Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

December 17, 2018

SISTEMA DE POSICIONAMENTO INDOOR POR VISÃO COMPUTACIONAL, CONSTRUÇÃO E CONTROLE DE ROBÔ CAPAZ DE IDENTIFICAÇÃO E CONDUÇÃO DE ALVO ATÉ DETERMINADO DESTINO

Nícolas Mocaiber Cardoso Cury

PUC RIO

Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMA DE POSICIONAMENTO INDOOR POR VISÃO COMPUTACIONAL, CONSTRUÇÃO E CONTROLE DE ROBÔ CAPAZ DE IDENTIFICAÇÃO E CONDUÇÃO DE ALVO ATÉ DETERMINADO DESTINO

Aluno(s): **Nícolas Mocaiber Cardoso Cury**

Orientador(es): **Wouter Caarls**

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

Agradecimentos

Agradeço aos meus pais por esta conquista e por sempre estarem disponíveis para todas as dificuldades que enfrentei na vida. Nada disso seria possível sem vocês.

Ao meu irmão pelas referências e companheirismo que construíram minha personalidade e meu caráter.

Aos meus amigos, que aguentaram meu péssimo humor nessa reta final de formatura. A paciência de vocês é admirável.

À todos meus companheiros de trabalho, pela compreensão nas dificuldades que enfrentei no término de minha formatura e disponibilidade em me auxiliar para conclusão deste projeto.

Ao engenheiro Luiz Felipe Costa dos Santos pelo auxílio na elaboração deste projeto.

Ao professor e orientados Wouter Caarls, por me dar a oportunidade de executar este projeto, pelo aprendizado que foi fundamental nas disciplinas no final do meu curso, e pela paciência.

Resumo

O trabalho abordou o desenvolvimento de um algoritmo para um sistema de posicionamento *indoor* (IPS) para múltiplos alvos por visão computacional, utilizando o software *LabView* em conjunto com o módulo de visão *Imaq Vision*. Cada alvo a ser identificado dispõe de uma referência em um padrão específico de forma que possa ser atribuído um *ID* único a este objeto.

Como forma de validação do sistema, o mesmo será implementado em um controlador PID aplicado em um robô capaz de identificar um alvo e conduzi-lo até um determinado destino, tendo sido feito a especificação dos componentes e construção do robô.

Uma proposta para atender estes requisitos, que será desenvolvida ao longo do projeto, é a elaboração de um robô capaz de jogar uma partida de hóquei de forma autônoma utilizando o sistema de posicionamento *indoor* desenvolvido.

Palavras-chave: IPS, Sistema de Posicionamento Indoor, Robô Autonômo, Controlador PID

INDOOR POSITION SYSTEM BASED ON COMPUTER VISION, CONSTRUCTION AND CONTROL OF ROBOT CAPABLE OF IDENTIFY AND MOVE TARGET TO SPECIFIC DESTINATION

Abstract

This work approached the development of an indoor position system algorithm for multiple targets based on computer vision, using LabView software in conjunction with Imaq Vision module. Each target to be identified has an individual reference in a specific pattern, in a way that can be attributed an unique ID to this object.

As a form of the system's validation, it will be implemented on a PID controller applied in a robot able of identify a target and move it to a specific destination, having been made the specification of the robot's components and its construction.

A proposal to attend these requirements, which will be developed and implemented along this project, is the elaboration of a robot able to play a hockey match autonomously, using the indoor position system developed.

Keywords: IPS, Indoor Position System, Autonomous Robot, PID Controller

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

a Considerações iniciais

Trabalho com desenvolvimento de drones e sistemas robóticos, onde frequentemente nossas aplicações são em ambientes *indoor*. Há tempo nos deparamos com o obstáculo de rastrear com precisão objetos nestes ambientes para implementação de telemetria ou autonomia dos projetos.

Com incentivo da empresa onde estagio, desenvolvi para meu projeto de conclusão de curso, tanto no estágio como em meu tempo livre, e em conjunto com o engenheiro Luiz Felipe Costa dos Santos, o sistema de posicionamento *indoor* utilizado neste projeto. Os equipamentos utilizados no projeto também foram disponibilizados pela empresa.

b Contextualização

Hoje, muitas aplicações cotidianas valem da utilização do GPS (*global position system*), seja para identificação do posicionamento do usuário, do objeto em que se deseja rastrear, execução de mapeamento e topografia, ou assistência à navegação.

Trata-se de um sistema de navegação de posicionamento por satélites em que é comparado a diferença de tempo entre o envio de um sinal do satélite e o recebimento em um receptor, sendo calculada a distância entre ambos. Como a posição dos satélites são conhecidas, é possível calcular o posicionamento do alvo de acordo com o triangulamento desses sinais recebidos por mais de um satélite. Dependendo da robustez do sistema, sua precisão pode variar de dezenas de metros à frações de centímetros. [7]

Figura 1: Triangulamento de sinal de GPS

Este recurso é uma das principais fontes de informação para o processamento de um sistema de controle para um veículo autônomo, o qual possui sua base do cálculo através da leitura de diversos sensores, sendo as coordenadas de posicionamento um dos mais relevantes para a eficácia de um deslocamento preciso e seguro.

Apesar da precisão ao utilizar este recurso e a facilidade em obter um sensor para este propósito, quando se trata de um ambiente fechado sua utilização não se mostra tão eficiente. Com a obstrução física para recebimento do sinal vindo dos satélites, impossibilita-se sua leitura para implementação em localizações como fábricas ou no interior de construções. Nestas situações, ocorre reflexão ou deslocamento do sinal emitido pelo satélite, sendo influenciando na leitura do relógio do dispositivo receptor, impossibilitando a confiabilidade de sua leitura.

É necessária uma aplicação com resultados semelhantes, mas sem a obrigatoriedade do cálculo da leitura pelo recebimento da informação através de um satélite.

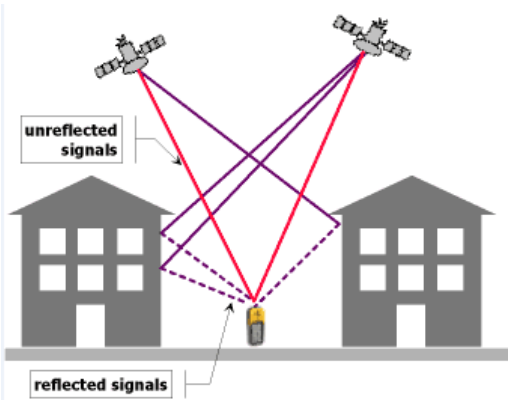

Figura 2: Obstrução e reflexão do sinal do satélite

c Motivação

Com base nesta impossibilidade de utilização do GPS em determinados âmbitos, foi desenvolvido para este projeto um sistema de posicionamento indoor (IPS) para múltiplos alvos, com a finalidade de reconhecer as coordenadas de um robô e de um determinado objeto, implementar o sistema de controle para que o robô se desloque até o objeto e o conduza até o destino desejado, tendo uma lógica para quando está no modo de "busca" e "posse" deste objeto.

Um IPS trata-se de um sistema de navegação em espaço fechado, onde em muitos casos o sinal de GPS não é suficiente para realizar a leitura das coordenadas do alvo a ser rastreado.

Com isso, busca-se o estudo prático na área de visão computacional para aplicação robótica em sistemas de controle em ambientes restritos utilizando câmeras, com utilidade no mercado para desenvolvimento de fabricas autônomas ou localização de pessoas dentro de construções fechadas.

d Objetivo

Com o objetivo de colocar em prática os tópicos mencionados, é proposto para este projeto, utilizando o sistema descrito, a elaboração de um robô capaz de jogar uma partida de hóquei de forma autônoma.

O sistema identifica a posição e angulação da direção em que o robô se encontra juntamente com a posição do disco de acordo com um referêncial e executa um algoritmo de controle, de forma não embarcada, para a atuação do robô até a posição do alvo. Encontrando-se no seu destino, será efetuado uma rotina de verificação para saber se é possível ser considerado o disco sob posse do robô, que será executada continuamente. Caso se encontre neste estado, o disco será conduzido até uma área definida como gol. Durante este processo, caso a rotina de verificação de posse acuse que o disco não se encontra mais sendo conduzido, o sistema retorna ao estado de deslocamento e controle da angulação em relação à posição do disco. Após o gol, o robô retorna à posição de início e aguarda até o disco ser colocado no centro da quadra para reiniciar o processo.

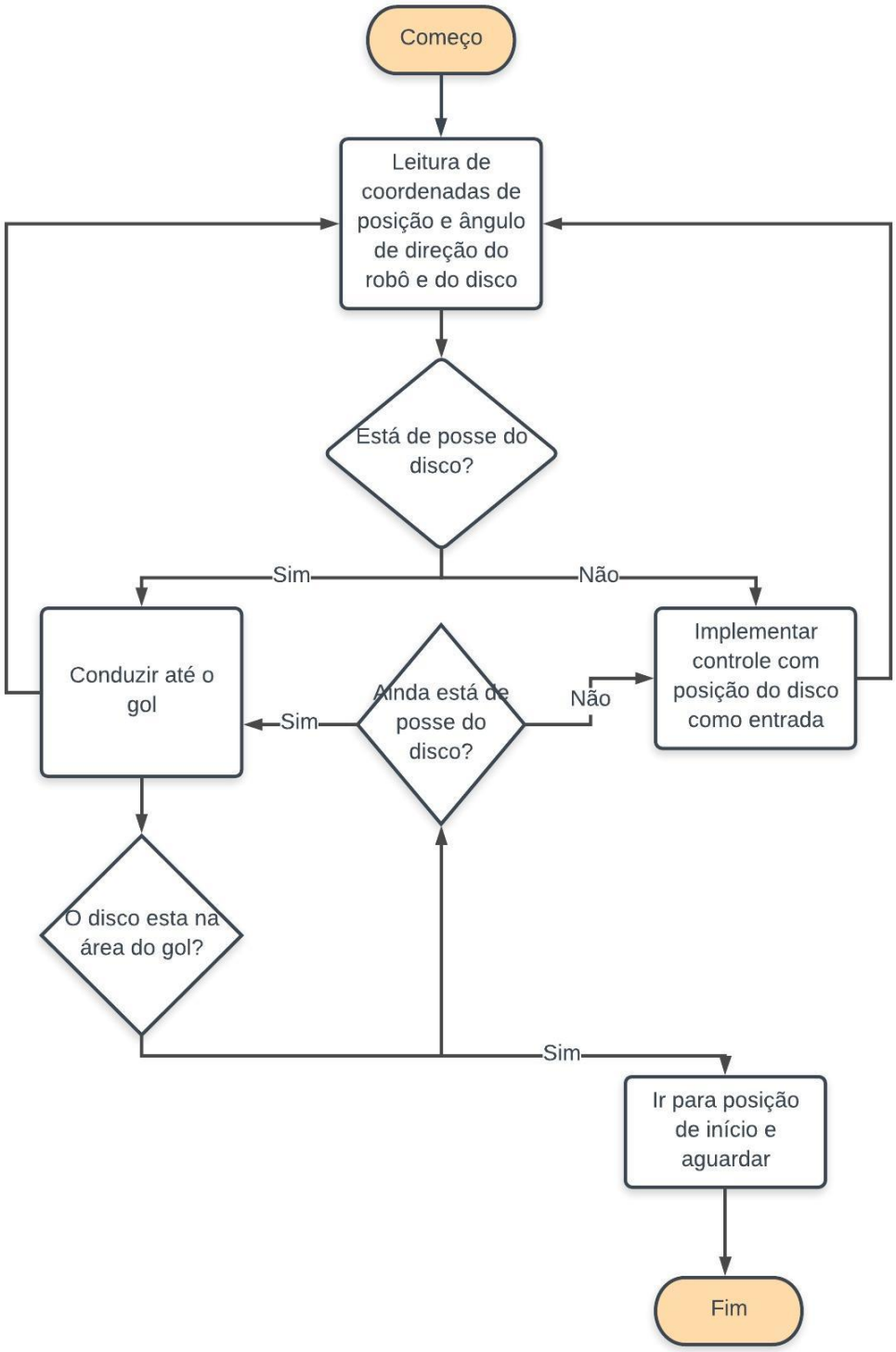

Figura 3: Fluxograma do sistema

2 Revisão Bibliográfica

a Sinais

Entende-se como sinal a representação através de funções de uma ou múltiplas variáveis descrevendo o comportamento de algum fenômeno de acordo com determinados sinais de entrada, que respectivamente alteram a resposta do sistema, gerando novos sinais de saída. Usualmente os casos mais estudados no âmbito da elétrica são as abordagens de sinais para descrever comportamentos de diferenças de potenciais e correntes de um sistema, sendo trabalhados nos domínios continuo e discreto, dependendo da natureza do sistema. [8]

b Filtros

Trata-se de um procedimento para processamento de sinal com o objetivo de analisar sua saída sob ponto de vista da frequência ou do tempo, alterando as amplitudes relativas dos componentes e atenuando comportamentos indesejados a partir de um sinal de entrada, como ruídos na leitura de um sensor. Tendo como contrapartida um atraso na saída do sinal em relação à sua entrada devido ao tempo de processamento deste sinal, característica presente tanto nos filtros analógicos como digitais, sendo imprescindível uma análise sobre a relação de eficiência entre este atraso e o resultado do processamento a ser executado.

Para a compreensão deste trabalho é expressivo o entendimento do comportamento e utilização de filtros digitais.

Utilizando um processador, que pode ser um computador ou um microcontrolador específico para processamento digital de sinais (*digital signal processor*), são aplicadas operações matemáticas no domínio do tempo discreto. Geralmente consiste de um conversor analógico-digital para comutar o domínio da amostra de contínuo para discreto. Com o dado convertido, as operações são aplicadas de acordo com o filtro de interesse e o tipo de atenuação desejada em sua saída, tendo este sinal novamente convertido desta vez por um conversor digital-analógico.

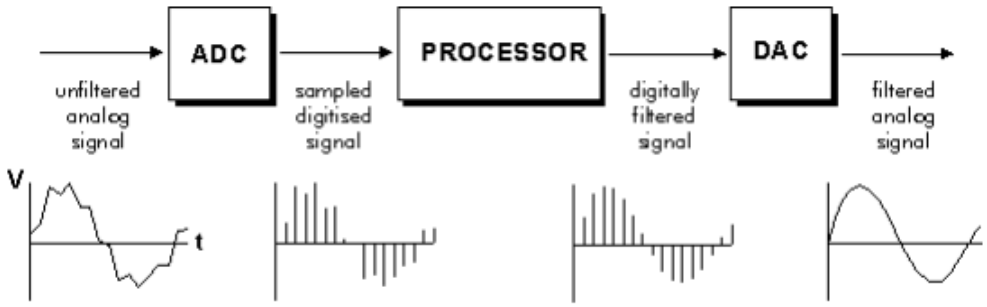

Figura 4: Comportamento do sinal ao ser processado por um conversor analógico-digital, filtrado e processado de volta por um conversor digital-analógico [1]

Este tipo de filtro tem nível de praticidade e versatilidade maior em relação aos filtros analógicos por serem programáveis, não necessitando a elaboração de um determinado circuito para cada comportamento desejado, sendo facilmente testados e implementados e possuindo melhor resposta a sinais de baixa frequência em relação aos filtros analógicos. [1]

c Modulação de Sinal

1 PWM

Refere-se a *Pulse Width Modulation*, sendo uma técnica utilizada para transmissão de dados de acordo com a variação de largura de um pulso.

Com a informação a respeito da frequência do sinal, são transmitidos pulsos com intervalo contínuo, sendo medido o tempo entre o envio de cada pulso e seu retorno ao nível baixo. De posse destes tempos, é possível determinar a proporção da onda que foi enviada durante este período fixo, sendo essa proporção a largura do sinal.

Atribuindo-se faixas de valores referentes largura máxima do pulso e a largura mínima, é possível o envio de dados utilizando esta modulação.

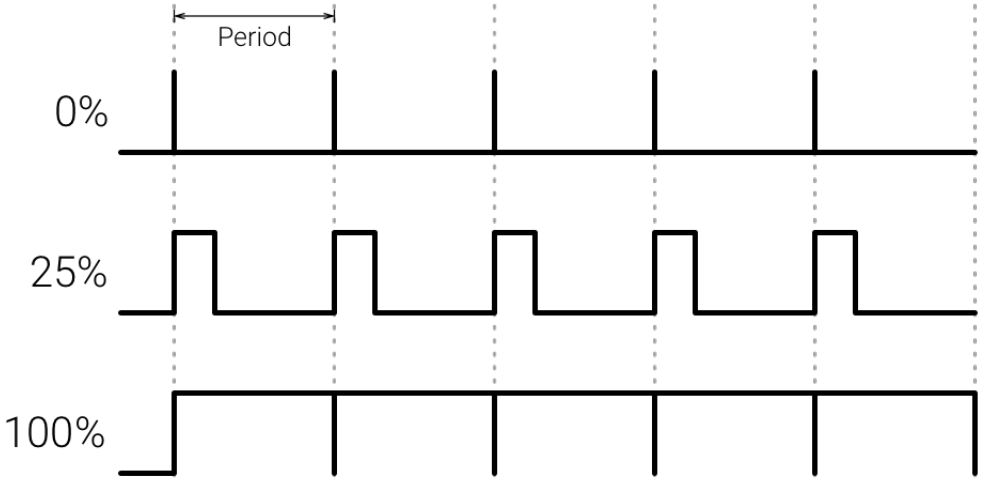

Figura 5: Exemplo de modulação em *PWM*

2 PPM

Refere-se à *pulse position modulation*, transmitindo dados de acordo com a posição do pulso em relação ao período de amostragem.

