Very Small Size Soccer (VSSS) RobotBulls - Time e simulador

Alexandre B. Lugli[1], Egídio R. Neto[2], Leonardo E. Nogueira[3], João Henrique S. Delfino[4], João Victor de O. G. Ribeiro[5], Rodrigo O. Salles[6], Wanderson E. Saldanha[7]

Instituto Nacional de Telecomunicações - Inatel
[1]baratella@inatel.br, [2]egidio.neto@inatel.br, [3]leonardoeufrazio@inatel.br, [4]joao.h@gec.inatel.br,
[5]joao.v@ges.inatel.br, [6]rodrigo.salles@gea.inatel.br, [7]wsaldanha@inatel.br

***Abstract*— Este trabalho descreve o desenvolvimento do time da equipe de robótica do Instituto Nacional de Telecomunicações (INATEL), RobotBulls, para a categoria *IEEE Very Small Size Soccer*. As informações aqui apresentadas envolvem os aspectos de eletrônica, mecânica, visão computacional e princípios de inteligência artificial, bem como as novas implementações para o modelo de competição exclusivo adotado para esse ano. O objetivo desse documento é compartilhar o conhecimento adquirido através das pesquisas realizadas para obter-se, no final, um time com características que poderão ser referência para outras equipes.**

***Palavras-chave*: Controle, Robótica, Simulador, Visão Computacional.**

I. INTRODUÇÃO

A categoria *IEEE Very Small Size Soccer (VSSS)* [7], tratada nesse estudo, motiva pesquisas na área de robótica devido à sua complexidade. Tal dificuldade se dá por conta da abrangência de diferentes conceitos relacionados a *hardware, software, firmware* e, agora com o modelo de competição remoto, também conceitos de redes de computadores.

Esse estudo foi realizado pela equipe de robótica e inteligência artificial do Instituto Nacional de Telecomunicações - INATEL, a RobotBulls. A equipe atualmente possui quatro pesquisadores e três professores que orientam os trabalhos na categoria, o que vale ressaltar é a presença de alunos de diferentes cursos de graduação na equipe.

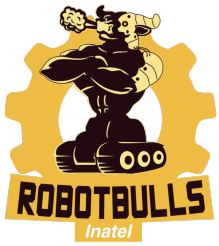

Fig. 1. Logo da equipe RobotBulls

A equipe RobotBulls participa de competições oficiais desde a LARC (Competição Latino-Americana de Robótica - Latin American Robotics Competition) de 2015, na cidade de Uberlândia/MG. A partir da edição de 2016 da LARC, realizado em Recife/PE a equipe conseguiu resultados positivos em sua pesquisa e se tornou conhecida ao terminar como vice-campeã. Em 2017, a LARC aconteceu em Curitiba/PR e a equipe RobotBulls encerrou a competição invicta, sendo consagrada campeã. Na última edição dessa competição, em 2019, a RobotBulls conquistou o primeiro lugar novamente, se mantendo entre os melhores times.

O detalhamento dos estudos é realizado pelos capítulos seguintes. O capítulo II apresenta os recursos utilizados na parte eletrônica, sendo este responsável pela descrição das características referentes aos componentes e circuitos necessários. Também destaca a mecânica, responsável pela parte estrutural do rôbo e disposição das partes que o constituem. No capítulo III, é reportada a Visão Computacional, que cuida da identificação e localização dos rôbos, além do tratamento da imagem capturada pela câmera. No capítulo IV, é descrito o desenvolvimento do software e a estratégia é apresentada com a metodologia de testes e tomada de decisões utilizados pela equipe, bem como a forma de utilizar a inteligência artificial embarcada para navegação dos robôs. O capítulo V apresenta o simulador FIRASim, o qual será utilizado na competição remota deste ano. Além disso, é apresentado seu funcionamento e seus benefícios. O capítulo VI mostra os testes e resultados obtidos ao longo do estudo.

