TRABALHO DE GRADUAÇÃO

SISTEMA AUTOMATIZADO PARA DETECÇÃO DE ALVOS E DISPARO DE PROJÉTEIS DE AIRSOFT

Por,

Kaio Mastelo Marcondes

Brasília, Julho de 2018

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

SISTEMA AUTOMATIZADO PARA DETECÇÃO DE ALVOS E DISPARO DE PROJÉTEIS DE AIRSOFT

POR,

Kaio Mastelo Marcondes

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Prof. Walter de Britto Vidal Filho, UnB/ ENM (Orientador) _____

Prof. Carlos Humberto Llanos Quintero, UnB/ENM _____

Dr. Fernando Merege, ANTAQ _____

Brasília, Julho de 2018

FICHA CATALOGRÁFICA

KAIO, MARCONDES
Sistema automatizado para detecção de alvos e disparo de projéteis de Airsoft ,

[Distrito Federal] Ano.

x, 89p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, Ano). Trabalho de Graduação – Universidade de Brasília, Faculdade de Tecnologia.

1.Airsoft	2.Torreta automática
3.Reconhecimento de alvo	4.Controle servo-visual
I. Mecatrônica/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

MARCONDES, K. M., (2018). Sistema automatizado para detecção de alvos e disparo de projéteis de Airsoft. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 05/2018, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 90p.

CESSÃO DE DIREITOS

AUTOR: Kaio Mastelo Marcondes.

SISTEMA AUTOMATIZADO PARA DETECÇÃO DE ALVOS E DISPARO DE PROJÉTEIS DE AIRSOFT: Concepção, projeto e construção de um robô capaz de reconhecer um alvo e disparar contra o mesmo utilizando uma arma de Airsoft.

GRAU: Engenheiro ANO: 2018

_____

Kaio Marcondes
Rua Alecrim Lote 6 Bloco A Ap. 1703 – Águas Claras
71938-720 Brasília – DF – Brasil.

RESUMO

O presente texto apresenta o projeto e implementação de uma torreta sentinela automática capaz de reconhecimento de alvos e disparo de projéteis de Airsoft. São analisados os requisitos desejados para o sistema, calculadas as especificações necessárias para atingir os requisitos. São apresentados cálculos, análises e experimentos para se definir a faixa de limites físicos operacionais para o robô. São apresentadas algumas técnicas de segmentação de imagem para identificação do alvo e soluções de processamento para as limitações do hardware disponível.
É realizada a implementação do sistema com as modificações necessárias sobre conceito original para a construção de um protótipo, coleta e análise dos resultados.

Palavras Chave: Airsoft, torreta automática, reconhecimento de alvo, controle servo-visual.

ABSTRACT

This current work presents the project and implementation of an automatic sentry turret capable of target recognition and Airsoft projectile shooting. System requisites are analyzed and the specifications for them are calculated. It is shown calculations, analysis and experiments defining physical operational ranges for the robot. It is shown some image segmentation techniques for target identification and processing solutions for the available hardware limitations.
The system is implemented with the necessary modifications over the original concept for a prototype construction, data collection and results analysis.

Keywords: Airsoft; sentry gun; auto turret; target recognition, servo-vision control;

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

A	Área	[m^2]
F_D	Força de arrasto	[N]
v	Velocidade	[m/s]
F_M	Força de Magnus	[N]
J	Momento de inércia angular	[N.m.s^2/rad]

Símbolos Gregos

ρ	Densidade	[kg/m^3]
T	Torque	[N.m]
θ	Ângulo	[rad]

Grupos Adimensionais

C_D	Coeficiente de arrasto
C_L	Coeficiente de sustentação

Subscritos

max	máximo
min	mínimo

Sobrescritos

•	Variação temporal

Siglas

IBGE	Instituto Brasileiro de Geografia e Estatística

CAPÍTULO 1 – INTRODUÇÃO

Este capítulo apresenta informações gerais preliminares relacionadas ao desenvolvimento do projeto. São abordados diferentes aspectos do esporte Airsoft e questões a serem tratadas para a sulução proposta.

1.1 O ESPORTE

Airsoft é um esporte de simulação de combate armado no qual são utilizadas armas não letais que simulam armas de fogo. Este jogo tem várias modalidades como combates em campos abertos, fechados, instalações prediais, infiltrações e vários outros cenários que tentam imitar situações próximas à realidade de operações policiais ou militares.

As armas do Airsoft que foram escolhidas para o desenvolvimento do projeto, são armas de pressão automáticas que disparam projéteis esféricos de plástico. Os projéteis têm 6 mm de diâmetro e seus materiais variam para se obter esferas com massas diferentes.

1.2 MOTIVAÇÃO

A ideia para o desenvolvimento deste projeto, foi inspirada em elementos de videogames que tornam o jogo mais dinâmico e divertido. Um desses elementos é a torreta automática, mecanismo que atira em inimigos automaticamente sem necessidade de operação pelo jogador, também chamadas nos jogos, de *Turret, Autoturret, Sentry, Sentry Gun, Auto Sentry*, entre outros nomes. Na Figura 1 pode ser visto uma dessas torretas como encontrada em jogos.

Figura 1: Sentry Gun do jogo Call of Duty: Ghosts [1]

Este elemento de jogo, torna a experiência do jogo virtual mais interessante em vários aspectos, pode ser usado para dar cobertura, proteger aliados e guardar objetivos sem necessidade de expor o jogador humano. Tendo isso em vista, este projeto busca trazer as mesmas vantagens e dinâmica do jogo virtual, para o jogo no mundo real.

1.3 OBJETIVOS

Este projeto tem como objetivo a implementação de um robô capaz de atuar como sentinela em jogos de airsoft, diferenciar inimigos de aliados, guardar objetivos do jogo, impedir inimigos de avançar em áreas protegidas e proteger a retaguarda dos aliados.

1.3.1 Objetivos específicos

Para identificação de inimigos, será implementado um software de visão computacional capaz de reconhecer possíveis alvos através do vídeo de uma câmera, e a partir dos alvos encontrados, identificar quais deles devem ser considerados hostis.

Ao ser encontrado o alvo a ser “eliminado”, o robô se posicionará utilizando técnicas de controle servo-visual para um mecanismo de movimento *pan-tilt* [2] operado por servo-motores, e acionará o gatilho da arma de Airsoft através de outro servo-motor. É desejado que o robô tenha precisão em seus disparos de modo a acertar nos alvos a maior parte dos projéteis.

CAPÍTULO 2 – REVISÃO BIBLIOGRÁFICA

Este capítulo consta a coleta de informações sobre projetos existentes, estudo de algoritmos para implementação de software e pesquisa de hardware para o desenvolvimento.

2.1 SISTEMAS COMERCIAIS EXISTENTES

Por se tratar de um tipo de equipamento comum em videogames, já houveram algumas iterações para adaptação do sistema para o mundo real. Aqui serão analisados alguns desses casos.

2.1.1 Gladiator II (Project Sentry Gun)

Figura 2: Gladiator II [3]

A torreta “Gladiator II” (Figura 2) é um produto da empresa ProjectSentryGun [3]. É um robô voltado para paintball com funcionamento similar ao que será proposto neste trabalho. O robô “Gladiator II” pesa aproximadamente 27 kg e tem pernas removíveis, utiliza um marcador de paintball semi-automático com acionamento mecânico servo-operado e utiliza servo-motores para movimentação nos dois eixos. O processamento de imagem e lógica de acionamento é realizado através de um computador não-incluso em que o usuário deve instalar o software necessário e conectar ao robô para sua operação.

O robô faz uso de duas portas USB do computador em questão, uma para comunicação com o controlador e outra para aquisição de dados da câmera.

O site vende também separadamente seus controladores e software para o usuário construir sua própria torreta automática. É fornecido gratuitamente o firmware para o controlador que é baseado em Arduino, sendo de fácil adaptação caso o usuário deseja implementar seu próprio controlador. Entretanto, o software necessário para se comunicar com o controlador e realizar o acionamento dos servo-motores é proprietário da empresa e de código fechado.

O mesmo site mostra outros projetos em que são utilizados seus controladores e software para desenvolvimento de diferentes torretas, porém com funcionamento idêntico. Um exemplo disso é a torreta Целуйко Е. (chamado também de “Sentry Eugene”) apresentado na Figura 3, que é um sistema mais robusto e visualmente mais elegante

Figura 3: Robô Целуйко Е. [4]

2.1.2 Heavy Turret (RealSentryGun)

Figura 4: Heavy Turret da empresa RealSentryGun [5]

A empresa RealSentryGun tem uma abordagem um pouco diferente em seu produto “Heavy Turret” (Figura 4). Ele é vendido como um kit de montagem sem arma, bateria ou computador no qual o comprador deve adquirir ou utilizar seus dispositivos já existentes, sendo de paintball, airsoft, bolas de tênis ou qualquer sistema desejado. Neste kit vem toda a parte mecânica, e controlador desenvolvido pela empresa.

Não é especificado o peso da versão atual do produto, mas algumas versões anteriores, o desenvolvedor aborda esta questão dizendo que o peso da versão antiga era de aproximadamente 30 kg e que era necessário diminuir.

O robô funciona com um software proprietário da empresa em que o usuário deve instalar em seu computador para conectar à torreta. É disponibilizado no site gratuitamente, uma versão de demonstração do software para testar. Entretanto não foi possível realizar o teste pois é necessário o controlador vendido pela empresa para seu funcionamento adequado.

O site também vende somente o controlador e software caso o usuário queira construir seu próprio robô.

2.1.3 Mini Turret (RealSentryGun)

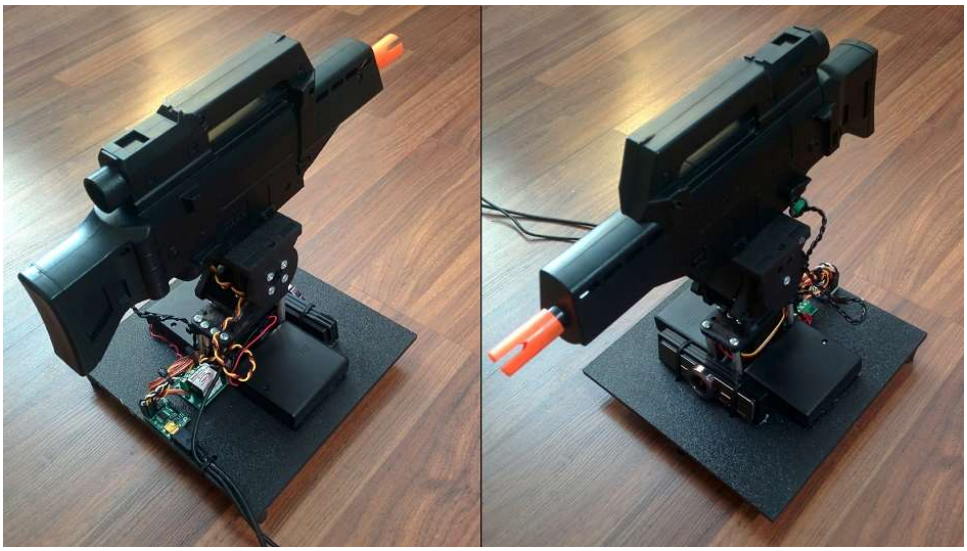


Figura 5: Mini Turret da empresa RealSentryGun [6]

A empresa RealSentryGun também vende uma versão já montada e pronta para uso de seu sistema, chamada de “Mini Turret” (Figura 5). É uma versão pequena e mais leve com arma elétrica modificada de Airsoft.

O usuário precisa apenas adquirir um computador para conectar o robô e poder rodar o programa do “Heavy-Turret”.

2.1.4 Super aEgis II (DoDAMM)

Figura 6: Super aEgis II [7]

A empresa Norte-coreana DoDAAM tem uma solução similar, porém para fins militares com a "Super aEgis II" (Figura 6). Este robô tem a capacidade de travar em um alvo localizado a uma distância de 2,2 km em escuridão total, graças a seus sistemas de mira com laser, visão notura e zoom óptico de 35x. Ele também conta com um sistema de melhoramento de imagem para adquirir o alvo com precisão em tempo ruim.

É utilizado para guardar a fronteira entre as duas Coréias e é preparada para atirar à vista.

2.1.5 SGR-A1 (Samsung)

Figura 7: Samsung SGR-A1 [8]

Similarmente, a empresa Samsung desenvolveu sua própria solução militar (Figura 7) para guardar a fronteira, só que do lado da Coréia do Sul. Portando um sistema de busca de alvos com câmeras infravermelho, laser e visão estéreo, este robô tem um alcance de disparo de 3.2 km e consegue detectar e rastrear os alvos a uma distância de 4 km durante o dia e 2 durante a noite.

Ambos os sistemas (Super aEgis II e SGR-A1) foram projetados para efetuar na Zona Demilitarizada (DMZ), região de 4 km que separa as duas Coréias.

A “SGR-A1”, segundo o fabricante, tem a capacidade de detectar qualquer inimigo que invada aquela região e agir conforme desejado de acordo com as características da ameaça. Ela tenta reconhecer o alvo detectado por reconhecimento de voz, pode soar um alarme para avisar e esperar um código de acesso para desativação, disparar balas de borracha e por fim, disparar com suas armas letais.

Nenhum sistema deste tipo voltado comercialmente para Airsoft foi desenvolvido no Brasil.

2.2 PROCESSAMENTO

Como o robô deverá ter uma central de processamento pequena e que seja capaz de fazer processamento complexo de imagem para identificação de alvos e detecção de movimento, foram analisadas algumas opções de placas de desenvolvimento existentes no mercado. Foram elas, Arduino, Raspberry Pi, Intel Galileo e BeagleBone Black entre outros.

2.2.1 Arduino

Figura 8: Arduino MEGA [9]

As placas de Arduino (Figura 8) são projetadas usando microcontroladores da Atmel de 8 bits para aplicações simples que não exigem processamento intenso e complexo. Por isso foram descartados como central de processamento principal no robô, podendo ser usado secundariamente para acionamento de hardware.

2.2.2 Raspberry Pi

Figura 9: Raspberry Pi 3 Model B [10]

Placas de desenvolvimento Raspberry Pi (Figura 9) são amplamente conhecidas em assuntos sobre internet das coisas, são placas de desenvolvimento baseadas em processadores ARM com alto poder de processamento, podendo ser usadas para as mais diversas aplicações. São placas multi-propósito em que pode ser instalado um sistema operacional de computador desktop, como versões mais completas de linux.

2.2.3 BeagleBone Black

Figura 10: Beaglebone Black [11]

O BeagleBone Black (Figura 10) é semelhante ao Raspberry Pi, melhor e com chip de processamento gráfico 3D. Porém com menor disponibilidade no Brasil e maior preço. Usa a mesma arquitetura ARM.

2.2.4 Intel Galileo

Figura 11: Intel Galileo [12]

Semelhante ao Arduino, utilizando processador Intel Pentium ou Quark de 400 MHz, utiliza arquitetura x86 que facilita a programação. Embora melhor que um Arduino, o Intel Galileo (Figura 11) não é ideal para aplicações de processamento pesado. Foi descontinuado pela Intel recentemente.

Outras placas, processadores e microcontroladores foram analisados. Das placas analisadas, a melhor para o projeto seria a BeagleBone Black, entretanto pela disponibilidade e acessibilidade financeira, foi adquirido um Raspberry Pi 3 Model B para esta aplicação.

2.3 SOFTWARE

Para o desenvolvimento do software, foi escolhido usar OpenCV e Python, pelas suas grandes popularidades e compatibilidades com o Raspberry Pi, centro de processamento escolhido para o projeto.

O OpenCV (Open Source Computer Vision Library) é uma biblioteca de código aberto amplamente conhecida e muito utilizada para aplicações de processamento de imagem e visão computacional. É uma biblioteca otimizada para maior eficiência computacional e com grande foco em aplicações em tempo real [13]. Está em constante desenvolvimento e melhoramento, e por ser de código aberto, de fácil acesso para qualquer usuário desenvolver e explorar cada uma de suas funções.

Foram analisadas outras bibliotecas para visão computacional como VLFeat [14] e VXL [15], entretanto, a quantidade de material online, guias, documentação, tutoriais e projetos sobre OpenCV são muito mais abundantes do que qualquer outra, tornando qualquer solução de problemas que possa vir a se apresentar, muito mais rápida. Além disso, OpenCV tem maior compatibilidade com Python, linguagem de programação amplamente utilizada no Raspberry Pi.

Python é uma linguagem de programação criada para melhorar a inteligibilidade do código e portanto, a velocidade de programação. Python é muito utilizado juntamente com Raspberry Pi para se desenvolver aplicações no âmbito da Internet das Coisas (IoT), para interconexão de dispositivos inteligentes.

Tendo sido decidido hardware, software e a linguagem de programação, o próximo passo foi analisar os métodos para implementação do programa do robô, para rastreamento de alvo e disparo. Para isso, foram analisadas diversas soluções e sua base de funcionamento.

2.3.1 Optical Flow

Optical Flow é um método que tem como base, descobrir como cada elemento de uma dada imagem, se moveu em relação à imagem anterior [16], normalmente é gerado um campo vetorial para análise. Desta forma pode ser descoberto se o referencial ou câmera se moveu e quanto, ou se foi uma secção da imagem como demonstra a Figura 12.

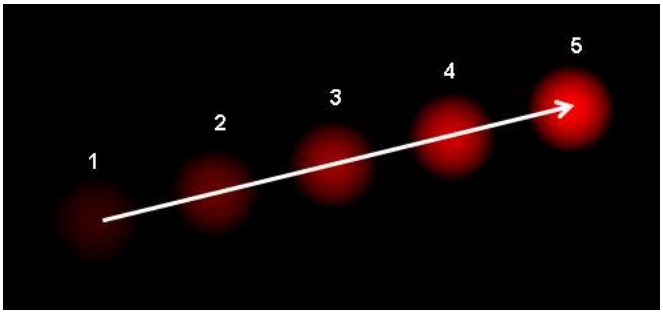


Figura 12: Vetor de movimento de um objeto [17]

Implementado computacionalmente, o Optical Flow trabalha com o pressuposto de que a intensidade de um pixel não vai mudar entre frames consecutivos de um vídeo, isto é, ele terá a mesma cor nos dois frames, e também com o pressuposto de que pixels vizinhos terão movimentos similares.

Considerando um pixel $I(x, y, t)$ no primeiro frame, onde x é a dimensão horizontal, y é a dimensão vertical e t é o instante em que o frame foi capturado. Este pixel se moveu em dx e dy durante um tempo dt, então pode ser escrito pela equação (1)

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \quad (1)$$

Fazendo-se a aproximação por séries de Taylor do lado direito da equação e dividindo por dt, se obtém a equação de optical flow:

$$f_x u + f_y v + f_t = 0;\ f_x = \frac{\partial f}{\partial x};\ f_y = \frac{\partial f}{\partial y};\ u = \frac{dx}{dt};\ v = \frac{dy}{dt} \quad (2)$$

Na equação (2) encontramos f_x, f_y e f_t que são os gradientes da imagem e do tempo, e para encontrar u e v será utilizado o método de Lucas-Kanade [17].

Continuando do pressuposto que os pixels vizinhos se moverão de forma similar, é selecionada uma área de 3x3 pixels ao redor do pixel em análise, de forma que se considera que os 9 pontos tenham a mesma movimentação, têm-se então 9 equações e 2 incógnitas. E por sua vez, a solução final para u e v é obtida como mostra a equação (3).

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \Sigma_i f_{x_i}^2 & \Sigma_i f_{x_i} f_{y_i} \\ \Sigma_i f_{x_i} f_{y_i} & \Sigma_i f_{y_i}^2 \end{bmatrix} \begin{bmatrix} -\Sigma_i f_{x_i} f_{t_i} \\ -\Sigma_i f_{y_i} f_{t_i} \end{bmatrix} \quad (3)$$

E para se obter o campo vetorial inteiro de uma imagem, é executada a subdivisão dela em várias partes pequenas de 3x3 pixels, podendo ser variado o tamanho para maior velocidade computacional ou precisão.

Na Figura 13 pode ser observado um exemplo de funcionamento deste algoritmo implementado utilizando-se as funções do OpenCV, no qual a mão se balança no vídeo, o campo vetorial (u, v) é calculado e mostrado na imagem.

Figura 13: Campo vetorial de optical flow criado com tutorial de Optical Flow da documentação do OpenCV

Com este método podemos detectar se há movimento na imagem e determinar a direção e a velocidade de movimentação de cada ponto na mesma.

2.3.2 Subtração de *Background*

Para o caso de uma câmera estática, o ambiente monitorado tende a continuar sempre o mesmo, com mínimas flutuações de intensidade de pixels ao longo do tempo, majoritariamente devido ao ruído inerente ao dispositivo de captura, ou à mudança nas condições de luminosidade para lugares monitorados em que há presença de iluminação

natural. O conceito de subtração de background é bastante simples, a partir de um frame ou uma média de frames capturados inicialmente, em que não exista a presença de objetos em movimento, é executada a subtração modular pixel a pixel de toda a matriz da imagem presente com a imagem ou média da imagem inicial, desta forma, qualquer objeto que não existia na cena, e que passar a existir, será detectado imediatamente por um resultado não-nulo nos valores dos pixels que marcam sua posição.

Após a subtração dos pixels, e que exista um resultado não nulo, isto é, existem pixels diferentes da imagem original, vários tipos de filtros podem ser aplicados. As flutuações podem ser ignoradas dependendo do tamanho da perturbação, desta forma se ignora pequenos objetos na cena, como por exemplo, um inseto presente que o usuário considera insignificante para seus objetivos. Podem ser ignoradas variações muito pequenas no valor da intensidade do pixel, algo que pode ser causado por ruído presente na imagem ou uma lenta transição de condições de luminosidade, neste segundo caso, uma boa prática seria atualizar a imagem de referência automaticamente com o passar do tempo, uma vez que tenha-se a certeza que não exista um objeto detectado na imagem, um exemplo do resultado deste método pode ser observado na Figura 14 e Figura 15.

Figura 14: Imagem antes de passar pela rotina de subtração do background [18]

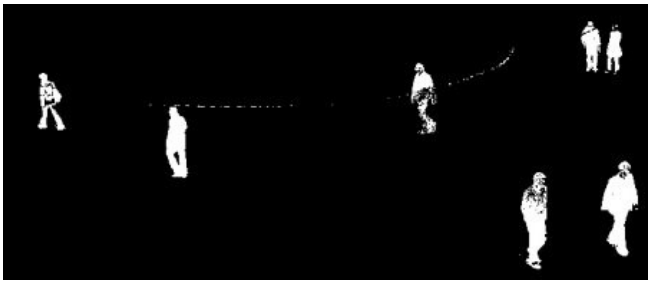


Figura 15: Objetos detectados após subtrair-se a imagem corrente da primeira imagem somente com o plano de fundo [18]

Um dos problemas deste método de subtração do plano de fundo, é sua incapacidade de lidar com câmeras em movimento, caso este que todos os pixels da imagem estariam se movendo, e portanto, a subtração das matrizes seria não-nula para pontos em que não existe real movimento.

Este problema pode ser atenuado combinando-se a subtração de background com o campo vetorial do Optical Flow, que por sua vez, apresentaria um campo inteiro com um deslocamento similar, e desta forma, os pixels a serem subtraídos podem ter suas novas coordenadas ajustadas de acordo.

2.3.3 Rastreamento por cor

Uma das formas mais simples de rastreamento, e possivelmente a única para conseguir detectar adversário e aliado para o caso do jogo de airsoft, é o rastreamento por cor.

O rastreamento por cor separa cada canal de cor da imagem (azul, verde e vermelho) em uma matriz de intensidades de pixels, e para cada matriz, é testado se os valores dos pixels azul, verde e vermelho estão dentro da faixa de valores em que ele pode ser considerado da cor desejada.

Considerando desejado rastrear objetos da cor verde, primeiramente deve-se definir o que é o verde e como o espaço de cores virtual funciona.

2.7.3.1 Espaço de cores RGB

Um dispositivo de captura de imagem, como uma câmera, possui 3 tipos de sensores passa-faixa, um que captura somente a luz vermelha, um que captura a luz verde e outro que captura a luz azul, filtrando as demais frequências de onda.

Uma vez capturada, a imagem é armazenada numa matriz de 3 dimensões, sendo altura e largura, e outra dimensão com os 3 canais de cor, que podem ser consideradas 3 matrizes, uma para cada cor. Este é chamado o espaço de cores RGB (*Red*, *Green*, *Blue*), como pode ser visto na Figura 16.

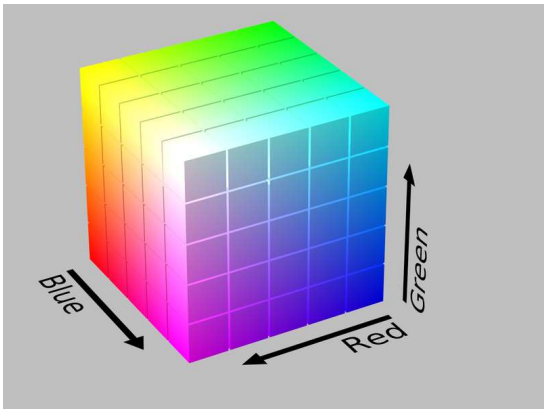


Figura 16: Representação 3D do espaço de cores RGB [19]

Este modelo de cores apresenta problemas claros para o usuário definir uma cor, afinal, para selecionar uma faixa específica, deve-se considerar os valores de todos os canais, afinal,

todos eles contribuem em parte para a cor final. Para resolver este problema, o espaço de cores em questão é convertido para o espaço HSV.

2.7.3.2 Espaço de cores HSV

No espaço de cores HSV (*Hue, Saturation, Value* ou Matiz, saturação, valor ou intensidade), as cores são transferidas de coordenadas retangulares para coordenadas polares, desta forma, para se definir uma faixa de cor, basta se definir a variação do ângulo onde a cor deve ser considerada a cor desejada, diminuindo-se a ocorrência de erros ao se fazer a manipulação simultânea de 3 coordenadas diferentes. O funcionamento deste pode ser visto na Figura 17.

