Construção de Robôs Jogadores de Futebol
(2ª Parte)

Wânderson de Oliveira Assis, Alessandra Dutra Coelho, Marcelo Marques Gomes,
Cláudio Guércio Labate, Daniel Franklin Calasso,
João Carlos Gonçalves Conde Filho

Escola de Engenharia Mauá – Instituto Mauá de Tecnologia (IMT)
Praça Mauá 1 – CEP 09580-900 – São Caetano do Sul – SP - Brasil

1. Introdução

Uma das maiores dificuldades para as equipes de futebol de robôs é a de conseguir fazer com que os robôs apresentem na prática os resultados que seriam previstos ou desejados com a programação. O desafio é o de conseguir controlar de forma autônoma todos os movimentos dos robôs.

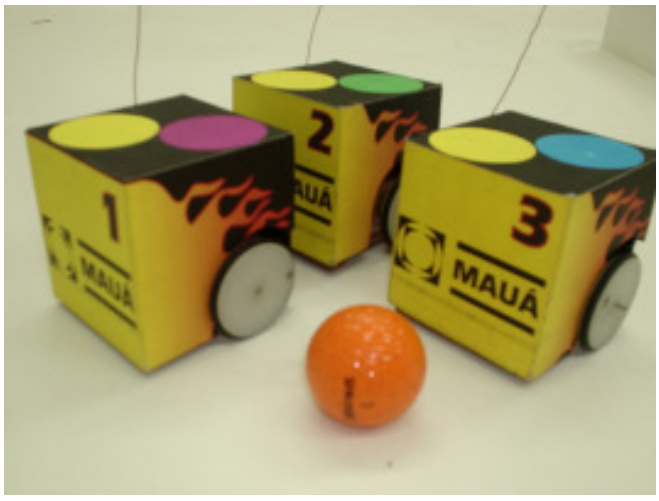

Figura 1 – Equipe de Futebol de Robôs da Mauá

Neste artigo apresenta-se a solução desenvolvida pela equipe de futebol de robôs da Escola de Engenharia Mauá para a programação dos robôs jogadores de futebol (Figura 1). Pretende-se demonstrar de forma clara e simples a programação dos

microcontroladores inseridos nos robôs a qual permite desenvolver a comunicação entre o PC e o robô por meio de radiofreqüência. Serão abordados os seguintes aspectos:

- desenvolvimento de um protocolo de comunicação para a transferência de dados;
- desenvolvimento de um algoritmo em assembly para o microcontrolador PIC, que permite efetuar a recepção serial de dados utilizando-se o protocolo, além da geração do sinal PWM para controle dos motores do robô.

2. Protocolo de Comunicação

Conforme se apresentou no primeiro artigo da série "Construção de Robôs Jogadores de Futebol", o computador deve monitorar continuamente as imagens captadas do campo, jogadores e bola e, de posse da localização de cada um dos elementos, enviar os sinais de controle para comandar cada um dos robôs da equipe. Para isso é necessário desenvolver um sistema de comunicação por meio de radiofreqüência (RF), por meio do qual o computador envia as mensagens e estas podem então ser recebidas pelo robô.

Para conseguir efetuar a transferência com maior velocidade, com qualidade e sem perda de dados deve-se definir um protocolo de comunicação. O protocolo de comunicação deve considerar:

- o número de pacotes de dados necessários para transmitir toda a mensagem;
- a utilização de padrões para identificar cada pacote de dados e permitir aos robôs decodificar a informação como, por exemplo, o destino da informação (robô 1, 2 ou 3);
- a máxima velocidade de transmissão admissível para os módulos de RF ;

a utilização de padrões e/ou filtros para garantir a qualidade da transmissão.

A transmissão da informação por ondas de rádio tem como vantagem a propagação em todas as direções e o grande alcance, porém está sujeito a interferências e ruídos. Uma das principais características a ser considerada na definição do protocolo de comunicação é a qualidade da transmissão. Para evitar o recebimento de mensagens com erros, o pacote deve apresentar alguns *bits* de controle que permitam a codificação

e o endereçamento da mensagem. Diante disso, o protocolo de comunicação utilizado na transmissão de dados do PC para os robôs é constituído de três *bytes* conforme se ilustra na Figura 2.

