RoboCIn - Equipe de Robótica do Centro de Informática

André Luís Damázio de Sales Júnior[1], Caio Carvalho de Abreu e Lima[1], Carlos Henrique Caloete Pena[1], Cristiano Santos de Oliveira[1], David Riff de França Tenório[1], Esdras Barbosa Lima da Silva Júnior[1], Gabriel Marques Bandeira[1], Heitor Rapela Medeiros[1], João Gabriel Machado da Silva[1], João Lucas Oliveira Canhoto[1], José Nilton de Oliveira Lima Júnior[1], Juliana do nascimento Damurie da Silva[1], Lucas Henrique Cavalcanti Santos[1], Lucas Oliveira Maggi[1], Marvson Allan Pontes de Assis[1], Mateus Gonçalves Machado[1], Raphael Cândido Brito[1], Renato Sousa Bezerra[1], Roberto Costa Fernandes[1], Vinicius Bezerra Araújo da Silva[1], Hansenclever de Franca Bassani[1], Edna Natividade da Silva Barros[1]

***Resumo*—Este *Team Description Paper* (TDP) tem como objetivo descrever o projeto desenvolvido pela equipe RobôCIn, do Centro de Informática da UFPE para participação na competição IEEE *Very Small Size Soccer*. Neste TDP são descritos os principais sistemas desenvolvidos: o sistema de localização por visão computacional, a técnica de planejamento de caminho, baseada em campos potenciais, as tomadas de decisão, bem como a comunicação entre os robôs e o sistema de controle, e o sistemas eletrônico e mecânico do robô projetado para a competição.**

I. INTRODUÇÃO

Para participar da competição da IEEE na categoria *Very Small Size Soccer*, é necessário quatro pilares essenciais: detecção de objetos, planejamento de caminho e decisões, mecânica, eletrônica e comunicação. O primeiro estágio do ciclo de execução do projeto é a detecção de objetos, nesta etapa, a equipe optou por criar um módulo de detecção da bola e dos robôs utilizando visão computacional. Após a etapa de detecção, as informações de posição, direção e velocidade são passadas para o módulo de planejamento. O módulo de planejamento define um caminho a ser seguido pelos robôs, e assim calcula as velocidades a serem enviadas para o módulo de controle do robô (*hardware* embutido no robô). Com as velocidades definidas, o módulo de controle local do robô realiza o controle do robô.

Na próxima seção, é explicado o módulo de localização dos objetos em campo. Na seção seguinte, é detalhado o planejamento de caminho e ações realizado pelo *software* da equipe. Em seguida as seções de mecânica e eletrônica dos robôs criados e finalizando com a seção da comunicação entre os módulos e a conclusão e agradecimentos, além das referências.

[1]Todos os autores estão no RobôCIn no Centro de Informática, Universidade Federal de Pernambuco, Brazil robocin@cin.ufpe.br

II. LOCALIZAÇÃO

Para identificação e detecção dos robôs e bola em campo, foi desenvolvido pela nossa equipe um software de Visão Computacional em C++, utilizando a biblioteca *open-source* OpenCV [1] baseado em detecção de cores de etiquetas, localizadas na parte superior de cada robô. Para captura das imagens necessárias para detecção, é utilizada uma câmera posicionada a 2 metros de altura sobre o campo. O fluxo de técnicas de visão computacional pode ser visto como um *software* de arquitetura de Pipes e Filtros [2] com as seguintes etapas: Pré-processamento, Segmentação, Identificação de objetos e Transformação de sistema de coordenadas.

A. PRÉ-PROCESSAMENTO

Na etapa de pré-processamento são utilizados primariamente configurações internas da própria câmera, como controle de brilho e ajuste de foco automático. Tais configurações eram definidas através de um *script* de comandos *bash* gerado pela equipe. A etapa de pré-processamento foi atualizada e a versão atual conta com comunicação direta com o drive da câmera para manipular as configurações que são definidas internamente, em nível de *hardware*. Além disso, estamos desenvolvendo pesquisas em métodos para controlar mudanças na iluminação, além da aplicação de filtros padrões que melhoram os dados para etapa de segmentação.