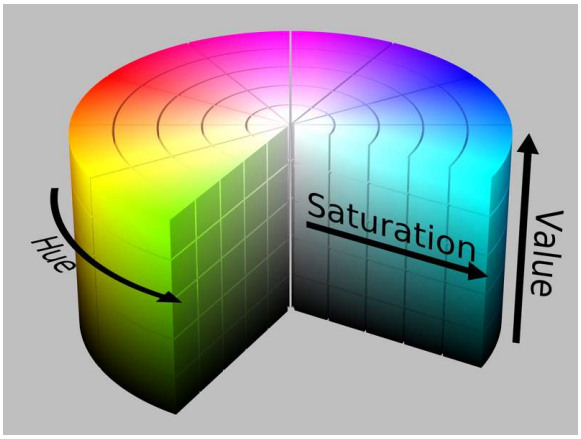


Figura 17: Representação 3D do espaço de cores HSV [20]

2.7.3.3 Transformação dos espaços de cores

O espaço de cores padrão RGB utilizado na maioria dos diferentes tipos de arquivos de imagem e vídeo, incluindo os padrões de DVD e Blu-Ray, até a data de publicação deste, é o sistema “True-color”, que utiliza 24 bits de cores, 8 bits para cada canal de cor Red, Green, e Blue, o que significa 256 níveis de intensidade para cada um dos três canais, totalizando 16.777.216 cores diferentes [21]. Existem sistemas de cores com maiores quantidades de bits, na ferramenta *Adobe Photoshop* por exemplo, pode ser utilizado um espaço de cores de 32 bits por canal RGB, totalizando 96 bits de cores, o que são aproximadamente 8×10^{28} cores diferentes. No entanto, de acordo com diferentes estudos sobre a capacidade de percepção de cores do olho humano, em seu melhor resultado, o olho humano tem a capacidade de distinguir aproximadamente 10 milhões de cores diferentes [22], o que torna o valor de 8 bits por canal, muito mais que o necessário para qualquer aplicação que não envolva frequências de cores além do espectro de luz visível.

Tendo definido o espaço RGB 24-bits como suficiente, deve-se converter o espaço de cores para HSV para ser aplicado no software do robô, para isso, primeiro se divide os valores de cada canal por 255, desta forma, é convertido o valor de 8 bits para um valor entre 0% e 100%

$$R' = \frac{R}{255}; G' = \frac{G}{255}; B' = \frac{B}{255} \quad (4)$$

Em seguida são definidos os parâmetros $C_{max} = \max(R', G', B')$, que é o maior valor entre os canais RGB, $C_{min} = \min(R', G', B')$, que é o menor valor, e $\Delta = C_{max} - C_{min}$ que é a variação dos dois. Após definir esses parâmetros, serão calculados os valores no novo espaço HSV.

Para calcular o valor *Hue* (matiz), é utilizada a seguinte fórmula:

$$H = \begin{cases} 0° & ; \Delta = 0 \\ 60° \times \left(\frac{G'-B'}{\Delta} mod\ 6\right) & ; C_{max} = R' \\ 60° \times \left(\frac{B'-R'}{\Delta} + 2\right) & ; C_{max} = G' \\ 60° \times \left(\frac{R'-G'}{\Delta} + 4\right) & ; C_{max} = B' \end{cases} \quad (5)$$

Para calcular o valor *Saturation* (saturação) é utilizada a seguinte fórmula:

$$S = \begin{cases} 0 & ; C_{max} = 0 \\ \frac{\Delta}{C_{max}} & ; C_{max} \neq 0 \end{cases} \quad (6)$$

E para calcular o *Value* (valor ou intensidade), simplesmente se toma $V = C_{max}$ [23].

Para se calcular a cor **verde** por exemplo, toma-se como verde a cor $RGB = (0, 255, 0)$ e fazendo-se os cálculos, descobre-se que a cor verde $RGB(0, 255, 0) = HSV\ (120°, 1, 1)$, ou

$$H = 120°,\ S = 100\%,\ V = 100\% \quad (7)$$

A tabela a seguir, mostra algumas cores comuns, seu código hexadecimal, que é uma representação binária do valor dos canais RGB, o valor RGB decimal e o valor HSV.

Color	Color name	Hex	(R,G,B)	(H,S,V)
	Black	#000000	(0,0,0)	(0°,0%,0%)
	White	#FFFFFF	(255,255,255)	(0°,0%,100%)
	Red	#FF0000	(255,0,0)	(0°,100%,100%)
	Lime	#00FF00	(0,255,0)	(120°,100%,100%)
	Blue	#0000FF	(0,0,255)	(240°,100%,100%)
	Yellow	#FFFF00	(255,255,0)	(60°,100%,100%)
	Cyan	#00FFFF	(0,255,255)	(180°,100%,100%)

	Magenta	#FF00FF	(255,0,255)	(300°,100%,100%)
	Silver	#C0C0C0	(192,192,192)	(0°,0%,75%)
	Gray	#808080	(128,128,128)	(0°,0%,50%)
	Maroon	#800000	(128,0,0)	(0°,100%,50%)
	Olive	#808000	(128,128,0)	(60°,100%,50%)
	Green	#008000	(0,128,0)	(120°,100%,50%)
	Purple	#800080	(128,0,128)	(300°,100%,50%)
	Teal	#008080	(0,128,128)	(180°,100%,50%)
	Navy	#000080	(0,0,128)	(240°,100%,50%)

Tabela 1: Algumas amostras de cores em diferentes espaços de cores [23]

Subjetivamente, ainda pode ser considerado verde, a cor dentro da faixa de 90° até 150°, permitindo uma variação de 30° para cada direção. Escolhe-se arbitrariamente 50% de saturação e 50% de intensidade e está definido o verde dentro desta faixa, o que seria complicado de fazer no espaço RGB, onde deveriam ser executados múltiplos testes para cada faixa de valores em relação aos outros canais.

Definindo-se a cor desejada para ser rastreada, é executado o teste para cada pixel da imagem, e os pixels que estão dentro dessa faixa de valores serão tratados como objeto-alvo da matriz, como mostra a Figura 18.

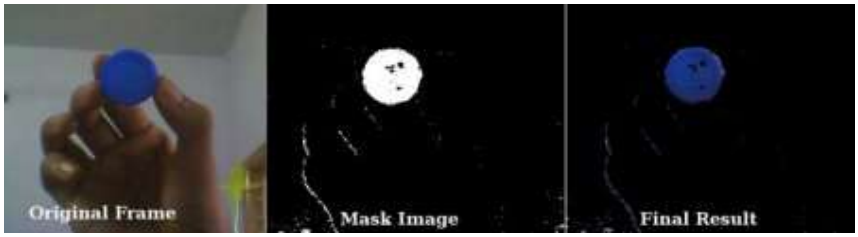


Figura 18: Demonstração de rastreamento por cor [24]

A partir destas abordagens, pode ser criado um código robusto e com grande tolerância a erro, unindo-se os pontos fortes de cada um dos métodos.

CAPÍTULO 3 – CONCEITUALIZAÇÃO

Neste capítulo serão abordados diversos aspectos do projeto, problemas, requisitos e soluções para obtenção do resultado desejado.

3.1 LEVANTAMENTO DE DADOS

Foram feitas algumas visitas em arenas CQC (*Closed Quarters Combat* ou Combate Fechado) de Airsoft. Nestas arenas foram feitos alguns levantamentos sobre a dinâmica do jogo, estes aspectos foram discutidos com os donos da arena e com os jogadores que lá estavam presentes, para assim serem definidos os requisitos do projeto, definir importâncias e ordens de prioridades para serem projetados. A seguir serão explicitados alguns desses elementos não necessariamente em ordem de importância ou de implementação. Sendo que alguns não serão implementados.

3.1.1 Portabilidade

O robô deve ter uma bateria para ser facilmente ativado e realocado (se necessário) entre cada partida do jogo. Ele não deve ficar fixo no mesmo ponto sempre pois os jogadores o irão evitar, diminuindo a área de efetiva da arena e afetando negativamente toda a dinâmica do jogo.

3.1.2 Desativação do Robô ao Receber um Tiro

As opiniões divergiram sobre este assunto, nem sempre é desejado que o robô possa ser desativado. Foram citadas diferentes situações de jogo em que em algumas delas, era melhor ser desativado e em outras, que não fosse desativado. A partir disso, é idealizado um sistema que permita ao usuário que escolha na hora do jogo.

3.1.3 Desativação Remota

Os jogadores não devem ser atingidos ao irem até o robô para realocar sua posição. Ativação/desativação por celular ou controle remoto é o recomendado.

3.1.4 *Override*

Foi sugerido por muitos jogadores que o robô tivesse um modo manual, desta forma, ele poderia ser operado remotamente através de um celular ao invés de ficar apenas no modo de mira automático.

3.1.5 Altura Variável

Por conter obstáculos de diversos tamanhos, os jogadores sugerem que o robô seja desenhado de modo que não consiga atirar através de alguns obstáculos e que possa através de outros, permitindo que os jogadores consigam fugir da torre uma vez que ela engaje em combate.

3.1.6 Delay de Ativação

É desejado que o robô tenha um tempo de resposta programável antes de atirar nos jogadores, deixando o jogo mais justo dando a eles a chance de conseguir se esconder antes do robô travar a mira.

3.1.7 Áudio/Sinalizador

É desejado que o robô emita algum tipo de som característico quando estiver travando a mira em um jogador para o mesmo tentar se esconder antes do robô atirar. Também é desejado que o robô tenha luzes de identificação se está ligado, armado ou desligado.

3.1.8 Identificador de Aliados/Inimigos

Sugere-se que o robô consiga operar em uma equipe, atacando apenas inimigos dessa equipe e consiga também operar em modo “cada um por si”, atirando em tudo que passar em sua frente.

3.1.9 Distância Máxima de Tiro

Para analisar este aspecto, primeiramente analisou-se a geometria geral do campo em questão mostrado na Figura 19.

Figura 19: Vista superior da arena de Airsoft.

Como pode-se observar, a arena tem caminhos com áreas abertas de tiros longos, e áreas fechadas para tiros de curta distância e alta dinâmica de movimento do jogador. Foi-se analisado como seria a dinâmica do jogo com o robô em cada uma das posições. Primeiramente se analisa o robô em um ponto de longo alcance representado na Figura 20.

Figura 20: Vista interior da arena de Airsoft com longo alcance.

Foi feita uma “simulação” com uma câmera nesta posição, devida à grande distância, os jogadores tem pouca chance contra a torre automática, com grandes trajetórias na mesma direção, restringe-se o jogo para o campo “fora da guarda” do robô.

Foi então analisada uma posição mais estratégica com grande rotatividade e velocidade dos jogadores, apresentada na Figura 21.

Figura 21: Vista interior da arena de Airsoft de curto alcance.

Foi unanimidade dentre todos os presentes no local, que o robô deveria cobrir apenas regiões de curta distância com muita dinamicidade de jogadores quando estiver no modo automático. Combinado com o delay de ativação e aviso de mira travada, todos acharam que é o melhor

modo de uso do robô. Houveram algumas opiniões a favor de usar o robô nos pontos de tiros de longas distâncias apenas se estiver no modo manual.

3.1.10 Peso

Ficou claro que o robô não pode ser muito pesado, pois deverá ser carregado por um jogador para dentro do campo ao início de cada partida, mas não pode ser muito leve para resistir a eventuais esbarrões e os tiros que ele possa a vir tomar.

3.1.11 Resistência

O robô tem que ter uma carenagem resistente a tiros que ocorrerão dentro do campo.

3.1.12 Parar de Atirar Quando o Jogador Estiver Fora de Jogo

É desejado que o robô possa detectar quando um jogador é dado como “morto” e pare de atirar, pois uma sequência grande de disparos pode machucar.

3.1.13 Frequência de Tiro

O robô deve atirar pelo menos na metade da frequência de um jogador para compensar pela vantagem da mira automática. Ideal que haja ajuste na frequência de tiro.

3.2 ESPECIFICAÇÕES DE REQUISITOS

3.2.1 Altura do Robô

Para se definir o tamanho mínimo e máximo do robô, foi consultada uma tabela de médias de alturas do IBGE (Instituto Brasileiro de Geografia e Estatística) e obtidos os valores para médias de alturas de brasileiros jovens adultos (idades entre 18 e 25 anos). Segundo os dados do IGBE de 2009, a média de altura do brasileiro é de 1,72 m para o sexo masculino e 1,61 m para o sexo feminino. Após isso, foi utilizada uma figura de proporções humanas (Figura 22) para ser definido o intervalo ideal para o tamanho do robô. Para se fazer este cálculo, primeiramente se considera o sexo feminino para se calcular a maior altura, pois a média de altura das mulheres é menor. Com base na figura, tentando evitar que acidentes ocorram com o cano da arma na altura do rosto, escolhe-se que a maior altura do robô deve ficar a 7/8 da altura média da mulher, obtendo-se um valor de 1,40 m. Depois é definida a altura mínima do robô acima do quadril por questões de conforto ao posicionamento, para isso, é utilizada a altura média do homem, que por ser maior, requer menor deslocamento do corpo. Com base na mesma figura, percebe-se que 4/8 da altura do homem é o tamanho mínimo ideal para o robô, calcula-se então que esta altura mínima deve ser de 0,86 m.

Com isto fica definida que a altura, de acordo com o tópico 2.1.5, deve ser variável dentro do intervalo 0,86 m e 1,40 m.

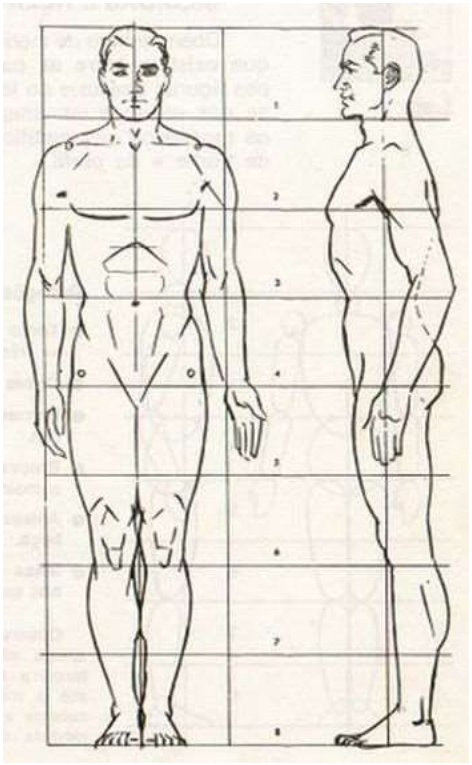


Figura 22: Proporção humana [25]

3.2.2 Peso do Robô

De acordo com a Norma Regulamentadora NBR 17, o peso máximo que que uma pessoa pode carregar em atividades frequentes, é de 18 kg para o homem e 12 kg para a mulher. Estabelece-se então que o robô não deve pesar mais de 12 kg. [26]

CARGAS PARA LEVANTAMENTO (EM Kg)				
	ADULTOS JOVENS		ADOLESCENTES APRENDIZES	
	Homem	Mulher	Homem	Mulher
Raramente	50	20	20	15
Freqüentemente	18	12	11-16	7-11

Fonte: Grandjean, 1980

Figura 23: Carga Máxima de levantamento de peso [26]

3.2.3 Ângulo da Abertura de Giro do Robô

Para se fazer o controle servo-visual, será usada uma câmera USB comercial pela facilidade e preço. Além disso, tem que ser levado em conta como será feita a detecção de movimento do robô, com a câmera fixa, o processamento da imagem é menos complexo, primeiramente então será considerada esta possibilidade.

Webcams comerciais têm entre 50° e 70° de campo de visão, então para uma única câmera fixa, obtém-se uma liberdade de movimento de cerca de 60°, para câmera acoplada no elemento que gira, ele pode ter um campo de visão de até 360°, ficando limitado por outros fatores físicos e estruturais do robô.

Analisando o campo e posições em que o robô seria colocado, é desejado que ele consiga atingir 210° de abertura, que é equivalente a ser colocado próximo a uma parede e sobrar 15° para cada lado como margem de segurança, deste modo é necessário 4 câmeras fixas ou uma móvel no eixo de giro.

3.2.4 Velocidade Angular de Giro do Robô

Uma vez com a mira travada no alvo, o robô não deve perdê-lo, para isso deve-se fazer algumas suposições. Primeiramente, analisa-se a trajetória do alvo, o pior caso possível, é quando a velocidade do alvo tem ângulo mais próximo de 90° em relação à reta paralela à linha de tiro, portanto assume-se o alvo correndo descrevendo uma circunferência de raio constante ao redor do robô. De acordo com a equação de velocidade angular $v = \omega \times r$, com uma certa velocidade v do jogador, quanto menor for o raio da circunferência, maior será a velocidade angular ω que o robô precisará alcançar, toma-se então o menor raio possível para se definir a velocidade angular do robô em relação à velocidade do jogador. Como regra do jogo, um jogador não pode atirar em outro a uma distância menor do que 3 m, esta distância é escolhida para que os tiros não machuquem o outro jogador. Estando definido o raio mínimo permitido no jogo, obtém-se que a velocidade angular do robô em radianos por segundo será 1/3 da velocidade média do jogador em metros por segundo. Sabe-se que a velocidade máxima já registrada de um ser humano, é de 44,72 km/h ou 12,42 m/s [27], então a velocidade angular do eixo do robô para que uma vez travado, ele não perca o alvo, será no pior caso, de 4,14 rad/s ou 237,25°/s.

3.3 EXPERIMENTOS

3.3.1 Velocidade dos Projéteis

Para se medir a velocidade dos projéteis, foi usado um Cronógrafo. O Cronógrafo é composto de 2 sensores ópticos que medem o tempo que o projétil leva para, depois de passar por um sensor, passar pelo outro. A distância entre os sensores é conhecida e dividindo pelo tempo

encontrado, é calculada a velocidade de lançamento do projétil. Um desenho explicativo do funcionamento deste equipamento pode ser visto na Figura 24.

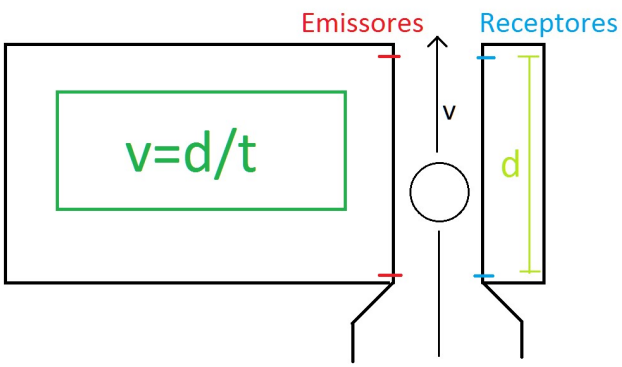


Figura 24: Esquemático de um cronógrafo

O equipamento utilizado foi um cronógrafo da marca Madbull modelo Madbull Chrono v.2 (Figura 25).

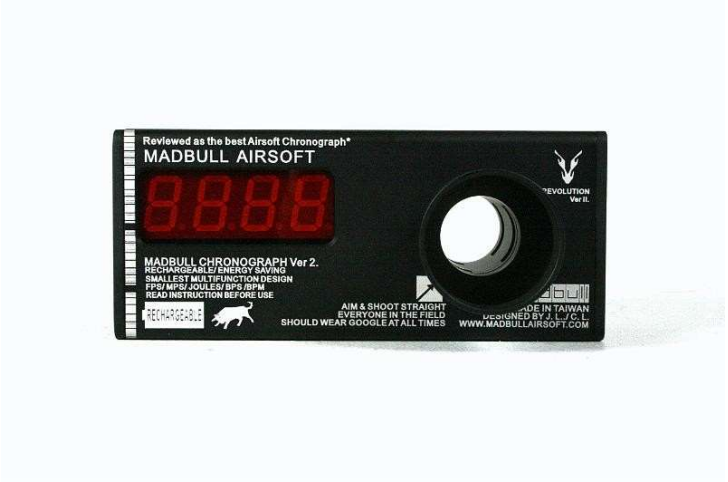


Figura 25: Cronógrafo utilizado para fazer as medidas

Segundo seu manual, este equipamento oferece precisão na medida de <0,001 Pés por segundo (FPS) ou <0,0003048 m/s.

Foram utilizados diferentes projéteis para se fazer o experimento de velocidades de tiro, foram utilizados projéteis de 0,20g e de 0,25g, ambos com diâmetros de 5,945 ± 0,005 mm. Foram feitas 10 medições das velocidades de disparo do projétil em FPS, calculada a velocidade em metros por segundo, calculada a média e desvio-padrão das amostras, como pode ser visto na Tabela 2: Velocidades experimentais de disparo.Tabela 2.

0,20 g		0,25 g	
FPS	m/s	FPS	m/s
341	103.9368	304	92.6592
340	103.632	305	92.964
341	103.9368	307	93.5736
342	104.2416	304	92.6592
340	103.632	304	92.6592
340	103.632	305	92.964
341	103.9368	310	94.488
340	103.632	305	92.964
351	106.9848	304	92.6592
345	105.156	304	92.6592
342.1 ± 3.4785	104.27 ± 1.0602	305.2 ± 1.9322	93.02 ± 0.5889

Tabela 2: Velocidades experimentais de disparo.

3.3.2 Trajetória dos Projéteis

Existem vários tipos de forças diferentes que agem sobre a trajetória do projétil, aqui será falado das mais importantes entre elas [28].

3.3.2.1 Gravidade

Uma das forças mais importantes que regem sobre a trajetória de uma esfera, é a atração gravitacional, a atração gravitacional da terra sobre o projétil pode ser descrita como $F_g = m \times g$, onde F_g é a força-peso exercida sobre o projétio, m é a massa do projétil e g é a aceleração da gravidade no local.

3.3.2.3 Arrasto

A força do arrasto é definida pela equação (8), onde C_D é o coeficiente de arrasto, ρ é a densidade do ar, A é a área da secção transversal da esfera e v é a velocidade da mesma [28].

$$F_D = \frac{1}{2} \times C_D \times \rho \times A \times v\text{^}2 \quad (8)$$

3.3.2.3 Força de Magnus

A força de Magnus é a mais importante para se definir a trajetória do projétil. A força de magnus é a força de sustentação gerada pela diferença de pressão entre as superfícies superiores e inferiores de um objeto que gira [29]. Na parte inferior da esfera, a velocidade do ar em relação à superfície é maior do que na parte superior, fazendo com que a força de Magnus aja no sentido vertical para cima como mostra a Figura 26.

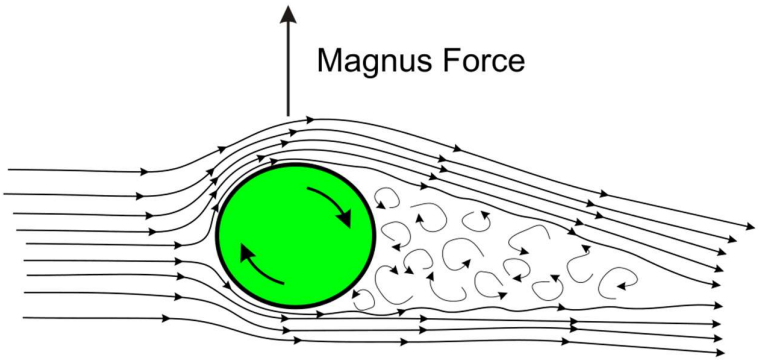


Figura 26: Diagrama da força de Magnus [30]

Para se calcular a força de magnus, usa-se a equação $F_M = C_L \times \rho \times v^2 \times A$ onde F_M é a força de Magnus, C_L é o coeficiente de sustentação, ρ é a densidade do ar e A é a área da secção transversal da esfera [28].

O movimento rotacional do projétil é causado por um dispositivo chamado *hop-up*. Este dispositivo se encontra no início do cano das armas de Airsoft. Seu principal componente é um rolo de borracha de perfil côncavo com o mesmo raio de 6 mm dos projéteis esféricos de Airsoft. Pode ser ajustado para maior ou menor contato com a esfera de modo a aumentar ou diminuir a velocidade angular do projétil este mecanismo pode ser visto na Figura 27.

Figura 27: Mecanismo de *Hop-up* de uma arma de Airsoft [31].

Este mecanismo visa se ajustar a velocidade angular de rotação de modo a se encontrar o ponto ideal onde a trajetória do projétil se mantém retilínea pela maior parte do tempo.

Sem o efeito de compensação causado pelo *hop-up* e o efeito Magnus, o projétil disparado com uma energia de 1,14 Joules segue uma trajetória parabólica como mostra a Figura 28.

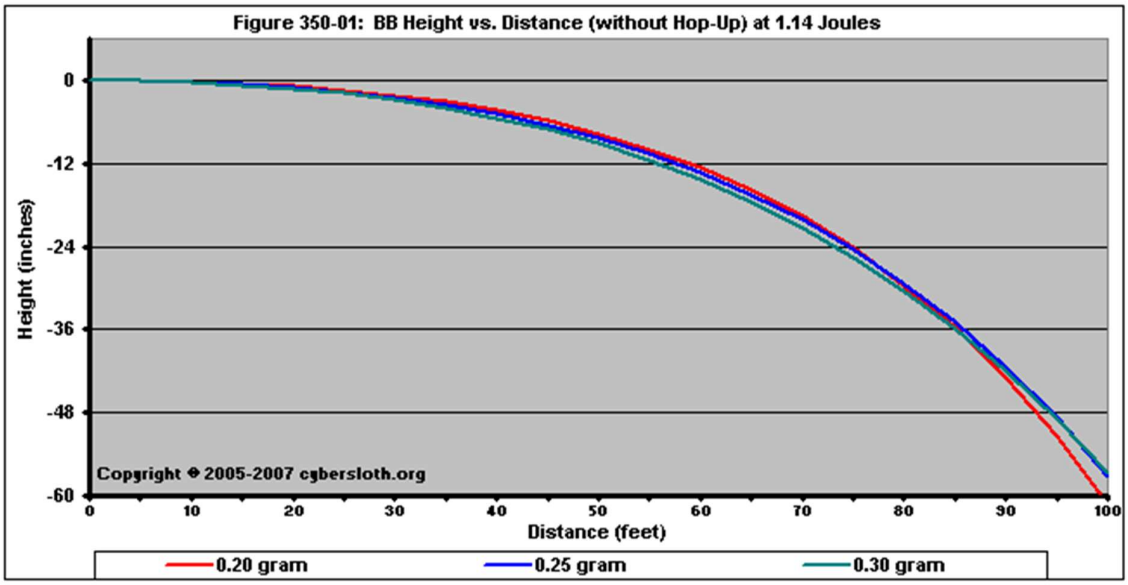


Figura 28: Trajetória de projéteis de diferentes massas sem compensação do *hop-up* [32].

A Figura 29 mostra um gráfico dos mesmos projéteis disparados com mesma energia, porém desta vez, compensados com o efeito da força de Magnus.

