TDP RoboIME VSS: LARC 2019

***Abstract*— Este artigo descreve os projetos eletrônicos, mecânicos e de software desenvolvidos pela Equipe RoboIME para participar da LARC 2019. Os conceitos gerais estão de acordo com as regras da Very Small Size League 2019. Esta será a segunda vez que a equipe participa da LARC.**

I. INTRODUÇÃO

RoboIME é um time de Very Small Size do Instituto Militar de Engenharia, IME, no Rio de Janeiro. Essa é a segunda vez que a equipe participa da competição. 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, abordando os seguintes tópicos:

II. DESIGN DE SOFTWARE

Nosso software atual foi desenvolvido em Labview 2017 devido ao seu potencial de escalabilidade e às ferramentas que ele dispõe 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.

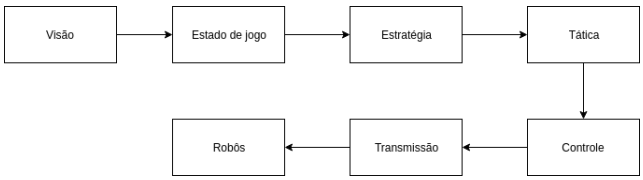

Fig. 1. Fluxo computacional do software

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 e chegar ao seu objetivo. Posteriormente, no estágio de transmissão, o módulo de transmissão envia essas velocidades para cada robô, para que então possam processar esses valores e executar a movimentação estabelecida. O estágio de Tomada de Decisão - que é composto pela Estratégia (Módulo de Personalidades) - e a Tática são descritos posteriormente.

*This work was not supported by any organization

[1]Albert Author is with Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7500 AE Enschede, The Netherlands albert.author@papercept.net

[2]Bernard D. Researcheris with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435, USA b.d.researcher@ieee.org

A. Visão

O sistema de visão deve reconhecer a bola e os padrões escolhidos para os robôs com o intuito de identificá-los. Sendo assim, optamos por utilizar o ssl-vision e adaptar seus padrões para o VSS, conforme figura 2. Como a bola não muda entre as duas categorias, não tivemos 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 que viabilizasse o reconhecimento dos robôs adversários sem depender de seus padrões.

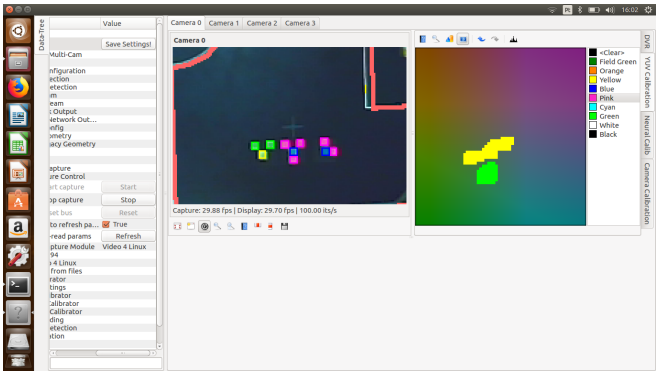

Fig. 2. SSL-Vision adaptado para VSS

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 e filtrando os dados de ruído no processo, estabilizando o estado de jogo obtido.

C. Sistema de IA

1) Personalidades: A atual arquitetura do nosso sistema de IA recebe informações do Gamestate e determina de acordo com esse estado de jogo qual estratégia utilizar. Existem três personalidades principais - Atacante, Defensor e Goleiro -, que comportam-se de maneira diferente de acordo com a estratégia adotada. Cada personalidade tem um papel essencial no jogo e executam suas funções de maneiras diferentes como serão descritas abaixo, podendo ser intercambiadas entre cada robô.

D. Módulo Tático

- Atacante:
  O atacante tem como objetivo final e principal marcar gols, portanto ele age de maneira a tentar conseguir posse de bola e posicioná-la em locais onde seja mais provável a chance de pontuar. Uma vez que a bola esteja em posição propícia, é estabelecido um alvo estrategicamente utilizando informações como posição dos robôs inimigos e da bola e então executa a trajetória definida pra chegar à bola e levá-la até o gol adversário, sendo essa trajetória mutável a medida que o estado de jogo muda, podendo sofrer alterações para desvio de obstáculos ou reposicionamento da bola.

- Defensor:
  Nosso robô defensor terá o objetivo de impedir a bola de chegar no nosso goleiro, para tal ele não abandonará o lado aliado do campo, sendo assim, quando a bola estiver no lado adversário, ele apenas acompanhará de longe para evitar contra-ataques. Se a bola estiver no lado aliado do campo, o robô irá servir como suporte para nosso robô atacante, para ajudar no controle da bola 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 que a bola está direcionada no instante atual. Se esta posição estiver fora do gol, o goleiro permanecerá na extremidade do gol mais próxima à posição final estimada da bola. Estando a bola próxima o suficiente do robô, ele tentará direcioná-la para longe da área do gol, evitando jogadas indesejadas do adversário.

