

EDSON BERNARDES FERREIRA FILHO

**DESENVOLVIMENTO DE UM TIME DE FUTEBOL DE ROBÔS
CATEGORIA *IEEE VERY SMALL SIZE***

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Profa Dra Karla Boaventura Pimenta
Palmieri

Ouro Preto
Escola de Minas – UFOP
Abril/2013

Monografia defendida e aprovada em 12 de abril de 2013 pela comissão avaliadora
constituída pelos professores:

Karla Palmieri

Prof. Karla Boaventura Pimenta Palmieri - Professora Orientadora

Henor Artur de Souza

Prof. Dr. Henor Artur de Souza - Professor Convidado

Ricardo Augusto Rabelo Oliveira

Prof. Dr. Ricardo Augusto Rabelo Oliveira - Professor Convidado

EPÍGRAFE

“I always choose a lazy person to do a difficult job because he will find an easy way to do it.” (Bill Gates)

DEDICATÓRIA

Dedico este trabalho à minha Mãe.

AGRADECIMENTO

À minha mãe que sempre me apoiou e respeitou minhas decisões e por que sem ela nada disto seria possível.

Às minhas irmãs Aninha e Nina e a minha sobrinha Marina por confiarem em mim e por estarem presentes na minha vida.

A todos os outros parentes, aos quais sempre tive uma incomensurável relação de amizade.

A todos meus amigos, de Ouro Branco e de Belo Horizonte por fazerem parte da minha formação humana e aos amigos de Ouro Preto e da Automação por tornarem minha estadia nesta maravilhosa cidade mais alegre.

A toda a equipe RODETAS: Prof. Dra. Karla Boaventura Pimenta Palmieri, João Paulo Gonçalves Simim, João Paulo Ferreira, Allan Augusto Almeida Ribeiro, Arthur Caio Vargas e Pinto, Arthur Oliveira Santos, Érica Silva Pinto, Sávio Palmieri, Felipe Otávio Pinto Figueiredo, Fernando dos Santos Alves Fernandes, Jéssica Ferreira Soares, Jonathan Alves Melquiades, Leonardo Henrique Pinho, Rafael Gustavo Alves e Sávio Nazareno Júnior por terem sido parte fundamental do projeto e pela amizade desenvolvida durante este processo.

Aos professores da PUC-MG por terem contribuído na minha formação acadêmica enquanto lá estive.

A todos os professores e funcionários da histórica Escola de Minas, em especial a professora Karla pelas oportunidades.

A gloriosa REPÚBLICA EXÍLIO, seus moradores e ex-alunos pela acolhida e irmandade.

RESUMO

O desenvolvimento de novas tecnologias atualmente está mudando a forma como as pessoas vivem. Por esta razão, juntar várias destas tecnologias em um mesmo projeto será de grande utilidade na formação de engenheiros. A construção de um time de futebol de robôs categoria *IEEE very small size* é apresentada evidenciando todas as partes que foram necessárias neste desenvolvimento. O time consiste de três robôs com dimensões máximas de um cubo de 7,5 centímetros. Os robôs são autônomos, utilizando uma câmera para percepção do ambiente, um computador para o planejamento das estratégias e motores fazendo os robôs se deslocarem atuando neste ambiente. Foram utilizadas tecnologias nas áreas de redes sem fio, processamento de imagem, campos potenciais, lógica *Fuzzy*, sistemas embutidos e acionamento elétrico. Foi feito um estudo bibliográfico de várias destas tecnologias e de alternativas que não foram utilizadas. A rede sem fio ficou por conta do módulo XBee®, o processamento de imagem pela plataforma Adobe® Flash® e o sistema embutido utilizado foi o Arduino e seus Shields. Além disso, há a explanação de um sistema do jogo de futebol de robôs com todas estas ferramentas trabalhando juntas.

Palavras-Chave: Rede sem fio, processamento de imagem, sistemas embutidos, robôs, futebol de robôs.

ABSTRACT

The development of new technologies in these days is changing the way people live. For this reason, putting together many of these technologies in a same project will be of great usefulness in the formation of engineers. The construction of a robot soccer team category IEEE very small size is presented demonstrating all the necessary parts in this development. The team consists of three robots with maximum dimension of a 7,5 centimeters cube. The robots are autonomous. Is used a camera to recognize the environment, a computer to plan the strategies and motors to make the robot move interacting with this environment. It was utilized technologies in the field areas of wireless networks, image processing, potential fields, Fuzzy logic, embedded systems and electric drive. It was made a bibliographical study of those technologies and of alternatives that were not used as well. The wireless network was made with the XBee® module, in the image processing was utilized the Adobe® Flash® platform, the embedded system utilized was the Arduino and its shields. Furthermore, there is an explanation of the game system covering all those tools working together.

Keywords: Wireless networks, image processing, embedded systems, robots, robot soccer.

SUMÁRIO

1. INTRODUÇÃO	16
1.1 Considerações iniciais	16
1.2 Objetivos.....	18
1.2.1 Objetivos Específicos	18
1.3 Motivação	19
1.4 Metodologia.....	19
1.5 A equipe RODETAS	20
1.6 A Estrutura do trabalho	21
2 SISTEMAS EMBARCADOS	22
2.1 Definição	22
2.2 Elementos de um sistema embarcado.....	22
2.2.1 CPU	23
2.2.2 Memória.....	23
2.2.3 Entradas e Saídas.....	24
2.3 A placa Arduino	24
2.3.1 Linguagem de programação do Arduino	25
2.3.2 Shield Ardumoto	26
2.3.3 Shield XBee	27
3 COMUNICAÇÃO SEM FIO	29
3.1 Introdução.....	29
3.2 Padrões das redes sem fio	29
3.2.1 IEEE 802.11.....	30
3.2.2 Bluetooth	31
3.2.3 ZigBee	32
3.3 Comparativo entre os padrões.....	38
4 RECONHECIMENTO DE IMAGEM.....	40
4.1 Introdução.....	40
4.2 Digitalização	40
4.2.1 Resolução.....	41

4.2.2 Cores.....	41
4.3 Reconhecimento.....	43
4.4 A plataforma Flash.....	44
4.5 Sistema de reconhecimento da imagem.....	45
5 O JOGO E SUAS REGRAS.....	46
5.1 Introdução.....	46
5.2 História.....	46
5.3 Algumas das categorias do futebol de robôs.....	46
5.2.1 Humanoide.....	47
5.2.2 <i>Small Size</i>	48
5.2.3 <i>IEEE Very Small Size</i>	49
6 ESTRATÉGIAS.....	53
6.1 Introdução.....	53
6.2 Campos Potenciais.....	53
6.2.1 Força de repulsão.....	54
6.2.2 Força de atração.....	55
6.2.3 Força resultante.....	56
6.3 Lógica <i>Fuzzy</i>	57
7 DESENVOLVIMENTO.....	58
7.1 Introdução.....	58
7.2 Desenvolvimento Arduino.....	58
7.2.1 Protocolo de identificação do robô.....	58
7.2.2 Shields Arduino.....	60
7.2.3 Envio aos motores.....	60
7.2.4 Código no Arduino.....	61
7.3 Motores.....	62
7.4 Baterias.....	62
7.5 Estrutura.....	64
7.6 Xbee.....	65
7.7 Desenvolvimento no computador.....	66

7.7.1 Reconhecimento da imagem.....	67
7.7.2 Estratégia do time.....	69
7.7.3 Comunicação com a porta serial	71
8 RESULTADOS	73
9 CONSIDERAÇÕES FINAIS	75

Lista de Figuras

- Figura 1.1 Sistemas geral
- Figura 2.1 Arquitetura de um sistema embarcado
- Figura 2.2 Arduino UNO R3 visto de frente
- Figura 2.3 *Sketch* exemplo Arduino
- Figura 2.4 Ardumoto conectado ao Arduino
- Figura 2.5 Esquema da ponte H
- Figura 2.6 *Shield* XBee
- Figura 3.1 Módulo XBee
- Figura 3.2 Topologia em estrela
- Figura 3.3 Topologia em árvore
- Figura 3.4 Topologia em malha
- Figura 3.5 Camadas da arquitetura protocolar *ZigBee*
- Figura 3.6 Detalhes da arquitetura protocolar *ZigBee*
- Figura 4.1 Plano cartesiano de cores RGB e cubo RGB
- Figura 4.2 Representação de cores HSI
- Figura 5.1 NAO®, exemplo de robô humanoide
- Figura 5.2 Padrões de cores usados na *RoboCup* 2012
- Figura 5.3 Regras referentes ao robô e a bola
- Figura 5.4 Exemplos de robôs com seus *patches*
- Figura 5.5 Dimensões do campo
- Figura 5.6 A bola do jogo
- Figura 6.1 Campo repulsivo
- Figura 6.2 Representação de uma força repulsiva
- Figura 6.3 Campo de atração
- Figura 6.4 Representação de uma força atrativa
- Figura 6.5 Exemplificação de uma trajetória
- Figura 7.1 Motor e caixa de redução
- Figura 7.2 *Shield* ardumoto ligado aos motores
- Figura 7.3 Bateria de NiMH

Figura 7.4 Estrutura do robô não finalizada

Figura 7.5 Estrutura do robô

Figura 7.6 Módulo XBee com antena adaptada

Figura 7.7 Esquema completo da rede

Figura 7.8 Interface criada no Adobe® Flash®

Figura 7.9 Câmera utilizada

Figura 7.10 Trajetória do atacante

Figura 8.1 Time completo

Figura 8.2 Troféu obtido na competição

Lista de Tabelas

Tabela 1.1 – Primitivas de um robô

Tabela 3.1 – Comparativo entre padrões de comunicação sem fio

Tabela 7.1 – Protocolo de identificação de cada robô

Tabela 7.2 – Exemplo de uma mensagem

Tabela 7.3 – Significado do segundo caractere de cada mensagem

Tabela 7.4 – Relação entre velocidade da mensagem e velocidade enviada ao motor

Tabela 7.5 – Pseudocódigo do Arduino

Tabela 7.6 – Entradas do controlador *Fuzzy*

Tabela 7.7 – Exemplo das regras do controlador *Fuzzy*

Lista de Siglas

AF - *Application Framework*

APL – Camada de aplicação

APS - *Application Support Sublayer*

CLP – Controlador Lógico Programável

CPU – *Central Processing Unit*

CSMA-CA - *Carrier Sense Multiple Acesses – Collision Avoidance*

EEPROM - *Electrically Erasable Programmable Read Only Memory*

ETSI - *European Telecommunications Standardisation Institute*

FFD – *Full Function Device*

HSI – *Hue, Saturation and Intensity*

IEEE – *Institute of Electrical and Eletronics Engineers*

IrDA – *Infrared Data Association*

ISM – *Industrial, Scientifical and Medical*

ISO – *International Standards Orgnization*

LARC – *Latin American Robotics Competition*

LED – *Light Emitting Diode*

MAC – *Medium Access Control*

NWK – *Network*

OSI – *Open System Interconnection*

PC – *Personal Computer*

PHY – Camada Física

PWM – *Pulse Width Modulation*

RFD – *Reduced Function Device*

RGB – *Red, green and blue*

SIG – *Special Interest Group*

SRAM – *Static ramdom acess memory*

ULA – Unidade Lógica Aritmética

USB – *Universal Serial Bus*

WLAN - Wireless Local Area Network

WMAN - Wireless Metropolitan Area Network

WPAN - Wireless Personal Area Network

WWAN - Wireless Wide Area Network

ZDO - ZigBee Device Object

1. INTRODUÇÃO

1.1 Considerações iniciais

O desenvolvimento de elementos robóticos é uma crescente área no campo da engenharia e ciência da computação e possibilita avanços na precisão, velocidade e custo de sistemas industriais e em aplicações no campo da domótica.

Um robô é definido como uma máquina que sente, pensa e age. Um sistema autônomo é definido como sendo um sistema capaz de operar no ambiente do mundo real sem nenhuma forma de controle externo por prolongados períodos de tempo (BEKEY, 2005).

Um robô autônomo é uma máquina que percebe as condições do ambiente ao qual está inserido e após um processamento feito, de forma que não haja interação do ser humano, toma decisões e ações modificando este determinado ambiente.

Esta autonomia depende de todos os dados que o robô consegue capturar do ambiente e depende também do ser humano, pois todo o “pensamento” do robô está pré-programado para que ele possa responder aos estímulos do ambiente.

Segundo Murphy (2000), as três primitivas dos robôs aceitas atualmente são: Sentir, planejar e agir conforme é apresentado na Tabela 1.1.

Tabela 1.1 - Primitivas de um robô.

Primitivas do robô	Entrada	Saída
Sentir	Dados do Sensor	Informação “sentida”
Planejar	Informações (“sentidas” ou cognitivas)	Diretivas
Agir	Informações “sentidas” ou diretivas	Comando no atuador

Fonte: MURPHY, 2000.

Traduzindo estas primitivas em termos práticos:

- Sentir (*Sense*) – É a parte de sensoriamento do robô, aqui o robô irá perceber o ambiente a sua volta. Imaginando essa primitiva como uma caixa preta, a entrada é o ambiente e a saída é alguma medida palpável para o tratamento eletrônico, como por exemplo, tensão ou corrente.

- Planejar (*Plan*) – Aqui a informação obtida do sensor irá ser tratada afim de que possa ser analisada por alguma unidade de processamento, seja ela embutida ou não. Essa unidade de processamento faz com que o robô “pense” e planeje a próxima ação a ser tomada.
- Agir (*Act*) – Depois de decidido o que fazer, é hora de dar a resposta planejada ao ambiente.

De acordo com Stone (2007), *RoboCup* ou *Robot Soccer World Cup* é uma iniciativa de pesquisa mundial projetada para avançar-se nos campos de robótica e inteligência artificial.

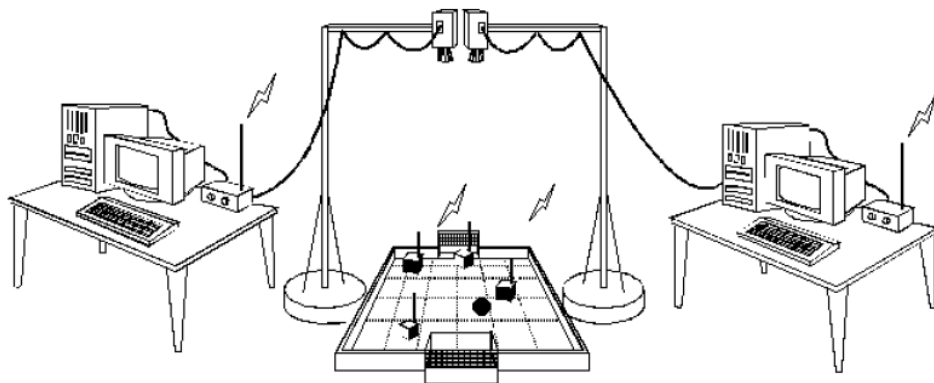
A *RoboCup* é responsável pela organização de competições de futebol de robôs. Atualmente existem várias categorias com regras específicas para cada uma delas.

No caso da categoria *IEEE Very Small Size* temos que o que será responsável pelo “sentir” será feita por uma câmera digital *webcam* colocada acima do campo. Essa é responsável por perceber o estado a cada momento de cada robô, assim como o estado da bola.

Ligada à câmera está um computador pessoal (PC) que é designado para fazer a transcrição desta imagem capturada para que ela possa ser usada no planejamento dos robôs.

Depois de ter tudo planejado, deve-se transmiti-las aos robôs de forma que eles possam transformar o planejamento em ações.

A figura 1.1 mostra um esquema do sistema como um todo, unindo cada uma das partes



utilizadas.

Figura 1.1 - Sistema geral.

Fonte: CBROBOTICA, 2008.

1.2 Objetivos

- Estudar, projetar e construir um time completo de robôs autônomos categoria *IEEE very small size*;
- Capaz de disputar uma partida inteira de futebol de robôs, segundo as regras da *IEEE*, com eficiência.

