PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Sistema de Posicionamento Robótico Guiado por Visão Computacional

Michel Feinstein

PROJETO FINAL DE GRADUAÇÃO

Centro Técnico Científico – CTC

Departamento de Informática

Curso de Graduação em Engenharia da Computação

Rio de Janeiro, Dezembro de 2013

PUC
RIO

Michel Feinstein

Sistema de Posicionamento Robótico Guiado por Visão Computacional

Relatório de Projeto Final, apresentado ao programa de **Engenharia de Computação** da PUC-Rio como requisito parcial para a obtenção do titulo de Engenheiro de Computação.

Orientador: Alberto Barbosa Raposo

Co-Orientador: Marco Antonio Meggiolaro

Rio de Janeiro,

Dezembro de 2013

“A new idea comes suddenly and in a rather intuitive way. But intuition is nothing but the outcome of earlier intellectual experience.”

Albert Einstein, 3 de Maio de 1949

Agradecimentos

Por se tratar de um projeto complexo, jamais teria alcançado qualquer resultado prático sem o apoio e a ajuda de professores, amigos e familiares.

Agradeço a todos que me influenciaram e ajudaram no andamento desse projeto, em especial aos professores Alberto Barbosa Raposo, Marco Antonio Meggiolaro, Manuel Eduardo Loaiza Fernandez, Luiz Antonio Pereira Gusmão e Marbey Manhães Mosso por terem me recebido e orientado, aos amigos Luiz Fernando Santarelli Conrado Nobre, Eduardo Von Ristow, João Carlos Virgolino Soares, Aline Wilm Senna Pinto e Igor Lins e Silva pelas sugestões, comentários e apoio, por último, porém não menos importante, agradeço aos meus familiares pelo apoio incondicional, sempre me estimulando e apoiando a alcançar os meus objetivos.

Resumo

Feinstein, Michel. Raposo, Alberto Barbosa. Meggiolaro, Marco Antonio. Sistema de Posicionamento Robótico Guiado por Visão Computacional. Rio de Janeiro, 2013. 44p. Relatório de Projeto Final II – Centro Técnico Científico – CTC, Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Este trabalho tem como objetivo apresentar o desenvolvimento de uma plataforma robótica de posicionamento angular com dois graus de liberdade que direciona um par estéreo de câmeras a serem aplicadas em um algoritmo autônomo de identificação de círculos brancos em um fundo preto. O sistema utiliza comunicação USB, com protocolo HID, para o controle do posicionamento e aquisição de dados.

Palavras-chave

Robótica, Controlador PID, Visão Computacional, Sistemas Embarcados, USB HID

Abstract

Feinstein, Michel. Raposo, Alberto Barbosa. Meggiolaro, Marco Antonio. Computer Vision Guided Robotic Positioning System. Rio de Janeiro, 2013. 44p. Final Year Project Report II – Centro Técnico Científico – CTC, Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

This work aims to present the development of an angular positioning robotic platform with two degrees of freedom that drives a stereo pair of cameras to be applied on an autonomous algorithm to identify white circles on a black background. The system uses the USB communication HID protocol, for the positioning control and data acquisition.

Keywords

Robotics, PID Controller, Computer Vision, Embedded Systems, USB HID

Sumário

1. Introdução

A interação entre homens e máquinas vem se tornando cada dia mais comum e natural, porém ainda está longe do ideal. Apesar de computadores pessoais e smartphones estarem conquistando as massas, os robôs de uso pessoal ainda são raros.

A robótica visa unificar três grandes áreas da engenharia: Mecânica, Eletrônica e Computação. Com uma engenharia mecânica bastante amadurecida e sistemas eletrônicos bastante potentes, o grande desafio ultimamente está na área de computação. Algoritmos inteligentes para detecção e processamento de informações relativas ao ambiente em que o sistema se encontra e um processo autônomo e confiável de tomada de decisão possuem em geral uma complexidade alta e exigem conhecimentos prévios de diversas áreas do ensino de engenharia, o que acaba não sendo motivado em muitos cursos de graduação.

Para estimular estudantes e engenheiros a desenvolver novas tecnologias e buscar novas áreas do conhecimento, diversas instituições internacionais promovem competições de engenharia, onde os competidores podem compartilhar as suas ideias e projetos e assim evoluir os seus conhecimentos.

Uma das competições de robótica mais desafiadoras é a Robot Shooting Gallery. Criada originalmente na DEFCON [1], e posteriormente movida para a atualmente extinta RoboGames [2], nela o sistema robótico deve identificar círculos brancos em um fundo preto, dispostos a três metros de distância do robô. Dois robôs competem ao mesmo tempo, tendo cada um a sua área de atuação com seus alvos, um robô não pode interferir na área de atuação do outro. No menor tempo possível dentro de dois minutos, deve-se acertar o máximo de círculos brancos, para cada círculo branco acertado será creditado um ponto. Círculos pretos também serão espalhados pela plataforma e a cada círculo preto atingido serão descontados 5 pontos. Ganha o competidor que acumular mais pontos e em caso de empate vence quem realizar a prova em menos tempo.

A proposta deste projeto é criar um sistema robótico que será capaz de identificar por visão computacional todos os círculos brancos em uma determinada área de atuação, posicionando, por meio de um sistema eletrônico, um sistema mecânico de propulsão em um ângulo correto para que assim consiga impelir uma esfera de plástico em cada círculo e derrubá-lo.

Para a realização desse projeto é necessário uma base forte em

Programação e Eletrônica, aonde ambas as disciplinas são o foco central do curso de Engenharia de Computação.

O projeto foi intitulado de “S.A.R.A.”, significando “Sistema Autônomo de Reconhecimento de Alvos”.

2. Marco Teórico

O projeto tem como base duas etapas, a identificação dos círculos e o posicionamento da plataforma robótica. Para a identificação dos círculos foram estudadas duas técnicas, a Transformada de Hough, que será apresentada em seguida, e a identificação de contornos com o posterior encaixe em formas elípticas, que será apresentada na seção dedicada à execução do projeto por se tratar mais de uma implementação especifica da plataforma do que uma teoria em si.

Já o posicionamento dos eixos angulares do robô utiliza o algoritmo do Controlador PID, uma técnica que apresenta facilidade de configuração e robustez.

2.1 Identificação de círculos

Diversas técnicas podem ser empregadas para a detecção de círculos em uma imagem, porém como o objetivo da plataforma desenvolvida é a detecção de círculos brancos em uma imagem com fundo preto, podem-se dispensar técnicas mais complexas e computacionalmente caras como redes neurais.

A Transformada de Hough é um método para a identificação de formas simples em uma imagem [3]. Apesar de sua forma básica aplicada à visão computacional ter sido feita para a detecção de linhas (originalmente o algoritmo foi criado para experimentos físicos [4]), o algoritmo pode ser estendido a detectar outras formas, sendo o método mais comum para a identificação de círculos em uma imagem [5].

Para a identificação de formas a Transformada de Hough necessita de uma imagem binária, ou seja, uma imagem que só possua dois tipos de valores, geralmente preto e branco. Para a obtenção de uma imagem binária uma série de filtros podem ser aplicados à imagem original de forma a realçar os formatos a serem detectados e possivelmente excluir dados irrelevantes que possam influenciar negativamente a procura por padrões.

Para a análise do reconhecimento de formas pela Transformada de Hough vamos primeiramente nos ater ao caso restrito de identificação de linhas. Vale lembrar que uma linha pode ser definida por sua inclinação e interseção, porém

para efeitos computacionais, a forma polar da equação da reta é mais desejada:

$$\rho = x\cos\theta + y\,\mathrm{sen}\,\theta$$

Equação 1: Forma Polar da equação de uma reta

Se considerarmos a imagem com as linhas a serem detectadas como um plano (x, y) e que existem uma infinidade de possíveis retas nesse plano (x, y), todas essas retas podem ser representadas como pontos em um plano (ρ, θ) onde cada ponto no plano (ρ, θ) corresponde a uma reta no plano (x, y). Dessa forma, para cada ponto (x, y) na imagem, obtemos uma senóide no plano (ρ, θ) que representará todas as possíveis retas que podem passar nesse ponto (ver Figura 1).

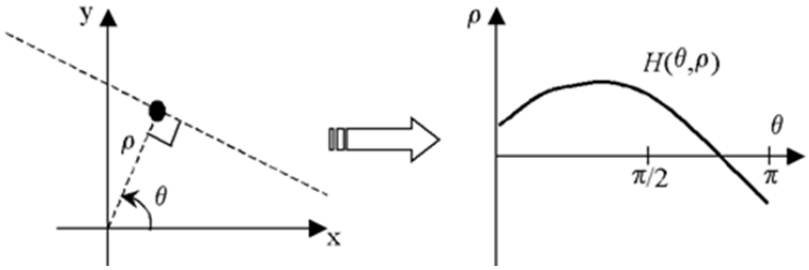

Figura 1: Uma reta no plano (x, y) é representada por um ponto no plano (ρ, θ), logo para cada ponto no plano (x, y), há uma senóide no plano (ρ, θ) que representa todas as possíveis retas que passam por esse ponto.

O algoritmo funciona percorrendo a imagem binária e para cada pixel cujo valor seja não nulo, todas as possíveis retas que podem passar por esse ponto são computadas como uma senóide no plano (ρ, θ) e os pontos dessa senóide são adicionados como uma contribuição a um plano acumulador. Ao terminar de percorrer a imagem o plano acumulador terá máximos locais para cada reta, já que cada pixel pertencente a uma mesma reta adicionará a mesma senóide ao plano acumulador. Dessa forma, o plano acumulador pode ser visto como um histograma de duas dimensões e para se identificar as retas basta percorrer o histograma, definindo um valor mínimo onde qualquer ponto que ultrapasse esse valor acumulado será considerado uma reta na imagem original. Um exemplo visual do plano acumulador pode ser visto na Figura 2.

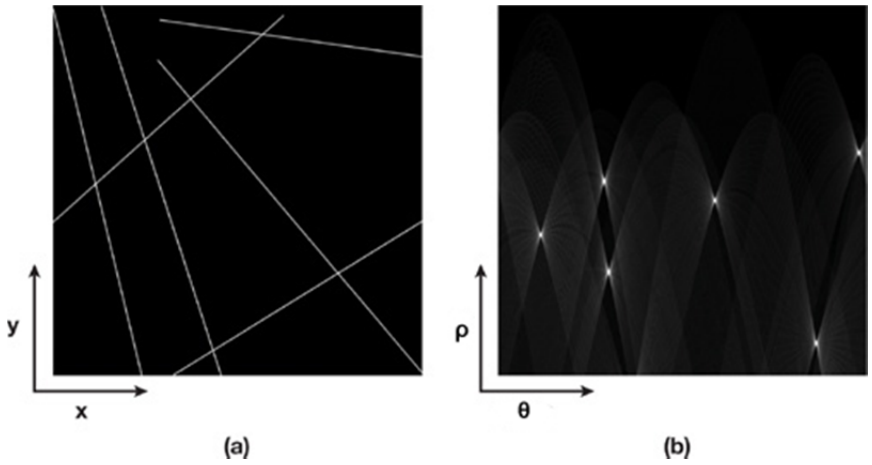

Figura 2: Linhas em uma imagem binária em (a) e o plano acumulador com seus máximos locais, aparecendo como pontos brancos, em (b)

De forma, análoga pode-se estender o método para a identificação de círculos [5], onde haveria uma variável a mais para ser encontrada, dado que a equação que descreve um círculo é definida como:

$$r^2 = (x - a)^2 + (y - b)^2$$

Equação 2: Um círculo de raio r, com centro em (a, b)

Logo para cada pixel não nulo na imagem, haveria uma contribuição em um volume acumulador de três dimensões para uma coordenada de centro (x, y) e um raio. Para evitar um grande esforço computacional e uma alta utilização de memória no reconhecimento de círculos, o método foi aprimorado para a detecção de derivadas, no que é conhecido como o método do gradiente de Hough.

O método do gradiente de Hough calcula para cada pixel não nulo na imagem binária o valor do gradiente nesse ponto. Considerando que o vetor gradiente de um círculo aponta para o centro do círculo, todos os pontos na reta associada a este gradiente, dado uma distância mínima e máxima do ponto em questão, são incrementados em um plano acumulador e a posição (x, y) do pixel é adicionada a uma lista. Os máximos locais no plano acumulador são os candidatos a centros de círculos e para se determinar o raio basta percorrer a lista de pixels não nulos e verificar a distância entre o centro e os pixels não nulos, caso certa distância apareça com frequência, esse centro e raio são escolhidos como um círculo válido.

Vale ressaltar que esse algoritmo possui certas falhas, onde círculos concêntricos tendem a serem encarados como um único círculo, excluindo um deles dos resultados, e uma imagem com muito ruído pode acabar gerando

círculos falsos. Como a Transformada de Hough deve procurar máximos locais, é natural que o algoritmo fique muito acoplado a valores que limitem um mínimo para que um valor seja considerado um máximo, isto pode acabar descartando círculos válidos em imagens diferentes fazendo o algoritmo ser muito dependente da aplicação.

Como um círculo visto de um ângulo não perpendicular ao seu plano aparece como uma elipse (Figura 3) é de se esperar que a Transformada de Hough não detecte corretamente uma elipse, e apesar de poder ser estendido para a detecção de elipses, outras formas de detecção apresentam um resultado melhor para a aplicação desejada neste trabalho.