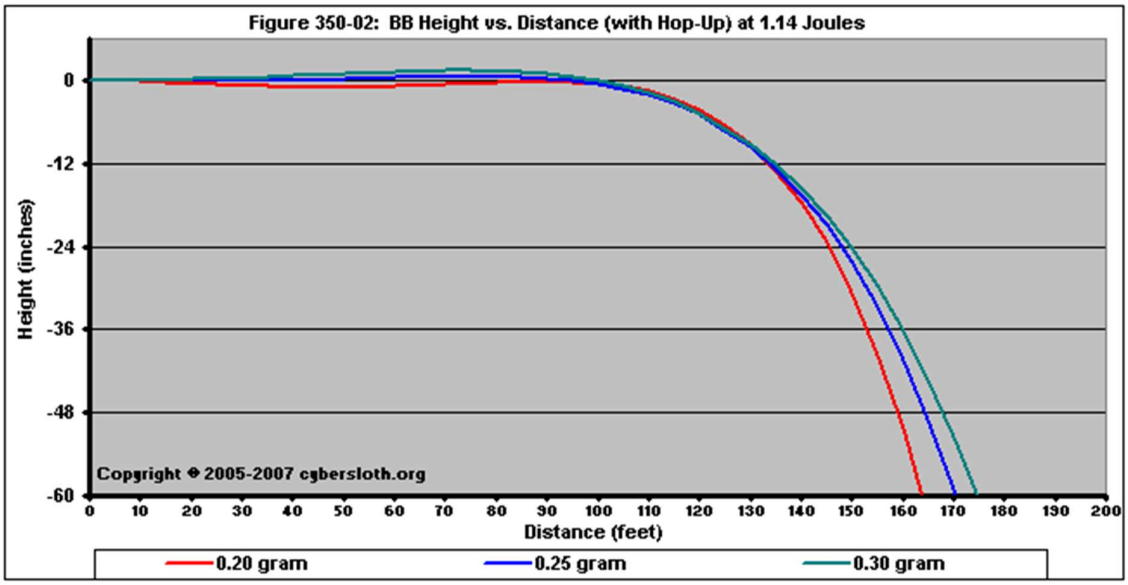


Figura 29: Trajetória de projéteis de diferentes massas com compensação do *hop-up* [32].

Pode ser notado que não somente a força de Magnus aumenta a distância atingida pelos projéteis, como também torna a trajetória quase perfeitamente retilínea durante grande parte de sua trajetória. A trajetória se mantém retilínea até a energia rotacional ser dissipada o suficiente de modo que a força de Magnus deixa de se cancelar com a força de atração gravitacional e a trajetória volta a ser parabólica.

Para o disparo não compensado, o projétil apresenta uma cadência de 12 polegadas (30,5 cm) à distância de aproximadamente 58 pés (17,7 m). Com a compensação da força de Magnus, esta cadência de 12 polegadas acontece à distância de aproximadamente 135 pés (41,1 m), sendo que a trajetória se mantém praticamente retilínea até os 100 pés (30,5 m).

3.3.2.4 Experimento

A partir dos dados encontrados sobre a trajetória, foi idealizado e realizado um experimento para verificar a distância máxima de disparo. Foram realizados disparos utilizando primeiro projéteis de 0,25 g com um anteparo que era deslocado 5m para trás a cada disparo para se traçar a trajetória do projétil. Devida à compensação do efeito Magnus, o projétil segue uma trajetória praticamente retilínea até os 30m, com uma variação de poucos milímetros, o que é desprezível para disparos em alvos de dimensões de pessoas. Após essa marca de 30m, os projéteis entram em turbulência devido ao arrasto do ar e a dispersão da energia de giro, desviando-se então por trajetórias aleatórias.

3.3.3 Centro de Massa da Arma

Foi utilizada uma arma de Airsoft Modelo M4A1 Assault Carbine Full Metal da marca Cybergun, esta arma tem aproximadamente 3,65 kg e deve-se definir o centro de massa para posteriormente ser calculado o momento de inércia de rotação, parâmetro essencial para escalamento dos motores do robô.

Para se calcular o centro de massa da arma, foram feitos experimentos como mostra a Figura 30. A arma foi pendurada em um ponto de apoio, no mesmo ponto de apoio foi pendurada uma massa e foi aguardado a posição de equilíbrio. Com o sistema em equilíbrio, foi traçada uma reta ao longo do fio em que a massa se encontrava pendurada. Após isso o mesmo procedimento foi repetido mudando-se o ponto de fixação da arma e da massa. Através desse procedimento foram marcadas na arma, duas retas concorrentes, o ponto de intersecção dessas duas retas, é o centro de massa da arma.

Figura 30: Experimento para encontrar o centro de massa

Visando calcular como o centro de massa varia à medida em que os projéteis são disparados, foram feitos 4 ensaios diferentes, todos eles com a coronha da arma retraída.

A: Arma com o pente vazio

B: Arma com o pente carregado com 330 projéteis de massa 0,20 g

C: Arma com o pente carregado com 330 projéteis de massa 0,25 g

D: Arma com dois pentes colados um no outro e ambos carregados com 330 projéteis de massa 0,20 g.

Na Figura 31 podem ser vistos os centros de massa encontrados se repetindo este experimento para cada uma das configurações citadas acima.

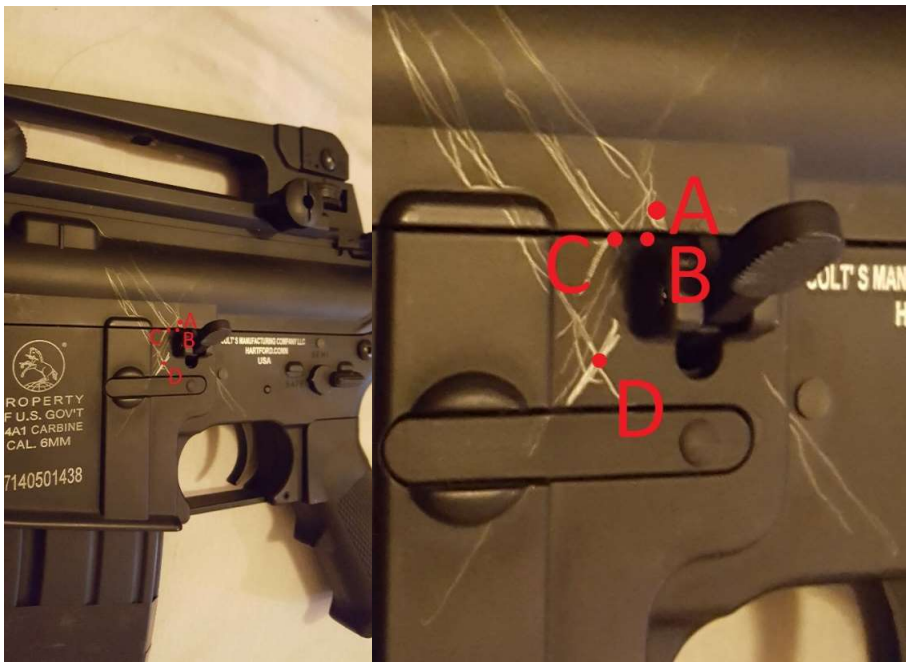


Figura 31: Centros de massa encontrados experimentalmente

3.3.4 Momento de Inércia Angular

Depois de se encontrar o centro de massa, a arma foi fixada pelo centro de massa (Figura 32) para se medir o maior torque que o motor terá que fazer para movimentar a arma com aceleração angular mínima para perseguir o alvo. Para isso, o experimento seguinte foi realizado no caso D do tópico 3.3.3, que é o pior caso para o motor lidar.

Figura 32: Arma fixada pelo centro de massa

Foi feito uma escala de posição angular com resolução de 5° numa cartolina, como mostra a Figura 33. Esta escala será utilizada para medir a aceleração angular com uma câmera de alta velocidade e assim se conseguir definir o momento de inércia de rotação da arma.

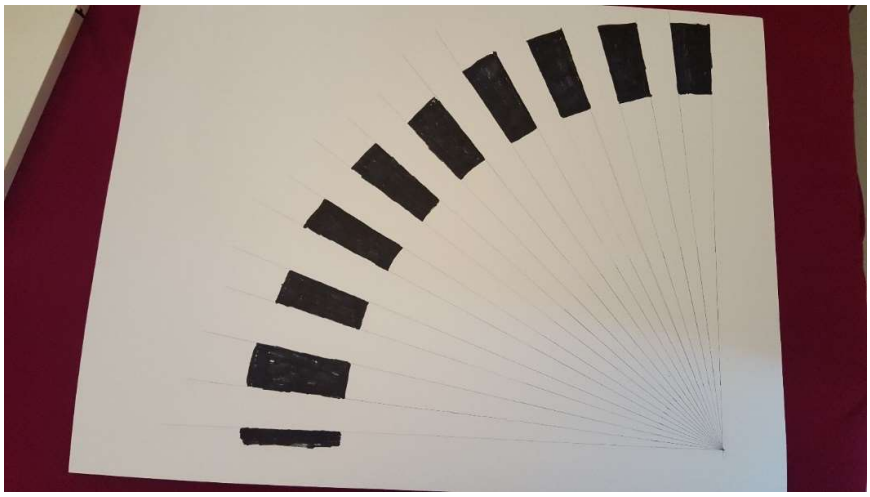


Figura 33: Encoder de posição angular

A câmera utilizada neste experimento foi a câmera do celular Samsung Galaxy S6 edge, que pode gravar a 240 quadros por segundo.

Foi fixada uma massa de 440g na ponta do cano da arma (Figura 34), de modo que ela crie um torque na arma, este torque cause uma aceleração angular, e esta aceleração angular possa ser medida usando-se a câmera de alta velocidade.

Figura 34: Experimento em execução.

Após feitas as filmagens, as mesmas foram transferidas para o computador onde foi rodado o software *Media Player Classic – Home cinema*, este software nos permite passar o filme quadro a quadro e ver o respectivo tempo de cada quadro.

A distância da ponta da arma até o centro de massa da mesma é de 472±0,5 mm. A distância da ponta da arma até o centro da massa é 32,5±0,5 mm.

Portanto a massa gera um torque na arma de (504±10)*10^{-3}*m*g*cos(θ), onde g é a gravidade, m é a massa medida de 440±0,1 g e θ é o ângulo com a horizontal.

A partir das imagens da câmera, pode se calcular a aceleração angular e substituir na fórmula:

$$\mathrm{T} = J \times \ddot{\theta} \quad (9)$$

Obtendo-se:

$$\frac{(504 \pm 10) \times 10^{-3} \times m \times g \times \cos(\theta)}{\ddot{\theta}} = J \quad (10)$$

A partir das imagens da câmera de alta velocidade rodando a 240 frames por segundo, são obtidos os dados apresentados na Tabela 3.

Filmagem 1					Filmagem 2				
Tempo (s	θ (°)	θ' (°/s)	θ'' (°/s²)	J (kgf*cm	Tempo (s	θ (°)	θ' (°/s)	θ'' (°/s²)	J (kgf*cm
2.138	0	-----	-----		1.427	0	-----	-----	
2.255	10	85.4701	-----		1.604	10	56.4972	-----	
2.328	20	136.986	705.702	1.77311	1.684	20	125	856.285	1.4613
2.385	30	175.439	674.602	1.76988	1.743	30	169.492	754.094	1.58331
2.434	40	204.082	584.552	1.88241	1.792	40	204.082	705.921	1.55877
Filmagem 3					Filmagem 4				
Tempo (s	θ (°)	θ' (°/s)	θ'' (°/s²)	J (kgf*cm	Tempo (s	θ (°)	θ' (°/s)	θ'' (°/s²)	J (kgf*cm
2.172	0	-----	-----		1.534	0	-----	-----	
2.269	10	103.093	-----		1.635	10	99.0099	-----	
2.334	20	153.846	780.821	1.60253	1.701	20	151.515	795.534	1.57289
2.386	30	192.308	739.645	1.61424	1.754	30	188.679	701.209	1.70272
2.431	40	222.222	664.767	1.65526	1.8	40	217.391	624.175	1.76291

Tabela 3: Aproximações de valores do momento de inércia angular

Para ângulos maiores que 40°, o centro de massa começa a se deslocar, inviabilizando realização precisa de mais medidas de aceleração.

Para cada filmagem foi feita uma média aritmética dos valores encontrados para o momento de inércia angular. Em seguida foi feita uma média das médias das filmagens, obtendo-se assim, o valor do momento de inércia a ser utilizado no projeto.

3.4 SOLUÇÕES DOS REQUISITOS

3.4.1 Cálculo do torque dos motores de movimentação (*Pan-Tilt*)

Para se calcular o torque necessário para movimento dos motores, considera-se o pior caso, o alvo surgirá na imagem, no limite do campo de visão do robô e o robô deve travar a mira no alvo dentro do tempo desejado. Neste cenário, utilizando a equação $\mathrm{T} = J \times \ddot{\theta}$, São calculados os torques necessários para diferentes campos de visão do robô, que depende do campo de

visão da câmera conectada. É criado um gráfico de correlação de Torque pelo tempo necessário de deslocamento até o alvo, como consta a Figura 35.

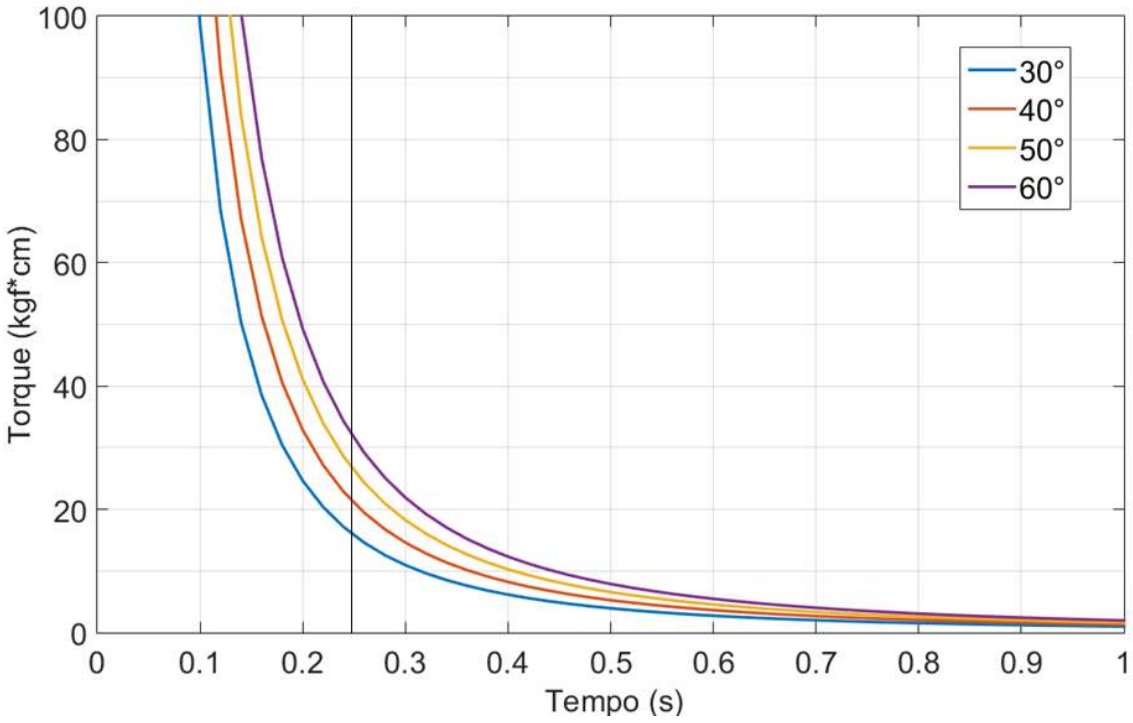


Figura 35: Torque X tempo para diferentes campos de visão

Para se encontrara o torque necessário, deve-se definir o tempo de travamento da mira no alvo desejado. Espera-se que o robô acerte o alvo, portanto ele deve disparar antes que o alvo possa reagir. De acordo com o experimento *How Fast Your Brain Reacts To Stimuli* [33], o tempo de reação médio de um humano a um estímulo visual é de 0,25 segundos, portanto o torque mínimo necessário para os motores reagirem neste intervalo de tempo, se encontra à esquerda da reta vertical tempo=0,25 s (Figura 35). Para selecionar qual curva será selecionada, deve-se saber o ângulo de abertura da câmera, o que será analisado posteriormente.

3.4.2 Acionamento do Gatilho

A forma de acionamento do mecanismo de disparo de uma arma de Airsoft do tipo AEG (*Automatic Electric Gun*), é feito de forma elétrica, o gatilho fecha o contato de um motor elétrico que é acoplado em uma engrenagem parcial, ao girar, esta engrenagem tem seus dentes acoplados aos dentes de um êmbolo de uma câmara de pressão, este êmbolo por sua vez comprime uma mola, e quando a engrenagem parcial termina uma volta, retorna para a posição sem dentes, liberando a mola que comprime rapidamente o êmbolo na câmara de pressão, realizando o disparo do projétil. O esquemático deste dispositivo pode ser visto na Figura 36.

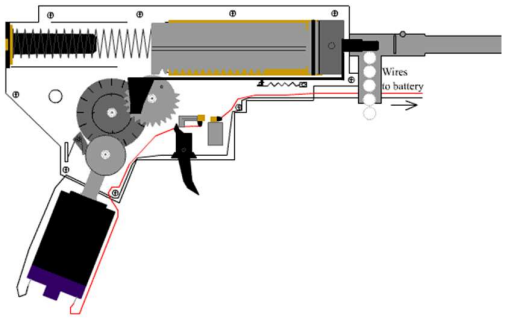


Figura 36: Demonstração de funcionamento de uma AEG [34]

O acionamento do motor pode ser feito utilizando-se um solenóide ou outro tipo de acoplador elétrico, entretanto, para tornar o projeto mais adaptável para os diferentes tipos de AEGs presentes no mercado e evitar quer que abrir a arma em questão para fazer o by-pass do sistema de acionamento elétrico, decidiu-se fazer o acionamento do disparo mecanicamente pressionando o gatilho, para isso, pode ser utilizado relê, servo ou outros mecanismos eletromecânicos. O servo é escolhido pela disponibilidade e por ter resistência a distúrbios, como a vibração da arma, uma vez que o gatilho estiver acionado e disparos estiverem sendo efetuados.

O motor do gatilho poderá ficar acoplado junto do mecanismo de movimentação de dois graus de liberdade da arma.

3.4.3 Mecanismo Pan-Tilt

Para movimentação nos dois eixos de movimento do robô, é elaborado o rascunho do mecanismo Pan-Tilt de movimentação usando o software de modelagem 3D Autodesk 3DS Max. Com este software foi desenvolvido este conceito para que a arma seja afixada por meio de fusos com porcas borboleta, o fuso gira no sentido vertical movido por um motor. O motor que gira o sistema no sentido horizontal, é fixado na parte inferior, e todo o sistema fica acoplado no próprio eixo do motor. Por fim, o servo que aciona o gatilho é fixado no mecanismo e aciona o gatilho por meio de alavanca acoplada no eixo do servo. O modelo bruto deste mecanismo pode ser visto na representação da Figura 37.

Figura 37: Conceito do mecanismo Pan-Tilt e de acionamento do gatilho

3.4.4 Fixação da Câmera

Para fixar a câmera, foi modelado um adaptador para trilhos de 20mm, de modo que aproveite o trilho de acessórios superior da arma de Airsoft como mostra a Figura 38.

Figura 38: Câmera e adaptador para o trilho [35]

3.4.5 Tripé

Para desenvolver o tripé necessário para o robô, foram analisadas algumas considerações.

O robô deve ser leve, fácil de carregar, fácil de mudar de posição, rápido para armar e deve resistir a trombadas de jogadores. Para isso, foram analisadas algumas ideias comerciais encontradas na internet.

Primeiramente foram pesquisados tripés para câmeras fotográficas, binóculos e lunetas, e assim fazer análise de viabilidade, dos vários tripés pesquisados, poucos conseguem suportar mais de 2kg, e os que conseguem, estavam completamente fora do orçamento. Devido a isso, ficou decidido construir o próprio tripé tendo as características desejadas em vista.

Para se analisar as opções de tripé, foram analisados alguns suportes reais para armas mostrados nas três figuras a seguir, suas características de estabilidade e facilidade para construção dentro dos laboratórios e oficinas da UnB.

Figura 39: Exemplo de tripé de um morteiro [36]

Figura 40: Exemplo de tripé de um fuzil de paintball [37]

Figura 41: Tripé telescópico [38]

A partir desses exemplos, foi feita uma mistura de ideias e modificações nos conceitos para preencher os requisitos necessários (Figura 42). Utilizando pés fixos na parte de baixo, diminui-se o risco de queda que é maior com tripés telescópicos, também são escolhidos pés fixos devida à facilidade de realocação rápida do robô. Além disso, uma estrutura com pés fixos, pode ter uma parte telescópica no eixo vertical, fixa com porcas borboletas ou pinos. Tudo pode ser feito de tubos de metal ou de PVC, facilitando o processo de fabricação.

Figura 42: Conceito do tripé modelado

3.5 CONCEITO FINAL

A partir dos conceitos discutidos anteriormente, chegou-se a uma ideia geral de como o robô deverá ser implementado. O protótipo proposto pode ser visualizado na próxima figura.

Figura 43: Protótipo conceitual modelado

CAPÍTULO 4 – PROJETO E IMPLEMENTAÇÃO

Neste capítulo serão demonstrados os processos de construção e desenvolvimento do software do robô.

4.1 COMPONENTES ELETRÔNICOS

4.1.1 Raspberry Pi

Para este projeto foi adquirido um Raspberry Pi 3 Modelo B, que possui processador quad-core ARM Cortex-A53 1.2GHz, memória RAM 1GB LPDDR2 (900 MHz), processador gráfico Broadcom VideoCore IV, capacidade de rede Wi-Fi 2.4GHz 802.11n, 40 pinos de comunicação GPIO, Portas HDMI e USB [10].

No Raspberry Pi foi instalado o sistema operacional oficial Raspbian GNU/Linux 8 Jessie. Dentro dele foi instalada a versão 3.2.0 do OpenCV e o pacote de drivers fswebcam para maior compatibilidade com câmeras USB, recomendado pela documentação do Raspberry Pi [39].

4.1.2 Câmera

Foram testadas 5 câmeras diferentes juntamente com Raspberry Pi, dentre estas, somente uma foi compatível com o sistema, esta câmera foi a Microsoft LifeCam HD-3000, com resolução de 1280x720, taxa de atualização máxima de 30 quadros por segundo e campo de visão diagonal de 68,5° [40]. Com isto, calcula-se o campo de visão vertical e horizontal resolvendo-se $68{,}5^2 = x^2 + y^2$, onde $x = 16r$ e $y = 9r$ devida à razão de aspecto da imagem. Então é encontrado um campo de visão de 59,70° horizontal e 33,58° vertical, informação que será levada em conta na escolha dos motores.

4.1.3 Motores

A partir das especificações da câmera, obtém-se a curva de torque necessária para o dimensionamento dos motores de acordo com a tabela mostrada no item 2.4.1, utilizando-se a maior dimensão, arredonda-se para 60° e obtém-se um torque necessário de 32 kgf*cm.

Tendo o valor do torque mínimo necessário para atingir o requisito previamente proposto, foi realizada uma busca em sites de vendas [41] para pesquisar alguns motores que atendessem ao requisito proposto e suas disponibilidades.

Por acessibilidade de preço, foi escolhido utilizar dois servos motores digitais JX PDI-6221MG por eixo de movimento e um para o gatilho. Este servo possui um torque de 20,32 kgf*cm e velocidade de 0.16 s/60° quando operando em 6 Volts [42], o suficiente para atingir aos requisitos de projeto.

Figura 44: Servo motor JX PDI-6221MG, escolhido para o projeto [42]

4.1.3.1 Operação do servo digital

Servo motores digitais possuem 3 fios, dois de alimentação e um para entrada do sinal de controle. Os fios de alimentação são conectados na fonte, no caso deste projeto, 6V para utilização de seu torque máximo. O fio da entrada do sinal é conectado a um gerador de PWM, que transmite o sinal de controle.

O servo motor é composto por uma placa de controle, um motor de corrente contínua e um sensor de posicionamento. No caso do servo em questão, o sensor de posição é um potenciômetro. O sinal do potenciômetro é convertido para um sinal digital dentro da placa de controle, é comparado com o sinal de entrada, e se a posição não for a mesma desejada de acordo com o sinal de entrada, o motor é acionado. O esquemático de funcionamento do servo digital pode ser visto na

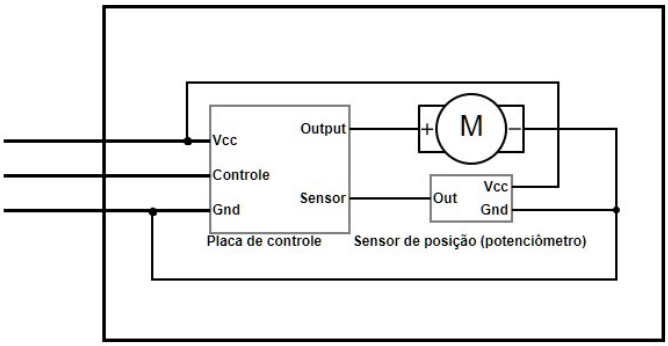


Figura 45: Esquemático do servo motor.

4.1.3.2 Gerador de PWM

PWM (Pulse Width Modulation) ou modulação de largura de pulso é um tipo de sinal digital de frequência fixa e que se controla a largura do pulso, como o nome sugere. É o tipo de sinal de controle utilizado pelo servo digital.

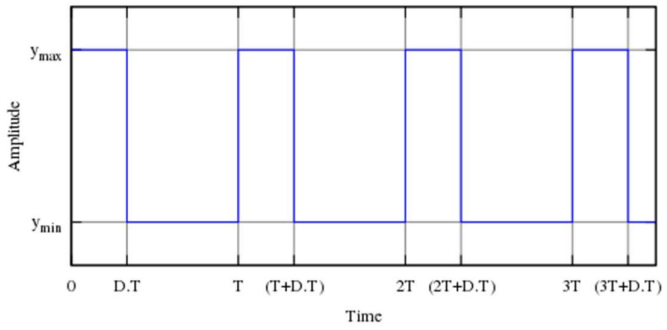


Figura 46: Gráfico de sinal PWM [43]

Este tipo de sinal (Figura 46) é frequentemente utilizado para se simular um sinal analógico usando-se sinal digital e controlando o tempo que o sinal fica na posição máxima e o tempo que fica na posição mínima, ou seja, a largura do pulso. Desta forma, a média do sinal pode ser ponderada de acordo com este tempo e se controlar um componente analógico, por exemplo, a intensidade luminosa de um LED, comumente utilizado para aplicações em Arduino [44].

Entretanto, o servo motor adquirido é digital e não analógico [42], neste caso, o controle de posicionamento não é feito por uma leitura analógica do sinal de controle, e sim pela largura do pulso independente de sua amplitude. Isto cria maior resistência a ruído da rede e permite grande precisão, uma vez que flutuações da tensão não são consideradas.