1.2.1 Objetivos Específicos

No caso da *RoboCup* o objetivo final é criar um time que consiga ganhar da seleção brasileira de futebol (KITANO, 1997). Obviamente este objetivo está longe de ser alcançado, mas mostra como, no futebol de robôs, sempre ter-se-á espaço para melhorias.

Mas, atendo-se a tecnologia e recursos atuais, o objetivo é a criação de um modelo de robô adequado para a categoria de futebol de robôs *IEEE Very Small Size*. Depois que esse modelo for criado, aplicar esse modelo para criação de um protótipo de um robô totalmente autônomo capaz de participar de um jogo de futebol de robôs.

- Seguindo as primitivas de um robô (sentir, planejar e agir), definir todo o material a ser usado, micro controlador, plataforma de controle, tipo de comunicação sem fio e todos os demais componentes;
- Posteriormente fazer a revisão bibliográfica de todas as ferramentas que serão usadas no projeto. O micro controlador o dispositivo de comunicação sem fio, o reconhecimento de imagem e a plataforma de controle;
- Desenvolver todo o reconhecimento de imagem e atrelá-la a plataforma de comando escolhida. O reconhecimento de imagem garantirá que o robô tenha a percepção do ambiente em que estará inserido;
- Definir todo o controle necessário através de um computador pessoal (PC). O

computador será responsável pelo “pensar” do robô, ou seja, aqui é feito seu planejamento;

- Desenvolver toda a programação necessária do micro controlador de cada robô. Este fará com que finalmente, o robô aja de acordo com o planejamento;
- Finalmente, traçar estratégias de jogo e designá-las aos robôs a fim de formar uma equipe totalmente apta a disputa de um jogo e um campeonato;
- Desenvolver toda a comunicação entre o microcontrolador e o dispositivo de comunicação escolhido. Esta comunicação fará com que as sensações obtidas do reconhecimento de imagem, os “pensamentos” obtidos do computador controlador e as ações executadas pelos atuadores estejam coordenados;
- Documentar todo o processo de criação dos robôs, como forma de poder-se dar continuidade ao projeto.

1.3 Motivação

A motivação primordial no desenvolvimento de um time de futebol de robôs advém da necessidade de aplicar parte do conhecimento obtido do curso de graduação na prática.

A programação de robôs móveis é agradável e inspiradora para estudantes, isso pode ser amplificado introduzindo competições de robôs onde dois times se digladiam na solução de algum problema (BRÄUNL, 1999).

Competições de robótica seguem esse preceito e fazem com que todos aprendam no desenvolvimento de seus robôs.

A competição em questão se trata da *Latin American Robotics Competition (LARC)* – 2012, com realização entre os dias 17-21 de Outubro de 2012 na Universidade de Fortaleza (UNIFOR).

1.4 Metodologia

- Levantamento sobre as tecnologias utilizadas em competições de futebol de robô;

- Pesquisa acerca dos materiais que podem ser utilizados;
- Definição e obtenção dos materiais;
- Desenvolvimento do software de reconhecimento de imagem;
- Desenvolvimento do software de cada robô;
- Desenvolver a comunicação entre os robôs e computador;
- Desenvolvimento do software da estratégia da equipe;
- Testes e correções das possíveis estratégias da equipe;
- Análise de resultados obtidos na competição.

1.5 A equipe RODETAS

A equipe RODETAS Robô Clube é uma equipe de futebol de robôs da Universidade Federal de Ouro Preto. Esta equipe realiza seus trabalhos no Laboratório de Tecnologias Industriais orientados pela professora Karla Boaventura Pimenta Palmieri. Todo o desenvolvimento do projeto foi realizado por toda a equipe, eventualmente dividindo-a em módulos de trabalho.

A equipe existe desde agosto de 2011, inicialmente foi formada por alunos que então estavam no primeiro período do curso de engenharia de controle e automação e um aluno de ciência da computação. Posteriormente foram aceitos na equipe, por meio de entrevistas de seleção, alunos mais avançados no curso, de acordo com a necessidade da equipe.

A equipe conta atualmente com treze membros dos cursos de engenharia de controle e automação e de ciência da computação que desenvolvem projetos na área de robótica, entre eles a competição *IEEE very small size*.

Inicialmente a equipe dividiu em dois grandes blocos de pesquisa: *Software e Hardware*, no qual cada um dos integrantes partiu para a busca e coleta de informações que fosse útil ao projeto.

Atualmente a equipe está dividida em três grandes blocos: *Software*, Eletrônica e *Hardware*. O bloco do *Software* é responsável pelo processamento de imagem e pela estratégia da equipe. O bloco da Eletrônica é responsável por toda eletrônica embarcada, além da programação do sistema embarcado. O bloco do *Hardware* atualmente é responsável pela construção física dos robôs, do campo e das outras estruturas necessárias.

1.6 A Estrutura do trabalho

No capítulo um é feita uma breve introdução sobre a robótica, e sobre o futebol de robôs.

Dos capítulos dois ao seis é apresentado o estudo bibliográfico do projeto.

O capítulo dois é referente à plataforma embutida usada, mas especificamente, o Arduino®.

No capítulo três, faz-se uma descrição de tecnologias da comunicação sem fio e do módulo de comunicação sem fio ZigBee.

O capítulo quatro apresenta o processamento e reconhecimento de imagem, ou seja, a visão computacional além de apresentar a plataforma Flash®.

No capítulo cinco, são descritas resumidamente as regras de um jogo de futebol de robôs, bem como uma breve descrição da história desse jogo.

São apresentadas no capítulo seis as ferramentas estratégicas que podem ser usadas para tal jogo.

No capítulo sete tem-se a união de todas as ferramentas utilizadas para construção do time de robôs propriamente dito, mostrada em etapas do desenvolvimento.

E no capítulo oito é apresentado os resultados obtidos.

No capítulo nove são apresentados uma conclusão sobre o projeto e também sugestões para possíveis melhorias e implementações futuras no mesmo.

2 SISTEMAS EMBARCADOS

2.1 Definição

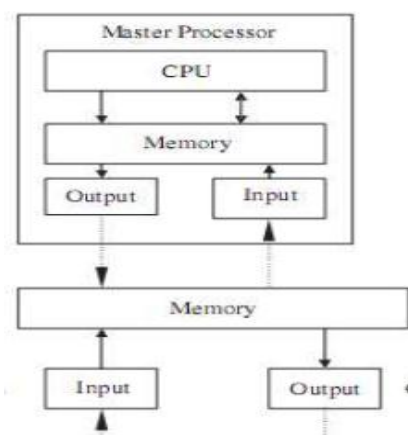
Sistemas embarcados, também chamados de sistemas embutidos e até de sistemas embebidos, são utilizados na maioria das vezes em pequenas automações onde a robustez do controlador não é uma exigência.

Os sistemas computacionais embarcados estão cada vez mais presentes na vida das pessoas, mesmo que elas não os veja. Isto se deve ao fato de estarem cada vez menores e mais acessíveis economicamente.

São similares aos sistemas computacionais não embarcados, no seu modo de funcionamento. Porém, os sistemas embarcados geralmente são especificados de acordo com a sua aplicação. Por exemplo, em aplicações remotas e com difícil acesso, é importante escolher um sistema embarcado com alta autonomia e baixo consumo de bateria.

2.2 Elementos de um sistema embarcado

Em linhas gerais, um sistema embutido deve ser composto de pelo menos três elementos, uma Unidade Central de Processamento (CPU), uma memória, e um ou mais dispositivos de entrada/saída. Na figura 2.1 apresenta-se a arquitetura básica de um sistema embarcado. Como se pode observar, além do CPU, memória, entradas e saídas, normalmente existem outros elementos como uma memória fora do microcontrolador e novas saídas/entradas para



esta memória.

Figura 2.1 – Arquitetura de um sistema embarcado.

Fonte: Adaptada de NOERGAARD, 2005.

2.2.1 CPU

CPU, *central processing unit*, ou ainda, unidade central de processamento é a parte responsável por todo o processamento do micro controlador, é composto por uma unidade lógica aritmética (ULA) e por registradores diversos. Estes registradores são uma memória interna da CPU, normalmente de pequena capacidade, mas que são capazes de atingir grandes velocidades na comunicação com a ULA

Segundo BORGES (2010?), em sistemas de Controle e Automação, são raros os exemplos em que não há necessidade do uso de alguma unidade de processamento.

2.2.2 Memória

Memória é o elemento de um sistema computacional onde são gravados os dados a serem lidos/escritos por tal sistema. Existem vários tipos de memória, voláteis ou não.

Usualmente, os vários tipos de memória apresentam uma relação velocidade/capacidade de armazenamento/custo de fabricação. Por exemplo, tem-se a memória *cache* em computadores pessoais, que são uma memória fisicamente bem próxima do processador. Memórias do tipo *cache* são extremamente rápidas, porém com baixa capacidade de armazenamento e alto custo de produção.

Em sistemas embutidos, comumente usam-se memórias não voláteis (que não perdem seu conteúdo quando não energizadas), memórias de alta velocidade e baixa capacidade de armazenamento. Como a maioria das aplicações em que se usam sistemas embutidos é de baixa robustez e pequeno porte, a necessidade de grandes espaços de armazenamento na memória é desnecessária.

Em sistemas embutidos são usadas memórias do tipo *Flash*. Estas memórias são classificadas como EEPROM (*Electrically Erasable Programmable Read Only Memory*), operam em grande velocidade, consomem pouca energia e permitem muitas regravações por meio de sinais elétricos (WILMSHURST, 2007).

2.2.3 Entradas e Saídas

Entradas e saídas de um sistema embarcado determinam o número de interações que ele irá receber/atuar no ambiente em um ciclo de *clock*, ou seja, em um “passo” de processamento.

A comunicação na entrada/saída de cada porta pode ser serial ou paralela. No caso da comunicação serial, tem-se que vários *Bits* (unidade mínima no armazenamento/transmissão de dados) são transmitidos um atrás do outro, por uma mesma porta, ou seja, vários *Bits* fazem o mesmo caminho de um ponto a outro no sistema elétrico. Já na comunicação paralela, tem-se que cada saída será responsável por apenas um *Bit*. Por exemplo, transmitem-se oito *bits* (um *byte*) em oito portas diferentes por vez. Assim, quando se fizer necessária uma comunicação mais veloz, seriam necessárias várias portas na configuração paralela.

Nos sistemas microcontrolados em geral, é mais prático o uso de uma comunicação serial porque usa apenas uma porta para entrada de dados e uma para saída, fazendo assim com que o *hardware* fique mais simples e barato de ser implementado.

2.3 A placa Arduino

Arduino é uma plataforma eletrônica de protótipo *open-source* baseada em um *software* e *hardware* flexível, e de fácil uso (ARDUINO, 2012).

O Arduino usa um microcontrolador ATmega328 da ATMEL© de alta performance, combina 32KB de memória flash, com 1KB de memória EEPROM, 2KB de memória SRAM, 23 linhas de entrada/saída em geral, como frequência de operação máxima de 20 Mhz[...] (ARDUINO, 2012).

O Arduino pode sentir o ambiente em que está inserido, recebendo um entrada de uma grande variedade de sensores e pode afetar este ambiente controlando luzes, motores e outros atuadores (ARDUINO, 2012).

Como os esquemas do Arduino estão disponíveis, uma placa do Arduino pode ser a princípio, montada manualmente. Porém, uma placa produzida industrialmente consegue atingir um preço popular e trazer uma performance superior, o que permite a eliminação de problemas de natureza elétrica no desenvolvimento (DIMITROV, 2011).

A figura 2.2 mostra um exemplo da placa Arduino que está comercialmente disponível.

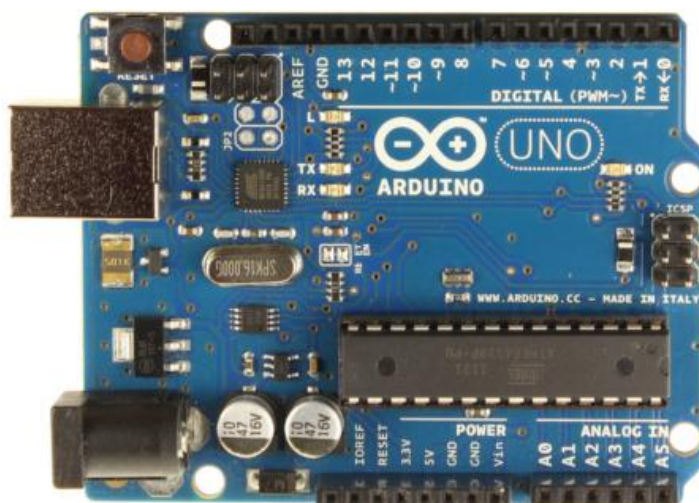


Figura 2.2 – Arduino UNO R3 visto de frente.

Fonte: ARDUINO, 2012.

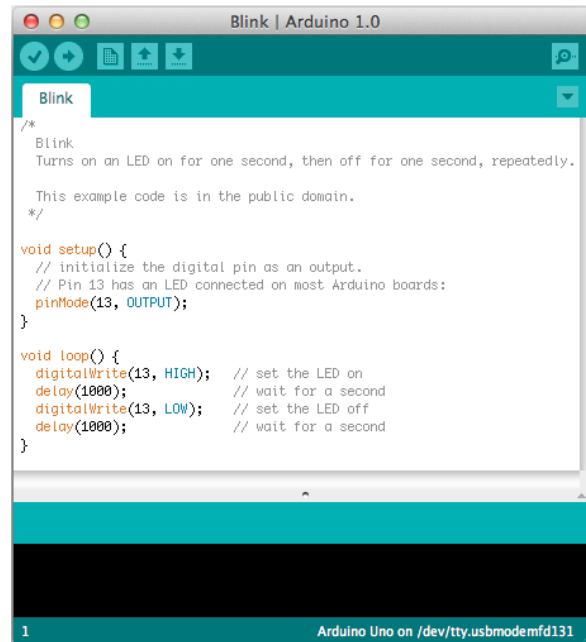
Além da placa do Arduino existem muitas outras placas, disponibilizadas pelo fabricante ou não, que adicionam funcionalidades a placa original encaixando-se perfeitamente na mesma. Dentre elas existem placas para diferentes tipos de comunicação (*Bluetooth*, *wi-fi*, *zigbee*, *ethernet*). Existem também placas de sensores, placas de acionamentos externos de motores, luzes, e outros atuadores. Todas estas placas são chamadas de *Shields*.

2.3.1 Linguagem de programação do Arduino

A linguagem de programação do Arduino é uma implementação do *Wiring*, uma plataforma computacional fisicamente similar ao Arduino, que é baseada em um ambiente de processamento de programação multimídia (ARDUINO, 2012).

Todos os programas feitos em Arduinos são chamados de *sketches*. Existe um *software* compilador disponibilizado pela Arduino para diferentes sistemas operacionais.

A figura 2.3 mostra um exemplo de um programa (*sketch*) simples.



```

Blink | Arduino 1.0
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/
void setup() {
// initialize the digital pin as an output.
// Pin 13 has an LED connected on most Arduino boards:
pinMode(13, OUTPUT);
}

void loop() {
digitalWrite(13, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(13, LOW); // set the LED off
delay(1000); // wait for a second
}

1 Arduino Uno on /dev/tty.usbmodemfd131

```

Figura 2.3 – *Sketch* exemplo Arduino.

Fonte: ARDUINO, 2012.

2.3.2 Shield Ardumoto

Para acionar atuadores, se faz necessário o uso de *drivers* de potência como o *shield* chamado de Ardumoto como o exemplificado na figura 2.4.

