VSS LARC 2020: RoboIME em uma competição virtual - desafios e soluções

Eduardo W. P. Padovany, Victoria S. D. Ramos, Marcos A. da S. M. Filho, Bernardo F. Tepedino, Lucas M. C. Bezerra Lucas I. de A. Andrade, Pedro L. B. dos Santos, Nicolas S. M. M. de Oliveira, Henrique W. A. S. de Sousa, Leticia V. da Fonseca Arthur de A. R. dos Santos, Lucas Vinicius S. de Lima, Luiz R. L. Rodrigues, Paulo F. F. Rosa
Instituto Militar de Engenharia, Rio de Janeiro, Brasil
rpaulo@ime.eb.br
http://roboime.com.br

***Abstract*— Este artigo descreve os projetos eletrônicos, mecânicos e de software desenvolvidos pela Equipe RoboIME para participar da LARC 2020. Os conceitos gerais estão de acordo com as regras da Very Small Size League 2020. Esta será a terceira vez que a equipe participa da LARC.**

I. INTRODUÇÃO

A RoboIME é composta por alunos do curso de graduação no Instituto Militar de Engenharia. Essa é a terceira vez que a equipe participa da modalidade Very Small Size. Todos os estudantes que trabalham no projeto da VSS são membros do Laboratório de Robótica e Inteligência Computacional do IME. Esse artigo descreve as informações e avanços que a equipe obteve nesse ano.

II. DESIGN DE SOFTWARE

O software atual foi desenvolvido em Labview 2018, devido ao seu potencial de escalabilidade e às ferramentas disponíveis para a medição de entidades físicas e para análise gráfica de sistemas. O fluxo de informações do software - como descrito na figura 1 - é composto de cinco estágios: Recepção, Pré-processamento, Tomada de Decisão (Módulo de Inteligência Artificial), Controle e Transmissão.

Na Recepção, pacotes do sistema de visão são recebidos e, em seguida, pré-processados no módulo Gamestate.

No Gamestate, a informação da Visão é filtrada usando Filtro de Kalman. Foi utilizado o módulo de programação orientada a objeto do LabView, devido à facilidade para futuras modificações e padronização do projeto.

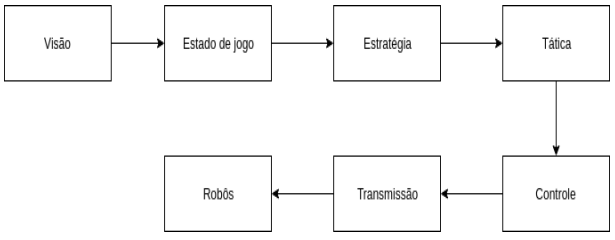

Fig. 1. Fluxo computacional do software

O estágio de controle recebe comandos do Módulo de Inteligência Artificial e executa algoritmos com o objetivo de fornecer velocidades apropriadas para o robô se movimentar e executar com precisão a trajetória calculada, ao mesmo tempo em que mantém a bola sob seu domínio até chegar ao seu objetivo. A seguir, o estágio de transmissão envia essas velocidades para cada robô, para que os valores sejam processados e a movimentação estabelecida. O estágio de Tomada de Decisão, que é composto pela Estratégia (Módulo de Personalidades) e a Tática, são descritos posteriormente.

A. Visão

O sistema de visão deve reconhecer e identificar a bola e os padrões escolhidos para os robôs. Sendo assim, optou-se por utilizar o SSL-vision e adaptar para os padrões atuais da VSS, conforme figura 2. Como a bola não muda entre as duas categorias, não houve problemas nessa parte, mas foi necessário mudar os padrões dos robôs e do campo para que o sistema os reconheça. Além disso, foi necessária uma alteração no SSL-vision que viabilizasse o reconhecimento dos robôs adversários apenas pela cor central.

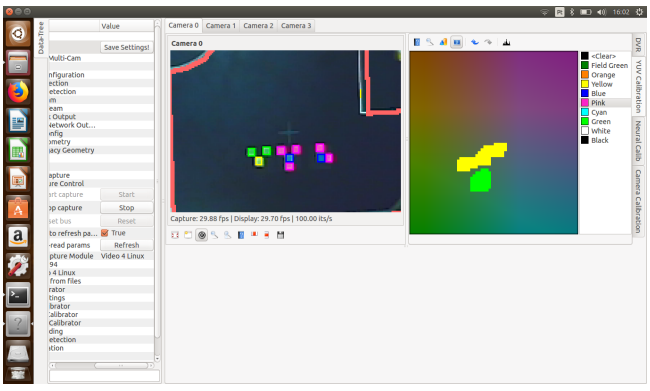

Fig. 2. SSL-Vision adaptado para VSS.