Diferentemente do *PWM*, a largura do pulso será fixa, sendo registrado o tempo em que o sinal é enviado. Sua posição em relação a frequência do sinal determinará o dado que será transmitido.

A principal vantagem em relação ao *PWM* é a capacidade de enviar múltiplos canais através de uma única saída de modulação [2]. Determinando-se quantos dados serão transmitidos, atribui-se a posição de cada pulso como um dado individual referente a um valor proporcional a sua posição.

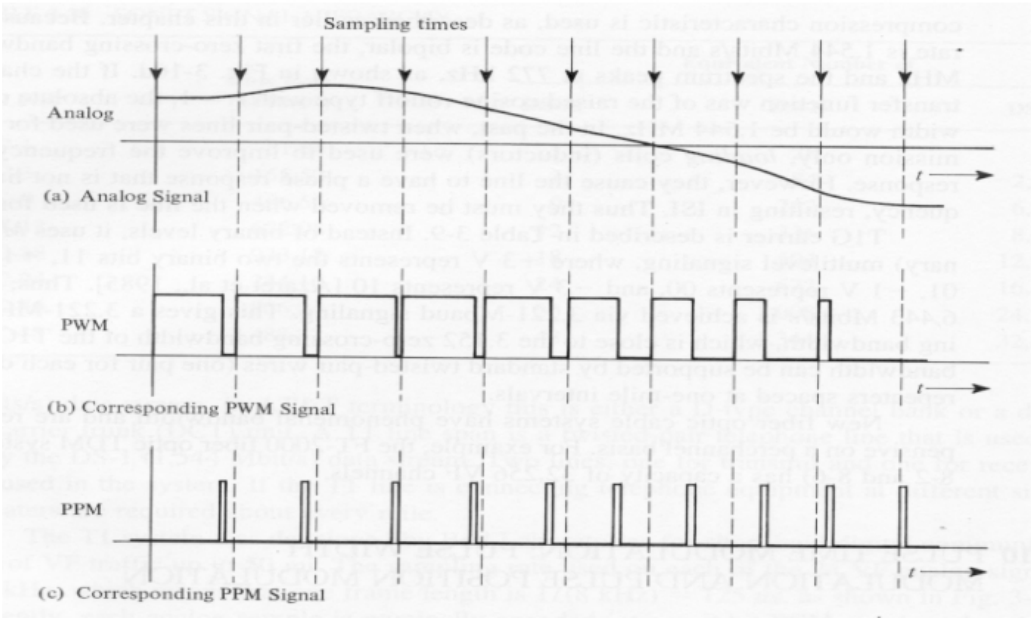

Figura 6: Exemplo de sinal sendo representado em *PWM* e *PPM* [2]

d Comunicação serial

Trata-se do processo de envio de um *bit* de cada vez de forma sequencial através de um canal de comunicação ou barramento, sendo o método de transferência de dados mais utilizados entre equipamentos e periféricos.

Na comunicação serial, os dados são representados na forma de pulsos binários, onde as portas se comunicam através do envio de sinais de nível lógico alto, por exemplo 5V, ou nível lógico baixo.

Estas conexões se dão por meio de duas conexões, ligando saídas transmissores e receptoras de ambos os equipamentos, com um terra em comum para referenciamento do nível lógico dos sinais [9].

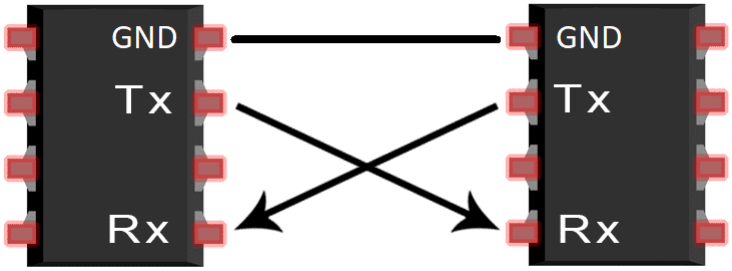

Figura 7: Conexões de dois hardwares por comunicação serial

e Controle PID

Trata-se de um sistema de controle em malha fechada que, ao utilizar componentes proporcionais, integrativos e derivativos, busca corrigir uma variável controlada baseado num *setpoint*, que é o valor de referência que se deseja alcançar, calculando continuamente o erro do sistema através da diferença entre o valor lido pelo sensor em comparação com este *setpoint*, utilizando componentes proporcionais, integrativos e derivativos.

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

Figura 8: Equação do controlador PID baseado nos ganho proporcional, integrativo e derivativo

1 Componente proporcional Kp

O fator proporcional resulta na multiplicação do erro medido por um valor de ganho proporcional *Kp*, fazendo com que sua proporção na atuação seja maior conforme aumenta-se o erro ou o ganho. Um valor excessivamente alto faz com que o valor lido pelos sensores ultrapasse o *setpoint*, ocasionando *overshoot*, que se trata da parte que excede o valor desejado, retardando a estabilização do sistema. Caso o novo erro, após *overshoot*, seja maior do que o erro medido anteriormente, o sistema apresentará comportamento instável.

O problema de se utilizar um controlador somente com componente proporcional é de que não será possível eliminar o erro estacionário, que é a diferença entre a saída lida após o sistema assentar e o valor de *setpoint* desejado.

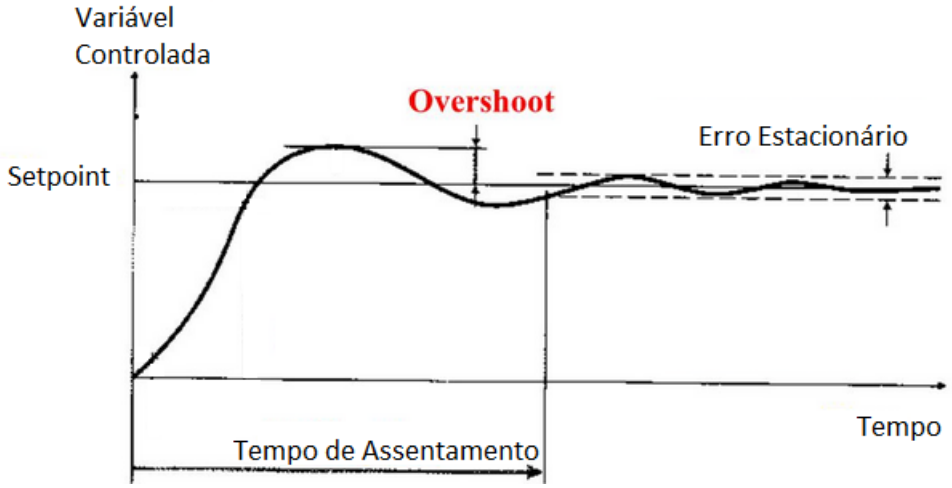

Figura 9: Exemplo de comportamento de um controlador PID, explicitando *overshoot*, tempo de assentamento e erro estacionário

2 Componente integrativo Ki

De forma a se eliminar o erro estacionário, é utilizado um componente integrativo que visa acumular o erro medido com o tempo, sendo que este acumulo pode ser reduzido ou até negativado em caso de erro negativo. O ponto negativo é que este componente tende a aumentar o *overshoot* a medida que a saída se aproxima do *setpoint*.

Outro problema comum é o chamado *wind-up*, que ocorre quando o controlador alcança o limite de atuação ou quando o atuador está saturado, problema recorrente neste projeto. Nesta situação, o controlador tenta eliminar o erro estacionário, mas não oferece saída suficiente para haver atuação, o que ocorre é um acumulo indefinido do erro, o que ocasionará em um *overshoot* excessivamente desproporcional caso o sistema consiga atuar, causando grande oscilação no sistema. Uma forma simples de contornar este problema é colocar um limite de saturação para este acumulo de erro [10].

3 Componente derivativo Kd

Atua comparando o erro atual com o medido anteriormente, atuando na diferenciação da atuação. Este componente tem o comportamento de amortecer o sistema, minimizando as oscilações.

Deve-se ter atenção à sua aplicação em sistemas ruidosos, visto que a presença de interferências tenderá a uma maior proporção deste componente. Tem melhor atuação após filtragem do sinal, atenuando os ruídos.

3 Projeto do robô e especificação de componentes utilizados

A inspiração para este projeto veio dos robôs de combate e sumô. Foram escolhidos componentes que priorizassem velocidade em detrimento ao torque, já que não haveria a situação de um robô ter que empurrar o outro como no sumô, assim como resistência não foi um fator determinante como nos casos de combate.

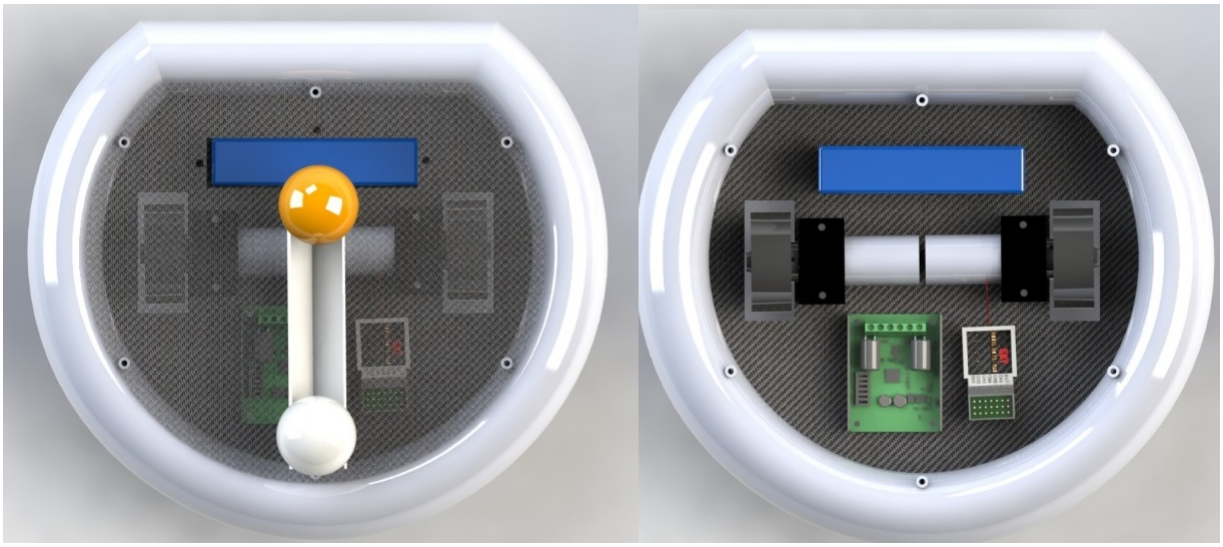

Figura 10: Modelo em 3d feito no SolidWorks do robô com e sem tampa

As duas esferas na tampa do robô atuam como um material difusor para LEDS identificadores, de forma que o sistema de posicionamento reconheça e atribua uma identificação individual para cada combinação de cores utilizadas.

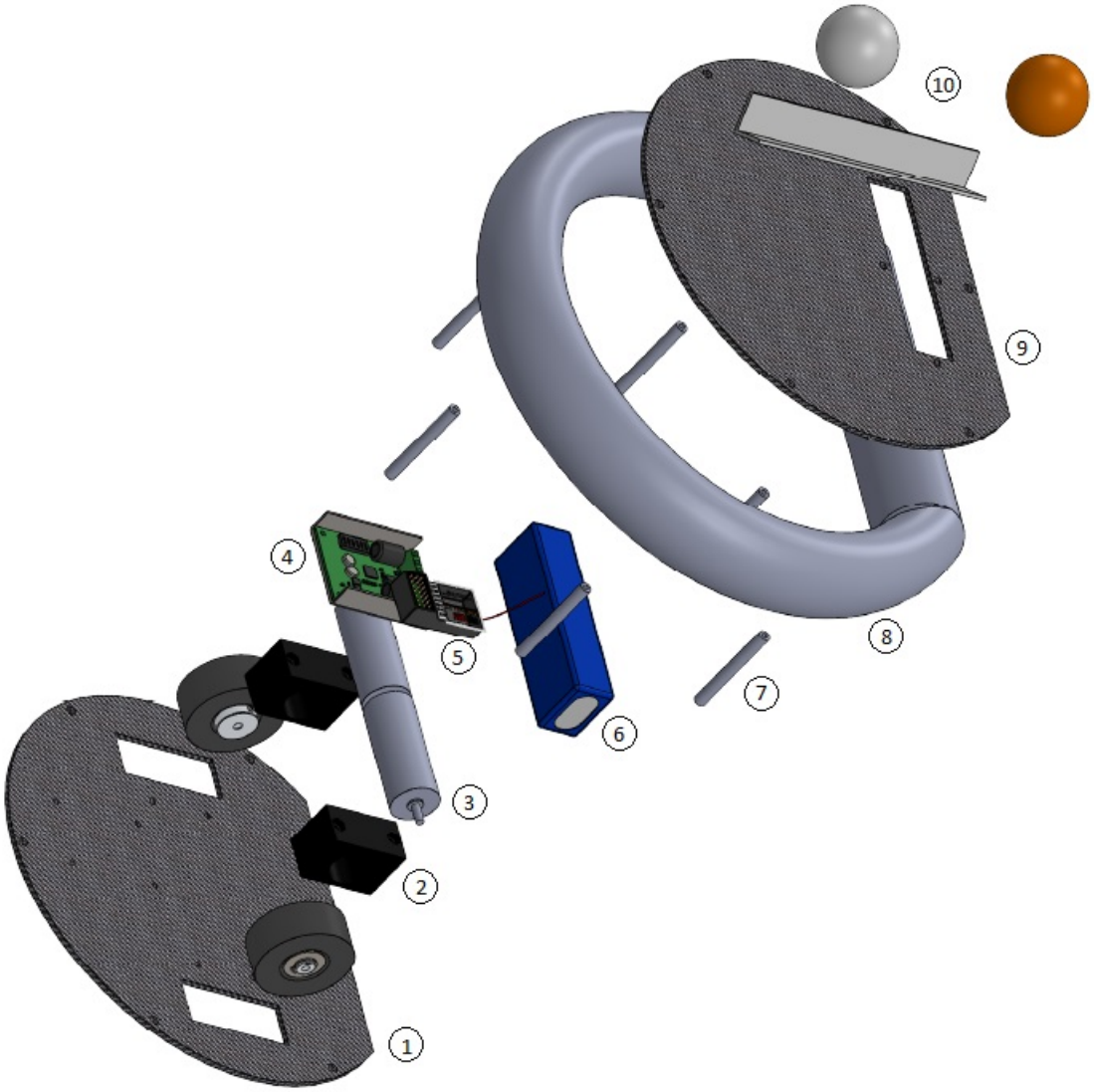

Figura 11: Vista explodida dos componentes do robô

Nº do Item	Descrição	Qtd.
1	Base	1
2	Berço do motor em PLA	2
3	Motor DC 12V 1000RPM	2
4	SaberTooth 2x12A V1.00	1
5	Radio Receiver	1
6	Bateria LiPo 12V	1
7	Espaçador	6
8	Espuma Protetora	1
9	Tampa	1
10	Material Difusor com LEDS	2

Tabela 1: Lista de componentes

a Controladora dos Motores – SaberTooth 2x12A V.100

Para maior precisão e garantir a saída desejada dos motores escovados, principalmente por conta do sistema de controle implementado, optou-se por utilizar uma controladora comercial encontrada no mercado ao invés de um circuito amplificador de tensão.

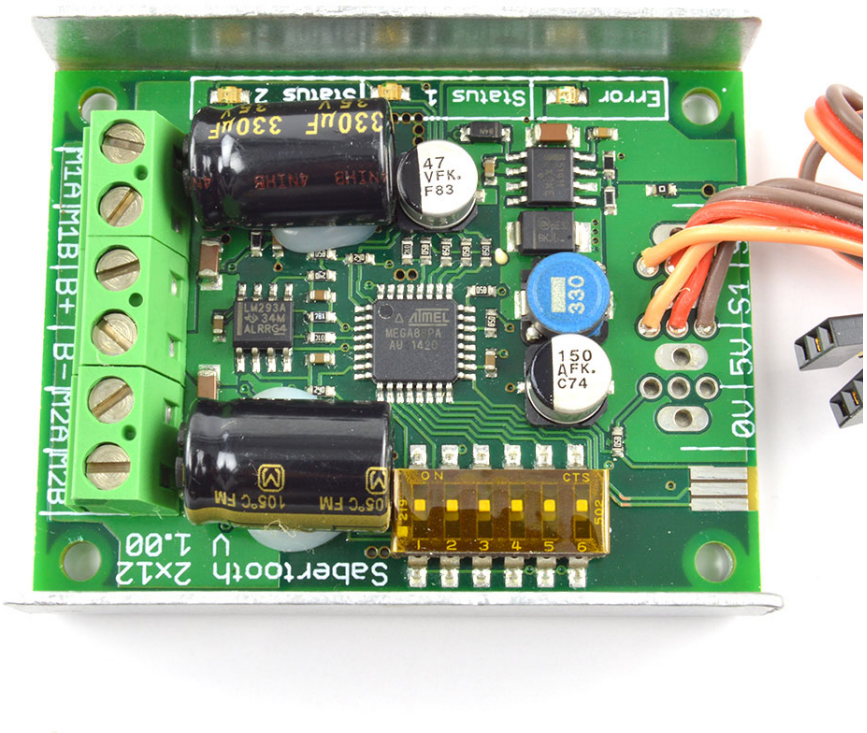

Figura 12: Controladora Sabertooth

A controladora SaberTooth opera dois motores DC escovados simultaneamente de até 12A cada, suportando uma corrente de pico de 25A por alguns segundos, estando incluídos proteção térmica e contra excesso de corrente, garantindo a integridade do circuito caso os motores se encontrem em *stall*. Um dos principais fatores considerados nesta escolha foi a versatilidade de controlar ambos os motores tanto por um sinal analógico, comunicação serial, ou rádio frequência. [11]

A saída do sistema de controle ligada à um microcontrolador enviou um sinal de saída no formato *PWM* para o rádio transmissor, o qual enviou o comando ao sabertooth através do radio receptor, permitindo um controle a distância do robô.