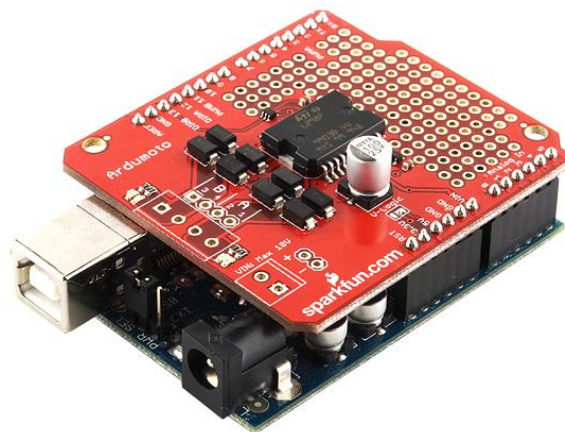


Figura 2.4 – Ardumoto conectado ao Arduino

O *shield* Ardumoto contém um circuito integrado L298 que apresenta duas pontes H internas. Cada ponte H tem quatro transistores funcionando como chaves que são capazes de acionar dois motores, em dois sentidos cada um. Um exemplo de uma ponte H é mostrado na figura

2.5.

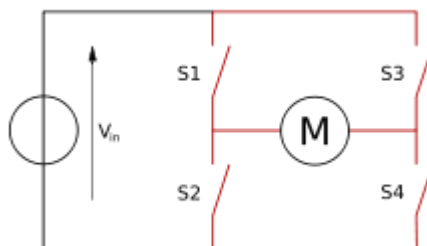


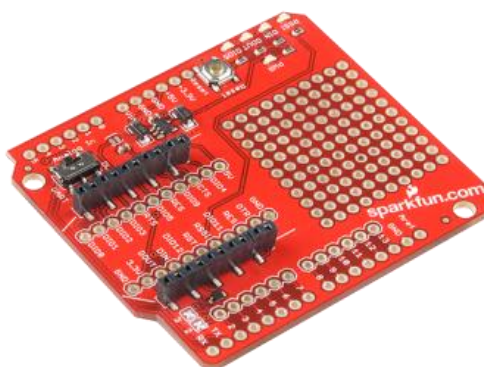
Figura 2.5 – Esquema da ponte H.

O comando das chaves 1 e 4 e das chaves 2 e 3 respectivamente são simultâneos. Assim, quando as chaves 1 e 4 estão acionadas a corrente pelo motor flui em um sentido fazendo o motor girar no sentido horário, quando as chaves 2 e 3 estão acionadas a corrente pelo motor flui no sentido inverso fazendo o motor girar no sentido anti-horário (FORESTI, 2006).

2.3.3 Shield XBee

Como uma alternativa para os padrões de comunicação, tem-se o protocolo ZigBee ou Xbee. Este protocolo será mais bem explorado no capítulo III. Como interface entre o microcontrolador e o XBee existe uma placa chamada de *shield* XBee como mostrado na figura 2.6. Este *shield* transmite a entrada/saída serial do Arduimoto diretamente para a placa do XBee. Como o Arduino funciona com uma tensão de pelo menos cinco volts (5V) e o XBee funciona com uma tensão de três vírgula três volts (3,3V), esse módulo faz essa conversão dos dados.

Este módulo também apresenta uma interface amigável com LEDs chaves e botões para que



se possa facilmente acessar os comando no Arduino através da placa do próprio *shield* XBee.

Figura 2.6 – *Shield* XBee

3 COMUNICAÇÃO SEM FIO

3.1 Introdução

A comunicação sem fio se faz necessária em várias aplicações de engenharia atualmente, na maior parte delas esta tecnologia é aplicada para garantir maior comodidade, portabilidade e segurança. Além disso, existem aplicações em que é inviável fisicamente o uso de fios para a comunicação.

A falta de padrão nas tecnologias sem fio, fez com que esta tecnologia demorasse a ganhar espaço entre os tipos seguros de transmissão de dados. Com esta falta de padrão, a comunicação sem fio poderia facilmente perder pacotes de dados conflitando com outras redes sem fio.

Com o objetivo de resolver a questão de incompatibilidade entre dispositivos de diversos fabricantes, o *Institute of Electrical and Electronic Engineers (IEEE)* e o *European Telecommunications Standardisation Institute (ETSI)* desenvolveram vários padrões para redes locais sem fio (DORNAN, 2001).

Com isso, as faixas de frequência de redes sem fio locais foram reguladas e licenciadas para o uso de determinadas faixas de equipamento.

A necessidade de comunicação sem fio começou a ser pesquisada para que se pudesse fazer transmissão de dados como a voz ou imagens. Com o passar do tempo viu-se que também existem aplicações onde pode-se usar com eficácia uma rede sem fio com baixa taxa de transmissão de dados, como por exemplo, sensores de campo em uma rede.

3.2 Padrões das redes sem fio

Nos dias atuais existem vários padrões de comunicação sem fio, sem os quais seria impossível que uma ou mais redes sem fio coexistissem. Existem diversos padrões e cada qual tem suas características técnicas que faz com que sejam mais adequados em determinada aplicação. Com isso, o desenvolvedor da aplicação na qual será usada a rede sem fio tem várias opções de escolha padronizadas que, a princípio, não irão conflitar com nenhuma outra rede.

3.2.1 IEEE 802.11

O IEEE 802.11 foi o primeiro a ser definido e utiliza como referência os protocolos da Ethernet convencional (MONTEBELLER, 2006).

Silva (2007, p. 15) diz que o grupo de trabalho 11 do IEEE, responsável pelos *standards* IEEE 802.11, considera a existência de quatro grandes grupos, em termos de redes sem fio. São eles WPAN, WLAN, WMAN e WWAN.

3.2.1.1 WPAN

WPAN (*Wireless Personal Area Network*) é definida pela IEEE como sendo a responsável pelas redes de pequeno alcance, até 100 metros. O grupo 15 da IEEE é o responsável pelas normas desse tipo de rede sem fio. Pertencem a esse grupo de normas os PC's e seus periféricos, *gadgets* em geral e também as tecnologias muito usadas atualmente *Bluetooth*, *IrDA* e o *ZigBee*. Estes últimos têm cada um, sua própria subdivisão dentro da IEEE. O subgrupo 802.15.1 rege as normas utilizadas pelo *Bluetooth*, assim como o subgrupo 802.15.4 é responsável pelas normas de utilização do padrão *ZigBee*.

3.2.1.2 WLAN

O grupo WLAN (*Wireless Local Area Network*), que hoje em dia está bem difundido no uso residencial devido à grande evolução e conseqüente redução de custos de fabricação de equipamentos para internet sem fio, é a tecnologia de redes sem fio com alcance de até 300 metros.

3.2.1.3 WMAN e WWAN

Os grupos WMAN (*Wireless Metropolitan Area Network*) e WWAN (*Wireless Wide Area Network*) são responsáveis pelas redes com maiores alcances, da ordem de quilômetros. Enquanto o WMAN é relativo aos acessos banda larga em regiões metropolitanas, o WWAN é dedicado a redes de alcances ainda maiores (da ordem de até milhares de quilômetros) e normalmente utilizado para serviços de transmissão de voz e dados.

3.2.2 Bluetooth

O padrão de comunicação *Bluetooth* foi desenvolvido pela Ericsson em 1994, apesar de somente ser formalizado em 1999, ano em que foi formado o SIG (*Bluetooth Special Interest Group*).

O nome *Bluetooth* tem sua origem no nome do rei Harald Bluetooth Gormson (911-986), conhecido como Harald I *of Denmark*, que foi um rei do século dez que se ocupou com a diplomacia, facilitando a negociação entre diferentes grupos (MONSIGNORE, 2007).

Este padrão é normalmente indicado para quando existe uma rede com uma grande taxa de transmissão de dados (em torno de 1Mbps), existem poucos nós e não é necessária grande distância na comunicação.

A aplicabilidade do *Bluetooth* em eletrônicos deve-se ao fato de que é uma tecnologia barata e com capacidade de transmissão de dados que atende grande parte dos eletrodomésticos. Em contrapartida, essa tecnologia não permite em sua rede uma grande quantidade de nós, o que faz com que ela não tenha quase nenhuma aplicação industrialmente.

3.2.3 ZigBee

ZigBee é um novo padrão para redes sem fio, sua principal característica é o baixo consumo de energia.

Anteriormente a responsável por esta tecnologia era a empresa Philips. O nome do protocolo nessa época era Home RF, porém não existia uma padronização bem estabelecida.

ZigBee corresponde ao ziguezaguear das abelhas (*Bees*, em inglês) daí o nome, Zig – Bee. O nome ZigBee corresponde ao padrão que utiliza a norma IEEE 802.15.4, que é fruto da ZigBee Alliance. Esta é uma associação que foi criada no ano de 2002 com o objetivo de criar e padronizar um dispositivo de baixo e com a principal vantagem sendo o baixo consumo de energia.

Uma rede ZigBee também se caracteriza por poder conter uma grande quantidade de nós. De acordo com Maxtream (2013) um coordenador aguenta até 65535 nós em sua rede.

O padrão ZigBee possui diversas funcionalidades que foram desenvolvidas para atender as necessidades do mercado, principalmente na área de automação (ESCHNER, 2011).

O ZigBee opera na banda ISM (*Industrial, Scientific and Medical*). Estas bandas ISM são isentas de licenciamento, ou seja, seu uso é liberado para equipamentos de pequeno alcance e somente dentro das faixas permitidas, que são dependentes do país que será usado.

O módulo mais conhecido da ZigBee Alliance chama-se XBee, como mostra a figura 3.1.



Figura 3.1 – Módulo XBee.

Fonte: MAXSTREAM, 2013.

Existem diferentes módulos XBee, cada qual com sua particularidade. Em geral estes módulos se diferenciam pelo seu alcance e conseqüente consumo energético.

3.2.3.1 Topologias de Rede

O padrão ZigBee permite o uso de três tipos diferentes de topologias de rede (*Star*, *Cluster tree* e *Mesh*). Cada uma delas possui suas características que, dependendo da aplicação, podem ser implementadas. Elas serão detalhadas a seguir.

O ZigBee se encaixa dentro da categoria de uma WPAN (*Wireless Personal Area Network*) que, como foi falado anteriormente, são redes de pequeno alcance.

Existem três tipos de funções que um dispositivo XBee pode executar em uma rede do padrão ZigBee. São elas coordenador, roteador e dispositivo final.

- O coordenador (*coordinator*) é responsável por reconhecer todos os outros dispositivos, designando seus endereços na rede;
- O roteador (*router*) pode ser usado para poder aumentar o alcance da rede, reproduzindo o sinal recebido para os próximos nós;
- O dispositivo final (*end device*) é o que funciona comumente como sensor, coletando dados a sua volta ou então comandando um atuador, modificando o

ambiente a sua volta.

Quando usado como dispositivo final, pode-se usar um dispositivo da classe de função reduzida (*Reduced Function Device* - RFD) o que possibilita com que ele gaste menos energia elétrica. De acordo com Silva (2007) um dispositivo RFD tem construção mais simples comparado com um dispositivo de função completo (*Full Function Device* - FFD), assim sendo, ele não pode atuar como coordenador em uma rede ZigBee. Como foi projetado para que pudesse ser usado em lugares afastados, o consumo de pouca energia faz com que esse dispositivo possa ter longa vida usando como fonte de energia apenas baterias.

Existem também dois modos de operação de cada nó XBee, o modo *beaconing* e o *non-beaconing*:

- No modo *beaconing*, os nós ZigBee *routers* transmitem periodicamente sinalização (*beacons*) a confirmar sua presença aos outros nós da mesma rede (SILVA, 2007). Assim, os módulos economizam energia não precisando estar totalmente ativos o tempo inteiro para receber alguma mensagem;
- No modo *non-beaconing* todos os módulos ficam ativos o tempo inteiro, nunca entrando em *sleep*.

3.2.3.1.1 Star (Estrela)

Na topologia estrela mostrada na figura 3.2, existe apenas um coordenador de rede que envia as informações da rede igualmente para todos os dispositivos finais. Nessa topologia, os dispositivos finais não se comunicam entre eles, para uma informação ir de dispositivo final para outro dispositivo final esta informação deve antes passar pelo coordenador. Essa forma de topologia é apropriada em casos onde se concentra o controle da aplicação em um único ponto, e os outros pontos são sensores que enviam informações ou então apenas agem nos seus atuadores individuais. Usualmente o dispositivo coordenador está conectado a algum dispositivo de controle, seja ele um microcontrolador, um PC ou um dispositivo lógico programável (CLP) ou algum outro tipo de controlador, dedicado ou não.



Figura 3.2 – Topologia em estrela.

Fonte: Adaptado de ESCHNER, 2011.

3.2.3.1.2 Cluster Tree (Árvore)

Na topologia em árvore mostrada na figura 3.3 existe uma hierarquia que não se encontra presente na topologia estrela. Nela, o coordenador é o dispositivo principal da rede e envia os dados para os roteadores repassarem aos dispositivos finais ou então os dados são passados diretamente para um dispositivo final. Cada “galho” pode ou não conter um dispositivo roteador, mas para ser chamada de topologia em árvore tem que haver pelo menos um “galho” com pelo menos um dispositivo roteador e também pelo menos um dispositivo final.

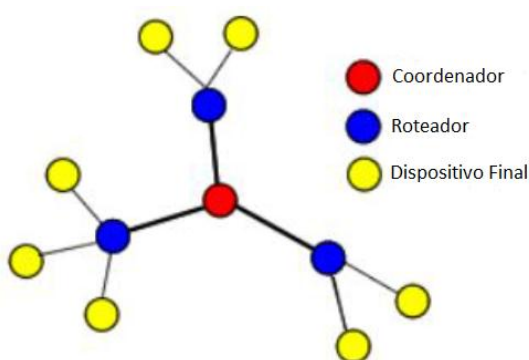


Figura 3.3 – Topologia em árvore.

Fonte: Adaptado de ESCHNER, 2011.

3.2.3.1.3 Mesh (Malha)

A topologia em malha que também é chamada de ponto-a-ponto é a mais complexa que pode existir dentre as três. Como há um tráfego de dados fluindo entre o coordenador e os roteadores o tempo todo, como mostrado na figura 3.4, a própria rede acha o caminho mais fácil e rápido para estas informações trafegarem. Esta rede é mais eficaz do que a rede em árvore do ponto de vista da redundância dos roteadores, porque caso algum roteador dê algum

problema ou for retirado da rede, a mesma pode achar outro caminho para que não haja perda de dados. O problema é que essa redundância tem um viés: São necessários muitos dispositivos roteadores para que a rede seja eficaz.

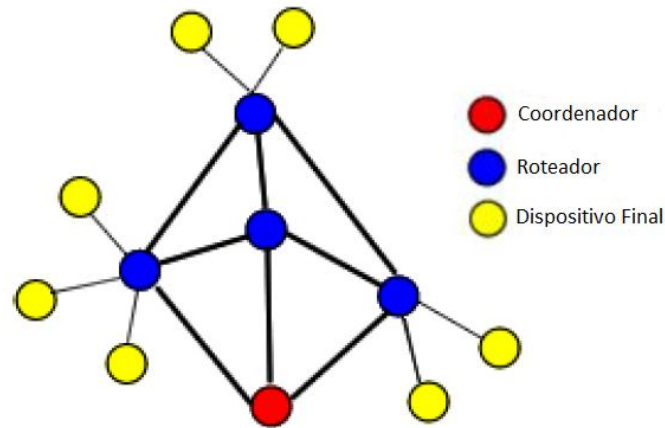


Figura 3.4 – Topologia em malha.

Fonte: Adaptado de ESCHNER, 2011.

3.2.3.1 Modelo OSI

A arquitetura do padrão ZigBee é baseado no modelo de camadas OSI (*Open Systems Interconnection*) criado pela ISO (*International Standards Organization*). Como define Forauzan et al. (2008) são sete as camadas da arquitetura OSI: Camada de aplicativo, camada de apresentação, camada de sessão, camada de transporte, camada de rede, camada de enlace de dados e camada física.

