Paulo Jerônimo Fagundes

SISTEMA ELETRÔNICO PARA CONTROLE DE UMA MÁQUINA DE LANÇAMENTO DE BOLAS DE TÊNIS

Horizontina-RS

2023

Paulo Jerônimo Fagundes

SISTEMA ELETRÔNICO PARA CONTROLE DE UMA MÁQUINA DE LANÇAMENTO DE BOLAS DE TÊNIS

Trabalho Final de Curso apresentado como requisito parcial para a obtenção do título de bacharel em Engenharia de Controle e Automação da Faculdade Horizontina, sob a orientação do Prof. Me Douglas de Castro Karnikowski.

Horizontina-RS

2023

FAHOR - FACULDADE HORIZONTINA

CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

A Comissão Examinadora, abaixo assinada, aprova o trabalho final de curso

“Sistema eletrônico para controle de uma máquina de lançamento de bolas de tênis”

Elaborada por:

Paulo Jerônimo Fagundes

Como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Controle e Automação

Aprovado em: 08/12/2023

Pela Comissão Examinadora

_____

Me. Douglas de Castro Karnikowski

Presidente da Comissão Examinadora - Orientador

_____

Dr. Geovane Webler

FAHOR – Faculdade Horizontina

_____

Dr. Rafael Luciano Dalcin

FAHOR – Faculdade Horizontina

Horizontina - RS

2023

RESUMO

O presente trabalho tem pro objetivo principal o projeto, prototipação e aplicação de um sistema eletrônico de controle e comunicação para uma máquina de lançamento de bolas. Ao abordar o tema, a possibilidade de aprofundamento e aplicação de conceitos e especificações de componentes eletrônicos se apresenta e contribui para uma melhor aplicação dentro de um sistema completo afim de se atender objetivos previamente definidos, chamados de requerimentos. A partir de um conjunto mecânico previamente construído e a necessidade de um sistema para controle, iniciou-se o desenvolvimento e prototipação do sistema para possibilitar o controle e configuração da máquina através de um *smartphone*. A iniciativa de desenvolvimento preenche a lacuna eletrônica da máquina e abre portas para o desenvolvimento completo de um produto, sendo esse produto uma máquina de lançamentos com aplicação em esportes de raquete, trazendo melhorias à treinamentos e repetição e eliminando atividades repetitivas através do controle e automação de um sistema.

Palavras-chave: Controle, Automação, IoT, Desenvolvimento de produto, Tênis.

ABSTRACT

The main objective of this work is the design, prototyping and application of an electronic control and communication system for a ball throwing machine. When approaching the topic, the possibility of deepening and applying concepts and specifications of electronic components presents itself and contributes to a better application within a complete system in order to meet previously defined objectives, called requirements. Based on a previously built mechanical set and the need for a control system, the development and prototyping of the system began to enable control and configuration of the machine using a smartphone. The development initiative fills the machine's electronic gap and opens doors for the complete development of a product, this product being a throwing machine with application in racket sports, bringing improvements to training and repetition and eliminating repetitive activities through control and automation of a system.

Keywords: Control, Automation, IoT, Product development, Tennis.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O tênis é um esporte apaixonante que exige habilidades técnicas, agilidade e resistência física dos jogadores. Para aprimorar suas habilidades e treinar adequadamente, os jogadores de tênis muitas vezes dependem de máquinas de lançamento de bolas de tênis. Essas máquinas desempenham um papel crucial no treinamento, permitindo que os jogadores aprimorem seu jogo, aperfeiçoem a precisão dos seus golpes e a eficiência dos seus movimentos.

Nesse contexto, o desenvolvimento de um projeto de automação para máquina de lançamento de bolas de tênis torna-se extremamente relevante. Esse projeto busca criar um sistema de controle preciso e eficiente, capaz de fornecer um treinamento personalizado e de alta qualidade para jogadores de tênis de todos os níveis. A automação dessa máquina oferece várias vantagens significativas. Em primeiro lugar, ela permite um controle preciso da velocidade, direção e efeito dos lançamentos das bolas de tênis, permitindo aos jogadores adaptarem seus treinamentos de acordo com suas necessidades específicas. Além disso, a automação elimina a necessidade de um operador manual, proporcionando uma experiência mais independente e flexível para o jogador.

Um aspecto fundamental desse projeto é a eficiência energética. Ao desenvolver um sistema de automação otimizado, é possível reduzir o consumo de energia, aumentando a autonomia da máquina e minimizando os custos operacionais. Isso torna a máquina de lançamento de bolas de tênis mais sustentável e economicamente viável para jogadores, treinadores e academias. Além disso, a precisão e a eficiência proporcionadas pelo projeto de automação permitem que os jogadores se concentrem em aprimorar suas habilidades, sem se preocupar com a consistência dos lançamentos. Isso resulta em um treinamento mais produtivo e eficaz, contribuindo para o desenvolvimento técnico dos jogadores e para seu desempenho em competições.

Em resumo, o projeto de automação para máquina de lançamento de bolas de tênis oferece uma solução inovadora para aprimorar o treinamento e o desempenho dos jogadores de tênis. Com um controle preciso e eficiente, essa tecnologia proporciona um treinamento personalizado, aumenta a autonomia da máquina e melhora a experiência geral do jogador.

1.1 JUSTIFICATIVA DO TEMA

A justificativa para o desenvolvimento do projeto de automação para máquina de lançamento de bolas de tênis baseia-se na necessidade de aprimorar o treinamento dos jogadores, proporcionando um controle preciso e eficiente dos lançamentos. As máquinas de lançamento de bolas de tênis existentes no mercado muitas vezes possuem limitações em relação à personalização dos treinamentos, à consistência dos lançamentos e ao consumo de energia. Portanto, a automação desse equipamento surge como uma solução promissora para atender às demandas dos jogadores, treinadores e academias, proporcionando um treinamento mais produtivo, sustentável e eficaz. Segundo Rodrigues (2019), a busca por aprimoramento técnico através do treinamento com máquinas de lançamento pode elevar muito a capacidade e velocidade de desenvolvimento do praticante, dessa forma, trazendo grande benefício ao usuário final.

1.2 PROBLEMA DE PESQUISA

O problema de pesquisa deste projeto é: Como desenvolver um sistema de automação para máquina de lançamento de bolas de tênis que ofereça controle preciso e eficiente dos lançamentos, permitindo um treinamento personalizado e de alta qualidade para jogadores de tênis de diferentes níveis? Diversas literaturas expõe a importância do treino e repetição para aperfeiçoamento técnico, sendo assim, a aplicação de uma máquina capaz de trazer dinamismo e customização ao treino incrementa o desenvolvimento exponencialmente.

1.3 DELIMITAÇÃO DO TEMA

O projeto de automação para máquina de lançamento de bolas de tênis concentra-se na melhoria do controle, da eficiência energética e da personalização dos lançamentos. Serão consideradas tecnologias e estratégias que permitam ajustar a velocidade, direção e efeito das bolas de tênis lançadas, bem como a automatização dos processos de controle, visando aprimorar o treinamento dos jogadores.

1.4 OBJETIVOS

1.4.1 Objetivo geral

O objetivo geral deste projeto é desenvolver um sistema de automação para máquina de lançamento de bolas de tênis que proporcione controle preciso e eficiente dos lançamentos, visando aperfeiçoar o treinamento dos jogadores de tênis.

1.4.2 Objetivos específicos

a) Projetar e implementar um sistema de controle avançado que permita ajustar a velocidade, direção e efeito dos lançamentos das bolas de tênis**;**
b) Desenvolver uma interface de usuário intuitiva e amigável, que possibilite aos jogadores personalizar seus treinamentos de acordo com suas necessidades e níveis de habilidade.

1.5 HIPÓTESE

A automação da máquina de lançamento de bolas de tênis, com um sistema de controle preciso e eficiente, resultará em um treinamento mais produtivo, sustentável e personalizado para os jogadores de tênis, melhorando sua performance técnica e competitiva.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CONCEITOS BÁSICOS DE AUTOMAÇÃO E CONTROLE

De acordo com Silva *et al*. (2020), o controle é o processo de monitorar e regular o desempenho de um sistema, garantindo que ele opere de acordo com os parâmetros desejados e preestabelecidos. Na visão de Torres e Souza (2019), a automação e o controle estão intimamente relacionados, pois a automação fornece os meios para executar tarefas automaticamente, enquanto o controle garante que essas tarefas sejam realizadas de maneira eficiente e precisa.

Para Oliveira e Santos (2017), a automação e o controle são fundamentais para otimizar processos industriais, reduzir erros e aumentar a produtividade, permitindo a tomada de decisões mais rápidas e precisas. Seguindo a linha de pensamento de Marques e Costa (2021), a automação e o controle são áreas de estudo interdisciplinares, que envolvem conhecimentos em eletrônica, mecânica, programação e sistemas de informação, visando a melhoria contínua de sistemas e processos.

2.1.1 Funcionamento e características das máquinas de lançamento de bolas de tênis

Segundo Mendes (2019), as máquinas de lançamento de bolas de tênis são dispositivos projetados para simular o lançamento de bolas de forma repetitiva e controlada, proporcionando aos jogadores uma prática consistente e aprimorando suas habilidades. De acordo com Santos e Almeida (2020), as máquinas de lançamento de bolas de tênis são equipadas com mecanismos que podem variar a velocidade, a direção e a altura dos lançamentos, permitindo treinos específicos para diferentes aspectos do jogo, como o saque, a recepção e as jogadas de fundo de quadra.

Na visão Oliveira (2018), as máquinas de lançamento de bolas de tênis são construídas com componentes eletrônicos e mecânicos, como motores, rodas de arremesso e sensores, que trabalham em conjunto para controlar o lançamento das bolas de forma precisa e consistente. Para Sousa *et al.* (2020), as máquinas de lançamento de bolas de tênis podem ser ajustadas para fornecer diferentes padrões de lançamento, como *topspin, slice* e bolas planas, replicando as características dos

golpes executados pelos jogadores profissionais. Seguindo a linha de pensamento de Rodrigues (2019), as máquinas de lançamento de bolas de tênis são equipamentos versáteis que podem ser utilizados tanto por jogadores amadores quanto por profissionais, oferecendo uma forma eficiente de treinamento e aprimoramento técnico.

2.2 ESTUDO DAS TÉCNICAS DE CONTROLE UTILIZADAS EM MÁQUINAS DE LANÇAMENTO DE BOLAS DE TÊNIS

De acordo com Silva e Santos (2022), o controle proporcional-integral-derivativo (PID) é uma das técnicas mais comumente utilizadas em máquinas de lançamento de bolas de tênis, permitindo ajustar e manter a velocidade de lançamento de forma precisa e estável. Segundo Mendonça e Almeida (2021), o controle por realimentação é amplamente empregado em máquinas de lançamento de bolas de tênis, utilizando sensores para monitorar e ajustar continuamente o desempenho do mecanismo de lançamento, garantindo uma resposta rápida e precisa às variações de velocidade e trajetória.

Na visão de Costa e Pereira (2023), o controle por malha aberta é utilizado em máquinas de lançamento de bolas de tênis para realizar lançamentos programados com parâmetros predefinidos, sem a necessidade de feedback em tempo real, sendo ideal para treinos padronizados. Para Sousa *et al.* (2020), o controle baseado em algoritmos de aprendizado de máquina tem sido objeto de estudo em máquinas de lançamento de bolas de tênis, permitindo a otimização dos parâmetros de lançamento com base em análise de dados e adaptação contínua às características do jogador.

2.3 O EFEITO DE MAGNUS (*MAGNUS EFFECT*)

O efeito de Magnus, ou *Magnus Effect*, é um efeito gerado na trajetória de uma esfera em função da rotação em seu próprio eixo, causando uma diferença de pressão e escoamento de fluido no entorno da esfera, tendo como consequência uma elevação ou queda acelerada em sua trajetória.

Para Kenyon *et al*. (2016), as diferentes aplicações do efeito de Magnus dificultam a compreensão da física dos fluídos envolvida na equação, dificultando no estabelecimento de um modelo matemático claro e conciso na magnitude e direção

do efeito. O efeito de Magnus é facilmente visualizado em esportes como tênis ou beisebol, o popularmente chamado *spin* é uma técnica altamente aprimorada para benefício e êxito dos jogadores em suas funções, com diferentes aplicações em diferentes situações de jogo.

2.4 PROJETO DO SISTEMA DE CONTROLE

Segundo Santos (2022), o projeto cuidadoso do sistema de controle é essencial para garantir a precisão e consistência dos lançamentos de bolas de tênis. Um sistema de controle eficiente é fundamental para personalizar os treinamentos e atender às necessidades individuais dos jogadores de tênis.

De acordo com Martins (2021), a automação do sistema de lançamento de bolas de tênis requer um projeto de controle preciso, capaz de ajustar a velocidade, direção e efeito dos lançamentos. O projeto do sistema de controle para máquinas de lançamento de bolas de tênis deve levar em consideração a segurança do jogador e a prevenção de acidentes. Um sistema de controle eficiente e personalizado proporciona um treinamento mais produtivo e ajuda os jogadores de tênis a aprimorarem suas habilidades Fernandes (2018).

2.4.1 Descrição das etapas de controle: detecção da bola, cálculo da trajetória, acionamento do mecanismo de lançamento

A detecção precisa da bola é crucial para garantir o controle adequado do sistema de lançamento. Sensores avançados e algoritmos de processamento de imagem são utilizados para detectar e rastrear a bola em tempo real Rodrigues (2022). Segundo Santos (2021), o cálculo da trajetória da bola envolve a análise de diversos fatores, como velocidade inicial, ângulo de lançamento, efeito desejado e condições ambientais. Algoritmos matemáticos são empregados para determinar a trajetória ideal de acordo com os parâmetros definidos.

O acionamento do mecanismo de lançamento é realizado com base nos cálculos da trajetória e nas informações obtidas pela detecção da bola. Sistemas de controle de atuadores são responsáveis por acionar o mecanismo com precisão e no momento adequado para garantir o lançamento correto da bola (Silva, 2020).

De acordo com Oliveira (2022), a integração harmoniosa entre a detecção da bola, o cálculo da trajetória e o acionamento do mecanismo de lançamento é

essencial para proporcionar um controle preciso e eficiente dos lançamentos. O sincronismo adequado entre essas etapas é fundamental para alcançar os resultados desejados.

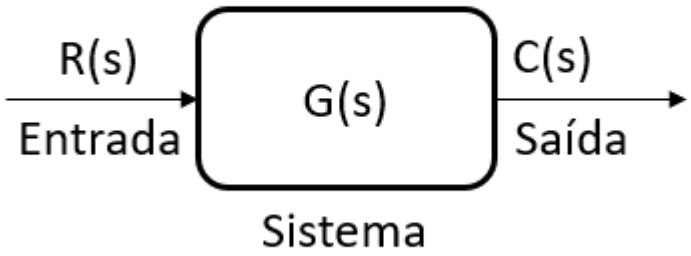
A detecção da bola, o cálculo da trajetória e o acionamento do mecanismo de lançamento devem trabalhar em conjunto para garantir a consistência e a qualidade dos lançamentos. Um sistema de controle bem projetado e integrado é a chave para atingir essa sincronia e oferecer um treinamento eficaz aos jogadores de tênis Fernandes (2023).