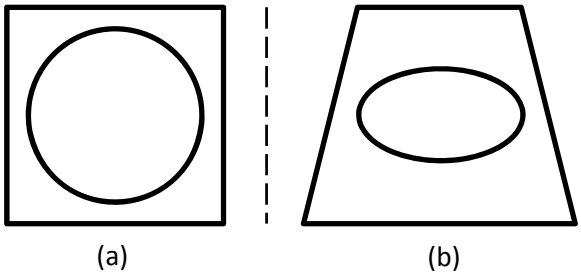

Figura 3: Um círculo visto de frente em (a) e o mesmo círculo visto inclinado em (b). Observa-se que o círculo aparece como uma elipse

Apesar de muito difundida, a Transformada de Hough não é o algoritmo ideal para o caso específico em estudo. Duas formas de detecção de círculos foram estudadas, a Transformada de Hough e a detecção de contornos para encaixe em formas elípticas. Enquanto a Transformada de Hough se apresentou muito vulnerável ao ambiente, produzindo por vezes círculos que não constavam na imagem original, a detecção de contornos e o posterior encaixe desses contornos em formas elípticas se mostrou uma técnica muito mais robusta, uma vez que o ambiente favorece esse algoritmo.

Por se tratar de uma implementação particular do *framework* OpenCV, a técnica de detecção de contornos será apresentada nas secções referentes à implementação do projeto em si.

2.2 Controlador PID

O Controlador Proporcional Integrativo Derivativo, ou mais simplesmente denominado de Controlador PID, é um método genérico de controle de processos altamente difundido na indústria e geralmente aplicado quando o modelo matemático do processo é desconhecido ou de difícil obtenção [6]. O Controlador PID reage em resposta ao erro medido no processo, tendendo a

minimizar esse erro.

De uma forma geral o controlador PID sempre tem uma variável que está sendo medida (como por exemplo, uma temperatura), chamada de Variável do Processo, uma variável que está sendo manipulada (como o valor da posição que uma válvula de ajuste da saída de uma caldeira deve ficar), a Variável Manipulada e um valor selecionado a ser atingido pela variável do processo (uma temperatura desejada a ser atingida), geralmente denominado como *setpoint*. A diferença entre o a variável do processo sendo medida e o valor selecionado como objetivo é chamado de erro.

O controlador PID possui esse nome em decorrência dos seus três blocos fundamentais: Proporcional, Integrativo e Derivativo, após executar cada um desses blocos aplicados ao erro previamente calculado e somá-los o controlador PID possui um novo valor a ser aplicado na variável manipulada.

A fórmula básica do controlador PID é:

$$VM(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

Equação 3: A fórmula geral do controlador PID

Conforme se observa na fórmula acima, o valor da variável manipulada (VM) é definido como uma constante Kp que multiplica o erro no instante avaliado (t), somado a uma constante Ki que multiplica a integral do erro no tempo, deste o início do processo até o instante sendo avaliado e somado a uma constante Kd que multiplica a derivada do erro medido naquele instante.

Sendo assim, fica evidente que ajustando as constantes Kp, Ki e Kd para valores adequados ao processo em questão, o controlador PID irá reagir ao erro medido alterando a variável manipulada, levando o erro a um valor cada vez menor.

Um diagrama de bloco pode ser visto na Figura 4 onde os termos proporcional integrativo e derivativo ficam bastante evidentes.

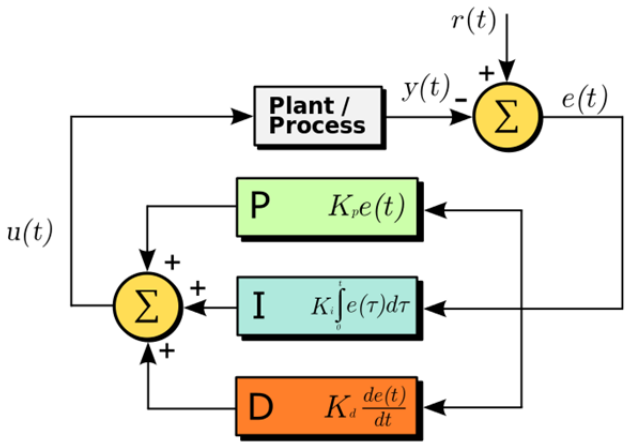

Figura 4: O controlador PID como um diagrama de bloco

O termo proporcional (Figura 5) possui um funcionamento bastante simples, ele somente multiplica o erro por uma constante, logo quanto maior for o erro, maior será o resultado da multiplicação e maior será o valor atribuído à variável manipulada. Com isso espera-se que o erro vá diminuindo até chegar a zero.

Entretanto devido a características limitantes próprias de cada sistema o erro tende a se estabilizar em um valor fixo, pois na maioria dos casos, é necessário um valor mínimo a ser atribuído à variável manipulada para que o controle tenha qualquer efeito. Um fator comum é o atrito que em geral demanda uma potência mínima para que qualquer movimento seja produzido.

Outro problema relacionado ao termo proporcional é a instabilidade caso a constante Kp seja muito alta. Nesse caso o sistema oscila devido à inércia da resposta do sistema em si. Para corrigir esses problemas, em geral o termo integral é necessário.

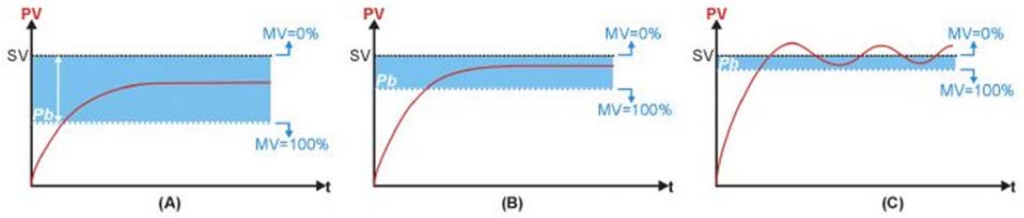

Figura 5: O termo proporcional do controlador PID. Em (a) e (b) observa-se como o sistema não consegue se estabilizar no valor desejado SV, utilizando diferentes valores para Kp e em (c) fica clara a oscilação devido a um valor muito alto de Kp

O termo integral (Figura 6) gera uma contribuição à variável manipulada proporcional ao erro acumulado no tempo. Dessa forma, conforme o erro se mantem constante, a integral cresce de valor e por sua vez a variável manipulada também crescerá de valor. Com isso, o termo integral consegue estabilizar o termo proporcional em caso de oscilações e evitar que o sistema

fique preso em valores abaixo do mínimo para que o controle surta efeito.

Porém o termo integral, se extrapolado, pode acumular muito erro e gerar oscilações na variável do processo. Em geral, ele acaba levando o sistema a valores excedentes ao desejado em uma oscilação amortecida conhecida como *overshoot*. O *overshoot* não necessariamente é um problema e muitas vezes acaba sendo desejado, pois ele acelera o tempo de estabilização na maioria dos casos, entretanto, em sistemas mais sensíveis que não toleram uma extrapolação dos seus valores, deve-se escolher um valor mais baixo para Ki, a fim de se evitar um *overshoot*, a troco de um provável maior tempo para o sistema atingir o seu *setpoint*.

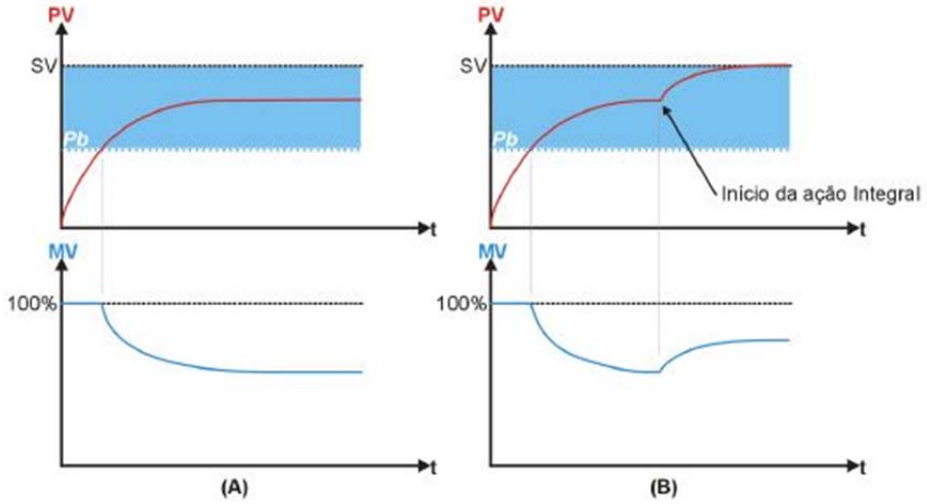

Figura 6: Em (a) um controle puramente proporcional, limitado a um erro fixo por não gerar valores mínimos para o controle surtir efeito. Em (b) percebe-se o efeito da inclusão do termo integral, resultando em um controle PI.

O termo derivativo (Figura 7) visa prever a ação do erro medindo a sua derivada no tempo e reagir de maneira proporcional. Dessa forma, no caso de uma reação abrupta por parte do sistema, o termo derivativo vai gerar um valor igualmente elevado que possa compensar o distúrbio inicial.

Em pequenas quantidades o termo derivativo costuma aumentar a estabilidade do sistema, porém se extrapolado ele pode levar a instabilidades por ser muito vulnerável a ruídos e modificações abruptas no valor de destino desejado.

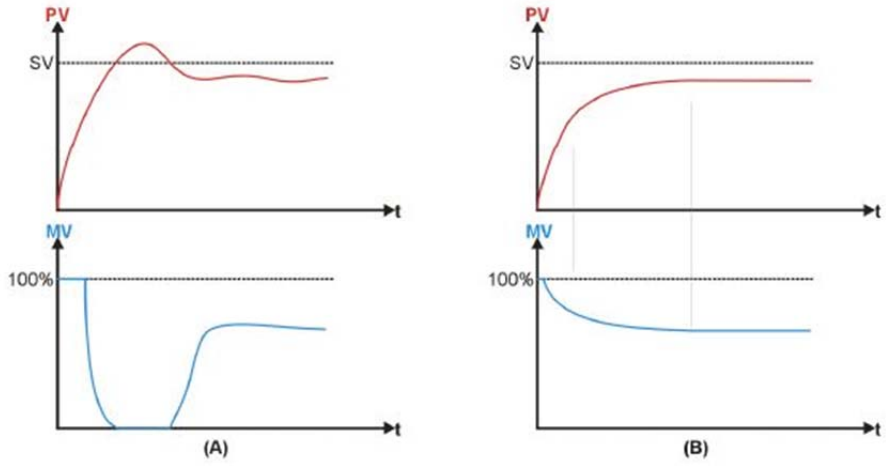

Figura 7: Um controle puramente proporcional em (a) e a adição do termo derivativo em (b)

O controlador PID apresentado é a forma mais básica a ser implementada, existem diversas modificações e melhorias que aumentam a segurança e a confiabilidade do controlador PID, porém fogem do escopo desse documento.

A definição das constantes Kp, Ki e Kd podem ser feitas de maneira experimental ou derivadas por métodos matemáticos da modelagem do sistema em si. Um método empírico muito utilizado é o de Ziegler–Nichols [7]. Ferramentas automáticas para a escolha desses valores estão disponíveis no mercado.

3. Especificação

Como a competição a qual o sistema se destina tem como objetivo identificar círculos brancos em um fundo preto, bem como acertá-los, a plataforma necessita de meios para identificar formas visuais no espaço tridimensional, bem como ser capaz de se posicionar de maneira a engajar os alvos previamente identificados.

A especificação do sistema foi baseada não somente em atender aos requisitos mínimos da aplicação final, mas também a deixar o sistema modular e expansível para outras aplicações futuras que possam se beneficiar de uma plataforma robótica que possua dois eixos angulares, posicionáveis por um protocolo genérico e que consiga movimentar um par de câmeras e uma carga útil.

Para atender a essas demandas um projeto multidisciplinar foi elaborado, envolvendo as principais três áreas da robótica, a engenharia de computação, eletrônica e mecânica.

3.1 Computação

A parte computacional do projeto se divide no software destinado ao usuário final que irá controlar o sistema como um todo em um computador e o software embarcado que lida com o posicionamento dos eixos angulares do robô.

Por se tratar de uma competição baseada em tempo, um software leve e de resposta rápida é o mais desejado, assim foi escolhida a linguagem C para o software embarcado e a linguagem C++ utilizando os *framework*s Qt e OpenCV para a interface com o usuário e o processamento de imagens, respectivamente.

Como o sistema deve detectar os alvos enquanto se movimenta, câmeras com altas taxas de aquisição são necessárias, para se evitar imagens borradas que dificultem a identificação correta dos círculos. Entretanto não há a necessidade de alta resolução de imagem já que as formas a serem detectadas são simples e somente se deseja o seu contorno.

O protocolo de comunicação entre o software do usuário final e a eletrônica responsável pelo posicionamento do robô deve ser robusto e de alta taxa de transmissão de dados, afinal é essencial para o posicionamento correto do sistema que a comunicação seja livre de erros e que qualquer nova posição a ser alcançada, seja recebida pelo controle eletrônico o mais rápido possível.

