Engenharia de Controle e Automação

Projeto de Graduação

22/06/2014

PROJETO E CONTROLE DE ROBÔ MÓVEL SEMIAUTÔNOMO COM TÉCNICAS DE LÓGICA FUZZY

Luiz Fernando Santarelli Conrado Nobre

PUC RIO

PROJETO E CONTROLE DE ROBÔ MÓVEL SEMIAUTÔNOMO COM TÉCNICAS DE LÓGICA FUZZY

Aluno: Luiz Fernando Santarelli Conrado Nobre

Orientador: Marco Antônio Meggiolaro

Coorientador: Alexandre Ormiga Galvão Barbosa

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia de Controle e Automação na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil

PUC RIO

Resumo

Este trabalho de fim de curso tem como objetivo desenvolver um veículo robótico semiautônomo para competições de sumô robótico, englobando diversas áreas da Engenharia de Controle e Automação. Foram utilizados conceitos de projeto e modelagem mecânica em Software CAD e CAM (Computer Aided Design/Machining), desenvolvimento de interface eletrônica, técnicas de fabricação e usinagem e a aplicação algoritmos de controle inteligente baseados em Lógica Fuzzy.

A navegação do robô é feita por meio de comandos do operador. Ao detectar o seu oponente, utilizando seus sensores, o robô assume o controle e persegue o adversário, com o objetivo de empurrá-lo para fora da arena, mas mantendo-se dentro dela.

Palavras-chave: Lógica Fuzzy; Sumô Robótico; mbed; CAD; CAM; Robótica;

Project and control of semi-autonomous mobile robot using Fuzzy Logic Technics

Abstract

The objective of this work is to develop a semi-autonomous robot to compete in robot sumo tournaments, encompassing several areas of the Mechatronic Engeneering.

Concepts of project and mechanic design applied to CAD and CAM (Computer Aided Design/Machining) Software, electronic interface development, manufacturing and machining techniques, as well as the application of control algorithms based on Fuzzy Logic were used.

The robot navigates by its operator's comand. When the robot's sensors detect the opponent, the robot takes control and autonomously chases the other robot, in order to push it outside the arena, withouth falling itselft out as well.

Keywords: Robotics; Robot Sumo; mbed; Fuzzy Logic; CAD; CAM;

Sumário

Lista de Abreviaturas e Siglas

R/C – Rádio Controlado

RF – Rádio Frequência

CC – Corrente Contínua

PCB – Printed Circuit Board – Placa de circuito impresso

PPM – Position Pulse Modulation

PWM – Pulse Width Modulation

RF – Rádio Frequência

TI – Tecnologia da Informação

Índice de Figuras

“Everything should be made as simple as possible, but no simpler. ”

- Albert Einstein

1 Introdução

1.1 Motivação

Os avanços tecnológicos das últimas décadas estão permitindo que hoje se desenvolva cada vez mais a autonomia de maquinas e equipamentos. Com o surgimento do conceito da Internet das Coisas, a interação de objetos do nosso cotidiano entre si e com os humanos está cada vez mais presente e ainda tem muito o que evoluir.

Uma forma de interação é o desenvolvimento de técnicas de controle semiautônomo. São recomendados em situações onde, em momentos críticos, a inteligência robótica se julga mais capaz que o operador e assume o controle, garantindo assim fatores como eficiência e segurança nos processos.

Mais especificamente no que diz respeito a competição, este projeto foi desenvolvido para competir no *24th All Japan Robot-Sumo Tournament* [1], maior torneio internacional de sumô robótico do mundo que acontece anualmente em Tóquio, Japão. O maior desafio foi vencer a estatística de que jamais um robô estrangeiro ganhou um round contra um japonês em 23 anos de competição.

1.2 Descrição do trabalho

O objetivo deste projeto é abordar todas as áreas da Engenharia de Controle e Automação, começando pela modelagem mecânica do robô, passando pelos processos de fabricação do mesmo até chegar no controle inteligente.

A complexidade da mecânica e as limitações físicas do projeto fez com que fosse necessário o emprego de diversas técnicas de fabricação além do uso de materiais de diversas naturezas.

Na área da eletrônica, foi desenvolvido uma placa de interface para integrar todos os componentes eletrônicos com o micro controlador.

Dado que a duração média de um round de sumô robótico é por volta de 3 segundos e que os competidores atingem altas velocidades, foi proposta a programação um controle inteligente usando técnicas de Lógica Fuzzy para auxiliar o operador durante os momentos em que a pilotagem se torna humanamente inviável.

O robô desenvolvido será referido como C3D4.

O C3D4 foi desenvolvido para competir em torneios de Sumô Robótico, uma das modalidades robóticas mais populares do mundo.

O Sumô Robótico, assim como o sumô humano, consiste em um duelo entre dois oponentes, com peso máximo de 3Kg e dimensões máximas de 20cmX20cm e altura livre, em uma arena circular em que o objetivo é empurrar o outro para fora. Existem duas categorias nesta modalidade, Sumô Autônomo e Sumô R/C (Rádio Controlado). O C3D4 foi desenvolvido para competir em ambas, porém, o foco do projeto foi na Rádio Controlado uma vez que essa seria a categoria em que ele viria a competir no Japão. Apesar do nome sugerir que haveria apenas o controle humano sobre o robô, as regras japonesas permitem e incentivam que o robô possua algum tipo de controle inteligente que auxilie o operador durante as partidas.

A arena de competição é uma chapa de aço circular de 1,5m de diâmetro e 5mm de espessura com a superfície pintada de preto fosco. Ela possui uma faixa branca de 5cm de largura na borda com objetivo de que o robô reconheça o limite da arena (Figura 1).

Figura 1 Arena oficial da competição

O projeto do robô C3D4 possui três grandes sistemas que podem ser divididos em subsistemas da seguinte maneira:

1.2.1 Mecânica

A Figura 2 representa a composição do subsistema mecânico:

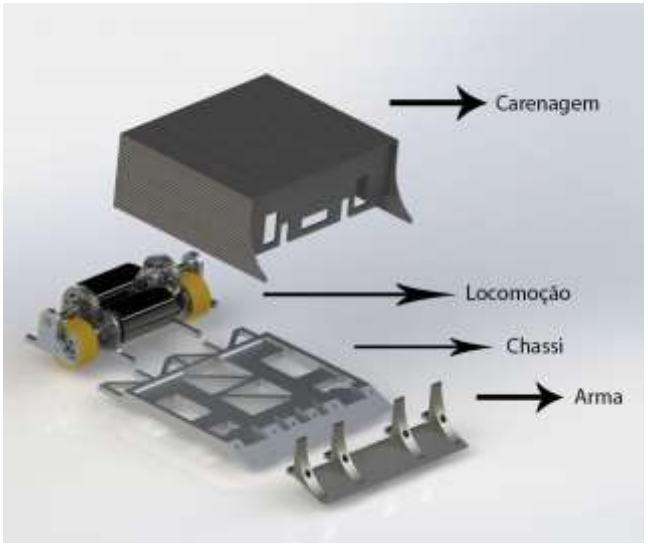

Figura 2 Vista explodida do subsistema mecânico

1.2.2 Eletrônica

A Figura 3 representa a composição do subsistema eletrônico:

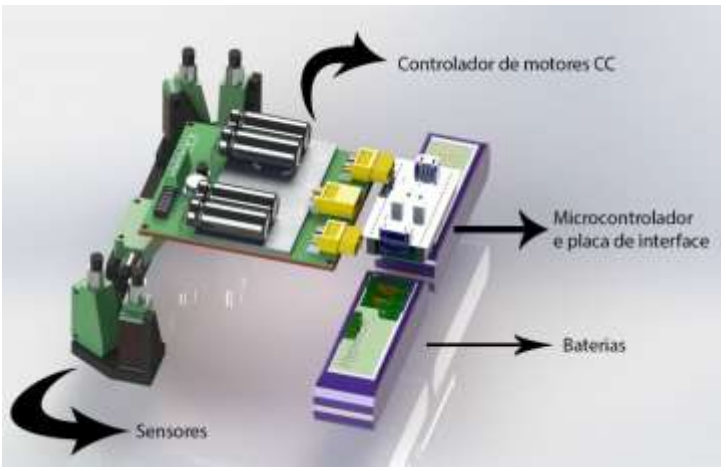

Figura 3 Vista explodida do subsistema eletrônico

1.2.3 Controle

A Figura 4 representa o diagrama de blocos do controle do robô:

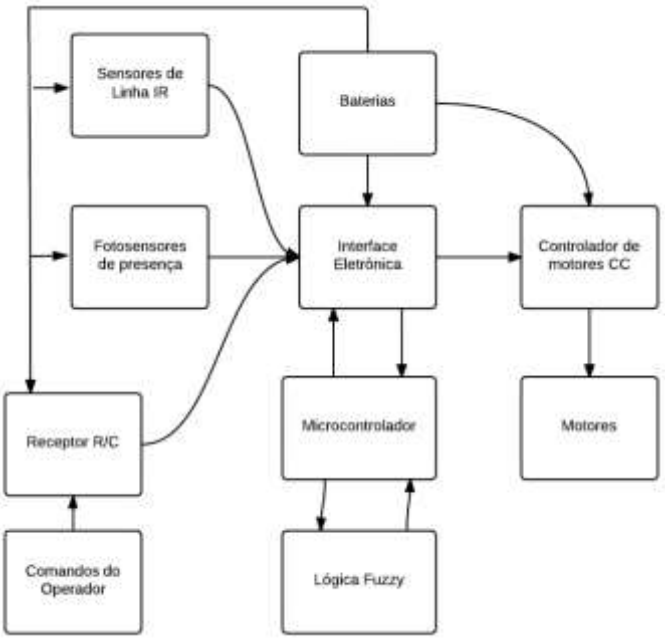

Figura 4 Diagrama de blocos do controle do C3D4

Os próximos capítulos serão divididos da seguinte maneira: O capítulo 2 abordará brevemente diversos conhecimentos que foram utilizados durante o projeto, nas áreas de eletrônica, mecânica e lógicas de programação. No capítulo 3 é apresentado o projeto e a construção do robô C3D4 com a aplicação dos conceitos abordados no capítulo 2. No capítulo 4 serão apresentados os resultados obtidos com o robô. Por fim, no capítulo 5 serão abordadas as conclusões do trabalho.

2 Conceitos Teóricos

2.1 Sumô Robótico

Em 1988 aconteceu o primeiro All Japan Robot Sumo Tournament, sendo ele o primeiro torneio de sumô robótico que se tem documentado. Hoje, 25 anos depois, existem diversas ligas que disputam torneios por todo o mundo.

2.1.1 Competições

Idealizado por Hiroshi Nozawa, Fundador e CEO da FujiSoft Incorporated, uma das maiores empresas de Tecnologia da Informação (TI) do Japão, o All Japan Robot Sumo Tournament é hoje considerado o torneio mais importante do mundo na modalidade (Figura 5).

Até o ano de 2013, o maior evento de robótica de todas as américas acontecia anualmente na Califórnia, EUA. A RoboGames possuía 50 categorias diferentes dentre elas o sumô robótico. Infelizmente em 2013 ocorreu a última edição, na qual a equipe RioBotz foi campeã na categoria Sumô RC de 3Kg, o que rendeu o convite para competir no Japão no fim do mesmo ano e motivou o desenvolvimento deste projeto.

Além da RoboGames, existem competições na Europa que também qualificam para o torneio no Japão.

No Brasil, acontecem uma ou duas vezes por ano eventos organizados pela RoboCore, conhecidos como Winter e Summer Challenge. Devido ao fim da RoboGames e o desempenho excepcional atingido pelo C3D4 no Japão, está sendo negociada a inclusão do Winter Challenge na lista de torneios classificatórios para o evento da FujiSoft.

Figura 5 24th All Japan Robot Sumo Tournament

Apesar de existirem diversas ligas onde ocorrem torneios de sumô robótico, as regras são praticamente as mesmas por todo o mundo.

2.1.2 Regras

Existem apenas duas limitações físicas impostas pelas regras, são elas:

- O Robô precisa ter largura e comprimento igual ou inferior à 20cm no início da partida, tendo a sua altura ilimitada.
- O Robô deve ter no máximo 3Kg.

A arena é composta por uma chapa de aço com 1,5m de diâmetro e 5mm de espessura, sendo ela pintada de preto e possuindo uma borda branca de 50mm em todo o seu perímetro, para que os robôs possam identificar os limites da arena.

A partida tem duração máxima de 3 minutos, porém, em competições de alto nível, a duração raramente excede 10 segundos.

Após o início do round, é considerado perdedor o robô que encostar na superfície fora dos limites da arena primeiro.

Em alguns casos, o juiz pode declarar vencedor um robô que esteja demonstrando agressividade enquanto o outro se encontra parado ou excessivamente passivo.

A figura 6 que explica a regra foi retirada do flyer distribuído no torneio japonês para os espectadores.

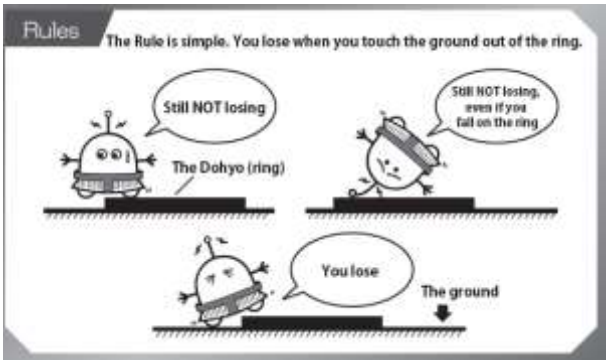

Figura 6 Flyer distribuído no Japão

2.2 Materiais

A escolha dos materiais é o primeiro passo no projeto de um robô que possui limitações de peso e volume. É necessário encontrar materiais com as propriedades que atenda às necessidades de cada peça e que a sua densidade seja ótima, não necessariamente mínima, pois este projeto também possui limitações de volume. A seguir, serão apresentadas as propriedades e as aplicações dos diversos materiais que foram utilizados neste projeto.