2.4.2 Detalhamento do processo de implementação do sistema

2.4.2.1 Sistema Malha-Aberta

São sistemas que apresentam um valor pré-determinado na saída, ou um valor conhecido a partir de uma entrada. Em sistemas de malha aberta, não tem como mudar o controle durante o processo. Por isso muitas vezes este tipo de sistema é dito que não tem controle. Os sistemas de malha aberta são caracterizados por apresentar apenas uma entrada e uma saída, conforme o diagrama de blocos mostrado na Figura 1:

Figura 1 – Sistema de malha aberta



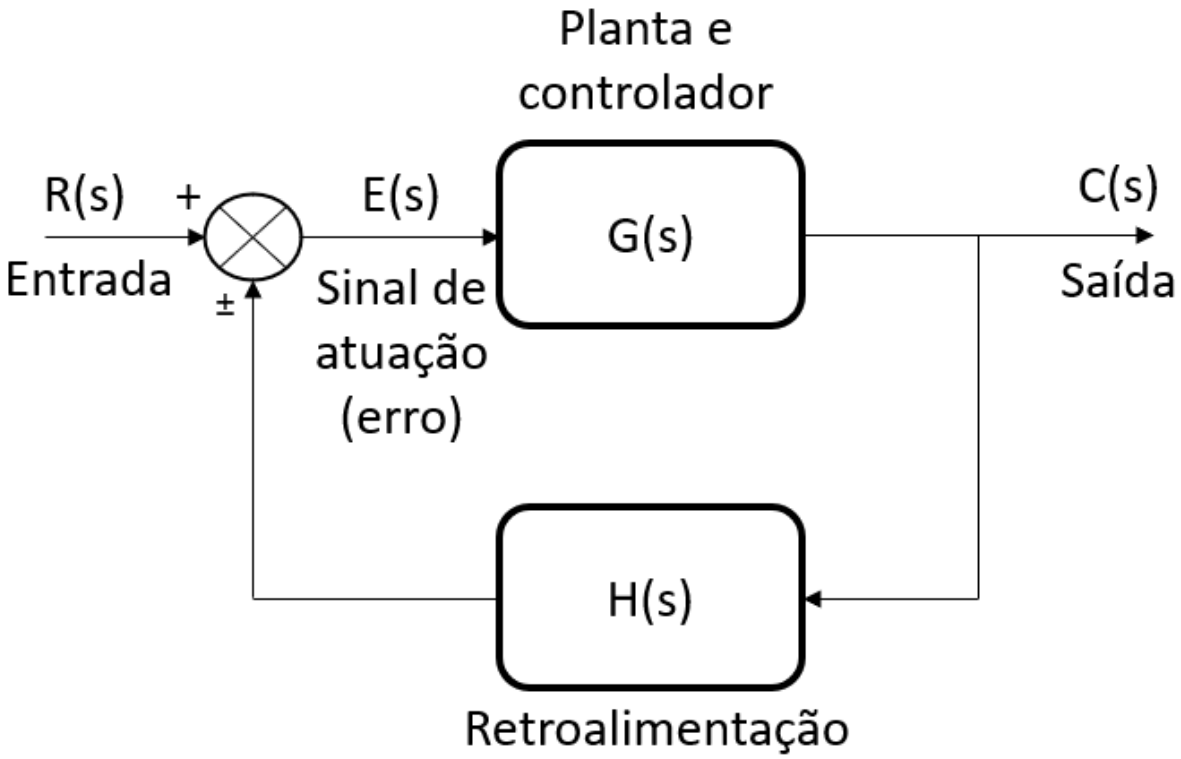
Fonte: Ogata, 2010

2.4.2.2 Sistema Malha-Fechada

São sistemas que podem ter uma saída controlada a partir de uma entrada. Em sistemas de Malha fechada, temos a possibilidade de ter controle durante o processo. Para ter controle num sistema, obrigatoriamente precisa estar em Malha Fechada. Os Sistemas de Malha Fechada são monitorados constantemente e

apresentam realimentação negativa. Um sistema Malha fechada pode ser representado no diagrama de blocos da Figura 2:

Figura 2 – Sistema de malha-fechada



Fonte: Ogata, 2010

Em um sistema de Malha Fechada, o valor de Saída do Sistema é comparado constantemente com o valor de Entrada (Valor Desejado-*SetPoint*) do sistema. À diferença entre a entrada e saída irá gerar um sinal de erro. Quanto mais o erro for próximo de zero, melhor será o controle. Um sistema de controle em malha fechada é um sistema no qual o sinal de saída possui um efeito direto na ação de controle, isto é, trata-se de um sistema de controle com realimentação (OGATA, 2010)

2.4.2.3 Sistema De Controle Pid

O PID é o controlador mais usado na indústria. Existem bilhões de loops de controle e os PIDs são usados por mais de 95% deles Åström e Murray (2012). O controlador PID é ensinado na maioria dos cursos de automação.

Controlador proporcional integral derivativo, controlador PID ou simplesmente PID, é uma técnica de controle de processos que une as ações derivativa, integral e proporcional, fazendo assim com que o sinal de erro seja minimizado pela ação proporcional, zerado pela ação integral e obtido com uma velocidade antecipativa

pela ação derivativa. De acordo com Ioannidis (2014) o PID é a metodologia mais utilizada quando falamos de algoritmos de controle de malha fechada de alta precisão. Ele é constituído pela junção dos controladores Proporcional, Integrativo e o Derivativo.

É baseado na resposta de uma malha de processo industrial a ser controlada Segundo Ogata (2010), mais da metade dos controladores industriais em uso atualmente emprega controle PID.

O controle PID é dividido em três formas de controles distintas são elas:

- Controle Proporcional:

O controle proporcional nada mais é que um controle proporcional ao erro atual no sistema, quanto maior o erro maior será o ajuste realizado pelo sistema, sempre buscando garantir uma melhor resposta. Segundo Nise (2012) a ação proporcional ajusta a saída proporcionalmente ao erro atual do sistema.

- Controle Integral:

O controle integral, ou ação integral atua junto com o controle proporcional para remover os erros que o controle proporcional sozinho não consegue eliminar segundo Kuo e Golnaraghi (2010) a ação integral é responsável por corrigir o erro em regime permanente, especialmente quando há uma perturbação constante no sistema. Através da acumulação do erro ao longo do tempo e do ajuste proporcional à integral do erro, a ação integral permite que o controlador PID atinja um estado de equilíbrio em que o erro permanece próximo de zero.

- Controle Derivativo:

O controle derivativo é o último tópico do sistema de controle PID, ele usa a ação derivativa para eliminar um erro da ação integral ajudando a diminuir a oscilação do sistema segundo Ogata (2010) A ação derivativa utiliza a taxa de variação do erro para prever a tendência futura. Através desse componente, a saída é ajustada proporcionalmente à taxa de variação do erro.

3 METODOLOGIA

A metodologia utilizada neste trabalho foi qualitativa e descritiva. A abordagem qualitativa foi escolhida para permitir uma compreensão mais profunda e contextualizada do fenômeno estudado, ou seja, o desenvolvimento de um sistema de controle para uma máquina de lançamento de bolas de tênis.

A pesquisa descritiva foi adotada para descrever e analisar detalhadamente o processo de desenvolvimento do sistema de controle. Essa abordagem visa retratar com precisão os passos, procedimentos e etapas seguidos durante a implementação do sistema, fornecendo informações claras e objetivas sobre suas características, funcionamento e desempenho.

Ao considerar o desenvolvimento e construção de um protótipo funcional e bem estruturado, uma fundamentação teórica bem desenvolvida e clara delimitação do problema e funcionalidades são inteiramente necessárias. O levantamento teórico realizado previamente embasa conceitos de controle e lançamento de bolas de tênis à fundamentos de eletrônica básica, afim de suportar o desenvolvimento e implementação de um sistema de controle de uma máquina de lançamento de bolas de tênis.

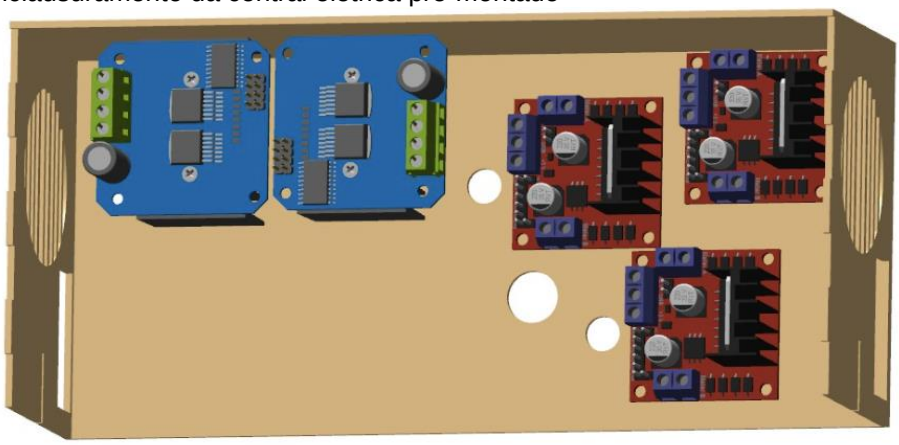
Ao final do trabalho, as considerações finais englobam o levantamento teórico inicialmente realizado, a construção do protótipo e aplicação de conceitos bem como os problemas enfrentados e soluções desenvolvidas para se atingir o objetivo final de obter um sistema eletrônico capaz de controlar diversas funções da máquina e simplificar a interação com o usuário.

3.1 RECURSOS NECESSÁRIOS

O protótipo, em seu desenvolvimento e construção, visa a utilização de componentes presentes no mercado de eletrônica, que atendam diretamente o que lhes é proposto possuindo as especificações necessárias para comportar os requerimentos do projeto. Para a construção do protótipo foram necessários recursos como um *software* de modelagem 3D para projeto da central elétrica, local de fixação para os componentes junto à máquina. Os desenhos detalhados dos componentes estão dispostos no Apêndice A. As peças foram projetadas com ranhuras para encaixe e fixadas com pequenas quantidades de cola, considerando

que nenhuma está sujeita esforços, apenas para manter os componentes unidos, organização exibida na Figura 3.

Figura 3 – Enclausuramento da central elétrica pré-montado



Fonte: o autor, 2023

A proposta do projeto eletrônico consiste em componentes compatíveis com a plataforma de prototipagem comum, sendo os componentes já construídos como módulos para interação com um microcontrolador, sendo ele o ESP32-WROOM-32U. Para *design* e especificação do esquema elétrico foi utilizado o *software* de código aberto Frizting, a mesma está disposta no Apêndice B. O Quadro 1 apresenta as características presentes no microcontrolador:

Quadro 1 – Características do ESP32-WROOM-32U

Microcontrolador	LX6
Tensão de alimentação	5V
Tensão de entrada (recomendada)	3.3 a 9 V
Tensão de entrada (limites)	6-20 V
I/O	34 pinos
Pinos de entrada analógica	15
Corrente contínua por pino I/O	40 mA
Corrente contínua para o pino 3.3V	90 mA @ 5V
Memória	4MB flash, 448kB ROM, 536kB SRAM
Velocidade de Clock	2.4GHz

Fonte: Espressif, 2023

Componentes de eletrônica embarcada utilizados no projeto do sistema:

- ESP32;
- Sensor de efeito Hall;
- Motores de passo NEMA23;
- Motores DC 12V Imobras 101407512;

- Drive L298N;
- Drive BTS7960B;
- Regulador de tensão 7805;
- Bateria 12V.

Para desenvolvimento do sistema e interface com o usuário:

- Microsoft Visual Code;
- Linguagem C/C++;
- Linguagem HTML e CSS.

Para o desenvolvimento do aplicativo de interface com o usuário, fora utilizada a ferramenta *Microsoft Visual Studio Code*, sendo um *software* grátis com diversos recursos para desenvolvimento de programação em diversas linguagens. Desde a construção da programação do sistema em linguagem C/C++ utilizando os recursos do *Visual Studio Code* e também a programação da interface do usuário em linguagem HTML, facilmente testada em qualquer navegador de internet.

3.2 DEFINIÇÃO DE REQUERIMENTOS

Para a construção de um projeto eletrônico, é necessário a definição de requerimentos que o projeto atenda, sendo eles diretamente objetivos a serem cumpridos ao longo do projeto. Sendo eles:

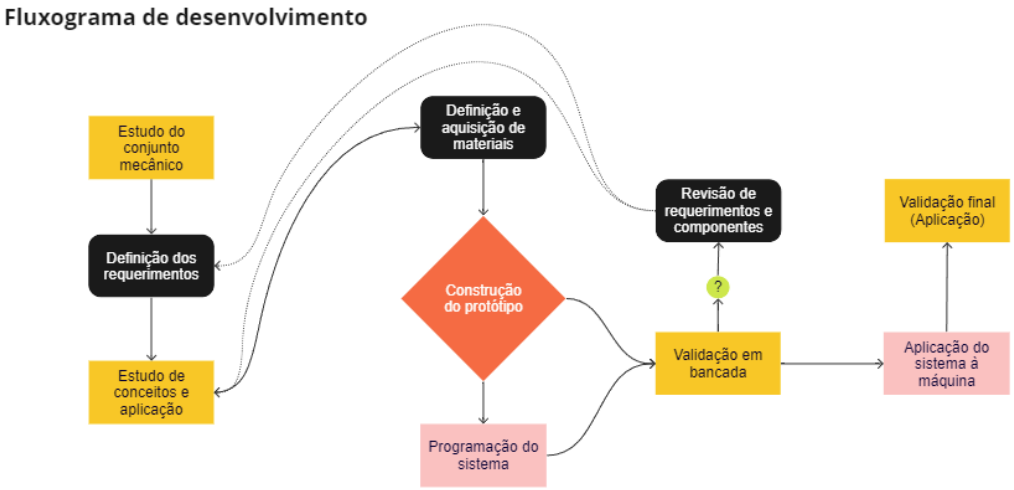
- Controle de rotação variável e independente entre as polias para viabilizar escolha de potência de lançamento e realização de *topspin/underspin*, a critério do usuário;
- Controle de sobrecarga na bandeja para identificar travamento de bolas na alimentação;
- Reversão automática do sentido de giro da bandeja quando detectado travamento;
- Controle de rotação variável na bandeja de alimentação, resultado em uma frequência de lançamento variável a critério do usuário.
- Ajuste horizontal de lançamento entre esquerda, direita e centro, viabilizando escolha de treino a critério do usuário.
- Ajuste vertical automático para compensação de trajetória considerando potência de lançamento e *topspin/underspin*;
- Interface do usuário amigável e intuitiva;

- Todos os lançamentos, independente da configuração do usuário, são lançamentos realizados em que a bola cai em alguma posição válida dentro dos limites da quadra oposta.

4 APRESENTAÇÃO E ANÁLISE DE RESULTADOS

A partir desse capítulo serão apresentados os resultados obtidos ao longo do caminho trilhado durante o desenvolvimento do sistema eletrônico. Uma abordagem detalhada da escolha, aplicação e interação de cada componente dentro do sistema, validação da funcionalidade e precisão de lançamentos. Todos os passos estão demonstrados na Figura 4, que apresenta o fluxograma de desenvolvimento do projeto.

Figura 4 – Conceito da máquina de lançamento

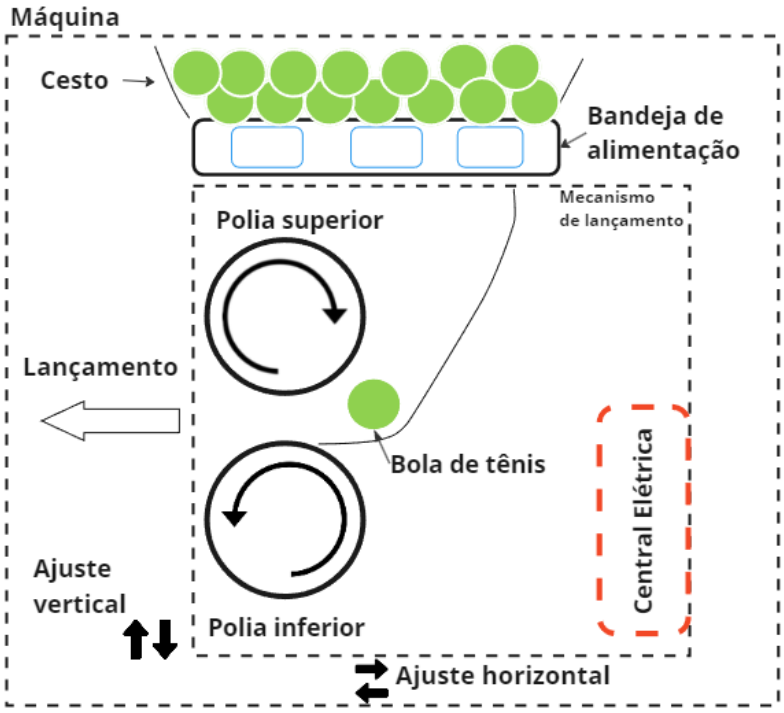


Fonte: o autor, 2023

4.1 CONCEITO DA MÁQUINA

Para definição do sistema, um conjunto mecânico previamente concebido e construído foi analisado afim de se desenvolver um sistema de controle. A máquina consiste em duas polias, uma superior e uma inferior com sentidos de giros opostos para lançar uma bolinha quando alimentada, como demonstrado com uma vista lateral pela Figura 5. O conceito de obter uma polia superior e uma polia inferior permite a aplicação de *topspin* e *underspin*, recursos importantes e muito presentes no jogo de tênis.