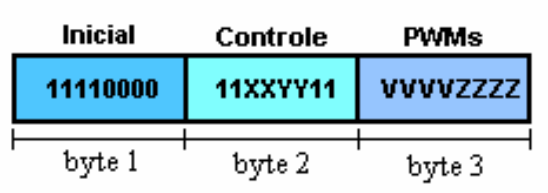

Figura 2 – Protocolo de Comunicação

O protocolo é o mais simples possível e cada um dos *bytes* tem a seguinte função:

- *Byte* Inicial – Armazena uma seqüência padronizada com valor binário dado por 11110000_2 (ou $F0_{16}$ no formato hexadecimal) para identificar o início de um pacote de dados. Este dado poderia ser repetido várias vezes para garantir uma "máscara" de controle e evitar que outros dados aleatórios pudessem ser entendidos como *byte* de início. Essa solução foi adotada inicialmente no projeto para minimizar os erros na transmissão. Contudo, na solução apresentada neste artigo, optou-se por uma configuração mais simplista que utiliza somente um *byte* inicial. Ainda assim, observou-se que, utilizando o módulo RF da Telecontrolli, o *byte* inicial seria perdido na transmissão. Isso ocorre porque, no início da transmissão de um pacote de dados, os módulos transmissor e receptor não estão sincronizados e produzem uma situação transitória. Por isso, o formato do primeiro *byte* no receptor não tem obrigatoriamente o mesmo formato que o dado transmitido. Exatamente por isso, o byte inicial foi utilizado somente na transmissão e foi simplesmente descartado na recepção.

- *Byte* de Controle – Armazena um pacote constituído de um "sanduíche" de 1's contém no interior uma "máscara" que identifica cada robô (código XX) e o sentido de cada motor (código YY) conforme detalhado na Tabela 1. Os dois *bits* mais significativos e os dois menos significativos devem ser obrigatoriamente 1. Portanto podem-se testar esses *bits* no microcontrolador para identificar a chegada do *byte* de controle.

XX	Identificação do Robô
00	Não utilizado
01	Robô 1
10	Robô 2
11	Robô 3

YY	Ação no Robô	Motor 1	Motor 2
00	Anda para frente	Frente	Frente
01	Gira no sentido horário	Frente	Trás
10	Gira no sentido anti-horário	Trás	Frente
11	Anda para trás	Trás	Trás

Tabela 1 – Códigos do Byte de Controle

- *Byte* PWMs – Apresenta dois números de quatro *em que* os quatro *bits* mais significativos são para o servomotor da esquerda (Motor 1) e os 4 bits menos significativos são para o servomotor da direita (Motor 2). Os valores transmitidos pelo computador para cada um dos motores correspondem à referência para o sinal PWM e, quanto maior o valor, maior será a velocidade do motor. Portanto, como temos quatro *bits* para cada servomotor, é possível obter-se 2^4 possíveis valores, que correspondem a 16 possíveis velocidades para cada servomotor (mais do que o suficiente para a aplicação).

Como se explicou, , o protocolo de comunicação é constituído de um pacote relativamente pequeno, o que simplifica os algoritmos de transmissão e recepção, além de tornar a transmissão mais rápida. Contudo o sistema fica bastante susceptível a interferências que possam ocorrer durante a transferência por RF. Por isso pode ser necessário enviar o mesmo pacote um maior número de vezes para garantir que os dados sejam recebidos nos robôs.