4.1.3.3 Resolução do servo motor

De acordo com as especificações do servo-motor [42], sua zona-morta inferior é de $2\mu s$, e o servo tem uma faixa de operação entre 500 μs e 2500 μs, com estas informações, pode-se calcular sua resolução, a quantidade de estados distintos possíveis para os possíveis sinais aplicados. Dividindo-se a faixa de operação pela zona morta, obtém-se o resultado de 1000 estados possíveis. E de acordo com as especificações, o servo tem um alcance rotativo de 180°. Sua resolução então é calculada como sendo 50 estados a cada 9°.

Na mesma página é especificada a zona-morta superior como 1520 μs, o que define o limite de frequência de pulsos no qual o servo motor consegue resolver, para aproximadamente 330 Hz.

4.1.3.4 Corrente necessária pelo servo motor

Para testar a corrente máxima consumida pelo servo, foi montado um circuito de testes como mostra a Figura 47. No teste, o eixo do motor foi travado, foi conectado a um amperímetro, por sua vez a um regulador de tensão e uma bateria de 12V que serão discutidos

posteriormente. A entrada do sinal foi conectada ao Raspberry Pi, onde foi simulado um sinal de controle a 100 Hz e largura de pulso de 1000 μs.

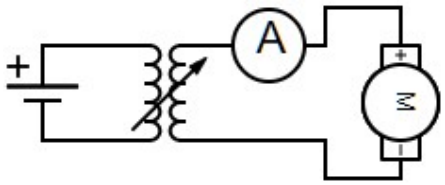


Figura 47: Teste de Corrente do servo.

A partir deste teste, foi medida a corrente máxima de 3,15 A.

Como o Raspberry Pi 3 Modelo B só possui duas portas de PWM nativas [45] e deve-se controlar 5 servos motores simultaneamente, torna-se necessário o uso de uma placa de controle ou simulação de PWM via software em outras portas. Simulação de PWM via software não é a melhor solução pois basicamente é realizada interrupções de sistema baseada na velocidade do *clock* do processador, e é contado o número de ciclos com tensão no nível alto e baixo para se gerar o sinal de PWM. Não é desejado que ocorra tantas interrupções de software pois isso afeta diretamente a velocidade de processamento. E é desejado que se tenha todos os recursos computacionais possíveis para o software de detecção e rastreamento de alvo.

4.1.4 Placa de potência de PWM

Foi adquirida a placa Adafruit PCA8695, mais recomendada pela comunidade dos fóruns do Raspberry Pi para controle de servo motores.

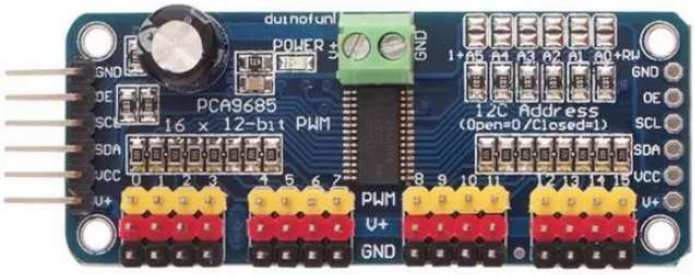


Figura 48: Placa de PWM PCA8695 [46]

A placa PCA8695 (Figura 48) possui 16 canais de 12 bits cada, permitindo a resolução de 4096 valores diferentes para a largura do pulso PWM. Sua biblioteca para controle com o Raspberry Pi é de fácil acesso e uso utilizando a plataforma do GitHub (plataforma de códigos fontes, recursos e programação em comunidade). Foi adquirida a biblioteca na linguagem Python que foi a escolhida para implementação deste projeto [47].

Entretanto, a placa PCA8695 suporta a máxima corrente por canal de 400 mA, muito abaixo dos 3,15 A consumidos pelo servo motor, tornando necessária a aquisição de um conversor de tensão para cada um dos servos e que eles sejam conectados externamente, sem passar pela placa PCA8695.

4.1.5 Reguladores de tensão

Foram adquiridos 6 reguladores de tensão LM2596, um para cada servo e um para o Raspberry Pi.

Figura 49: Regulador de tensão LM2596 [48]

O LM2596 (Figura 49), suporta entrada de tensões entre 3,2 V e 40 V, e saída de tensões entre 1,5 V e 35 V [48] tornando mais que adequado para o projeto.

De acordo com as especificações do LM2596 [49], Ele suporta corrente de até 3 A, por este motivo, foram ajustados para operar em 5,5 V ao invés de 6 V, fornecendo uma corrente de 2,91 A ao ser realizado o teste de corrente do servo novamente.

4.1.6 Bateria

Para alimentar o sistema, foi adquirida uma bateria de chumbo de 12 V, 7 Ah da marca Unipower. Considerando o consumo de 2,91 A máximo de cada servo e o máximo de consumo de corrente de 3 A do Raspberry Pi [10]. Obtém-se a corrente máxima do sistema de 17,55 A, então a bateria consegue alimentar o sistema por 24 minutos de movimentos ininterruptos do robô. O valor real experimentado entretanto, se estende por mais de uma hora, uma vez que os motores não estão em pleno funcionamento o tempo todo.

4.2 ESTRUTURA

A partir do conceito desenvolvido no item 3.4, começa-se o desenvolvimento do protótipo técnico.

Devido a dificuldades encontradas em disponibilidade da oficina do SG-9 da UnB para usinagem dos metais necessários para o desenvolvimento do projeto, foi decidido optar por

projetar uma estrutura de MDF, que pode ser trabalhado com mais facilidade no laboratório do GRACO, UnB.

A única espessura de MDF disponível no momento de desenvolvimento do protótipo no laboratório, foi 9mm de espessura, portanto todo o projeto foi baseado neste.

Foi utilizado o software Autodesk Inventor versão de estudante para o desenvolvimento dos desenhos e modelos 3D.

4.2.1 Eixo de movimento horizontal (*pan*)

Um dos componentes principais da estrutura mecânica do robô, e mais complicado para usinagem é o eixo de movimento horizontal, ele não pode ser feito de MDF como o resto da estrutura do robô e precisa de maior precisão de usinagem para encaixe de rolamento.

Com isto em mente, foi pesquisadas lojas de metais e foi adquirida uma barra cilíndrica de aço de 30 mm de diâmetro e 190 mm de comprimento, então o eixo de movimento do robô foi desenhado de acordo com esta barra.

Após adquirir a barra de aço, foi visitada uma loja de rolamentos para encontrar rolamentos axiais que seriam compatíveis com a barra de aço, o melhor rolamento encontrado, que necessitava menor quantidade de material removida da barra, foi um rolamento axial de esferas com menor diâmetro interno de 25 mm, maior diâmetro interno de 27 mm, diâmetro externo de 47 mm e altura de 15 mm como mostra a Figura 50.

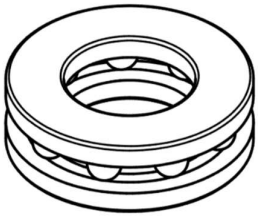


Figura 50: Rolamento axial para o eixo de 30 mm

A partir deste rolamento, foi desenhado o perfil necessário para o eixo de movimento horizontal (Figura 51). Seu desenho técnico se encontra no Anexo 1.

Figura 51: Eixo de movimento horizontal

Foram criados 4 furos com rosca de parafusos M9x1,25 para fixação do eixo na base como pode ser visto no Anexo 1.

4.2.2 Base telescópica

Baseado no conceito original dos requisitos do robô, foi adaptada a estrutura telescópica para ajuste de altura a cada 50 mm como mostram as figuras Figura 52 e Figura 53. Com desenhos técnicos no Anexo 2 e Anexo 3.

Figura 52: Estrutura interna da base telescópica

Figura 53: Estrutura externa da base telescópica

Por fim é desenhada uma plataforma para acomodação dos motores e componentes eletrônicos (Figura 54), e sua fixação na estrutura interna da base telescópica. Seu desenho técnico se encontra no Anexo 4.

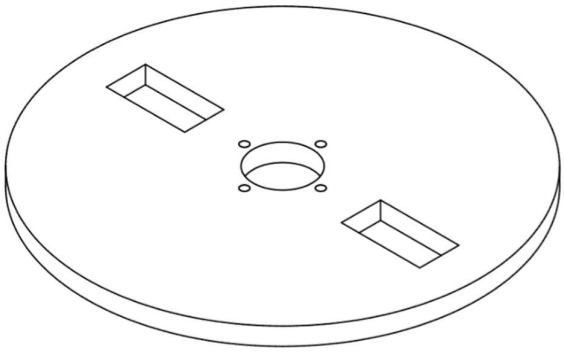


Figura 54: Plataforma de fixação dos motores e componentes eletrônicos

Na Figura 55 pode ser vista toda a base telescópica, a plataforma de montagem dos componentes eletrônicos e o eixo de movimento horizontal.

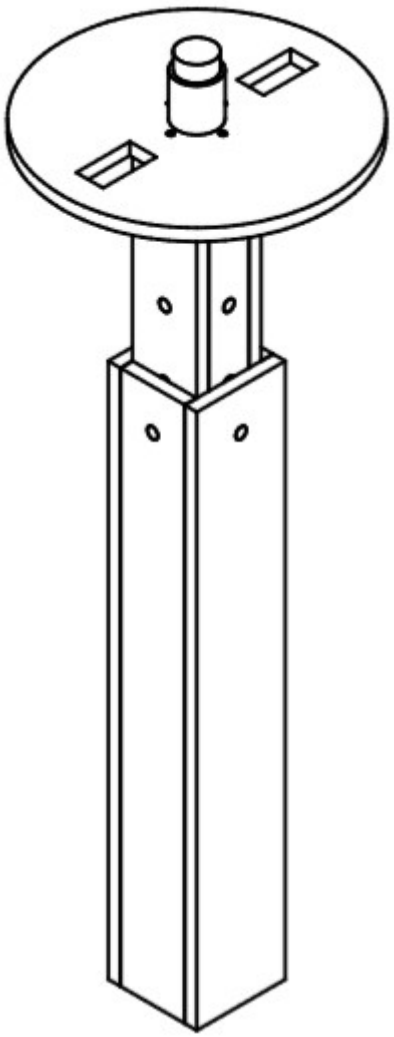


Figura 55: Base telescópica montada com eixo acoplado

4.2.3 Suporte da arma, estrutura *tilt* (movimento vertical)

Neste suporte foi desenhado dois furos de modo a se passar um eixo de cada lado no centro de massa da arma de modo que ela fique em equilíbrio e um furo retangular de cada lado para encaixe do servo motor do gatilho em qualquer um dos dois lados. A estrutura e suas dimensões podem ser vistas na Figura 56 e no Anexo 5.

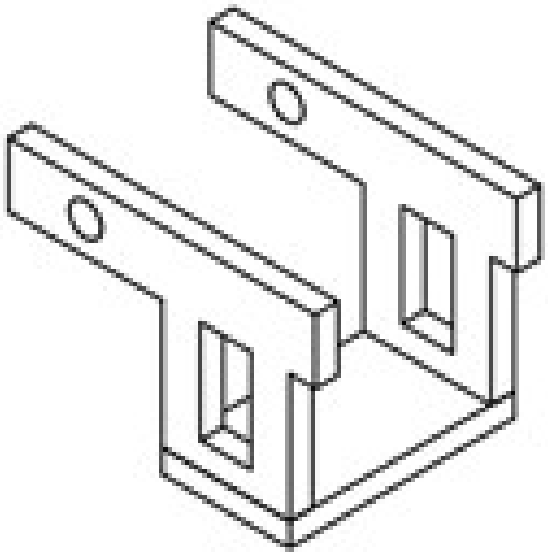


Figura 56: Suporte da arma, estrutura tilt

Para fixação da arma no suporte, foram desenhadas mais três peças para serem fabricadas utilizando-se técnicas de impressão 3D. A primeira delas (Figura 57) vai diretamente na parte inferior do gatilho para a fixação na guarda de dedo, esta peça possui furos sextavados para o encaixe de porcas e furos transpassantes para fixação no MDF através de parafusos. Seu desenho técnico se encontra no Anexo 6.

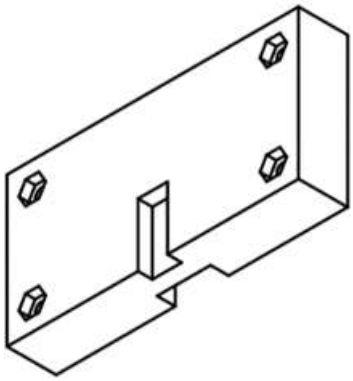


Figura 57: Suporte da arma, fixação inferior

A segunda peça (Figura 58) é fixada na parte superior da estrutura principal de MDF, ela vai segurar a arma por sua parte superior e servir de fixação para a câmera, ela possui algumas áreas abertas para permitir o encaixe da geometria da arma e furos para fixação da câmera através de parafusos. O sulco principal no meio de 12 mm por 21,2 mm servirá de canal de movimentação para a próxima peça, que efetuará a fixação superior. Seu desenho técnico se encontra no Anexo 7.

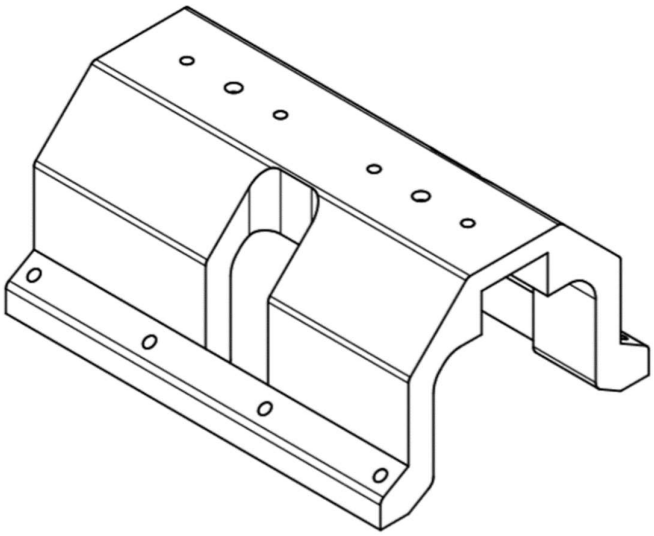


Figura 58: Suporte da arma, suporte da câmera, fixação superior

A peça final (Figura 59) é encaixada no trilho superior da arma e fixada na superior por meio de parafusos que por sua vez, permite o ajuste de altura, desta forma, provendo maior facilidade no encaixe e desencaixe da arma no suporte. Desenho técnico se encontra no anexo 8.

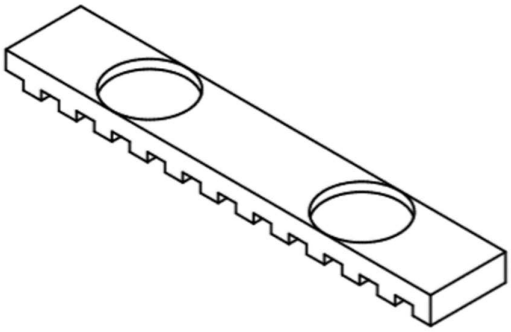


Figura 59: Suporte da arma, fixador no trilho

As duas identações cilíndricas fazem uso dos braços circulares que vem junto do servo motor (Figura 60) como apoio para os parafusos de ajustes, é uma repropositação dos braços do servo que não serão usados.

Figura 60: Braço circular do servo-motor

Para a finalização do suporte da arma, foram encaixados dois eixos de aço de 12,5 mm de diâmetro por 100 mm de comprimento que estavam disponíveis para uso no laboratório e a estrutura de movimentação vertical está completa (Figura 61).

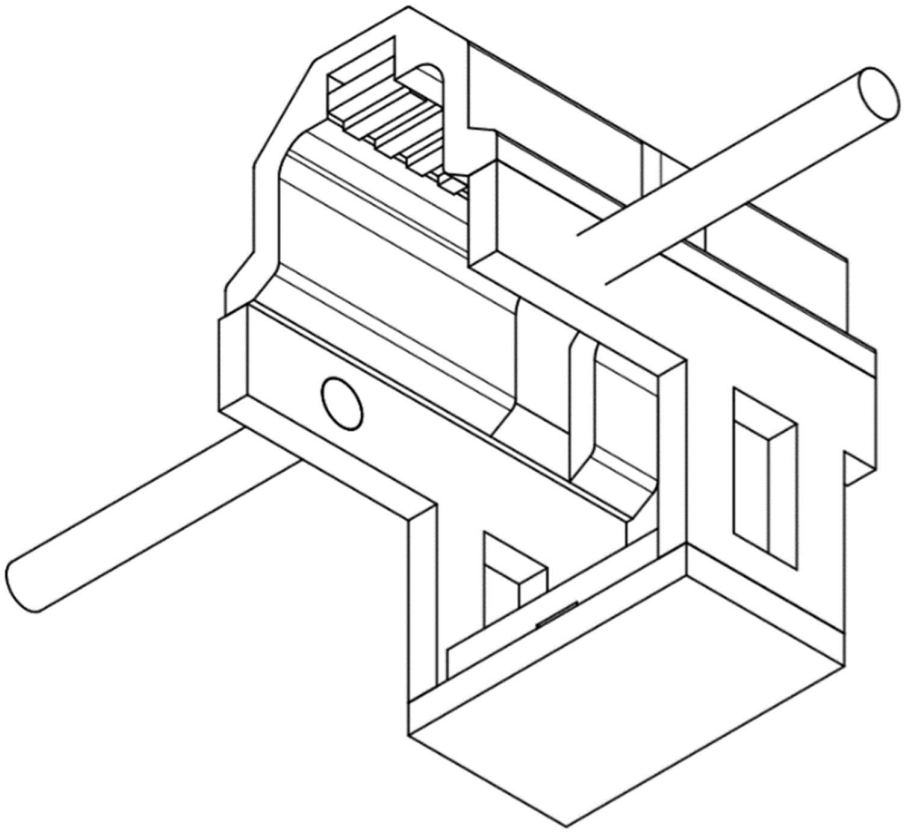


Figura 61: Estrutura tilt completa.

4.2.4 Estrutura *pan* (movimento horizontal)

Mais uma vez, a presente parte a ser discutida teve que ser desenvolvida a partir dos materiais disponíveis. Para os eixos de 12,5 mm, foram facilmente encontrados rolamentos radiais de mesmo diâmetro interno e diâmetro externo de 32 mm (Figura 62).

Figura 62: Rolamentos radiais.

Com base nestes rolamentos radiais, o rolamento axial e o eixo de movimento horizontal foi criado o restante da estrutura de movimento horizontal como mostra o esquemático da Figura 63 e o Anexo 9.

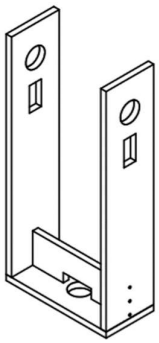


Figura 63: Estrutura de movimento horizontal.

Com a estrutura finalizada como mostra a Figura 64 e sua vista explodida na Figura 65, falta a transmissão de movimento dos servos para seus respectivos eixos, que será discutido no próximo tópico.

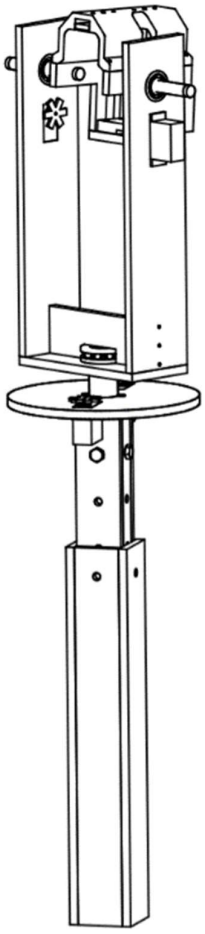


Figura 64: Esquemático da estrutura finalizada com rolamentos e servos.

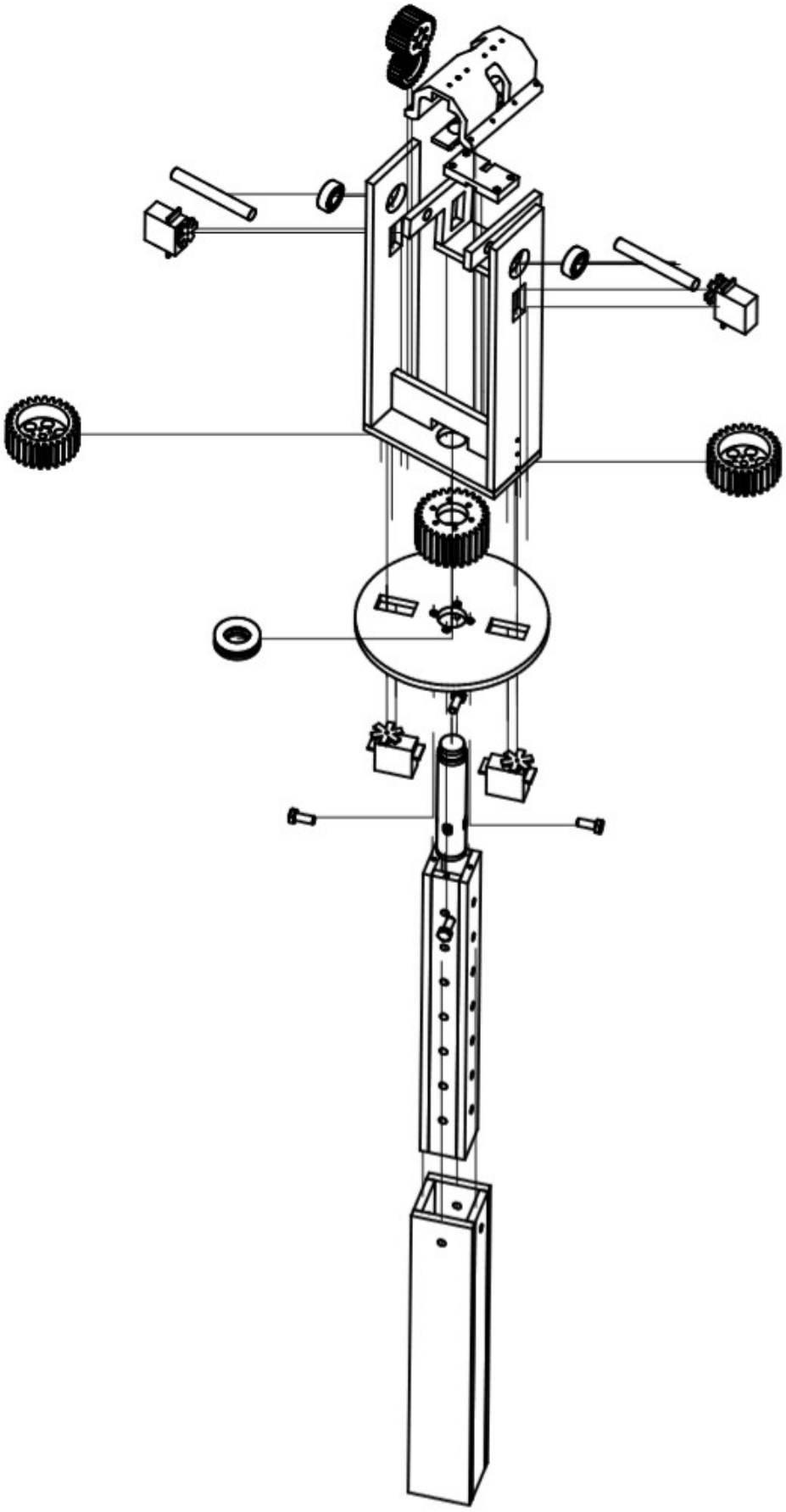


Figura 65: Vista explodida da estrutura do robô.

4.2.5 Engrenagens para transmissão de torque para os eixos

O software Autodesk Inventor possui uma ferramenta embutida para cálculo geométrico de engrenagens de acordo com os parâmetros desejados. No caso deste projeto, a razão desejada das engrenagens é de 1:1, pois os servos possuem torque o suficiente e dispensam a necessidade de redução para aumento do torque, o parâmetro principal que deve ser levado em conta neste caso, é a distância entre os eixos de movimento.

Na Figura 66 pode-se ver uma captura de tela da ferramenta de cálculo de engrenagem do Autodesk Inventor.

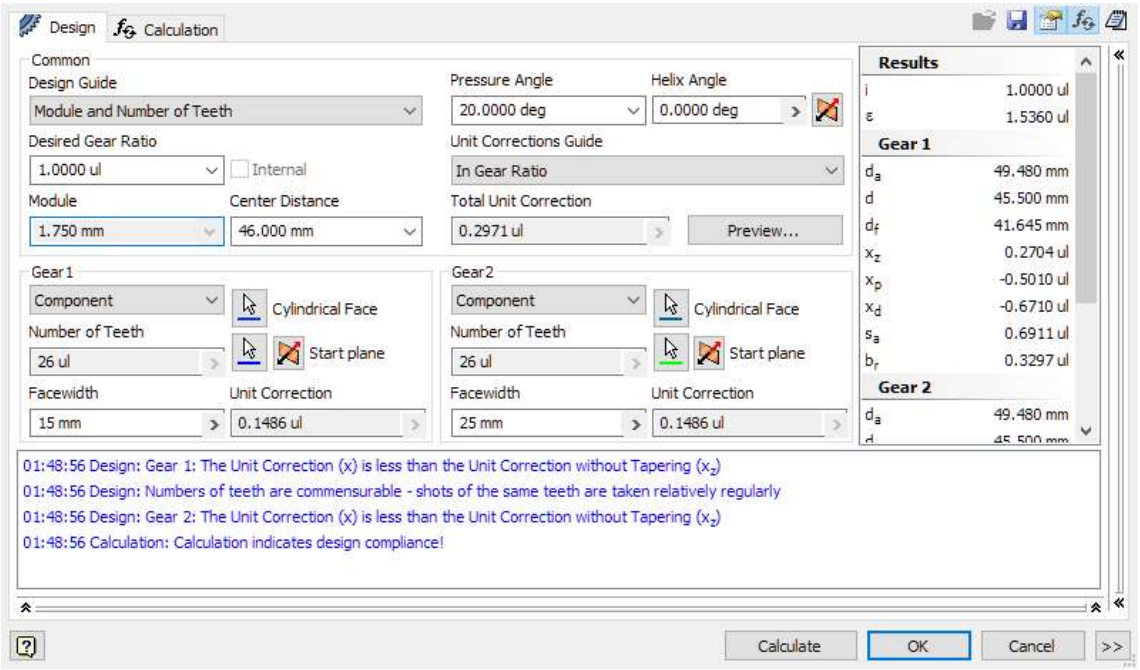


Figura 66: Captura de tela da ferramenta de cálculo de engrenagens do Autodesk Inventor

Para calcular os pares de engrenagens do mecanismo *tilt* (Figura 67), utiliza-se a distância entre o eixo do servo motor e o eixo de aço, que é de 46 mm, é escolhida a razão de 1 entre as duas engrenagens, e o comprimento de 15 mm para a engrenagem que ficará no servo e 25 mm para a engrenagem do eixo. O resto dos parâmetros é calculado automaticamente pelo Autodesk Inventor.

