

Equipe Poti de Futebol de Robôs da UFRN 2020

Daniel S. Morais¹, Luís G. P. Condados¹, Victor D. Queiroz², Marcos B. Barros¹
Adelardo A. D. Medeiros¹ e Pablo J. Alsina¹

Resumo—Este TDP apresenta a equipe Poti, desenvolvida no Departamento de Engenharia de Computação e Automação (DCA/UFRN), para disputar em competições. Apresentaremos aspectos do sistema do futebol de robôs, o desenvolvimento de um controle local para os robôs e a conexão do simulador FIRASim com o software da Equipe Poti.

I. INTRODUÇÃO

A Equipe Poti de Futebol de Robôs surgiu no Departamento de Engenharia de Computação e Automação da Universidade Federal do Rio Grande do Norte (DCA-UFRN), sendo participante na categoria IEEE Very Small Size Soccer (VSSS) e penta campeã brasileira.

Nesse tipo de categoria três robôs de cada time são utilizados de forma cooperativa com o objetivo de fazer o gol. Cada robô possui uma função no jogo, como: o goleiro, o atacante e o defensor, de forma que dependendo da estratégia de jogo eles podem se alternarem para melhor desempenho do time. De acordo com as regras, cada time deve ser identificado com uma cor, amarela ou azul, além disso, cores auxiliares podem ser utilizadas para melhor identificar cada robô de um time. No Poti, a cor auxiliar é utilizada para definir a função do jogador.

A arquitetura do sistema do Futebol de Robôs do Time Poti é formada por vários módulos: visão, localização, estratégia, controle e transmissão.

O módulo de visão consiste de uma câmera que fotografa o campo com os jogadores e a bola. O módulo de localização fornece a posição da bola e dos jogadores através das cores que foram predefinidas. O módulo de estratégia é responsável pelo próximo passe que os robôs deverão executar, a partir das imagens atuais. O módulo de controle é responsável por gerar as referências das posições em forma de sinal que os robôs deverão se deslocar e o módulo de transmissão envia esse sinal para os jogadores para serem executados.

Para este ano, 2020, algumas alterações nos robôs foram realizadas, bem como o desenvolvimento de um controle local embarcado nos robôs. Devido a pandemia do Covid-19, a utilização de um simulador para a competição se fez necessário, sendo então descrito neste TDP a sua conexão com o software da Equipe Poti.

¹Departamento de Engenharia de Computação e Automação, Universidade Federal do Rio Grande do Norte, Natal, RN, Brasil. E-mail: danielmorais@outlook.com.br, gabriellgpc@hotmail.com, adelardo@dca.ufrn.br, pablo@dca.ufrn.br

²Departamento de Engenharia Mecânica, Universidade Federal do Rio Grande do Norte, Natal, RN, Brasil. E-mail: victor_duarte9@hotmail.com

II. ESTRUTURA ROBÓTICA

A. Mecânica

A estrutura dos robôs é essencial para organizar e proteger os componentes eletrônicos de colisões que podem ocorrer nas partidas de Futebol de Robôs. O Equipe Poti já utilizou de diferentes materiais para construir a estrutura dos robôs, entre eles usando: prototipagem rápida em impressora 3D com acrilonitrila butadieno estireno (ABS), que é um polímero rígido e leve; o PVC expandido, que é também leve e fácil de manusear; placas de fenolite, sendo o robô montado em camadas e também o acrílico.

A nova versão da estrutura robótica é organizada em camadas e produzidas em impressora 3D com material ABS, que obedecem as dimensões para a categoria IEEE VSSS, possuindo uma base de 7,5 cm por 7,5 cm e uma altura de 6,25 cm. No total, são utilizadas quatro estruturas principais, a base onde ficam as baterias, o suporte dos motores, o suporte da placa eletrônica e a capa. A Fig. 1 mostra a estrutura com as camadas explodidas. A Fig. 2 mostra a estrutura montada.

