CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Alexsander Ramos da Silva

PADRÃO DE ARQUITETURA PARA FUTEBOL DE ROBÔS

Timóteo

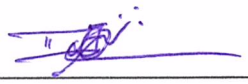
2018

Alexsander Ramos da Silva

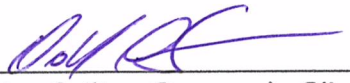
PADRÃO DE ARQUITETURA PARA FUTEBOL DE ROBÔS

Monografia apresentada ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais para a obtenção do título de Engenheiro de Computação.

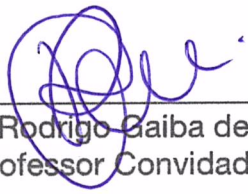
Trabalho aprovado. Timóteo 06 de dezembro de 2018:



Prof. Dr. Elder de Oliveira Rodrigues
Orientador



Prof. Me. Odilon Correa da Silva
Professor Convidado



Prof. Dr. Rodrigo Gaiba de Oliveira
Professor Convidado

Timóteo
2018

Alexsander Ramos da Silva

PADRÃO DE ARQUITETURA PARA FUTEBOL DE ROBÔS

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Dr. Elder de Oliveira Rodrigues

Timóteo

2018

Aos meus pais, José Amauri e Elizângela.

Agradecimentos

Agradeço aos meus pais, irmã, familiares e amigos pelo incentivo e carinho.

Agradeço a meu orientador por me auxiliar neste trabalho e em outras atividades acadêmicas durante a graduação.

”É melhor acender uma vela do que praguejar contra a escuridão”.
Adágio

Resumo

O futebol de robôs é um esporte multidisciplinar, que envolve essencialmente as áreas de visão computacional, estratégia, comunicação sem fio, eletrônica e mecânica. Estas áreas devem trabalhar em conjunto para o funcionamento coordenado de um time. A maior parte do processo do futebol de robôs dá-se no computador, o que remete ao desenvolvimento de software. Nesta parte do desenvolvimento é ausente um padrão que comprovadamente permita a criação de um software para o esporte. Para isso, criou-se um padrão arquitetural genérico, que permite a simples escalabilidade e manutenibilidade do software que o implementa e executou-se testes sobre uma implementação simplista que o seguiu, provando a eficácia do padrão para os casos especificados do futebol de robôs categoria IEEE *Very Small Size Soccer*. Além da arquitetura, o trabalho reúne uma parte da fundamentação teórica das disciplinas do esporte, fazendo dele uma fonte unificada destas informações. O uso do padrão arquitetural permite que o esforço de desenvolvimento se concentre na visão computacional, estratégia e comunicação sem fio, as partes que realmente agem no jogo, deixando que o fluxo das informações e execução sejam geridos pela arquitetura, sem perder a flexibilidade, que é fornecida pelo uso da tipagem genérica e a simples customização na substituição de componentes, provida pelo uso de interfaces para os atuadores.

Palavras-chave: Visão Computacional, Padrão de Arquitetura, Futebol de Robôs, IEEE Very Small Size Soccer.

Abstract

Robot soccer is a multidisciplinary sport that essentially involves the areas of computer vision, strategy, wireless communication, electronics and mechanics. These areas should work together for the coordinated operation of a team. Most of the robot soccer process takes place in the computer, which refers to software development. In this part of the development is absent a pattern that demonstrably allows the creation of a software for the sport. So, a generic architectural pattern was created that allows the simple scalability and maintainability of the software that implements it, and tests were executed on a simplistic implementation that followed, proving the effectiveness of the pattern for the specified cases of category IEEE Very Small Size Soccer. Besides architecture, the work brings together a part of the theoretical foundation of sports disciplines beeing an unified source of this information. The architectural pattern use allows the development effort focus on computer vision, strategy, and wireless communication, the parts that actually act in the game, letting the flow of information and execution be managed by the architecture itself, without losing the flexibility, which is provided by the use of the generic typing, and the simple customization in the substitution of components, provided by the use of interfaces in the actuators.

Keywords: Computer Vision, Architectural Pattern, Robots soccer, IEEE Very Small Size Soccer.

Lista de ilustrações

Lista de tabelas

Lista de Quadros

Lista de Algoritmos

Sumário

1 Introdução

“Nada é maravilhoso demais para ser verdade”.
Comentário atribuído a Michael Faraday (1791-1867)

Futebol de robôs é uma iniciativa internacional voltada à pesquisa e à educação, visando promover desenvolvimentos ligados às áreas de inteligência artificial e robótica inteligente (KITANO et al., 1997).

> Construir um time *RoboCup Small Size League* (SSL) capaz de competir no campeonato mundial requer uma pesquisa multidisciplinar em campos como desenvolvimento de hardware robótico, aprendizado de máquina, sistema multi robô, visão computacional, teoria de controle e mecânica, entre outros (GURZONI et al., 2011, tradução nossa, p.69)[1].

O mesmo afirmado por Gurzoni et al. (2011, p.69) aplica-se à modalidade *Very Small Size Soccer*, que apesar de seguir regras próprias IEEE (2008), abrange as mesmas áreas de conhecimento da SSL.

Os diversos trabalhos acadêmicos, realizados sobre esta segunda modalidade, apresentam uma abordagem específica, onde não são explorados todos os campos do esporte eletrônico em conjunto, apenas um ou alguns destes são apresentados. Como exemplo disso, pode-se citar o trabalho de Barros et al. (2015), que trata da visão computacional, a produção de Shim et al. (1999), que detalha certa estratégia de jogo, e a publicação de Jusoh e Omar (2011), que esclarece a parte de *hardware*.

Estes trabalhos de abordagem específica são fontes para quem busca melhorar ou aprender sobre um determinado aspecto inerente de um time, contudo, quem pretende criar um time do início encontra diversas dificuldades, seja para reunir material bibliográfico, quanto para identificar o que é realmente essencial para fazer um time funcionar.

1.1 Justificativa

Sabendo da ausência de um compilado de informações, para a produção de um time, e sendo conhecido o potencial contributivo do esporte para o aprendizado, é necessária uma produção que instrua o interlocutor por um caminho, que permita a construção de um time, crie um padrão de arquitetura completo do processo e defina uma abordagem simples que possam ser utilizados como base para o desenvolvimento futuro.

Segundo That, Sadou e Oquendo (2012), um dos principais problemas do desenvolvimento de software está no estágio de manutenção e evolução. Ainda afirmam que, de fato,

[1] To build a RoboCup Small Size League (SSL) team able to compete in the world championship requires multidisciplinary research in fields like ro- botic hardware development, machine learning, multi-robot systems, computer vision, control theory, and mechanics, among others. (GURZONI et al., 2011, p.69).

dados os altos custos associados a esse estágio (cerca de 80% do custo total), torna-se importante encontrar uma solução para reduzi-los e, além disso, afirmam que os principais fatores deste problema são:

1. O não cumprimento das práticas estabelecidas
2. Falta de explicitação das escolhas feitas ao longo do processo de desenvolvimento.

That, Sadou e Oquendo (2012) afirmam que, se a causa for a primeira e precisarmos aplicar uma evolução, devemos primeiro reconstruir o que já foi construído incorretamente. Aplicar uma evolução em um sistema mal construído só pode torná-lo mais complexo e, finalmente, incapaz de evoluir ainda mais (segunda lei de Lehman). No segundo caso, o sistema está bem desenvolvido, exceto que as intenções por trás de cada escolha não são explícitas. Há sempre diferentes soluções para conseguir uma mudança, mas algumas delas podem estar em contradição com certas intenções implícitas. Além disso, pode levar vários passos entre a criação da contradição e a sua detecção. Isso requer desfazer o que já foi construído, resultando em custos adicionais importantes.

Sabendo destas características de um padrão de arquitetura e dos problemas causados por sua não adoção, faz-se necessário criar e aderir a esta prática para o futebol de robôs, pois neste sistema é de interesse que a solução implementada seja customizável a qualquer tempo de forma rápida, descomplicada e barata.

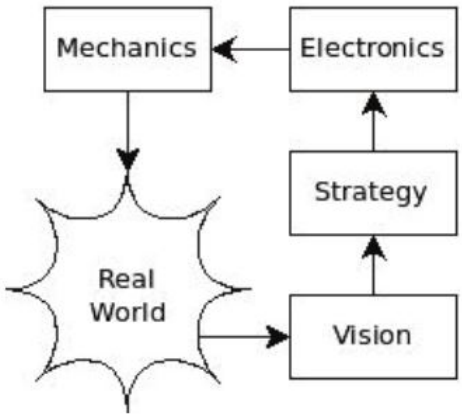
Um sistema orientado a serviços fracamente acoplado é fácil e barato de se desenvolver e tem o potencial de crescer e escalar rapidamente [2]. Tendo estes princípios em vista, busca-se trazer esta orientação para o padrão de arquitetura proposto, dando a este padrão os atributos do baixo acoplamento.

O ciclo de funcionamento do futebol de robôs é definido, genericamente, por Silva et al. (2010) e pode ser observado na Figura 1. Cada caixa presente nesta representação demonstra um estágio do processo e pode conter quaisquer soluções que permitam o time executar suas funções coordenadas, ou seja, é realizado de acordo com a necessidade do criador. Gurzoni et al. (2011), por exemplo, completou as caixas com suas soluções particulares, que podem ser observadas na Figura 2.

Assim como os demais trabalhos encontrados na literatura, a Figura 1 não define como estas partes se comunicam ou o que trafega entre elas. Esta informação é de valia para a compreensão do funcionamento da arquitetura por trás do esporte eletrônico. Sabendo disso, é interessante definir estes detalhes no padrão de arquitetura, elucidando aos leitores sobre os mesmos.

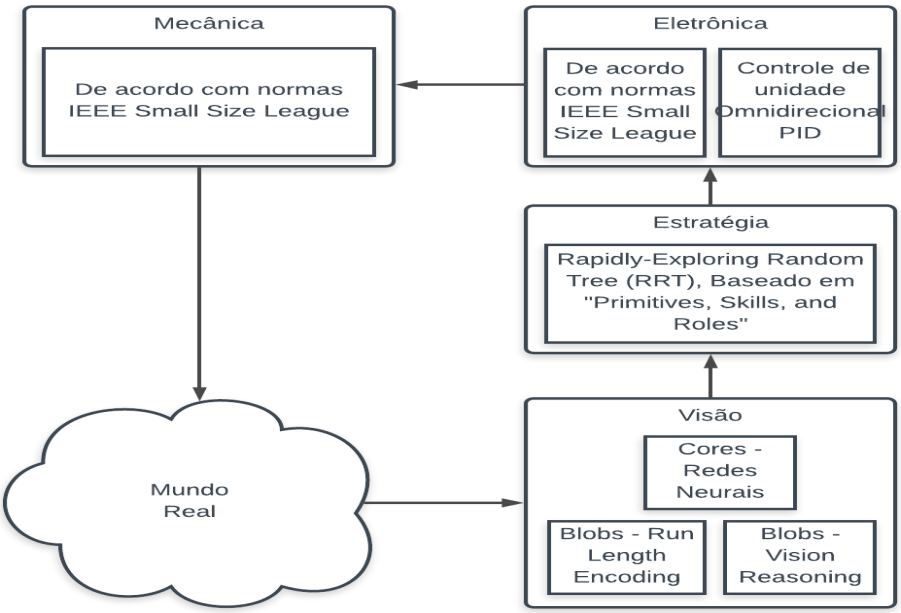
[2] A loosely coupled service-oriented system is thus easy and cheap to evolve and has the potential to grow as rapidly[...] (PAUTASSO; WILDE, 2009, p.911)

Figura 1 – Estrutura de um sistema real de robôs de futebol.



Fonte: Silva et al. (2010).

Figura 2 – Abordagem utilizada por Gurzoni et al. (2011).



Fonte: O Autor.

1.2 Objetivos

Almeja-se desenvolver um padrão de arquitetura para definir uma prática referência de programação do futebol de robôs categoria *Very Small Size* e, sobre este padrão definido, realizar testes, através de sua utilização no desenvolvimento de um algoritmo simplista das partes de visão computacional, estratégia e comunicação sem fio, em conjunto com um robô protótipo.

Tal abordagem é escolhida, pois, a implementação simplista apesar de não explorar o

limite do padrão arquitetural apresentado, cuida que ele tenha sua funcionalidade comprovada e, ainda, permite que seja facilmente reproduzida caso seja o primeiro contato do interlocutor com o assunto.

O padrão de arquitetura proposto também anseia o baixo acoplamento entre as áreas do esporte eletrônico, visando permitir o desenvolvimento concorrente, num cenário de refatoração ou criação de algoritmos que tenham o padrão proposto como base.

Busca-se mais especificamente:

1. Definir Padrão de Arquitetura para o futebol de robôs;
2. Comprovar efetividade do padrão de arquitetura através de testes em algoritmo simplista desenvolvido a partir dele;
3. Definir protocolo de comunicação que controle e possibilite a transferência da informação sobre a velocidade das rodas do computador para os robôs.

1.3 Estrutura do Texto

Este texto está estruturado em seis capítulos. No Capítulo 2 são relatados os trabalhos relacionados, usados como base neste, e é detalhado o referencial teórico. No Capítulo 3 são apresentados os procedimentos metodológicos usados para desenvolvimento do projeto. O que foi desenvolvido, encontra-se no Capítulo 4. O Capítulo 5 apresenta os experimentos realizados e os resultados obtidos. Conclusão e trabalhos futuros são apresentados no Capítulo 6.

2 Revisão Bibliográfica

“Esperamos pela luz, mas contemplamos a escurião”.
Isaías, 59:9

O futebol de robôs tem sido campo para o desenvolvimento em diversas vertentes da computação. Devido a sua grande gama de possibilidades, os trabalhos existentes na literatura apresentam abordagens específicas, propondo soluções para as diversas áreas. Com estas produções, é possível compreender o enorme potencial da esporte eletrônico para o desenvolvimento e consolidação de tecnologias. Neste Capítulo 2, os trabalhos relacionados são descritos e o necessário para a compreensão do tema e desenvolvimento realizado é retratado. Primeiramente, são analisados alguns trabalhos relacionados. Posteriormente, as regras básicas do esporte são abordadas. Finalmente, a teoria utilizada na implementação é relatada.

2.1 Trabalhos Relacionados

Dada a multidisciplinaridade do trabalho, esta seção 2.1 agrupa os trabalhos, relacionando-os pelas áreas, e os que abrangem as diversas áreas.

A princípio é necessário compreender as regras da competição, sendo que para isto existe o manual da IEEE (2008), que apresenta técnica e detalhadamente as obrigatoriedades do time para a participação da competição. Então, após esta visão geral, é possível entrar especificamente nas disciplinas abrangidas pelo esporte eletrônico.

2.1.1 Diversas Áreas

Um notório trabalho é o de Gurzoni et al. (2011), que apresenta um desenvolvimento nas áreas de visão computacional, controle de movimentação dos robôs e estratégia. O autor adota um sistema de rede neurais *Multi Layer Perceptron* (MLP) para a detecção de cores, uma vez que existe variância luminosa dos ambientes, a detecção estática pode deixa a desejar. Como o treinamento de uma rede MLP tem elevado custo computacional, há um pré treinamento e uma *lookup table* é preenchida antes do jogo, com todas as possibilidades de valores do espaço de cores HSV, que são ($360 \times 100 \times 100$). Ainda na visão computacional, Gurzoni et al. (2011) utiliza um algoritmo *Run Length Encoding* (RLE) para a geração de *blobs*, extraindo informação das regiões, tais como contorno, centroide e área, que são necessários para a análise do reconhecimento de padrões e uso por algoritmos heurísticos. Os dados capturados são transformados na informação para a estratégia, através de algoritmos que seguem diversas heurísticas, que procuram padrões nos *blobs* que sejam compatíveis com o formatos predefinidos e que correspondam aos dos elementos do jogo.

Na parte de eletrônica de Gurzoni et al. (2011) por se tratar da categoria SSL, devido ao formato dos robôs o autor realiza o controle omnidirecional nas quatro rodas.