Embora se baseie no modelo OSI de sete camadas, a arquitetura protocolar ZigBee apenas define, no entanto, as camadas de interesse para atingir as funcionalidades desejadas (SILVA, 2007).

Conforme a figura 3.5 as camadas PHY (camada física) e MAC (*Medium Access Control*) são definidas pelo padrão IEEE 802.15.4, e as três camadas superiores são definidas pela *ZigBee Alliance*.

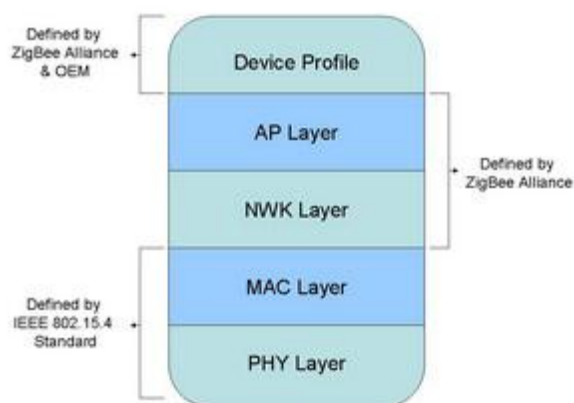


Figura 3.5 – Camadas da arquitetura protocolar *ZigBee*

Fonte: SILVA, 2007.

A camada PHY coordena as funções exigidas para transportar um fluxo de *bits* por um meio físico. Ela lida com as especificações mecânicas e elétricas da interface e do meio de transmissão (FOROUZAN et al., 2008).

À camada MAC cabe fundamentalmente o papel de controlar o acesso aos canais RF (rádio frequência), utilizando mecanismos de prevenção de colisão CSMA-CA (*Carrier Sense Multiple Accesses – Collision Avoidance*) (SILVA, 2007).

De acordo com Forouzan et al. (2008) a camada NWK, análoga à camada de rede do modelo OSI, é responsável pelo envio de um pacote, da origem ao destino, passando possivelmente por várias redes (*links*).

Quanto a camada de aplicação (APL), contém a sub-camada *Application Support Sublayer* (APS), o *ZigBee Device Object* (ZDO) e a *Application Framework* (AF). Esta camada pretende assegurar uma correta gestão e suporte para as diversas aplicações (SILVA, 2007).

A figura 3.6 mostra com mais detalhes a arquitetura protocolar *ZigBee*, focando em melhor especificar a camada de aplicação e suas subcamadas.

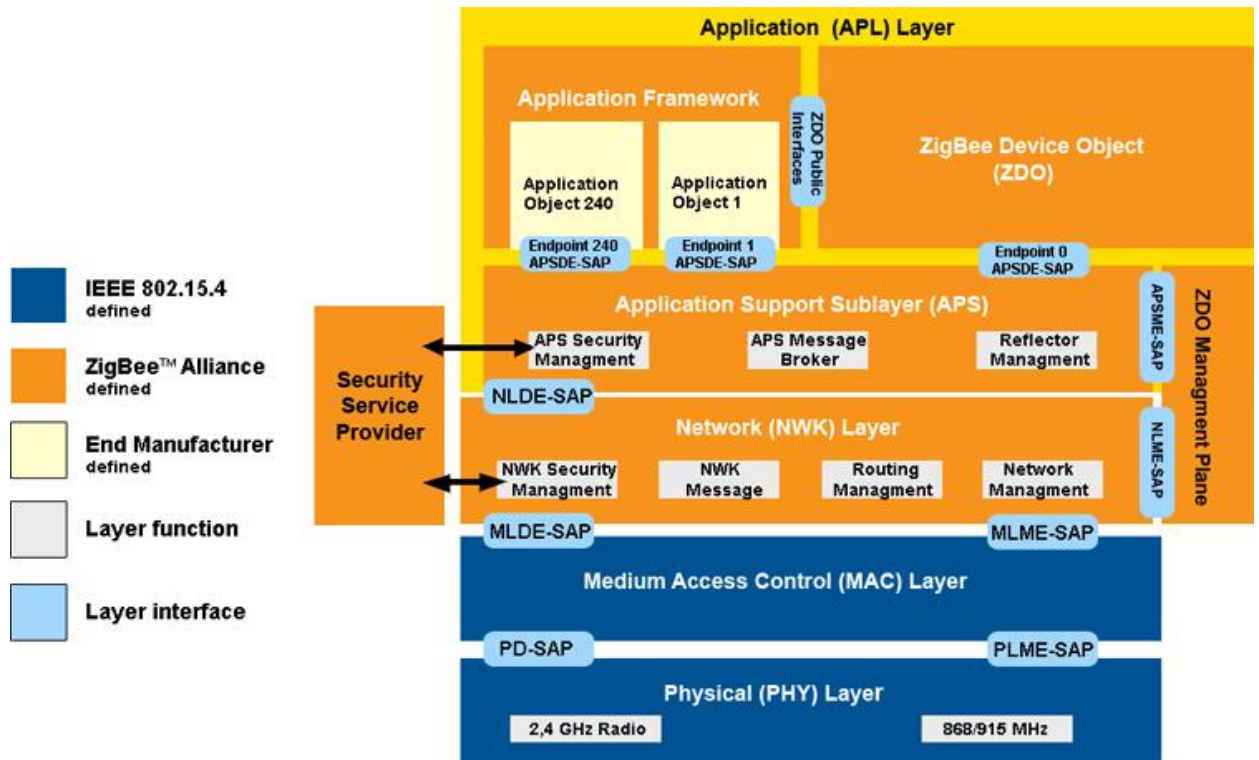


Figura 3.6 – Detalhes da arquitetura protocolar *ZigBee*.

Fonte: SILVA, 2007.

3.3 Comparativo entre os padrões

Em qualquer aplicação de engenharia uma ferramenta é dita melhor que a outra, se atende melhor as especificações de determinada aplicação, ou seja, não existe ferramenta “melhor” ou “pior” absolutamente, tudo depende de onde e como esta ferramenta será usada. Esta ideia é válida para o caso de padrões de comunicação, em cada aplicação será escolhido e utilizado o melhor padrão que atende aquela aplicação.

Com isso em mente a tabela 3.1 faz um comparativo entre as principais características de três padrões diferentes de comunicação sem fio.

A figura 3.7 mostra uma relação entre a taxa de transmissão e o alcance de alguns dos padrões de comunicação sem fio mais usados atualmente.

Tabela 3.1 – Comparativo entre padrões de comunicação sem fio.

ESPECIFICAÇÃO	DÉBITO	CONSUMO	PILHA PROTOCOLAR	VANTAGENS	PRINCIPAIS APLICAÇÕES
Wi-Fi (IEEE 802.11b/g)	54Mbps	>400mA TX, <i>standby</i> 20mA	1MB +	Elevada Taxa de Transferência	Internet, Transferências de dados, Vídeo/Aúdio
Bluetooth (IEEE 802.15.1)	1Mbps	>400mA TX, <i>standby</i> 0.20mA	≈ 250KB	Interoperabilidade, substituição de cabos	Periféricos de PC e Celulares, PDA's
ZigBee (IEEE 802.15.4)	100kbps	30mA TX, <i>standby</i> 0.20µA	≈ 32KB	Consumo, Latência, N.º de Nós, Fiabilidade, R\$	Controle Remoto, Sensores, Dispositivos alimentados por bateria

Fonte: Adaptado de SILVA, 2007.

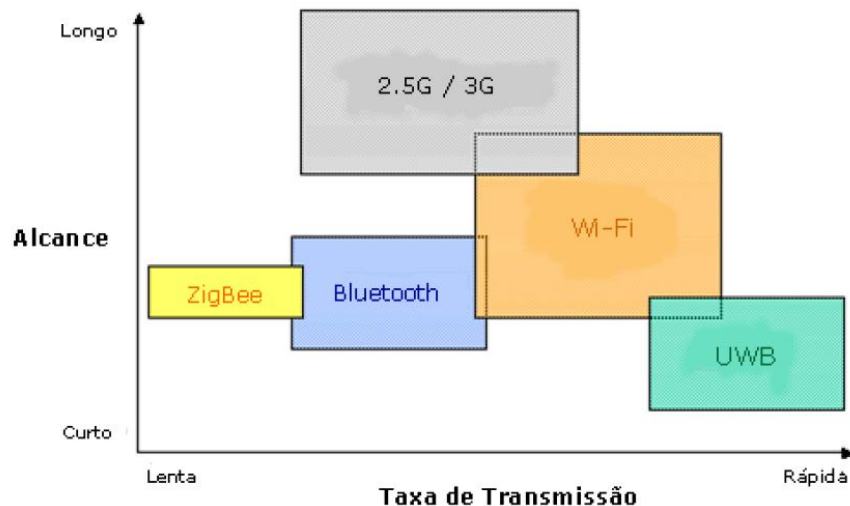


Figura 3.7 – Representação gráfica da taxa de transferência de dados *versus* alcance.

Fonte: MAXTREAM, 2013.

Analisando a tabela 3.1 e a figura 3.7 conclui-se, em grosso modo, que o padrão Wi-Fi é mais adequado para aplicações que não tem consumo de energia limitado e necessitam de uma grande taxa de transferência de dados. O padrão *ZigBee* é mais adequado para uma aplicação que não necessita de muita velocidade na transmissão dos dados, porém deve ter baixo consumo energético. Enquanto que o padrão *Bluetooth* é um intermediário entre estes outros dois padrões.

4 RECONHECIMENTO DE IMAGEM

4.1 Introdução

O reconhecimento de imagem designa uma área de estudos que está em constante crescimento nos últimos anos. Uma das razões deste crescimento consiste na sua aplicabilidade. Vários campos da ciência podem se beneficiar de seu uso adequado, como por exemplo, áreas da medicina, de segurança e industriais.

4.2 Digitalização

Diferentemente de uma imagem capturada pelos olhos humanos, que possuem noção de espaço tridimensional, uma imagem capturada por um dispositivo eletrônico possui apenas duas dimensões. Quando uma imagem for capturada por tal dispositivo, esta consiste em um plano (x,y) formado por diversos *pixels*.

Um pixel é uma unidade elementar de uma imagem digitalizada. Quando uma imagem é digitalizada, significa que ela foi discretizada transformando uma quantidade imensa de pontos em um número palpável de pontos.

Dependendo de vários fatores como a resolução da câmera, sua capacidade de processamento e sua capacidade de foco uma imagem pode ser obtida com diferentes resoluções e qualidades. Quanto mais *pixels* esta imagem tiver, mais parecida com a visão humana ela será, porém exigirá com que o equipamento utilizado para capturá-la tenha uma capacidade maior e melhor de processamento e dependendo da aplicação, isto não é necessário.

4.2.1 Resolução em pixels

O termo resolução em *pixels* indica quantos *pixels* uma imagem digitalizada contém. Por exemplo, se uma imagem é dita como 320x240 *pixels*, significa que ela contém 320 pontos de largura e 240 pontos de altura. Estes valores não definem o tamanho da imagem, mas sim a quantidade de pontos. Existe outro atributo que fornece a quantidade de *pixels* por unidade de área, este atributo é comumente relacionado à quantidade de *pixels* por polegada (ppi – *pixel per inch*).

Para uma imagem então ser próxima do que o olho humano vê, ela deve conter uma grande quantidade de *pixels* por polegada além de uma grande quantidade de *pixels* total, e também deve ter sido bem focada e processada por um dispositivo de boa qualidade.

4.2.2 Cores

A maioria de imagens no mundo real não é monocromática, mas sim colorida e por isso, além da quantidade de pontos, ao digitalizar uma imagem é necessário que cada um destes pontos esteja relacionado a uma cor.

4.2.2.1 RGB

O espaço de cores RGB (*red*, *green* e *blue*) é a designação de cores mais usada em sistemas computacionais. De acordo com Ribeiro (2005) cada uma destes componentes de cor faz uso de oito *bits* ou um *byte* cada, perfazendo o total de 24 *bits*. O cubo mostrado na figura 4.1 representa as possibilidades de cor que o espaço de cores RGB pode conter. Este sistema permite a formação de 16,8 milhões de possibilidades de cor.

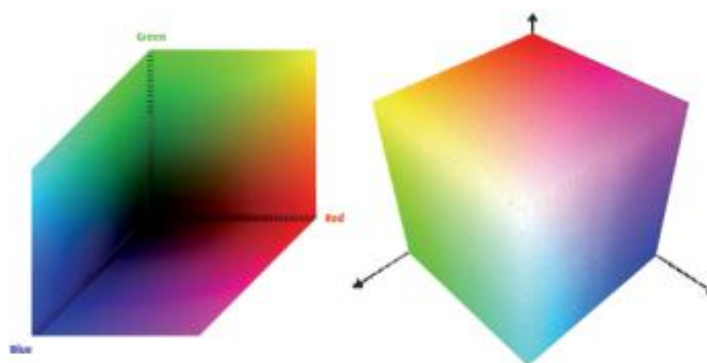


Figura 4.1 – Plano cartesiano de cores RGB e cubo RGB.

Fonte: RIBEIRO, 2005.

Utilizando oito *bits* para cada uma das cores fundamentais há um número entre 0 e 255 que é usado no reconhecimento de imagem para definir o “quanto” de vermelho, verde e azul existe em determinado pixel. Desta forma, depois de definidos alguns filtros e limites, o computador pode usar uma imagem e “saber” qual cor tem uma determinada imagem.

4.2.2.2 HSI

Uma outra forma de classificação das cores é a chamada HSI (*hue, saturation e intensity*). Segundo Facon et al. (1997) este modelo de cores é uma forma mais próxima da manipulação natural de cor utilizada pelo ser humano.

Hue, “H”, tonalidade ou ainda matiz descreve a cor pura. O “S” de *saturation* ou saturação representa o grau de pureza desta cor e o “I” de *intensity* ou intensidade diz respeito a intensidade de luz da cor (brilho).

As vantagens desse sistema de representação da cor encontram-se na possibilidade de separar a intensidade da informação tonalidade e saturação, bem como, na relação que existe entre essas componentes [...] (FACON et al., 1997).

A figura 4.2 mostra a representação de cores HSI onde “I” é a intensidade da cor, “H” a tonalidade e “S” a saturação.

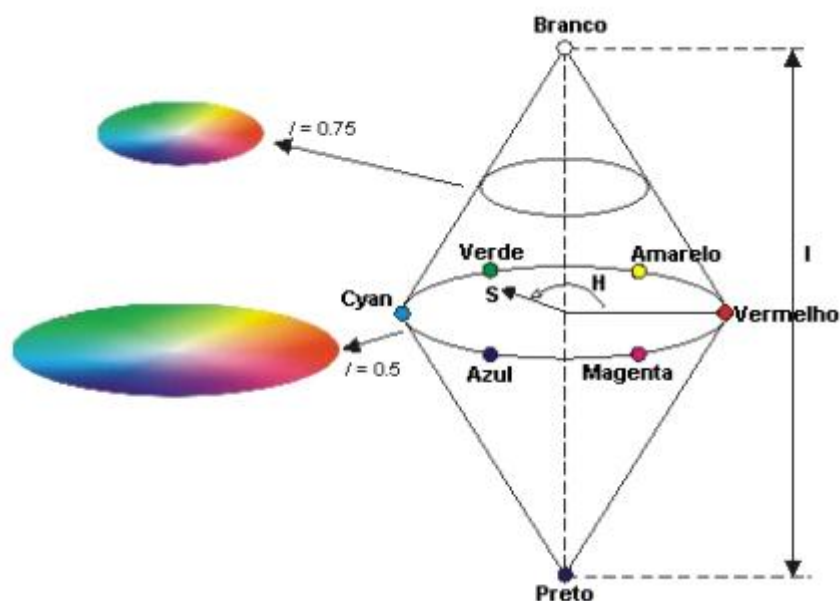


Figura 4.2 – Representação de cores HSI.

Fonte: MATOS et al., 2007.

