RobôCIn IEEE Very Small Size Soccer 3v3 LARC 2022 Team Description Paper

Arthur Silva, Breno Cavalcanti, Carlos Silva, Elisson Araújo, Erick da Silva, Felipe Martins, Geovany Nunes, José Douglas Silva, Leonardo Silva, Luisa Cavalcante, Mariana Barros, Mateus Machado, Matheus Albuquerque, Morgana Galamba, Pedro Coutinho, Pedro Oliveira, Pedro Silva, Ryan Morais, Thiago Souza, Edna Barros[1]

***Abstract*— Este *Team Description Paper* 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 *Team Description Paper* são descritos os principais sistemas desenvolvidos: o sistema de localização por visão computacional, o sistema de tomadas de decisão, bem como a comunicação entre os robôs, o sistema de controle, e o sistema eletrônico e mecânico do robô projetado.**

I. INTRODUÇÃO

Para participar da competição da categoria IEEE *Very Small Size Soccer* (VSSS), são necessários cinco pilares essenciais: detecção de objetos, tomada de decisões, planejamento de caminhos, controle, 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 seção II, é explicado o módulo de localização dos objetos em campo. Na seção III, é detalhada tomadas de decisões realizadas pelo módulo de estratégia. Em seguida as seções IV e V descrevem a mecânica e eletrônica dos robôs utilizadas pela equipe e a seção VI que explica a comunicação entre os módulos. Em seguida, são apresentadas as seções de conclusão, agradecimentos e 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] e baseado na detecção de pares de cores, 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 coordenadas, descritas a seguir.

A. PRÉ-PROCESSAMENTO

Na etapa de pré-processamento, são configurados os parâmetros internos da câmera, como controle de brilho e ajuste de foco automático. 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 em nível de *hardware*. Além disso, estão sendo desenvolvidas 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 classificação de cada *pixel* da imagem, removendo ruídos para facilitar a identificação dos objetos. 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 dentro de um dos intervalos da cor característica de alguma etiqueta ou bola. Calculamos previamente a classe para todos os valores possíveis de *pixels*, isto é, definimos a classe de cor para cada cor possível. Nós utilizamos uma *Look Up Table* (LUT) [3] para armazenar esses resultados, de tal maneira que indexamos os pixels a partir da soma dos valores dos canais (R,G,B) multiplicados por 65536, 256, e 1, respectivamente. Dessa forma, em tempo de execução é necessário somente realizar uma consulta à LUT para saber a classe de um *pixel*. A LUT aparece como uma otimização da técnica de segmentação através de cores, tornando o processo de segmentação eficiente, realizando apenas operações de leitura e escrita durante a partida.

Para tornar a segmentação mais eficiente, os valores guardados na LUT são de intensidades no espaço RGB. Este é o espaço de cor no qual recebemos a imagem da câmera. Porém, o processo em si de segmentação precisa ser realizado em um espaço de cor que seja capaz de ser robusto a variações na iluminação e nas cores.

A geração da LUT possui as seguintes etapas:

- Normalização do espaço RGB.
- Filtragem de tons de cinza.

[1]Centro de Informática - Universidade Federal de Pernambuco, Av. Jornalista Anibal Fernandes, s/n - CDU 50.740-560, Recife, PE, Brasil. robocin@cin.ufpe.br

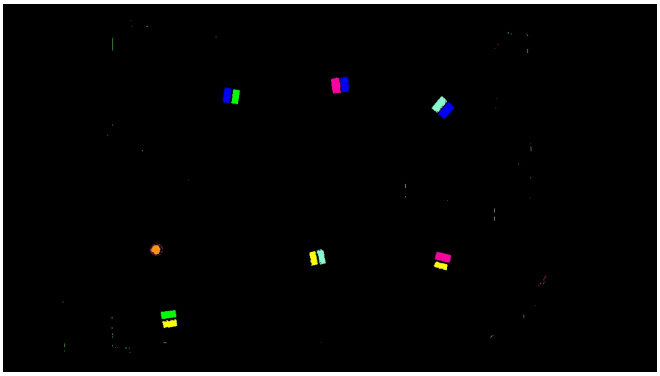


Fig. 1. Elementos de interesse identificados.