B. SEGMENTAÇÃO

Na etapa de segmentação, é feita a separação das imagens em informações úteis para as próximas etapas, eliminando partes das imagens que não são necessárias para a identificação dos objetos. Foi escolhido o desenvolvimento de uma segmentação de cores com a utilização de uma *Look Up Table* (LUT) [3]. A segmentação de cores se baseia na análise das intensidades dos canais de cor de cada pixel, onde o pixel se configura como um *pixel* de

interesse se a intensidade dos canais de cor do *pixel* está presente dentro de um dos intervalos da cor característica de alguma etiqueta ou bola. A utilização de uma LUT calcula previamente a classe de todos os *pixels*, isto é, se cada pixel pertence ou não a algum intervalo de cor. Dessa forma, em tempo de execução é necessário somente fazer a consulta a LUT para saber a classe do pixel em questão. A LUT aparece como uma complementação da técnica de segmentação através de cores, tornando a segmentação mais veloz, possibilitando realizar poucas operações. Para melhorar a segmentação, os valores guardados na LUT são de intensidades no espaço YUV, espaço que garante mais confiabilidade, pois as características que sofrem influência da luminosidade estão presente em somente 2 dos 3 canais de intensidade, tornando o processo mais robusto à variações de luminosidade se comparado aos espaços RGB e HSV. Dentre os estudos feitos no processo de segmentação para a competição, está presente a adaptação das equações de conversão RGB para YUV, para equações com operadores binários, tanto na inicialização quanto na consulta à LUT. Esse formato de equação permite uma otimização no uso da CPU. Foram também criadas ferramentas na interface do programa de forma a acelerar o tempo de calibração da segmentação, e detecção de erros na mesma. Tais ferramentas incluem a possibilidade de salvar e carregar configurações de segmentação através de diferentes arquivos em tempo de execução; e a adição de novas visualizações de depuração da imagem da câmera. Após a etapa de segmentação, a matriz com os elementos de interesse (Figura 1) será enviada para a etapa seguinte.

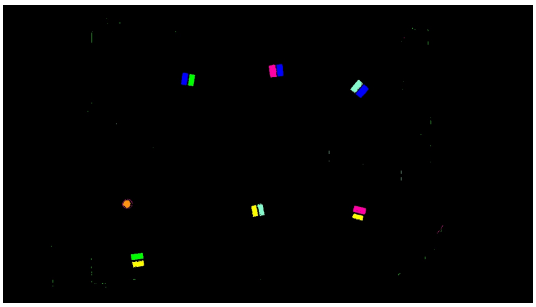

Fig. 1. Elementos de interesse identificados.

C. IDENTIFICAÇÃO DE OBJETOS

Dada a imagem segmentada do campo, a equipe optou por utilizar dois métodos de localização dos robôs, a escolha dependerá do formato das marcações nos robôs das equipes adversárias. Primeiramente foi estudado o método de clusterização *K-Means* [4], no qual os robôs de um mesmo time são reconhecidos através do agrupamento das cores em comum. Inicialmente, são posicionados N *clusters* aleatoriamente, de tal maneira que existam dois *clusters* para cada robô e um para a bola. Tal processo é apresentado na Figura 2, simplificado com apenas dois robôs e a bola. Cada *cluster* tem sua cor de referência e este só irá se incorporar com *pixels* da mesma cor. Então cada *pixel* tem sua distância comparada a todos os *clusters* que contenha a mesma cor e será incorporado ao centróide que estiver a menor distância euclidiana, como pode ser visto na Figura 3. Após a varredura da matriz, o centróide irá para o ponto que contém a média das coordenadas entre os pontos associados. O processo será repetido até que gere uma convergência dos núcleos. Na próxima iteração o algoritmo é iniciado novamente, entretanto ao invés dos núcleos serem escolhidos de forma aleatória, é escolhida a posição do *frame* anterior.

Fig. 2. Campo com Clusters posicionados aleatoriamente.

Fig. 3. Clusters com as futuras posições.

O outro método implementado pela equipe foi baseado em uma busca em profundidade (DFS) onde a matriz segmentada é varrida e ao detectar alguma a cor primária ou secundária é iniciado a DFS na qual a cada interação é feita o somatório das coordenadas. Ao final da varredura é calculada a média do somatório para adquirir o centro de cada cor. Nos testes realizados pela equipe foi constatado que o método por DFS, possui melhor eficiência nos casos gerais entretanto, em casos no qual a equipe adversária utiliza padrões como o adotado na *RoboCup - Small Size League*, na qual a cor secundária é descontínua o *K-Means* possui uma melhor precisão.