Algumas facilidades ao usuário final também são desejadas. As capacidades de calibração do robô sem a necessidade de reprogramar o sistema eletrônico e uma interface que mostre ao usuário o estado geral do sistema ajudam na detecção de erros e agilizam a instalação do robô na competição.

3.2 Eletrônica

O projeto eletrônico deve ser capaz de receber comandos de posicionamento, informar a atual posição em que os eixos do robô se encontram e posicionar os mesmos conforme os comandos recebidos.

É desejado que a eletrônica do sistema possua um tempo de resposta baixo, podendo calcular e informar a posição do robô sem demoras e com precisão.

Para o controle da movimentação do robô são necessários sensores de posição angular de alta precisão, bem como motores que consigam movimentar o robô sem sofrerem esforço excessivo. Para lidar com a leitura de posição por

parte dos sensores, o controle do acionamento dos motores, a comunicação com o computador e todos os cálculos a serem executados é imprescindível que o projeto eletrônico inclua tanto uma parte de eletrônica de potência quanto uma parte de processamento e aquisição de dados digitais.

Interfaces externas como botões para controle rápido e monitoramento do estado do sistema eletrônico por luzes indicativas também são desejadas, porém mais como um adendo e não um requisito.

3.3 Mecânica

Para que se construa um sistema de precisão, a execução do projeto mecânico é de extrema importância. Como o sistema deve carregar tanto as câmeras quanto o disparador, é necessária uma estrutura rígida e resistente.

A estrutura deve ser capaz de se movimentar em dois eixos angulares independentes, *yaw* e *pitch*, de forma que o sistema eletrônico possa posicionar os mesmos conforme necessitar.

Folgas e desalinhamentos não são toleráveis já que como o sistema deve funcionar a uma distância media de 3 metros dos alvos, qualquer pequeno erro de alinhamento na estrutura pode significar um erro muito maior no alvo em si.

Para evitar esforços desnecessários nos motores, a estrutura deve estar balanceada com os centros de gravidade de todas as partes moveis alinhados com os seus respectivos eixos, a fim de se evitar um torque externo resultante que se oponha ao movimento dos motores.

É desejado que a estrutura seja leve e pequena para facilitar o transporte do robô à competição, bem como diminuir gastos com material e evitar que a massa do sistema influencie negativamente a movimentação do sistema, adicionando inércia.

4. Atividades Realizadas

Por se tratar de uma plataforma bastante complexa, nem todas as atividades planejadas puderam ser executadas, entretanto grande parte do que foi desejado no inicio do projeto pode ser alcançado.

4.1 Computação

A plataforma desenvolvida possui dois sistemas computacionais independentes: o sistema embarcado, realizando o controle de posicionamento, comunicação com o computador e ativação do mecanismo disparador, e o software que será executado no computador e permitirá ao usuário o controle

total do sistema. Um diagrama relacionando todas as partes funcionais que compõem o sistema computacional, bem como seus periféricos é apresentado na Figura 8.

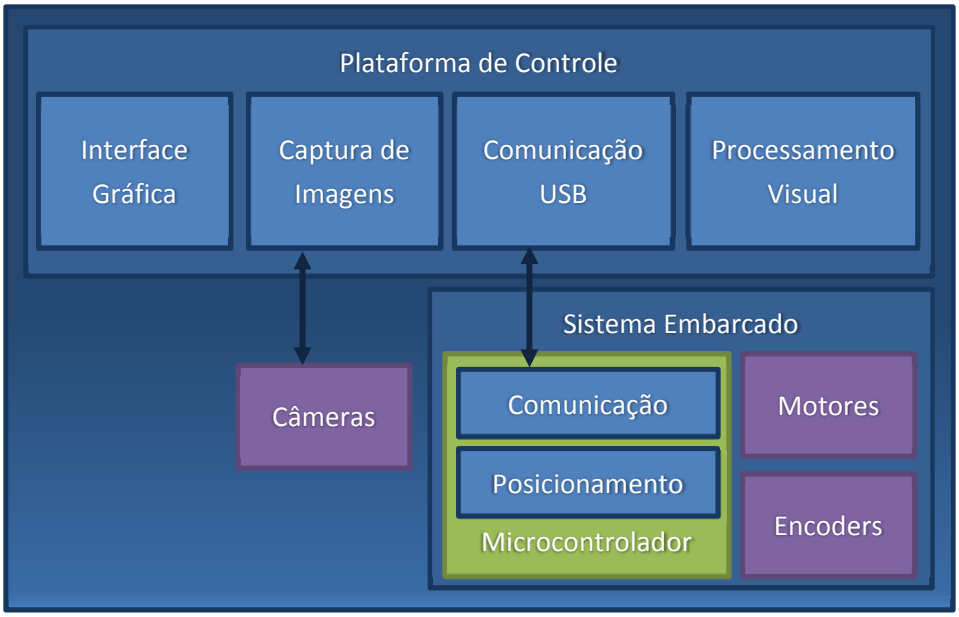

Figura 8: O Diagrama do Sistema

4.1.1 Sistema Embarcado

O sistema embarcado foi desenvolvido utilizando o Atmel Software *Framework*, mais conhecido como ASF, que abstrai o acesso às funções de baixo nível do microcontrolador Atmel como funções de linguagem C.

Apesar do ASF ser um *framework* bastante difundido e robusto, foi detectado e corrigido um erro na biblioteca, relativo à configuração dos pinos do microcontrolador. Além dessa modificação, outras funções também foram criadas dentro do ASF, para complementar as funcionalidades já providas pelo *framework*.

Dentre as funções incluídas pode-se citar a criação de uma função de configuração assíncrona de PWM, onde o ASF somente permitia a configuração síncrona, a inclusão de uma função de envio e recebimento mútuo de dados para comunicações que utilizam o protocolo SPI, enquanto que o ASF somente permitia a escrita ou o recebimento de dados, enquanto que o protocolo SPI permite o envio e recebimento na mesma transação. E por fim, a adição de uma função que permite a leitura de 16 bits de dados em uma única transação, o que facilita e agiliza a leitura dos sensores encoders de 12 bits.

Para a comunicação entre o computador e o sistema embarcado foi escolhida a especificação USB, utilizando a classe de periféricos HID, Human Interface Device, com transferência por interrupção, já que se trata de uma

conexão com detecção automática de erro e latência garantida, além de possuir uma interface genérica e de fácil configuração para a transferência dos dados.

Todo o trafego de dados é feito pela transferência de 64 bytes entre o computador e o robô, cabendo a ambos preencher o vetor de bytes na ordem correta e identificarem cada posição no vetor como um dado especifico. Dessa forma, a ordem em que os dados estão dispostos no vetor fixo de 64 bytes define o protocolo de comunicação entre o software de controle e a eletrônica controlada.

O controle de posicionamento foi baseado no controlador PID, onde o aluno desenvolveu uma biblioteca genérica de controle PID que permite tanto uma referência de controle linear quanto circular. A diferença entre os dois é que no caso da referência circular, o controlador PID sempre irá movimentar o sistema de forma a percorrer o menor caminho, seja ele em sentido horário ou anti-horário, enquanto que em uma referencia linear, o controlador PID sempre realiza o posicionamento sem considerar que o sistema possui uma ligação entre o seu valor de posição máxima e mínima, ficando impossibilitado de realizar um movimento que cruze esses limites.

O problema fica evidente se considerarmos que o algoritmo do controlador PID se baseia na medição do erro entre a posição desejada e a posição atual, com isso, caso o sistema queira realizar um giro da posição 10º à posição 300º, o controlador PID, sem considerar que há a possibilidade de cruzamento, acabaria concluindo um erro de 290º, enquanto que para um sistema que permite tal cruzamento, o erro na realidade é de apenas 70º.

Dessa forma, caso o usuário informe que deseja utilizar uma referência circular e calibra no sistema o valor relativo à medição de 180º, o sistema será capaz de sempre percorrer o menor caminho entre duas posições desejadas, enquanto que se o mesmo fosse tentado com uma referência linear, o sistema iria ignorar a possibilidade de cruzamento entre a posição 0º e 360º.

Foram adicionadas à biblioteca do controlador PID algumas melhorias em relação ao algoritmo básico. A biblioteca agiliza o cálculo dos termos por estar localizada em uma interrupção de timer, que garante uma frequência de execução constante. A biblioteca também prevê o controle do termo derivativo caso o usuário mude o valor de destino desejado, causando uma alta derivada no erro. É possível calibrar o PID enquanto o sistema está funcionando sem que as variáveis internas dependentes de valores passados sejam influenciadas, bem como é possível ligar e desligar o controlador PID, prevendo inicializações

abruptas.

Para manter o controle do sistema bastante fluido, o controlador PID é executado a cada 1 milissegundo, porém o envio de dados ao computador somente se dá a cada 50 milissegundos, a fim de se evitar um volume excessivo de dados a ser processado pelo computador. Entretanto, os comandos enviados pelo computador ao sistema embarcado são processados no momento em que chegam. Vale ressaltar que o tempo de 50 milissegundos pode ser alterado para garantir uma boa resposta em aplicações de rastreamento, porém esse período se mostrou satisfatório na aplicação do projeto e se necessário pode facilmente ser reconfigurado para um intervalo menor.

A cada transferência, o sistema embarcado envia a posição absoluta dos seus eixos, os ângulos máximos e mínimos que ele consegue se posicionar, o valor considerado como zero para a calibração, os ângulos de destino a serem alcançados, a tensão a ser aplicada nos motores como um resultado calculado pelo controlador PID e se caso o sistema se encontra em uma posição que não deveria, extrapolando os seus limites angulares, um aviso é enviado que indica essa condição de erro, incluindo a posição em si. Dessa forma o sistema conclui que os motores foram desligados como uma reação de segurança.

Os possíveis comandos a serem enviados ao sistema embarcado são: a configuração das constantes do controlador PID, Kp, Ki e Kd, a ativação e desativação do controle PID, o envio de uma nova posição de destino, o envio de um acréscimo posicional em que o sistema apenas informa quantos graus a mais ou a menos o sistema necessita girar, sem informar posições absolutas e o envio de uma velocidade fixa em que o sistema deve girar. Vale ressaltar que a velocidade fixa não é recomendada e deve somente ser enviada em caso de testes de potência ou no caso em que o usuário deseja controlar o robô como um todo pelo computador. O protocolo também define o comando de tiro em que o usuário informa o numero de projeteis a serem atirados, porém devido a limitações mecânicas o mesmo não foi desenvolvido. Todos os valores enviados ao sistema embarcado são testados quanto a sua validez, sendo assim, caso o robô receba uma nova posição de destino que está além dos seus limites, ele ignorará o comando enviado.

Por possuir funções que funcionam pelo mecanismo de interrupção, onde uma função pode interromper o funcionamento de outra, gerando possíveis problemas de concorrência a recursos compartilhados, foi integrado ao sistema um mecanismo de exclusão mútua que garante o acesso atômico a todos os

recursos compartilhados.

De forma a se evitar altos torques na estrutura, foi adicionando ao sistema a opção de controle de aceleração, garantindo que não haverá mudanças repentinas de velocidade, ajudando a manter a integridade estrutural do sistema e evitando desgastes desnecessários.

4.1.2 Plataforma de Controle

O controle do sistema com um todo foi estruturado para ser executado em um computador pessoal que utiliza o sistema operacional Windows®. Para atender aos requisitos de velocidade e modularidade, a linguagem de programação C++ foi escolhida para o desenvolvimento do software.

4.1.2.1 A Estrutura

O *framework* Qt é utilizado de maneira extensa no projeto, tanto para a criação da interface gráfica quanto para o funcionamento geral do sistema. O software possui uma estrutura *multithread*, onde linhas de execução independentes adicionam modularidade e velocidade ao sistema.

A *thread* principal é a responsável pelo gerenciamento do sistema, recebendo comandos da interface gráfica do usuário e gerenciando as outras *threads*. Há uma *thread* somente responsável pela leitura e envio de dados ao robô, encapsulando todo o acesso ao protocolo de comunicação. Outra *thread* é responsável pela captura de imagens do par estéreo de câmeras e outra *thread* lida com o processamento das imagens em si, é nessa *thread* que todo o controle autônomo deve ser desenvolvido.

Dessa forma, o sistema possui um alto nível de desacoplamento onde não somente existem classes separadas para cada atividade, como cada uma executa em uma linha de execução independente.

Figura 9: A estrutura geral do software

A comunicação entre processos é feita pelo mecanismo de "*Signal and Slots*" do *framework* Qt [8]. Esse mecanismo possibilita que caso um sinal (*Signal*) seja disparado, um ou mais métodos (*Slots*), registrados a esse sinal serão executados (Figura 10). Além de possibilitar um alto grau de modularidade, já que cada emissor de um sinal não necessita saber de antemão

quantos ou quais *slots* estão conectados ao seu sinal, esse mecanismo também torna possível a execução de diversos métodos ao mesmo tempo, mesmo que eles estejam em *threads* de execução diferentes.

