

## Sumário

Sumário .....	1
Introdução .....	5
1. Conceitos básicos .....	8
1.1. Robôs móveis .....	8
1.1.1. Sistema de controle e supervisão .....	11
1.1.2. Sistema de percepção .....	13
1.1.2.1. Sensores de distância .....	14
1.1.2.2. Sensores de proximidade.....	14
1.1.2.3. Sensores de posicionamento.....	14
1.1.2.4. Sistemas de visão .....	15
1.1.2.5. Sensores de estado .....	15
1.1.3. Sistema de acionamento.....	15
1.1.3.1. Efetuadores.....	15
1.1.3.2. Atuadores .....	16
1.2. Futebol de robôs .....	17
1.3. O projeto Furgbol.....	20
1.4. Movimentação omni-direcional.....	21
1.4.1. Outros tipos de movimentação.....	21
1.4.1.1. Tração diferencial.....	22
1.4.1.2. Tração síncrona.....	22
1.4.1.3. Triciclo.....	22
1.4.2. A movimentação omni-direcional.....	23
2. Arquitetura de controle.....	24
2.1. Especificações .....	24
2.2. Proposta de arquitetura.....	26
2.2.1. Módulo de comunicação .....	27
2.2.2. Módulos comportamentais .....	27
2.2.3. Módulo de decisão.....	28
2.2.4. Módulo de controle .....	31
3. Instanciação da arquitetura .....	32
3.1. Módulos comportamentais .....	32
3.1.1. Módulo de direção a uma coordenada.....	32
3.1.2. Módulo de evasão de obstáculo .....	34
3.2. Módulo de controle .....	36
3.3. Módulo de comunicação.....	38
3.4. Módulo de decisão .....	38
4. Implementação.....	39
4.1. Microcontroladores .....	41
4.2. Estrutura.....	43
4.3. Placa de controle .....	44
4.3.1. PWM .....	45
4.3.2. Recebendo um vetor de posição.....	47

4.3.3. Decompondo o vetor de posição.....	49
4.3.4. Gerando os sinais de controle .....	50
4.3.5. Movimentando os motores .....	55
4.3.6. Controlando os motores .....	57
4.4. Placa de deliberação.....	61
4.4.1. Leitura e interpretação dos sensores .....	62
4.4.2. Comunicação com o computador externo .....	65
4.4.3. Geração de vetores velocidade para a placa de controle .....	67
4.5. Execução.....	69
5. Testes e resultados.....	72
6. Considerações finais.....	75
Bibliografia.....	78

**Lista de figuras**

Figura 1: diagrama de um robô .....	9
Figura 2: arquitetura clássica.....	12
Figura 3: arquitetura comportamental.....	13
Figura 4: elementos do futebol de robôs.....	19
Figura 5: análise ambiental dos módulos.....	26
Figura 6: exemplo de comportamento emergente.....	30
Figura 7: fluxo de controle do módulo de controle .....	31
Figura 8: fluxo de informações no módulo .....	33
Figura 9: sugestão de configuração para os sensores .....	35
Figura 10: evasão de obstáculos .....	36
Figura 11: disposição das rodas.....	37
Figura 12: fluxo de informações no módulo de decisão.....	39
Figura 13: PIC 16F84.....	42
Figura 14: PIC 16F877 .....	42
Figura 15: concepção inicial da estrutura do robô.....	43
Figura 16: diagrama completo da placa de controle .....	45
Figura 17: trecho do código para recebimento de um byte.....	49
Figura 18: trecho do código que decompõe o vetor de posição.....	50
Figura 20: código da função de mapeamento do PWM.....	54
Figura 21: trecho de código para geração dos sinais PWM .....	54
Figura 22: esquemático de uma Ponte H.....	55
Figura 23: diagrama do L293D.....	56
Figura 24: encapsulamento do L293D .....	56
Figura 25: encoder óptico típico .....	58
Figura 26: disco refletor para encoder.....	59
Figura 27: diagrama de ligação do sensor Hamamatsu.....	59
Figura 28: disposição do disco de encoder no motor.....	60
Figura 29: foto de um receptor infra-vermelho.....	63
Figura 30: vibrador astável para 30KHz.....	63
Figura 31: trecho do código de leitura dos sensores e evasão .....	65
Figura 32: diagrama do SILRX-433-5, com circuito de ligação típico.....	66
Figura 33: trecho de código para dirigir a uma coordenada.....	68
Figura 34: trecho de código do módulo de decisão .....	68
Figura 35: placas de circuito impresso geradas.....	70
Figura 36: foto do robô finalizado .....	71
Figura 37: resultados dos testes realizados .....	74

**Lista de Tabelas**

Tabela 1: níveis de PWM versus velocidade do motor .....	51
Tabela 2: lista de comandos.....	66

## **Introdução**

Desde a criação das primeiras ferramentas mecânicas, o ser humano vem tentando facilitar e automatizar diversas tarefas que exigem esforço físico. Inicialmente, essas ferramentas utilizavam unicamente a energia do próprio homem, sob controle total deste. As máquinas surgiram de modo a converter outros tipos de energia em trabalho útil para o homem. Surgiram os tratores, carros, navios e outras tantas invenções. Essas máquinas operam totalmente sob controle do homem, tornando possíveis novos tipos de tarefas, antes impossíveis e facilitando a maioria das tarefas existentes. Porém, o homem ainda necessitava estar fisicamente presente, controlando cada aspecto dessas máquinas.

Os robôs surgiram de modo a também retirar o controle da responsabilidade do homem, substituindo totalmente o ser humano na execução de tarefas, sejam por serem elas repetitivas, insalubres ou mesmo anteriormente inviáveis. A utilização de robôs no lugar do trabalho manual traz três vantagens principais [BOT96]: melhoria na qualidade, aumento de produtividade e redução de acidentes envolvendo humanos. Para que essas vantagens se tornem evidentes, é necessário que toda a rota desde a concepção até a implementação dos robôs seja executada da melhor forma possível.

A robótica móvel surgiu da necessidade de se ter robôs capazes de locomoção em ambientes, substituindo com vantagens, em tarefas específicas, o ser humano. Porém, ao permitir que os robôs se locomovam, estende-se um ambiente muito mais complexo e dinâmico a esses. Como resposta, os robôs devem ser também mais complexos e capazes de interações inteligentes com os objetos ao seu redor.

O desenvolvimento de arquiteturas para robôs têm papel fundamental tanto na concepção quanto na construção de melhores robôs. A arquitetura estuda o

comportamento do robô, suas relações com o ambiente e define seus limites e capacidades. Diversas propostas de arquiteturas já foram apresentadas, que serviram de incentivo para este trabalho [BRO85] [MIC96] [DES92] [BRO90].

A importância atual dos robôs na sociedade e a potencialidade de utilizar-se robôs móveis para um número muito maior de tarefas, envolvendo desde tarefas cotidianas e caseiras até a exploração espacial, serve de motivação para o estudo e desenvolvimento de arquiteturas e técnicas de implementação de robôs móveis.

O Projeto Furgbol, que é o projeto de futebol de robôs da FURG, oferece motivação adicional para este trabalho. O projeto exige a especificação e implementação de robôs móveis para os jogos e o presente trabalho terá aplicação imediata neste projeto.

Ainda, a realização desta pesquisa oferece contribuição para o crescimento da área de robótica na FURG, gerando conhecimentos para outros projetos correlatos.

O objetivo deste trabalho é desenvolver uma arquitetura para robôs móveis omni-direcionais e oferecer uma proposta para implementá-la para uso em jogos de futebol de robôs, em particular aos ligados ao projeto Furgbol da FURG. O robô implementado deve oferecer características de robôs autônomos, como capacidade de lidar com obstáculos e cumprir objetivos. Faz parte deste trabalho o estudo de técnicas de implementação tanto da arquitetura em um nível teórico quanto da parte eletrônica do robô.

É também objetivo deste trabalho realizar uma revisão das arquiteturas mais utilizadas atualmente, bem como das técnicas de implementação de robôs móveis. Serão feitos testes para acompanhar a eficácia da arquitetura projetada, visando o controle omni-direcional. Ao longo do trabalho, com a revisão bibliográfica realizada e desenvolvimento de diversos algoritmos e produtos secundários, estar-se-á contribuindo com o laboratório de Robótica e Inteligência Artificial da FURG.

O presente **Capítulo 1** faz uma introdução ao trabalho, descrevendo as motivações, objetivos e estrutura geral. No **Capítulo 2**, conceitos básicos necessários para compreender o contexto do trabalho são apresentados, juntamente com a revisão bibliográfica do tema. O **Capítulo 3** apresenta a Arquitetura de Controle desenvolvida

para o controle de robôs omni-direcionais e no **Capítulo 4** é feita uma instanciação desta arquitetura para uso no futebol de robôs. O **Capítulo 5** discute a implementação efetiva da arquitetura em um robô físico real. No **Capítulo 6** os testes e resultados obtidos são apresentados. Finalizando, o **Capítulo 7** disserta sobre as conclusões tiradas do trabalho e apresenta trabalhos futuros para continuidade.

## 1. Conceitos básicos

A idealização de uma arquitetura de controle para robôs móveis omni-direcionais para jogos de futebol de robôs, bem como a implementação de uma instância desta arquitetura, demanda o conhecimento básico das principais áreas envolvidas nessas tarefas. Nesta seção são apresentados os robôs móveis, seguido por uma descrição do futebol de robôs, onde os robôs móveis são utilizados, e a apresentação do projeto de futebol de robôs na FURG, o projeto Furgbol, finalizando com a exposição da movimentação omni-direcional, que caracteriza este trabalho.

### 1.1. Robôs móveis

Russell e Norving [RUS95] definem uma entidade robótica como sendo: *Um agente ativo, artificial, cujo ambiente seja o mundo físico, capaz de perceber seu ambiente e atuar sobre esse.* São máquinas desenvolvidas e utilizadas para realizar funções mecânicas antes efetuadas pelo homem.

Basicamente se pode dividir um robô em três partes principais<sup>1</sup> [BOT96]: sistema de percepção, que gera informações sobre o ambiente; sistema de acionamento, composto por atuadores e efetadores, responsável por agir sobre o ambiente, seja para locomover o robô ou interagir com outros elementos presentes; sistema de controle e supervisão, responsável por implementar uma arquitetura de controle, que interpreta os sensores e comandos externos, gerando ações coerentes

---

<sup>1</sup> Essas partes compõem a essência de um robô em um nível mais abstrato. Porém, outras partes são necessárias e essenciais, sendo elas: fonte de energia, responsável por fornecer a energia necessária para as partes funcionarem; sistema de comunicação, necessário para troca de informações do robô com o mundo externo, sejam elas comandos para o robô ou dados coletados por este; estrutura física (corpo), necessária para manter as partes juntas, compondo o robô. Há ainda uma *glue logic*, que é a *interface* dessas diversas partes.

sobre o ambiente. A Figura 1 mostra um diagrama simplificado dos principais componentes de um robô.

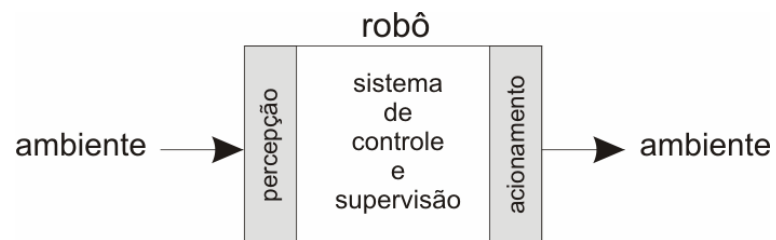


Figura 1: diagrama de um robô

Um robô móvel é, portanto, uma máquina capaz de locomover-se com alguma autonomia por um ambiente, realizando desde processos repetitivos até tarefas “inteligentes” neste ambiente, isto é, atuando sobre e interagindo com este. Se diferenciam de robôs fixos pela sua mobilidade, já que estes últimos apresentam uma base fixa e apenas movimentam seu efetuador no ambiente. E diferem de máquinas comuns (como tratores, esteiras, etc.) pela capacidade de não exigir um controle total do homem sobre esta. Ou seja, há um grau de *autonomia* associado a máquina, capaz de realizar tarefas designadas sem auxílio humano.

A autonomia de um robô móvel está na sua capacidade de locomover-se e realizar ações no ambiente sem intervenção humana. Há diversos graus de autonomia possíveis. O menor grau de autonomia está nos robôs tele-operados. Nestes, um operador humano controla cada aspecto de movimentação do robô, substituindo quase por completo o sistema de controle deste. Alguns graus acima, estão os robôs que, apesar de serem operados por humanos, possuem capacidade de efetuar tarefas simples e reagir a modificações no ambiente, sem intervenção humana, se necessário. Por exemplo, um robô poderia ser controlador por um *joystick* mas ainda assim ser capaz de perceber quando haverá um choque com uma parede e parar, independente das ordens dadas. Em um grau mais elevado, estão os robôs totalmente autônomos. Estes não sofrem intervenção humana de qualquer tipo, são capazes de traçar seus próprios objetivos e executá-los. A complexidade da arquitetura cresce com o grau de autonomia. Também, quanto maior for a autonomia de um robô, maior quantidade e

qualidade de sensores serão necessários, além de uma maior habilidade no controle de seus efetadores.

Robôs móveis incluem todo e qualquer tipo de robô capaz de se locomover. Este trabalho, porém, focalizará os robôs móveis terrestres, realizando apenas eventuais comentários em relação a robôs aéreos e aquáticos.

Diversos são os desafios de se especificar e implementar um robô móvel. Um robô é efetivamente um agente físico, operando sobre um ambiente real. Ambientes reais oferecem uma série de problemas [MAX00][RUS95]: o conhecimento sobre o ambiente é geralmente incompleto, incerto e aproximado. Além disso, as medidas geradas pelos sensores possuem ruídos, limitações de alcance e visibilidade e ainda há a possibilidade de erros na interpretação das informações. Mais ainda, ambientes reais tendem a serem extremamente dinâmicos, com objetos em movimento, mudança em condições de luz e terreno. Como resultado, a execução de ações pelo robô não é completamente livre de falhas.

Os robôs móveis, com alguma autonomia, possuem como principais desafios [BOT96]:

- microinformática embarcada potente: para assegurar que o robô não necessite de dispositivos externos, deve esse possuir de forma embarcada toda a estrutura para processamento de dados;
- processamento em tempo real: os sinais gerados pelos sensores devem ser tratados de forma rápida, de modo a gerar reações pertinentes no ambiente;
- comportamento adaptivo: o sistema deve ser capaz de se adaptar ao ambiente, tornando-se mais eficiente e robusto;
- facilidade para tratamento de ruídos: sistemas físicos são imperfeitos por natureza, o que acarreta na presença de ruídos nos sinais captados por sensores, que devem ser tratados de forma adequada;
- capacidade de mapear ambientes: o sistema deve ser capaz de gerar representações internas do ambiente em que está inserido, de modo a permitir a execução de determinadas tarefas.

Para uma melhor compreensão das partes componentes de um robô, a seguir serão apresentadas as principais, focando especificamente em robôs móveis.

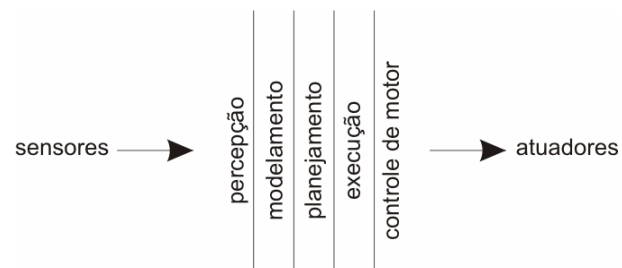
### 1.1.1. Sistema de controle e supervisão

É o sistema de controle que constitui o meio físico para a implementação da arquitetura do robô. Atualmente, é tipicamente um sistema digital capaz de ler os sensores, executar algoritmos de controle e gerar saídas para os atuadores. Em robôs simples de pequeno porte esse sistema pode ser composto de um ou mais microcontroladores. Robôs maiores utilizam sistemas computacionais mais completos, com processadores mais rápidos. Sistemas de controle analógicos são também utilizados em alguns casos [ALV1].