D. CONVERSÃO DE COORDENADAS

A última etapa de detecção de objetos é a conversão dos pontos adquiridos no plano da imagem para pontos em escala real, no sistema de coordenadas do campo. Para alcançar o resultado desejado, utilizamos métodos de interpolação de coordenadas, utilizando correspondências já conhecidas de pontos da imagem com os pontos reais para gerar uma transformada que execute a conversão para entregar posições e velocidades no sistema de coordenadas desejado.

III. PLANEJAMENTO

O planejamento de caminho tem como principal objetivo definir as posições para as quais os robôs têm que se locomover. A partir das coordenadas de cada elemento no campo de jogo (bola e robôs), a estratégia de decisão decide o comportamento de cada robô a cada instante de jogo. Para execução de um comportamento utiliza-se o algoritmo de campos potenciais [7] para definir uma posição alvo para o robô e o caminho até esta. Estes dados são convertidos em velocidades e sentido de rotação dos motores responsáveis por conduzir as rodas dos robôs.

Utilizou-se o software de simulação VREP para a realização de testes do algoritmo, visto que foi possível desenvolver um ambiente similar ao real e, assim, modular diversos cenários de jogo sem haver uma dependência técnica das outras áreas da equipe (Figura 4).

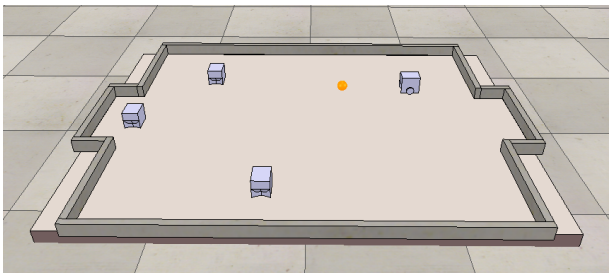

Fig. 4. Exemplo de situação de jogo simulada no VREP.

A. TOMADA DE DECISÃO

A primeira etapa da tomada de estratégia é determinar qual será o comportamento de cada robô para um determinado instante de jogo. Para realizar essa funcionalidade é levado em conta a posição de todos os elementos no campo, com isso será possível determinar qual comportamento dos robôs, ou seja, se ele será o goleiro, zagueiro ou atacante.

A abordagem escolhida para a escolha de cada comportamento foi a distância para a bola e a distância para o gol que está sendo defendido. O robô mais próximo da bola será o atacante, o mais próximo do gol defendido será o goleiro e o que restar será o zagueiro. Se forem identificados apenas dois robôs no campo, só serão utilizados os comportamentos de atacante e de goleiro. Se apenas um robô for identificado no campo, ele será o atacante.

Determinadas situações de jogo também podem ser identificadas, o que facilita o uso de jogadas específicas para cada situação. Pode-se separar essas situações em duas categorias: estáticas e dinâmicas. As estáticas são as que podem ser identificadas a partir de um momento em que o jogo encontra-se parado, como por exemplo: falta, pênalti ou saída de jogo. Por outro lado as dinâmicas são as identificadas durante o andamento do jogo, como por exemplo qual time tem a posse de bola ou se o adversário possui um comportamento mais ofensivo ou mais defensivo. Com todas essas informações é possível implementar diversos comportamentos para diferentes situações de jogo.

O algoritmo de campos potenciais é uma técnica muito utilizada para planejamento de rota de robôs. Neste, o robô é um ponto que sofre a influência de um campo potencial gerado pelos outros robôs, pela bola, pelo seu objetivo final e pelo campo de jogo, onde as posições a serem evitadas são pontos com maiores potenciais e as posições mais próxima do objetivo do robô são pontos com menores potenciais. Para definir o alcance do campo, de cada obstáculo e do objetivo, foram geradas gaussianas (positivas ou negativas) que utilizam a equação 1, onde os parâmetros σ_x e σ_y definem o alcance da Gaussiana, onde, A é a amplitude máxima ou mínima da Gaussiana, x_o e y_o são as coordenadas do obstáculo, ou do objetivo, e x e y são as coordenadas do ponto que está sendo analisado. Um exemplo de campos potenciais gerados para a Figura 4 ilustrado na Figura 5.

$$f(x,y) = Ae^{-(\frac{(x-x_O)^2}{2\sigma x} + \frac{(y-y_O)^2}{2\sigma y})} \quad (1)$$