- Conversão para o espaço de cor HSV.
- Segmentação do espaço de cor HSV usando pivôs.

No processo de Normalização do espaço RGB é utilizada uma normalização vetorial ponderada, fortalecendo a cromaticidade das cores (isto é, se a cor for avermelhada, ela será normalizada para uma cor mais próxima do vermelho). De forma similar, isto ocorre para as outras cores primárias e secundárias, porém fazendo com que a região de ambiguidade tenha um distanciamento maior entre os seus elementos iniciais. Já a filtragem em tons de cinza é utilizada para remover falsos positivos das partes que compõem o campo, que geralmente são regiões muito claras (muito saturadas), ou muito escuras (trazem pouca precisão na cor), mas apresentam ainda algum nível de cromaticidade. Esta filtragem é feita a partir de uma região cilíndrica orientada na diagonal principal do espaço RGB. Após isso, convertemos todas as cores resultantes em cores no espaço HSV. Este espaço de cor distribui as cores por cromaticidade no canal ”Hue” (matiz), onde iremos realizar a segmentação das cores pela sua matiz, sendo descartados os valores dos canais S (Saturation) e V (Value). Associamos cada cor à cor do pivô que está mais próximo dela. Cada pivô está posicionado de tal forma que as fronteiras de decisão coincidem com as fronteiras da percepção humana, resultando em uma segmentação mais robusta à variação da iluminação, e torna-se mais fácil de segmentar por ter menos parâmetros. Temos um número para cada pivô: o Hue da cor que ele representa. Temos 7 cores para segmentar, portanto 7 parâmetros apenas para as cores. Além disso, temos mais um parâmetro para o filtro de tons de cinza, que delimita o quanto queremos filtrar com o cilindro no espaço RGB. Assim, há uma totalização de 8 parâmetros, e com isso é demonstrado que precisamos de menos parâmetros para configurar a segmentação do que uma segmentação por regiões retangulares em 3D, por exemplo, no espaço RGB. Após a etapa de segmentação, a matriz com os elementos de interesse (Figura 1) será enviada para a etapa seguinte.

Foram também criadas ferramentas na interface do programa de forma a acelerar o tempo de calibração da segmentação e de detecção de erros na mesma. Tais ferramentas incluem a possibilidade de salvar e carregar configurações de segmentação automaticamente em tempo de execução; e a adição de novas visualizações de depuração da imagem da câmera.

C. IDENTIFICAÇÃO DE OBJETOS

De forma a otimizar o tempo de processamento, a imagem segmentada é comprimida através do algoritmo *run length encoding* [4], 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* vizinhos da mesma cor, calculando o centróide de cada conjunto a cada junção nova. Por fim, o número de objetos será o número de conjuntos disjuntos formados cujo tamanho seja superior a um threshold.

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. Além disso, ele permite a paralelização do processo de compressão, que pode ser fragmentado em um *run* para cada linha, e depois se fazer o *union-find* desses runs.

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 caminhos a serem tomados pelos robôs, utilizando como fontes de informação a posição, orientação e velocidade dos robôs e da bola presentes em campo, informações essas que são fornecidas pelo 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.

Para a competição desse ano, o RobôCIn vai unificar suas equipes de VSSS, a equipe de abordagens determinísticas e a de aprendizado profundo. Na seção III-A descrevemos a escolha de formação, o primeiro modulo no nosso processo de decisão, em seguida descrevemos os nossos comportamentos de atacante de aprendizado por reforço e goleiro determinístico nas seções III-B e III-C respectivamente, na seção III-D descrevemos nossa implementação de posicionamento automático, que pretendemos utilizar pela primeira vez nesta competição para aumentar a autonomia da nossa estrategia.