2.2.1 Alumínio

O principal atrativo para o uso de alumínio em estrutura de robôs é a sua baixa densidade quando comparado com ligas de aço, aproximadamente 1/3 (2,7g/cm^3 contra 7,8g/cm^3). Mais especificamente se tratando das ligas de alumínio aeronáutico (Ex. AL7075-T6 e AL2024-T3), a sua aplicação se torna ainda mais vantajosa, uma vez que sua resistência se equipara a de algumas ligas de aço como a 1020, com a mesma diferença de densidade, 3x menor.

Na Figura 7 encontram-se as peças do robô que foram feitas em alumínio.

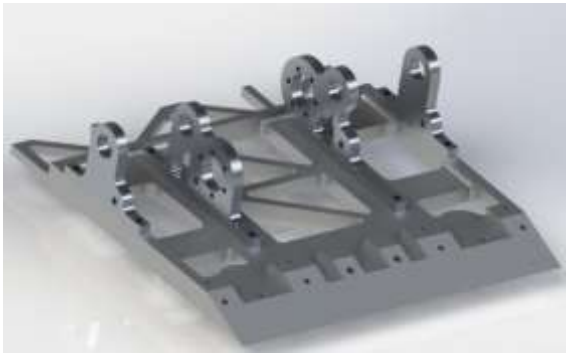

Figura 7 Peças de alumínio do C3D4

2.2.2 Ligas de Aço

As ligas de aço são compostas basicamente de ferro e alguns outros elementos de liga. Dependendo da liga, eles podem ser extremamente resistentes, porém, não é possível utilizar aço no robô inteiro pois o deixaria pesado demais devido à sua alta densidade. No que diz respeito a projetos de sumô robótico, o aço é utilizado em algumas peças como engrenagens, eixos e a lâmina do robô.

Nas engrenagens, são comumente utilizados aços de baixo carbono como o aço 1020. Uma das vantagens do seu uso é seu baixo custo e a facilidade de usinagem/fabricação. Sua baixa dureza combinada com a aplicação de um tratamento térmico superficial chamado cementação minimizam o risco de propagação de trincas nos dentes, aumentando assim a durabilidade das engrenagens.

A lâmina é uma das peças mais importantes do projeto, pois ela é a única peça que deverá entrar em contato com o oponente. Conseguir entrar por baixo da lâmina do adversário durante o round é a maior vantagem mecânica que se pode ter, pois a tração do outro robô ficará drasticamente prejudicada, facilitando assim a vitória.

O aço da lâmina deverá ser extremamente duro para que se possa ter uma boa retenção de corte (dificuldade de perder o fio) e que ainda assim tenha alta resistência para que a lâmina não quebre durante o impacto contra o oponente.

Por mais que não seja desejável que a lâmina quebre por inteiro durante o impacto, devido à sua alta dureza, uma característica que deve-se procurar é o efeito de "chipping". Consiste no fenômeno de pequenas lascas serem arrancadas durante o impacto ao invés do aço simplesmente ficar deformado. Deformações no fio da lâmina podem diminuir drasticamente a efetividade da mesma durante uma luta.

Dada as características necessárias para a aplicação, os aços para trabalho a frio como o D3 e o D6 são recomendados, assim como o VC-131 da Villares Metals. Após o tratamento térmico, a lâmina deverá estar com dureza entre 58 e 60 HRC.

Na figura 8 temos exemplos de peças do C3D4 feitas com ligas de aço.

Figura 8 Peças de aço do C3D4

2.2.3 Polímeros (Plásticos)

A definição química de um polímero diz que são compostos de elevada massa molecular formada pela sequência de monômeros resultantes de reações conhecidas como polimerização. Dentre os diversos tipos de polímeros existentes, foram utilizados dois tipos neste projeto, um termoplástico e um elastômero:

- **ABS (Acrilonitrila butadieno estireno)**
  O ABS é o termoplástico mais utilizado como filamento para impressão 3D e possui boa resistência mecânica. Um bom exemplo de um artigo do cotidiano que seja constituído de ABS são as peças de Lego® (Figura 9), que por diversas gerações já provou o quão resistente o material é.
  Neste projeto, o ABS foi utilizado no processo de impressão 3D do suporte dos sensores do robô.

Figura 9 Peças de LEGO® feitas com ABS

- **Poliuretano (PU)**
  Seu desenvolvimento original tinha como objetivo criar um substituto da borracha natural. Descobriu-se que com pequenas alterações em sua formulação, é possível modificar bastante propriedades como a sua dureza. Tal versatilidade o tornou um produto ideal para aplicações que exijam elastômeros ("borrachas") com durezas específicas. Uma aplicação comum bem próxima da que foi feita no projeto são as rodas de skate (Figura 10) que, nos dias de hoje, são, em sua maioria, de poliuretano.
  As rodas do C3D4 foram feitas a partir de um produto comercial chamado Poly 75-59, fabricado pela PolyTec, uma borracha de poliuretano para moldagem a frio com dureza de 60 Shore A.

Figura 10 Roda de Poliuretano

2.2.4 Compósitos

Não se trata de um material em si, compósitos são associações de materiais de naturezas e propriedades distintas que quando associados apresentam novas propriedades únicas. No C3D4, a fibra de carbono foi o compósito utilizado (Figura 11). As fibras carbônicas sozinhas não são apropriadas para uso, porém, ao serem combinadas com materiais matrizes como a resina epóxi, estas resultam num material com propriedades mecânicas excelentes. O resultado desta combinação é conhecido como CFRP (Carbon Fiber Reinforced Plastic) e seu uso vem crescendo cada vez mais em indústrias como a aeroespacial e automobilística de alto desempenho dado seu alto fator rigidez/densidade (E/ρ entre 44 e 96).

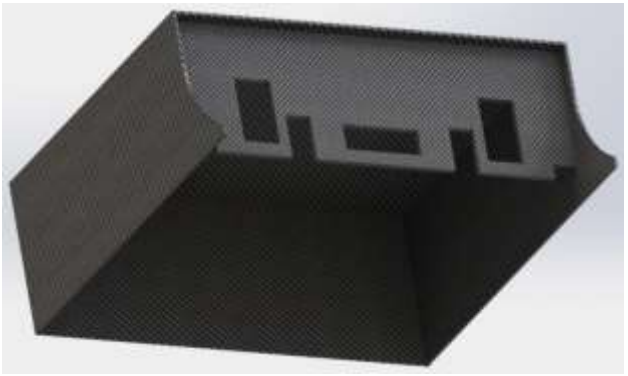

Figura 11 Carenagem de Fibra de Carbono do C3D4

2.3 Processos de Fabricação

Ao se desenvolver um robô, é necessário projetá-lo pensando sempre em como cada peça será fabricada a fim de otimizar a produção da mesma e consequentemente minimizar o custo. A seguir, serão apresentados diversos processos de fabricação que foram utilizados durante a realização deste projeto.

2.3.1 Fresa

Fresamento é um processo de usinagem que se caracteriza pelo movimento combinado de rotação de ferramenta e translação relativa da peça. O corte é realizado por uma ferramenta muiticortante (fresa), na qual cada aresta retira uma pequena quantidade de material na forma de cavacos não contínuos.

Uma fresadora pode realizar uma grande variedade de trabalhos tridimensionais, o que gera uma vantagem em relação a outros processos de usinagem. O corte pode ser realizado em planos paralelos, perpendiculares, ou formando ângulos diversos: construir ranhuras circulares, elípticas, fresagem em formas esféricas, côncavas e convexas, com rapidez e precisão.

Existem diversos tipos de fresadoras, mas podemos citar dois tipos principais: vertical e universal. Na fresadora vertical, usinam-se tradicionalmente peças de grandes dimensões, na qual a peça pode se deslocar em dois eixos sendo o eixo-árvore da máquina perpendicular à mesa da mesma. Na fresadora universal, a peça pode se deslocar em até três eixos (dependendo do modelo) e é equipada com diversos acessórios especiais que permitem a usinagem de quaisquer superfícies tridimensionais.

Na Figura 12 é possível observar o trabalho de fresagem sendo feito no chassi do C3D4.

Figura 12 Chassi do C3D4 sendo fresado

2.3.2 Torno

O torno é uma máquina de usinagem utilizada na fabricação de peças com geometria de revolução, com o auxílio de uma ou mais ferramentas monocortantes (com apenas uma superfície de saída). Para isso, a peça gira em torno de um eixo principal de rotação da máquina e a ferramenta se desloca simultaneamente com uma trajetória no mesmo plano do eixo. Dessa forma, várias operações podem ser realizadas: furação, torneamento cilíndrico, cônico, radial e curvilíneo.

No torno foram feitos os hubs das rodas e os eixos dos sistemas de redução.

2.3.3 Centro de usinagem CNC

CNC é o acrônimo de Computer Numeric Control, ou controle numérico computadorizado. Esta tecnologia permite o controle, por meio de um código computacional, dos movimentos de máquinas operatrizes, sendo largamente utilizado em tornos, fresas e centros de usinagem na produção de peças complexas e de grande precisão.

Apresenta vantagem na produção de peças seriadas, reduzindo tanto o tempo de usinagem quanto o número de erros humanos na produção. Esta vantagem também se estende a peças exclusivas com grande complexidade ou necessidade de precisão.

Devido à necessidade de grande precisão, o chassi do robô foi feito em uma fresa CNC de 3 eixos e este trabalho pode ser visto na Figura 13.

Figura 13 Chassi do C3D4 sendo usinado em um centro de usinagem CNC

2.3.4 Eletroerosão à fio

É um processo de tipo térmico na qual o material condutor sofre erosão através de sucessivas descargas elétricas que fundem localmente o material.

Esta técnica possibilita gerar quaisquer geometrias bi ou tridimensionais e trabalhar com quaisquer ligas metálicas, obtendo-se ótimo acabamento superficial quando controlado corretamente os parâmetros do processo (Tempo de ciclo, tempo de descarga elétrica, corrente de descarga e polaridade).

O processo foi utilizado na produção de 11 cópias dos "chifrinhos" (Figura 14) do C3D4 pois, devido a sua complexa geometria as técnicas tradicionais de usinagem não se aplicavam.

Figura 14 Peças feitas com o processo de eletroerosão a fio

Figura 15 Equipamento de eletroerosão a fio onde foram cortadas peças do C3D4

2.3.5 Retífica

A retificação é um processo de usinagem por abrasão de acabamento superficial que possibilita a obtenção de baixas tolerância (IT4 e IT6) e rugosidade (Ra de 0,2 a 1,6 mícrons).

Tem como objetivo a adequação de dimensões que necessitem de exatidão, remoção de camadas finas de materiais endurecidos e redução de saliências e/ou rebaixos em superfícies usinadas.

Como pode-se observar na Figura 16, as engrenagens do sistema de redução do C3D4 foram retificadas para chegar com mais exatidão na medida de sua espessura.

Figura 16 Engrenagens do C3D4 que foram retificadas

2.3.6 Laminação de carbono

O processo elementar de laminação de compósitos consiste em banhar as fibras com a matriz de resina, segundo uma determinada proporção que varia com o projeto. São utilizados principalmente dois tipos de resina: Poliéster que é mais fraca mas também mais barata, e epóxi que é a mais resistente mas também mais cara. A matriz então se infiltra entre os filamentos, colando-os e mantendo-os no lugar. Assim o componente toma forma.

O processo consiste basicamente em banhar as fibras com a matriz sobre um molde, camada por camada, retirando o excesso de resina com uma espátula e distribuindo-a com um rolo, que ajuda também a evitar a formação de bolhas de ar que prejudicarão significativamente as propriedades mecânicas do componente.

Para obter peças mais leves e resistentes utiliza-se o método de laminação a vácuo, também conhecido com a expressão em inglês "vacuum bagging". A laminação a vácuo é um refinamento, um aprimoramento do processo de laminação manual. No entanto, a peça a ser produzida é selada em uma bolsa plástica que por sua vez é conectada através de tubos, mangueiras e válvulas, a uma ou mais bombas de vácuo. Uma vez acionadas as bombas, o ar é retirado de dentro da bolsa dentro da qual está contido o laminado, criando uma pressão em seu interior que é menor que a pressão atmosférica normal. Isso ajuda a compactar a peça, minimizar as bolhas de ar e, através de filmes absorventes, remover o excesso de resina, obtendo-se assim uma peça com alta qualidade.

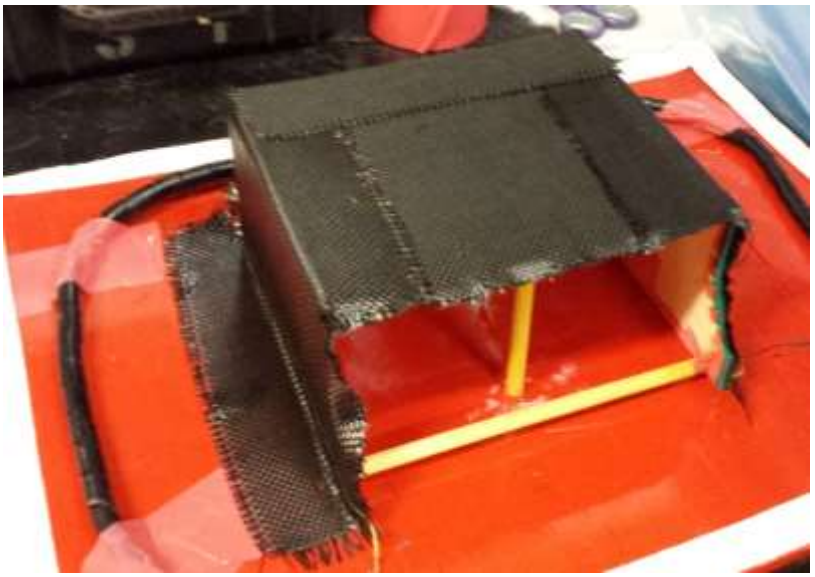

Figura 17 Carenagem do robô C3D4 sendo produzida

2.3.7 Corte à Laser

Acrônimo de Light Amplification by Stimulated Emission of Radiation, é um processo do tipo térmico que desfruta da interação entre um feixe de luz coerente e monocromática com a peça a ser trabalhada. Tem como base física o conceito de emissão estimulada de um átomo (decaimento energético de um nível superior a um inferior, provocando a liberação de um fóton, o qual formará o feixe Laser).