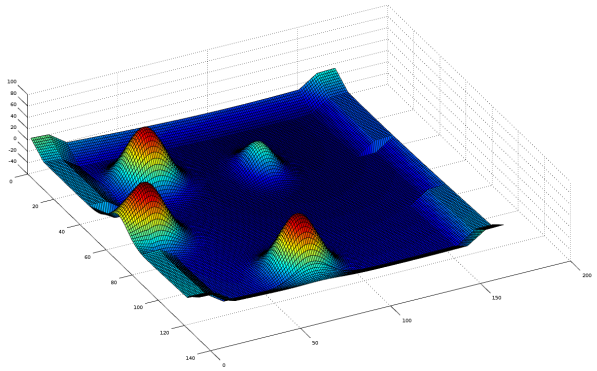

Fig. 5. Campos potenciais gerados através do campo e 3 robôs.

Um dos problemas encontrados com o uso de campos potenciais foi que em algumas situações são gerados mínimos locais indevidos. Nestas posições, o robô tende a ficar estagnado, pois o algoritmo de planejamento de caminhos tende a achar que esse local é o objetivo final, apesar de não o ser. Para tratar esse problema foi utilizado uma função de custo de busca em grafos, que tende buscar caminhos com menor custo. Para o cálculo do custo de um dado ponto leva-se em conta o custo para se chegar

a esta posição, o valor do campo potencial desta posição e a distância do ponto até o objetivo. Esse algoritmo computa os custos de cada rota de navegação antes de enviar as informações ao robô. Com esse cálculo caminhos com mínimos locais ficam com um custo muito alto o que tornará este caminho indevido, e leva o algoritmo a buscar caminhos com custos menores, que levam ao caminho certo.

Para a navegação acontecer da maneira esperada, foi implementado uma função que calcula a velocidade de cada roda e corrige pequenas alterações no caminho, que é um problema recorrente no âmbito da robótica. Na maioria das vezes, os valores calculados para que o robô chegue ao objetivo não são os mesmos que o *hardware* executa. Isso acontece devido a pequenas variações de um motor para o outro como algumas outras calibrações necessárias dentro do *hardware*. Então essa função compara via *software* o movimento executado pelo robô com o que é esperado e com isso faz as correções em tempo de execução. Para exemplificar, uma rota em linha reta, sem essas correções, pode se tornar uma curva causada por um erro de calibração em algum dos motores.

B. POSICIONAMENTO DO GOLEIRO

Na estratégia atual, é destinada a um robô a função de ser o último jogador em campo e guardar o gol. Então, é preciso dar uma atenção especial ao seu posicionamento. Foi decidido utilizar a abordagem descrita por Seesink [8], que define pontos de um semicírculo como possíveis objetivos para o goleiro. Esse semicírculo tem o centro no meio da reta entre as duas traves do gol a ser defendido. Para definir qual o melhor ponto da semicircunferência em que o goleiro deve estar, são traçadas duas retas com a mesma origem, a bola, e as traves do gol. Com a bissetriz do ângulo formado pelas duas retas encontradas anteriormente, é calculada a sua intersecção com a semicircunferência e é definido como objetivo para o goleiro, como ilustra a Figura 6. A angulação do robô será sempre a mesma de uma reta tangente a semicircunferência, com isso, o goleiro irá sempre se posicionar de lado com bola, facilitando o afastamento da mesma. Utilizando essa abordagem ao invés da movimentação na linha do gol, não é preciso cobrir todo gol, mas sim uma área menor.

IV. MECÂNICA

O robô foi projetado com o intuito de ser modular, permitindo uma fácil montagem e manutenção. Portanto toda a estrutura para construção do robô foi realizada através de impressão 3D (Figura 7). Os módulos da estrutura são encaixáveis, os quais não necessitam de parafusos para sua montagem. Para projetar todas as peças, utilizou-se o Solid Works 3D CAD [5], que permite realizar a modelagem e exportação de peças para impressão 3D.

