UNIVERSIDADE DO OESTE PAULISTA
UNOESTE
PRESIDENTE PRUDENTE - SP

FACULDADE DE INFORMÁTICA DE PRESIDENTE PRUDENTE – CURSO DE SISTEMAS DE INFORMAÇÃO

FUTEBOL DE ROBÔS, UMA APLICAÇÃO DE ROBÓTICA

RODOLFO MARENGO SOBREIRA

Presidente Prudente – SP
2003

UNIVERSIDADE DO OESTE PAULISTA
UNOESTE
PRESIDENTE PRUDENTE - SP

FACULDADE DE INFORMÁTICA DE PRESIDENTE PRUDENTE – CURSO DE SISTEMAS DE INFORMAÇÃO

FUTEBOL DE ROBÔS, UMA APLICAÇÃO DE ROBÓTICA

RODOLFO MARENGO SOBREIRA

Trabalho monográfico apresentado no curso de graduação, Bacharelado em Sistemas de Informação, como requisito parcial para sua conclusão.
Área de concentração: Arquitetura de Sistemas de Computação.

Orientador:
Francisco Assis da Silva

Co-orientador:
Renato Luciano Ros

Presidente Prudente – SP
2003

FICHA CATALOGRÁFICA

004 Sobreira, Rodolfo Marengo
Futebol de robôs, uma aplicação de robótica / Rodolfo Marengo Sobreira. - Local :[s.n), 2003.
44p. : il

Monografia de Graduação
Bibliografia
Universidade do Oeste Paulista – Unoeste: Presidente Prudente – SP, 2003

1. Robótica 2. Microcontrolador PIC. I. Autor. II. Título.

DEDICATÓRIA

Dedico este trabalho todos aqueles que apoiaram, confiaram e foram pacientes no convívio e partilha dos momentos difíceis, pela vida, sabedoria e oportunidades; Osvaldo Marengo e Sidney Zaparoli Marengo, meus avós e minha mãe Neucy Mara Marengo pelo apoio e incentivo e pela educação e formação do meu caráter.

AGRADECIMENTOS

A minha família que, em todos os momentos de realização desta pesquisa, esteve presente.

Agradeço a Deus por me permitir sonhar e acreditar que, com trabalho, esforço e dedicação, tudo pode ser conquistado. Agradeço ainda, às pessoas que conheci ao longo dessa passagem, que foram o instrumento escolhido por Ele para que eu pudesse aprender essa lição. Principalmente aos orientadores Francisco Assis da Silva e Renato Luciano Ros e ao professor Emerson Silas Dória.

EPÍGRAFE

"Não há nada que melhor defina uma pessoa do que aquilo que ela faz quando tem toda a liberdade de escolha."

Willian M.Bulger

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

SOBREIRA, Rodolfo Marengo. **Futebol de robôs, uma aplicação de robótica**. Presidente Prudente, 2003. Monografia de Graduação – Universidade do Oeste Paulista.

RESUMO

O futebol de robôs é uma área de pesquisa que está se tornando muito difundida nas universidades de todo o mundo, essa área pode envolver extrema complexidade, principalmente na programação do software controlador. Partindo desta perspectiva, a pesquisa e o desenvolvimento de times de futebol de robôs torna-se de importante valia no meio acadêmico.

O objetivo principal do projeto é a construção de um robô jogador de futebol e um software controlador. Para a construção do robô são utilizadas algumas tecnologias não muito usuais nos cursos de computação, como por exemplo: microcontroladores PIC e transmissão de dados via ondas de rádio. O microcontrolador PIC é programado com um algoritmo que se destina à interpretação dos comandos enviados pelo computador controlador para rotacionar os motores e movimentar o robô de um local para outro. O software controlador faz uso de uma câmera de vídeo para capturar as imagens do campo e com isso realizar a identificação e posicionamento do robô.

Nesta monografia, são descritos alguns conceitos e informações importantes a respeito de campeonatos de futebol de robôs realizados por federações competentes. Adicionalmente, encontra-se descrições de funcionamento, características e arquitetura do microcontrolador PIC e do transceptor de ondas de rádio utilizado para a comunicação do robô com o computador controlador.

SOBREIRA, Rodolfo Marengo. **Robot soccer, a robotic application**. Presidente Prudente, 2003. Monografia de Graduação – Universidade do Oeste Paulista.

ABSTRACT

The robot soccer is a research area that is becoming very disseminate in the whole world universities, this area can involve extreme complexity, mainly in the controller software programming. Leaving of this perspective, the research and the development of robot soccer team become of important value in the academic class.

The main objective of the project is the construction of robot soccer player and controller software. Some technologies not very usual in the computation courses are used for the robot construction, for example: PIC microcontrollers and transmission of data by radio waves. PIC microcontroller is programmed with an algorithm that destinates the interpretation of the commands sent by the controller computer to rotate the motors and to move the robot of a place for another one. The controller software makes use of a video camera to capture the images of the field and make the robot identification and positioning.

In this job, are described some important concepts and information about robot soccer championships maked for competent federations. Additionally, are described things about functioning, characteristics and architecture of the PIC microcontroller and transceptor of radio waves used for the communication of the robot with the controller computer.

1 INTRODUÇÃO

Antigamente os robôs eram vistos como máquinas ameaçadoras para a humanidade ou simplesmente vistos em filmes de ficção científica. Nos dias de hoje, pode-se ver robôs em indústrias, em explorações a locais de difícil acesso por humanos (por exemplo: na inspeção de planetas, vulcões, fundo do oceano, tubulações, etc) ou em situações perigosas (por exemplo: no desarme de bombas e de minas terrestres). A indústria automobilística tem utilizado de forma rotineira robôs de soldagem, com os quais é possível a construção de complexas estruturas dos modernos automóveis com baixo custo e altíssima qualidade.

A robótica vem despertando o interesse dos jovens estudantes, como exemplo disso, surgiu o futebol de robôs e suas federações, promovendo campeonatos em diversas modalidades, tendo modalidades em que o custo para montagem de um time não é muito alto. Por trás dessa aparente diversão está um estudo complexo em vários âmbitos, como: inteligência artificial, visão computacional, eletrônica, linguagens de programação, estudo dos movimentos dos robôs, etc.

Partindo dessa perspectiva, a pesquisa e o desenvolvimento de futebol entre robôs, além de serem extremamente motivantes por possibilitar o surgimento de um espírito de ciência e tecnologia nas universidades, constituem uma atividade que possibilita a realização de experimentos reais para o desenvolvimento e testes de robôs que apresentam comportamento inteligente e que cooperam entre si para a execução de uma tarefa, formando um time [MAIA, 2000].

O presente projeto de pesquisa tem por objetivo principal estudar assuntos relacionados à eletrônica, tais como: microcontrolador PIC e transmissão de dados via ondas de rádio utilizando a porta serial do computador. O estudo de visão computacional também é assunto relevante que compõe o objetivo desse trabalho.

Foi construído um robô com dimensões de 8 x 8 cm, dotado de dois motores de corrente contínua e um circuito eletrônico para o controle dos motores e recepção dos comandos enviados pelo computador controlador. A comunicação do computador controlador com o robô foi idealizada e desenvolvida utilizando transmissão de ondas de rádio, para isto, também foi construído um circuito eletrônico.

O projeto também está constituído do desenvolvimento de um software para capturar as imagens de uma câmera de vídeo, e, a partir dessas imagens definir o posicionamento do robô.

Os próximos parágrafos estão organizados conforme a estrutura que segue:

O capítulo 2 aborda as características e conceitos sobre o futebol de robôs, suas federações e suas regras.

No capítulo 3 é feita a descrição dos conceitos de portas seriais e microcontroladores PIC.

2 FUTEBOL DE ROBÔ

Apesar do aspecto à primeira vista de um mero brinquedo, na verdade a proposta de uma partida de futebol jogada por robôs autônomos envolve aspectos de extrema complexidade sob a ótica da ciência robótica. Além de serem necessários robôs especialistas, o duelo principal no futebol de robôs é tanto entre *software* como *hardware*, ou seja, a programação dos robôs e o melhor *hardware*, onde quem conseguir um melhor desempenho vence a partida.

O futebol de robô foi criado em 1996 pelo professor Jong–Hwan Kim, do Departamento de Engenharia Elétrica do Kaist (Korean Advanced Institute of Science and Technology), da República da Coréia (Coréia do Sul). Em 1997 a idéia foi trazida para o Brasil pelo Instituto de Automação da Fundação Centro Tecnológico da Informática – CTI (Campinas – SP), por meio do pesquisador Roberto Tavares Filho [UFPR].

Segundo Kraetzchmar et al (1998), “Dispositivos mecatrônicos, hardware especializado para o controle de sensores e atuadores, teoria de controle, interpretação e fusão sensorial, redes neurais, computação evolutiva, visão, e sistemas multi agentes são exemplos de campos envolvidos nesse desafio” [BIANCHI, 2000].

No ano de 1998, começaram a ficar bem conhecidos os campeonatos mundiais de futebol de robôs, envolvendo diversas universidades, surgindo a necessidade de criar regras definidas para garantir a compatibilidade dos times.

2.1 Federações

Os pesquisadores coreanos fundaram a Fira (Federação Internacional de Futebol de Robôs), estabelecendo que os jogos são disputados entre dois times compostos cada qual de três micro–robôs, em formatos de cubos metálicos, e devendo ser dotados de baterias e comunicação sem fio [UFPR].

E paralelamente a empresa Sony do Japão incentivou o surgimento de competições de futebol de robôs em escolas e universidades, o que levou à criação de uma outra federação denominada RoboCup. E também havia uma liga parecida com a Fira, denominada Small, com a diferença de que são cinco jogadores e o campo tem as

dimensões de uma mesa semelhante à que é usada em jogos de tênis de mesa. Os robôs da RoboCup não precisam ter formato de cubos [UFPR].

Hoje em dia, na Robocup, há cinco modalidades: a liga de pequeno porte, para robôs menores; a liga de médio porte; a liga de simulação, que é realizada utilizando apenas o computador, sem a presença física de robôs; a liga de humanóides que utiliza robôs com formas humanas; e a liga dos quadrúpedes da Sony, que tem os famosos cães denominados de Aibo, como jogadores. Há também, além dessas cinco modalidades, a competição de robôs que simulam uma missão de resgate em um cenário de desastre.