B. Gamestate

1) Estimativa usando Filtro de Kalman: O filtro de Kalman utiliza os dados fornecidos pela visão para estimar o próximo estado de jogo, gerando informações[1] como estimativas de velocidadade. Ele também filtra os dados de ruído no processo, estabilizando o estado de jogo obtido.

C. Sistema de IA

Nosso atual sistema de IA segue a arquitetura STP *(Skills, Tatics and Plays)* e com as informações recebidas pelo Gamestate, determina pelo estado de jogo qual *Play* deve ser executada. Em cada *Play* existem 3 personalidades principais: Atacante, Defensor e Goleiro, os quais possuem diferentes *Tatics* em cada *Play*. Todas as personalidades possuem um papel crucial para funcionamento correto da jogada, conforme será descrito abaixo.

- Atacante:
  O atacante tem como objetivo marcar gols, enquanto evita o time inimigo. Ele age de maneira a tentar conseguir a posse da bola e levá-la para posições onde a chance de gol é maior. Uma vez que a bola esteja em posição propícia, é estrategicamente estabelecido um alvo, utilizando informações como posição dos robôs inimigos e da bola. Então é executada uma trajetória para chegar à bola e levá-la ou chutá-la até o gol adversário, sendo essa trajetória variável a medida que o estado de jogo muda, podendo sofrer alterações para desvio de obstáculos ou reposicionamento da bola.

- Defensor:
  O robô defensor tem o objetivo de impedir a bola de chegar no goleiro, não abandonando o lado aliado do campo. Assim, quando a bola está no lado adversário, ele apenas acompanha de longe para evitar contra-ataques. Se a bola estiver no lado aliado do campo, o robô irá servir como suporte para o robô atacante, para ajudar no controle da bola ou chutá-la de volta para o outro lado, ao mesmo tempo que impede o avanço adversário.

- Goleiro:
  A principal função do goleiro é interceptar a bola antes que ela alcance o gol. Desse modo, se a bola estiver suficientemente distante do goleiro ele será enviado para a posição a qual a bola está direcionada no instante atual. Caso esta posição esteja fora do gol, o goleiro permanecerá na extremidade do gol mais próxima à posição final estimada da bola. Se a bola estiver próxima o suficiente do robô, ele tentará direcioná-la para longe da área do gol, evitando jogadas indesejadas do adversário.

- Personalidade Dinâmica:
  Conforme a situação do jogo muda, as vezes mostra-se vantajoso trocar entre os robôs as funções que estão assumindo, para chegar à bola mais rápido ou fazer com que um robô assuma a função de outro que está muito distante por exemplo. O módulo de personalidade dinâmica utiliza critérios como distância à bola de cada robô e a posição dos inimigos. Assim, ela determina qual personalidade cada robô assumirá em cada momento do jogo, levando em conta as personalidades de atacante e defensor, já que o goleiro evitará trocar de posição.

D. Planejamento de trajetória

Com o intuito de evitar colisões com outros robôs, é utilizado o algoritmo ERRT[2],que é um processo de amostragem de posições navegáveis de um ambiente através de um grafo do tipo árvore. O funcionamento consiste em expandir a árvore de modo aleatório de um nó inicial (raiz) até que uma de suas ramificações alcance uma posição final. Como cada nó possui informação de seu nó antecessor, a rota é traçada desta posição até a posição seguinte, de modo a atingir a posição final através do menor caminho possível. O algoritmo ERRT *(Extended Rapidly-exploring Random Tree)* é uma melhoria do algoritmo RRT visando à redução da extensão das rotas planejadas.

E. Controle

Para o controle da posição foi utilizado PID, levando em consideração que o robô possui duas frentes e trajetória direcional. Assim, é definido um caminho que o robô deve fazer, seja para atacar ou defender, conforme figura 3.Para tentar seguir esse caminho, usa-se uma malha PID na correção dos erros, levando em conta a velocidade angular e a tangencial.