As fontes de Laser são variadas (Gás, sólidos ou Diodos), com diversos tipos de transporte do feixe (espelhos planos, côncavos e fibras) e sua sucessiva focalização (lentes, espelhos ou fibras óticas). Todos estes fatores são escolhidos de acordo com a potência do Laser e sua aplicação.

A técnica fui utilizada na fabricação dos moldes (Figura 18), em madeira, para a posterior laminação em fibra de carbono do robô. Utilizou-se um Laser a gás (CO_2) com fonte selada e movimentação de espelhos planos.

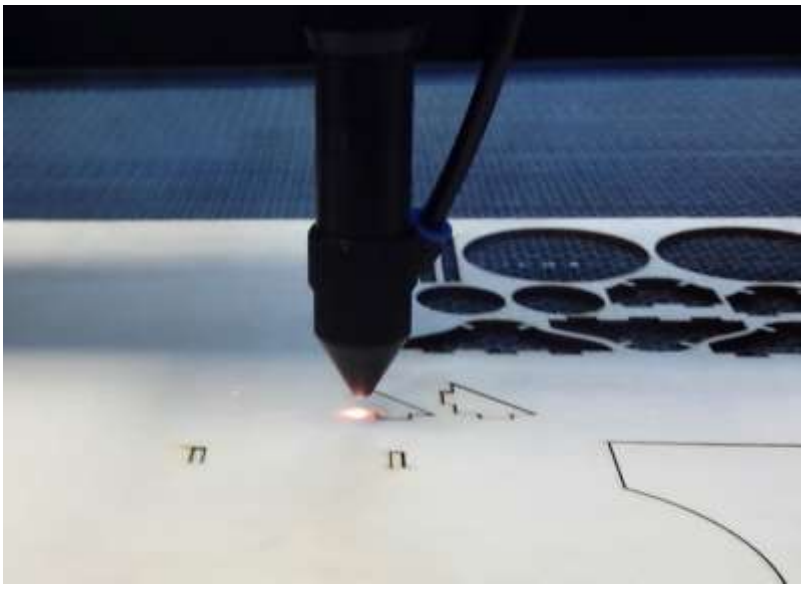

Figura 18 Molde da carenagem sendo cortado à Laser

2.3.8 Impressão 3D

Impressoras 3D montam objetos, camada por camada, a partir de pedaços de materiais, da mesma forma que as impressoras tradicionais criam imagens de pontos de tinta ou toner - mas em três dimensões.

Esse método de fabricação é chamado de aditivo, em oposição à produção subtrativa, que remove as partes de que não se precisa a partir do material bruto. A impressão 3D tem início a partir do nada: começa adicionando materiais, camada por camada, até que o item esteja pronto.

A peça tridimensional é criada em um software CAD e posteriormente enviada à impressora 3D.

Neste projeto, o suporte dos sensores foi impresso em uma impressora 3D idêntica à da Figura 19.

Figura 19 Impressora 3D utilizada para imprimir o suporte dos sensores

2.3.9 Moldagem a frio

O processo consiste na mistura de um polímero com seu catalisador, iniciando-se assim o processo de endurecimento do material (curagem). Após a mistura ficar homogênea, despeja-se no molde que dará a forma final à peça. Devido à alta viscosidade do liquido e da presença de bolhas obtidas durante o processo de mistura, é recomendado retirar o excesso de ar por meio de um processo conhecido como "degassing". O processo consiste em colocar o molde preenchido dentro de uma câmara de vácuo para que as bolhas tendam a emergir para a superfície da mistura até que não existam mais bolhas no interior da mesma.

A Figura 20 mostra a roda do robô C3D4 sendo moldada.

Figura 20 Roda do robô sendo moldada

2.4 Software de CAD

Acrônimo de Computer Aided Design, os Softwares de CAD são utilizados para modelar peças mecânicas que serão fabricadas ou já existentes. Com as peças modeladas, é possível montar sistemas mecânicos e posicioná-las de acordo com o projeto, montando assim uma versão virtual completa do que se deseja fazer. O projeto do C3D4 foi totalmente desenvolvido no software Solidworks da Dassault Systemes (Figura 21).

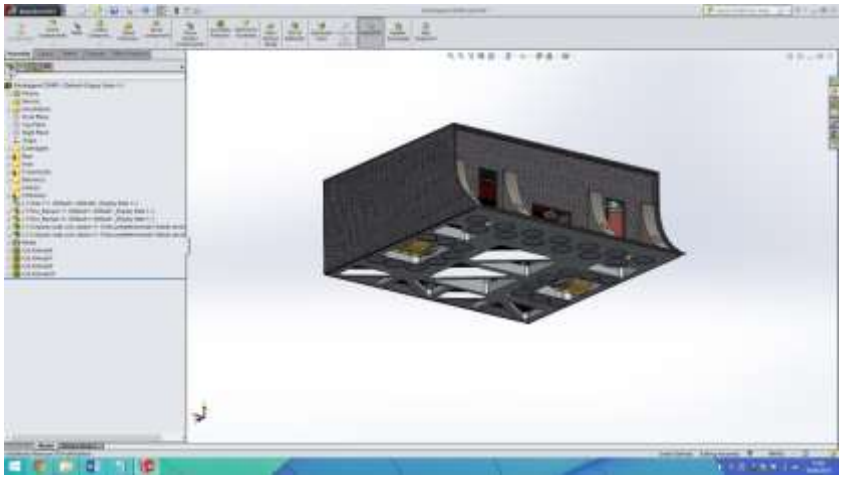

Figura 21 C3D4 no Software SolidWorks

2.5 Caixas de redução

Motores muitas vezes possuem sua rotação de trabalho acima da que é preciso na saída do sistema e para reduzir esta rotação, conjuntos de engrenagens são acoplados entre o eixo do motor e o eixo de saída de modo que a relação de diâmetros entre as engrenagens dará o fator de desmultiplicação (redução) da rotação e paralelamente multiplicará o torque resultante.

Figura 22 Redução do C3D4

2.6 Rodas

A roda é outro elemento de grandíssima importância no projeto, pois é o responsável por transferir a potência dos motores para o solo gerando assim o movimento do robô. Três fatores são importantes na escolha do material: Aderência, dureza e durabilidade. A aderência é quem vai garantir que toda a rotação do eixo será convertida em movimento, não havendo assim escorregamento (derrapagem). A dureza está altamente ligada à aderência, pois quando menos dura for a roda, maior será a deformação da mesma e consequentemente haverá maior área de contado com o solo, aumentando assim a aderência final. Já a durabilidade, está diretamente ligada à dureza, pois quanto mais dura for a roda, maior será a sua durabilidade. O desafio é justamente equilibrar a aderência e a durabilidade variando a dureza do material escolhido.

2.7 Imãs

Como a arena da competição é feita de aço, todos os robôs utilizam imãs permanentes em seu fundo para aumentar a aderência das rodas e a dificuldade para ser empurrado para fora. Como a força magnética do imã em relação à chapa de aço é inversamente proporcional à distância entre os dois, o objetivo é deixar o imã o mais próximo possível do chão sem que se encostem. A partir do momento em que o contato ocorre, passa a existir atrito entre o fundo do robô e o chão, dificultando bastante a locomoção do mesmo.

Dentre os diversos tipos de imãs que existem no mercado, os de neodímio são os que possuem maior força. Comercialmente, esse tipo de imã possui uma escala gradual que vai de N35 à N52, do mais fraco pro mais forte. Além do grau, existem imãs que possuem uma armadura de aço que potencializa sua força de atração em chapas. Na figura 23 é possível comparar que as linhas de campo ficam visivelmente mais concentradas no imã com armadura (esquerda) do que no sem (direita).

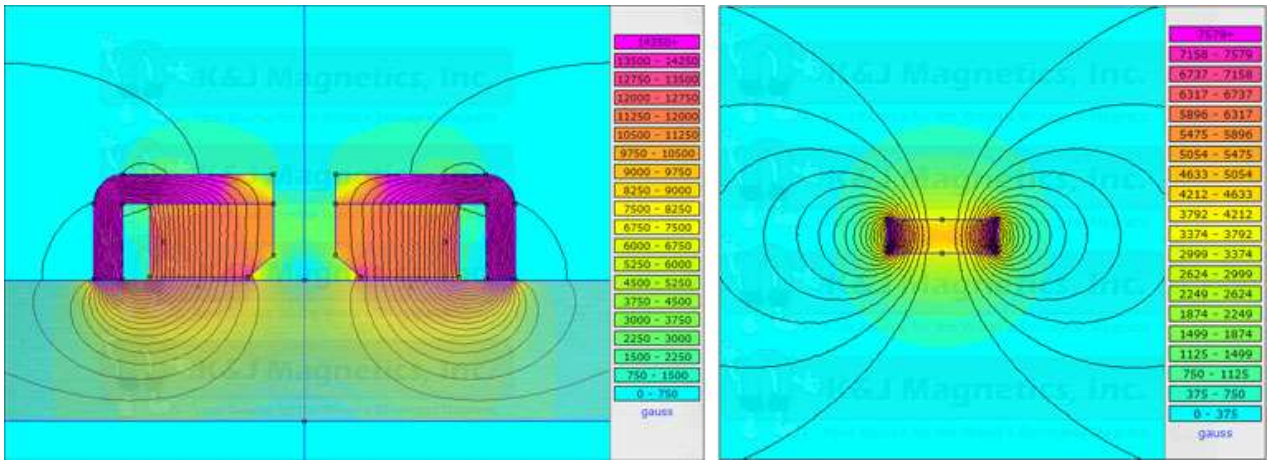

Figura 23 Fluxo de campo magnético em ima com armadura (esquerda) e sem (direita)

2.8 Baterias

A bateria é o componente do robô que determinará a sua autonomia, além de ter grande influência no peso final do projeto. Dentre todas as químicas existentes no mercado, as de Lítio Polímero (LiPo) são as que possuem maior densidade de carga por peso, tornando-as unanimidade em projetos de sumô robótico. O único contra que ela possui é a instabilidade de sua química, que pode entrar em combustão caso seja descarregada abaixo da tensão permitida (3V por célula), acima da corrente máxima de descarregamento ou em caso da célula ser rompida.

2.9 Controladores de velocidade de motores de corrente contínua (CC)

No caso específico dos motores de corrente contínua (CC) o controlador é responsável por controlar a tensão (V) que será enviada aos motores, sendo ela uma fração da tensão máxima fornecida pelas baterias do sistema. Além da velocidade, os controladores também são responsáveis por comandar o sentido em que o motor vai rodar.

2.10 Microcontroladores

Um microcontrolador pode ser considerado como um computador reduzido à apenas um chip. Ele sozinho é capaz de rodar um programa que lê entradas analógicas e digitais, realizar operações lógicas e aritméticas e disponibilizar saídas digitais e analógicas com os resultados do processamento. Neste projeto, foi adotado o uso do mbed, um Microcontrolador/plataforma desenvolvido pela NXP. Assim como o Arduíno, seu concorrente mais famoso, o mbed é vendido pronto para ser usado, possui uma grande comunidade de programadores que disponibilizam diversas bibliotecas para serem reaproveitadas em novos projetos. O diferencial dele em relação ao Arduino é o seu superior poder de processamento, pois o clock do processador roda à 96MHz contra apenas 16MHz.

2.11 Sensores fotoelétricos

Dentre os diversos tipos de sensores fotoelétricos existentes, todos os sensores utilizados neste tipo de aplicação se encaixam na mesma família de sensores, são sensores fotoelétricos de detecção por reflexão difusa. Este processo de detecção consiste no princípio de que um emissor irá emitir uma luz, seja ela infravermelha ou visível, esta luz será refletida (ou não) por um objeto e de acordo com a intensidade/presença de luz recebida de volta no receptor, este é capaz de aferir a presença ou não de um objeto à sua frente, em alguns casos o receptor é capaz de aferir também a distância em que o objeto se encontra.

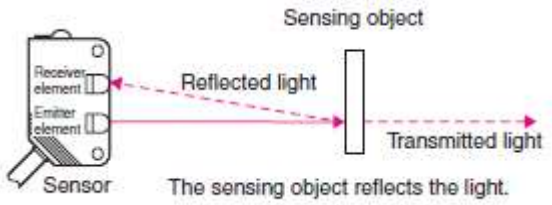

Figura 24 Diagrama de um sensor fotoelétrico

2.12 Pulse Width Modulation (PWM)

O PWM pode ser utilizado como um método de modularizar a tensão de saída em relação a tensão de entrada em um sistema de potência mas o seu conceito também pode ser usado como um protocolo de comunicação. No segundo caso, este protocolo é amplamente utilizado na indústria de rádio controles para a prática de modelismo.

Como é possível observar na figura 25, o protocolo consiste em um pulso que deve ser enviado à cada 20ms e a largura do pulso é que contêm a informação. Sendo 1ms a largura mínima e 2ms a largura máxima. Na aplicação deste projeto, um pulso de 1ms recebido pelo receptor do rádio controle é convertido para o valor -100 pelo Microcontrolador e um pulso de 2ms é convertido para +100, sendo o valor zero equivalente à um pulso de 1,5ms.

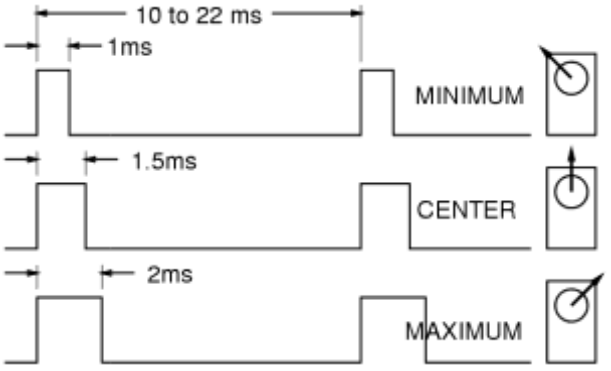

Figura 25 Como funciona o PWM

2.13 Lógica Fuzzy

A Lógica Fuzzy é uma generalização da lógica booleana que admite valores lógicos intermediários entre o "falso" (0) e o "verdadeiro" (1), como por exemplo o talvez (0,5).