Após o cálculo das engrenagens, os modelos são modificados para fixação no braço do servo e na estrutura *tilt*. Também são modificados de modo a ficarem mais eficientes para impressão 3D.

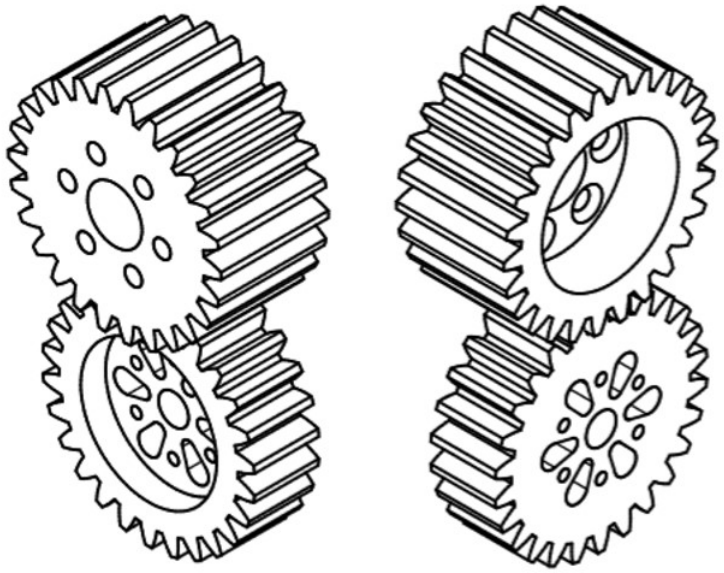


Figura 67: Modelos dos pares de engrenagens para o movimento vertical.

Em seguida são calculadas as engrenagens do movimento *pan* (Figura 68). São projetadas duas engrenagens em contato com uma engrenagem central acoplada ao eixo. A distância entre os eixos dos servos e o eixo horizontal é de 70 mm, a razão entre as engrenagens é de 1, o comprimento da engrenagem do eixo é de 40 mm, o comprimento das engrenagens dos servos é de 30 mm.

Após o calculo das engrenagens, da mesma forma que foi feito com as engrenagens *tilt*, os modelos são modificados para impressão 3D e fixação no servo e no eixo.

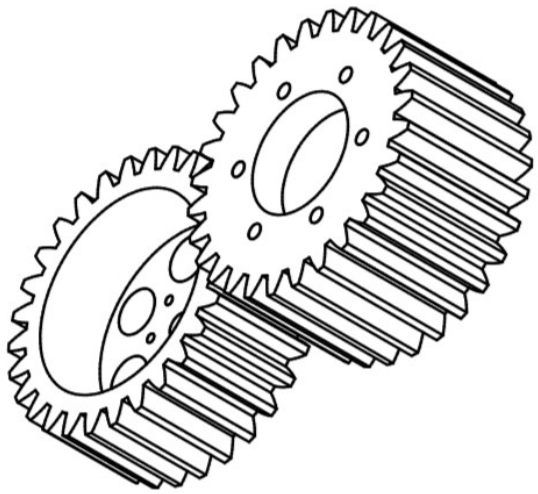


Figura 68: modelos das engrenagens para o movimento horizontal.

4.2.6 Estrutura finalizada

Com todos os componentes mecânicos calculados e colocados em seu devido lugar, é renderizada uma imagem 3D da estrutura finalizada, que pode ser vista na Figura 69.

Figura 69: Renderização da estrutura do robô

4.2.7 Montagem do protótipo

Estando satisfatório o modelo 3D do robô, é realizada a montagem final do protótipo, a arma de Airsoft, o Raspberry Pi, placa de potência e reguladores de tensão são colocados em seus devidos lugares segundo o circuito elétrico mostrado na Figura 70.

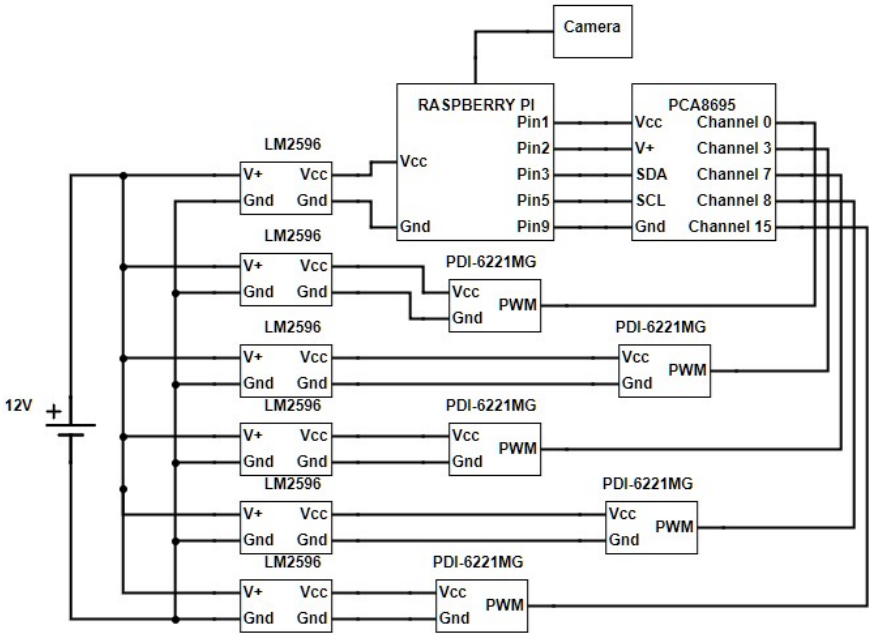


Figura 70: Diagrama eletrônico do robô

4.2.7.1 Fixação da câmera

Para a fixação da câmera e prevenção de ruído na imagem devido à vibração da arma de Airsoft enquanto estiver atirando, foi utilizado um suporte com amortecedor de câmera para drone [50]. O suporte foi parafusado no topo da peça de fixação superior e o resto dos componentes encaixados e parafusados na estrutura, como mostra a Figura 71.

Figura 71: Montagem do protótipo finalizada.

Na Figura 72 pode ser observada a parte frontal do robô, onde se apresenta o Raspberry Pi, as engrenagens do movimento *pan* e as conexões do Raspberry Pi com a placa PCA8695.

Na Figura 73 pode ser observada a parte superior do robô, onde são apresentadas as peças de fixação superiores, o suporte da câmera com amortecedores parafusado na peça de fixação superior, os parafusos que prendem o cabo da câmera e os parafusos de ajuste da peça que se encaixa no trilho.

Na Figura 74 pode ser observada a peça de fixação inferior da arma na estrutura *tilt*, que é posicionada entre o gatilho e o guarda-mato e fixada na estrutura *tilt* por meio de parafusos.

Na Figura 75 pode ser observada a parte traseira do robô, onde se apresenta a placa PCA8695, o circuito LM2596, sua fixação na plataforma, conexões com o Raspberry Pi e seus fios de conexão com a bateria.

Na Figura 76 pode ser observada a parte inferior da plataforma de componentes, onde são apresentados os servo-motores e suas posições de fixação, as posições de fixação do circuito LM2596 e suas conexões com os servos e os fios de conexão com a bateria.

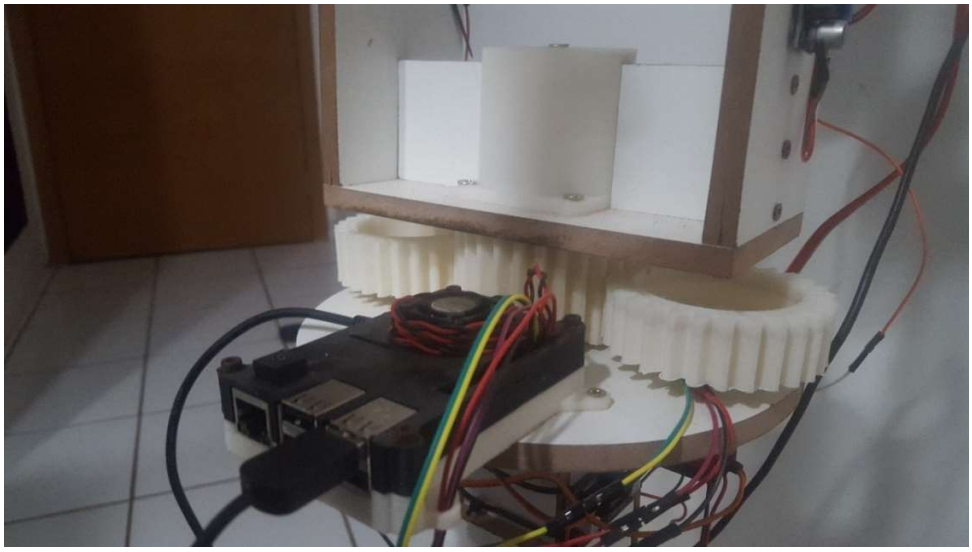


Figura 72: Conexões no Raspberry Pi e engrenagens de movimentação horizontal.

Figura 73: Peça de fixação superior da arma e fixação da câmera.

Figura 74: Peça de fixação inferior da arma emabixo do gatilho.

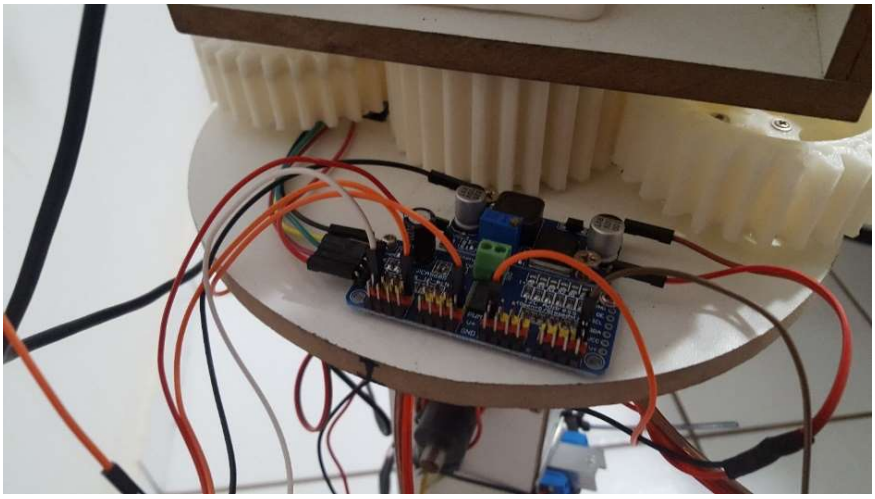


Figura 75: Conexões na placa PCA8695 e LM2596 de acionamento de gatilho.

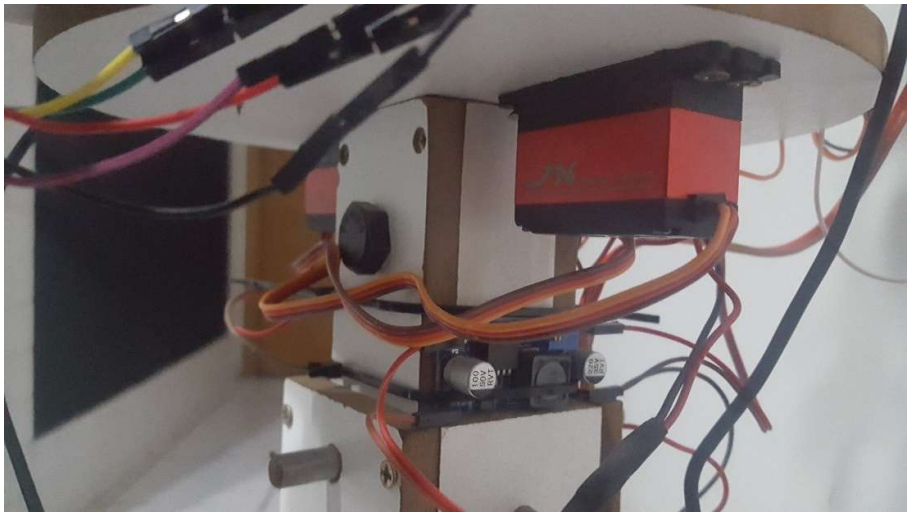


Figura 76: Servo-motor preso à plataforma e placa LM2596.

4.3 PROGRAMA

O sistema de controle do robô pode ser sumarizado pelo diagrama de blocos da Figura 77.

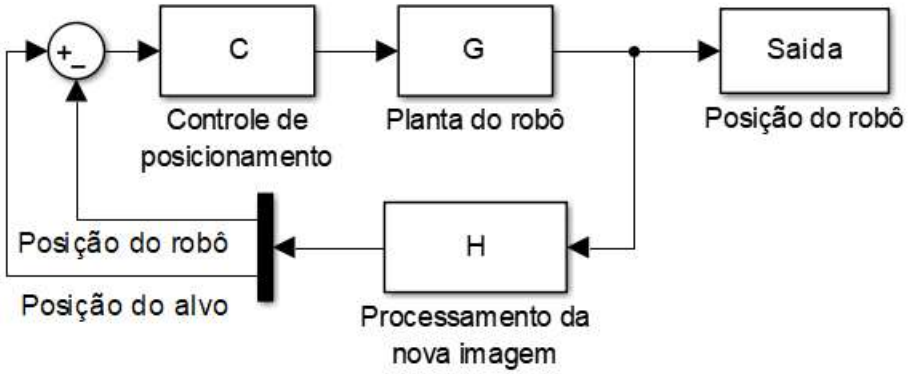


Figura 77: Diagrama de blocos do sistema de controle do robô

Para auxiliar o desenvolvimento do software de rastreamento e disparo, foi prototipado um modelo em escala de um mecanismo pan-tilt através de impressão 3D (Figura 78). Desta forma, pode ser programado um modelo equivalente ao que virá a rodar no robô antes de sua implementação na planta definitiva.

Figura 78: Mecanismo pan-tilt auxiliar prototipado

Foram desenvolvidos múltiplos softwares para o robô, nos tópicos seguintes eles serão abordados.

4.3.1 Movimentação dos motores

Para se mover os motores, foi utilizada a biblioteca de PWM da placa PCA8695 no Raspberry Pi e criadas duas funções para movimento. Uma função de movimento absoluta para ser utilizada no software de calibração, que não envolve qualquer sistema de controle, e uma função com controle proporcional para ser utilizada no software de rastreamento.

4.3.1.1 Movimento absoluto

Primeiramente, para cada motor dos eixos, foram aplicados manualmente diversos pulsos PWM para encontrar o valor em que os motores estão na posição central. Encontrados esses valores, eles são definidos como “zeros” dentro do programa, de modo que um incremento negativo faz este valor diminuir e o motor gira para um lado, e um incremento positivo faz girar para o outro. Uma coisa que tem que ser notada, é que os motores de *tilt* foram montados em direções opostas, então enquanto para um deles, o incremento positivo gira no sentido horário, para o outro, este mesmo incremento deve ter sinal oposto.

A função de movimento criada tem como entrada um valor binário para direção, e um valor inteiro para incremento. Ao ser chamada, ela testa o valor direção, se o movimento desejado for vertical, ela soma o incremento em um motor e subtrai no outro, e se for horizontal ela soma o incremento em ambos os motores. O fluxograma desta função pode ser visto na Figura 79.

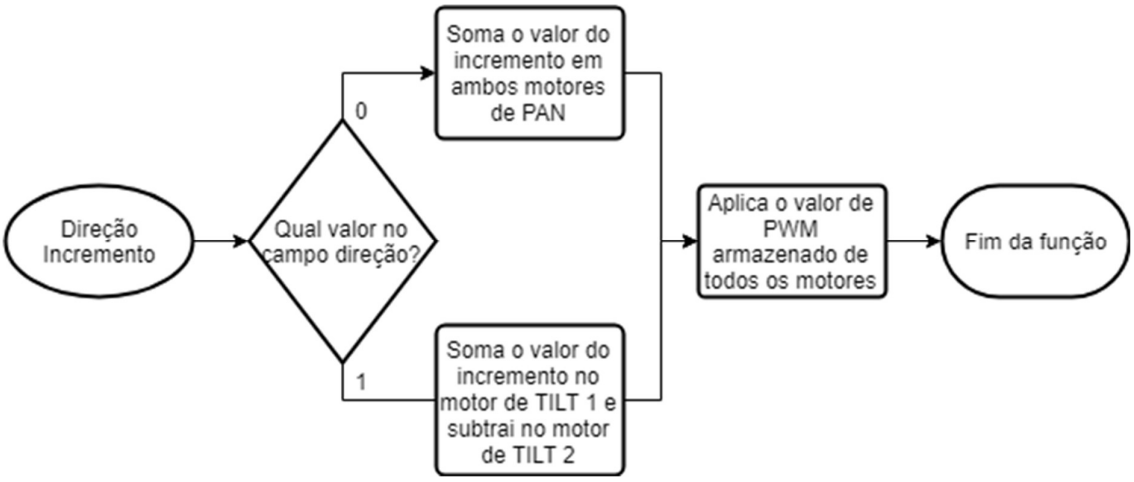


Figura 79: Fluxograma da função de movimento absoluto

A função de movimento absoluto tem como objetivo responder a entradas do teclado, por isso no programa de calibração ela é chamada sempre com incrementos de ± 1.

4.3.1.2 Movimento controlado

A função de movimento controlado, recebe o valor da diferença de posição entre a posição do a mira do robô e a posição do alvo nas duas direções, ela calcula qual deve ser a posição nova da mira e faz duas chamadas na função de movimento absoluto utilizando esses valores, uma chamada para cada direção.

Para cada uma das duas direções das quais a função de controle recebe o erro, ela o multiplica pelo valor escolhido do ganho proporcional (0<Kp<1), de modo a evitar sobressinal.

Por exemplo, se o alvo estiver na posição 100 pixels de altura e -20 pixels de largura a partir do centro da mira, a função de controle que esteja com o Kp ajustado em 0,5 chamará a função de movimento absoluto com entradas (0;50) e (1;-10). Na próxima interação, se a posição do alvo não mudar, a função de movimento absoluto será chamada com entradas (0;25) e (1;-5). E assim sucessivamente.

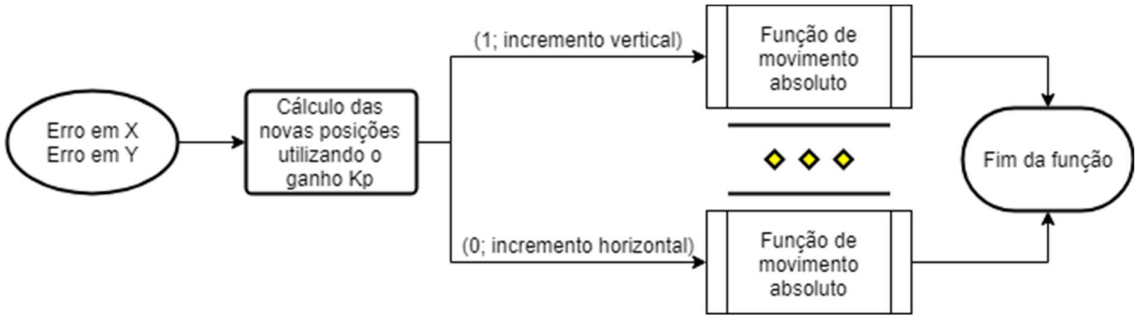


Figura 80: Fluxograma da função de controle do movimento

4.3.2 Calibração da mira

Antes de começar a rastrear seus alvos, a mira da câmera precisa ser calibrada, o que significa que o programa tem que saber qual região da imagem capturada pela câmera, o projétil irá atingir. Para isto foi criado um experimento onde um anteparo de papelão é posicionado a 10 m do robô e foi criado um software para calibração da mira.

O programa para calibração da mira desenha um retículo na tela onde é suposta de ser a mira não calibrada, em seguida, através de comandos do teclado, o usuário posiciona o robô na posição desejada e efetua disparos no anteparo de papelão. O usuário utiliza um laser para apontar a posição do furo realizado no anteparo de papelão, de modo que fique mais visível e em seguida posiciona um segundo retículo na posição dos furos no alvo e aperta um botão no teclado para redefinição da mira para a nova posição.

O fluxograma e uma captura de tela do programa para calibração podem ser vistos nas figuras a seguir.

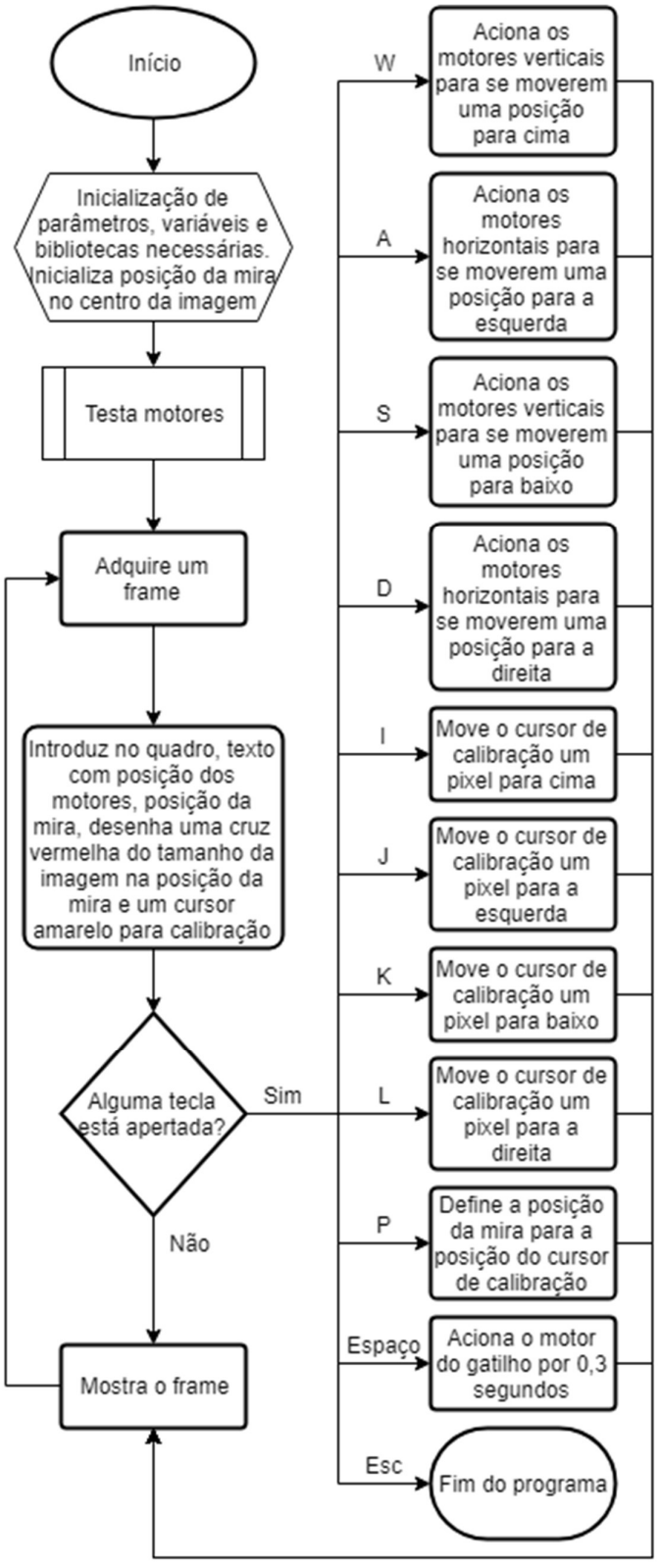


Figura 81: Fluxograma do programa de calibração da mira

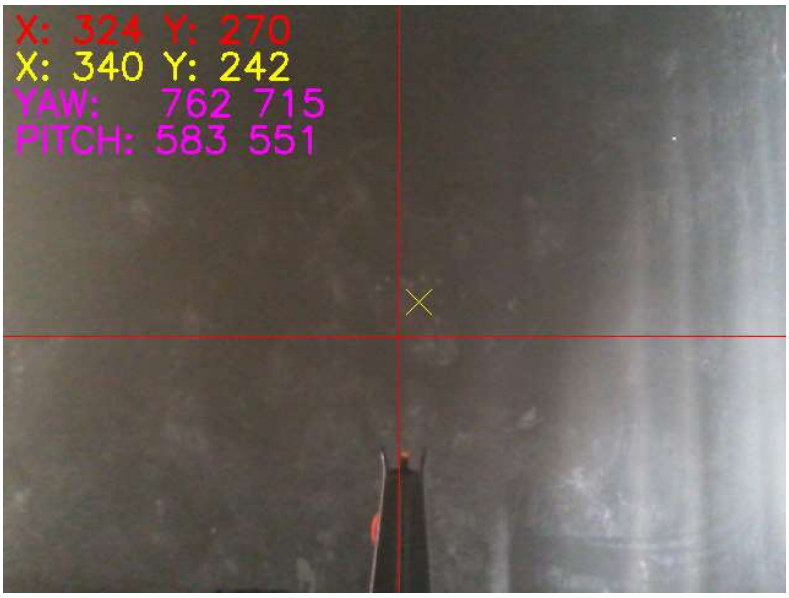


Figura 82: Captura de tela do programa de calibração da mira

Na Figura 82 pode-se observar a posição da mira em vermelho, o cursor de calibração em amarelo e os valores PWM aplicados nos 4 motores de movimentação em roxo.

4.3.3 Rastreamento

Apesar da robustez apresentada pela combinação dos diferentes métodos de rastreamento estudados, não foi possível sua implementação no hardware do Raspbeey Pi como idealizado. Por ser muito intensivo em recursos da máquina, o software começa a apresentar um processamento muito lento e longe do objetivo desejado em tempo real. Levando mais de 3 segundos para se processar um único quadro da imagem, não é aceitável para se tentar atingir um alvo em movimento. Para se obter uma resposta mais rápida, o software teve que ser simplificado ao máximo possível, neste processo, alguns métodos tiveram que ser removidos. O desenvolvimento do programa passou a ser focado principalmente em rastreamento por cor, pois este é o único dos métodos apresentados que permite a identificação de alvo aliado e adversário.

Para o programa de rastreamento do alvo, são inicializados os parâmetros adquiridos pelo programa de calibração. São eles os valores dos “zeros” dos servo-motores, posição da mira na imagem capturada pela câmera, distância mínima do alvo para abrir fogo, zona-morta em que o erro é desconsiderado para movimento dos motores, tamanho mínimo do alvo para ser considerado um alvo, ganho proporcional Kp, faixa de valores HSV da cor desejada para detecção do alvo, inicialização da placa PCA8695 e resolução da câmera.