De acordo com [RUS95], a arquitetura de um robô é o modo com que este organiza as tarefas de geração de ações a partir de suas percepções. De forma menos ampla, podemos dizer que a arquitetura de um robô é a forma com que o sistema de controle gera ações a partir da leitura de sensores e do recebimento de comandos externos. A arquitetura define o comportamento do robô no ambiente e, conseqüentemente, seu grau de *autonomia*. A complexidade de uma arquitetura cresce com o grau de autonomia, complexidade do ambiente, complexidade das tarefas e necessidade de precisão. Com a complexidade da arquitetura, cresce também a necessidade de maior capacidade de processamento pelo sistema de controle.

A arquitetura clássica [RUS95], desenvolvida pela empresa *SRI* e implementada inicialmente no robô *SHAKY*, trata o problema do controle de forma semelhante a uma programação imperativa, conforme ilustrado na Figura 2, onde rotinas de baixo nível são combinadas para gerar rotinas de mais alto nível, até chegar no nível de tarefas ou comandos externos. Nessa arquitetura, a camada de baixo nível (chamada *LLA – low-level actions*) trata de aspectos físicos do robô, como movimento das rodas e leitura de sensores. Uma camada intermediária (*ILA –*

*intermediate-level actions*) é capaz de tratar comandos de mais alto nível, como “mover o robô para coordenada (x,y)” ou “esticar o braço” e, também, gerar planos e trajetórias de movimento. Essas ações são composições de rotinas presentes na LLA. Os elementos básicos dessa arquitetura são utilizados em grande parte dos sistemas modernos.

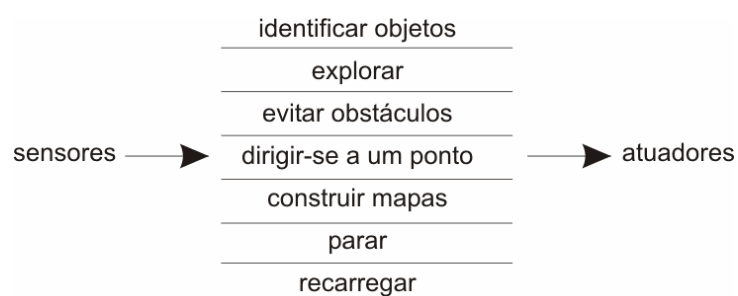


**Figura 2: arquitetura clássica**

Nos anos 80, novas pesquisas surgiram na área de arquiteturas para robôs [RUS95]. Conforme a necessidade de interações mais complexas com o ambiente cresce, a quantidade de processamento começa a se tornar limitante para que o robô consiga reagir em tempo real. Esse problema, na arquitetura clássica, era parcialmente resolvido utilizando macro-operadores, que funcionam como uma *cache* de ações, compilando conjuntos de ações que foram úteis no passado para uso posterior, sem necessidade de deliberar essa seqüência. Uma outra solução é eliminar completamente deliberações explícitas. Essa é a idéia por trás dos chamados “autômatos situados” e dos modelos baseados em comportamento.

Um autômato situado é uma máquina de estados cujas entradas são fornecidas pelos sensores e cujas saídas são conectadas aos efetadores [RUS95]. Neste paradigma, o *automaton* é especificado em termos declarativos, em um nível simbólico. Esta especificação é compilada, *off-line*, para uma máquina digital, que satisfaz as especificações das declarações. A partir daí, a máquina digital não faz qualquer tipo de manipulação simbólica, todas as respostas do sistema a estímulos conhecidos são mapeados de forma fixa no sistema. Este tipo de técnica é por vezes conhecida como *reativa*.

Modelos baseados em comportamento foram inicialmente propostos por Rodney Brooks em [BRO85]. A idéia por trás dessa arquitetura é que a especificação do robô pode ser decomposta não em unidades funcionais, como planejamento, movimento, sensoreamento, mas em comportamentos como evitar obstáculos, explorar ambiente, seguir objetivos [RUS95]. A Figura 3 ilustra o conceito desse modelo. O sistema é dividido em módulos comportamentais e cada módulo tem acesso aos sensores que lhe interessam, de forma independente, e geram sinais independentes para os atuadores. Há uma hierarquia de módulos, onde os de mais alto nível podem acessar informações dos de mais baixo nível, além de modificar ou inibir suas saídas. A vantagem da arquitetura baseada em comportamento é a eliminação da necessidade de um controle centralizado e de um modelo do mundo (mapeamento do ambiente). Comportamentos mais complexos tendem a emergir da interação de comportamentos simples com o ambiente. Porém, [RUS95] discute a dificuldade de gerar comportamentos que cumpram tarefas mais complexas, uma vez que eventualmente a interação dos diversos módulos tornar-se-ia excessivamente complexa, necessitando de outros controladores para resolver conflitos e cumprir a tarefa designada, sacrificando a proposta de simplicidade original.



**Figura 3: arquitetura comportamental**

### 1.1.2. Sistema de percepção

Para que o sistema de controle seja capaz de gerar ações sobre o ambiente, de modo a apresentar um comportamento inteligente, é necessário, antes, que este tenha

algum conhecimento do ambiente. Por exemplo, para se locomover, é preciso saber onde há área livre para realizar a locomoção. Os sensores são responsáveis pelo fornecimento de informações do ambiente para o robô. Realizam algumas medidas sobre o ambiente e retornam sinais para o sistema de controle e supervisão, proporcionais as medidas efetuadas.

Há diversas classes de sensores, cada uma com um propósito específico. A seguir, apresenta-se as classes mais comuns utilizadas nos robôs atuais [GRA1].

#### **1.1.2.1. Sensores de distância**

Esses sensores realizam uma medida de distância no ambiente. São utilizados quando é necessário ter informações sobre a distância que um objeto se encontra do robô ou, ainda, fornecem informações necessárias para o mapeamento de uma determinada região, ao medir as diversas distâncias ao redor do robô.

Sensores comuns utilizam técnicas baseadas em sonares, lasers, sensores eletrostáticos e radares.

#### **1.1.2.2. Sensores de proximidade**

Similares aos sensores de distância, porém apenas informam a existência ou não de um objeto a uma distância fixa, pré-determinada, do robô. São utilizados quando não interessa a distância do objeto, mas apenas saber se ele está ou não em uma região próxima ao robô. Por exemplo, podem ser utilizados para que se detecte uma parede e se gere uma ação de parar o robô para que não haja o choque. Sensores infravermelhos e sensores de toque são os de uso mais comum.

#### **1.1.2.3. Sensores de posicionamento**

Fornecem informações sobre a posição e/ou orientação do robô no ambiente. São utilizados para que o robô saiba onde se encontra e qual seu estado no ambiente, de modo a gerar a ação necessária de acordo com sua posição. São úteis tanto para localização em um mapa quanto para detectar uma possível posição de desequilíbrio, que poderia ocasionar uma queda ou choque. Esta classe engloba acelerômetros, giroscópios, bússolas e GPS.

#### **1.1.2.4. Sistemas de visão**

São sensores que captam imagens do ambiente, de forma que se possa fazer interpretações sobre essas imagens posteriormente. Esta classe de sensores não fornecem informações de imediato, sendo um processamento necessário sobre as imagens adquiridas. Porém, a quantidade de informações que se pode retirar de uma imagem é grande, sendo estes sensores os de maior potencial de uso atualmente. Compõem essa classe basicamente as câmeras de vídeo, tanto do tipo CCD (digitais) quanto as convencionais analógicas.

#### **1.1.2.5. Sensores de estado**

Sensores internos são sensores que tem por função não medir o ambiente, mas medir recursos e estados do próprio robô. São necessários para realizar o controle de partes mecânicas internas ou, ainda, realizar medições de energia. Os mais utilizados são os encoders, *reed switch*, sensores de efeito hall e voltímetros.

### **1.1.3. Sistema de acionamento**

#### **1.1.3.1. Efetuadores**

Os efetadores são as partes que farão contato com o ambiente, impulsionados pelos atuadores. São eles que farão a interação do robô com seu ambiente, seja para locomoção ou manipulação. Estão incluídas nesta categoria as rodas, pernas, garras, braços, etc.

Em robótica móvel, são diversos os efetadores utilizados, dependendo do tipo de locomoção desejado. O tipo de locomoção pode ser por rodas, esteiras, pernas, hélices, ventosas, entre outros.

Entre os robôs móveis terrestres, há uma tendência inicial a se pensar em pernas para locomoção, já que, na natureza, essa é a estratégia mais utilizada nos animais. Porém, a locomoção com pernas é de grande dificuldade para robôs, uma vez que a complexidade de controle é alta. Há necessidade de controlar o equilíbrio, posição de cada perna, cada uma contendo diversas juntas e outras variáveis.

Rodas e esteiras são muito mais eficientes atualmente para locomoção de robôs, na maioria dos ambientes. São mais eficientes na utilização de energia e provêm suporte estático, isto é, permite que o robô permaneça parado sem gastos de energia para manter equilíbrio. O controle é bastante facilitado também, já que modelos cinemáticos relativamente simples descrevem com precisão a maioria dos sistemas com rodas ou esteiras.

#### **1.1.3.2. Atuadores**

Atuadores são transdutores, responsáveis por converter sinais do sistema de controle em movimentos mecânicos. Eles são responsáveis por movimentar os efetadores, que são as partes que farão contato efetivo com o ambiente em que o robô está inserido.

Atuadores incluem motores elétricos, pistões hidráulicos, pistões pneumáticos, entre outros. Motores elétricos são os preferidos na maioria dos robôs móveis, dado sua alta eficiência de conversão de energia e tamanho compacto.

Os atuadores definem se um robô é holonômico ou não. Um robô holonômico é tal que o número de movimentos independentes controláveis (graus de liberdade) é igual ao número de graus de liberdade do robô como um todo [LAT91]. Um robô não-holonômico é o que possui menor graus de liberdade controláveis do que os graus de liberdade totais. Por exemplo, um carro é não-holonômico, pois possui três graus de liberdade totais (duas posições espaciais e uma orientação), mas apenas dois graus de liberdade controláveis: direção das rodas e tracionamento. Sabe-se que é possível colocar um carro em qualquer posição com qualquer orientação no plano, mas a quantidade de manobras necessárias para isso pode ser grande. Robôs holonômicos, apesar de terem uma estrutura mais complexa, permitem movimentações mais simples, sem necessidade de muitas manobras para atingir uma posição.

## 1.2. Futebol de robôs

O futebol de robôs foi proposto originalmente, em 1992, por Alan Mackworth, professor na Universidade de Columbia, no Canadá, em um artigo denominado “On seeing robots” [MAC93b].

Um jogo de futebol de robôs é, de forma simplificada, uma partida, sobre um campo delimitado, entre dois times com um certo número de robôs. Cada time deve conduzir uma bola até o gol adversário, marcando assim pontos. As regras seguidas são semelhantes às de futebol, devidamente adaptadas.

Na sua forma mais comum, o futebol de robôs é composto pelos seguintes elementos, ilustrados na Figura 4:

- **Times:** dois times, de número variável de robôs. A quantidade e tamanho máximo de cada robô são definidos dependendo da categoria e diferentes robôs podem ser utilizados, desde que todos respeitem as regras;

- **Campo:** o campo usualmente é feito de material liso. Possui paredes que limitam o campo, impedindo que os robôs ou a bola saiam de campo. Há duas goleiras, uma em cada extremidade;
- **Visão:** pode-se ter uma câmera de vídeo colocada em posição acima do campo, chamada visão global, de modo que todo o campo seja visível, fornecendo imagens do jogo para ambos os times, ou cada robô pode possuir seu próprio sistema de visão ou, ainda, outra forma de sensoriamento;
- **Computadores externos:** um ou mais computadores externos, em cada time, processam as imagens fornecidas pelas câmeras e/ou geram estratégias para o time, repassando comandos para os robôs;
- **Bola:** uma bola é utilizada para a partida, de tamanho dependente do porte dos robôs.

Os jogos de futebol de robôs compõem ambientes extremamente dinâmicos, envolvendo diversos desafios para a robótica e áreas afins. Relacionado à área de robótica, existe a necessidade de robôs pequenos, ágeis e robustos, que precisam lidar com elementos estáticos (paredes) e móveis (outros robôs e a bola) em campo. No que tange ao processamento de imagens, há a necessidade de reconhecer elementos em campo, tanto estáticos quanto móveis, mapeando o ambiente de forma adequada, em tempo real, para que se possa extrair informações de forma rápida. Ainda, o ambiente oferece desafios relacionados à tática e à estratégia a ser adotada, já que cada time precisa ser controlado de forma inteligente, exigindo, talvez, cooperação entre os robôs, com regras de convívio. Esse conjunto de fatores, aliado ao entretenimento, faz do jogo um bom ambiente para testes e desenvolvimento de robótica móvel, processamento de imagens e controle multi-agente.



**Figura 4: elementos do futebol de robôs**

Atualmente, algumas organizações são responsáveis pela elaboração das regras em campeonatos internacionais. As duas principais são a *Federation of International Robot-soccer Association* (FIRA) e a Robocup. Cada uma divide-se em categorias, que vão desde pequenos robôs com rodas até robôs humanóides, cada categoria almejando objetivos específicos de desenvolvimento. As categorias, para cada organização, são as seguintes:

#### **FIRA [FIR1]**

- **Simurosot:** categoria que não envolve robôs físicos, sendo as partidas realizadas no nível de simulação;
- **KheperaSot:** jogo realizado com os robôs Khepera [KHP1], um por time, sendo estes totalmente autônomos com visão ou outros tipos de sensores embarcados;
- **MiroSot:** times de três robôs, cada um com tamanho máximo de um cubo de 7,5 cm de aresta. Utiliza-se de uma câmera global e um computador externo por time;
- **NaroSot:** times de cinco robôs, cada um com tamanho máximo de 4cm x 4cm x 5.5cm

- **RoboSot**: times de um até três robôs, com visão embarcada, de tamanho máximo de 20cm x 20cm x 40cm e um computador externo opcional;
- **HuroSot**: times humanóides, ainda sem regras definidas.

### **Robocup [ROB1]**

- **Simulation League**: semelhante a Simurosot, opera apenas em simulações;
- **Small-Size League (F-180)**: times de até cinco robôs, com tamanho máximo de um cilindro de 18 cm de diâmetro e 15 cm de altura. Visão global ou embarcada, com um computador externo opcional;
- **Middle-Size League (F-2000)**: times de até onze robôs, cada um não podendo ultrapassar no chão um quadrado de 50cm x 50cm. A altura deve estar entre 30cm e 80cm. Não é permitido o uso de visão global, ou qualquer sensor externo. Um computador externo é opcional;
- **Sony Four Legged Robot League**: times compostos dos robôs quadrúpedes Aibo, da Sony;
- **Humanoid League**: times compostos de robôs humanóides.

### **1.3. O projeto Furgbol**

O projeto Furgbol surgiu da necessidade de se ter um ambiente e agentes físicos para testes de sistemas de tomada de decisão distribuída, além de gerar conhecimentos em robótica móvel na Universidade.

Como objetivos específicos do projeto, estão a participação da FURG em campeonatos nacionais e internacionais, de modo a divulgar a produção e o nome da Universidade, e a construção de robôs móveis. De modo mais geral, o projeto visa aproveitar etapas desenvolvidas para o futebol de robôs em outros projetos, como, por exemplo, o projeto de Robótica Submarina, além de aproveitar os próprios robôs para tarefas mais nobres de controle multi-agente. Assim, há a necessidade de que os robôs

gerados no projeto sejam o mais genéricos e flexíveis possíveis, isto é, capazes de se adaptarem a outras tarefas.

Para guiar as atividades do projeto, optou-se por seguir as regras da categoria F-180 da Robocup. Esta categoria foi escolhida por ser a mais flexível, não exigindo robôs excessivamente pequenos (que aumentaria custos e dificultaria a mecânica do robô) e permitindo o uso tanto de visão global quanto de embarcada. Neste ponto, optou-se por iniciar o projeto utilizando-se visão global e evoluir posteriormente para visão local.