No planejamento de caminho é utilizado *Rapidly-Exploring Random Trees* (RRTs) para definir o caminho a ser tomado pelo robô. A estratégia faz uso deste chamado *path planning* para definir a movimentação para cada robô que possui papel definido.

2.1.2 Visão Computacional

A área de visão computacional no futebol de robôs é desenvolvida em diversos trabalhos, de muitas maneiras diferentes. Estas soluções apresentadas diferem umas das outras e têm maior ou menor precisão, dependendo de diversos fatores.

Barros et al. (2015) apresentaram uma interessante solução, que foge a detecção de cores largamente utilizada pelos autores e utiliza a detecção por formas, num algoritmo de pontos chave, que se prova suficientemente rápido para a utilização no futebol de robôs.

Martins, Tonidandel e Bianchi (2006) apresentaram tanto a detecção de uma única forma geométrica independente de cores, utilizando a Transformada de Hough, quanto uma baseada na detecção objetos de uma cor independente da forma, através de um algoritmo de detecção em cruz. Neste trabalho é identificado que ambas as abordagens, detecção por cor e formas, são viáveis para o futebol de robôs, pois extrapolam as necessidades de tempo, precisão e tolerância à variação de iluminação exigidas pelo esporte.

2.1.3 Estratégia

Guarnizo e Mellado (2016) apresentaram a abordagem selecionada para utilização neste trabalho, máquina de estados finitos. A estratégia é dividida em táticas, que são selecionadas por uma máquina de estados. Uma vez que a tática tenha sido selecionada, é atribuída funções aos robôs, dependendo das condições do jogo. Cada função executa comportamentos definidos, selecionados pela máquina de estados.

2.1.4 Eletrônica e Mecânica

Jusoh e Omar (2011) realizaram um estudo sobre os desenvolvimentos de robôs dos 10 anos anteriores de sua publicação, na área de mecânica e circuito do futebol de robôs, para apresentar o melhor desempenho e desenho presente no esporte eletrônico. Este artigo analisa o desenvolvimento de robôs futebol, baseado em subcategorias de robô. A pesquisa destes autores ressalta a atenção prioritária à seleção de um projeto robusto: restringir o tamanho do robô, o peso, projeto e equilíbrio do robô, para obter assim um sistema de alto desempenho em um pacote muito pequeno.

2.2 Regras da Categoria *Very Small Size Soccer*

As regras necessárias para o desenvolvimento de um time de robôs de futebol encontram-se disponíveis tanto em inglês quanto em português, disponibilizado pela IEEE. Todas as informações presentes nesta seção 2.2 podem ser encontradas no link da referência IEEE (2008) e foram adaptadas, para mais fácil compreensão do interlocutor sobre os aspectos mais necessários do esporte e para o desenvolvimento deste trabalho.

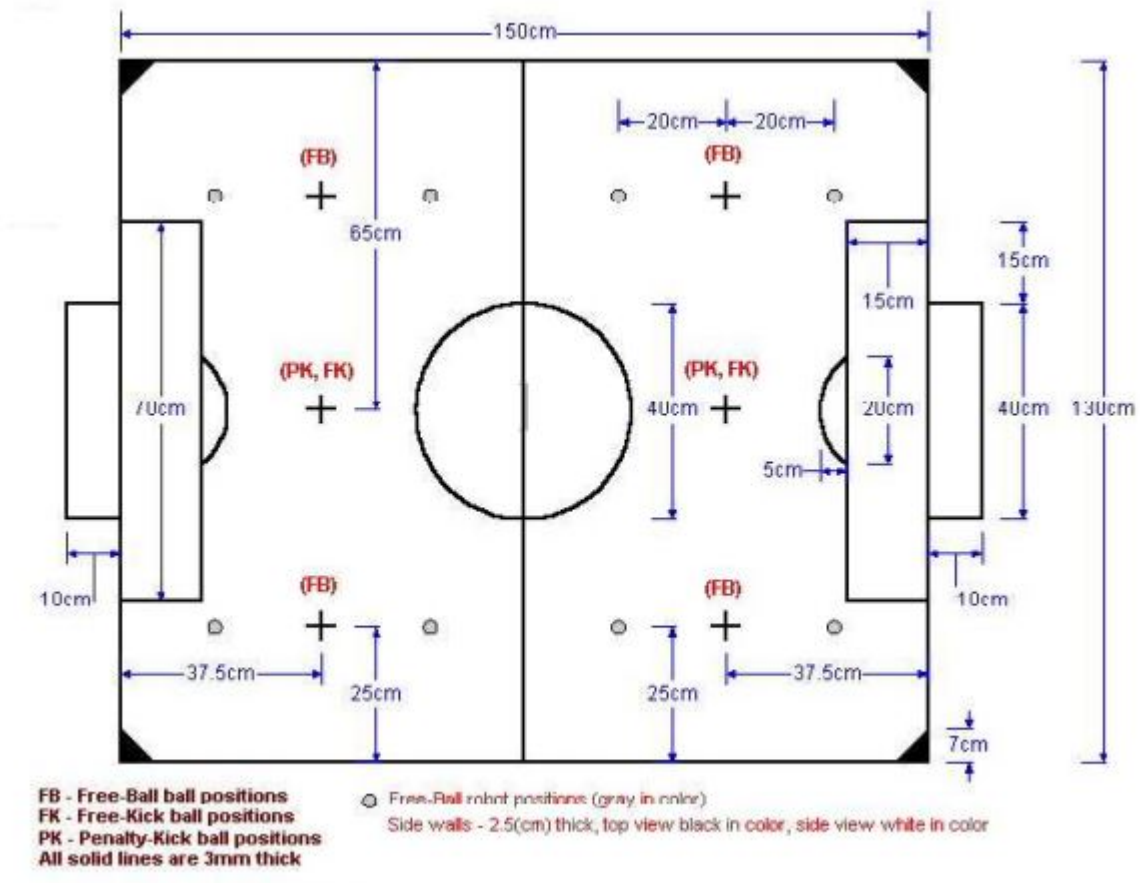
2.2.1 Campo e Bola

O campo e a bola são os elementos passivos do esporte. O campo é a superfície onde ocorre a movimentação dos demais elementos do jogo e a bola é o elemento que deve ser conduzido pelos robôs até a área denominada gol do time adversário.

2.2.1.1 Campo

O campo é composto por uma chapa plana e rígida de madeira medindo 150cm x 130cm, em cor preta fosca (não reflexiva) com laterais medindo 5cm de altura por 2.5cm de largura. A parte superior das laterais deverá ser na cor preta (a mesma cor do campo) e as paredes deverão ser na cor branca. Este campo deve ser localizado em um ambiente interno, coberto e sob intensidade luminosa fixa de 1000*Lux*. Medidas e marcações do campo podem ser visualizadas na Figura 3.

Figura 3 – Especificações do campo de futebol de robôs modalidade *Very Small Size Soccer*.



Fonte: IEEE (2008).

2.2.1.2 Bola

Deve ser utilizada uma bola de golfe laranja, com 42.7mm de diâmetro e 46g de massa, conforme apresenta a Figura 4. A bola ser da cor laranja, elimina esta tonalidade das etiquetas dos robôs.

Figura 4 – Bola de Golfe.

Fonte: Found Golf Balls (2018)

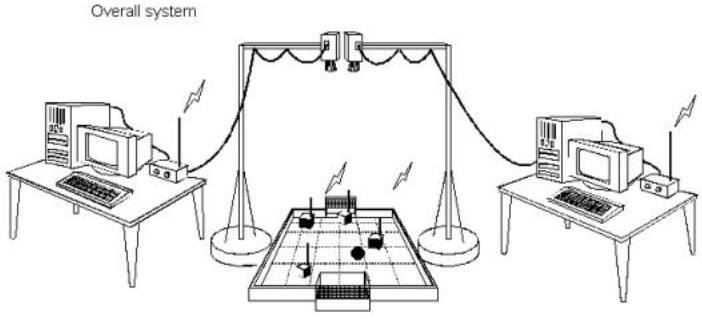
2.2.2 Jogadores

Os jogadores são a parte ativa do jogo. Estes são os robôs e onde está localizada a parte eletrônica e mecânica do esporte eletrônico, assim como o sistema de visão, estratégia e comunicação sem fio.

2.2.2.1 Sistema completo

Uma partida deve ser disputada por dois times, cada qual constituído por três robôs. Um dos robôs pode ser o goleiro. Somente três pessoas de cada equipe devem ficar no local da competição. Apenas um único computador por equipe pode ser usado, sobretudo dedicado ao processamento visual e para reconhecimento de posições. Isto pode ser observado na Figura 5.

Figura 5 – Visão geral dos elementos da modalidade *Very Small Size Soccer*.



Fonte: IEEE (2008).

2.2.2.2 Robôs

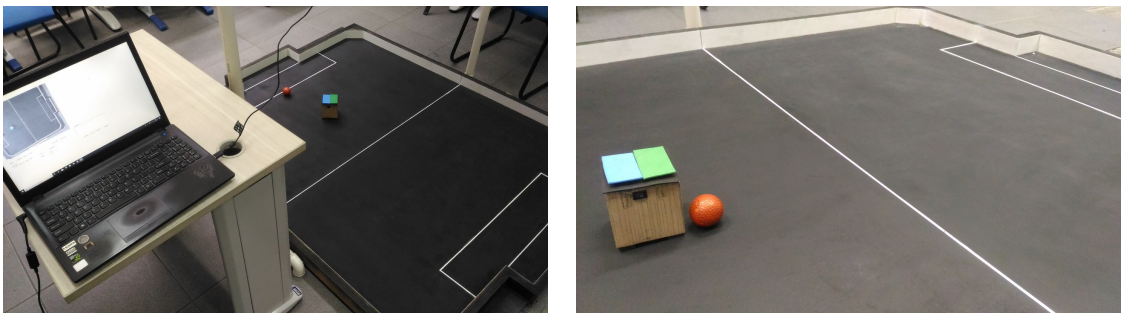
O tamanho de cada robô deve ser limitado a 7.5cm x 7.5cm x 7.5cm. A altura da antena de radio-frequência não deve ser considerada neste limite.

A parte superior dos robôs não deve conter as cores laranja, branco ou cinza e também não deve ser colorida com mais de duas cores diferentes da cor preto. Uma cor, que pode ser azul ou amarelo, a ser definida pelos organizadores, identificará times distintos. Todos os robôs de um time vestem uma das duas cores enquanto os outros a cor restante.

Uma etiqueta azul ou amarela, definida pelos organizadores, irá identificar os robôs de cada time. Todos os robôs devem ter visivelmente, no topo, uma região sólida da etiqueta do time, seja ela na cor azul ou amarelo. Essa região pode ser de qualquer forma, mas deve ser capaz de conter (pelo menos) um quadrado com 3.5cm de lado ou um círculo com 4cm de diâmetro. A cor de identificação de cada time poderá mudar de partida a partida, portanto a etiqueta usada deverá ser destacável. Quando os times estiverem com as etiquetas com as cores (azul ou amarelo) definidas, os robôs não devem ter nenhuma etiqueta da cor do time adversário.

Cada robô dever ser totalmente independente, com baterias e motores contidos em si mesmo. Somente comunicação sem fio deverá ser permitida para todos os tipos de interações entre o computador *host* e um robô. Um exemplo de ambiente de testes do esporte pode ser visualizado na Figura 6.

Figura 6 – Exemplo de campo da modalidade *Very Small Size Soccer*.



Fonte: O Autor.

2.3 Visão Computacional

A visão computacional é a área deste esporte que exige mais atenção, pois ela é a porta de entrada para a tomada de decisões. Assim, se ela for executada de maneira incorreta, o erro se propagará para todo o restante do processo, resultando em um time que responde incorretamente.

> [...] Mas o principal desafio do futebol de robôs se encontra nas áreas relacionadas à Inteligência Artificial (I.A.), como sistemas multi-agentes, aprendizado de máquina e visão computacional. Estes problemas citados não são triviais,

> visto que ambiente do Futebol de Robôs é dinâmico, incerto e probabilístico (MARTINS; TONIDANDEL; BIANCHI, 2006, p.173).

Um sistema de visão computacional para o futebol de robôs, segundo Martins, Tonidandel e Bianchi (2006), deve ser rápido e tolerante à variação de luminosidade e é desejável que seja capaz de tolerar ruídos e variações de intensidade luminosa.

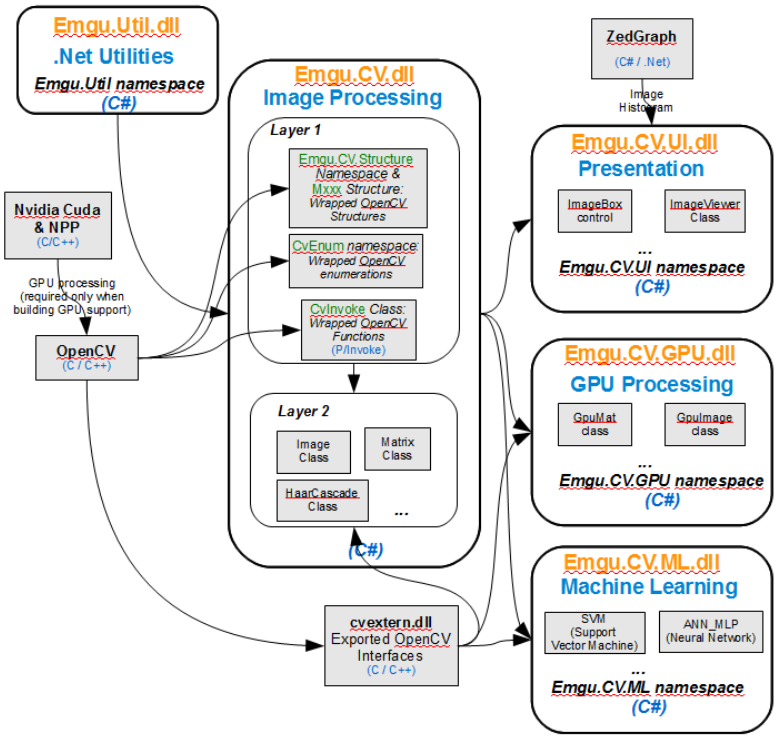
Neste trabalho, busca-se atender às necessidades da visão computacional no futebol de robôs, utilizando os elementos descritos nesta seção 2.3.

2.3.1 Biblioteca EmguCV

A biblioteca EmguCV é um *wrapper* do OpenCV em C++, já muito difundida e utilizada no processamento e detecção de imagens. Ela permite a utilização das funções do OpenCV no ambiente .NET compatível com diversas linguagens (WIKI, 2018, Tradução nossa).

Como é esclarecido na Figura 7, o *EmguCV* é uma biblioteca recente e apresenta, além das funcionalidades herdadas da *OpenCV* (Camada 1 (um) *Layer 1*), diversas classes que permitem aproveitar também da programação .NET (Camada 2 (dois) *Layer 2* por exemplo).

Figura 7 – Visão geral da arquitetura EmguCV.



Fonte: Wiki (2018)

Com o uso desta biblioteca, faz-se possível a implementação de diversas soluções na parte de visão computacional, desde transformações e conversões simples nas imagens, até elementos mais complexos como, por exemplo, aprendizado de máquina.

2.3.2 Coleta de Imagens

Segundo Martins, Tonidandel e Bianchi (2006) o sistema de visão computacional usado no futebol de robôs deve ser, dentre outros atributos, rápido e, para isso, é necessário que a entrada de dados neste sistema ocorra também de forma veloz.

Tendo isto em vista, a coleta de imagens deve ser realizada por uma câmera capaz de detectar 30 quadros por segundo ou mais, de maneira constante. Quanto à resolução de captura, deve-se ter em vista que o tempo de processamento da imagem é diretamente proporcional ao seu tamanho, então esta deve ser suficientemente grande para a extração de dados e não tão extensa, a ponto de prejudicar a performance do sistema.