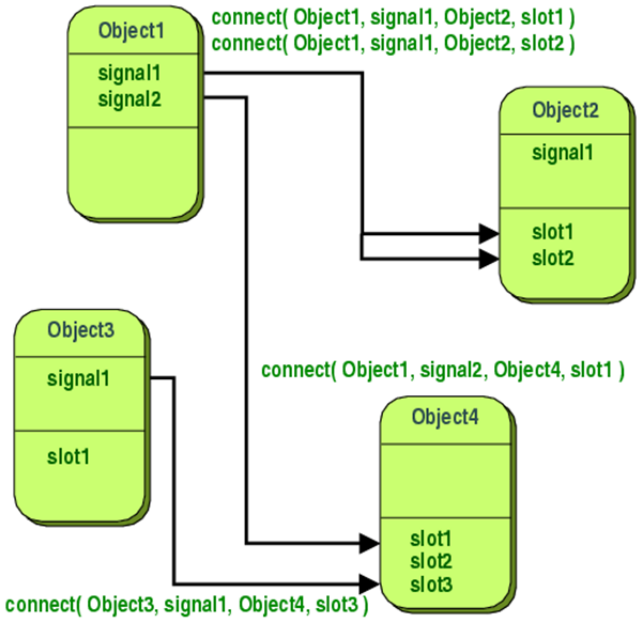

Figura 10: Framework Qt Signal and Slots

Com essa estrutura de comunicação *multithread*, o sistema consegue enviar os dados para todos os processos que necessitem dos mesmos. Dessa forma, a *thread* responsável pela captação do estado do sistema embarcado, dispara um sinal contendo o estado do sistema e todas as *threads* que necessitam dessa informação a recebem praticamente ao mesmo tempo. O mesmo acontece com as câmeras, a *thread* de captura dispara um sinal contendo os dois frames capturados e tanto a *thread* de processamento visual quanto a *thread* de interface gráfica recebem essas imagens.

Há ainda uma interface separada somente para a calibração do controlador PID, onde o usuário pode ver com mais detalhes todo o estado atual do robô. Ambas as interfaces disponibilizam gráficos que informam a posição atual e destino do sistema, porém somente na interface de calibração que existem gráficos mais complexos que mostram o avanço de cada eixo do robô em relação ao tempo, permitindo uma visualização do tempo de resposta e comportamento do controlador PID.

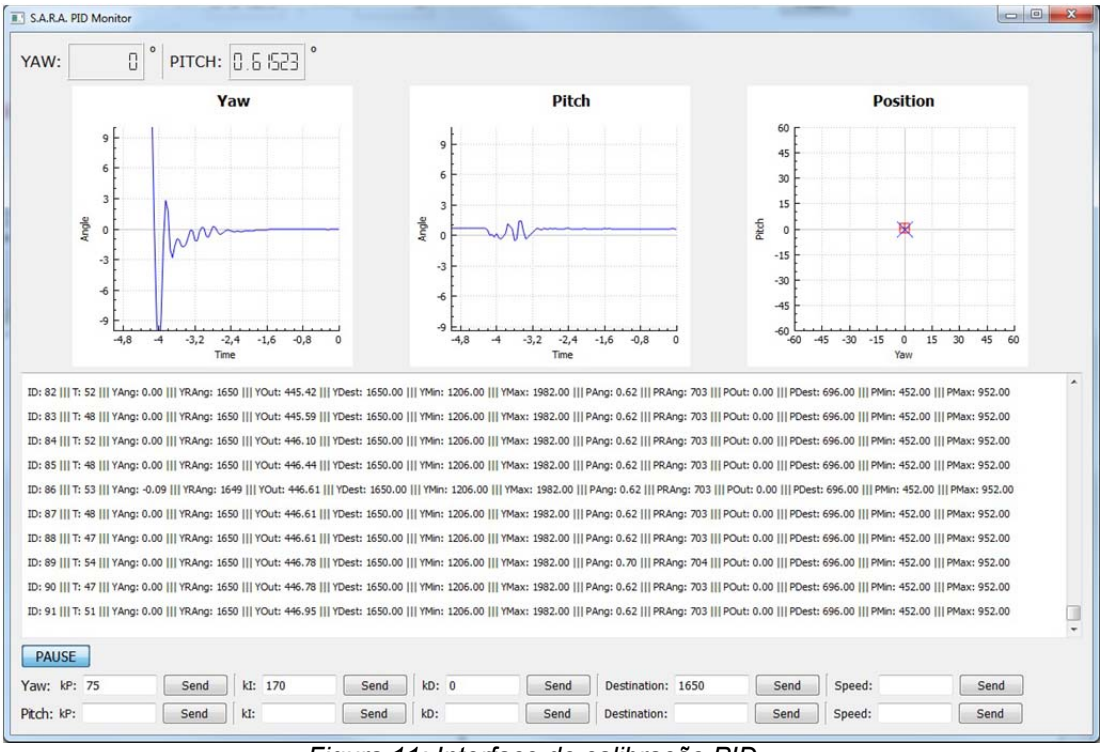

Figura 11: Interface de calibração PID

Na interface da Figura 11 é possível ver um gráfico dedicado a cada eixo mostrando o avanço da posição angular em relação ao tempo e um terceiro gráfico, onde é possível ver a posição atual do robô, relacionando os dois eixos, denotada pelo quadrado vermelho com uma cruz em seu centro, e a posição desejada de destino, como o xis azul. Há também uma caixa de texto que imprime todas as variáveis sendo transmitidas do robô, bem como o tempo em que cada informação foi recebida.

A interface geral do sistema é apresentada na Figura 12. Pode-se ver que o usuário recebe de maneira instantânea as informações principais do robô. Ao topo é possível ver diretamente o valor numérico dos ângulos em que a plataforma se encontra e logo ao lado são informados estados de controle geral do sistema, como se ele está armado ou seguro e se está em modo de controle manual ou automático. Caso o usuário pressione o botão "*Run*" a *thread* de controle autônomo toma o controle do sistema, porém somente se o circuito se encontra conectado e há um número mínimo de câmeras também presente. Para interromper o controle autônomo, basta pressionar o mesmo botão, que agora exibe o texto de "*Stop*". Durante o controle autônomo os controles manuais do sistema são desabilitados.

Há um gráfico de posição absoluta idêntico ao do controlador PID, com a adição de um controle de posicionamento por seleção, onde, caso o controle

autônomo não esteja em execução, ao clicar em uma coordenada do gráfico, o robô irá se mover para a mesma. Logo abaixo do gráfico de posição é possível ver duas áreas dedicadas a mostrar a imagem capturada pelo sistema. Selecionando as abas o usuário pode ver as imagens diretamente como elas são enviadas das câmeras, ou como a imagem fica logo após o processamento de identificação de círculos, bem como misturar as duas opções na terceira aba.

Caso a ordem das câmeras foi invertida, ou a imagem que deveria aparecer como a câmera esquerda está local da câmera direita e vice-versa, o usuário pode selecionar uma opção no menu superior que permite a troca da ordem das câmeras fazendo com que o sistema interprete a imagem esquerda como a direita e vice-versa.

Ao lado do gráfico há um controle direcional que permite tanto um rápido posicionamento do robô, como a configuração de novos limites (selecionados pela troca de um botão), para evitar que o mesmo não venha a se posicionar na área de outro competidor ou mire em posições em que há pessoas passando. Porém, conforme mencionado anteriormente, para um controle rápido, o usuário também pode clicar no gráfico e o robô irá se mover para a posição selecionada.

Na parte inferior da janela é apresentada uma barra de status, sendo dividia em três áreas para informações relativas à conexão do circuito eletrônico, o erro de posicionamento do controle autônomo e o número câmeras conectadas.

A intensidade do filtro de *Threshold*, que será explicado mais adiante pode ser ajustada deslizando o controle direcional que fica logo acima das imagens, de forma a adequar melhor o filtro ao ambiente em questão.

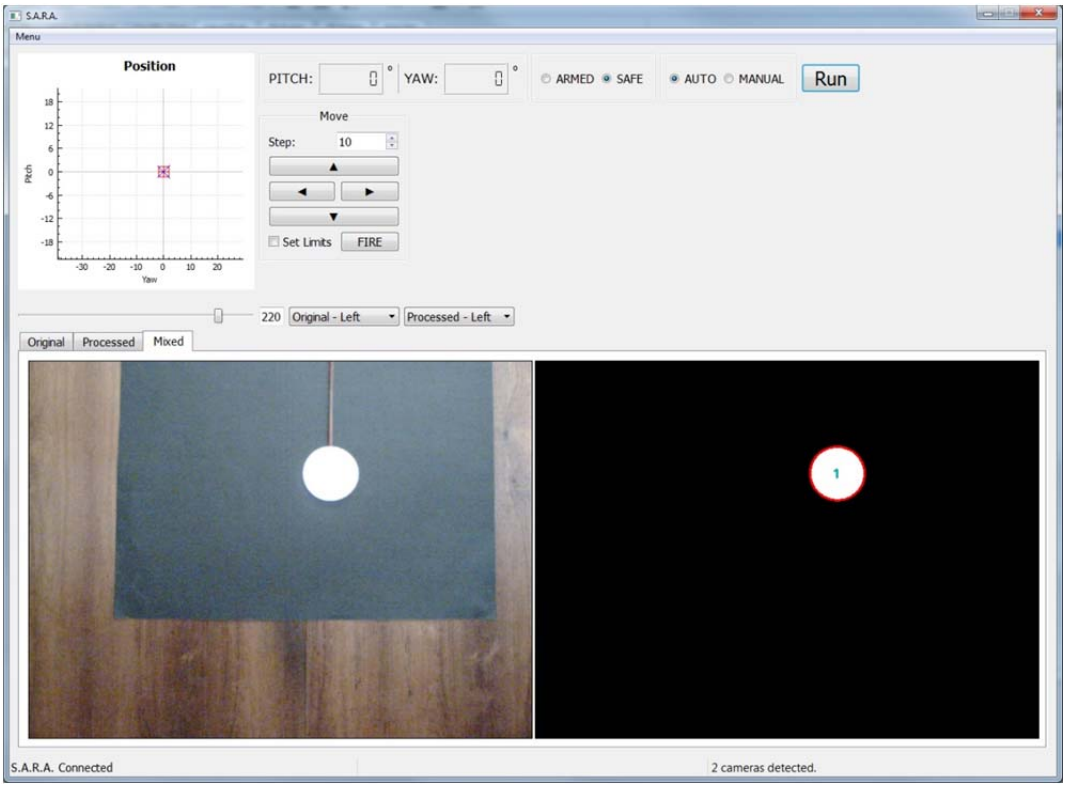

Figura 12: A interface geral do sistema

4.1.2.2 Detecção de Círculos

Para atender a especificação original do projeto, duas câmeras PlayStation Eye® foram escolhidas. Essas câmeras possuem uma alta taxa de quadros por segundo, podendo chegar a 187 fps em resolução QVGA (320 x 240 pixels) e 75 fps a VGA (640 x 480 pixels). Apesar da resolução das imagens não ser alta, essas câmeras são capazes de retornar ótimas imagens para rastreamento o que é essencial para o projeto. A configuração escolhida para as câmeras foi de resolução VGA a 60 fps.

Conforme dito anteriormente a Transformada de Hough não se mostrou a melhor solução para a aplicação final. Um método mais robusto foi elaborado que consiste na detecção de contornos presentes na imagem e o encaixe dos mesmos em formas elípticas. O *framework* OpenCV foi utilizado para o todo o processamento de imagens [9].

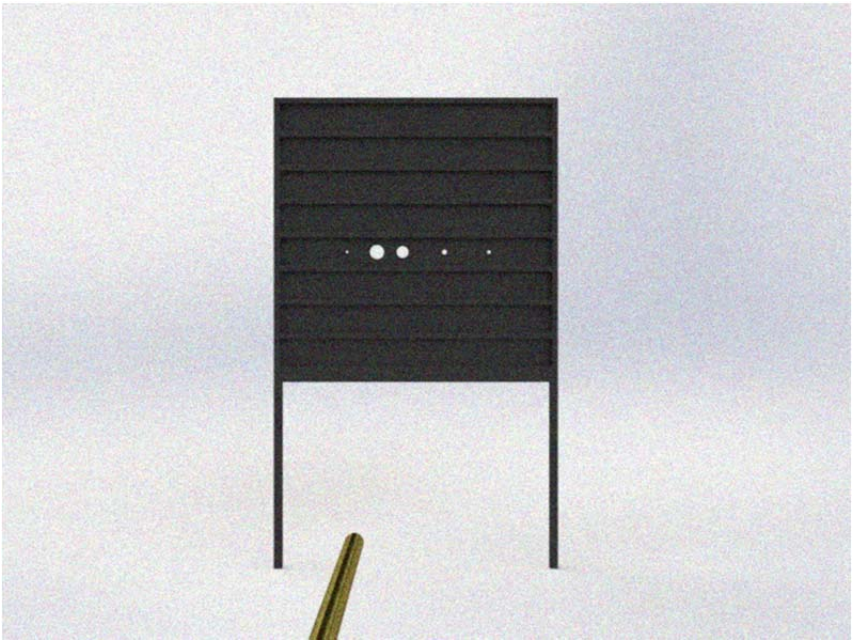

Figura 13: Imagem sem processamento

Para exemplificar a extração dos contornos, uma imagem simulando uma das câmeras na plataforma robótica foi gerada no software de desenho técnico *SolidWorks 2013*.