Tensão de entrada	6 – 24V
Corrente	12A por canal
Corrente de pico	25A por canal por alguns segundos
Peso	62,37g
Dimensões	59 x 75 x 17mm

Tabela 2: Especificações do *SaberTooth*

A controladora também possui um módulo de *switches* capazes de configurar seu modo de operação, com funções como controle independente de cada motor, modo exponencial e controle de segurança para baterias *lithium* [11].

- Switch 1 – Determina o modo mixado, responsável por controlar de forma independente cada motor. Este modo ficará acionado, de forma que o canal 1 controlará ambos os motores quando for acionado o comando para o robô se deslocar para frente ou para trás. O canal 2 fará o controle diferencial destes motores, responsável pelo movimento de giro. Caso este switch não fosse acionado seria permitido um controle independente de cada motor.
- Switch 2 – Habilita ou desabilita o controle exponencial embutido na própria controladora. Este modo será desabilitado visto que o sistema implementará um de controle próprio.
- Switch 3 – *Lithium Cutoff*. Nesta configuração, a alimentação será cortada caso alguma célula da bateria tenha menos de 3V de tensão, prevenindo dano à mesma. Como será utilizada uma bateria *LiPo*, este modo se encontra ativado.
- Switch 4 – Atribui a aceleração um formato de rampa. Assim como o controle exponencial, não é interessante a aplicação para este projeto visto que quanto menor interferência na entrada do sistema, melhor para o controle.
- Switches 5 e 6 – Atribuídas a calibração automática e manual, relacionando uma entrada mínima e máxima a seus respectivos endpoints. Está calibração será feita manualmente pelo controle implementado.

b Motores com Escovas (*brushed*)

É necessário levar em consideração alguns aspectos ao escolher entre motores com ou sem escovas, como a relação torque/velocidade requeridas, fonte de tensão disponível, tempo de vida e fatores externos, como humidade e temperatura.

Motores *brushless* (sem escovas) oferecerem certas vantagens, disponibilizando maior torque, precisão, resistência física e menores dimensões, mas em contrapartida sua construção é mais cara e trabalha com 3 fases distinta, necessitando um controle mais complexo [12].

Para esta aplicação não será necessário um torque elevado ao ponto de optar por um motor *brushless*. A complexidade de seu controle também é um ponto contra sua integração ao sistema de controle que será utilizado, evitando o uso de mais componentes. Sendo assim, optou-se pela utilização de motores escovados, que são bem populares em competições de batalha e robôs de sumô [13].

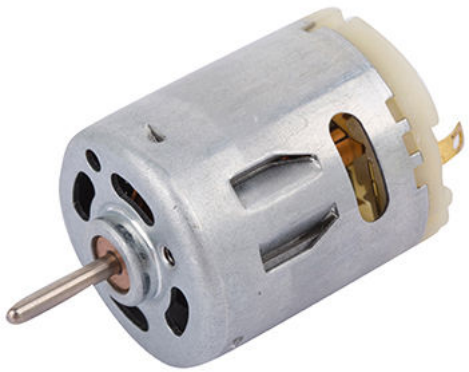

Figura 13: Motor DC com escovas 12V 1000RPM

Inicialmente, foi considerado que o torque não seria um fator prioritário em relação a velocidade para aplicação prática neste projeto, mas à medida que a atuação foi implementada para o sistema de controle, foi verificado um problema, que será descrito mais adiante, onde existiu a necessidade de troca do motor por um de maior torque, sendo escolhido um motor de tensão 12V com rotação 1000RPM quando operado sem carga.

Tensão de operação	4 – 12V
RPM (sem carga)	1000 RPM a 12V
Corrente sem carga	50mA a 12V
Corrente com carga máxima	300mA (stall)
Dimensões	46 x 36 x 25mm
Peso	100g

Tabela 3: Especificações do motor [4]

c Bateria

Para alimentação de toda a eletrônica e motores, foram consideradas as opções tradicionais de bateria comumente utilizadas em aplicações de baixa potência, baterias de polímeros de lítio (*LiPo*) e baterias íons de lítio (*li-ion*). As baterias *li-ion* utilizam um solvente inflamável de líquido orgânico responsável pela troca de íons entre os eletrodos, enquanto as do tipo *LiPo* são secas, sendo mais utilizadas no mundo do rádio controle, também sendo mais leves e compactas que as do tipo *li-ion* [13].

Foi feita uma análise em cima de uma bateria *Li-Po* disponível para utilização, uma bateria da *fullymax* de 2200mAh, 3 células (3S) e capacidade de carga e descarga de 30C.

Células	**3S**	**Descrição**
Mínimo	9	Voltagem com risco de dano permanente
Descarregada	10,2	Baixa voltagem Segura
Nominal	11,1	Voltagem nominal de cada pack
Armazenada	11,4 V1.00	Voltagem de armazenamento
Carregada	12,6	Voltagem com carga máxima armazenada

Tabela 4: Tensões de referência de uma bateria 3S [5]

Figura 14: Bateria *fullymax* 2200aH 3S

Baterias do tipo *Lipo* são compostas por células ligadas em série para se obter determinada tensão. Cada célula é capaz de armazenar 3,7V, podendo chegar a até 4,2V quando carregada. Tratando-se de uma bateria de 3 células (3S), ela é capaz de armazenar uma tensão nominal de 11,1V, chegando à 12,6V quando carregada. Caso uma célula descarregue abaixo de 3V, ela poderá ser permanente danificada, exigindo um cuidado com o monitoramento de sua tensão. Como dito anteriormente, este risco foi minimizado com a utilização do controlador *SaberTooth*, já que foi acionado o modo "*Lithium Cutoff*", onde a força é cortada se uma célular estiver abaixo dos 3V [11].

Também é possível utilizas as células ligares em paralelo, para aumentar a carga mAh das baterias.

A capacidade da bateria é classificada em miliamperes por hora (mAH), que determina quanto de corrente ela fornece no período de uma hora, no caso temos 2,2A em uma hora. Supondo uma situação em que haja o consumo contínuo somente dos motores, considerando a pior situação possível, com ambos em *stall*, utilizando o valor de 300mA para cada um, como descrito na tabela [4], teríamos um consumo de 600mA.

$$tempo = \frac{consumo}{capacidade} \tag{1}$$

Com esta capacidade, neste exemplo a bateria duraria: 3 horas e 39 minutos.

Quanto a taxa de carga e descarga, ela é representada pela nomenclatura em C (*continuous discharge rate*), e em teoria representa a capacidade de fornecer corrente a uma taxa multiplicando-se este valor com a capacidade da bateria. Neste caso, com capacidade de 2200mAh e taxa de carga e descarga de 30C, ela é capaz de fornecer corrente a uma taxa de 66000mAh, 30 vezes sua capacidade. Esta capacidade de carga e discarga é diretamente proporcional ao preço e peso das baterias.

d Radio receptor

Para receber a saída do sistema de controle de forma remota, foi utilizado a comunicação por rádio frequência, sendo utilizado o módulo receptor *Turnigy 9x8C*.

Figura 15: Rádio receptor *Turnigy 9x8C*

Este módulo opera numa frequência de 2.4GHz, com até 8 canais, sendo que somente 2 foram utilizados. Uma contrapartida é a impossibilidade de se trabalhar com sinais em *PPM*, apenas *PWM*. Neste projeto foi utilizado os dois canais através de largura de pulso (*PWM*), atendendo os requisitos necessários.

e Radio transmissor

Para que o receptor possa trabalhar com a saída enviada pelo sistema de controle, é necessário gerar um sinal com frequência compatível com o mesmo, sendo utilizado o rádio transmissor *turnigy GTX3* de 3 canais. O diferencial deste transmissor foi a presença de um contato fêmea formato P2 (3,5mm) com suporte a *PWM*, ao qual foi o formato utilizado para enviar a saída do sistema de controle de posicionamento.

Figura 16: Rádio transmissor *turnigy GTX3*

f Arduíno

Um dos problemas enfrentados foi falta de capacidade do sistema gerar como saída um sinal em *PWM* que pudesse se comunicar com o rádio, tendo a necessidade de um componente que intermediasse esta função.

Um microcontrolador cabe neste papel, sendo utilizado um arduíno nano devido ao seu baixo preço e vasto acervo de bibliotecas disponíveis.

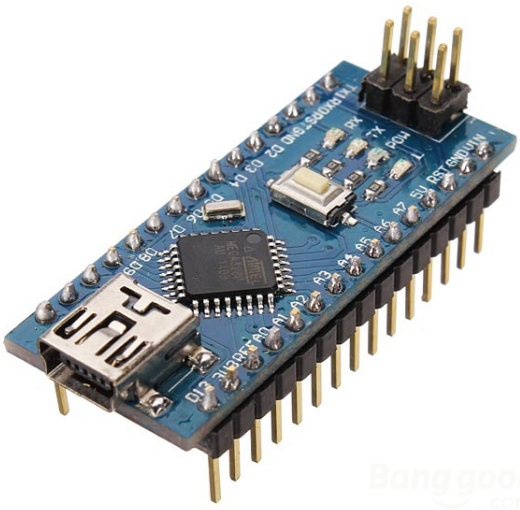

Figura 17: Arduíno nano

Desta forma, o sistema irá gerar uma saída que terá seu valor convertido em uma sequência de caracteres. Esta sequência será enviada por comunicação serial ao arduíno, que converterá seus valores para uma saída *PWM* compatível com o rádio transmissor [6].

Microcontrolador	ATmega328
Tensão de operação	5 V
Memória Flash	32 KB
SRAM	2KB
Clock Speed	16 MHz
Pinos Analógicos	8
EEPROM	1 KB
Corrente CC por pino	40mA
Tensão de alimentação	7-12 V
Pinos I/O digitais	22
Saídas PWM	6
Consumo	16 mA
Dimensões	18 x 45 mm

Tabela 5: Especificações do Arduíno [6]

g *LabView*

O software será desenvolvido em linguagem gráfica G, que deriva do *C++*, no ambiente de programação *LabVIEW*, em conjunto com o módulo de visão *Imaq Vision*. Os principais campos de aplicação do *LabVIEW* são a realização de medições e a automação. A programação é feita de acordo com o modelo de fluxo de dados, o que oferece a esta linguagem vantagens para a aquisição de dados e para a sua manipulação. [3]

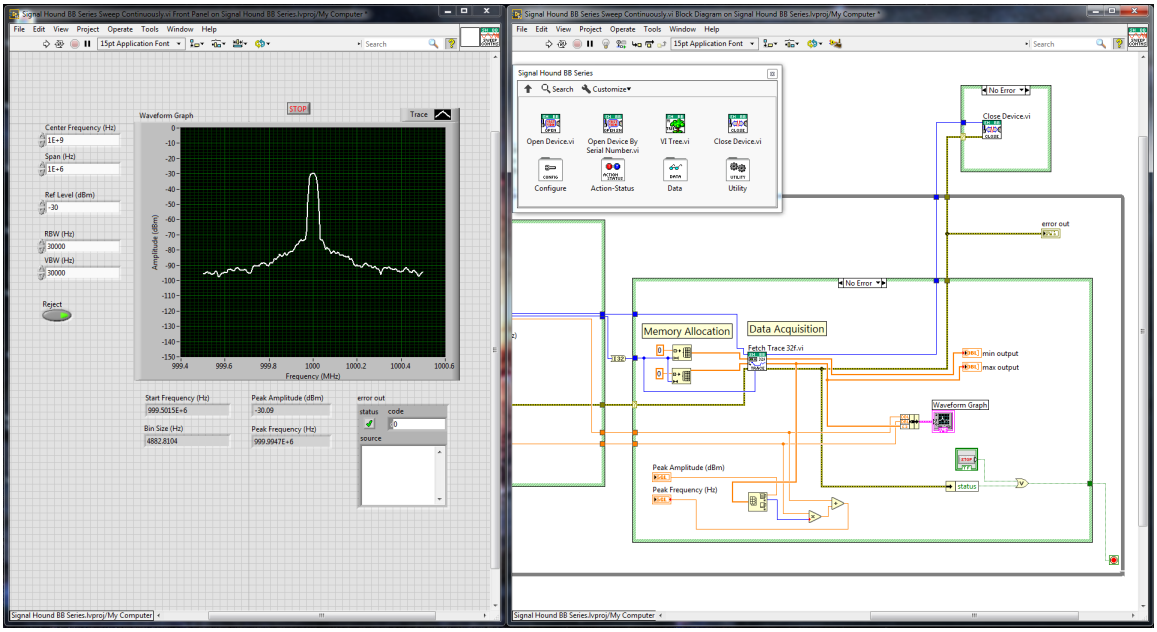

Figura 18: Exemplo de medição utilizando o *LabView*

O programa possui uma interface gráfica intuitiva e que facilita sua programação, sendo especialmente útil para medições e alterações de parâmetros em tempo real, o que facilitou bastante os processos de verificação do reconhecimento da posição do robô, assim como as saídas do sistema de controle e verificação e calibração do controle *PID*. Em contrapartida, como mencionado, o programa não é o ideal para gerar um sinal de saída no formato de *PWM*, obrigando o uso de um microcontrolador para esta etapa.

h *SolidWorks*

Para modelagem do projeto, assim como impressão de peças necessárias em uma impressora 3d, foi utilizado o software de modelagem 3d *SolidWorks*.

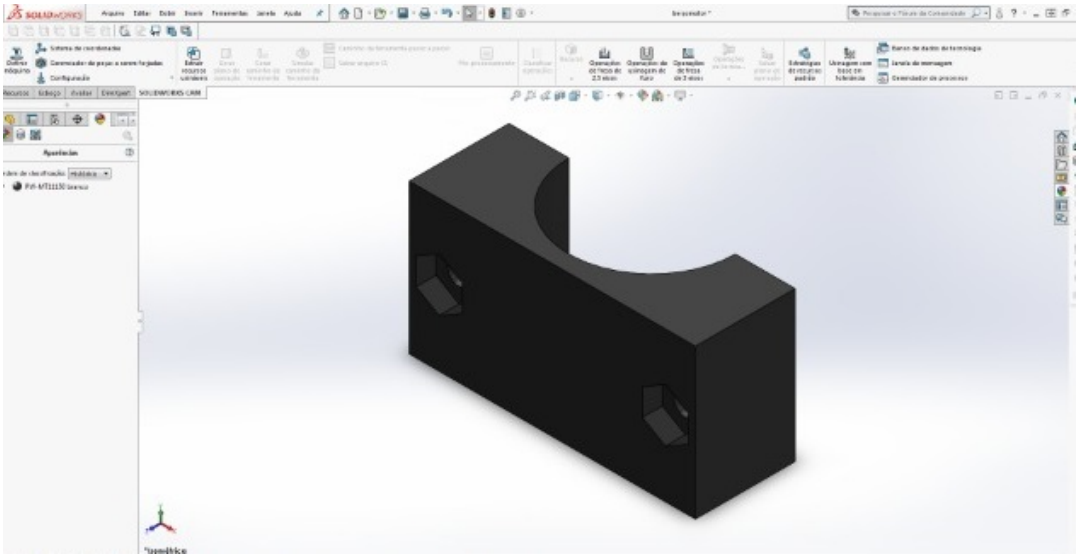

Figura 19: Modelagem da peça do berço do motor a ser impressa

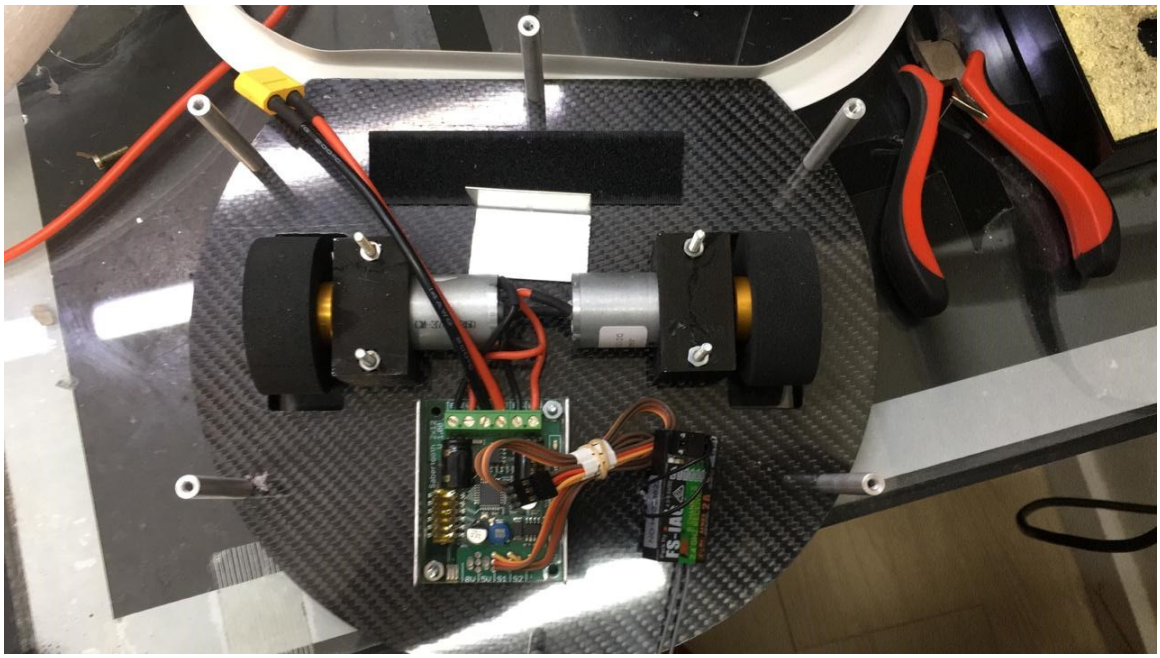

Figura 20: Resultado final do robô sem tampa com eletrônica exposta

4 Desenvolvimento de sistema de posicionamento indoor (IPS)

a Identificação Individual

Para detecção de múltiplos alvos, foi necessário determinar alguma forma de identificação individual (que chamaremos de *ID*) para cada robô ou alvo desejado.