2.3.3 Processamento de Imagens

De acordo com Albuquerque e Albuquerque (2000), o processamento de imagens vem do processamento de sinais e processar uma imagem consiste em transformá-la, sucessivamente, com o objetivo de extrair mais facilmente a informação nela presente, ou seja, o processamento de imagens é onde a imagem bruta capturada do mundo real passa por transformações, de maneira que resulte em uma imagem ou conjunto de imagens que podem ser analisadas, em muitos casos, algorítmicamente e, assim, retirando dos dados, informações como contornos, formas, dentre outras propriedades possíveis.

As subseções a seguir são utilizadas no trabalho para explanar sobre a solução escolhida para desenvolvimento. Utilizar o padrão de arquitetura proposto significa que tanto ela pode ser utilizada quanto substituída, desde que sua saída, o posicionamentos dos elementos do jogo a serem interpretados pela estratégia, seja mantida.

2.3.3.1 Detecção por Cores pelo Espaço HSV

Segundo Alves (2010), a representação do espaço de cores no sistema RGB *(red green blue)* pode ser feita através do chamado *cubo RGB*, conforme observado na Figura 8. Ainda de acordo com o autor, para se obter uma determinada cor é utilizado um intervalo pré-especificado, que pode ser, por exemplo, de 0 a 255, onde a cor preta é obtida por (0,0,0) a cor branca por (255,255,255), (255,0,0) resulta em vermelho, (0,255,0) em verde e (0,0,255) no azul. Os demais vértices representam o complemento de cada cor primária, sendo estas amarelo, ciano e magenta, cada ponto no interior do cubo corresponde a uma cor representada pela tripla (R,G,B) e os tons de cinza são representados ao longo da diagonal principal do cubo.

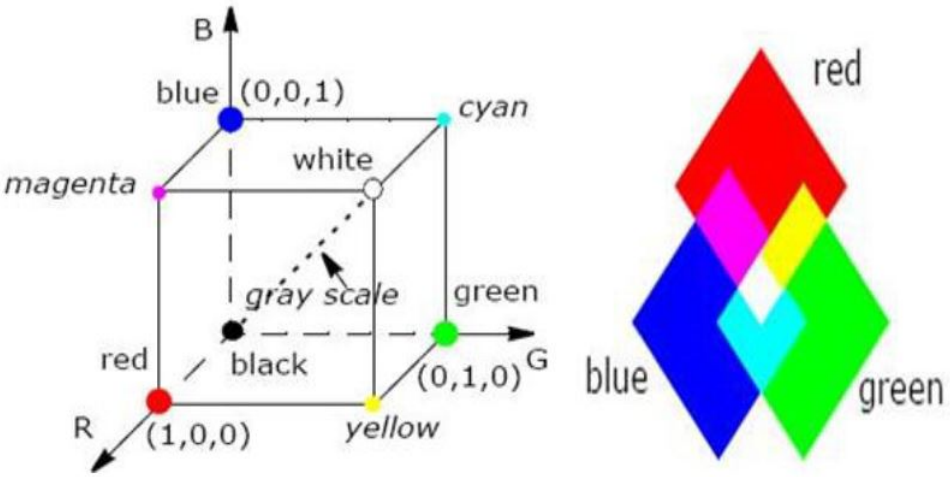
Apesar de ser útil na representação de cores, o modelo RGB não é interessante para a definição de cores, baseando-se no sistema de percepção visual humano.

> O modelo RGB possui uma desvantagem muito forte: ele não é adequado para definição de cores baseando-se no sistema de percepção visual humano. Isso significa que nada garante que cores com representação próxima no espaço RGB sejam próximas em termos de percepção visual. Fica difícil determinar, visualmente, com exatidão, se uma cor é ou não a de interesse. (ALVES, 2010, p.28)

Para contornar esse problema, é necessária uma mudança de abordagem, que por muitos autores, como Alves (2010), Ganesan e Rajini (2014), Ganesan et al. (2014) e Barros et al. (2015), é solucionado pela adoção do espaço de imagem *HSV (hue saturation value)*.

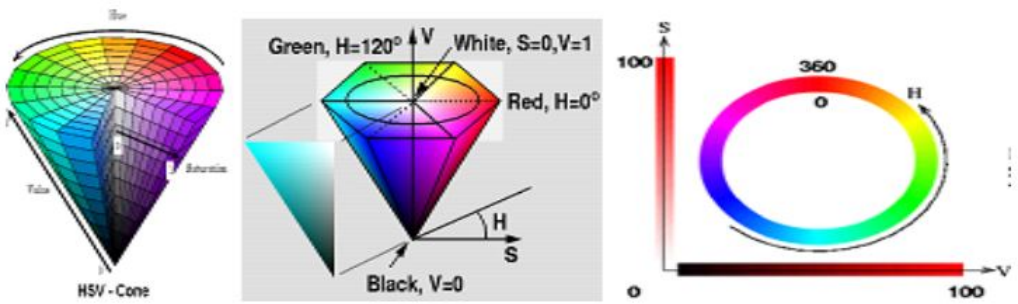
O sistema HSV é representado por coordenadas em um espaço dentro do qual o modelo definido é um cone em forma hexagonal conforme ilustra a Figura 9. Segundo Ganesan e Rajini (2014) este espaço de cores, assim como o RGB, é uma representação matemática das cores em um modelo tridimensional, sendo suas coordenadas hue ou matiz, saturação e valor.

Figura 8 – Espaço de cores RGB



Fonte: Ganesan et al. (2014).

Figura 9 – Espaço de cores HSV.



Fonte: Ganesan et al. (2014).

> The hue is an angle from 0 to 2π (360 degrees), typically 0 or 360 degree represents red color, 60 degree represents yellow color, 120 degree represents green color, 180 degree represents cyan color, 240 degree represents blue color, and 300 degree represents magenta color. In this way, all the colors are represented in the HSV color space. The saturation defines the purity of the color or hue i.e., what amount of white color is mixed with hue. The third component in HSV color space is the value which represents the brightness of the color. (GANESAN; RAJINI, 2014, p.2)

Os componentes HSV são os seguintes:

1. O hue ou matiz verifica o tipo de cor, abrangendo todas as cores do espectro, desde o vermelho até violeta, mais o magenta. Atinge valores de 0 a 360, mas para algumas aplicações, esse valor é normalizado de 0 a 100%.
2. Saturação define a pureza da cor, ou seja, quanto menor esse valor, mais com tom de cinza aparecerá a imagem. Quanto maior o valor, mais pura é a imagem. Atinge valores de 0 a 100%.
3. O valor define o brilho da cor. Atinge valores de 0 a 100%.

$$H = \arccos \frac{\left(\frac{1}{2}\right)(2R - G - B)}{\sqrt{(R-G)^2 - (R-B)(G-B)}} \tag{2.1}$$

$$S = \frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)} \tag{2.2}$$

$$V = \max(R,G,B) \tag{2.3}$$

Esta conversão é facilmente realizada através da utilização das funções presentes na biblioteca *EmguCV*. Por exemplo, converter uma imagem do espaço RGB para o HSV faz-se:

Algoritmo 1 – Código C# EmguCV conversão entre espaços de imagem. Fonte: O Autor.

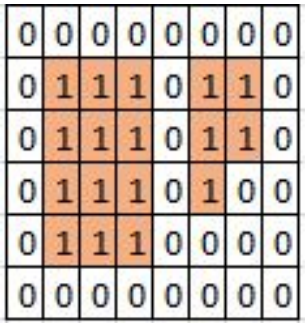
```
Emgu.CV.Image<Hsv, Byte> novaImgHSV = imgRGB.Convert<Hsv, Byte>();
```

2.3.3.2 Agrupamento em *Blobs*

Após a captura e conversão entre espaços de imagem (*RGB* para *HSV*) e a aplicação de filtros para redução de ruído, é possível identificar onde estão localizadas as cores dos elementos de jogo. A maneira como são selecionadas estas cores está descrito no Capítulo 3. Onde as cores são identificadas é denominada as *ROI (region-of-interest)* ou *AOI (area-of-interest)* da imagem.

Para esta solução escolhida, cada região de interesse é identificada por agrupamentos de pixeis semelhantes (mesma cor), originando assim as áreas denominadas *blobs*. A Figura 10 mostra como este agrupamento é realizado. Neste exemplo, as regiões de interesse

Figura 10 – Região de Interesse, *ROI (region-of-interest)*



Fonte: O Autor.

são onde há os algarismos 1. Um algoritmo identificador de *Blobs* reconhece onde há o agrupamento dos semelhantes.

A biblioteca *EmguCV* também possui algoritmos para este reconhecimento. Estes podem ser utilizados da seguinte forma:

Algoritmo 2 – Código C# EmguCV Identificando Blobs. Fonte: O Autor

```
Emgu.CV.Cvb.CvBlobDetector blobDetector;
blobDetector = new Emgu.CV.Cvb.CvBlobDetector();
CvBlobs blobsIdentificados = new CvBlobs();
blobDetector.Detect(imagemBinaria, blobsIdentificados);
```

Neste trecho a *imagemBinaria*, que é uma imagem resultante do pré-processamento, passa pelo algoritmo da biblioteca, que identifica os *blobs* desta imagem e os coloca no objeto *blobsIdentificados*. Estes blobs contém diversas informações, que são utilizadas para a interpretação do posicionamento dos robôs.

2.3.3.3 Interpretação dos *Blobs*

Após serem definidos os *blobs*, é necessário interpretá-los. Foi então adotado que o centroide de cada *blob* é também a posição dos objetos que eles representam.

Através da interpretação dos *blobs*, são obtidos os pontos que são as localizações de cada elemento do jogo. Esta informação então segue para a estratégia, para serem tomadas as decisões.

2.4 Estratégia

Segundo Guarnizo e Mellado (2016), o futebol de robôs apresenta-se como um ambiente multi-agente hostil com incertezas, onde robôs devem operar na busca de um objetivo comum. No futebol robôs, a estratégia é definida como o plano de equipe com o que se destina a ganhar o jogo, a tática é definida como a organização da equipe durante um dado momento (a cada *frame*). Os papéis correspondem à combinação de comportamentos dos robôs junto a

sua localização no campo de jogo, por exemplo goleiro, meio-campista, defensor ou atacante. Os comportamentos correspondem aos movimentos que realizam os robôs para executar tarefas específicas, como ir para a bola, bloqueando um adversário ou fazendo um passe.

A estratégia é a camada onde os pontos identificados pela visão computacional são analisados e são definidas as ações que cada robô deve tomar. Como há a divisão de papeis, cada robô assume uma postura, de acordo com a cena obtida. No caso da *Very Small Size Soccer* há três robôs, o goleiro que deve impedir a entrada da bola na área definida como gol, o zagueiro que deve bloquear ataques desferidos pelo time adversário e o atacante deve conduzir a bola ao gol inimigo.

Há diversas possibilidades de solução para esta camada, como por exemplo, máquinas de estados finitos, que foi utilizada por Guarnizo e Mellado (2016), ou inteligência artificial, usada por Gurzoni et al. (2011).

Para o desenvolvimento da estratégia utilizada neste trabalho, foi usado como base o simulador *SimuroSot*, que além de possuir estratégias prontas, oferece um ambiente de simulação para testes destas e de novas estratégias.

2.4.1 SimuroSot da FIRA

A *Federation of International Robot-Sport Association* (FIRA) é uma das grandes organizações internacionais que promovem eventos de fomento à pesquisa em robótica, com enfoque na aplicação de futebol de robôs.

> Iniciativas de âmbito mundial têm surgido como forma de fomento à pesquisa em robótica com enfoque na aplicação de futebol de robôs. Algumas destas iniciativas surgem em forma de competições científicas entre institutos de pesquisa e universidades, que costumam inscrever anualmente seus times robóticos. A FIRA (Federation of International Robot-soccer Association) é uma das grandes organizações internacionais que promovem este tipo de competição [...] (RABELO; MACEDO; FREIRE, 2018, p.686).

Segundo Rabelo, Macedo e Freire (2018), uma das modalidades de competição proposta pela FIRA é a *SimuroSot*, categoria de simulação que consiste de um servidor e dois programas clientes, que executam estratégias de jogo pré-programadas. Como este simulador utiliza a mesma entrada e saída da estratégia utilizada pelo padrão de arquitetura proposto deste trabalho, posicionamento dos elementos do jogo e velocidades para as rodas direita e esquerda de cada robô.

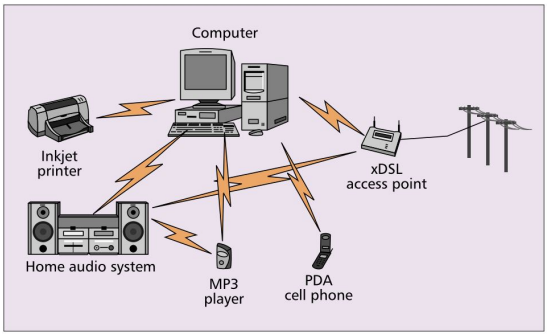
> Os robôs dessa modalidade são dispositivos movidos por dois motores elétricos de corrente contínua, dispostos em uma configuração denominada tração diferencial, onde são usadas duas rodas, uma à direita e outra à esquerda, cada uma acionada de forma independente por um dos motores. A navegação do robô no campo de jogo é feita com base em informações a respeito do ambiente de jogo, como a posição da bola e de cada jogador. Para levar o robô de um ponto a outro, são atribuídos valores de velocidade para as rodas direita e esquerda, que determinam a velocidade e direção de movimento do robô (RABELO; MACEDO; FREIRE, 2018, p.687).

2.5 Comunicação Sem Fio

Segundo Sairam, Gunasekaran e Rama Reddy (2002) o *bluetooth* é um método para comunicação de dados que usa links de rádio de curto alcance, para substituir cabos entre computadores e suas unidades conectadas. Ele foi inventado em 1994 por L. M. Ericsson, da Suécia.

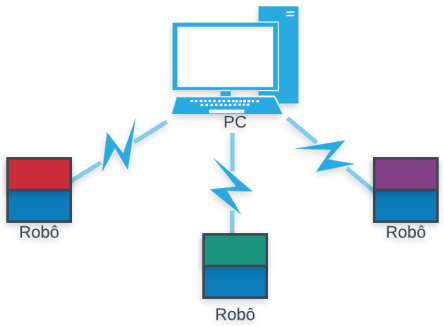
O funcionamento da rede *bluetooth* permite precisamente o que a rede de futebol de robôs necessita, uma conexão sem fio capaz de suportar um dispositivo mestre e os demais escravos (três), além de proporcionar uma conexão estável à curta distância e numa velocidade para transmissão de dados suficiente, já que as mensagens que os robôs consomem são ordens que demandam baixa quantidade de dados. A Figura 11 ilustra o funcionamento de uma rede *bluetooth* e a Figura 12 exibe como ocorre a comunicação do sistema do futebol de robôs.

Figura 11 – Rede *Bluetooth*.



Fonte: Sairam, Gunasekaran e Rama Reddy (2002).

Figura 12 – Rede *Bluetooth* no ambiente futebol de robôs.



Fonte: O Autor.

Os robôs desconhecem a existência uns dos outros, esta coordenação é toda realizada no mestre, identificado na visão computacional e tratado na estratégia.

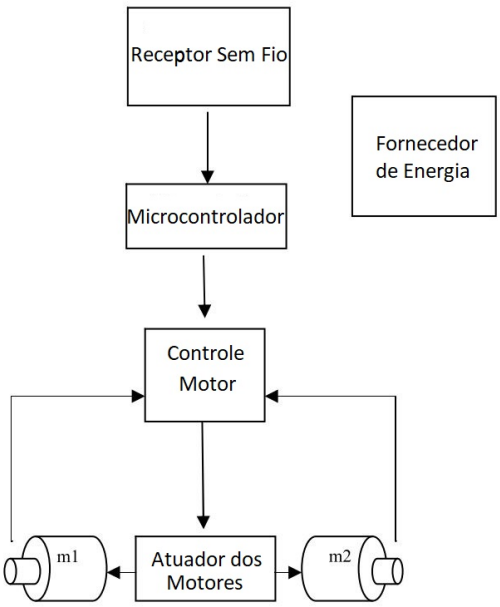
2.6 Mecânica e Eletrônica dos Robôs

Segundo Jusoh e Omar (2011), vários fatores influenciaram o projeto do robô como: peso do robô, material usado, tamanho e tipo do motor. A movimentação do robô também está diretamente relacionada a geometria dos motores, projeto das rodas e relação de engrenagem entre o motor e a roda, se os robôs têm rodas largas e pneus com alta fricção.