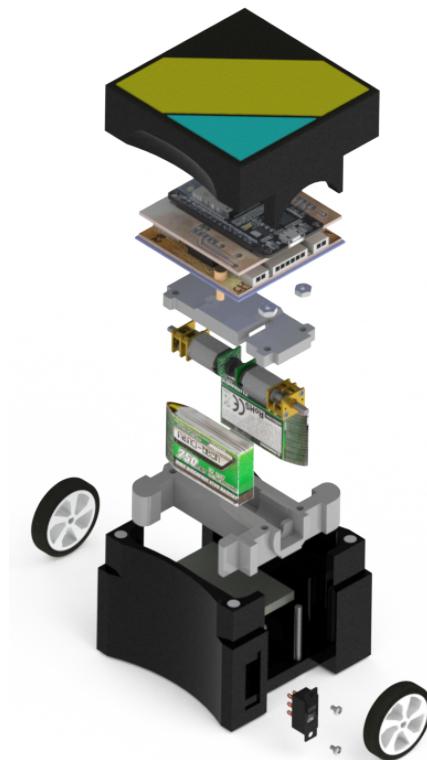


Fig. 1: Estrutura mecânica explodida.



Fig. 2: Estrutura mecânica montada.

B. Eletrônica

Os robôs da Equipe Poti possuem acionamento diferencial feitos com dois micromotores HP da Pololu com redução de 30:1. Esses motores possuem eixo estendido para encaixe de encoders magnéticos da Pololu, que faz parte do controle de velocidade local dos robôs. Foi necessário cortar parte dos eixos para que os motores coubessem na dimensão de 7,5 centímetros. A Fig. 3 mostra o motor e encoder utilizados. São acopladas aos motores rodas com diâmetro de 3,2 centímetros com aproximadamente 0,7 centímetros de espessura.

Para o acionamento e controle da velocidade dos motores utilizou-se o driver motor Pololu TB6612FNG, capaz de controlar dois motores de corrente contínua, recomendável para o acionamento de micromotores. A Fig. 4 mostra o driver utilizado.



Fig. 3: Motor com Encoder Magnético

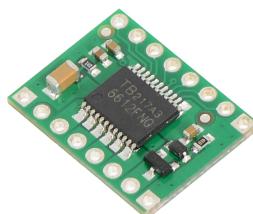


Fig. 4: Driver motor.



Fig. 5: Placa de desenvolvimento com ESP-WROOM-32.

É utilizado o microcontrolador de 32 bits da Espressif Systems (ESP32), com alimentação de 5V e que pode suportar até 12V, para controlar o robô. A utilização do ESP32 permite um design compacto, pois o microcontrolador além de possuir considerável poder computacional, para a categoria, possuindo dois núcleos de 32 bits que podem trabalhar até 240MHz, fazendo com que ele possa executar o trabalho de mais de um microcontrolador *single core* convencional, o mesmo também possui hardware para comunicação sem fio, como Bluetooth e Wi-Fi, eliminando assim a necessidade de um módulo ou hardware externo extra para a tarefa de comunicação.

Para comunicação entre os robôs e o computador foi empregado tecnologia bluetooth. Os módulos bluetooth dos robôs foram configurados como modo escravo e no módulo do computador como modo mestre.

A alimentação do robô é feita com duas baterias de 3,6 V, do tipo LiPo, ligadas em série.

III. CONTROLE LOCAL

Todos os robôs possuem um controle local de velocidade angular dos motores, que visa assegurar que eles atinjam as velocidades de referência em suas rodas, e também visa minimizar as assimetrias no tempo de resposta de cada par de motor. O controle é do tipo feedforward mais um controlador PID, como ilustrado na figura 6, onde ω_{ref} é a referência da velocidade angular para um motor, ω_{motor} é a velocidade do motor, ω_{enc} é a velocidade angular medida pelo encoder (Enc.), $f(\omega)$ é a função que relaciona velocidade angular e tensão que deve ser somada à saída do controlador PID para ser aplicada no motor, mas antes esse valor passa por um bloco de saturação (Sat.). Essa arquitetura possibilita erro de regime igual a zero e um bom controle do tempo de resposta.