Como existem várias formas de se implementar um modelo Fuzzy, a Lógica Fuzzy deve ser vista mais como uma área de pesquisa sobre tratamento da incerteza, ou uma família de modelos matemáticos dedicados ao tratamento da incerteza, do que uma lógica propriamente dita.

Fuzzy, em inglês, significa incerto, duvidoso. Expressa exatamente os valores com que esta lógica lida. Com Lógica Fuzzy, não se trata uma variável como tendo apenas um estado atual, mas sim com 'n' estados, cada um com um grau de associação. Em outras palavras, não se afirma que uma casa é grande, mas sim que ela é 0,8 grande, 0,2 média e 0,0 pequena. Isto faz com que se definam conjuntos em que um dado valor pode ser enquadrado. Voltando ao exemplo da casa teríamos três conjuntos: casas grandes, médias e pequenas. Nada impede que se tenham cinco conjuntos: casas enormes, grandes, médias, pequenas e minúsculas. O número de conjuntos deve ser ajustado de acordo com o tipo de dinâmica do problema e com a precisão que se deseja tratar o problema.

Finalmente, define-se um sistema Fuzzy, que será uma coleção de variáveis de entrada (sendo cada uma, uma coleção de conjuntos), uma coleção de conjuntos para a variável de saída e uma coleção de regras que associam as entradas para resultar em conjuntos para a saída.

É necessária ainda uma função que "defuzzifique" a saída, ou seja, que a partir dos graus de pertinência de cada variável linguística de uma regra, implique em um grau de pertinência da variável de saída, que é defuzzificado para um valor crisp.

2.13.1 Conjuntos Fuzzy e Defuzzificação

Os conjuntos Fuzzy constituem uma "ponte" no caminho de aproximar o raciocínio humano ao da lógica executada pela máquina.

Matematicamente, são funções que atribuem graus de pertinência dos valores de entrada àquele conjunto. Como no raciocínio humano, tais conceitos e definições dos conjuntos se sobrepõem, de modo que um valor não precisa deixar de pertencer a um conjunto para pertencera outro.

Assim, um conjunto definido como, por exemplo, "muito grande" receberá um grau de pertinência de acordo com a entrada, bem como o conjunto "grande" terá também o seu grau definido, podendo ambos ser diferentes de zero. A figura 26 mostra a sobreposição de 3 conjuntos Fuzzy.

Este processo de atribuição de graus de pertinência é chamado Fuzzificação.

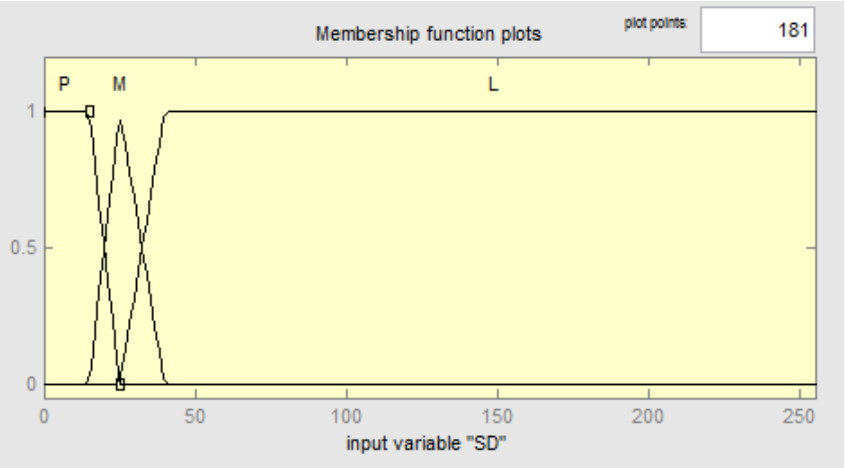

Figura 26 Sobreposição dos conjuntos Fuzzy

2.13.2 Inferência

A inferência é o processo pelo qual o controlador Fuzzy combina os sinais de entrada (já fuzzificados) com a base de regras definida para a obtenção dos conjuntos Fuzzy de saída.

Esta inferência pode ser realizada por diferentes métodos. Um dos métodos é o "Máximo dos Mínimos". Nesse método, todas as regras recebem os graus de pertinência relevantes àquela regra e é calculado o mínimo das pertinências, atribuindo assim, um grau de pertinência daquela regra. Feito isso, a regra com maior pertinência é selecionada a partir de uma função de máximo e será incluída no processo de Defuzzificação.

2.13.3 Deffuzificação

A Defuzzificação pode ser realizada através de diversos métodos, os mais comuns são o método do centroide, que calcula o centroide da área composta pelas funções de pertinência de saída; o método do maior dos máximos (LOM); e o método da média dos máximos (MOM).

3 O Projeto

3.1 Mecânica

O Projeto do robô C3D4 foi bastante influenciado por projetos anteriores e sofreu inspiração de projetos de outros competidores internacionais. O robô foi inteiramente projetado no programa de CAD SolidWorks. A seguir, será detalhada toda a construção mecânica do robô.

3.1.1 Chassi

Para o desenvolvimento do chassi, foi feito um desenho baseado em treliças para redução do peso total da estrutura sem perda de resistência. Uma estrutura compacta (não treliçada), foi usada como base para a fixação dos imãs, os quais geram a forma de atração com a placa metálica da arena, sendo estes as fontes de flexão presentes no chassi.
Utilizou-se um perímetro inteiriço de seção retangular (de altura 12 mm e largura 6 mm) para auxiliar no aumento da rigidez da estrutura. Este incremento de rigidez se faz necessário, pois a força gerada pelos imãs fixados a estrutura não treliçada do chassi aumenta ao reduzir do clearance. Tais forças de atração geradas pelos imãs são as indutoras de flexão na estrutura, o que se não controlado pode acarretar no contato entre chassi e arena, no posterior bloqueio do sistema de locomoção, colocando os motores em stall e consequentemente queimando por superaquecimento.

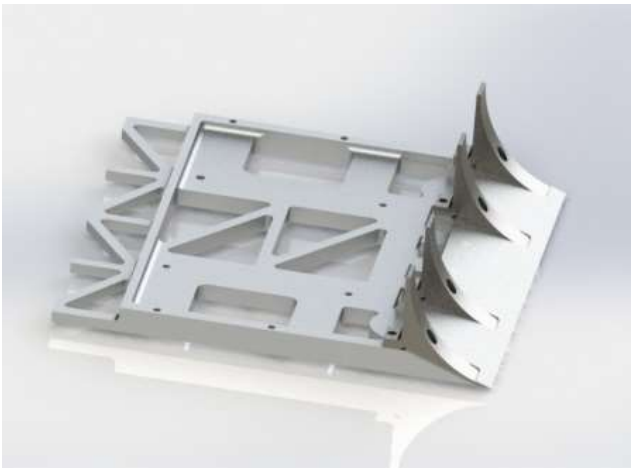

Figura 27 Visão geral do chassi

A lâmina utilizada foi feita sob medida em uma empresa especializada em fabricar facas industriais para corte de papel e celulose. Feita em aço VC-131 da Villares Metals, a peça com 2mm de espessura foi temperada à uma dureza de 60HRC.
O ângulo escolhido para a lâmina foi de 18 graus, 2 graus à menos que a inclinação em que ela viria a ser fixada no robô em relação ao chão (20 graus). Esses 2 graus de diferença garantem que apenas o fio da lâmina estará em contato com o solo o tempo todo (Figura 28).

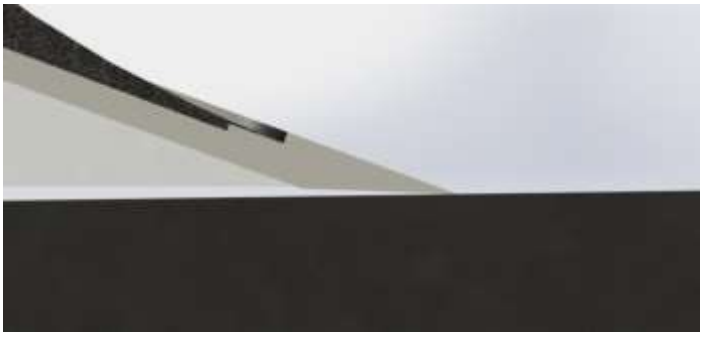

Figura 28 Detalhe da ponta da lâmina

3.1.2 Transmissão

Para a transmissão do robô, foi desenvolvida uma redução por engrenagens de dentes retos de dois estágios. O eixo que contém as engrenagens intermediárias é apoiado nos dois extremos em rolamentos de agulha, enquanto o eixo em que se encontra a roda é fixo, ficando assim os rolamentos no interior do conjunto roda-engrenagem. A redução total do sistema é de 6,2:1. O sistema completo pode ser observado na figura 29.

Figura 29 Transmissão completa

3.1.3 Suporte dos sensores

O suporte dos sensores foi modelado de forma que eles fiquem posicionados com um ângulo de 45 graus entre cada um deles. A peça foi impressa em uma impressora 3D em ABS (Figura 30).

Figura 30 Suporte dos Sensores

3.1.4 Carenagem

A carenagem do C3D4 foi feita inteiramente de fibra de carbono, tendo sido necessário cortar a laser o molde em madeira onde a fibra viria a ser moldada (Figura 31).

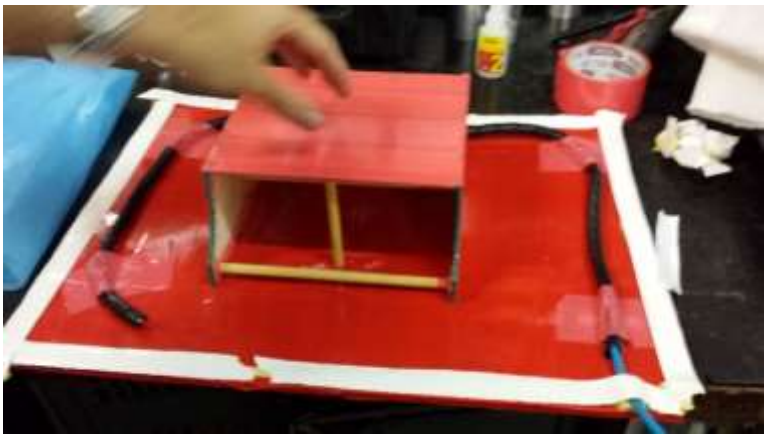

Figura 31 Molde de madeira sendo preparado

Após a montagem do molde, as camadas de carbono foram banhadas em resina epóxi (Figura 32).

Figura 32 Fibra de carbono sendo banhada

Já impregnadas de resina epóxi, as camadas foram sobrepostas sobre o molde e uma camada de feltro foi colocada por cima para ajudar na absorção do excesso de resina na peça final (Figura 33).

Figura 33 Feltro sendo colocado por cima da fibra de carbono

Por fim, o molde é hermeticamente fechado com um plástico especial para que seja aplicado o vácuo (Figura 34).

Figura 34 Molde hermeticamente fechado

Após 12 horas sob vácuo, a peça já está finalizada e apenas precisa ser removida do molde.

3.2 Eletrônica

Para este projeto foi necessário especificar diversos componentes eletrônicos e desenvolver uma placa de interface para interligar todos os periféricos.

3.2.1 Microcontrolador

Foi selecionado como Microcontrolador o mbed modelo LPC1768 para controlar o C3D4. Seu maior poder de processamento e sua fácil utilização foram os diferenciais para escolher ele no lugar do Arduino. A seguir veja um quadro comparativo entre os dois principais competidores:

	NXP mbed LPC1768	**Arduino nano 3.0**
Microcontrolador	ATmega328	ARM Cortex M3
Velocidade do Clock	96MHz	16MHz
Nível Lógico	3,3V	5V
Pinos de Entrada/Saída (I/O)	26	14
Periféricos	2xI2C,3xUART,2xSPI,1xCAN,1xEthernet,1xUSB-Host	1xI2C,1xUART,1xSPI

Figura 35 mbed LPC1768

Outro diferencial do mbed que foi introduzido na sua última atualização de firmware é o suporte ao protocolo CMSIS-DAP da ARM que permite fazer debug em tempo de execução, acompanhando o funcionamento do programa em tempo real.

3.2.2 Sensores

Os sensores utilizados possuíam duas funcionalidades distintas: Detectar a linha branca na extremidade da arena e detectar o oponente.

Para a primeira funcionalidade, foi escolhido o sensor QRD1114 da Fairchild (Figura 36).

Figura 36 Sensor QRD1114

Ele consiste em um LED infravermelho que atua como emissor e um fototransistor como receptor. Afastado de qualquer superfície ou próximo à uma superfície escura, a luz refletida não é suficiente para saturar o fototransistor e a saída do mesmo é mantida em nível lógico 0, ao se aproximar de uma superfície clara, a mesma reflete grande parte da luz emitida pelo LED e isso faz com que o fototransistor sature e o nível lógica troca para 1.

Foram utilizados dois sensores e foram posicionados nas extremidades frontais do C3D4, de acordo com a figura 37.

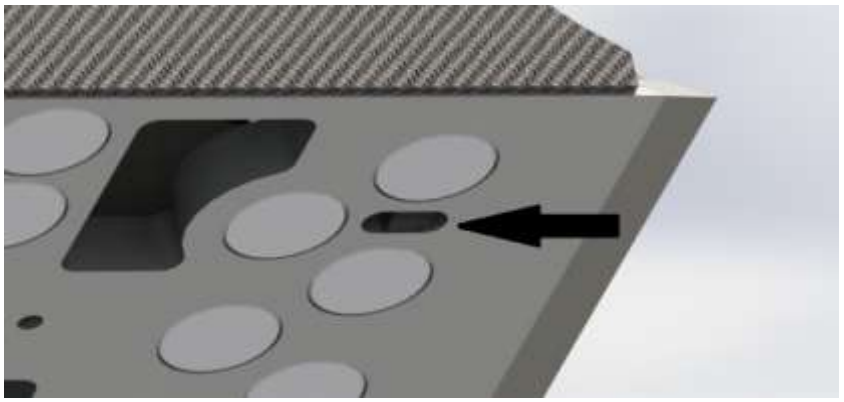

Figura 37 Posição do sensor de linha

Para a detecção do oponente, foram selecionados dois modelos de sensores com características distintas:

O Pepperl Fuchs ML100 (Figura 38) é um sensor industrial de luz vermelha que detecta objetos distantes em até 1 metro. Sua principal característica é a sua frequência de leitura e consequentemente seu tempo de resposta para detectar o oponente. Porém, a saída do sensor é binária, 0 ou 1, não informando a distância do objeto detectado e com isso, inviabiliza o uso de técnicas de Lógica Fuzzy para controlar o robô.