Alternativamente um protocolo mais elaborado e mais eficiente poderia ser desenvolvido para garantir melhor qualidade na transmissão. Algumas medidas poderiam ser utilizadas tais como a codificação de dados, utilização de mais dados de controles, introdução de um *preâmbulo* no início do pacote para sincronizar transmissor e receptor além da utilização de um *checksum* para fazer a verificação da informação. **Caso o leitor deseje desenvolver um algoritmo mais eficiente, pode consultar o artigo "Sistema Inteligente de Identificação *Smart Label* para Automação em Supermercados" publicado na revista ??????, edição do mês de ????, no qual todas essas medidas foram utilizadas para garantir um protocolo mais complexo, mas muito mais eficiente e imune a interferências.**

3. Programação do Microcontrolador

A utilização de um microcontrolador no robô é essencial para receber o sinal captado pelo módulo de RF, manipular os dados para identificar cada componente do protocolo e produzir sinais de controle para um módulo de potência (circuito integrado L298N) a fim de se definir a direção de acionamento e ajustar a velocidade de cada motor. Para desenvolver essas ações, a equipe de futebol de robôs da Mauá optou pela utilização do microcontrolador PIC16F628 da Microchip. Esse modelo permite comandar simultaneamente vários dispositivos periféricos, desenvolver a comunicação serial por meio de um pino de entrada, além de ter disponíveis temporizadores internos que podem ser manipulados por interrupções. Uma dessas interrupções (interrupção por *overflow* em Timer 0) será utilizada para produzir os sinais PWM para os servomotores.

Os sinais de controle utilizados ou produzidos pelo microcontrador estão listados na Tabela 2.

Porta	**Bit**	**Tipo**	**Função**
PORTB	1	Entrada	Recepção serial da USART
	3	Saída	Habilita sentido horário para motor 1
	4	Saída	PWM para motor 1
	5	Saída	Habilita sentido horário para motor 2
	6	Saída	Habilita sentido reverso para motor 2
	7	Saída	PWM para motor 2
PORTA	1	Saída	Habilita sentido reverso para motor 1

Tabela 2 – Sinais de Controle no Microcontrolador

O programa para o microcontrolador, desenvolvido em Assembly pode ser dividido em três partes: programa principal, interrupção por *overflow* em Timer 0 e interrupção pela USART (*Universal Syncronous and Asynchronous Receiver Transmitter*). O programa principal permite iniciar os registradores e as funções especiais do microcontrolador. Após a execução dessas ações, o sistema fica travado aguardando interrupções. Apresenta-se o fluxograma do programa principal, na Figura 3.