O projeto Furgbol foi separado em três áreas: visão, robótica e estratégia. A visão engloba a captura das imagens do campo e seus elementos, convertendo esses dados em informações úteis para gerar uma estratégia, como informações de posição de cada elemento em campo (bola, adversários, jogadores). A estratégia é responsável por traçar uma estratégia de jogo, isto é, a partir de informações obtidas da visão e de diretrizes internas, gerar novas posições para cada jogador em campo, de modo a cumprir metas. A robótica é responsável por desenvolver os robôs que participarão dos jogos, sob comando da etapa de estratégia, e é onde este presente trabalho se enquadra.

#### **1.4. Movimentação omni-direcional**

Há diversas maneiras de um robô móvel se locomover, que dependem da configuração de seus atuadores e efetadores. Aqui apenas serão tratadas as formas de locomoção baseadas em rodas, focando na movimentação dita omni-direcional. Porém, como foi visto, há categorias de futebol de robôs baseadas em robôs com “pernas” e outras formas de locomoção.

##### **1.4.1. Outros tipos de movimentação**

#### **1.4.1.1. Tração diferencial**

A tração diferencial utiliza-se de duas rodas tracionadas de forma independente, isto é, com atuadores independentes. Esses atuadores são as únicas variáveis de controle para locomoção do robô. Trata-se de um sistema não-holonômico, já que há dois graus de liberdade controláveis (velocidades das duas rodas) e o robô pode se locomover com três graus de liberdade (duas posições espaciais  $(x,y)$  e uma orientação). É simples de ver como o robô pode ser locomover com três graus de liberdade se notarmos que é sempre possível para o corpo girar em torno do seu eixo vertical (centro entre as rodas). A partir daí, para chegar em qualquer ponto no plano basta rotacionar o corpo para apontar na direção desse ponto e mover o robô até lá, rotacionando-o após em qualquer orientação.

Esse é o tipo mais comum de tração, por ser mecanicamente simples de implementar e possuir diversas técnicas já consagradas para controle de trajetórias.

#### **1.4.1.2. Tração síncrona**

Na tração síncrona, três rodas são dispostas nos vértices de um triângulo equilátero e podem ser tracionadas e direcionadas de forma conjunta. Todas as rodas possuem mesma orientação e velocidade. Isso garante um comportamento holonômico ao robô.

Esta técnica de locomoção possui a vantagem de tornar bastante simples a movimentação do robô no plano, bastando direcionar as rodas na direção desejada e tracioná-las. Porém, o sistema para movimentar de forma síncrona todas as rodas pode ser mecanicamente bastante complexo. Geralmente utiliza-se algum tipo de esteira que transmita a rotação de um motor para todas as rodas.

#### **1.4.1.3. Triciclo**

Nos robôs triciclos, há duas rodas de giro livre e uma roda de tração capaz de ser direcionada. É um sistema bastante simples de ser implementado, pois apenas uma roda precisa ser movimentada e controlada. Porém, a cinemática pode ser bastante complexa e, especialmente para movimentos em curtas distâncias, é possível haver dificuldades para posicionar o robô na posição e orientação desejada.

#### **1.4.2. A movimentação omni-direcional**

A movimentação omni-direcional é um tipo de locomoção holonômica no plano. Como toda movimentação holonômica, é obtida igualando o número de atuadores ao número de graus controláveis de movimento. Naturalmente, estes atuadores estão ligados diretamente ao movimento. A omni-direcionalidade é a capacidade de posicionar-se e se locomover no plano em qualquer direção e com qualquer orientação.

Um veículo omni-direcional é capaz de se locomover no plano, em qualquer direção e sentido, sem necessidade de rotacionar seu corpo. Além disso, é capaz de girar sobre seu próprio eixo e se mover ao mesmo tempo em que realiza o giro, de modo a atingir uma determinada posição já com uma orientação especificada.

A movimentação omni-direcional possui a grande vantagem de simplificar o processo de locomoção no plano, ao não impor restrições à mobilidade. Em um jogo de futebol de robôs, isso é particularmente útil, uma vez que aumentam as possibilidades de movimentação em campo e, em particular, permite que se mova o robô mantendo sua orientação, isto é, podendo-se manter o atuador de chute direcionado ao gol adversário durante a translação. Isso é uma vantagem já que uma mudança de orientação pode ser muito lenta em relação a velocidade da bola em campo.

## 2. Arquitetura de controle

### 2.1. Especificações

Para desenvolver o projeto, partiu-se de algumas premissas básicas: o robô a ser desenvolvido deve poder ser utilizado em outras atividades simples que não futebol de robôs; o tamanho e características físicas, porém, devem obedecer as regras da Robocup F-180; o custo total do robô deve ser mantido baixo, de modo a permitir sua replicação e a formação de um time completo.

Boa parte dos robôs para futebol de robôs, na categoria F-180, são controlados pelo computador externo, usualmente com realimentação através da câmera de vídeo global. O computador externo é responsável por gerar diretamente sinais de controle aos motores. Esse tipo de abordagem torna o sistema altamente dependente de um computador externo para realizar o controle, apesar de eficiente para futebol de robôs em específico.

De modo a tornar o robô um pouco mais genérico e modular, optou-se pela interação entre o robô e o computador externo se dar sob a forma de coordenadas. O robô recebe uma coordenada do tipo  $(x,y,?)$ , especificando uma posição e orientação no espaço. Cada coordenada é passada de forma assíncrona e aperiódica, estando a frequência de comando limitada unicamente pela velocidade de transmissão. Além disso, cada coordenada é relativa a um referencial externo pré-definido, que pode ser o ponto de início de operação ou, no caso do futebol de robôs, um ponto fixo no campo. Esse tipo de abordagem exige que o próprio robô gere uma trajetória de sua posição atual até a posição desejada, gerando seus próprios sinais de controle para os

atuadores. Mais ainda, exige que o robô tenha conhecimento da sua posição no ambiente, de modo que possa, a partir dessa posição, atingir o objetivo.

Ainda, o robô deve permitir a agregação de novas funções. Um sistema para evitar choques é imediatamente necessário, mas a evolução do projeto Furgbol prevê a utilização de câmera embarcada para sensoramento, também exigindo outros sensores para complementar as informações sobre o ambiente. Essas e outras novas funções devem ser passíveis de ser agregadas de forma simples ao robô, de forma modular, sem necessitar de grandes modificações nas partes já operantes.

A comunicação deverá ser feita através de *link* de rádio. O computador externo envia um sinal único (*broadcast*) que cada robô recebe e interpreta. A comunicação, em princípio, será unidirecional. Porém, o robô deve suportar também comunicação bidirecional, para que futuras evoluções possam usufruir dessa capacidade.

A movimentação omni-direcional foi especificada como forma de simplificar o processo de movimentação em campo, além de ser algo que ainda não foi implementado no País. O caráter vetorial do posicionamento é utilizado amplamente para o desenvolvimento de uma arquitetura modular, como será visto adiante.

Das considerações acima, pode-se tirar as especificações para o robô:

- Capacidade de processamento embarcada;
- Modular;
- Movimentação omni-direcional;
- Comunicação por *link* de rádio;
- Comandos passados na forma de coordenadas (x,y,?).

Estas especificações têm impacto direto na definição do tipo de arquitetura a ser utilizada no robô, pois esta dará suporte a todas essas funções no momento da implementação efetiva.

## 2.2. Proposta de arquitetura

Para preencher as especificações do sistema, a arquitetura de controle do robô foi idealizada como um modelo híbrido entre a arquitetura clássica, baseada em camadas hierárquicas, e a arquitetura comportamental *subsumption* de Brooks. Toda a arquitetura provê facilidades para a movimentação omni-direcional, sendo bastante dirigida para esta. A Figura 5 mostra os diversos módulos envolvidos na arquitetura e seus relacionamentos.

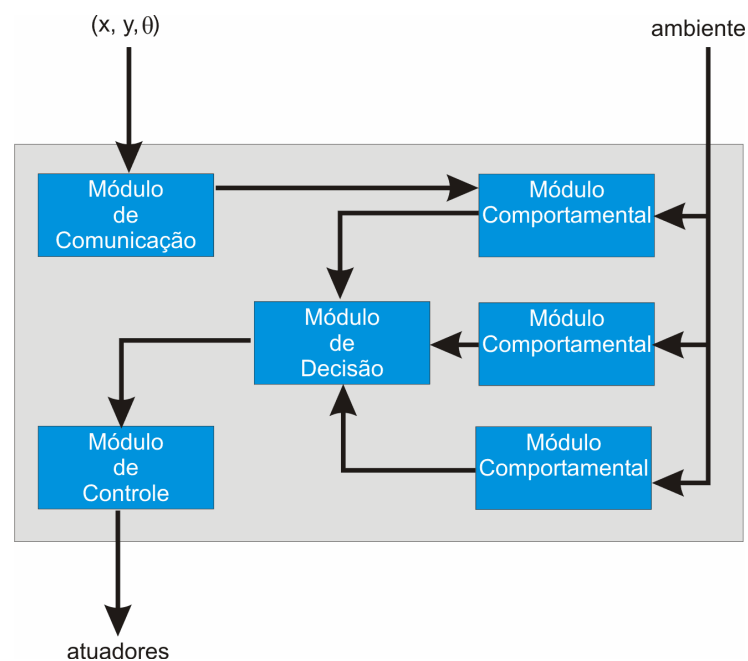


Figura 5: análise ambiental dos módulos

Note-se na Figura 5 que há módulos comportamentais, característicos da arquitetura de Brooks. Estes, porém, não se comunicam entre si e apenas fornecem “sugestões” de comportamento para o robô. A decisão de qual sugestão aceitar ou ainda, quais sugestões devem ser combinadas, é feita de forma centralizada. A seguir, cada módulo é explicado de forma detalhada.

### 2.2.1. Módulo de comunicação

O módulo de comunicação é responsável por receber novos comandos e por enviar informações para agentes externos. Todas as partes de baixo nível nas comunicações são tratadas neste módulo e seu produto, na recepção, é o comando conforme foi transmitido.

Por comando entende-se uma informação externa fornecida ao robô, composto de um identificador e um ou mais parâmetros. Cada comando recebido é passado para todos os módulos comportamentais, através de um barramento ou posição de memória, que então escolhem se aceitam ou não o comando e o tratam da forma conveniente. Comandos simples típicos incluem “vá para”, que comanda o robô a ir para uma determinada posição, passada como parâmetro e “pare”, que comanda o robô a parar totalmente.

### 2.2.2. Módulos comportamentais

Um módulo comportamental é um módulo independente capaz de gerar reações para o robô, a partir de dados dos sensores e de comandos externos. Cada módulo comportamental possui um objetivo local a ser seguido, definido pela sua pré-programação e as informações recebidas. A idéia é semelhante à originalmente discutida por Brooks, na sua arquitetura *subsumption* [BRO85].

A idéia básica é que cada módulo cuide de um comportamento simples, instintivo, do robô, realizando a conexão sensor-efetuador ou ordem-efetuador. Isto é, em cada módulo um estímulo externo gera diretamente um comportamento para o robô. Da interação de diversos módulos com o ambiente, podem surgir comportamentos mais complexos. Alguns tipos de módulos que poderiam ser integrados à arquitetura são:

- Evitar choques: a proximidade de objetos do robô, sentida pelos sensores, causam a reação do robô se afastar desses objetos, de modo a evitar o choque.

- Perseguir um objetivo: dado um objetivo (coordenada), o módulo gera seqüências de movimentos que tentem atingir tal coordenada.
- Recarregar: quando o robô estiver próximo de estar sem energia, deve procurar um local para se recarregar ou parar em local seguro e indicar a falta de energia.
- Exploração: o robô pode mostrar um comportamento exploratório, andando sem rumo definido, de modo a mapear um ambiente, quando não há tarefas mais importantes designadas.

Nesta arquitetura para robô omni-direcional, cada módulo gera um *vetor velocidade*, composto de duas velocidades de translação ( $v_x, v_y$ ) e uma de rotação (?), que procura atingir o objetivo específico do módulo. Um módulo que evite choques geraria um vetor que afasta o robô do obstáculo detectado; um módulo de recarga geraria um vetor que direciona o robô até o ponto de recarga.

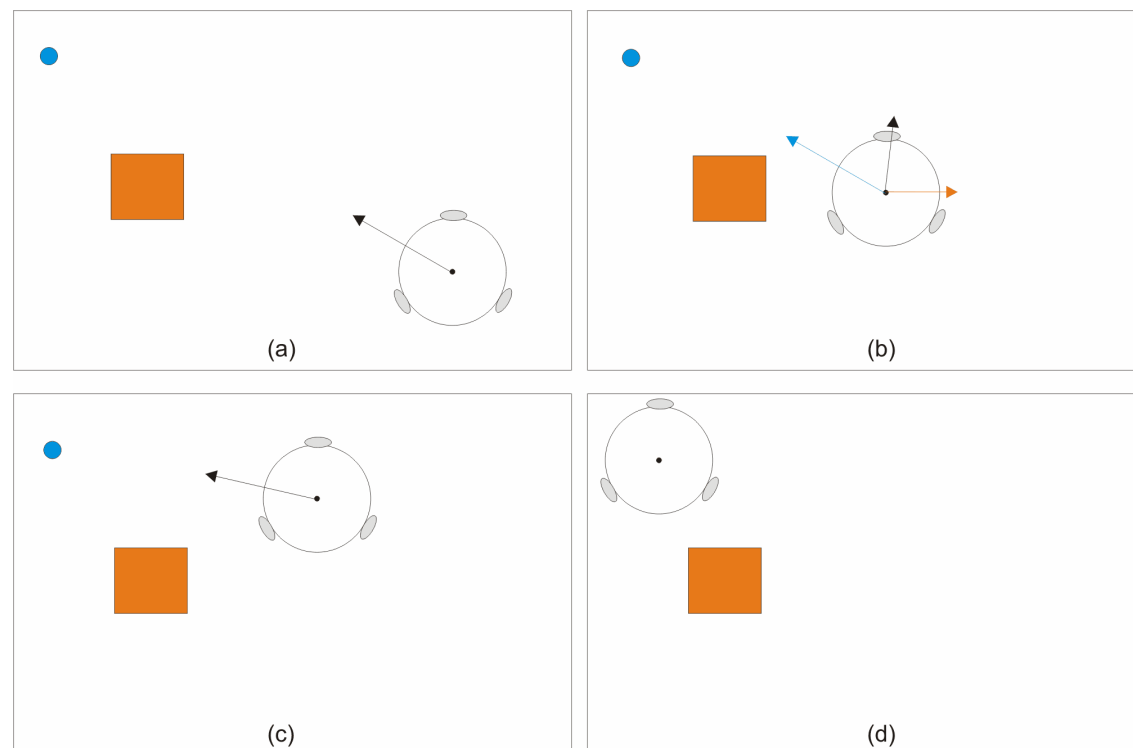
Uma arquitetura baseada em módulos comportamentais apresenta características de robustez e escalabilidade. Cada módulo opera de forma independente das demais, podendo, inclusive, rodar em processadores distintos. Assim, por exemplo, se o sistema de comunicação falhar e não houver novos comandos para o robô seguir, este não ficará parado, outros comportamentos garantirão a operacionalidade parcial do sistema. A escalabilidade surge uma vez que novos comportamentos podem ser adicionados ao sistema sem necessidade de grandes modificações nos módulos existentes, que é uma das especificações para o sistema.

### 2.2.3. Módulo de decisão

No módulo de decisão há a geração do vetor velocidade final que será passado para o módulo de controle. Esse vetor é produto dos diversos módulos comportamentais.

São duas as decisões que podem ser tomadas em um grupo de vetores de velocidade gerado pelos diversos módulos comportamentais: somar vetorialmente, de forma ponderada ou não, ou fazer com que um prevaleça sobre os demais. Vale ressaltar que se pode separar as saídas dos módulos comportamentais em diversos grupos, aplicando-se decisões parciais em cada grupo e, após, decidindo entre esses resultados.

A soma vetorial tem a meta de tentar unir dois ou mais objetivos distintos. O vetor resultante tenderá a levar o robô para uma posição que seja mais próxima dos vários objetivos, mesmo que não tão próximo quanto se apenas um, ou poucos, dos objetivos prevalecesse. A Figura 6 mostra como o desvio de obstáculos pode emergir da arquitetura, unicamente a partir da soma vetorial de dois comportamentos básicos: afastar-se de obstáculos e perseguir uma coordenada. Considera-se que o comportamento de afastar-se de obstáculos gera vetores que sempre apontam no sentido contrário ao obstáculo, com magnitude proporcional ao quadrado da distância, conforme a Teoria dos Campos Potenciais [LAT91]. O comportamento de perseguir uma coordenada sempre gera vetores que apontam para a coordenada desejada.