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

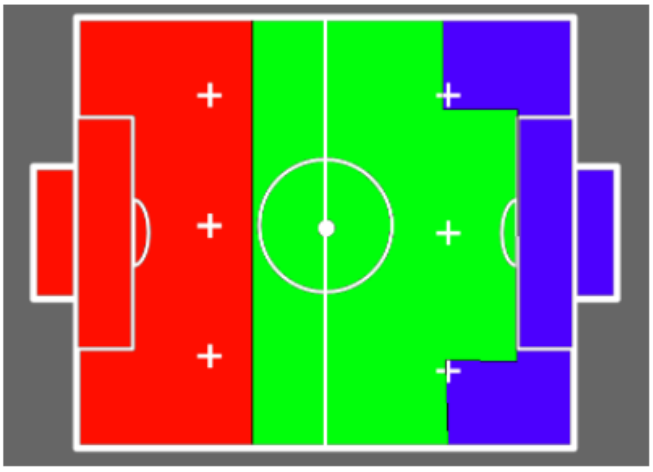


Fig. 2. Zonas de divisão do campo

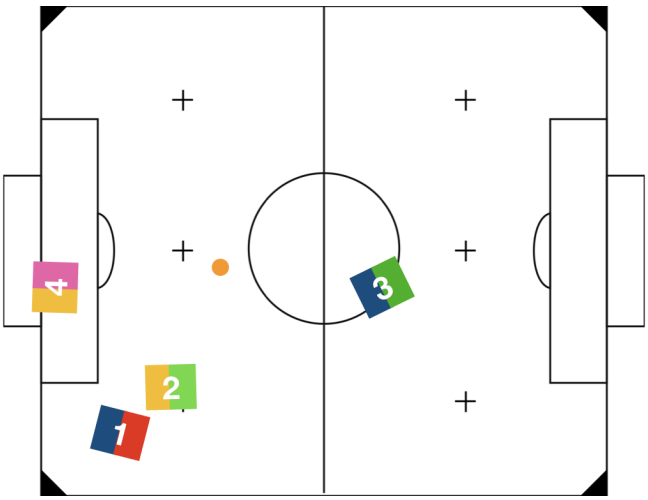


Fig. 3. Situação de jogo onde a troca de funções entre os jogadores é vantajosa

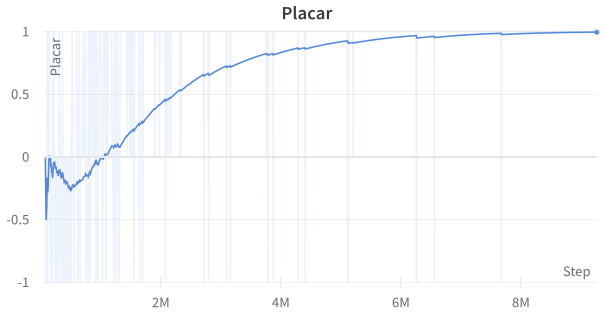


Fig. 4. Placar dos episódios durante o treinamento do atacante. Cada episódio equivale a 30 segundos do ambiente real.

conjunto de relações entre as ações que o robô pode tomar e as situações em que cada ação é tomada. Sua definição é análoga à 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.

Para gerir as diferentes combinações de comportamentos, que denominamos como formações, com pequenas variações no posicionamento de campo e nas tomadas de decisão, adicionamos uma camada extra, pois precisamos escolher primeiro qual formação seria a mais eficaz dada a condição atual do jogo.

Identificamos através de testes que formações diferentes possuem vantagens em partes diferentes do campo. Para se aproveitar dessa característica, bolamos uma formação híbrida que varia de acordo com o posicionamento da bola no campo.

Dividimos então o campo em 3 zonas, como consta na figura 2. Se a bola estiver na zona em vermelho, vamos focar numa formação que abuse de comportamentos defensivos; se a bola estiver na zona verde, escolhemos comportamentos que armem o jogo e empurrem a bola pra frente; e se a bola estiver na zona azul, escolhemos comportamentos bons na finalização.