Além disso, o projeto mecânico precisa realizar a tarefa com a maior precisão possível, permanecer no caminho dado, encaixar em um cubo de comprimento de borda de 7,5 cm, ter baixo ponto de gravidade para reduzir escorregões, não ter frente ou verso (robô simétrico), ter rolamentos de rolos para montagem das rodas, possibilidade de impulsionar a bola, possuir engrenagem de um único estágio, motor com codificadores de alta resolução e pequena tolerância de movimento.

Ainda de acordo com Jusoh e Omar (2011), o projeto eletrônico consiste em três módulos principais: módulo de controle do sistema, de comunicação e de potência (fonte de alimentação para cada módulo), que podem ser observados na Figura 13. A complicada placa eletrônica, em diversos projetos, precisa de vários níveis de tensão para fornecer energia ao seus módulos.

Figura 13 – Arquitetura básica do circuito de um robô.



Fonte: Adaptado de Jusoh e Omar (2011).

Com o intuito de simplificar a implementação desta etapa, opta-se por utilizar uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador *Atmel AVR*, com suporte de entrada/saída embutido. Esta abordagem permite unir a parte de recepção sem fio, microcontrolador e o controle de movimento das rodas. Com isso ainda é possível a utilização de uma única bateria para fornecer energia ao sistema de controle e aos motores, com a utilização de uma ponte H.

3 Materiais e Métodos

“Os métodos são as verdadeiras riquezas”.
Friedrich Wilhelm Nietzsche

A proposta deste trabalho consiste em definir um padrão de arquitetura desacoplado para o futebol de robôs, comprovar efetividade do padrão, através de testes em algoritmo simplista desenvolvido a partir dele e, dentro deste segundo, definir um protocolo de comunicação entre mestre e robôs, sendo este mestre um computador pessoal. Sendo assim, por se tratar de uma pesquisa que acessa e usa teorias, conhecimentos e técnicas acadêmicas, buscando uma solução para o problema apresentado, segundo Roll-Hansen (2009), por apresentar estas características, este trabalho trata-se de uma pesquisa aplicada.

Os procedimentos utilizados neste trabalho subdividem-se então em elaboração de um padrão de arquitetura desacoplado para o futebol de robôs (seção 3.1), desenvolvimento do algoritmo baseado no padrão (seção 3.2) e testes comprobatórios da efetividade do padrão arquitetural, proposto através da utilização do sistema desenvolvido (seção 3.4).

3.1 Construção do Padrão de Arquitetura de Baixo Acoplamento

Um padrão arquitetural é um conceito que resolve e delineia alguns elementos coesos, essenciais de uma arquitetura de software. Segundo That, Sadou e Oquendo (2012), padrões arquiteturais são padrões de software que oferecem soluções bem estabelecidas, para problemas arquiteturais em engenharia de software. Ele fornece uma descrição detalhada de seus elementos e relações, juntamente com um conjunto de restrições. Um padrão arquitetural expressa um esquema fundamental para o sistema de software, que consiste em subsistemas, suas responsabilidades e inter-relações.

That, Sadou e Oquendo (2012) ainda afirmam que um padrão de arquitetura é um conceito que captura elementos essenciais da arquitetura de software. Muitas arquiteturas diferentes podem implementar o mesmo padrão e, portanto, compartilhar as mesmas características.

Este novo padrão arquitetural visa definir precisamente as responsabilidades de cada parte do processo, propondo assim uma solução geral e reutilizável para o problema, um sistema para o futebol de robôs categoria *Very Small Size*. Cada subárea do futebol de robôs será separada em um componente e uma arquitetura, pensada de modo a desacoplar os elementos do esporte, ordenar estes componentes e definir como eles se comunicam. As características a se alcançar neste padrão de arquitetura são:

- Baixo acoplamento;
- Escalabilidade;
- Encapsulamento de responsabilidades.

As desvantagens, fraquezas e impossibilidades pontuais de conseguir o desacoplamento total neste padrão arquitetural serão apontadas e justificadas no desenvolvimento.

3.2 Desenvolvimento do Sistema Baseado no Padrão

Tendo como base o padrão de arquitetura proposto, são preenchidos os componentes e, de acordo com a solução desejada, tem-se:

1. Visão computacional: Processamento de imagens com uso da biblioteca EmguCV.
2. Estratégia: Maquina de estados finitos;
3. Comunicação sem fio: *Bluetooth*;
    a) utilização do protocolo proposto.
4. Eletrônica e mecânica: *Arduino*, ponte H e motores *Pololu*.

As etapas, agora chamadas de componentes, são desenvolvidas obedecendo as definições feita no padrão de arquitetura.

3.3 Construção do Robô

Os robôs, dentro dos padrões IEEE para a competição, não são encontrados como produtos prontos no mercado atual. Sabendo disso, para a construção dos robôs, devem ser confeccionadas as peças da carcaça, a parte eletrônica, mecânica e programável, sendo livre a escolha dos materiais utilizados, desde que sujeitos às regras da IEEE.

Esta construção é realizada dentro dos preceitos propostos por Jusoh e Omar (2011), onde são definidas boas práticas para a organização do robô como distribuição e ordem dos componentes.

3.4 Testes Validadores

Para avaliar o correto funcionamento do algoritmo desenvolvido, com base no padrão arquitetural, e a eficácia dele, os seguintes casos de teste são utilizados:

1. Implementação de algoritmo, dentro do padrão arquitetura, que reja a movimentação de um robô;
2. Os testes para comprovação do funcionamento do robô, tendo sucesso, comprovam a eficácia total ou parcial da eletrônica, mecânica e do protocolo utilizados. Estes testes são:
    a) Robô vai até um ponto;
    b) Robô segue trajetória de n pontos.

Caberá a trabalhos futuros forçar o padrão proposto, definindo seus limites. Os testes sobre o algoritmo e o padrão são feitos analisando os tempos de execução. São realizados,

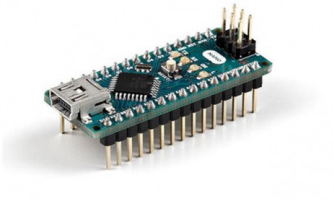
primeiramente, sobre os componentes isolados (visão computacional, estratégia e comunicação sem fio) e posteriormente todos juntos. Neste segundo caso, o teste abrangerá desde a captura da imagem do ambiente até a movimentação do robô, e será realizado utilizando os casos de teste de movimentação (Item 2), para comprovação do funcionamento do robô supracitado.

3.5 Materiais Utilizados

Foi utilizado para os testes o seguinte ambiente:

- Campo feito em placa de fibra de média densidade segundo as regras IEEE;
- Câmera Logitech C920 HD Pro Webcam;
- Computador Pessoal com a seguinte configuração:
  Processador Intel(R) Core(TM) i7-4810MQ CPU @ 2.80GHz 2.80GHz;
  Ram: 8,00 GB;
  Sistema Operacional: Windows 10, 64 bits, processador com base em x64.
- Robô composto por:
  Motores: Pololu (Figura 14b);
  Corpo impresso em poliácido láctico (PLA): Pololu;
  Capa externa em policloreto de polivinila (PVC);
  Etiquetas de várias cores em espuma vinílica acetinada (EVA);
  Bateria 7400Mah 7,4V (Figura 14d);
  Plataforma de prototipagem eletrônica programável (Figura 14a);
  Módulo *bluetooth* (Figura 14c);

Figura 14 – Componentes utilizados nos robôs do presente trabalho.



(a) Plataforma de prototipagem eletrônica de hardware livre e de placa única, Arduino Nano. Fonte: Arduino (2018).

(b) Motor Pololu de eixo único. Fonte: Pololu (2018).

(c) Modulo *bluetooth* HC-06 para *Arduino*. Fonte: Portal Vida de Silício (2018).

(d) Bateria 7400Mah 7,4V. Fonte: O Autor.

4 Padrão de Arquitetura para Futebol de Robôs Categoria VSS

“O apetite, ligado à crença de conseguir, chama-se esperança”.
Thomas Hobbes de Malmesbury, Leviatã

O desenvolvimento organiza-se com a criação do padrão de arquitetura propondo e explicando as decisões adotadas e em seguida há a utilização do padrão na implementação.

4.1 Padrão de Arquitetura Proposto

O padrão arquitetural proposto, busca estabelecer o fluxo e regras para a implementação de um software, que coordene um time de Futebol de Robôs, categoria *Very Small Size*, de forma baixamente acoplada. Para isso, ele utiliza tipagem genérica e é empacotado em uma *Dynamic-link library* (DLL). Esta DLL pode ser encontrada num repositório da plataforma GitHub[1].

Para explicar as decisões de arquitetura, primeiro são detalhadas as informações que fluem pelas partes do esporte eletrônico relatado, como um processo contínuo. Então são definidas as informações transportadas entre os atuadores do processo e estes mesmos atuadores. Depois é exibido o que é a DLL disponibilizada e como usá-la. Na seção 4.2 é disposta a implementação realizada, seu uso de forma padrão e escalável.

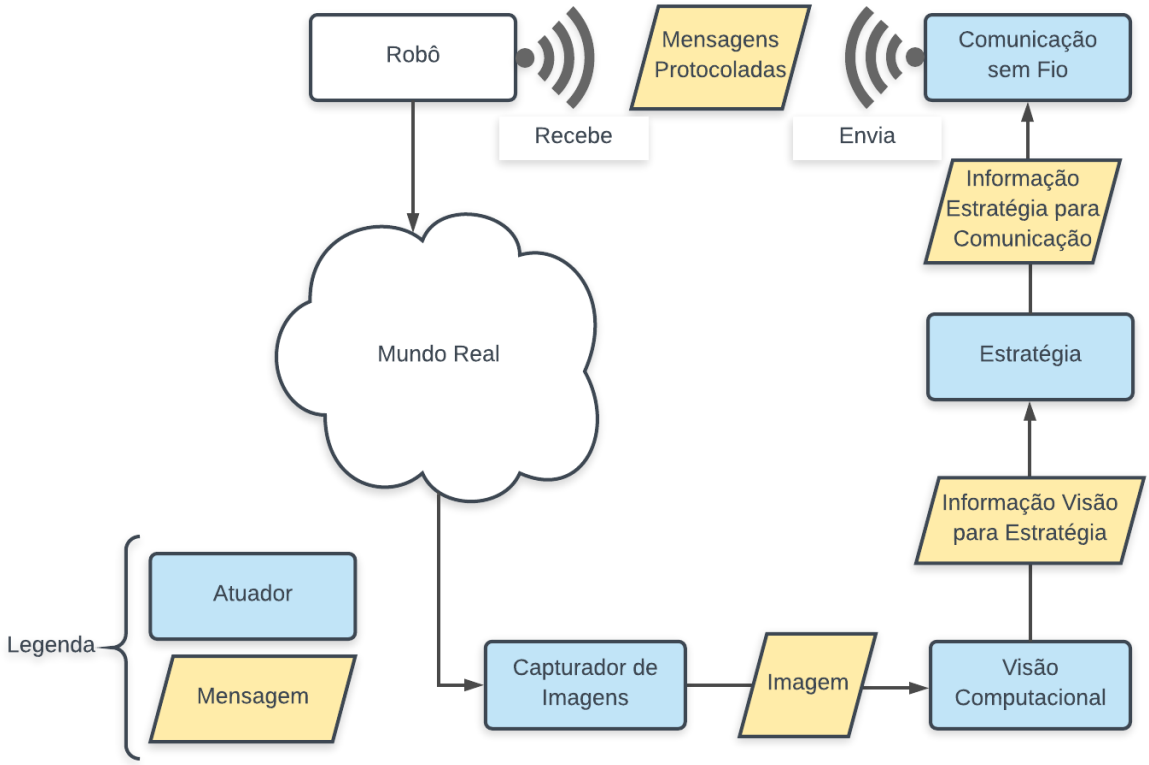
4.1.1 Fluxo de Informações na Arquitetura

Deve-se ter em mente que os atuadores dentro de um software de Futebol de Robôs, categoria *Very Small Size*, seguem um caminho que vai da captura da imagem até o envio da mensagem para o robô. Dentro deste processo, tem-se os atuadores e as mensagens que trafegam entre eles, que pode ser visualizado na Figura 15.

Internamente, ao se iniciar o processo de execução de uma partida, o Controle segue o seguinte processo de execução. Quando uma imagem é enviada pelo Capturador de Vídeo ao Controle o processo é inicializado, essa imagem é enviada para os componentes responsáveis para detectar a Bola e o Campo retornando o *VtoEBola* e *VtoECampo*. Então, para cada robô presente, é executada a etapa de visão, resultando para cada um uma individual *VtoERobo*. Após isso, as etapas de estratégia dos robôs recebem seus respectivos *VtoERobo* e os *VtoEBola* e *VtoECampo* resultando em um *EtoCRobo* para cada robô, que por fim é enviado a seus respectivos componentes responsáveis pela comunicação sem fio, que protocolam e enviam a mensagem para a eletrônica do robô, seguindo o ciclo demonstrado na Figura 15.

[1] Repositório de nome GitHub RobotSoccerLib disponível em: https://github.com/alexsilvar/RobotSoccerLib Publicado em: 30/09/2018 por Alexsander Ramos da Silva.

Figura 15 – Processo do Software do Futebol de Robôs categoria *Very Small Size*.



Fonte: O Autor.

Para reger este fluxo e as informações, a arquitetura proposta conta com uma classe controladora, denominada Controle. Este, por sua vez, permite a exibição de informações relevantes (imagens com e sem processamento), permite a definição do ambiente e o mais importante, onde manifesta-se a utilização genérica, o desenvolvedor implementa livremente a visão computacional, estratégia e comunicação, desde que essa cumpra o contrato fornecido pela biblioteca. Este contrato pode ainda ser elaborado pelo próprio desenvolvedor devido a utilização de tipos genéricos.

4.1.2 Tipagem Genérica

A tipagem genérica permite que, no momento da criação de uma instância do controle os tipos sejam definidos, a partir daí todos os parâmetros passarão a seguir o que foi definido no momento da instanciação.

Os tipos genéricos estabelecidos pelo controle são: *Img, VtoERobo, EtoCRobo, VtoEBola, VtoECampo* e *PlaceToDraw*.

Algoritmo 3 – Código C# Controle e seus tipos genéricos. Fonte: O Autor

```
Controle<Img, VtoERobo, EtoCRobo, VtoEBola, VtoECampo, PlaceToDraw>
```

O tipo *Img* é o tipo de imagem que trafega para para os atuadores de visão computacional. Os tipos *VtoERobo, VtoEBola* e *VtoECampo* são movidos da visão computacional para a estratégia. O tipo *EtoCRobo* vai da estratégia do robô para sua comunicação sem fio, e o tipo *PlaceToDraw* é o tipo que receberá o desenho das imagens, tanto capturadas quanto processadas, servindo para a visualização na interface de usuário.

Assim sendo, para criar um Controle primeiramente é necessário definir quais as informações vão trafegar nele. Estas informações podem tanto ser tipos mais básicos de dados como inteiros, caracteres ou vetores até classes ou estruturas. Literalmente qualquer tipo comportado pela linguagem pode ser utilizado.

4.1.3 Interfaces dos Atuadores

Para ser possível a substituição de atuadores, para promover uma alta customização, foram definidas interfaces para cada um deles (captura de vídeo, visão computacional, estratégia e comunicação sem fio). Ao implementar estas interfaces, a classe deve utilizar os mesmos tipos definidos na instanciação do Controle. As interfaces definidas podem ser visualizadas abaixo.

