RobôCIn 2018 SSL Description Paper

Camila Oliveira de Souza[1], Cecília Virgínia Santos da Silva[1], Felipe Bezerra Martins[1], João Gabriel Machado da Silva[1], Lucas Henrique Cavalcanti Santos[1], Renato Sousa Bezerra[1], Roberto Costa Fernandes[1], Victor Hugo Sabino dos Santos Araújo[1], Vinícius Bezerra Araújo da Silva[1], Hansenclever de França Bassani[1], Edna Natividade da Silva Barros[1]

***Abstract*— Este *Team Description Paper* (TDP) tem por objetivo introduzir a equipe RobôCIn na categoria RoboCup Small Size League (SSL) da *Latin American and Brazilian Robotics Competition* (LARC/CBR) 2018, competição que ocorrerá de 6 a 10 de novembro em João Pessoa/PB. Serão descritos os primeiros avanços nas áreas de mecânica, software e embarcados, destacando a arquitetura de software de alto nível, o planejamento de caminhos, o projeto mecânico da base, do *dribbler* e do chute, e a arquitetura de baixo nível.**

I. INTRODUÇÃO

A equipe do RobôCIn vem participando desde 2016 da categoria *IEEE Very Small Size Soccer* (VSSS) [1], que consiste em uma disputa de futebol de robôs autônomos. Para o ano de 2018 foi decidido expandir a equipe para novas categorias e uma das escolhidas foi a *RoboCup Small Size League* (SSL) [2]. Além da similaridade com a categoria anterior, algumas das motivações para a escolha da categoria foram a maior complexidade dos robôs (quantidade, tamanho e componentes) e o aumento no campo de jogo, permitindo o uso de mais recursos e estratégias, como chute e *dribbling*.

Esse artigo descreve as pesquisas e o desenvolvimento feito pela equipe para a participação na categoria. Para atingir tal objetivo, a equipe foi dividida em três subequipes: software, mecânica e embarcado. A equipe de software desenvolveu todas as etapas de controle do time e definiu as estratégias, seus estudos serão detalhados na seção II. A equipe de mecânica ficou responsável pela modelagem e montagem dos componentes mecânicos do robô, e seu trabalho será apresentado na seção III. E a equipe de embarcados elaborou toda a eletrônica para o controle de baixo nível dos robôs, e suas descobertas serão apresentadas na seção IV.

II. SOFTWARE

Como a equipe está participando da categoria pela primeira vez, os primeiros estudos de software foram dedicados a definição da arquitetura e da estratégia de planejamento de caminhos. Essas duas atividades foram priorizadas, pois, ambas as definições, se forem bem feitas, poderão ser aproveitadas para os próximos anos da equipe na categoria.

Na subseção II-A será apresentada a arquitetura de software proposta pela equipe e o fluxo de dados nessa arquitetura. Na subseção II-B será detalhada a escolha do algoritmo de planejamento de caminho utilizado.

[1]Todos os autores estão no RobôCIn no Centro de Informática, Universidade Federal de Pernambuco, Brasil robocin@cin.ufpe.br

A. Arquitetura de software alto nível e fluxo de dados

Foram utilizadas duas arquiteturas como base para a construção da arquitetura proposta. A primeira arquitetura fez uso da abordagem do *Skill, Tactics and Play* [3]. Esta arquitetura propõe o uso de 3 diferentes níveis de tarefas. A tarefa de mais alto nível é a *Play*, que define as jogadas em que todos os jogadores estão participando. Em uma *Play* cada jogador executa uma *Tactic*, que consiste em uma máquina de estados, onde cada estado é uma *Skill*. Uma *Skill* consiste no conjunto de funções de mais baixo nível, como ir para um local predefinido, passar a bola para outro jogador, ou chutar a bola ao gol.