Enquanto a tecla “Esc” não for pressionada, o robô captura uma imagem da câmera (frame) no formato BGR (equivalente ao RGB porém com os canais na ordem azul, verde e vermelho), formato padrão de captura do OpenCV. Esta etapa é mostrada na Figura 83.

Figura 83: Imagem capturada pela câmera do robô

Em cima da imagem capturada, é aplicado um filtro de *Blur* Gaussiano, que borra a imagem de modo a homogeneizar mais as diferentes regiões de cores. Como mostra a Figura 84.

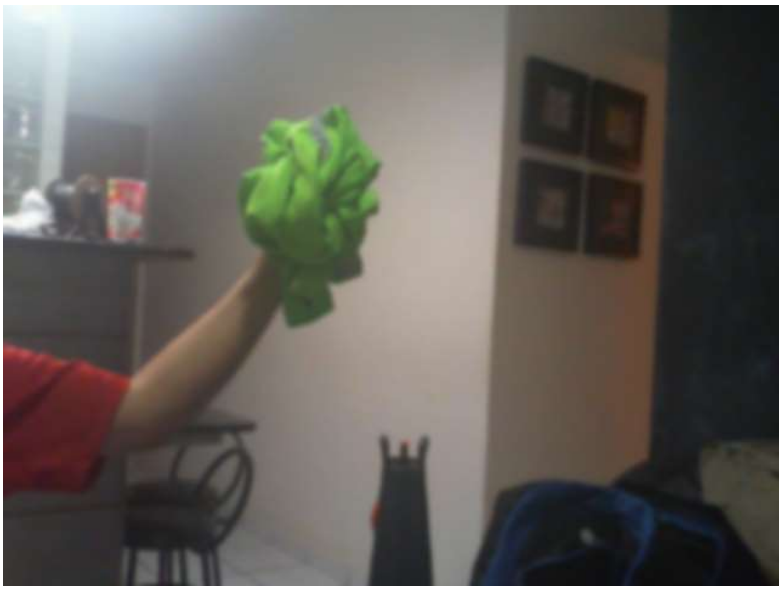


Figura 84: Imagem com filtro Gaussiano aplicado

Após aplicar o filtro, a imagem é convertida para o padrão de cores HSV. Convertida para HSV, cada pixel da imagem é comparado com a faixa de valores HSV da cor desejada para detecção de alvo, estes pixels são copiados em uma imagem binária para servir de máscara. Esta máscara é erodida e dilatada, processo que diminui a quantidade de pixels da máscara e depois aumenta, este processo se livra de uma boa parte do ruído da imagem que tenha passado para a máscara (Figura 85).

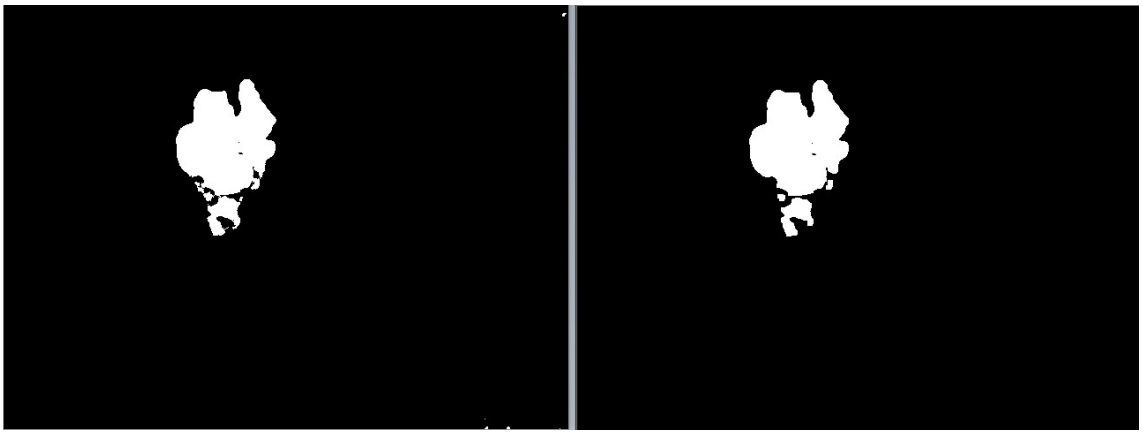


Figura 85: Máscara original e máscara depois do processo de erosão e dilatação.

Depois da eliminação do ruído indesejado, é utilizada a função do OpenCV *findContours*, esta função identifica e guarda todos os contornos que podem ser definidos pela máscara binária.

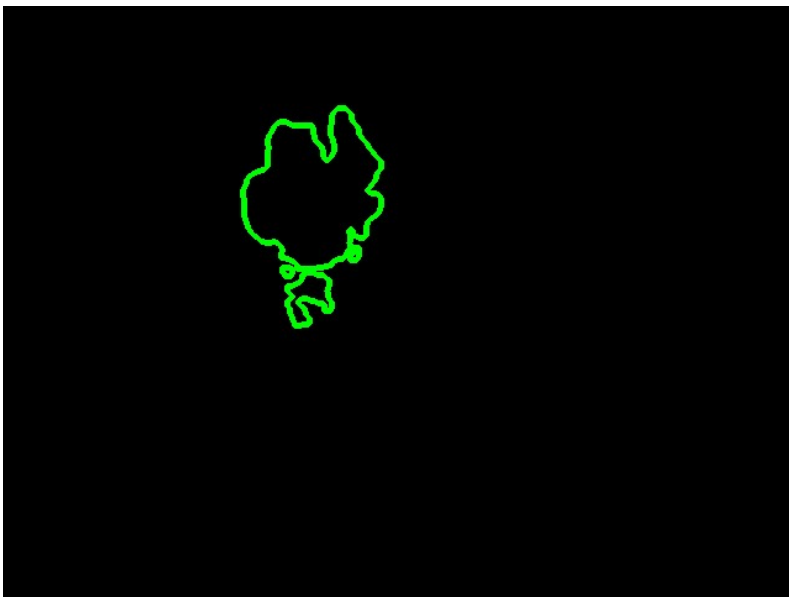


Figura 86: Imagem dos 4 contornos encontrados pela função de encontrar contornos do OpenCV

Tendo os contornos encontrados (Figura 86), é utilizada a função *moments* do OpenCV. Esta função retorna diversas características dos contornos encontrados previamente e será utilizada para encontrar o centro geométrico do alvo. Utilizando-se a função *moments* se encontra o centro geométrico de cada um dos contornos, a partir disso, é utilizada a função *findMinEnclosingCircle* que encontra o menor círculo que contém o contorno, se o raio deste círculo for maior que um valor predefinido, o contorno é descartado. Desta forma consegue-se reduzir o erro de detecção induzido por ruído que tenham conseguido passar por todos os filtros utilizados até então. Depois de descartar os contornos indesejáveis, é feita uma média ponderada da posição dos centros dos contornos restantes com a área de cada contorno e

este é considerado o centro do alvo. Com o centro do alvo encontrado, é calculada a distância do centro da mira que o alvo se encontra, como demonstrado na Figura 87.

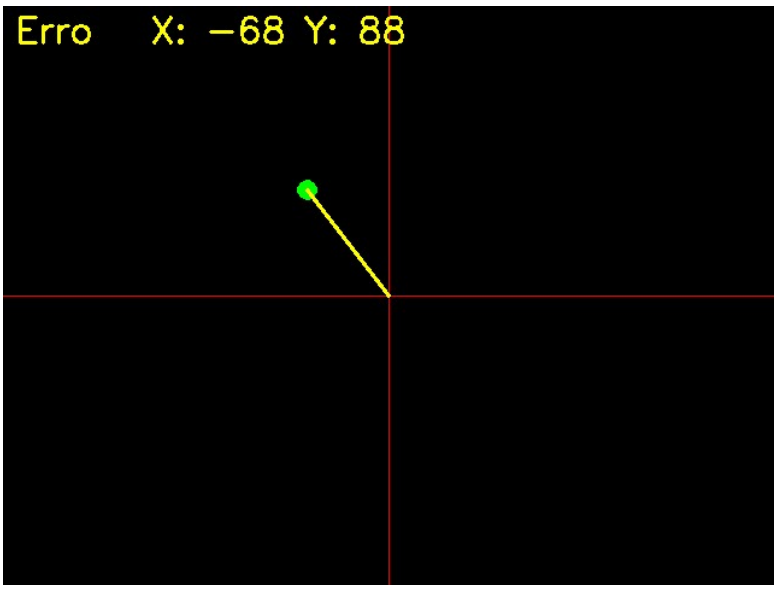


Figura 87: Diferença da posição do alvo para a posição do robô

Esta informação é considerada o erro de posicionamento do robô. Se o erro for menor que um valor de tolerância predefinido, significa que o robô está com a mira no alvo, e neste caso o gatilho é acionado, se o erro for maior, o gatilho é desligado e o erro é alimentado à função de movimento controlado.

Em seguida a imagem é desenhada com as informações adquiridas, é mostrada para o usuário (Figura 88) e o laço se repete.

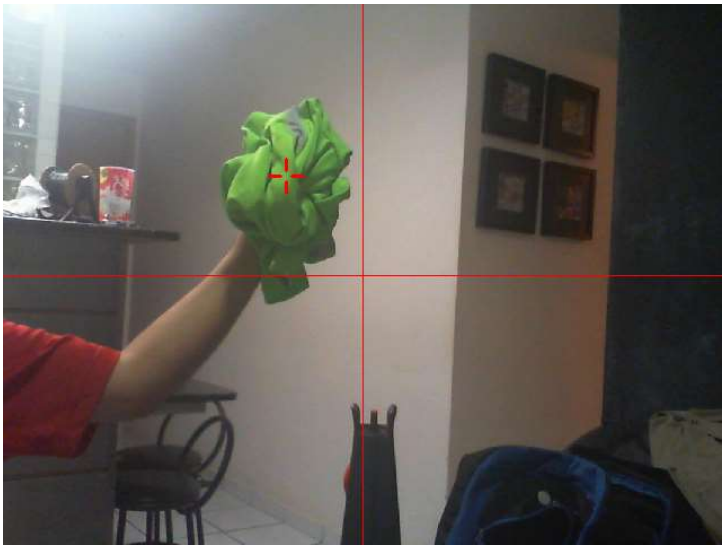


Figura 88: Imagem final com a posição da mira e posição do alvo marcados.

Na Figura 89 pode ser visto o fluxograma do software de rastreamento.

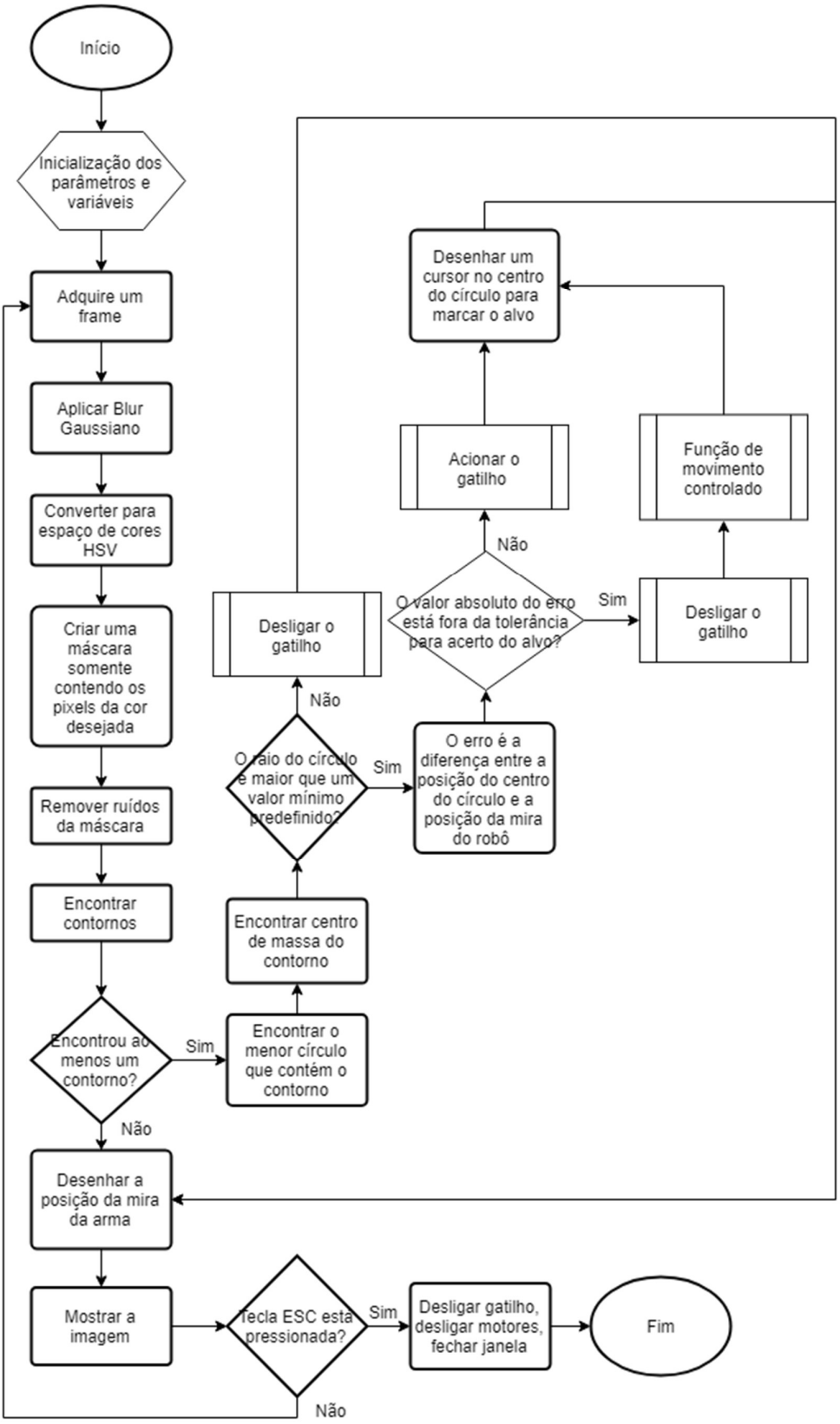


Figura 89: Fluxograma do software de rastreamento

4.3.4 Controle e acesso remoto

Para se controlar o robô remotamente, foi utilizado o programa VNC (incluso no sistema Raspbian do Raspberry Pi), este programa permite que o usuário controle o teclado e mouse do computador remoto, no caso o Raspberry Pi, via rede. Para isso, o Raspberry Pi foi conectado a um roteador Wi-Fi e foi usado um *laptop* para poder controlar, criar seu programa e instalar as bibliotecas necessárias.

CAPÍTULO 5 – RESULTADOS E CONCLUSÕES

Neste capítulo será analisada a performance do robô em diversas situações e a robustez de seu programa.

5.1 TEMPO DE RESPOSTA

Baseando-se no tempo necessário de cada ciclo completo de processamento de um frame em diferentes resoluções da câmera e tempo de travamento da mira no alvo, visa-se a análise de viabilidade do sistema.

5.1.1 Taxa de FPS

Utilizando-se a resolução máxima da câmera (1280 x 720), o tempo de processamento de um único frame passou de 5 segundos, o que é completamente inviável em um cenário que o robô tem que responder em tempo real. Reduzindo-se a resolução da câmera para 640 x 480, observou-se um grande aumento de velocidade, o Raspberry Pi passou a conseguir processar 1,52 FPS, porém muito longe do ideal. Reduzindo-se mais uma vez a resolução da imagem para 320 x 240, observou-se um grande aumento na taxa de frames, passando para 12,17 FPS, ainda assim não viável pois o sistema de controle leva mais de um frame para chegar na posição final dependo da diferença para a posição do alvo. Por fim foi utilizada a resolução de 160x120 e a taxa de FPS conseguida foi de 29,21, valor que é quase o máximo da câmera de acordo com suas especificações, então foi a resolução adotada para funcionamento nominal.

5.1.2 Ganho do sistema

Para se medir o tempo de travamento da mira no alvo baseado nos diferentes valores possíveis para o ganho do sistema, foi realizado um experimento fixando um apontador laser no cano da arma e o apontado para a escala angular criada previamente no experimento de momento de inércia angular.

O experimento consiste em apresentar o alvo em determinado instante para o robô e filmar com a câmera de alta velocidade a resposta do robô com diferentes valores de ganho. Em seguida é retirado dos vídeos, os tempos e posições do laser para os valores de ganho testados. Este experimento pode ser visualizado na Figura 90.

Figura 90: Experimento para se medir a influência do ganho

Os valores obtidos são comparados no gráfico da Figura 91.

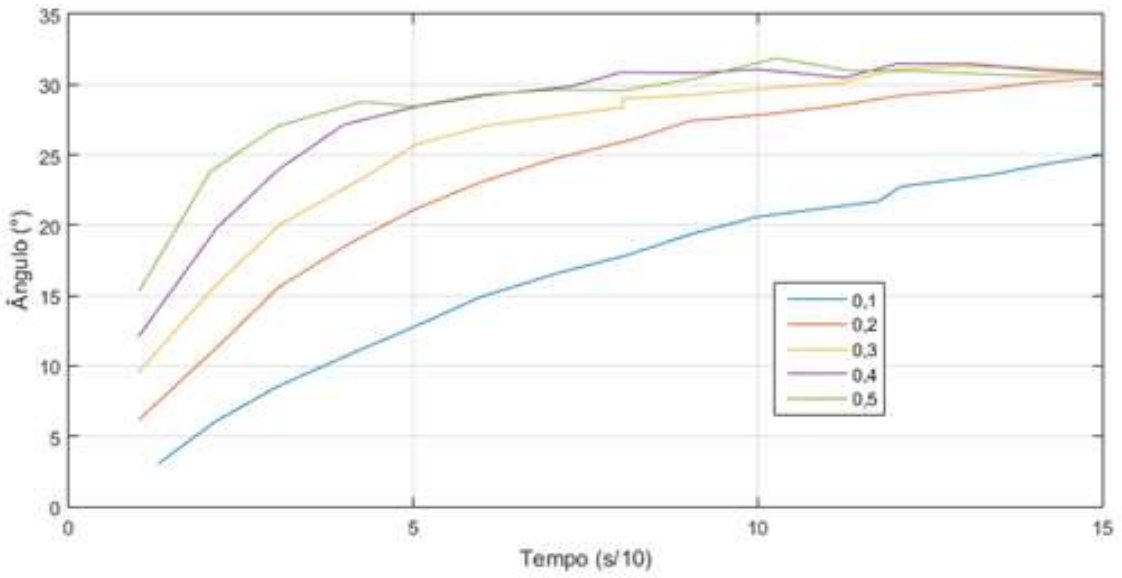


Figura 91: Gráfico mostrando o deslocamento com diferentes ganhos.

De acordo com os dados observados do experimento, pode se observar que embora a resposta seja mais rápida, a partir do valor de ganho 0,5 o sistema começa a apresentar sobressinal e oscilações em torno do valor final. Para evitar estas oscilações, o valor do ganho escolhido para funcionamento nominal do sistema foi de 0,425.

5.2 PRECISÃO DO SISTEMA

Como calculado anteriormente, o servo motor tem a resolução de 1000 estados por 180°. Comparando-se com o campo de visão horizontal da câmera de aproximadamente 60°, seria o equivalente a 333 estados possíveis dentro de uma mesma imagem.

Com a diminuição da resolução da câmera, o número de pixels dentro dos 60° de campo de visão passou a ser 160, a resolução máxima do servo motor também passa a ser 160 estados, ao invés do máximo possível de 333. E isto afeta diretamente a resolução do robô, que passou a ser 0,375°/pixel

5.2.1 Precisão estática

Para se medir a precisão estática do robô, foi utilizado o software de calibração. A mira foi calibrada como na descrição do mesmo. Foi colocado um anteparo de papelão a 8 m de distância e o robô foi posicionado manualmente utilizando o software de calibração. O fuzil de Airsoft foi colocado em sua configuração automática. Desta maneira foi utilizada sua frequência máxima de disparos. Foram realizados disparos durante 3 segundos, o que resultou em um total de 31 disparos. A dispersão de disparos devido a vibrações da estrutura, incertezas de posicionamento e sincronia dos dois motores em cada eixo, pode ser vista na Figura 92. O desvio padrão encontrado foi de 62 mm.

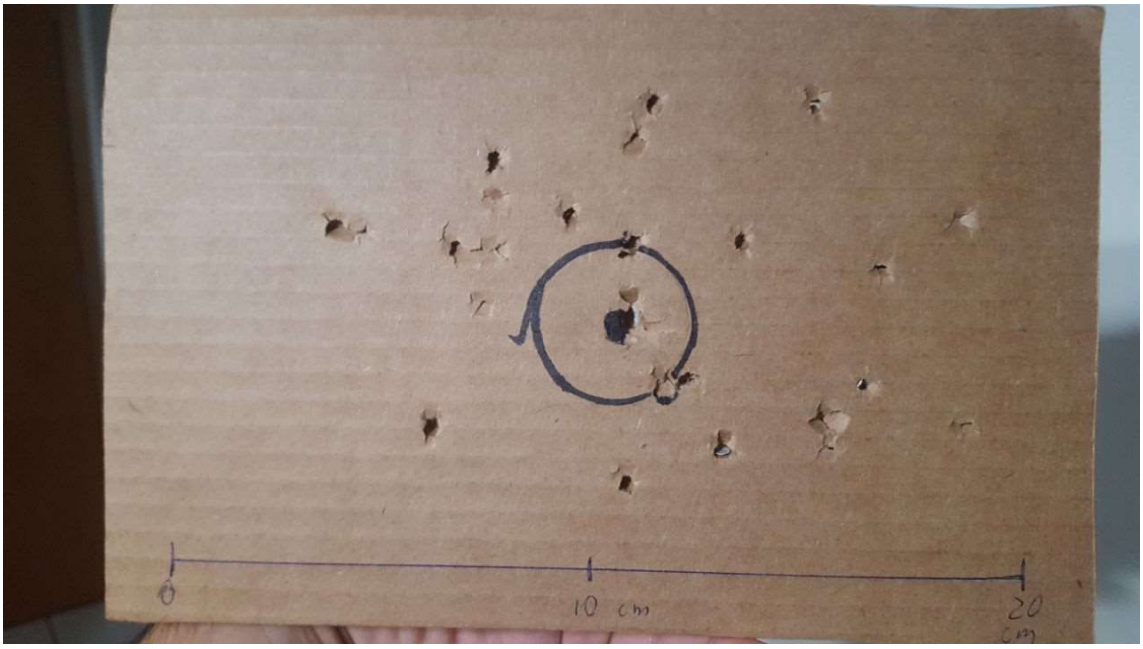


Figura 92: Dispersão de disparos em teste de precisão estática.

5.2.1 Precisão dinâmica

Para se testar a precisão do robô em alvos em movimento, foram cortados pedaços de papelão de 50 cm x 50 cm, uma pessoa segurou estes alvos a distâncias de 5, 10, 15, 20 e 25 metros e passava na frente do robô andando normalmente, correndo e correndo mudando

de direção no meio do caminho, para cada caso, o robô disparou 10 projéteis a uma taxa de 2 projéteis por segundo. Os pedaços de papelão foram analisados em seguida e a quantidade de furos devidos aos projéteis foi anotada como mostra a Tabela 4.

	5m	10 m	15 m	20 m	25 m
Andando	10	10	8	4	2
Correndo	4	8	6	4	0
Aleatório	6	8	7	5	1

Tabela 4: Disparos atingidos em diferentes situações de movimento e distância

Desta tabela, pode-se perceber que o ponto ótimo de funcionamento do robô é a uma distância entre 10 e 15 metros. A uma distância de 5m, pode-se perceber uma taxa de acerto menor do que a 10, isto se deve ao fato de a velocidade angular do robô precisar ser maior quanto mais perto o alvo está. A distâncias acima de 20 m, a resolução mais baixa do robô e vibração da arma ao atirar se combinam de modo a cair muito a precisão como o esperado.

O experimento foi conduzido na arena CQC de Airsoft mostrada na Figura 20, com condições de iluminação controladas e alvos pintados de laranja fluorescente para fácil identificação.

Após isso, o mesmo experimento foi realizado em campo aberto no local da Figura 93.

Figura 93: Campo aberto de Airsoft

Seguindo o mesmo modelo de experimento no novo cenário, são obtidos os dados apresentados na Tabela 5.

	5m	10 m	15 m	20 m	25 m
Andando	9	9	7	2	1
Correndo	3	8	6	2	0
Aleatório	5	7	6	1	0

Tabela 5: Disparos atingidos em diferentes situações de movimento e distância em campo aberto.

O robô teve um desempenho pior em quase todos os cenários, mas seu ponto de operação ótimo continua praticamente o mesmo.

Deve-se tomar nota que no campo aberto, um vento de intensidade mediana é suficiente para mudar as condições de iluminação e o auto-ajuste de exposição da câmera fazer o robô perder completamente o alvo e detectar grandes pontos de ruído como alvo erroneamente.

5.3 CONCLUSÕES FINAIS

Embora todos os requisitos primariamente impostos não tenham sido atingidos, este projeto demonstra o conceito e solução válidos para a proposição inicial. O protótipo do robô não foi testado em um jogo real por questões de segurança dos jogadores e do sistema, mas ao se implementar os artifícios de segurança propostos inicialmente, pode ser feito no futuro.

5.4 TRABALHOS FUTUROS

Para melhoramento do projeto e sua adequação a um produto comercial, devem ser tomados alguns aprimoramentos em relação ao que faltou ser implementado. Realizar a detecção de que foi atingido, trocar a placa de controle para um BeagleBone Black ou melhor e implementar as opções de controle e processamento de imagem propostas. Inserir um sensor de luminosidade de modo que os valores de cor do alvo possam ser modificados dinamicamente e tentar se reduzir os problemas de causados pela auto-exposição da câmera. Implementar uma forma de ajustar o ganho do sistema de acordo com a distância que o alvo está do robô, desta forma ele pode se ajustar dinamicamente para acertar mais alvos a distâncas diferentes. Da mesma forma pode-se ajustar a tolerância para efetuação de disparo, de acordo com a distância, aumentando a taxa de acertos a longa distância e diminuindo a necessidade de movimentos redundantes a curta distância. É desejado também melhorar a estabilidade da estrutura, melhorar o mecanismo de transmissão e a utilização de um único motor para eliminar problemas de sincronias.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Sentry Gun CODG. Disponível em: <http://callofduty.wikia.com/wiki/File:Sentry_Gun_CODG.png>. Acesso em: 12/07/2017.