- Personalidade Dinâmica:
  À medida que a situação do jogo muda, muitas vezes mostra-se vantajoso trocar entre os robôs as personalidades que estão assumindo com o objetivo de chegar à bola mais rápido ou fazer com que um robô assuma a função de outro, que seja essencial em determinados estados de jogo, quando este está preso, por exemplo. O módulo de personalidade dinâmica utiliza de critérios como distância à bola de cada robô, posição dos inimigos e disputa entre robôs para determinar qual personalidade cada robô assumirá em cada estado de jogo.

E. Planejamento de trajetória

Com o intuito de evitar colisões com outros robôs, o algoritmo ERRT é utilizado [2], 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.

F. Controle

Para o controle da posição foi utilizado PID, levando em consideração que o robô possui duas frentes e trajetória direcional. Para isso, é definido um caminho que o robô deve fazer, seja para atacar ou defender, conforme figura 3. O robô irá tentar seguir o caminho, sendo que a medida que não conseguir irá ser feito um correção na sua trajetória através de uma malha PID, levando em conta a velocidade angular e tangencial.

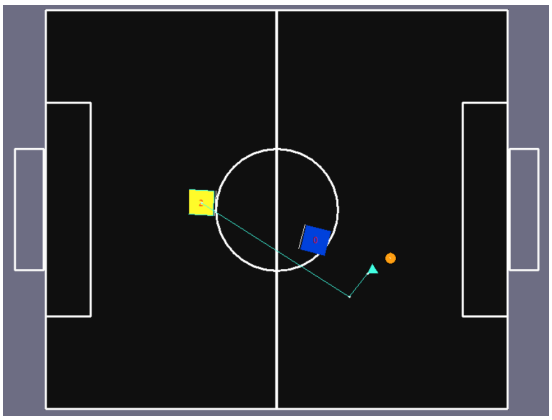

Fig. 3. Trajetoria definida para o robô amarelo

G. Ambiente de testes

A estratégia adotada e módulo de controle são testadas no simulador do projeto VSS-SDK [3], do SirLab, onde o mesmo foi adaptado para rodar em Windows, ambiente onde o Labview opera, como mostra a figura 4. O simulador permite emular os estados de jogo e testar a estratégia e movimentação dos robôs num ambiente "ideal"onde fatores práticos, tais como 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 2019, desenvolvemos uma nova placa no Altium Designer, conforme figura 5. Em relação a placa do ano passado, essa nova possui um display de 7 segmentos com a função de indicar valores, tal como a identidade do robô se comunicar com a placa de comunicação. Além disso, foi adicionado um expansor de porta (STMPE811QTR) e um botão de tal forma que seja possível alterar os valores do display e assim mudar a identidade do robô, sem que seja necessário conectá-lo no computador. Por fim, foi adicionado um giroscópio (MPU-9250), a fim de que o controle da trajetória do robô seja feita pela velocidade angular e tangencial do robô.

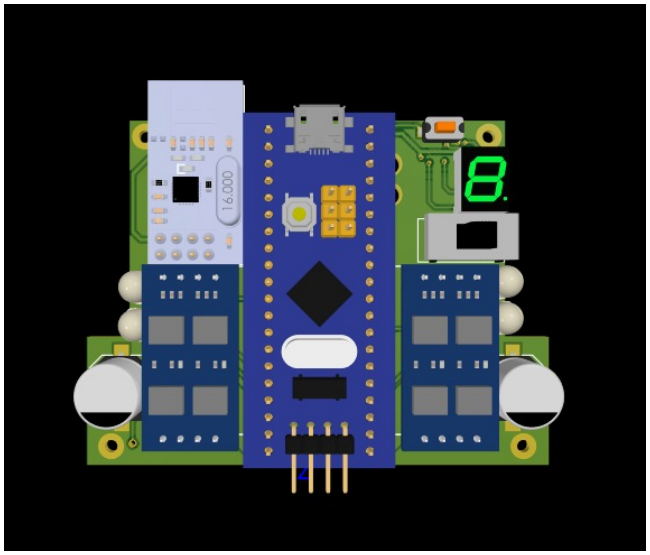

Fig. 5. Design da nova placa

B. Firmware

Para Larc 2019, o time desenvolveu um firmware em C++, usando conceitos de abstração e orientação ao objeto. As classes abstratas representam os sensores e atuadores e os recursos do microcontrolador são as classes concretas, tal como a interrupção. Dessa forma, o firmware fica aberto para ser estudado pelos membros e pode ser desenvolvido e aprimorado no futuro, ou até utilizado em outros projetos.