A outra arquitetura utilizada como inspiração foi a proposta pela equipe ER-Force [4]. Nesta arquitetura é proposto que cada jogador aja como um agente autônomo para tarefas de baixo nível, mas ainda com a presença de um software central que determina as funções de alto nível para todos os jogadores. Como os jogadores são agentes, eles podem se comunicar entre si, e assim, conseguem combinar jogadas e tomar decisões mais rapidamente. Porém, a utilização total dessa arquitetura aumenta a quantidade de troca de mensagens entre os jogadores, podendo criar um gargalo na rede de comunicação.

Utilizando conceitos das duas arquiteturas a equipe propõe o uso de uma arquitetura híbrida, que será primeiramente implementada com base em um controle central, mas pode ser facilmente evoluída para uma abordagem descentralizada, com os jogadores sendo agentes autônomos. Uma visão geral dessa arquitetura pode ser observada na Figura 1, que mostra as classes utilizadas nessa arquitetura. Outra vantagem da arquitetura escolhida é facilitar as tarefas paralelas, onde cada classe pode ser tornar uma *thread*. Nas próximas subseções será detalhada cada classe presente na arquitetura.

1) DataWorld: Classe responsável pelo recebimento de dados externos, que vêm tanto do *SSL-Vision* [5], quanto do *SSL Referee*. Foi escolhido destinar uma *thread* somente para essa função, por conta da alta taxa de recebimento de pacotes externos, que podem chegar à $800Hz$. Além disso também ficou previsto que esta classe ficaria responsável pelo tratamento de qualquer possível recebimento de dados dos jogadores. Esses dados recebidos dos jogadores podem aprimorar suas posições, utilizando filtros, como por exemplo o filtro de Kalman ou o filtro de Markov.

Como saída para classes subsequentes é entregue a estrutura definida pela equipe e nomeada de *Frame*. Essa estrutura

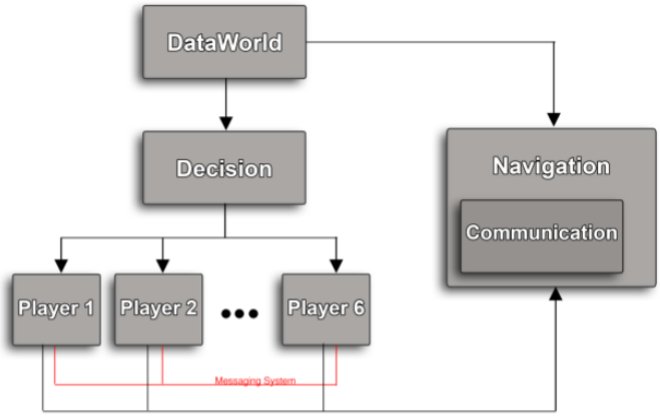

Fig. 1. Visão geral da arquitetura de software proposto.

contém as informações mais atualizadas das posições dos elementos no campo, robôs e bola, e dos comandos recebidos pelo *SSL Referee*. Dessa forma, as outras classes da arquitetura requisitam o *Frame* mais atual quando necessitam e não causam um gargalo no recebimento de dados.

2) Decision: Classe destinada a escolha da jogada (*Play*) que irá ser executada pelos jogadores. Essa classe é executada em paralelo com as outras *threads*, com uma taxa preestabelecida de 60*Hz*. No começo de cada execução é requisitado o *Frame* mais atual de *DataWorld*. A escolha da *Play* a ser executada recebe maior influência da situação de jogo e dos comandos enviados pelo árbitro. Essa influência ocorre devido a situações estáticas, como escanteios e pênaltis, onde pode ser escolhida uma jogada predefinida.

A situação de jogo que permite uma escolha mais refinada da jogada ocorre quando a bola está livre e o jogo está sendo disputado. Neste tipo de lance, a tática (*Tactic*) dos defensores e dos goleiros deve ser escolhida de forma que eles possam bloquear um eventual chute ao gol. A tática dos jogadores de ataque e de suporte é escolhida dentre as possíveis jogadas, para que os jogadores possam se posicionar para receber a bola em condições de chutar ao gol. A tática dos jogadores avançados também é influenciada pela situação geral do jogo.