Figura 5 – Conceito da máquina de lançamento



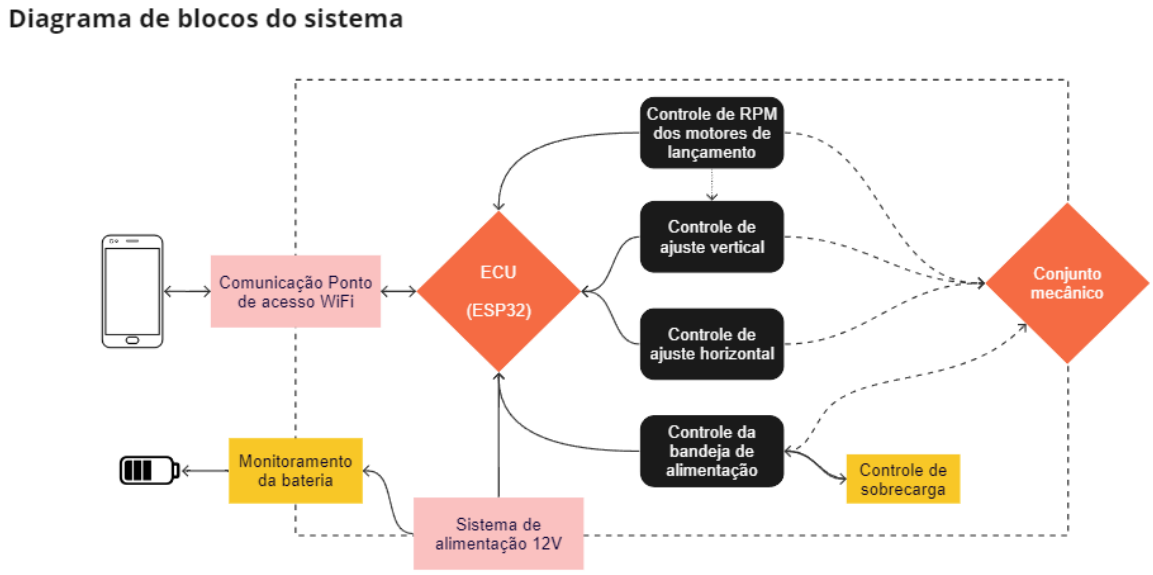
Fonte: o autor, 2023

A máquina também possui uma bandeja de alimentação com quatro aberturas do tamanho de uma bola, a bandeja rotaciona 360º infinitamente alimentando uma bola por vez para a canaleta que fica posicionada abaixo e direciona a bolinha até as polias. Acima da bandeja à um cesto articulado com capacidade de comportar 100 bolas. Dentro da estrutura da máquina e abaixo do cesto, há uma estrutura independente com um ponto de pivotamento permitindo o ajuste horizontal da direção de lançamento. Integrado à essa mesma estrutura independente se encontra um mecanismo de pivotamento das polias, permitindo o ajuste vertical da direção de lançamento.

4.2 PROJETO ELETRÔNICO

A execução de um projeto requer diversas etapas preparatórias como requerimentos, levantamento de dados, e construção de um diagrama de blocos do funcionamento geral do sistema, afim de obter uma compreensão macro do sistema e simplificar a compreensão e as relações entre módulos. A Figura 6 demostra as o diagrama de blocos para o proposto projeto, ilustrando suas aplicações e diferentes subsistemas.

Figura 6 – Motor elétrico DC 12V 101407512



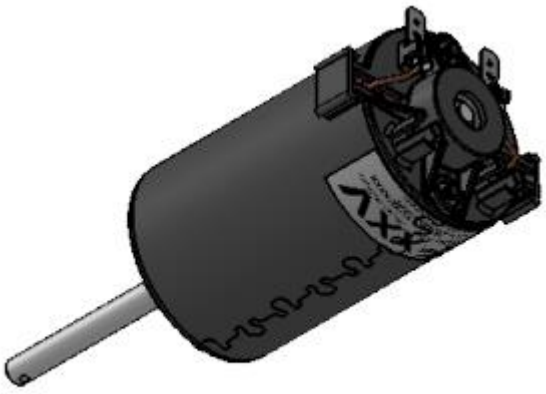
Fonte: o autor, 2023

Todos os subsistemas, componentes definidos e suas aplicações são descritas nas sessões seguintes.

4.2.1 Acionamento polias

Para acionamento das polias, fora escolhido o motor elétrico DC 12V 101407512 da Imobras demonstrado na Figura 7, as especificações técnicas do motor estão descritas no Quadro 2.

Figura 7 – Motor elétrico DC 12V 101407512



Fonte: Imobrás, 2023

Quadro 2 – Características motores DC

Valores Nominais	
Tensão (V)	13
Corrente (A)	9.5
Potência (W)	85
RPM (min-¹)	4050
Torque (N.m)	0,2
Massa(kg)	0,8

Fonte: Imobrás, 2023

Os motores DC 12V de acionamento das polias são controlados pelos *drives* BTS7960B, um drive de ponte H mais robusto construído para controlar maiores cargas de corrente, demonstrado na Figura 8.

Figura 8 – Drive BTS7960B

Fonte: Usinainfo, 2023a

4.2.2 Acionamento bandeja

O acionamento da bandeja também é realizado com um motor DC 12V AK360/25PL12S3500S da Akiyama Motors de menor capacidade, afinal requer menor RPM e não realiza esforço, apenas conduz as bolas para a canaleta. O motor está representado na Figura 9 e suas especificações no Quadro 3.

Figura 9 – Motor elétrico DC 12V Akiyama AK360/25PL12S3500S

Fonte: Techmakers, 2023a

Quadro 3 – Características motores DC

Valores Nominais	
Tensão (V)	12
Corrente (mA)	40
Potência (W)	0.7
RPM (min-¹)	3500
Torque (N.m)	0,02
Massa(kg)	0,5

Fonte: Techmakers, 2023a

O motor DC 12V da bandeja é controlado pelo drive L298N, um drive de ponte H construído para controlar cargas indutivas, solenoides e motores, demonstrado na Figura 10.

Figura 10 – Drive L298N

Fonte: Usinainfo, 2023b

Afim de atender um dos requerimentos definidos pela máquina, caracterizado no item 4.1, o sistema elétrico da bandeja também consta um sensor de efeito Hall ACS712, demonstrado na Figura 11:

Figura 11 – Sensor de efeito Hall ACS712

Fonte: Usinainfo, 2023c

O ACS712 é integrado em série com o motor DC 12V da bandeja, assim que o travamento de bolas ocorrer na bandeja, a corrente do motor DC 12V da bandeja se eleva gerando um campo magnético no sensor, capturado e interpretada pelo circuito integrado e convertido em tensão proporcional.

4.2.3 Ajustes horizontal e vertical

Os requerimentos também englobam ajustes vertical e horizontal na direção de lançamento da bola, para isso os mecanismos de ajuste descritos no item 4.1 são controlados por dois motores de passo NEMA-23, demonstrado na Figura 12 e caracterizado tecnicamente pelo Quadro 4.

Figura 12 – Motor de passo NEMA-23

Fonte: Techmakers, 2023b

Quadro 4 – Especificação técnica Motor de Passo NEMA-23

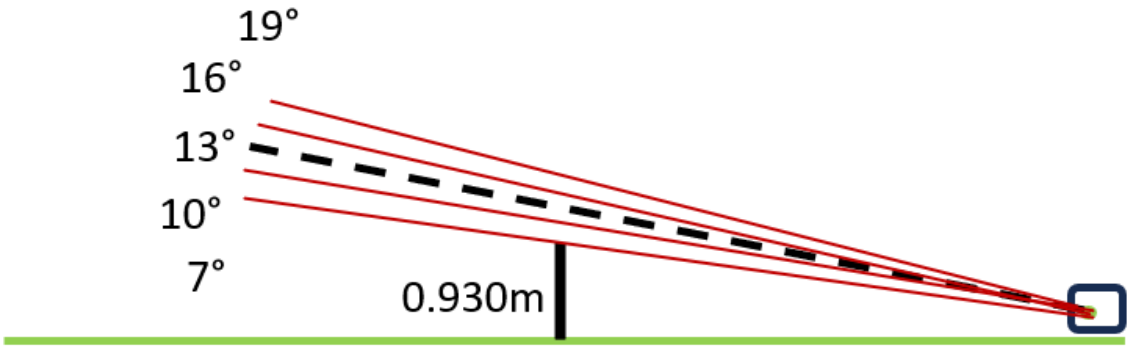
Item	Especificação
Ângulo de passo	1.8º
Número de passos	200
Enrolamento	Espiras bifilares
Temperatura máxima de operação	80º C
Temperatura ambiente	-10ºC – 50º C
Corrente / Fase	3A
Nº de Fases	2
Resistência	1.3 Ohms
Peso	0.6 kg

Fonte: Techmakers, 2023b

O drive de ponte H L298N, descrito no item 4.2.1, também é utilizado para realizar o controle e acionamento dos motores de passo NEMA-23, em uma razão de 1:1, cada NEMA-23 possuindo um drive L298N individual. O L298N, através de sua capacidade de controlar dois motores DC independentes, é conectado às duas fases do NEMA-23 podendo realizar o controle independente das fases para execução do passo. A lógica de programação realiza o controle fazendo uso da biblioteca *myStepper*, criando objetos *Stepper* para cada motor e executando o acionamento intercalado das fases.

O ajuste vertical é realizado condicionalmente conforme as escolhas do usuário para potência de lançamento e *spin*, realizando a compensação automática de ângulo de lançamento para garantir a trajetória adequada de lançamento. O ângulo inicial calibra em 13º, a variação de ângulo máxima é ±6 º, conforme demonstrado na Figura 13:

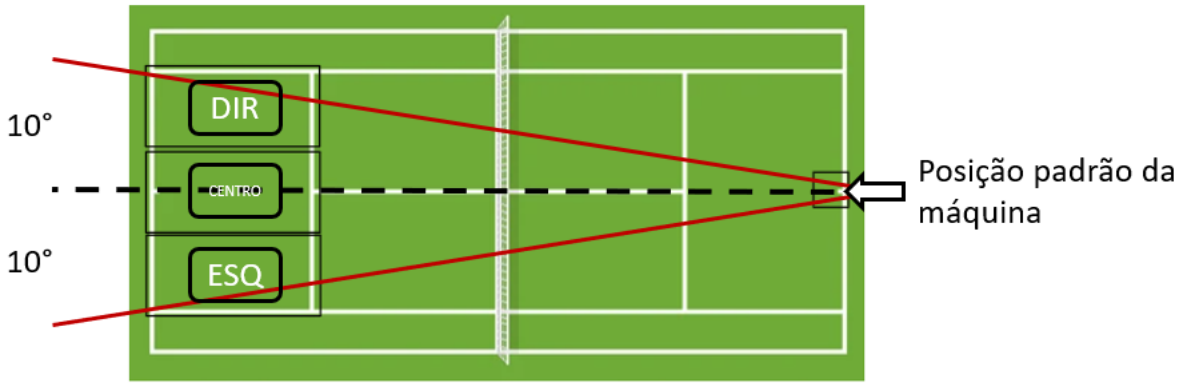
Figura 13 – Esquema de ajuste vertical



Fonte: o autor, 2023

O ajuste horizontal é realizado conforme as escolhas do usuário para direção de lançamento, variando 10º para cada lado de acordo com a exemplificação demonstrada na Figura 14:

Figura 14 – Esquema de ajuste horizontal



Fonte: o autor, 2023

As posições selecionadas para ajuste horizontal e vertical são gravadas na EEPROM para armazenamento e retomada dos dados após o ciclo de alimentação na máquina.

4.2.4 Microcontrolador ESP32-WROOM-32U

Para ser o cérebro por trás de toda a operação, foi escolhido o ESP32 por suas características e principalmente a conectividade. O ESP32-WROOM-32U, demonstrado na Figura 15, apresenta características interessantes, descritas no Quadro 1, que agregam ao projeto como as portas GPIO, *General Purpose In/Ou*, em português portas Entrada/Saída de Propósito Geral e a conectividade *WiFi/Bluetooth* disponível, atendendo às necessidades do projeto.

Figura 15 – Microcontrolador ESP32-WROOM-32U

Fonte: Espressif, 2023

4.2.4.1 Ponto de Acesso *WiFi*

Uma forma de utilizar o ESP32 e seu recurso de conectividade *WiFi* é na forma de Ponto de Acesso. O Ponto de Acesso consiste em transformar o ESP32 para além de um microcontrolador para atuar como emissor de uma rede privada e *off-line*, hospedando uma página *web* e exibindo os recursos para interação e exibição de informações com usuário através da conexão WiFi com o microcontrolador. O benefício para a aplicação em questão é que elimina a necessidade de componentes individuais para controle e comunicação, unindo ambos em um componente apenas. Também facilita a hospedagem da página *web* através do sistema de arquivos *SPIFFS.*

4.2.4.2 - SPIFFS

O *SPIFFS*, ou SPI *Flash File System*, é um recurso disponível a partir da inclusão da biblioteca SPIFFS que viabiliza o gerenciamento e armazenamento interno de arquivos a memória flash do ESP32, executando operações de leitura, gravação, renomeação e exclusão de arquivos. O *SPIFFS* também permite a definição de formato e tamanho de partição do armazenamento destinado ao sistema de arquivos viabilizando diversos tipos de operações como coleta e exportação de dados, EEPROM e entre outros mais.

Os sistemas de ajustes vertical e horizontal, pelo fato de não possuírem *input* ou *feedback* de posição, foi adotado o uso de gravação dos dados das posições horizontal e vertical na EEPROM, afim de armazenar e restaurar os dados em uma nova utilização e evitar a perda da informação de posição. A utilização da EEPROM também permite a criação e inserção de treinos pré-definidos para seleção.

4.3 SENSORIAMENTO DE LANÇAMENTO

Com o objetivo de monitorar o funcionamento da máquina constantemente, um sistema de sensoriamento do lançamento é necessário. O sensor TCRT5000 foi escolhido para desempenhar o papel de interpretar a alimentação de bolas para as polias. O sensor, demonstrado na Figura 16, é posicionado na calha que conduz as bolas da bandeja até as polias, dessa forma, a bola passa em frente ao emissor e fototransistor, gerando oscilação na leitura e interpretando a passagem.

Figura 16 – Sensor óptico reflexivo TCRT5000

Fonte: Usinainfo, 2023d

4.4 SISTEMA DE ALIMENTAÇÃO

A alimentação do sistema consiste em uma bateria 12V 12Ah para suprir todo o sistema, seja eletrônico ou de potência. O sistema de potência consiste na alimentação dos motores diretamente pela bateria, sendo eles motores das polias, bandeja e ajustes horizontal e vertical. O sistema eletrônico é alimentado a partir do ESP32, que por sua vez recebe alimentação do regulador de tensão 7805, demonstrado na Figura 17:

Figura 17 – Regulador de tensão 7805

Fonte: Usinainfo, 2023e

O regulador de tensão, em conjunto com capacitores eletrolíticos de 1uF, desempenham o papel de regular a tensão de 12V fornecida pela bateria, rebaixando para a tensão necessária para alimentação do microcontrolador ESP32. O mesmo também auxilia na estabilidade da tensão de alimentação. O sistema de alimentação também consta com um indicador e testador de carga da bateria com display LCD BW-LY6W, demonstrado na Figura 18. Os parâmetros técnicos do componente estão descritos no Quadro 5.

Figura 18 – Indicador e testador de bateria BW-LY6W

Fonte: Usinainfo, 2023f

Quadro 5 – Especificação técnica Motor de Passo NEMA-23

Parâmetro	Min	Max
Tensão de operação (V)	8	63
Consumo de corrente (mA)		5.0
Range de temperatura (º C)	0	40
Peso (g)	15	

Fonte: o autor, 2023

4.5 LED INDICADOR

Visando obter algum tipo de indicador visual que possa ser interpretado a uma certa distância ou sem fazer uso do telefone, um LED RGB foi incluído no sistema. O Quadro 6 demonstra as cores exibidas pelo LED e seus significados.

Quadro 6 – Codificação de cores LED RGB

Cor	Significado
Azul	Em funcionamento
Vermelho	Problema enfrentado
Verde	Máquina ativada, mas estática
Amarelo	Reversão da bandeja
Ciano	Aguardando conexão *WiFi*
Roxo	Calibrando / Ajustando

Fonte: o autor, 2023

4.6 CUSTO DE DESENVOLVIMENTO

Afim de obter uma perspectiva de custo da implementação eletrônica e uma estimativa de custo final da máquina, o Quadro 7 detalha especificamente o custo de implementação do sistema eletrônico.

Quadro 7 – Codificação de cores LED RGB

Item	QTD	Valor (R$)	Total (R$)
ESP32-WROOM-32U	1	69,90	69,90
Motores DC 12V	2	110,15	220,30
Motor DC 12V (bandeja)	1	33,20	33,20
Motores de passo NEMA-23	2	70,91	141,82
Drive L298N	3	22,90	68,70
Drive BTS7960B	2	84,84	169,68
Módulo ACS712	1	13,90	13,90
Módulo TCRT5000	1	9,90	9,90
Regulador de tensão 7805	1	3,10	3,10
Capacitores 1uF	2	2,45	4,90
Fiação	-	50,00	50,00
MDF	-	30,00	30,00
Total	-	-	**815,40**

Fonte: o autor, 2023

Visando gerar uma estimativa aproximada do custo total da máquina, considerando o valor de R$ 1200,00 reais para construção de toda a estrutura e conjuntos mecânicos, somados aos R$ 815,40 reais de custo total do sistema elétrico, chegamos à um custo final de R$ 2015,40 reais. Ao comparar com o valor de mercado máquinas de lançamento de bolas similares, com valores variando entre R$ 6000,00 à R$ 15000,00, visualiza-se uma boa oportunidade de comercialização devido à enorme diferença de valor.