[2] Azevedo, D.A., Lima, P.P.P., Leite, A.C., Zachi, A.R.L. Controle de orientação de uma câmera pan-tilt por servovisão baseada em umagem com realimentação de saída..

[3] Vídeo de demonstração disponível em: <http://projectsentrygun.rudolphlabs.com/products/gladiator-ii-paintball-turret>. Acesso em: 12/07/2017.

[4] Целуйко Е. Disponível em: <http://projectsentrygun.rudolphlabs.com/successful-projects>. Acesso em: 15/07/2017.

[5] Heavy Turret. Disponível em <http://www.realsentrygun.com/Paintball%20Airsoft%20Turret%20Kit2.htm>. Acesso em: 15/07/2017.

[6] Mini Turret. Disponível em: <http://www.realsentrygun.com/Paintball%20Airsoft%20Turret%20Kit.htm>. Acesso em: 15/07/2017.

[7] Blain, L. South Korea's autonomous robot gun turrets: deadly from kilometers away. Disponível em: <http://newatlas.com/korea-dodamm-super-aegis-autonomos-robot-gun-turret/17198/>. Acesso em: 15/07/2017.

[8] Samsung SGR-A1. Disponível em: <de http://www.theregister.co.uk/2007/03/14/south_korean_gun_bots/>. Acesso em: 15/07/2017.

[9] Arduino MEGA. Disponível em: <https://store.arduino.cc/usa/arduino-mega-2560-rev3>. Acesso em 13/06/2018.

[10] Raspberry Pi 3 Model B. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em 12/06/2018.

[11] BeagleBone Black. Disponível em: <https://beagleboard.org/black>. Acesso em: 12/06/2018.

[12] Intel Galileo. Disponível em: <https://software.intel.com/en-us/iot/hardware/discontinued>. Acesso em: 13/06/2018.

[13] OpenCV. Disponível em: <https://opencv.org>. Acesso em: 20/09/2017.

[14] VLFeat. Disponível em: <http://www.vlfeat.org>. Acesso em 20/09/2017.

[15] VXL. Disponível em: <http://vxl.sourceforge.net>. Acesso em: 20/09/2017.

[16] S. M. Smith, Finding Optic Flow. Disponível em: <https://users.fmrib.ox.ac.uk/~steve/review/review/node1.html>. Acesso em 21/09/2017.

[17] OpenCV: Optical Flow. Disponível em: <https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html>. Acesso em 24/09/2017.

[18] OpenCV: Background Subtraction Tutorial. Disponível em: <https://docs.opencv.org/3.3.1/db/d5c/tutorial_py_bg_subtraction.html>. Acesso em 24/09/2017.

[19] RGB Cube. Disponível em: <https://en.wikipedia.org/wiki/File:RGB_Cube_Show_lowgamma_cutout_a.png>. Acesso em 13/06/2018.

[20] HSV Cylinder. Disponível em: <https://en.wikipedia.org/wiki/File:HSV_color_solid_cylinder_alpha_lowgamma.png>. Acesso em 13/06/2018.

[21] Color Depth. Disponível em <https://en.wikipedia.org/wiki/Color_depth>. Acesso em: 13/06/2018.

[22] J. Leong, Number Of Colors Distinguishable By The Human Eye. Disponível em: <https://hypertextbook.com/facts/2006/JenniferLeong.shtml>. Acesso em: 13/06/2018, 2006.

[23] RGB to HSV color conversion. Disponível em: <https://www.rapidtables.com/convert/color/rgb-to-hsv.html>. Acesso em 13/06/2018.

[24] Changing colorspaces. Disponível em: <https://docs.opencv.org/3.3.1/df/d9d/tutorial_py_colorspaces.html>. Acesso em 13/06/2018.

[25] A. S. Coutinho, Elemento da forma: a proporção. Disponível em: <http://portaldoprofessor.mec.gov.br/fichaTecnicaAula.html?aula=40628>. Acesso em: 20/07/2017.

[26] NBR 17 - Ergonomia. Disponível em: <http://www.ebah.com.br/content/ABAAAA3NIAE/nbr-17-ergonomia>. Acesso em: 20/07/2017.

[27] How fast does Usain Bolt run, Disponível em: <https://www.eurosport.com/athletics/how-fast-does-usain-bolt-run-in-mph-km-per-hour-is-he-the-fastest-recorded-human-ever-100m-record_sto5988142/story.shtml>. Acesso em: 23/07/2017.

[28] The Airsoft Trajectory Project. Disponível em: <http://mackila.com/airsoft/atp/01-d-01.htm>. Acesso em 30/07/2017.

[29] Analytic Functions, The Magnus Effect, and Wings. Disponível em: <http://www.mathpages.com/home/kmath258/kmath258.htm>. Acesso em: 12/07/2017.

[30] Imagem criada pelo usuário “Rdurkacz” na wikipedia e disponível em: https://en.wikipedia.org/wiki/Magnus_effect#/media/File:Sketch_of_Magnus_effect_with_streamlines_and_turbulent_wake.svg>. Acesso em 01/08/2017.

[31] G&G Airsoft Metal Hop Up Chamber For GR25 Series AEG. Disponível em: <https://www.hobbytron.com/GGMetalHopUpChamberForGR25SeriesAEG.html>. Acesso em: 25/03/2017..

[32] Mass Comparison of Projectiles Fired at 1.14 Joules. Disponível em: <http://mackila.com/airsoft/atp/07-b-07.htm>. Acesso em: 06/03/2018..

[33] Experiment: How fast your brain reacts to stimuli. Disponível em: <https://backyardbrains.com/experiments/reactiontime>. Acesso em: 11/01/2018.

[34] Animação de funcionamento de uma AEG. Disponível em: <http://www.instructables.com/id/The-Ultimate-Guide-To-AirsoftingReally/step10/So-how-do-airsoft-guns-work/> Acesso em 05/08/2017.

[35] Modelo da câmera adaptado. Disponível em: <http://archive3d.net/?a=download&id=e5f47143>. Acesso em 06/08/2017.

[36] Disponível em: <http://browningmgs.com/Mounts/Mounts.htm>. Acesso em 20/03/2018.

[37] Prototype ver.6. Disponível em: <http://www.realsentrygun.com/Prototypes.htm#version_six>. Acesso em: 12/07/2017.

[38] ATPG18 - ALL TERRAIN POD WITH FM18 HEAD. Disponível em: <http://www.tiffen.com/displayproduct.html?tablename=davissanford&itemnum=ATPG18>. Acesso em 14/09/2017.

[39] Using a standard USB webcam. Disponível em: <https://www.raspberrypi.org/documentation/usage/webcams/>. Acesso em 13/06/2018.

[40] Manual da câmera Microsoft LifeCam HD-3000. Disponível em: <http://download.microsoft.com/download/0/9/5/0952776D-7A26-40E1-80C4-76D73FC729DF/TDS_LifeCamHD-3000.pdf>. Acesso em 13/06/2018.

[41] Site da internet do Mercado Livre. Disponível em: <www.mercadolivre.com.br>. Acesso em 23/10/2017.

[42] Especificações do Servo Jx PDI-6221MG. Disponível em: <http://www.jx-servo.com/English/Product/0951873936.html>. Acesso em 05/10/2017.

[43] Duty cycle general - Pulse Width Modulation. Disponível em: <https://en.wikipedia.org/wiki/Pulse-width_modulation#/media/File:Duty_cycle_general.svg>. Acesso em:13/06/2018.

[44] Tutorial de PWM do Arduino. Disponível em: <https://www.arduino.cc/en/Tutorial/PWM>. Acesso em: 13/06/2018.

[45] Pin Numbering. Disponível em: <http://pi4j.com/pins/model-3b-rev1.html>. Acesso em 13/06/2018.

[46] PCA9685. Disponível em: <https://www.dhgate.com/store/product/pca9685-16-channel-12-bit-pwm-servo-motor/384786052.html>. Acesso em 13/06/2018.

[47] GitHub PCA8695 Python. Disponível em: <https://github.com/adafruit/Adafruit_Python_PCA9685>. Acesso em 13/06/2018.

[48] LM2596. Disponível em: <http://www.baudaeletronica.com.br/modulo-regulador-de-tensao-lm2596.html?gclid=CjwKCAjwgYPZBRBoEiwA2XeupQEDqL-BxgM_xuLpZIugSEOC8bLxlyN2eDxjBBmDP33M0oEliUL2choCrQ8QAvD_BwE>. Acesso em: 13/06/2018.

[49] Datasheet LM2696. Disponível em: <http://pdf1.alldatasheet.com/datasheet-pdf/view/134372/ETC1/LM2596.html>. Acesso em 13/06/2018.

[50] Suporte amortecido de câmera para drone. Disponível em: <https://produto.mercadolivre.com.br/MLB-858064872-suporte-gimbal-amortecedor-p-cmera-gopro-acosyma-x8g-_JM>. Acesso em 13/06/2018.

[51] How to find frame rate of a video in OpenCV?. Disponível em: <https://www.learnopencv.com/how-to-find-frame-rate-or-frames-per-second-fps-in-opencv-python-cpp/>. Acesso em 12/01/2018.

[52] Kikuchi, D.Y. Sistema de controle servo visual de uma câmera pan-tilt com rastreamento de uma região de preferência. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3152/tde-27072007-163810/pt-br.php>. Acesso em 15/08/2017.

[53] Lu, N., Wang, J., Wu, Q. H., Yang, L. An improved Motion Detection Method for Real-Time Surveillance. Disponível em: <http://alumni.cs.ucr.edu/~ychen053/motion_detection.pdf>. Acesso em: 29/08/2017.

[54] Rosebrock, A. Basic Motion detection and tracking with Python and OpenCV. Disponível em: <http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/>. Acesso em: 01/10/2017.

ANEXOS

Anexo 1 – Desenho do eixo de movimento horizontal

Anexo 2 – Desenho da estrutura interna da base telescópica

Anexo 3 – Desenho da estrutura externa da base telescópica

Anexo 4 – Desenho da plataforma de fixação dos motores e componentes eletrônicos

Anexo 5 – Desenho da estrutura *tilt*, suporte da arma de Airsoft

Anexo 6 – Desenho da peça de fixação da arma na estrutura *tilt*

Anexo 7 – Desenho da peça de fixação superior da arma e câmera na estrutura *tilt*

Anexo 8 – Desenho da peça de fixação no trilho da arma

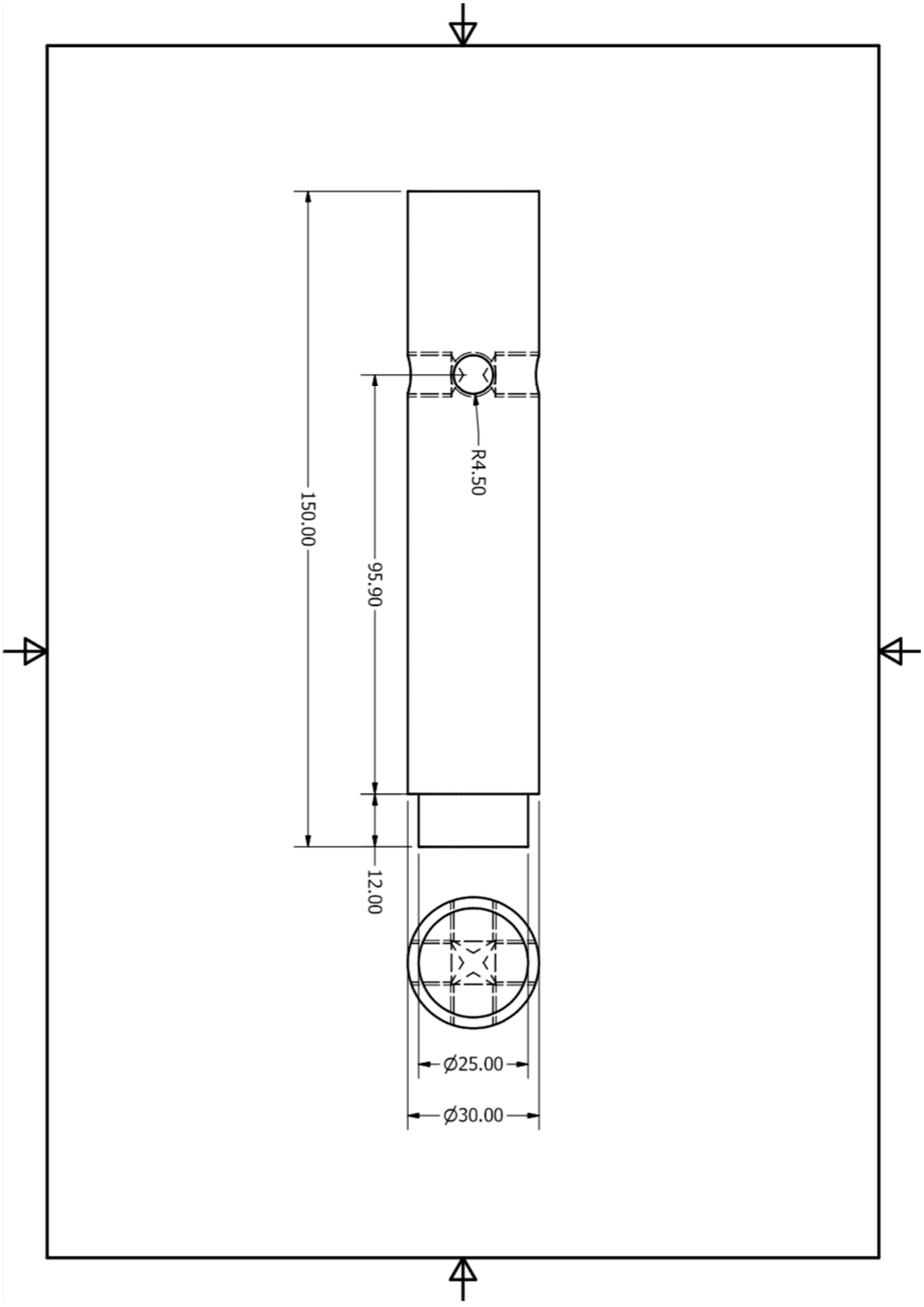
Anexo 9 – Desenho da peça de fixação no trilho da arma

Anexo 10 – Desenho das engrenagens de movimento horizontal

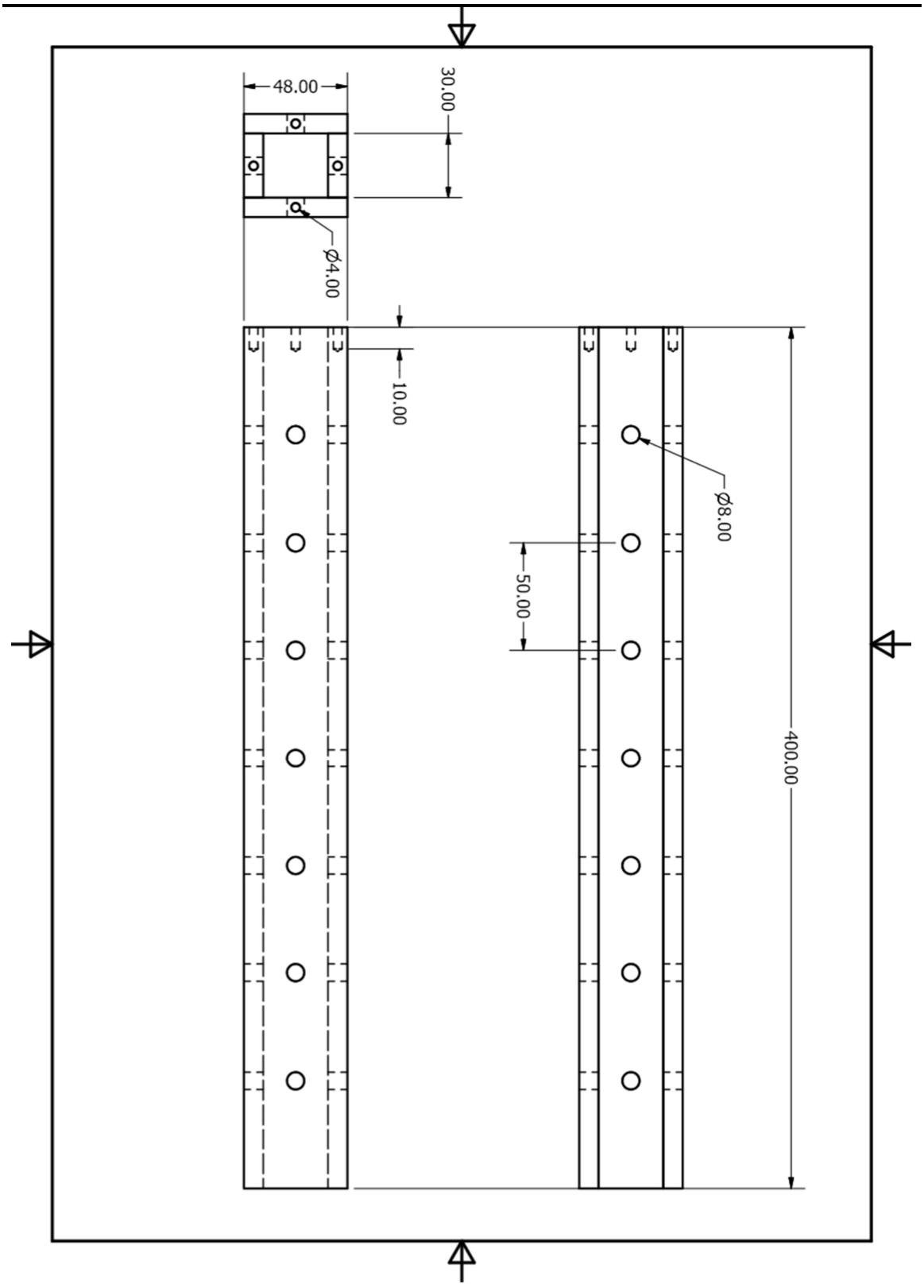
Anexo 11 – Desenho das engrenagens de movimento vertical

Anexo 12 – Programas utilizados

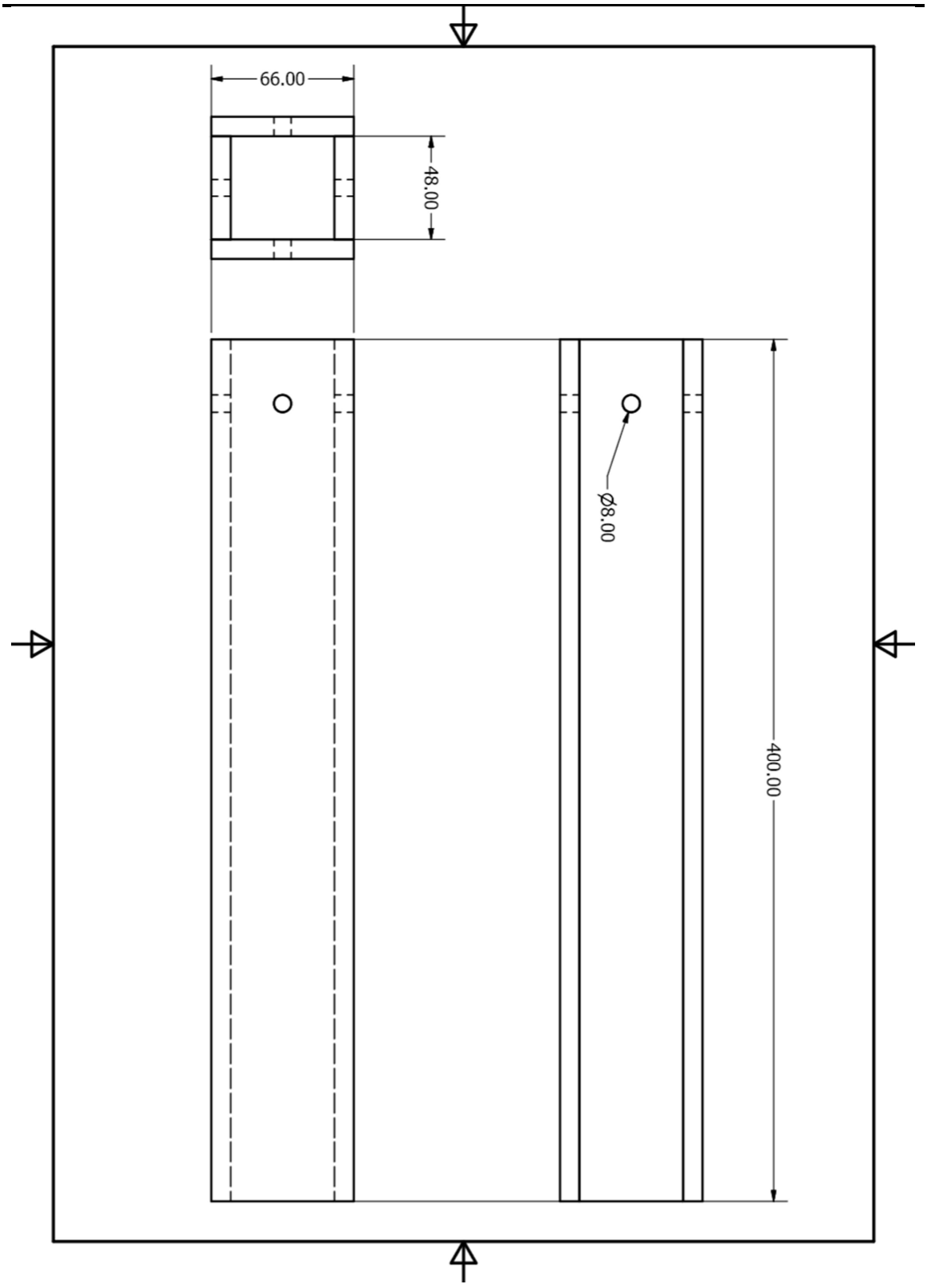
ANEXO 1: Desenho do eixo de movimento horizontal



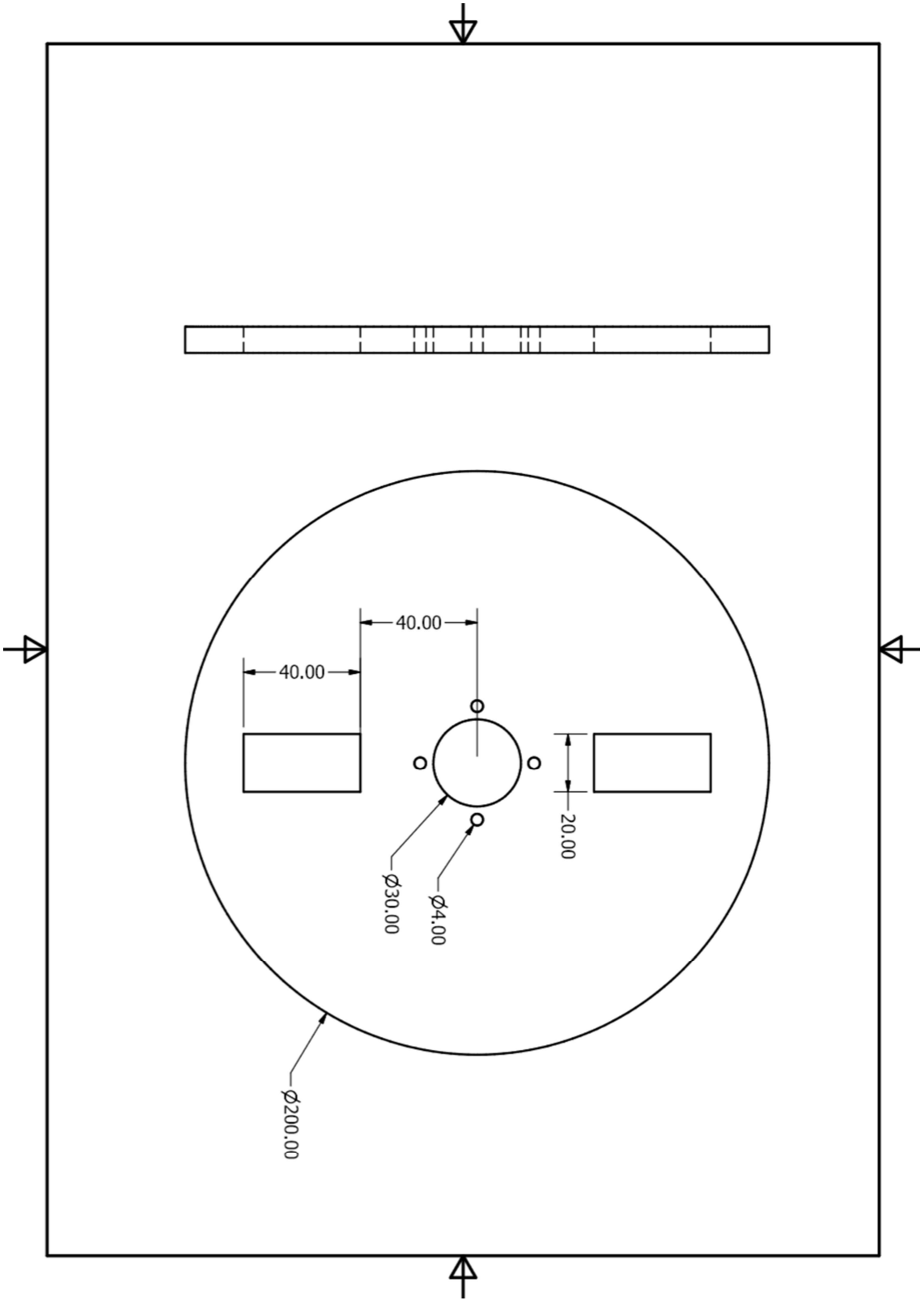
ANEXO 2: Desenho da estrutura interna da base telescópica