3) Player: A classe *Player* é responsável pela definição de cada tática previamente escolhida. Por isso, tal classe recebe como entrada a posição de todos elementos no campo e a jogada definida em *Decision*, e consequentemente, qual tática cada jogador irá executar. Assim, é na classe *Player* que são definidas as máquinas de estados de cada tática que todos os jogadores podem executar de forma independente. Além disso é previsto uma estrutura de troca de mensagens das *threads* via software, para permitir a cooperação entre jogadores.

Na Figura 2 é possível observar a arquitetura interna de *Player*, composta das máquinas de estado das táticas e das definições das *Skills* utilizadas. Dentro das *Skills* estão definidos, por exemplo, algoritmos de planejamento de caminhos, posicionamento para prevenir um chute e algoritmos de decisão para acorrer ou não um chute. Devido ao fato de que cada jogador é comandado por uma *thread* separada, é esperado como saída de cada *thread* apenas a posição objetivo de cada jogador.

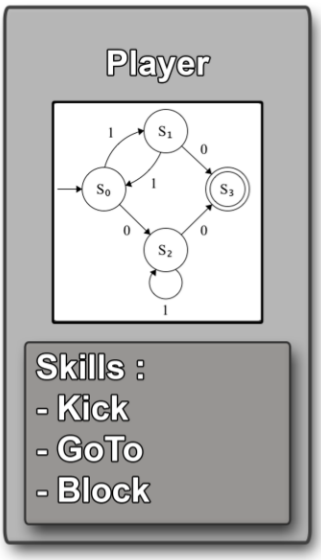

Fig. 2. Visão do diagrama interno da classe *Player*.

4) Navigation: Em *Navigation* é definida a estratégia de navegação para calcular as velocidades lineares e angulares para cada robô. Para realizar tal tarefa, são utilizados como entradas a posição mais atualizada do jogador vindo de *DataWorld* e as posições-objetivo vindo de cada *Player*. Para garantir uma alta taxa de correção na velocidade enviada para o robô, a tarefa de navegação é executada em paralelo com o restante do código. A taxa de atualização de *Navigation* é de 60*Hz*, pois já garante a corretude na execução do caminho planejado.

Dentro da classe *Navigation* é definido a classe *Communication*, que é responsável pelo envio de informações do computador central para todos os robôs em campo. *Communication* foi escolhido para ser unicamente definido em *Navigation*, ao invés de uma vez em cada *Player*, para evitar a concorrência de várias *threads* tentando acessar o mesmo recurso. Dessa forma o rádio só é utilizado pela *thread* de *Navigation*, quando as novas velocidades para cada robô são calculadas.

B. Planejamento de caminhos

A *Skill* de mais alta importância é a de se movimentar dentro do campo, por isso foi a primeira a ser implementada. O primeiro passo na movimentação é o planejamento de caminho a ser realizado, de tal forma que esse caminho seja ótimo e que evite a colisão do robô com obstáculos. Para o planejamento do caminho é necessário uma representação do mundo em um mapa navegável, para isso existem várias abordagens [6], como campos potenciais, diagrama de Voronoi e grafo de visibilidade.

A primeira possibilidade considerada pela equipe foi o uso do método de campos potenciais, pois a equipe já utilizou

tal abordagem na categoria VSSS. Na categoria VSSS, a equipe utilizou uma representação com $1cm$ de precisão, pois o campo mede $1,5m \times 1,3m$, e tal representação gasta somente 200KB de memória. Porém, para representar todo o campo do SSL [7] de $9m \times 6m$ com a mesma precisão de $1cm$, seria necessário utilizar 8MB de memória por mapa para cada robô, além de gerar um grande *overhead* de tempo para percorrer todo esse mapa com essa precisão.