A imagem original (Figura 13) primeiramente passa por um filtro gaussiano que retira componentes de alta frequência da imagem, como ruído, deixando a imagem mais uniforme e levemente borrada e então a imagem é convertida para níveis de cinza (Figuras 14 e 15 respectivamente), logo após um filtro de Limiar (mais conhecido como *Threshold*) é aplicado, onde um valor é definido como um limite e pixels com valores de tons de cinza abaixo desse valor são convertidos para preto e acima para branco, formando uma imagem binária, contendo apenas valores brancos ou pretos, conforme a Figura 16. Como o sistema deve atuar em um fundo preto com círculos brancos, qualquer ruído ou diferença de tonalidade é automaticamente excluída nessa etapa.

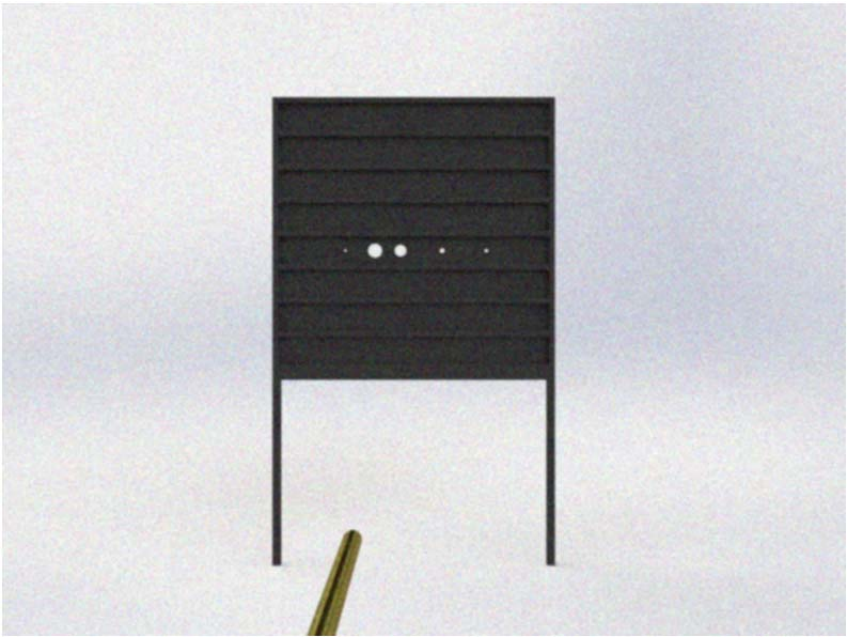

Figura 14: Imagem com filtro gaussiano

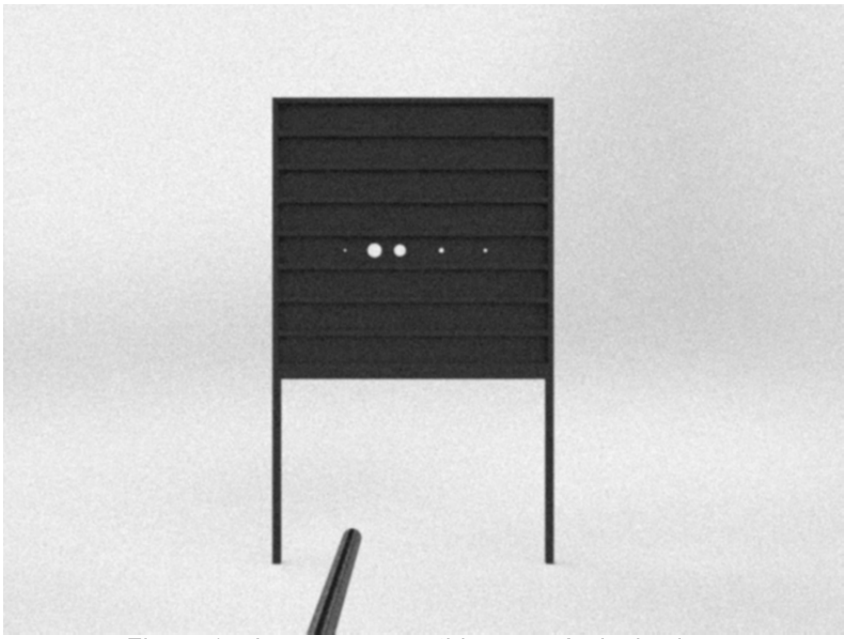

Figura 15: Imagem convertida para níveis de cinza

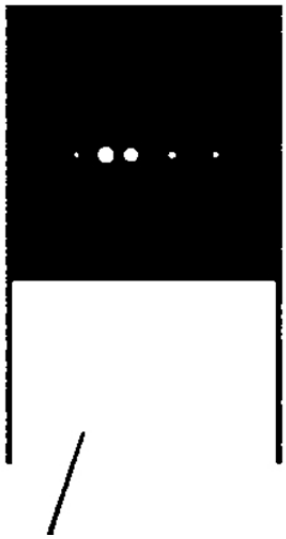

Figura 16: Imagem binária, após passar pelo filtro de Limiar (Threshold)

De posse dessa imagem binária, basta realizar a detecção de contornos, que no OpenCV é realizada com a função *findContours*, utilizando o algoritmo de Suzuki [10],retornando uma lista de contornos encontrados, onde cada contorno é um conjunto de pontos. Cada contorno possui uma caixa envolvente que é constituída pela largura máxima e altura máxima do contorno, ao mesmo tempo que o OpenCV permite a criação de uma elipse na imagem utilizando apenas uma caixa envolvente que contenha os eixos da elipse e o ângulo de inclinação da mesma.

Sendo assim, utilizando a caixa envolvente do contorno para a criação da elipse, sem nenhuma inclinação (já que a projeção dos círculos em formas elípticas não produzirá nenhuma elipse inclinada), podemos definir uma elipse que terá a forma exata de cada contorno circular encontrado, podendo assim, identificar facilmente os círculos na imagem, bem como as suas dimensões e posição no espaço.

Entretanto é normal que outros contornos, além dos círculos brancos, possam ser encontrados. Isso se dá principalmente por focos de brilho na imagem, que podem criar áreas brancas que não seriam retiradas pelo filtro de *Threshold*. Para garantir um método de identificação mais robusto, foi adicionado uma etapa a mais ao algoritmo que realiza uma comparação de áreas, de forma a ver se a área do contorno detectado é semelhante a área da elipse atribuída a ele.

A área do contorno é calculada pelo OpenCV por aproximações dos

pontos que definem a sua forma utilizando a Fórmula de Green [11], enquanto que a área da elipse é dada pela Equação 4, onde *a* e *b* são os semi-eixos da elipse. Assim, no algoritmo adotado, é tomado o valor absoluto da diferença das áreas do contorno e da elipse, caso essa diferença seja menor que 0,1% da área da elipse, o contorno encontrado é considerado uma elipse, caso contrário, ele é descartado.

$$A_{elipse} = a * b * \pi$$

Equação 4: Área de uma Elipse

O resultado fica claramente ilustrado na Figura 17, onde o algoritmo foi completamente aplicado somente em uma área da imagem, evitando esforço computacional desnecessário. Cada elipse detectada foi marcada com numero único em azul turquesa em seu centro e o seu contorno foi acentuado com uma linha vermelha.

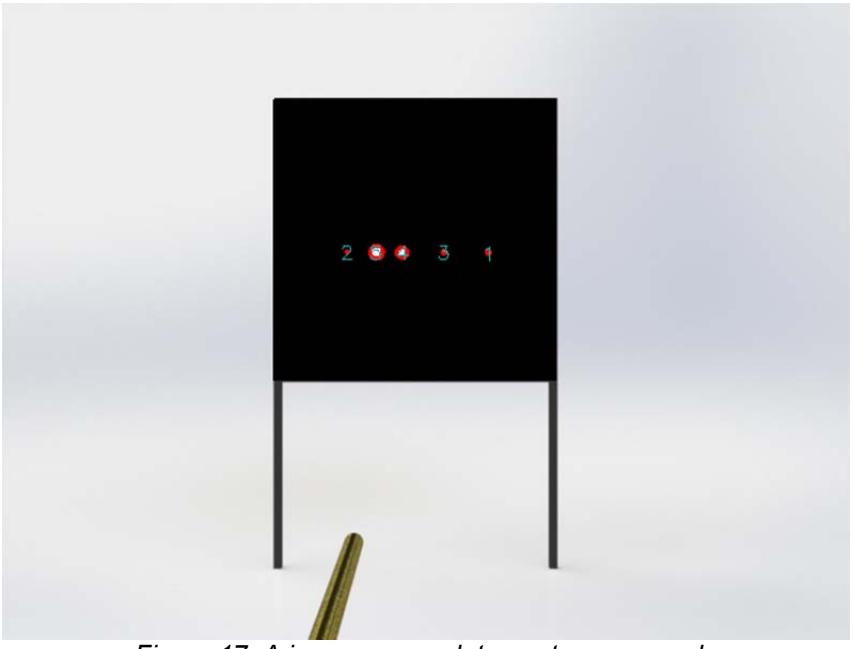

Figura 17: A imagem completamente processada

Mesmo com uma certa distorção o sistema ainda identifica formas elípticas claramente conforme pode ser visto na Figura 18.

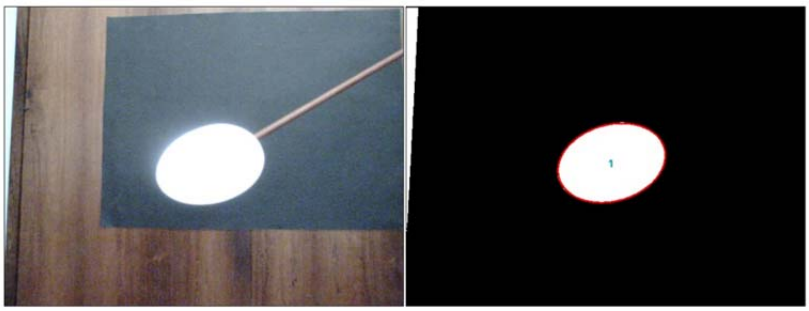

Figura 18: Uma elipse corretamente identificada e numerada

4.1.2.3 Controle Autônomo Sugerido

Por ser altamente dependente do sistema mecânico, o controle autônomo de posicionamento e seleção de alvos não pode ser desenvolvido a tempo, já que houve diversos problemas de logística para a construção do robô. Entretanto, toda a estrutura para o seu desenvolvimento foi feita, fazendo com que a implementação do controle seja bastante modular e necessite de configurações externas mínimas.

O controle original foi estruturado de forma que inicialmente o usuário defina limites para ambos os eixos de atuação do robô, garantido que o robô não os ultrapassará, invadindo a área de atuação de outro robô durante a competição. De posse desses limites, o algoritmo iria controlar o robô a primeiramente percorrer a extensão dos seus limites, incluindo em uma lista todos os alvos identificados, junto com seu tamanho, se são fixos ou móveis e sua posição no espaço.

A posição no espaço tridimensional de cada alvo é importante para se manter uma alta precisão no alimento do robô com os alvos, entretanto a obtenção automática dessas coordenadas não é uma tarefa muito trivial. A princípio foi escolhido um par estéreo de câmeras para se capturar uma imagem tridimensional dos alvos, porém não é garantido que um par estéreo de câmeras irá sempre calcular de maneira correta a triangulação tridimensional de um alvo, uma vez que esse cálculo depende das diferenças de posicionamento de um mesmo alvo nas imagens de ambas as câmeras [12].

Como os alvos são muito parecidos, o algoritmo pode ter uma dificuldade de casar as semelhanças entre as imagens, levando a dois alvos distintos serem identificados erroneamente como o mesmo alvo, produzindo uma triangulação errada.

Outras técnicas como a utilização de sensores de distância ou a informação prévia da distância perpendicular ao plano dos alvos também não são ideias, uma vez que sensores de distância que atuam acima de três metros costumam ser muito caros, e uma informação prévia fornecida pelo usuário faz com que o sistema somente possa atuar em casos estáticos, desconsiderando qualquer desalinhamento entre o robô e o plano dos alvos.

Ainda assim, provavelmente essas técnicas apresentarão uma maior confiabilidade que a identificação de distancia por meio de visão estéreo, porém vale ressaltar que nenhuma medida de distância chegou a ser testada e comparada nesse projeto.

Uma vez que essa lista seja preenchida o algoritmo iria traçar uma rota que percorreria os alvos de forma a garantir uma maior pontuação em menor tempo. Essa decisão poderia ser ajustada para primeiramente percorrer os alvos maiores, que são mais fáceis de acertar, ou de simplesmente engajar todos os alvos de uma vez, traçando a menor rota que percorra todos os alvos. Essas decisões teriam que ser testadas com a plataforma totalmente fabricada, já que elas são extremamente dependentes da precisão final do sistema.

Para a seleção da menor rota que percorra todos os alvos desejados, um algoritmo que resolva o problema do Caixeiro Viajante, como a heurística de Lin–Kernighan [13], pode ser utilizado, ou a simples seleção do alvo que se encontra mais perto da posição atual do robô também pode ser adotado, embora não seja ótimo, para as distâncias a serem percorridas é bastante provável que seja suficiente para uma boa velocidade final de engajamento, embora não tenha sido testado e é apenas uma suposição.