**Figura 6:** (a) o obstáculo (em laranja) está fora do alcance dos sensores; (b) o obstáculo está próximo, o robô reage criando um vetor que tenta evitar o obstáculo, mas ainda se aproximar do objetivo; (c) o obstáculo deixa de afetar o robô; (d) o robô atinge o objetivo.

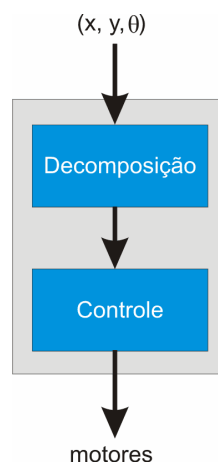
A decisão de fazer um comportamento prevalecer sobre os demais se aplica em casos emergenciais ou em situações especiais. Por exemplo, poderia haver um comportamento que fizesse o robô parar quando detectasse sobrecarga nos motores. Esse comportamento, quando ativo, deve prevalecer sobre os demais, já que é uma questão atípica que poderia ocasionar danos ao sistema.

Este módulo é o que mais difere da arquitetura *subsumption* original, já que se trata de um módulo centralizador. A decisão de incluí-lo vem da necessidade de se tratar de conflitos e incoerências eventuais nas decisões de cada módulo comportamental. Apesar da arquitetura *subsumption* apresentar formas de contornar esses problemas, através da inibição hierárquica de saídas, estas apresentam complexidades e problemas próprios. Um módulo centralizador simplifica o processo de resolução, ao custo de reduzir a robustez e paralelismo do sistema.

#### 2.2.4. Módulo de controle

Este módulo é responsável por receber um vetor velocidade para o corpo do robô e decompor este em velocidades angulares para cada roda. Além disso, é responsabilidade desta camada utilizar leis de controle de forma a atingir, de forma rápida, sem oscilações, a velocidade desejada e mantê-la pelo período que for determinado.

O fluxo de controle desta camada é explicitada na Figura 7.



**Figura 7: fluxo de controle do módulo de controle**

A decomposição é feita utilizando as equações 4.4, 4.5 e 4.6 citadas no capítulo anterior. O controle dos motores pode utilizar qualquer lei de controle, como PID [OGA00], ou, ainda, operar em malha aberta, em casos que o robô operará em ambientes suficientemente controlados.

### **3. Instanciação da arquitetura**

A arquitetura desenvolvida na seção anterior é genérica e flexível. Para uma aplicação prática, é necessário fazer a instanciação do problema, no caso para o futebol de robôs. Para tanto, devem ser especificados os módulos comportamentais e as restrições e exigências para os demais módulos da arquitetura.

#### **3.1. Módulos comportamentais**

A especificação de quais módulos comportamentais estarão presentes na arquitetura implementada é o que definirá o comportamento do robô como um todo, seu grau de autonomia e sua inteligência.

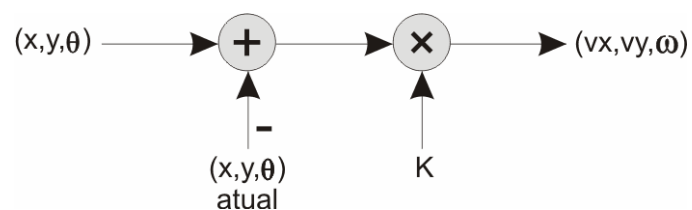
Para um jogo de futebol de robôs, tal como foi especificado na seção anterior, o robô deve poder ser direcionado até uma coordenada específica e tratar eventuais obstáculos que se encontrem durante o movimento. Essas ações podem ser implementadas utilizando dois módulos comportamentais, cada um responsável por uma das ações.

##### **3.1.1. Módulo de direção a uma coordenada**

Este módulo deve direcionar o robô a uma coordenada recebida do módulo de comunicação. Ao chegar à coordenada, deve parar e permanecer no ponto. Note-se que as coordenadas são passadas como valores relativos a um ponto fixo no ambiente, usualmente o ponto de partida do robô. Para que seja possível o deslocamento até o ponto desejado, o módulo deve ter conhecimento da posição do robô em cada

momento. Isto é, o módulo deve conhecer sua posição no espaço e atualizar seu conhecimento conforme houver locomoção.

A Figura 8 mostra o fluxo de informações no módulo. A entrada é uma coordenada objetivo, no formato de um vetor posição. O vetor posição é composto por três parâmetros: duas posições espaciais ( $x,y$ ) e uma orientação ( $\theta$ ). O módulo possui internamente o conhecimento de sua posição atual, com mesma referência do vetor recebido. A subtração do vetor posição recebido com a posição atual fornece um vetor *velocidade*, composto por três componentes de velocidade:  $v_x$ ,  $v_y$  e  $\omega$ . Este vetor é multiplicado por uma constante, de modo a normalizá-lo e passado então para o módulo de decisão. A normalização é necessária para que, no módulo de decisão, todos os vetores velocidades recebidos sejam tratados de forma igual. Existindo unicamente este módulo, o módulo de decisão faria com que o robô se dirigisse na direção e sentido especificados por  $v_x$  e  $v_y$ , com uma rotação de velocidade  $\omega$ . O módulo de direção a uma coordenada deve então manter atualizada a informação de sua posição corrente, de modo a cessar o movimento (gerando um vetor velocidade nulo) quando a posição desejada for atingida.



**Figura 8: fluxo de informações no módulo**

Deve-se notar que o comportamento de parar ao atingir a coordenada alvo é automaticamente realizado pelo processo de subtração de vetores. Quando a posição corrente for igual à posição desejada, o vetor velocidade resultante é nulo. Porém, o comportamento apresentado pelo robô é o de ter uma alta velocidade no início do movimento e tê-la reduzida a medida que se aproxima do alvo, o que, dependendo da situação, pode tornar o movimento por demais lento.

Para que o módulo saiba sua posição corrente, é necessário que este utilize algum método de atualização do seu conhecimento. Idealmente, um robô autônomo

deve saber sua posição corrente no ambiente unicamente utilizando sensores internos ou o próprio ambiente em que está inserido. Um método simples é basear-se na fidelidade do conhecimento sobre a velocidade, direção e sentido do robô, contando o tempo desde o início do movimento e inferindo a posição a partir dessas informações. Outro método, mais eficiente e bastante utilizado, é o da odometria. Este método consiste em medir o movimento das rodas para, então, inferir o deslocamento de posição do robô. Possui como grande desvantagem a imprecisão gerada, principalmente, pela derrapagem das rodas, que não geram movimento do robô mas que os sensores interpretam como se gerassem. Esse erro introduzido é cumulativo e, por isso, outras técnicas usualmente são utilizadas em conjunto de modo a reduzir o erro global.

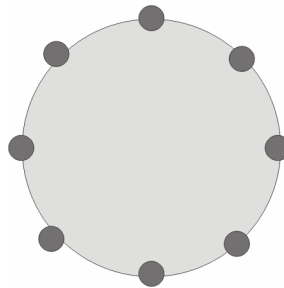
A técnica utilizada em conjunto com a odometria, neste caso, é aproveitar-se da visão global para atualizar regularmente a informação de posição atual do módulo. Isso é feito com o computador externo identificando a posição do robô pelo sistema de visão e enviando essa posição para o módulo. Dessa forma, o erro acumulado é zerado a cada atualização, evitando que cresça de forma a impedir o controle do robô.

### **3.1.2. Módulo de evasão de obstáculo**

Este módulo deve gerar um movimento na direção e sentido contrário ao obstáculo detectado, de modo que o robô tenha um comportamento de “fuga” de um obstáculo. Apesar deste não ser um comportamento inteligente, conforme foi visto na seção anterior, espera-se que, em conjunto com os demais módulos, surjam comportamentos mais inteligentes.

Este módulo depende principalmente de sensores de obstáculos, podendo ser do tipo de proximidade ou de distância. A utilização de sensores que permitam a medição da distância do obstáculo é interessante, já que permite utilizar comportamentos mais complexos, que gerem uma tendência do robô se afastar de obstáculos que seja tão forte quanto a proximidade destes, de forma semelhante a Teoria de Campos Potenciais [LAT91].

Para que o módulo seja eficiente, diversos sensores devem estar espalhados pelo corpo do robô e suas posições devem ser conhecidas pelo módulo. A Figura 9 mostra uma sugestão de configuração para os sensores. É importante que os sensores cubram todos os lados do robô, uma vez que a movimentação omni-direcional não possui uma “frente” propriamente definida.

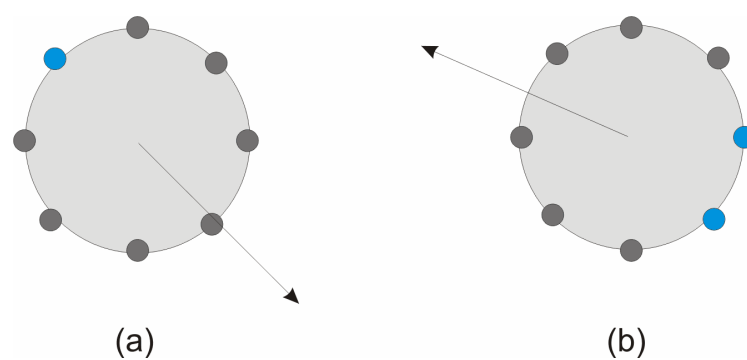


**Figura 9: sugestão de configuração para os sensores**

O módulo deve constantemente verificar cada sensor e, quando um deles indicar a presença de um obstáculo, deve gerar um vetor velocidade que direcione o robô no sentido contrário àquele apontado pelo sensor, como mostrado na

Figura 10 (a). Quando dois ou mais sensores indicarem presença de obstáculo, deve-se efetuar a soma vetorial dos vetores gerados de modo a gerar o comportamento final do módulo, como exemplificado na

Figura 10 (b). Em uma situação limite, cada sensor pode representar um módulo comportamental independente, ligado diretamente ao módulo de decisão. Porém, o grande número de sensores tornaria a arquitetura confusa sem necessidade e adota-se que os módulos de comportamentos semelhantes são agrupados dentro de um único módulo.

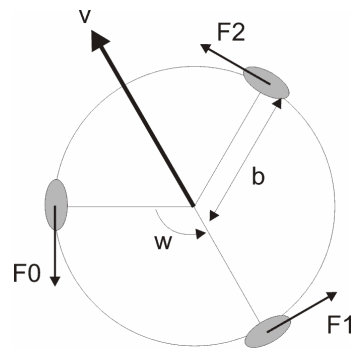


**Figura 10: (a) vetor que direciona o robô para se afastar do obstáculo; (b) soma dos vetores resultantes de dois sensores**

### **3.2. Módulo de controle**

A precisão do módulo de controle, ao controlar as velocidades dos motores, é decisiva na precisão dos movimentos do robô como um todo. Em um jogo de futebol de robôs, não é necessário que haja um posicionamento extremamente preciso, já que o ambiente é extremamente dinâmico e outros erros (como os de identificação de objetos pela câmera global em tempo real) praticamente anulam os benefícios que um controle preciso trariam. Porém, apesar de talvez não oferecer problemas na precisão, o controle em malha aberta pode oferecer dificuldades para o controle da trajetória.

A principal tarefa do módulo de controle é converter o vetor velocidade recebido em velocidades para as rodas. Essa conversão é dependente da configuração utilizada para obter a movimentação omni-direcional. Neste projeto, esta é obtida com rodas omni-direcionais dispostas em ângulos de 120 graus entre seus eixos, como no diagrama da Figura 11. Rodas omni-direcionais são rodas especiais que possuem rolamentos de giro livre transversais ao eixo de rotação normal, de forma a reduzir o atrito quando a roda é “arrastada”. Cada roda possui seu próprio motor, ligado ao seu eixo. Assim, cada uma tem controle independente e impõe uma força em uma única direção, com dois sentidos possíveis. A composição dessas diferentes forças acaba por gerar uma força resultante sobre o corpo do robô, que dá exatamente a direção e sentido de movimentação do robô, além de um possível movimento de giro em torno do centro de massa do corpo.



**Figura 11: disposição das rodas**

Analisando a cinemática do corpo, na Figura 11 e de [PPR1] temos que:

- $F_0, F_1, F_2$  – vetores unitários de força de cada roda,
- $b$  – distância de cada roda até o centro de massa do corpo,
- $w$  – velocidade angular do corpo,
- $v$  – vetor velocidade do corpo.

Sendo, ainda:

- $r$  – raio da roda,
- $w_0, w_1, w_2$  – velocidades angulares das rodas.

Colocando a origem no centro de massa do robô, temos:

$$F_0 = (-1, 0) \quad (4.1),$$

$$F_1 = (1/2, -\sqrt{3}/2) \quad (4.2),$$

$$F_2 = (1/2, \sqrt{3}/2) \quad (4.3).$$

A partir dessas informações, temos as equações que governam os movimentos do robô:

$$w_0 = (v \cdot F_0 + b \cdot w) / r \quad (4.4),$$

$$w_1 = (v \cdot F_1 + b \cdot w) / r \quad (4.5),$$

$$w_2 = (v \cdot F_2 + b \cdot w) / r \quad (4.6).$$

Isto é, a partir de um vetor velocidade e uma velocidade angular do corpo desejados, obtém-se a decomposição em velocidades angulares para cada roda. Essas

velocidades angulares devem ser convertidas para tensões efetivas para os motores, de modo a gerar fisicamente o movimento.

### **3.3. Módulo de comunicação**

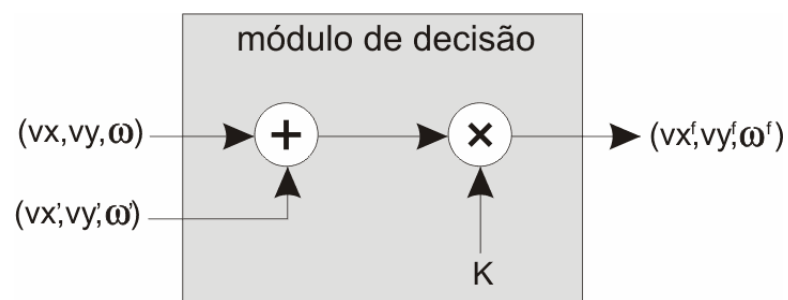
A comunicação no futebol de robôs deve ser feita através de uma conexão *wireless*. No projeto Furgbol, essa conexão é unidirecional, sendo o transmissor o computador externo e os robôs os receptores. Assim, o módulo de comunicação deve implementar um sistema de recepção dos dados do computador externo, repassando-os para os demais módulos.

Os dados recebidos do computador externo são comandos para o robô. São dois comandos possíveis, sendo um do tipo “vá para”, especificando uma coordenada para onde o robô deve se dirigir e outro do tipo “setar posição”, que ajusta o conhecimento interno sobre a posição atual do robô. O segundo comando é necessário para inicializar o ponto de partida do robô e para, eventualmente, reduzir os erros no posicionamento causados pela odometria.

O módulo deve indicar os módulos comportamentais da chegada de um comando, para que estes o tratem da forma correta. Não é responsabilidade do módulo de comunicação realizar quaisquer interpretações sobre os comandos recebidos.

### **3.4. Módulo de decisão**

Com base nos módulos comportamentais previstos, o módulo de decisão tem a simples tarefa de realizar uma soma vetorial de suas entradas e enviar o resultado para o módulo de controle. A Figura 12 mostra o fluxo de informações nesse módulo.



**Figura 12: fluxo de informações no módulo de decisão**

Nenhum sistema de inibição ou prevalectimento das entradas é utilizado, já que ambas estão em um mesmo nível, isto é, possuem mesma hierarquia e autoridade no controle do robô.

#### 4. Implementação

De modo a validar a arquitetura proposta, um protótipo de robô móvel omnidirecional foi desenvolvido. Esse protótipo visa o futebol de robôs em específico, porém mantendo-se genérico o suficiente para outras aplicações.

A implementação foi altamente baseada no uso de microcontroladores. Outras formas de implementação possíveis se baseiam em FPGA [KON98], circuitos analógicos [ALV1] e microcomputadores. A opção por utilizar-se microcontroladores se deu por fatores como preço, disponibilidade e facilidade de uso.

