10,0 dez

**Eduardo Pasianot**

# FUTEBOL DE ROBÔS AUTÔNOMOS : TRANSMISSÃO DE DADOS VIA RF E VISÃO COMPUTACIONAL

**Trabalho de formatura para graduação em Engenharia Mecatrônica na Escola Politécnica da USP**

**Orientador: Prof. Marcos Ribeiro Pereira Barretto**

**São Paulo**

**2001**

## Sumário

## Índice de figuras

## Índice de tabelas

## Resumo

Este trabalho é o resultado das pesquisas e projetos realizados sobre o tema futebol de robôs autônomos, mais precisamente sobre os sub-sistemas de transmissão de dados via RF e visão computacional.

Primeiramente será feita uma exposição geral do tema futebol de robôs autônomos bem como um apanhado acerca das dificuldades envolvidas. Em seguida será iniciada a etapa de projeto da transmissão de dados via RF, situando-a no tema e discorrendo sobre suas viabilidades técnica e financeira. Serão apresentadas algumas soluções para o problema e uma delas será escolhida para implementação.

Na implementação constarão requisitos de desempenho, listagens dos programas envolvidos e esquemas do hardware empregado na solução. Finda a implementação serão expostos resultados de testes de desempenho e falhas, além de apresentação de possíveis refinamentos.

A seguir serão discutidas a viabilidade técnica e financeira da visão computacional para futebol de robôs autônomos, situando-a no tema e abordando as dificuldades relacionadas à sua implementação. Após isso haverá uma exposição das soluções propostas e a escolha de uma delas para estudo e realização. Terminada essa etapa as características necessárias do sistema serão discutidas e apresentar-se-á o software envolvido no sistema, que será testado e terá seus resultados analisados.

Por fim, a conclusão levantará, além de um apanhado dos dois projetos, uma observação pessoal sobre os estudos necessários para a realização deste trabalho.

# 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 falhas e ruídos 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, 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 Carnelos e Rodrigo de Deus Reinaldo. Este trabalho se propõe a documentar todos os testes, protótipos e resultados referentes ao projeto da visão computacional e transmissão de dados via RF.

## 1.1. Motivação

O tema futebol de robôs diz respeito a 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 desde 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 auxílio 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 específicas também as dimensões dos jogadores, campo e bola.

A equipe que será modelada corresponde à categoria MiroSot segundo o regulamento da FIRA.

## 2. Definição do problema

Tomando o futebol de robôs como um sistema, pode-se dividi-lo em quatro grandes células interligadas [7]:

⇒ Controle dos motores (hardware do robô)

⇒ Interface computador - robô (transmissão via rádio freqüência)

⇒ Estratégia (software de controle)

⇒ Visão computacional (reconhecimento de imagens)

A seguir cada módulo será descrito resumidamente.

### 2.1. Controle dos motores

O controle dos motores pode ser realizado de várias formas distintas; uma delas consiste no emprego de servo-motores comerciais 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 por uma 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 [4][5][6].

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 para o acionamento dos 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.

## 2.2. 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 é feita pela porta serial de um computador pessoal e um transmissor RC Futaba Skysport, com um canal do receptor direcionado para cada motor; ou ainda utilizando a saída paralela do computador, como fazem os coreanos da equipe ROBOTIS [8], 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 módulo de rádio freqüência. 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.

## 2.3. 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 tão grande relevância a 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 campeã mundial ROBOTIS utilizam um processamento de 60 ou mais quadros por segundo.

## 2.4. 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.

Já que as possibilidades de posicionamento dos robôs e da bola são infinitas, 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.

# 3. Aprofundamento Teórico

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

## 3.1. Sistema Futebol de Robô

Analisando 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.

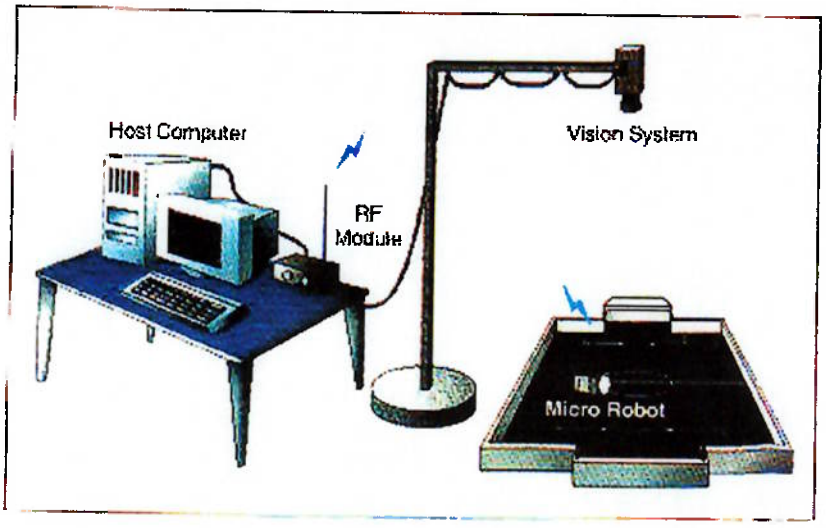



*Figura 3.1.2: Esquema de funcionamento.*

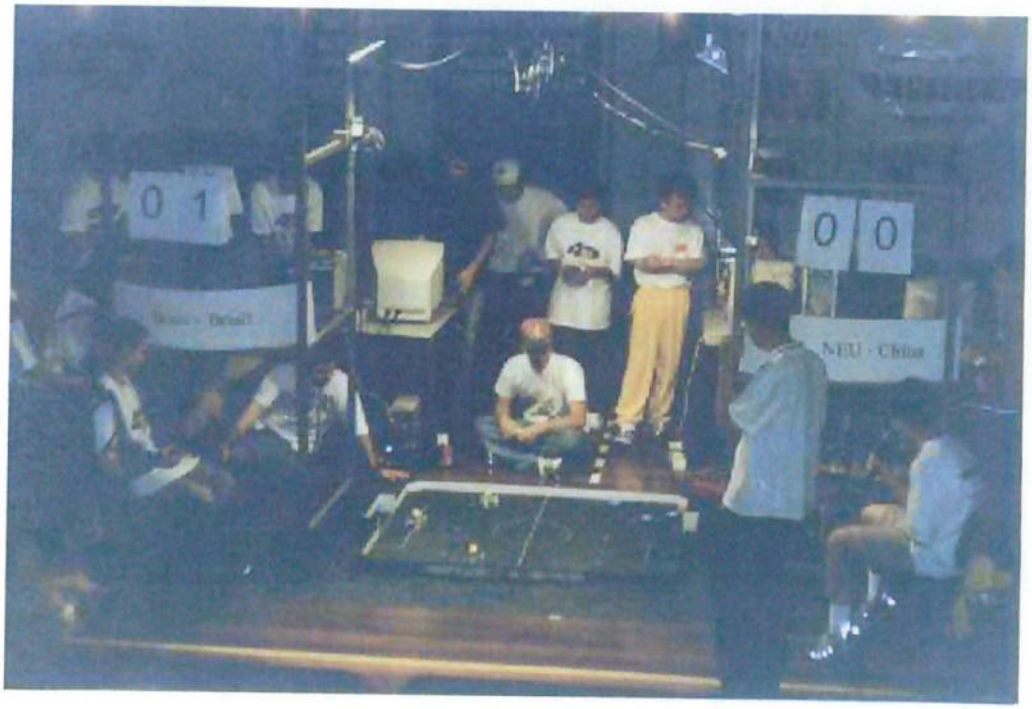



*Figura 3.1.1: Foto ilustrativa de um jogo.*

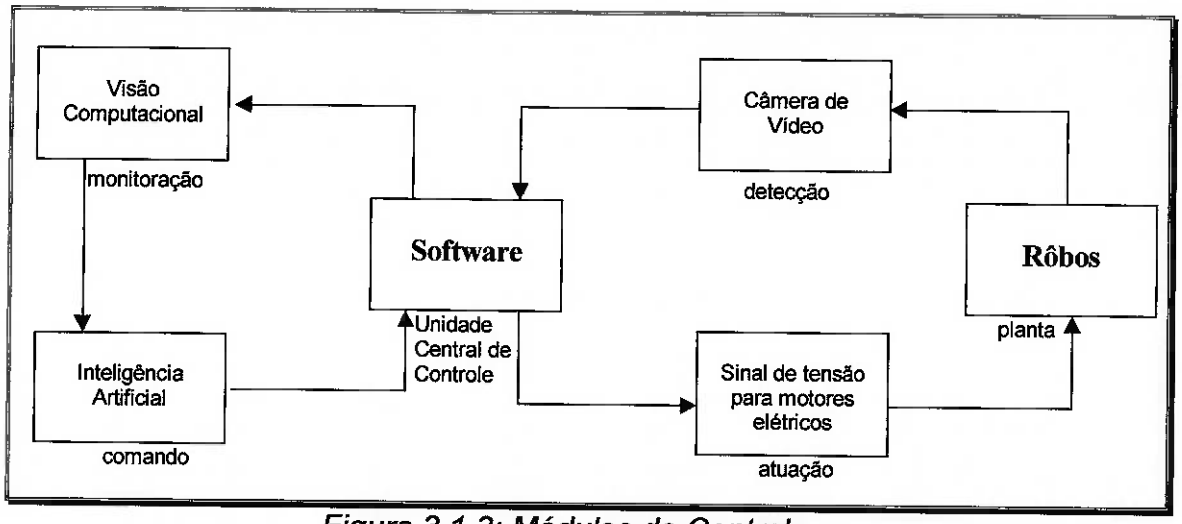



*Figura 3.1.3: Módulos de Controle.*

Na figura acima tem-se 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 como coordenadas a outro módulo do software de controle, o módulo de estratégia 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.

## 3.2. Estrutura montada

Para a viabilização da equipe e dos testes que possibilitaram este projeto, primeiramente foi montada toda um infra-estrutura de laboratório, desde o espaço físico ate a instrumentação, que será detalhada nesta seção.


*Figura 3.2.1: fonte reguladora de tensão.*

Como será visto na seqüência do projeto, microprocessadores 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.

*Figura 3.2.2: Apagador de memória EPROM.*

O campo de jogo foi confeccionado em madeira segundo as regras da FIRA para 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 consta das regras.

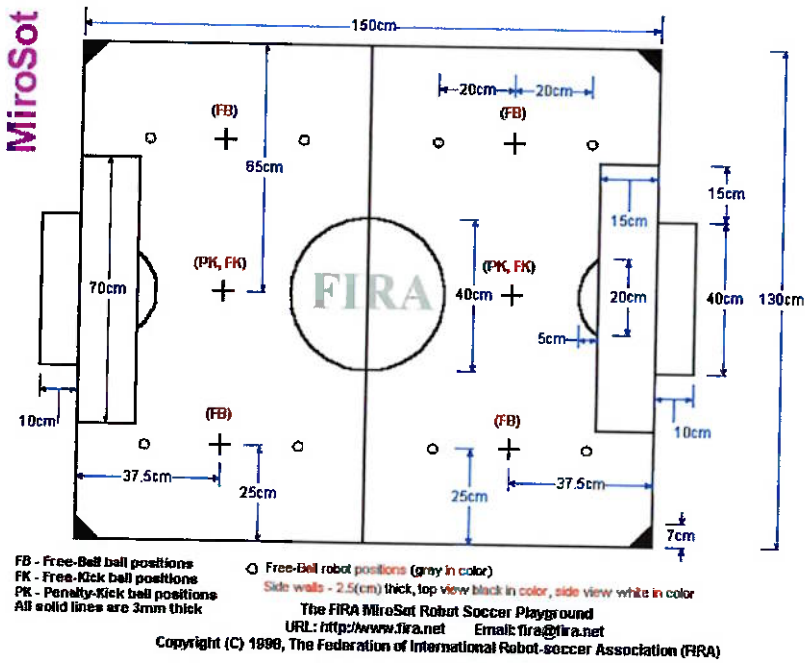



*Figura 3.2.3: dimensões do campo.*


*Figura 3.2.4: dimensões do campo.*

Também foi necessária a construção de estrutura que posicionasse a câmera de vídeo a uma altura de 2,40 m de altura (regra), e esta foi projetada e construída (2,20 m no laboratório) com cantoneiras de 19,3 x 40,9 x 2,9 mm, e parafusos M8 de cabeça sextavada. Além da câmera, estão presentes na estrutura duas lâmpadas de 500W cada (halógenos 500W IP44) presas com mãos francesas de alumínio, para a iluminação do campo.

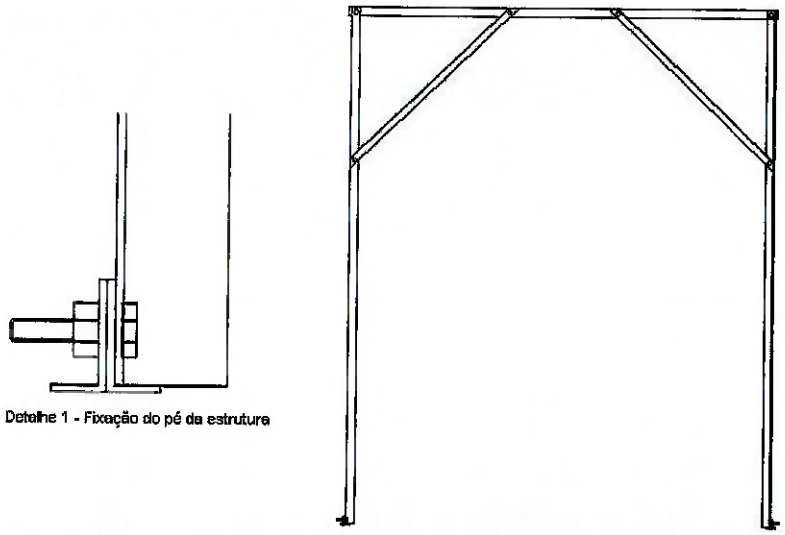



*Figura 3.2.5: desenho da estrutura.*


*Figura 3.2.6: foto da estrutura com a câmera.*

# 4. Estudo da viabilidade da transmissão de dados via RF

Nos próximos dois sub itens, serão discutidas a viabilidade técnica e financeira do projeto de transmissão de dados via RF. Vale salientar neste momento que serão discutidas apenas a viabilidade da transmissão de dados e da visão computacional, sendo que os outros módulos serão desenvolvidos nos outros trabalhos referentes ao futebol de robôs.

## 4.1. Viabilidade técnica da transmissão de dados por RF

A transmissão de dados via rádio é comprovadamente viável sob o ponto de vista técnico dada a variedade de veículos de comunicação vigentes na atualidade que utilizam como portadoras de sinal as ondas de rádio freqüência (RF). Além disso, existem inúmeras equipes de futebol de robôs que participam das competições organizadas pela FIRA que empregam estes métodos de comunicação.

A transmissão de dados via rádio caracteriza-se pelo modo de transmissão de comandos entre o computador que calcula as estratégias e os jogadores. Para realizar esta tarefa é possível empregar diversos meios, entre eles a modulação em freqüência, amplitude e micro-ondas.

Competitivamente, há dois grupos principais de soluções empregadas em maior número nas equipes: rádios inteiramente comerciais para uso em aeromodelos e rádios específicos para o futebol de robôs, normalmente desenvolvidos pelas próprias equipes. Ambos constituem soluções possíveis e serão explicados mais adiante. Por hora, estes modelos indicam algumas necessidades para o projeto da transmissão: interface digital com o computador, capacidade de comunicação unidirecional, isto é, o fluxo de dados sempre se dirige do computador para os robôs, endereçamento, que consiste na possibilidade de tornar os dados acessíveis apenas ao robô

desejado e também é característica requerida a velocidade de transmissão, que deve ser tal a não comprometer o desempenho geral do sistema.

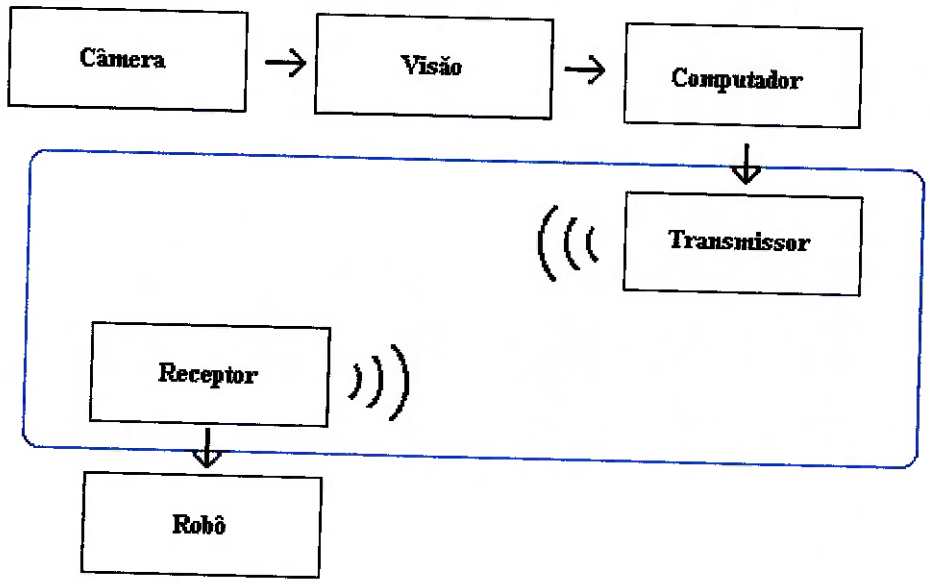



*Figura 4.1.1: Transmissão de dados (em azul).*

## 4.2. Viabilidade financeira da transmissão de dados via RF

Já que a viabilidade técnica não impõe restrições, o grande fator limitante do projeto será a dificuldade para a aquisição de equipamentos necessários para a execução do mesmo. Como a verba limitada, o projeto fica restrito a utilizar alguns equipamentos disponíveis a baixo custo, que invariavelmente dispõe de menos recursos e têm pior desempenho. Apesar disso, a pouca disponibilidade de equipamentos não limitará a formulação das soluções propostas para o problema.

Pode-se concluir desta forma que este trabalho propõe-se 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 verba reduzida.

## 4.3. Apresentação de possíveis soluções

Dentre todas as soluções que serão propostas é necessário dizer que o projeto mecânico dos robôs e a eletrônica embarcada serão afetados e modificados para cada caso. A fim de facilitar o entendimento será justaposta a cada solução para a transmissão de dados uma síntese da mecânica e eletrônica envolvidas.