Figura 38 Sensor ML100 da Pepperl Fuchs

Já o Sharp GP2Y0A21YK0F (Figura 39) é um sensor de luz infra vermelha que detecta objetos entre 10 e 80 centímetros. Sua principal característica é sua saída analógica que indica a distância para o objeto. Porém, por não ser um sensor tão robusto quanto o da Pepperl Fuchs, seu tempo de resposta é baixo, tornando sua utilização incompatível com as velocidades envolvidas na competição. Ele foi apenas utilizado para tornar viável o uso da Logica Fuzzy.

Figura 39 Sensor Sharp GP2Y0A21YK0F

A seguir temos o quadro comparativo entre os dois modelos de sensor:

	Pepperl Fuchs ML100	**Sharp GP2Y0A21YK0F**
Tipo de luz emitida	Luz Vermelha	Infra-vermelho
Distância de detecção	0-100cm	10-80cm
Frequência de leitura	1KHZ	N/A
Tempo de Resposta	0,5ms	40ms
Saída	Digital (0 ou 1)	Analógica (10-80cm)

Para a detecção do oponente foram utilizados 5 sensores apontando em diferentes ângulos, permitindo assim definir também a direção em que o oponente se encontra. Defasados de 45 graus, os sensores permitem um campo de visão de 180 graus (Figura 40).

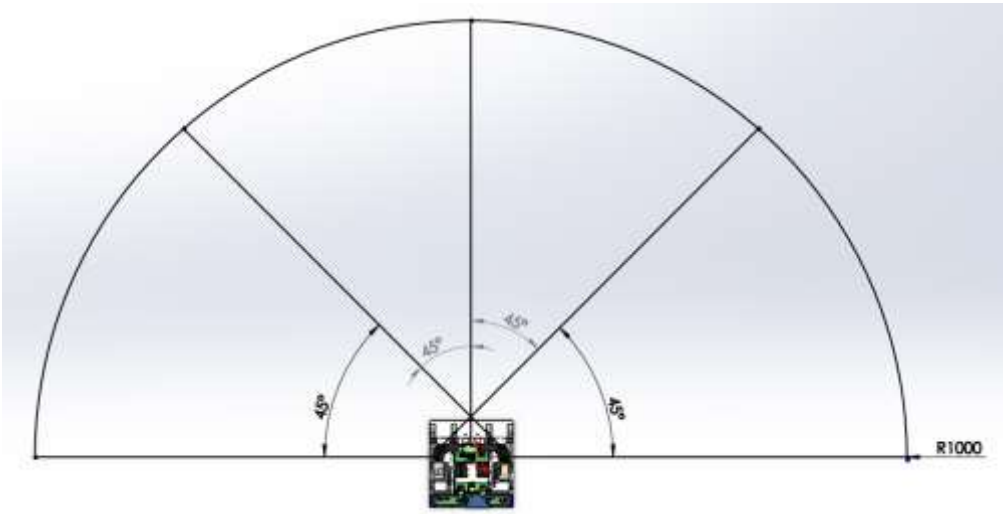

Figura 40 Campo de Visão dos sensores

3.2.3 Controlador de motor CC

Para este projeto, foi selecionado o controlador Sabertooth 2x60 da Dimension Engineering. É um controlador que possui dois canais para controlar dois motores simultaneamente e é capaz de fornecer até 60A de corrente em regime contínuo por canal.

3.2.4 Controle remoto por Rádio Frequência (RF)

Para receber comandos do operador, é necessário o uso de um sistema de transmissão por rádio frequência. Neste projeto foi utilizado o Conjunto transmissor e receptor da marca Hobby King e modelo HK-300 (Figura 41). Foram utilizados 3 canais de comando, sendo eles:

- Canal 1

Controla a potência que deverá ser enviada aos motores controlando assim a velocidade do robô.

- Canal 2

Controla a diferença de potência que deve ser enviada aos motores, permitindo assim que o robô faça curvas.

- Canal 3

Apenas uma chave liga desliga que habilita/desabilita o loop de controle inteligente.

Figura 41 Controle por RF da Hobby King

3.2.5 Placa de Interface

Devido à grande quantidade de periféricos à serem ligados ao Microcontrolador, foi necessário o desenvolvimento de uma placa de circuito impresso para servir de interface entre o mbed e os diversos dispositivos conectados a ele (Figura 42).

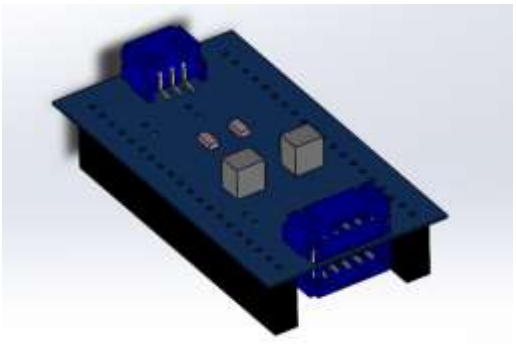

Figura 42 Imagem 3D da placa de interface

A seguir podemos ver o esquemático do circuito mostrando todas as ligações de entrada e saída utilizadas no mbed:

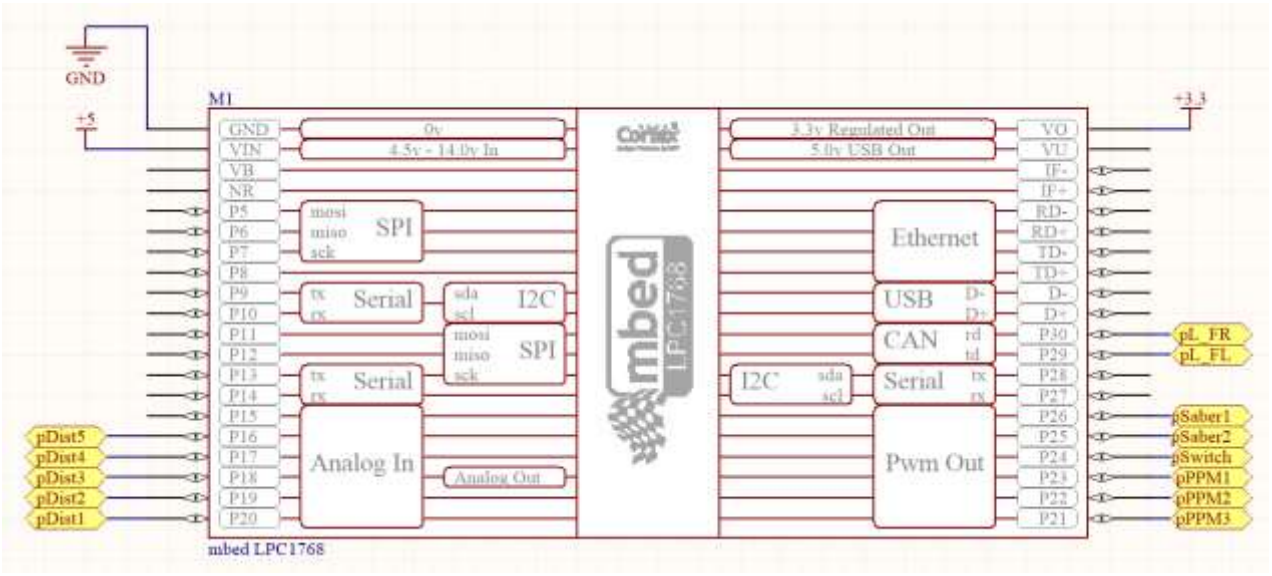

Figura 43 Esquema das entradas e saídas do mbed

Para a ligação dos sensores de detecção de oponente, foi necessário fazer um divisor de tensão para cada entrada pois a tensão de saída dos sensores ML100 são acima do máximo suportado pelo mbed, além disso o conector também era responsável por alimentar todos os sensores.

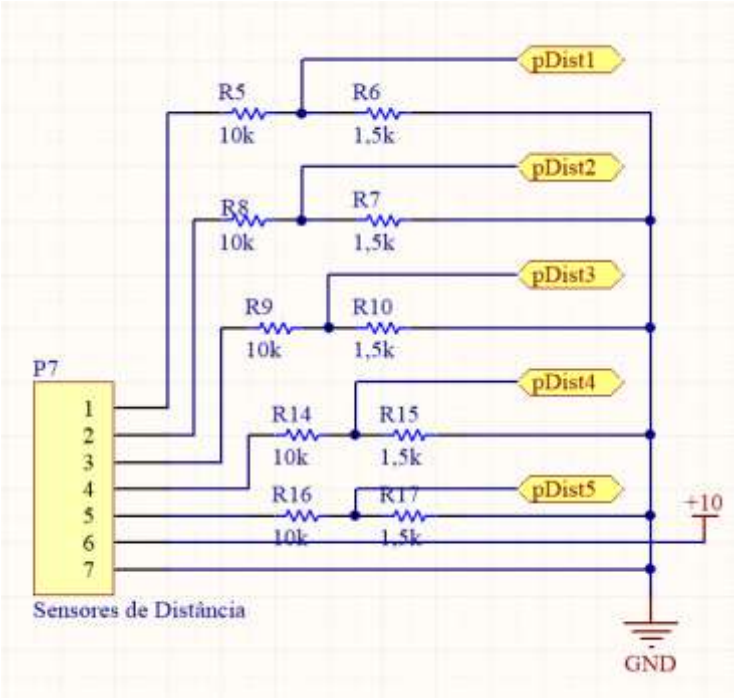

Figura 44 Esquemático da ligação com os sensores

Para a comunicação com o controlador de velocidade dos motores, foi necessário desenvolver um filtro RC para filtrar os dois sinais de entrada na Sabertooth, uma vez que a mesma é controlada por um sinal analógico para cada motor.

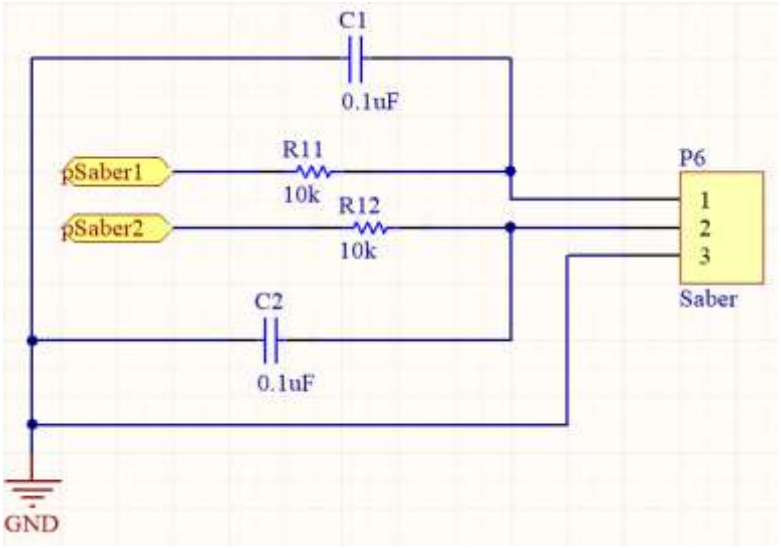

Figura 45 Esquemático da ligação com o controlador de velocidade

Já para a ligação dos sensores de linha foram utilizados resistores variáveis para ajustar a sensibilidade do fototransistor de acordo com a luminosidade do local e as cores da arena, além da alimentação dos LEDs infravermelhos.

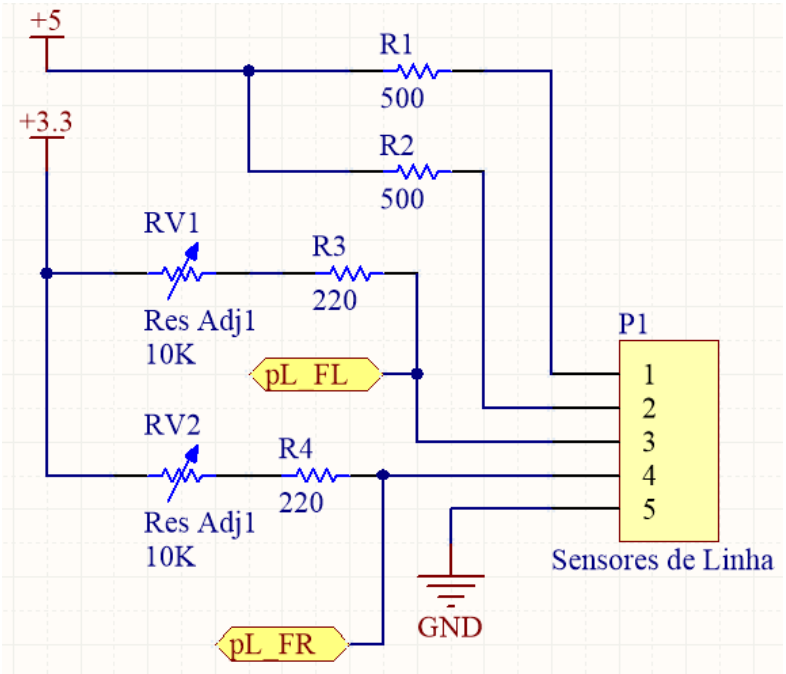

Figura 46 Esquemático de ligação com os sensores de linha

No que diz respeito a conexão entre a placa e o receptor do sistema RF, o circuito necessitava apenas alimentar o receptor e uma linha de comunicação para cada canal.