Foram descartadas reconhecimento de forma pois a tendência tanto para esta como para outras aplicações seria a de que os robôs tenham projetos semelhantes, logo uma diferença de design seria muito sutil. Sendo assim, foi necessário utilizar algo que, visualmente, destacasse cada objeto como possuindo um padrão único.

Figura 21: Exemplo de *QR code*

Uma ideia considerada foi a utilização de *QR codes*, porém, nos testes realizados, foi observado que os padrões não eram identificados com precisão quando o objeto estivesse a uma distância razoável, sendo necessário uma câmera de alta resolução. Além de encarecer o projeto, isto implicaria no processamento de imagens em alta definição, o que aumentaria consideravelmente o tempo de execução do algoritmo.

Foi testado uma simplificação deste conceito, utilizando imagens com padrões únicos, com desenhos mais simples, porém o mesmo problema de falta de nitidez e necessidade de uma câmera com foco de alta definição à distância se mostrou presente.

A solução que se mostrou viável foi a utilização de *LEDS* em padrões que atribuíssem uma identificação individual a cada objeto. Cada alvo tem dois *LEDS* na sua parte superior, voltada à câmera. Desta forma, o sistema reconhece o *LEDS* como ponto de referência de acordo com uma intensidade de brilho pré-calibrada, verificando se existe alguma outra referência numa determinada distância para confirmar se o alvo pode ser atribuído como um objeto individual de *ID* único.

b RGB x Infravermelho

1 RGB

Inicialmente, foi utilizado um plano de cores *RGB* para determinar o padrão de cada objeto. Utilizando somente as cores primárias de referência: vermelho, verde e azul. O resultado obtido atendeu as necessidades do sistema, mesmo à distância, também apresentando uma boa velocidade de processamento.

Um problema encontrado foi a indispensabilidade de calibração do intervalo de faixa de intensidade para cada ambiente testado, havendo necessidade de recalibração de acordo com a luminosidade presente. Como o ambiente de testes tinha contato com o exterior, era necessária uma recalibração de acordo com o horário do dia devido a iluminação externa. Esta recalibração se mostrou especialmente problemática para a diferenciação do reconhecimento entre as cores primárias verde e azul, sendo necessária uma calibração precisa para que não houvesse uma troca de valores entre elas.

Possíveis iluminações e animações com luzes presentes no ambiente também eram detectados pelo sistema. Apesar de não se enquadrarem no padrão especificado, uma iluminação isolada era detectada,

acarretando em gasto de processamento para verificar se outra referência se encontrava num raio próximo. Ambientes claro também exigiam alta intensidade no brilho dos LEDS, necessitando muitas vezes um ambiente de controle que não fosse excessivamente iluminado.

2 Infravermelho

A alternativa utilizada para minimizar os problemas apresentados com o plano *RGB* foi a utilização de *LEDS* infravermelho, juntamente com a retirada do filtro infravermelho das câmeras, tornando visível a visualização da luz nesta frequência.

Figura 22: Exemplo de leitura de câmera com e sem filtro infravermelho

Semelhante ao processo anterior, em que foi utilizado 3 opções de referência (vermelho, verde e azul), neste caso foram atribuídos *LEDS* IR com 3 intensidades distintas, mantendo a lógica de identificação individual baseado nos padrões das diferentes intensidades de brilho.

Esta configuração apresentou resultados mais precisos, exigindo menor esforço na calibração das faixas de intensidades em comparação com o plano *RGB*. Houve menos erros de resultado ao confundir uma intensidade com outra, assim como a minimização de excesso de processamento com outras fontes de iluminação infravermelho presentes no ambiente. Um problema a se destacar é a interferência da luz do dia utilizando esta configuração, problema contornável ao se utilizar um ambiente controlado que limite a entrada de luz do sol.

Apesar desta configuração apresentar melhores resultados, houve um problema de *hardware* que impossibilitou sua viabilidade. A câmera disponível para testes em que foi possível a retirada do filtro infravermelho possuía resolução baixíssima, assim como baixo tempo de exposição da imagem. Na prática, o sistema funcionou perfeitamente enquanto o robô estava parado. No momento em que começou a implementação da atuação, a câmera enxergava as referências como borrões, impossibilitando uma leitura precisa. Com a falta de tempo e orçamento para compra de uma câmera de boa qualidade compatível com a remoção do filtro infravermelho, a opção escolhida foi retornar ao filtro no plano *RGB*, visto que a câmera disponível atendia os requisitos, não havendo perda de qualidade com a movimentação das referências.

Apesar dos melhores resultados com o infravermelho, a utilização das referências com *LEDS RGB* atende satisfatoriamente as condições deste projeto.

c Conceito de Visão Artificial

Segundo Bianchi (2001) [14], a Visão Artificial é definida como sendo um sistema composto por uma estrutura na qual uma imagem passaria por três processos distintos e sequenciais.

A primeira etapa, de implementação física, é responsável pela aquisição da imagem, ou seja, a etapa sensorial e perceptiva do processo de visão. Na segunda etapa, de algoritmo e tipo de representação, a problemática é como a informação vinda dos sensores pode ser representada de forma que seja útil para

o usuário da imagem. E finalmente na terceira etapa, da teoria computacional teríamos o que é definido como o objetivo da visão, sobre a qual devem existir teorias sobre como utilizar as representações para se atingir os objetivos desejados. Dessa forma pode se dizer que as principais operações realizadas por um sistema de visão artificial são o processamento, a análise e interpretação de imagens.

As principais operações realizadas no tratamento de uma imagem podem ser expressas através da sequência de operações de aquisição e processamento expressas na figura a seguir [14].

Figura 23: Sequência para aquisição e processamento de uma imagem

d Aquisição da Imagem

Nesta primeira fase o principal objetivo é definir um meio de conversão de uma imagem ou informação visual para um sinal digital ou analógico. Esta conversão se dá por meio de câmeras ou sensores óticos que capturam a imagem ou informação e as transformam em sinais elétricos. Segundo Tommaselli et al. (2000) [15], as câmeras digitais são compostas de um sistema de lentes, um chip sensor, que pode ser do tipo *CCD* (*Charge Coupled Device*) ou *CMOS* (*Complementary Metal Oxide Semiconductor*), processadores e uma memória para coleta e armazenamento de imagens digitais.

As células de uma matriz *CCD* são mais conhecidas como pixel que é a abreviatura de picture element. A luz incidente sobre a *CCD* passa por um conjunto de lentes que convergem a luz vinda do objeto de interesse e a distribui sobre a área da *CCD*, sensibilizando-a.

Esta sensibilização provoca uma excitação nas células capacitivas, e estas armazenam uma energia elétrica proporcional à intensidade luminosa que atinge cada pixel. Desta maneira, a distribuição de carga sobre a superfície do *CCD* é uma representação discretizada da cena que foi capturada em um determinado instante [15].

A conclusão da aquisição da imagem ocorre através de um processo realizado por um conversor A/D (Analógico/ Digital) que tem por objetivo a transformação do sinal elétrico gerado pelo CCD em um sinal digital que é armazenado em uma memória temporária.

Como são construídas como uma matriz, a forma mais fácil de obter informações sobre cada célula é varrer a matriz na forma de linhas e colunas. Na verdade, cada linha da matriz é lida de uma vez e o valor de cada célula é armazenado em um registrador paralelo, e então os dados deste registrador são enviados serialmente.

1 Pré-Processamento

O pré-processamento é responsável por fazer o tratamento da imagem para minimizar o custo computacional e facilitar a extração de características. Segundo Gonzalez e Woods (2002) [16], as técnicas de pré-processamento são utilizadas para aprimorar a qualidade de uma imagem para corrigir efeitos de iluminação, contraste, distorções e nitidez.

2 Segmentação

Na etapa de segmentação, divide-se a imagem em conjunto de *pixels* ou objetos para sua representação, com objeto simplificar sua análise [16]. Sendo comumente utilizado para identificação de objetos

e formas.

3 Identificação de objeto

Esta etapa não será utilizada, mas diz respeito a utilização de um algoritmo por aprendizado, como uma rede neural, para que seja feito um treinamento com o objetivo de reconhecer múltiplos objetos ou um algo em específico.

4 Reconhecimento de padrões

Será aplicado um reconhecimento para identificação das referências luminosas de acordo com sua intensidade de brilho e cor de referência, seja na solução por RGB ou por infravermelho. Se enquadrando nas especificações adequadas para este projeto, é feito um outro processamento para verificar a distância até outras referências próximas, reconhecendo o objeto com uma identificação única caso haja um retorno positivo para esta verificação no reconhecimento dos padrões pré-estabelecidos.

e Desenvolvimento do Algoritmo

Como mencionado anteriormente, foi feita a escolhe de desenvolver o algoritmo no *software LabView*, utilizando o módulo de visão *ImaqVision*.

1 Inicialização

Utilizando a função *Vision Acquisition*, dentro do módulo Vision Express, é possível inicializar o processamento vindo da câmera:

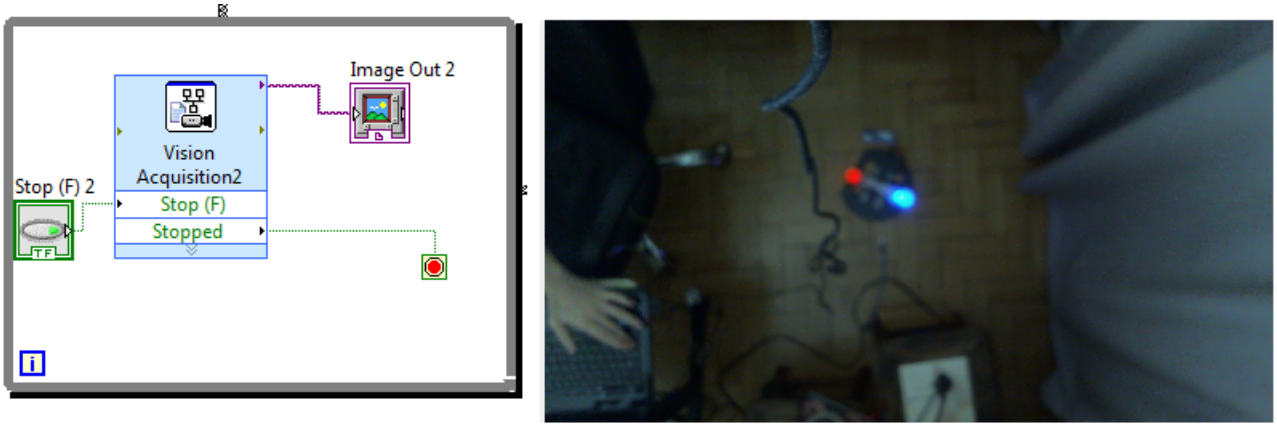

Figura 24: Inicialização da aquisiçao de imagem

2 Parâmetros para melhorar visualização da saída

Foi necessário definir alguns parâmetros em relação a saída da imagem para melhor visualização do resultado a ser analisado.

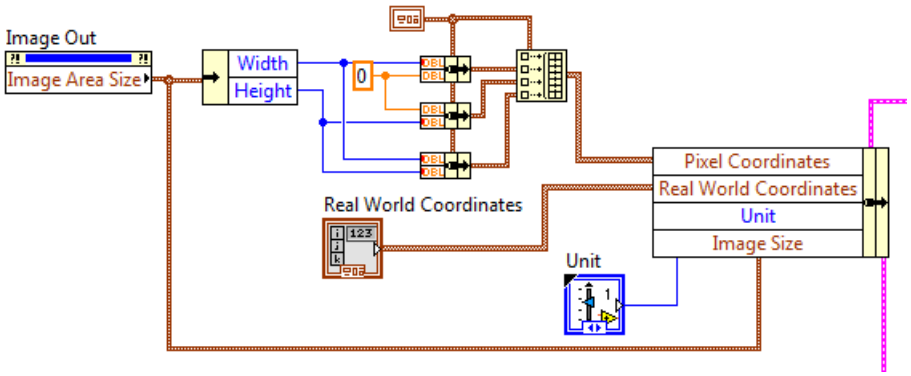

Figura 25: Definição de tamanho da janela da saída e coordenada a ser utilizada de referência

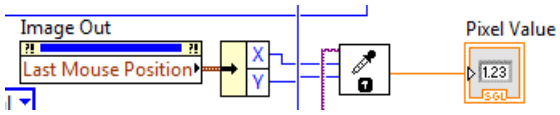

Figura 26: Definição de valor do pixel de acordo com coordenada do mouse sobre imagem – facilitar verificação se resultados são pertinentes

3 Divisão nos planos RGB

A seguir, separou-se a imagem em 3 planos individuais *RGB*, através do *subVI ExtractSingleColorPlane* [3].

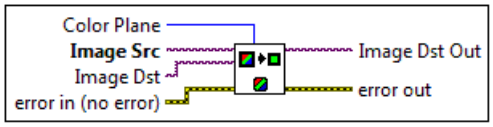

Figura 27: *Subvi ExtractSingleColorPlane*

Com este *subVI*, foram criadas 3 classes distintas de saídas para a imagem baseada nas cores que estamos analisando como referência: planos vermelho, verde e azul.

4 Pré-Processamento

Como forma de aprimorar os resultados, assim como ajudar na calibração de cada plano de cor de acordo com o ambiente a ser testado, foram implementados alguns parâmetros para ajudar na segmentação de cada plano de cor, estas saídas são parâmetros de referência para o *subVi* de inicialização *Vision Acquisition*. Como este *subVI* roda continuamente, é possível alterar os parâmetros de pré-processamento para o *frame* seguinte a ser captado:

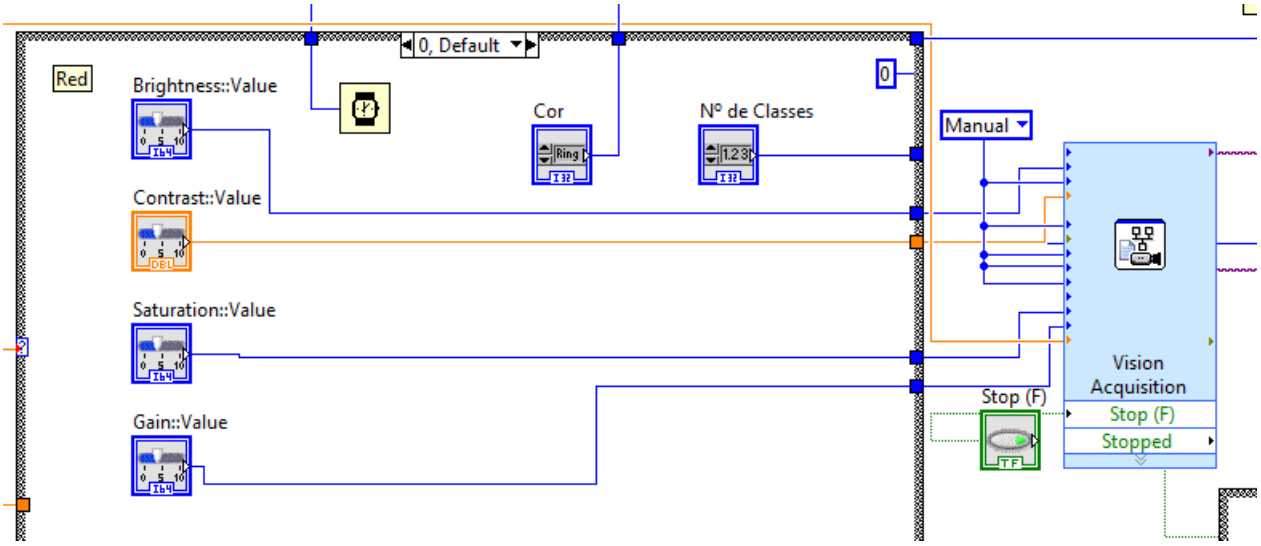

Figura 28: Parametros utilizados no pré-processamento

5 *Brightness* (brilho)

Corrige a claridade da imagem, tanto nas partes iluminadas como nos contornos.