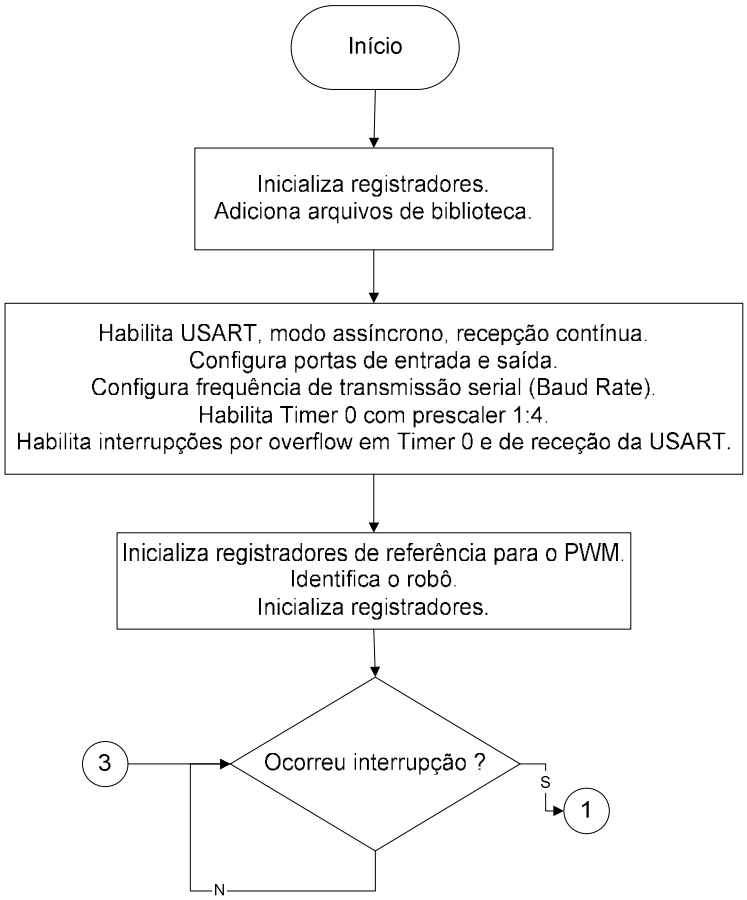

Figura 3 – Fluxograma do Programa Principal

l

Uma das interrupções (interrupção por *overflow* em Timer 0) depende do funcionamento de um temporizador e ocorrerá após um intervalo de tempo T_V correspondente ao tempo necessário para que ocorra estouro na contagem de 8 *bits* do temporizador, dado por:

$$T_V = N\,.\,\text{Prescaler}\,.\frac{1}{f_{osc}}$$

onde:

- N é o número de incrementos no Timer 0, de 8 *bits*, dado por $2^8 = 256$;
- *prescaler* é um divisor de freqüência programável que permite ajustar a freqüência de incremento do Timer 0. Nesta aplicação, utilizou-se um divisor de freqüência 1:4.

- f_{OSC} é a freqüência de oscilação do temporizador no caso de 1 MHz.

Substituindo-se na equação, resulta:

$$T_V = 256.4.\frac{1}{1.10^6} = 1{,}024ms$$

A interrupção por *overflow* no Timer 0 é utilizada para produzir os sinais PWM (*Pulse Width Modulation*) para os servomotores. Na Figura 4 apresenta-se o fluxograma da estratégia utilizada para produzir esses sinais. Serão utilizadas duas variáveis com valores admissíveis de 0 a 15: COUNT e PWM. A variável COUNT é um contador de interrupções incrementada a cada interrupção até chegar ao limite 15: ao ser incrementada após esse limite é zerada novamente. A variável PWM apresenta a largura de pulso desejada para o PWM. Por exemplo, se PWM=10, a saída apresentará nível lógico 1 somente durante o período de 10 interrupções de um total de 15.

A interrupção de recepção pela USART é utilizada para receber serialmente os dados enviados pelo computador. Os dados serão transmitidos pelo computador numa freqüência conhecida como Freqüência BAUD RATE, limitada pela velocidade de transmissão dos módulos RF. O cálculo da freqüência é dado pela equação abaixo, onde Fator representa o valor de 8 *bits* ajustado no registrador especial SPBRG.

$$f_{BAUD\,RATE} = \frac{f_{OSC}}{64(Fator+1)}$$

Em nosso caso, utilizamos Fator = 26 e f_{OSC} de 4 MHz que resultou numa Freqüência BAUD RATE de 2400 bits/s.

O fluxograma do programa de interrupção de recepção pela USART é apresentado na Figura 5.

O programa completo em assembly para o microcontrolador PIC dos robôs é apresentado no Anexo 1.