4.2.2.3 Comparação entre HSI e RGB

Os modelos de representação de cor RGB e HSI são os mais utilizados atualmente, apesar de existirem outros modelos, suas vantagens e desvantagens dependem basicamente de sua aplicação.

De acordo com Facon et al. (1997) o modelo RGB deve se destacar em aplicações onde não haja variação de luminosidade, como a formação de sombras, e quando seus componentes de cor e intensidade estão muito correlacionados.

Segundo Facon et al. (1997) o modelo HSI se destaca no reconhecimento de cores onde há uma variação na luminosidade do local, porque nele é possível que se isole o “I” que computacionalmente será um número responsável por essa intensidade. Com isso, o reconhecimento fica menos dependente da intensidade da cor e mais dependendo da sua tonalidade e da pureza dessa tonalidade.

4.3 Reconhecimento

O reconhecimento da imagem propriamente dito consiste em classificar esta imagem

digitalizada de acordo com padrões para que a imagem se transforme em uma informação que o computador correlacione com o mundo real.

Para que se faça o reconhecimento de imagem de um determinado problema, a resolução da imagem digitalizada deve ser adequada. Se a resolução for muito grande, talvez o processamento seja custoso e inviável e caso a resolução seja muito pequena pode ser que não seja possível reconhecer devidamente o elemento desejado.

O grande desafio no reconhecimento de imagem consiste na variação de cor, tonalidade e brilho das imagens obtidas, o que normalmente acontece em aplicações que interagem com o ambiente.

4.4 A plataforma Flash

Para fazer o processamento e reconhecimento da imagem digitalizada se faz necessário o uso ou desenvolvimento de um *software* adequado para tal fim.

Uma dessas possibilidades é o *software* da empresa Adobe® chamado Flash® para tal processamento. Este *software* adquire as imagens em intervalos constantes de uma interface com o padrão USB (*Universal Serial Bus*) e faz o seu processamento. Esta plataforma já possui várias ferramentas integradas desenvolvidas especificamente para a captura da imagem.

A plataforma Flash® utiliza a linguagem de desenvolvimento orientada a objetos própria chamada ActionScript. A linguagem ActionScript já está em sua terceira versão (ActionScript 3.0). Apesar de ser própria, tal linguagem apresenta muita similaridade com outras linguagens que são mais difundidas como o Java e o C++.

4.5 Sistema de reconhecimento da imagem

Para separar cores normalmente traça-se limites para o que será considerado uma cor com relevância para a aplicação. Como no sistema RGB existem 16,8 milhões de possibilidades de cor, normalmente esse número é reduzido através de filtros pra que o sistema aproxime cada uma dessas possibilidades a uma cor definida *a priori* que tenha relevância na aplicação.

Em sistemas de engenharia dependentes de um processamento de imagem, esta não deve nunca falhar, pois no caso de uma falha ou interpretação errônea de uma imagem, o sistema de controle irá considerar esta informação como certa e atuar no ambiente equivocadamente.

Um sistema de reconhecimento de imagem deve também estar preparado para várias situações de variação de cor, tonalidade e brilho na imagem, porque tais variações podem também produzir uma resposta diferente e equivocada do controlador computacional.

5 O JOGO E SUAS REGRAS

5.1 Introdução

Um jogo de futebol de robôs pode ter diversos tipos e regras, dependendo de sua categoria e do órgão ou associação que o organiza. Todavia todos eles consistem em um mesmo princípio básico análogo ao futebol praticado por pessoas: Quem fizer mais gols vence.

5.2 História

Existem divergências na literatura sobre quem realmente idealizou os primeiros robôs jogadores de futebol.

De acordo com Fira (2013), esta federação foi a primeira a organizar competições de futebol de robôs no mundo, clamando que a ideia de futebol de robôs foi originada na Korea em setembro de 1995 pelo Professor Jong-Hwan Kim da cidade de KAIST.

Segundo Kitano (1997), a primeira competição mundial de futebol de robôs, chamada de *RoboCup*, aconteceu em Nagoya, no Japão, no ano de 1997 e foi originada a partir da menção sobre futebol de robôs feita em 1992 pelo Professor Alan Mackworth da *University of British Columbia* no Canadá.

5.3 Algumas das categorias do futebol de robôs.

Existem diversas categorias de futebol de robô e existem principalmente dois órgãos responsáveis pelas definições de tais categorias em competições, são eles: A FIRA (*Federation of International Robot-Soccer Association*) e a *RoboCup*.

Neste trabalho apresentar-se-á algumas das categorias mais difundidas no Brasil, que em sua maioria seguem as divisões e regras da *RoboCup*. Atualmente a *RoboCup* também rege competições de resgate, simulações e outras competições, porém neste trabalho serão somente tratadas as competições relacionadas ao futebol.

5.2.1 Humanóide

Na liga de robôs humanóides da categoria RoboCup Humanoid os robôs devem ser construídos de forma a assemelhar-se ao corpo humano. De acordo com as regras da IEEE (2008) esta categoria se subdivide em duas classe: A *KidSize*, (com robôs de 30 a 60 centímetros de altura) e a *TeenSize* com robôs maiores (entre 80 e 130 centímetros de altura).

Todos os robôs devem ser autônomos, sendo controlados por um processador interno. Na classe *KidSize* os times são compostos por até dois robôs enquanto que na classe *TeenSize* cada time tem apenas um robô que se reveza no papel de atacante e defensor (goleiro).

Esta categoria está entre uma das que apresentam maior dificuldade de implementação, porque além de apresentar os mesmos desafios de outras categorias na questão na visão computacional e programação da estratégia, e ainda apresenta uma grande dificuldade na construção mecânica do robô.

Andar dinamicamente, correr e chutar a bola enquanto gerenciar o equilíbrio, a percepção visual da bola, de outros jogadores, do campo, a localização própria e o jogo em equipe estão dentre os vários desafios da pesquisa a ser desenvolvida na liga Humanóide (CBROBOTICA, 2013).

Existem competições de robôs humanóides que utilizam apenas robôs que são comercializados, como o ultrapassado AIBO® da Sony® e atualmente o NAO®.



Figura 5.1 – NAO®, exemplo de robô humanóide

Fonte: ALDEBARAN ROBOTICS, 2013.

5.2.2 *Small Size*

A categoria que possui o maior número de participantes no Brasil é a categoria *Small Size* também chamada de F180. Nesta categoria, cada time tem até cinco robôs com dimensões máximas de 15 centímetros de altura e 18 centímetros de diâmetro.

Nesta categoria os robôs podem ser controlados por um processador interno ou então por um computador externo se comunicando com eles. Como a quantidade de dados a ser processada é muito grande, comumente se usa um computador externo para este processamento.

A visão computacional em competições da *RoboCup* para a categoria F180 é processada e fornecida pelo computador da organização, fazendo com que este seja um desafio a menos desta, em relação a outras categorias.

Cada time deve ter *patches* coloridas padronizadas colocadas em cima de seus robôs para identificação dos mesmos.

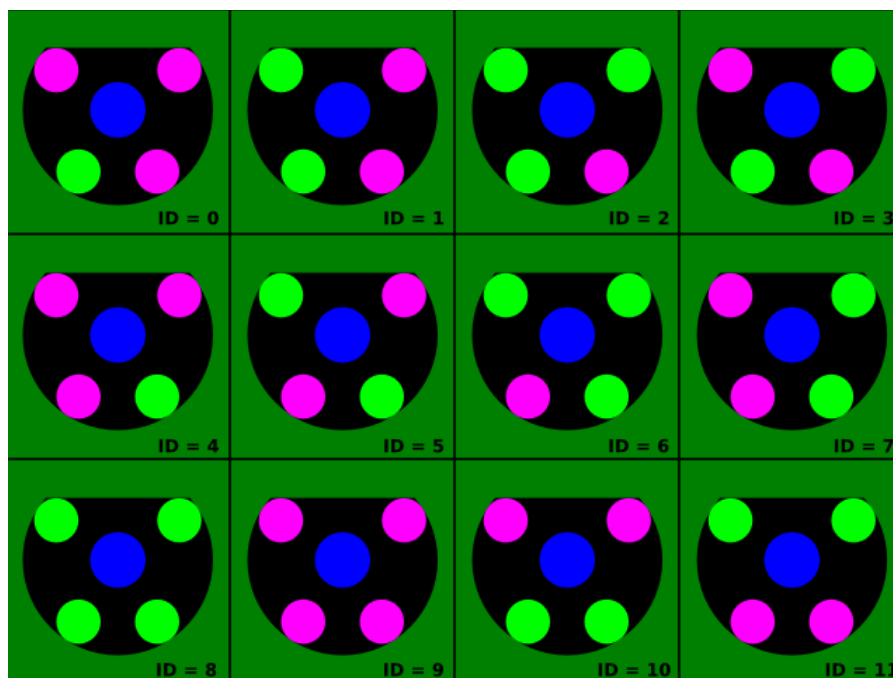


Figura 5.2 – Padrões de cores usadas na *RoboCup* 2012.

Fonte: CBROBOTICA, 2013.

5.2.3 IEEE Very Small Size

A categoria *IEEE Very Small Size* é regulamentada no Brasil pela IEEE e suas regras são baseadas na categoria MIROSOT da FIRA (CBROBOTICA, 2013).

Para jogar uma partida desta categoria, um time deve ter entre um e no máximo de três jogadores, sendo que as dimensões máximas de cada robô não podem exceder um cubo de 7,5 centímetros de lado, exceto por sua antena de comunicação que pode ultrapassar estas medidas, e um uniforme com no máximo 0,5 centímetros que pode ser colocado por cima de cada robô.

5.2.3.1 Resumo das regras IEEE Very Small Size

Como todas as categorias de competições de robôs, a categoria *IEEE Very Small Size* apresenta um conjunto de regras que devem ser seguidas e suas consequentes punições em caso de não seguimento.

5.2.3.1.1 Dos robôs

Cada robô pode possuir braços, pernas ou dispositivos de chutes desde que não ultrapassem as medidas estabelecidas e que a bola não seja encoberta a mais de 30% da sua área, conforme mostra a figura 5.3.

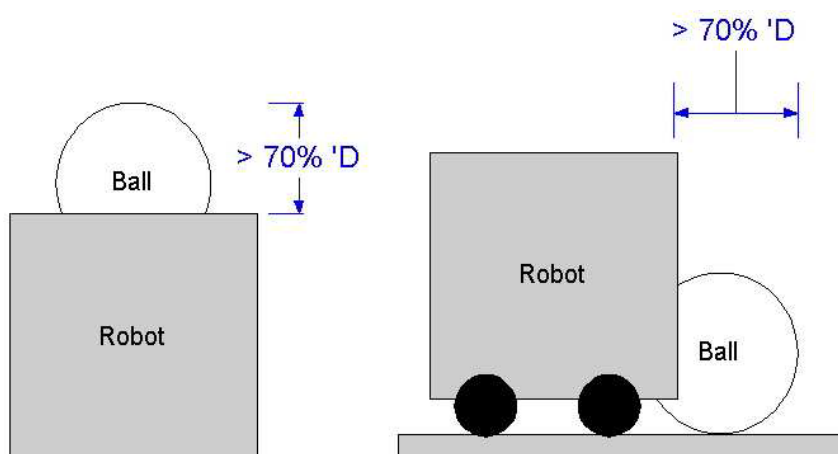


Figura 5.3 – Regras referentes ao robô e a bola.

Fonte: CBROBOTICA, 2013.

A parte de cima do robô não deve conter as cores laranja, branco ou cinza e não devem ter mais de duas cores diferentes de preto. Um *patch* de cor azul ou amarelo vai ser designado pelos organizadores, irão identificar os robôs de cada time (CBROBOTICA, 2013).

Estes *patches* azuis e amarelos irão ser utilizados pela visão computacional para identificar todos os robôs de um determinado time, que deve saber qual é a cor de seu *patch* (azul ou amarelo) antes de ter início a partida.

Todos os robôs podem ter também, visível em seu topo, uma região sólida de qualquer outra cor (que não seja amarelo, azul, preto, branco ou cinza) para identificação individual de seus jogadores como é exemplificado na figura 5.4.

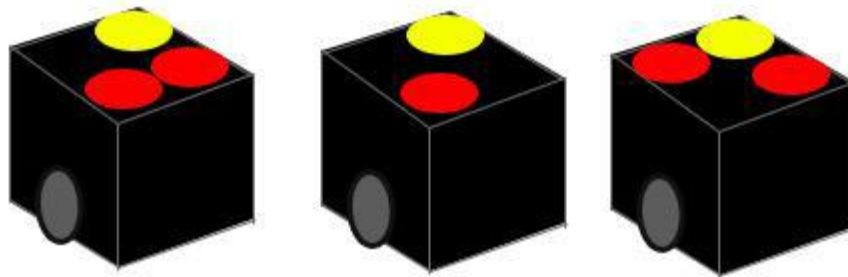


Figura 5.4 – Exemplos de robôs com seus *patches*.

Cada robô deve ser totalmente independente, com mecanismos de motorização e fornecimento de energia próprios. Somente comunicação sem fio será permitida para todas as interações entre o computador controlador e o robô (CBROBOTICA, 2013).

5.2.3.1.2 Do campo

Um campo preto (não reflexivo) de madeira retangular com as medidas 150 por 130 centímetros com cinco centímetros de altura e 2,5 centímetros de espessura das paredes laterais deve ser usado (CBROBOTICA, 2013).

No campo devem conter seis marcações brancas para designar os pontos como especificados na figura 5.5 onde serão cobradas as faltas (*freekicks* ou FK), os “*freeball*” (FB) e os pênaltis (*pênalti-kicks* ou PK).

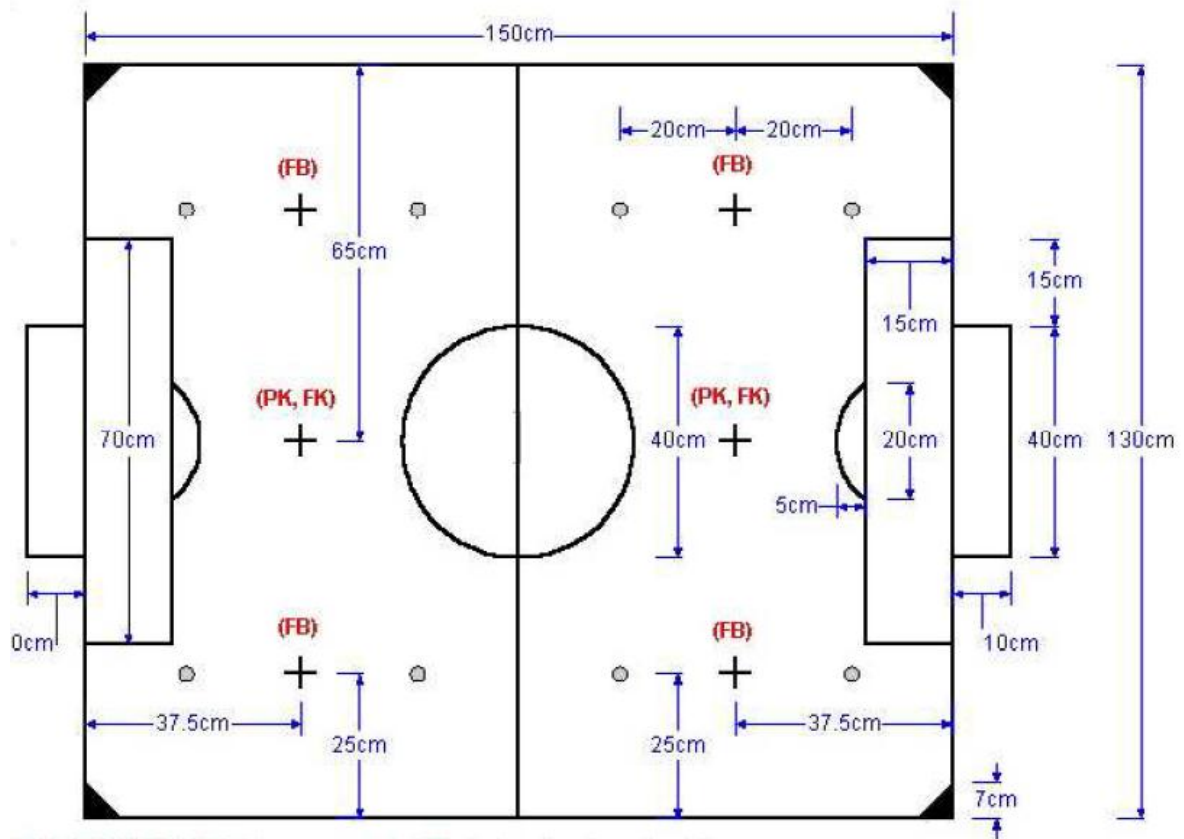


Figura 5.5 – Dimensões do campo.