II. DESENVOLVIMENTO DE *HARDWARE*

Nesse capítulo, serão apresentados os principais componentes e circuitos utilizados pelos robôs do time RobotBulls, bem como sua estrutura básica, motores e rodas, sendo estes referenciados a seguir em eletrônica e mecânica.

A. Eletrônica

A eletrônica básica do robô é composta da seguinte maneira:

- *Step-down*: regulador de tensão ST1S41 [13], opera na faixa de 4V a 18V, com saída ajustada para operar em 6V.
- Microcontrolador: STM32WB55CE [14] multiprotocolo wireless de 32 bits com *FPU* (*float point unit* - unidade de ponto flutuante), com frequência de processamento de 64 MHz, com transmissor RF que suporta *Bluetooth* 5 e IEEE 802.15.4-2011 *PHY* (*physical layer*) o que facilita a comunicação.
- Ponte-H: IHM13A1, ferramenta de desenvolvimento de gestão de energia que trabalha com PWM (*pulse width modulation*) e serve como placa de expansão de driver de motor brush DC de baixa tensão para ser usado com placa núcleo STM32.

- *Encoder*: O ENX 10 EASY/QUAD [6] converte os movimentos de giro do motor em pulsos elétricos, trabalha com 3 canais fazendo um máximo de 1024 leituras por giro e pode fornecer a leitura até uma velocidade máxima de 30000 rpm (rotações por minuto).

B. Mecânica

Aqui serão descritas as informações estruturais do robô e as características dos motores e rodas utilizados no projeto.

1) Chassi: O chassi foi planejado de modo que o robô ficasse o mais próximo ao chão possível com a maior parte de seu peso (motores) concentrada na parte mais inferior da estrutura, baixando seu centro de massa ao máximo a fim de garantir mais estabilidade na navegação. Para desenvolver o projeto foi utilizado o software *SolidWorks*. Através dessa ferramenta foi possível simular as dimensões da estrutura, do motor (com encoder), da bateria e da placa. O material utilizado para confecção do rôbo através de impressão 3D foi o PETG-XT que apresenta alta resistência a choques mecânicos, evitando assim rachaduras em sua estrutura, além de não sofrer alterações de cor e deformação. A Figura 2 mostra estrutura básica do robô.

Fig. 2. Escopo do projeto do chassi

2) Motores e rodas: O motor escolhido para ser agregado ao projeto foi o DC-MAX22S [5] que possui rotação de 7030rpm e torque nominal 11,5mNm. Devido à disposição do encoder, alogando ainda mais seu comprimento, houve a necessidade de deslocar o motor do eixo central do robô, o que acarretou na necessidade de engrenagens que, além de corrigir o posicionamento do eixo, ainda reduzem o giro e ajustam a rotação das rodas que, por sua vez, foram pensadas para evitar a derrapagem nas curvas mais bruscas visando aumentar a velociade sem perder precisão. Sendo assim, o material escohido foi o poliuretano líquido (poly 74-20) [8] que ao enrijecer adere bem ao campo, evitando giros em falso do motor.

III. Visão Computacional

As imagens capturadas pela câmera durante a partida são tratadas e identificadas através de métodos que são descritos abaixo.

A. Câmera

A câmera utilizada é ELP- USBFHD01M-MFV. Ela possui uma resolução de 480*pixel*, com uma taxa de quadros de 120fps (quadros por segundo - *frames per second*). O conector de sua entrada é 2.0. Junto a ela, utilizou-se uma lente de 2.8-12mm varifocal.

B. Transformação Linear 2D

Um dos processos mais custosos computacionalmente dentro do módulo da visão, é a transformação da imagem da câmera para o plano de duas dimensões. Esta transformação é de extrema importância, pois aumenta significativamente a precisão da identificação dos objetos na imagem, sendo possível remover eventuais distorções na lente da câmera, selecionando apenas a área útil. Um exemplo de transformação pode ser visto na Figura 3.