Algoritmo 4 – Código C# Interfaces e seus tipos. Fonte: O Autor

```
public interface IVideoCaptura<Img, PlaceToDraw>
public interface IVisao<Img, VtoERobo, PlaceToDraw>
public interface IEstrategia<VtoERobo, EtoCRobo, VtoEBola, VtoECampo>
public interface IComunicacao<EtoCRobo>
```

Exemplificando este trecho de código, uma classe chamada “VisaoRobo”, que faz parte da solução criada neste trabalho, implementa a interface providenciada pela DLL “IVisao” cujos tipos *Img* foram definidos como *Bitmap*, *VtoERobo* definido como uma classe chamada *InfoVtoERobo* e *PlaceToDraw* definido como *PictureBox*. A mesma interface é utilizada para a bola e campo, porém sendo modificado o tipo *VtoERobo* para *VtoEBola* e *VtoECampo*, respectivamente, sendo estes os tipos que conterão as informações da bola e do campo e são destinados para uso nas etapas de estratégia dos robôs.

4.1.4 O Piloto

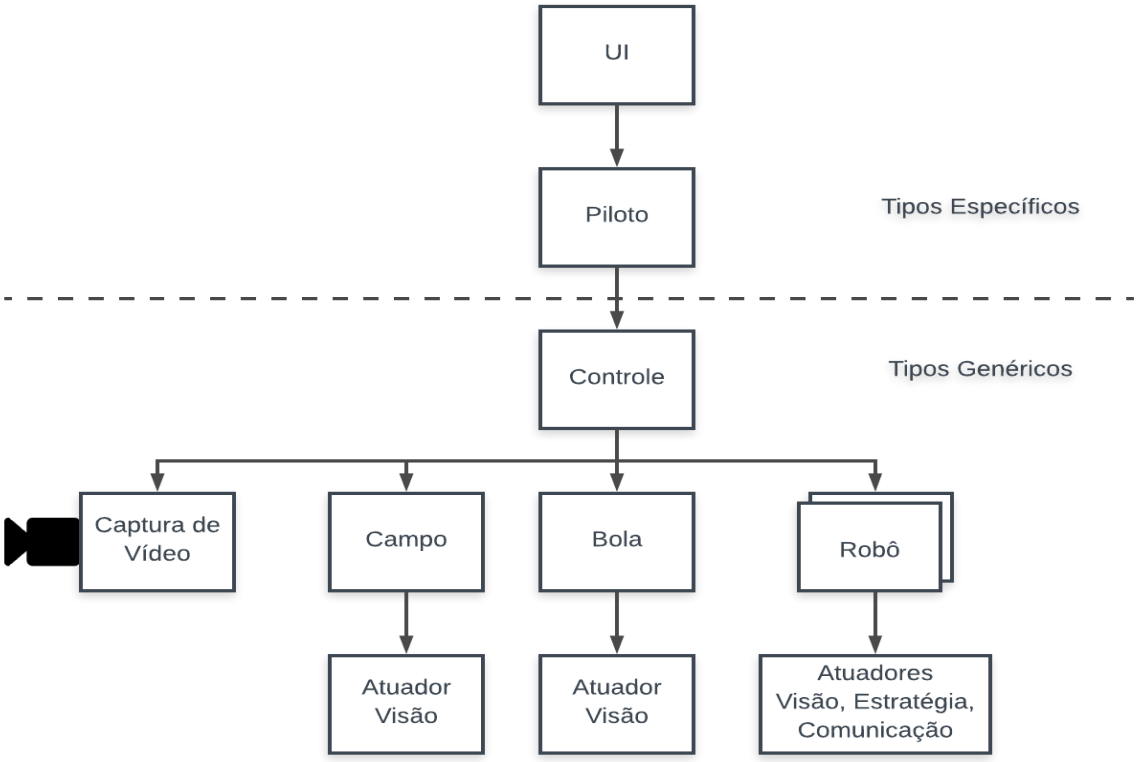
Por ser genérico, o Controle demanda um elemento que o instancie, definindo finalmente os tipos de mensagens a serem utilizados. Este elemento trata-se da real implementação sobre a arquitetura, ou seja, nele se dá o desenvolvimento do software do futebol de robôs. Este software possuirá as características proporcionadas pelo padrão.

Assim, quem é realmente utilizado pela UI é o Piloto. Para a criação do Piloto são selecionados, no momento da instanciação de um elemento Controle, quais as informações serão trafegadas entre as partes, ou seja, o piloto enfim fixa como os atuadores se comunicam.

O Piloto possui em geral as mesmas funções do controle, porém, pode criar regras específicas para o seu funcionamento.

A exemplo de regras específicas, supondo um robô com duas rodas que se movimentam com velocidade de -255 a +255, o Piloto pode definir em sua função de controle manual que, ao invés de receber um *EtoCRobo*, receba as velocidades das rodas, crie um elemento específico que tenha definido, atribua os valores das rodas recebidos e então passe ao seu Controle o objeto gerado. A Figura 16 demonstra a separação entre a parte que utiliza tipos genéricos e a que não o faz.

Figura 16 – Divisão entre a parte Genérica e Específica da Arquitetura.



Fonte: O Autor.

4.1.5 Comportamento da Arquitetura

A arquitetura propõe uma nova divisão em que há um controlador (Controle) que permita a substituição de seus componentes pelo utilizador da biblioteca. O controlador possui seus componentes: capturador de imagens, bola, campo e uma lista de robôs.

O Controle permite que os atuadores destes componentes sejam substituídos por outros, desde que criados tendo como base a tipagem especificada na instanciação do Controle, feita pelo Piloto, e implementem as respectivas interfaces do componente que almeja-se redefinir.

A exemplo disso, pode-se especificar que, dado um caso de implementação onde o Controle tenha sido instanciado utilizando tipos específicos da maneira abaixo, este controle

passará a aceitar a livre troca de componentes atuadores que utilizem a mesma tipagem que ele.

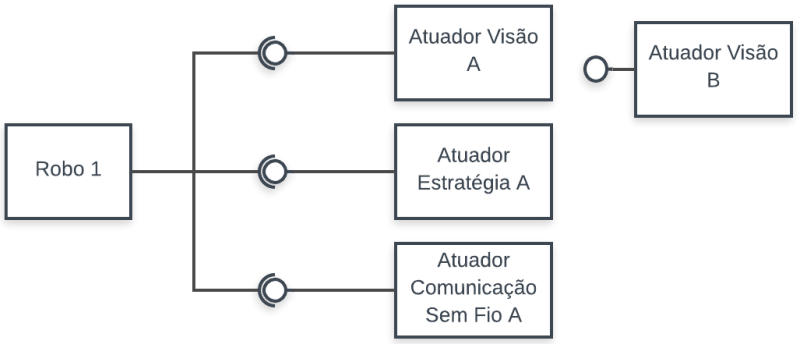
Algoritmo 5 – Código C# Implementação do controle específico. Fonte: O Autor.

```
Controle<Bitmap , InfVERbo , InfECRbo , InfVEBol , InfVECamp , PicBox>controle ;
```

De acordo com os componentes do Controle, além desses atuadores poderem ser alterados, promove-se o desacoplamento, principalmente, quando se diz respeito aos robôs. Este desacoplamento ocorre uma vez que cada robô receberá uma instância de cada atuador (visão, estratégia e comunicação). É possível, por exemplo, que um robô A utilize uma visão computacional A, estratégia A e comunicação A, e um robô B utilize uma visão computacional B, estratégia B e comunicação B, onde A é diferente de B. Fica manifesto então que robôs até mesmo com comunicação sem fio diferentes (a exemplo *bluetooth* e rádio) podem ser utilizados simultaneamente desde que sejam atribuídos corretamente os atuadores. A Figura 17 e Figura 18 demonstram como se dá a substituição de uma aplicação da visão computacional de um robô por outra.

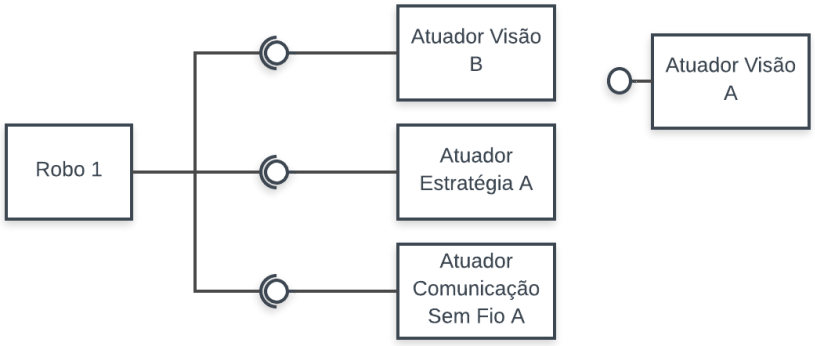
Através deste desacoplamento, fica explicitado a diminuição no esforço de manutenção ou alteração em uma área do jogo. Basta que seja criada uma nova classe, que implemente a respectiva interface do atuador a ser substituído, que utilize como tipagem as mensagens definidas no Piloto para o Controle. Feito isso, esta nova implementação pode ser passada por parâmetro na definição de algum elemento (bola, campo, capturador de imagens e robôs) e o Controle utilizará esta implementação quando o fluxo do jogo solicitar a atuação do definido.

Figura 17 – Representação conceitual do processo de substituição de um componente por outro dentro da arquitetura. Uso do componente A.



Fonte: O Autor.

Figura 18 – Representação conceitual do processo de substituição de um componente por outro dentro da arquitetura. Uso do componente B.



Fonte: O Autor.

4.2 Uso da Arquitetura

Juntamente com o Controle é disponibilizado um Piloto que implementa alguns atuadores, denominado *SimpleVSSS*. Essa é a aplicação prática da arquitetura, utilizando os tipos específicos explanados na Quadro 4.1.

Quadro 4.1 – Relação entre tipos genéricos do Controle e sua implementação no *SimpleVSSS* e suas dependências.

Tipo Genérico	**Tipo Específico**	**Dependência**
Img	Bitmap	System.Drawing
VtoERobo	InfoVtoERobo	RobotSoccerLib
EtoCRobo	InfoEtoCRobo	RobotSoccerLib
VtoEBola	InfoVtoEBola	RobotSoccerLib
VtoECampo	InfoVtoECampo	RobotSoccerLib
PlaceToDraw	PictureBox	System.Windows.Forms

Fonte: O Autor.

Os atuadores foram criados seguindo a tipagem definida na instanciação do Controle interno deste Piloto. Esta instancia foi definida usando os tipos da Quadro 4.1 e a relação de como fica definido os tipos das interfaces implementadas pelos atuadores seguem a Quadro 4.2.

Para utilizar a *SimpleVSSS* em sua condição padrão, basta adicionar a DLL a um projeto de aplicação C# juntamente das bibliotecas EMGU.CV v3.4.1.2976 e ZedGraph v5.1.7. Estas bibliotecas externas podem ser instaladas facilmente através do gerenciador de pacotes NuGet do Ambiente de Desenvolvimento Integrado (IDE) Visual Studio (versão utilizada para desenvolvimento foi a Microsoft Visual Studio Community 2015 Version 14.0.25431.01 Update 3).

Quadro 4.2 – Relação entre os atuadores, suas interfaces implementadas e tipagem específica usado na *SimpleVSSS*.

Atuador	**Interface**	**Tipos Específicos**
CapturaVideo	IVideoCaptura	Bitmap, PictureBox
ComunicadorBluetooth	IComunicacao	InfoEtoCRobo
EstrategiaBasica	IEstrategia	InfoVtoERobo, InfoEtoCRobo, InfoVtoEBola, InfoVtoECampo
VisaoBola	IVisao	Bitmap, InfoVtoEBola, PictureBox
VisaoCampo	IVisao	Bitmap, InfoVtoECampo, PictureBox
VisaoRobo	IVisao	Bitmap, InfoVtoERobo, PictureBox

Fonte: O Autor.

4.2.1 Livre Customização do *SimpleVSSS*

Por utilizar a arquitetura proposta, o *SimpleVSSS* permite que seus atuadores sejam substituídos por outros desenvolvidos externamente. Para isso, basta que tal atuador implemente a interface seguindo o relacionado no Quadro 4.2 e utilizem as funções do Piloto em questão, para realizar a definição dos atuadores customizados. Essa possibilidade de simples modificação ressalta os benefícios da arquitetura no baixo acoplamento, manutenibilidade e escalabilidade.

4.3 Implementação do *SimpleVSSS*

As decisões de implementação da *SimpleVSSS* são relatadas abaixo. Em cada atuador as seguintes estratégias de implementação foram utilizadas.

Seguindo as premissas do padrão proposto, esta seção descreve, levando em conta as dependências, a implementação de uma solução que siga as práticas propostas. Usando-a como exemplo, pode-se desenvolver uma solução própria, que terá as mesmas qualidades e limitações.

4.3.1 Visão Computacional

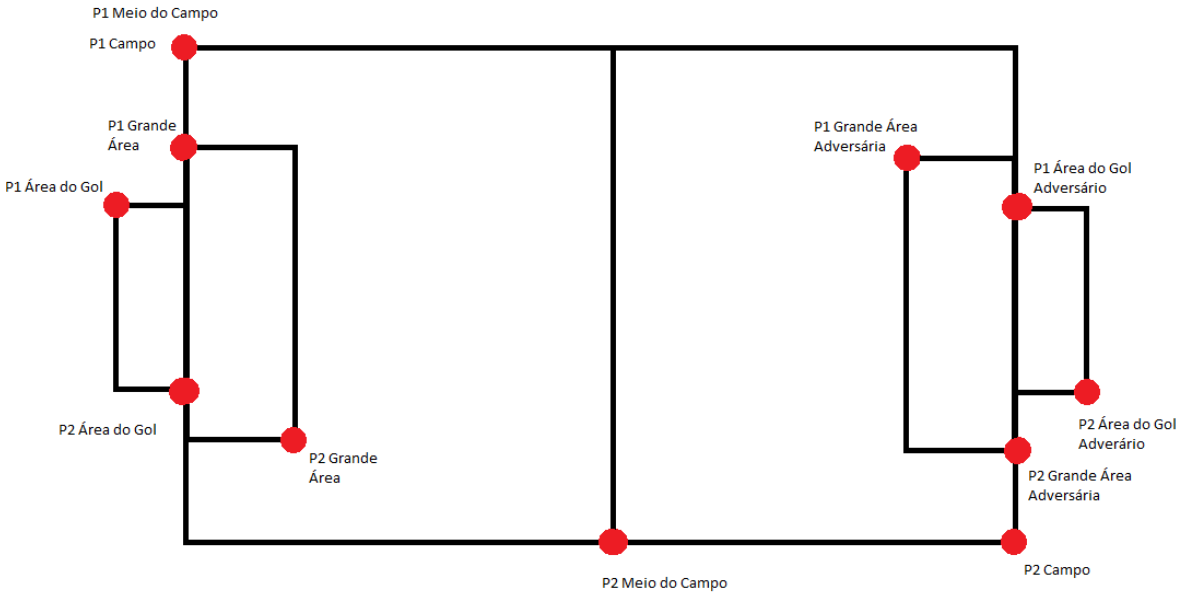
Nesta parte, a entrada da visão computacional é uma imagem capturada do mundo real. Esta então deve passar pelas seguintes etapas até gerar a saída para a etapa seguinte:

1. Recebimento da Imagem;
2. Aplicação de Filtro *Smooth Gaussian* e *Median Blur* para redução de ruídos;
3. Conversão RGB para HSV;
4. Identificação das *ROI* para cada cor de interesse (time e individuais);
5. Calcular pontos chave do jogo.

Para definir os retângulos que representam o campo e suas divisões, é adotada a seguinte abordagem: o usuário através da interface de usuário (UI), seleciona os pontos capazes de identificar os retângulos, como pode ser observado na Figura 19, pois tendo dois pontos de

uma diagonal é possível desenhar um retângulo. Como o campo não se move em relação a câmera, que é fixada a 2 metros acima do centro deste, é possível utilizar esta estratégia.

Figura 19 – Seleção de pontos fixos para definir campo.



Fonte: O Autor.