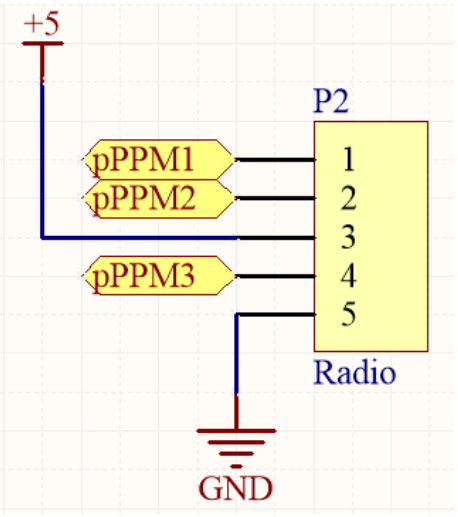

Figura 47 Esquemático de ligação com o receptor RF

Por fim, restaram as conexões de alimentação do mbed (5V), alimentação dos sensores industriais (10V) e também foi adicionado um conector para ligar um switch que poderia vir a ser útil em algum momento. Os 5 Volts foram fornecidos por um regulador de tensão chaveado que também era alimentado pelas baterias do robô.

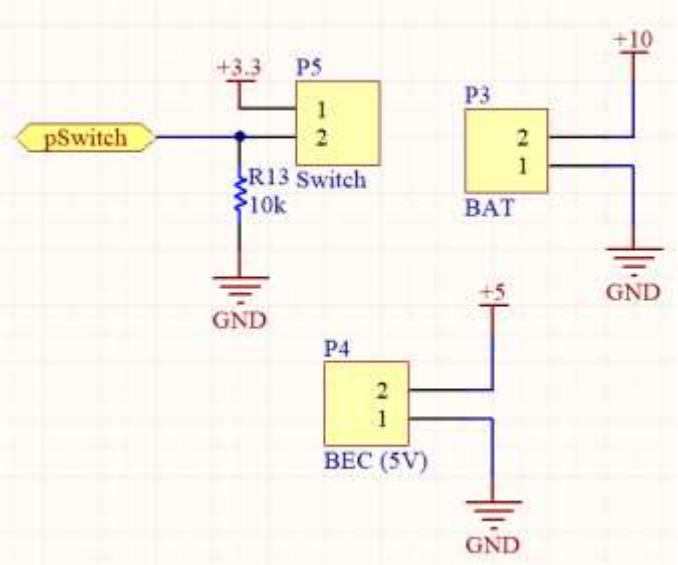

Figura 48 Esquemático das alimentações

Após o desenvolvimento dos esquemáticos, foi necessário definir a posição dos componentes e os caminhos que as trilhas de cobre percorreriam nas duas camadas da placa (Figura 44).

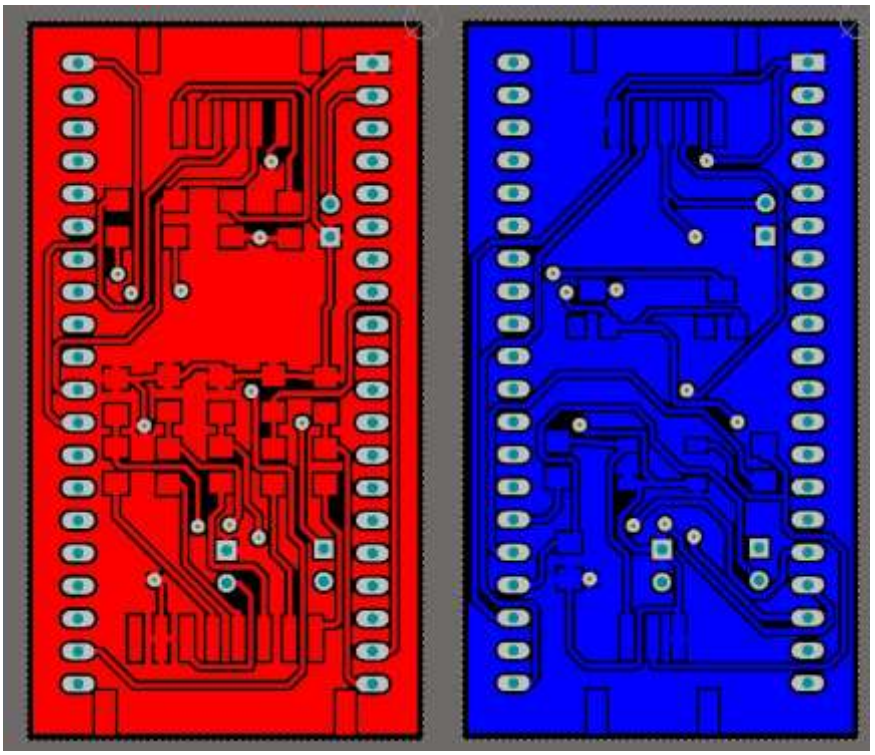

Figura 49 As duas camadas de cobre da placa de interface

Com tudo definido, o projeto da placa foi enviado para ser produzido em uma empresa especializada e o resultado obtido foi o seguinte:

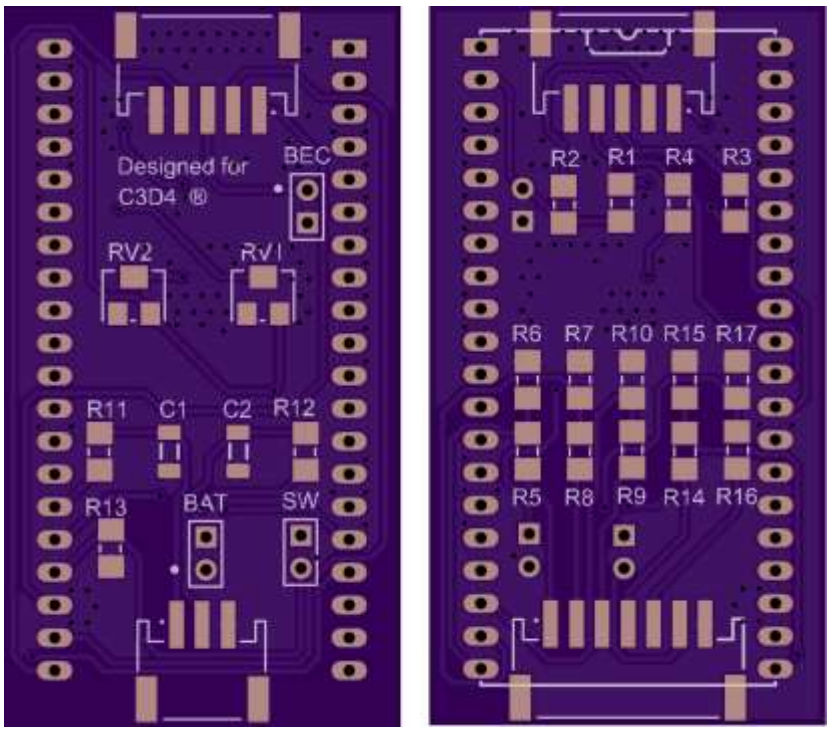

Figura 50 Vista superior e inferior da PCB da placa de interface

Com as PCBs em mãos, foi necessário soldar todos os resistores, capacitores e conectores para que a placa ficasse pronta (Figura 46).

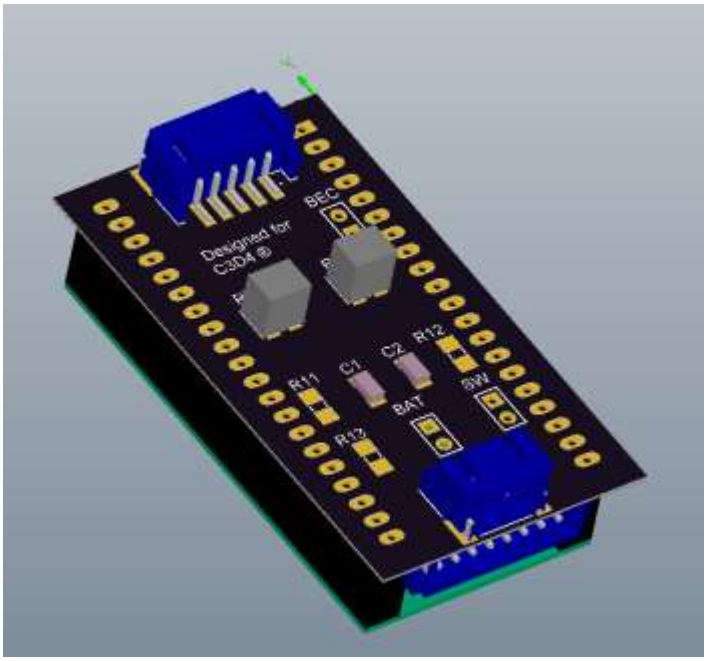

Figura 51 Placa de interface pronta

3.3 Software

3.3.1 Plataforma de desenvolvimento

Para programar o mbed de forma gratuita, é necessário se conectar ao portal http://www.mbed.org e utilizar o compilador online. Outra opção, porém proprietária, é a utilização da IDE uVision da Keil desenvolvida especificamente para este microcontrolador. Esta segunda opção é altamente recomendada pois não é necessário ter uma conexão com a internet para programar e apenas ela possui o recurso de debugging em tempo de execução, ferramenta extremamente útil.

A seguir, serão apresentadas as funções básicas desenvolvidas para o funcionamento do robô.

3.3.2 Definição das variáveis dos sensores

Para definir que pinos do Microcontrolador correspondiam aos sensores de detecção de adversário foram adicionadas as seguintes linhas de código:

```
//Pinos dos sensores de distancia
#define pd1 p20
#define pd2 p19
#define pd3 p18
#define pd4 p17
#define pd5 p16
//Entradas dos sensores de distancia
DigitalIn sd1(pd1);
DigitalIn sd2(pd2);
DigitalIn sd3(pd3);
DigitalIn sd4(pd4);
DigitalIn sd5(pd5);
```

Já para definir os pinos correspondentes aos sensores de linha:

```
//Pinos dos sensores de linha
#define plFL p29
#define plFR p30
//Entradas dos sensores de linha
DigitalIn slFL(plFL);
DigitalIn slFR(plFR);
```

3.3.3 Funções para leitura dos comandos do operador

Para criar a função de leitura dos comandos do operador foi criada uma biblioteca de leitura do protocolo PWM utilizado em receptores de sistemas de controle por RF. A biblioteca na íntegra se encontra no apêndice 1 junto com todos os arquivos de código. Para ler o valor atual de cada canal do rádio foi criada a função ReadRadio que possui como entrada o número do canal que se deseja ler o estado.

```
//Entrada de radio
RadioIn radio;

float ReadRadio(int chan){  // Retorna o valor do canal entre -1 e 1
radio.Update();
if (chan==1){return radio.chan1;}
if (chan==2){return radio.chan2;}
if (chan==3){return radio.chan3;}
else return 0;
}
```

3.3.4 Funções para comandar os motores

Para comandar os motores, foi criada a função Drive que possuía como entradas dois valores entre -1 e 1, correspondentes à potência que deveria ser enviada a cada motor.

```
//Pinos das saidas analogicas para SaberTooth
#define psaber1 p26
#define psaber2 p25
//Saidas analogicas para SaberTooth
PwmOut saber1(psaber1);
PwmOut saber2(psaber2);

//Saber2=> Motor esquerda Saber1=> Motor Direita
void Drive(float m_esq, float m_dir){ // Valores de -1 a 1 em cada motor
float gain=1;
float trime=0;
float trimd=0;
saber2.write(((-m_esq*gain/2)+0.5+trime));
saber1.write(((-m_dir*gain/2)+0.5+trimd));
return;
}
```

3.3.5 Função para evitar o sair da arena

Evitar o suicídio (sair voluntariamente com o robô da arena), é uma das tarefas de maior importância, para isso, foi desenvolvida a função LineCheck.

```
int LineCheck(){

   result=!slFL*10+!slFR;
   led1=!slFR;
   led4=!slFL;

   if (result==0 && anterior==0){return 0;}
   if (result==0 && anterior==1){anterior=0; wait(0); return 0;}
   switch (result){

   case 10:{ // Somente FR
          Drive(-1,-1);
          wait(0.15);
          Drive(-1,1);
          wait(0.15);
          anterior=1;
          break;
   }
   case 1:{ // Somente FL
          Drive(-1,-1);
          wait(0.15);
          Drive(1,-1);
          wait(0.15);
          anterior=1;
          break;
   }
   case 11:{ // Os dois da frente
          Drive(-1,-1);
          wait(0.15);
          Drive(-1,1);
          wait(0.15); //Mesmo valor do wait final
          anterior=1;
          break;
   }
   default: {anterior=1; break;}
   }
   return 1;
}
```

Assim que um dos sensores detecta a linha branca, o C3D4 interrompe a perseguição ou a trajetória em que se encontrava para efetuar uma manobra defensiva até que o mesmo esteja fora de perigo.

3.3.6 Rotina de combate baseada em lógica Crisp

Como os sensores utilizados para a competição possuem apenas saídas 0 ou 1, não faz sentido desenvolver a rotina de controle baseada em Lógica Fuzzy. Com isto, a função Chase foi desenvolvida para perseguir o oponente.

```
int Chase(){
   int detectados=0;
   detectados=sd1+sd2+sd3+sd4+sd5;
   if (detectados==0){return 0;}
   while(detectados)
   {
    //LineCheck();
                    if (ReadRadio(3)<-0.5){return 0;}
    switch(detectados){
    case 1:{
        if(sd5){Drive(-1,1); break;}
        else if(sd4){Drive(0.25,1); break;}
        else if(sd3){Drive(1,1); break;}
        else if(sd2){Drive(1,0.25); break;}
        else if(sd1){Drive(1,-1); break;};
        break;
    }
    case 2:{
        if(sd5&&sd2){Drive(-0.7,-0.3); break;}
        else if(sd4&&sd3){Drive(1,0.75); break;}
        else if(sd3&&sd2){Drive(0.75,1); break;}
        else if(sd1&&sd4){Drive(-0.3,-0.7); break;}
        break;
    }
    case 3:{
         if(sd2&&sd3&&sd4){Drive(1,1); break;}
         else if(sd1&&sd4&&sd3){Drive(-0.3,-0.7); break;}
         else if(sd3&&sd2&&sd5){Drive(-0.7,-0.3); break;}
        break;
        }
    default: {break;}
    }
    detectados=sd1+sd2+sd3+sd4+sd5;
   }

   return 1;
}
```

Inicialmente a função verifica se algum dos sensores detectou o adversário, caso negativo, a função já termina, caso seja detectado, a função controla os motores de modo a perseguir o adversário à menos que a linha seja detectada ou que o canal 3 do rádio seja desativado por desejo do operador.

