RobôCIn VSSS Description Paper

João Amaro de Assunção Bisneto[1], João Gabriel Machado da Silva[1], José Douglas Pontes Silva[1], José Victor Silva Cruz[1], Juliana do Nascimento Damurie da Silva[1], Lucas Dias Maciel[1], Lucas Henrique Cavalcanti Santos[1], Lucas Oliveira Maggi[1], Mariana da Silva Barros[1], Matheus Viana Coelho Albuquerque[1],Thalisson Moura Tavares[1], Tiago da Silva Barros[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, 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 cinco 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 estratégia. O módulo de estratégia 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 segunda seção, é explicado o módulo de localização dos objetos em campo. Na seção estratégia, 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.

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 Dutos 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 configurados os parâmetros internos câmera, como controle de brilho e ajuste de foco automático. Inicialmente, tais configurações eram definidas através de um *script* de comandos *bash* gerado pela equipe. Porém, a necessidade da execução desse script antes do início do código principal aumenta a possibilidade de erro. Por isso a etapa de pré-processamento foi embutida no software principal do RobôCIn, que conta com comunicação direta com o drive da câmera para manipular as configurações que são definidas a 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

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

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.

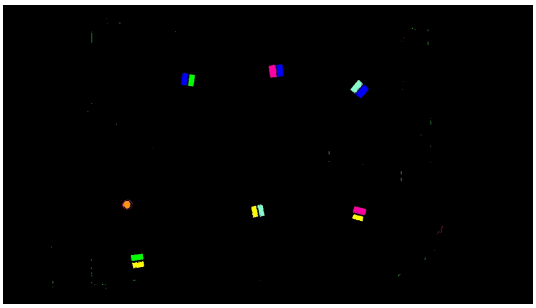

Figura 1. Elementos de interesse identificados.

C. IDENTIFICAÇÃO DE OBJETOS

Uma abordagem comum e utilizada pelo RoboCIn em anos anteriores para identificação de objetos se dá através de uma busca em profundidade na matriz segmentada, tirando a média das posições de pixels agrupados com a mesma cor para achar o centróide de cada objeto. Este ano, foram trazidas melhorias no sistema de identificação de objetos. De forma a otimizar o tempo de processamento, a matriz segmentada é comprimida através do algoritmo *run length encoding*, comprimindo *pixels* que sejam da mesma cor e estejam na mesma linha em uma única célula de informação, chamada de *run*, que contém a cor e a quantidade de *pixels* correspondentes. Essa compressão ajuda a otimizar o tempo para encontrar os centróides, pois possibilita utilizar o algoritmo *union-find* para juntar *runs* vizinhas da mesma cor, calculando o centroide de cada conjunto a cada junção nova. Por fim, o número de objetos será o número de conjuntos disjuntos formados.

A utilização da compressão *run length encoding* em conjunto com o *union-find* tem como principal vantagem a diminuição no número de iterações pela matriz segmentada completa. Com essa melhoria, o sistema de visão do RoboCIn é capaz de processar uma imagem com tamanho médio de 800x600 em cerca de 2 ms.

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. Esse método consiste em utilizar 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. ESTRATÉGIA

O módulo de estratégia é responsável por determinar as melhores ações e caminho à serem tomados pelos robôs, utilizando como fonte de informação a posição, orientação e velocidade dos robôs e bola presentes em campo vindas do módulo de visão. É importante que o módulo de estratégia seja rápido e preditivo, de forma a prover uma reação rápida dos robôs em função das situações de jogo e assim garantir uma maior probabilidade de vitória.

O módulo de estratégia do RobôCIn pode ser separado em camadas, nomeadas escolha de formação, escolha de ação, planejamento de caminho e controle de movimento. Tais camadas são executadas em ordem e tem como saída final as velocidades de roda que devem ser enviadas para cada robô de forma a executar o plano de jogo, como pode ser observado na Figura 2. As mesmas serão descritas nas subseções A, B, C e D, respectivamente.