Fonte: CBROBOTICA, 2013.

5.2.3.1.3 Da bola

Deverá ser usada uma bola de golfe laranja com 42,7 milímetros de diâmetro e 46 gramas de peso (CBROBOTICA, 2013). A figura 5.6 exemplifica a bola que deve ser usada.



Figura 5.6 – A bola do jogo.

5.2.3.1.3 Da partida

Segundo as regras oficiais CBROBOTICA, (2013) a duração da partida deverá ser de dois tempos de cinco minutos cada, com um intervalo entre eles de dez minutos e se algum time não estiver pronto para retornar, conceder-se-á mais cinco minutos.

Cada time também tem direito a dois tempos de dois minutos cada um a serem pedidos a qualquer momento da partida, mas que só serão concedidos a critério do árbitro.

Existem diversas circunstâncias que serão marcadas faltas, algumas delas são:

- Mais de um robô de um mesmo time ao mesmo instante dentro da área;
- Um robô de um time empurrar o robô do outro time;
- Falha de um robô designado como goleiro (aquele que está dentro de sua área) em tirar a bola de sua área em dez segundos.

O chamado *freeball* (bola-livre) consiste em uma infração em que o jogo fique de alguma forma travado, seja porque a bola está presa ou porque todos os robôs estão presos em algum lugar. Quando ocorre o *freeball*, a bola é colocada pelo arbitro no quadrante em que ela ficou presa em um ponto que é mais facilmente alcançado pelos robôs e os robôs de cada time são colocadas equidistancialmente da bola em marcações específicas.

Ao final da partida, o time que tiver feito mais gols e não tiver cometido nenhuma infração punível de desclassificação vence.

6 ESTRATÉGIAS

6.1 Introdução

Na categoria *IEEE Very Small Size*, bem como na categoria *Small Size* normalmente se faz uso de um computador para definições em tempo real das estratégias a serem tomadas em um jogo, isso pelo fato dos robôs serem pequenos e não suportarem dentro deles processadores com tamanho suficiente para suprir toda a demanda de cálculos necessária.

Em uma competição de robôs as estratégias tomadas e sua eficiente implementação são a maior diferença entre um time e outro. Esta estratégia consiste na tomada de decisões em tempo real do que cada robô deve fazer individualmente, para que, em conjunto estes robôs formem um time.

Existem diversas ferramentas teóricas que podem ser aplicadas na elaboração de uma estratégia para um time de futebol de robôs. Dentre elas se destacam ferramentas de inteligência artificial como as redes neurais artificiais (RNA) e a lógica *Fuzzy* implementada em um controlador. Uma outra ferramenta matemática que pode ser usada na tomada da decisão da trajetória de um robô é a técnica dos “campos potências”.

Serão mostradas algumas dessas técnicas, mas especificamente a utilização da lógica *Fuzzy* para tomada de decisão de qual comportamento cada robô deverá ter em determinado momento e mostrara a técnica dos “campos potenciais” aplicada ao futebol de robôs.

6.2 Campos Potenciais

A decisão da trajetória de cada robô em determinado momento é de fundamental importância em uma partida de futebol de robôs, pois se um robô tomar uma trajetória errônea seus efeitos podem ser desastrosos na questão da competitividade da equipe. Por exemplo, em cada categoria existe um conjunto de regras que ditam faltas e pênaltis e se um jogador exercer uma trajetória que resulte em um destes eventos, o seu time ficará amplamente prejudicado.

A ideia desse método é que os obstáculos exerçam uma força repulsiva e o destino do robô uma força atrativa, como um campo magnético (SIMIM et al., 2012).

6.2.1 Força de repulsão

Uma força de repulsão cria um campo repulsivo que tende a repelir um obstáculo que está imerso no mesmo. Este campo é mais forte perto do obstáculo e vai ficando mais fraco a medida que se afasta deste obstáculo, assim como mostrado na figura 6.1.

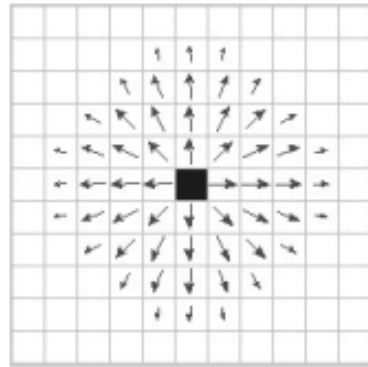


Figura 6.1 – Campo repulsivo.

Fonte: FARIA et al., 2006.

Uma força repulsiva contém duas componentes conforme mostrado na figura 6.2, são elas a componente módulo e a componente direção. Na figura 6.2, “O” é o obstáculo criador do campo repulsivo e “R” é o objeto imerso neste campo.

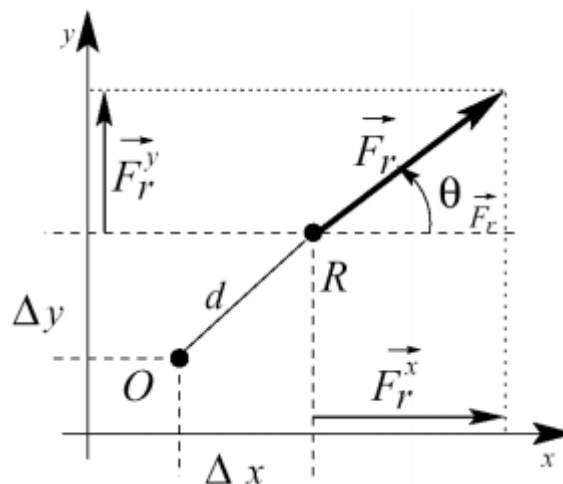


Figura 6.2 – Representação de uma força repulsiva.

Fonte: FARIA et al., 2006.

De acordo com Faria et al. (2006) Quando um objeto está imerso em vários campos repulsivos deve-se fazer o somatória dos vetores das forças de repulsão geradas para se obter uma resultante de forças repulsivas total.

6.2.2 Força de atração

Uma força de atração criada por um objeto pontual gera um campo potencial de atração em seu entorno. Analogamente ao campo potencial repulsivo, porém invertida em sinal, um campo atrativo ou de atração gera forças mais fortes quando se está perto do objeto que o criou e forças menos fortes a medida que se afasta de tal objeto, como mostrado na figura 6.3.

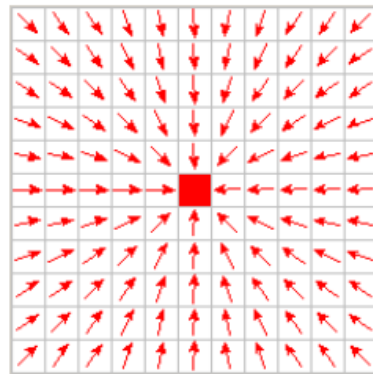
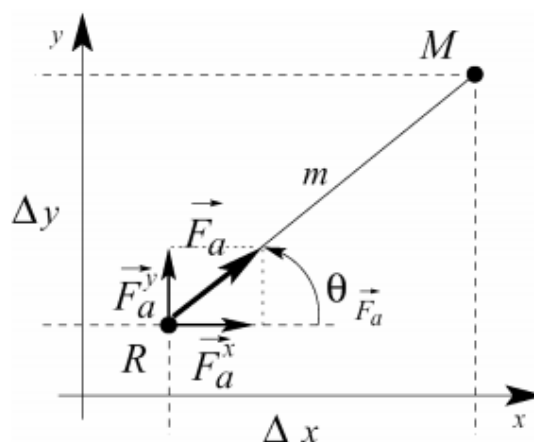


Figura 6.3 – Campo de atração.

Fonte: FARIA et al., 2006.

Na figura 6.4 tem-se a representação de uma força repulsiva onde “M” é o ponto de atração e



“R” é o objeto a ser atraído.

Figura 6.4 – Representação de uma força atrativa.

Fonte: FARIA et al., 2006.

No futebol de robôs ou em qualquer aplicação de campos potenciais para o planejamento de uma trajetória o que exerce a função de um campo atrativo é o destino em que se deseja levar o robô (SIMIM, et al., 2012)

6.2.3 Força resultante

Após determinadas todas as forças de repulsão e atração deve-se então definir a trajetória que cada robô deverá seguir (SIMIM, et al., 2012).

Uma novo ponto resultante deve ser calculado a cada momento, pois os objetos criadores de campos repulsivos e atrativos são dinâmicos e mudam de lugar a todo instante. Com isso, ao longo do tempo cria-se uma trajetória momento conforme é exemplificado na figura 6.5

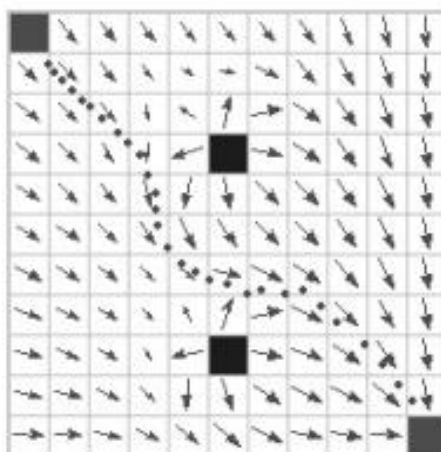


Figura 6.5 – Exemplificação de uma trajetória.

Fonte: FARIA et al., 2006)

Em futebol de robôs são usadas constantes na formulas dos cálculos das forças de atração e repulsão. O cálculo destas constantes apresenta a maior dificuldade na implementação da técnica de campos potenciais aplicada ao futebol de robôs e comumente são calculadas empiricamente apesar de existirem métodos de otimização que podem ser usados para este cálculo.

6.3 Lógica *Fuzzy*

A lógica *Fuzzy* ou nebulosa é a lógica baseada em um conjunto *Fuzzy*. Segundo Gudwin et al. (1994) a lógica *Fuzzy* é a lógica que suporta os modos de raciocínio que são aproximados ao invés de exatos. Na teoria clássica um elemento pertence a um conjunto ou não. Esta lógica tradicional não permite que se diga que um elemento é mais ou menos membro de determinado conjunto. Atualmente diversas aplicações de engenharia utilizam a lógica *Fuzzy* como forma de resolver problemas.

Nos sistemas lógicos clássicos, o modificador mais utilizado é negação enquanto que na lógica *Fuzzy* uma variedade de modificadores de predicados são possíveis (e.g.: *muito*, *mais ou menos*, ...). Estes modificadores são essenciais na geração de termos linguísticos (e.g.: *muito alto*, *mais ou menos perto*, etc.) (GUDWIN, et al., 1994).

Aplicada ao futebol de robôs a lógica *Fuzzy* pode ser utilizada na escolha da tática de um time em determinado momento, por exemplo. Pode se definir algumas táticas diferentes (e.g.: *ataque*, *defesa*, *contra-ataque*) para que o sistema *Fuzzy* determine uma destas táticas a ser utilizada em cada instante.

7 DESENVOLVIMENTO

7.1 Introdução

Para se construir um time de futebol de robôs, inicialmente é necessária a construção de um único robô e testar seu comportamento para assim aprimorá-lo. Depois que este robô estiver atendendo todas as necessidades de um jogador de futebol de robôs, basta fazer outros robôs baseados no primeiro para que sejam os outros jogadores.

7.2 Desenvolvimento Arduino

Cada robô contém dentro de sua carcaça um microcontrolador Arduino. Isto é necessário por que cada robô deve receber informações e processá-las de modo que estas informações possam ser usadas devidamente.

Como cada robô tem seu tamanho limitado (em um cubo de 7,5 centímetros), optou-se por deixar em seu processamento interno estritamente o que for necessário para evitar que ele esquente podendo causar danos aos seus componentes, além de se poder usar um microprocessador menor e com conseqüente menor capacidade de processamento.

7.2.1 Protocolo de identificação do robô

Como são três robôs em cada time, cada robô deve receber apenas os seus comandos, e processá-los internamente. Para que isso aconteça, foi criado um protocolo de comunicação próprio que auxilia cada robô a entender somente o comando a ele destinado.

Este protocolo contém sete caracteres em cada mensagem. Para que cada robô entendesse apenas o seu comando foi usado um caractere inicial e um final específico de cada robô conforme a tabela 7.1.

Tabela 7.1 – Protocolo identificação de cada robô.

{-----}	Robô 1
[-----]	Robô 2
(-----)	Robô 3

Conforme a tabela 7.1, cada robô tem um caractere inicial e final definido para si. Por exemplo, o robô 1 só entenderá uma mensagem que comece com o caractere “{”, descartando todas as outras mensagens que capturar. Este mesmo robô entende como o fim da mensagem quando se depara com o caractere “}”.

Em cada mensagem tem-se ainda outros cinco caracteres, cada qual com sua função como mostra o exemplo na tabela 7.2.

Tabela 7.2 – Exemplo de uma mensagem.

{	A	3	0	8	0	}
1º Caractere	2º Caractere	3º Caractere	4º Caractere	5º Caractere	6º Caractere	7º Caractere

O segundo caractere (mostrado na tabela 7.2 em verde) define se o robô vai para frente, para trás, para direita ou para esquerda ou fica parado. Este caractere pode assumir apenas cinco valores diferentes, são eles “A”, “B”, “C”, “D” e “E” conforme a tabela 7.3.

Tabela 7.3 – Significado do segundo caractere de cada mensagem.

A	Robô vai para frente
B	Robô vai para trás
C	Robô fica parado
D	Gira em torno do próprio eixo em um sentido
E	Gira em torno do próprio eixo em sentido oposto.

Os outros quatro caracteres definem as velocidades que serão enviadas a cada motor. Sendo que os caracteres dois e três (mostrado na tabela 7.2 em amarelo) definem a velocidade do motor A e os caracteres quatro e cinco (mostrado na tabela 7.2 em azul) definem a velocidade do motor B.

Estes quatro caracteres podem assumir valores entre “00” e “99” dois a dois. Sendo que quando é enviado o valor “00” para determinado motor, significa que ele receberá em sua entrada 0 Volts, e ficará parado. E quando receber o valor máximo “99” ele receberá em sua entrada 5 Volts fazendo com que ele gire em velocidade perto da sua máxima (os motores usados são de 6 Volts nominal). Esta conversão de dois caracteres para a tensão enviada aos motores através do seu PWM é mostrada na tabela 7.4.

Tabela 7.4 – Relação entre velocidade da mensagem e velocidade enviada ao motor.

Caracteres de velocidade (dois a dois)	Correspondente no PWM
00	0
99	255

Através da tabela 7.4 tem-se uma equação determinada por uma regra de três simples que devolve o correspondente no PWM aos caracteres determinantes da velocidade. A equação 7.1 mostra essa correlação, sendo “C” os caracteres de velocidade dois a dois.

$$PWM = C \times \left(\frac{255}{99}\right) \quad (7.1)$$

Os comandos enviados pelo segundo caractere sobrepõem os comandos de velocidades dos quatro caracteres que determinam a velocidade dos motores. Sendo assim, quando é enviada uma mensagem para girar em um sentido com velocidades “30” e “50” por exemplo, apenas uma destas velocidades será lida.