Outra possibilidade de representação de mapa seria o uso do diagrama de Voronoi [8], que utiliza como possíveis caminhos, aqueles que, maximizam a distância para os obstáculos. Assim, essa abordagem visa fazer o caminho mais seguro, porém tal caminho pode se tornar muito longo e muito lento para se usar em uma competição. A construção do diagrama de Voronoi tem custo computacional $O(n \times log(n))$, porém como é necessário que seja executado o caminho mais rápido para o objetivo, não foi utilizado o diagrama de Voronoi.

Dentre as abordagens analisadas, o uso de grafo de visibilidade [9] apresenta um custo computacional menor que campos potenciais e consegue encontrar o menor caminho até a posição objetivo. O grafo de visibilidade constrói uma representação em grafos, onde os obstáculos são polígonos e os pontos inicial e final são arestas do grafo. Como os robôs da competição são circulares, na representação construída pela equipe, os robôs foram representados como triângulos equiláteros circunscritos ao círculo que representa o robô. Com isso, as arestas dos polígonos que representam os obstáculos são consideradas como arestas do grafo.

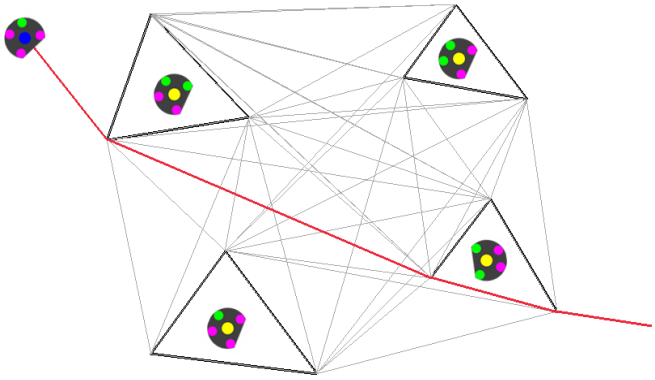

Fig. 3. Exemplo de aplicação do gráfico de visibilidade para SSL.

A partir da representação dos obstáculos e dos pontos inicial e objetivo, o grafo de visibilidade é construído conectando todas as arestas do grafo, exceto as que cruzam um obstáculo. Assim, serão criados possíveis caminhos que passam perto dos obstáculos, mas não colidem com nenhum objeto, e são menores que os propostos pelo diagrama de Voronoi. O custo computacional dessa abordagem é $O(n^3)$, onde n é quantidade de arestas no grafo, ou seja, o custo é proporcional à quantidade de obstáculos e à representação escolhida para os obstáculos. Um exemplo de uso de grafo de visibilidade na SSL pode ser visto na Figura 3.

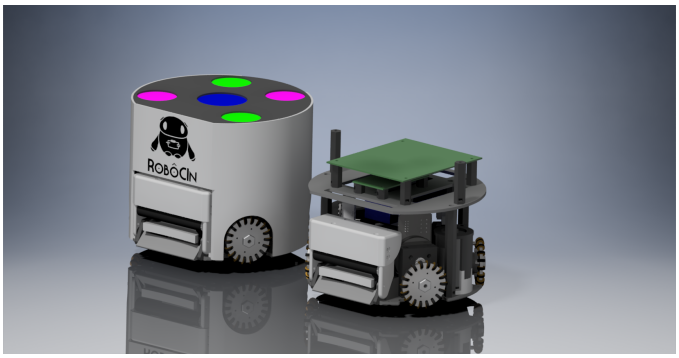

Fig. 4. Visualização do Modelo 3D da Mecânica do Robô

III. MECÂNICA

A mecânica do robô foi projetada considerando todas as especificações do regulamento da categoria [7]. Na Figura 4 é apresentada uma visualização interna e externa do robô completo, feita em ferramenta de modelagem 3D. Com o proposito de centralizar e facilitar a localização das informações descritas nesta seção, as especificações mecânicas completas são encontradas na Tabela I ao final desta seção.