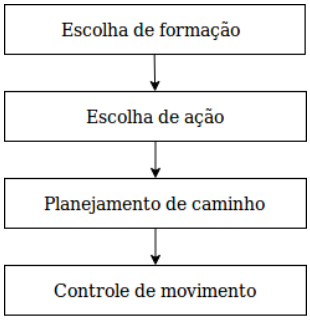

Figura 2. Fluxo das camadas do módulo de estratégia.

A. ESCOLHA DE FORMAÇÃO

Para explicar a camada de escolha de formação, é necessário primeiramente introduzir o conceito de comportamento. O comportamento de um robô é definido como o conjunto de relações entre as ações que o robô pode tomar e as situações em que cada ação é tomada. Sua definição é análoga a definição de função de um jogo de futebol. Por exemplo, semelhante a um goleiro de um time de futebol que, ao perceber que a bola está se aproximando de seu gol, deve ir em direção à bola com objetivo de desviar sua trajetória, um robô programado com o comportamento

de goleiro deve executar ação semelhante em campo. O software do RobôCIn apresenta os seguintes comportamentos que são utilizados atualmente: Atacante, Defensor e Goleiro.

A camada de escolha de formação tem como função analisar as posições dos robôs e da bola no campo e assim decidir qual comportamento é o mais adequado para cada robô executar. É importante observar que a escolha de formação é executada *frame* a *frame*, ou seja, um robô que começou o jogo sendo goleiro pode terminar sendo atacante, e vice-versa. Tal característica possibilita uma maior adaptabilidade ao time de robôs. Essa adaptabilidade pode ser observada na Figura 3. Nessa situação, o robô atacante, marcado pelo identificador 1, não está em boa posição para chegar na bola pois está sendo marcado pelo robô inimigo de identificador 2. Já o robô defensor, marcado com o identificador 3, tem um caminho quase retilíneo em direção ao gol. Nessa situação, a troca de funções faz com que este defensor tome o comportamento de atacante e aproveite da chance para fazer o gol. Para decidir que robô é o melhor para cada posição, são utilizadas as distâncias dos robôs para a bola e para o gol aliado, priorizando a escolha do goleiro em detrimento das outras.

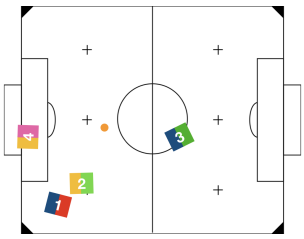

Figura 3. Situação de jogo onde a troca de funções entre os jogadores é vantajosa

B. ESCOLHA DE AÇÃO

A camada de escolha de ação, como é sugerido pelo nome, diz respeito à avaliação da melhor ação a ser tomada, dado o comportamento de cada robô. A escolha de ações é feita através de uma máquina de estados, onde os estados são ações e as arestas são situações de jogo. O exemplo de uma máquina de estados para um comportamento de atacante pode ser observada na Figura 4. Atualmente no módulo de estratégia em questão, as seguintes ações podem ser tomadas: buscar a bola, conduzir a bola, Girar e acompanhar a bola.

C. PLANEJAMENTO DE CAMINHO

A escolha da ação a ser executada implica na definição de uma coordenada e orientação definidas como objetivo, que deve ser alcançado o mais rápido possível para que a ação tenha sucesso. É trabalho da camada de planejamento de caminhos definir qual a melhor trajetória a ser seguida de forma a chegar no objetivo sem maiores problemas.Uma das abordagens usadas pela equipe é definir o caminho a ser seguido por curvas de Bézier, geradas com