7.2.2 Shields Arduino

Para conectar eletricamente os contatos de saída e entrada do microcontrolador Arduino aos motores e ao módulo de comunicação XBee utiliza-se um *Shield* de potência e um *Shield* de condicionamento de sinais.

O *Shield* de condicionamento de sinais para o XBee, converte sinais seriais em 5 Volts para sinais em 3,3 Volts, que é o padrão que o módulo XBee utiliza. Após essa conversão e a conexão elétrica, todos os valores recebidos na porta serial (sem fio) pelo XBee são encaminhados para o Arduino.

7.2.3 Envio aos motores

Após ser feita a identificação e quebra de cada mensagem recebida, o Arduino irá mandar para suas saídas a direção e velocidade que cada motor deverá seguir a cada momento. Cada

Arduino está imediatamente ligada a uma placa Ardumoto que contém *drivers* para os motores. Estes *drivers* enviam as saídas dos Arduinos para os motores com a potência desejada pois recebem uma fonte de alimentação externa ao Arduino.

O Shield que contém *o driver* de potência recebe de cada micro controlador os dados de direção e velocidade (em PWM) de cada motor. Assim, através de uma ponte H contida no circuito integrado L298, esta potência definida pelo micro controlador será enviada diretamente ao motor.

7.2.4 Código no Arduino

Na tabela 7.5, é mostrado um pseudocódigo comentado correspondente ao que está implementado no Arduino:

Tabela 7.5 – Pseudocódigo do Arduino.

<p>Configura Arduino // Configura quais portas do Arduino serão entradas e quais serão saídas.</p> <p>Loop //A partir daqui, todo o resto do código se repete até o robô ser desligado.</p> <p>Lê porta Serial //Função que lê a entrada serial e a guarda em uma variável.</p> <p>Compara Serial //Função que identifica se aquele mensagem é para este robô ou não, caso seja deste robô, ele continua lendo a mensagem, senão ele espera a próxima mensagem.</p> <p>Se mensagem é deste robô:</p> <p>Quebra Mensagem //Função que quebra a mensagem em partes, determinando qual velocidade deve ser mandada para qual motor e em qual sentido isto ocorrerá.</p> <p>Envia para Saída //Função que envia os valores correspondentes para as saídas do Arduino, fazendo em última análise, que os motores girem com a velocidade e sentido desejados.</p>
--

7.3 Motores

Os motores utilizados em cada robô, são motores de corrente contínua. Esta escolha foi feita por que motores de corrente contínua são adequados para o uso com circuitos eletrônicos alimentados por baterias.

Cada motor deve ser alimentado com uma tensão máxima de 6 Volts. Porém, eles serão alimentados com tensões variáveis (controladas por modulação por largura de pulso) dependentes do comando enviado pelo microcontrolador. A figura 7.1 mostra o motor utilizado acoplado a sua caixa de redução.

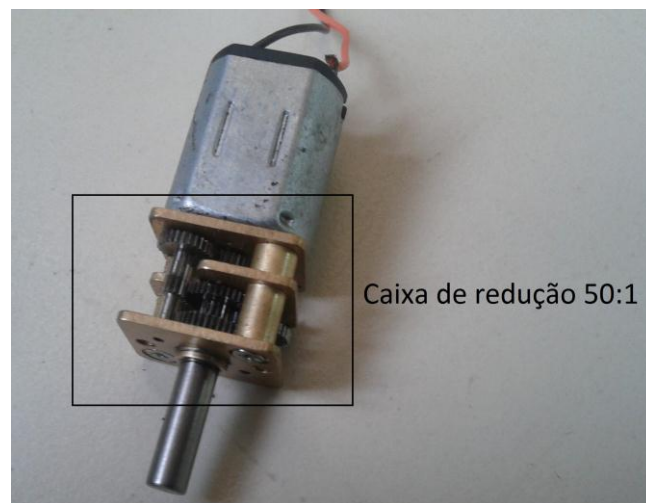


Figura 7.1 – Motor e caixa de redução.

O *Shield* de potência (Ardumoto) está ligado em dois motores de corrente contínua com uma caixa de redução acoplada 50:1 como mostrado na figura 7.2. Esta caixa de redução permite que o motor tenha um maior torque em detrimento de uma menor velocidade de rotação. Com isso os motores são capazes fazer os robôs se deslocarem.

7.4 Baterias

As baterias são parte fundamental de um robô autônomo, por que sua autonomia só dura enquanto durar a energia das baterias.

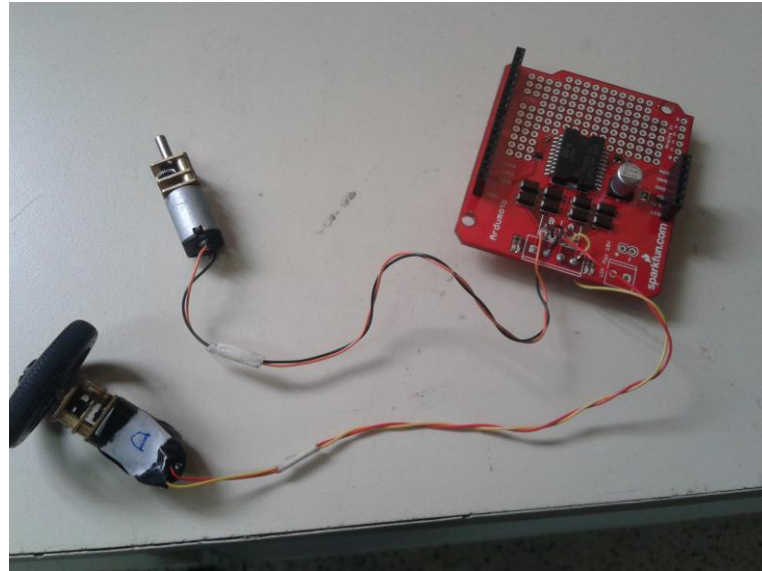


Figura 7.2 – *Shield* arдумoto ligado aos motores.

Para robôs com limitação de peso, deve ser encontrada uma bateria com relação peso e potência adequada. Uma bateria que, para esta necessidade foi adequada, é uma da marca importada *Pololu*, que apresenta tensão de 8,4V. Esta bateria é feita de NiMH, contém 7 células de 1,2V ligadas em série e é capaz de fornecer 900mAh.

Como o uso estimado de corrente de cada robô ficou em torno de 500mA e a tensão de alimentação do circuito é de 5V para o circuito de controle e de no máximo 6V para os motores, então esta bateria se mostrou muito boa para sua aplicação, sendo capaz de suportar um jogo inteiro de futebol de robôs com pouca variação na sua tensão. A figura 7.3 mostra a bateria utilizada.



Figura 7.3 – Bateria de NiMH.

7.5 Estrutura

A estrutura dos robôs em uma partida de futebol de robôs deve ser capaz de suportar o próprio peso dos robôs além de suportar a força exercida por robôs do time adversário em prováveis colisões.

Em se tratando de robôs com peso máximo de 650 gramas, a estrutura não precisa ser muito resistente por que não irá receber muita força aplicada. Para que os robôs não excedam o limite de peso estabelecido, a estrutura deve ser leve.

Como estes robôs foram construídos manualmente a estrutura deve ser de fácil manuseio para que não sejam necessárias máquinas de usinagem em sua construção.

O material escolhido para estrutura foi o fenolite, que é mais conhecido na eletrônica por ser o material mais usado na confecção de circuitos eletrônicos. A figura 7.4 mostra um robô com sua estrutura em fenolite não acabada.



Figura 7.4 – Estrutura do robô não finalizada.

A figura 7.5 mostra a estrutura do robô finalizada e pintada de preto, o que é uma regra da competição, para que a cor do robô não influencie no processamento de imagem das equipes.

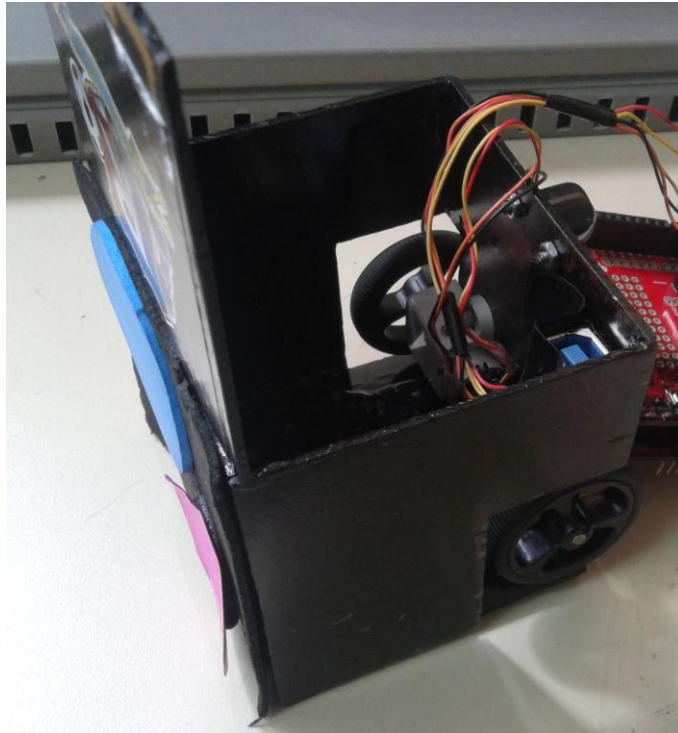


Figura 7.5 – Estrutura do robô.

7.6 Xbee

Cada robô contém seu módulo de comunicação XBee, além de um módulo conectado ao computador PC em que é executada a lógica das estratégias dos robôs. A figura 7.6 apresenta o módulo XBee com uma antena maior do que a original adaptada, de modo que a comunicação tenha um alcance um pouco maior.

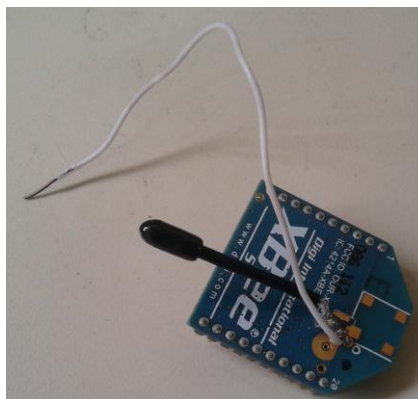


Figura 7.6 – Módulo XBee com antena adaptada.

Estes módulos formam uma rede que está ligada em uma topologia em estrela, onde apenas um XBee é coordenador e todos os outros são dispositivos finais. Isto é feito para que os XBee dispositivos finais, que ficam dentro de cada robô, não precisem enviar informações, apenas recebê-las e o dispositivo coordenador, que fica conectado ao computador, apenas envie informações. Com isso tem-se a rede completa mostrada na figura 7.7.

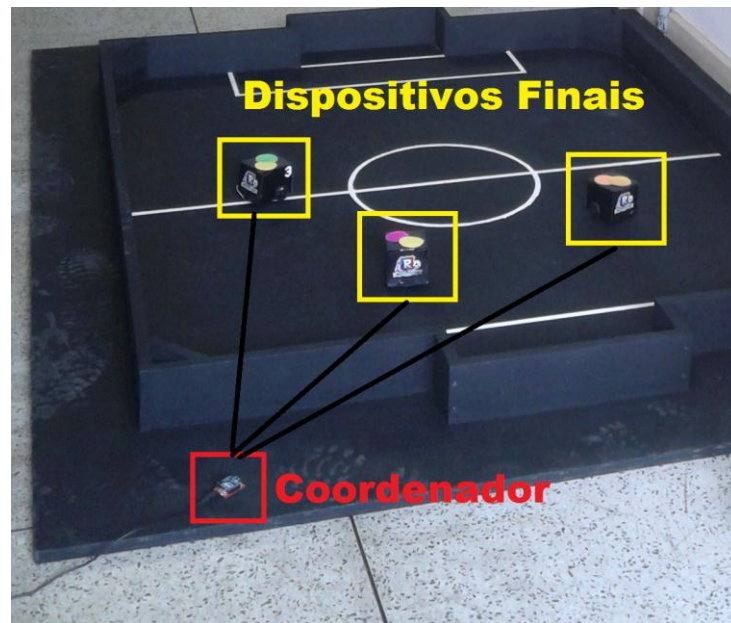


Figura 7.7 – Esquema completo da rede

7.7 Desenvolvimento no computador

Além do processamento realizado no Arduino, uma parte essencial do projeto é o desenvolvimento realizado no computador. Nele são realizadas principalmente duas funções:

- Reconhecimento de imagem, onde o computador recebe as imagens da câmera e reconhece seus padrões a fim de identificar o campo, os robôs e a bola;
- Desenvolvimento da estratégia, onde o computador decide, baseado em uma estratégia desenvolvida no código, o que cada robô deve fazer a seguir.

Toda a programação realizada no computador PC foi realizada na plataforma Adobe® Flash® na linguagem Action Script 3.0. Isto porque esta plataforma é bastante apropriada para aplicações com reconhecimento de imagem por já conter várias bibliotecas próprias neste sentido. A plataforma Adobe® Flash® também permite a criação de interfaces amigáveis como a mostrada a interface criada pelo time RODETAS na figura 7.8.

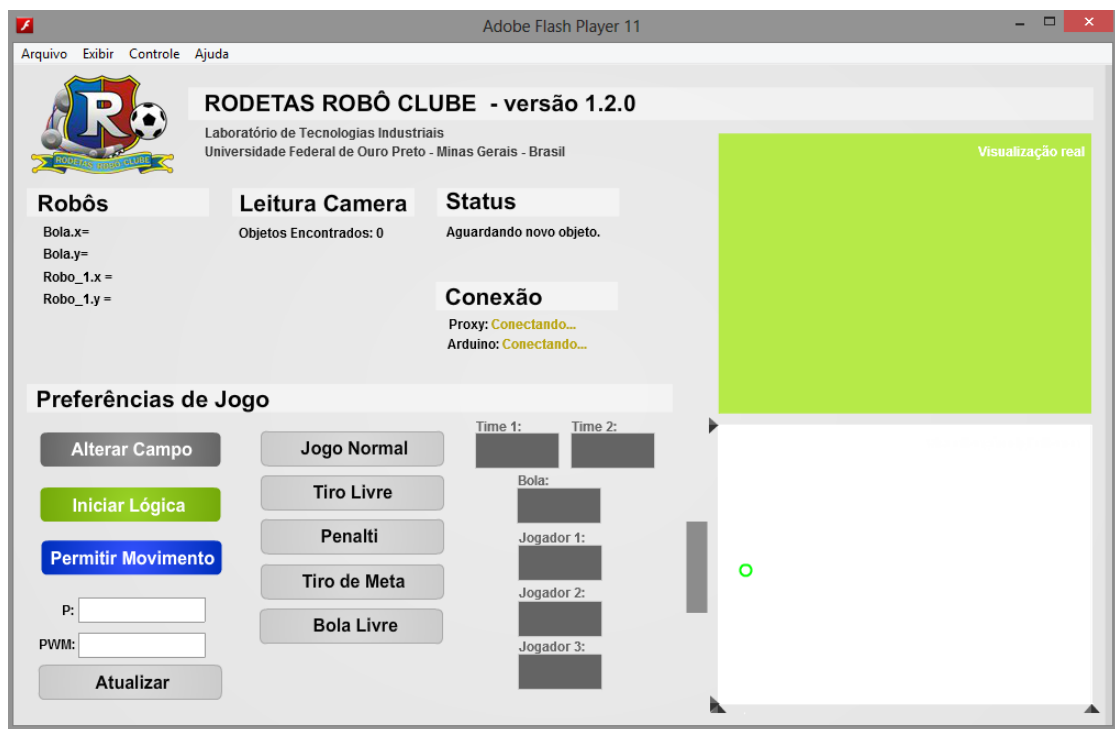


Figura 7.8 – Interface criada no Adobe® Flash®.