A. Mecânica da Base

Inicialmente, foram feitas pesquisas para definir o posicionamento ideal para os motores de movimentação, levando em consideração a eficiência, a durabilidade, as dimensões e o custo. Assim, a escolha feita foi pelo motor *brushless Maxon EC-45 flat 50W* [10] com *encoders*, que é amplamente utilizado pelas equipes da categoria e atende aos requisitos necessários do robô. Para uma maior otimização do espaço na base, foram escolhidos motores da série *flat* e também definidas as distâncias angulares de 120 graus entre os motores frontais e de 90 graus entre os traseiros dispostos simetricamente em relação a um eixo longitudinal, disponibilizando mais espaço para o *dribbler* e para o mecanismo de *kicker*. Na Figura 5 é possivel visualizar tais componentes descritos, posicionados na base do robô.

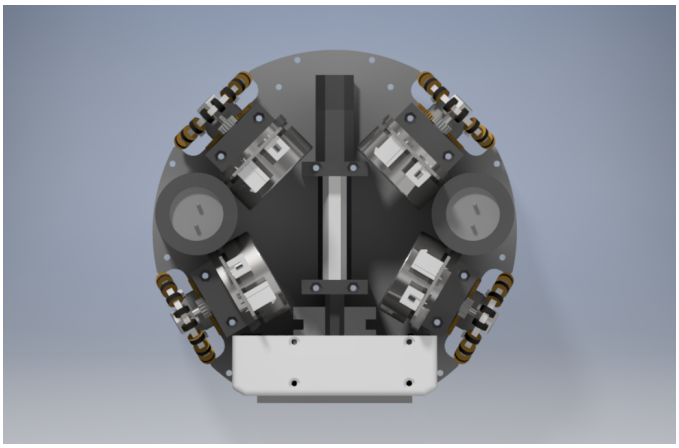

Fig. 5. Vista superior da Base do Robô em 3D, com as estruturas de fixação dos motores e roda, mecanismo de chute, *dribbler* e capacitores.

O robô possui movimentação holonômica, ou seja, se movimenta para todas as direções e rotaciona no próprio eixo. Este movimento é proporcionado por quatro rodas omnidirecionais modelo SW-504 da *GTF Robots* [11], que tem 50mm de diâmetro com 18 roletes feita de alumínio. A formação escolhida com quatro rodas possibilita maior potência, mais firmeza e maior espaço para o eixo central.

Tanto para o *dribbler*, quanto para as rodas, foram escolhidos pares de engrenagens de dentes retos para um melhor aproveitamento do espaço interno do robô. Para as rodas, a relação de transmissão é de $10 : 3$. Para atingir esta configuração, foram feitos testes com modelos 3D utilizando também engrenagens internas para as rodas, e correias para o *dribbler*, como utilizado por algumas equipes internacionais em outras edições da competição.

Um estudo realizado a partir de [12], determinou que para um melhor engrenamento, é necessário que as engrenagens estejam distantes em relação ao diâmetro da circunferência primitiva, como mostra a Figura 6.

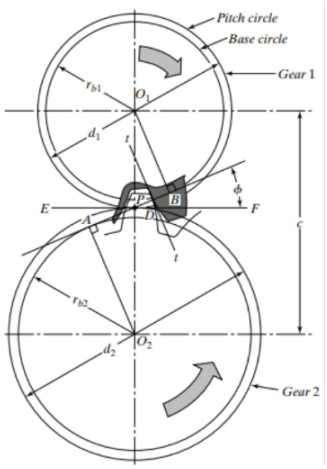

Fig. 6. Distância entre eixos de duas engrenagens de dentes retos. Fonte: [13].

