rl
        retlw   00H

;*************************************************************************
; Loop de tempo para movimento                                           *
;*************************************************************************
; Modifica o divisor do timer para a base de tempo de movimentação      *
; A rotina permanece em loop até o valor do timer ser igual ao valor    *
; da variável cte, que é um valor adequado ao motor de passo empregado.*
; Para verificar se o timer é igual a cte, subtrai-se um do outro       *
; e testa-se o bit 2 do registrador STATUS, afetado em operações        *
; matemáticas                                                            *
;*************************************************************************

loopm   movlw   H'06'
        OPTION
        clrf    TMR0
m1      movf    TMR0,0
        subwf   cte,0
        btfss   STATUS,2
        goto    m1
        retlw   00H
```

```
;*******************************************************************************
; Rotina de recepção e identificação                                           *
;*******************************************************************************
; A rotina a seguir é responsável pela decodificação da onda recebida          *
; do rádio e o dado fica armazenado na variável dado.                          *
; O processador fica em loop até receber o gatilho de recepção, utilizado      *
; para estabelecer uma constante de tempo para a amostragem do sinal.          *
; A cada intervalo da constante de tempo deve estar presente na via            *
; de comunicação um bit diferente do dado inteiro, composto por oito bits.     *
; A passagem do sinal recebido para a variável dado é feita da seguinte forma: *
; testa-se o estado do bit 0 da porta A de i/o e o bit correspondente da        *
; variável dado é setado ou resetado conforme o caso.                          *
; Além disso a rotina testa os bits 6 e 7 para identificar o robô correspondente. *
; Se os bits de identificação correspondem aos do robô, a rotina de decodificação *
; continua, caso contrário o programa retorna para start.                      *
;*******************************************************************************
; Variáveis afetadas: dado, tempo                                              *
;*******************************************************************************

recep   clrf    dado
        clrf    tempo
        btfss   PORTA,0
          goto  recep
        movlw   H'03'
        OPTION
        clrf    TMR0
p1      btfsc   PORTA,0
          goto  p1
        movf    TMR0,0
        movwf   tempo
        call    loopr
        btfss   PORTA,0
         bcf    dado,0
        btfsc   PORTA,0
         bsf    dado,0
        call    loopr
        btfss   PORTA,0
         bcf    dado,1
        btfsc   PORTA,0
         bsf    dado,1
        call    loopr
        btfss   PORTA,0
         bcf    dado,2
        btfsc   PORTA,0
         bsf    dado,2
        call    loopr
        btfss   PORTA,0
         bcf    dado,3
        btfsc   PORTA,0
         bsf    dado,3
        call    loopr
        btfss   PORTA,0
         bcf    dado,4
        btfsc   PORTA,0
         bsf    dado,4
        call    loopr
        btfss   PORTA,0
         bcf    dado,5
        btfsc   PORTA,0
         bsf    dado,5
        call    loopr
        btfss   PORTA,0
         bcf    dado,6
        btfsc   PORTA,0
         bsf    dado,6
        call    loopr
        btfss   PORTA,0
```

```
        bcf     dado,7
       btfsc    PORTA,0
        bsf     dado,7
       call     loopr
       btfsc    dado,7
         goto   start
       btfsc    dado,6
         goto   start
       goto     trata

;*****************************************************************************************
; Rotina de tratamento                                                                   *
;*****************************************************************************************
; Esta subrotina continua a decodificação do dado recebido.                             *
; Uma vez que o dado é endereçado a este robô, é preciso saber se diz respeito a        *
; uma rotação, uma translação ou disparo. Para isso, testa-se os bits 4 e 5 de dado     *
; repitindo-os nos bits 0 e 1 da variável trat. Ela é então decrementada até zerar      *
; e quando isso acontece o processamento continua com a decodificação, caso             *
; contrário o programa retorna para start.                                              *
;*****************************************************************************************
; Obs: A variável treto é responsável pelo decremento e tem valor 01H                   *
;*****************************************************************************************
; Variáveis afetadas: trat                                                              *
;*****************************************************************************************