Ainda na UI o usuário seleciona a cor do time e cores individuais dos robôs como Figura 20, sendo que a cor do time precisa ser selecionada apenas uma vez, já que é compartilhada, e as demais cores, uma para cada robô.

Figura 20 – Seleção das cores dos robôs.

Fonte: O Autor.

Então, após os pontos fixos e as cores dos robôs serem definidos é possível dar início a partida. A partir daí a estratégia utiliza os pontos fixos e o que é reconhecido pelo processamento de imagem para sua tomada de decisões.

A detecção da bola consiste em selecionar a cor da bola e onde aquela cor é identificada seu centroide é definido como a posição efetiva desta. Nesta parte, os dados que transitarão para a etapa seguinte de estratégia, devem estar representadas nos objetos que, pelo Controle, são definidos por tipos genéricos *VtoERobo*, *VtoECampo* e *VtoEBola* onde no

SimpleVSSS foram especificados, respectivamente, por *InfoVtoERobo*, *InfoVtoECampo* e *InfoVtoEBola*.

4.3.1.1 Desafios na Visão Computacional

Por ser a etapa detectora do meio, a visão computacional conta com diversas interferências e variações do mundo real. Por exemplo, a diferença da iluminação em diferentes partes do campo, pode ser tratada utilizando apenas luz artificial, bloqueando a entrada de luz solar no ambiente do campo, e o trânsito de pessoas pelas proximidades durante a execução dos teste.

Para identificar as cores tendo uma tolerância de variação na iluminação, é definido um intervalo de valor em que a componente *"Value"* da cor HSV escolhida, fazendo com que o robô possa transitar no campo, e embora sua cor modifique da inicial, devido a diferentes incidências de iluminação na extensão do campo, o algoritmo continua a identificar aquela variação de tonalidade.

Para a homogeneização da imagem recém processada, onde é possível que haja falsos positivos que mostre erroneamente pontos de presença de determinada cor, filtros são utilizados para reduzir estes ruídos. Da própria biblioteca *EmguCV*, são utilizados sobre a imagem, na ordem apresentada, os filtros: *Smooth Gaussian*, *Erode* e *Dilate*. Com esta combinação é possível ter uma imagem para a detecção de *blobs* menos ruidosa.

Um cuidado que deve-se tomar ao capturar a imagem, é sempre certificar-se de que ela deixa de ser referenciada após seu envio para processamento. Caso contrário, os quadros se acumularão na memória, consumindo espaço, resultando em falha crítica do programa.

4.3.2 Estratégia

A entrada desta etapa é um objeto contendo os posicionamentos dos elementos de jogo, que é constantemente atualizado pela etapa da visão computacional, ou seja, é uma região crítica de dados.

A partir dos pontos detectados pela etapa anterior, como previamente especificado, há a tomada de decisões estáticas através da utilização da abordagem de máquinas de estados finitos. O algoritmo proposto para a solução é uma adaptação do disponibilizado pela FIRA em seu simulador *SimuroSot*.

O resultado então liberado por esta etapa é a movimentação para cada roda de cada robô. Nesta parte, os dados que transitarão para a etapa seguinte de comunicação sem fio devem estar representadas nos objetos que, pelo Controle, são definidos pelo tipo genérico *EtoCRobo*, onde no *SimpleVSSS* é respectivamente especificado por *InfoEtoCRobo*.

4.3.2.1 Estratégia Implementada

Na estratégia, é necessário definir diversos artifícios matemáticos, com o intuito de extrair informação dos pontos obtidos na etapa de visão. Estes pontos são interpretados num plano discreto de coordenadas x e y, de valores inteiros, cuja unidade de variação é 1.

Para encontrar a angulação do robô em relação ao destino, primeiramente é definido o centro do robô (ponto médio entre a cor individual e cor de time). Para calcular o ângulo entre as semi-retas que podem ser traçadas, do ponto em comum, no centro do robô, seguindo para o centro da posição individual, e o destino desejado, o ponto central é definido como a origem (0,0), e os demais pontos citados são deslocados proporcionalmente, tendo suas coordenadas subtraídas das do centro. Então estes pontos são definidos como vetores, para, utilizando a equação de produto interno (Equação 4.1), obter o cosseno do ângulo entre eles.

$$\cos\theta = \frac{x_1 \cdot x_2 + y_1 \cdot y_2}{\sqrt{{x_1}^2 + {y_1}^2} \cdot \sqrt{{x_2}^2 + {y_2}^2}} \tag{4.1}$$

O ângulo obtido pelo produto interno não faz distinção entre direita e esquerda, pois o cosseno varia de -1 (180 graus) até 1 (0 graus). Para definir o lado do ponto em relação a esquerda ou direita do robô, é feita a utilização da verificação pela Equação 4.2, que se $d < 0$ então o ponto de destino está a direita da linha do robô, se $d > 0$ então está a esquerda. Se $d = 0$ está concorrente a linha.

$$d = (x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1) \tag{4.2}$$

Além disso é calculada a distância do centro do robô a seu destino, que é dado pela Equação 4.3.

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.3}$$

Utilizando estas três equações, é possível já executar a movimentação básica de um robô através de tomadas simples de decisões.

4.3.3 Comunicação Sem Fio

A comunicação sem fio tem um problema inerente, o acoplamento entre a eletrônica e a comunicação sem fio, já que a tecnologia de transmissão e recepção precisa ser compatível.

A comunicação sem fio tem a função de estabelecer a conexão com os robôs e sempre que houver uma mudança na velocidade de alguma roda de algum robô, informada pela estratégia, deve informar o robô correto no menor tempo possível. Entre a comunicação e a estratégia há uma segunda região crítica, já que a estratégia escreve e a comunicação lê.

4.3.3.1 Protocolo

Sabendo que as mensagens interpretadas pelos robôs consistem na informação da velocidade de suas rodas, o seguinte protocolo foi desenvolvido tendo, em vista que as rodas são independentes tem uma velocidade mínima (0) e máxima (255) e podem girar em duas direções.

O protocolo consiste de uma *String* composta pelas seguintes partes e é ilustrado pela Figura 21:

1. NNN: Número inteiro de 0 a 255, define a intensidade de giro da roda esquerda;
2. F ou T: Frente ou Trás, sentido que a roda esquerda girará.
3. NNN: Número inteiro de 0 a 255, define a intensidade de giro da roda direita;
4. F ou T: Frente ou Trás, sentido que a roda direita girará.

Figura 21 – Protocolo de comunicação.

Fonte: O Autor.

Exemplo de mensagem válida: ”100F100T”, o robô que receber a mensagem ligará sua roda esquerda no sentido para frente, com a intensidade 100, e a roda direita no sentido contrário ao da roda direita, com intensidade de 100, numa escala de -255 a +255.

4.3.3.2 Problemas e Tratativas

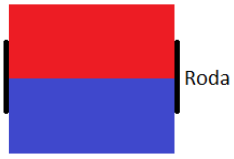
Um problema durante a comunicação com o robô é que caso sejam enviadas mensagens na velocidade do processamento do computador, há um mal funcionamento, onde, por escrever muitas vezes na saída para as rodas, estas não se movimentam da maneira desejada, reduzindo sua velocidade e até mesmo parando.

Para solucionar este problema, dentro do componente de comunicação desenvolvido, foi realizada uma tratativa de modo que as mensagens passam por uma verificação em que, caso a mensagem atual seja igual a mensagem anterior, ela não é enviada, pois esta segunda mensagem igual é desnecessária. Com esta tratativa realizada, também é reduzido o tráfego na comunicação sem fio, elevando assim a vida útil da bateria que alimenta o robô.

4.3.4 Construção, Eletrônica e Mecânica dos Robôs

A superfície visível do robô foi dividida em duas partes retangulares, uma com a cor do time e a outra com a individual, orientadas de maneira que o segmento de reta, que pode ser formado ligando o centro dos dois retângulos, forma um angulo reto com o alinhamento das rodas. Isto pode ser observado na Figura 22.

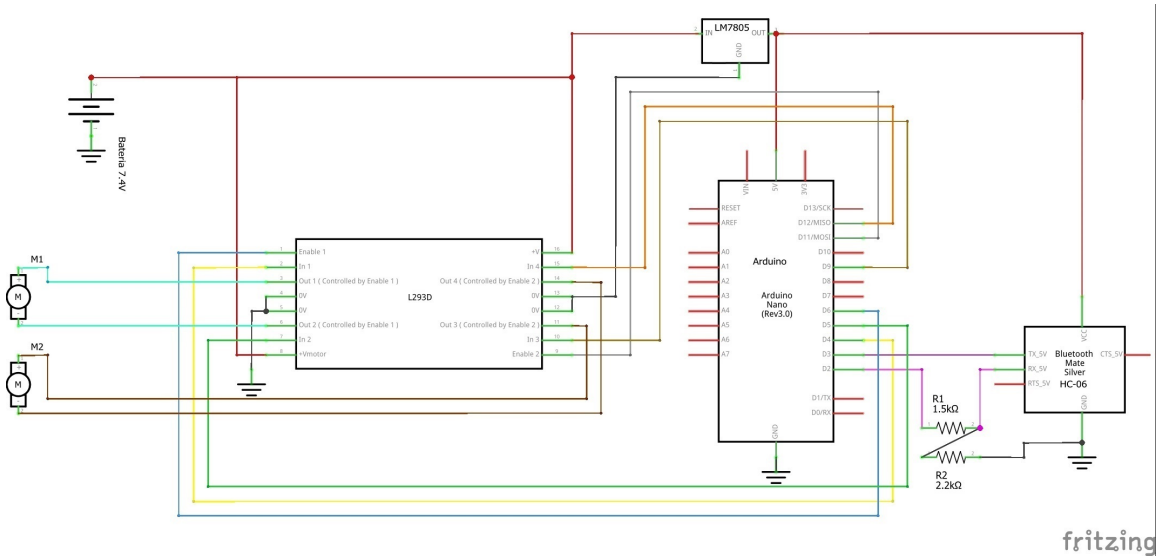
Figura 22 – Esquema de vista do robô por cima.



Fonte: O Autor.

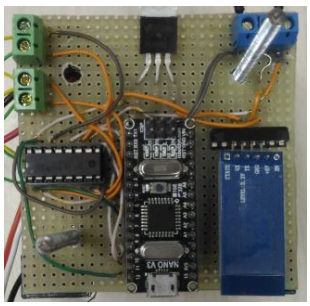
A eletrônica consiste em um receptor *bluetooth* que estabelece conexão entre mestre (computador) e escravo (robô), e aguarda ordens via mensagem protocolada. Nele a mensagem é decifrada e ordens são enviadas a ponte H, que libera a corrente que ativa os motores na ordem e intensidade corretas.

Figura 23 – Desenho do circuito do robô.



Fonte: O Autor.

Figura 24 – Circuito do robô.



Fonte: O Autor.

O código do receptor segue o seguinte procedimento. Recebe 8 caracteres, interpreta e executa a ordem passada, segundo o protocolo, enviando sinais aos motores e então aguarda o recebimento de mais mensagens, reexecutando o processo num *loop* infinito. O desenho do circuito criado pode ser visualizado na Figura 23 e seu estado real dentro de um robô de futebol na Figura 24.

O resultado desta etapa é a efetiva movimentação dos robôs dentro de campo. Isto então ocasiona a modificação do ambiente capturado pela câmera, formando assim um ciclo como demonstrado anteriormente na Figura 15.

5 Experimentos e Resultados

Ubi dubium ibi libertas: Onde há dúvida, há liberdade.
Provérbio Latino

O tempo de execução é um aspecto que deve ser minimizado no sistema que rege o jogo, pois, por lidar com o mundo real, este deve responder aos estímulos recebidos do meio no menor tempo possível. Sabendo disso, os testes realizados visam os tempos de execução (em milissegundos), das etapas separadamente e em conjunto, dentro do padrão de arquitetura proposto, e por meio da análise dos dados como média e desvio padrão, do tempo de execução e das saídas obtidas, quando estas transitam para outra etapa dentro do software identificar particularidades do sistema.

Os experimentos deste trabalho, realizados em ambiente controlado, definido pelo autor, visam a comprovação do funcionamento, tanto da implementação quanto da arquitetura, sendo a primeira utilizada dentro da segunda.

5.1 Testes Individuais

Os testes individuais consistem em testar separadamente os componentes desenvolvidos para a validação da arquitetura. É importante destacar que os testes abaixo visam apenas verificar o comportamento dos algoritmos e, através da análise dos resultados, propor hipóteses que justifiquem este comportamento.

5.1.1 Visão Computacional

Os testes sobre a visão computacional tiveram como intuito a verificação da acurácia da identificação dos pontos chave do jogo, detectando um robô imóvel em relação à câmera. Este teste foi executado num ambiente com iluminação de 1000 lux, sendo o pixel a unidade de medida dos posicionamentos (x e y num plano cartesiano) e a câmera situada a um metro de altura perpendicular a superfície observada.

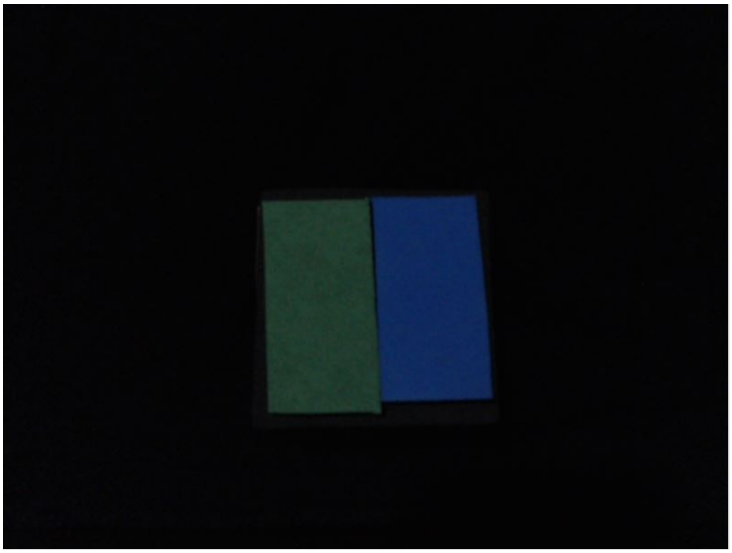
O teste foi realizado processando dez imagens capturadas pela câmera e registrando o tempo de execução entre o início do processamento da primeira imagem e o término da décima, fazendo uma comparação total de tempos, para 100 execuções, e os valores do posicionamento da cor individual do robô e sua cor de time. Uma amostra do ambiente analisado pela câmera pode ser visualizado na Figura 25.

Tabela 1 – Tempos de execução dos testes da visão computacional.

Teste	1	2	3	4	5	6	7	8	9	10
Tempo (ms)	83	78	91	79	82	87	83	100	83	85

Fonte: O Autor.

Figura 25 – Amostra de ambiente processado pela etapa de visão.



Fonte: O Autor.

Tabela 2 – Média e desvio padrão dos tempos de teste da visão computacional.

Média	**Desvio Padrão**
85,1	6,090156

Fonte: O Autor.

Tabela 3 – Média e desvio padrão das posições definidas pela visão computacional.

	Média	**Desvio Padrão**
Cor Individual Posição X	471,9	0,3
Cor Individual Posição Y	324,66	0,551725
Cor Time Posição X	351	0
Cor Time Posição Y	335,1	0,3

Fonte: O Autor.