Nesta interface é possível obter informação sobre o que está acontecendo no momento. São mostradas as posições dos robôs, o *status* da conexão com os robôs, a imagem real obtida pela câmera e a imagem digitalizada. Esta interface é uma excelente ferramenta no diagnóstico de erros. Nela também é possível realizar a calibração das cores dos robôs antes do início da partida.

7.7.1 Reconhecimento da imagem

Na categoria *IEEE Very Small Size* de futebol de robôs, cada equipe é responsável pelo desenvolvimento de seu próprio sistema de reconhecimento de imagem. Isto apresenta uma das partes mais complicadas de todo o projeto. Isto ocorre por que é difícil prever as condições exatas de luz do ambiente da competição, apesar das regras estabelecerem estas condições. Qualquer variação no ambiente pode ser sentida pela câmera e atrapalhar o reconhecimento de imagem. A fim de minimizar estes problemas, é feita uma programação neste sentido.

A câmera utilizada é uma *Webcam* de alta resolução (*Full HD*) da marca Microsoft como a mostrada na figura 7.9. A captura dos vídeos é feita em formato 640x480, para que haja maior velocidade no processamento desta imagem, não são capturadas imagens em alta resolução apesar desta câmera assim permitir. A câmera deve ser fixada há dois metros de altura do campo.



Figura 7.9 – Câmera utilizada.

Primeiramente, é capturada a imagem dinâmica (vídeo) da câmera utilizada e quebrada em imagens estáticas para facilitar o processamento. Cada segundo de vídeo é transformado em dez imagens. Estas imagens então são analisadas. A relação de quantas imagens por segundo seriam necessárias foi determinada de forma empírica, se houvesse mais imagens analisadas, o processamento ficaria lento, e se houvesse menos imagens, não daria tempo hábil do robô reagir às interações do ambiente.

Cada imagem é sobreposta a uma imagem do campo e é feita uma diferença entre estas imagens, de modo que seja eliminada da primeira imagem, o campo. Desta imagem obtida irão sobrar apenas os robôs dos dois times e a bola.

Assim, para determinar os robôs da equipe RODETAS, é feita uma varredura (*pixel por pixel*) na imagem que verifica onde estão os maiores agrupamentos de *pixels* da cor correspondente ao time (azul ou amarela). Após determinado onde estão estes maiores agrupamentos (no máximo três) tem-se onde estão posicionados cada robô, porém ainda não há uma identificação de cada robô individual nem pra qual ângulo ele está virado.

Para que haja uma identificação individual o programa procura por agrupamentos de cores pré-estabelecidas que estão perto das cores da equipe. As cores usadas normalmente são o rosa, o roxo e o verde por que foram as cores com maior sucesso na identificação determinadas empiricamente.

A bola é o objeto no campo de mais fácil identificação por que a bola é o único objeto laranja, para identificá-la basta procurar pelo maior agrupamento de *pixels* laranja na imagem.

Os robôs da outra equipe são identificados apenas pelo *patch* do time (azul ou amarelo), por que não é de interesse da estratégia do time, saber pra qual ângulo o robô adversário está virado e nem é de interesse a identificação de cada robô adversário individualmente.

7.7.2 Estratégia do time

A estratégia do time é responsável por decidir qual trajetória cada robô deve seguir em cada momento e traduzir isto para qual tensão cada motor deve receber neste determinado momento.

A estratégia do time define que sempre irá existir um robô atacante, um robô de apoio e um goleiro. O goleiro será fixo, ou seja, a um robô será atribuído a função de goleiro no início do jogo e nunca irá mudar esta função. Os outros dois robôs terão suas funções variadas dependendo de sua posição em campo e do seu ângulo em relação a bola. Para que a decisão de qual jogador será o atacante e qual será o apoiador em cada momento, é utilizado um controlador baseado em lógica *Fuzzy*.

7.7.2.1 Controlador *Fuzzy*

As entradas deste controlador *Fuzzy*, são mostradas na tabela 7.6. Estas entradas consideram todos os fatores que afetam a decisão de qual robô é atacante e qual é apoiador.

Tabela 7.6 – Entradas do controlador *Fuzzy*.

Distância do robô 1 a bola -> R1 (perto - médio - longe)
Distância do robô 2 a bola -> R2 (perto - médio - longe)
Ângulo do robô 1 em relação a bola -> A1 (bom – médio - ruim)
Ângulo do robô 2 em relação a bola -> A2 (bom – médio - ruim)

Após serem colocadas as entradas, as regras do controlador são definidas, de modo com que cada regra decida qual robô será o atacante como mostrado no exemplo da tabela 7.7. Nesta estratégia é definida que a distância de cada robô a bola é mais decisiva do que seu ângulo em relação a bola.

Tabela 7.7 – Exemplo das regras do controlador *Fuzzy*.

<i>SE</i>	<i>Condição</i>	<i>ENTÃO</i>	<i>Decisão</i>
SE	R1 perto e R2 perto e A1 bom e A2 médio	ENTÃO	Robô 1 ataca
SE	R1 longe e R2 perto e A1 bom e A2 ruim	ENTÃO	Robô 2 ataca

Depois disso é feita uma inferência *Fuzzy* que, depois de atribuídas pertinências às entradas, infere uma saída utilizando uma determinada função. Esta saída é em termos de qual robô será atacante. Tudo isto é feito na mesma taxa que cada imagem é processada, ou seja, esta decisão é tomada 10 vezes por segundo.

7.7.2.2 Campos potenciais

Depois de definido quem é o atacante e que é o apoiador em determinado momento, deve se ter um vetor resultante determinando ângulo e força (velocidade) nesse momento. A estratégia utilizada para decidir isto é baseada em campos potenciais.

Isto é feito através do código do computador utilizando o artifício da física para tal. Cada robô repele ou atrai objetos no campo, artificialmente. Um robô atacante é atraído pela bola a fim de que este tente chegar na bola para empurrá-la para o gol, da mesma forma o robô apoiador é atraído pela bola, porém é repelido pelo robô atacante, de tal forma que os dois não tentem ir para o mesmo lugar em um mesmo instante.

Os robôs atacante e apoiador, são ambos repelidos pelos robôs adversários, de modo que eles desviem suas trajetórias para não chocar com um robô adversário como exemplifica a figura 7.10. Na figura 7.10 os robôs adversários são mostrados de vermelho, a bola mostrada em laranja e a trajetória do robô atacante mostrada em pontilhado. A trajetória é tal que o robô atacante desvia dos adversários e encontra um caminho livre para achar a bola.



Figura 7.10 – Trajetória do atacante.

Estas repulsões e atrações geram um vetor resultante para cada robô. Este vetor resultante tem informações quanto à velocidade e ao ângulo que estes robôs devem seguir. Estas informações são traduzidas pelo código em termos de sinal elétrico. O sinal elétrico então é enviado aos robôs, que através do seu *driver* de potência, acionam os motores e finalmente, as rodas.

7.7.3 Comunicação com a porta serial

Para ser possível a comunicação com os robôs, é necessário um módulo XBee conectado a uma porta USB de um computador PC. Esta ligação é feita através de um adaptador chamado XBee USB Explorer que simplesmente transmite o que chega na porta serial da USB para o módulo XBee.

A plataforma Adobe® Flash® possui bibliotecas para comunicação com uma porta USB através do *software* chamado serProxy. Para configurar este *software* serProxy, deve-se procurar em qual porta o USB Explorer está conectado e assim esta informação deve ser colocada no arquivo de configurações do serProxy.

Depois de configurado, toda vez que for encontrado na programação do Adobe® Flash® um comando para jogar na porta serial do USB do computador, o serProxy fará este interfaceamento de modo que, por fim, os dados enviados pelo Adobe® Flash® sejam recebidos e retransmitidos pelo XBee.

8 RESULTADOS

O objetivo do trabalho foi atingido com sucesso, a figura 8.1 mostra três robôs capazes de participar de um jogo como proposto.



Figura 8.1 – Time completo.

O principal objetivo do trabalho foi o desenvolvimento de um time de futebol de robôs e a principal motivação era a possibilidade de participar de uma competição. No começo do projeto esta possibilidade estava distante e só depois que ela foi ficando mais real. Quando a participação na competição *Latin American Robotics Competition - LARC 2012* se tornou possível por razões técnicas, o principal objetivo do projeto passou a ser conquistar um bom resultado nesta competição no primeiro ano da equipe RODETAS.

O torneio LARC 2012 teve inscritos doze equipes, porém somente sete apareceram em condições de jogar. Com estas sete equipes foram decididas as regras da competição. As regras estabeleciam que primeiramente iria se eliminar uma das sete equipes em disputas de pênalti para que pudesse se formar as chaves com seis equipes. Nesta disputa de pênaltis a equipe ficaria sem nenhum adversário no campo por três minutos para marcar o maior número de gols possíveis.

A equipe RODETAS ficou em segundo lugar nesta eliminatória, ganhando assim vantagens na próxima fase.

A fase de grupos consistiu de várias chaves eliminatórias, ou seja, quem perdesse uma partida estaria fora da competição.

No primeiro jogo a equipe RODETAS empatou em 1x1 e só prosseguiu por ter conseguido uma boa colocação na fase dos pênaltis.

No segundo jogo a equipe RODETAS ganhou de 3x1 da equipe UNBall da UnB e prosseguiu para a próxima fase da competição.

A próxima partida foi contra a equipe Ararabots da UFMS e o resultado foi uma vitória da equipe RODETAS por 6x2.

A partida da grande final foi realizada contra a USP de São Carlos, equipe Warthogs. Nela, a equipe RODETAS obteve sua única derrota, por 3x0 e ficou assim com o segundo lugar na competição de futebol de robôs categoria *IEEE very small size* na *Latin American Robotics Competition 2012* realizada em Fortaleza - CE.



Figura 8.2 – Troféu obtido na competição.

9 CONSIDERAÇÕES FINAIS

Analisando os resultados fica evidenciado que o projeto foi bem realizado e atingiu todos os objetivos propostos e ainda foi um pouco além e se tornou, em seu primeiro ano, a segunda melhor equipe da América Latina.

Com a realização deste trabalho, foi possível perceber que uma competição de robótica é de grande valia no ensino de ferramentas que são úteis em muitas aplicações de engenharia. Conclui-se também que um projeto como este necessita de uma equipe com uma sinergia afinada a fim de que obtenha bons resultados.

Conclui-se que em uma competição o objetivo deve ser sempre superar os adversários e como eles estão em constante evolução, sempre haverá espaço para melhorias e otimizações.

Futuramente a equipe RODETAS irá desenvolver um novo time, feito de maneira mais profissional para que se tenham robôs exatamente iguais, além disso um próximo passo é o uso de *encoders* nas rodas. Estas abordagens são para que se possa tirar um modelo matemático preciso dos robôs para análises computacionais no campo da teoria de controle.

Como continuação deste trabalho sugere-se a modelagem matemática do comportamento de um robô para que possa-se utilizar artifícios matemáticos para sintonia de todos seus parâmetros.

REFERÊNCIAS

ALDEBARAN ROBOTICS: Aldebaran Robotics Web Site. Disponível em: <<http://www.aldebaran-robotics.com>>. Acesso em: 08 Jan. 2013.

ARDUINO: Arduino HomePage. Disponível em: <<http://www.arduino.cc>>. Acesso em: 26 Out. 2012.

BEKEY George A. Autonomous Robots: From Biological Inspiration to Implementation and Control: 2005.

BORGES, Marcos Augusto F.; BORTHOLOTTO, Julio C. Integração de dispositivos robóticos a sistemas de apoio ao aprendizado utilizando plataforma Arduino. Universidade Estadual de Campinas, Limeira - SP - Brasil: 2010?

BRÄUNL, Thomas. Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems: 1999.

CBROBOTICA: Latin American Robotics Competition. Disponível em: <<http://www.cbrobotica.org/>>. Acesso em: 8 Jan. 2013.

DIMITROV, Smilen; SERAFIN, Stefania. Audio Arduino - an ALSA (Advances Linus Sounds Architecture) audio driver for FTDI-based Arduinos. In: Proceeding of the international Conference on New interfaces for Musical Expression, Oslo - Noruega: 2011.

DORNAN, A. Wireless communication: o guia essencial de comunicação sem fio, Rio de Janeiro - RJ - Brasil, Campus: 2001.

ESCHNER, Rodrigo Hommerding. Sistema de Automação Residencial Baseado em Sensores ZigBee, Porto Alegre - RS - Brasil: 2011.

FACON, Jacques; JÚNIOR, Alceu De Souza Britto e NETO, Anselmo Chaves. Uso da Análise de Componentes Principais na Segmentação interativa de Imagens Coloridas. In: XXIX CONGRESSO DA SOCIEDADE BRASILEIRA DE PESQUISA OPERACIONAL, 1997, Salvador : XXIX SBPO, 1997.

FARIA, Gedson; **MARTINS**, Priscila Da Silva e **PEREIRA**, Mauro Conti. **Time de Robôs controlado por Campos Potenciais**. In: ANAIS DO XXVI CONGRESSO DA SBC. - Campo Grande, MS : GPEC - Grupo de Pesquisa em Engenharia e Computação - Universidade Católica Dom Bosco: 2006.

FIRA: Federation of International Robot-Soccer Association. FIRA | FIRA Cup History. Disponível em: <<http://www.fira.net/?mid=firahistory>>. Acesso em: 8 Jan. 2013.

FORESTI, Henrique Braga. **Desenvolvimento de um robô bípede**, Recife: 2006.

FOROUZAN, Behrouz A. e **FEGAN**, Sophia Chung. **Protocolo TCP/IP**, McGrawHill: 2008.

MATOS, Leonardo; **MENDONÇA**, Gabriel Matos Meryelle; **FREIRE**, Eduardo e **MONTALVÃO**, Jugurta. **Sistema de visão artificial baseado em detecção de cores (para sistema de controle de robôs celulares com realimentação visual)**. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, 2007, Florianópolis - SC: 2007.

GUDWIN, Ricardo Ribeiro e **GOMIDE**, Fernando Antonio Campos. **Modelagem, controle, sistemas e lógica fuzzy**. In: Revista da SBA Controle & Automação. Departamento de Engenharia de Computação e Automação Industrial (DCA) - Faculdade de Engenharia Elétrica (FEE), - 3. - Vol. 4: 1994

IEEE: Regras Categoria IEEE very small size | crobotica. Disponível em: <http://www.crobotica.org/regras/VerySmall2008_en.pdf>. Acesso em: 8 Jan. 2013.

KITANO, Hiroaki. **RoboCup-97: Robot Soccer World Cup I**, Nagoya: 1997.

MAXTREAM: Datasheet XBee. Disponível em: <https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Manual.pdf>. Acesso em: 22 Fev. 2013.

MONSIGNORE, Ferdinando. **Sensoriamento de ambiente utilizando o padrão ZigBee**, São Carlos: 2007.

MONTEBELLER, Sidney José. **Estudo sobre o emprego de dispositivos sem fios - wireless na automação do ar condicionado e de outros sistemas prediais**, São Paulo: 2006.

MURPHY, R. R. Introduction to AI Robotics, MIT Press: 2000.

NOERGAARD, T. Embedded Systems Architecture: A comprehensive Guide for Engineers and Programers, Estados Unidos: 2005.

RIBEIRO, A. Fernando. Virtual sensors for autonomous mobile robots through the use of image processing tools. In: REVISTA ROBÓTICA: Abr. 2005.

SILVA, André Teixeira Da. Módulos de Comunicação Wireless para Sensores, Porto: 2007.

SIMIM, João Paulo Gonçalves e PALMIERI, Karla Boaventura Pimenta. Descrição de uma estratégia para competição de futebol de robôs. In: MOSTRA BRASILEIRA DE ROBÓTICA 2012, Fortaleza. Fortaleza - CE - Brasil: 2012.

STONE, Peter. Intelligent Autonomous Robotics: A Robot Soccer Case Study: 2007.

WILMSHURST, Tim. Designing Embedded Systems With Pic Microcontrollers: Principles and Aplications: 2007.