3.3.7 Rotina de combate baseada em lógica Fuzzy

Como os sensores reais do robô não seriam capazes de dar informações suficientes para desenvolver o controle baseado em Lógica Fuzzy, o modelo Fuzzy será proposto apenas no campo teórico.

Foram definidas duas variáveis de entrada, sendo elas a distância em relação ao oponente e o ângulo relativo entre o robô e o oponente detectado e duas variáveis de saída, motor esquerdo e motor direito, de acordo com o diagrama da Figura 47.

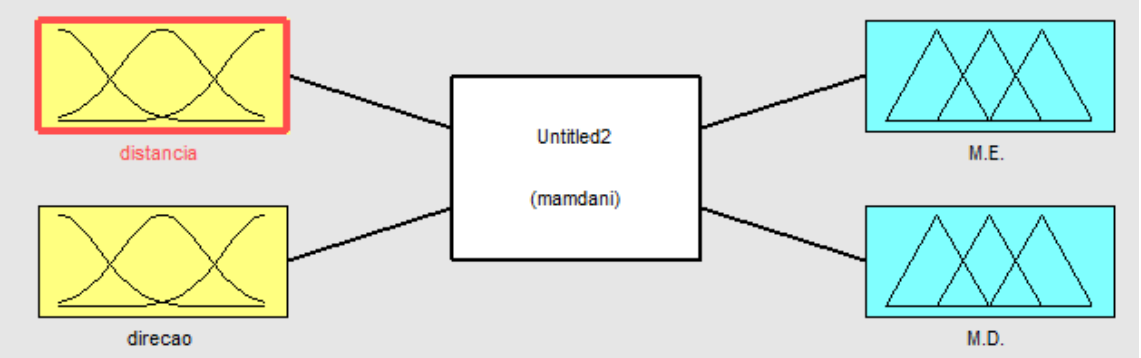

Figura 52 Diagrama Fuzzy

Os parâmetros de Fuzzificação e Defuzzificação escolhidos foram os seguintes:

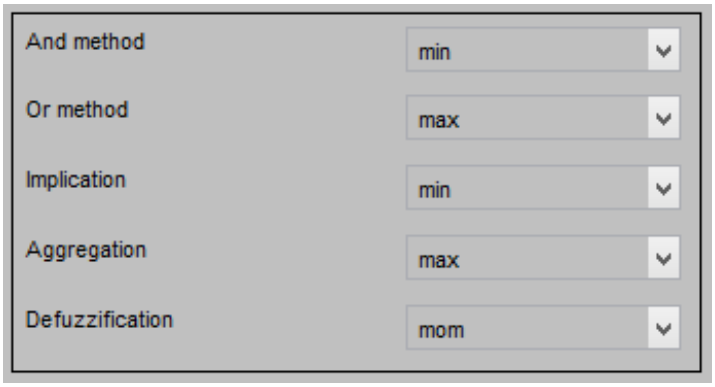

Figura 53 Parâmetros Fuzzy

Para a variável distância, foram definidos 4 conjuntos Fuzzy: Perto, Médio, Longe e Fora (do alcance). O valor desta entrada é definido pela média normalizada das distâncias medidas pelos sensores, descartando os valores daqueles que não conseguirem enxergar o oponente.

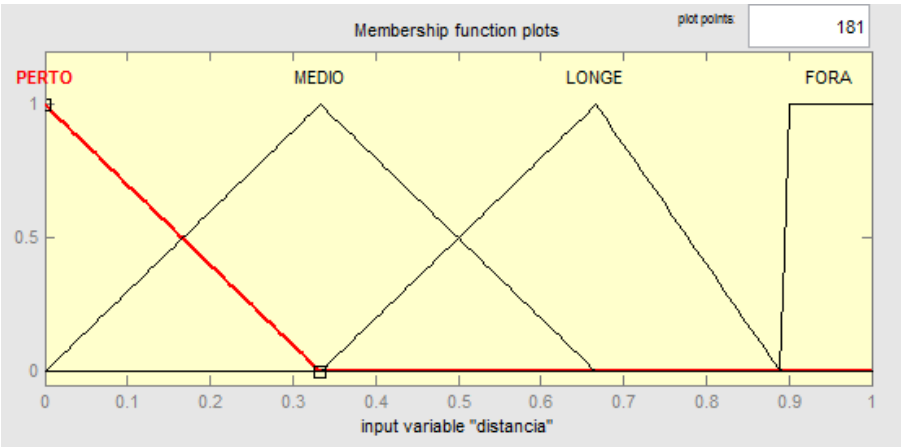

Figura 54 Funções de Pertinência da distância

Para a variável direção, foram definidos 4 conjuntos Fuzzy: Esquerda Alto, Esquerda Médio, Zero, Direita Médio e Direita Alto. O valor desta variável é obtido pela média normalizada dos ângulos correspondentes de cada sensor que tenha detectado o oponente.

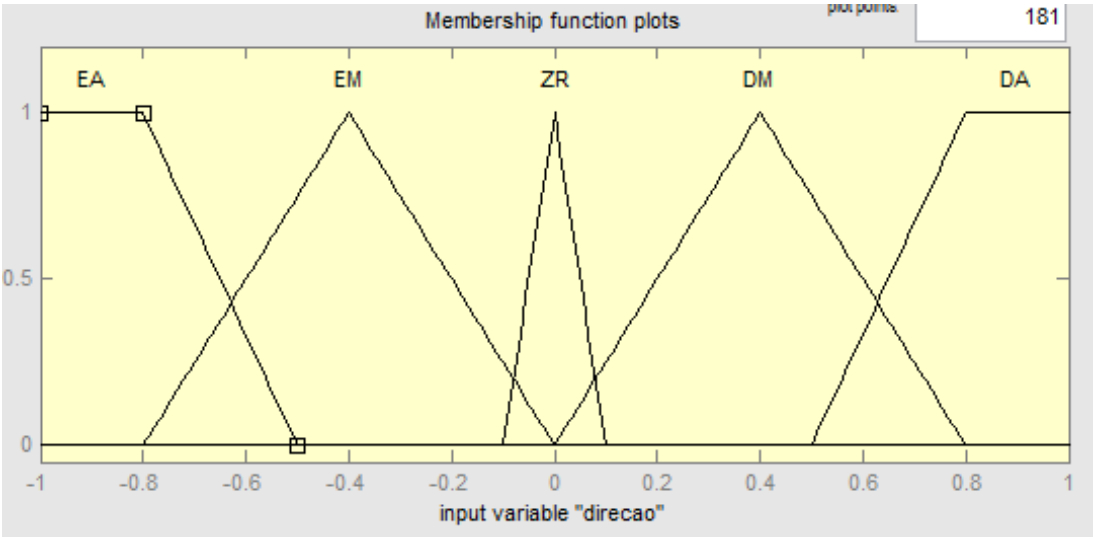

Figura 55 Funções de Pertinência da direção

Para as variáveis de saída dos motores, foram definidos 5 conjuntos Fuzzy: Negativo Alto, Negativo Médio, Zero, Positivo Médio e Positivo Alto.

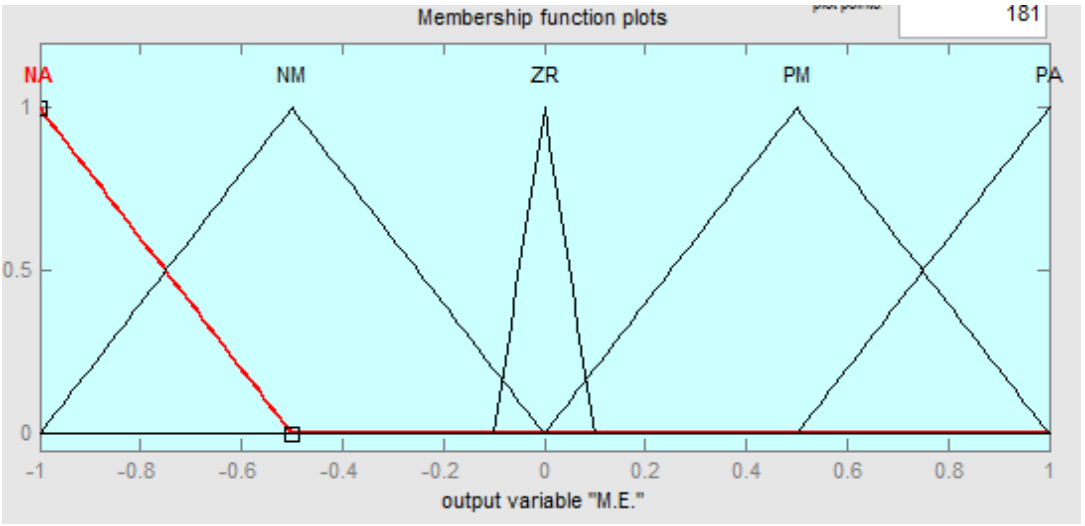

Figura 56 Funções de Pertinência dos motores

Baseado nas variáveis de entrada e saída do sistema, foram definidas as regras da seguinte maneira:

Distância	Direção	M.E.	M.D.
P	EA	NA	PA
M	EA	NA	ZR
L	EA	ZR	PM
P	EM	NA	ZR
M	EM	ZR	PM
L	EM	PM	PA
P	ZR	PA	PA
M	ZR	PM	PM
L	ZR	PM	PM
P	DM	ZR	NA
M	DM	PM	ZR
L	DM	PA	PM
P	DA	PA	NA
M	DA	ZR	NA
L	DA	PM	ZR
F	X	NM	PM

Legenda	
P	Perto
M	Médio
L	Longe
EA	Esquerda Alto
EM	Esquerda Médio
ZR	Zero
DM	Direita Médio
DA	Direita Alto
NA	Negativo Alto
NM	Negativo Médio
PM	Positivo Médio
PA	Positivo Alto

4 Resultados Obtidos

Após a conclusão do projeto, foram levados dois exemplares do robô para o Japão para competir na 24th All Japan International Sumo Tournament e foram conquistados o segundo e terceiro lugar na competição (Figura 52), um feito inédito para qualquer estrangeiro. Por se tratar de um robô desenvolvido para competir, os resultados obtidos em batalha são a maior prova de que o projeto foi um sucesso.

Infelizmente a falta de sensores de distância rápidos o suficiente para serem usados em uma luta de verdade não permitiu a validação do modelo teórico de controle por Lógica Fuzzy.

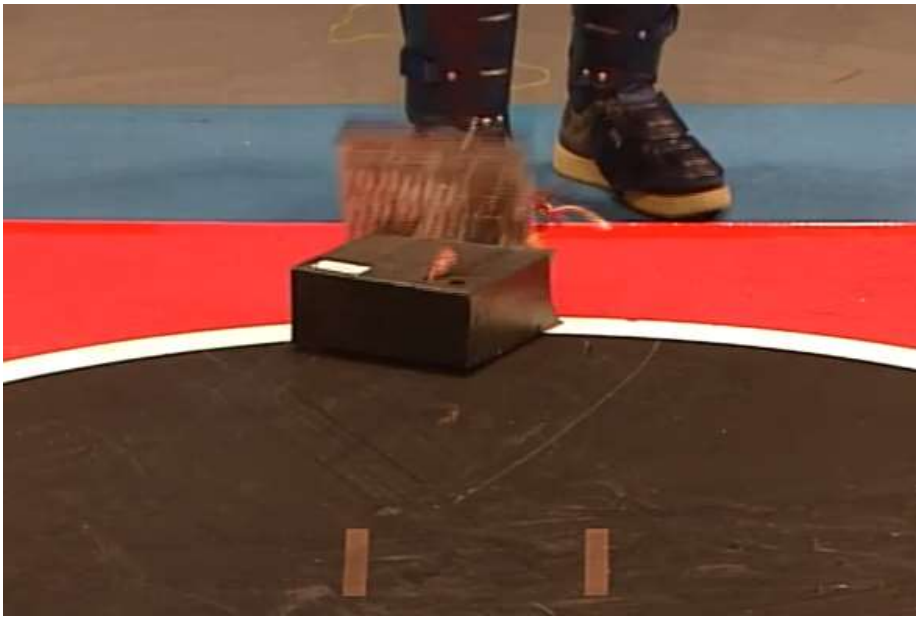

Figura 57 C3D4 vencendo uma das lutas no Japão

Figura 58 Prêmios Obtidos

5 Conclusões

- O chassi se provou extremamente robusto mecanicamente suportando a grande força feita pelos imãs (aproximadamente 75Kg) sem defletir.
- A lâmina se mostrou extremamente eficiente, sendo mais efetiva que as lâminas especiais produzidas no Japão por fabricantes de lâminas de barbear.
- Apesar de precisar de alguns ajustes em relação à durabilidade, as rodas de Poliuretano moldado a frio tiveram um bom desempenho.
- O sistema de redução desempenhou seu papel com perfeição sendo necessário apenas pensar em uma maneira de isolar as engrenagens do resto do robô para evitar esmagamento de fios.
- O suporte dos sensores impresso em 3D foi frágil pois não suportou o impacto durante a final da competição e quebrou. É necessário repensar o material que deverá ser utilizado em uma nova versão.
- A placa de interface desempenhou perfeitamente o seu papel e se mostrou mais robusta que a versão anterior que possuía conectores mais delicados que não resistiam aos impactos durante as lutas.