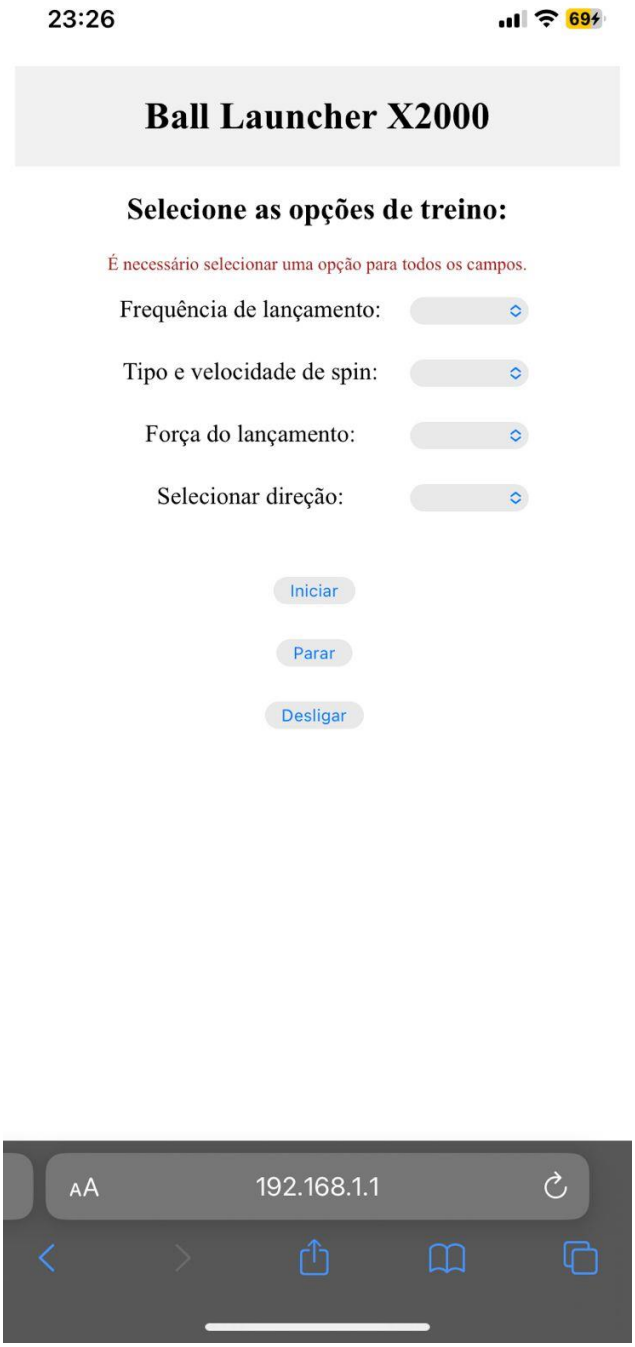
4.7 INTERFACE DO USUÁRIO

A partir da funcionalidade Ponto de Acesso do ESP32 descrita no item 4.2.4, a interface do usuário foi construída como uma página *web*, programada em HTML e

estilizada através do CSS. Os códigos estão dispostos nos Apêndices C e D. Visando atender o requerimento de obter uma interface do usuário simples e intuitiva, demonstrada na Figura 19, a página dá acesso direto ao menu de opções com aspecto limpo, opções claras e diretas e simplicidade na escolha através dos menus seletores do tipo *dropdown*. Além dos seletores, a página disponibiliza três botões de comando, sendo eles Iniciar, Parar e Desligar.

Entende-se que a interface do usuário possui fácil acesso, comum para qualquer usuário que possua um *smartphone*, situação bem comum atualmente, e disponibiliza o acesso de maneira simples, facilitando a interação com qualquer nível de conhecimento e intimidade com tecnologia. A máquina apresenta funcionalidade simples e direta, opções de escolhas claras e pertinentes à prática do esporte, conhecidas por qualquer praticante.

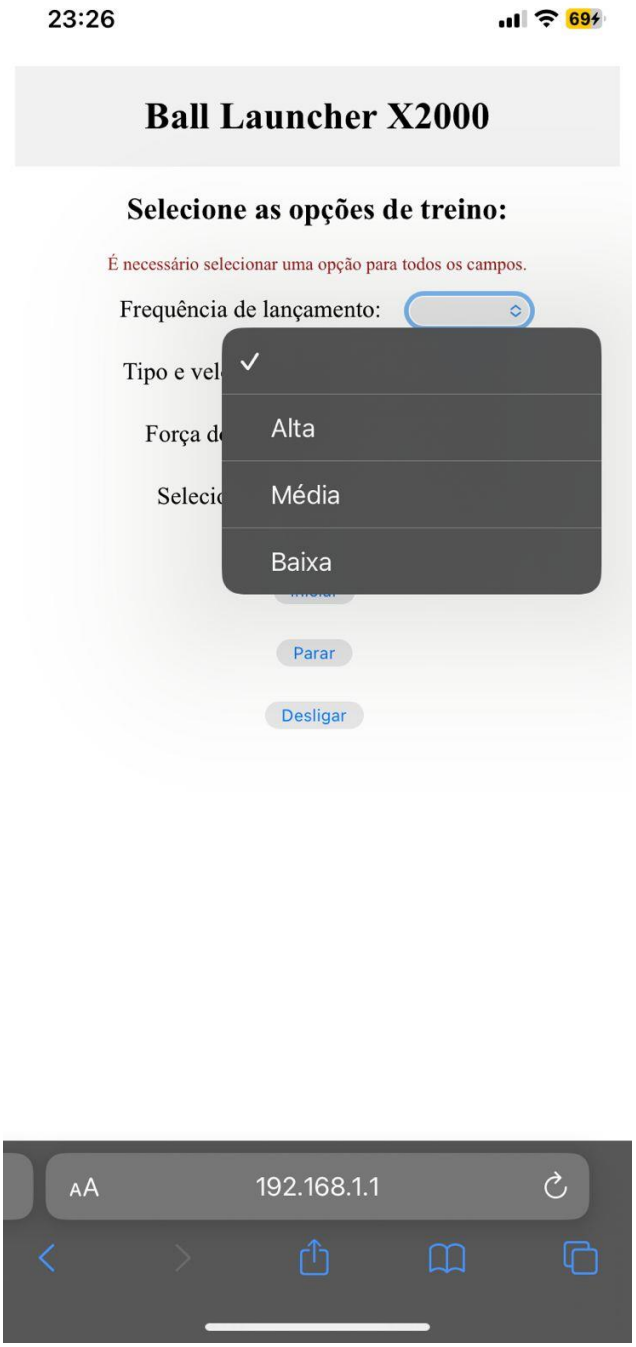
Figura 19 – Interface do usuário



Fonte: o autor, 2023

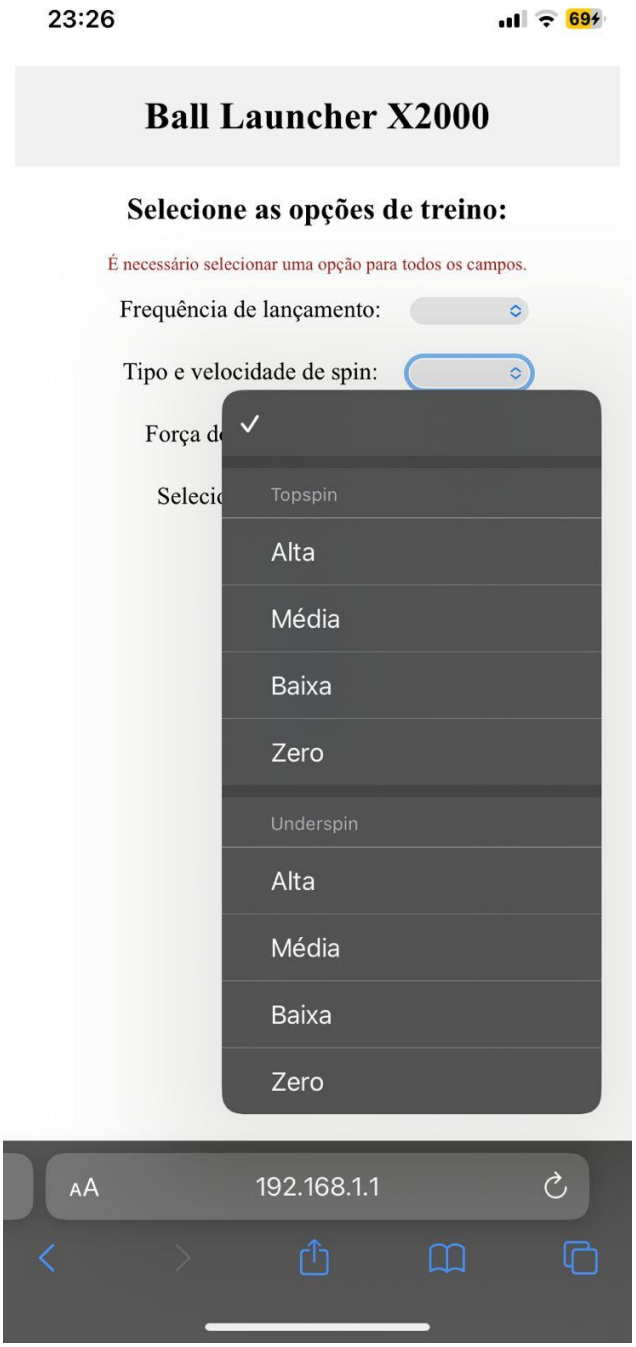
Ao considerar os critérios de escolha do usuário, fora introduzido um menu com quadro opções de seleção, sendo eles a frequência de lançamento, potência do lançamento, *topspin/backspin*, direção de lançamento e 3 botões, de início, parada e desligamento, todos demonstrados nas Figuras 20, 21, 22 e 23.

Figura 20 – Menu de seleção de frequência de lançamento



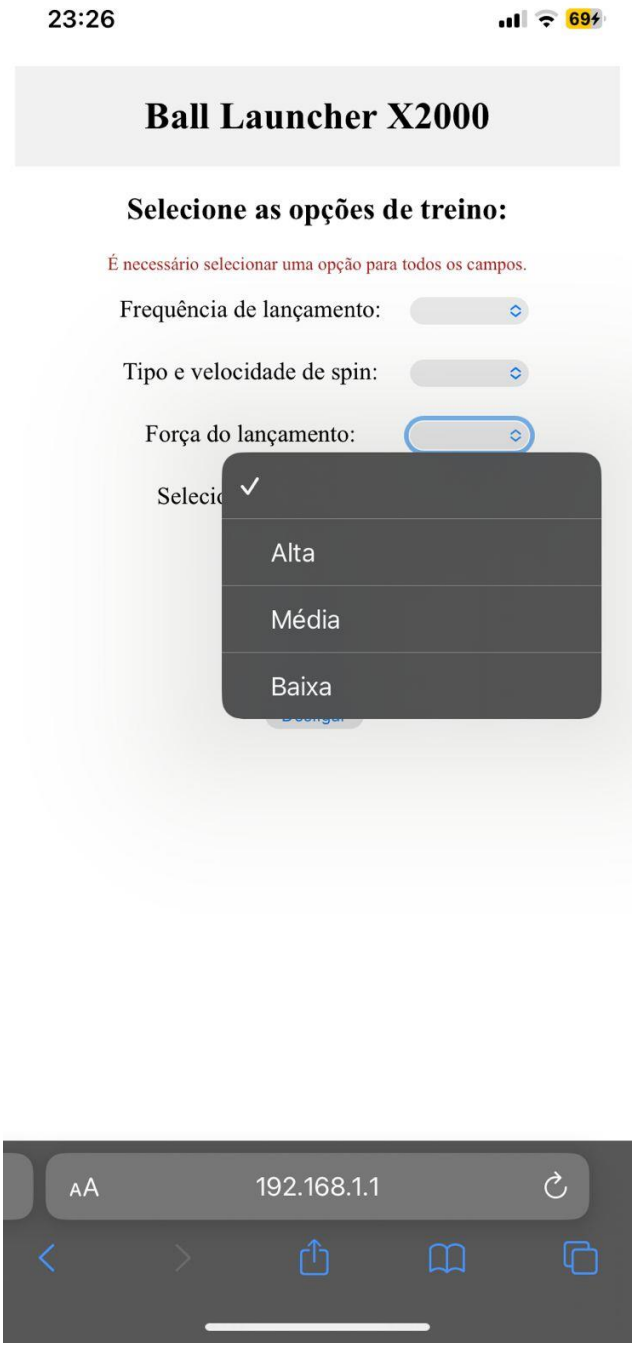
Fonte: o autor, 2023

Figura 21 – Menu de seleção *spin*



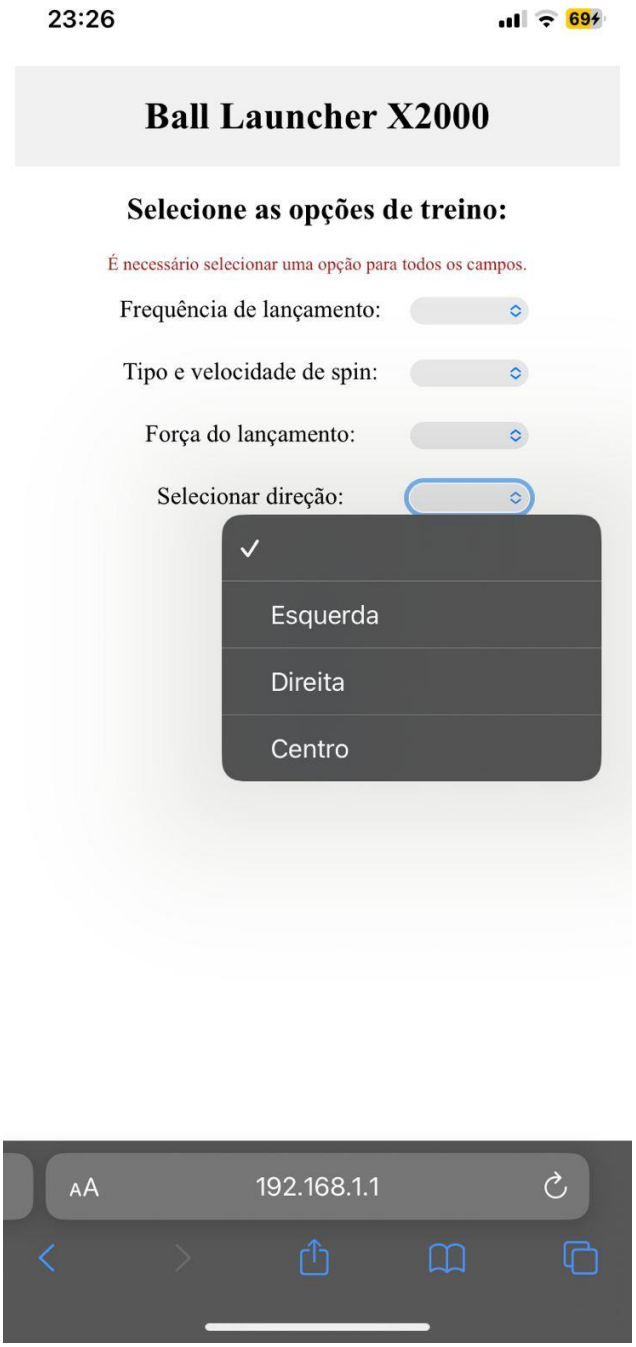
Fonte: o autor, 2023

Figura 22 – Menu de seleção potência de lançamento



Fonte: o autor, 2023

Figura 23 – Menu de seleção de direção de lançamento



Fonte: o autor, 2023

5 TESTES E VALIDAÇÃO DO SISTEMA

Conforme a aplicação do sistema, uma série de testes e coleta de dados foi planejada para proporcionar a melhor execução e construção do protótipo ao longo das etapas, afim de identificar os pontos mais críticos e endereçar soluções ou alternativas.

5.1 AMOSTRAGEM DE CARGAS

Uma instrumentação para coleta de dados referente a cargas de corrente geradas pelos motores DC 12V das polias de lançamento e bandeja de alimentação foi realizada afim de obter informações acerca do funcionamento da máquina em ócio e picos de corrente durante lançamentos. O Quadro 8 apresenta os dados obtidos.