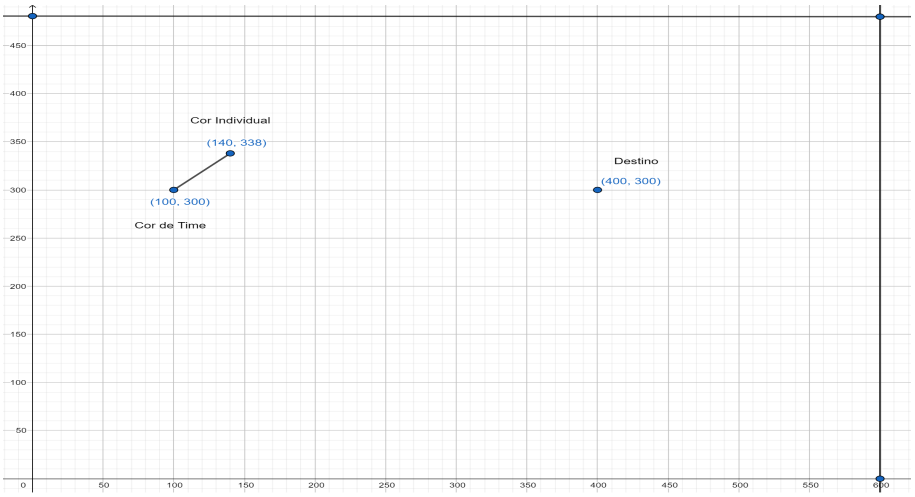
Quanto ao tempo de execução, pode-se afirmar, analisando a Tabela 1 e Tabela 2, que este tem baixa variação, observando seu desvio padrão. Quanto ao posicionamento identificado pela cor individual e de time, é notável pela Tabela 3 que há alguma variação no posicionamento das etiquetas.

5.1.2 Estratégia

Os testes de estratégia consistem no processamento de pontos condizentes com estado de jogo do futebol de robôs para um robô, isto é ele recebe os posicionamentos de suas etiquetas, da bola e do campo, e deve fornecer as velocidades das rodas do robô. Esta etapa ocorre sem a interação direta com o meio, por isso sua execução pode ser manipulada livremente, fornecendo diversos pontos definidos, sem a interferência do ambiente. A estratégia utilizada para teste tem como entrada um mesmo conjunto de pontos, para posicionamento do robô e seu destino. Sua execução terá o tempo e saídas monitorados e analisados para possíveis conclusões.

O cenário de teste pode ser observado na Figura 26, onde estão presentes o que esta estratégia leva em consideração, a localização do robô e o destino que deseja alcançar.

Figura 26 – Cenário de teste individual para estratégia.



Fonte: O Autor.

Diferentemente das demais etapas, o tempo de execução foi medido entre o início da primeira execução e o fim da centésima, ou seja, foram realizados cem execuções, totalizando 1000. As saídas desta etapa tiveram todos os valores iguais, pois para a mesma entrada terá sempre a mesma saída.

Tabela 4 – Tempos de teste de estratégia.

Teste	1	2	3	4	5	6	7	8	9	10
Tempo(ms)	225	210	203	213	218	230	208	213	224	201

Fonte: O Autor.

Tabela 5 – Média e desvio padrão do tempo de teste da estratégia.

Média	**Desvio Padrão**
214,5	9,135097

Fonte: O Autor.

Analisando os tempos e desvio padrão observáveis na Tabela 4 e Tabela 5 podemos afirmar que a etapa da estratégia é consistente e, se utilizada a implementação apresentada, não será um gargalo para o sistema do esporte eletrônico. Como esta etapa é composta apenas de operações independentes de entrada ou saídas externas (câmera ou comunicação sem fio), ela é executada mais rapidamente em relação às outras etapas.

5.1.3 Comunicação Sem Fio

Os testes de comunicação sem fio consistem na utilização do robô já configurado com o protocolo proposto e respondendo às mensagens recebidas. Os testes foram realizados tendo o transmissor *bluetooth* e o robô numa distância entre um e dois metros.

O teste foi realizado enviando dez mensagens distintas para o robô protótipo, utilizando a implementação de comunicação *bluetooth*. O tempo de execução entre o início do envio da primeira mensagem e o término da décima, fornece o tempo de execução medido. Este teste foi executado dez vezes, totalizando 100 execuções, tendo como resultado o exibido na Tabela 6.

Tabela 6 – Tempos de teste de comunicação.

Teste	1	2	3	4	5	6	7	8	9	10
Tempo (ms)	43	51	10	46	12	80	50	46	48	37

Fonte: O Autor.

Tabela 7 – Média e desvio padrão dos tempos de testes da comunicação.

Média	**Desvio Padrão**
42,3	18,989734

Fonte: O Autor.

Analisando o desvio padrão da Tabela 7, pode-se afirmar que estes tempos apresentam uma certa variabilidade. Esta variação se dá, muito provavelmente, devido a instabilidade das comunicações sem fio, neste caso via *bluetooth*.

5.1.4 Análise dos Testes Individuais

Pela análise dos testes individuais, é possível afirmar que, tratando-se do tempo de execução, o processamento de um *frame* (etapa de visão computacional) dá-se em média em 8,5ms. O trabalho de Martins, Tonidandel e Bianchi (2006) apresenta uma análise sobre processamento de imagem e exibe uma tabela com o tempo médio de execução de 25ms, deixando claro assim que, mesmo as máquinas sendo de especificações diferentes, os tempos obtidos no cenário atual são aceitáveis para a visão computacional de um time de futebol de robôs, pois o tempo do mundo real é o mesmo, e o intuito é agir o mais próximo do tempo real possível. Então, sabendo que o maior gargalo de processamento encontra-se na visão computacional, pode-se afirmar que o padrão de arquitetura é aplicável no contexto do esporte eletrônico.

Destaca-se, ainda na visão computacional, que a posição identificada das etiquetas possui uma pequena variação (vide Tabela 3), isto ocorre, possivelmente, devido a diversos

fatores, como variação na iluminação ambiente, deslocamento imprevisto da câmera, entre outros, porém mesmo com todos os interventores, há uma elevada acurácia na identificação dos elementos de jogo.

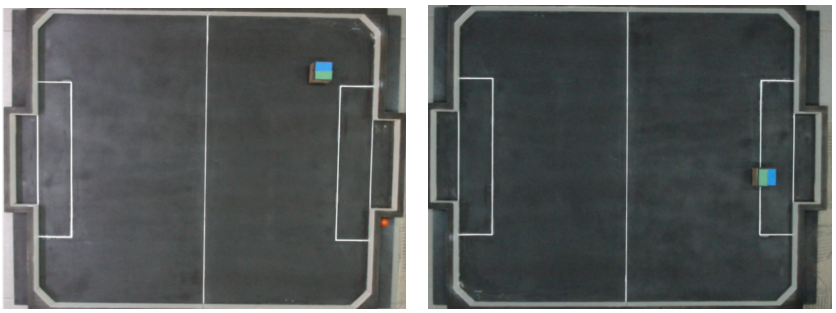
Na etapa de estratégia há uma maior constância no tempo de execução e constância total das saídas, pois não há as variações na entrada, como na visão computacional (subseção 5.1.1), nem a inconstância das saídas, como as transmissões realizadas pela comunicação sem fio (subseção 5.1.3).

5.2 Testes Integrados

Nos testes integrados é avaliado o trajeto do robô, sendo sua implementação feita seguindo do padrão de arquitetura proposto. Dois tipos de testes foram realizados, no primeiro caso, o robô partindo de uma posição inicial desloca-se até um dado ponto, na segunda bateria de testes há a definição de dois pontos que são visitados pelo robô.

A Figura 27 exibe a imagem do ambiente capturada pela câmera e que alimenta o processo do software atuador no jogo.

Figura 27 – Casos iniciais de testes integrados.



(a) Estado inicial dos testes relatado na subseção 5.2.1 e subseção 5.2.2.

(b) Estado inicial do teste relatado na subseção 5.2.3.

Fonte: O Autor.

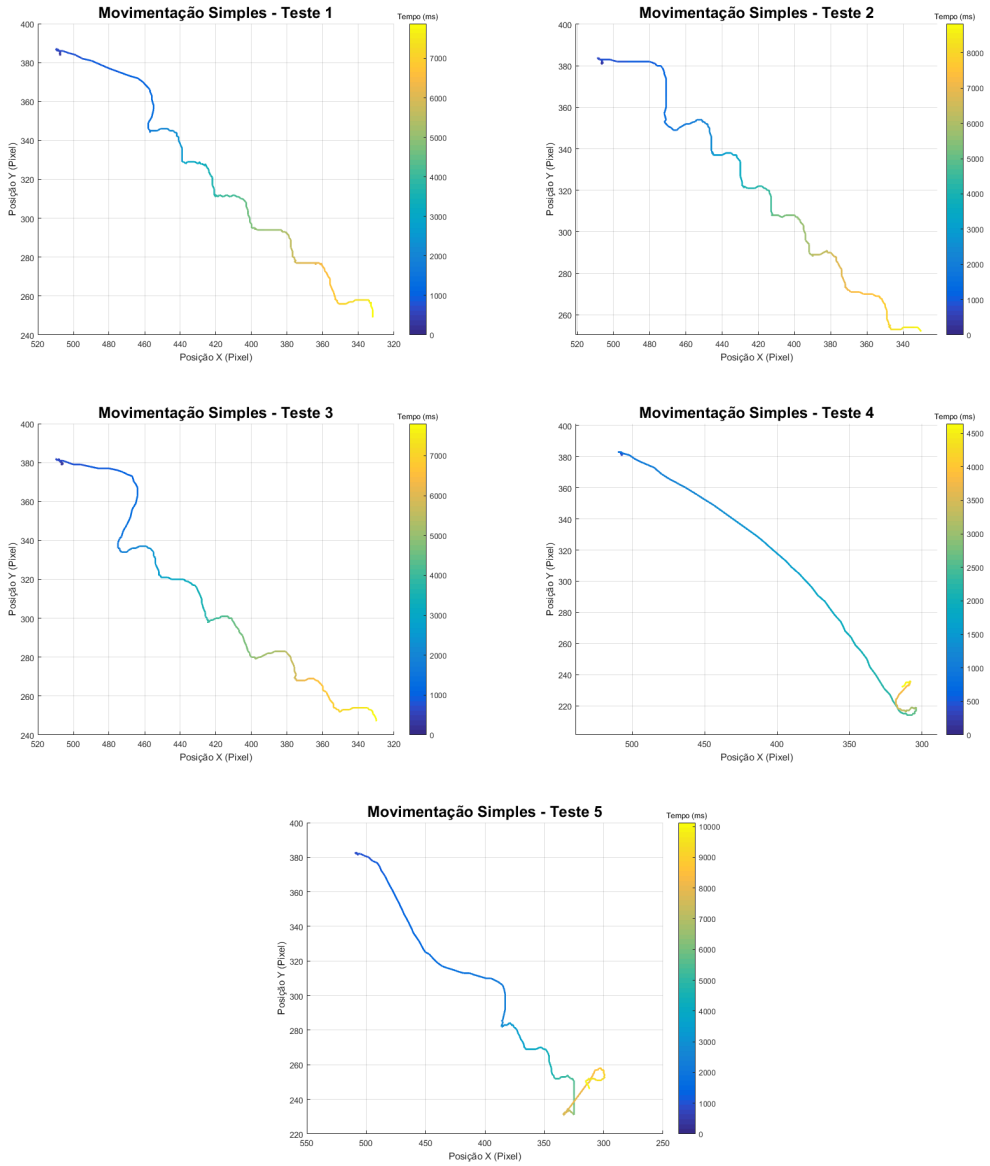
Os gráficos utilizados para a representação da movimentação do robô, nos casos de teste, seguem o seguinte padrão: seu eixo horizontal é decrescente e exibe a posição X na imagem do robô, o eixo vertical mostra a posição Y do robô e a barra lateral com cor representa o tempo em que o robô estava naquela determinada posição.

5.2.1 Robô Vai a Um Ponto

Este teste foi definido da seguinte maneira, o robô é posicionado sempre na mesma posição inicial, (510, 380) e como ponto destino é definido o centro do campo (320 ,240). A estratégia utilizada tem como uma tolerância estabelecida de 15 pixeis para a distância do robô ao ponto destino.

Os gráficos retratados na Figura 28 apresentam os resultados coletados empiricamente pela execução do caso de teste supracitado.

Figura 28 – Testes de movimentação do robô do ponto inicial ao ponto definido.



Fonte: O Autor

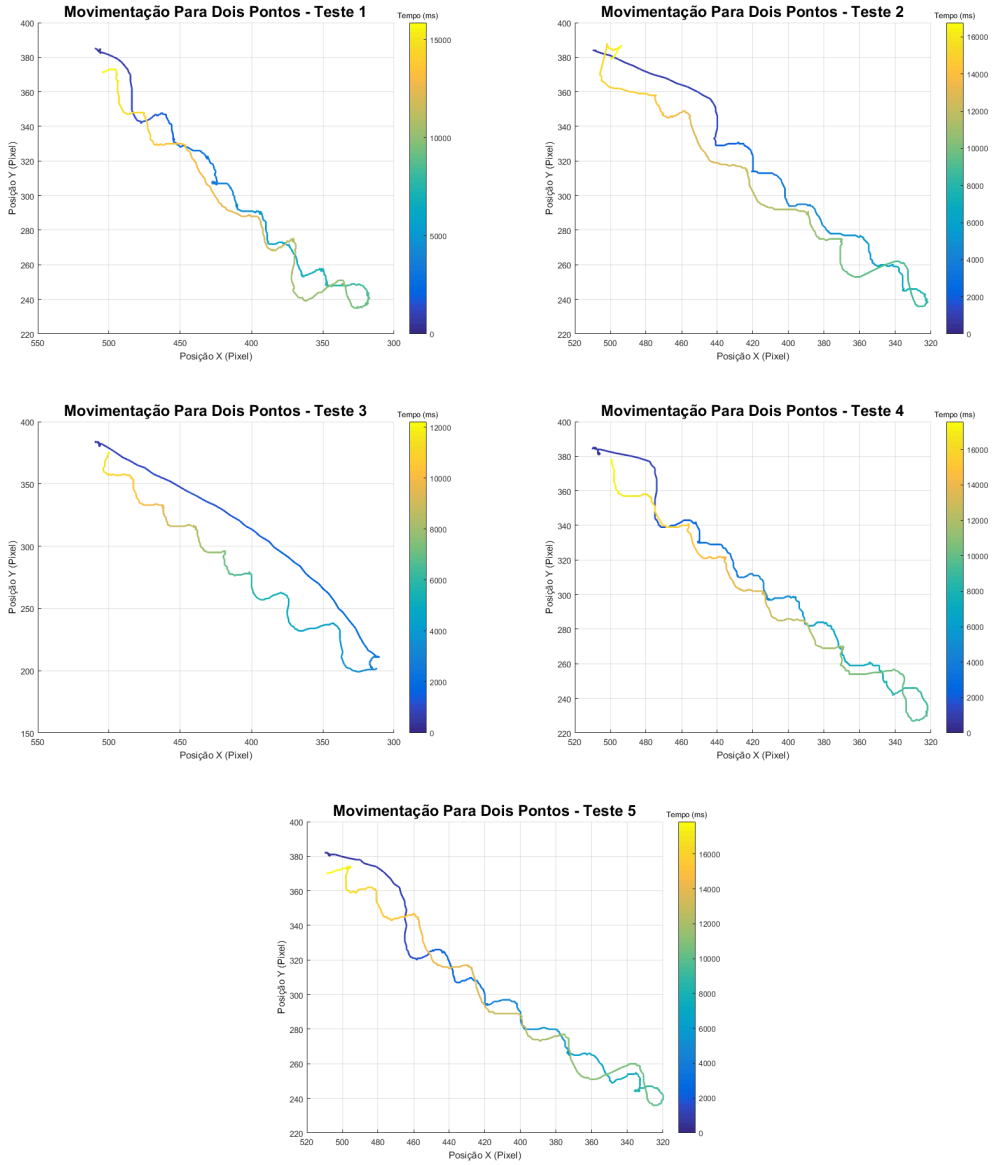
Analisando a Figura 28, pode-se notar as curvas realizadas pelo robô e seu ajuste precário para chegar ao ponto determinado. Isto se deve a estratégia utilizada, composta apenas por três tipos de movimento, giro para um sentido, giro para o sentido contrário e seguir em frente, sempre comparando o ângulo entre a orientação do robô com seu destino e assim determinando qual das ações tomar. Apesar de seus desvios o robô atingiu seu objetivo, dentro do limite predeterminado em todos os casos de teste.