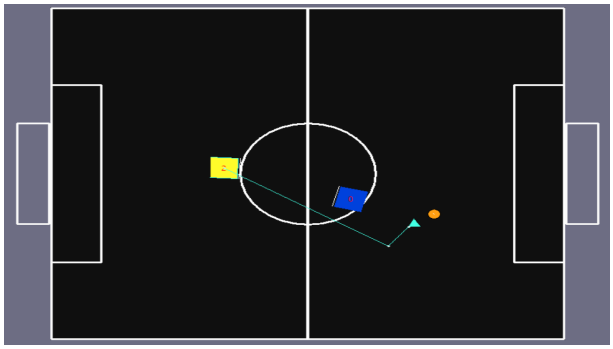

Fig. 3. Trajetória definida para o robô amarelo.

F. Ambiente de testes

A estratégia adotada e o módulo de controle são testados no simulador FIRASIM[3], como mostra a figura 4.Este permite emular os estados de jogo e testar a estratégia, a movimentação e o posicionamento dos robôs em um ambiente "ideal", onde fatores práticos, como a luz do ambiente ou bateria com pouca carga, não interfiram no jogo.

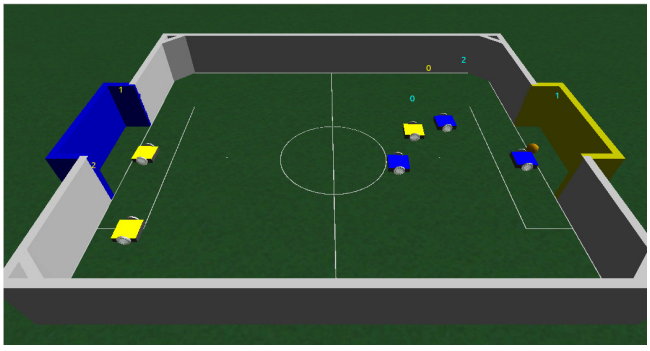

Fig. 4. Teste de uma partida no simulador.

III. Projeto Elétrico

A. Design da Placa

Para a Larc 2020, foi desenvolvida uma nova placa no Altium Designer, conforme figura 6. Em relação a placa do ano passado, essa nova possui entrada para um display de 7 segmentos com a função de indicar valores, tal como a identidade do robô. Além disso, foi inserido um expansor de porta (STMPE811QTR) e um botão que permite alterar o valor do display, fazendo com que não seja preciso conectar o robô à um computador para alterar sua identidade.

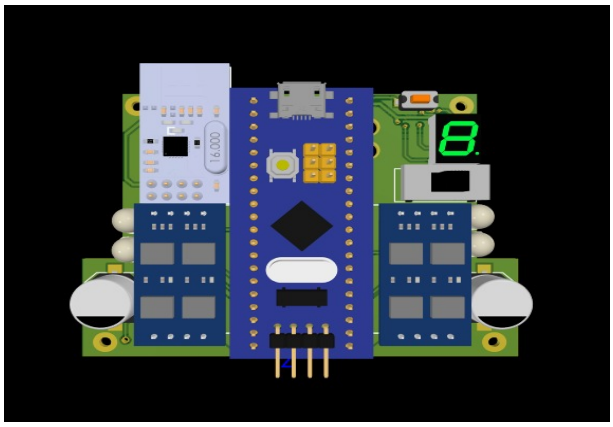

Fig. 5. Design da nova placa

B. Firmware

O firmware é feito em C++ e possui duas abordagens: conceitos de abstração e orientação do objeto. As classes abstratas envolvem os sensores e atuadores, enquanto os recursos do microcontrolador são as classes concretas, assim como a interrupção. O firmware está disponível tanto para o estudo quanto para possíveis melhorias futuras, além da possibilidade de utilização em outros projetos.

O microcontrolador utilizado é o STM32f103C8T6- Blue Pill, usando a bibliotaca HAL. A linguagem C++ foi escolhida tanto pelas suas vantagens, como o polimorfismo, quanto por ser conteúdo de estudo de cadeiras na graduação de alguns alunos da equipe.

A principal função do firmware é enviar e receber informação do robô para o computador e vice-versa. A comunicação ocorre em um loop infinito, e em cada ciclo ocorre uma interrupção, na qual é feita o controle do robô.