Quadro 8 – Bateria de testes de carga

Bateria de testes											
Corrente na bandeja			**Rotação Motor Inferior**				**Rotação Motor Superior**				
Baixa rotação livre			Rotação Livre				Rotação Livre				
01	80	mA	01	20%	710	mA	01	20%	700	mA	
02	70	mA	02	40%	920	mA	02	40%	950	mA	
03	70	mA	03	50%	1200	mA	03	50%	1150	mA	
04	80	mA	04	60%	1500	mA	04	60%	1300	mA	
05	90	mA	05	70%	1550	mA	05	70%	1500	mA	
Alta rotação livre			06	80%	1600	mA	06	80%	1600	mA	
01	120	mA	07	90%	1800	mA	07	90%	1750	mA	
02	110	mA	08	100%	2050	mA	08	100%	2200	mA	
03	100	mA	Lançamento				Lançamento				
04	120	mA	01	20%	1000	mA	01	20%	1300	mA	
05	110	mA	02	40%	1400	mA	02	40%	1500	mA	
Sobrecarga			03	50%	1920	mA	03	50%	2000	mA	
01	500	mA	04	60%	2100	mA	04	60%	2100	mA	
02	420	mA	05	70%	2320	mA	05	70%	2500	mA	
03	510	mA	06	80%	2750	mA	06	80%	2800	mA	
04	550	mA	07	90%	2850	mA	07	90%	3050	mA	
05	590	mA	08	100%	3000	mA	08	100%	3500	mA	

Fonte: o autor, 2023

A partir dessa tomada de dados de carga dos motores, foi possível visualizar o subdimensionamento do drive de ponte H inicialmente proposto para os motores de lançamento. Os picos de corrente gerados no momento de lançamento da bola,

em função do aumento da carga, estavam ultrapassando os limites de corrente do *drive* e se fez necessário a aquisição de novos *drives*.

5.2 VALIDAÇÃO DO SISTEMA EM BANCADA

Para realizar a validação do sistema em bancada, antes da aplicação na máquina, um *smartphone* foi utilizado para simular a interação com o usuário através da página *web* e gerados os sinais de comunicação para o ESP32 e a lógica de programação através do *debugging* utilizando a publicação de mensagens na rede serial, visualizada no computador através da IDE Arduino. Mensagens estratégicas para compreender e validar o funcionamento da lógica foram colocadas no código e uma vez que concluída e obtidos os resultados esperados, foram removidas.

5.3 VALIDAÇÃO FINAL

Para a validação final, o equipamento foi aplicado em uma quadra de tênis, posicionado de acordo com a instrução e realizadas baterias de testes nas mais diversas configurações, demonstrado nas Figuras 24 e 25.

Figura 24 – Testagem em quadra 1

Fonte: o autor, 2023

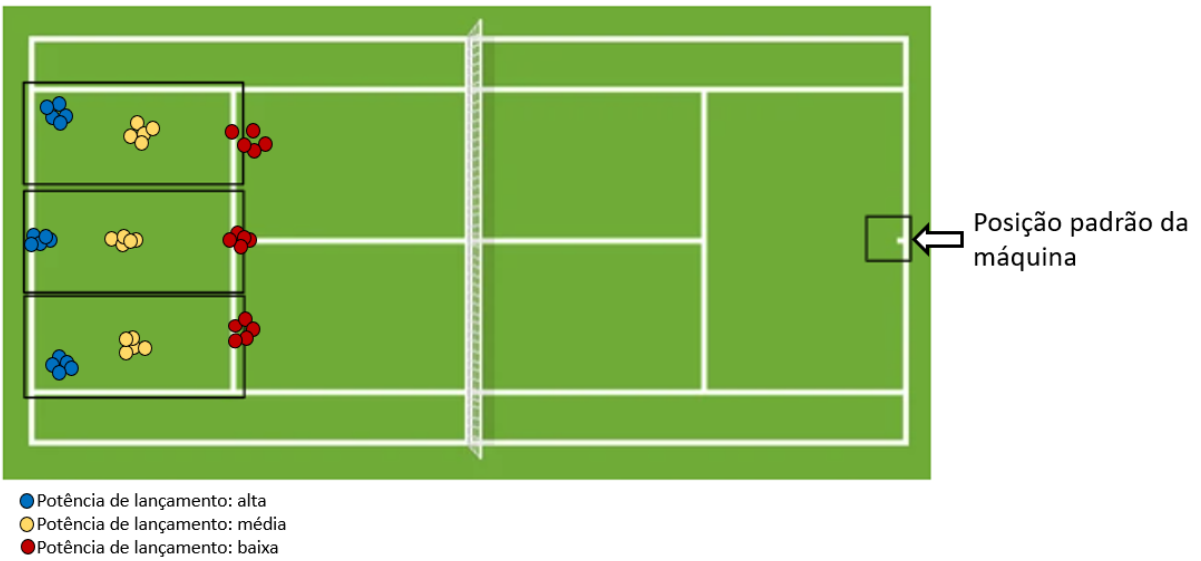
Figura 25 – Testagem em quadra 2

Fonte: o autor, 2023

O primeiro objetivo de validação era garantir que as velocidades de lançamento máxima e mínima sem *spin*, estavam lançando bolas dentro das delimitações da quadra oposta. O ajuste vertical não é utilizado para correção de lançamentos sem

spin, dessa forma, o ângulo de lançamento é mantido e a variação da potência de lançamento deve entregar lançamentos dentro da quadra. Foram realizadas 5 baterias de teste para cada direção e os resultados apresentados na Figura 26.

Figura 26 – Bateria de testes de lançamento



Fonte: o autor, 2023

Após a conclusão do *tunning* da potência de lançamento, iniciou-se os testes e *tunning* da potência de lançamento com a aplicação de *spin*, que apresentou mudanças significativas na trajetória de lançamento, conforme o item 2.3 previa.

O *spin* se dispõe em seis opções, divididas em duas categorias com três opções cada, alta, média ou baixa para *topspin* e *backspin.* Para testes e validação de todas as configurações, três baterias de testes foram executadas para cada variação de configuração, como descrito no Quadro 9.

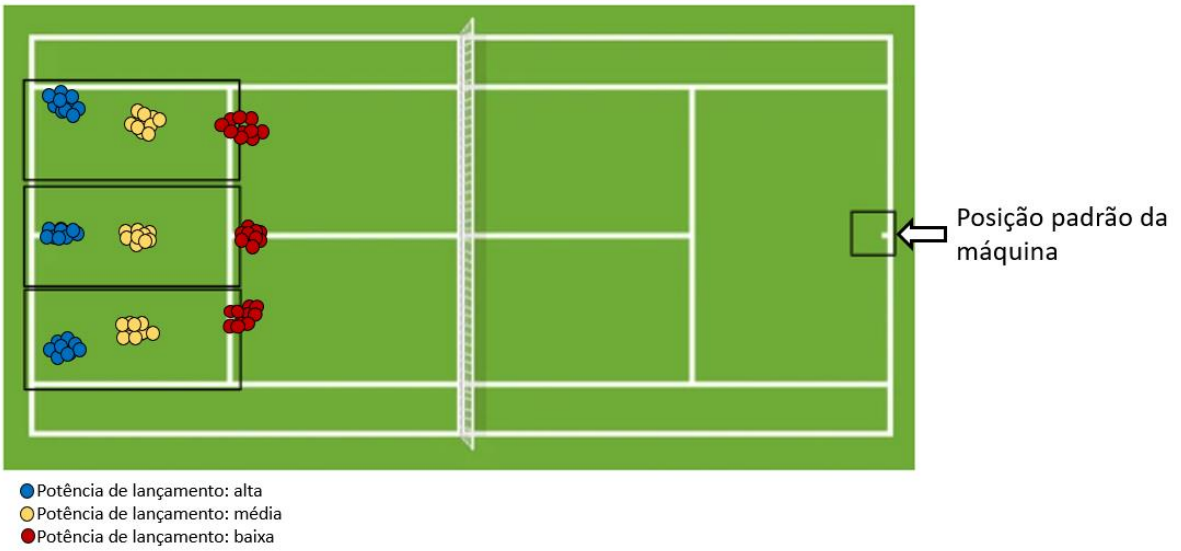
Quadro 9 – Bateria de testes variações de configuração

Potência	Spin	Potência	Spin	Potência	Spin
Alta	**Topspin**	Média	**Topspin**	Baixa	**Topspin**
	Alto		Alto		Alto
	Medio		Medio		Medio
	Baixo		Baixo		Baixo
	Backspin		**Backspin**		**Backspin**
	Alto		Alto		Alto
	Medio		Medio		Medio
	Baixo		Baixo		Baixo

Fonte: o autor, 2023

Ao contemplar todas as variações de configurações, obtemos dezoito situações diferentes para testar, sendo três baterias para cada configuração, chegamos ao total de 54 baterias de teste totais para validação da configuração utilizando *spin*. Os resultados obtidos estão resumidos na Figura 27.

Figura 27 – Bateria de testes de lançamento com *spin*



Fonte: o autor, 2023

À medida que o *tunning* foi concluído, a máquina se apresentava em condição de pleno funcionamento, entregando o que lhe está proposto de acordo com os requerimentos e funcional definidos.

5.4 DIFICULDADES ENCONTRADAS

Durante a etapa de validação, foi possível identificar alguns potenciais problemas e possíveis falhas no design e escopo do projeto. Parte dos problemas levantados foram facilmente contornados e ajustados ao longo do desenvolvimento. Outros problemas requerem uma análise mais detalhada ou alterações inviáveis dentro do presente cronograma.

5.4.1 Subdimensionamento *drive* ponte H

Conforme mencionado no item 5.1, os *drives* de ponte H L298N incialmente definidos para controle dos motores DC 12V das polias de lançamento apresentaram limitação de corrente. Afim de evitar riscos de falha do componente ao longo prazo,

os mesmos foram substituídos por *drives* de maior capacidade, o BTS7960B, referenciado no item 4.2.1.

5.4.2 Consumo de bateria

O sistema, que consiste basicamente em motores elétricos, apresenta um consumo de bateria significativo, tornando uma carga da bateria de 12V 12Ah suficiente para 3-4h apenas em potência baixa-média ou 2h em potência máxima. Alternativas para essa situação são a utilização de um jogo de baterias auxiliar, duas ou três unidades, ou também a introdução de uma fonte chaveada em paralelo ao sistema de bateria ou integrado para carregamento, visando a utilização de energia elétrica se disponível.

5.4.3 Central elétrica

Por não possuir projeto 3D do conjunto mecânico, o projeto da central elétrica apresentou interferências de montagem e inviabilizou a sua aplicação, dessa forma, uma central elétrica teve de ser improvisada até que um novo *design* seja realizado e solucione os problemas encontrados.

5.5 IDEIAS DE MELHORIA

Correções ou alterações consideradas muito incisivas inviabilizam sua implementação dentro do presente cronograma, mas são consideradas pertinentes ao projeto atual e são propostas como melhorias.

5.5.1 Sistema de alimentação ligado à energia elétrica e recarga da bateria

Afim de viabilizar a utilização da máquina por longos períodos de tempo, se faz necessário a implementação de um sistema capaz de utilizar energia elétrica e fonte de conversão AC/DC para funcionamento.

Em conjunto, a introdução de um sistema de recarga da bateria, embora agressiva ao design atual, se mostra como uma melhoria obrigatória devido à atual necessidade de recarga manual da bateria, atividade onerosa e desnecessária.

5.5.2 Melhorias, incrementos e polimento na interface do usuário

Devido às limitações de conhecimento na programação em HTML, a interface do usuário possui uma grande margem para melhorias e adição de funções que podem agregar a experiência do usuário, seja funcional ou apenas visual.

Aplicação da máquina em outros esportes

O *padel* e o *beach tennis* são esportes que se tornaram febre nacional, ambos fazem uso do mesmo conceito de jogo com raquete e bolinhas.

Propõe-se que, com pequenas inclusões e/ou alterações, é possível tornar a máquina compatível com estes esportes, expandindo seu alcance no esporte e mercado.

CONSIDERAÇÕES FINAIS

O cenário de esportes como tênis, *padel*, *beach tennis* e demais esportes de raquete cresce e ganha cada vez mais espaço no Brasil. São esportes que demandam treinamento, e treino significa repetição. A partir dessa premissa, a aplicação de máquinas para lançamentos de bolas é imprescindível para realização de um bom treinamento e desenvolvimento de habilidades e fundamentos. O não uso de uma máquina para automatizar o lançamento constante e frequente de bolas demanda de uma segunda pessoa realizando essa atividade, que acaba dificultando e onerando o processo de treinamento.

Ao observar o mercado desse produto, nota-se uma dificuldade na obtenção do equipamento pela carência de nacionalização do item e pelo alto custo, sendo uma *commodity* acessível apenas pela elite. O equipamento, podendo ser utilizado preferencialmente individual ou no máximo em duplas, gera um custo elevado para a instituição que o disponibiliza, acarretando em um incremento de custo no treinamento visando retorno do investimento, e consequentemente o treinamento e aprendizado do esporte passa a ser menos atrativo ou até não viável.

O desenvolvimento e prototipação de um sistema eletrônico de controle para a máquina de lançamentos evidencia a possibilidade de desenvolvimento de um produto nacional de baixo custo para competir no segmento. O sistema aborda conceitos fundamentais aprendidos ao longo da graduação e os aplica de maneira concisa e simplificada, concluindo a proposta de desenvolvimento de maneira satisfatória e capaz de atender os requerimentos e funções presentes no conceito do produto. O produto, ainda com grandes margens de melhoria, demonstra eficácia no atendimento de sua função, trazendo benefícios e baixo custo ao aprendizado e treinamento de esportes com raquete, nesse primeiro momento, focado ao tênis.

REFERÊNCIAS

ASTRÖM, K. J.; MURRAY, R. M. ***Feedback systems:*** *an introduction for scientists and engineers. Princenton: Princeton University Press,* 2012. Disponível em: http://www.cds.caltech.edu/~murray/amwiki. Acesso em 22 jul. 2023.

COSTA, E.; PEREIRA, F. Controle por malha aberta em máquinas de lançamento de bolas de tênis: treinos padronizados e lançamentos programados. **Revista de Tecnologia e Inovação**, v. 8, n. 2, p. 87-95, 2023.

ESPRESSIF. **ESP32-WROOM-32D & ESP32-WROOM-32U.** 2023. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf. Acesso em: 1 set. 2023.

FERNANDES, E. **Sistemas de controle integrado para consistência e qualidade em lançamentos:** trabalho conjunto da detecção da bola, cálculo da trajetória e acionamento do mecanismo em treinamento de jogadores de tênis. 2023.

FERNANDES, J. Sistema de controle eficiente para máquinas de lançamento de bolas de tênis: produtividade no treinamento e aprimoramento das habilidades dos jogadores. **Revista Brasileira de Tecnologia Aplicada ao Esporte**, v. 5, n. 2, p. 78-92, 2018.

IMOBRAS. **Documentação Motor DC 12V 101407512.** 2023. Disponível em: https://www.imobras.ind.br/site-2017/ajax/action.produto-download?id=141&idArquivo=1. Acesso em: 1 set. 2023.

IOANNIDIS, F. ***Intelligent controller based on raspberry Pi*****.** Dissertação (Mestrado) – *University of Manchester, Faculty of Engineering and Physical Sciences*, 2014.

KENYON, K. *On the Magnus Effect.* ***Scientific Research Publishing***, v. 8, p. 49-52, 2016.

KUO, B. C.; GOLNARAGHI, F. **Sistemas de controle automático.** Rio de Janeiro: LTC, 2010.

MARQUES, J.; COSTA, F. **Estudo interdisciplinar em automação e controle:** integrando conhecimentos em eletrônica, mecânica. Programação e sistemas de informação para a melhoria contínua de sistemas e processos. 2021.

MARTINS, I. Automação do sistema de lançamento de bolas de tênis: projeto de controle preciso e segurança do jogador. *In:* ANAIS DO SIMPÓSIO BRASILEIRO DE ENGENHARIA AUTOMÁTICA, pp. 234-246. 2021.

MENDES, A. Máquinas de lançamento de bolas de tênis: prática consistente e aprimoramento de habilidades. **Jornal de Tecnologia Esportiva**, v. 7, n. 2, p.45-58, 2019.

MENDONÇA, C.; ALMEIDA, D. Controle por realimentação em máquinas de lançamento de bolas de tênis: resposta rápida e precisa às variações de velocidade

e trajetória. *In:* ANAIS DO CONGRESSO INTERNACIONAL DE AUTOMAÇÃO E ROBÓTICA, pp. 123-135, 2021.

NISE, Norman. **Engenharia de sistemas de controle**. Rio de Janeiro: LTC, 2012.

OGATA, K. **Engenharia de controle moderno**. Rio de Janeiro: Pearson Prentice Hall, 2010.

OLIVEIRA, D. **Integração harmoniosa para controle eficiente de lançamentos:** sincronismo entre detecção da bola, cálculo da trajetória e acionamento do mecanismo de lançamento. 2022.

OLIVEIRA, P.; SANTOS, L. **Automação e controle na otimização de processos industriais:** redução de erros e aumento da produtividade para decisões rápidas e precisas. 2017.

OLIVEIRA, J. Componentes eletrônicos e mecânicos em máquinas de lançamento de bolas de tênis: precisão e consistência no treinamento. **Engenharia Mecânica no Esporte**, v. 5, n. 1, p. 23-36, 2018.