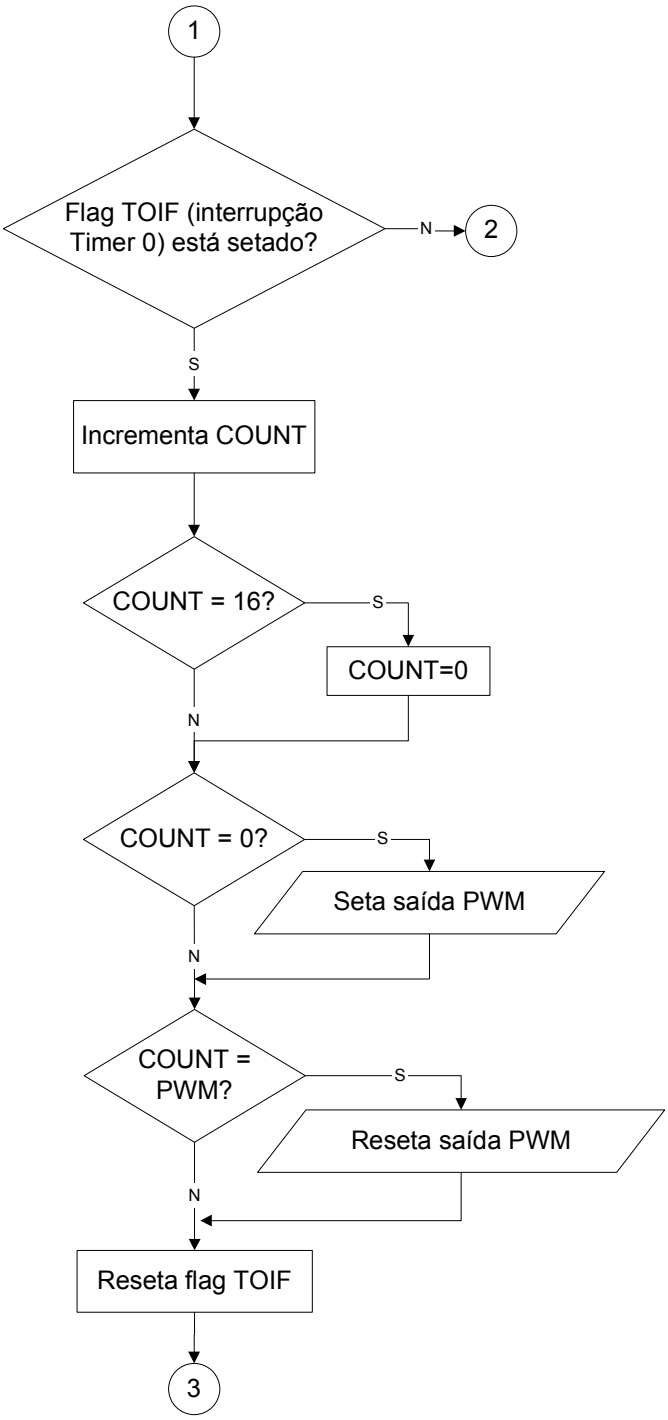

Figura 4 – Fluxograma da Interrupção por Overflow em Timer 0

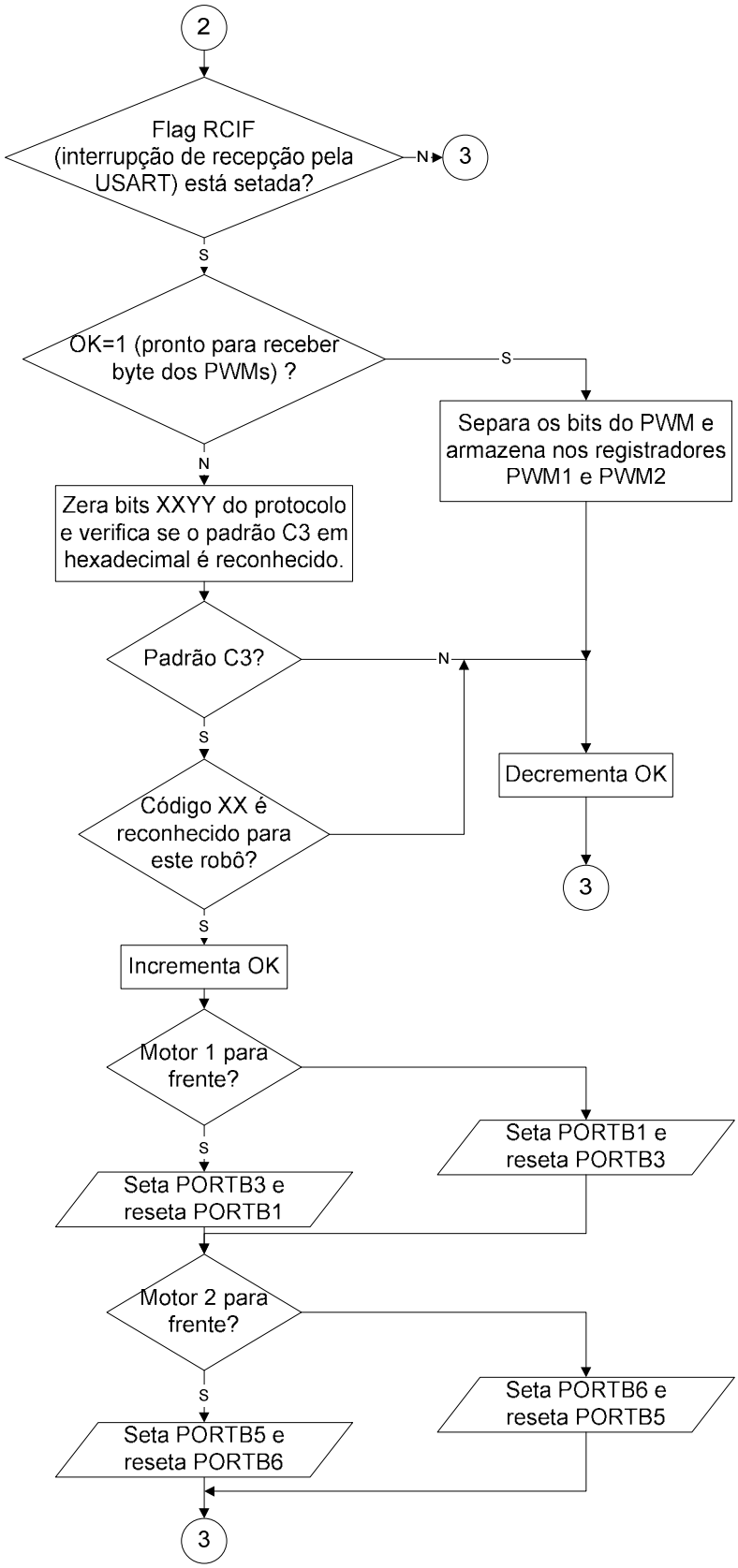

Figura 5 – Fluxograma da Interrupção de Recepção pela USART

4. Controle dos Robôs por Computador

O principal objetivo do futebol de robôs é o de implementar de forma autônoma o controle dos robôs para desenvolver uma estratégia de jogo. Isso significa que, nas competições, os robôs não podem ser controlados de forma remota. Contudo, na fase de desenvolvimento da estratégia e utilização do sistema de visão, o desempenho fica totalmente comprometido caso ocorram falhas no *hardware* dos robôs. Por isso é essencial que os robôs possam ser testados de forma independente antes de partir para a parte de programação da estratégia; neste caso, com os robôs efetivamente funcionando de forma autônoma. Por isso a equipe de futebol de robôs desenvolveu um algoritmo que permite efetuar o controle remoto dos robôs por computador. Esse algoritmo permite comandar os robôs em todas as direções por meio do acionamento de teclas. Esta solução permite detectar falhas no sistema de transmissão por meio de radiofreqüência, erros na programação do microcontrolador e principalmente maus contatos e falhas na soldagem dos componentes eletrônicos sobre a placa de circuito impresso do robô.

Na Figura 6 apresenta-se a janela de comando do algoritmo de controle e teste dos robôs desenvolvido em Delphi®. Com esse algoritmo é possível controlar de forma remota os robôs em todas as direções (para trás, para frente, girar no sentido horário e anti-horário), ajustar a velocidade e o ângulo de orientação, etc.