### 4.3.1. Solução 1 – Rádios comerciais de aeromodelos

Esta solução é baseada no emprego de um rádio comercial de no mínimo 6 canais que possui interface serial com um software de controle.

### 4.3.2. Solução 2 – Transmissão digital TTL

Esta solução emprega circuitos integrados TTL para transmitir dados da porta paralela do computador para os robôs utilizando um módulo RF comercial [9] [10].

### 4.3.3. Solução 3 – Transmissão por micro-controlador tipo A

A solução 3 traria ao projeto o uso de micro-controladores, representando uma nova área da engenharia mecatrônica a ser abordada. Seu funcionamento seria parecido com o da solução 2, porém fisicamente menor e mais dispendioso.

### 4.3.4. Solução 4 – Transmissão por micro-controlador tipo B

Como principal diferença em relação às demais soluções está o micro-controlador a ser empregado. Muito poderoso, poderia agregar as funções de recepção de dados e controle dos robôs.

# 5. Detalhamento das soluções para transmissão de dados

Neste tópico, pretende-se detalhar e documentar melhor todos as soluções propostas. Serão reportados todos os desenhos e circuitos, além das mudanças impostas por cada uma nas características gerais do projeto.

## 5.1. Solução 1

Os rádios comerciais de no mínimo 6 canais com porta serial são empregados no controle de helicópteros miniatura e dispõe de interface serial para aulas de treino em pilotagem, onde o transmissor do aluno fica conectado ao do professor. Além disso, o transmissor também é utilizado em softwares de simulação de pilotagem fornecidos pelos fabricantes. A eletrônica embarcada nos robôs neste caso ficaria restrita a apenas um receptor que controlaria dois servo-motores em cada um. Os servo-motores normalmente acompanham o pacote, mas precisam ser modificados para possibilitar o movimento de rotação contínua:

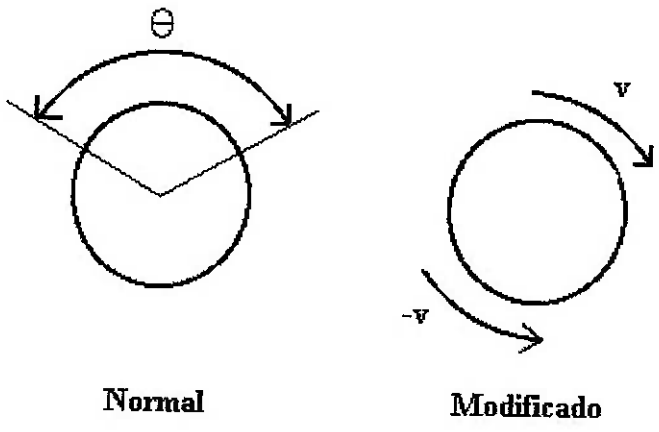



*Figura 5.1.1: servo-motor modificado.*

Quanto aos receptores, precisam ser adquiridos separadamente outros dois, pois são necessários três no total (um por robô), ao passo que no pacote só é acompanhado por um receptor. O projeto da transmissão neste caso fica restrito às rotinas de software de comunicação que o programa de estratégia utilizará como ferramenta para enviar os dados.

As rotinas de geração de trajetória também deverão estar no programa de controle, pois não há processamento de dados nos robôs.

Apesar de não ter sido testada pela equipe, o desempenho desta solução pode ser considerado pobre já que ela desperdiça a capacidade do robô de calcular sua própria trajetória ocupando tempo de processamento no programa de estratégia. Além disso, é a mais dispendiosa financeiramente. Por outro lado, é a de mais fácil implementação, pois é possível eliminar várias etapas do projeto tradicional.

Como curiosidade pode-se citar que esta é a solução mais empregada nas equipes nacionais, pois a maioria é constituída de faculdades de ciências da computação, não interessando a elas o desenvolvimento de hardware.

## 5.2. Solução 2

A solução 2 propõe uma abordagem radicalmente diferente e mais complexa. Constitui-se na utilização de circuitos integrados TTL tanto no transmissor quanto no receptor e lança mão da porta paralela como canal de comunicação com o computador. Totalmente digital, reproduz o estado da porta paralela nos receptores em cada um dos robôs. Esta solução foi implementada em protótipo e testada utilizando-se como módulo RF o modelo Weishing HS-435 (435 MHz), de capacidade de transmissão nominal de 1000 bits por segundo.

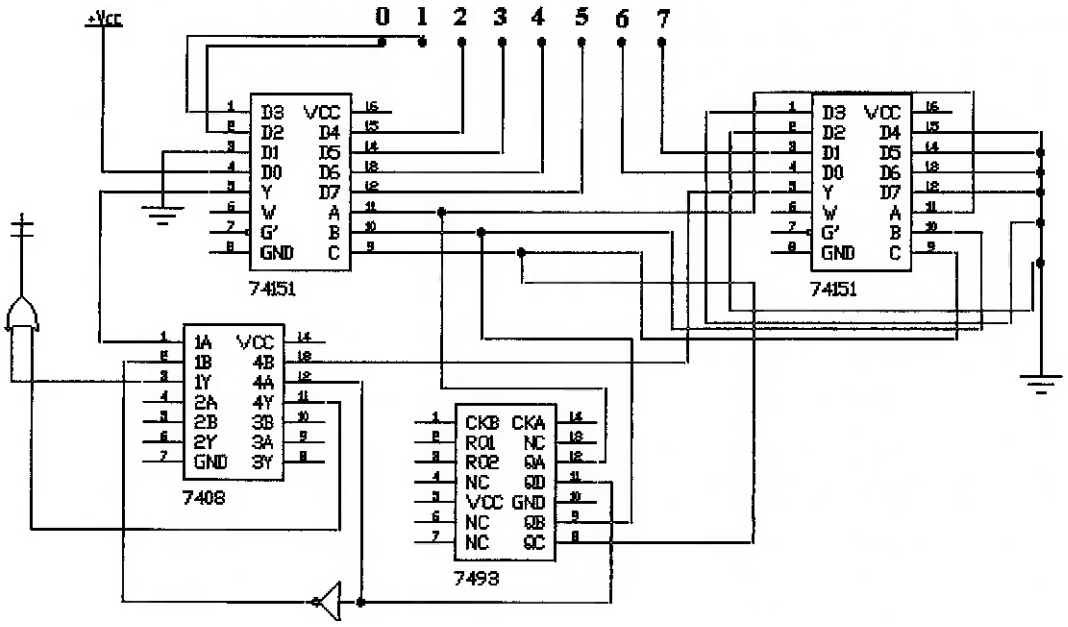



*Figura 5.2.1: transmissor.*

O transmissor funciona da seguinte maneira: o contador de 4 bits 7493, em seu funcionamento controla com seus 3 bits menos significativos os dois multiplexadores 74151 e portanto qual o bit da porta paralela que está na saída Y de cada mux. O quarto bit mais significativo do contador vai habilitar através da porta NAND 7408 qual mux vai mandar a sua saída para o rádio.

O receptor funciona de maneira inversa. Os sinais advindos do rádio são guardados num registrador de deslocamentos 74164, cuja freqüência de clock deve ser o mais próxima possível daquela do transmissor. Quando o contador 7493 atinge o número equivalente a 10 decimal, a lógica do contador passa os últimos 8 bits registrados no 74164 para o octo-flip-flop tipo D 74377, onde se tem então o estado da porta reproduzido.

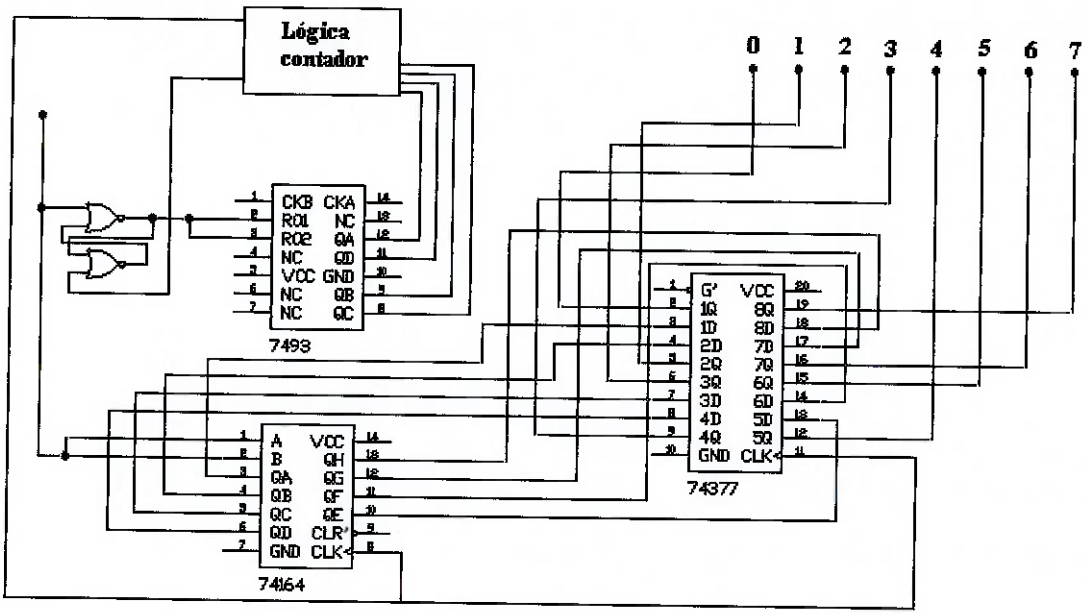



*Figura 5.2.2: receptor.*

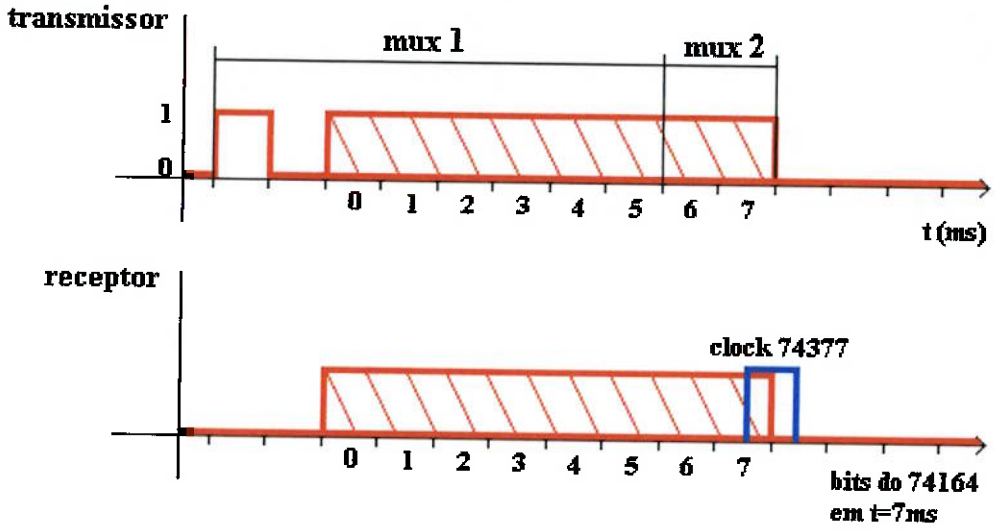



*Figura 5.2.3: diagrama de funcionamento.*

Como o módulo RF Weishing não se mostrou confiável em velocidades de transmissão além de 800 bits/s, a velocidade do transmissor-receptor foi mantida em 500 bits/s por segurança. O principal problema do

conjunto era a falta de sincronismo entre os sinais de clock do transmissor e do receptor, o que ocasionava falhas na transmissão. Além disso, nada impedia que a porta paralela mudasse de estado durante a varredura dos multiplexadores, acarretando mais erros. Extremamente barato, poderia ser utilizado para robôs sem cálculo de trajetória, isto é, apenas amplificando seu sinal para controlar os motores dos robôs. Esta solução também implicaria em cálculo de trajetória no programa de estratégia, proporcionando desempenho similar ao da solução 1.

## 5.3. Solução 3

A solução 3 introduz o uso de micro-controladores da família PIC 16C54 com EPROM. Além de mais atrativa sob o ponto de vista de aprendizado, surge como resposta às falhas da solução anterior [28]. A transmissão de dados confiável é fundamental para a viabilidade do controle de trajetória, função que passa a ser desempenhada por uma eletrônica embarcada mais elaborada, aliviando o computador desta tarefa e otimizando o tempo de resposta do sistema futebol de robôs como um todo.

O novo controle seria feito então da seguinte maneira: a visão computacional informaria o computador das coordenadas dos jogadores, e este, através do software de estratégia calcularia os comandos chave da movimentação. Estes comandos seriam então transmitidos aos robôs e a eletrônica embarcada executaria a trajetória conforme havia sido solicitada pelo software de estratégia.

O micro-controlador PIC permitiu a correção das falhas do transmissor-receptor TTL primeiramente através da inclusão, no transmissor, de um novo bit de interface com a porta paralela. Este bit é representado na figura a seguir como S7 e sua função é comunicar ao computador o estado do periférico com o qual ele está conectado. No caso do transmissor, ele informa ao computador o momento de trocar o estado da porta para não provocar erros de transmissão.

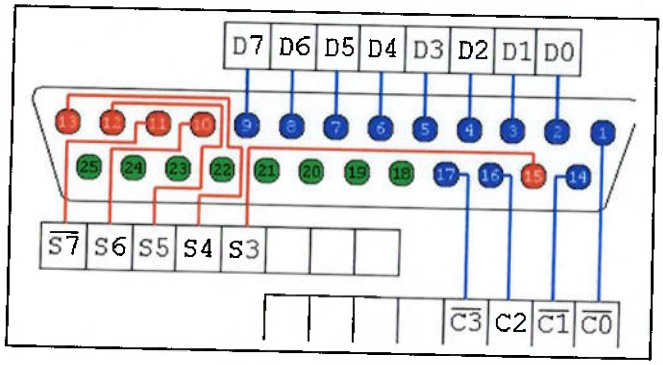



*Figura 5.3.1: porta paralela.*

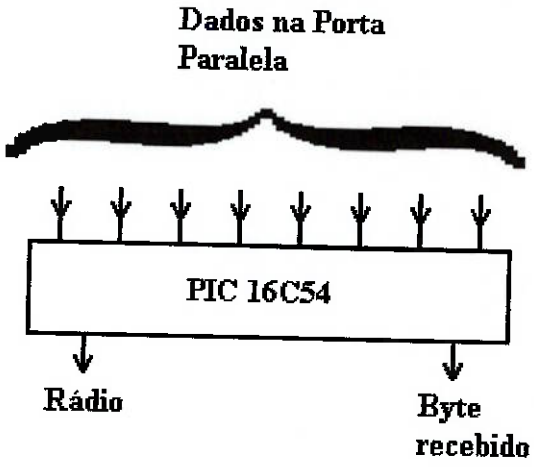



*Figura 5.3.2: esquema do transmissor.*

Para solucionar o erro de sincronismo, foi implementado, graças ao micro-controlador, um laço de amostragem que calcula a base de tempo da transmissão a partir do tempo de duração do gatilho. Esta constante é utilizada então para montar o dado bit a bit da maneira correta. Além disso, esta solução permite o endereçamento de dados. Isto foi implementado utilizando dois pinos do PIC receptor, responsáveis pela identificação do robô. Os robôs podem assumir as identidades 00, 01, 10 e 11. Os dois primeiros bits do dado recebido são comparados à identidade do robô. Caso sejam iguais, o receptor do robô

muda o estado dos seus pinos e produz um sinal de aviso para a eletrônica embarcada a fim de alertá-la sobre o novo comando enviado pelo software de estratégia. Caso os bits de endereçamento sejam diferentes da identidade do robô, o receptor não excita a eletrônica embarcada.

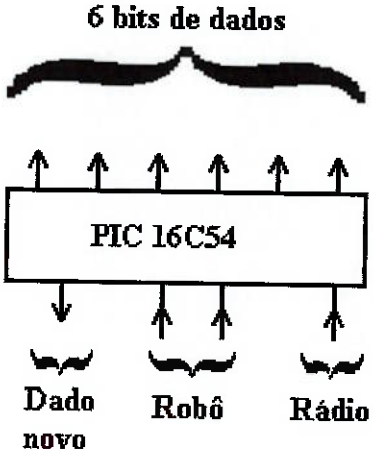



*Figura 5.3.4: esquema do receptor.*

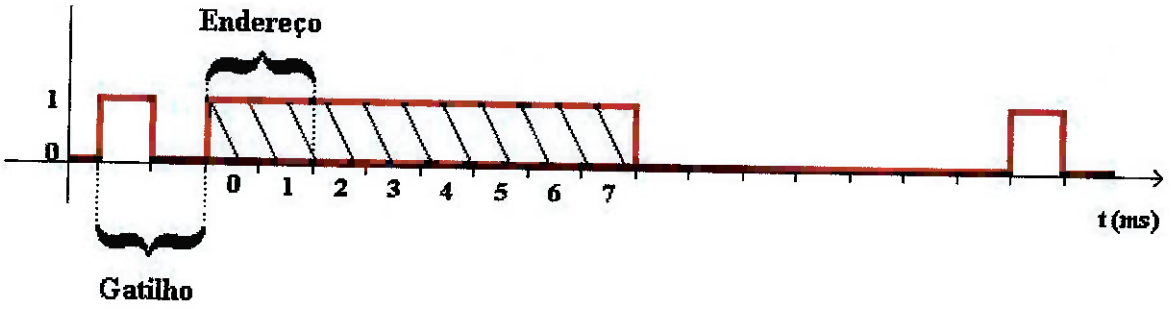



*Figura 5.3.5: protocolo de transmissão.*

Um protótipo desta solução foi implementado e seu desempenho foi comprovado. Sua velocidade de transmissão, de 500 bits/s é mais uma vez limitada pelo módulo RF, como no protótipo da solução anterior.

## 5.4. Solução 4

A solução 4 tem como principal diferencial uma integração plena da eletrônica embarcada no robô. É uma variação mais bem acabada da solução 3 e implicaria no uso de um micro-controlador de maior complexidade, que reuniria em si todas as funções desempenhadas no robô, isto é, a recepção de dados e o controle de trajetória. Como vantagem, este sistema apresentaria uma facilidade ainda maior de atualizações e modificações. Como desvantagem, o micro-controlador necessário teria custo mais elevado [21].