Figura 29: (a) referencia, (b) brilho alto, (c) brilho baixo

6 *Contrast* (contraste)

Refere-se a diferença entre a região clara e escura da imagem, ajustando este contorno.

Figura 30: (a)referencia, (b) contraste alto, (c) contaste baixo

7 *Saturation* (Saturação)

Refere-se a percepção que temos da intensidade da cor da imagem.

Figura 31: (a) Referencia, (b) saturação alta, (c) saturação mínima

8 *Gain* (Ganho)

O ganho é o equivalente fotográfico ao processamento de filme em que você força o aumento da sensibilidade removendo informações do fundo subexposta. Então o fundo perde os detalhes [17].

9 Resultados por plano de cor

Aplicando e ajustando estes parâmetros individualmente em cada um dos planos *RGB* divididos anteriormente, temos os seguintes resultados:

Figura 32: Resultado da separação por plano de cor com os parâmetros determinados num ambiente com pouco controle sobre a luz – (a) imagem original, (b) plano vermelho, (c) plano verde, (d) plano azul

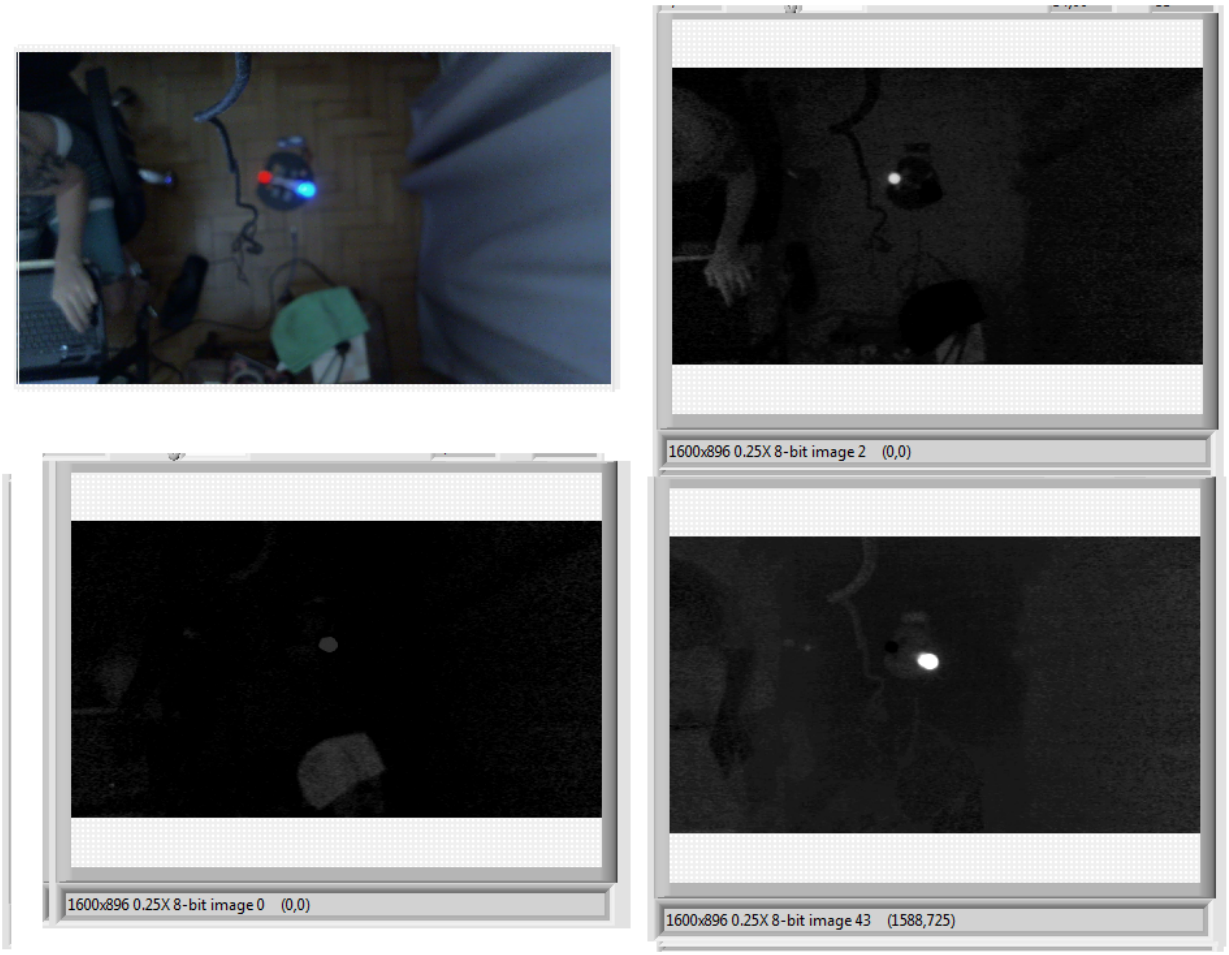

Figura 33: Resultado da separação por plano de cor com os parâmetros determinados num ambiente controlado com menos iluminação – (a) imagem original, (b) plano vermelho, (c) plano verde, (d) plano azul

10 Pós Processamento

Assim como na etapa de pré-processamento, foram utilizados parâmetros para pós processamento individualmente de acordo com cada plano de cor.

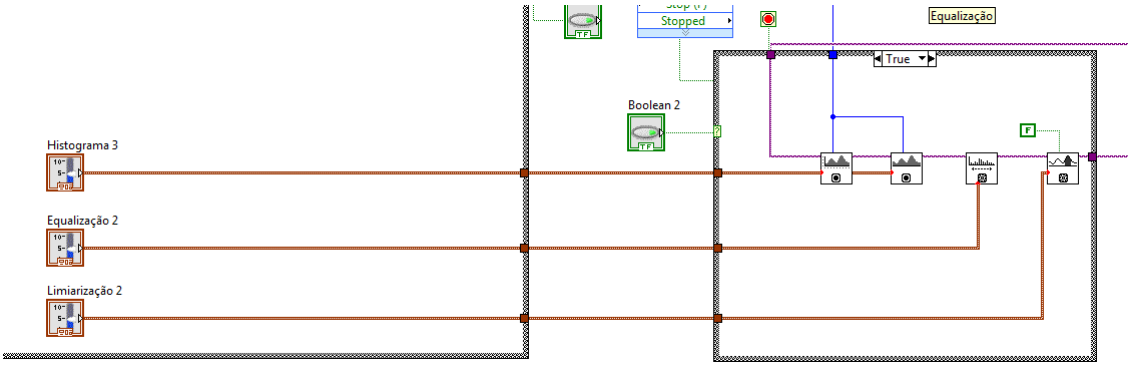

Figura 34: Implementação no *LabView* do histograma, equalização e limiarização

11 Histograma

O histograma de uma imagem indica o percentual de *pixels* que a imagem tem de determinado nível de cinza ou cor [18]. O *LabView* possui um *subVI* próprio para isso [3]: O *LabView* possui um *subVI* próprio para isso:

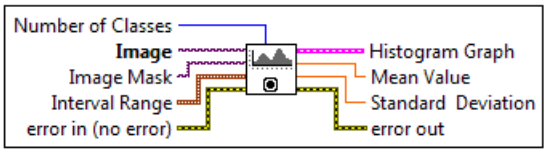

Figura 35: *SubVI* do histograma [3]

12 Equalização

Tem como objetivo a obtenção da melhor variância do histograma de uma imagem, obtendo um resultado final com melhor contraste.

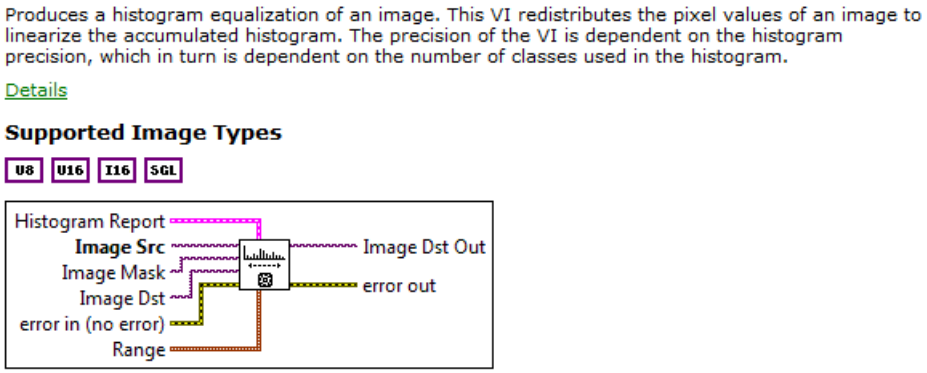

Figura 36: *SubVI* do Equalização [3]

13 Limiarização

Se trata de um processo de segmentação da imagem baseado na diferença de níveis de cinza que compõe diferentes objetos de uma imagem, podendo ser selecionados os grupos de *pixels* em cinza com níveis abaixo e acima do limiar, atribuindo-se um valor fixo a todos os pixels do mesmo grupo, de forma que a saída será uma imagem em que os *pixels* podem ter somente valores 1 ou 0 [19].

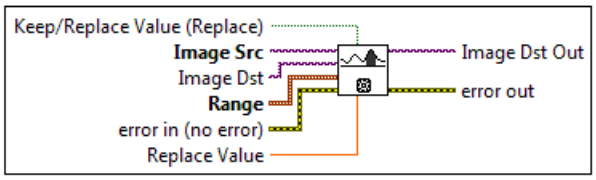

Figura 37: *SubVI* do Limiarização [3]

14 Resultados

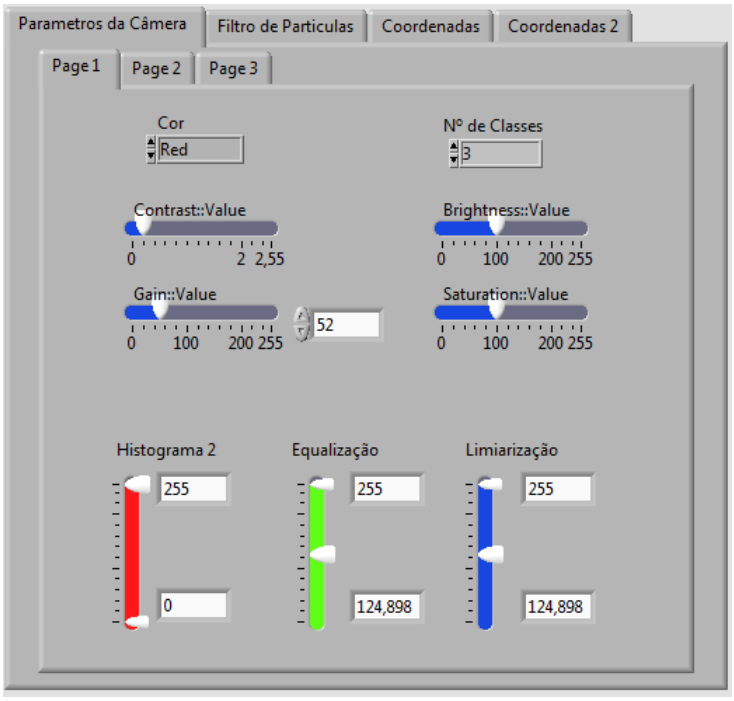

Figura 38: Painel de controle dos parâmetros

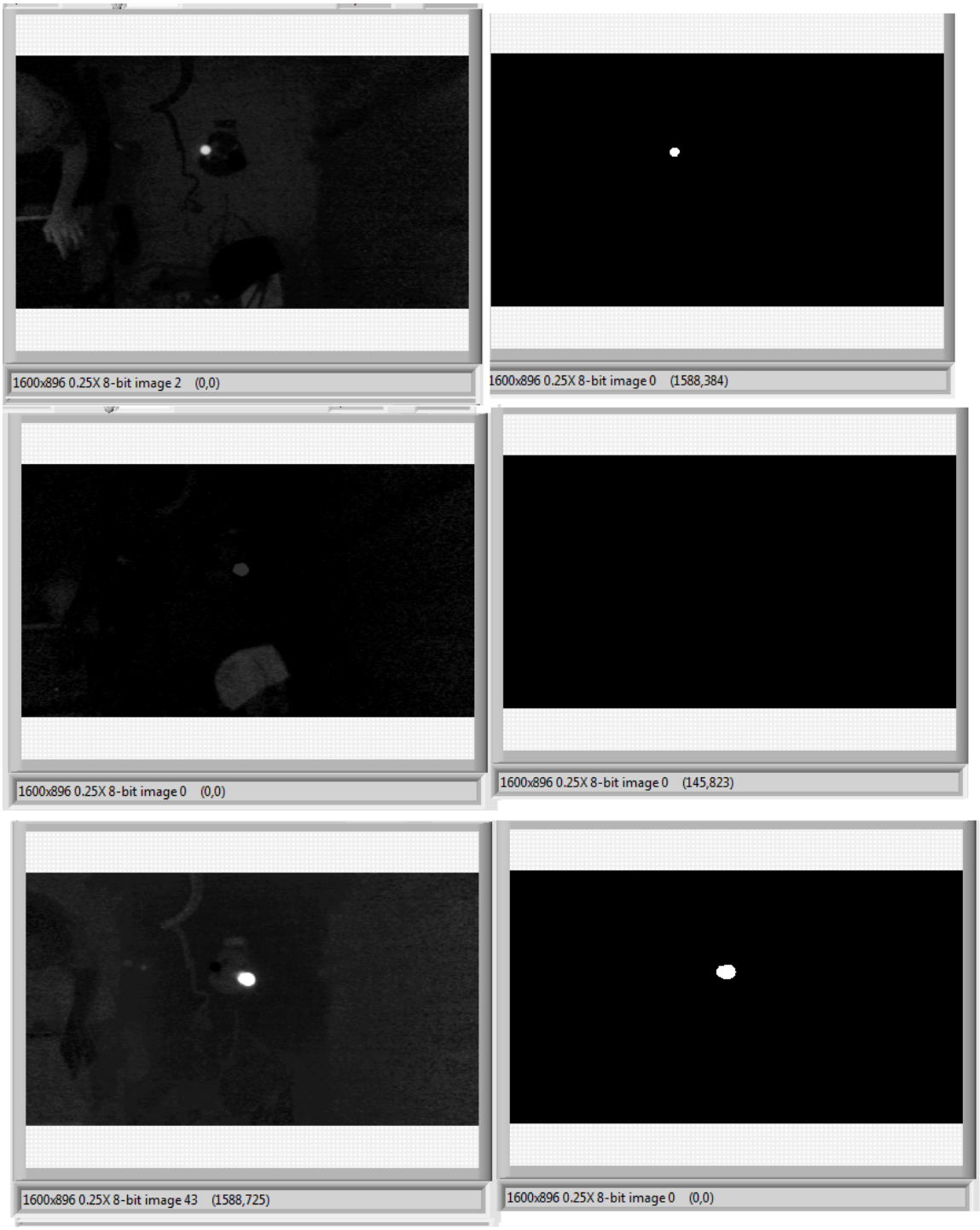

Figura 39: Resultado dos processamentos descritos em cada camada de cor – (a) Vermelho, (b) Verde, (c) Azul

15 Erosão

Trabalhando em cima da saída do filtro de limiarização, em que os *pixels* têm valor 1 ou 0, o filtro de dilatação verifica os valores de cada componente da matriz deste resultado, fazendo com que os vizinhos dos *pixels* de valor 0 também tenham valor 0, removendo uma camada da imagem. O objetivo deste filtro é a diminuição de ruídos e a redução de *pixels* isolados que possam interferir na análise desejada após aplicação dos filtros anteriores.

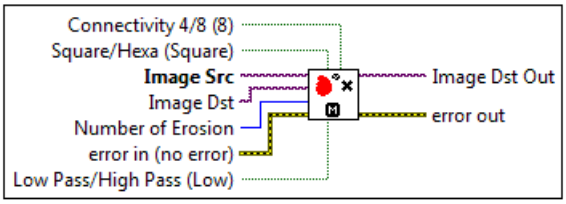

Figura 40: *SubVI* do filtro de erosão [3]

16 Dilatação

Semelhante ao filtro de erosão, é verificado os valores de cada componente da matriz deste resultado, fazendo com que os vizinhos dos componentes de valor 1 tenham valor 1, de forma a adicionar mais uma camada de *pixels* a imagem. Este filtro tem como objetivo o fechamento de vãos no resultado final após aplicação final dos filtros.