Portanto, a distância entre os eixos do motor e do *dribbler* é dada pela Equação 1, onde $d1$ é o diâmetro da circunferência primitiva do pinhão, e $d2$ é o diâmetro da circunferência primitiva da coroa.

$$c = \frac{d1 + d2}{2} \tag{1}$$

B. Dribbler

Para o mecanismo de *dribbler* foi escolhido o motor *Maxon EC-22 max* [14], acoplado a um par de engrenagens de dentes retos com relação de transmissão de $13 : 25$. A transmissão se situa entre motor e o eixo do *dribbler*, que possui $14mm$ de diâmetro. Para encontrar o material mais aderente e de melhor desempenho, estão sendo realizados testes com os seguintes materiais da barra: selante de silicone, emborrachado e silicone sólido. Também está sendo desenvolvido um protótipo que permita centralização da bola no *dribbler*, apresentado pela equipe Op-AmP em 2017 [15]. Este mecanismo possui duas helicoides contrárias nas extremidades da barra que, ao entrar em contato com a bola, agem de forma a manter a mola no seu centro.

Para o posicionamento do *dribbler* no chassis, foi feito um estudo matemático baseado na regra 80/20 [7], a qual requer que 80% da bola precisa estar livre de contato e visível pela parte de cima do robô. Esse estudo nos ajudou a encontrar a altura ideal entre eixo do dribbler e o chão $(43,8mm)$, e o ponto de contato com a bola para maximizar o seu domínio, que irá formar um ângulo de aproximadamente $51,5$ graus.

Versão do robô	2018
Motores de Condução	*Maxon EC-45 flat - 50W*
Encoder	*MILE 1024 CPT*
Engrenagem	18 : 60
Tipo de Engrenagem	Engrenagem Externa
Roda	*GTF Robots SW-504*
Diâmetro Roda	50mm
Motor Dribbler	*Maxon EC-22 max*
Engrenagem Dribbler	50 : 26
Velocidade do chute	máx. 8 m/s
Microcontrolador	STM32F746G
Área de cobertura da bola	19,55%

TABLE I
ESPECIFICAÇÕES DO ROBÔ

IV. EMBARCADO

O sistema eletrônico do RobôCIn consiste em duas placas principais: a *Main Board*, responsável por toda a computação embarcada e pelo controle dos motores, e a *Kicker Board*, responsável por elevar a tensão da bateria e acionar o chute. Descrições mais detalhadas da *Main Board* e da *Kicker Board* são dadas nas seções IV-A e IV-B, respectivamente.

A. Main Board

A *Main Board* conta com um processador STM32F746 da STMicroelectronics com núcleo ARM Cortex M7, 320kB de memória RAM, *clock* máximo de 216MHz, além de 6 interfaces SPI, todos os pinos preemptíveis e até 18 timers disponíveis. Esse processador é responsável por tarefas como o controle dos motores do robô, acionamento do chute, comunicação *wireless*, interpretação da leitura dos sensores, entre outras.

Para realizar a comunicação entre o computador central e cada um dos robôs, foi escolhido o transceptor nRF24L01+, que trabalha em uma frequência de 2.4GHz e a uma taxa de transmissão máxima de 2Mb/s. O módulo nRF24L01+ já é utilizado pela equipe RobôCin na categoria Very Small Size e vem mostrando boa performance nas competições. A comunicação entre o transceptor e o processador é feita através da interface SPI.

O acionamento dos quatro motores Maxon EC-45 flat 50W, responsáveis pela movimentação do robô, e do motor Maxon EC-max22 25W, responsável pelo *dribbler*, é feito utilizando o MOSFET IRF8113. Dos 18 timers do processador, 2 deles são de 32 bits, no qual um deles é usado para gerar os 4 sinais de PWM para controlar a velocidade dos motores de movimentação. Todos os motores

de movimentação possuem ainda um *encoder* que permite a aferição de suas velocidades, que por sua vez, o outro timer de 32 bits é resposável por receber e ler os sinais dos *encoders* aumentando assim a precisão no controle da movimentação do robô. Um outro timer de 16 bits é usado para gerar o sinal PWM que controla a velocidade do motor do *dribbler*.

Além dos motores, outros atuadores existentes no robô são LEDs, para a comunicação visual de informações importantes, e um *buzzer*, utilizado com o intuito de aumentar a segurança no manuseio do robô.