O controle no firmware é feito da seguinte forma: o computador envia a velocidade desejada para o robô, quando a comunicação é bem sucedida, ocorre uma interrupção no programa e placa eletrônica executa o comando de velocidade nas rodas através de PWM. O robô possui um encoder óptico para medir a velocidade da roda. Através da velocidade real e velocidade desejada, é feito uma malha de controle PID em ciclo de 13 ciclos, que é a quantidade máxima que o firmware consegue executar em um ciclo de interrupção, conforme os cálculos feitos pela equipe.

IV. Projeto Mecânico

Os robôs foram projetados de forma simples e compacta, a fim de garantir maior mobilidade, agilidade e facilidade de condução da bola. A estrutura dos robôs é cúbica, com dimensões máximas de 75x75x75mm.

A. Design

O software utilizado para a confecção do design 3D em CAD dos robôs foi o SolidWorks. Para aumentar a capacidade dos robôs de controlarem a bola, foram adicionadas cavidades ligeiramente rasas nas partes frontal e traseira, que podem ser vistas na figura 6. O formato desta cavidade é parabólico, de forma que quando a bola colide com robô, ela tende a ir em direção do foco da parábola, favorecendo, assim, o controle da bola. Os robôs possuem duas rodas laterais que garantem a movimentação destes. No entanto, tal sistema permite a mudança de direção apenas pela diferença de velocidade entre as rodas.

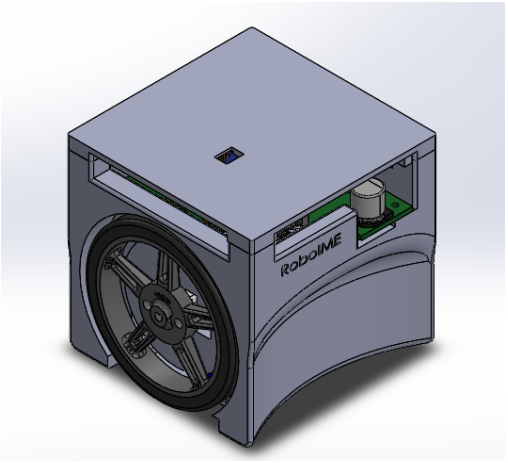

Fig. 6. Modelo CAD do robô.

O robô possui duas tampas, uma inferior e superior. Como mostrado na figura 7, o encaixe delas é feito através de ímãs, o que garante facilidade na desmontagem caso haja necessidade acessar algum componente interno. Os imãs da parte superior também são colocados de forma que cada robô possua apenas uma tampa.

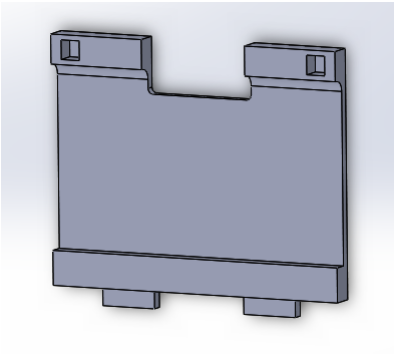

Fig. 7. Tampa inferior atual

Também foi desenvolvimento de um alojamento para os fios da bateria, como mostrado nas figura 8.

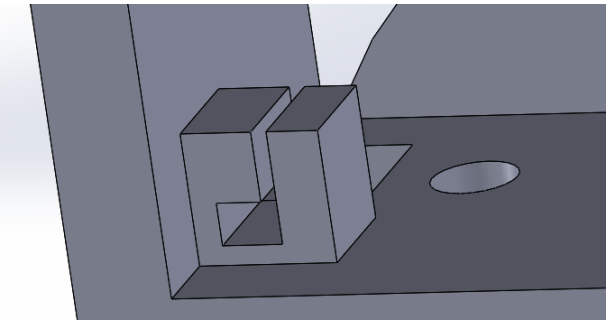

Fig. 8. Alojamento para os fios da bateria

B. Manufatura

Com o design 3D em CAD pronto, a equipe optou pelo uso de impressora 3D com material ABS para a impressão das peças. Tal opção justifica-se pela vantagem na substituição e troca de componentes quebrados do robô.

REFERENCES

[1] Bzarg http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/ , 2015

[2] Ramon Jansen. Waypoint navigation with obstacle avoidance for mav's. 2016.

[3] FIRASim. https://github.com/robocin/FIRASim