Uma vez em frente ao alvo, o disparador seria acionado por um período curto enquanto que o alvo selecionado seria monitorado, de forma a ver se ele já foi derrubado. Caso o alvo não tenha sido derrubado um percurso de padrão espiral pode ser adotado como forma de compensar erros na trajetória dos projéteis, contanto que a espiral não se expanda muito, pois existem alvos escondidos em preto que caso sejam derrubados ocasionam na perda de pontos.

O cálculo de posicionamento deve ser calibrado uma vez que a plataforma esteja completamente montada, afinal o eixo de tiro do robô é teoricamente paralelo ao eixo das câmeras, porém é de se esperar que erros de fabricação adicionem uma certa translação e rotação entre os eixos, que afete o cálculo como um todo. Sendo assim, no estado em que o projeto se encontra, esses cálculos não foram definidos.

Uma vez que a posição de destino é calculada, basta enviá-la ao robô que ele mesmo se encarregará dos movimentos necessários para se atingir essa posição. Conforme mencionado anteriormente, também é possível enviar ao robô um acréscimo a ser percorrido em cada eixo, de forma a se evitar o cálculo de uma nova posição, baseado na posição atual, quando somente se quer mover a plataforma em uma dada direção. Esse controle é especialmente útil para o posicionamento inicial do sistema por parte do usuário.

4.1.2.4 Controle Autônomo Realizado

Embora o controle autônomo necessário para a aplicação final do sistema não tenha sido desenvolvido, outro controle autônomo, mais simples, foi feito de forma a testar as capacidades do sistema de realizar um posicionamento guiado por visão computacional.

Nesse controle um único círculo é identificado e o sistema se movimenta de forma a posicionar o centro da imagem da sua câmera esquerda no centro do círculo, assim, conforme o círculo se mexe, o robô acompanha o seu movimento.

O posicionamento é feito utilizando um controlador PID bastante simplificado, onde apenas um termo proporcional é necessário, tomando a diferença em pixels entre o centro do círculo encontrado e o centro de imagem da câmera como medida de erro para o algoritmo, gerando um avanço nos eixos de *pitch* e *yaw* como saída.

Um exemplo do posicionamento guiado por visão computacional pode ser visto na Figura 19. Um único círculo foi apresentado ao sistema e corretamente identificado e automaticamente posicionado ao centro da imagem. A imagem original da câmera esquerda é apresentada à esquerda e a mesma imagem processada é apresentada à direita. Na barra de status é possível ver o erro de posicionamento no momento da captura da imagem. Vale ressaltar que devido a falta de um termo integrativo e derivativo nesse controlador PID, o erro nunca será completamente removido, conforme explicado anteriormente.

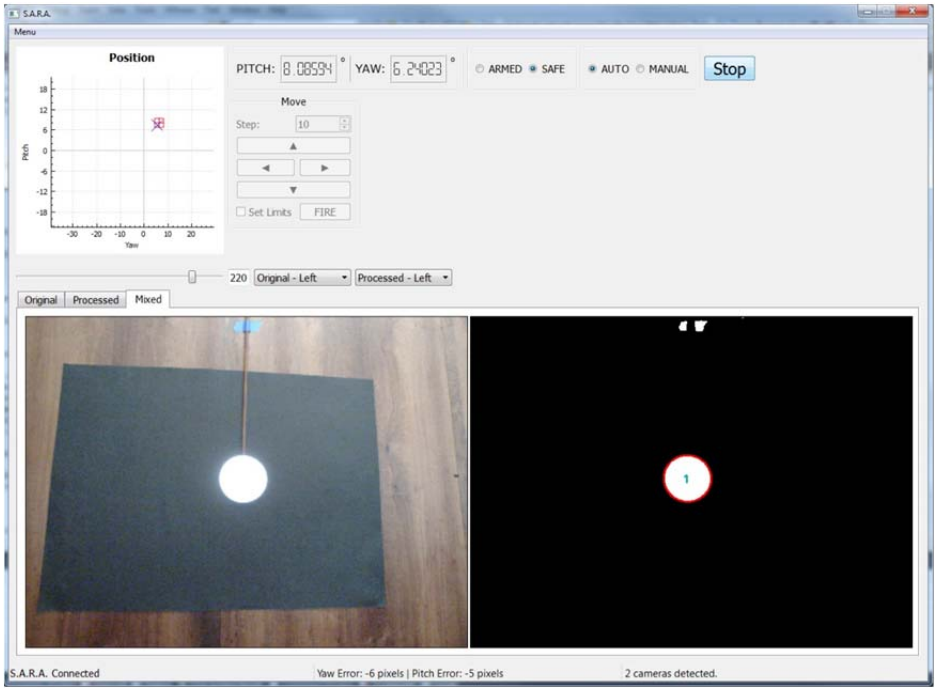

Figura 19: Um círculo identificado e numerado, posicionado ao centro da imagem

Caso o círculo seja movido para outro lugar, o sistema corrige automaticamente o seu posicionamento, de forma a garantir que o círculo se encontre ao centro da imagem capturada, conforme demonstrado na Figura 20.

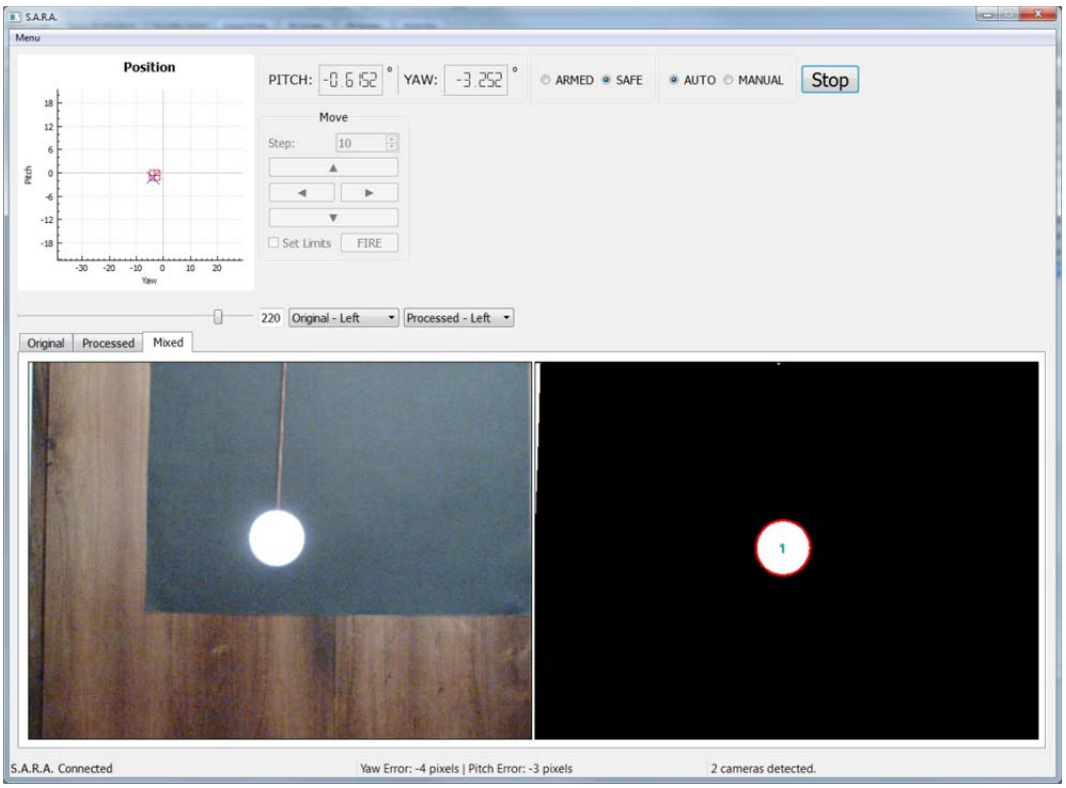

Figura 20: Movendo o círculo, o sistema o acompanha

4.1.3 Testes

Durante o andamento do projeto foi avaliado que a criação de classes de testes unitários para todos os módulos criados, apesar de muito apreciada, iria atrasar em muito o cronograma de desenvolvimento, podendo impactar de maneira drástica o resultado final esperado. Porém o descarte completo dos testes também não era aceitável, uma vez que levaria uma perda muito grande da qualidade do sistema, logo foi decidido manter uma solução intermediária, onde os testes seriam realizados de maneira pontual e manual a cada criação ou alteração de uma funcionalidade do projeto.

4.2 Eletrônica

Para minimizar custos o projeto eletrônico foi desenvolvido para ser o mais compacto possível, logo somente componentes de superfície (SMT) foram utilizados. Sempre que disponível, o encapsulamento 0805 foi adotado, o que possibilitou uma alta densidade de componentes e grandes escalas de miniaturização.

A placa de circuito impresso possui quatro camadas, tanto para que seja

possível conectar todas as malhas internas do circuito de forma compacta, quanto para minimizar a impedância da mesma, a fim de se diminuir possíveis problemas de integridade de sinal e compatibilidade eletromagnética [14].

Nas figuras 21 e 22 abaixo é possível ver o circuito eletrônico dividido em seções. Cada seção numerada provê uma funcionalidade em particular ao sistema, que será explicada nos parágrafos a seguir.

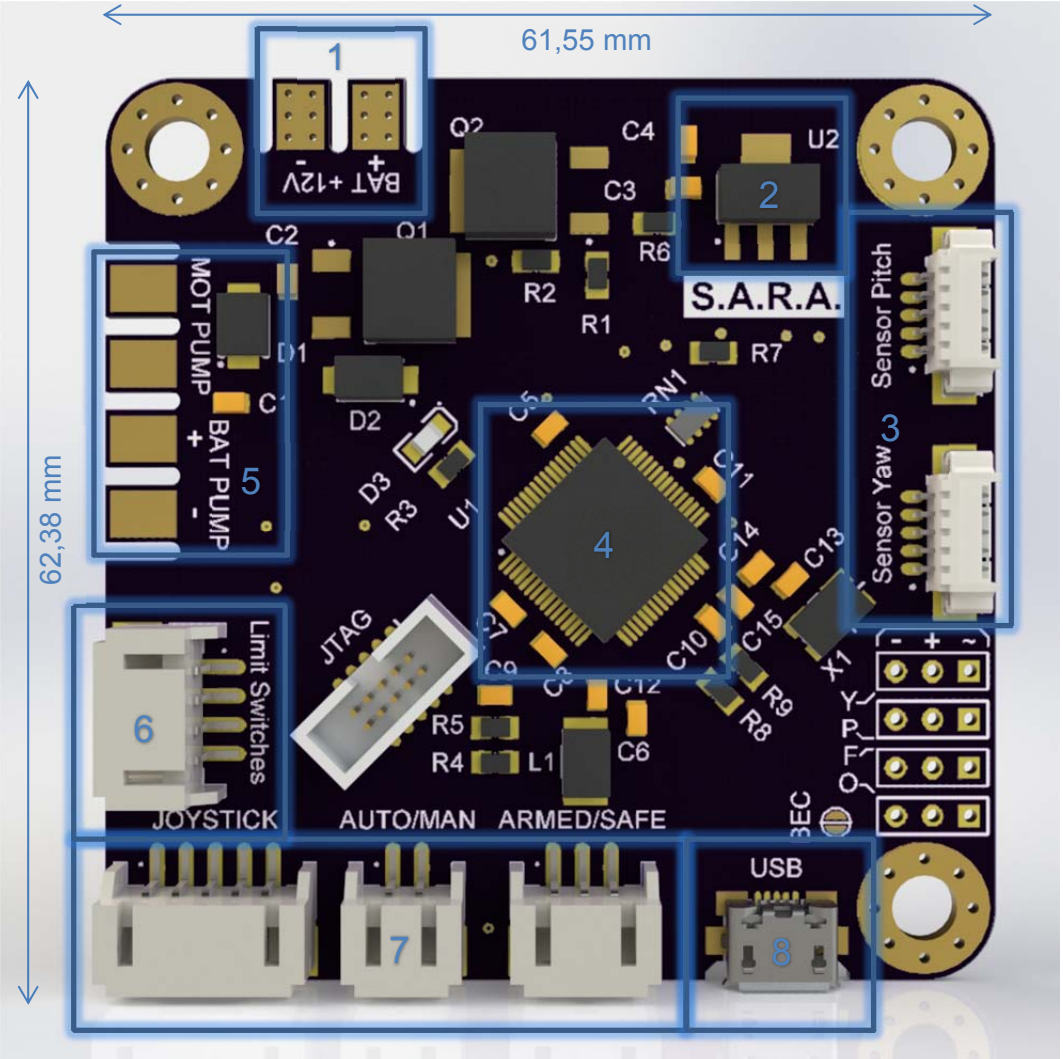

Figura 21: O circuito eletrônico visto de cima

1	Alimentação 12V
2	Conversor LDO 5V
3	Conexão dos Encoders
4	Microcontrolador
5	Disparador
6	Chaves limitadoras
7	Conexão do Painel
8	Micro USB