Juntamente a uma estrutura impressa em 3D, o robô possui dois micro motores com redução integrada de 75:1, e permitem 400 RPM com torque de 1,6 Kg-cm. Para

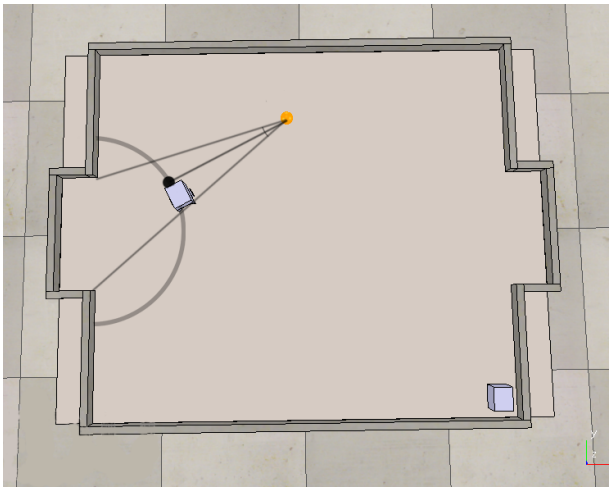

Fig. 6. Pontos de um semicírculo servindo como possíveis objetivos para o goleiro.

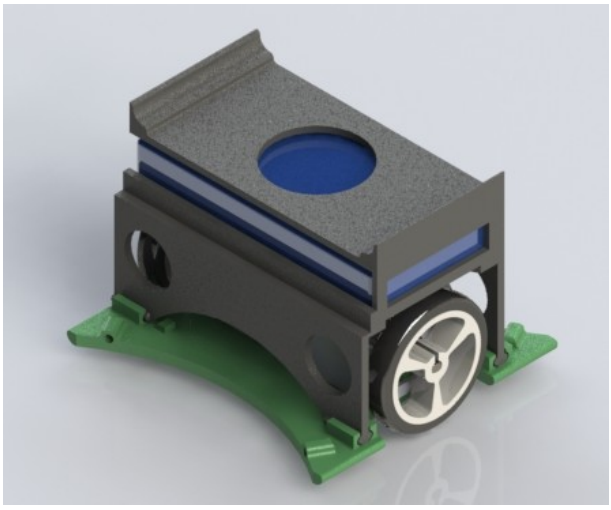

Fig. 7. Estrutura interna do robô e seus módulos estruturais.

completar a estrutura de locomoção do robô existem duas rodas com 32mm de diâmetro, localizadas nas laterais do robô, e 4 pivôs de apoio, já inclusos na estrutura impressa.

V. ELETRÔNICA

Para controlar o robô é utilizado um Arduino Pro Mini de 16 MHz, escolhido pela praticidade, tamanho, e capacidade de controlar todos os dispositivos requisitados no robô. Para controle de potência nos motores é utilizado o driver de motor TB6612FNG, que possui limite de corrente contínua de 1A por motor e 3A para corrente de pico.

O circuito do robô é composto também por um módulo *wireless*, responsável pela comunicação com o computador, e uma IMU (*Inertial Measurement Unit*), composta por um acelerômetro, magnetômetro e um giroscópio, a qual permite monitorar a orientação do robô, assim como sua velocidade de rotação.

Visando melhorar a precisão no deslocamento dos robôs, o controle dos motores que inicialmente era feito através de malha aberta, ou seja, a potência elétrica era enviada aos motores, mas não existia nenhum feedback da velocidade com que as rodas estavam. Foi acrescentado ao robô um dispositivo *encoder* no aro da roda, com imãs e sensor de efeito *hall*, para aferir a velocidade de rotação das rodas. Atualmente, a equipe está desenvolvendo seu próprio *encoder*, pois o espaço atualmente disponível dentro do robô, não permite o uso de um *encoder* comercial. Também será incluído um sensor óptico, utilizado em mouses, que aliado ao IMU permitirá realizar a odometria do robô.

Para otimizar o espaço do robô, o circuito foi impresso em placa de dupla camada através da prototipadora LPKF ProtoMat S63. Para desenhar o layout do circuito impresso foi utilizado o Eagle [6] (Figura 8), software especializado em produzir placas de circuito impresso.

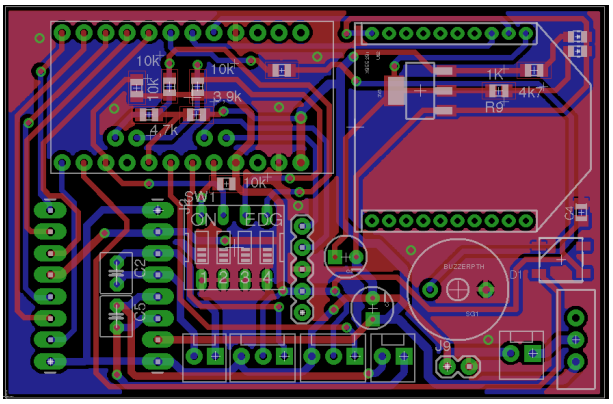

Fig. 8. Projeto do circuito feito através do Eagle Circuit CAD.