Dividiu-se a implementação em duas placas físicas, que se comunicam por um barramento. Mais placas podem ser adicionadas ao sistema com facilidade, mantendo a modularidade da arquitetura. A primeira placa é responsável pela etapa de potência para os motores e também pelo controle destes, implementando internamente o módulo de controle proposto. Esta é chamada “placa de controle”. A segunda placa é

responsável por ler os sensores, comunicar-se com o computador externo e gerar trajetórias (seqüências de coordenadas) para serem passadas à placa de controle. Isto é, implementa os módulos comportamentais, de decisão e de comunicação da arquitetura. Esta é chamada “placa de deliberação”.

Para se ter um robô funcional, também foram especificados a estrutura, motores, rodas e alimentação. A seguir cada uma das placas e dos itens especificados serão vistos com mais detalhes.

#### 4.1. Microcontroladores

Os microcontroladores são a base da implementação do robô, e uma visão geral destes é fundamental para compreender como o resto da implementação é estruturada.

Dois microcontroladores são utilizados na implementação, ambos da família PIC, da fabricante Microchip. É uma família que tem ganho muito destaque e uso atualmente, devido a sua robustez, eficiência e disponibilidade. Tratam-se de microcontroladores RISC rodando sob um arquitetura Harvard [SIU00]. A utilização desta família se deu devido aos fatores disponibilidade, preço e a já utilização desta em outros projetos na Universidade, o que reduziu a curva de aprendizado necessária para utilizar corretamente os microcontroladores.

Ambos microcontroladores são da classe 16FXXX, que especifica microcontroladores de 8 bits com memória do tipo Flash. Esse tipo de memória é particularmente útil por permitir rápida programação, não necessitando utilizar luz ultravioleta para apagá-las como no caso das EPROMS. Esta classe é a mais comum na família PIC e de maior disponibilidade.

O primeiro microcontrolador é utilizado para implementar o módulo de controle. Para este, foi escolhido o modelo PIC 16F84, que contém todas as características da família, mencionadas anteriormente e, mais especificamente, é um microcontrolador com 20 pinos no total, 13 pinos configuráveis de entrada ou saída, 1K palavras de memória Flash, 68 bytes de memória EEPROM, 1 timer e suporte a *clocks* de até 10MHz, na versão utilizada.

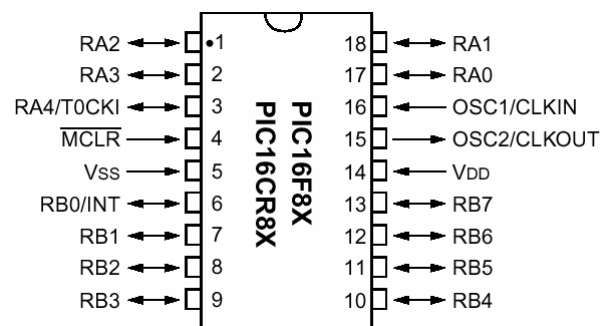


Figura 13: PIC 16F84

O segundo microcontrolador é um PIC 16F877. Mais completo que o 16F84, possui 40 pinos, 8K palavras de memória Flash de instrução, 368 bytes de memória RAM e 256 bytes de memória EEPROM, 33 pinos configuráveis de entrada ou saída, 3 timers, conversor analógico / digital de 10 bits, USART integrada e suporte a *clocks* de até 20MHz.

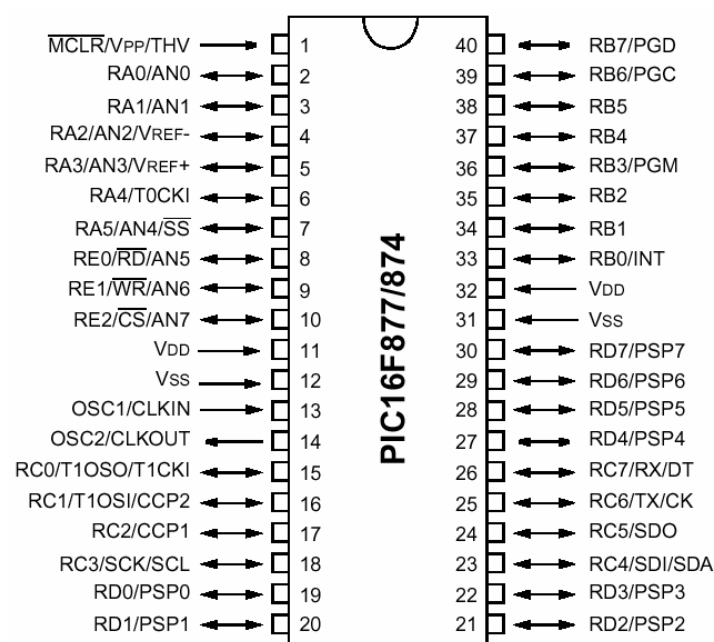


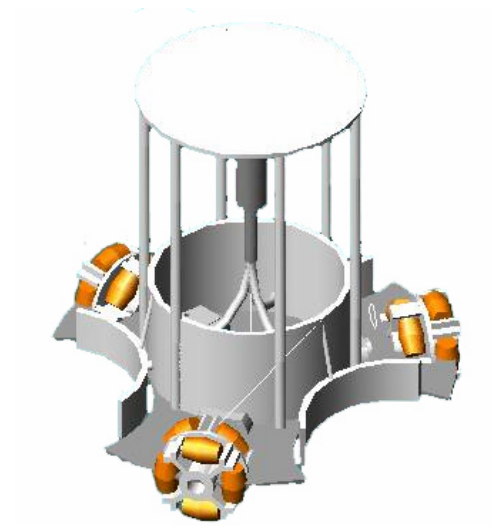
Figura 14: PIC 16F877

A família PIC possui diversos aplicativos para auxiliar na programação. A programação foi realizada em linguagem C, utilizando-se o compilador CC5X, da empresa noruega *B Knudsen Data*, integrado no ambiente de desenvolvimento MPLAB da própria Microchip. O programador, utilizado para passar o *software*

compilado para o microcontrolador, foi implementado baseando-se no programador especificado em [LAT91].

#### 4.2. Estrutura

A etapa de estrutura envolve a criação da base mecânica e a especificação e agregação dos motores e rodas. Esta etapa é crucial na etapa de construção de robôs móveis, porém em robôs móveis omni-direcionais esta é ainda mais crítica, já que o alinhamento dos motores na base são fundamentais para o correto controle. Esta etapa não é escopo deste trabalho, tendo sido realizado por outros integrantes do projeto Furgbol. A Figura 15 mostra uma modelagem da concepção inicial da estrutura do robô.



**Figura 15: concepção inicial da estrutura do robô**

Três motores ligados a três rodas são necessários para compor a estrutura de locomoção do robô. Esses motores devem possuir torque suficiente para movimentar o corpo totalmente carregado, já com todas placas, baterias e sensores. Devem também possuir velocidade adequada para o jogo de futebol de robôs.

Os motores utilizados foram servo-motores adaptados. Servo-motores são motores DC que já agregam caixa de redução e eletrônica para controle de posição, controle este feito através de pulsos digitais. Esse tipo de motor usualmente é utilizado

para controle de posição, sendo que não oferecem possibilidade para controle de velocidade ou mesmo rotação contínua. A escolha destes se deu por fatores de disponibilidade, facilidade e preço, já que o robô a ser implementado é apenas um protótipo que visa validar a arquitetura desenvolvida, sem se preocupar em excesso com velocidade final do robô ou capacidade de carga. O motor pode ser alimentado com até 9 volts, tendo nessa faixa uma velocidade máxima, na saída do redutor, de 2 rotações por segundo.

Para utilização nesta implementação, toda eletrônica do servo-motor foi removida, mantendo-se apenas a caixa de redução, devidamente adaptada para permitir rotação contínua. Desse modo, é possível controlar a velocidade de rotação aplicando-se uma tensão diretamente nos terminais dos motores.

Para o movimento omni-direcional, utilizou-se as rodas omni-direcionais da *North American Roller Products (NARP)*, em seu modelo de 4 centímetros com rolamentos de plástico, que pode ser observada na Figura 15. A escolha destas se deu pela sua consagração nesse uso em outros robôs omni-direcionais [PPR1] [CAR01].

### **4.3. Placa de controle**

A etapa de controle e potência são duas etapas distintas, que foram unidas em uma mesma placa na implementação de modo a reduzir o tamanho total do circuito final e otimizar o desempenho de ambas as etapas.

A etapa de potência tem o objetivo de converter sinais de controle, em forma digital, gerados pela etapa de controle em tensões e correntes efetivas para serem aplicadas aos motores. A etapa de controle é a implementação efetiva do módulo de controle e realiza todas as funções deste: receber um vetor de velocidade, decompô-lo e gerar sinais de controle para os motores.

A placa compõe-se do microcontrolador PIC 16F84 e de uma etapa de potência para os motores. O microcontrolador é responsável por ler novos vetores de posição e gerar os sinais PWM necessários para executar o comando. Os sinais PWM,



tensão média sobre o motor. Essa tensão média será tão alta quanto for a largura dos pulsos. Dessa forma, ao variar a largura dos pulsos aplicados estaremos variando a tensão sobre o motor e, conseqüentemente, sua velocidade.

Dois fatores estão envolvidos no controle de um motor DC por PWM: resolução e freqüência da portadora. A resolução é o número de passos possíveis (níveis de modulação) entre uma largura nula de pulso até uma largura máxima. Essas larguras são especificadas em percentagens da largura máxima. A resolução, por sua vez, é dada usualmente em bits. Um PWM de  $n$  bits possui  $2^n$  níveis. Como exemplo, considere um PWM de 4 bits, resultando em 16 níveis: 0%, 6.25%, 12.5%, 25% e assim por diante até 100%. Quanto maior a resolução maior o controle que se obtém sobre a velocidade do motor. Porém, o crescimento não é linear e há um limite de resolução acima do qual o motor é insensível a variações. Esse limite varia com diversos fatores, entre eles o tipo de motor, fabricante, carga, etc.

A freqüência da portadora determina o período com que um novo pulso será gerado. Uma freqüência muito baixa dará tempo ao motor para ter a tensão no seu terminal decaída, causando uma variação e oscilação na velocidade mesmo com uma largura fixa de pulso. A freqüência também determina a eficiência com que o motor aproveita a energia fornecida através do PWM. A freqüência ideal também varia com as características do motor sendo usado e, de modo geral, deve ser tão alta quanto possível, usualmente na faixa de KHz. Valores típicos comerciais estão acima de 20KHz, em especial por ser uma freqüência acima da faixa de sensibilidade do ouvido humano, evitando que se escute o motor zumbindo com o chaveamento.

Também deve-se levar em consideração, ao implementar sistemas baseado em PWM, que, diferentemente do controle por tensão contínua, a velocidade do motor não cresce linearmente com os níveis de PWM aplicados. Isso pode ser visto traçando a curva de tensão RMS para diversos níveis de PWM ou através de simples experimentos. Portanto, ao menos que se utilize algum sistema que compense essa não-linearidade, como um controle PID, deve-se ter consciência desse fator no momento da implementação.

As vantagens de se utilizar PWM são várias. A principal é a possibilidade de se gerar tensões analógicas sobre motores a partir de um único pino do microcontrolador<sup>2</sup>. Dessa forma, poupam-se pinos de entrada e saída, reduzindo o tamanho total do circuito e o custo do microcontrolador. Adicionalmente, a utilização de PWM oferece uma melhor eficiência na utilização da energia, já que ao motor não se fornece constantemente uma corrente. A principal desvantagem no uso desta técnica está na dificuldade de controle, dada a não-linearidade apresentada, especialmente em baixas velocidades.

#### 4.3.2. Recebendo um vetor de posição

O processo tem início com o recebimento de um vetor de velocidade da placa de deliberação. Este vetor é recebido de forma assíncrona e, enquanto um novo vetor não é especificado, o sistema mantém operando o último vetor recebido. Isto é, ao enviarmos um vetor que especifique movimento para frente, o robô manter-se-á andando para frente até que um novo vetor (informando para parar, por exemplo) seja recebido.

O recebimento deste vetor se dá através de dois pinos do microcontrolador, com funções de notificação e dados. O microcontrolador constantemente verifica o pino de notificação, aguardando uma mudança de estado. Quando este ocorre, significa que será iniciado o processo de comunicação. A partir daí, os bits compondo o vetor de posição chegam serialmente pelo pino de dados, de forma síncrona com o pino de notificação. Isto é, o pino de notificação muda de estado para validar cada bit no pino de dados.

Estes bits são então organizados para formarem o vetor de posição, armazenado em áreas da memória. São três parâmetros que compõem o vetor de posição, cada um com 8 bits. Os 24 bits são passados conjuntamente, sendo o

---

<sup>2</sup> Com a devida *interface* entre os dois.

algoritmo de recepção responsável por organizá-los. A Figura 17 mostra o trecho do código responsável por efetuar o recebimento de um byte.

```

char receive_data()
{
    char c;

    while (clk==1);
    c.0 = data;
    while (clk==0);
    c.1 = data;
    while (clk==1);
    c.2 = data;
    while (clk==0);
    c.3 = data;
    while (clk==1);
    c.4 = data;
    while (clk==0);
    c.5 = data;
    while (clk==1);
    c.6 = data;
    while (clk==0);
    c.7 = data;
    return c;
}

```

**Figura 17: trecho do código para recebimento de um byte**

### 4.3.3. Decompondo o vetor de posição

A próxima etapa no processo está em, uma vez recebido um vetor de posição, decompô-lo para gerar as velocidades angulares para cada motor. Esta etapa implementa as equações 4.4, 4.5 e 4.6 vistas anteriormente. Porém, o microcontrolador em conjunto com o compilador utilizado apenas trabalha com matemática inteira. Portanto, as equações precisam ser convertidas para uso com inteiros ao invés de números reais.

Essa conversão acarreta numa perda de precisão. Porém, a proporção entre os valores gerados se mantém e, mesmo que não se tenha um controle preciso sobre a velocidade final, não há perdas significativas, para os objetivos almejados, na orientação de locomoção do robô como um todo. A Figura 18 mostra o trecho de código responsável por realizar a decomposição.

```

void movimenta(int16 vx, int16 vy, int16 omega)
{
    int16 omega1;
    int16 omega2;
    int16 omega3;
    int16 bb;
    int16 bx;
    int16 by;
    int16 rot;

    bb = 86*vy;
    rot = 50*omega;

    omega1 = -100*vx;
    omega2 = 50*vx - bb;
    omega3 = 50*vx + bb;

    omega1 = omega1 + rot;
    omega2 = omega2 + rot;
    omega3 = omega3 + rot;

    omega1 = omega1 / 100;
    omega2 = omega2 / 100;
    omega3 = omega3 / 100;

    omega1=conv_vel(omega1);
    omega2=conv_vel(omega2);
    omega3=conv_vel(omega3);

    set_pwm(1, omega1);
    set_pwm(2, omega2);
    set_pwm(3, omega3);
}

```

**Figura 18:** trecho do código que decompõe o vetor de posição

#### 4.3.4. Gerando os sinais de controle

Uma vez encontrada a velocidade angular de cada motor, estas precisam ser mapeadas em níveis de PWM capazes de gerarem tais velocidades. A não-linearidade do controle por PWM deve ser levada em consideração nesta etapa.

Para gerar o mapeamento, é necessário conhecer o comportamento do motor sendo utilizado. Para tanto, aplica-se diversos níveis de PWM a este e verifica-se a velocidade efetiva na saída, traçando-se a curva de resposta do motor. Os níveis PWM

para o teste podem ser gerados diretamente no microcontrolador ou, ainda, utilizando um gerador de sinais que permita o controle do *duty-cycle* da onda gerada. Nos testes realizados, utilizou-se o microcontrolador para essa função. A velocidade de saída foi medida utilizando-se um potenciômetro de giro contínuo ligado ao eixo de saída do motor, obtendo-se a tabela presente na Tabela 1 e graficada na

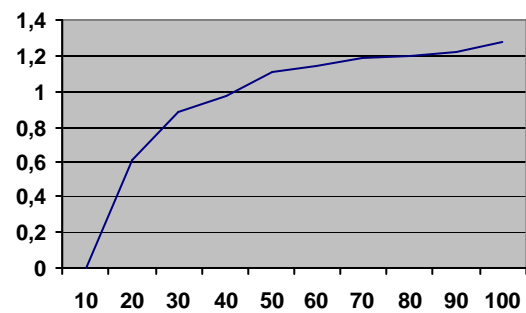


Figura 19.