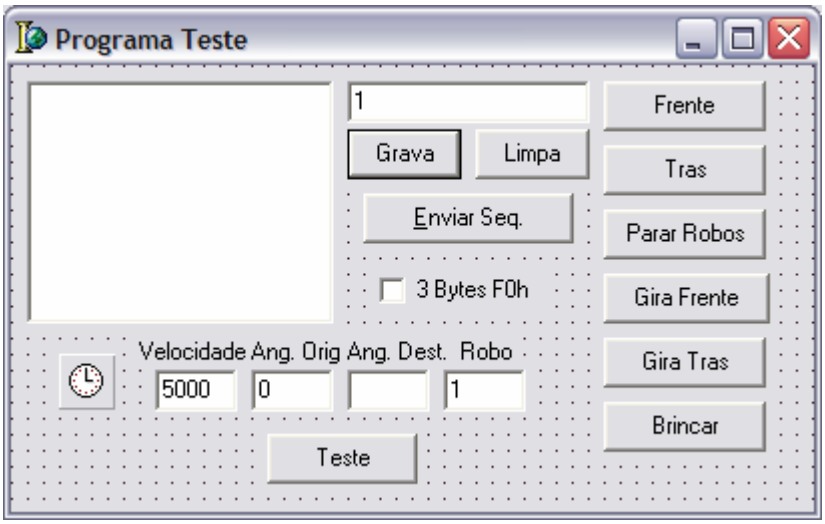

Figura 6 – Janela de Comando do Algoritmo de Controle e Teste dos Robôs

5. Considerações Finais

Neste artigo foi apresentada uma solução simples para o protocolo de comunicação entre o PC e os robôs jogadores de futebol da equipe da Escola de Engenharia Mauá. O trabalho apresenta também uma descrição detalhada do algoritmo de programação dos microcontroladores nos robôs.

Para o desenvolvimento do futebol de robôs, várias outras áreas de pesquisa estão envolvidas e várias etapas devem ainda ser implementadas, que incluem:

- desenvolvimento do algoritmo de estratégia para controle dos robôs de forma autônoma utilizando-se qualquer linguagem tais como: C^{++}, Delphi®, ou mesmo desenvolver a programação em sistema operacional Linux;

- desenvolvimento de um sistema de visão computacional;

- utilização de técnicas de inteligência artificial para melhorar a eficiência do sistema.

O objetivo principal é o de que o projeto apresente uma evolução constante, permitindo o desenvolvimento de novas pesquisas e incentivando cada vez mais o aprendizado dos alunos.