## 6. Escolha da melhor solução

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

- Custo de fabricação: este critério leva em conta o preço de compra dos componentes envolvidos em cada solução e dos equipamentos necessários à sua implementação. Como já foi dito, o custo de fabricação recebeu um dos mais altos pesos.
- Confiabilidade: este critério mensura a capacidade de cada solução de não produzir erros e também a facilidade de corrigi-los caso ocorram.
- Controlabilidade: este critério mede a facilidade que o software de estratégia encontrará para utilizar a ferramenta de transmissão de dados.
- Adaptação às dimensões: este critério avalia a melhor capacidade de cada solução em influir nas dimensões totais dos robôs (limitadas por regra).

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.

| Critério | Peso | Solução 1 | Solução 2 | Solução 3 | Solução 4 |
|---|---|---|---|---|---|
| Custo | 10 | 0 | 10 | 6 | 5 |
| Confiabilidade | 8 | 5 | 2.5 | 10 | 10 |
| Controlabilidade | 2 | 10 | 5 | 10 | 10 |
| Dimensões | 8 | 5 | 2.5 | 6 | 10 |
| Total | - | 100 | 147.5 | 208 | 230 |

| | | | | | |
|---|---|---|---|---|---|
| Média | - | 3.57 | 5.27 | 7.42 | 8.21 |

*Tabela 6.1: matriz de decisão para transmissão de dados via RF*

Como é possível observar, a solução 4, foi a que teve melhor desempenho na matriz, obtendo uma média de 8.21, sendo portanto a solução escolhida para a implementação definitiva pela equipe.

# 7. Implementação da transmissão de dados via RF

A implementação do par transmissor-receptor compreende as seguintes etapas:

- Definição das características de desempenho;
- Construção do hardware de transmissão;
- Implementação do software do micro-controlador do transmissor e do receptor;
- Testes de desempenho;
- Refinamentos;

## 7.1. Definição das características de desempenho

O transmissor digital deve, conforme a apresentação da solução 3 e 4, contornar as possibilidades de erro. Como principal fonte de erro, observa-se a troca de estado da porta paralela durante o envio de um byte. Para tornar o sistema imune a esse erro a interface do micro-controlador com a porta deve ser ampliada em um bit correspondente ao bit de retorno dos dispositivos normalmente conectados à porta paralela e que tem a função de permitir ou não a mudança de estado da porta. Desta forma, o micro-controlador do transmissor deverá possuir pelo menos 8 entradas de dado e 2 saídas, uma para o módulo RF e outra para o bit de retorno. O micro-controlador do receptor, por sua vez, necessitará de 1 entrada para o módulo RF e 8 saídas, dada a necessidade de controlar os 2 motores de passo de 4 fases cada empregados em sua construção. Como o PIC 16C54C possui 12 pinos de i/o, ele é adequado para uso.

A velocidade de transmissão é limitada pelo módulo RF empregado, e é de 500 bits/s. Admitindo que o laço de controle do sistema futebol de robôs como um todo (captura e processamento de imagem, tomada de decisões e

movimentação dos jogadores) consuma por volta de 2 segundos e sabendo que um jogo oficial dura 5 minutos = 300 segundos, o sistema de transmissão precisa enviar 150 programações completas para os 3 jogadores.

Para explicar em que consiste uma programação completa dos 3 jogadores é necessário conhecer a trajetória desempenhada por cada jogador e os parâmetros envolvidos para que este cumpra a tarefa pretendida. Essa trajetória é convenção da equipe e consiste em uma rotação sobre o centro de massa do jogador e uma translação, após a qual o robô aguardará nova programação. Para efetuar esta programação, o software de estratégia deverá informar ao robô o sentido e a magnitude da rotação e da translação, além de uma palavra de disparo, totalizando 3 palavras de 8 bits cada uma: rotação, translação e disparo. Como são 3 os jogadores e 150 programações durante o jogo, o sistema de transmissão de dados via RF deverá ser capaz de transmitir 150 x 3 x 3 x 8 = 10800 bits ou 1350 bytes. Estes números serão utilizados para os testes da confiabilidade da transmissão, mais adiante.

O receptor digital, por sua vez, possui como principal fonte de erro a falta de sincronismo entre a freqüência de sinal transmitida e a freqüência de sinal a qual ele é sensível. Para corrigir este problema, o receptor deve ter a capacidade de ajustar sua freqüência de amostragem do sinal para não cometer erros na decodificação dos bits enviados. Outra característica de desempenho do receptor é a sua capacidade de endereçamento, isto é, deve ser possível ao software de estratégia conversar apenas com um jogador em especial, se for o caso.

## 7.2. Construção do hardware de transmissão

O hardware de transmissão envolve a compra e montagem dos dispositivos requeridos para a implementação do sistema. São eles:

- Placa universal 30 x 60 mm com ilhas isoladas;
- 4 capacitores cerâmicos de 30 pf;
- Conector macho para porta paralela (25 pinos);

- Conector para fonte de computador (terra, +5V, +12V);
- 2 micro-controladores PIC 16C54C;
- 2 cristais de oscilação de 3,540 Mhz;
- Soquetes para circuitos integrados (18 pinos);
- Par de módulos RF Weishing HS-435 (435 MHz);

Apesar do item 5.4 explicar a solução 4 como dependente de um micro-controlador mais poderoso, foi possível sua implementação com os mesmos PIC 16C54C bastando para isso uma maior integração com o desenvolvimento da eletrônica embarcada dos robôs, fato flagrante quando se analisa o programa de recepção como uma rotina dentre muitas outras contidas no micro-controlador do robô. O transmissor deve ser montado conforme as figuras 18 e 19, enquanto os receptores devem ser montados nos robôs.

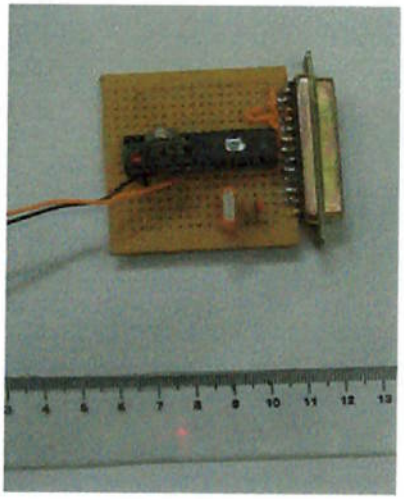


*Figura 7.2.1: protótipo do transmissor.*

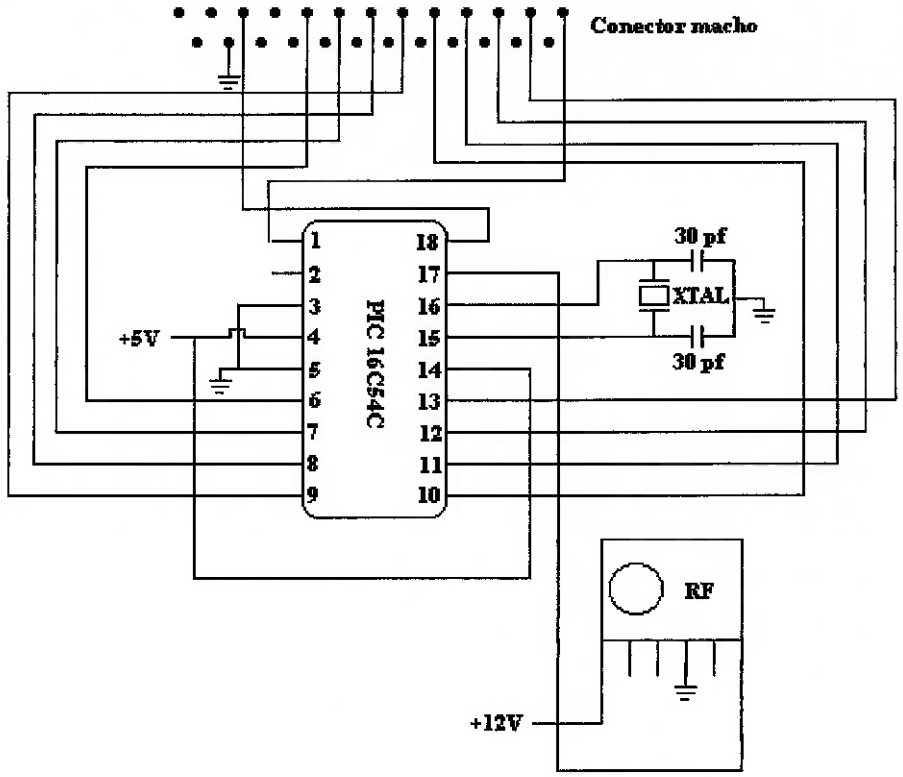
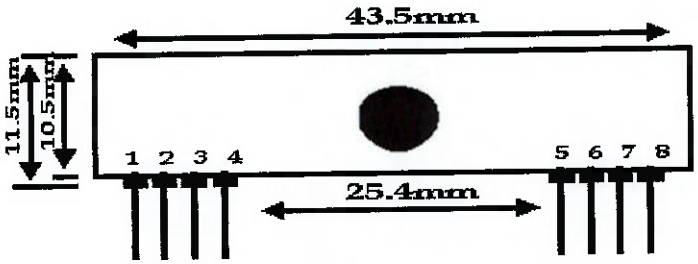



*Figura 7.2.2: esquema da placa do transmissor.*

Além disso, os diagramas dos módulos RF são:

*Figura 7.2.3: receptor Weishing HS-435.*




**pin 1 : Gnd**
**pin 2 : DigitaL Output**
**pin 3 : Linear Output**
**pin 4 : Vcc**
**pin 5 : Vcc**
**pin 6 : Gnd**
**pin 7 : Gnd**
**pin 8 :Ant ( About 30 - 35 cm )**

*Figura 7.2.4: pinagem do módulo receptor RF.*

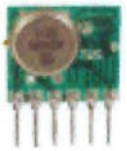
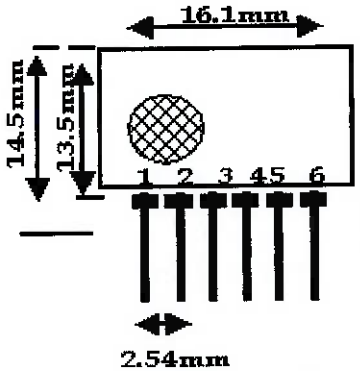


*Figura 7.2.5: transmissor Weishing HS-435.*




**pin 1 : Vcc**
**pin 2 : Vcc**
**pin 3 : Gnd**
**pin 4 : Gnd**
**pin 5 : RF Output**
**pin 6 : Code Input**

*Figura 7.2.6: pinagem do módulo transmissor de RF.*

Como o módulo receptor mostrou-se pouco confiável, foi necessário adicionar alguns componentes na tentativa de solucionar a instabilidade. A primeira medida foi utilizar a saída analógica do módulo, cujo funcionamento era mais consistente e modificar o sinal desta para posterior interface com o micro-controlador. Em seguida foi inserido um capacitor de poliéster de 25 nF na alimentação do módulo para diminuir as oscilações.

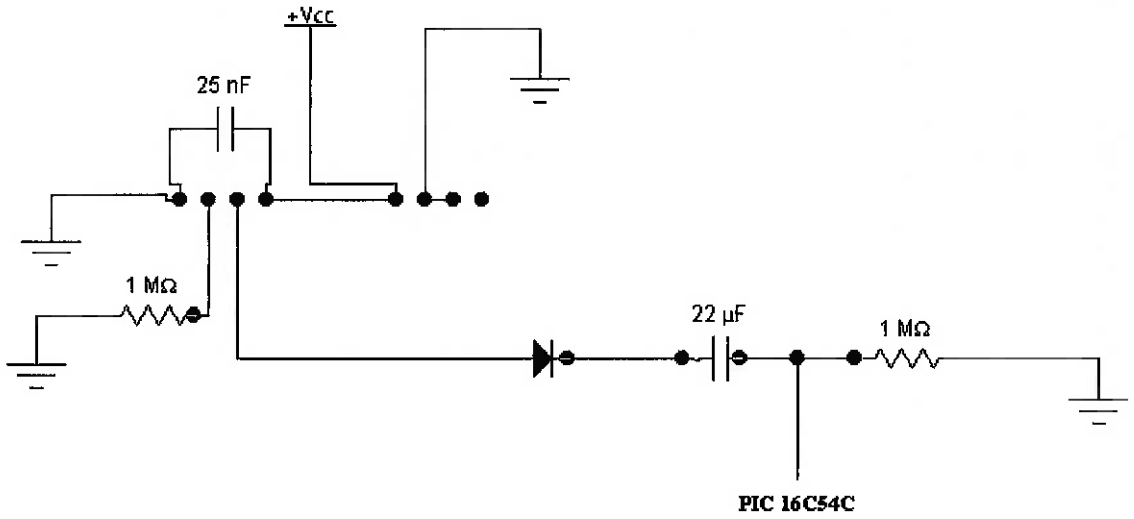



*Figura 7.2.7: circuito de ligação do receptor.*

A saída analógica passa por um diodo IN 4002 e em seguida por um capacitor eletrolítico de 22 µF e um resistor de 1MΩ em paralelo com o pino de entrada do micro-controlador. Esta ligação foi feita para manter o nível alto durante transmissão de 1 lógico através do capacitor e para garantir 0 V na transmissão de 0 lógico através do resistor. O diodo impede a descarga do capacitor pelo módulo receptor, o que poderia danificá-lo.

## 7.3. Implementação do software do micro-controlador do transmissor e do receptor

Para o micro-controlador do transmissor foi implementada rotina em assembler descrita no APÊNDICE 1: Programa transmissor.

O programa funciona da seguinte maneira: o micro-controlador inibe a troca do estado da porta paralela elevando o bit 1 da porta de i/o A para o nível lógico 1. O dado presente na porta é então lido e comparado ao último transmitido. Se o dado atual é igual ao anteriormente enviado, o micro-controlador volta para a posição inicial e habilita a troca de dado na porta paralela. Caso o dado seja novo, a transmissão do dado é feita. Ela acontece primeiro pela transmissão de uma gatilho, composto de um sinal em nível lógico 1 com duração de um ciclo de loop, seguido por um sinal de nível lógico 0 com duração de meio loop. Em seguida o dado na porta paralela é lido e transmitido bit a bit e após o oitavo bit o transmissor permanece em nível 0 por dois ciclos de loop. Como evento final o micro-controlador habilita novamente a porta paralela a trocar de dado.

Para o micro-controlador do receptor foi implementado o programa em assembler descrito no APÊNDICE 2: Programa receptor.

Basicamente, a rotina do receptor está contida num programa muito maior responsável por todas as funções propostas para o robô. A rotina de recepção é a mais ativa durante o processamento e funciona da seguinte maneira:

O gatilho enviado pelo receptor excita a rotina e inicia a contagem da constante de tempo da recepção. Terminada a contagem, a cada loop de tempo a rotina amostra o pino de entrada e monta o dado recebido. Como última tarefa antes de entregar o dado às rotinas decodificadoras, a recepção compara os bits mais significativos do dado recebido, isto é, os bits 6 e 7, aos bits de endereço gravados na rotina. Para o programa listado acima, o endereço é 00.

Uma abordagem mais completa do programa como um todo não será feita neste trabalho, mas sim no trabalho referente à eletrônica embarcada.

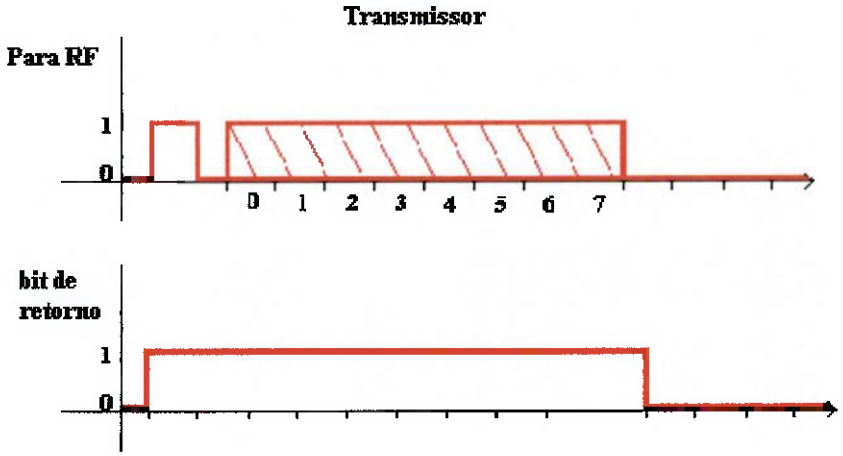



*Figura 7.3.1: carta de tempo do transmissor.*

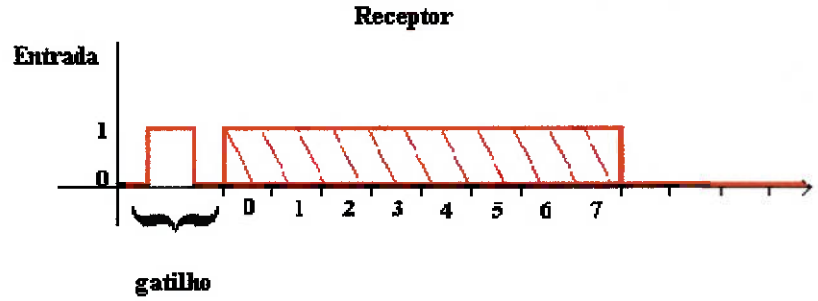



*Figura 7.3.2: carta de tempo da entrada do receptor.*

A tabela para programação do robô, necessária para o software de estratégia é:

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

<table>
<tr><td>D7<br>D6</td><td>00 – robô 00<br>01 – robô 01<br>10 – robô 10</td></tr>
<tr><td>D5<br>D4</td><td>00 – rotação<br>01 – translação<br>10 – Disparo do movimento</td></tr>
<tr><td>D3</td><td>Se rotação: 0 para sentido anti-horário, 1 para horário<br>Se translação: 0 para trás, um para frente;</td></tr>
<tr><td>D2<br>D1<br>D0</td><td>Magnitude do movimento:
<table>
<tr><td></td><td>Rotação</td><td>Translação</td></tr>
<tr><td>000</td><td>0°</td><td>0 cm</td></tr>
<tr><td>001</td><td>25,7°</td><td>5 cm</td></tr>
<tr><td>010</td><td>51,4°</td><td>10 cm</td></tr>
<tr><td>011</td><td>71,1°</td><td>15 cm</td></tr>
<tr><td>100</td><td>102,85°</td><td>20 cm</td></tr>
<tr><td>101</td><td>128,57°</td><td>25 cm</td></tr>
<tr><td>110</td><td>154,28°</td><td>30 cm</td></tr>
<tr><td>111</td><td>180°</td><td>35 cm</td></tr>
</table></td></tr>
<tr><td></td><td></td></tr>
</table>

*Tabela 7.3.1: comandos dos robôs.*

Conforme a convenção estabelecida na equipe, tem-se então o seguinte significado para a programação-exemplo 00010110 (22 em decimal), 00000001 (um em decimal) e 00101101 (45 em decimal):

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Valor | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

*Tabela 7.3.2: exemplo de programação.*

Como bit 7 e 6 são 00 o dado é referente ao robô 00. Com os bits 5, 4 assumindo o valor 01 significa translação, e do bit 3 igual a 0 depreende-se que é para trás. O valor dos bits 2, 1 e 0 é 6 em decimal, isto é, 30 cm.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Valor | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

*Tabela 7.3.3: exemplo de programação.*

Como bit 7 e 6 são 00 o dado é referente ao robô 00. Com os bits 5, 4 assumindo o valor 00 significa rotação, e do bit 3 igual a 0 depreende-se que é no sentido anti-horário. O valor dos bits 2, 1 e 0 é 1 em decimal, isto é, 25,7° .