RODRIGUES, K. Detecção precisa da bola em máquinas de lançamento de bolas de tênis: sensores avançados e algoritmos de processamento de imagem em tempo real. *In*: ANAIS DO CONGRESSO NACIONAL DE ENGENHARIA ELETRÔNICA, p. 205-218, 2022.

RODRIGUES, F. Versatilidade e eficiência: utilização de máquinas de lançamento de bolas de tênis por jogadores de diferentes níveis. **Jornal de Treinamento e Desenvolvimento de Tênis**, v. 14, n. 2, p. 55-68, 2019.

SANTOS, B. **Cálculo da trajetória da bola em sistemas de lançamento:** análise de fatores como velocidade inicial, ângulo de lançamento e condições ambientais utilizando algoritmos matemáticos. 2021.

SANTOS, H. **Projeto do sistema de controle em máquinas de lançamento de bolas de tênis:** personalização dos treinamentos e atendimento às necessidades individuais dos jogadores. Tese (Doutorado) – Universidade Federal de Tecnologia, São Paulo, 2022.

SANTOS, B.; ALMEIDA, C. Aprimorando o jogo: variação e controle nas máquinas de lançamento de bolas de tênis. **Jornal Internacional de Ciência do Tênis**, v. 12, n. 4, p. 112-125, 2020.

SILVA, A.; *et al*. **Controle de sistemas:** monitoramento e regulação do desempenho de acordo com parâmetros preestabelecidos. 2020.

SILVA, A.; SANTOS, B. Controle proporcional-integral-derivativo em máquinas de lançamento de bolas de tênis. **Revista de Engenharia e Automação**, v. 15, n. 3, p. 45-60, 2022.

SILVA, C. **Acionamento preciso do mecanismo de lançamento:** controle de atuadores com base em cálculos de trajetória e informações da detecção da bola. 2020.

SOUSA, G.; *et al.* Controle baseado em algoritmos de aprendizado de máquina em máquinas de lançamento de bolas de tênis. ***Journal of Artificial Intelligence in Sports***, v. 7, n. 1, p. 32-48, 2020.

TECHMAKERS. **Documentação Akiyama AK360/ 25PL12S3500S.** 2023a. Disponível em: https://cdn.awsli.com.br/945/945993/arquivos/MicroMotorDC-R3-Sem-Red.pdf. Acesso em 30 set. 2023.

TECHMAKERS. **Documentação NEMA-23.** 2023b. Disponível em: https://recursos.techmakers.com.br/MediaCenter/NEO%2057CM23-3A-ID.pdf. Acesso em: 1 set. 2023.

TORRES, R.; SOUZA, M. **Automação e controle:** integração para a execução eficiente e precisa de tarefas automatizadas. 2019.

USINAINFO. **Documentação ACS712.** 2023c. Disponível em: https://www.usinainfo.com.br/index.php?controller=attachment&id_attachment=306. Acesso em: 1 set. 2023.

USINAINFO. **Documentação BTS7960B.** 2023a. Disponível em: https://www.usinainfo.com.br/index.php?controller=attachment&id_attachment=490. Acesso em: 1 out. 2023.

USINAINFO. **Documentação indicador de bateria BW-LY6S.** 2023f. Disponível em: https://www.usinainfo.com.br/index.php?controller=attachment&id_attachment=924. Acesso em: 1 set. 2023.

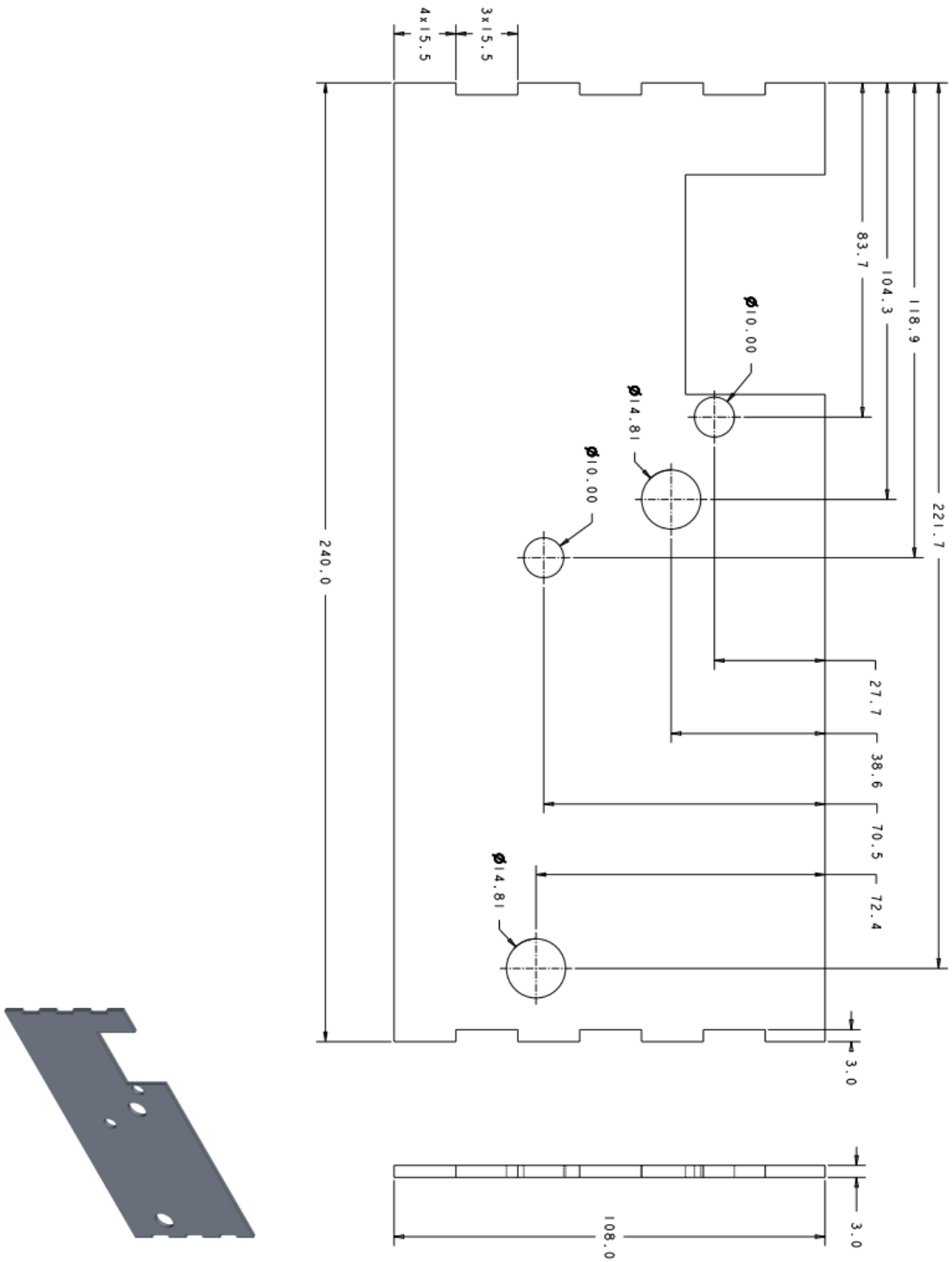
USINAINFO. **Documentação L298N.** 2023b. Disponível em: https://www.usinainfo.com.br/index.php?controller=attachment&id_attachment=82. Acesso em: 2 set. 2023.

USINAINFO. **Documentação regulador de tensão 7805.** 2023e. Disponível em: https://www.usinainfo.com.br/index.php?controller=attachment&id_attachment=568. Acesso em: 1 set. 2023.

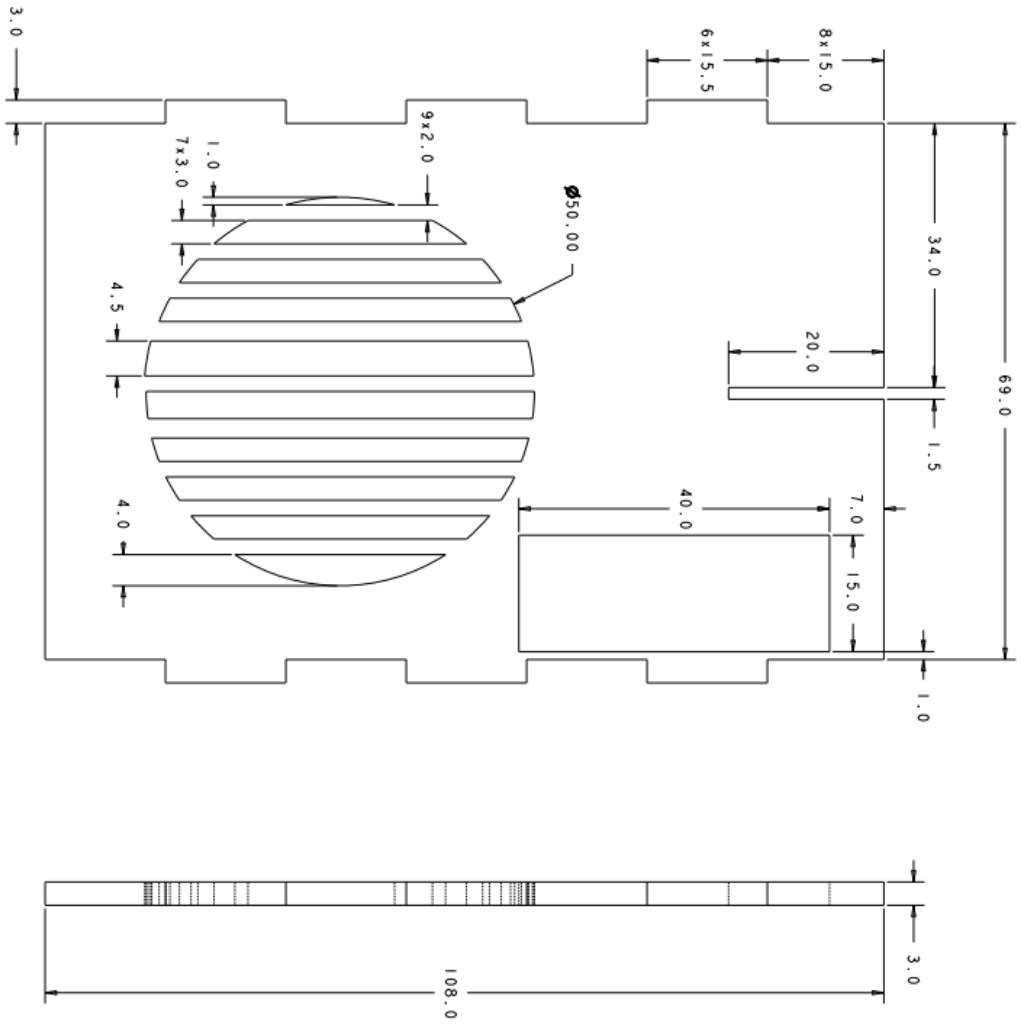
USINAINFO. **Documentação TCRT5000.** 2023d. Disponível em: https://www.usinainfo.com.br/index.php?controller=attachment&id_attachment=70. Acesso em: 2 set. 2023.

APÊNDICE A – DETALHAMENTO PEÇAS CENTRAL ELÉTRICA

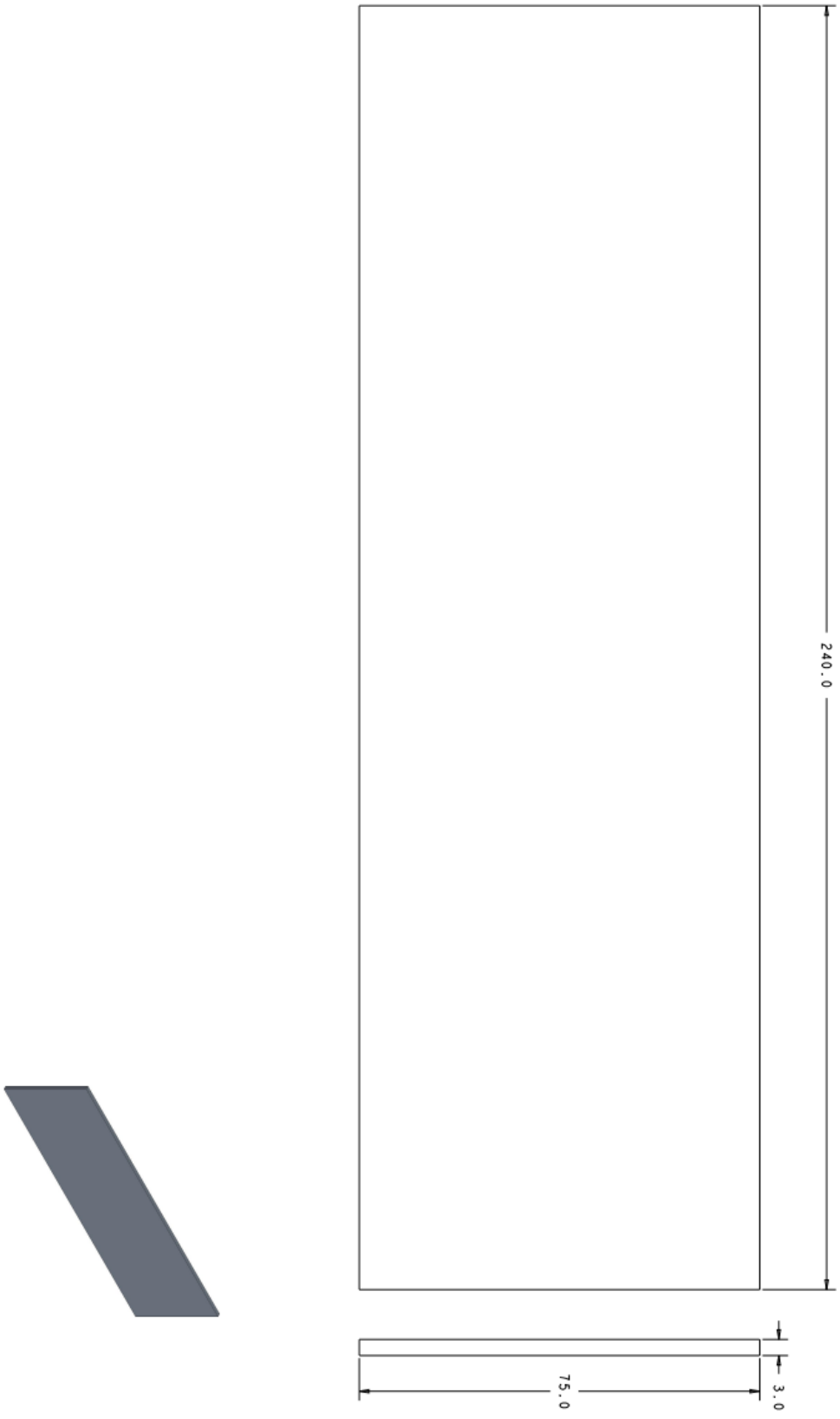
Peça 01: face (2x)