5.2.2 Robô Visita Dois Pontos

Este teste foi definido da seguinte maneira, o robô é posicionado sempre na mesma posição inicial, (510, 380) e como ponto destino é definido o centro do campo (320 ,240) e seu ponto de partida. A estratégia utilizada tem como uma tolerância estabelecida de 15 pixeis para a distância do robô ao ponto destino. Os gráficos retratados na Figura 29 apresentam os

resultados coletados empiricamente pela execução do caso de teste supracitado.

Figura 29 – Testes de movimentação do robô do ponto inicial a um ponto e voltando ao início.



Fonte: O Autor

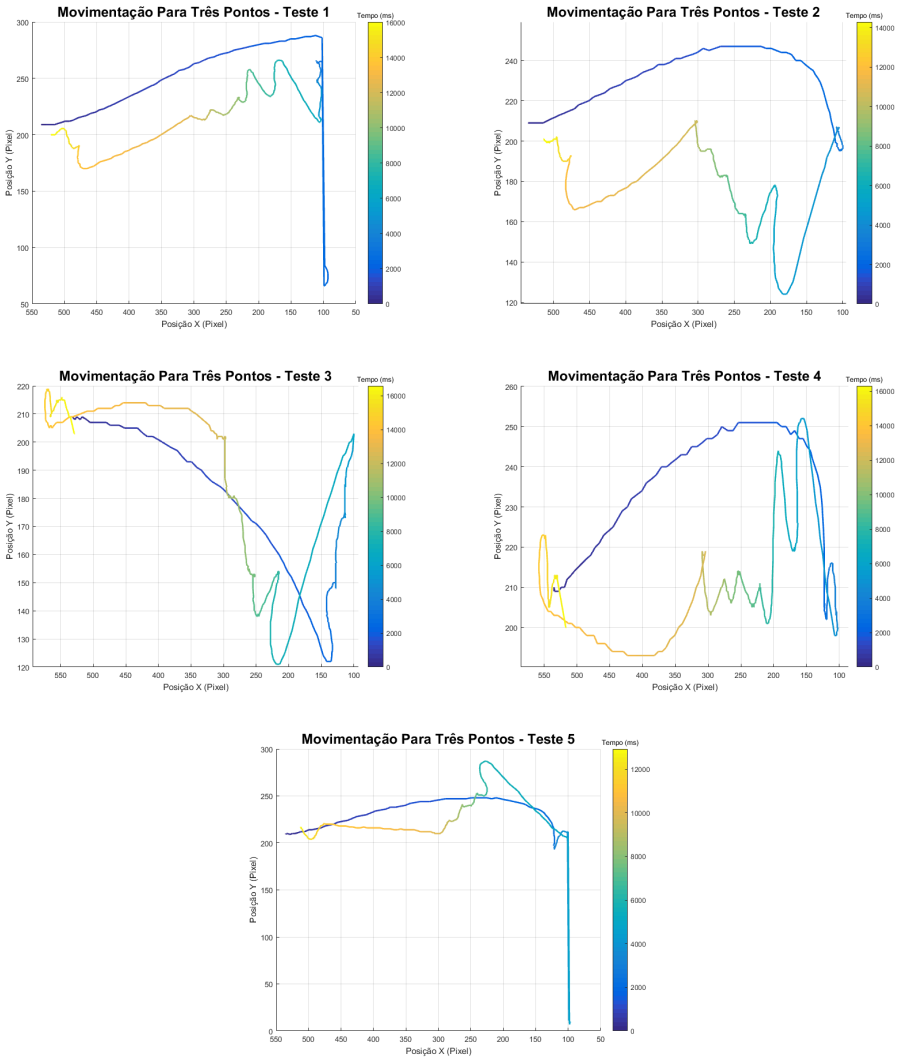
Analisando a Figura 29, pode-se afirmar que, assim como o primeiro caso integrado, o robô apresenta a movimentação com um movimento de “ziguezague” ocasionado pelo uso da estratégia implementada e pela não realização de um controle fino a nível de eletrônica e mecânica. Porém, cumpre sua premissa que é a movimentação do robô para os pontos determinados.

5.2.3 Robô Visita Três Pontos

O robô é posicionado sempre na mesma posição inicial (520,210) e então deveria visitar, na ordem, o início da área do gol do lado oposto de partida (100,210), o centro do campo (310,210) e, finalmente deve voltar a posição inicial (520,210). O diferencial desta bateria de

testes é que o robô percorre os dois lados do campo, diferente dos testes anteriores que mantiveram-se no lado direito do campo (selecionado arbitrariamente).

Figura 30 – Testes de movimentação do robô do ponto inicial ao ponto definido do outro lado do campo.



Fonte: O Autor

A execução desta bateria de testes, ilustrada pela Figura 30, permite analisar a movimentação do robô por toda a extensão do campo e a movimentação horizontal do robô, diferente dos testes anteriores que cuidaram da movimentação diagonal do robô. O robô pôde visitar todos os pontos definidos ao longo do campo.

5.3 Análise Geral dos Testes

Os testes individuais e integrados possibilitaram a comprovação da eficácia do padrão arquitetural, no que diz respeito a sua funcionalidade e validade para a utilização no futebol de robôs, já que a execução integrada foi efetiva no deslocamento do robô para os destinos selecionados.

O tempo para o robô alcançar os pontos definidos e a trajetória tortuosa que este segue, é devido à implementação feita na parte da estratégia, o que felizmente pode ser substituído sem impactar as outras etapas, graças ao padrão arquitetural utilizado. Outro ponto onde este problema pode ser solucionado é na eletrônica e mecânica, pela utilização de encoders para o controle refinado

Na terceira bateria de testes proposta, nos casos um e cinco, há um comportamento que destaca-se, onde a posição do robô desloca-se muito rapidamente a um ponto muito fora da curva. Este comportamento ocorre, possivelmente, devido a interferências na etapa de visão, porém o desvio e tão pontual que não chega a prejudicar o comportamento do sistema em geral. Este teste também permite afirmar que a implementação dentro da arquitetura possibilita a movimentação ao longo da extensão horizontal do campo.

Por meio do padrão dos gráficos apresentados na subseção 5.2.1 e subseção 5.2.2, é possível visualizar o posicionamento do robô no tempo, informação importante quando se trata do esporte eletrônico, já que quem vence é aquele que marca mais pontos primeiro. Por meio deste padrão de gráfico é possível comparar o tempo gasto para a movimentação do robô por aquele percurso.

6 Conclusão e Trabalhos Futuros

"Não me venham com conclusões! A única conclusão é morrer".
Fernando Pessoa

Neste capítulo apresenta-se o que foi percebido sobre o padrão arquitetural proposto, a implementação simplista que o seguiu e os testes realizados, em seguida são ressaltados os possíveis trabalhos futuros identificados.

6.1 Conclusão

Pontos de interesses são distinguíveis dentro desta obra, em diversas partes. Sua contribuição com um padrão arquitetural regente do futebol de robôs, a informação obtida através da análise dos testes executados sobre implementação e quanto a contribuição do trabalho como base de conhecimento. Tais pontos a saber são:

- Neste trabalho é apresentado um padrão arquitetural para o futebol de robôs. Este padrão regeu o processo do esporte, isolou as etapas do processo, permitiu a substituição dos componentes das etapas e ainda a utilização de implementações distintas simultaneamente, esta mesma substituição também contribui na escalabilidade do software. Pelo uso da tipagem genérica, remove ainda a dependência de tipos específicos, permitindo que haja a customização das informações que transitam entre os atuadores;
- O protocolo de comunicação provou-se eficaz para ser utilizado na comunicação entre o software e o robô de futebol, usando a tecnologia de comunicação bluetooth;
- Os testes realizados confirmam que a arquitetura é promissora e pode ser utilizada, pois, os testes de tempo realizados são indicativos que permitem comandar um time de robôs de maneira que os mesmos atuem em jogo efetivamente, para a catergoria *Very Small Size*, onde cada time possui três robôs;
- No meio acadêmico, os trabalhos sobre o futebol de robôs abordam, muitas vezes, apenas uma área do esporte (ou visão computacional ou estratégia ou eletrônica, etc), diferente deste, que além de revisar brevemente a bibliografia existente destas áreas, dá enfoque na interação entre elas, mostrando o esporte como um processo iterativo, composto por componentes de responsabilidades distintas que se comunicam por meio de mensagens estabelecidas prévia e arbitrariamente.

Com a execução deste trabalho houve a produção de uma infraestrutura capaz de suportar um time completo de futebol de robôs da categoria *Very Small Size* no Centro Federal de Educação Tecnológica Campus Timóteo, provendo um ambiente favorável para o desenvolvimento de outras pesquisas sobre o futebol de robôs.

6.2 Trabalhos Futuros

Esta seção destina-se a apresentar os pontos de melhoria identificados pelo autor. Dada a multidisciplinaridade abordada neste trabalho, cobrir todos os pontos de melhoria possíveis é inviável. O ressaltado pelo autor nesta etapa a saber é que:

- O primeiro problema evidenciado aqui é que atualmente todos os *frames* são processados e geram saídas para o robô. Na atual abordagem, como a execução de todas as etapas pode ter um tempo de execução total maior que o da aquisição da imagem, o sistema pode trabalhar com informações defasadas da realidade. Uma outra solução que forneça sempre uma imagem mais recente do meio é um ponto de melhoria;
- A arquitetura foi desenvolvida abordando os pontos para a confecção de um time conforme os critérios necessários segundo a visão do autor, novas necessidades como por exemplo, tratativas de exceções identificadas e detecção e utilização de dados sobre o time adversário, podem ser adicionadas, mas sempre tendo em vista os preceitos de baixo acoplamento e simples escalabilidade proporcionados pela arquitetura;
- A implementação utilizada neste trabalho é uma forma simplista que soluciona o problema para os casos de teste utilizados. Estes devem ser substituídos por implementações que contemplem as etapas de um jogo real da categoria *Very Small Size*, comunicando com três robôs. Graças à arquitetura apresentada neste trabalho, atingir este objetivo bastará na implementação isolada dos algoritmos responsáveis pelo novo comportamento;
- A movimentação do robô (definida na estratégia) mostrou nos testes empíricos que cumpre o proposto, porém de maneira primária e, provavelmente, não aplicável num jogo real de futebol de robôs. Uma modelagem matemática para a movimentação das rodas do robô deve ser elaborada e utilizada na etapa de estratégia, para que resulte numa movimentação mais satisfatória;
- Finalmente, é evidenciável as possíveis melhorias na eletrônica e mecânica do robô, como por exemplo a utilização de *encoders* para a garantia da movimentação balanceada das rodas do robô.

Referências

ALBUQUERQUE, M. P.; ALBUQUERQUE, M. P. Processamento de Imagens. *Information Retrieval*, 2000. Citado na página 24.

ALVES, D. R. Avaliação dos Modelos de Cores RGB e HSV na segmentação de Curvas de Nível em Cartas Topográficas Coloridas. 2010. Citado nas páginas 24 e 25.

Arduino. *Arduino Nano*. 2018. Disponível em: <https://store.arduino.cc/usa/arduino-nano>. Citado na página 34.

BARROS, R. da S. et al. Identificação De Objetos Do Futebol De Robôs Utilizando Algoritmo De Descrição De Pontos Chave. *Colloquium Exactarum*, v. 7, n. 2, p. 101–118, 2015. ISSN 21788332. Disponível em: <http://revistas.unoeste.br/revistas/ojs/index.php/ce/article/view/1427/1463>. Citado nas páginas 14, 19 e 25.

Found Golf Balls. *Orange Mix used golf balls - Foundgolfballs.com*. 2018. Disponível em: <https://www.foundgolfballs.com/products/orange-mix>. Citado na página 21.

GANESAN, P.; RAJINI, V. Assessment of satellite image segmentation in RGB and HSV color space using image quality measures. *2014 International Conference on Advances in Electrical Engineering (ICAEE)*, p. 1–5, 2014. Disponível em: <http://ieeexplore.ieee.org/document/6838441/>. Citado nas páginas 25 e 26.

GANESAN, P. et al. HSV Color Space Based Segmentation of Region of Interest in Satellite Images. p. 101–105, 2014. Citado na página 25.

GUARNIZO, J. G.; MELLADO, M. Robot Soccer Strategy Based on Hierarchical Finite State Machine to Centralized Architectures. *IEEE Latin America Transactions*, v. 14, n. 8, p. 3586–3596, 2016. ISSN 1548-0992 VO - 14. Citado nas páginas 19, 27 e 28.

GURZONI, J. A. et al. On the construction of a RoboCup small size league team. *Journal of the Brazilian Computer Society*, v. 17, n. 1, p. 69–82, 2011. ISSN 01046500. Citado nas páginas 8, 14, 15, 16, 18 e 28.

IEEE. Rules for the IEE Very Small Competition. v. 2008, n. d, p. 1–14, 2008. Disponível em: <http://www.cbrobotica.org/wp-content/uploads/2014/03/VerySmall2008_en.pdf>. Citado nas páginas 14, 18, 19, 20 e 21.

JUSOH, S. B.; OMAR, K. The Mechanical an nd Circuit Design of Robot Soccer : A Study. n. June, p. 119–124, 2011. Citado nas páginas 14, 19, 30 e 32.

KITANO, H. et al. Robocup: The robot world cup initiative. In: *Proceedings of the First International Conference on Autonomous Agents*. New York, NY, USA: ACM, 1997. (AGENTS '97), p. 340–347. ISBN 0-89791-877-0. Disponível em: <http://doi.acm.org/10.1145/267658.267738>. Citado na página 14.

MARTINS, M. F.; TONIDANDEL, F.; BIANCHI, R. A. a. Reconhecimento de Objetos em Tempo Real para Futebol de Robôs. p. 173–182, 2006. Citado nas páginas 19, 23, 24 e 50.

PAUTASSO, C.; WILDE, E. Why is the Web Loosely Coupled ? A Multi-Faceted Metric for Service Design. p. 911–920, 2009. Citado na página 15.

POLOLU. *Pololu - Micro Metal Gearmotors*. 2018. Disponível em: <https://www.pololu.com/category/60/micro-metal-gearmotors>. Citado na página 34.

Portal Vida de Silício. *Módulo Bluetooth HC-05 e HC-06 - Arduino | Portal Vida de Silício*. 2018. Disponível em: <https://portal.vidadesilicio.com.br/modulo-bluetooth-hc-05-e-hc-06/>. Citado na página 34.

RABELO, R. A.; MACEDO, H. T.; FREIRE, E. O. The SimuroSot Strategy Development Kit : a High- Level Approach to Robot Soccer Coding. *IEEE Latin America Transactions*, v. 16, n. 2, p. 686–693, 2018. Citado na página 28.

ROLL-HANSEN, N. *Why the distinction between basic (theoretical) and applied (practical) research is important in the politics of science*. [S.l.], 2009. 1–30 p. Citado na página 31.

SAIRAM, K. V.; GUNASEKARAN, N.; Rama Reddy, S. Bluetooth in wireless communication. *IEEE Communications Magazine*, v. 40, n. 6, p. 90–96, 2002. ISSN 01636804. Citado na página 29.

SHIM, H.-S. et al. Design of action level in a hybrid control structure for vision based soccer robot system. *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, v. 3, p. 1406–1411, 1999. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=811676>. Citado na página 14.

SILVA, W. C. et al. USPDroidsSS robot soccer simulator for Very Small Size Category. *Proceedings - 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting, LARS 2010*, p. 138–143, 2010. Citado nas páginas 15 e 16.

SIRLab. *SIRLab: VSS*. 2018. Disponível em: <http://sirlab.github.io/vss.html>. Nenhuma citação no texto.

THAT, M. T. T.; SADOU, S.; OQUENDO, F. Using architectural patterns to define architectural decisions. *Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA 2012*, p. 196–200, 2012. Citado nas páginas 14, 15 e 31.

WIKI, E. *Emgu CV: OpenCV in .NET (C# VB, C++ and more)*. 2018. Disponível em: <http://www.emgu.com/wiki/index.php/Main_Page>. Citado na página 23.