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Valor | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |

*Tabela 7.3.4: exemplo de programação.*

Como bit 7 e 6 são 00 o dado é referente ao robô 00. Com os bits 5, 4 assumindo o valor 10 significa disparo e o restante dos bits não têm importância. O robô 00, caso programado desta forma executará uma rotação no sentido anti-horário de 25,7° e uma translação para trás de 30 cm.

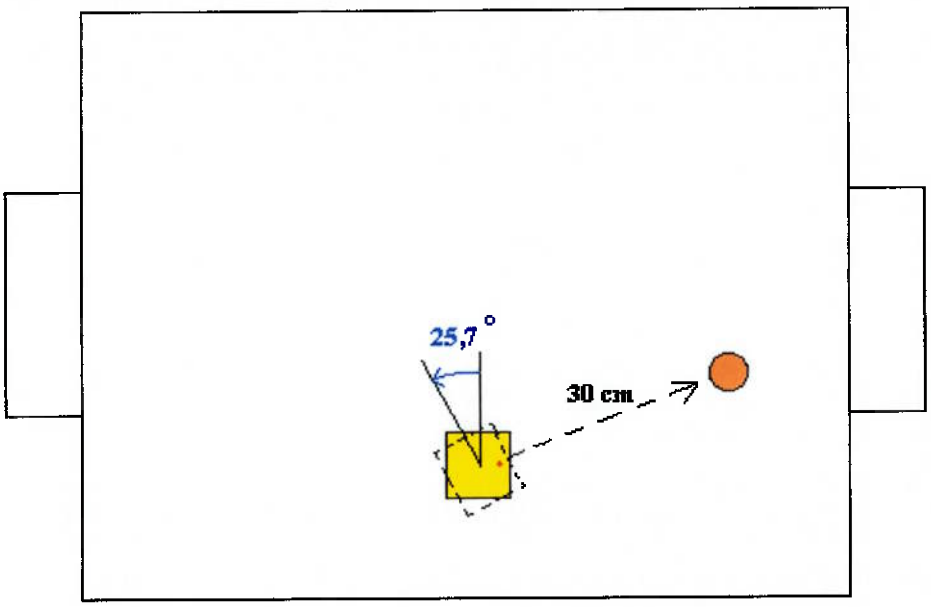



*Figura 7.3.3: comportamento do robô 00 após a programação-exemplo.*

## 7.4. Testes de desempenho

O término da concepção física do transmissor e do receptor possibilitou a verificação da taxa de ocorrência de erros na transmissão. Esta verificação foi feita da seguinte maneira: primeiramente implementou-se em linguagem C uma rotina para geração aleatória de palavras, enviadas de 2 em 2 segundos para a porta até que se atingisse a marca de 10800 bits transmitidos (ou 1350 bytes enviados). Em seguida foi montado um circuito receptor com um micro-controlador PIC 16C54C, Modificando-se ligeiramente o programa receptor para comparar o estado da porta paralela com o dado recebido e acusar o erro num pino de saída, consegui-se contar o número de erros injetando esse sinal em um contador binário de 8 bits 74LS93. Os testes foram feitos tanto com transmissão via RF quanto por fio, para analisar a confiabilidade do módulo RF Weishing HS-435.

Os programas implementados para os testes de desempenho estão listados no APÊNDICE 3: Programas de teste de desempenho.

Para a transmissão por fios:

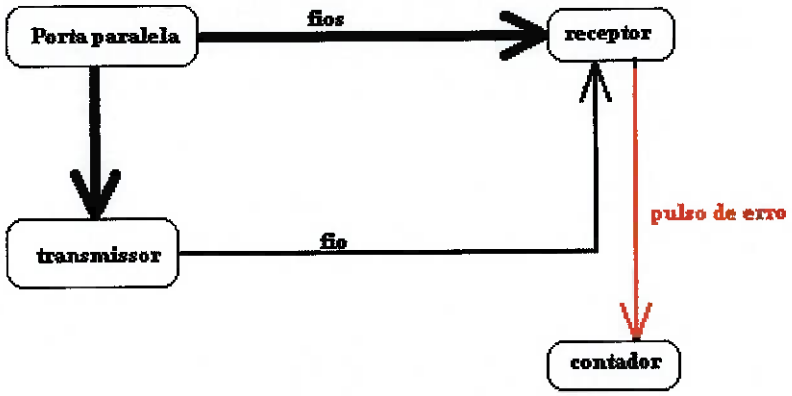



*Figura 7.4.1: circuito para medição de erros por fio.*

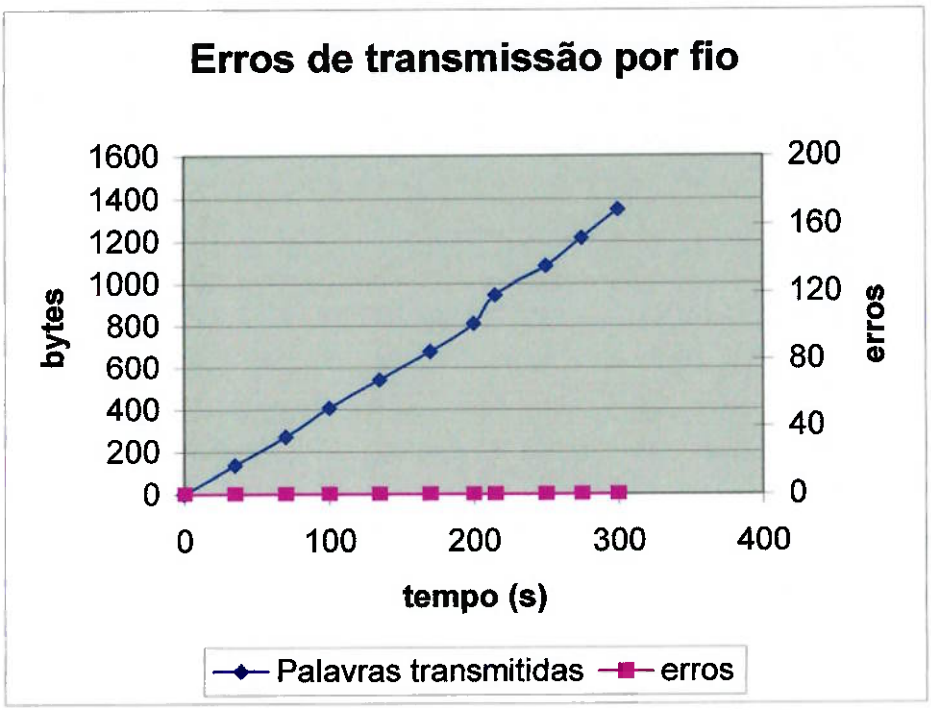



*Figura 7.4.2: gráfico de erros na transmissão por fio.*

Para a transmissão por RF:

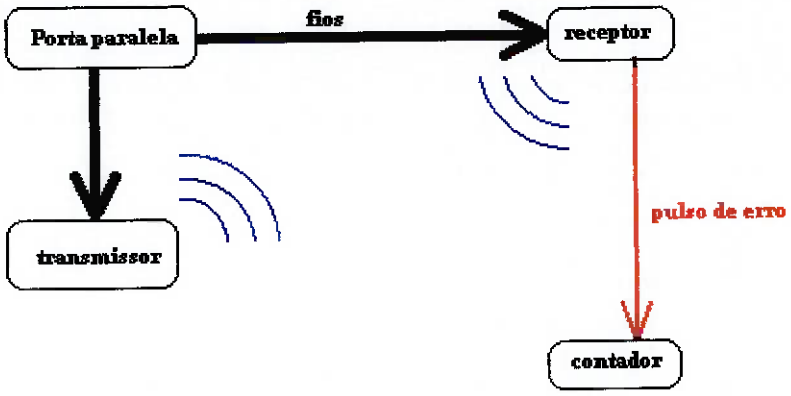



*Figura 7.4.3: circuito para medição de erros por RF.*

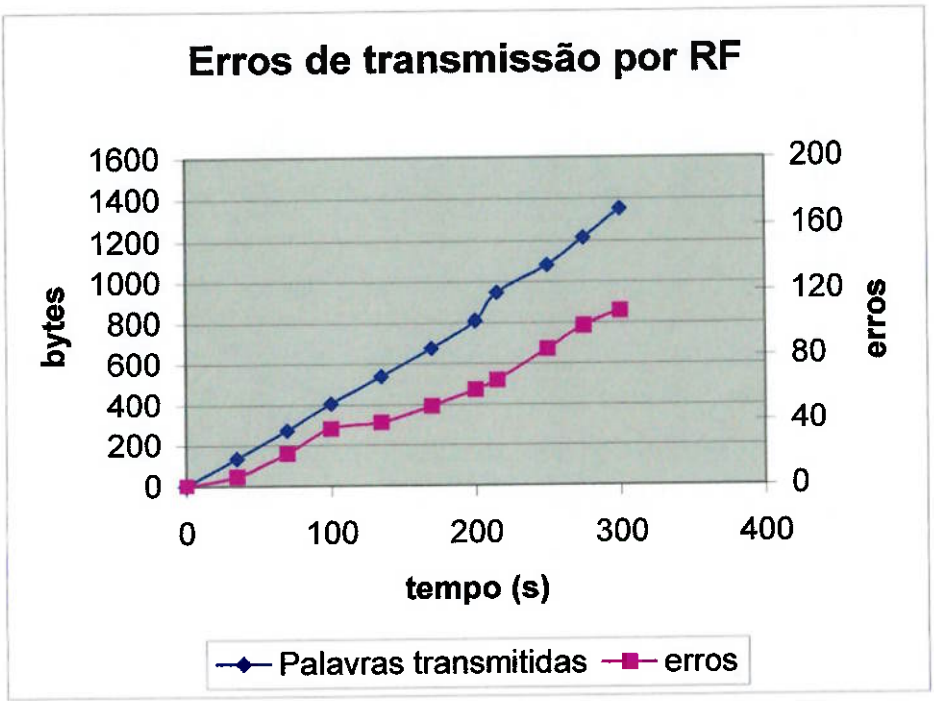



*Figura 7.4.4: gráfico de erros na transmissão por RF.*

A taxa de erro na transmissão por fio foi zero, indicando a alta confiabilidade dos micro-controladores. Por outro lado, na transmissão por RF, a taxa de erros foi de aproximadamente 8% do total de palavras transmitidas, um índice alto.

## 7.5. Refinamentos

Os resultados apontados no item anterior indicam claramente o caminho para melhorar ainda mais o sistema de transmissão recepção de dados via RF. É preciso mudar o módulo RF para um com maiores velocidades de transmissão e confiabilidade. Com a atual taxa de transmissão em 500 bits/s e uma carga de dados de 10800 bits/jogo, durante os 5 minutos da partida 22 segundos serão gastos na transmissão de dados. Existem módulos comerciais que embora mais caros transmitem a até 21000 bits/s, reduzindo o tempo gasto para apenas 0,6 segundo durante um jogo inteiro.

# 8. Estudo de viabilidade da visão computacional

Será discutida no restante do trabalho a viabilidade da visão computacional, bem como as perspectivas para o futuro deste sistema.

## 8.1. Viabilidade técnica da visão computacional

A visão computacional é uma área de estudo que teve seu desenvolvimento muito acelerado pelas suas características para medições de processos industriais. Tecnicamente viável, transformou-se em equipamento de sensoriamento e é com essa finalidade que tem sido utilizada em futebol de robôs [13].

Constituído por câmera, placa de vídeo e um software de processamento, o sistema de visão computacional é responsável pela monitoração dos elementos do jogo: jogadores e bola.

Devido ao frenesi em busca de vitórias nos campeonatos de futebol de robôs, os sistemas de visão computacional têm sido alvo dos principais fabricantes de hardware de processamento de imagem, que têm utilizado as competições como laboratório para seus equipamentos [14].

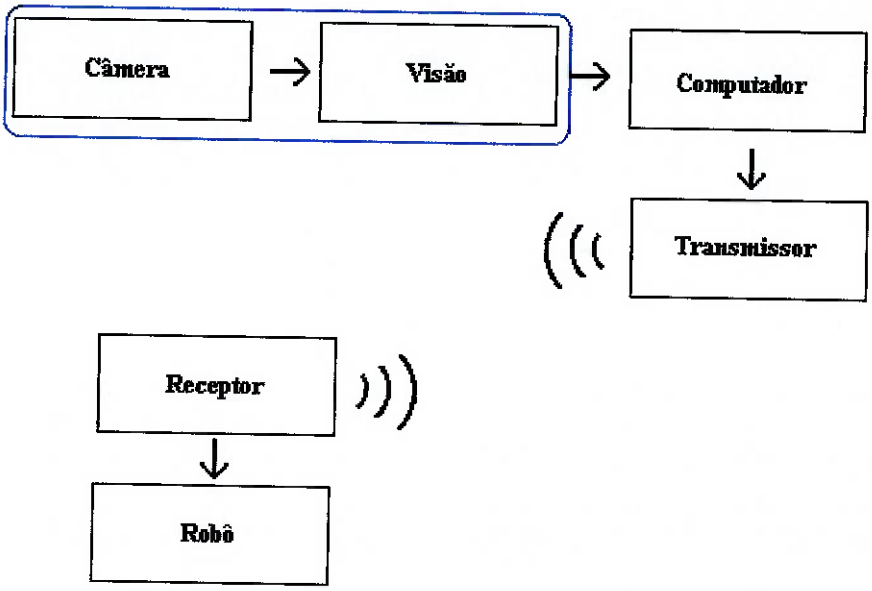



*Figura 8.1.1: visão computacional (azul).*

## 8.2. Viabilidade financeira da visão computacional

Dada a relevância do sistema de visão computacional para o sistema futebol de robôs e dos fatores anteriormente descritos no item 8, há uma imensa variedade de equipamentos disponíveis para aplicação. Entretanto, pela limitação de recursos da equipe há a necessidade de se utilizar os equipamentos disponíveis. Apesar disso, a pouca disponibilidade de equipamentos não limitará a formulação das soluções propostas para o problema.

## 8.3. Apresentação das possíveis soluções

Para efetuar a apresentação das soluções, faz-se necessária uma explicação acerca dos tipos de hardware mais utilizados:

Câmeras: podem ser analógicas (VHS) ou digitais (CCD). Equipamentos digitais compõe a imensa maioria em competições de futebol de robôs assim como nos sistemas industriais de medição baseados em imagens. A câmera utilizada pela equipe, porém, é analógica para uso doméstico.

Placas de vídeo: dividem-se em dois grupos: as placas de aquisição, normalmente utilizadas para gravação de filmes em computadores, e as placas de processamento de vídeo, dedicadas ao tratamento de imagens em alta velocidade, cujo emprego é amplo nos sistemas industriais e também nas equipes de futebol de robôs. Como diferença fundamental, as placas de aquisição normalmente não dispõem de memória própria sendo obrigadas a utilizar a memória RAM de um computador para alocar as imagens digitalizadas e processá-las. No caso do futebol de robôs, este computador é o mesmo que abriga o software de estratégia e por isso as duas funções, tanto a visão computacional quanto a decisão de estratégias ficam prejudicadas. Na prática o que se observa é a impossibilidade de realizar o processamento em tempo real, vital para competições. Por esta razão, dispositivos deste tipo não encontram aplicação em futebol de robôs. As placas de processamento de vídeo, por outro lado, são dedicadas à função de processar as imagens despejadas em memórias RAMDAC de altíssima velocidade. Alguns modelos apresentam processamento paralelo e a implementação dos algoritmos de processamento é feita diretamente na memória da placa, sendo possível programar o tipo de interface com software de estratégia.

### 8.3.1. Solução 1- Equipamento para processamento em tempo real

Esta solução relaciona-se ao uso de uma câmera digital colorida e uma placa de processamento de imagem [25].

### 8.3.2. Solução 2- Equipamento de aquisição de vídeo

Composta basicamente dos equipamentos disponíveis para a equipe, isto é, uma câmera Panasonic VHS e uma placa de aquisição de imagens Vídeo Blaster FS200, além de um computador IBM PC 486 DX2 66 MHz.

# 9. Detalhamento das soluções

Neste tópico serão explicados as soluções propostas para o problema de visão computacional, seus desempenhos e alguns estudos realizados acerca da solução dois.

## 9.1. Solução 1

A solução um é composta do equipamento padrão entre as equipes de futebol de robôs. Uma câmera digital (CCD) colorida e uma placa de processamento de imagens capaz de processar em torno de 50 quadros por segundo a uma resolução de 640x480 pixels. A observação dos campeonatos mostra a capacidade destes sistemas de suprir os softwares de estratégias com informações em tempo real de modo a permitir uma cadência de jogo imperceptível para olhos humanos. Esta alta velocidade faz com que as diferenças de posição analisadas entre dois quadros subseqüentes sejam muito pequenas, a ponto de não ser necessária a implementação de algoritmos para corrigir a distorção da lente da câmera.