Por fim, precisamos atribuir os comportamentos da formação escolhidos na camada anterior para os robôs. Analisamos então o posicionamento dos robôs e decidimos qual robô está mais apto a exercer cada comportamento na formação. É importante observar que a escolha de comportamentos 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 a chance para fazer o gol.

Para decidir qual robô é o melhor para cada função, ordenamos os robôs de acordo com a sua distância para o ponto objetivo daquela função. Esta técnica é utilizada da mesma forma tanto para o time com 3 robôs, quanto para o time com 5 robôs.

B. ATACANTE DE APRENDIZADO POR REFORÇO

Na unificação das equipes, decidimos manter apenas o atacante treinado com Aprendizado por Reforço [5]. Treinamos o atacante usando a técnica Deep Deterministic Policy Gradients (DDPG) [6] no ambiente de VSS-v0 do *framework* rSoccer [7]. O agente consegue uma taxa de aproximadamente 100% de acerto ao final do treinamento, como mostra a Figura 4.

O treino do agente segue o trabalho do rSoccer [7] com os mesmos hiperparâmetros e mesma quantidade de passos no ambiente. Utilizamos o *framework* Pytorch [8] para salvar e carregar os parâmetros da rede atuante no ambiente que usamos no agente de atacante durante o jogo. Um tutorial geral de como fazemos isto está descrito em `https://pytorch.org/tutorials/advanced/cpp_export.html`.

C. GOLEIRO DETERMINÍSTICO

Desenvolvemos um novo agente de goleiro para a competição. Analisando nosso jogo, verificamos que nossos agentes de goleiro não interagem bem com os atacantes, pois nossos goleiros assumiam uma estratégia muito ofensiva. Decidimos recuar o goleiro e fazer apenas um agente que segue a bola no eixo y. Este comportamento reduz em 80% a quantidade de gols que recebemos no simulador e é um comportamento que deixaremos em uso durante a competição física. Se for verificada uma queda de performance com este comportamento, adaptaremos a estratégia antiga.

O novo goleiro contará com um modelo básico de controle. Dado o objetivo e angulação instantâneos para nosso agente, é necessário mapear tais valores para velocidades de rodas que produzam o resultado desejado. Para tal, é utilizado um controle PD como mostrado nas equações 1, 2, 3, 4, que mapeiam as velocidades necessárias para que o robô saia do ponto R para o ponto O, onde θ_{ref} é o ângulo desejado, θ é o ângulo atual do robô, ϵ_θ é o erro angular, D é a distância euclidiana entre o robô e o ponto objetivo e os valores K_p^θ, K_d^θ, K_p^v, K_d^v são constantes que dependem de fatores físicos do robô.

$$\epsilon_\theta[t] = \theta[t] - \theta_{ref}[t] \tag{1}$$

$$\omega = K_p^\theta \epsilon_\theta[t] + K_d^\theta(\epsilon_\theta[t] - \epsilon_\theta[t-1]) \tag{2}$$

$$D[t] = \sqrt{(R_y[t] - O_y[t])^2 + (R_x[t] - O_x[t])^2} \tag{3}$$

$$v = K_p^v[t]\epsilon_v + K_d^v(D[t] - D[t-1]) \tag{4}$$

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 5 calcula o ângulo futuro que o robô vai estar com base no seu ângulo atual e sua velocidade angular. As equações 6 e 7 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.

$$\theta(t + \Delta t) = \theta t + \Delta t.wt \tag{5}$$

$$x(t + \Delta t) = x(t) + \Delta t.\cos((\theta.t + \theta(t + \Delta t))/2).V_x(t) \tag{6}$$

$$y(t + \Delta t) = y(t) + \Delta t.\cos((\theta.t + \theta(t + \Delta t))/2).V_y(t) \tag{7}$$