Na Fira existem outras categorias como a de robôs humanóides sendo a HuroSot, a KheperaSot, a liga de simulação chamada de SimuroSot, existe também a QuadroSot, RoboSot e as duas modalidades mais conhecidas que são a Mirosot e a NaroSot. Em ambas, os robôs são pequenos em forma de cubos metálicos com rodinhas. Apesar de a regra não impedir que tenha outros formatos e membros, mas é necessário ficar dentro das dimensões previstas para cada categoria, o que acaba sendo uma forte limitação. Por isso, geralmente utiliza-se o formato cúbico [RAMALHO].

2.2 Regras

O funcionamento de cada time segue uma mesma fórmula básica: cada time realiza a aquisição da imagem por meio da sua câmera, processa a imagem usando técnicas de visão computacional para descobrir a posição de todos os robôs e da bola. Com essa imagem, um sistema de decisão define a melhor estratégia a aplicar e os movimentos instantâneos de cada robô. Todo esse processamento é realizado em um único computador. Com a decisão de movimentação tomada, um sistema de comunicação por rádio envia para os robôs uma mensagem com o movimento a ser realizado. A figura 1 mostra um esquema do funcionamento do futebol de robô. A figura 2 ilustra o funcionamento do futebol de robô.

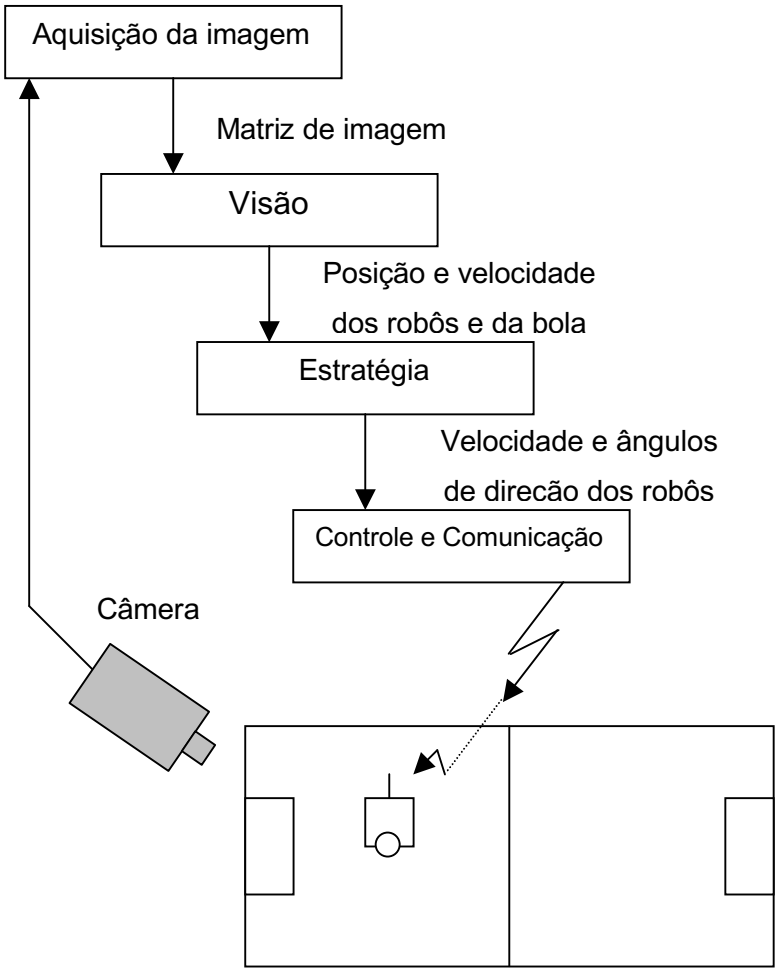

Figura 1 - Esquema do funcionamento do futebol de robô.
Fonte: [COSTA, 2000].

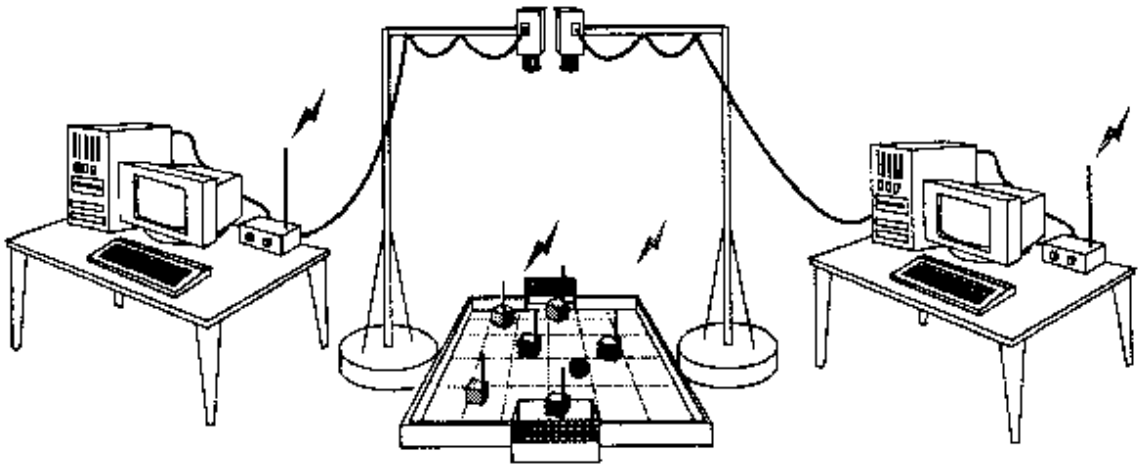

Figura 2 - Ilustra o funcionamento do futebol de robô
Fonte: [COSTA, 2000]

Entre os desafios e problemas a serem estudados, Shen (1998) afirma que “jogadores robóticos precisam realizar processos de reconhecimento visuais em tempo real, navegar em um espaço dinâmico, rastrear objetos em movimento, colaborar com outros robôs e ter controle para acertar a bola na direção correta”. Para atingir esse objetivo os robôs devem ser autônomos, eficientes, cooperativos, com capacidades de planejamento, raciocínio e aprendizado, além de atuarem sob restrições de tempo real.

As partidas de futebol de robôs são dirigidas por um juiz (humano). Após o apito de início da partida, é proibida qualquer forma de intervenção humana enquanto os robôs estiverem jogando. Exceto em caso de falta, pênalti, escanteio e gol ou mesmo reiniciar a partida quando ordenado pelo juiz.

Um time de futebol de robôs é comandado por um sistema computacional (*software*) composto por: sistema de visão computacional, sistema estrategista, sistema de comando e comunicação e sistema de comando dos robôs.

2.2.1 Sistema de visão computacional

O sistema de visão computacional é responsável por obter a imagem do campo, a partir da imagem adquirida da câmera disposta sobre o campo de jogo, a posição dos robôs e a posição da bola em coordenadas no plano (X,Y).

Os elementos principais da visão computacional consistem em: aquisição da imagem, calibração do sistema, rastreamento das cores dos objetos no espaço de cores e localização dos robôs e da bola no campo.

No caso dos robôs do time, o algoritmo de visão determinará também a orientação e identificação do jogador, por meio do reconhecimento de identificadores coloridos colocados sobre o robô. A realização da identificação de cada robô torna o sistema independente da seqüência de resultados do rastreamento dos objetos nas imagens. Assim, se um objeto não for encontrado em uma imagem, isso não afeta o resultado para a imagem seguinte. Além disso, não usar rastreamento elimina problemas como a troca de identificação dos objetos quando esses se encontram ou cruzam suas trajetórias.

O reconhecimento dos objetos é baseado nas cores da imagem capturada pela câmera, em que um conjunto de *pixels* adjacentes de cor laranja identifica a bola; de

cor azul, os robôs de um time e de cor amarela, os robôs de outro time. Para completar a identificação, localização e determinação da orientação dos robôs, além de etiqueta da cor do time, a cor rosa foi utilizada, ambas margeadas por uma moldura preta, conforme mostrado na figura 3. A cor rosa foi escolhida por não ser uma das cores reservadas na categoria MiroSot (verde escuro, branca, preta, laranja, amarela e azul) e por apresentar razoável separabilidade no espaço de cores restantes.

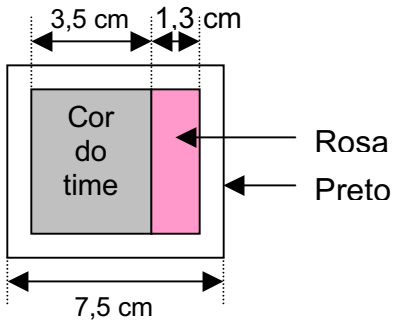

Figura 3 - Cores para identificação dos robôs.
Fonte: [COSTA, 2000].

A finalidade da borda preta usada nas etiquetas do robô é para evitar que duas etiquetas de uma mesma cor, porém colocadas em dois robôs distintos que estejam em contato, possam ser identificadas como um único elemento, evitando um indesejado alinhamento acidental.

O ciclo total de controle do sistema foi definido pela taxa de aquisição de imagens, é de 30 quadros por segundo, ou seja, um quadro a cada 33 ms (microsegundos). Assim em período de 33 ms, uma nova imagem refletindo o estado atual do campo torna-se disponível ao computador para processamento.

O módulo de visão computacional possui uma fase inicial (*off-line*) de calibração, executada previamente ao início da partida. Durante a partida (fase *on-line*), a visão identifica, localiza e rastreia a bola e os jogadores no campo.

A calibração é feita antes do início de uma partida, as equipes devem preparar os times, instalando equipamentos e calibrando o módulo de visão computacional. Nessa fase de calibração, a visão computacional é preparada para adaptar-se à iluminação do ambiente, reconhecer cores, localizar objetos, relacionar o espaço-imagem (*pixel*) com o espaço-campo (centímetros), definir cores do time e do adversário, identificar campos de ataque e de defesa, etc.