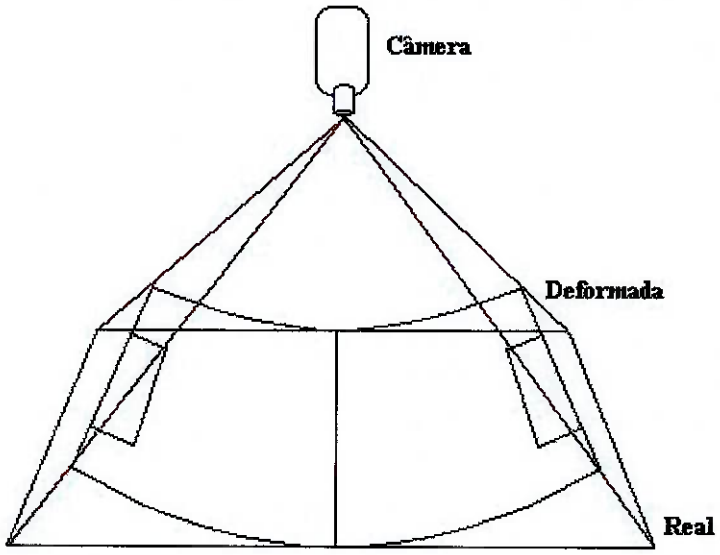



*Figura 9.1.1: deformação da imagem pela lente.*

## 9.2. Solução 2

A solução dois é composta pelo equipamento descrito anteriormente. Com ele foi possível realizar alguns testes de desempenho e os resultados serão tratados aqui.

O primeiro teste realizado foi o da velocidade de aquisição. Apesar de não considerar o processamento de imagens, já basta para estabelecer uma referência de desempenho em relação aos níveis dos equipamentos de outras equipes (com placas de processamento).

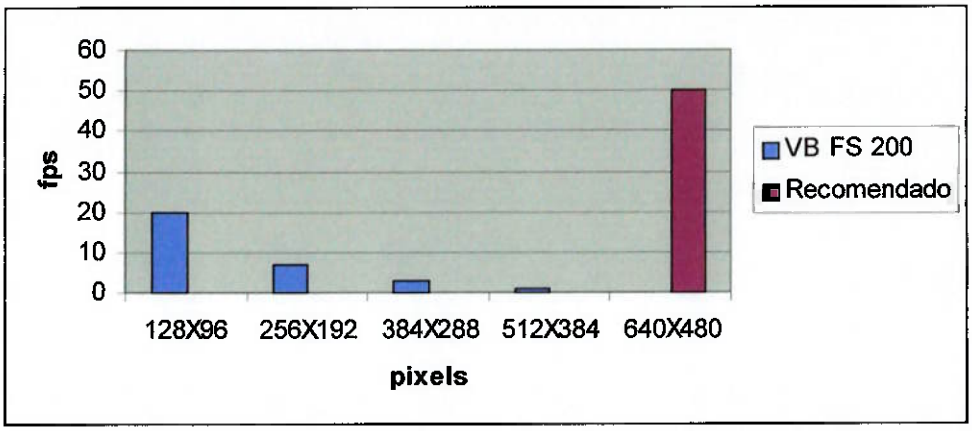



*Figura 9.2.1: velocidade em quadros por segundo X resolução para aquisição.*

Esta tabela mostra a obsolescência do equipamento e sua incapacidade de equiparar-se às placas convencionais empregadas no futebol de robôs.

O segundo teste realizado consiste em decompor as imagens geradas no padrão true color (24 bits) e analisar as imagens quanto à qualidade. A primeira imagem é referente ao campo vazio:

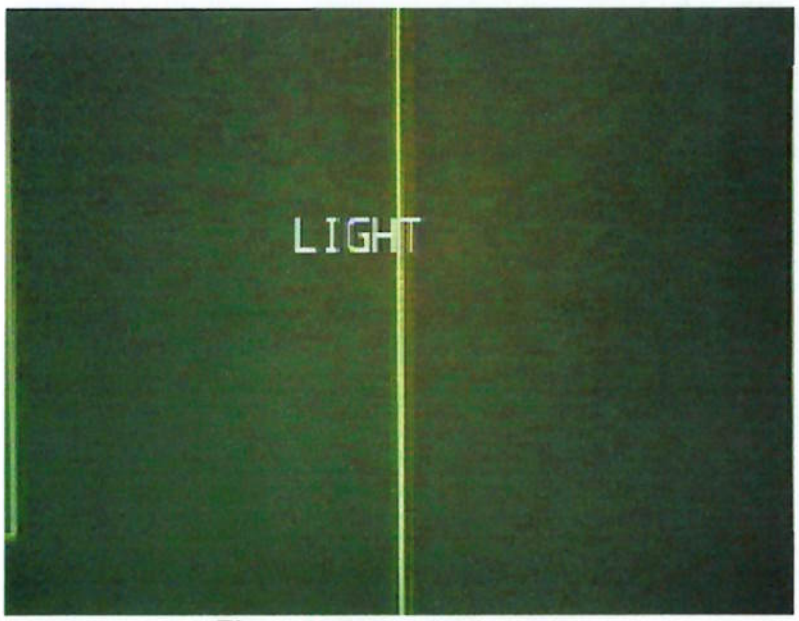



*Figura 9.2.2: campo vazio.*

A mensagem "LIGHT" é proveniente da câmera e significa que há falta de luz no ambiente. As matrizes resultantes da decomposição true color são:

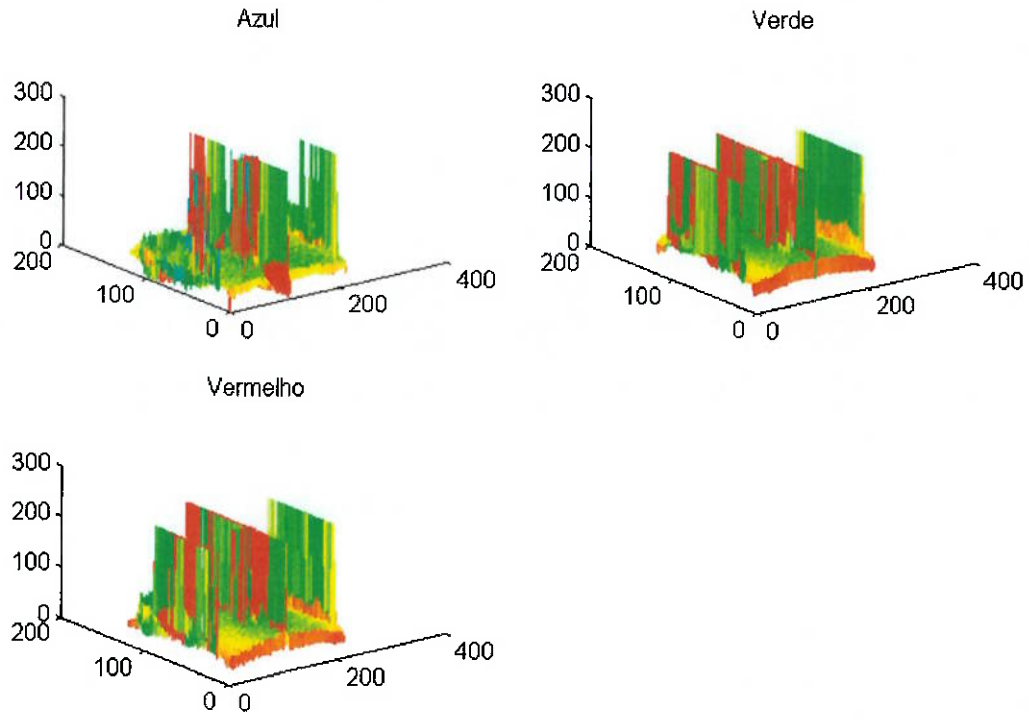



*Figura 9.2.3: RGB da figura 21.*

A observação das matrizes, principalmente a verde, mostra uma deficiência na homogeneidade das cores, apesar da iluminação uniforme. Também há um fato curioso na matriz azul, onde as linhas das áreas do campo não correspondem ao esperado para cor branca, isto é, deveriam estar no máximo da escala, mas não estão.

A próxima imagem corresponde ao campo vazio com a luminosidade mínima necessária para suprir as necessidades da câmera:

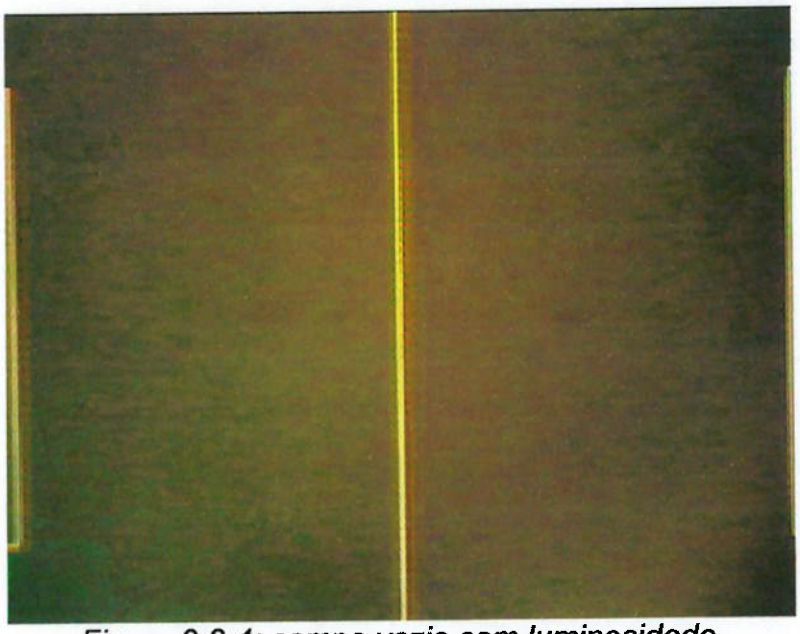


*Figura 9.2.4: campo vazio com luminosidade.*

As matrizes correspondentes são:

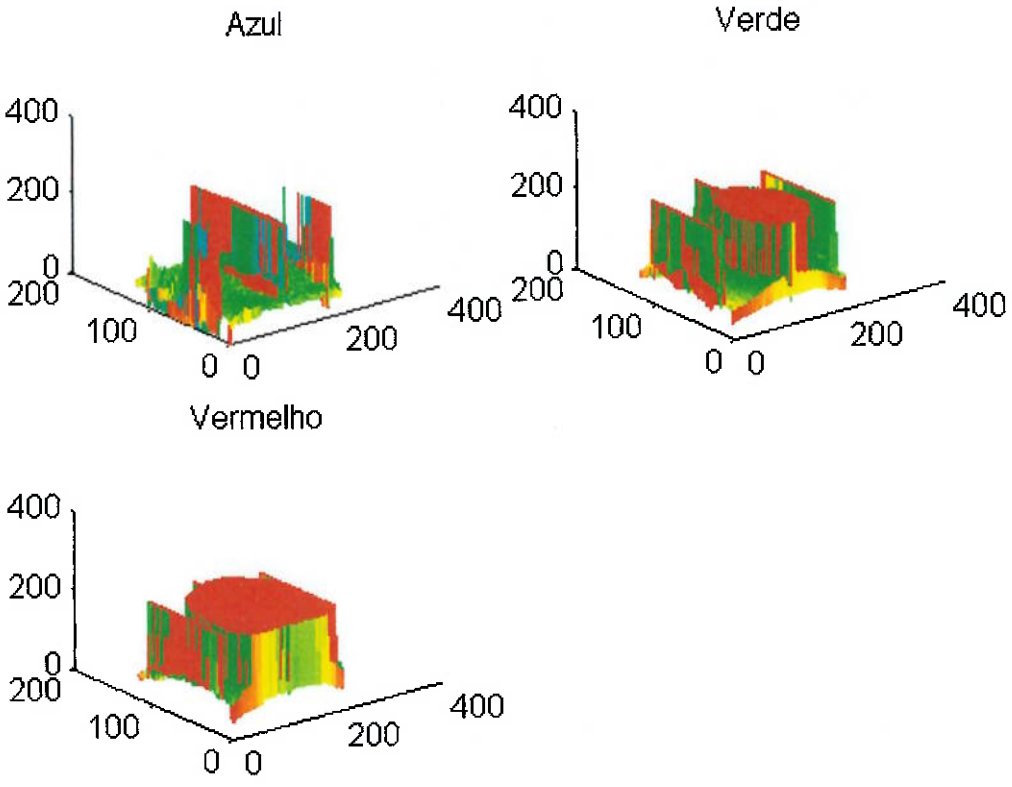



*Figura 9.2.5: RGB da figura 23.*

A observação das matrizes da figura 24 mostra que não é possível realizar qualquer medição, pois quase toda a imagem está em fundo de escala.

Isso mostra num primeiro momento falta de confiabilidade do equipamento, porém os estudos devem prosseguir até que as possibilidades sejam esgotadas.

# 10. Escolha da melhor solução

Para a decisão acerca da melhor solução, foi escolhida a técnica da matriz de decisão, com os critérios de seleção definidos abaixo:

- Custo de implementação: por não serem equipamentos que precisam de desenvolvimento, precisam necessariamente ser comprados. Como já foi dito, o custo um dos fatores limitantes deste projeto.
- Confiabilidade: este critério mensura a capacidade de cada solução de não produzir erros e também a facilidade de corrigi-los caso ocorram.
- Velocidade: este critério mede a rapidez que o sistema oferece no processamento de imagens.

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.

| Critério | Peso | Solução 1 | Solução 2 |
|---|---|---|---|
| Custo | 10 | 1 | 10 |
| Confiabilidade | 3 | 10 | 0 |
| Velocidade | 1 | 10 | 0 |
| Total | - | 50 | 100 |
| Média | - | 3.57 | 7.14 |

*Tabela 10.1: matriz de decisão para transmissão de dados via RF.*

Como é possível observar, a solução 2, foi a que teve melhor desempenho na matriz, obtendo uma média de 7.14, sendo assim a solução escolhida para a implementação definitiva pela equipe.

# 11. Implementação da visão computacional

A implementação da visão computacional é um projeto que compreende as seguintes etapas [26] [27]:

- Aperfeiçoamento do campo e da CPU;
- Definição das características do software;
- Implementação de software;
- Testes e avaliação do desempenho;

## 11.1. Aperfeiçoamento do campo e da CPU

Conforme a evidência da pouca confiabilidade do equipamento, mesmo as marcações do campo são empecilhos para a execução de algum tipo de processamento de imagem. Por este motivo procedeu-se a remoção da pintura original e a re-pintura do campo, desta vez sem linhas brancas, na tentativa de minimizar o problema de precisão e repetibilidade do hardware envolvido.

Também foi modificada a CPU utilizada pela equipe para as tarefas de processamento de imagem e tomada de decisões. Definida anteriormente como um IBM 486 DX2 66 Mhz, foi substituída por um PENTIUM 166 Mhz com sistema operacional Windows 95.

Outra alteração feita foi a mudança da bola. Via de regra, deveria ser uma bola de golfe laranja, mas foi adotada uma bola comum de pingue-pongue branca. Para justificar a mudança há diversos motivos, entre os quais seu menor peso (melhor para o robô empurrar) e maior opacidade (favorece o processamento de imagem). Definiu-se também as marcas na parte superior dos jogadores como sendo:

*Figura 11.1.1: topo do robô*

Esta marca não permite a identificação direta da orientação do robô, motivo pelo qual o sistema é dinâmico e baseado na condição inical.

## 11.2. Definição das características do software

Para definir as características do software é necessário conhecer os requisitos de hardware inerentes ao equipamento disponível para a equipe. Tratando especificamente da placa de aquisição Vídeo Blaster FS200, constata-se a impossibilidade de empregá-la sob qualquer sistema operacional diferente de DOS versão 3.2 ou superior, apesar de a placa apresentar drivers para Windows 3.1. Esse comportamento deve-se provavelmente a algum defeito associado ao tempo de uso da placa já que é pouco realista a idéia de o driver para Windows 3.1 não ser operacional, pois trata-se de software comercial. Além disso, o compilador C disponível é antigo e pobre, dificultando a realização de cálculos numéricos. Por outro lado, a obtenção de uma nova CPU com Windows 95 abriu a possibilidade de realizar processamento em multi-tarefa, além do emprego de softwares matemáticos e de manipulação simbólica como o MatLab. Desta forma, abandonou-se totalmente a intenção de otimizar a velocidade de processamento em favor da manipular de dados para a obtenção de resultados, mesmo que isso significasse um aumento demasiadamente longo do tempo de processamento.

Como característica do software fica evidente então o uso de duas linguagens de programação: o C e a linguagem característica de MatLab. Além disso, é importante frisar a função principal do sistema de processamento de imagem: alimentar o programa de estratégia com dados sobre a posição e a orientação dos entes envolvidos do jogo de futebol de robôs, o que significa, no caso específico da equipe, a bola e os três robôs.

## 11.3. Implementação do software

A análise das características de software e hardware envolvidas sugeriu a comunicação entre os programas em C e em MatLab através de arquivos. Genericamente o sistema de futebol de robôs é um grande loop de software que envolve as seguintes etapas:

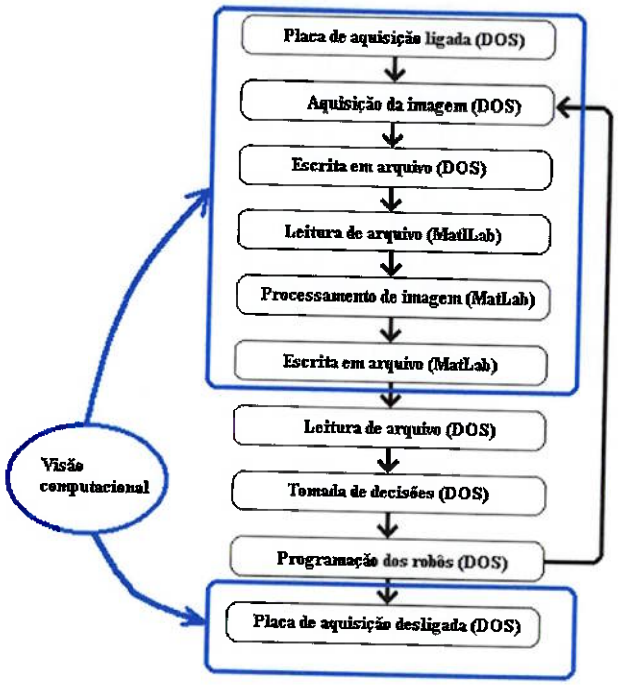



*Figura 11.3.1: loop de software genérico.*

A etapa do acionamento da placa é feita pelo software em C [29] descrito no APÊNDICE 4: Programas em C para captura de imagem.