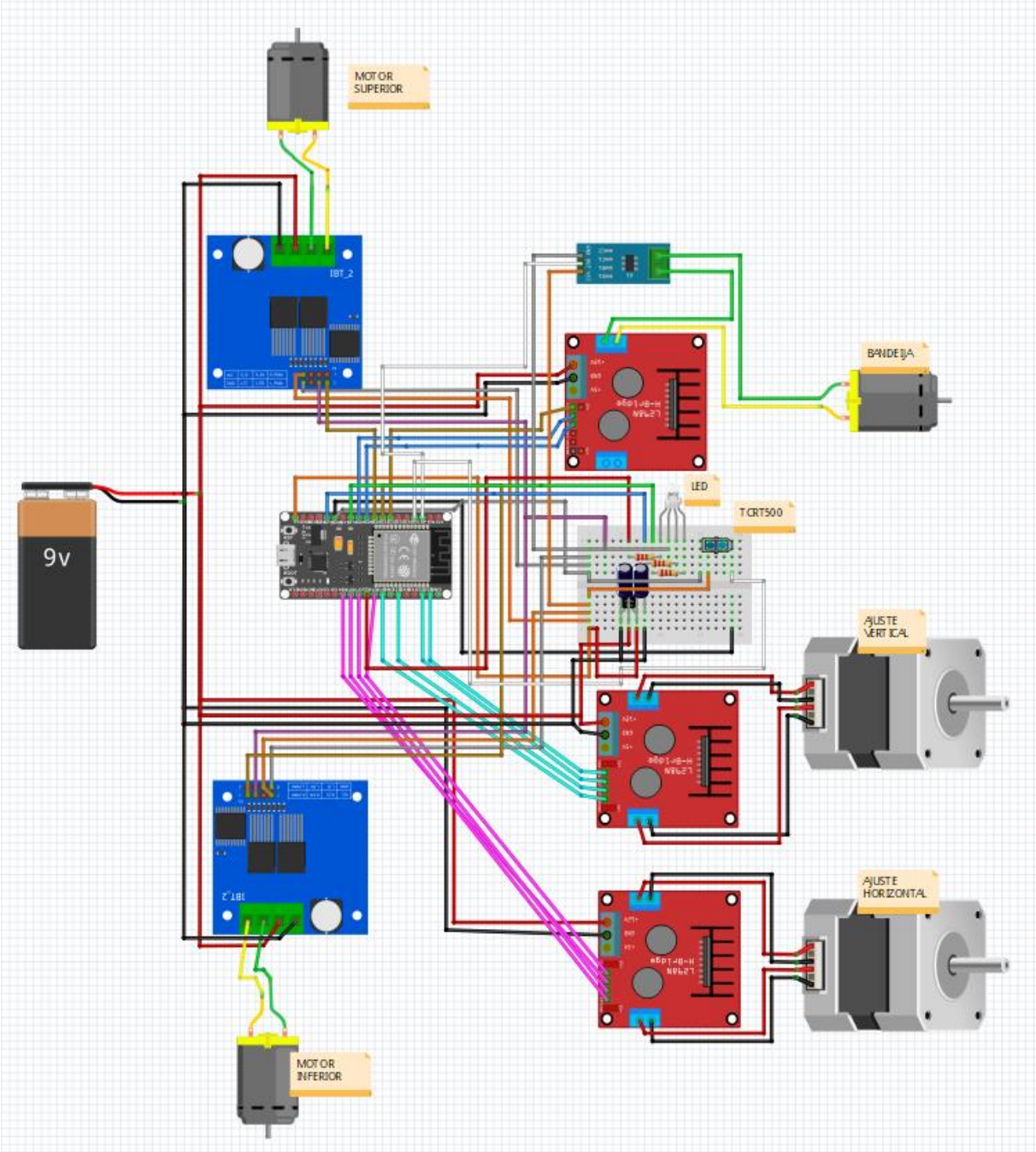
Peça 02: lateral (2x)