2.2.2 Sistema Estrategista

O algoritmo responsável pelo sistema estrategista recebe as informações do sistema de visão computacional e as traduz em decisões que resultam em ações que os robôs deveram executar, tais como: ataque, defesa, etc. Ou seja, como resultado final dessa etapa tem-se as informações de velocidade e ângulo de deslocamentos necessários para cumprir as ações solicitadas.

2.2.3 Sistema de comando e comunicação

Esse módulo é responsável por calcular as velocidades e ângulos de deslocamento dos motores dos robôs. O resultado é transmitido para o robô via transmissor de rádio acoplado à porta serial do microcomputador.

2.2.4 Sistema de comando dos robôs

O receptor capta os sinais, vindo do microcomputador, contendo as informações necessárias para o deslocamento dos robôs. Com esses dados o sistema de comando interpreta o quanto cada motor deverá se deslocar e em que sentido. Na figura 4 é mostrado um esquema de recepção do robô.

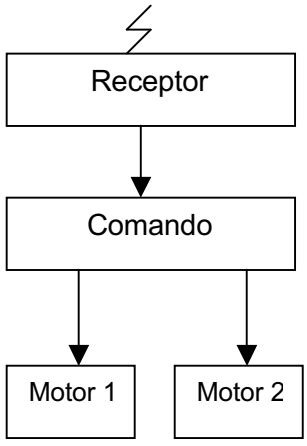

Figura 4 - Esquema de recepção do robô.

3 PROJETO

O objetivo do projeto é estudar os diversos assuntos relacionados às áreas de pesquisa que envolvem futebol de robôs e despertar nos acadêmicos da Faculdade de Informática de Presidente Prudente (FIPP) o interesse em realizar pesquisas nessa área da robótica.

O robô construído tem como principais atrativos o transceptor de ondas de rádio usado para receber os comandos de movimentação provenientes do computador controlador e o microcontrolador PIC com uma programação exclusiva para o robô.

Foi desenvolvido um *software* com o modulo de visão computacional, para reconhecimento do robô e definir a sua localização. Nesse módulo foram definidas rotinas para captura e reconhecimento de imagens, podendo ser utilizadas para outros objetos como outros robôs e a bola.

O campo de futebol foi projetado com dimensões de 1,30m x 0,90m, para que seja usado futuramente em outros projetos para aprimorar e inovar as investigações já realizadas.

3.1 Robô

Inicialmente foi construído um protótipo do robô com microcontrolador PIC ligado diretamente à porta serial. Com esse protótipo, foram realizados testes de comunicação com o computador e testes para rotacionar os motores do robô.

Principais características do protótipo:

- Microcontrolador PIC, responsável pela movimentação dos motores do robô, por meio de uma programação para recepção e interpretação dos dados recebidos pelo sistema controlador (computador). O microcontrolador utiliza 5 volts.
- Cristal de 4 Mhz, utilizado em conjunto com o microcontrolador PIC.
- Ponte H é constituída de transistores de potência para rotacionar os motores.

- O Max232 é ligado diretamente à porta serial e converte o padrão serial (+12v e –12v) para o microncontrolador PIC (0v e +5v).
- Regulador de tensão, usado para regular a alimentação de 5 volts para o cristal e microcontrolador PIC, e 9 volts utilizados no circuito para acionamento dos motores.
- Acoplador ótico isola a parte de potência (9V) onde estão os motores da parte lógica (5V) onde está o PIC.

Após o teste no protótipo foi projetado e construído um robô artesanalmente, que possui uma dimensão de 8 x 8 cm, tendo como diferença do protótipo dois motores de 9 volts de corrente contínua (C.C.), uma bateria de alimentação e o transceptor para comunicação via ondas de rádio do computador com o robô, eliminando a utilização do Max232. A figura 5 mostra uma imagem do robô montado com todos os componentes eletrônicos.

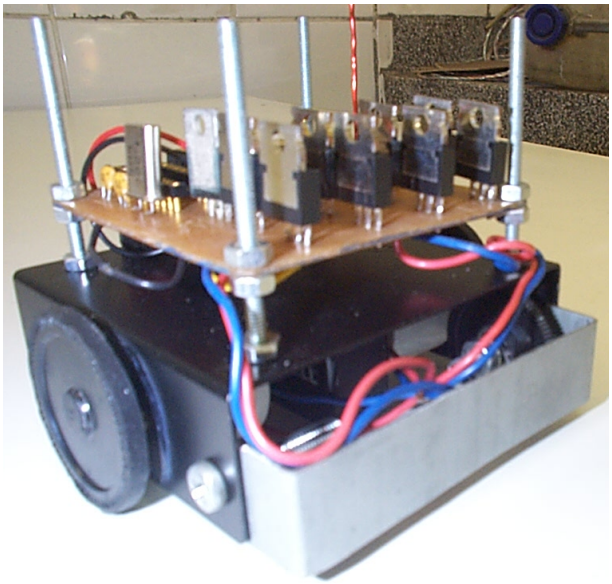

Figura 5 - Imagem do robô.

A figura 6 demonstra o circuito eletrônico do robô. Nessa, são indicados os principais componentes que compõem o circuito, tais como: transceptor (responsável por

recepção dos comandos via ondas de rádio); microcontrolador PIC e circuito para acionamento dos motores.

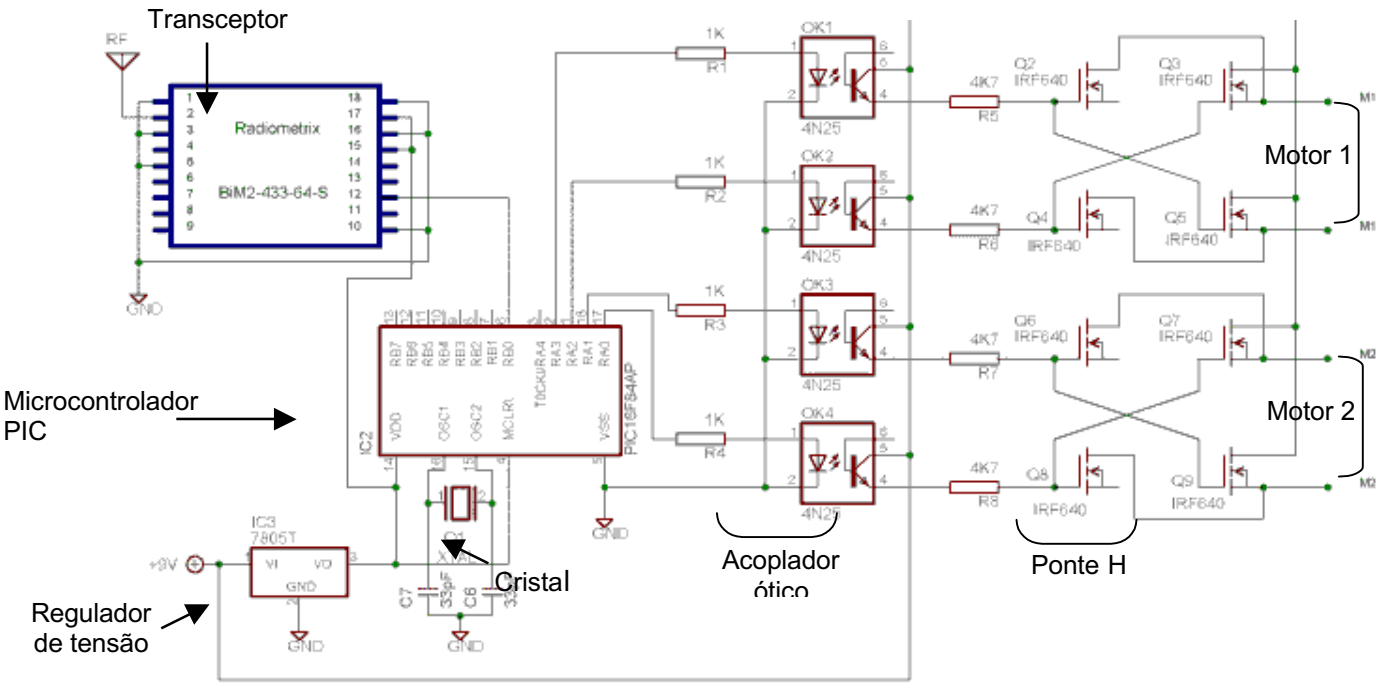

Figura 6 - Circuito eletrônico do robô.

O transceptor Radiometrix pertencente ao circuito do robô, ele capta os sinais de rádio provenientes do circuito acoplado à porta serial do computador controlador e repassa para o microcontrolador PIC, que controla os motores para realizar os movimentos do robô. A figura 7 mostra o circuito transmissor que é conectado à porta serial (RS232) do microcomputador. O transceptor usado é Radiometrix Bim2, opera sobre a faixa de 433,92 Mhz, é capaz de atingir 200 metros em área livre e 50 metros em área de construção. Em média, sua taxa de transmissão é de 160kbit/s [RadioMetrix, 2003].

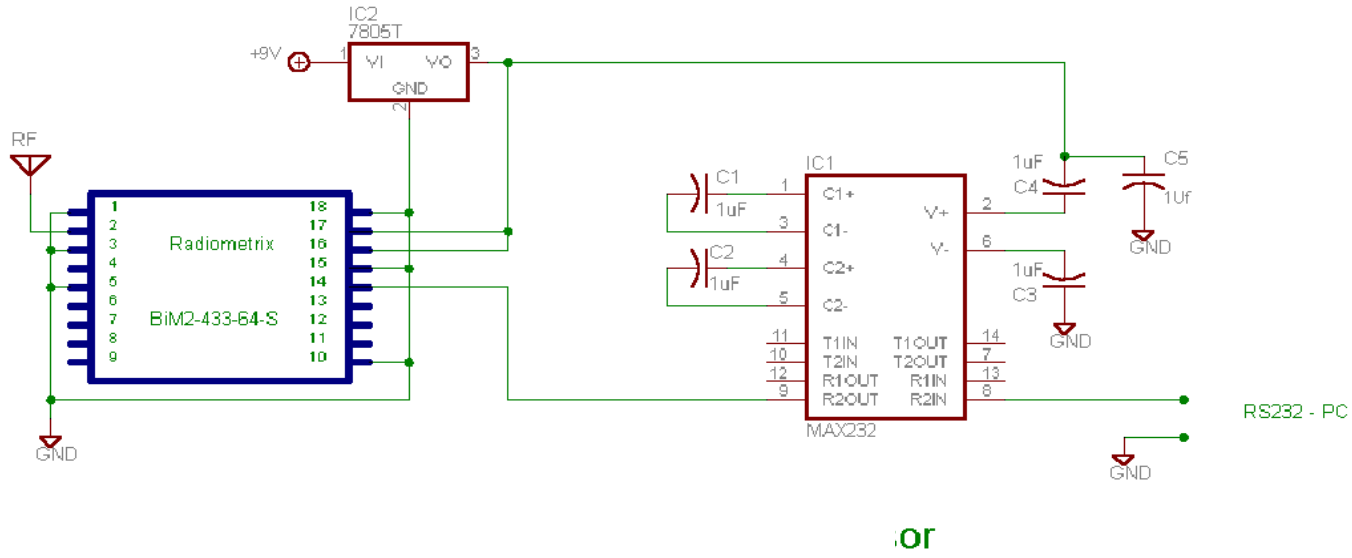

Figura 7 - Circuito eletrônico responsável por enviar dados ao robô.