D. POSICIONAMENTO AUTOMÁTICO

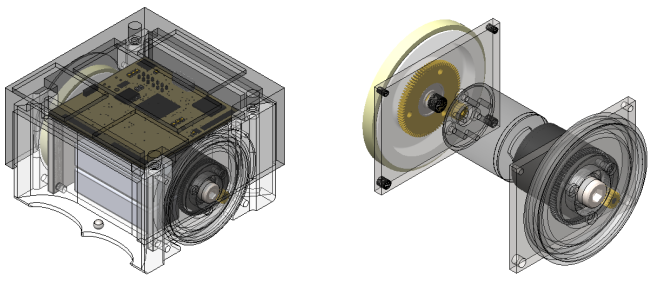
Sendo o VSSS uma categoria de robótica autônoma, a visão é que tenhamos partidas com a menor quantidade de interferência humana possível. Com o objetivo de autonomia em mente, implementamos uma estratégia de posicionamento automático para que os robôs se preparar durante as interrupções da partida na posição ideal para o reinicio, sem que seja necessária uma interferência. Abordamos esse problema como a sequência de decisão de seleção de posicionamento desejado, atribuição de posições para cada robô, planejamento de caminho e controle.

Esta implementação é um exemplo dos benefícios que a unificação de softwares entre as categorias dentro do RobôCIn nos trouxe, o desafio de posicionamento é um problema que abordamos a fundo no *Small Size Soccer*, e conseguimos rapidamente adaptar as soluções já desenvolvidas para esta categoria. A seleção de posicionamento é feita utilizando posições pré definidas para cada situação de jogo, tendo um conjunto de posições objetivo, atribuímos a cada robô uma posição utilizando algoritmo de otimização combinatória *Hungarian* [9]. Por fim, tendo os conjuntos de robôs e posições objetivo, podemos utilizar abordagens de planejamento de caminhos e controle. A abordagem usada pela equipe para o planejamento de caminho consiste em definir o caminho a ser seguido utilizando um campo de vetores unitários [10], que podem ditar a direção de movimento do robô.

IV. MECÂNICA

Durante o hiato de 3 anos de competições presenciais, uma análise sobre aspectos tanto ergonômicos quanto de performance foram realizados, através do estudo dos robôs desenvolvidos na LARC e no MIROSOT [11]. Em suma, os aspectos gerais para o desenvolvimento do novo robô permeiam três fatores: Caixa de redução, sistema de pontos de contato e otimizar as constantes físicas, estas descritas nas subseções IV-A, IV-B, e IV-C, respectivamente.

Na figura 5a, está o protótipo resultante dessa análise composto dos seguintes módulos: chassi de impressão 3D, mancais, rodas 50mm, motores DC 22mm, baterias de 300mAh, placa do sistema embarcado e capa.



(a) (b)

Fig. 5. (a) Mecânica RobôCIn 2022; (b) Unidade de potência;

A. Caixa de redução

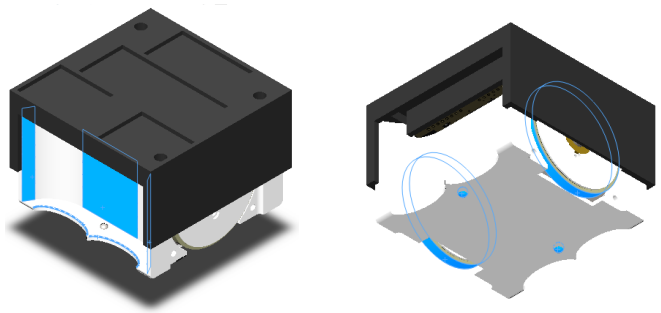
Na mecânica de 2019 [5], o eixo dos motores eram diretamente ligados à roda por conta de sua caixa de

redução inclusa. Já o atual por sua utiliza uma transmissão descentralizada do pinhão de 16 dentes do motor para a engrenagem interna da roda, em uma redução de 5:1 (Figura 5b). Essa nova disposição dos motores traz impacto direto no centro de massa do robô.

B. Pontos de contato

Para carregar a bola, o formato elipsoidal se provou mais eficiente em sua função. Com isso, foi proposto um sistema de três toques [12], e dessa maneira, quando a bola tender a sair desse sistema, a elipse inferior gera um atrito que a recentraliza (Figura 6a).