Sobre o programa para ligar a placa existem algumas considerações sobre o set-up da imagem que será gerada. Entre as principais estão o padrão NTSC compatível coma filmadora PANASONIC e o tamanho da tela de 30% em relação ao total disponível. De modo geral foi possível aumentar o tamanho para até 60% do total sem travar o computador por erro de memória, mas com os 30% escolhidos já foi possível obter imagens com um bom número de pontos (120 X 216 pixels).

Para a aquisição da imagem e escrita em arquivo foi implementado o programa em C descrito também no APÊNDICE 4: Programas em C para captura de imagem.

Sobre o programa de aquisição da imagem e escrita em arquivo vale explicar que o arquivo gerado é numérico de nome foto3.r. Este arquivo contém uma matriz 120 X 216 com os valores da decomposição em vermelho da imagem. Para isso é necessário decompor a imagem no padrão RGB 24 bits, conhecido como *true color*. Poder-se-ia utilizar também para o processamento de imagens as componentes verde e azul, porém estas demonstraram uma completa deficiência em precisão e repetibilidade durante os testes realizados e por isso foram descartadas.

As próximas etapas correspondem à implementação dos algoritmos em MatLab e para explicar melhor seu funcionamento será utilizada a seguinte imagem, composta do campo com a bola:

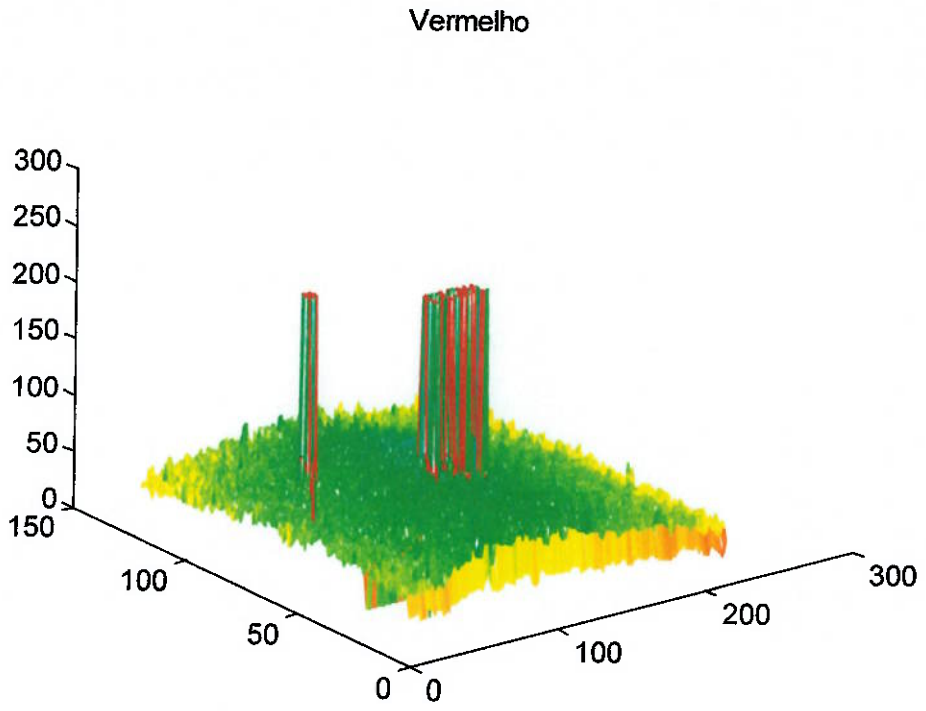



*Figura 11.3.1: decomposição em vermelho da imagem exemplo.*

A leitura do arquivo e seu tratamento inicial são feitos pela rotina em MatLab descrita no APÊNDICE 5: Rotinas em MatLab para processamento de imagem.

O funcionamento da rotina é: primeiro executa-se o comando em DOS referente ao arquivo que faz a aquisição da imagem. Em seguida ele é filtrado para eliminar ruídos de baixa amplitude. A seguir ocorre o rebatimento da imagem para coincidir a origem do sistema de coordenadas com o canto inferior esquerdo da imagem:

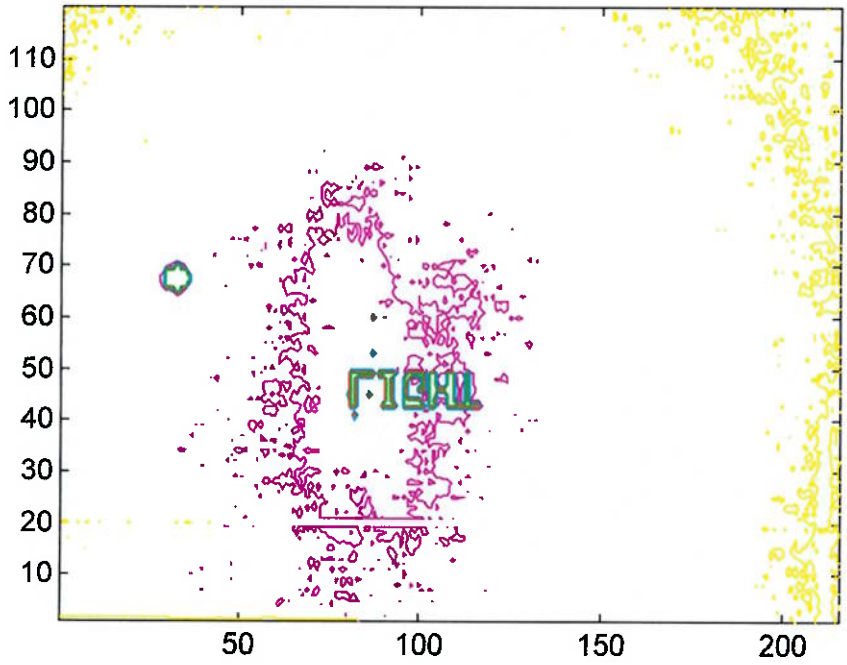



*Figura 11.3.2: antes do rebatimento.*

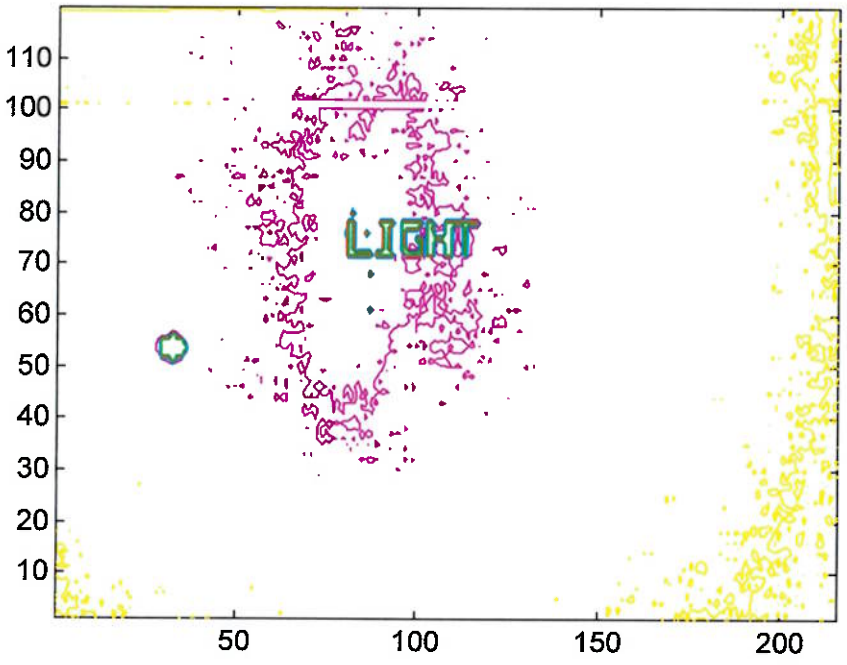



*Figura 11.3.3: depois do rebatimento.*

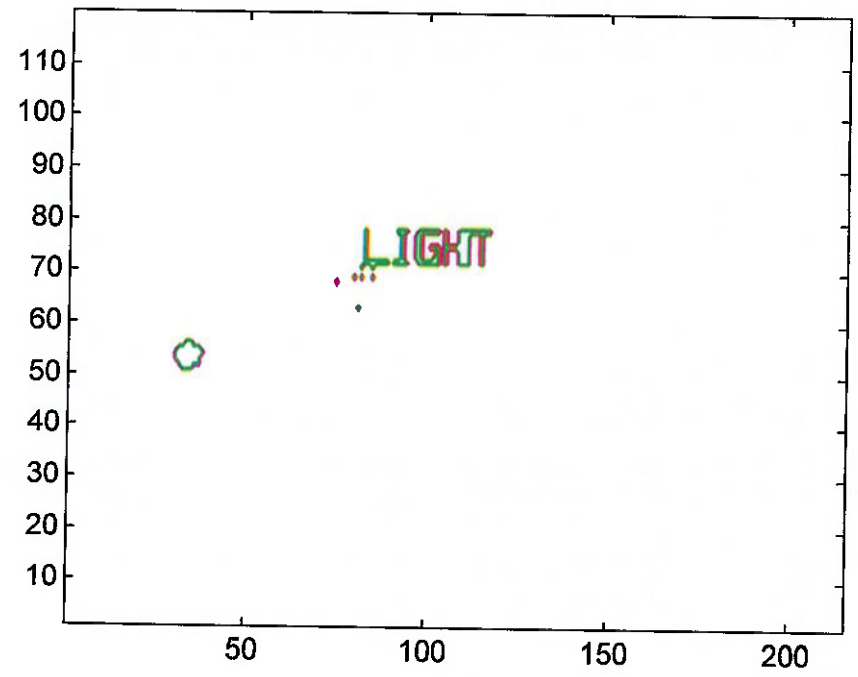



*Figura 11.3.4: imagem sem ruído de baixa amplitude.*

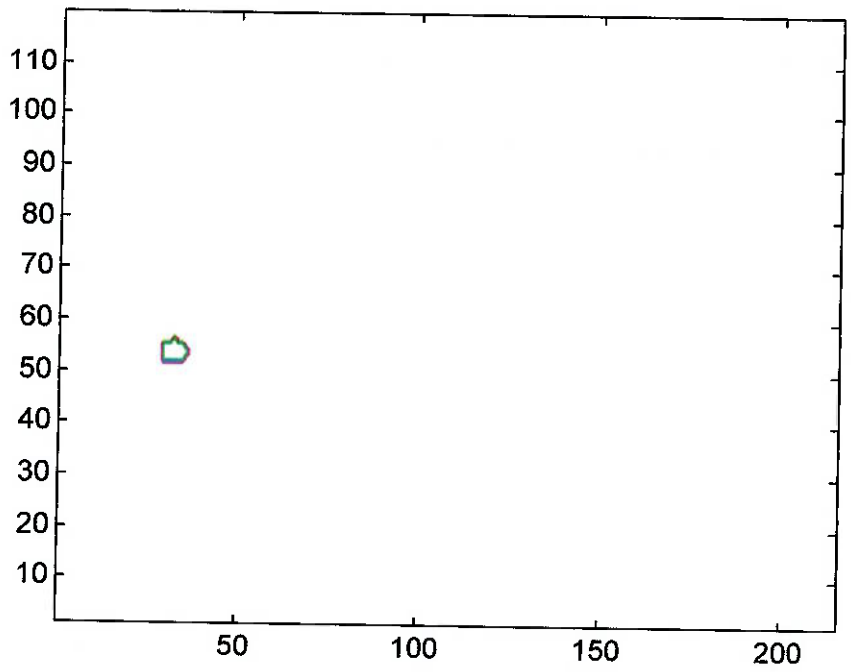



*Figura 11.3.5: LIGHT removido.*

Como observado no item 9.2, a mensagem LIGHT é proveniente da câmera e pode ser removida aumentando-se a luminosidade ambiente. Porém isso causa uma saturação da imagem e inviabiliza qualquer medição posterior. A solução empregada é simplesmente apagar toda a área onde aparece a mensagem LIGHT. Isso facilita o processamento, mas torna uma área de cerca de 3% da matriz nula para efeitos de cálculo. O próximo passo é o cálculo de contornos e seu resultado final está mostrado na figura 11.3.5.

A próxima rotina é continuação da anterior e é responsável pela detecção de objetos na matriz de contorno. A rotina consiste numa manipulação de matrizes de forma a transformar a matriz de contorno numa matriz mais adequada para os cálculos que virão a seguir. A matriz de contorno tem o seguinte formato:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $V_1$ | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $V_2$ | $X_1$ | $X_2$ | $V_3$ | ... | $X_n$ |
| $N_1$ | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $N_2$ | $Y_1$ | $Y_2$ | $N_3$ | ... | $Y_n$ |

*Tabela 11.3.1: matriz de contorno.*

Onde os $V_n$ são os valores dos contornos (patamares) e os $N_n$ são os números de pontos que formam cada contorno. Os $X_n$ e $Y_n$ correspondem às coordenadas destes pontos:

| | | | | | |
|---|---|---|---|---|---|
| 100 | 2 | 3 | 200 | 1 | 2 |
| 2 | 4 | 2 | 2 | 8 | 8 |

*Tabela 11.3.2: exemplo de matriz de contorno.*

Desta matriz é possível depreender a existência de 2 contornos:

- Valor 100 e pontos (2,4) e (3,2);
- Valor 200 e pontos (1,8) e (2,8);

O objetivo da próxima rotina implementada é transformar a matriz de contorno na matriz de objetos:

| valor | N | $X_{médio}$ | $Y_{médio}$ | $X_{máx}$ | $Y_{ass}$ | $X_{mín}$ | $Y_{ass}$ | $X_{ass}$ | $Y_{máx}$ | $X_{ass}$ | $Y_{mín}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 2 | 2.5 | 3 | 3 | 2 | 2 | 4 | 2 | 4 | 3 | 2 |
| 200 | 2 | 1.5 | 8 | 2 | 1 | 8 | 8 | 1 | 8 | 2 | 8 |

*Tabela 11.3.3: matriz objeto.*

Na matriz objeto cada linha representa um contorno com seu valor de contorno (patamar) e as coordenadas médias, além de os pontos de máximo e mínimo valor em X e Y, com seus respectivos pares associados. Esta matriz possibilita a identificação entre os objetos de: ruídos, bola e robôs, através do tamanho do contorno, além de sua orientação. Para a imagem da figura 11.3.5 tem-se a seguinte matriz de objeto:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 25 | 32,8 | 53,83 | 36,8 | 54 | 29,2 | 52 | 33 | 56,8 | 35 | 51,2 |
| 100 | 25 | 32,8 | 53,82 | 36,6 | 54 | 29,4 | 52 | 33 | 56,6 | 35 | 51,4 |
| 150 | 25 | 32,8 | 53,81 | 36,4 | 54 | 29,6 | 52 | 33 | 56,4 | 35 | 51,6 |
| 200 | 25 | 32,8 | 53,8 | 36,2 | 54 | 29,7 | 52 | 33 | 56,2 | 35 | 51,7 |
| 250 | 25 | 32,8 | 53,8 | 36 | 54 | 29,9 | 52 | 33 | 56 | 35 | 52 |

*Tabela 11.3.4: matriz objeto da imagem exemplo*

A rotina está listada no APÊNDICE 5: Rotinas em MatLab para processamento de imagem.

Com a matriz exemplo computada, é preciso corrigir as coordenadas apresentadas em função da distorção imposta pela câmera, pois da forma como se apresenta, um objeto esférico, por exemplo, não possui tamanho em x igual ao tamanho em y. Para a matriz apresentada, a esfera possui tamanho em $X=X_{max} - X_{min}= 8$ pixels e tamanho em $Y=Y_{max} - Y_{min}= 6$ pixels. Esta distorção é corrigida aplicando-se um fator de escala para a matriz objeto:

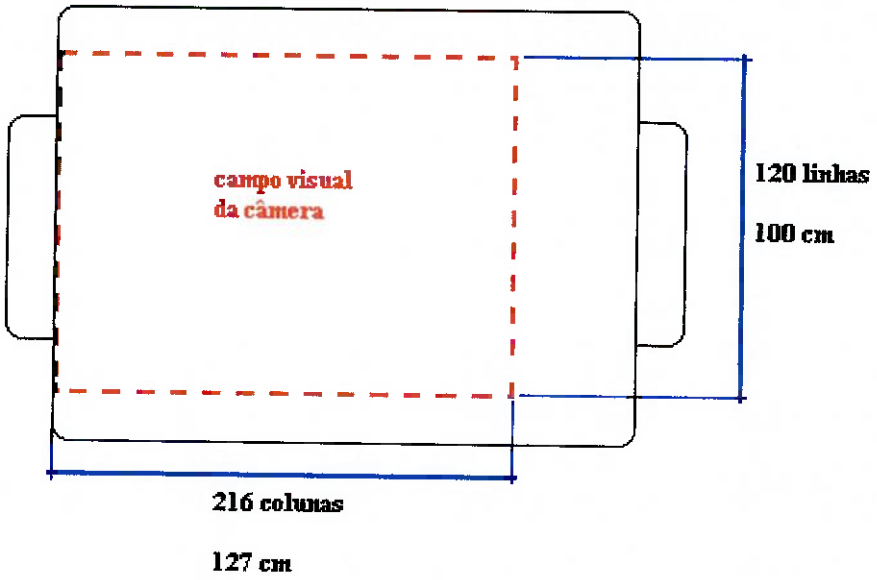



*Figura 11.3.6: escala da matriz.*

A rotina responsável pela escala está descrita no APÊNDICE 5: Rotinas em MatLab para processamento de imagem.

Após a correção da escala, pode-se efetuar a filtragem de objetos, isto é, ruídos tão intensos que chegaram a esse ponto do processamento de imagem interpretados como objetos reais. Para isso será adotada daqui para diante uma nova imagem composta do campo com a bola, dos símbolos representativos dos robôs e alguns ruídos.