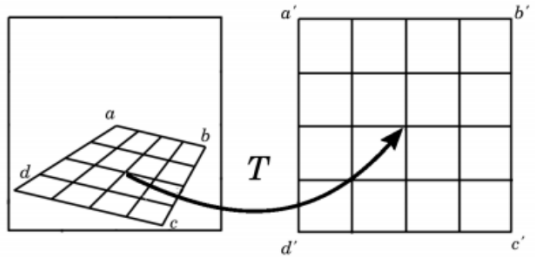

Fig. 3. Método de conversão de 3D para 2D implementado

C. Filtro de ruídos, conversão HSV e identificação

Após a filtragem da imagem, ela é convertida do espaço de cores RGB (vermelho, verde e azul - *red, green and blue*) para o HSV (matriz, saturação e valor - *hue, saturation and value*), a fim de localizar o centroide dos objetos e agrupá-los. A figura 4 mostra uma *tag* com seus respectivos centros e pontos de identificação.

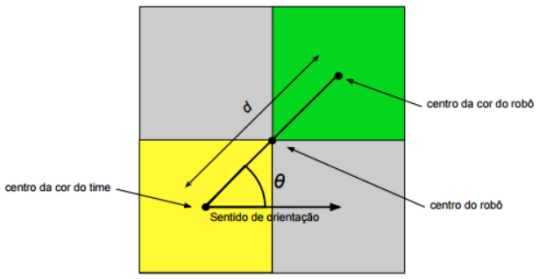

Fig. 4. Sistema de cores para identificação e orientação dos robôs

Essa filtragem é feita através de uma matriz de convolução. A interface que permite o ajuste das cores para identificação das informações pertinentes a localização dos objetos pode ser vista na Figura 5.

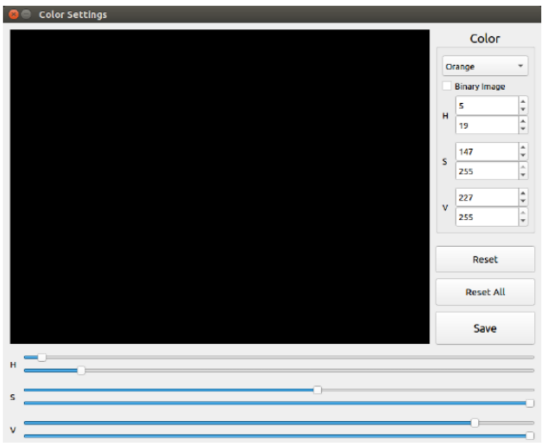

Fig. 5. Ajuste das cores para HSV

Nessa interface é possível visualizar o campo e calibrar as cores de acordo com a incidência de luz, pois em determinados pontos, uma cor pode perder ou ganhar saturação em função da iluminação no local, o que é um problema, visto que ao alterar a intensidade de uma cor o comportamento dos robôs pode mudar. Visando corrigir isto, antes de iniciar a partida, sempre se calibra todas as cores em todas as posições do campo em que é possível perceber que há mudança de luminosidade. Sendo assim, o risco de perder-se a identificação de uma cor durante a partida é muito menor e, eventualmente os robôs poderão navegar com mais acertividade.

D. Mapeamento

A fim de melhorar a precisão das delimitações do campo, desenvolveu-se um programa que recebe a imagem da câmera e permite ao usuário certificar quais partes do campo estão sendo captadas, podendo redimensionar o campo via *software*. A tela do programa é apresentada na figura 6.

Fig. 6. Mapeamento do campo através do programa

Esse foi um programa que sentiu-se a necessidade de desenvolver para resolver determinados problemas relacionados às linhas do campo e posicionamento de câmeras, visto que durante a competição a fixação da câmera pode influenciar no tratamento da imagem e, consequentemente no comportamento dos robôs.