O sistema de três pontos de toque também é utilizado para as tangências do robô no campo, através esferas de rolamento entre as elipses, garantindo estabilidade nos toques junto às rodas durante a movimentação (Figura 6b).



(a) (b)

Fig. 6. (a) Toques com a bola; (b) Toques com o campo;

C. Constantes Físicas

O robô da equipe RobôCIn de 2022 permite 1380 RPM com torque de 2,16 Kg-cm. Ademais, é de suma importância o coeficiente de atrito da roda, pois é a única constante manipulável para alterar a aceleração máxima do robô [13]. Esse coeficiente pode ser alterado através da dureza e tempo de cura do material do pneu da roda.

V. ELETRÔNICA

Atualmente, o sistema embarcado desenvolvido para os robôs se baseia no microcontrolador ATmega328. Neste são utilizados dois microcontroladores, sendo um deles configurado como mestre e o outro como escravo. Eles foram escolhidos pela praticidade, tamanho, e capacidade de controlar todos os dispositivos requisitados no robô, no entanto um novo sistema baseado no microcontrolador STM32F446 está em desenvolvimento. A criação deste novo sistema se justifica pelo ganho de desempenho trazido pelo processador do tipo ARM e, também, por este anular a necessidade de serem utilizados dois microcontroladores no mesmo robô.

Para controle de potência dos dois motores DC presentes no robô, é utilizado o *driver* de motor TB6612FNG, que possui limite de corrente contínua de 1A por motor e 3A para corrente de pico. Além disso, os motores contam com *encoders* para que possa ser feito um controle de rotação

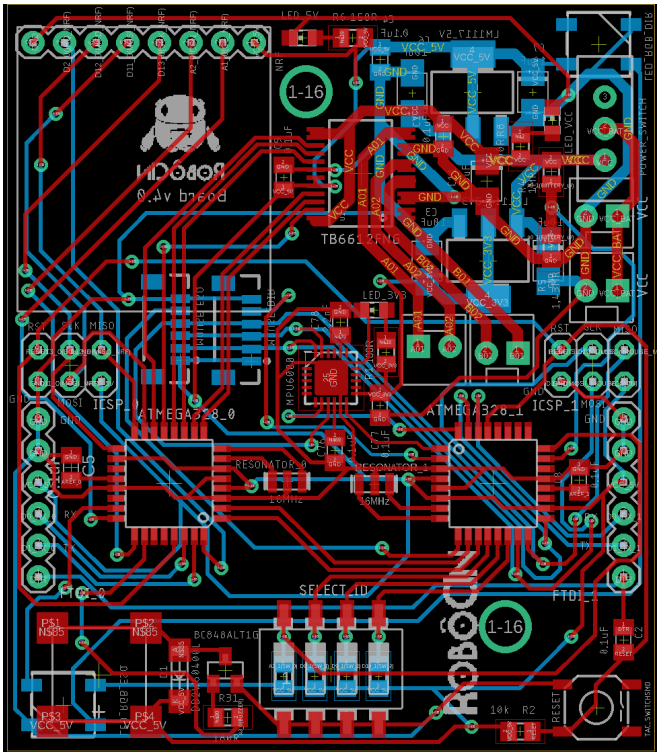


Fig. 7. Projeto do circuito principal feito através do Eagle Circuit CAD.

mais preciso por meio de *software*. Além disso, o circuito do robô também é composto pelo nRF24L01+, um módulo *wireless*, responsável pela comunicação com o computador.

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 impresso Eagle [14]. A placa principal foi desenvolvida conforme Figura 7 para compor a parte superior do robô.

VI. COMUNICAÇÃO

Para realizar a comunicação entre os robôs e o computador, na competição do VSSS em 2017 o circuito utilizado 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 VSSS de 2016. Assim, a partir da competição de 2018, a equipe utiliza apenas o módulo nRF24L01+ para comunicação.