Anexo 1: Programa em Assembly para Microcontrolador PIC16F628

```
;*****************************************************************
;
; Projeto de um Robô comandado pelo computador
;
*****************************************************************

        CBLOCK      0X70   ; Definição de variáveis - posição comum a todos bancos

        TEMP
        RBID
        OK
        PWM1
        PWM2
        COUNT1
        COUNT2

        ENDC
```

```
#INCLUDE P16F628.inc

#DEFINE     BANK1       BSF    STATUS,RP0 ; Seleciona Banco 1 para RAM
RAM
#DEFINE     BANK0       BCF    STATUS,RP0 ; Seleciona Banco 0 para RAM

        ORG   0X00
        GOTO        INICIO

        ORG   0X04                      ; Vetor de início das interrupções
        BTFSC       INTCON,T0IF         ; Verifica se ocorreu interrupção de Timer
        GOTO        TIMER0              ; Se sim, desvia para rotina do Timer0
        BTFSS       PIR1,RCIF           ; Verifica interrupção de recepção USART
        RETFIE

RX      MOVFRCREG,W                     ; Tratamento da interrupção pela USART
        MOVWF       TEMP
        MOVLW       .1                  ; Byte de Controle já recebido?
        SUBWF       OK,W
        BTFSC       STATUS,Z
        GOTO RECEBE                     ; Sim, então desvia p/ receber dados PWM

        MOVLW       0XC3                ; Caso seja o primeiro byte recebido (byte
                                        ; de controle)
        ANDWF       TEMP,W              ; Verifica padrão 11XXYY11.
        SUBLW       0XC3
        BTFSSSTATUS,Z
        GOTO        SAI                  ; Se não, sai do procedimento.

        MOVLW       B'00XX0000'         ; Ajuste XX = 01 caso este seja o robô 1,
                                        ; XX = 02 para robô 2 e assim por diante.
        ANDWF       TEMP,W
        SUBWF       RBID,W
        BTFSS       STATUS,Z            ; Caso não seja para esse robô,
        GOTO        SAI                 ; sai do procedimento
        INCF  OK,F                      ; Se for, incrementa 1 na variável OK,
                                        ; indicando que recebeu o byte de controle
MOTOR1      BTFSC       TEMP,2          ; Verifica sentido motor 1
        GOTO REVERTER                   ; Caso seja para trás, pula para rotina de
                                        ; reversão
        BSF   PORTB,3
        BCF   PORTA,1

MOTOR2      BTFSC       TEMP,3          ; Verifica sentido motor 2
        GOTO REVERTER2                  ; Caso seja para  trás , pula para rotina
        BSF   PORTB,5
        BCF   PORTB,6
        GOTO SAI
```

```
REVERTER
        BSF    PORTA,1          ; Inverte acionamento dos bits para motor 1
        BCF    PORTB,3
        GOTO MOTOR2

REVERTER2
        BSF    PORTB,6          ; Inverte acionamento dos bits para motor 2
        BCF    PORTB,5
        GOTO SAI

RECEBE                          ; Recebe informação dos PWMs
                                ; (velocidade dos motores)
        MOVLW       0X0F
        ANDWF       TEMP,W      ; Lê byte PWMs
        MOVWF       PWM1        ; Separa parte baixa referente ao PWM1

        MOVLW       0XF0
        ANDWF       TEMP,F      ; Separa parte alta referente ao PWM2
        RRF    TEMP,F
        RRF    TEMP,F
        RRF    TEMP,F           ; Rotaciona 4 vezes para ajustar
        RRF    TEMP,F           ; parte alta do byte (PWM2)

        MOVFW       TEMP
        MOVLW       0X0F
        ANDWF       TEMP,W
        MOVWF       PWM2        ; Guarda o valor em PWM2
SAI1    DECF        OK,F        ; Zera OK; indica recebimento byte PWM
SAI     BCF         PIR1,RCIF   ; Limpa flag da interrupção por USART
        RETFIE

TIMER0                          ; Interrupção de Timer0
        INCF   COUNT1,F         ; Incrementa COUNT1 (PWM do Motor1)
        MOVLW       0X0F
        ANDWF       COUNT1,F    ; Verifica se COUNT1 passou de 15.
        INCF        COUNT2,F    ; Incrementa COUNT2 (PWM do Motor 2)
        ANDWF       COUNT2,F

; Rotina do PWM1
        MOVF        COUNT1,W
        BTFSC       STATUS,Z    ;Verifica se COUNT está em 0
        BSF         PORTB,4     ; Se estiver,  liga o PWM
        SUBWF       PWM1,W      ; Testa para ver se o COUNT1=PWM1
        BTFSC       STATUS,Z
        BCF    PORTB,4          ; Se for, desliga PWM

; Rotina do PWM2
```

```
        MOVFCOUNT2,W            ;Mesmo procedimento para o PWM2
        BTFSC       STATUS,Z
        BSF    PORTB,7
        SUBWF       PWM2,W
        BTFSC       STATUS,Z
        BCF    PORTB,7

        BCF    INTCON,T0IF      ; Limpa flag da interrupção por timer
        RETFIE

; Programa Principal

INICIO
        BANK0
        MOVLW       B'10010000'  ; Habilita USART para recepção contínua e
        MOVWF       RCSTA        ; Modo Assíncrono (TXSTA = 0 inicial)

        BANK1
        MOVLW       B'00000001'
        MOVWF       TRISA        ; Configura direção das portas A
        MOVLW       B'00000010'
        MOVWF       TRISB        ; Configura direção das portas B
        MOVLW       .25          ; Controle do gerador BAUD RATE
        MOVWF       SPBRG        ; ~2400 bits/s
        MOVLW       B'00000001'  ; Habilita Timer 0 para trabalhar com
        MOVWF       OPTION_REG   ; freqüência 1:4
        MOVLW       B'11100000'  ; Habilita interrupção por overflow Timer 0
        MOVWF       INTCON       ; e interrupção de grupo de periféricos
        BSF         PIE1,RCIE    ; Habilita interrup. recepção pela USART

        BANK0
        MOVLW       .15
        MOVWF       COUNT1       ; Inicia os PWMs
        MOVWF       COUNT2
        MOVLW       B'00XX0000'  ; Identificação do robô: p/ robô 1 XX=01,
        MOVWF       RBID         ; para robô 2 XX=10, e assim por diante.

VOLTA
        GOTO VOLTA               ; Fica aguardando interrupção: Timer ou
                                 ; Recepção.
        END
```