Apendice 1 – main.cpp – Programa Principal

```
#include "Servo.h"
#include "mbed.h"
#include "RadioIn.h"
#include "Config.h"
int result=0;

//Saber2=> Motor esquerda Saber1=> Motor Direita
void Drive(float m_esq, float m_dir){ // Valores de -1 a 1 em cada motor
float gain=1;
float trime=0;
float trimd=0;
saber2.write(((-m_esq*gain/2)+0.5+trime));
saber1.write(((-m_dir*gain/2)+0.5+trimd));
return;
}

void Monitor(){
int a=sd1;
int b=sd2;
int c=sd3;
int d=sd4;
int e=sd5;
int f=!slFR;
int i=!slFL;
int linha=!slFL*10+!slFR;
float j=-(saber1.read()-0.5)*2;
float k=-(saber2.read()-0.5)*2;
float l=radio.chan3;
int sw=sSwitch;

pc.printf("s1=%d|s2=%d|s3=%d|s4=%d|s5=%d",a,b,c,d,e);
pc.printf("Linha=%d|",linha);
pc.printf("|FR=%d|FL=%d|",f,i);
pc.printf("ME=%.3f|MD=%.3f|",k,j);
pc.printf("Ch3=%.1f|",l);
pc.printf("S=%d\r",sw);
return;
}

void DriveRC(){
        radio.Update();
        Drive(radio.chan1,radio.chan2);
        return;
}


float ReadRadio(int chan){  // Retorna o valor do canal entre -1 e 1
radio.Update();
if (chan==1){return radio.chan1;}
if (chan==2){return radio.chan2;}
if (chan==3){return radio.chan3;}
else return 0;
}
```

```
// Faz o Mix para pilotar com rádio pistola
void DriveMixedRC(){
        radio.Update();
        saber1.write((((radio.chan1+radio.chan2)+1.0)/2.0));
        saber2.write((((radio.chan1-radio.chan2)+1.0)/2.0));
        return;
}

int LineCheck(){

   result=!slFL*10+!slFR;
   led1=!slFR;
   led4=!slFL;

   if (result==0 && anterior==0){return 0;}
   if (result==0 && anterior==1){anterior=0; wait(0); return 0;}
   switch (result){

   case 10:{ // Somente FR
        Drive(-1,-1);
        wait(0.15);
        Drive(-1,1);
        wait(0.15);
        anterior=1;
        break;
   }
   case 1:{ // Somente FL
        Drive(-1,-1);
        wait(0.15);
        Drive(1,-1);
        wait(0.15);
        anterior=1;
        break;
   }
   case 11:{ // Os dois da frente
        Drive(-1,-1);
        wait(0.15);
        Drive(-1,1);
        wait(0.15); //Mesmo valor do wait final
        anterior=1;
        break;
   }
   default: {anterior=1; break;}
   }
   return 1;
}

int Chase(){
   int detectados=0;
   detectados=sd1+sd2+sd3+sd4+sd5;
   if (detectados==0){return 0;}

   while(detectados)
   {
    //LineCheck();
                if (ReadRadio(3)<-0.5){return 0;}
    switch(detectados){

    case 1:{
        if(sd5){Drive(-1,1); break;}
        else if(sd4){Drive(0.25,1); break;}
        else if(sd3){Drive(1,1); break;}
        else if(sd2){Drive(1,0.25); break;}
```

```
        else if(sd1){Drive(1,-1); break;};
        break;
    }
    case 2:{
        if(sd5&&sd2){Drive(-0.7,-0.3); break;}
        else if(sd4&&sd3){Drive(1,0.75); break;}
        else if(sd3&&sd2){Drive(0.75,1); break;}
        else if(sd1&&sd4){Drive(-0.3,-0.7); break;}
        break;
    }
    case 3:{
                                        if(sd2&&sd3&&sd4){Drive(1,1); break;}
        else if(sd1&&sd4&&sd3){Drive(-0.3,-0.7); break;}
        else if(sd3&&sd2&&sd5){Drive(-0.7,-0.3); break;}
        break;
        }
    default: {break;}
    }
    detectados=sd1+sd2+sd3+sd4+sd5;
   }

   return 1;
}

void Search(){

    Drive(0.8,0.8);  //Anda reto até bater na linha
   return;
}

int main()
{
saber1.period_us(5);
saber2.period_us(5);
radio.Init();

while(1){
      while(ReadRadio(3)>-0.5)
      { // Modo RC Assistido
           Monitor();
           Chase();
                                                    DriveMixedRC();
      }
                                  // Modo RC Não assistido
          DriveMixedRC();
                                Monitor();
      }

}
```

Apendice 2 – config.h – Header do programa principal

```
//Pino de Entrada do Switch
#define pswitch p24

//Pinos das saidas analogicas para SaberTooth
#define psaber1 p26
#define psaber2 p25

//Pinos dos sensores de linha
#define plFL p29
#define plFR p30

//Pinos dos sensores de distancia
#define pd1 p20
#define pd2 p19
#define pd3 p18
#define pd4 p17
#define pd5 p16

//Comunicacao serial
Serial pc(USBTX,USBRX);

//Saida dos LEDs
DigitalOut led1(LED1);
DigitalOut led2(LED2);
DigitalOut led3(LED3);
DigitalOut led4(LED4);

//Entrada do Switch
DigitalIn sSwitch(pswitch);

//Entradas dos sensores de linha
DigitalIn slFL(plFL);
DigitalIn slFR(plFR);

//Entradas dos sensores de distancia
DigitalIn sd1(pd1);
DigitalIn sd2(pd2);
DigitalIn sd3(pd3);
DigitalIn sd4(pd4);
DigitalIn sd5(pd5);

//Saidas analogicas para SaberTooth
PwmOut saber1(psaber1);
PwmOut saber2(psaber2);

//Entrada de radio
RadioIn radio;

//Variavel Global da ultima leitura do LineCheck()
int anterior=0;
```

Apendice 3 – RadioIn.cpp – Biblioteca do R/C

```
#include "mbed.h"
#include "RadioIn.h"


//Initialization
RadioIn::RadioIn() :
   ChInt0(InterruptIn(p21)),
   ChInt1(InterruptIn(p22)),
   ChInt2(InterruptIn(p23))
{}

void RadioIn::Init()
{
   Time.start();

   for(int i= 0; i < 2; i++)
   {
      LastRise[i]= 0;
      dTime[i]= 1000;
   }

   ChInt0.mode(PullDown);   ChInt0.rise<RadioIn>(this, &RadioIn::Rise0);
ChInt0.fall<RadioIn>(this, &RadioIn::Fall0);
   ChInt1.mode(PullDown);   ChInt1.rise<RadioIn>(this, &RadioIn::Rise1);
ChInt1.fall<RadioIn>(this, &RadioIn::Fall1);
   ChInt2.mode(PullDown);   ChInt2.rise<RadioIn>(this, &RadioIn::Rise2);
ChInt2.fall<RadioIn>(this, &RadioIn::Fall2);
   Update();
}

//Update Method
void RadioIn::Update()
{
   for(int i= 0; i<2; i++)
      RawChannels[i]= dTime[i];

   //time to float conversion
   chan1= float(dTime[0]-1500) * 0.002;
   if (chan1>1){chan1=1;} if (chan1<-1){chan1=-1;}
   chan2= float(dTime[1]-1500) * 0.002;
   if (chan2>1){chan2=1;} if (chan2<-1){chan2=-1;}
   chan3= float(dTime[2]-1500) * 0.002;
   if (chan3>1){chan3=1;} if (chan3<-1){chan3=-1;}
 }
```

Apendice 4 – RadioIn.h – Header da biblioteca RadioIn

```
#pragma once

//Interrupt-Callback Function
#define RadioIn_RISE_FALL(Ch)\
void Rise##Ch()\
{\
   LastRise[Ch]= Time.read_us();\
}\
void Fall##Ch()\
{\
   int dT= Time.read_us() - LastRise[Ch];\
   if(dT > 900 && dT < 2100)\
       dTime[Ch]= dT;\
}


class RadioIn
{
private:
   InterruptIn ChInt0;          //Interrupt-Handler
   InterruptIn ChInt1;
   InterruptIn ChInt2;

   Timer Time;                  //Timer for all 3 Channels
   volatile int LastRise[3];      //Time of the last rise
   volatile int dTime[3];         //Pulse in us [1000...2000]

public:
   int RawChannels[3]; //Raw Data [1000...2000]

   //Channels (0.0 - 1.0)
   float chan1;
   float chan2;
   float chan3;

   //Initialization
   RadioIn();
   void Init();

   //Interrupt-Callbacks
   RadioIn_RISE_FALL(0);
   RadioIn_RISE_FALL(1);
   RadioIn_RISE_FALL(2);

   //Update Method
   void Update();
};
```

Apêndice 5 – Servo.cpp – Biblioteca para enviar PWM

```
#include "Servo.h"
#include "mbed.h"

static float clamp(float value, float min, float max) {
    if(value < min) {
        return min;
    } else if(value > max) {
        return max;
    } else {
        return value;
    }
}

Servo::Servo(PinName pin) : _pwm(pin) {
    calibrate();
    write(0.5);
}

void Servo::write(float percent) {
    float offset = _range * 2.0 * (percent - 0.5);
    _pwm.pulsewidth(0.0015 + clamp(offset, -_range, _range));
    _p = clamp(percent, 0.0, 1.0);
}

void Servo::position(float degrees) {
    float offset = _range * (degrees / _degrees);
    _pwm.pulsewidth(0.0015 + clamp(offset, -_range, _range));
}

void Servo::calibrate(float range, float degrees) {
    _range = range;
    _degrees = degrees;
}

float Servo::read() {
    return _p;
}

Servo& Servo::operator= (float percent) {
    write(percent);
    return *this;
}

Servo& Servo::operator= (Servo& rhs) {
    write(rhs.read());
    return *this;
}

Servo::operator float() {
    return read();
}
```

Apendice 6 – Servo.h – Header da biblioteca Servo

```
#ifndef MBED_SERVO_H
#define MBED_SERVO_H

#include "mbed.h"

/** Servo control class, based on a PwmOut
 *
 * Example:
 * @code
 * // Continuously sweep the servo through it's full range
 * #include "mbed.h"
 * #include "Servo.h"
 *
 * Servo myservo(p21);
 *
 * int main() {
 *     while(1) {
 *         for(int i=0; i<100; i++) {
 *             myservo = i/100.0;
 *             wait(0.01);
 *         }
 *         for(int i=100; i>0; i--) {
 *             myservo = i/100.0;
 *             wait(0.01);
 *         }
 *     }
 * }
 * @endcode
 */
class Servo {

public:
    /** Create a servo object connected to the specified PwmOut pin
     *
     * @param pin PwmOut pin to connect to
     */
    Servo(PinName pin);

    /** Set the servo position, normalised to it's full range
     *
     * @param percent A normalised number 0.0-1.0 to represent the full range.
     */
    void write(float percent);

    /**  Read the servo motors current position
     *
     * @param returns A normalised number 0.0-1.0  representing the full range.
     */
    float read();

    /** Set the servo position
     *
     * @param degrees Servo position in degrees
     */
    void position(float degrees);

    /**  Allows calibration of the range and angles for a particular servo
     *
     * @param range Pulsewidth range from center (1.5ms) to maximum/minimum position in seconds
     * @param degrees Angle from centre to maximum/minimum position in degrees
```

```
     */
    void calibrate(float range = 0.0005, float degrees = 45.0);

    /**  Shorthand for the write and read functions */
    Servo& operator= (float percent);
    Servo& operator= (Servo& rhs);
    operator float();

protected:
    PwmOut _pwm;
    float _range;
    float _degrees;
    float _p;
};

#endif
```

Apêndice 7 – Fuzzy.fis – Código MATLAB para o modelo Fuzzy proposto

```
[System]
Name='Untitled'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=2
NumRules=16
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='mom'

[Input1]
Name='distancia'
Range=[0 1]
NumMFs=4
MF1='PERTO':'trimf',[-0.3333 0 0.3333]
MF2='MEDIO':'trimf',[0 0.3333 0.6667]
MF3='LONGE':'trimf',[0.3333 0.6667 0.89]
MF4='FORA':'trapmf',[0.89 0.9 1 1.01]

[Input2]
Name='direcao'
Range=[-1 1]
NumMFs=5
MF1='EA':'trapmf',[-1.1 -1 -0.8 -0.5]
MF2='EM':'trimf',[-0.8 -0.4 0]
MF3='ZR':'trimf',[-0.1 0 0.1]
MF4='DM':'trimf',[0 0.4 0.8]
MF5='DA':'trapmf',[0.5 0.8 1 1.1]

[Output1]
Name='M.E.'
Range=[-1 1]
NumMFs=5
MF1='NA':'trimf',[-1.5 -1 -0.5]
MF2='NM':'trimf',[-1 -0.5 0]
MF3='ZR':'trimf',[-0.1 0 0.1]
MF4='PM':'trimf',[0 0.5 1]
MF5='PA':'trimf',[0.5 1 1.5]

[Output2]
Name='M.D.'
Range=[-1 1]
NumMFs=5
MF1='NA':'trimf',[-1.5 -1 -0.5]
MF2='NM':'trimf',[-1 -0.5 0]
MF3='ZR':'trimf',[-0.1 0 0.1]
MF4='PM':'trimf',[0 0.5 1]
MF5='PA':'trimf',[0.5 1 1.5]

[Rules]
1 1, 1 5 (1) : 1
2 1, 1 3 (1) : 1
3 1, 3 4 (1) : 1
1 2, 1 3 (1) : 1
2 2, 3 4 (1) : 1
```

3 2, 4 5 (1) : 1
1 3, 5 5 (1) : 1
2 3, 4 4 (1) : 1
3 3, 4 4 (1) : 1
1 4, 3 1 (1) : 1
2 4, 4 3 (1) : 1
3 4, 5 4 (1) : 1
1 5, 5 1 (1) : 1
2 5, 3 1 (1) : 1
3 5, 4 3 (1) : 1
4 0, 2 4 (1) : 1

Bibliografia

[1] [Online]. Available: http://www.fsi.co.jp/sumo/index.html.

[2] A. O. G. Barbosa, "Controle de Robô Usando Técnicas Inteligentes," 2008.

[3] F. F. Dede. [Online]. Available: http://jsumo.com.

[4] M. A. Meggiolaro, Riobotz Combat Robot Tutorial, CreateSpace, 2009.

[5] J. E. Carryer, Introduction to Mechatronic Design, Prentice Hall, 2010.

[6] [Online]. Available: http://mbed.org.

[7] E. Oberg, Machinery's Handbook, 29th, Industrial Press, 2012.