A figura 8 é a imagem do circuito acoplado ao computador para transmissão de dados para o robô.

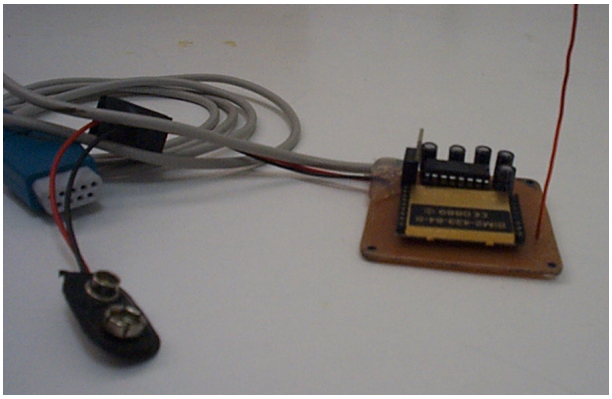

Figura 8 – Imagem do circuito transmissor montado.

3.2 Software

Para o protótipo foi desenvolvido um *software* em Borland C++ Builder, utilizando as bibliotecas de comunicação serial. Na figura 9, é mostrado o aplicativo desenvolvido para comunicação com o protótipo.

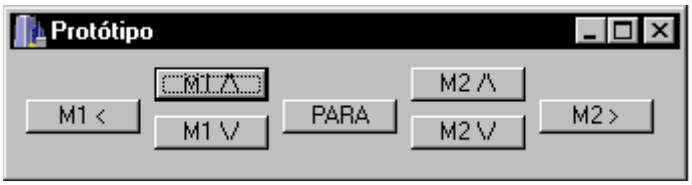

Figura 9 – Aplicativo desenvolvido para teste de comunicação.

Os dados enviados para a porta serial, são dados numéricos de 0 a 8 e no microcontrolador PIC foi feito um programa para coordenar a rotação dos motores conforme o número recebido. A função utilizada para o envio dos dados para a porta serial foi a TransmitCommChar(<configurações da porta serial>, <dados a serem enviados>).

Nos testes feitos foram utilizados como configurações:

- a porta COM1
- taxa de transferência de 4800 bits por segundos
- sem paridade no pacote de transmissão
- pacote de dados de 8 bits

A figura 10 mostra o código fonte do *software* de comunicação demonstrado na figura 9.

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"


// GLOBAL VARIABLES

HANDLE hComm = NULL;
COMMTIMEOUTS ctmoNew = {0}, ctmoOld;

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  DCB dcbCommPort;

  // OPEN THE COMM PORT.
  // REPLACE "COM2" WITH A STRING OR "COM1", "COM3", ETC. TO OPEN
  // ANOTHER PORT.

  hComm = CreateFile("COM1",
                      GENERIC_READ | GENERIC_WRITE,
                      0,
                      0,
                      OPEN_EXISTING,
                      0,
                      0);

  // IF THE PORT CANNOT BE OPENED, BAIL OUT.

  if(hComm == INVALID_HANDLE_VALUE) Application->Terminate();

  // SET THE COMM TIMEOUTS IN OUR EXAMPLE.

  GetCommTimeouts(hComm,&ctmoOld);
  ctmoNew.ReadTotalTimeoutConstant = 100;
  ctmoNew.ReadTotalTimeoutMultiplier = 0;
  ctmoNew.WriteTotalTimeoutMultiplier = 0;
  ctmoNew.WriteTotalTimeoutConstant = 0;
  SetCommTimeouts(hComm, &ctmoNew);

  // SET BAUD RATE, PARITY, WORD SIZE, AND STOP BITS.
  // THERE ARE OTHER WAYS OF DOING SETTING THESE BUT THIS IS THE EASIEST.
  // IF YOU WANT TO LATER ADD CODE FOR OTHER BAUD RATES, REMEMBER
  // THAT THE ARGUMENT FOR BuildCommDCB MUST BE A POINTER TO A STRING.
  // ALSO NOTE THAT BuildCommDCB() DEFAULTS TO NO HANDSHAKING.

  dcbCommPort.DCBlength = sizeof(DCB);
  GetCommState(hComm, &dcbCommPort);
  BuildCommDCB("4800,N,8,2", &dcbCommPort);
  SetCommState(hComm, &dcbCommPort);

  // ACTIVATE THE THREAD. THE FALSE ARGUMENT SIMPLY MEANS IT HITS THE
  // GROUND RUNNING RATHER THAN SUSPENDED.

//  ReadThread = new TRead(false);

}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
  // TERMINATE THE THREAD.

//  ReadThread->Terminate();

  // WAIT FOR THREAD TO TERMINATE,
```

```
  // PURGE THE INTERNAL COMM BUFFER,
  // RESTORE THE PREVIOUS TIMEOUT SETTINGS,
  // AND CLOSE THE COMM PORT.

  Sleep(250);
  PurgeComm(hComm, PURGE_RXABORT);
  SetCommTimeouts(hComm, &ctmoOld);
  CloseHandle(hComm);