IV. DESENVOLVIMENTO DE *SOFTWARE*

A tomada de decisões é baseada em inúmeros fatores, os quais permitem um melhor desempenho dos robôs em relação às ações executadas. Dessa forma, os conceitos de inteligência artificial são aplicados a fim de alcançar os objetivos dentro de uma partida.

A. Estratégia

A estratégia tem como principal objetivo obter um melhor desempenho por parte do time. Sendo assim, o método adotado pela equipe visa analisar as condições de jogo e ,em determinados momentos da partida e definir a melhor formação. Tal análise leva em consideração o placar da partida, o que permite definir uma formação mais defensiva ou ofensiva, dependendo das condições de jogo. Ainda assim, é feito um estudo acerca de cada equipe, de modo a não ser surpreendido em um início de partida.

B. Máquina de estado

Para processar a estratégia que o time deseja colocar em jogo em determinado momento, é utilizada uma máquina de estados [2]. Em todos os casos, a bola que está em campo é colocada como prioridade em relação à todos os jogadores identificados. Tendo em vista, é atribuído a cada jogador do time uma função. Com o intuito de otimizar o time, novos estudos estão sendo realizados e trabalhados, a fim de obter métodos mais eficientes. O conceito de árvore de comportamento é um dos métodos a serem aplicados, uma vez que melhora, consideravelmente, o desempenho do time.

C. Software

Para a visualização dos robôs pela câmera e melhor entendimento de sua movimentação no campo, bem como a escolha das jogadas - respeitando as condições definidas pela máquina de estados apresentadas no tópico anterior - foi criado um software no qual é possível observar a movimentação da bola e a trajetória do robô até seu ponto futuro a fim de realizar análises do movimento e, mediante à resposta corrigir os erros antes mesmo das partidas acontecerem. Também é possível selecionar a formação desejada para os robôs, observando e entendendo o melhor tipo de estratégia a ser usada durante o jogo ou antes do início da partida.

D. Navegação

Entre os diversos tipos de navegações estudadas e testadas, a equipe continua utilizando o *UVF (Unity Vector Field)* [1], dessa forma a navegação dos robôs apresenta um resultado esperado como planejado durante os testes. A figura 7 demonstra o seu funcionamento.

Com a finalidade em manter a estabilidade da trajetória dos robôs para alcançar o objetivo final, foram usados dois tipos de controle PID (Proporcional Integral Derivativo), um deles é via *software*, configurado pelo código da inteligência que usa a angulação do robô como erro, e outro via *firmware*, através da leitura do *encoder*.

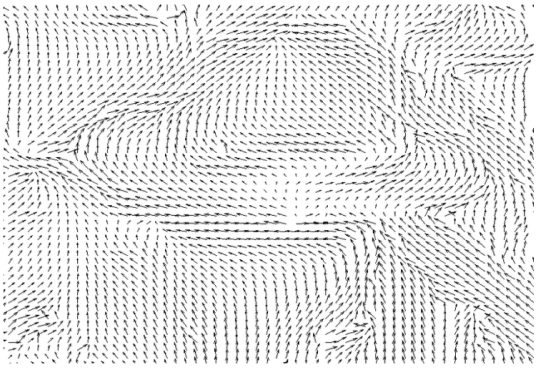

Fig. 7. Plotagem de um sistema *UVF*

V. SIMULADOR

O *software* FIRASim [3] é um simulador desenvolvido pela ParsianLab, que possibilita a execução de uma partida de *VSSS* com todos os requisitos necessários. Tal ferramenta será utilizada nessa competição e contribui para o desenvolvimento da equipe em relação às pesquisas realizadas. Isso se dá devido ao aumento da produtividade na elaboração de novas estratégias, pois não se fez necessário ter uma estrutura de *hardware* montada a fim de efetuar inúmeros testes. Dessa forma, é possível prever quaisquer possíveis problemas relacionados a lógica da tática pré estabelecida pela equipe.