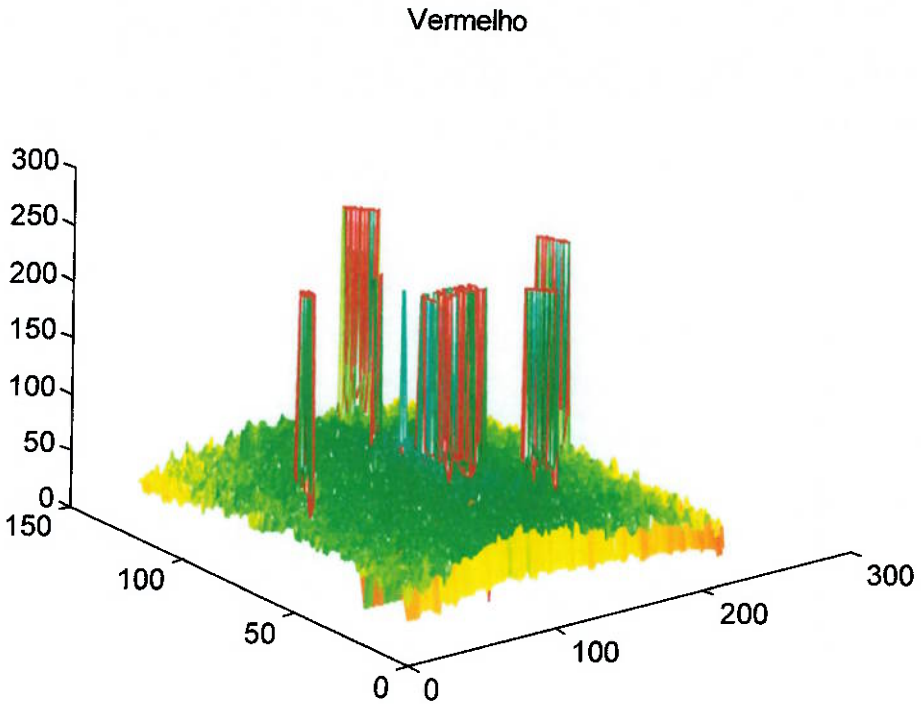



*Figura 11.3.7: imagem completa com ruídos.*

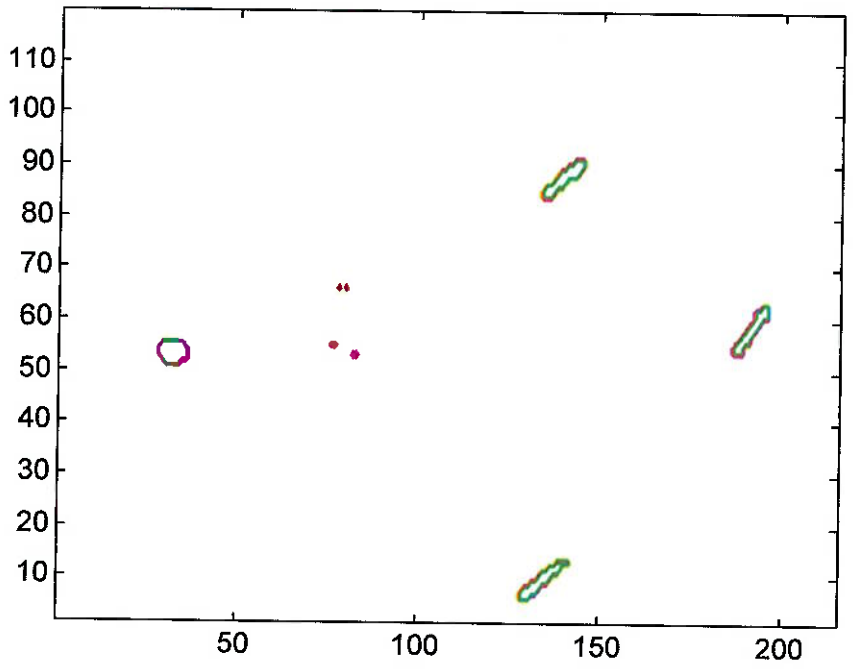



*Figura 11.3.8: bordas.*

Observando a matriz de objetos pode-se perceber que o menor objeto pertinente ao processamento de imagens é a bola. Realizados alguns testes, verificou-se que raramente o número de pontos utilizados em seu contorno era inferior a 15. Alem disso, seu tamanho raramente era menor que 3,5 cm nas direções x e y. Aproveitando-se deste conhecimento é possível retirar da matriz de objetos os objetos que são menores que a bola. Da mesma maneira, verificou-se que os jogadores dificilmente possuíam contornos com menos de 35 pontos. Apesar de a orientação mudar, o comprimento absoluto das marcas dos jogadores era aproximadamente constante e de valor superior a 8 cm. Essas informações são valiosas para ajudar não só na filtragem posterior como também para identificar os jogadores e a bola.

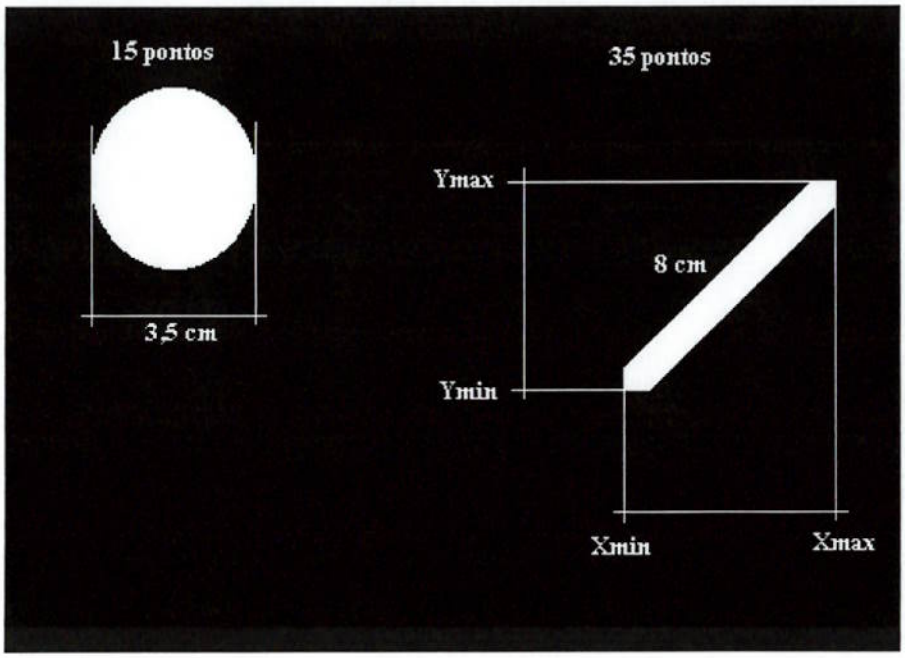



*Figura 11.3.9: dimensões dos objetos*

A rotina que desempenha a filtragem posterior está listada no APÊNDICE 5: Rotinas em MatLab para processamento de imagem.

Em seguida da filtragem posterior é realizada a eliminação de informações redundantes da matriz objeto. As informações redundantes são provenientes de múltiplos contornos encontrados para um mesmo ente físico e ocorrem porque durante o processo de aquisição de imagem a função de

geração de contorno impõe um gradiente de cor, de modo que a borda entre o campo preto e a bola branca, por exemplo, não é um degrau, mas sim uma escada com alguns patamares. A rotina de eliminação de informações redundantes condiciona a existência de um objeto na matriz objeto a uma não ocorrência de outro ente com centro distante no máximo 1 cm. A rotina é a descrita no APÊNDICE 5: Rotinas em MatLab para processamento de imagem.

A matriz objeto da figura 11.3.8 fica então:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 41 | 112,68 | 48,8 | 115,25 | 50,83 | 109,94 | 45,83 | 115,24 | 52,52 | 109,95 | 44,98 |
| 250 | 27 | 19,05 | 44,29 | 21,18 | 45 | 17,04 | 44,17 | 17,64 | 45,85 | 19,99 | 42,48 |
| 250 | 41 | 82,07 | 73,02 | 85,27 | 75 | 78,77 | 70,83 | 85,25 | 75,85 | 78,79 | 69,98 |
| 250 | 45 | 79,61 | 8,18 | 83,5 | 18,83 | 75,84 | 5,83 | 83,49 | 10,85 | 75,85 | 4,98 |

*Tabela 11.3.5: matriz objeto da figura 11.3.8.*

Pode-se perceber claramente a bola com 27 pontos e os jogadores com 41, 41 e 45 pontos cada. Falta agora escrever em arquivo as coordenadas dos entes físicos. Esta escrita, por convenção da equipe, deverá ser da seguinte maneira: para cada objeto, um arquivo diferente deverá ser criado. Deverão ser bola.txt para as coordenadas da bola e r1.txt, r2.txt e r3.txt para os jogadores. A eliminação da indeterminação do ângulo e da identidade deverá ser feita pelo software de estratégia dinamicamente da seguinte maneira: no instante inicial do jogo os jogadores deverão estar dispostos de tal forma que o robô 00 seja o com coordenada X mais elevada, o robô 01 seja o com coordenada Y mais elevada e o robô 10 seja o com coordenada Y mais baixa, todos eles voltados de frente para o eixo Y. A rotina que desempenha a escrita dos arquivos está listada no APÊNDICE 5: Rotinas em MatLab para processamento de imagem.

Para a matriz objeto da tabela 11.3.5, são criados os seguintes arquivos:

-bola.txt = 19.05 44.29;

com bola(1)=$X_{bola}$ e bola(2)=$Y_{bola}$;

-r1.txt= 112.68 48.8 0.755;

com r1(1)=$X_{r1}$, r1(2)=$Y_{r1}$ e r1(3)=$\phi_{r1}$;

-r2.txt= 79.61 8.18 0.58;

com r1(1)=$X_{r1}$,r1(2)=$Y_{r1}$ e r1(3)=$\phi_{r1}$;

-r3.txt= 79.61 8.18 0.58;

com r3(1)=$X_{r3}$,r3(2)=$Y_{r3}$ e r3(3)=$\phi_{r3}$;

## 11.4. Testes e avaliação de desempenho

A realização de testes foi importante para as considerações empregadas durante o desenvolvimento do software e medição da taxa de processamento e erro. Para medição de taxa de processamento foi utilizado um loop para 10 processamentos que consumiram aproximadamente 60 segundos, ou seja, cerca de 1quadro a cada 6 segundos. Vale lembrar que equipes medianas de futebol de robôs, utilizando equipamento indicado na solução 1, processam no mesmo período de tempo (6 segundos) cerca de 200 quadros.

Avaliando o sistema de visão computacional como um instrumento de medição, é possível medir sua precisão da seguinte maneira: mantendo a configuração física inalterada, realizam-se várias medições e comparam-se os resultados obtidos. Tem-se então para 5 medidas:

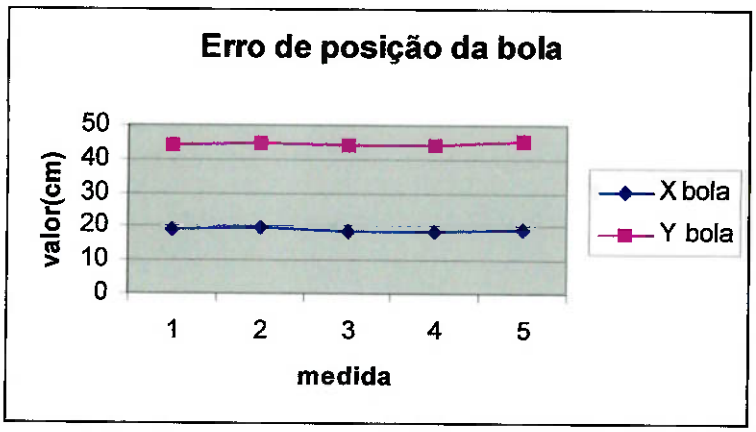



*Figura 11.4.1: erro de medição da bola*

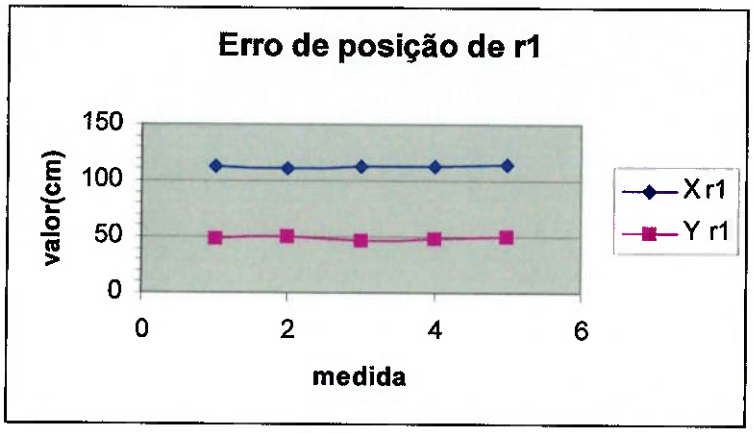



*Figura 11.4.2: erro de posição de r1.*

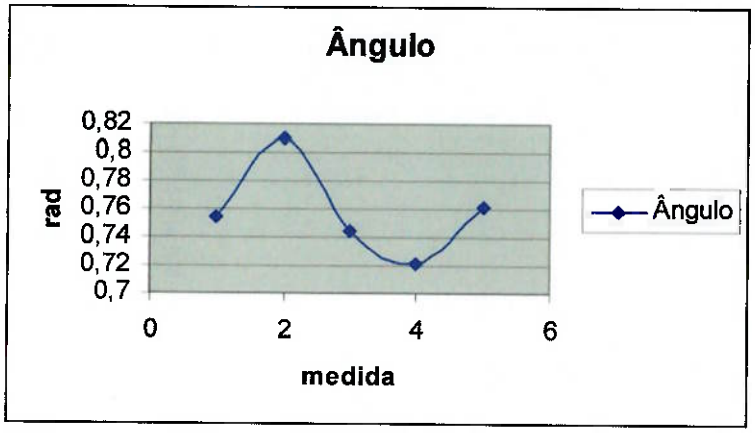



*Figura 11.4.3: erro de orientação de r1.*

Como observado nos gráficos, para a bola houve um erro de 4% nas medidas. Já para r1, os erros de posição em X e Y e de orientação são respectivamente: 3%, 6% e 11%. São considerados pequenos e melhores que o esperado.

# 12.Conclusões

O futebol de robôs mostrou-se, no decorrer deste trabalho, uma fonte enorme de problemas de engenharia mecatrônica e possibilitou aos alunos integrantes da equipe o conhecimento e compreensão de diversas áreas na engenharia e robótica.

A equipe acredita ser possível refinar ainda mais alguns pontos do sistema futebol de robôs, mas está claro que apesar do grande esforço desprendido, com o hardware de processamento de imagem atual a equipe dificilmente terá condições de competir oficialmente algum dia.

## Apêndice 1: Programa transmissor

```
;*****************************************************************
;       Programa do transmissor  CALC
;*****************************************************************
        LIST    p=16C54C ; PIC16C54C is the target processor
          #include "P16C5x.INC" ; Include header file
; endereços das variáveis
tempo1          equ     08H
tempo2          equ     09H
dado0 equ       0AH
          org     0
          goto    config
;*****************************************************************
; Loop cheio de tempo para recepção
;*****************************************************************
; A rotina permanece em loop1 até o valor do timer ser igual ao valor
; da variável tempo1, que é a cte de tempo da transmissão via rádio.
; Para verificar se o timer é igual a tempo1, subtrai-se um do outro
; e testa-se o bit 2 do registrador STATUS, afetado em operações
; matemáticas.
;*****************************************************************
loop1   movf    TMR0,0
        subwf   tempo1,0
        btfss   STATUS,2
         goto loop1
        clrf    TMR0
        retlw   00H
;*****************************************************************
; Meio-loop  de tempo para recepção
;*****************************************************************
; A rotina permanece em loop2 até o valor do timer ser igual ao valor
```

```
; da variável tempo2, que é (cte de tempo da transmissão via rádio)/2.
; Para verificar se o timer é igual a tempo2, subtrai-se um do outro
; e testa-se o bit 2 do registrador STATUS, afetado em operações
; matemáticas.
;*****************************************************************
loop2   movf    TMR0,0
        subwf   tempo2,0
        btfss   STATUS,2
         goto loop2
        clrf    TMR0
        retlw   00H
;*****************************************************************
; Configuração
;*****************************************************************
; Esta subrotina configura as portas de i/o e os endereços das variáveis.
;*****************************************************************
config  movlw   04H
          TRIS  PORTA
        movlw   H'FF'
          TRIS  PORTB
          movlw   H'7E'
        movwf   tempo1
          movlw   H'3F'
        movwf   tempo2
        movlw   H'00'
          movwfdado0
          movlw H'03'
        OPTION
          goto    start
;***********************
; Início do programa   *
;***********************
```

```
; A0 é saída para o rádio.A1 é saída para Busy.
start   bsf     PORTA,1
        movf    PORTB,0
        subwf dado0,0                   ;Verifica se o dado na porta é igual ao
anterior
        btfsc   STATUS,2
          call  igual                   ;ou se é um dado novo
        btfss   STATUS,2
          call  novo
igual   bcf     PORTA,1                 ;Se igual, nada acontece
        call    loop2
        goto    start
novo    bsf     PORTA,0                 ;Se novo, a transmissão começa
        call    loop1
        bcf     PORTA,0
        call    loop2
        btfss   PORTB,0
       bcf    PORTA,0
      btfsc   PORTB,0
       bsf    PORTA,0
        call    loop1
        btfss   PORTB,1
       bcf    PORTA,0
      btfsc   PORTB,1
       bsf    PORTA,0
        call    loop1
        btfss   PORTB,2
       bcf    PORTA,0
      btfsc   PORTB,2
       bsf    PORTA,0
        call    loop1
        btfss   PORTB,3
```

```
 bcf   PORTA,0
btfsc  PORTB,3
 bsf   PORTA,0
  call    loop1
  btfss  PORTB,4
 bcf   PORTA,0
btfsc  PORTB,4
 bsf   PORTA,0
  call    loop1
  btfss  PORTB,5
 bcf   PORTA,0
btfsc  PORTB,5
 bsf   PORTA,0
  call    loop1
  btfss  PORTB,6
 bcf   PORTA,0
btfsc  PORTB,6
 bsf   PORTA,0
  call    loop1
  btfss  PORTB,7
 bcf   PORTA,0
btfsc  PORTB,7
 bsf   PORTA,0
  call    loop1
  movf   PORTB,0
  movwfdado0
  bcf    PORTA,0
  bcf    PORTA,1
  call    loop1
  call    loop1
  goto   start
```

END

## Apêndice 2: Programa receptor

```
;*****************************************************************
;     Programa Receptor  C.A.L.C.  Robô 00    full step 2 phase on  *
;*****************************************************************
        LIST   p=16C54 ; PIC16C54C 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
r1      movf   TMR0,0
        subwf  tempo,0
        btfss  STATUS,2
        goto   r1
        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
fi1    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    fi1
        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

# Apêndice 3: Programas para testes de desempenho