}

//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TransmitCommChar(hComm, 1);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  TransmitCommChar(hComm, 2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  TransmitCommChar(hComm, 3);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
  TransmitCommChar(hComm, 4);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{
  TransmitCommChar(hComm, 5);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
{
  TransmitCommChar(hComm, 6);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button9Click(TObject *Sender)
{
  TransmitCommChar(hComm, 8);
}
//---------------------------------------------------------------------------
```

Figura 10 - Código fonte do *software* de comunicação.

No módulo de visão computacional, inicialmente foi utilizado o componente TvideoControl, no ambiente de programação Borland C++ Builder 5, mas o componente

passou a apresentar uma captura de quadros por segundo abaixo do esperado, além de poucas opções para sua utilização.

Posteriormente, passou-se a utilizar o componente Videocap para Borland Delphi e, conseqüentemente, houve a mudança de ambiente de programação de Borland C++ Builder 5 para Borland Delphi 5. Na figura 11 é mostrado o *software* de visão computacional.

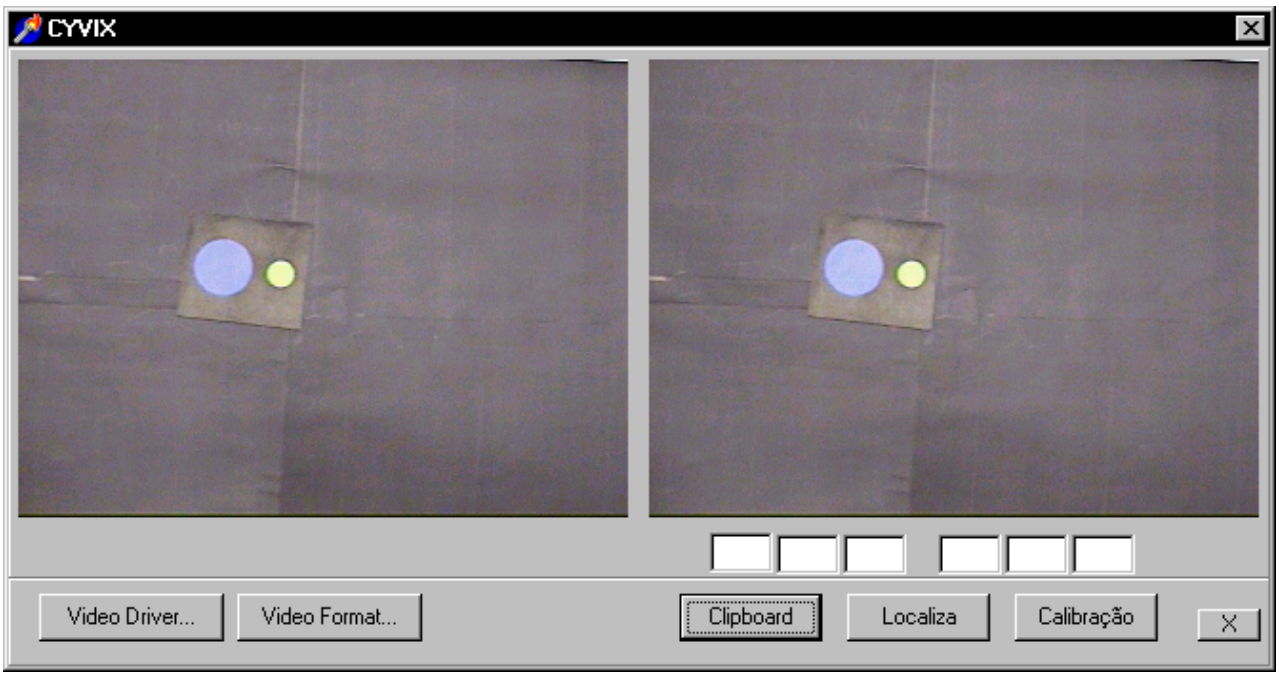

Figura 11 - *Software* de visão computacional.

Foram desenvolvidos métodos para varredura e reconhecimento dos objetos, que durante os testes foi utilizado um quadrado nas dimensões do robô, com dois círculos, simulando a superfície dele. Foram utilizados dois círculos de tamanhos diferentes para definir a frente do robô e as cores utilizadas foram azul do círculo maior simulando a cor de identificação de um time e a cor verde no círculo menor simulando o número de um dos robôs jogadores do time azul.

Ao iniciar o *software* de visão computacional é necessário fazer a calibração das cores. O método de calibração é responsável para definição da faixa de cores do objeto que o usuário selecionou. Como por exemplo, ao acionar o botão de calibração, o usuário é informado para selecionar a área onde se encontra o círculo azul;

o sistema faz uma varredura nessa área calculando a média e a diferença encontrada na área correspondende aos pontos azuis. Após isso é requerido ao usuário para fazer o mesmo com o círculo verde e com a área preta, onde o sistema faz o mesmo processo. Após isso é calculado o raio de distância entre o centro dos dois círculos.

Ao acionar o botão localiza, é feita uma varredura na imagem capturada para localizar um ponto azul, após isso, é calculado o centro do círculo azul.

As principais funções utilizadas no *software* de visão computacional são:

- CalculaRGB: utilizado na calibração. É passada a área que o usuário selecionou e é feita a varredura nessa área para definir o maior e o menor índice encontrado RGB (red=vermelho, green=verde, blue=azul), para definir a média e a diferença encontrada.
- VerificaCor: utilizado para comparar se o RGB do *pixel* está dentro da faixa correspondente. Ambos são passados como parâmetro de entrada da função, retornando verdadeiro ou falso.
- AchaCentroCírculo: é passado como parâmetro de entrada X e Y, e a cor que não identifica o círculo, essa função faz uma varredura para encontrar o início e o fim do círculo na horizontal e vertical, e retorna como parâmetro os pontos X e Y correspondentes ao centro do círculo.
- AchaCírculoRotacao: o parâmetro de entrada é o ponto X, Y (corresponde ao centro do círculo azul no teste) e a cor que deverá ser encontrada (o círculo verde por exemplo). Esta função procura o centro do círculo verde, por meio de varreduras circulares em torno do ponto X e Y. A função retorna o primeiro ponto (X,Y) encontrado no círculo verde.

3.3 Campo de Futebol

O campo foi adaptado para o projeto com dimensões menores do que o sugerido pela MiroSot. As dimensões do campo são 1,30m X 0,90 m. Exemplo do campo na figura 12 e na figura 13 uma visão da lateral do campo.

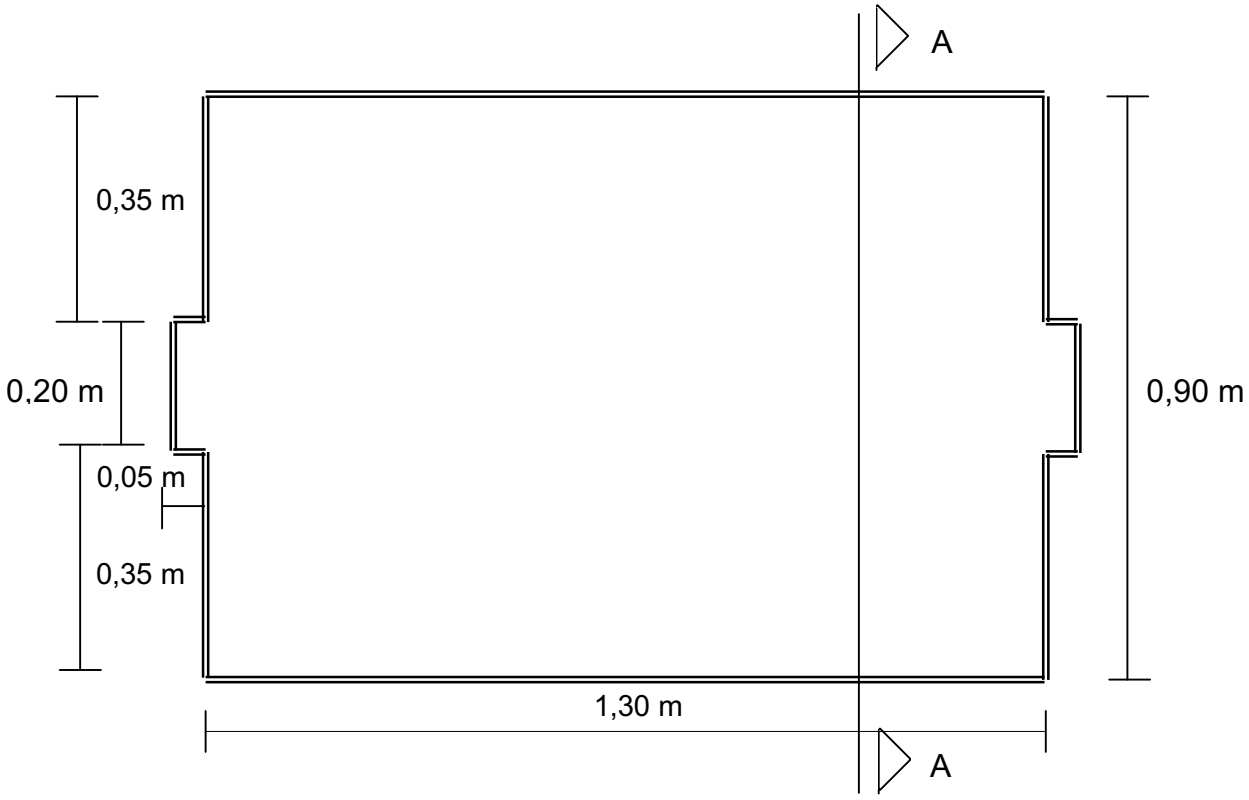

Figura 12 - Campo de futebol.

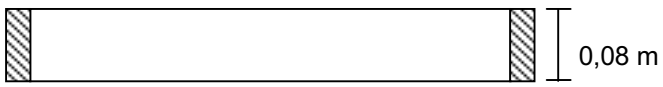

Figura 13 - Lateral do campo de futebol.

4 CONCEITOS

Neste capítulo, são apresentados os conceitos básicos para uma melhor compreensão dos assuntos abordados. Inicialmente, são descritas informações relevantes a respeito de porta serial do microcomputador; posteriormente, as características do microcontrolador PIC 16F84A.

4.1 Porta Serial do Microcomputador

O padrão para comunicação serial será o RS-232 encontrado nas portas seriais do microcomputador. Esse termo foi desiguinado pela *Eletronics Industries Association* (Associação das Indústrias Eletrônicas) e conseqüentemente, definiu padrões para a sua utilização, que nem sempre são usados.

A comunicação serial pode ser:

-síncrona: existe um canal de transmissão de dados e um canal de sincronismo, usado para mostrar ao receptor onde começa e onde termina cada conjunto de dados que está sendo transmitido pelo canal de dados;

-assíncrona: o mesmo canal que transmite os dados é também utilizado para a transmissão de sinais para identificar o início e o término do conjunto de dados.

A transmissão serial é feita *bit* a *bit* mas, para conseguir uma sincronia e evitar em caso de uma falha ao enviar todo o conteúdo novamente, os dados são enviados em grupos de oito bits, em que primeiro são enviados os bits menos significativos (de menor peso) e no final os mais significativos (de maior peso).

Utiliza-se para isso o *start bit* para iniciar um grupo e o *stop bit* para encerrar. Para efetuar a transmissão o processador envia oito bits para o endereço 3F8h e a porta transmite um por um. Ao terminar a transmissão de um grupo o receptor envia um sinal para a porta transmissora que recebeu os dados. A figura 14 mostra um exemplo de transmissão e a figura 15 mostra um exemplo da situação do cabo de transmissão.

Quando a porta serial está ociosa, sua saída está em nível lógico 1. Ao receber nível lógico 0, indica o *start bit*, e inicia-se a contagem para receber um grupo de oito bits.O *stop bit* equivale ao nível lógico 1 novamente encerrando o grupo.