A. Comunicação

A comunicação [10] é um dos fatores mais relevantes, visto que a interação e o aproveitamento dos serviços e dados fornecidos pelo simulador, só podem ser manipulados a partir dela. A estrutura se torna semelhante a arquitetura de comunicação cliente/servidor. Para isso, faz-se uso do *Protocol Buffers* [4], sendo este um método para serialização de dados estruturados desenvolvido pelo Google, semelhante ao XML, porém seu desempenho é notavelmente melhor.

Primeiramente, o escopo da mensagem é definido por meio de arquivos *".proto"*, os quais contém as informações referentes às mensagens enviadas pelo simulador, por meio de pacotes via *Multicast*. As estruturas de dados a serem serializadas são descritas, de modo a identificar cada campo (dado) com seu respectivo tipo primitivo (*bool, int32, float, double e string*). A figura 8 apresenta o esquema de comunicação no simulador.

A partir da definição das mensagens, o arquivo *".proto"* é compilado, gerando, automaticamente, os arquivos com as classes referentes às informações. Ainda assim, são gerados todos os métodos acessores e modificadores relacionados a cada dado pré-definido. A fim de efetuar o recebimento das informações, o cliente deve ser configurado a receber pacotes referentes ao *Multicast* enviado pelo simulador. Torna-se evidente, portanto, que este protocolo de comunicação mostra-se efetivo no envio das mensagens.

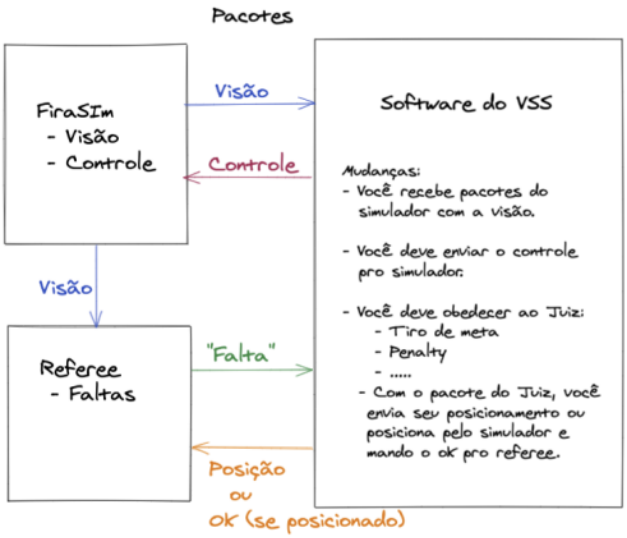

Fig. 8. Esquema de comunicação no simulador

B. Implementação

A implementação ocorre a partir da aquisição do código do Referee, do desenvolvimento referente a parte do cliente de cada time e a instalação das bibliotecas necessárias. A figura 9 mostra a tela do simulador com os parâmentros sendo recebidos no terminal.

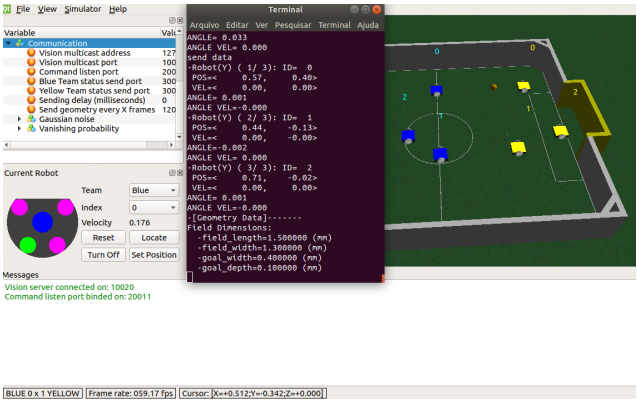

Fig. 9. Implementação do simulador

As subseções a seguir tratam cada um dos tópicos apresentados individualmente.