trata   clrf    trat
       btfsc    dado,5
        bsf     trat,1
       btfsc    dado,4
        bsf     trat,0
       movf     trat,0
       subwf    tgiro,0
       btfsc    STATUS,2
        goto    tgirar
       movf     trat,0
       subwf    treto,0
       btfsc    STATUS,2
        goto    ttransl
       movf     trat,0
       subwf    tfire,0
       btfsc    STATUS,2
        goto    fire
       goto     start

;*****************************************************************************************
; Armazena dado de rotação                                                              *
;*****************************************************************************************
; Esta subrotina armazena valor referente à rotação.                                    *
; Uma vez que o dado é referente à rotação, o valor da variável dado é gravado na       *
; variável giro e modificado para melhor funcionamento dos motores de passo.            *
; Após esse cálculo, o programa retorna para start.                                     *
;*****************************************************************************************
; Variáveis afetadas: giro                                                              *
;*****************************************************************************************

tgirar  movf    dado,0
       movwf    giro
       bcf      giro,7
       bcf      giro,6
       bcf      giro,5
       bcf      giro,4
       rlf      giro,1
       bcf      giro,0
       goto     start

;*****************************************************************************************
; Armazena dado de translação                                                           *
;*****************************************************************************************
; Esta subrotina armazena valor referente à translação.                                 *
; Uma vez que o dado é referente à translação, o valor da variável dado é gravado na    *
; variável reto e modificado para melhor funcionamento dos motores de passo.            *
```

```
; Após esse cálculo, o programa retorna para start.                          *
;*****************************************************************************
; Variáveis afetadas: reto                                                   *
;*****************************************************************************
ttransl movf    dado,0
        movwf   reto
        bcf     reto,7
        bcf     reto,6
        bcf     reto,5
        bcf     reto,4
        rlf     reto,1
        bcf     reto,0
        goto    start

;*****************************************************************************
; Rotina de disparo                                                          *
;*****************************************************************************
; Esta subrotina desempenha a movimentação do robô.                          *
; Uma vez que o dado é referente ao disparo, a rotina gerencia a trajetória do robô,  *
; acionando as subrotinas de rotação e translação com os valores de giro e reto que   *
; estão armazenados. Após o cumprimento destas tarefas, o programa retorna para start.*
;*****************************************************************************
; Variáveis afetadas: nenhuma                                                *
;*****************************************************************************

fire    btfsc   giro,4
        goto    giracw
        goto    giraccw
fil     bcf     PORTA,1
        btfsc   reto,4
        goto    frente
        goto    trás
        goto    start

;*****************************************************************************
; Gira sentido horário                                                       *
;*****************************************************************************
; Esta subrotina desempenha rotação no sentido horário.                      *
; A rotina escreve as palavras correspondentes ao acionamento dos motores de passo    *
; para giro no sentido horário espaçadas no tempo de loopm. A cada loop da rotina a   *
; variável giro é decrementada até zerar, quando o processamento retorna para a       *
; rotina de disparo.                                                         *
;*****************************************************************************
; Variáveis afetadas: giro                                                   *
;*****************************************************************************

giracw  bcf     giro,4
        rlf     giro,1
        bcf     giro,0
gcw1    clrw
        subwf   giro,0
        btfsc   STATUS,2
        goto    fil
        bsf     PORTA,1
        decf    giro,1
        movlw   B'01010110'
        movwf   PORTB
        call    loopm
        movlw   B'10011010'
        movwf   PORTB
        call    loopm
        movlw   B'10101001'
        movwf   PORTB
        call    loopm
        movlw   B'01100101'
        movwf   PORTB
        call    loopm
        goto    gcw1
```