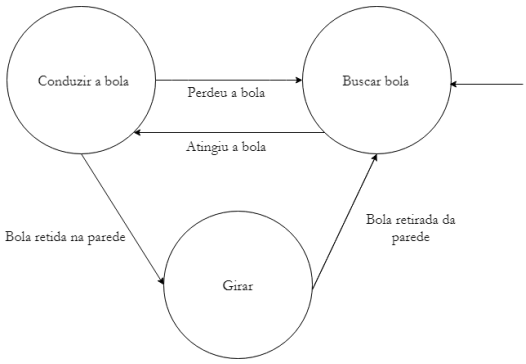

Figura 4. Máquina de estados que representa o comportamento do atacante

4 pontos de controle, onde definimos a posição e a pose de início, e a posição e a pose desejada no fim do trajeto. No

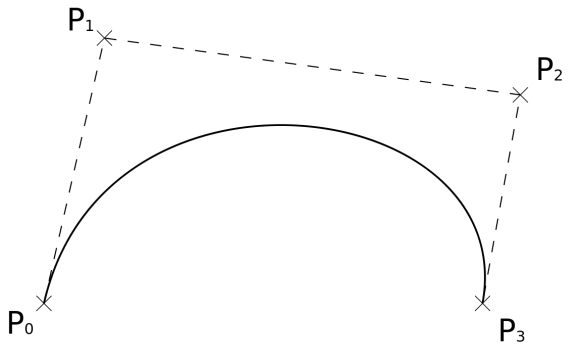

Figura 5. Curva Cúbica de Bézier

exemplo da Figura 5, a posição inicial é definida pelo ponto P_O, a pose inicial é definida pela reta $P_O{}^*P_1$, a posição final é definida pelo ponto P_O, e a pose final definida pela reta $P_2{}^*P_3$. Todos os pontos da trajetória gerada são calculados usando a Equação 1.

$$B(t) = (1-t^3)P_O + 3t(1-t^2)P_1 + 3t^2(1-t)P_2 + t^3P_3 \quad (1)$$

Para o desvio de obstáculos usando Bézier checamos se algum dos pontos na curva gerada, invade o espaço dos outros robôs, caso isso ocorra, geramos um novo trajeto, usando duas novas curvas que passam por um ponto estratégico em uma distância segura do objeto que estava em rota de colisão.

Outra abordagem da equipe é definir o caminho a ser seguido usando campos de vetores unitários [9] que podem ditar a direção de movimento do robô. Duas espirais hiperbólicas são utilizadas para gerar os vetores, uma para fazer o robô se direcionar à bola e uma para fazer o robô desviar de obstáculos.

O resultado do processamento do módulo de planejamento de caminho será um conjunto ordenado de pontos no plano que descrevem a trajetória a ser executada. A partir dessa trajetória, é gerado um objetivo e angulação

instantâneos, que são pontos do caminho alcançáveis num intervalo de *frame*.

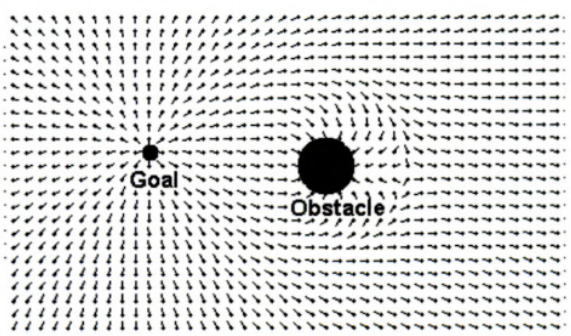

Figura 6. Univector Field [9]

D. CONTROLE DO MOVIMENTO

Dado o objetivo e angulação instantâneos gerados pelo planejamento de caminhos, é necessário mapear tais valores para velocidades de rodas que produzam o resultado desejado. Para tal, é utilizada a malha de controle fechada descrita em [8] que garante a movimentação do robô até o ponto desejado (x, y) com a angulação θ. A lei de controle que a malha descreve é dada pelas equações 2, 3, 4, 5 e 6, onde ϕ_v é a direção instantânea desejada, θ_e é o erro angular, L é a distância entre as rodas, p é o a posição atual do robô, g é o ponto destino, V_m e R_m são as velocidades linear e angular máximas e K_W e K_d são constantes.