O microcontrolador utilizado é o STM32f103c8t6, conhecido como Blue pill, e a biblioteca importada é a HAL. A linguagem C++ foi escolhida, devido ser uma das cadeiras de estudo na faculdade dos integrantes, além das vantagens trazidas pelo tipo dessa linguagem, tal como o polimorfismo.

As principais funções no firmware são de enviar e receber informação do robô para o computador e vice-versa. A comunicação ocorre em um loop infinito, e a cada ciclo, ocorre uma interrupção, na qual é feito o controle do robô.

O controle no firmware do robô é feito da seguinte forma: o computador envia a velocidade desejada pro 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

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

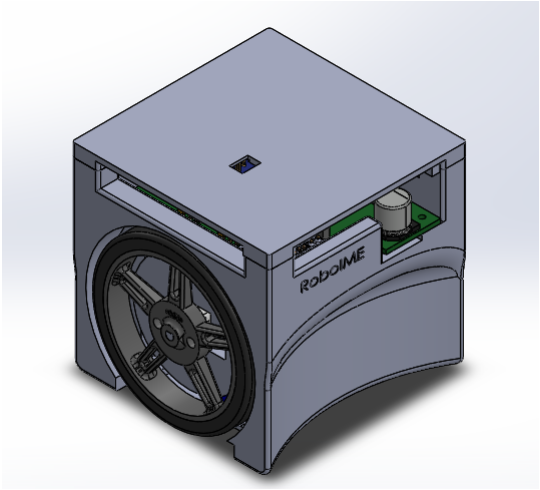

Fig. 6. CAD 3D do robô

A. Design

O software utilizado para a confecção do design 3D em CAD dos robôs (projeto de cada peça e junção destas) foi o SolidWorks. Para aumentar a capacidade dos robôs de controlarem a bola, foram adicionadas cavidades ligeiramente rasas nas partes frontal e traseira. O formato desta cavidade é parabólico, e, com isso, quando a bola colide com robô, ela tende a ir em direção do foco da parábola, que está estrategicamente posicionado, favorecendo, assim, o controle da bola por parte do robô. 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. Em ambas as tampas (inferior e superior) o encaixe é feito através de ímãs, o que garante facilidade na desmontagem caso haja necessidade acessar algum componente interno.

B. Manufatura

Com o design 3D em CAD pronto, haja vista a simplicidade dos robôs da VSS e a facilidade na produção, 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. A fim de adquirir o estágio atual, foram realizadas diversas alterações na estrutura do robô, como por exemplo,

a utilização de ímãs na tampa inferior e o desenvolvimento de um alojamento para os fios da bateria, como mostrado nas figuras a seguir.

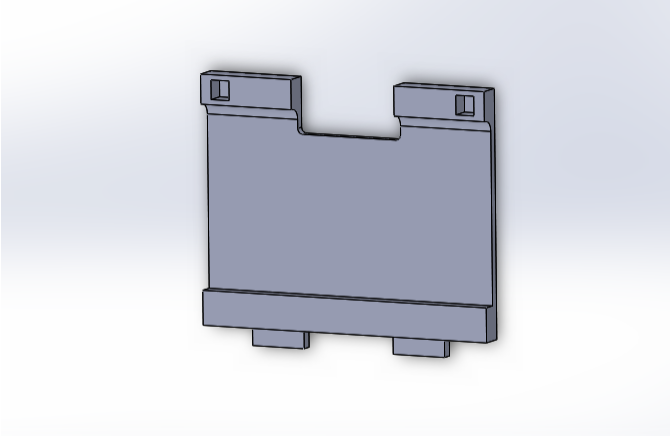

Fig. 7. Tampa inferior atual

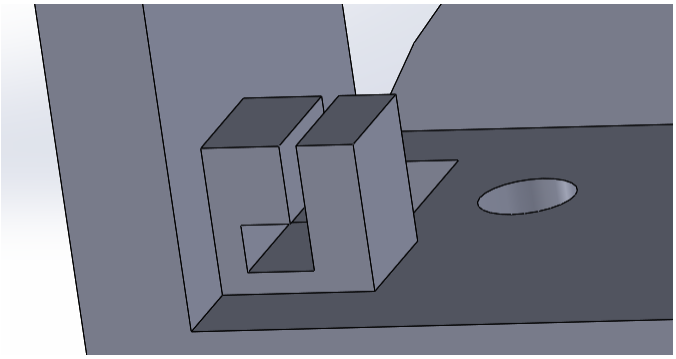

Fig. 8. Alojamento para os fios da bateria

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] VSS-SDK. https://github.com/VSS-SDK/VSS-SDK, 2019