O bit de paridade é opcional e é usado para checar os dados recebidos, evitando possíveis erros que possam ocorrer na transmissão dos dados.

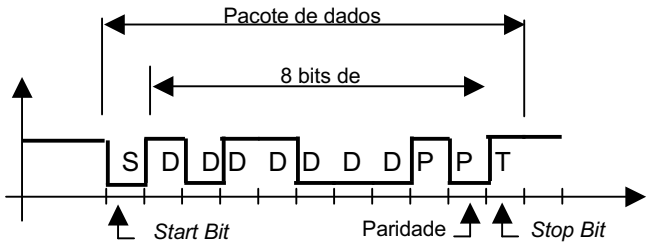

Figura 14 - Exemplo de transmissão da porta serial.

No caso do cabo para transmissão, o sinal 0 equivale a +12 V, e 1 a -12 V.

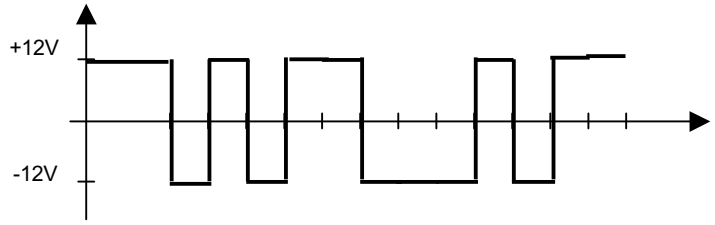

Figura 15 - Exemplo da situação do cabo de transmissão.

4.2 Conjunto de Caracteres ASCII

Os caracteres enviados por meio de uma interface serial geralmente seguem o padrão ASCII (American Standard Code for Information Interchange) de sete bits. A tabela 1 demonstra os caracteres ASCII usados na transmissão de dados.

Tabela 1 - Tabela de caracteres ASCII usados na transmissão.

HEX	DEC	CHR	HEX	DEC	CHR	HEX	DEC	CHR	HEX	DEC	CHR
00	00	NUL	20	32	SPC	40	64	@	60	96	`
01	01	SOH	21	33	!	41	65	A	61	97	A
02	02	STX	22	34	“	42	66	B	62	98	B
03	03	ETX	23	35	#	43	67	C	63	99	C
04	04	EOT	24	36	$	44	68	D	64	100	D
05	05	ENQ	25	37	%	45	69	E	65	101	E
06	06	ACK	26	38	&	46	70	F	66	102	F
07	07	BEL	27	39	‘	47	71	G	67	103	G
08	08	BS	28	40	(	48	72	H	68	104	H
09	09	HT	29	41	)	49	73	I	69	105	I
0A	10	LF	2A	42	*	4A	74	J	6A	106	J
0B	11	VT	2B	43	+	4B	75	K	6B	107	K
0C	12	FF	2C	44	,	4C	76	L	6C	108	L
0D	13	CR	2D	45	-	4D	77	M	6D	109	M
0E	14	SO	2E	46	.	4E	78	N	6E	110	N
0F	15	SI	2F	47	/	4F	79	O	6F	111	O
10	16	DLE	30	48	0	50	80	P	70	112	P
11	17	DC1	31	49	1	51	81	Q	71	113	Q
12	18	DC2	32	50	2	52	82	R	72	114	R
13	19	DC3	33	51	3	53	83	S	73	115	S
14	20	DC4	34	52	4	54	84	T	74	116	T
15	21	NAK	35	53	5	55	85	U	75	117	U
16	22	SYN	36	54	6	56	86	V	76	118	V
17	23	ETB	37	55	7	57	87	W	77	119	W
18	24	CAN	38	56	8	58	88	X	78	120	X
19	25	EM	39	57	9	59	89	Y	79	121	Y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	Z
1B	27	ESC	3B	59	;	5B	91	|	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	|	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	-	7F	127	DEL

Os caracteres não imprimíveis (00 a 31) são utilizados por diversos protocolos de comunicação. A nomenclatura dada para cada um deles é demonstrada na tabela 2:

Tabela 2 - Caracteres não imprimíveis.

DEC	CHR	CTRL+X	NOME
1	NUL	CTRL+@	Null
2	SOH	CTRL+A	Start of Heading
3	STX	CTRL+B	Start of Text
4	ETX	CTRL+C	End of Text
5	EOT	CTRL+D	End of Transmittion
6	ENQ	CTRL+E	Enquiry
7	ACK	CTRL+F	Acknowledge
8	BEL	CTRL+G	Bell
9	BS	CTRL+H	Backspace
10	HT	CTRL+I	Horizontal Tab
11	LF	CTRL+J	Line Feed
12	VT	CTRL+K	Vertical Tab
13	FF	CTRL+L	Form Feed
14	CR	CTRL+M	Carriage Return
15	SO	CTRL+N	Shift Out
16	SI	CTRL+O	Shift In
17	DLE	CTRL+P	Data Line Escape
18	DC1	CTRL+Q	Device Control 1
19	DC2	CTRL+R	Device Control 2
20	DC3	CTRL+S	Device Control 3
21	DC4	CTRL+T	Device Control 4
22	NAK	CTRL+U	Not Acknowledge
23	SYN	CTRL+V	Syncronous
24	ETB	CTRL+W	End of Transmition Block
25	CAN	CTRL+X	Cancel
26	EM	CTRL+Y	End of Médium
27	SUB	CTRL+Z	Substitute
28	ESC	CTRL+|	Escape
29	FS	CTRL+\	File Separator
30	GS	CTRL+|	Group Separator
31	RS	CTRL+^	Record Separator
32	US	CTRL+_	Unit Separator

A velocidade de transmissão pode ser de 300 bps a 115.200 bps. O endereçamento das portas seriais nos microcomputadores é mostrado na tabela 3.

Tabela 3 - Tabela de endereçamento da porta serial.

Porta serial	Endereço utilizado	Interrupção
COM1	3F8H	IRQ4
COM2	3F8H	IRQ3
COM3	3E8H	IRQ4
COM4	2E8H	IRQ3

A figura 16 mostra uma ilustração da porta serial com o conector padrão DB9. Na tabela 4 são demonstradas as identificações dos pinos.

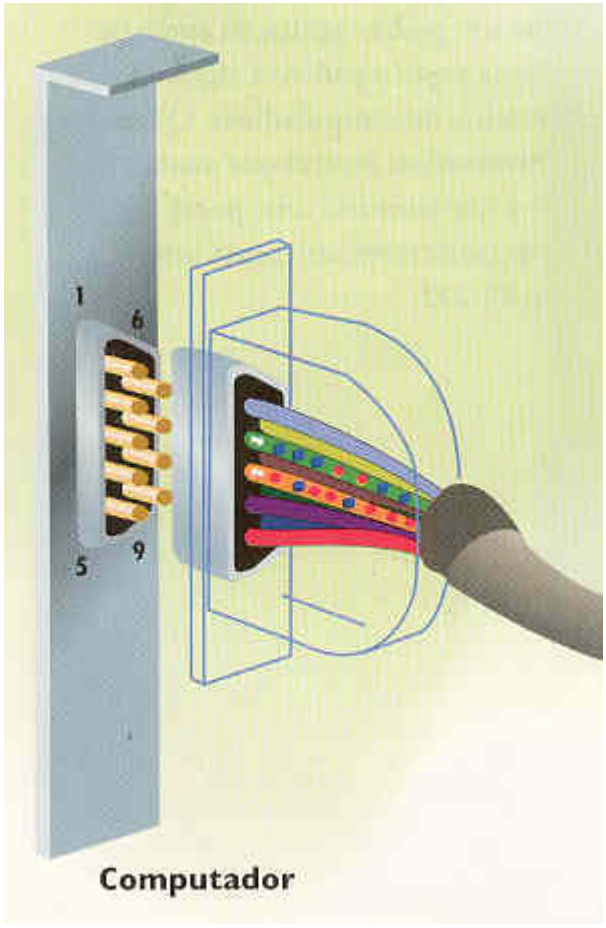

Figura 16 - Porta serial DB9.
Fonte: [WHITE, 1995]

Tabela 4 - Identificação dos pinos da porta serial.

Pinos	Identificação
1	Signal Ground
2	DTE Ready
3	Transmitted Data
4	Received Data
5	Received Line Signal Detect
6	Ring Indicator
7	Clear to Send
8	Request to Send
9	DCE Ready

4.3 Microcontrolador PIC

Os microcontroladores PIC apresentam uma otimização de recursos eletrônicos e uma boa relação custo/beneficio. Os microcontroladores PIC podem ser encontrados em diversos aparelhos, como: televisão, aparelho de som, teclados de computadores, celulares, *mouses*, refrigeradores, carros, brinquedos, sistemas de alarme, etc.

Os microcontroladores PIC possuem diversos periféricos, tais como: CPU (central processing Unit), ALU (arithmetic logical unit), linhas para dados, linhas para endereços e linhas para controle e acesso de memória externa. Também há periféricos como comunicação serial, *timers*, w*atchdog timer*, osciladores, I/O´s (portas para o acesso ao mundo exterior), etc.

O PIC 16F84 é fabricado pela Microchip, opera até 10 MHz (ciclo de instrução de 400 ns) e o PIC 16F84A pode operar até 20 MHz. Sua arquitetura é Risc (reduced instructions set computing) que é baseado na arquitetura Harvard, que separa a memória de dados da memória de programa tendo um caminho para memória de dado e outro para memória de programa. Isso permite que o microcontrolador carregue na CPU um *byte* que contém toda uma linha de programa em um único ciclo de máquina, aumentando a velocidade do *chip*.

Principais características:

- 1 KByte (1024) palavras de 14 bits para programa;
- 68 *bytes* de RAM para uso geral;
- *stack* com oito níveis;
- apenas 35 instruções;
- 15 registros específicos em RAM para controle do *chip* e seus periféricos;
- *timer* de 8 *bits* com opção de *prescaler* de 8 *bits*;
- 13 pinos que podem ser configurados individualmente como entrada e saída;
- alta capacidade de corrente nos pinos (podendo acender um *led*);
- capacidade de gerenciar interrupções (até cinco entradas), do *timer* e EEPROM;

- *watchdog* para recuperação e *reset* em caso de travas no *software*;
- memória de programa protegida contra cópias;
- modo *sleep* para economia de energia;
- várias opções de osciladores;
- faixa de alimentação de 2 a 6 volts – típico 5 volts
- consumo de controle:
    - < 2 mA a 5 volts a 4 MHz
    - 15 μA a 2 volts a 32 KHz
    - 2 μA a 2 volts em *stand by*

A figura 17 demonstra a pinagem e suas características elétricas básicas do PIC 16F84A.

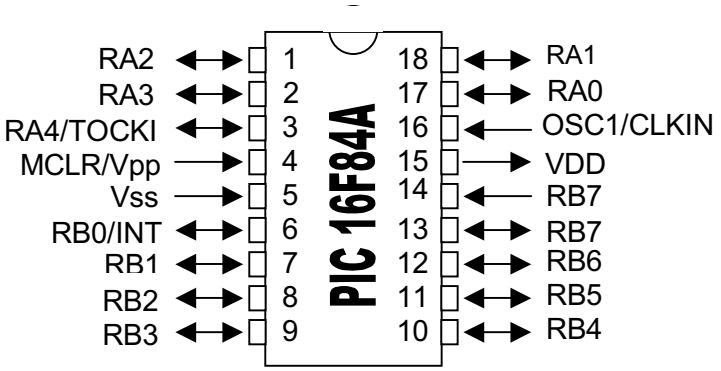

Figura 17 - Pinagem do PIC 16F84A.

A tabela 5 mostra a descrição do PIC 16F84A.

Tabela 5 - Descrição dos pinos do PIC 16F84A.

Nome	No.	Tipo	Estrutura	Descrição