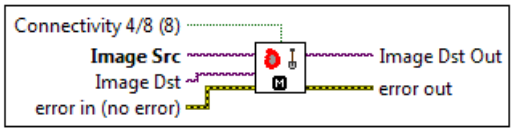

Figura 41: *SubVI* do filtro de Dilatação [3]

17 Filtro de Abertura

Trata-se da aplicação do filtro de erosão seguido do filtro de dilatação. Desta forma, remove-se os ruídos e *pixels* isolados na imagem, recuperando a camada perdida da imagem pós-processada, sem perda de informação.

18 Filtro de Partículas

Este filtro tem por objetivo calcular o centro de massa dos *pixels* da imagem. Uma forma de proceder para este resultado é multiplicar a matriz binária de saída do processamento da imagem por outra matriz com os índices de suas linhas, desta forma o resultado será uma matriz onde somente os componentes de valor 1 agora terão valor correspondente a sua linha. Ao aplicar uma média destes valores teremos o componente x referente ao centro de massa da coordenada x.

Aplicando a metodologia anterior, mas utilizando uma matriz onde os índices sejam colunas, teremos o centro de massa da coordenada y.

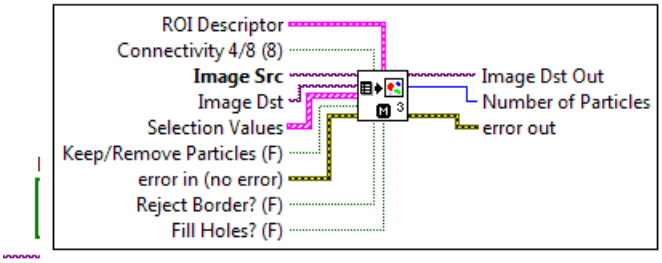

Figura 42: *SubVI* do filtro de partículas [3]

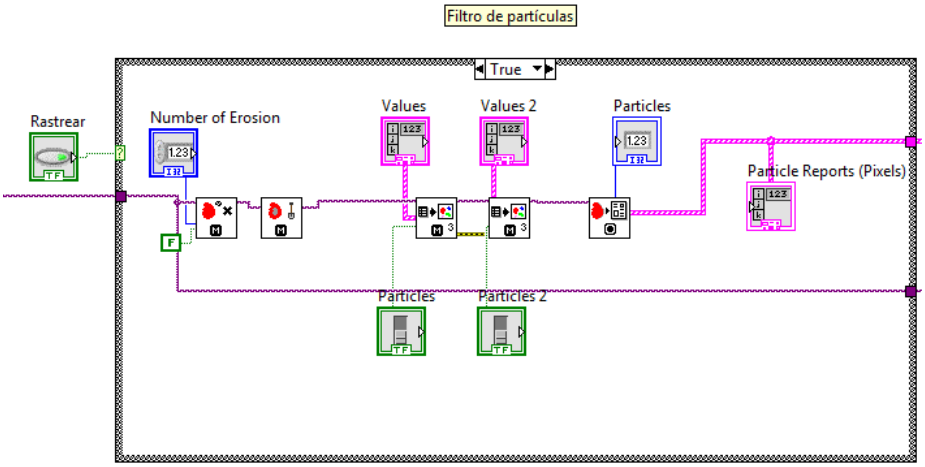

Figura 43: Implementação dos filtros de abertura e de partículas

Com a implementação do filtro de partículas após o processamento descrito pela aplicação dos filtros anteriores, é possível extrair a coordenadas de cada objeto utilizado como referência para detecção dos alvos:

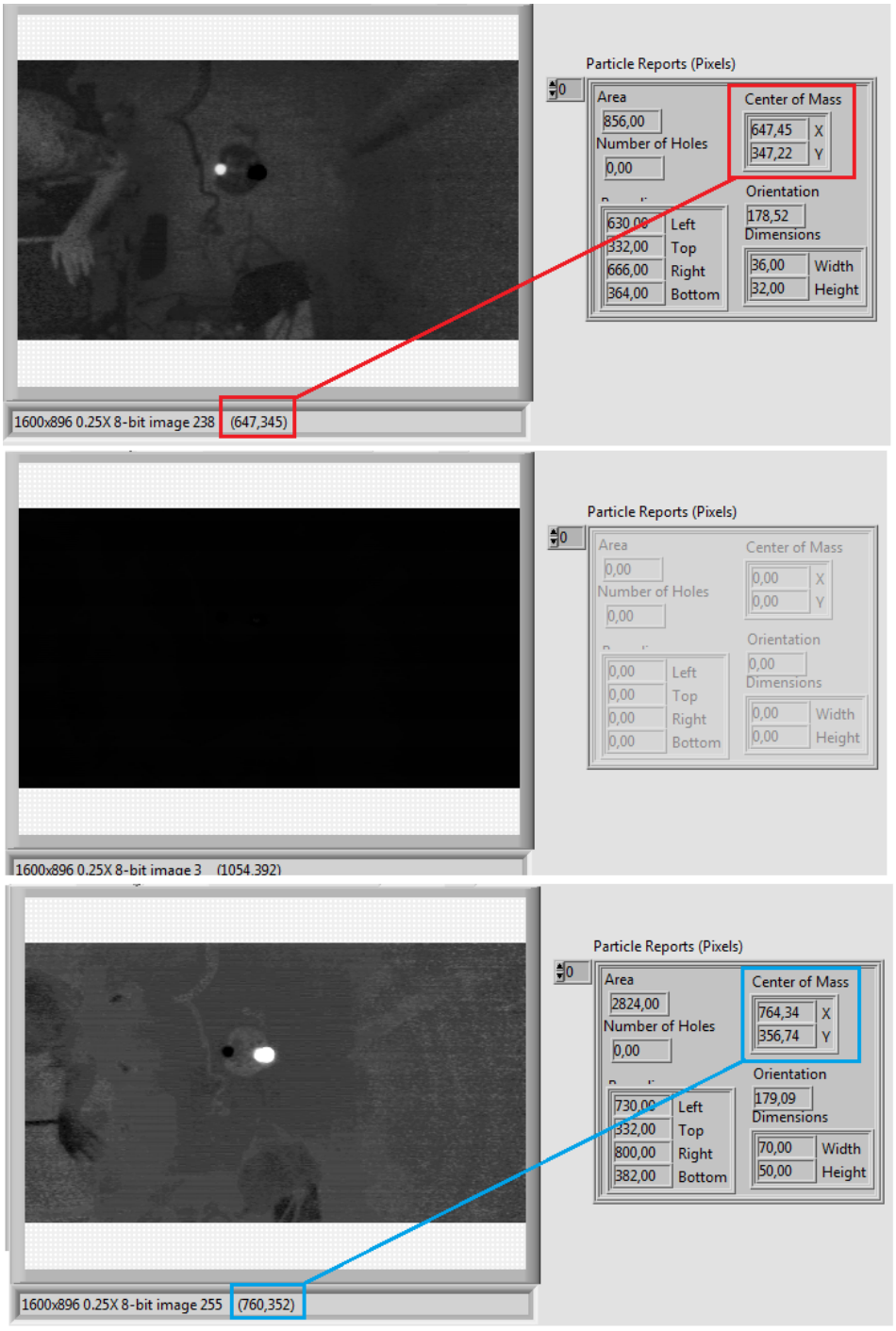

Figura 44: Saída do centro de gravidade das imagens em cada plano de cor após implementação do filtro de partículas – (a) Vermelho, (b) Verde, (c) Azul

Nota-se que não houve detecção no plano verde, visto que as cores utilizadas neste exemplo para referência foram azul e vermelho.

f Processamento das coordenadas

1 Identificação de Alvo

De posse dos valores das coordenadas das duas referências, calcula-se a distância entre elas expresso pela equação (2):

$$dist = \sqrt{(Xref1 – Xref2)^2 + (Yref1 – Yref2)^2} \quad (2)$$

Caso a distância entre as referências seja menor do que um determinado valor, considera-se que ambas as referências compõem o mesmo alvo, verificando qual o *ID* equivalente a este objeto. Nota-se que este valor de distância deve ser ajustado de acordo com a distância entre as referências e a distância da câmera até o alvo. Ele é correspondente a distância em *pixels*.

O sistema tratará que a coordenada do robô é a posição da referência localizada na sua parte de trás, fornecendo as saídas x e y de sua coordenada de posição.

2 Distância até destino

De forma semelhante ao calculado no item anterior, é utilizado a mesma equação para calcular a distância do robô até seu destino, sendo que agora os parâmetros são a posição do robô, obtido na etapa de sua identificação, e a posição do alvo.

$$dist = \sqrt{(Xrobo – Xalvo)^2 + (Yrobo – Yalvo)^2} \quad (3)$$

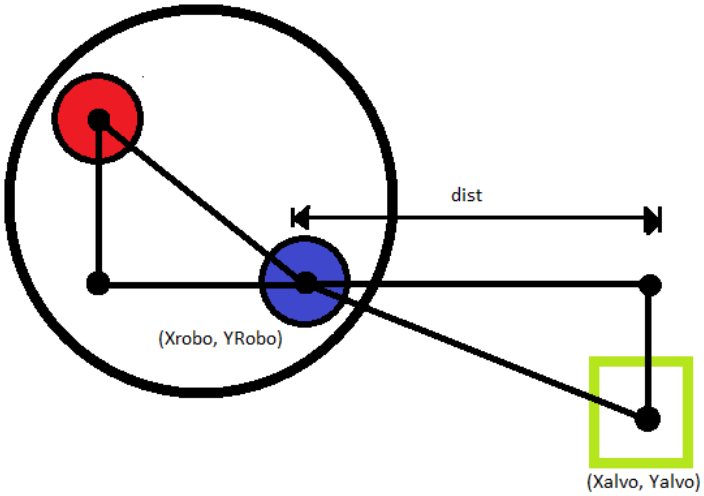

Figura 45: Representação das coordenadas do robô, alvo e a distância entre ambos. Bola azul representa a referência traseira, enquanto a bola vermelha representa a referência frontal.

3 Cálculo da direção do robô

De acordo com a posição das duas referências de luz, é calculado o ângulo global em que o robô se encontra, adotando o canto da imagem obtida, a coordenada (0 , 0) como referência. Para esta aplicação, foi adotado que o *LED* de referência vermelho seria sempre posicionado na frente do robô. Quando se utilizou a referência de luz infravermelha, o *LED* da frente era o de maior brilho.

$$globalAngle = \arctan \frac{Yfrente - Ytras}{Xfrente - Xtras} \quad (4)$$

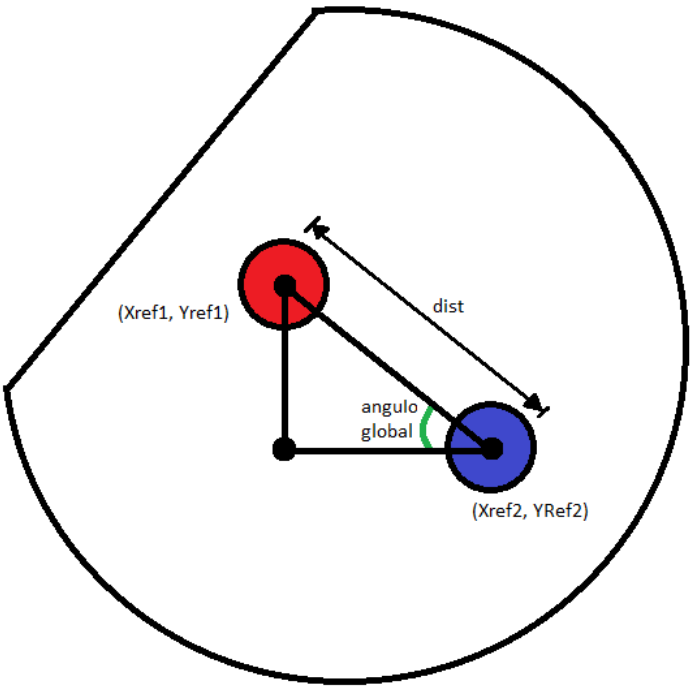

Figura 46: Representação do ângulo global do robô de acordo com suas referências em relação a coordenada (0,0)

O Ângulo será 0 quando o robô estiver em direção a coordenada (0,0).

4 Cálculo do Ângulo até o Alvo

Semelhante ao Item Anterior, é calculado o ângulo do alvo em relação ao eixo horizontal que passa pela referência traseira do robô:

$$targetAngle = \arctan \frac{Yrobo - Yalvo}{Xrobo - Xalvo} \tag{5}$$

O valor desejado para utilizar no sistema de controle é a diferença entre o ângulo global do robô e o ângulo do alvo em relação ao eixo horizontal na referência do robô.

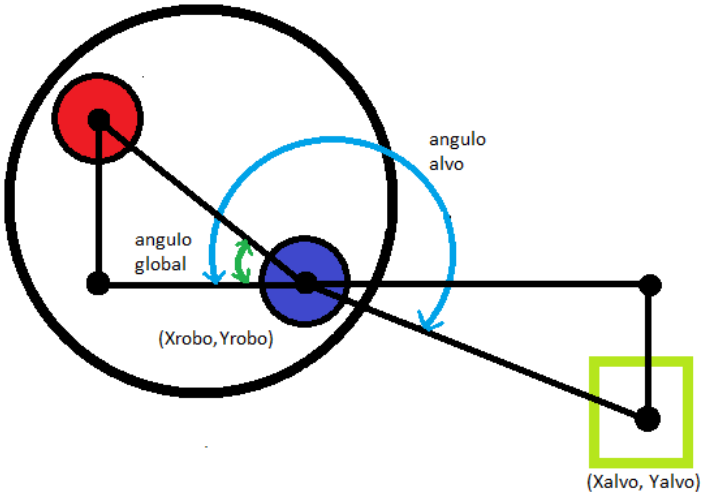

Figura 47: Representação do ângulo global e do ângulo em relação ao alvo

Por fim temos que o ângulo que o robô deve girar é a diferença entre o ângulo global e o ângulo até o alvo.

$$finalAngle = globalAngle - targetAngle \tag{6}$$

5 Controle e atuação

Após a aquisição de todos os dados na etapa de processamento de imagem, será implementado o sistema de controle para execução na atuação do robô.

O modelo de robô desenvolvido é do tipo não-holonômico, sendo o termo atribuido a *Hertz* [20], e significa ”universal”, ”integral”, ”integrável”. Portanto, sistemas não-holonômicos podem ser interpretados como sistemas não integráveis. Sistemas deste tipo se caracterizam por apresentar algum tipo de limitação ao seu estado. Neste caso, tendo restrições em relação a seus graus de liberdade para movimentação, visto que para se deslocar para um ponto que não esteja na direção em que aponta, o robô necessita girar até este ângulo para então realizar a movimentação no sentido em que aponta ou contrário a ele, indo para frente ou para trás.

Desta forma, foi implementado uma metodologia para controlar tanto seu ângulo de giro como seu deslocamento frontal, com a seguinte ordem de processos de atuação e verificação:

1. Verifica o ângulo em que robô se encontra em relação ao alvo
2. Caso não esteja dentro de uma faixa aceitável, implementa controle e atuação somente no ângulo
3. Entrando na faixa de ângulo pré-determinada, implementa controle e atuação no deslocamento
4. Atuação no ângulo continua sendo exercida, porém com uma limitação na saída do sistema de controle enquanto o robô se movimenta
5. Atuação no deslocamento para de ser acionada quando robô chega a uma distância pré-estabelecida do alvo. Idealmente essa distância seria a medida entre o robô e o alvo
6. Enquanto essa distância até o alvo se mantenha em uma faixa aceitável, é implementado novo controle na direção e no deslocamento, desta vez com uma posição fixa como alvo
7. Durante esta etapa, caso a distância para o alvo ultrapasse a medida pré-estabelecida, o sistema entende que o robô perdeu a posse do objeto e recomeça da etapa 1

Foram implementados dois controladores distintos *PID* para as variáveis de angulação e distanciamento até o destino, sendo ele o alvo quando não está de posse do objeto e qualquer coordenada dentro de uma área pré-estabelecida quando de posse do objeto.

Conforme dito anteriormente, o *LabView* não trabalha de forma ideal ao gerar um sinal no formato *PWM* como saída [3]. Então sua saída foi enviada por comunicação serial para um arduíno nano, o qual enviou o respectivo sinal *PWM* para o rádio transmissor. Como o transmissor possui uma entrada fêmea no formato P2, foi soldado um conector P2 em um pino com saída *PWM* do arduíno e em um terra.

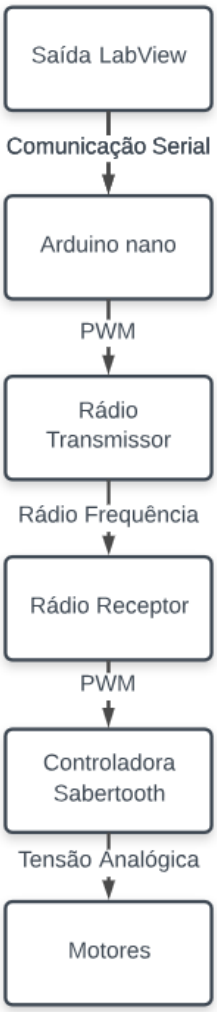

Figura 48: Fluxograma da comunicação entre *LabView*, arduíno, rádio transmissor, rádio receptor, *sabertooth* e motores.

a Controle PID

Devido a calibração necessária no sistema de posicionado indoor para cada ambiente de teste, há um ajuste necessário nos ganhos do sistema de controle para que fique compatível com a da leitura das coordenadas pela câmera. Sistemas em que a câmera esteja a distâncias diferentes dos objetos a serem reconhecidos, por exemplo, trará uma proporção diferente a suas distâncias quando medidas em *pixels*, o que por consequência dará comportamentos distintos a sistemas em que os controles possuem mesmos valores de ganhos.