$$v = min(v_1, v_2, v_3) \tag{2}$$

$$w = \phi_v v + K_W sgn(\theta_e)\sqrt{|\theta_e|} \tag{3}$$

$$v_1 = \frac{2V_m - LK_W\sqrt{|\theta_e|}}{2 + L|\phi_v|} \tag{4}$$

$$v_2 = \frac{\sqrt{K_W^2 + 4R_m|\phi_v|} - K_W\sqrt{|\theta_e|}}{2|\phi_v|} \tag{5}$$

$$v_3 = K_d||p - g|| \tag{6}$$

Tais velocidades angular e linear são então mapeadas para velocidades das rodas utilizando o modelo de cinemática de acordo com as equações 7 e 8.

$$V_L = v - \frac{L}{2}.w \tag{7}$$

$$V_R = v + \frac{L}{2}.w \tag{8}$$

Dentre as malhas de controle existentes, essa foi escolhida por tratar o problema do deslizamento do robô quando a velocidade linear e angular são muito altas. Isso garante que o robô se mova sempre com a maior velocidade possível sem causar esse efeito. Dentre as os trabalhos deste ano, foi dada uma atenção especial ao estudo da influência do *delay* da câmera sobre a performance dos robôs no erro entre o caminho desejado e o percorrido. Foi percebido que, com o aumento da velocidade do robô, o *delay* da câmera, que chega a cerca de 90 ms, pode aumentar o erro de trajetória, principalmente em curvas bruscas. Tal situação ocorre pois a percepção de que o robô chegou à um ponto desejado é atrasada, fazendo com que o robô, por exemplo, execute uma curva a partir de um ponto diferente do ponto planejado.

Para compensar o *delay* da câmera na tomada de decisão, implementamos um sistema de previsão que funciona usando o método de Runge-Kutta. Tal método se baseia em equações que levam em consideração o ângulo que o robô está se movimentando e sua velocidade para predizer onde o robô estará. A equação 9 calcula o ângulo futuro que o robô vai estar baseado no seu ângulo atual e sua velocidade angular. As equações 10 e 11 usam a média entre o ângulo atual e o futuro, a posição atual e a velocidade na coordenada para prever a posição futura dos objetos em campo.

$$\theta(t + \Delta t) = \theta t + \Delta t.wt \tag{9}$$

$$x(t + \Delta t) = x(t) + \Delta t.\cos((\theta.t + \theta(t + \Delta t))/2).V_x(t) \tag{10}$$

$$y(t + \Delta t) = y(t) + \Delta t.\cos((\theta.t + \theta(t + \Delta t))/2).V_y(t) \tag{11}$$

IV. MECÂNICA

O robô foi planejado com o propósito de ser uma peça única e sólida, proporcionando ao robô estabilidade. Toda a estrutura para construção do robô foi realizada através de impressão 3D (Figura 7). A estrutura é composta dos seguintes módulos chassi, tampa do motor, tampa das baterias, tampa chassi e capa. O módulo chassi tem como função dar suporte aos motores, as baterias e a placa principal. A tampa do motor segura o motor no chassi, a tampa das baterias tem a função de segurá-las e capa protege a placa do robô e é um suporte para as *tags*. Para projetar todas peças, utilizou-se o CAD 3D Autodesk Inventor [5], que permite realizar a modelagem e exportação de peças para impressão 3D.

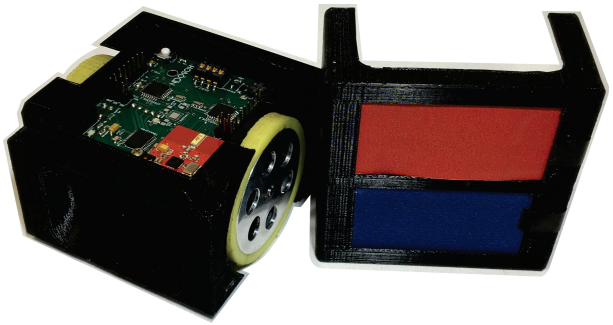

Figura 7. Robô desenvolvido pelo RobôCIn em impressão 3D.