Tabela 1: Seções do lado superior do circuito

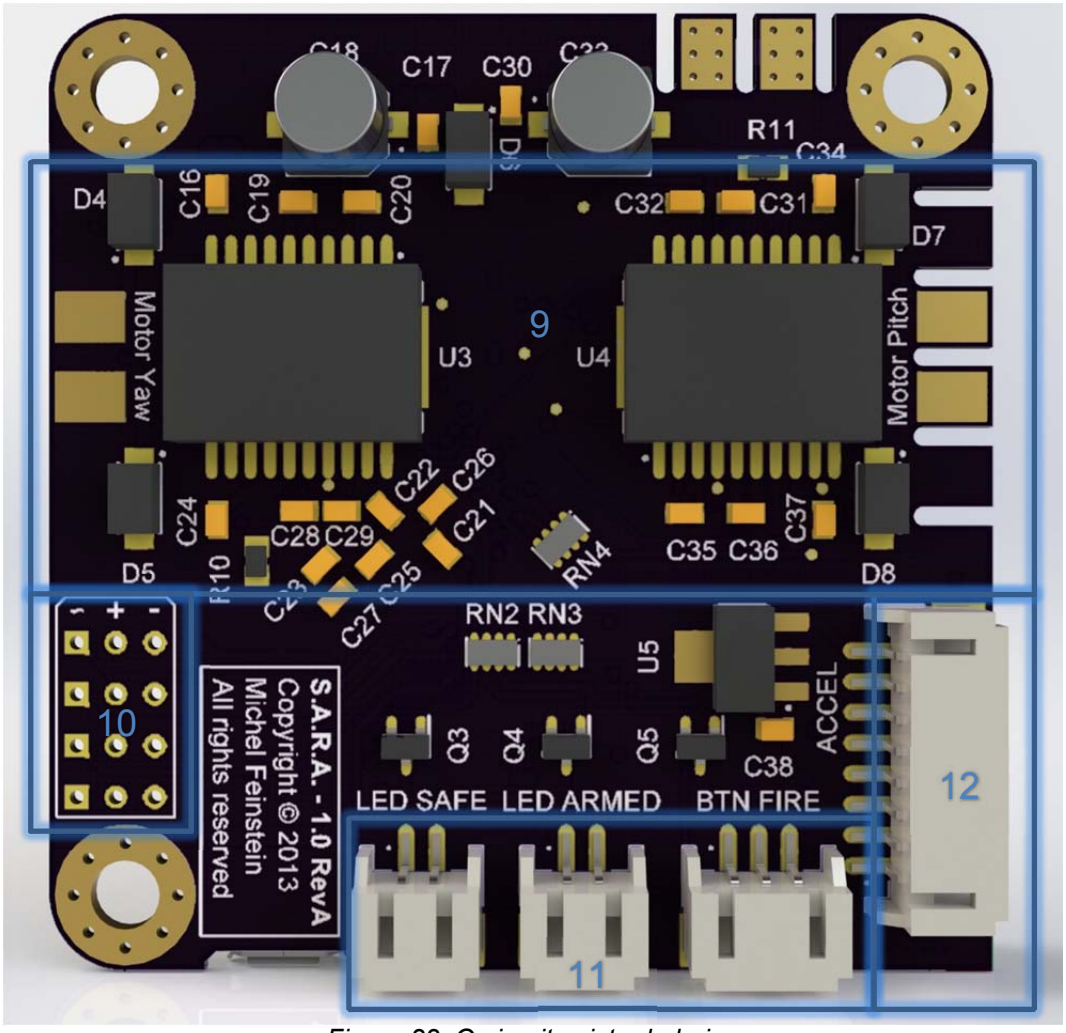

Figura 22: O circuito visto de baixo

9	Chips de Ponte-H
10	Conexão R/C
11	Conexão Painel
12	Acelerômetro

Tabela 2: Seções do lado inferior do circuito

O sistema embarcado desenvolvido consiste de um microcontrolador (4) de 32 bits Atmel AT32UC3C2512C configurado a 48MHz. Esse modelo de microcontrolador foi escolhido por sua grande quantidade de periféricos e pinos, o que possibilita conectá-lo as mais diversas funções, bem como por possuir uma arquitetura RISC modificada de 32 bits com uma excelente relação performance-frequência e consumir pouca energia.

Outro fator que influenciou na escolha deste microcontrolador é o ambiente de programação integrado gratuito, Atmel Studio, baseado na plataforma Visual Studio, que disponibiliza o *framework* ASF, Atmel Software *Framework*, contendo todas as funções necessárias para acessar as funções de baixo nível do

microcontrolador sem ser necessário o acesso a registradores internos, acelerando o tempo de desenvolvimento, bem como permitindo uma rápida adaptação a outras plataformas.

O sistema funciona com uma tensão nominal de 12V (1), e embora ele possa funcionar com tensões abaixo de 12V, esta prática não é recomendada, já que a velocidade dos motores é proporcional à tensão do sistema, assim toda a calibração dos motores teria que ser refeita. A tensão de entrada é convertida a 5V por um regulador linear (2) para alimentar o microcontrolador.

O microcontrolador possui um modulo periférico de comunicação USB que permite uma conexão Full-Speed USB 2.0 de 12Mbits/s. Para que o projeto fique compacto, um conector Micro USB tipo B (8) foi escolhido.

O controle dos motores é feito por meio de um sinal PWM e um sinal digital que indica a direção de giro dos motores, ambos enviados do microcontrolador a um circuito integrado dedicado ao acionamento dos motores. O sinal PWM consiste de uma onda quadrada que permanece em nível alto por uma porcentagem de tempo, denominado Duty Cycle, e permanece em nível baixo pelo restante do período da onda. Na Figura 23 é possível ver três exemplos de PWM para duty cycles de 20%, 50% e 80% onde nível alto é considerado 5V e nível baixo 0V.

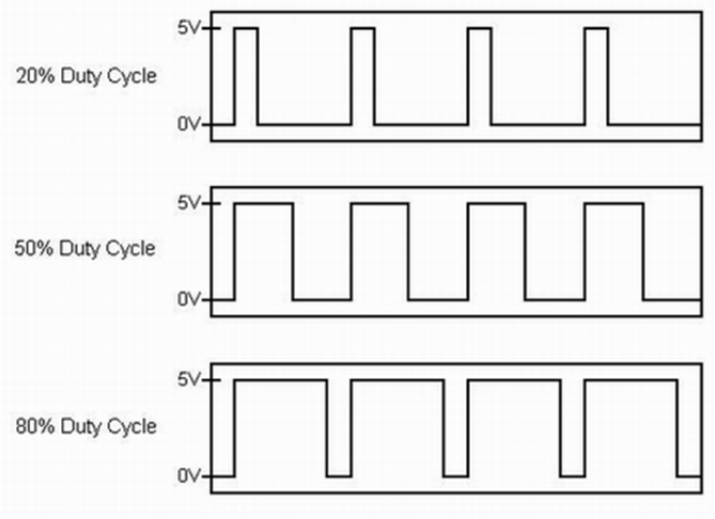

Figura 23: O sinal PWM

A cada nível alto gerado pelo PWM os motores são ligados e a cada nível baixo os motores são desligados, se esse processo se repete em uma frequência relativamente alta (10KHz no caso deste projeto) os motores acabam girando com uma velocidade proporcional ao duty cycle do PWM, sendo 100% a maior rotação possível para o sistema.

Para realizar o chaveamento de potência dos motores o circuito integrado

de Ponte-H (circuito clássico para controle de direção e ativação de motores) L9958 (9) foi escolhido. Esse circuito possui proteções internas que adicionam confiabilidade ao sistema como limitação de corrente, temperatura, mínimo tempo de acionamento, assim prevenindo distúrbios de emissão eletromagnética por parte das indutâncias internas dos motores, e uma interface de comunicação SPI para diagnóstico e configuração interna.

Os motores estão conectados em série com chaves mecânicas limitadoras de curso, de forma a garantir a integridade do sistema caso os eixos venham a ultrapassar os seus limites operacionais. Caso sejam acionadas, as chaves irão mecanicamente desligar os motores das suas respectivas fontes de energia, cabendo ao usuário o retorno manual do robô a uma posição válida, o que acarretará na reconexão automática dos motores ao sistema.

A leitura dos ângulos de *pitch* e *yaw* é realizada por dois encoders absolutos magnéticos com resolução de 12 bits. Com isso uma precisão máxima teórica de aproximadamente 0,088º é possível o que se converteria em um erro de aproximadamente 4,6 milímetros a 3 metros de distância.

Os encoders possuem conexão SPI e dividem o mesmo barramento (3) sendo lidos a cada 120 microssegundos o que garante fluidez ao controle de posicionamento.

O projeto eletrônico foi feito de forma a permitir expansões futuras, sendo assim, ele inclui conexões de rádio controle para aeromodelismo (10), permitindo que a plataforma seja controlada remotamente sem fios, uma conexão para um acelerômetro (12) podendo auxiliar em algum controle de estabilização, conexões diversas para chaves seletoras e luzes para um painel de controle (7, 11) e chaves limitadoras de curso (6) digital que seriam acionadas no caso do robô ultrapassar a posição das mesmas, emitindo um sinal digital ao microcontrolador permitindo uma reação imediata.

O acionamento do disparador é feito por meio de um transistor MOSFET (5) que, acionado pelo microcontrolador, conecta a bateria do disparador no motor do mesmo. Sendo assim, a conexão do MOSFET com o disparador permite que qualquer modelo de disparador seja conectado ao sistema.

4.3 Mecânica

A maior parte da plataforma mecânica foi construída utilizando madeira de MDF pelo seu baixo preço e a possibilidade de ser usinada em uma fresadora CNC ou cortada a laser, entretanto as peças que transmitem torque ao sistema

foram feitas com alumínio. Todo o projeto possui engastes que permitem o alinhamento de todas peças para uma montagem mais precisa.

O sistema de transmissão de torque consiste de motores ligados aos eixos por uma redução com rosca sem fim com a proporção de 96:1, garantindo uma velocidade teórica máxima de 50,875 RPM. A vantagem em se desenvolver o próprio sistema de posicionamento em relação a se comprar servomotores prontos é a maior flexibilidade no controle de posicionamento do robô podendo calibrá-lo com maior precisão para a aplicação final.

O sistema consiste basicamente de duas estruturas móveis interligadas que quando acionadas permitem o movimento independente dos eixos de *pitch* e *yaw* do robô. A estrutura externa realiza o movimento de *yaw* e serve de apoio para o motor de acionamento do eixo de *pitch* bem como o seu encoder, já a estrutura interna realiza o movimento do eixo de *pitch* e transporta o disparador, as câmeras e possui um trilho para a fixação de um laser. Essas estruturas ficam claras nas figuras 24 e 25.

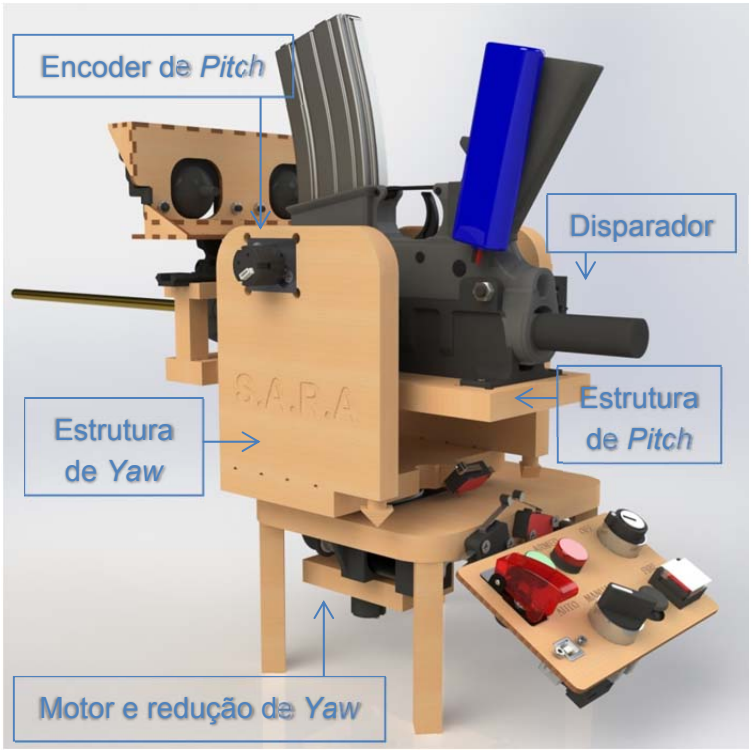

Figura 24: O robô visto de lado

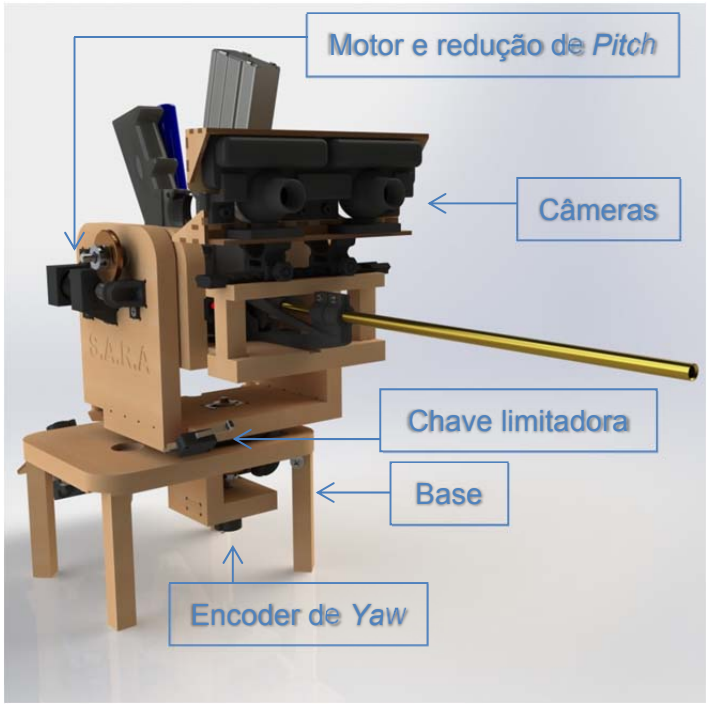

Figura 25: O robô visto de frente

Por limitações de custo e fabricação, o sistema possui liberdade de atuação de 68,2º (-39,02º a +29,18º) no eixo de *yaw* e de 43,94º (-21,44º a +22,5º) no eixo de *pitch*, conforme as figuras 26 e 27, respectivamente.