Informações adicionais para um melhor controle da movimentação do robô são trazidas pelo MPU6000, um módulo que contém um acelerômetro e um giroscópio; e também os 2 sensores de mouse. Ambos se comunicam com o processador através da interface SPI. Outros sensores presentes na *Main Board* incluem circuitos para detecção de bola por infravermelho e para o monitoramento das tensões da bateria e dos capacitores da placa de chute. É na placa principal ainda que se encontra o circuito responsável pela alimentação de todo o robô. A tensão da bateria é regulada para 5V pelo conversor *Buck* ST1S40IPHR e para 3.3V pelo regulador LT3080.

Na Figura 7 podemos visualizar um diagrama de blocos da arquitetura utilizando o processador escolhido e demais componentes descritos nesta seção.

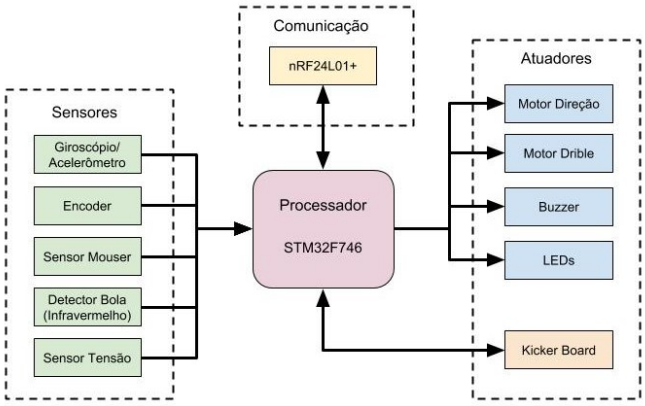

Fig. 7. Arquitetura da *Main Board*.

B. Kicker Board

Na *Kicker Board* encontramos um conversor DC-DC com topologia *Boost*, responsável por elevar os 14.8V fornecidos pela bateria 4S até 200V. Essa tensão é então utilizada para carregar dois capacitores de 3300uF, ligados em paralelo. Quando o comando de chute é enviado pelo processador, esses capacitores são descarregados em um solenóide, que por sua vez ativa o mecanismo de chute, seja esse o *kicker* ou o *chip kick*. Tanto o indutor presente no conversor *Boost*, quanto o solenóide utilizado para acionar o mecanismo do chute são de fabricação própria da equipe. A Figura 8 apresenta um diagrama de blocos dos componentes utilizados na realização de um chute.

A tensão presente nos capacitores é medida pelo conversor analógico digital ADC122S051 e enviada através do protocolo SPI para o processador na *Main Board*, a fim de que este possa controlar o carregamento e o descarregamento desses capacitores no momento oportuno, de acordo com a estratégia de jogo. Essa mesma tensão passa ainda por um divisor de tensão e por um amplificador operacional na configuração comparador, que aciona um LED quando os capacitores atingem uma tensão pré-definida. Esse indicador visual é utilizado para garantir a segurança dos membros da equipe, evitando que o robô seja manuseado com os capacitores ainda carregados. Um botão de segurança presente na placa permite o descarregamento manual dos capacitores no solenóide, sem a necessidade de que um comando seja dado pelo processador.

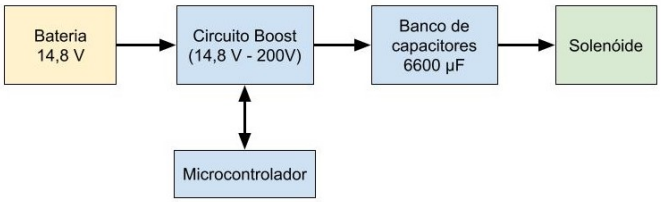

Fig. 8. Diagrama de Blocos dos Componentes do Chute.

C. Protocolo de comunicação

O protocolo de comunicação foi pensado baseado no módulo de comunicação nRF24L01+, que usa uma topologia estrela com até 6 pipes. Contudo, as mensagens são enviadas por *broadcast* e todos os robôs, juntamente com o computador, estão no mesmo canal e pipe. Então, além de campos como velocidades dos motores, existe um campo que identifica o robô destino para a mensagem. O protocolo de mensagem do computador para os robôs é construído como ilustra a Tabela II.