Ao decidir usar a comunicação do tipo radiofrequência (RF), a equipe implementou um protocolo através da 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 microprocessador

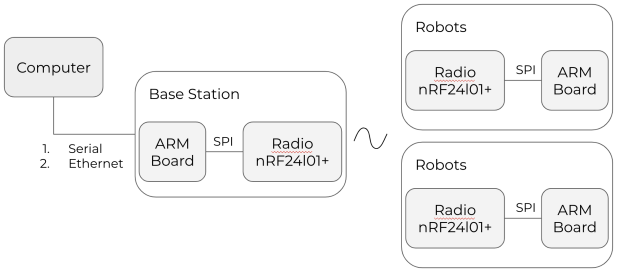


Fig. 8. Arquiteura da comunicação entre computador e robôs.

ARM 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 módulo RF que recebe os dados que o computador enviou e ao ser solicitado envia mensagens de resposta para o computador.

Para realizar a comunicação, é seguido um fluxo como está mostrado na Figura 8. Inicialmente, o computador processa a informação recebida através de imagens e converte nos movimentos que devem ser realizados pelos robôs. Para acessar o módulo nRF24l01+, o computador é conectado a uma "base station" através de uma interface serial ou Ethernet. Esta "base station" consiste de um sistema embarcado composto por uma placa ARM conectada por interface SPI a dois módulos nRF e que permite a sua comunicação com o computador. Em cada um dos robôs, um módulo nRF é utilizado para enviar ao robô as informações do computador sobre o movimento que deve realizar. Caso haja necessidade de resposta, o robô as envia para o computador também pelo seu módulo RF.

VII. CONCLUSÃO

No presente TDP, é mostrado como foi projetado e desenvolvido o sistema necessário para participar na categoria VSSS. Para isso, foi descrito um sistema de localização e detecção de objetos com objetivo de localizar a bola e os robôs em campo, o decisor de comportamentos a fim de permitir a elaboração de estratégias de jogo, o novo 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, à todo apoio dado pelos professores Edna Barros e Hansenclever Bassani. Também agradecemos a ajuda dos nossos patrocinadores: CESAR, Microsoft, Moura, HSBS, and Veroli Transportadora.

REFERÊNCIAS

[1] G. Bradski, "The opencv library." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.
[2] S. Ian, "Engenharia de software," *6a. edição, Addison-Wesley/Pearson*, 2003.
[3] G. S. Gupta and D. Bailey, "Discrete yuv look-up tables for fast colour segmentation for robotic applications," in *2008 Canadian Conference on Electrical and Computer Engineering*. IEEE, 2008, pp. 000 963–000 968.
[4] Z. Wen, J. Shi, B. He, J. Chen, K. Ramamohanarao, and Q. Li, "Exploiting gpus for efficient gradient boosting decision tree training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2706–2717, 2019.
[5] F. B. Martins, H. R. de Medeiros, L. H. C. Santos, M. G. Machado, P. H. M. Braga, R. de Azevedo Delgado, and E. N. da Silva Barros, "Robocin ia description paper," 2020.
[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.02971
[7] F. B. Martins, M. G. Machado, H. F. Bassani, P. H. M. Braga, and E. S. Barros, "rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer," in *RoboCup 2021: Robot World Cup XXIV*, R. Alami, J. Biswas, M. Cakmak, and O. Obst, Eds. Cham: Springer International Publishing, 2022, pp. 165–176.
[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[9] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.
[10] Y.-J. Kim, J.-H. Kim, and D.-S. Kwon, "Evolutionary programming-based univector field navigation method for past mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 3, pp. 450–458, 2001.
[11] C. F. Pana, N. G. Bizdoaca, I. C. Rescanu, and M. Niculescu, "Strategy planning for mirosot soccer's robot," 2008.
[12] Z. Huang, L. Chen, J. Li, Y. Wang, Z. Chen, L. Wen, J. Gu, P. Hu, and R. Xiong, "Zjunlict extended team description paper for robocup 2019," *arXiv preprint arXiv:1905.09157*, 2019.
[13] D. Otten, "Building mitee mouse iii," *Circuit Cellar INK*, 1990.
[14] Autodesk, Inc., "Eagle."