Além da uma estrutura impressa em 3D, o robô da equipe RobôCIn possui dois micro motores com redução integrada de 50:1, e permitem 600 RPM com torque de 0,86 Kg-cm. Para completar a estrutura de locomoção do robô existem duas rodas com 50mm de diâmetro,

localizadas nas laterais do robô, e 4 pivôs já inclusos na estrutura impressa em 3D.

V. ELETRÔNICA

Para controlar o robô são utilizados dois microcontroladores ATmega328, onde um atua como mestre e o outro como escravo. Eles foram escolhidos 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, 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, um Sensor de Mouse que retorna as posições em x e y do robô possibilitando uma descrição de trajetória do robô.

A fim de otimizar o espaço do robô, os circuitos foram impressos em placas de dupla camada. Para fazer os layouts dos circuitos impressos foi utilizado o software de desenvolvimento de placas de circuito impressos Eagle. Nesta nova versão do robô, tivemos o desafio de adicionar dois sensores óptico de mouse, com a função de rastrear o deslocamento do robô no campo, e auxiliar o controle de baixo nível. Os sensores de mouse necessitam estar perto do chão e por isso, desenvolvemos uma placa separada para comportar o sensor de mouse que se conecta a placa principal através de um conector, pois a placa principal fica na parte superior do robô. A placa principal foi desenvolvida conforme Figura 8 para compor a parte superior do robô e a placa do mouse foi desenvolvida conforme Figura9.

VI. COMUNICAÇÃO

Para realizar a comunicação entre os robôs e o computador, na competição do *Very Small Size* em 2017 o nosso circuito permitia o uso tanto do xBee Series 2 quanto o do módulo nRF24L01+. Depois de testar o desempenho do módulo nRF24L01+ nos jogos, julgou-se melhor optar pelo uso único dele e anular o uso do xBee Series 2 devido aos problemas dispostos pelo mesmo na competição *Very Small Size* de 2016.

Ao decidir usar a comunicação do tipo radiofrequência (RF), a equipe implementou um protocolo através de um classe *NRF24_Communication*, onde foram utilizadas as funções nativas do *RF24*. A comunicação foi desenvolvida de modo que incluísse não somente o envio de mensagens do computador para o robô, mas também a resposta do robô para o computador.

O módulo utilizado, nRF24L01+, se comunica com o microcontrolador seguindo o protocolo *Serial Peripheral Interface* (SPI). Assim, é utilizado um microcontrolador Arduino e dois módulos RF para a comunicação por parte do computador, onde um módulo atua como receptor, e o outro como transmissor. Contudo no robô é utilizado um único RF que recebe os dados que o computador enviou e ao ser solicitado envia mensagens de resposta para o computador.

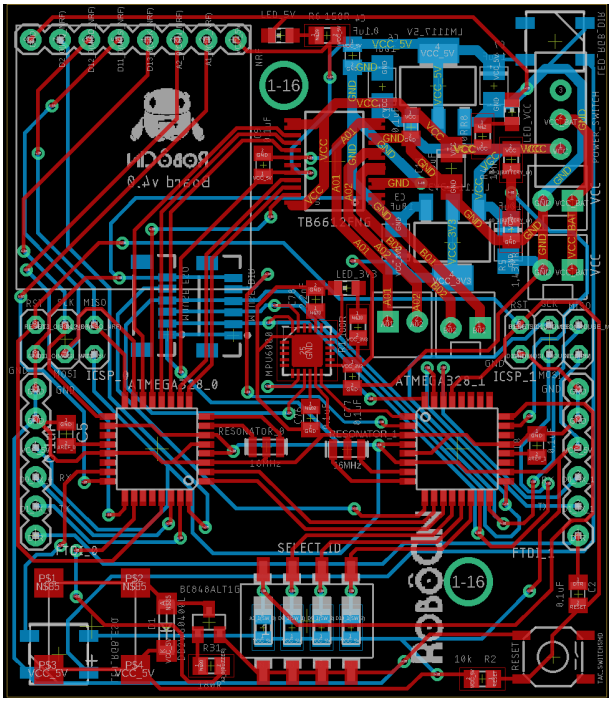

Figura 8. Projeto do circuito principal feito através do Eagle Circuit CAD.