Nível PWM (%)	10	20	30	40	50	60	70	80	90	100
Velocidade (Hz)	0	0,62	0,89	0,98	1,11	1,14	1,19	1,20	1,22	1,28

Tabela 1: níveis de PWM versus velocidade do motor

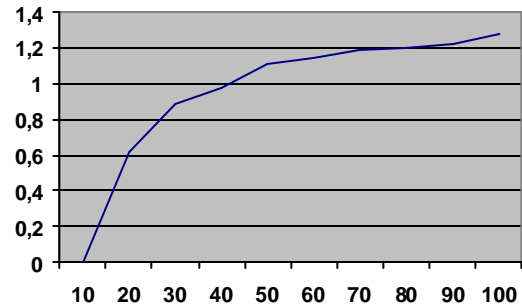


Figura 19: gráfico da Tabela 1

A implementação do tratamento da não-linearidade presente foi feita através de uma tabela em *software*. De modo a poupar memória no microcontrolador, apenas os pontos presentes na Tabela 1 foram mapeados diretamente. Pontos entre estes são

interpolados de forma linear. A Figura 20 mostra o trecho de código da função responsável por realizar o mapeamento. A saída da função é o nível PWM necessário para gerar a velocidade indicada.

```
int16 conv_vel(int16 vel)
{
    int16 pwm;
    char i;

    if (vel<0)
    {
        vel=-vel;
        i=1;
    }
    else
        i=0;

    if (vel>0)
        vel=vel+28;

    if ((vel<=128) && (vel>122)) {
        pwm = vel - 28;
    }

    if ((vel<=122) && (vel>119)){
        pwm = 90 - 122 + vel;
    }

    if ((vel<=119) && (vel>114)){
        pwm = 80 - 119 + vel;
    }

    if ((vel<=114) && (vel>111)){
        pwm = 60 - 114 + vel;
    }

    if ((vel<=111) && (vel>98)){
        pwm = 50 - 111 + vel;
    }

    if ((vel<=98) && (vel>89)){
        pwm = 40 - 98 + vel;
    }

    if ((vel<=89) && (vel>62)){
        pwm = 33 - 89 + vel;
    }

    if ((vel<=62) && (vel>50)){
        pwm = 20 - 62 + vel;
    }

    if (vel <= 50)
        vel = vel / 3;

    if (i==1)
        pwm=-pwm;
}
```

```

    return pwm;
}

```

**Figura 20: código da função de mapeamento do PWM**

Com o nível de PWM de cada motor, basta uma simples rotina que gere os sinais com tais níveis. A Figura 21 mostra o trecho de código responsável por gerar os sinais PWM.

```

#pragma bit out1    @ PORTB.0
#pragma bit out2    @ PORTB.2
#pragma bit out3    @ PORTB.4

...

int16 pwm1,pwm2,pwm3;

...

while (1)
{
    for (i=0;i<100;i++)
    {
        if (i>=pwm1) out1 = 0; else out1 = 1;
        if (i>=pwm2) out2 = 0; else out2 = 1;
        if (i>=pwm3) out3 = 0; else out3 = 1;
    }
}

```

**Figura 21: trecho de código para geração dos sinais PWM**

Ao se gerar sinais PWM digitalmente em um microcontrolador, há sempre um custo computacional associado, o que impossibilita que se tenha frequências e resoluções altas ao mesmo tempo, com uma capacidade de processamento limitada. Uma troca entre esses dois parâmetros é necessária. Para o projeto, especificou-se uma frequência de pelo menos 1KHz, com a maior resolução que se conseguisse após a implementação. A escolha da frequência se deu principalmente pela análise de experiências empíricas realizadas em [PWM1].

#### 4.3.5. Movimentando os motores

Devido a características elétricas, os sinais PWM (digitais) não podem ser aplicados diretamente aos respectivos motores. Uma etapa de potência é necessária para que se forneça a potência necessária para movimentar os motores.

Essa etapa de potência constitui-se de Pontes H controlados por PWM. Uma Ponte H é uma configuração de transistores capaz de amplificar sinais digitais, fornecendo a corrente e tensão necessárias para movimentar um motor em ambas as direções. A Figura 22 mostra uma Ponte H típica. A direção do motor é definida pelo modo com que os transistores são polarizados. A velocidade do motor é definida pela tensão aplicada na Ponte H e, portanto, não é controlada diretamente por este circuito.

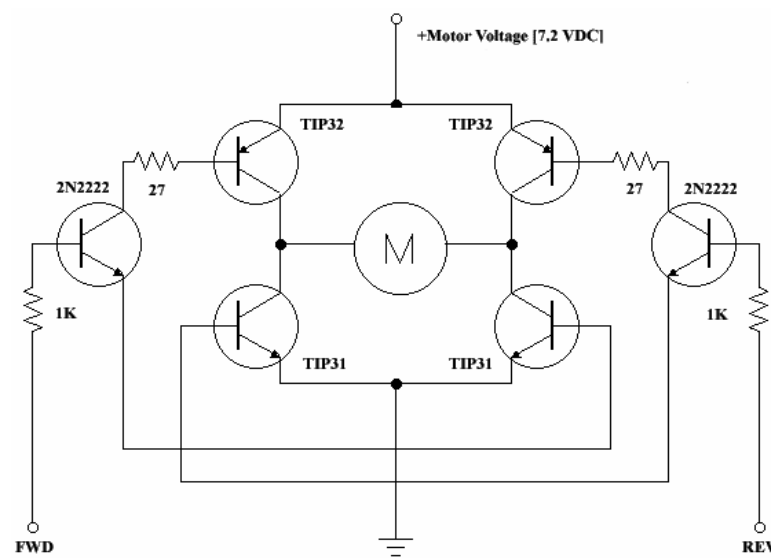


Figura 22: esquemático de uma Ponte H

O controle da velocidade provém do PWM. O sinal PWM é uma onda quadrada e, aplicada aos terminais de habilitação da Ponte H ocasiona na replicação do sinal sobre o motor, porém com os níveis de tensão amplificados. A aplicação de uma onda quadrada de tensão sobre o motor faz com que uma tensão média apareça sobre seus terminais, tensão esta proporcional ao nível de PWM aplicado, permitindo o controle da velocidade através dos sinais PWM.

A Ponte H utilizada é o circuito integrado L293D. Trata-se de duas pontes H encapsuladas e, portanto, dois CIs são necessários para controlar os três motores. A Figura 23 mostra o diagrama interno do L293D e a Figura 24 seu encapsulamento.

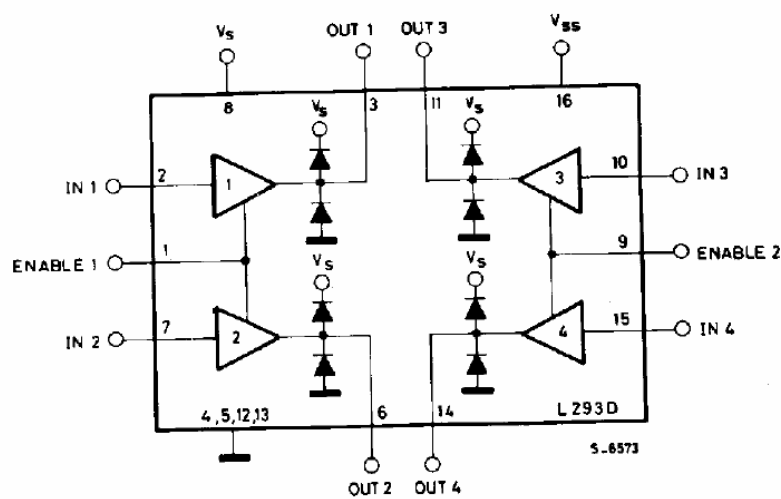


Figura 23: diagrama do L293D

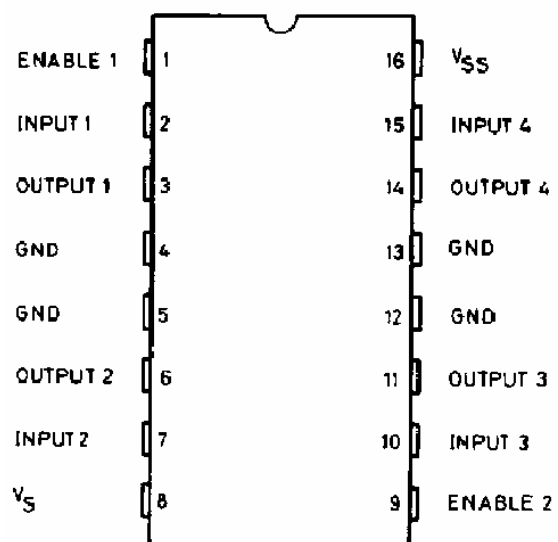


Figura 24: encapsulamento do L293D

São duas formas de controlar uma Ponte H utilizando-se PWM: controle por anti-fase e controle por sinal-magnitude.

No controle por anti-fase, apenas um sinal é necessário para controlar a direção e velocidade de rotação. Especifica-se que o motor deve permanecer parado quando

da aplicação de um sinal com largura de 50%. Abaixo dessa largura, há rotação em uma direção, sendo 0% a velocidade máxima nessa direção. Acima de 50%, há rotação na outra direção, sendo 100% a velocidade máxima. A grande vantagem desse método é a possibilidade de se utilizar um único pino do microcontrolador. A desvantagem, naturalmente, é a perda de resolução.

No controle por sinal-magnitude, dois sinais são necessários para controlar a Ponte H. Um dos sinais especifica a direção de rotação e o outro a velocidade. Há um aproveitamento total da faixa de resolução, porém ao custo de serem necessários dois pinos do microcontrolador por motor. Esta é a técnica utilizada para o controle nesta implementação.

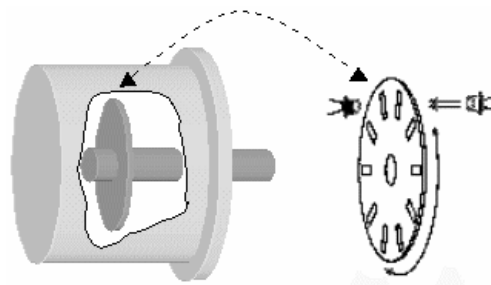
#### 4.3.6. Controlando os motores

O controle dos motores implementado neste pré-protótipo é diretamente em malha aberta, com níveis PWM mapeados diretamente a velocidades de motor, sem realimentação para correções. Esta não é a abordagem ideal e apenas foi adotada pela indisponibilidade de recursos para efetuar o controle em malha fechada. Porém, a placa de controle prevê a implementação de malha fechada, com mínimas modificações.

Para o controle em malha fechada, três pinos do microcontrolador são reservados para a leitura de sensores de rotação, ligados aos motores. Os sensores de rotação previstos para serem utilizados são sensores capazes de gerarem pulsos de frequência proporcional a velocidade de rotação do motor. Para essa função, existem duas formas mais conhecidas para realizar essa medição: Sensores de efeito Hall [HAL1] e *encoders* incrementais ópticos. Foi prevista na implementação a utilização deste último.

*Encoders* ópticos são sensores capazes de gerar pulsos a partir da interrupção de um feixe de luz, usualmente infra-vermelha. É o princípio utilizado nos *mouses* optomecânicos, onde a rolagem de uma esfera na parte inferior causa movimento de rotação de duas hastes internas, uma vertical e outra horizontal. Essas hastes possuem nas suas

extremidades um disco com diversas perfurações regulares. Um emissor infra-vermelho é colocado em um lado do disco e um receptor do outro. Dessa forma, conforme o disco gira, o emissor tem seu feixe interceptado pelas áreas opacas do disco ou liberado nas perfurações. O receptor, por sua vez, vê um feixe pulsado de luz, convertendo isso em um sinal também pulsado. Medindo-se a frequência dos pulsos, pode-se inferir a velocidade de rotação da haste e, também, do movimento do *mouse*. A Figura 25 mostra a operação simplificada de um encoder óptico. O encoder óptico desse tipo é dito incremental devido a sua incapacidade de dizer a posição angular do motor diretamente, apenas registrando o quanto o motor se moveu desde o início da geração dos pulsos, incrementalmente.



**Figura 25: encoder óptico típico**

A implementação prevista é uma variação do encoder óptico comum. Um emissor e um receptor infra-vermelhos são utilizados, mas são dispostos de forma paralela e, no local do disco perfurado, é utilizado um disco refletor interrompido de forma regular, conforme mostrado na Figura 26. O emissor e receptor são colocados de forma próxima a esse disco, de modo que o receptor apenas receba a luz do emissor quando houver reflexão no disco. Como as áreas pretas absorvem o infra-vermelho, nestas o receptor terá sua recepção interrompida, gerando os pulsos em frequência proporcional a rotação do disco.

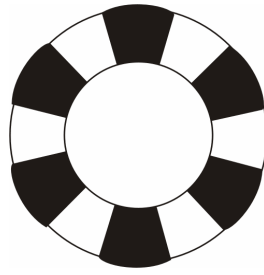


Figura 26: disco refletor para encoder

Sensores da fabricante Hamamatsu foram escolhidos. Cada sensor é um encapsulamento que já inclui um emissor e um receptor infra-vermelho, contendo também toda a eletrônica para amplificação dos sinais, bastando alimentá-lo e retirar os pulsos de um pino de saída. A Figura 27 mostra seu diagrama de ligação típico. Este sensor foi escolhido devido a seu pequeno tamanho (menos de 2 centímetros), necessário para ser colocado próximo as engrenagens do motor.

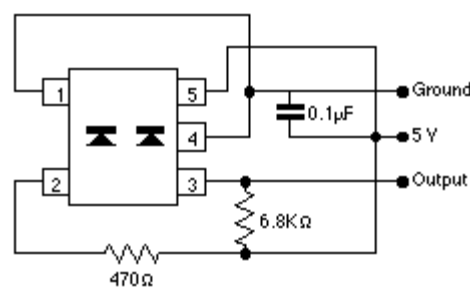


Figura 27: diagrama de ligação do sensor Hamamatsu

Para o disco, gerou-se em um programa de desenho técnico o padrão, que foi impresso, recortado e colado sobre uma das engrenagens do motor. Gerou-se um padrão com 6 listras, que geram seis pulsos por revolução. Em linhas gerais, quanto maior o número de pulsos gerados por revolução mais precisão de controle pode ser obtida, já que é possível inferir com maior precisão a posição do motor e sua velocidade. *Encoders* comerciais típicos são capazes de gerar de 100 a 1000 pulsos por revolução [ENC1]. O pequeno número de pulsos gerados é devido a incapacidade do sensor escolhido de distinguir faixas de larguras inferiores.

No motor, apenas uma engrenagem oferece a possibilidade para colocação do disco e do sensor, por possuir espaço suficiente em sua volta. Esta engrenagem possui uma redução de 10:1 em relação a roda. Ou seja, a cada giro da roda há dez giros desta engrenagem. Como a roda é o efetuator final que se deseja controlar, essa redução se traduz em um aumento da resolução do encoder utilizado, passando a ter 60 pulsos por revolução da roda. Isso equivale a um controle com resolução de 6 graus, insuficiente para se ter precisão na posição, mas capaz de fornecer boa informação sobre a velocidade. A Figura 28 mostra o disco já no motor.



**Figura 28:** disposição do disco de encoder no motor

Apenas foi especificado um sensor por motor. Isso permite unicamente ter informações sobre o deslocamento do eixo e sua velocidade, mas não é possível saber qual o sentido da rotação. Esse problema pode ser resolvido de duas formas: incluindo-se mais um sensor em cada motor ou confiando que o sentido que o motor foi instruído a girar realmente é o sentido de giro real. A primeira opção oferece a vantagem da inclusão de mais um sensor aumentar em até quatro vezes a resolução total do encoder [MAX00]. Porém, mais pinos do microcontrolador são necessários e uma lógica externa é frequentemente necessária para decodificar os sinais vindos dos dois sensores. A segunda opção é simples de ser implementada e, na maioria dos casos, realmente o sentido de giro do motor é aquele que lhe foi comandado.