```
/*****************************************
Programa de teste do transmissor-receptor em C
*****************************************/
#include <stdio.h>
#include <bios.h>
#include <dos.h>
void main(){
int i;
for(i=0;i<=1350i++)
   {
   biosprint(0,random(255),0);   // byte aleatório ,
   delay(2000);  // mandado para a porta paralela a cada 2000 //ms
   até encerrar o teste com 1350 palavras enviadas
   }
}
```

Em assembler:

```
;*********************************************************************
;    Programa Receptor  C.A.L.C.  Robô 00    teste de desempenho
;*********************************************************************
   LIST   p=16C54C ; PIC16C54 is the target processor
   #include "P16C5x.INC" ; Include header file
;
; posições RAM utilizadas no programa
;
tempo  equ   H'08'
dado   equ   H'09'
;*********************************************************************
```

```
; 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
r1      movf    TMR0,0
    subwf   tempo,0
    btfss   STATUS,2
     goto   r1
    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.
;********** ********************************************************************
```

```
; 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
   retlw  00H
;*********************************************************
; Rotina de tratamento
;**** ***************************************************
; Esta rotina compara o valor de dado e da porta C, que contém a
porta ;paralela, gerando um pulso no pino 1 da porta A caso as
palavras ;comparadas sejam diferentes
;************ ***********************************************
trata  bcf      PORTA,1
   movf   dado,0
```

```
        subwf  PORTB,0
        btfss  STATUS,2
        bsf    PORTA,1
        call   loopr
        bcf    PORTA,1
        goto   start
;*****************************************************************
; Configuração
;*****************************************************************
; Esta subrotina configura as portas de i/o e os endereços das
variáveis.
;*********************************************************************************
config  movlw  01H
        TRIS   PORTA
        movlw  FFH
        TRIS   PORTB
        goto   start
;************************
; Início do programa
;************************
start   btfss  PORTA,0
        goto   start
        call   recep
        goto   trat
        END
```

# Apêndice 4: Programas em C para captura de imagem

Para ligar a placa:

```
/*************************************************************
   Programa liga placa
   Liga.c
*************************************************************
   data da última atualização:21/11/2001
   Descrição: programa aciona a placa de aquisição
   de imagem
*************************************************************/
#include <conio.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <memory.h>
#include <graphics.h>

void limpa_tela(void);  //Protótipos das funções
void liga_placa(void);
int  acha_driver(char*);
void executa(union REGS* , struct SREGS*)
void janela_de_video(void);


#define ALTURA              300      //Eixos vao de 0 a 1000
#define LARGURA             300
#define ORIGEM_DA_JANELA_X   0
#define ORIGEM_DA_JANELA_Y   0
```

```
int          numero_da_interrupcao;  //Variáveis globais
union REGS     regs;
struct SREGS    segregs;
typedef  struct tagWINREC{
                              int wX ;
                              int wY ;
                              int wW ;
                              int wH ;
                              } WINREC ;
typedef WINREC far* LPWINREC ;
WINREC  winrec ;
unsigned int  wXpos , wYpos , wWidth , wHeight ;
char   huge* ptrbuffer;
char   huge* ptr        ;
long   memoria;
/*************************************************************
   Função limpa tela
*************************************************************
   data da última atualização:21/11/2001
   Descrição: função prepara a tela para receber
   vídeo
*************************************************************/
void limpa_tela(void)
{
regs.x.ax = 3;
int86(0x10, &regs, &regs);
}
/*************************************************************
   Função liga placa
*************************************************************
   data da última atualização:21/11/2001
```

```
    Descrição: função aciona a placa de aquisição
 ************************************************************/
 void liga_placa(void)
 {
 regs.x.bx = 0x01;
 executa(&regs, &segregs);
 }
 /************************************************************
    Função acha driver
 ************************************************************
    data da última atualização:21/11/2001
    Descrição: função procura driver da placa
 ************************************************************/
 int acha_driver(char *assinatura)
 {
 void far *lpIntrVect;
 int i,achou,comprimento;
 achou=0;
 comprimento=strlen(assinatura);
 for(i=0x80;i<0xC0 && !achou;i++)
   {
   lpIntrVect=(void far *)getvect(i);
   FP_OFF(lpIntrVect)=0x103;
   if(!_fstrnicmp(lpIntrVect,(char far *)assinatura,comprimento))
     {
     achou=i;
     break;
     }
   }
 return(achou);
 }
 /************************************************************
```

```
   Função executa
************************************************************
   data da última atualização:21/11/2001
   Descrição: função faz requisição da interrupção
   da placa com os registradores apropriados
************************************************************/
void executa(union REGS *regs, struct SREGS *segregs)
{
int86x(numero_da_interrupcao, regs, regs, segregs);
}
/************************************************************
   Função janela de vídeo
************************************************************
   data da última atualização:21/11/2001
   Descrição: função realiza o set-up da placa para
   receber vídeo
************************************************************/
void janela_de_video(void)
{
regs.x.bx=0x105;       //Logical window
regs.x.ax=1000;
regs.x.dx=1000;
executa(&regs,&segregs);
regs.x.bx=0x117;       //Mascara de cor
regs.x.ax=0;
executa(&regs,&segregs);
regs.x.bx=0x115;       //Posicao da janela
regs.x.ax=ORIGEM_DA_JANELA_X;
regs.x.dx=ORIGEM_DA_JANELA_Y;
executa(&regs,&segregs);
regs.x.bx=0x116;       //Tamanho da janela
regs.x.ax=LARGURA;
```

```
 regs.x.dx=ALTURA;
 executa(&regs,&segregs);
 regs.x.bx=0x108;       //Fit-mode
 regs.x.ax=4;
 executa(&regs,&segregs);
 regs.x.bx=0x112;       //Mostra-nao mostra a janela
 regs.x.ax=1;
 executa(&regs,&segregs);
 regs.x.bx=0x110;       //Sistema de video:NTSC
 regs.x.ax=1;
 executa(&regs,&segregs);
 regs.x.bx=0x26;        //Traduz coordenadas
 regs.x.ax=ORIGEM_DA_JANELA_X;
 regs.x.dx=ORIGEM_DA_JANELA_Y;
 executa(&regs,&segregs);
 wXpos=regs.x.ax;
 wYpos=regs.x.dx;
 regs.x.bx=0x26;
 regs.x.ax=LARGURA;
 regs.x.dx=ALTURA;
 executa(&regs,&segregs);
 wWidth=regs.x.ax;
 wHeight=regs.x.dx;
 if(wWidth & 3)
 wWidth=(wWidth &~3) +4 ;
 winrec.wX=wXpos;
 winrec.wY=wYpos;
 winrec.wW=wWidth;
 winrec.wH=wHeight;
 }
 /*************************************************************
    Função main
```

```
*************************************************************
   data da última atualização:21/11/2001
   Descrição: função realiza o seqüenciamento das
   funções para ligar a placa
*************************************************************/
void main(void)
{
limpa_tela();
numero_da_interrupcao=acha_driver("VBLAST");
liga_placa();
janela_de_video();
}
```

Para capturar imagem e escrever em arquivo:

```
/*************************************************************
   Programa aquisição de imagem e escrita
   Foto3.r
*************************************************************
   data da última atualização:21/11/2001
   Descrição: programa aciona a placa de aquisição
   de imagem
*************************************************************/
#include <conio.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <memory.h>
#include <graphics.h>
```

```
void executa(union REGS* , struct SREGS*) //Protótipos das funções
void janela_de_video(void);
void aloca_memoria(void);
void monta_arquivo_do_buffer(void);


#define ALTURA              300      //Eixos vao de 0 a 1000
#define LARGURA             300
#define ORIGEM_DA_JANELA_X  0
#define ORIGEM_DA_JANELA_Y  0

int          numero_da_interrupcao;  //Variáveis globais
union REGS     regs;
struct SREGS   segregs;
typedef  struct tagWINREC{
                          int wX ;
                          int wY ;
                          int wW ;
                          int wH ;
                          } WINREC ;
typedef WINREC far* LPWINREC ;
WINREC  winrec ;
unsigned int  wXpos , wYpos , wWidth , wHeight ;
char   huge* ptrbuffer;
char   huge* ptr        ;
long   memoria;

/*************************************************************
   Função executa
*************************************************************
   data da última atualização:21/11/2001
   Descrição: função faz requisição da interrupção
```

```
   da placa com os registradores apropriados
*************************************************************/
void executa(union REGS *regs, struct SREGS *segregs)
{
int86x(numero_da_interrupcao, regs, regs, segregs);
}
/*************************************************************
   Função janela de vídeo
*************************************************************
   data da última atualização:21/11/2001
   Descrição: função realiza o set-up da placa para
   receber vídeo
*************************************************************/
void janela_de_video(void)
{
regs.x.bx=0x105;       //Logical window
regs.x.ax=1000;
regs.x.dx=1000;
executa(&regs,&segregs);
regs.x.bx=0x117;       //Mascara de cor
regs.x.ax=0;
executa(&regs,&segregs);
regs.x.bx=0x115;       //Posicao da janela
regs.x.ax=ORIGEM_DA_JANELA_X;
regs.x.dx=ORIGEM_DA_JANELA_Y;
executa(&regs,&segregs);
regs.x.bx=0x116;       //Tamanho da janela
regs.x.ax=LARGURA;
regs.x.dx=ALTURA;
executa(&regs,&segregs);
regs.x.bx=0x108;       //Fit-mode
regs.x.ax=4;
```

```
executa(&regs,&segregs);
regs.x.bx=0x112;        //Mostra-nao mostra a janela
regs.x.ax=1;
executa(&regs,&segregs);
regs.x.bx=0x110;        //Sistema de video:NTSC
regs.x.ax=1;
executa(&regs,&segregs);
regs.x.bx=0x26;         //Traduz coordenadas
regs.x.ax=ORIGEM_DA_JANELA_X;
regs.x.dx=ORIGEM_DA_JANELA_Y;
executa(&regs,&segregs);
wXpos=regs.x.ax;
wYpos=regs.x.dx;
regs.x.bx=0x26;
regs.x.ax=LARGURA;
regs.x.dx=ALTURA;
executa(&regs,&segregs);
wWidth=regs.x.ax;
wHeight=regs.x.dx;
if(wWidth & 3)
wWidth=(wWidth &~3) +4 ;
winrec.wX=wXpos;
winrec.wY=wYpos;
winrec.wW=wWidth;
winrec.wH=wHeight;
}
/*************************************************************
   Função aloca memória
*************************************************************
   data da última atualização:21/11/2001
   Descrição: função aloca memória para
   onde será passada a imagem
```

```
 ********************************************************/
 void aloca_memoria(void)
 {
 memoria = wWidth*3 ;
 memoria *=wHeight  ;
 if( (ptr = (char huge *) halloc(memoria, sizeof(char)))==NULL)
  {
  desliga_placa();
  gotoxy(2,23);
  printf("Alocacao falhou") ;
  }
}
/***********************************************************
   Função monta arquivo do buffer
************************************************************
   data da última atualização:21/11/2001
   Descrição: função escreve as componentes vermelhas
   da imagem em arquivo
************************************************************/
void monta_arquivo_do_buffer(void)
{
int i,j,auxBlue,auxGreen,auxRed;
long contador;
FILE * imp1;
WINREC far* lpWinRec = (WINREC far*) &winrec    ;
ptrbuffer=ptr ;
regs.x.ax  = 1                  ;
segregs.es = FP_SEG((char far*) ptr) ;
regs.x.di  = FP_OFF((char far*) ptr) ;
segregs.ds = FP_SEG(lpWinRec) ;
regs.x.si  = FP_OFF(lpWinRec) ;
regs.x.bx  = 0x1C ;
```

```
    executa(&regs,&segregs);
    imp1=fopen("c:foto3.r","w");
    ptrbuffer++;ptrbuffer++;
    for( contador= 2 ; contador < memoria ;)
      {
      if ((unsigned)*ptrbuffer >255)
       fprintf(imp1," 255");
      else
       fprintf(imp1," %d",(unsigned)*ptrbuffer);
      ptrbuffer++;contador++;
      ptrbuffer++;contador++;
      ptrbuffer++;contador++;
      j++;
      if(j==((LARGURA/100)*72))
       {
       j=0;
       fprintf(imp1,"\n");
       }
     }
  fclose(imp1);
  }
  /*************************************************************
     Função main
  *************************************************************
     data da última atualização:21/11/2001
     Descrição: função realiza o seqüenciamento das
     funções para ligar a placa
  *************************************************************/
  void main(void)
  {
  janela_de_video();
  aloca_memoria(void);
```

```
monta_arquivo_do_buffer(void);
}
```

# Apêndice 5: Rotinas em MatLab para processamento de imagem

Para abrir arquivo e executar tratamento inicial:

```
%visao3
dos('foto3');
load foto3.r;
for i=1:120
for j=1:216
if foto3(i,j)<255;
foto3(i,j)=0;
end;
end;
end;
foto3=flipud(foto3);
for i=70:80
for j=80:120
foto3(i,j)=0;
end;
end;
contour(foto3);
c=contourc(foto3);
borda;
escala;
filtro;
elimina;
arq;
```

Para transformar a matriz contorno na matriz objeto :

```
%borda
[a b]=size(c);
ji=1;
jf=0;
xmax=0;
ymax=0;
xmin=216;
ymin=120;
yassxmax=0;
yassxmin=0;
xassymax=0;
xassymin=0;
x=0;
y=0;
pontos=0;
obj=1;
while jf~=b
jf=ji+c(2,ji);
for q=ji+1:jf
x=x+c(1,q);
y=y+c(2,q);
if c(1,q)>xmax
xmax=c(1,q);
yassxmax=c(2,q);
end
if c(1,q)<xmin
xmin=c(1,q);
yassxmin=c(2,q);
end;
if c(2,q)>ymax
ymax=c(2,q);
```

```
 xassymax=c(1,q);
 end;
 if c(2,q)<ymin
 ymin=c(2,q);
 xassymin=c(1,q);
 end;
 pontos=pontos+1;
 end;
 objeto(obj,1)=c(1,ji);
 objeto(obj,2)=pontos;
 objeto(obj,3)=x/pontos;
 objeto(obj,4)=y/pontos;
 objeto(obj,5)=xmax;
 objeto(obj,6)=yassxmax;
 objeto(obj,7)=xmin;
 objeto(obj,8)=yassxmin;
 objeto(obj,9)=xassymax;
 objeto(obj,10)=ymax;
 objeto(obj,11)=xassymin;
 objeto(obj,12)=ymin;
 ji=ji+c(2,ji)+1;
 obj=obj+1;
 x=0;
 y=0;
 pontos=0;
 xmax=0;
 ymax=0;
 xmin=216;
 ymin=120;
 yassxmax=0;
 yassxmin=0;
 xassymax=0;
```

```
xassymin=0;
end;
escala;
```

A rotina de escala é:

```
%escala
[a b]=size(objeto);
for i=1:a
objeto(i,3)=objeto(i,3)*(127/216);
objeto(i,4)=objeto(i,4)*(100/120);
objeto(i,5)=objeto(i,5)*(127/216);
objeto(i,6)=objeto(i,6)*(100/120);
objeto(i,7)=objeto(i,7)*(127/216);
objeto(i,8)=objeto(i,8)*(100/120);
objeto(i,9)=objeto(i,9)*(127/216);
objeto(i,10)=objeto(i,10)*(100/120);
objeto(i,11)=objeto(i,11)*(127/216);
objeto(i,12)=objeto(i,12)*(100/120);
end
```

Filtragem posterior:

```
%filtro
[a b]=size(objeto);
while i<=a
if objeto(i,2)<=15
objeto(i,:)=[ ];
i=1;
end;
[a b]=size(objeto);
i=i+1;
```

```
end;
[a b]=size(objeto);
while i<=a
if objeto(i,2)>=30,objeto(i,2)<=35
objeto(i,:)=[ ];
i=1;
end;
[a b]=size(objeto);
i=i+1;
end;
```

Eliminação de objetos redundantes:

```
%elimina
[a b]=size(objeto);
i=1;
i1=1;
while i<=a
while i1<=a
if objeto(i,3)<=(objeto(i1,3)+1),objeto(i,3)>=(objeto(i1,3)-
1),objeto(i,4)<=(objeto(i1,4)+1),objeto(i,4)>=(objeto(i1,4)+1)
if i1~=i
objeto(i,:)=[ ];
i1=0;
end;
end;
[a b]=size(objeto);
i1=i1+1;
end;
i=i+1;
[a b]=size(objeto);
end;
```

Escrita em arquivo:

```
%arq
[a b]=size(objeto);
i=1;
while i~=a
if objeto(i,2)<35
final(1,1)=objeto(i,3);
final(1,2)=objeto(i,4);
final(1,3)=0;
linha=i;
end;
i=i+1;
end;
objeto(linha,:)=[ ];
i=1;
[a b]=size(objeto);
while i~=a+1
final(i+1,1)=objeto(i,3);
final(i+1,2)=objeto(i,4);
final(i+1,3)=atan2(objeto(i,6)-objeto(i,8),objeto(i,5)-objeto(i,7));
i=i+1;
end;
[a b]=size(final);
if final(1,1)~=0,final(1,2)~=0,final(1,3)==0
fid = fopen('bola.txt','W');
fprintf(fid,'%d %d',final(1,1),final(1,2));
fclose(fid);
end;
if a>=2
fid = fopen('r1.txt','W');
fprintf(fid,'%d %d %d',final(2,1),final(2,2),final(2,3));
```

```
fclose(fid);
end;
if a>=3
fid = fopen('r2.txt','W');
fprintf(fid,'%d %d %d',final(3,1),final(3,2),final(3,3));
fclose(fid);
end;
if a>=4
fid = fopen('r3.txt','W');
fprintf(fid,'%d %d %d',final(4,1),final(4,2),final(4,3));
fclose(fid);
end;
```

# Apêndice 6: regras

## 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.

## 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 versão 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

**[25]** Levine, Martin D. "Vision in man and machine", ed. McGraw Hill

**[26]** Mitra, Sanjit K.; Kaiser, James F. "Handbook of Digital Signal Processing, ed. Wiley Interscience

**[27]** Barret, Anthony "Computer Vision and Image Processing, wd. Chapman & Hall

**[28]** Henson, C. S. "Solution of Problems in Telecommunications and Electronics", ed. Pitmam & Sons

**[29]** Schildt, Herbert "C Completo e Total", ed. McGraw Hill

**[30]** Knuth, Donald E. "The art of computer programming",ed. Addison-Wesley

**[31]** internet – www.matrox.com