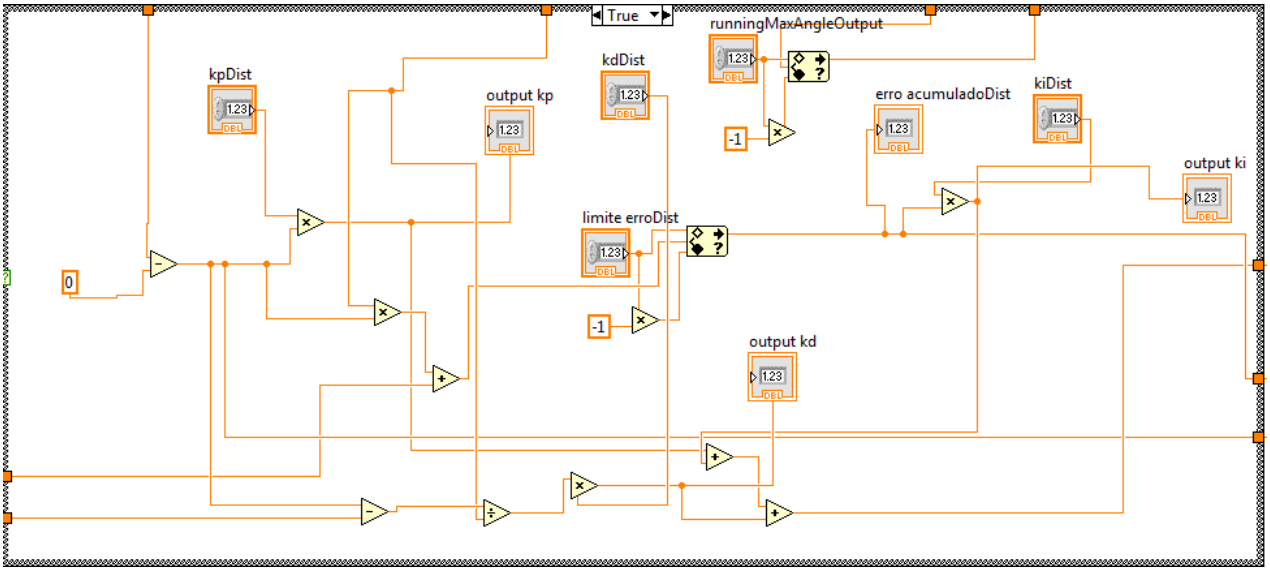

Figura 49: Implementação do controlador PID da distância até o alvo no *LabView*

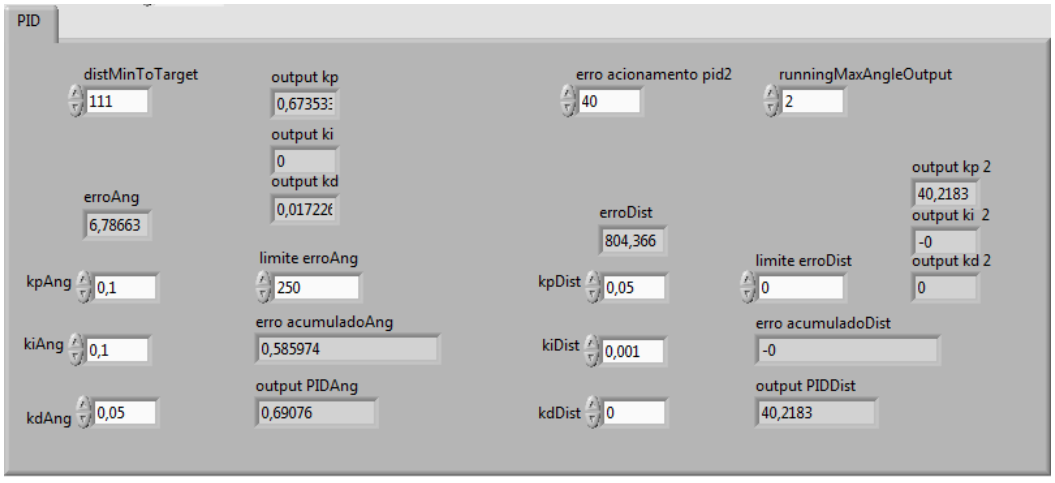

Figura 50: Painel de controle dos parâmetros dos dois controladores PID

b Controle do ângulo de direção em relação ao alvo

Esta se mostrou a parte mais crucial da calibração, e também a primeira a ser implementada.

Primeiramente, ao se chegar a esta etapa, se mostrou necessário um aprimoramento da sensibilidade da leitura da direção do robô pelo sistema de visão computacional. A leitura se mostrava errática ao identificar as coordenadas de ângulo com o objeto em movimentação, identificando as luzes de referência como borrões, comprometo a leitura do centro de gravidade pelo filtro de partículas. Este aprimoramento foi possível ao melhorar os ajustes dos parâmetros do processamento de imagem descritos anteriormente e ao trocar a câmera que estava sendo utilizada.

Um diferencial nesta calibração foi a diferença e atuação quando o sistema se encontra parado em relação a quando se movimenta nos eixos x e y, exigindo uma calibração precisa na primeira etapa de atuação do robô, o ângulo da direção em relação ao alvo.

Devido frequência de leitura do sistema, que oscilava entre 80 e 100ms, a leitura de sua angulação apresentava um sinal ruidoso no formato de ondas quadradas. Foi necessário a aplicação de um filtro para processamento da curva do sinal que suavizasse seu resultado, implementando-se um filtro digital de média móvel, através do próprio *LabView*.

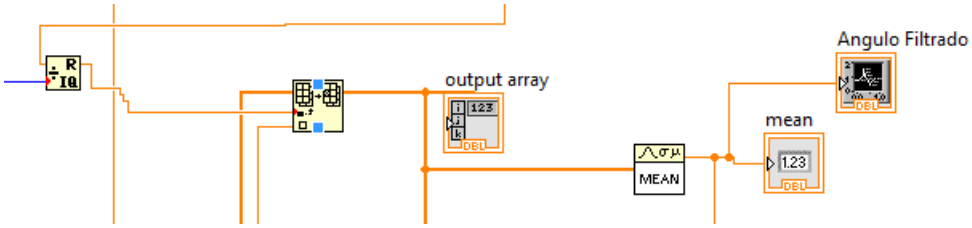

Figura 51: Implementação do filtro de média móvel no *LabView*

Trata-se de um filtro digital que calcula a média dos últimos *n* valores do sinal de entrada para cada valor de saída. Este filtro apresenta bom desempenho no domínio do tempo e mal desempenho no domínio da frequência [1]. Quanto maior a amostragem de elementos a se calcular a média, mais suave ficará o sinal de saída, mas também mais atrasado em relação ao sinal sem filtragem. É imprescindível determinar a relação entre atraso e resultado desejado que se almeja para o devido projeto. Neste caso, um filtro com muitas interações apresentou comportamento excessivamente atrasado para a estabilidade do sistema, enquanto uma aplicação de filtro com 5 interações apresentou estabilidade e comportamento mais suave do que sem filtragem.

O resultado foi um sinal com saída mais atenuada, mais propício a implementação do controle.

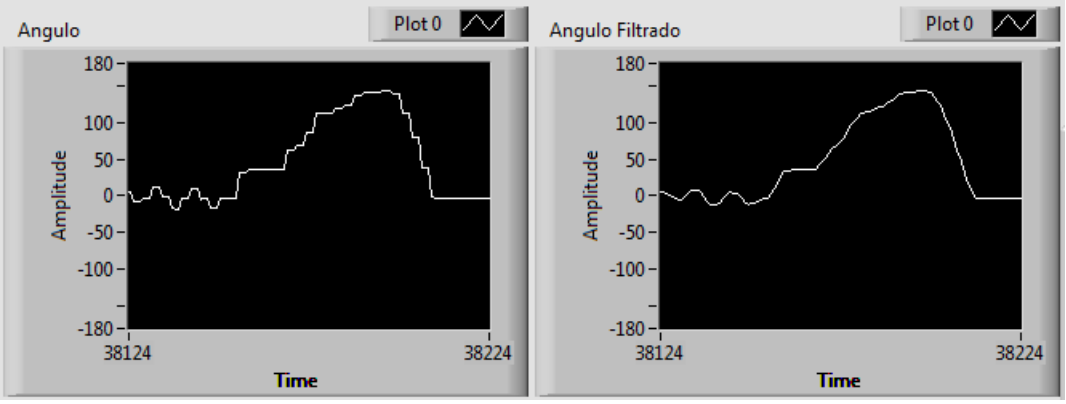

Figura 52: Sinal de saída do ângulo de direção – (a) sem filtro, (b)após aplicação do filtro de média móvel

Como o objetivo nesta etapa é fazer o robô ficar direcionado ao seu alvo, o *setpoint* desejado será sempre 0, visto que o ângulo de direção em relação ao alvo é o ângulo global subtraído do ângulo do alvo em relação ao eixo horizontal do robô. Um *setpoint* de 0 fará o sistema controlar o robô com objetivo de ficar, teoricamente, direcionado exatamente para o destino. A medição do ângulo de direção apresenta valores entre -180 a 180 graus. Apresentando uma não linearidade em sua representação.

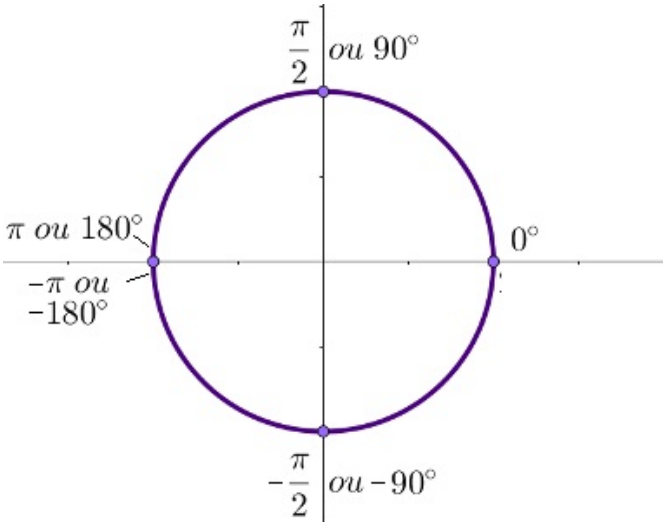

Figura 53: Representação das referências do ângulo de direção, valores vão de -180 a 180

Esta não linearidade representa um empecilho para a implementação do controlador PID, que atua idealmente em sistemas lineares. A solução seria uma linearização deste sistema. Ao alcançar o valor máximo de leitura de 180 graus, sua saída passa instantaneamente a -180 graus, acarretando em desproporcional neste instante.

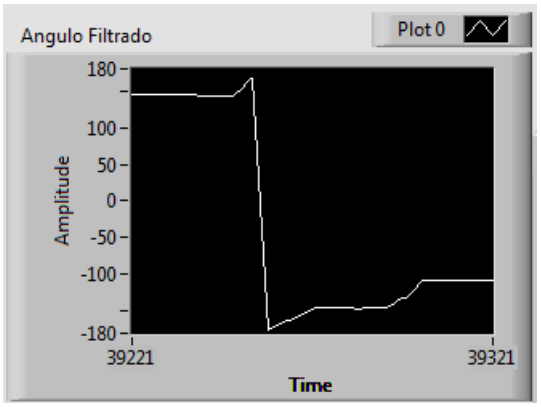

Figura 54: Representação da não linearidade do sistema, salto instantâneo de 180 a -180

A solução seria a linearização do sistema, fazendo com que se analisasse as duas distâncias do erro, adotando o menor valor como parâmetro.

Porém, ao analisar este caso específico do controlador, como o *setpoint* sempre será 0, a as duas possibilidades máximas de erro do ângulo neste valor de transição terão valor absoluto idênticos, não afetando na parte proporcional do controlador.

O mesmo não é verdade para os ganhos restantes. Uma variação instantânea geraria, teoricamente, um valor diferencial infinito, correspondendo a uma atuação desproporcional por parte do componente derivativo. Semelhante ocorre para o componente integrativo, onde o erro acumulado terá adição de um valor inconsistente com o que vinha sendo acrescentado. Para estes dois componentes, adota-se a análise mencionada em cima do valor do erro antes da implementação.

Devido a não linearidade da atuação do motor, quando o mesmo se encontra em repouso é necessária uma saída do controlador PID mínima para que o sistema comece a rotacionar. Uma vez em movimento, esta diferença no valor de saída passa a ter uma sensibilidade consideravelmente maior na proporção de sua atuação, sendo exigido uma limitação do valor de saída máximo para que o sistema não se apresente instável de acordo com a frequência de leitura de seu ângulo de direção. Também foi observado que o valor desta atuação mínima estava sendo elevado com o motor usado originalmente, influenciando o resultado da atuação. Ao se trocar o motor por um de maior torque sua resposta foi aprimorada.

Essa atuação mínima gerava um sinal insuficiente para atuação dos motores, acarretando no clássico problema de *wind-up*, mencionado no capítulo de referência bibliográfica [10]. Levando em consideração que para este projeto específico, não é necessário um erro estacionário nulo, já que este controlador atua somente até uma faixa de ângulo pré-determinada, acionando o segundo controlador PID responsável pelo deslocamento do robô. Uma vez neste segundo estágio de movimentação, com a saída do ângulo limitada, o comportamento esperado é uma diminuição deste erro estacionário, visto que o problema de atuação mínima para acionamento da rotação não ocorre quando em movimento. Porém, quando parado, esta atuação mínima pode ocasionar uma situação que a atuação dos componentes proporcionais e derivativos não sejam suficientes vencer a inércia do robô. O resultado final continua não sendo um erro estacionário nulo, mas este não é o objetivo neste caso. Como é necessário que o sistema entre no ângulo limite de atuação do segundo controlador, é razoável assumir este valor como uma espécie de *setpoint* simulado, de forma que não se tenha erro estacionário em relação a este limite. Sendo assim, foi acrescido um limitador que garante a atuação do componente integrativo somente em uma pequena faixa acima deste ângulo limite garantindo que não ocorra *wind-up* e que o robô não deixe de entrar no modo de movimentação por conta de falta de atuação suficiente [10].

O sistema apresentou um comportamento bastante oscilatório a medida que este controle foi sendo implementada. Para amortecer este efeito, também foi utilizado um componente derivativo.

Os ganhos foram calibrados de forma manual. Aumentou-se o valor do ganho proporcional até o sistema começar a oscilar, incrementando na sequência seu ganho derivativo com objetivo de atenuar as oscilações. Em seguida foram atribuídos limites de intervalo de ângulo em que o componente integrativo seria acionado, assim como um limite no valor de erro máximo que poderia ser acumulado, forçando-se a situação em que o componente integrativo seria necessário para acionamento da atuação, garantindo que o sistema sempre entre nos limites de ângulo para o controlador de distância até o alvo.

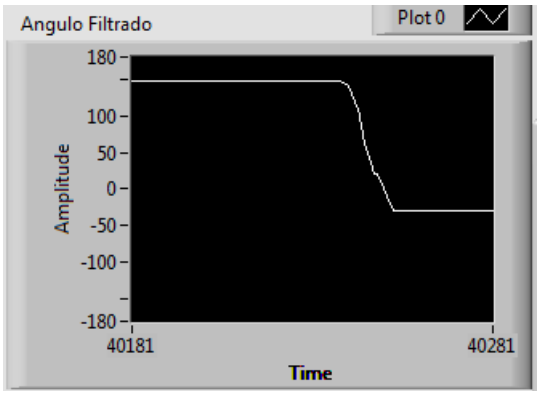

Figura 55: Controle em atuação quando sistema é acionado com robô em direção aleatória

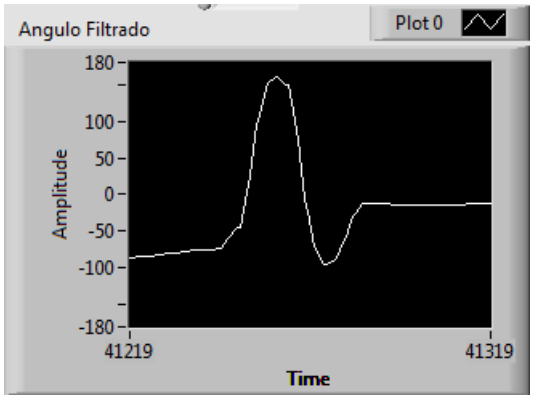

Figura 56: Resultado do PID quando robô sofre impacto e fica em direção oposta a desejada, sendo o pior cenário para o controle.