O sinal proveniente de cada sensor é dirigido a um pino do microcontrolador, que deve constantemente verificar os pulsos que chegam neste, registrando em

contadores internos o número de pulsos. O valor nesses contadores, em qualquer momento, indica quantos pulsos ocorreram desde o início da contagem, registrando o quanto a roda se movimentou, em incrementos de 6 graus. Ao se verificar o conteúdo desses registradores de forma periódica, a diferença entre a leitura atual e a anterior, dividida pelo período de verificação, indica a velocidade de rotação da roda.

Para que não se perca nenhum pulso, é imprescindível que o microcontrolador dê atenção a cada um dos pinos em uma taxa superior a taxa de Nyquist [HAY01], ou seja, superior a duas vezes a frequência máxima dos pulsos. A frequência máxima é dada por:

$$f_{\max} = N * V_{\max} \quad ,$$

onde  $N$  é o número de pulsos por revolução gerados pelo encoder e  $V_{\max}$  é a velocidade máxima de rotação do disco do encoder (em rotações por segundo). Neste caso, teremos uma frequência máxima de 120 Hz. O microcontrolador, portanto, deve ser capaz de verificar cada um dos pinos de entrada dos encoders a uma taxa superior a 240 Hz, o que significa que a verificação deve ser feita a cada 4 ms, aproximadamente.

Após ler as informações de posição e velocidade da roda, o microcontrolador deve executar um algoritmo que implemente uma lei de controle, como a PID [OGA00]. A saída desta lei de controle será um nível de PWM a ser aplicado ao motor, corrigindo eventuais desvios da velocidade desejada. Adicionalmente, as informações provenientes dos encoders podem ser utilizados para realizar o controle de posição do robô por odometria, conforme foi explicado em seção anterior.

#### **4.4. Placa de deliberação**

A placa de deliberação tem esse nome por ser a etapa “inteligente” do robô. É de responsabilidade desta:

- Ler e interpretar os sensores;
- Efetuar comunicação com o computador externo;
- Gerar vetores de posição para a placa de controle.

Efetivamente a placa implementa os módulos comportamentais, de comunicação e de decisão da arquitetura. O microcontrolador PIC 16F877 foi escolhido para coordenar e executar as tarefas da placa. Cada uma dessas tarefas serão vistas adiante, relacionando-as à arquitetura. A Figura DEL1 mostra o diagrama esquemático da placa.

#### **4.4.1. Leitura e interpretação dos sensores**

Oito sensores infra-vermelho compõem o conjunto de sensores do robô. Cada sensor é composto de um par emissor/receptor infra-vermelho. O emissor emite luz infra-vermelha em pulsos de frequência de 40KHz. O receptor, por sua vez, capta luz infra-vermelha e, ao detectar pulsos na frequência de 40KHz, sinaliza a recepção. É necessário que a luz seja pulsada para que fontes externas de luz infra-vermelha, como luzes fluorescentes e o Sol, não sejam detectadas, já que estas não apresentam modulação nas suas emissões. A frequência utilizada é devido aos receptores comerciais, utilizados nesta implementação, serem sintonizados nesta frequência.

O princípio de funcionamento de um sensor deste tipo é simples. O emissor emite luz pulsada em uma direção específica, onde se deseja detectar um obstáculo. Quando um obstáculo está presente, parte da luz que o atinge é absorvida e parte é refletida. A luz refletida atinge o receptor, que detecta os pulsos e sinaliza a presença de um obstáculo. O alcance de detecção depende, portanto, principalmente, da potência do emissor e da sensibilidade do receptor.

O receptor utilizado é um receptor comercial comum utilizado em televisores e outros aparelhos controlados por controle remoto, como o GPIU da Sharp, mostrado

na Figura 29. O emissor é um simples LED infra-vermelho, ligado a um vibrador astável implementado a partir de um 555, sintonizado na frequência correta. A sensibilidade do receptor é dependente da frequência de pulsação, sendo a maior sensibilidade localizada em 40KHz, decaindo nos arredores desta. Para especificar a distância de detecção do sensor, foi-se variando a frequência do emissor até que a detecção só ocorresse na distância desejada. Para uma distância de 10 centímetros chegou-se a uma frequência de 30KHz. O circuito para gerar essa frequência está na Figura 30.

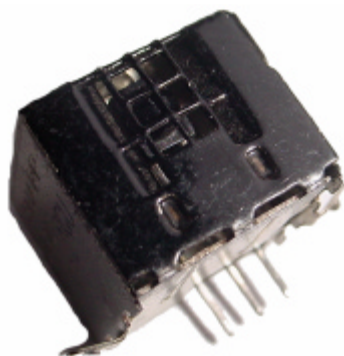


Figura 29: foto de um receptor infra-vermelho

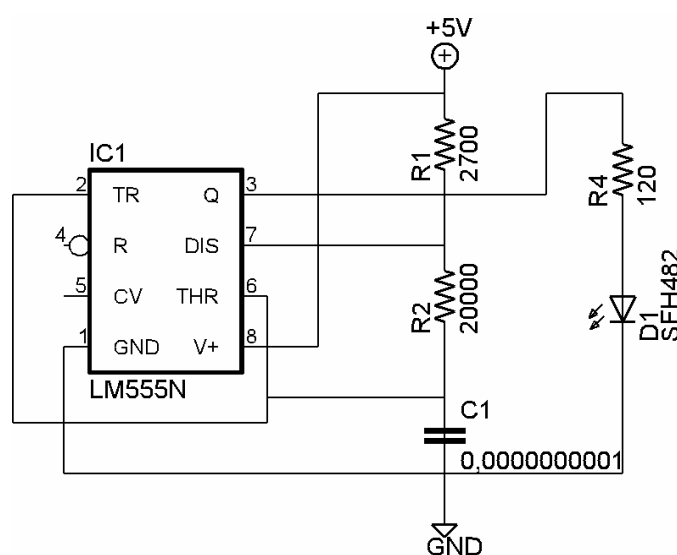


Figura 30: vibrador astável para 30KHz

Cada sensor deve ser monitorado, no aguardo de uma mudança de seu estado, que indica a presença de obstáculo à frente deste. Assim, cada sensor utiliza um pino de entrada do microcontrolador. Ao detectar um obstáculo, o microcontrolador ativa o

comportamento de evasão de obstáculo, gerando um vetor que faz com que o robô se afaste do ponto de detecção. A Figura 31 mostra um trecho do código exemplo, capaz de efetuar a leitura dos sensores e o comportamento de evasão.

```

#pragma bit s1    @ PORTE.0
#pragma bit s2    @ PORTE.1
#pragma bit s3    @ PORTE.2
#pragma bit s4    @ PORTE.3

char vx,vy;
...

while (1)
{
    ...

    if (s1==1) vx = vx + 100;
    if (s2==1) vx = vx - 100;
    if (s3==1) vy = vy + 100;
    if (s4==1) vy = vy - 100;

    ...
    evasao[0]=vx;
    evasao[1]=vy;
}

```

**Figura 31: trecho do código de leitura dos sensores e evasão**

No algoritmo da Figura 31 mostra o trecho suficiente para supervisionar quatro sensores, supondo que estes estão dispostos ortogonalmente entre si. A implementação para mais sensores é apenas questão de analisar a disposição destes.

O comportamento de evasão basicamente escreve em uma variável global, *evasao*, o vetor velocidade necessário para afastar o robô do obstáculo detectado, que será lido pelo módulo de decisão posteriormente.

#### 4.4.2. Comunicação com o computador externo

A comunicação com o computador externo é realizado utilizando-se um *link* de rádio unidirecional, onde o robô atua como receptor. A parte de mais baixo nível da comunicação, que inclui modulação/demodulação dos sinais digitais, é feita por componentes especializados comerciais, da fabricante Radiometrix. Especificamente, utiliza-se o transmissor TXM-433-5 e o receptor SILRX-433-5. Ambos operam na



Dependendo do tipo de comando recebido, o módulo escreve o comando em uma área específica da memória. Duas variáveis globais, na forma de vetores, existem no programa, uma especificando a posição atual do robô e outra a posição alvo. Ao receber um comando do tipo “vá para”, o módulo escreve os parâmetros na variável que especifica a posição alvo, *pos\_alvo*. No caso de receber um comando do tipo “setar posição”, os parâmetros são escritos sobre a variável de posição atual, *pos\_atual*.

#### 4.4.3. Geração de vetores velocidade para a placa de controle

Dois vetores velocidade são gerados pela placa, cada um com origem em um comportamento da arquitetura. Esses dois vetores são então somados para gerar o vetor que será passado para a placa de controle.

O módulo de direção a uma coordenada gera seu vetor a partir das variáveis globais contendo a posição atual do robô e a posição desejada. O algoritmo que implementa este módulo roda constantemente, de forma ininterrupta, realizando a subtração das duas variáveis, multiplicando o resultado por uma constante para normalizar o vetor gerado, e colocando o produto final em uma outra variável global *direcao\_alvo*. A Figura 33 mostra o trecho de código responsável por realizar essas funções.

```

char vx,vy,w;
...

while (1)
{
    ...

    vx = pos_alvo[0] - pos_atual[0];
    vy = pos_alvo[1] - pos_atual[1];
    w = pos_alvo[2] - pos_atual[2];

    direcao_alvo[0]=vx*K;
    direcao_alvo[1]=vy*K;
    direcao_alvo[2]=w*K;

    ...
}

```

```
}

```

**Figura 33: trecho de código para dirigir a uma coordenada**

Na seção 5.4.1. foi visto como o módulo de evasão gera seu vetor velocidade, armazenando-o na variável global *evasao*. O algoritmo que implementa o módulo de decisão é executado periodicamente, através de uma interrupção interna. Essa periodicidade deve ser definida com base em experiências empíricas, de modo a encontrar o maior período que permita o controle do robô. Períodos muito curtos causarão um *overhead* de tempo de processamento no tratamento de comunicações entre as placas, o que é indesejável.

Ao ser executado, o módulo de decisão soma os dois vetores presentes nas variáveis globais *evasao* e *direcao\_alvo* e repassa o resultado para a placa de controle. A Figura 34 mostra o trecho de código responsável por essas ações.

```
void decisao()
{
    int16 vx,vy,w;

    ...

    vx = evasao[0] + direcao_alvo[0];
    vy = evasao[1] + direcao_alvo[1];
    w = evasao[2] + direcao_alvo[2];

    if (vx>127) vx = 127;
    if (vy>127) vy = 127;
    if (w>127) w = 127;

    if (vx<-126) vx = -126;
    if (vy<-126) vy = -126;
    if (w<-126) w = -126;

    send_byte((char)vx);
    send_byte((char)vy);
    send_byte((char)w);

    ...
}

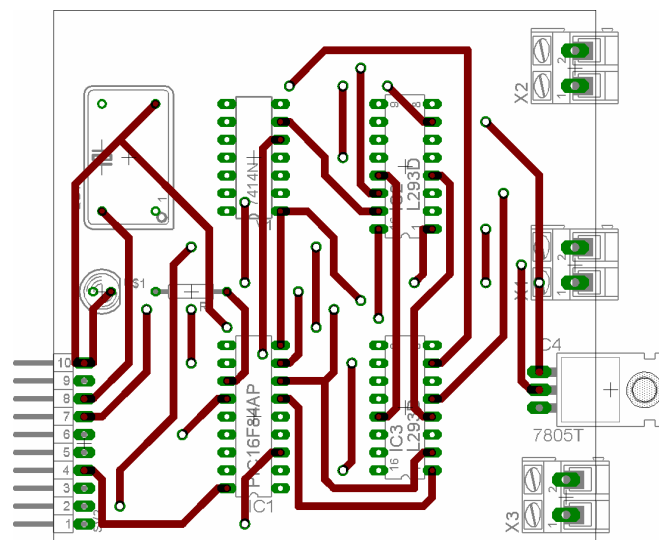
```

**Figura 34: trecho de código do módulo de decisão**

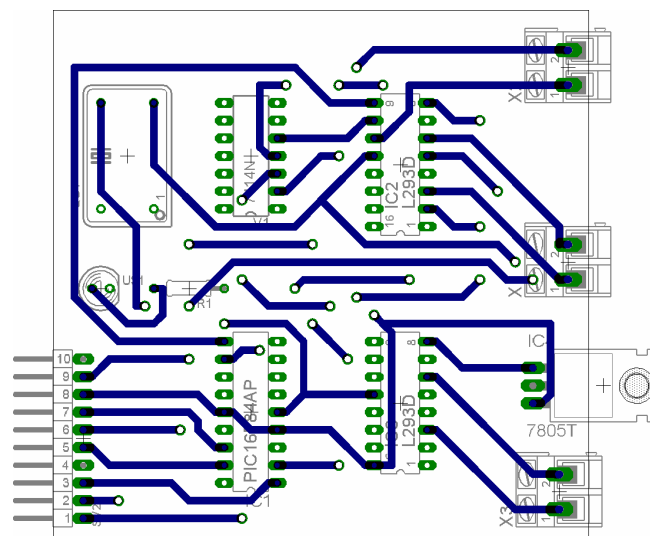
#### 4.5. Execução

Devido a problemas no cronograma e no custeio de determinadas partes necessárias, até o momento foi possível implementar efetivamente a placa de controle. Para permitir a realização de testes, adicionou-se ao algoritmo do microcontrolador um comportamento simples de movimentação, na forma de uma máquina de estados. Este comportamento realiza movimentos omni-direcionais em diversas direções, sempre retornando ao ponto original de partida após a realização de um movimento.

Para a execução da placa, utilizou-se o programa EAGLE Layout Editor, na sua versão 4.09. Este programa permite o desenho dos diagramas esquemáticos e realiza o roteamento automático das placas de circuito impresso a partir destes, resultando na placa da Figura 35. A partir dos desenhos gerados, realizou-se a passagem destes para uma placa de circuito impresso dupla face, onde, após ser corroída, foi feita a soldagem dos componentes.



(a)



(b)

**Figura 35: placas de circuito impresso geradas. (a) trilhas superiores e (b) inferiores**

A placa foi fixada sobre a base e os motores ligados nos terminais correspondentes. O conjunto foi alimentado com o uso de duas baterias de 9V ligadas em paralelo, de modo a fornecer a corrente necessária aos motores. A Figura 36 mostra o conjunto completo.

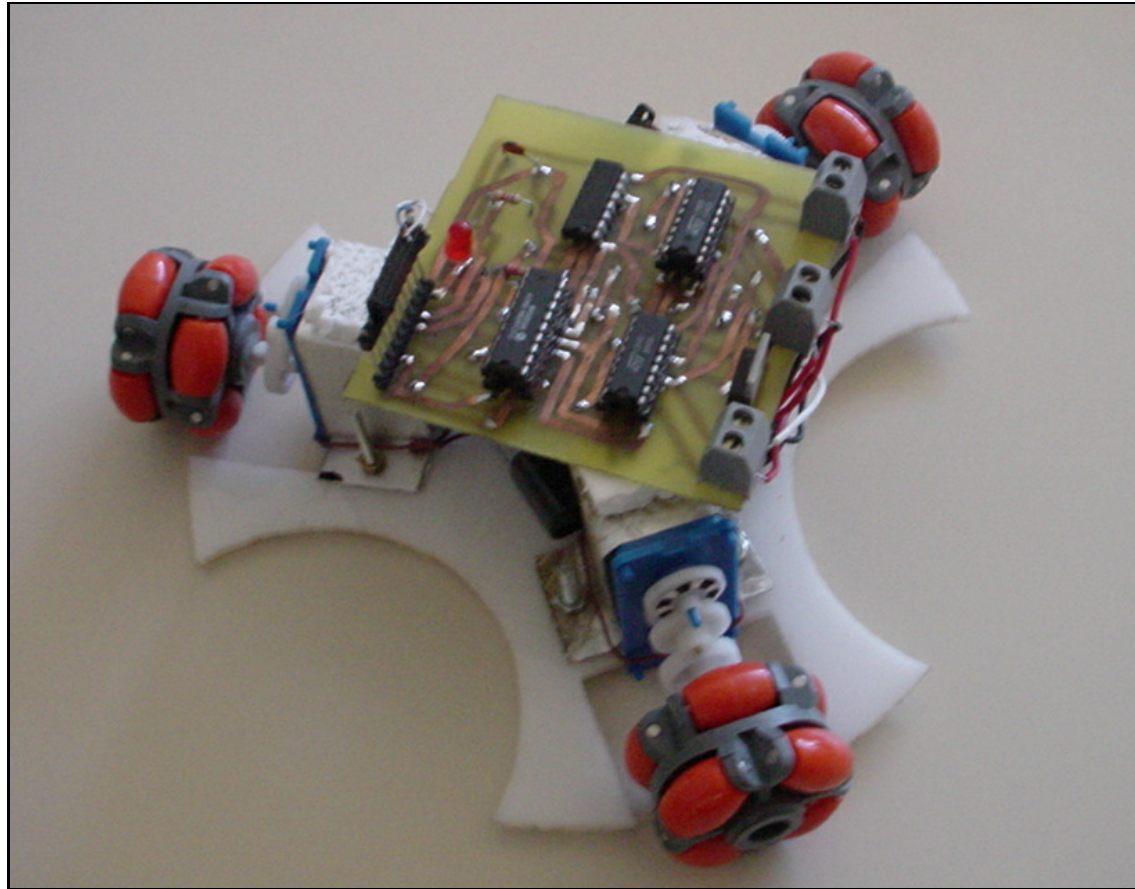


Figura 36: foto do robô finalizado

## 5. Testes e resultados

O principal a ser testado em uma arquitetura omni-direcional é a sua capacidade de realmente realizar movimentos omni-direcionais. Mais especificamente, há o interesse em realizar movimentos de translação mantendo a orientação do corpo constante.