MCLR / VPP Pino 4	4	Entrada/ Programação	Shimitt Trigger	Em condição normal de funcionamento desenvolve a função de Máster CLeaR ou seja *Reset* estará ativo a nível 0. Pode ser conectado a um circuito de *reset* externo ou simplesmente conectado-o ao positivo da alimentação. Quando o PIC vier posto em *Program Mode* será utilizado como entrada para a tensão de programação Vpp.
OSC1 / CLKOUT OSC2 / CLKIN	16 15	Entrada / Saída	Shimitt Trigger/ CMOS	É um pino de conexão no caso de se utilizar um cristal de *quartzo* para gerar o *clock*. E como saída de *clock* caso for aplicado um oscilador RC externo.
RA0 (Bit 0) RA1 (Bit 1) RA2 (Bit 2) RA3 (Bit 3)	17 18 1 2	Entrada/ Saída	TTL	É um pino de I/O programável em entrada ou saída. Trabalha com o Bit correspondente na Porta A.
RA4 / RTCC ou T0CKI	3	Entrada/ Saída	Shimitt Trigger	É um pino multi-função que pode ser programado como uma linha normal de I/O ou como linha de clock para entrada em sentido ao contador RTCC ou TMR0. Se programada como pino de I/O corresponde a o BIT 4 da Porta A ao contrario de outra linha de I/O. Quando esta funciona como saída, trabalha em coletor aberto.
RB0/Int (Bit 0)	6	Entrada/ Saída	TTL Shimitt Trigger	É um pino de I/O programável em entrada ou em saída. O port B possui *pull-ups* internos programáveis. Podendo ser programado para gerar interrupção.
RB1 (Bit 1) RB2 (Bit 2) RB3 (Bit 3)	7 8 9	Entrada/ Saída	TTL	É um pino de I/O programável em entrada ou em saída. Trabalha com o Bit correspondente na Porta B.
RB4 (Bit 4) RB5 (Bit 5)	10 11	Entrada/ Saída	TTL	É um pino de I/O programável em entrada ou em saída. Permite Interrupção se alterar o nivel Trabalha com o Bit correspondente na Porta B.
RB6 (Bit 6) RB7 (Bit 7)	12 13	Entrada/ Saída	TTL/Shimit t Trigger	É um pino de I/O programáve l em entrada ou em saída. Permite Interrupção se alterar o nivel Trabalha com o Bit correspondente na Porta B.
VSS	5	Alimentação		É um pino que vai conectado ao negativo da tensão de alimentação.
VDD	14	Alimentação		É um terminal positivo de alimentação do PIC. Em todas as três versões disponíveis do PIC16F84 (comercial, industrial e automotiva) a tensão pode assumir um valor que vai de 2 volts a 6 volts. Nomalmente +5V.

A figura 18 mostra a arquitetura interna do PIC 16F84A.

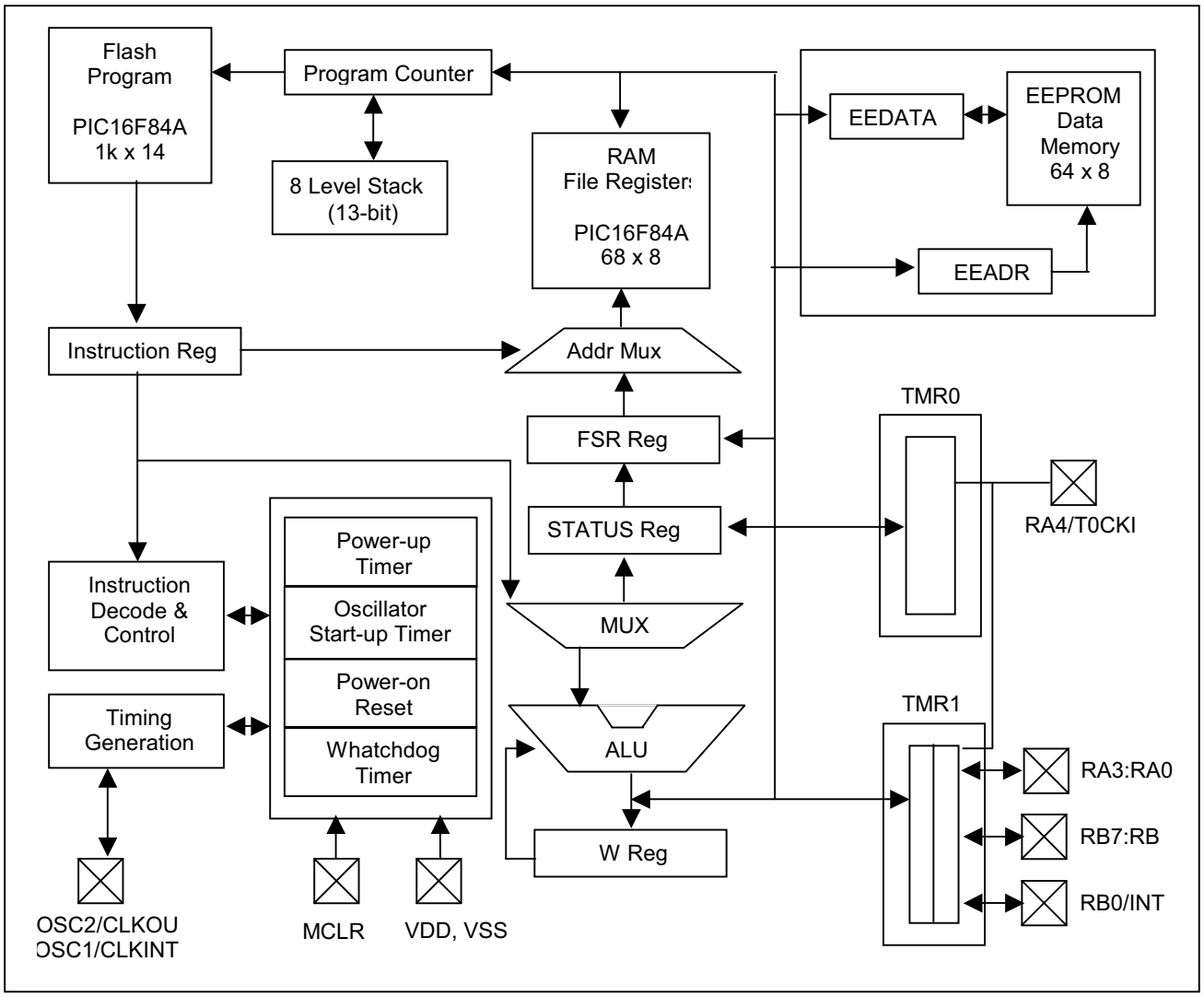

Figura 18 - Arquitetura interna do PIC.

A tabela 6 mostra a descrição dos registros do PIC 16F84A.

Tabela 6 - Tabela dos registros do PIC16F84A.

Nome	Banco	Endereço	Descrição
INDF	0/1	00h/80h	Não é um registro fisicamente implementado, ao acessa-lo na verdade esta acessando a posição indicada pelo registro FSR, que atua como um ponteiro para outras posições de memória.
TMR0	0	01h	Registro de contagem do timer 0.
PCL	0/1	02h/82h	Parte baixa do contador de programa .
STATUS	0/1	03h/83h	Configura os bancos de registros, flags da ULA e outros. Valor de *reset*: 00011XXX.
FSR	0/1	04h/84h	Ponteiro para o endereçamento indireto.
PORT A	0	05h	Registro dos pinos de I/O´s RA0 a RA4.
PORT B	0	06h	Registro dos pinos de I/O´s RB0 a RB7.
EEDATA	0	08h	Arnazena os 8 *bits* (*bytes*) para leitura ou escrita.
EEADR	0	09h	Armazena o endereço da EEPROM que será acessado.
PCLATH	0/1	0Ah/8Ah	Parte alto contador de programa.
INTCON	0/1	0Bh/8Bh	É para leitura e escrita, no qual se habili ta *bits* para selecionar todos os tipos de interrupção. Valor de *reset*: 0000000X.
OPTION	1	81h	Configura o *prescaler* de temporização, *timers* e outros. Valor de *reset*: 11111111.
TRIS A	1	85h	Direção dos pinos do PORT A.
TRIS B	1	86h	Direção dos pinos do PORT B.
EECON1	1	88h	Controle da EEPROM.
EECON2	1	89h	Controle da EEPROM.

4.3.1 Programação do PIC

O ambiente de programação que a *Microchip* (fabricante da família PIC16FXX) disponibiliza gratuitamente é o *MPLab*, que se utiliza da linguagem Assembly. Existem outros ambientes de programação para o PIC como por exemplo o CCS PICC, cuja a programação é desenvolvida em linguagem C. Esse já não é gratuito.

O PIC 16F84 possui um conjunto de 35 instruções em Assembly, divididas em três grupos: orientadas a *byte* (registradores), orientadas a *bit* e com constantes (literais) e de controle. A tabela 7 mostra a descrição do conjunto de instruções em Assembly.

Tabela 7 - Conjunto de instruções do PIC.

Intrução	Argumentos	Ciclos	Descrição
Operadores de *bytes* com registros			
ADDWF	f,d	1	Adcionar *w* e *f*.
ANDWF	f,d	1	Faz um *AND* lógico entre *w* e *f.*
CLRF	F	1	Faz todos os *bits* de *f* = zero.
CLRW		1	Faz todos os *bits* de *w* = zero.
COMF	f,d	1	Complementa *f* (inverte os *bits* de 0>1 ou 1>0)
DECF	f,d	1	Decrementa uma unidade em *f*.
DECFSZ	f,d	1 ou 2	Decrementa uma unidade em *f* e pula a próxima instrução se *f* = zero.
INCF	f,d	1	Incrementa uma unidade em *f*.
INCFSZ	f,d	1 ou 2	Incrementa uma unidade em *f* e pula a próxima instrução se *f* = zero.
IORWF	f,d	1	Faz um *OR* lógico entre *w* e *f.*
MOVF	f,d	1	Move *f*.
MOVWF	F	1	Move *w* para *f*.
NOP		1	Nenhuma operação.
RLF	f,d	1	Roda o *byte* à esquerda passando pelo *Carry*.
RRF	f,d	1	Roda o *byte* à direita passando pelo *Carry*.
SUBWF	f,d	1	subtrai *w* de *f*
SWAP	f,d	1	Troca *nibles* em *f*.
XORWF	f,d	1	Faz um *XOR* lógico entre *w* e *f.*
Operadores de *byts* com registros			
BCF	f,b	1	Zera o *bit b* em *f*.
BSF	f,b	1	Seta, (1) o *bit b* em *f*.
BTFSC	f,b	1 ou 2	Testa o *bit b* em *f* e pula a próxima instrução se o *bit* for zero.
BTFSS	f,b	1 ou 2	Testa o *bit b* em *f* e pula a próxima instrução se o *bit* for um.
Operadores com constantes e de controle			
ADDLW	K	1	Adiciona *w* em *k*.
ANDLW	K	1	*AND* lógico entre *w* em *k*.
CALL	K	2	Chama a sub rotina *k*.
CLRWDT		1	Zera o *timer* do *watchdog*.
GOTO	K	2	Vai para o *label k*.
IORLW	K	1	Faz um *OR* lógico entre *w* em *k*.
MOVLW	K	1	Move *l* para *w* (*w* = *k*).
RETFIE		2	Retorna da interrupção.
RETLW	K	2	Retorna com *w* = *k*.
RETURN		2	Retorna da subrotina.
SLEEP		1	Entra em modo *stand by*.
SUBLW	K	1	Subtrai *w* de *l*.
XORLW	K	1	Faz um *XOR* lógico.

4.4 Transceptor Radiometrix Bim2

O transceptor Radiometrix Bim2 é muito usado em comunicação do tipo *wireless* bidirecional em alta velocidade de transferência de dados em uma escala de 200 metros. O módulo opera sobre a faixa de 433,92 Mhz e permite a instalação conveniente do PWB [RADIOMETRIX, 2003].