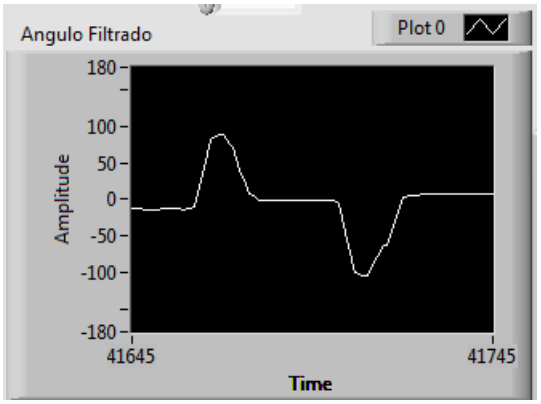

Figura 57: Resultado após sofrer dois impactos que o deixaram em direção aleatória.

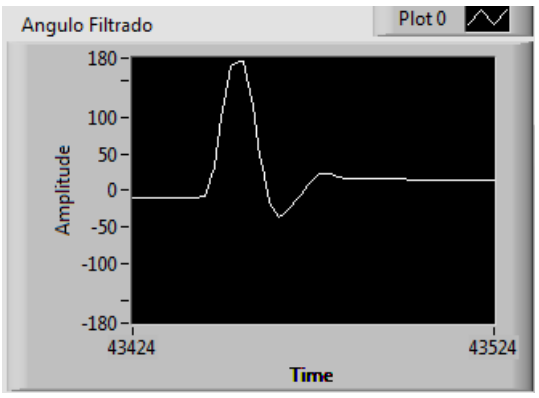

Figura 58: Resultado da saída do ângulo com ambos os sistemas de controle atuando. Controle sobre o ângulo e distância em relação ao alvo simultaneamente.

c Controle da distância até o alvo

Explicitado as condições de acionamento do segundo controle, referente a atuação no deslocamento, sua implementação foi muito mais simples. Como é trabalhado com a distância como sendo um valor absoluto, o robô não andará para trás nesta etapa de atuação. Esta escolha foi proposital, visto que não é interessante uma movimentação retrograda caso o robô erre o alvo. Neste caso, ao passar do alvo, haverá um súbito aumento do erro do ângulo de direção desejado, fazendo com que o sistema corrija esta angulação antes de retornar à movimentação. Na prática, ao invés de andar para trás, o robô corrigirá sua direção e andará para frente.

O *setpoint* para distância teoricamente seria 0 para este caso, mas tanto o robô como o alvo (disco, no caso) possuem suas respectivas dimensões, não se tratando de pontos infinitesimais e sendo aproximados como objetos esféricos para simplificação. Duas alternativas se mostram presentes, estabelecer o setpoint como a distância mínima, em *pixels*, possível entre o alvo e o robô, ou utilizar esta mesma distância como raio de atuação do controlador, fornecendo saída nula quando se encontrar dentro deste perímetro. Desta forma, é atribuído a necessidade de implementação do controle até o alvo estar muito próximo ao robô, passando a ideia de "posse" do disco.

A oscilação para este caso é mínima, sendo exigido um erro estacionário próximo à nulo, de forma que o controle implementado se trata do tipo PI, com ganhos proporcional e integrativo, adicionando-se um parâmetro referente à limitação do erro máximo acumulável pela parte integrativa.

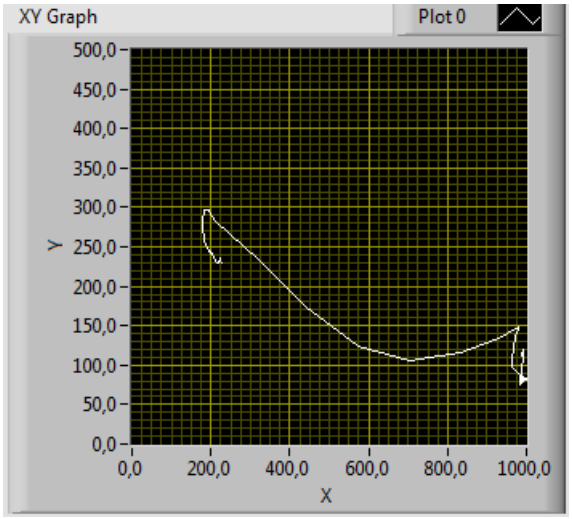

Figura 59: Deslocamento do robô saindo do ponto (980, 150) com alvo no ponto (220, 220)

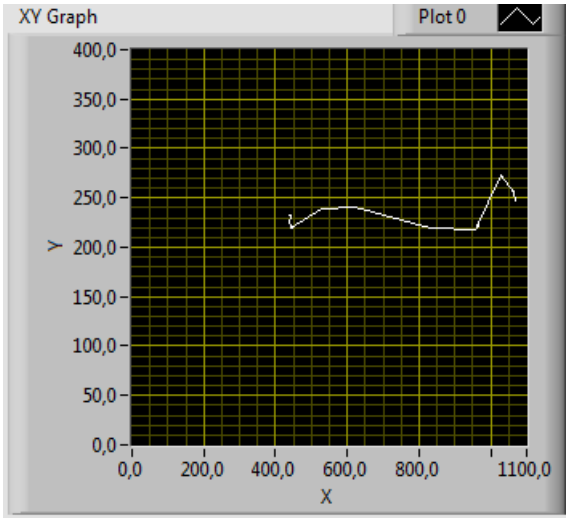

Figura 60: Deslocamento do robô saindo do ponto (1160, 150) com alvo no ponto (450, 250)

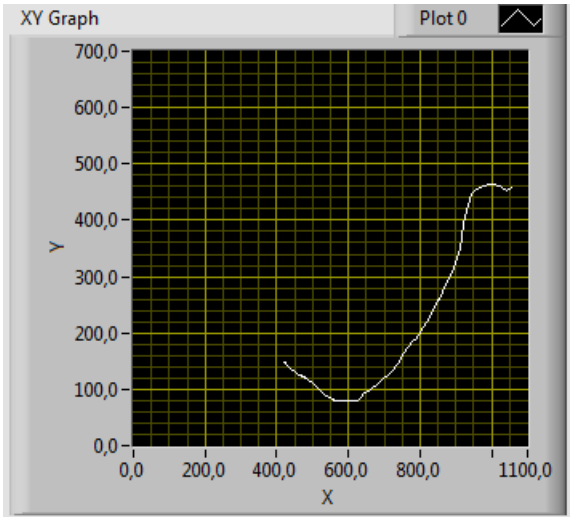

Figura 61: Deslocamento do robô saindo do ponto (420, 150) com alvo no ponto (1150, 450), neste caso robô apresentada direção de origem oposta ao destino.

6 Conclusão

A execução deste projeto comprovou a viabilidade da utilização de visão computacional para desenvolvimento de um sistema de posicionamento para ambientes fechados, atendendo os requisitos de aplicação na robótica desejados para este projeto. O objetivo de rastrear as coordenadas X e Y de um alvo, assim como seu ângulo de direcionamento em relação à uma referência, foram alcançados com êxito, apresentando bons resultados para tempo de processamento e precisão de suas saídas, mesmo para os casos de maior dificuldade, em ambientes de teste pouco controlados e em movimentação consideradas altas para os devidos propósitos.

As alternativas apresentadas ao longo de sua elaboração também se mostraram viáveis para aprimoramento dos resultados, mesmo nos casos em que não foi possível continuar sua implementação por falta de equipamento disponível.

Estas considerações tornaram possível a implementação de controladores PID para executar o deslocamento do robô até o destino desejado com tempo de assentamento considerados satisfatórios para a proposta do robô que jogue hóquei de maneira autônoma.

A calibração deste controle chegou em um ponto de gargalo por conta da pequena área disponível para testes e limitação física da altura da câmera devido ao tamanho de extensão de seu cabo. Sob ponto de vista de desenvolvimento, não havia propósito em aprimorar os ganhos e parâmetros implementados no controle para refinar sua resposta na área disponível para testes que estava longe de ter dimensões aceitáveis para a simulação do jogo proposto.

Infelizmente não foi possível implementar a localização do alvo até o prazo final deste projeto, sendo os resultados obtidos utilizando coordenadas pré-determinadas para simular o alvo, tendo os objetivos sido alcançados nestas condições. Utilizando as referências luminosas de identificação única dos objetos, pretendia-se utilizar uma combinação única, especificada no programa, para reconhecimento do alvo, utilizando os mesmos procedimentos descritos ao longo do projeto para rastrear as coordenadas deste alvo, usando estes valores de saída como posição para o disco.

a Implementações futuras

Meu principal sentimento ao fim deste projeto é a expectativa de ver esta implementação em uma área com dimensões apropriadas para uma partida de hóquei com vários robôs simultaneamente, o que pretendo providenciar assim que possível. Uma vez em um ambiente apropriado, trabalharei mais uma vez em cima dos ganhos e parâmetros dos controladores PID, de modo a tornar a resposta do robô a mais rápida possível sem desestabilizar o sistema.

A utilização do *LabView* foi devido a praticidade do sistema para implementações na área de automação, assim como o já conhecimento de minha parte do programa. Pretendo implementar o algoritmo do sistema de posicionamento descrito em outras linguagens open source, como o OpenCV [21], com o fim de verificar o quanto é possível melhorar o tempo de processamento do programa.

Por fim, também gostaria de implementar no controle um sistema de previsão da trajetória do alvo, com a finalidade que o robô fosse possível interceptar o disco durante sua trajetória, prevendo a provável posição em que o mesmo estaria ao final de seu curso.

Referências

[1] *Introduction to digital filters*, (acessado 17 de dezembro, 2018). [Online]. Available: http://123.physics.ucdavis.edu/week_5_files/filters/digital_filter.pdf

[2] *PWM and PPM difference and conversionl*, (acessado 17 de dezembro, 2018). [Online]. Available: https://oscarliang.com/pwm-ppm-difference-conversion/

[3] National Instruments, "Labview help."

[4] *12V 1000RPM Gear Motor*, (acessado 11 de dezembro, 2018). [Online]. Available: https://www.mepits.com/product/210/dc-motors/12v-1000rpm-gear-motor

[5] *Tensões de referência de uma bateria 3S*, (acessado 08 de dezembro, 2018). [Online]. Available: http://blog.droneng.com.br/baterias-lipo/

[6] *Arduino Nano*, (acessado 09 de dezembro, 2018). [Online]. Available: https://store.arduino.cc/usa/arduino-nanor

[7] Wikipedia, *Sistema de posicionamento global*, (acessado 9 de dezembro, 2018). [Online]. Available: https://pt.wikipedia.org/w/index.php?title=Sistema_de_posicionamento_global&oldid=53161048

[8] A. V. Oppenheiml and A. S. Willsky, *Sinais e Sistemas*, 2nd ed. Pearson Education do Brasil.

[9] *What is Serial Communication and How it works?*, (acessado 17 de dezembro, 2018). [Online]. Available: https://www.codrey.com/embedded-systems/serial-communication-basics/

[10] A. Baetica, *Integrator windup and PID controller design*, (acessado 17 de dezembro, 2018). [Online]. Available: https://www.cds.caltech.edu/~murray/wiki/images/6/6f/Recitation_110_nov_17.pdf

[11] D. Engineering, *Sabertooth 2x12 Datasheet*, (acessado 8 de dezembro, 2018). [Online]. Available: https://www.dimensionengineering.com/datasheets/Sabertooth2x12.pdf

[12] *What's The Difference Between Brush DC And Brushless DC Motors?*, (acessado 15 de dezembro, 2018). [Online]. Available: https://www.electronicdesign.com/electromechanical/what-s-difference-between-brush-dc-and-brushless-dc-motors

[13] M. A. Meggiolaro, *RioBotz Combat Robot Tutorial*, 2nd ed.

[14] R. Bianchi, "Visão computacional aplicada ao controle de micro robôsk," Master's thesis, FEI, São Bernardo do Campo, 2001.

[15] A. M. G. Tommaselli, J. K. Hasegawa, and M. Galo, "Modernas tecnologias de aquisição de imagens em fotogrametria," *Boletim de Ciências Geodésicas*, 2000.

[16] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Pearson Education, Inc.

[17] *What Is The Difference Between Gain And ISO?*, (acessado 11 de dezembro, 2018). [Online]. Available: https://www.videouniversity.com/articles/what-is-the-difference-between-gain-and-iso/r

[18] *Histograma de imagem digital*, (acessado 12 de dezembro, 2018). [Online]. Available: http://computacaografica.ic.uff.br/transparenciasvol2cap3.pdf

[19] *Limiarização*, (acessado 12 de dezembro, 2018). [Online]. Available: https://sites.google.com/site/imgprocgpu/limiarizacao

[20] V. I. A. V.I. and S. Novikov, *Dynamical Systems VII - Integrable Systems, Nonholonomic Dynamical Systems*. Springer-Verlag, New York, vol. 16.

[21] Intel, *OpenCV*, (acessado 12 de dezembro, 2018). [Online]. Available: https://opencv.org/

A Apêndice A: Especificação dos modos de operação da *Sabertooth*

Options: All options are set by using the option switches.

	Mixing Mode: With switch 1 in the UP position, the controller is in mixed mode. In this mode the signal sent to the FWD channel controls the forward/back motion of both motors, while the Turn channel controls the difference in speed between the two motors. Switch 1 in the DOWN position (shown) allows control of the motors independently.
	Exponential: If switch 2 is in the UP position the controller will be in exponential mode. This makes the response less sensitive in the center. This is useful to bring very fast robots under control. If switch 2 is in the DOWN position (shown) exponential is disabled and the response is linear.
	Lithium Mode: Switch 3 in the DOWN position (shown) enables lithium mode, for use with lithium batteries. This will shut the controller down at 3.0 volts per cell, preventing damage to a lithium battery pack. The detected cell count is flashed on the Status 2 LED. Switch 3 should be in the UP position when using NiCd, NiMH or lead-acid batteries.
	Acceleration Ramping: Switch 4 in the DOWN position (shown) turns on acceleration ramping. This will cause the Sabertooth to smoothly change speed over an approximately quarter second. Switch 4 in the UP position means no acceleration ramp is active. Ramping enables smoother control and reduces peak current draw in heavy robots.
	Auto-Calibrate: Switch 5 in the UP position sets auto-calibrate mode. In auto-calibrate mode the neutral position is read at power-up and the end points are automatically detected. Switch 5 in the DOWN position (shown) uses the saved calibration settings. The default saved settings are 1500us center, 1000us min and 2000us max. The saved settings can be changed to accommodate different radios.
	Calibrate and Store: Switch 6 DOWN and Switch 5 UP puts the controller in Calibrate and Store mode. In this mode, the first signals sent to the controller sets the stored center point when using saved calibration settings. The longest and shortest signals sent to the controller on each channel set the endpoints. Turn switch 6 UP before powering down to save the settings.
	Microcontroller Mode: Switches 5 and 5 both DOWN (shown) enables microcontroller mode. In this mode, timeout is disabled and the controller will run at the last commanded speed until a new command is given, using the saved calibration settings. This is useful when running from a microcontroller like a Basic Stamp.

B Apêndice B: Código do *LabView* do reconhecimento dos padrões de referência

```
g if ( (sqrt( ((Rgx - Grx)**2) + ((Rgy - Gry)**2) ) <= ref)  ( Rgx >0 )  ( Grx > 0) )
matchRG = 1;
Rx = Rgx;
Ry = Rgy;
Gx = Grx;
Gy = Gry;
X = Rgx;
Y = Rgy;
Bx = 0;
By = 0;
Drone = 0;
hdgRG = ( atan2 ( ( Rgy - Gry), ( (Rgx - Grx)) ) ) / pi * 180;
else
matchRG = 0;

if ( (sqrt( ((Bgx - Gbx)**2) + ((Bgy - Gby)**2) ) <= ref )  (sqrt( ((Bgx - Gbx)**2) + ((Bgy - Gby)**2) ) >0 ) )
matchGB = 1;
Bx = Bgx;
By = Bgy;
Gx = Gbx;
Gy = Gby;
X = Bgx;
Y = Bgy;
Rx = 0;
Ry = 0;
Drone = 1;
else

matchGB = 0;
if ( (sqrt( ((Brx - Rbx)**2) + ((Bry - Rby)**2) ) <= ref)  (sqrt( ((Brx - Rbx)**2) + ((Bry - Rby)**2) ) > 0) )
matchBR = 1;
Bx = Brx;
By = Bry;
Rx = Rbx;
Ry = Rby;
X = Brx;
Y = Bry;
Gx = 0;
Gy = 0;
Drone = 2;
else
matchBR = 0;
if ( (matchRG == 0)  (matchGB == 0)  (matchBR == 0)  (matchRR == 0)  (matchBB == 0)  (matchGG == 0))
var = 0 ;
else
var = 1;
Xm = X * escX;
Ym = Y * escY;
localRot = hdgRG;
```

C Apêndice C: Código do *LabView* para as saídas dos dois controladores PID

```
dist = (sqrt(((tgtX - navX)**2) + ((tgtY - navY)**2)));
globalRot = (atan ((0 - navY) / (0 - navX)));
tgtHdg = (atan2((tgtY - navY), (tgtX - navX)));

xout = navX; yout = navY;

localRot = localRot * pi / 180;

ch2 = tgtHdg - localRot;
ch2 = atan2(sin(ch2), cos(ch2));
ch2 = ch2 * 180/pi;
ch1 = dist;
```