O teste consistiu em configurar o algoritmo que implementa o comportamento de geração de trajetórias, descrito na seção anterior, para gerar trajetórias tipicamente omni-direcionais e analisar o comportamento do robô ao seguir as trajetórias. As trajetórias testes geradas são as seguintes:

- a) Locomover-se no eixo X;
- b) Locomover-se no eixo Y;
- c) Locomover-se em uma diagonal a 45 graus;
- d) Rotação sobre o próprio eixo.

Na Figura 37 encontram-se os resultados dos testes realizados. Para cada movimento especificado acima, foram registradas as posições inicial, intermediária e final. A posição inicial é o de princípio do movimento, a intermediária é o deslocamento até o ponto especificado e a final é o retorno ao ponto inicial. A eficácia de cada movimento está na capacidade de manter uma linha reta e a orientação fixa durante a translação e, no retorno, parar em uma configuração o mais próxima possível da inicial. No teste de rotação, o ponto central do robô deve permanecer sem deslocamentos entre a posição inicial e final, indicando que apenas há rotação realmente. Das figuras pode-se notar que a orientação é mantida de forma correta, porém uma leve curvatura

aparece na trajetória, causando erro na posição intermediária. A mesma curvatura aparece no retorno, reduzindo o erro na posição final.

Medições realizadas durante os testes de translação determinaram um erro de posicionamento, na posição intermediária, de 5 centímetros após um deslocamento de cerca de 40 centímetros, representando um erro de 12,5%. Este erro é medido em relação ao ponto central do robô e o eixo que este deveria seguir. O erro na posição final ficou um pouco acima de 3 centímetros. A orientação, por outro lado, manteve-se bastante estável durante todo o movimento, mas não foi possível realizar a medição efetiva do erro.

No teste de rotação, após girar 180 graus, o deslocamento do ponto inicial até o final ficou, na média, em torno de 6 centímetros.

Pode-se identificar a origem dos erros apresentados em três vertentes: tratamento simplificado da não-linearidade do controle por PWM, falta de um controle em malha fechada e mau alinhamento das rodas. O tratamento da não-linearidade parece ter sido insuficiente e métodos melhores devem ser empregados para controlar os motores. Essa parece ser a principal causa dos movimentos levemente curvos. O controle em malha fechada pode superar o problema da não-linearidade de controle e, adicionalmente, fornece maior confiabilidade nas velocidades dos motores e, por consequência, na trajetória do robô como um todo. O mau alinhamento das rodas também parece ter influência nos movimentos curvos, uma vez que a consequência do alinhamento errado é semelhante ao de se ter velocidades incorretas nas rodas.

Apesar de haverem erros, estes ficaram em níveis aceitáveis para um jogo de futebol de robôs. Havendo necessidade de um controle de maior precisão, os problemas identificados devem ser sanados. Os testes mostraram ser possível o controle omni-direcional a partir de componentes de baixo custo, além de uma baixa capacidade de processamento embarcada.

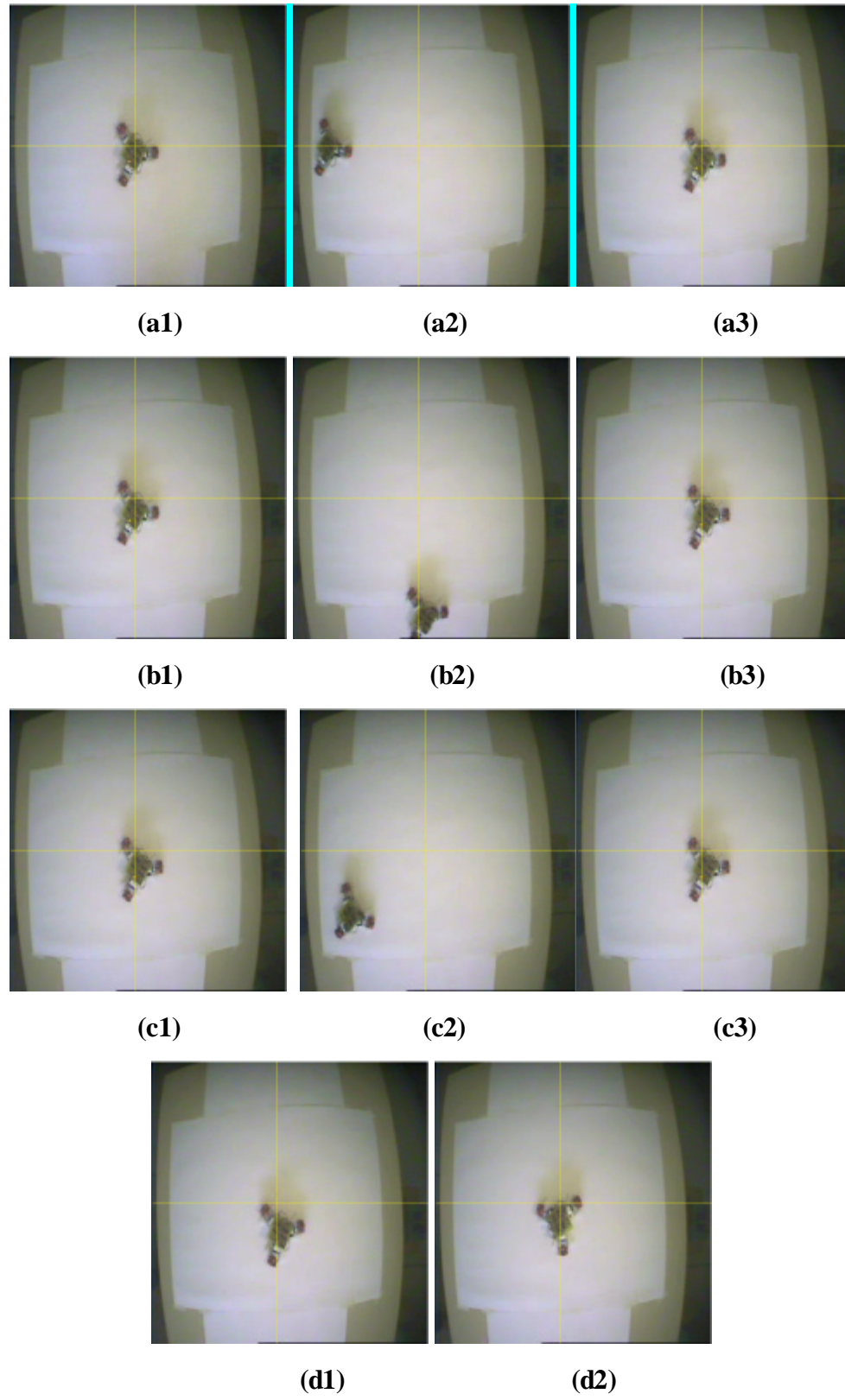


Figura 37: resultados dos testes realizados

## 6. Considerações finais

Neste trabalho, foi apresentado um estudo sobre arquiteturas de robôs móveis e foi proposta uma arquitetura modular para controle de robôs omni-direcionais, baseada em uma arquitetura híbrida entre o modelo clássico e o modelo comportamental. De modo a validar a arquitetura desenvolvida, estudos sobre implementação de robôs móveis foram feitos e o desenvolvimento de um protótipo foi realizado.

Durante o trabalho, foram desenvolvidos e estudados sistemas de controle de motores elétricos DC, microcontroladores para executar a arquitetura proposta, sensores de diversos tipos, diversos módulos de comunicação sem fio, entre outras áreas de microeletrônica, teoria de controle e telecomunicações.

Ao fim, obteve-se um protótipo capaz de realizar movimentos omni-direcionais com razoável precisão, suficiente para o jogo de futebol de robôs, validando parcialmente a arquitetura proposta.

O desenvolvimento teórico de uma arquitetura para robôs omni-direcionais apresentou uma série de desafios, sendo a maioria relativa a definição e delimitação do fluxo de controle dentro da arquitetura, de modo a permitir que esta se tornasse modular o suficiente para permitir implementações eficientes. Grande dificuldade foi encontrada em descobrir-se quais elementos influenciavam efetivamente o controle omni-direcional, de modo a isolá-los na arquitetura permitindo um tratamento específico.

A implementação de um protótipo para validar a arquitetura esbarrou em uma série de dificuldades. A primeira grande dificuldade foi a falta de recursos para adquirir componentes para a montagem. Dada a necessidade de manter os custos baixos,

muitas soluções comerciais disponíveis tiveram de ser descartadas em prol de soluções menos eficientes, muitas vezes implementadas utilizando-se peças reutilizadas. Outro desafio foi a parte mecânica, que mostrou-se bastante complexa. O alinhamento das rodas é crucial para a locomoção omni-direcional e a falta de equipamentos para realizar este alinhamento, bem como o corte da base, de forma eficiente, prejudicou as demais etapas.

A baixa disponibilidade de diversas partes necessárias para a implementação também mostrou ser um problema. Diversas vezes não se encontrava no país as partes necessárias, tendo-se que recorrer a importação, custosa e lenta, ou a utilização de similares nem sempre tão eficazes. Em dado momento, dois microcontroladores PIC16F84, de 20MHz, estragaram e não foi possível encontrar equivalente idêntico, tendo que se readaptar o projeto para um microcontrolador de velocidade mais baixa (de 4MHz), o que colaborou para aumentar os erros nos testes.

Este trabalho aborda diversos aspectos sobre robótica móvel, em especial sobre a efetiva implementação física de robôs móveis, a partir de especificações básicas. Dá ao leitor uma visão sobre as dificuldades e apresenta soluções para problemas envolvidos na especificação e implementação de uma arquitetura e no controle omni-direcional.

Para o projeto Furgbol, definiu-se a base para a construção dos robôs que efetivamente participarão dos jogos, especificando como cada robô deverá se comportar e como devem ser implementados.

Uma série de melhorias pode ainda ser feita no que foi implementado, além da implementação das partes que, devido ao pouco tempo e recursos, não foram feitas nesta etapa. Pode-se listar os trabalhos futuros como sendo, em ordem de prioridade:

- Implementação da placa de supervisão;
- Implementação do controle em malha fechada dos motores;
- Especificar-se melhores motores, mais rápidos e com melhor estrutura;
- Adquirir rodas omni-direcionais emborrachadas, para reduzir deslizamentos;

E, para o futebol de robôs em específico, visando a completar o robô, resta ainda especificar devidamente o mecanismo de chute, que impulsionará a bola, e definir uma melhor estrutura, mais robusta, para suportar os novos componentes. Futuramente, ainda pode-se implantar a visão local no robô, possibilitando torná-lo totalmente autônomo.

## Bibliografia

- [ACV1] **Intelligent Active Vision**. Online.  
<http://www.tcs.auckland.ac.nz/~jacky/teaching/courses/CompSci-766/current>
- [ALV1] ALVISO, Nataniel. **The Tutebot**. Online.  
<http://www.upd.edu.ph/~mobotlab/tutebot.html>.
- [BOT96] BOTELHO, S.C., **Desenvolvimento de sistemas inteligentes para controle de robôs móveis**. CPGCC da UFRGS. Porto Alegre, 1996.
- [BRO85] BROOKS, Rodney. **A Robust Layered Control System for a Mobile Robot**. A.I. Memo 864, MIT, 1985.
- [BRO90] BROOKS, Rodney. **Elephants don't play chess**. MIT Press, 1990.
- [CAR01] CARTER, Brian et al. **Mechanical design and modeling of an omni-directional robocup player**. Em *Proceedings RoboCup 2001 International Symposium*, Seattle, WA, 2001.
- [CAT1] CATENA, Mario. **PIC16F877 Programmer And Development System**. Online. <http://www.angelfire.com/ok3/masterbyte/>
- [DES92] DESAI, Rajiv e MILLER, David. **A simple reactive architecture for robust robots**. IEEE International Conference on Robotics and Automation, 1992.
- [ENC1] **Two Channel Optical Incremental Encoder Modules**. Online.  
<http://www.seattlerobotics.org/encoder/200006/heds9000.pdf>

- [FIR1] **FIRA**. Online. <http://www.fira.net>
- [GRA1] GRABOWSKI, Bob. **Small Robot Sensores**. Online.  
<http://www.contrib.andrew.cmu.edu/~rjg/index.html>
- [HAL1] **An Introduction to the Hall Effect**. Online.  
<http://www.sypris.com/library/documents/hallcatalog.pdf>.
- [HAY01] HAYKIN, Simon e VEEN, Barry. **Sinais e sistemas**. Bookman, 2001.
- [KHP1] K-TEAM. **Khepera – User Manual**. Lausanne: K-TEAM, 1994.
- [KON98] KONGMUNVATTANA, Angkul e CHONGSTITVATANA Prabhas. **A FPGA-based Behavioral Control System for a Mobile Robot**. IEEE APCCAS, 1998.
- [LAT91] LATOMBE, Jean-Claude, **Robot Motion Planning**. Kluwer Academic Publishers, 1991.
- [MAC93] MACKWORTH, Alan. **Computer Vision: System, Theory and Application**. World Scientific Press, 1993.
- [MAC93b] MACKWORTH, Alan. **On seeing robots**. Em A. Basu e X. Li, editores, *Computer Vision: Systems, Theory, and Applications*. World Scientific Press, Singapura, 1993.
- [MAX00] MAXON, Kenneth. **Reaching the next step in motion control and sensor-software fusion for mobile robots**. Encoder Newsletter, junho / 2000.

- [MIC96] MICHAUD, François et al. **A new control architecture combining reactivity, planning, deliberation and motivation for situated autonomous agent.** International Conference on Simulation of Adaptive Behavior, 1996.
- [OGA00] OGATA, Katsuhiko. **Engenharia de controle moderno.** LTC, 2000.
- [PIC01] **PIC 16F87X Datasheet.** Microchip, 2001.
- [PPR1] **PPRK: Palm Pilot Robot Kit.** Online. <http://www-2.cs.cmu.edu/~pprk/>
- [PWM1] **A thread discussion on PWM.** Online.  
<http://www.seattlerobotics.org/encoder/sep99/pwmmail.html>
- [RUS95] RUSSEL, S. e NORVING, P. **Artificial Intelligence: A Modern Approach.** Prentice Hall, 1995.
- [ROB1] **Robocup.** Online. <http://www.robocup.org>
- [SAF1] SAFFIOTI, Alessandro e DORIGO, Marco. **Using Fuzzy Logic in Autonomous Robot Navigation.** Online.  
<http://iridia.ulb.ac.be/Projects/flar.html>.
- [SIU00] SIUZA, David J. **Desbravando o PIC.** São Paulo: Érica, 2000.