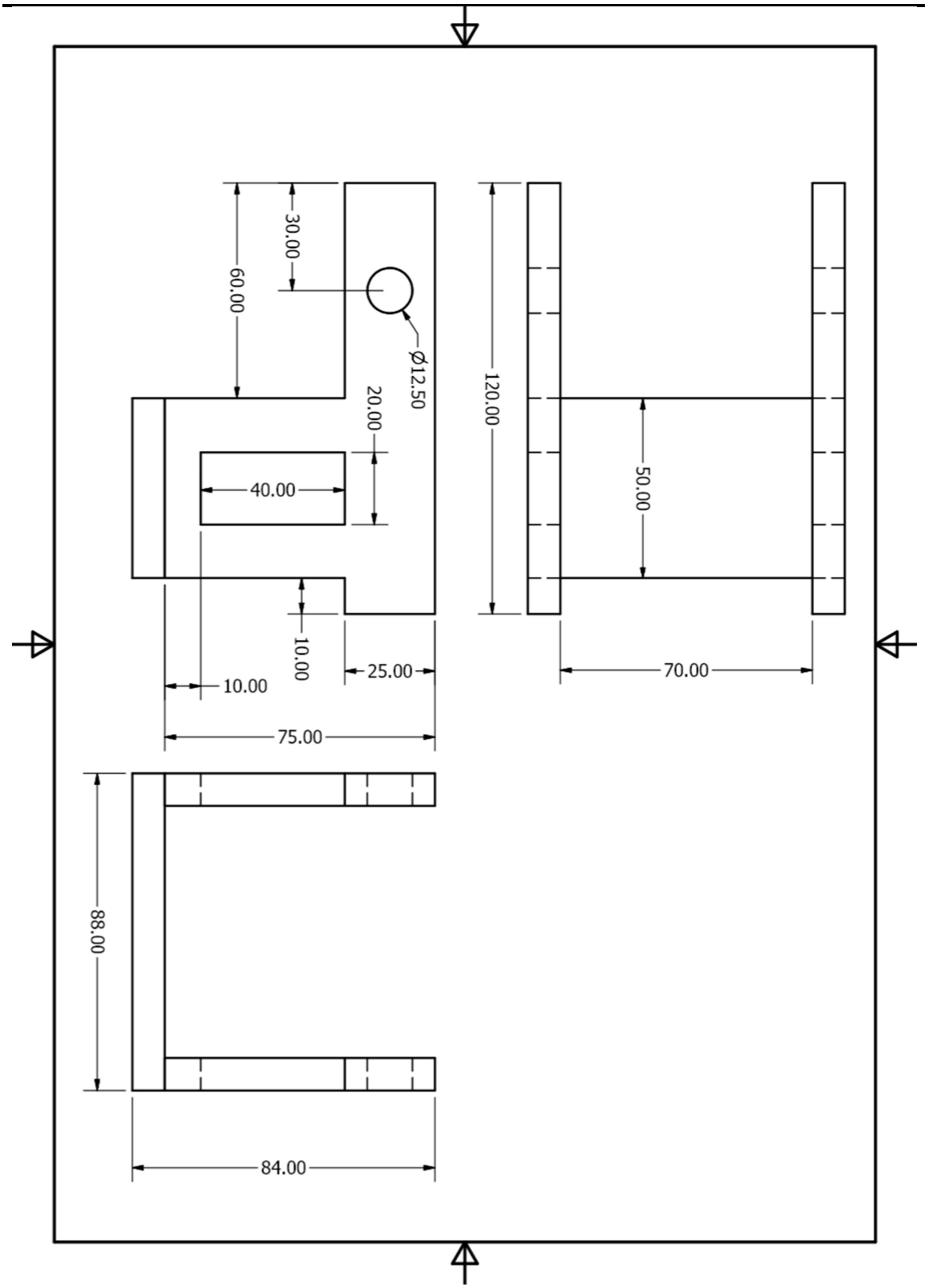
ANEXO 3: Desenho do eixo de movimento horizontal



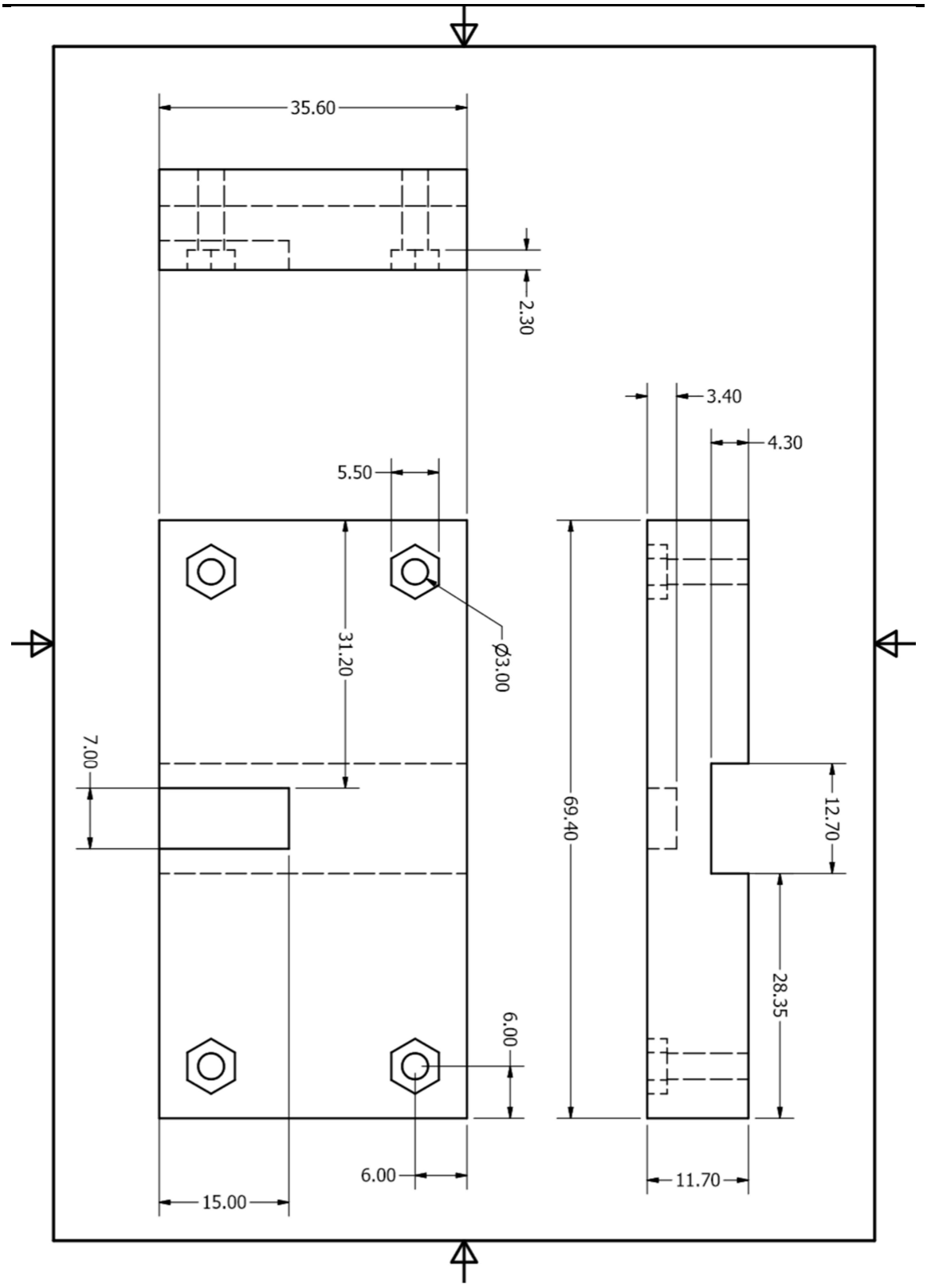
ANEXO 4: Desenho da plataforma de fixação dos motores e componentes eletrônicos



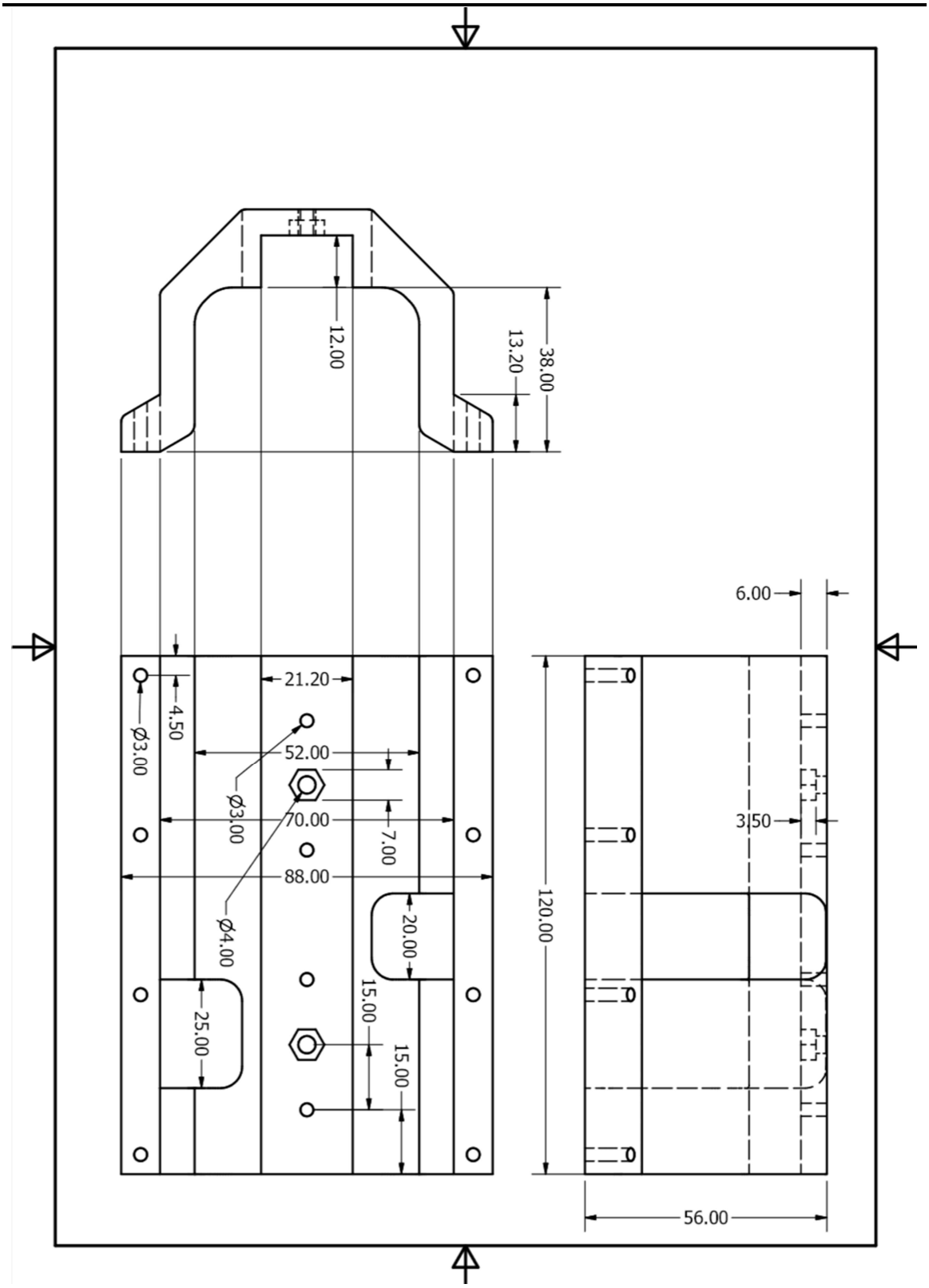
ANEXO 5: Desenho da estrutura *tilt*, suporte da arma de airsoft



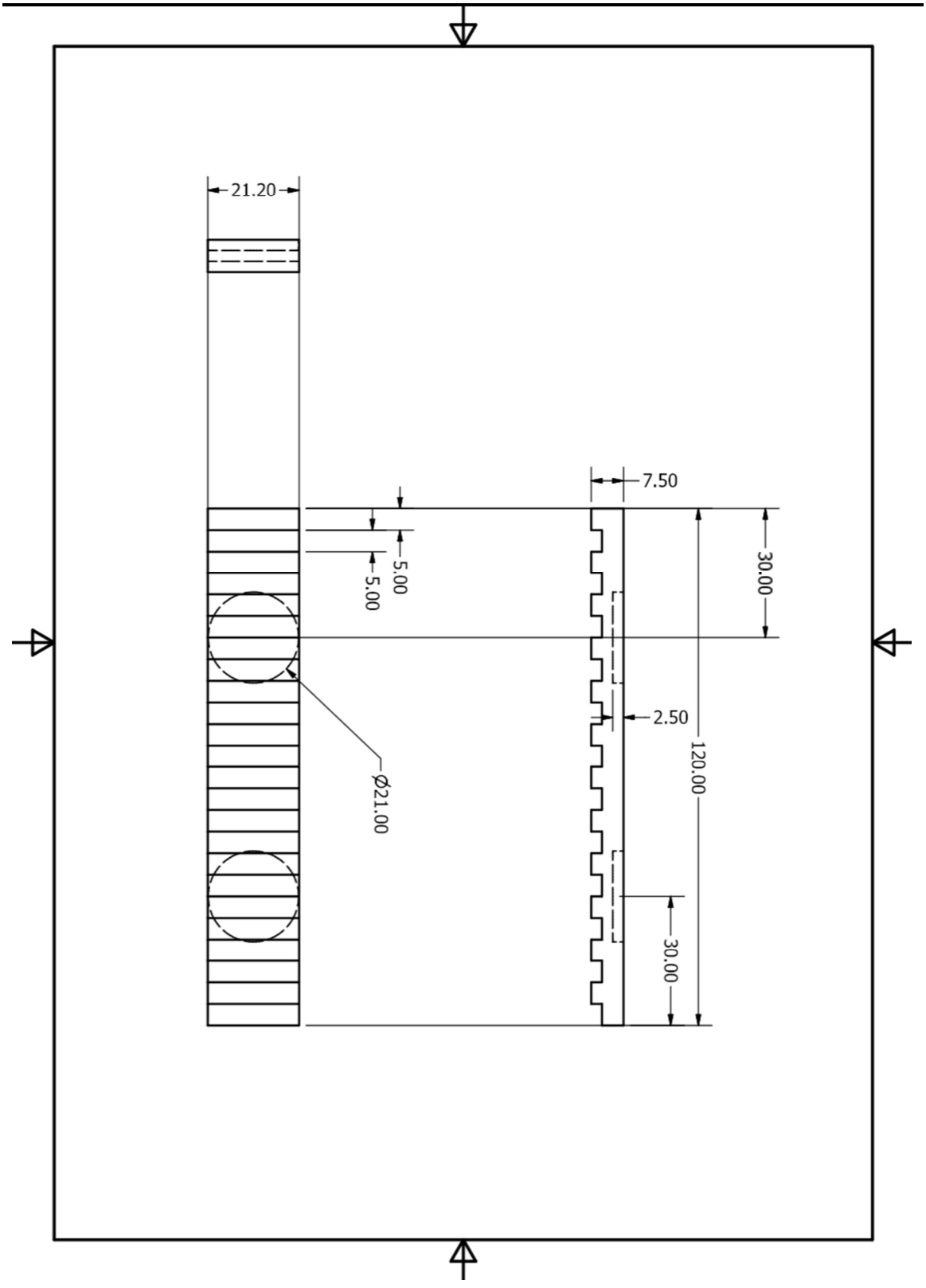
ANEXO 6: Desenho da peça de fixação da arma na estrutura *tilt*



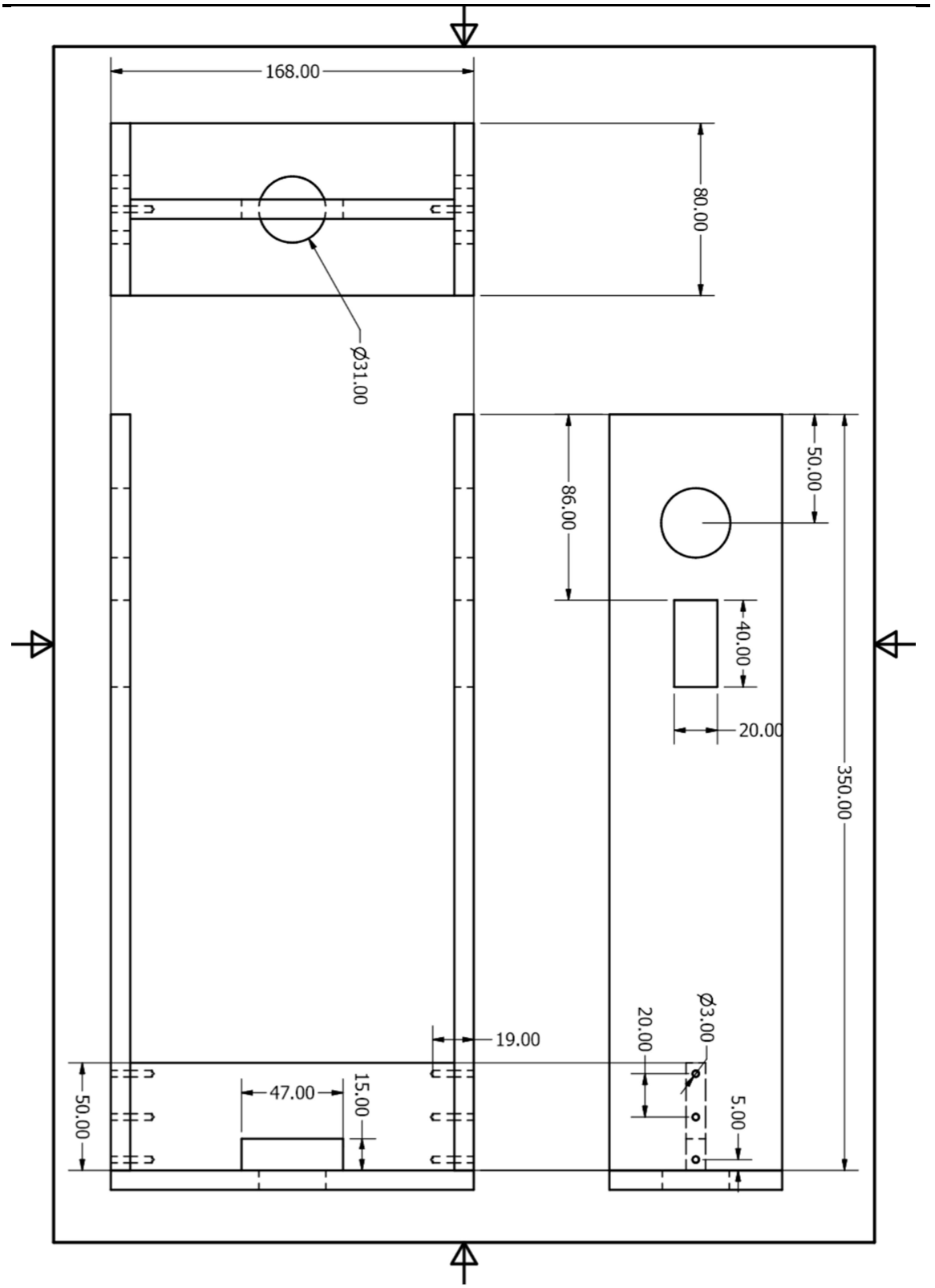
ANEXO 7: Desenho da peça de fixação superior da arma e câmera na estrutura *tilt*



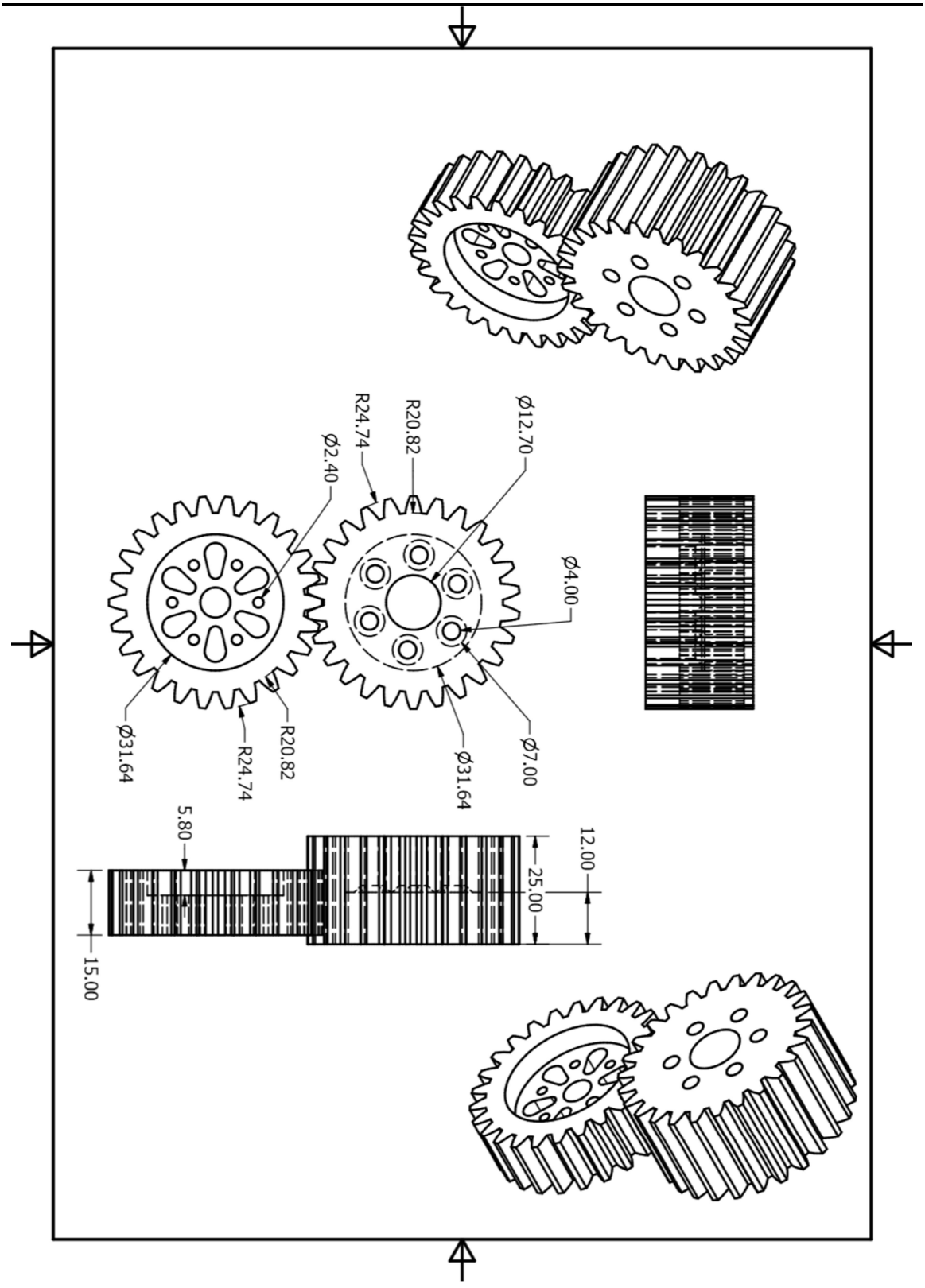
ANEXO 8: Desenho da peça de fixação no trilho da arma



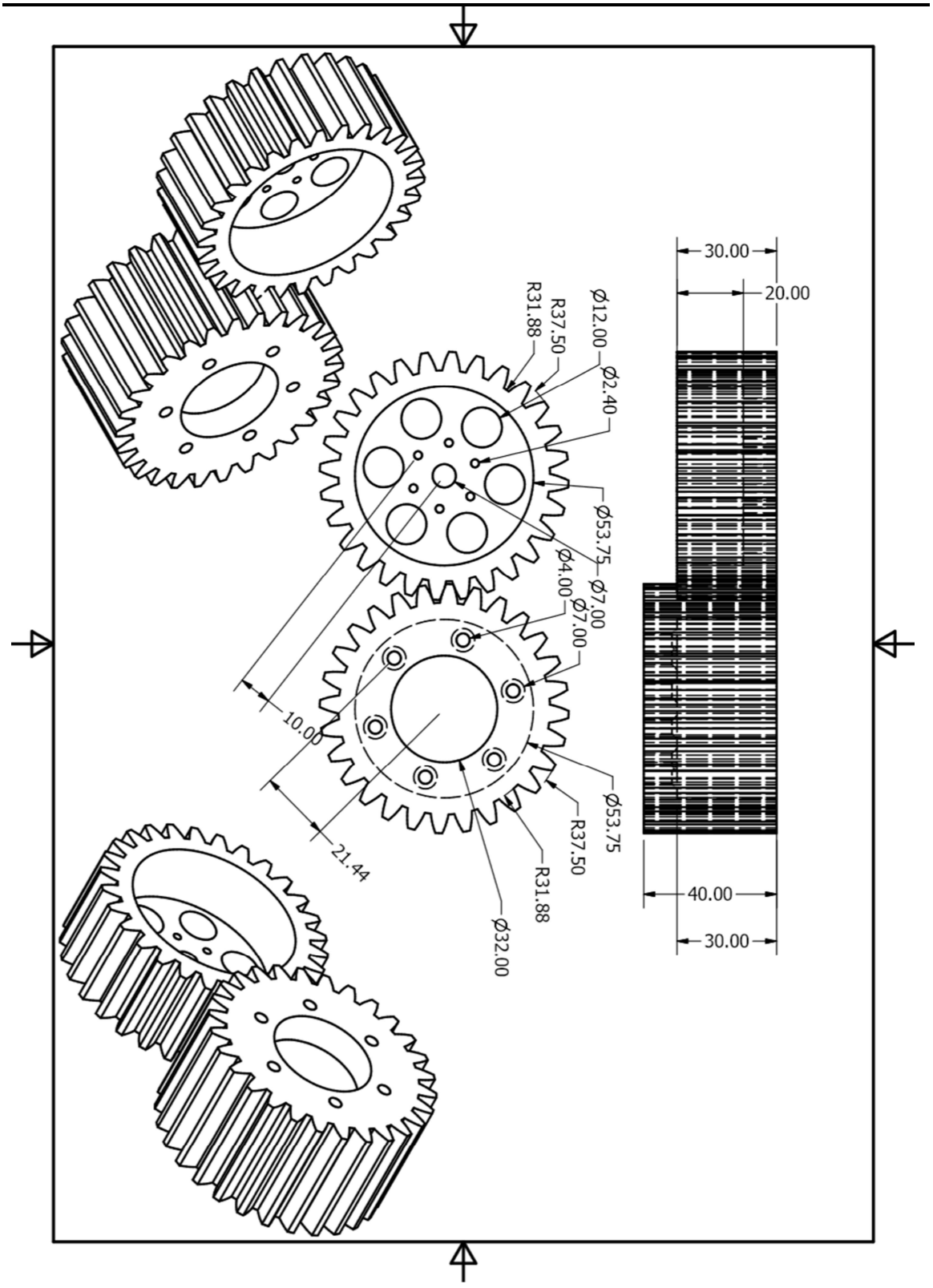
ANEXO 9: Desenho da estrutura *pan* de movimento horizontal



ANEXO 10: Desenho das engrenagens de movimento vertical



ANEXO 11: Desenho das engrenagens de movimento horizontal



Para o desenho dos fluxogramas apresentados neste projeto, foi utilizada a ferramenta de desenho online do site www.draw.io.

Para o desenho dos circuitos apresentados neste projeto, foi utilizada a ferramenta online de desenho do site www.digikey.com/schemeit/.

Para os desenhos técnicos foi utilizado o programa Autodesk Inventor 2017.

Para desenvolvimento do código do robô, foi utilizado o ambiente de desenvolvimento IDLE 2 para Raspbian OS.