1) Requisitos de funcionamento: Para o seu funcionamento, o *software* faz uso de inúmeras bibliotecas, o que possibilita, efetivamente, aproximar a simulação a uma situação real de jogo. Entre os módulos e recursos necessários para um bom funcionamento, vale ressaltar:

- *ODE (Open Dynamics Engine)*: Esta é uma biblioteca [12] *Open Source* de alto desempenho que possibilita simular a dinâmica dos corpos rígidos. Sendo assim, torna-se possível detectar colisões integradas com fricção.
- *OpenGL*: Esta é uma *API (Application Programming Interface)* [9], ou seja, é uma biblioteca com funções

específicas disponibilizadas, que permite a visualização dos robôs e demais objetos no simulador.
- *QT*: Esta é uma biblioteca *Open Source*, que oferece recursos relacionados ao desenvolvimento da interface do simulador.
- *Protobuf*: Pacote que oferece recursos para serialização dos dados, possibilitando que os clientes acessem as informações fornecidas pelo simulador.

2) Referee: O árbitro [11] é um cliente responsável por processar as informações do jogo em tempo real, bem como tomar decisões e definir as situações de jogo. A partir da identificação das condições de jogo, é feito uma classificação da circunstância, podendo ser: *KICKOFF, FREE BALL, GOAL KICK, PENALTY KICK, STOP e GAME ON*. Essas informações são passadas ao cliente de cada time, sendo este responsável por analisar e tomar uma decisão.

3) Client: Como dito anteriormente, o Simulador envia os pacotes via *Multicast*, contendo as informações sobre o jogo. Entre os dados fornecidos, vale ressaltar:

- Jogo: Tempo de jogo e o número de gols de cada time (azul e amarelo)
- Campo: informações referentes à largura e comprimento do campo, bem como a largura e profundidade do gol;
- Bola: posição (x, y e z) e velocidade (Vx, Vy e Vz) da bola;
- Robôs: informações referentes à posição (x, y e z), velocidade (Vx, Vy e Vz) e orientação dos jogadores.

A partir do recebimento desses dados, estes são processados e as decisões são tomadas. Portanto, as estratégias pré-definidas pela equipe dependem, diretamente, das condições de jogo e das informações fornecidas pelo simulador.

Entre os vários fatores, a bola se torna prioridade na visão de jogo. Após tal determinação, os jogadores recebem funções, as quais serão mantidas até que ocorra alguma mudança relacionada à formação do time ou às condições de jogo enviadas pelo árbitro (Tiro-livre, Pênalti, etc). Dessa forma, os comandos enviados a cada robô são definidos anteriormente, buscando a execução correta das ações, bem como um melhor posicionamento dos jogadores.

VI. Testes e resultados

Para validar os estudos realizados, a equipe participa de competições nas quais submete os robôs às mais diversas situações de jogo, colocando-os à prova. Além disso, também são realizados testes em laboratório simulando o cenário das competições a fim de verificar o desempenho dos robôs antes dos jogos oficiais e poder aplicar melhorias, sejam elas no código, na eletrônica ou na mecânica.

Com a implementação do simulador FIRASim os testes fluem ainda melhor, ao passo que o código pode ser testado separadamente do hardware, permitindo maior aproveitamento do tempo e uma análise mais detalhada de possíveis *bugs*,podendo prever erros antes mesmo do código ser testado nos robôs em campo, no laboratório ou em competições, otimizando tempo que pode ser usado para aprimorar novas jogadas. Alguns dos testes executados em laboratório com os robôs são:

- Teste de tração e aderência das rodas: realizado com auxílio de um dinamômetro que mede a força necessária para o robô entrar em movimento (atrito estático), a partir disso é possível calcular a potência mínima que os motores devem exercer, bem como determinar a aderência do material das rodas, dentre outros aspectos importantes para a performance do robô.
- Teste dos motores: os motores são testados aos pares para aferir quais tem o giro mais parecido e serem colocados no mesmo robô a fim de evitar variações na navegação. Os testes realizados foram visuais, porém existem meios mais eficazes de realizar tal comparação.
- Teste de velocidade: executado para verificar o limite máximo de velocidade que o robô pode atingir sem perder a precisão na navegação, normalmente realizado movimentando a bolinha aleatoriamente pelo campo e aumentando gradativamente a potência nos motores.