Motor1	Motor2	Motor3	Motor4	Motor Dribbler	ID	KickFlag	FbFlag
1 byte	1 byte	1 byte	1 byte	1 byte	3 bits	1 bit	4 bits
6 bytes							

TABLE II
DIAGRAMA DO PROTOCOLO DE MENSAGEM DO COMPUTADOR PARA ROBÔ

O campo de Feedback Flag indica que o computador precisa de um feedback do robô, então foi construído alguns tipos de mensagem de resposta. A resposta completa do robô para o computador é ilustrada como na Tabela III. Desse modo foi construído o protocolo RobôCIn SSL.

PosX	PosY	Theta	Charge	IR	ID	KickFlag	Battery
1 byte	1 byte	1 byte	1 bit	1 bit	3 bits	1 bit	2 bits
4 bytes							

TABLE III
DIAGRAMA DO PROTOCOLO DE MENSAGEM RESPOSTA DO ROBÔ PARA O COMPUTADOR

V. CONCLUSÃO

Através da subdivisão de áreas, foi possível atacar todos os aspectos da categoria SSL [2]. Nela foram encontrados desafios complexos, porém os artigos disponíveis foram essenciais para iniciar os desenvolvimentos. Os testes dos sistemas eletrônicos, como chute e *dribbler* se demostraram estáveis, assim como o software de planejamento e navegação de caminho estão sendo testados em simulador. O próximo passo é a construção do campo para testar todo o sistema integrado, a equipe também identificou funções do jogo, como posicionamento em bolas paradas, as quais serão otimizadas com o uso de aprendizado de máquina.

AGRADECIMENTOS

A equipe gostaria de agradecer o Centro de Informática da Universidade Federal de Pernambuco pelo apoio financeiro e de recursos durante todo o processo do projeto. Também gostaríamos de agradecer à todo apoio dado pelos professores Hansenclever Bassani e Edna Barros.

REFERENCES

[1] IEEE. Very small size soccer. [Online]. Available: http://www.cbrobotica.org/?page_id=81

[2] RoboCup. Small size league. [Online]. Available: http://wiki.robocup.org/Small_Size_League

[3] B. Browning, J. Bruce, M. Bowling, and M. Veloso, "Stp: Skills, tactics, and plays for multi-robot control in adversarial environments," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 219, no. 1, pp. 33–52, 2005.

[4] C. Lobmeier, D. Burk, A. Wendler, and B. M. Eskofier, "Er-force 2018 extended team description paper," 2018, robocup Small Size League, Montreal, Canada, 2018.

[5] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso, "Ssl-vision: The shared vision system for the robocup small size league," in *Robot Soccer World Cup*. Springer, 2009, pp. 425–436.

[6] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[7] *Laws of the RoboCup Small Size League 2016*, Small Size League Technical Committee, 4 2016.

[8] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, "Sensor-based exploration: Incremental construction of the hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 126–148, 2000.

[9] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[10] *Maxon EC 45 flat datasheet*, Maxon Motors, 5 2017.

[11] *Omni directional Wheel SW-504*, GTF Robots, 6 2017.

[12] R. L. Norton, *Kinematics and dynamics of machinery*, 1st ed. Boston, Mass. : McGraw-Hill, 2009.

[13] R. Norton, *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines*, ser. McGraw-Hill series in mechanical engineering. McGraw-Hill Higher Education, 2004.

[14] *Maxon EC-max 22 datasheet*, Maxon Motors, 3 2017.

[15] T. Yoshimoto, T. Horii, S. Mizutani, Y. Iwauchi, Y. Yamada, K. Baba, and S. Zenji, "Op-amp 2017 team description paper," 2017, robocup Small Size League, Nagoya, Japan, 2017.