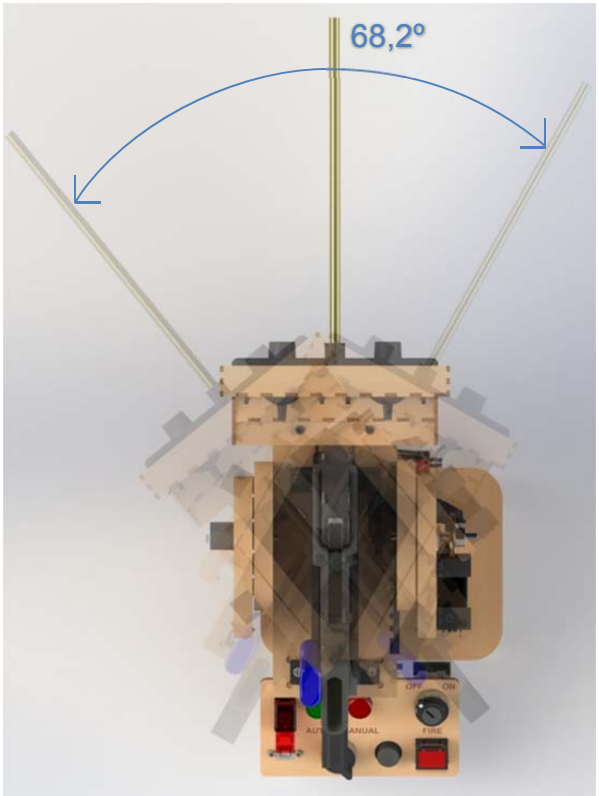

Figura 26: Ângulo máximo do eixo de Yaw

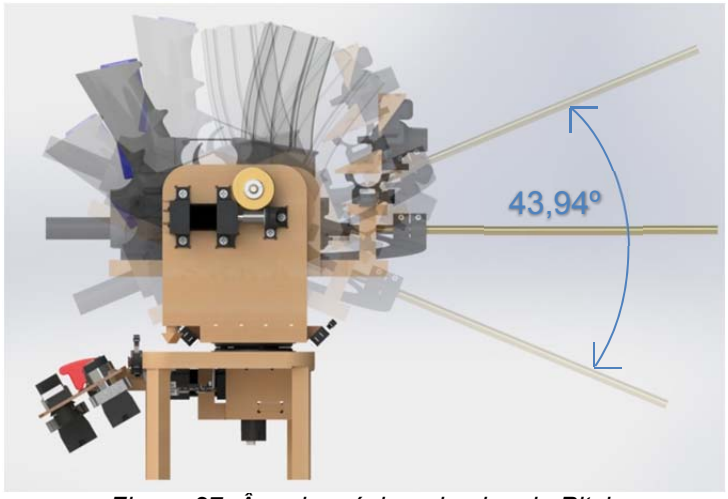

Figura 27: Ângulo máximo do eixo de Pitch

Um painel de controle também foi desenvolvido, embora não implementado, que permita ao usuário o rápido acesso ao sistema, sem depender de um computador e a identificação visual do estado do sistema. O painel projetado pode ser visto na Figura 28, nele há chaves seletoras de modo automático e manual, chave de segurança, chave de ativação geral do sistema, conexão USB tipo B e controles direcionais manuais do sistema.

Figura 28: Painel de Controle

Para permitir testes sem o disparador um contrapeso pode ser adicionado, conforme mostrado na Figura 29, a fim de equilibrar a estrutura em relação ao eixo de *pitch*, sem aumentar o esforço nas suas engrenagens.

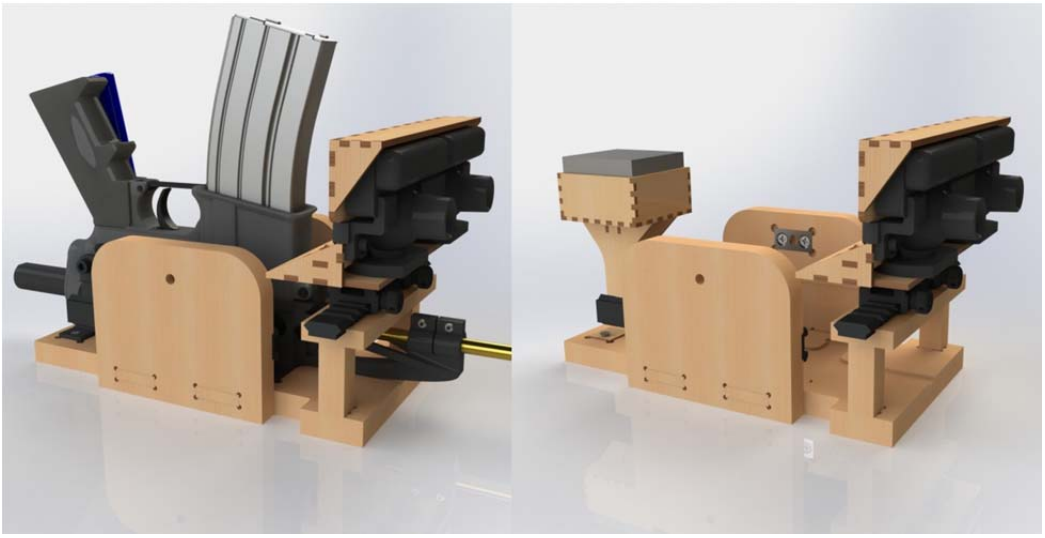

Figura 29: A estrutura do eixo de pitch com o disparador à esquerda e com o contrapeso à direita

5. Resultados

5.1 Computação

Os resultados de ambas as plataformas computacionais foram bastante satisfatórios, os softwares possuem um excelente tempo de resposta e não apresentam erro em seus algoritmos em todos os testes realizados.

Apesar da metodologia de testes adotada não ser ideal, ela foi capaz de detectar diversos erros durante a fase de desenvolvimento e manter a qualidade final do sistema elevada.

5.1.1 Sistema Embarcado

O sistema embarcado é capaz de processar o controlador PID de maneira a posicionar os eixos com a precisão total do sistema de aproximadamente 0,088º. Embora a massa do sistema interfira na resposta mecânica à mudança de posição, o algoritmo foi testado com a massa do robô reduzida e a precisão total foi alcançada em menos de um segundo.

Conforme mencionado anteriormente, o sistema envia um novo pacote de informações a cada 50 milissegundos, sendo assim o tempo entre envio de pacotes foi medido para garantir que não há latência ou atraso no processamento e foi constatado um tempo fixo de 50 milissegundos entre cada pacote de dados enviado.

5.1.2 Plataforma de controle

O software de controle apresenta uma interface fluida ao usuário onde nenhum atraso de resposta é perceptível, graças a sua estrutura *multithread*.

Todos os comandos são enviados e recebidos do circuito eletrônico de forma que nenhum atraso de processamento foi notado. As câmeras foram configuradas a 60 fps e todo o tratamento e processamento de imagens é exibido em tempo real, sem que seja perceptível nenhum atraso nas imagens.

Os comandos manuais de posicionamento da interface do usuário apresentaram o funcionamento esperado, permitindo posicionar o robô de maneira rápida e precisa.

A interface somente disponibiliza os comandos de controle do robô quando o circuito eletrônico encontra-se conectado ao computador, mostrando uma clara mensagem ao usuário caso ele esteja desconectado. O número de câmeras conectadas também é informado em uma barra de status, junto com o estado de conexão do circuito.

Todos os gráficos presentes na aplicação foram capazes de refletir a posição do robô, ao mesmo tempo em que os gráficos de posição *yaw* versus p*itch* mostram também os limites operacionais do sistema, de maneira implícita pelos limites dos eixos dos gráficos.

Apesar de não ter sido completamente desenvolvido, o controle autônomo de posicionamento pode ser facilmente alterado já que todas as partes necessárias para o seu funcionamento já foram desenvolvidas. O algoritmo implementado funciona de maneira satisfatória, apresentando um erro mínimo e imperceptível para a aplicação final.

De uma forma geral a interface de controle do robô possui um ótimo desempenho e robustez, encapsulando todas as suas funcionalidades, permitindo uma rápida manutenção e extensão do projeto.

5.2 Eletrônica

O sistema eletrônico apresentou uma ótima estabilidade, sem apresentar nenhuma falha durante o seu tempo de operação. Todos os circuitos integrados, bem como os circuitos passivos especificados operaram conforme projetado.

A velocidade de processamento e aquisição de dados atendeu ao desejado originalmente garantindo um controle e processamento fluídos.

5.3 Mecânica

O projeto mecânico se mostrou muito satisfatório, apresentando uma ótima precisão e baixo atrito, atendendo aos requisitos da aplicação final, entretanto algumas melhorias devem ser feitas para que o sistema funcione de maneira a garantir a sua duração de longo prazo.

Pelo centro de gravidade do robô ser muito elevado em relação a sua base, o sistema deve estar muito bem preso ao chão para que ele consiga se movimentar com segurança, caso contrário a base e o topo ambos giram, fazendo com que o sistema vibre e se locomova.

Devido a problemas com a importação de certos componentes, o sistema não pode ser testado com o disparador, ficando somente configurado com o contrapeso em seu lugar.

6. Conclusão

Apesar de ser um projeto desafiador, envolvendo diversas áreas da engenharia, muitas vezes desconexas, os resultados obtidos foram muito satisfatórios. A plataforma permite não somente atender ao seu propósito inicial, como também ser facilmente estendida a novas aplicações.

Por se tratar de um desenvolvimento de um produto, diversas técnicas de fabricação foram empregadas, o que acabou acarretando em problemas de logística, impedindo que o sistema fosse completado a tempo, com todas as suas especificações originais.

Entretanto, todas as partes desenvolvidas possuem um excelente resultado prático. A estrutura mecânica se mostrou bastante robusta, o circuito eletrônico está permitindo um controle de alta performance e o software de gerenciamento do sistema permite um controle rápido e seguro, bem como a fácil expansão e manutenção das suas funcionalidades. Uma demonstração do sistema pode ser encontrado na internet [15].

Para que se obtenha uma plataforma mais completa, é recomendado o aprimoramento do algoritmo de controle autônomo do sistema e a possível modificação da estrutura de madeira para alumínio, a fim de se prolongar a vida útil da plataforma robótica.

7. Referências

[1] DEFCON, “DEFCONBOTS,” [Online]. Available: http://defconbots.org/. [Acesso em 15 Dezembro 2013].

[2] RoboGames, “Robot Shooting Gallery,” [Online]. Available: http://robogames.net/rules/shooting-gallery.php. [Acesso em 15 Dezembro 2013].

[3] R. O. Duda and P. E. Hart, "Use of the Hough Transformation To Detect Lines and Curves in Pictures," *Communications of the Association for Computer Machinery,* no. 15, pp. 11-15, 1972.

[4] P. V. C. Hough, "Machine Analysis Of Bubble Chamber Pictures," in *Proceedings of the International Conference on High Energy Accelerators and Instrumentation*, 1959.

[5] C. Kimme, D. Ballard and J. Sklansky, "Finding circles by an array of accumulators," *Communications of the Association for Computing Machinery,* vol. 18, pp. 120-122, 1975.

[6] K. Ogata, Modern Control Engineering, Prentice Hall, 2010.

[7] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," *Transactions of the American Society of Mechanical Engineers,* no. 64, pp. 759-768, 1942.

[8] Qt Project, “Signals & Slots,” Qt Project, [Online]. Available: http://qt-project.org/doc/qt-4.8/signalsandslots.html. [Acesso em 15 Dezembro 2013].

[9] OpenCV, “OpenCV Documentation,” [Online]. Available: http://docs.opencv.org/. [Acesso em 15 Dezembro 2013].

[10] S. Suzuki and K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following," in *Computer Vision, Graphics, and Image Processing*, 1985.

[11] J. Stewart, Cálculo, 5ª ed., vol. 2, Cengage Learning, 2006.

[12] G. Bradski e A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, 1ª ed., O'Reilly Media, 2008.

[13] S. Lin e B. W. Kernighan, *An Effective Heuristic Algorithm for the Traveling-Salesman Problem,* vol. 21, Operations Research, 1973, pp. 498-516.

[14] E. Bogatin, Signal and Power Integrity - Simplified, 2nd ed., Prentice Hall, 2010.

[15] “YouTube,” [Online]. Available: http://www.youtube.com/user/michelfeinstein. [Acesso em 15 Dezembro 2013].