VI. COMUNICAÇÃO

Na etapa de comunicação, foi escolhida a tecnologia ZigBee, o xBee Series 2 especificamente, para a troca de mensagens entre o computador e os robôs. Com esta tecnologia, foi possível que o computador enviasse informações como velocidade dos motores e também receber dados como o nível de bateria de cada robô, por exemplo.

Para realizar a comunicação, foi definido um protocolo para a mensagem *broadcast* que o computador envia para o robô (Figura 9). Esse protocolo é dividido em seções de bits para conter informações do ID do robô de destino, a velocidade de cada motor, e se é preciso que o robô retorne o nível de bateria.

Após competir na Very Small Size em 2016, o time teve problemas de interferência com o rádio xBee e, consequentemente, houve falhas na comunicação com todos robôs. Isto porque o xBee Series 2 utiliza um protocolo que busca a entrega das mensagens, ou seja, caso algum robô não for encontrado na rede, todos os rádios na rede ecoavam a mensagem para tentar achá-lo, e, portanto, com um robô perdido, todos os outros ficavam ocupados ecoando a mensagem, mas sem obter sucesso.

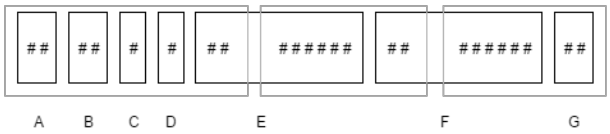

Fig. 9. Protocolo de comunicação. A: Início de comunicação. B: Identificador do robô de destino. C: Bit indicativo de retorno de nível de bateria. D: Bit indicativo de chute (não utilizado nesta versão). E: Velocidade do motor 1. F: Velocidade do motor 2. G: Fim de comunicação..

Reconhecendo que a comunicação é crucial para o funcionamento do sistema, testes em outras tecnologias foram realizados pela equipe. Surgiu, então, uma adaptação no circuito para permitir o uso tanto do xBee quanto do módulo nRF24L01+. Desta maneira, existe uma tecnologia de *backup* que permite se comunicar com até 5 robôs numa topologia de rede em estrela, além de que, esta é mais flexível na implementação do protocolo de comunicação que o xBee.

VII. CONCLUSÃO

No presente *Team Description Paper* (TDP), é mostrado como projetar e desenvolver o sistema necessário para participar na categoria *Very Small Soccer League*. Para isso, foi descrito um sistema de localização e detecção de objetos com objetivo de localizar a bola e os robôs em campo, um planejador de caminhos e decisor de comportamentos a fim de permitir a elaboração de estratégias de jogo, o projeto mecânico e eletrônico dos robôs e a comunicação entre eles e o sistema de controle.

AGRADECIMENTOS

A equipe gostaria de agradecer o Centro de Informática da UFPE pelo apoio financeiro e de recursos durante todo o processo do projeto. Também gostaríamos de agradecer à todo apoio dado pelos professores Edna Barros e Hansenclever Bassani.

References

[1] OpenCV. Disponível em: http://opencv.org/. Acessado por último em 20 de junho de 2016.
[2] Sommerville, Ian. Engenharia de Software, 9ª edição. Pearson Education.
[3] Discrete YUV Look-Up Tables for Fast Colour Segmentation for Robotic Applications
[4] Hartigan, J., & Wong, M. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1), 100-108. doi:1. Retrieved from http://www.jstor.org/stable/2346830
[5] CAD Solid Works. Disponível em: http://www.solidworks.com/. Acessado por último em 20 de junho de 2016.
[6] CAD Eagle. Disponível em: http://www.cadsoftusa.com/. Acessado por último em 20 de junho de 2016.
[7] Roland Siegwart and Illah Reza Nourbakhsh and Davide Scaramuzza Introduction to Autonomous Mobile Robots. The MIT Press, 2ª edição 2011.
[8] Seesink, R. A. (2003). Artificial Intelligence in multi-agent robot soccer domain (Doctoral dissertation, Masters Thesis, University of Twente).