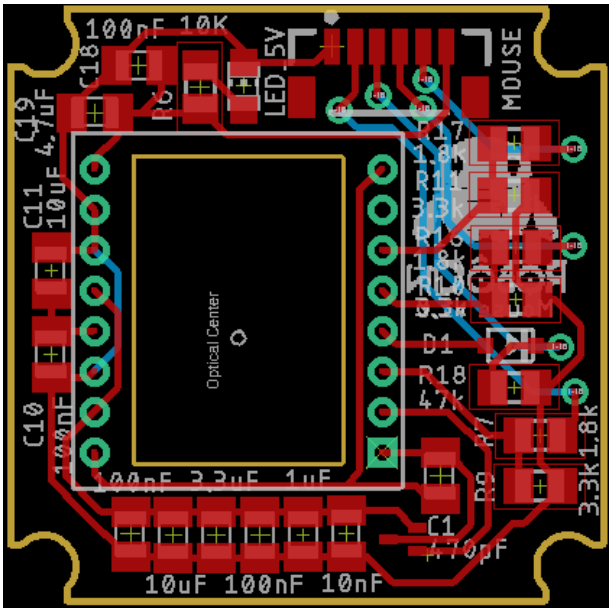

Figura 9. Projeto do circuito do mouse feito através do Eagle Circuit CAD.

Para realizar a comunicação, é seguido um fluxo como está mostrado na figura 10. Inicialmente, o computador envia o dado via porta serial para o Arduino que está

conectado a ele. O módulo RF conectado a este Arduino envia a mensagem para o NRF do robô, e este, ao receber o código, executa as ações. Caso haja necessidade de resposta, o robô as envia para o computador também pelo seu módulo RF.

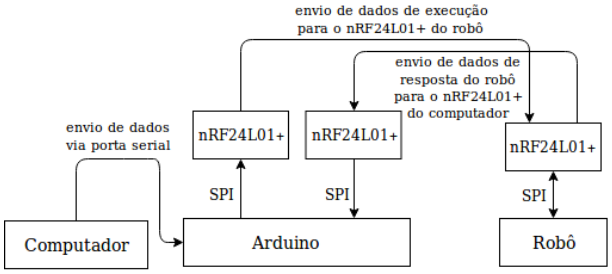

Figura 10. Fluxo das mensagens de comunicação, passagem dos dados do computador para o robô.

Para melhor controle do protocolo de comunicação, o time definiu cinco tipos diferentes de mensagens, para identificar: as velocidades dos motores dos robôs, a velocidade angular e linear a qual o robô deve seguir, as constantes do sistema PID do robô, a posição na qual o robô se encontra e o nível de bateria do robô. Estes dois últimos tipos caracterizam as mensagens de retorno do robô. Cada mensagem tem uma quantidade de bytes de acordo com as seus campos específicos e com a função desempenhada.

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.

Referências

[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 3D Autodesk Inventor. Disponível em: https://www.autodesk.com.br/products/inventor/overview. Acessado por último em 16 de junho de 2019.
[6] CAD Eagle. Disponível em: http://www.cadsoftusa.com/. Acessado por último em 20 de junho de 2016.
[7] Seesink, R. A. (2003). Artificial Intelligence in multi-agent robot soccer domain (Doctoral dissertation, Masters Thesis, University of Twente).
[8] Kim, Jong-Hwan, et al. Soccer robotics. Vol. 11. Springer Science & Business Media, 2004.
[9] Jong-Hwan Kim, Dong-Han Kim, Yong-Jae, 2004.