Para a utilização do componente feedforward ($f(\omega)$) o robô realiza uma auto-calibração que possibilita estimar a relação entre a velocidade de giro de cada roda com a tensão (via PWM) correspondente. Além disso, considerando o sistema de primeira ordem, podemos estimar o ganho e a constante de tempo para cada motor. Com essas informações,

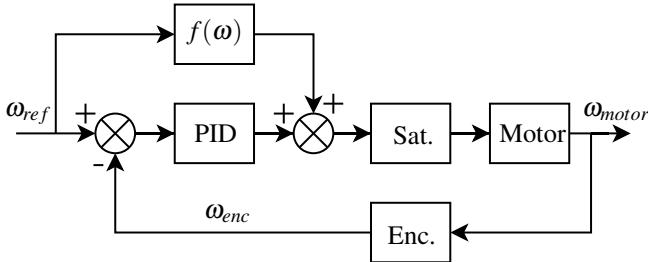


Fig. 6: Diagrama controle local Feedforward/Feedback.

é possível calcular as constantes do PID de forma a otimizar a resposta dos motores e diminuir a diferença no tempo de resposta entre eles.

IV. SIMULADOR

Os simuladores são sempre bem-vindos no desenvolvimento de projetos de robótica. Eles servem para realizar testes ou serem uma ambiente final para competições de robótica. Quanto mais próximo os sistemas dos simuladores forem dos sistemas reais, melhor será os resultados obtidos com ele.

A Equipe Poti possui um simulador para a realização dos testes de seus algoritmos, tendo uma estrutura de rede bem desenvolvida. Outro simulador bastante utilizado é o FIRASim [1], que foi modificado em 2020 para se adequar à categoria IEEE VSSS. Nesta seção, será mostrado como foi realizada a conexão entre esse simulador e o software da Equipe Poti.

A. Simulador FIRASim

O simulador FIRASim foi desenvolvido para simular o futebol de robôs da categoria IEEE VSSS. Ele é baseado em outro simulador, o grSim, desenvolvido para a categoria Small Size League (SSL) da RoboCup.

O FIRASim é construído em Qt5 [2] e utiliza o protobuf [3] como protocolo de comunicação com os softwares clientes, além de utilizar a biblioteca OpenGL [4] para a construção de seus gráficos. Para a simulação dos robôs é utilizada a biblioteca Open Dynamics Engine (ODE) [5], que permite simular a dinâmica de corpos rígidos e detectar colisões. A Fig. 7 mostra a interface do simulador.

B. Comunicação com o simulador

O simulador FIRASim trabalha em modo servidor, disponibilizando duas portas UDP para receber comandos contendo as velocidades das rodas dos robôs dos dois times. Além disso, o simulador cria e mantém um canal *multicast* por onde ele transmite o estado atual do jogo, contendo posições e orientações de cada robô e informações geométricas do campo, provendo assim a informação da etapa de aquisição para ambos os times.

A cada passo da simulação o servidor transmite de forma serial, no grupo *multicast*, um pacote contendo um **Environment**, que encapsula as informações de posição, orientação e velocidade dos objetos da cena no instante atual e cada cliente pode transmitir um pacote do tipo **Command** para



Fig. 7: Simulador FIRASim

controlar as velocidades das rodas de seus robôs. Essa comunicação é ilustrada na figura 8 e mais adiante será explicado com mais detalhes o que é transmitido.