Esses testes básicos permitem garantir a qualidade da performance dos robôs em campo. Vale ressaltar que, mesmo com a introdução do simulador os testes continuam sendo executados, no entanto também é testado o código através da simulação, aumentando ainda mais as chances de sucesso do projeto.

VII. Conclusão

Pode-se concluir, portanto, que o projeto têm sido bem aproveitado e está trazendo resultados muito positivos para a equipe. A implementação do simulador foi um fator muito significativo que pode trazer resultados ainda melhores a longo prazo, podendo ser usado para testar estratégias e códigos enquando o time físico é projetado, aumentando assim o rendimento e a produtividade da equipe. Porém existem ainda diversas possibilidades de melhoria, nas quais é possível aplicar os conhecimentos adquiridos nesse trabalho, bem como as experiênias vivenciadas com outras equipes nas competições anteriormente citadas.

Com os resultados obtidos, por meio dos experimentos práticos nos jogos durante as competições passadas, testes realizados - em laboratório remotamente via simulador - é evidente que as metas vêm sendo alcançadas e os estudos aplicados foram de extrema relevância. Para os próximos meses tem-se o interesse em aperfeiçooar o time e, até mesmo criar um novo, visando a participação na categoria 5 contra 5 e a consolidação do simulador como uma alternativa para competições futuras.

References

[1] Khan Academy. *Vector Fields*. URL: `https://www.khanacademy.org/math/multivariable-calculus/thinking-about-multivariable-function/ways-to-represent-multivariable-functions/a/vector-fields` (visited on 02/19/2019).

[2] M. Colledanchise. *Behavior Trees in Robotics and AI*. 1st. 2015.

[3] fira-simurosot. *FIRASim*. URL: https://github.com/fira-simurosot/FIRASim (visited on 06/29/2020).

[4] Google. *Protocol Buffers*. URL: https://developers.google.com/protocol-buffers/ (visited on 07/02/2020).

[5] Maxon Group. *DC-max 22 S Ø22 mm, graphite brushes, sintered bearings*. URL: https://www.maxongroup.com/maxon/view/product/motor/dcmotor/DC-max/DC-max22/DC-MAX22S01GBSL529 (visited on 04/19/2019).

[6] Maxon Group. *Encoder ENX 10 QUAD, 1 CPT*. URL: https://www.maxongroup.com/maxon/view/product/sensor/encoder/ENX/ENX10QUAD/ENX10QUAD01 (visited on 04/22/2019).

[7] IEEE. *Very Small Size Soccer (VSSS) Rules*. URL: http://www.cbrobotica.org/wp-content/uploads/2014/03/VerySmall2009_ptbr.pdf (visited on 06/10/2019).

[8] MoldFlex. *POLY 74-20 - BORRACHA DE POLIURETANO*. URL: https://www.moldflexmodelagem.com/poliuretano/poly-74-20 (visited on 08/28/2019).

[9] OpenGL. *OpenGL*. URL: https://www.opengl.org/ (visited on 07/04/2020).

[10] Robocin. *Introdução ao FIRASim*. URL: https://drive.google.com/file/d/14MZ13T2ztmO7dDnpCYRy4KFDlzB1HYhx/view (visited on 07/01/2020).

[11] Maracatronics Robotics. *VSSReferee*. URL: https://github.com/MaracatronicsRobotics/VSSReferee (visited on 07/13/2020).

[12] Russ Smith. *Open Dynamics Engine*. URL: http://www.ode.org/ (visited on 07/04/2020).

[13] STMicroelectronics. *ST1S41*. URL: https://www.st.com/en/power-management/st1s41.html (visited on 04/22/2019).

[14] STMicroelectronics. *STM32WB55CE*. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32wb55ce.html (visited on 04/27/2019).