Principais características:

- atinge 200 metros em área livre e 50 metros em áreas de construção;
- as taxas de dados chegam a 160kbit/s;
- transmissor FM controlado SAW 10mW;
- receptor superheteródico de FM;
- fonte 3V ou 5Volt abaixo de 20mA.

A figura 19 mostra a ilustração do modelo do Radiometrix, utilizado no projeto.

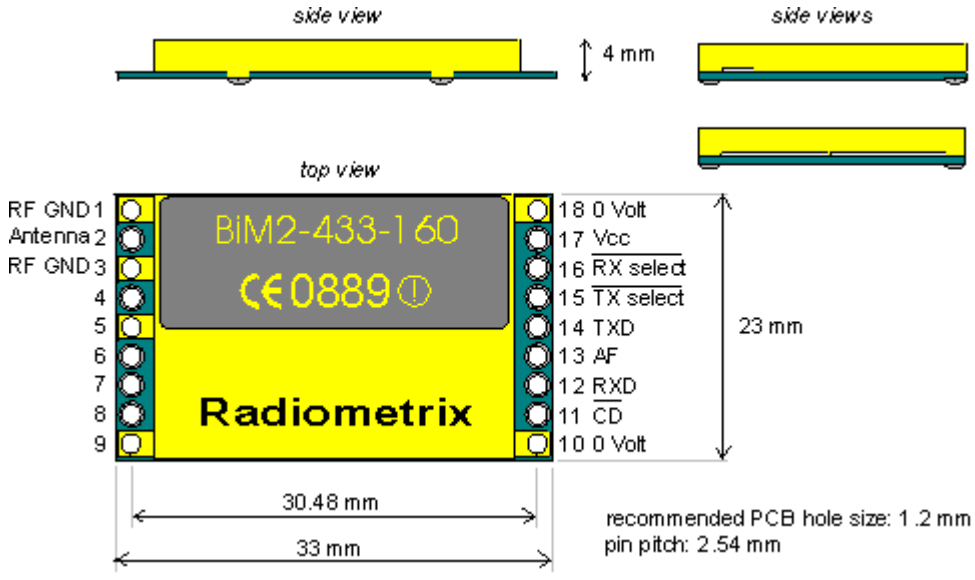

Figura 19 - Modelo do Radiometrix.
Fonte: [RADIOMETRIX, 2003].

Descrição dos pinos:

Pino 1 & 3 **terra do RF**: pino terra do RF, conectado internamente aos pinos 5, 9, 10, 18 (0 volts). Esse pino deve ser conectado ao retorno do RF.

Pino 2 **antena:** entrada de RF, 50Ω, isolada D.C. internamente.

Pino 11 **CD:** detector de portara – quando o receptor esta habilitado, um sinal de nível baixo acima do ponto inicial esta sento recebido. A saída tem alta impedância (50Ω) e deve somente ser usada com lógica CMOS.

Pino 12 **RXD:** saída digital proveniente de um *data slicer* interno, é uma versão quadrada do sinal presente no pino 13. pode ser ligada a decodificadores externos. A impedância da carga deve ser >1kΩ e <1nF

Pino 13 **AF:** essa é uma saída análoga protegida e filtrada do demodulador de FM. Tem uma polarização D.C. do sinal banda base de 1.2 volts e 400mV $_{P-P}$. É útil como um ponto de teste ou para ligar decodificadores lineares. A impedância da carga deve ser > 2kΩ e < 100pF.

Pino 14 **TXD:** esta entrada de modulação acoplada D.C., aceitará qualquer série de dado digital ou sinais lineares de nível elevado. A impedância de entrada é 10kΩ (de nível 0v a Vcc).

Pino 15 **TX:** quando em nível baixo, transmissor habilitado.

Pino 16 **RX:** quando em nível baixo, receptor habilitado.

A tabela 8 mostra a descrição dos pinos do Radiometrix.

Tabela 8 - Descrição do pinos do Radiometrix.
Fonte: [RADIOMETRIX, 2003].

Pn 15 TX	Pin16 RX	Função
1	1	Sistema inativo (< 1μA)
1	0	receptor habilitado
0	1	Transmissor habilitado
0	0	Auto teste *loop back*

5 CONCLUSÕES

Esta monografia foi motivada pela difusão cada vez maior da área de futebol de robôs nas universidades. Acredita-se que futebol de robôs é uma importante área da robótica e de grande valia para o meio acadêmico de computação. Partindo desse pressuposto, a pesquisa e o desenvolvimento de futebol entre robôs, contribui para despertar o surgimento do espírito da pesquisa cientifica e tecnológica.

Neste trabalho foram abordados os princípios básicos e alguns conceitos para iniciar um projeto de futebol de robôs. Como proposto inicialmente, desenvolveu-se um robô jogador e um *software* controlador. O robô foi construído utilizando um microcontrolador PIC, programado com um *software* para controlar as possíveis rotações dos motores, motores estes que produzem movimentos de deslocamento do robô. Um outro dispositivo utilizado no robô que merece destaque devido a sua complexidade é o transceptor, responsável pela transmissão dos dados via ondas de rádio do computador controlador para o robô. O *software* foi desenvolvido com mecanismos de visão computacional para reconhecimento do robô.

Estima-se que, com o estudo e resultados alcançados este trabalho venha a atingir as expectativas e os objetivos inicialmente idealizados para a iniciação de um projeto de futebol de robôs.

É de grande interesse que se tenha na Faculdade de Informática um time de robôs jogadores de futebol com possibilidades de participar de campeonatos com outras universidades, mas para que isso ocorra, há a necessidade da continuação deste projeto e de novas investigações no sentido de aperfeiçoar o robô e o *software* desenvolvido.

Pensando-se em novas pesquisas, há a necessidade de aperfeiçoamento da comunicação do robô jogador com o computador controlador. No que diz respeito ao *software* controlador precisa-se de novos mecanismos para comandos e comunicação, de um sistema estrategista provavelmente usando recursos da Inteligência Artificial, com isso, pré-programar jogadas e comportamentos mediante situações que possam surgir durante uma partida de futebol de robôs.

REFERÊNCIAS BIBLIOGRÁFICAS

BIANCHI, Reilnaldo A. C.; REALI-COSTA, Anna H.. **O Sistema de visão computacional do time Futepoli de futebol de robôs**. Escola Politécnica da Universidade de São Paulo, Laboratório de Técnicas Inteligentes, Departamento de Engenharia de Computação e Sistemas Digitais, 2000. Disponível em: **<http://www.lti.pcs.usp.br/~rbianchi/publications/CBA2000.pdf>**. Acesso em: 10 jan. 2003.

CANZIAN, Edmur. **MINICURSO Comunicação Serial – RS232**. CNZ Engenharia e Informática Ltda. E-mail: engenharia@cnz.com.br.

COSTA, Anna Helena Reali; PEGORARO, René Contruindo. **Robôs autônomos para partidas de futebol: o time Guaraná.** SBA Controle & Automação Vol. 11, no 03 / Set., Out., Nov., 2000
Disponível em: **<http://www.fee.unicamp.br/revista_sba/vol11/v11a259.pdf>**. Acesso em: 10 jan. 2003.

MAIA, Luiz Carlos Jr.; BIANCHI, Reinaldo A. C.. **Evolução de Agentes Jogadores de Futebol de Robôs**. Faculdade de Engenharia Industrial – FEI – Departamento de Engenharia Elétrica, 2000, Disponível em: **<http://www.lti.pcs.usp.br/~rbianchi/publications/ workcomp2000.pdf>**. Acesso em: 10 jan. 2003.

GUTIERRES, Adilson. **Treinamento em PIC Modulo 1 – Básico**. Bauru, Edutec Consultoria e treinamento S/C Ltda, 2000.

Manchester encoding using RS232 for Microchip PIC RF applications. Disponivel em **< http://www.quickbuilder.co.uk >**. Acesso em 20 de set. 2003.

RAMALHO, Luciano. **Vai que é tua Asimov!** Disponível em **<http://www.magnet.com.br/cosmo/copa_robos.html>**. acesso em 10 jan. 2003.

RadioMetrix BiM2-433 data sheet – BiM2-433-160; 2003. Disponível em: **<http://www.radiometrix.co.uk/products/bim2.htm>**. Acesso em: 2 Setembro 2003.

ROSH, Winn L. **Desvendando o hardware do PC**. Rio de Janeiro, 1990, 522 pg. Editora Campus.

TORRES, Gabriel; CAMELLO, Álvaro,. **Hardware, curso completo**. Rio de Janeiro, 1999, 1147 pg. Editora Axel Books do Brasil.

UFPR. **Futebol de robôs**. Curitiba, 2003. Disponível em **<http://pet.inf.ufpr.br/fut_robo>**. Acessado em 10 jan. 2003.

WHITE, Ron; DOWNS, Timothy Edward,. **Como funciona o computador**. São Paulo, 1995, 218 pg. Editora Quark.