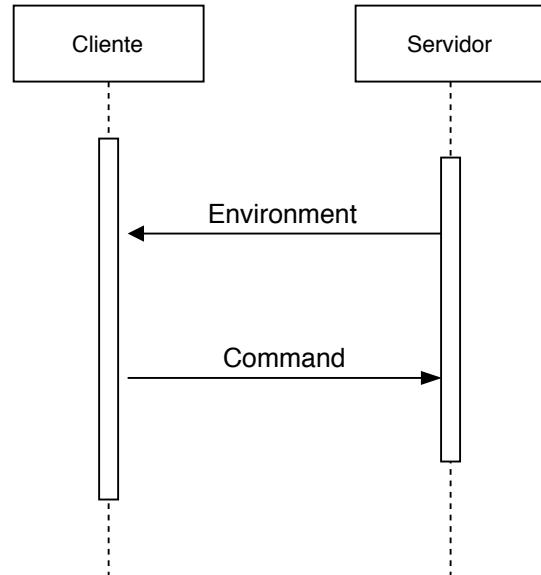


Fig. 8: Exemplo comunicação cliente-servidor

Como já mencionado, a comunicação utiliza-se do Protocol Buffers (protobuf), o protobuf é um método de serialização de dados estruturados de código aberto. A figura 9 mostra as principais estruturas de dados utilizadas na comunicação com o simulador. A estrutura **Environment** contém um contador de passos de simulação (variável step), um componente do tipo **Frame**, um componente do tipo **Field** que registra as informações da geometria do campo e variáveis para registrar a pontuação de cada equipe (variáveis goals_blue e goals_yellow).

Frame encapsula uma variável do tipo **Ball** (variável ball) que contém as informações referentes à bola e duas variáveis do tipo **Robot** (variáveis robots_yellow e robots_blue), uma para cada time, que possuem os dados de todos os robôs no passo atual.

Field armazena quatro informações, sendo elas largura, comprimento do campo, largura e profundidade da área do gol, nas variáveis width, length, goal_width e goal_depth respectivamente.

A estrutura **Ball** possuí informações de posição (x,y,z) e velocidade da bola (vx,vy,vz) com relação ao referencial global.

Cada estrutura do tipo **Robot** contém um identificador (robot_id) do robô, para esta categoria esse id assume os valores de 0,1 ou 2, em cada equipe, já que cada equipe possui 3 robôs no máximo, além do robot_id esta estrutura contem informação de posição (x[m],y[m]), orientação (orientation [rad]) e velocidade (vx[m/s],vy[m/s], vorientation[rad]) em relação ao referencial global.

Por fim, as estruturas do tipo **Command** contém uma variável id para identificar qual robô é o destinatário do comando, que pode assumir os valores 0, 1 ou 2. **Command** também possui um campo para identificar o time (azul ou amarelo) por meio de yellowteam. Para informar a velocidade de referência das rodas direita e esquerda do robô deve-se passar essa informação em wheel_right e wheel_left respectivamente.

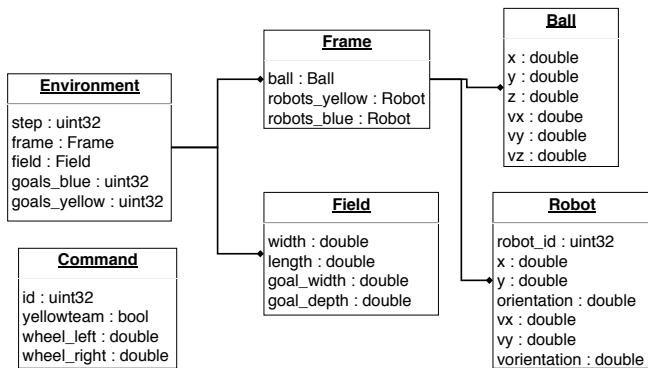


Fig. 9: Diagrama das estruturas utilizadas na comunicação.

V. CONCLUSÃO

Este TDP se propôs a descrever a construção e desenvolvimento da estrutura dos robôs, o controle local embarcado e a conexão com o simulador. Os materiais e dispositivos utilizados são populares e que facilitam as pessoas em suas criações. Além disso, esperamos incentivar e ajudar novas equipes a desenvolverem seus projetos.

REFERÊNCIAS

- [1] FIRASim. [Online]. Available: <https://github.com/IEEEVSS/FIRASim>
- [2] Q. Group. Qt5. [Online]. Available: <https://doc.qt.io/qt-5/index.html>
- [3] Google. Protocol Buffers. [Online]. Available: <https://developers.google.com/protocol-buffers>
- [4] K. Group. OpenGL. [Online]. Available: <https://www.opengl.org/>
- [5] R. L. Smith. Open Dynamics Engine. [Online]. Available: <https://www.ode.org/>