```
;*************************************************************************************
; Gira sentido anti-horário                                                          *
;*************************************************************************************
; Esta subrotina desempenha rotação no sentido anti-horário.                         *
; A rotina escreve as palavras correspondentes ao acionamento dos motores de passo   *
; para giro no sentido anti-horário espaçadas no tempo de loopm. A cada loop da      *
; rotina a variável giro é decrementada até zerar, quando o processamento retorna    *
; para a rotina de disparo.                                                          *
;*************************************************************************************
; Variáveis afetadas: giro                                                           *
;*************************************************************************************

giraccw bcf     giro,4
        rlf     giro,1
        bcf     giro,0
gccw1   clrw
        subwf   giro,0
        btfsc   STATUS,2
        goto    fi1
        bsf     PORTA,1
        decf    giro,1
        movlw   B'01100101'
        movwf   PORTB
        call    loopm
        movlw   B'10101001'
        movwf   PORTB
        call    loopm
        movlw   B'10011010'
        movwf   PORTB
        call    loopm
        movlw   B'01010110'
        movwf   PORTB
        call    loopm
        goto    gccw1

;*************************************************************************************
; Translada para frente                                                              *
;*************************************************************************************
; Esta subrotina desempenha translação.                                              *
; A rotina escreve as palavras correspondentes ao acionamento dos motores de passo   *
; para translação espaçadas no tempo de loopm. A cada loop da                        *
; rotina a variável reto é decrementada até zerar, quando o processamento retorna    *
; para a rotina de disparo.                                                          *
;*************************************************************************************
; Variáveis afetadas: reto                                                           *
;*************************************************************************************

frente  bcf     reto,4
        rlf     reto,1
        bcf     reto,0
fr1     clrw
        subwf   reto,0
        btfsc   STATUS,2
        goto    start
        decf    reto,1
        bsf     PORTA,2
        movlw   B'01010101'
        movwf   PORTB
        call    loopm
        movlw   B'10011001'
        movwf   PORTB
        call    loopm
        movlw   B'10101010'
        movwf   PORTB
        call    loopm
        movlw   B'01100110'
        movwf   PORTB
```

```
        call    loopm
        goto    fr1


;*****************************************************************************************
; Translada para trás                                                                    *
;*****************************************************************************************
; Esta subrotina desempenha translação.                                                  *
; A rotina escreve as palavras correspondentes ao acionamento dos motores de passo       *
; para translação espaçadas no tempo de loopm. A cada loop da                            *
; rotina a variável reto é decrementada até zerar, quando o processamento retorna        *
; para a rotina de disparo.                                                              *
;*****************************************************************************************
; Variáveis afetadas: reto                                                               *
;*****************************************************************************************

trás    bcf     reto,4
        rlf     reto,1
        bcf     reto,0
tr1     clrw
        subwf   reto,0
        btfsc   STATUS,2
        goto    start
        decf    reto,1
        bsf     PORTA,2
        movlw   B'01100110'
        movwf   PORTB
        call    loopm
        movlw   B'10101010'
        movwf   PORTB
        call    loopm
        movlw   B'10011001'
        movwf   PORTB
        call    loopm
        movlw   B'01010101'
        movwf   PORTB
        call    loopm
        goto    tr1

;*****************************************************************************************
; Configuração                                                                           *
;*****************************************************************************************
; Esta subrotina configura as portas de i/o e os endereços das variáveis.                *
;*****************************************************************************************

config  movlw   01H
        TRIS    PORTA
        movlw   00H
        TRIS    PORTB
        movlw   H'00'
        movwf   tgiro
        movlw   H'01'
        movwf   treto
        movlw   H'02'
        movwf   tfire
        movlw   H'FF'
        movwf   cte
        clrf    PORTB
        goto    start

;***********************
; Início do programa   *
;***********************

start   bcf     PORTA,2
        btfss   PORTA,0
        goto    start
        goto    recep
        END
```