Peça 03: fechamento superior



APÊNDICE B – ESQUEMA ELÉTRICO



APÊNDICE C – LÓGICA DE PROGRAMAÇÃO C/C++

```
1.  // Inclusão de bibliotecas
2.      #include <math.h>
3.      #include <stdio.h>
4.      #include <Stepper.h>
5.      #include "WiFi.h"
6.      #include "ESPAsyncWebServer.h"
7.      #include "SPIFFS.h"
8.
9.  // Definição pinagem
10.
11.     // Pinagem sensor de corrente
12.     #define ACS     A0
13.
14.     //Pinagem TCRT5000
15.     #define TCRT    A1
16.
17.     // Pinagem LED
18.     #define LED_R   25
19.
20.     #define LED_G   27
21.     #define LED_B   29
22.
23.     // Pinagem drive motores DC
24.     #define m_sup_PWM   7
25.     #define m_inf_PWM   6
26.
27.     // Pinagem drive motores passo
28.     #define bandeja_PWM    5
29.     #define bandeja_IN1    22
30.     #define bandeja_IN2    23
31.
32.     #define SPR            200                          // Steps Per Rev
33.     #define ANTIHORARIO     6                           // ESQ
34.     #define HORARIO        -6                           // DIR
35.     #define UP              3
36.     #define DOWN           -3
37.     Stepper stepperVertical(SPR, 39, 41, 43, 45);
38.     Stepper stepperHorizontal(SPR, 47, 49, 51, 53);
39.
40.
41.     // Definição velocidades
42.     #define RPM_HI      100             // TBD
43.     #define RPM_MID     70
44.     #define RPM_LOW     50
45.
46.     #define B_HI        100             // TBD
47.     #define B_MID       70
48.     #define B_LOW       50
49.
50.     #define TOPSPIN_HI          1.5          // TBD
51.     #define TOPSPIN_MID         1.35
52.     #define TOPSPIN_LOW         1.15
53.     #define TOPSPIN_ZERO        1
54.
55.     #define UNDERSPIN_HI        1.5          // TBD
56.     #define UNDERSPIN_MID       1.35
57.     #define UNDERSPIN_LOW       1.15
58.     #define UNDERSPIN_ZERO      1
59.
60.     // Definição de valores
61.     #define limCorrente     0.300   // 300mA overcurrent
62.     #define limTCRT          500     // 500 infravermelho
63.
```

```
64.
65. // Declara funções
66.     void initFunc();
67.     void start(bool input);
68.     bool precondic();
69.     void stop();
70.     void SPIFFS();
71.     void eeprom_read();
72.     void eeprom_write();
73.     void WiFiAP();
74.     void cntlBandeja(int status, int bandeja_RPM);
75.     void correnteBandeja();
76.     void cntlMotores(bool status, int RPM, int topspin, int underspin);
77.     int contLancamento();
78.     bool cntlLancamento();
79.     void statusLED(LED status);
80.
81.     void set_global_RPM(int BT_RPM);
82.     void set_global_spin(int BT_spin);
83.     void set_global_bandeja_RPM(int BT_bandeja_RPM);
84.     void set_global_direcao(int dir);
85.     void set_global_vertical_pos(int pos);
86.
87.     void cntlVertical(int inclinacao);
88.     void cntlHorizontal(int direcao);
89.
90. // Global variables
91.     int global_counter,
92.         global_RPM,
93.         global_topspin,
94.         global_underspin,
95.         global_bandeja_RPM;
96.         global_direcao;
97.         global_vertical_pos;
98.
99.     bool global_start;
100.
101.  // HTML INPUTS
102.      const char* PARAM_INPUT_1 = "myConfig1";
103.      const char* PARAM_INPUT_2 = "stop";
104.      const char* PARAM_INPUT_3 = "off";
105.
106.  // Credenciais do ponto de acesso
107.      const char* ssid = "Launcher";
108.      const char* password = "20252525";
109.
110.  // Cria o objeto na porta 80
111.      AsyncWebServer server(80);
112.
113.  // Parâmetros de rede
114.      IPAddress IP(192, 168, 1, 1);
115.      IPAddress gateway(192, 168, 1, 1);
116.      IPAddress subnet(255, 255, 255, 0);
117.
118.  enum motores
119.  {
120.      ON,
121.      OFF,
122.      REVERSE
123.  }
124.
125.  enum freq_lançamento
126.  {
127.      F_HIGH,
128.      F_MID,
129.      F_LOW
```

```
130.  }
131.
132.  enum spin
133.  {
134.      T_HIGH = 3,
135.      T_MID = 4,
136.      T_LOW = 5,
137.      ZERO = 6,
138.      U_HIGH = 7,
139.      U_MID = 8,
140.      U_LOW = 9
141.  }
142.
143.  enum RPM_motores
144.  {
145.      HIGH = 11,
146.      MID = 12,
147.      LOW = 13
148.  }
149.
150.  enum direcao
151.  {
152.      ESQ = 14,
153.      DIR = 15,
154.      CENTRO = 16
155.  }
156.
157.  enum LED
158.  {
159.      RED,
160.      GREEN,
161.      BLUE,
162.      YELLOW,
163.      CYAN,
164.      PURPLE
165.  }
166.
167.  enum INCLIN
168.  {
169.      I_NONE,
170.      I_UPUP,
171.      I_UP,
172.      I_DOWN,
173.      I_DOWNDOWN
174.  }
175.
176.  void setup()
177.  {
178.      Serial.begin(115200);
179.
180.      initFunc();
181.
182.      SPIFFS();
183.      WiFiAP();
184.  }
185.
186.  void loop()
187.  {
188.      start(precondit());
189.
190.      if(!cntlLancamento())
191.      {
192.          correnteBandeja();
193.      }
194.      else
195.      {
```

```
196.         stop();
197.     }
198. }
199. 
200. void initFunc()
201. {
202.     // Inicializa bandeja
203.     pinMode(bandeja_IN1,OUTPUT);
204.     pinMode(bandeja_IN2,OUTPUT);
205.     pinMode(bandeja_PWM,OUTPUT);
206.     cntlBandeja(OFF);
207. 
208.     // Inicializa velocidade dos steppers
209.     stepperVertical.setSpeed(60);
210.     stepperHorizontal.setSpeed(60);
211. 
212.     // Inicializa motores de lançamento
213.     pinMode(m_sup_PWM, OUTPUT);
214.     pinMode(m_inf_PWM, OUTPUT);
215.     cntlMotores(OFF, 0, 0)
216. 
217.     //Inicializa LED
218.     pinMode(LED_R, OUTPUT);
219.     pinMode(LED_G, OUTPUT);
220.     pinMode(LED_B, OUTPUT);
221.     statusLED(GREEN);
222. 
223.     // Inicializa TCRT
224.     pinMode(TCRT, INPUT);
225. 
226.     // Inicializa variáveis
227.     global_counter = 0;
228.     global_RPM = 0;
229.     global_topspin = 1,
230.     global_underspin = 1,
231.     global_bandeja_RPM = 0;
232.     global_direcao = 0;
233.     global_vertical_pos = 0;
234.     global_start = FALSE;
235. 
236. }
237. 
238. void start(bool input)
239. {
240.     if (global_start && input)
241.     {
242.         cntlBandeja(ON, global_bandeja_RPM);
243.         cntlMotores(ON, global_RPM, global_topspin, global_underspin);
244.     }
245.     else
246.     {
247.         Serial.println("Start falhou");
248.         statusLED(RED);
249.     }
250. }
251. 
252. bool precondic()    // TBD
253. {
254. 
255.     if (!eeprom_read()){
256.         Serial.println("Precondição falhou");
257.         statusLED(RED);
258.         return FALSE;
259.     }
260.     else
261.     {
```

```
262.         return TRUE;
263.     }
264. }
265.
266. void stop(bool input)
267. {
268.     cntlBandeja(OFF, global_bandeja_RPM);
269.     cntlMotores(OFF, global_RPM, global_topspin, global_underspin);
270.
271.     global_start = FALSE;
272.
273.     Serial.println("Start falhou");
274.     statusLED(RED);
275. }
276.
277. void SPIFFS()
278. {
279.     // Inicializa SPIFFS
280.     if (!SPIFFS.begin(true))
281.     {
282.         Serial.println("Erro ao montar o SPIFFS");
283.         statusLED(CYAN);
284.         delay(1000);
285.         global_start = FALSE;
286.         return;
287.     }
288.
289.     // Informações SPIFFS
290.     Serial.println("Informações: ");
291.     Serial.printf("totalBytes: %u\nusedByets: %u\nfreeBytes: %u\n",
292.                   SPIFFS.totalBytes(),
293.                   SPIFFS.usedBytes(),
294.                   SPIFFS.totalBytes() - SPIFFS.usedBytes());
295.
296.     // Listagem de arquivos
297.     Serial.println("\nArquivos --------");
298.
299.     if (!SPIFFS.exists("/launcher.html"))
300.     {
301.         Serial.println("Arquivo '/laucher.html' inexistente");
302.     }
303.
304.     if (!SPIFFS.exists("/style.css"))
305.     {
306.         Serial.println("Arquivo '/style.css' inexistente");
307.     }
308.
309.     File dir = SPIFFS.open("/");
310.     File file = dir.openNextFile();
311.     while (file)
312.     {
313.         Serial.printf(" %s - %u bytes\n", file.name(), file.size());
314.         file = dir.openNextFile();
315.     }
316. }
317.
318. void eeprom_read()
319. {
320.     if (!SPIFFS.exists("/eeprom.txt"))
321.     {
322.         Serial.println("Arquivo '/eeprom.txt' inexistente");
323.     }
324.     else
325.     {
326.         File file = SPIFFS.open("/eeprom.txt", "r");
327.         fprintf(file, global_dir);
```

```
328.         fclose(file);
329.     }
330.     set_global_vertical_pos(inputMessage[0].toInt());
331.     set_global_direcao(inputMessage[1].toInt());
332.     Serial.println("EEPROM lida com sucesso.");
333. }
334. 
335. void eeprom_write()
336. {
337.     if (!SPIFFS.exists("/eeprom.txt"))
338.     {
339.         Serial.println("Arquivo '/eeprom.txt' inexistente");
340.     }
341.     else
342.     {
343.         File file = SPIFFS.open("/eeprom.txt", "w");
344.         global_direcao = 0;
345.         global_vertical_pos = 0;
346.         fprintf(file, global_direcao + global_vertical_pos);
347.         fclose(file);
348.         Serial.println("EEPROM gravada com sucesso.");
349.     }
350. 
351. }
352. 
353. void WiFiAP()
354. {
355.     // Conectar:
356.     // 1. Conecte ao WIFI "Launcher"
357.     // 2. Acesse o IP http://192.168.1.1/
358.     // password válida precisa ter ao menos 7 char
359.     WiFi.softAP(ssid, password);
360.     delay(500);
361.     WiFi.softAPConfig(IP, gateway, subnet);
362.     IPAddress IP = WiFi.softAPIP();
363.     Serial.print("IP do ponto de acesso: ");
364.     Serial.println(IP);
365. 
366.     // Carregar a página
367.     server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
368.     {
369.         request->send(SPIFFS, "/launcher.html", String(), false);
370.     });
371. 
372.     // Carregar styling
373.     server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request)
374.     {
375.         request->send(SPIFFS, "/style.css", "text/css");
376.     });
377. 
378.     // Communicação com HTML
379.     server.on("/get", HTTP_GET, [](AsyncWebServerRequest *request)
380.     {
381.         String inputMessage;
382.         String inputParam;
383.         // Get myConfig1 <ESP_IP>/get?myConfig1=<XXXXXXX>
384.         if (request->hasParam(PARAM_INPUT_1))
385.         {
386.             inputMessage = request->getParam(PARAM_INPUT_1)->value();
387.             inputParam = PARAM_INPUT_1;
388. 
389.             String freq = inputMessage.substring(0, 1);
390.             String spin = inputMessage.substring(1, 2);
391.             String pot = inputMessage.substring(2, 4);
392.             String dir = inputMessage.substring(4, 6);
393. 
```

```
394.             set_global_bandeja_RPM(freq.toInt());
395.             set_global_spin(spin.toInt())
396.             set_global_RPM(pot.toInt());
397.             set_global_direcao(dir.toInt());
398.
399.             global_start = TRUE;
400.
401.             /* debug
402.             Serial.print("Freq: ");
403.             Serial.println(freq);
404.             Serial.println(freq.toInt());
405.             Serial.print("Spin: ");
406.             Serial.println(spin);
407.             Serial.println(spin.toInt());
408.             Serial.print("Potência: ");
409.             Serial.println(pot);
410.             Serial.println(pot.toInt());
411.             Serial.print("Direção: ");
412.             Serial.println(dir);
413.             Serial.println(dir.toInt());
414.             */
415.         }
416.         // Get stop <ESP_IP>/get?stop=0
417.         else if (request->hasParam(PARAM_INPUT_2))
418.         {
419.             inputMessage = request->getParam(PARAM_INPUT_2)->value();
420.             inputParam = PARAM_INPUT_2;
421.
422.             global_start = FALSE;
423.         }
424.         // Get off <ESP_IP>/get?off=0
425.         else if (request->hasParam(PARAM_INPUT_3))
426.         {
427.             inputMessage = request->getParam(PARAM_INPUT_3)->value();
428.             inputParam = PARAM_INPUT_3;
429.
430.             global_start = FALSE;
431.         }
432.         else
433.         {
434.             inputMessage = "No message sent";
435.             inputParam = "none";
436.         }
437.
438.         // Retorna à pagina inciial
439.         request->send(SPIFFS, "/launcher.html", String(), false, processor);
440.     });
441.
442.     // Inicia o servidor
443.     server.begin();
444. }
445.
446. void cntlBandeja(int status, int bandeja_RPM)
447. {
448.     switch(status){
449.         case ON:
450.             // Ativa rotação bandeja
451.             analogWrite(bandeja_IN1, HIGH);
452.             analogWrite(bandeja_IN2, LOW);
453.             analogWrite(bandeja_PWM, bandeja_RPM);
454.             statusLED(BLUE);
455.             break;
456.
457.         case OFF:
458.             // Desativa rotação bandeja
459.             analogWrite(bandeija_IN1, LOW);
```

```
460.             analogWrite(bandeija_IN2, LOW);
461.             analogWrite(bandeija_PWM, 0);
462.             statusLED(GREEN);
463.             break;
464.
465.         case REVERSE:
466.             // Inverte o sentido de rotação
467.             analogWrite(bandeja_IN1, LOW);
468.             analogWrite(bandeja_IN2, HIGH);
469.             analogWrite(bandeja_PWM, B_LOW);
470.             statusLED(YELLOW);
471.             delay(2500);
472.             break;
473.
474.         default:
475.             break;
476.     }
477. }
478.
479. void correnteBandeja()
480. {
481.     float tensao;
482.     unsigned int x = 0;
483.     float   valorACS   = 0.0,
484.             amostra    = 0.0,
485.             mediaACS     = 0.0,
486.             valorACS2  = 0.0;
487.
488.     for (int x = 0; x < 10; x++)                // Coletar 10 amostras
489.     {
490.         valorACS = analogRead(ACS);              // Leitura de corrente
491.         amostra = amosta + valorACS;            // Somatório amostras
492.         delay (3);                              // Delay até próx amostra
493.     }
494.     mediaACS=amostra/10.0;                      // Média das amotras
495.     tensao=mediaACS*(5.0 / 1024.0);             //((mediaACS * (5.0 / 1024.0))
    conversão da leitura para 0-5V
496.     valorACS2 = (2.5 - tensao)*1000/0.185;      //2.5 offset,   0.185v is rise in
    output voltage when 1A current flows at input
497.
498.     // Lógica
499.     if (valorACS2 >=  limCorrente){
500.         cntlBandeja(OFF);
501.         statusLED(RED);
502.         delay(100);
503.         cntlBandeja(REVERSE);
504.     }
505.
506.     /* Debug
507.         Serial.print("Raw Voltage:");
508.         Serial.print(tensao);
509.         Serial.print("\t");
510.         Serial.print("Motor Current :");
511.         Serial.print(valorACS2);                    //Print no monitor serial
512.         Serial.println(" mA");
513.     */
514. }
515.
516. void cntlMotores(bool status, int RPM, int topspin, int underspin)
517. {
518.     switch(status)
519.     {
520.         case TRUE:
521.             analogWrite(m_inf_PWM, (RPM * underspin));              // Motor
    inferior
```

```
522.             analogWrite(m_sup_PWM, (RPM * topspin));                // Motor
    superior
523.             statusLED(BLUE);
524.             break;
525.
526.         case FALSE:
527.             analogWrite(m_inf_PWM, 0);                  // Motor inferior
528.             analogWrite(m_sup_PWM, 0);                  // Motor superior
529.             statusLED(GREEN);
530.             break;
531.
532.         default:
533.             break;
534.     }
535.
536. }
537.
538. int contLancamento()
539. {
540.     int valorTCRT = analogRead(TCRT);
541.
542.     if(valorTCRT < limTCRT)
543.     {
544.         global_counter++;
545.     }
546.
547.     /* Debug
548.         Serial.print("Raw Value:");
549.         Serial.print(valorTCRT);
550.         Serial.print("\t");
551.         Serial.print("Contador :");
552.         Serial.print(global_counter);                    //Print no monitor serial
553.     */
554.     return global_counter;
555. }
556.
557. bool cntlLancamento()
558. {
559.     int aux1 = contLancamento();
560.     delay(3000);
561.     int aux2 = contLancamento();
562.
563.     if ( aux2 > aux1 )
564.     {
565.         return TRUE;                // Continua a lançar
566.     }
567.     else
568.     {
569.         return FALSE;               // Parou de lançar
570.     }
571. }
572.
573. void statusLED(LED status)
574. {
575.     switch(status)
576.     {
577.         case RED:
578.             digitalWrite(LED_G, LOW);
579.             digitalWrite(LED_B, LOW);
580.             digitalWrite(LED_R, HIGH);
581.             break;
582.
583.         case GREEN:
584.             digitalWrite(LED_G, HIGH);
585.             digitalWrite(LED_B, LOW);
586.             digitalWrite(LED_R, LOW);
```

```
587.             break;
588.
589.         case BLUE:
590.             digitalWrite(LED_G, LOW);
591.             digitalWrite(LED_B, HIGH);
592.             digitalWrite(LED_R, LOW);
593.             break;
594.
595.         case YELLOW:
596.             digitalWrite(LED_G, HIGH);
597.             digitalWrite(LED_B, LOW);
598.             digitalWrite(LED_R, HIGH);
599.             break;
600.
601.         case CYAN:
602.             digitalWrite(LED_G, HIGH);
603.             digitalWrite(LED_B, HIGH);
604.             digitalWrite(LED_R, LOW);
605.             break;
606.
607.         case PURPLE:
608.             digitalWrite(LED_G, LOW);
609.             digitalWrite(LED_B, HIGH);
610.             digitalWrite(LED_R, HIGH);
611.             break;
612.
613.         default:
614.             digitalWrite(LED_G, LOW);
615.             digitalWrite(LED_B, LOW);
616.             digitalWrite(LED_R, LOW);
617.             break;
618.     }
619. }
620.
621. void set_global_RPM(int RPM)
622. {
623.     switch(RPM)
624.     {
625.         case HIGH:
626.             global_RPM = RPM_HI;
627.             break;
628.
629.         case MID:
630.             global_RPM = RPM_MID;
631.             break;
632.
633.         case LOW:
634.             global_RPM = RPM_LOW;
635.             break;
636.
637.         default:
638.             break;
639.     }
640. }
641.
642. void set_global_spin(int spin)
643. {
644.         switch(spin)
645.     {
646.         case T_HIGH:
647.             global_topspin = TOPSPIN_HI;
648.             global_underspin = 1;
649.             cntlVertical(I_UPUP);
650.             break;
651.
652.         case T_MID:
```

```
653.             global_topspin = TOPSPIN_MID;
654.             global_underspin = 1;
655.             cntlVertical(I_UP);
656.             break;
657.
658.         case T_LOW:
659.             global_topspin = TOPSPIN_LOW;
660.             global_underspin = 1;
661.             cntlVertical(I_UP);
662.             break;
663.
664.         case U_HIGH:
665.             global_underspin = UNDERSPIN_HI;
666.             global_topspin = 1;
667.             cntlVertical(I_DOWNDOWN);
668.             break;
669.
670.         case U_MID:
671.             global_underspin = UNDERSPIN_MID;
672.             global_topspin = 1;
673.             cntlVertical(I_DOWN);
674.             break;
675.
676.         case U_LOW:
677.             global_underspin = UNDERSPIN_LOW;
678.             global_topspin = 1;
679.             cntlVertical(I_DOWN);
680.             break;
681.
682.         case ZERO:
683.             global_topspin = TOPSPIN_ZERO;
684.             global_underspin = UNDERSPIN_ZERO;
685.             cntlVertical(I_NONE);
686.             break;
687.
688.         default:
689.             global_topspin = TOPSPIN_ZERO;
690.             global_underspin = UNDERSPIN_ZERO;
691.             cntlVertical(I_NONE);
692.             break;
693.     }
694. }
695.
696. void set_global_bandeja_RPM(int bandeja_RPM)
697. {
698.    switch(bandeja_RPM)
699.     {
700.         case F_HIGH:
701.             global_bandeja_RPM = B_HI;
702.             break;
703.
704.         case F_MID:
705.             global_bandeja_RPM = B_MID;
706.             break;
707.
708.         case F_LOW:
709.             global_bandeja_RPM = B_LOW;
710.             break;
711.
712.         default:
713.             break;
714.     }
715. }
716.
717. void set_global_direcao(int dir)   //redundante
718. {
```

```
719.      global_direcao = dir;
720.      /*switch(dir)                          // Clean up
721.      {
722.          case CENTRO:
723.              global_direcao = CENTRO;
724.              break;
725.
726.          case ESQ:
727.              global_direcao = ESQ;
728.              break;
729.
730.          case DIR:
731.              global_direcao = DIR;
732.              break;
733.
734.          default:
735.              global_direcao = CENTRO;
736.              break;
737.      }*/
738.  }
739.
740.  void set_global_vertical_pos(int pos)
741.  {
742.      global_vertical_pos = pos;
743.      /*switch(pos)                                  // Clean up
744.      {
745.          case I_NONE:
746.              global_vertical_pos = I_NONE;
747.              break;
748.
749.          case I_UPUP:
750.              global_vertical_pos = I_UPUP;
751.              break;
752.
753.          case I_UP:
754.              global_vertical_pos = I_UP;
755.              break;
756.
757.          case I_DOWN:
758.              global_vertical_pos = I_DOWN;
759.              break;
760.
761.          case I_DOWNDOWN:
762.              global_vertical_pos = I_DOWNDOWN;
763.              break;
764.
765.          default:
766.              global_vertical_pos = I_NONE;
767.              break;
768.      }*/
769.  }
770.
771.  void cntlVertical(int inclinacao)
772.  {
773.      switch(inclinacao)
774.          {
775.              case I_NONE:
776.                 if(global_vertical_pos == I_UPUP)
777.                  {
778.                      stepperVertical.step(DOWN);          // Retorna ao UP
779.                      delay(500);
780.                      stepperVertical.step(DOWN);          // Retorna ao 0
781.                      delay(500);
782.                      set_global_vertical_pos(I_NONE);
783.                  }
784.                  else if (global_vertical_pos == I_UP)
```

```
785.                 {
786.                     stepperVertical.step(DOWN);         // Retorna ao 0
787.                     delay(500);
788.                     set_global_vertical_pos(I_NONE);
789.                 }
790.                 else if (global_vertical_pos == I_NONE)
791.                 {
792.                     set_global_vertical_pos(I_NONE);    // Mantém
793.                 }
794.                 else if (global_vertical_pos == I_DOWN)
795.                 {
796.                     stepperVertical.step(UP);           // Retorna ao 0
797.                     delay(500);
798.                     set_global_vertical_pos(I_NONE);
799.                 }
800.                 else if (global_vertical_pos == I_DOWNDOWN)
801.                 {
802.                     stepperVertical.step(UP);           // Retorna ao DOWN
803.                     delay(500);
804.                     stepperVertical.step(UP);           // Retorna ao 0
805.                     delay(500);
806.                     set_global_vertical_pos(I_NONE);
807.                 }
808.                 break;
809.
810.             case I_UPUP:
811.                 if(global_vertical_pos == I_UPUP)
812.                 {
813.                     set_global_vertical_pos(I_UPUP);    // Mantém UPUP
814.                 }
815.                 else if (global_vertical_pos == I_UP)
816.                 {
817.                     stepperVertical.step(UP);           // Avança ao UPUP
818.                     delay(500);
819.                     set_global_vertical_pos(I_UPUP);
820.                 }
821.                 else if (global_vertical_pos == I_NONE)
822.                 {
823.                     stepperVertical.step(UP);           // Avança ao UP
824.                     delay(500);
825.                     stepperVertical.step(UP);           // Avança ao UPUP
826.                     delay(500);
827.                     set_global_vertical_pos(I_UPUP);
828.                 }
829.                 else if (global_vertical_pos == I_DOWN)
830.                 {
831.                     stepperVertical.step(UP);           // Retorna ao 0
832.                     delay(500);
833.                     stepperVertical.step(UP);           // Avança ao UP
834.                     delay(500);
835.                     stepperVertical.step(UP);           // Avança ao UPUP
836.                     delay(500);
837.                     set_global_vertical_pos(I_UPUP);
838.                 }
839.                 else if (global_vertical_pos == I_DOWNDOWN)
840.                 {
841.                     stepperVertical.step(UP);           // Retorna ao DOWN
842.                     delay(500);
843.                     stepperVertical.step(UP);           // Retorna ao 0
844.                     delay(500);
845.                     stepperVertical.step(UP);           // Avança ao UP
846.                     delay(500);
847.                     stepperVertical.step(UP);           // Avança ao UPUP
848.                     delay(500);
849.                     set_global_vertical_pos(I_UPUP);
850.                 }
```

```
851.                 break;
852.
853.             case I_UP:
854.                 if(global_vertical_pos == I_UPUP)
855.                 {
856.                     stepperVertical.step(DOWN);         // Retorna ao UP
857.                     delay(500);
858.                     set_global_vertical_pos(I_UP);
859.                 }
860.                 else if (global_vertical_pos == I_UP)
861.                 {
862.                     set_global_vertical_pos(I_UP);      // Mantém
863.                 }
864.                 else if (global_vertical_pos == I_NONE)
865.                 {
866.                     stepperVertical.step(UP);           // Avança ao UP
867.                     delay(500);
868.                     set_global_vertical_pos(I_UP);
869.                 }
870.                 else if (global_vertical_pos == I_DOWN)
871.                 {
872.                     stepperVertical.step(UP);           // Retorna ao 0
873.                     delay(500);
874.                     stepperVertical.step(UP);           // Avança ao UP
875.                     delay(500);
876.                     set_global_vertical_pos(I_UP);
877.                 }
878.                 else if (global_vertical_pos == I_DOWNDOWN)
879.                 {
880.                     stepperVertical.step(UP);           // Retorna ao DOWN
881.                     delay(500);
882.                     stepperVertical.step(UP);           // Retorna ao 0
883.                     delay(500);
884.                     stepperVertical.step(UP);           // Avança ao UP
885.                     delay(500);
886.                     set_global_vertical_pos(I_UP);
887.                 }
888.                 break;
889.
890.             case I_DOWN:
891.                 if(global_vertical_pos == I_UPUP)
892.                 {
893.                     stepperVertical.step(DOWN);         // Retorna ao UP
894.                     delay(500);
895.                     stepperVertical.step(DOWN);         // Retorna ao 0
896.                     delay(500);
897.                     stepperVertical.step(DOWN);         // Avança ao DOWN
898.                     delay(500);
899.                     set_global_vertical_pos(I_DOWN);
900.                 }
901.                 else if (global_vertical_pos == I_UP)
902.                 {
903.                     stepperVertical.step(DOWN);         // Retorna ao 0
904.                     delay(500);
905.                     stepperVertical.step(DOWN);         // Avança ao DOWN
906.                     delay(500);
907.                     set_global_vertical_pos(I_DOWN);
908.                 }
909.                 else if (global_vertical_pos == I_NONE)
910.                 {
911.                     stepperVertical.step(DOWN);         // Avança ao DOWN
912.                     delay(500);
913.                     set_global_vertical_pos(I_DOWN);
914.                 }
915.                 else if (global_vertical_pos == I_DOWN)
916.                 {
```

```
917.                     set_global_vertical_pos(I_DOWN);      // Mantém
918.                 }
919.                 else if (global_vertical_pos == I_DOWNDOWN)
920.                 {
921.                     stepperVertical.step(UP);           // Retorna ao DOWN
922.                     delay(500);
923.                     set_global_vertical_pos(I_DOWN);
924.                 }
925.                 break;
926. 
927.             case I_DOWNDOWN:
928.                 if(global_vertical_pos == I_UPUP)
929.                 {
930.                     stepperVertical.step(DOWN);         // Retorna ao UP
931.                     delay(500);
932.                     stepperVertical.step(DOWN);         // Retorna ao 0
933.                     delay(500);
934.                     stepperVertical.step(DOWN);         // Avança ao DOWN
935.                     delay(500);
936.                     stepperVertical.step(DOWN);         // Avança ao DOWNDOWN
937.                     delay(500);
938.                     set_global_vertical_pos(I_DOWNDOWN);
939.                 }
940.                 else if (global_vertical_pos == I_UP)
941.                 {
942.                     stepperVertical.step(DOWN);         // Retorna ao 0
943.                     delay(500);
944.                     stepperVertical.step(DOWN);         // Avança ao DOWN
945.                     delay(500);
946.                     stepperVertical.step(DOWN);         // Avança ao DOWNDOWN
947.                     delay(500);
948.                     set_global_vertical_pos(I_DOWNDOWN);
949.                 }
950.                 else if (global_vertical_pos == I_NONE)
951.                 {
952.                     stepperVertical.step(DOWN);         // Avança ao DOWN
953.                     delay(500);
954.                     stepperVertical.step(DOWN);         // Avança ao DOWNDOWN
955.                     delay(500);
956.                     set_global_vertical_pos(I_DOWNDOWN);
957.                 }
958.                 else if (global_vertical_pos == I_DOWN)
959.                 {
960.                     stepperVertical.step(DOWN);         // Avança ao DOWNDOWN
961.                     delay(500);
962.                     set_global_vertical_pos(I_DOWNDOWN);
963.                 }
964.                 else if (global_vertical_pos == I_DOWNDOWN)
965.                 {
966.                     set_global_vertical_pos(I_DOWNDOWN);// Mantém
967.                 }
968.                 break;
969. 
970.             default:
971.                 break;
972.         }
973. }
974. 
975. void cntlHorizontal(int direcao)
976. {
977.     switch(direcao)
978.     {
979.         case DIR:
980.             if(global_direcao == CENTRO)
981.             {
982.                 stepperHorizontal.step(HORARIO);  // DIREITA
```

```
983.                 set_global_direcao(DIR);
984.             }
985.             else if (global_direcao == ESQ)
986.             {
987.                 stepperHorizontal.step(HORARIO);  // Retorna ao centro
988.                 delay(500);
989.                 stepperHorizontal.step(HORARIO);  // Posição DIREITA
990.                 set_global_direcao(DIR);
991.             }
992.             else if (global_direcao == DIR)
993.             {
994.                 set_global_direcao(DIR);
995.             }
996.             break;
997.
998.         case ESQ:
999.             if(global_direcao == CENTRO)
1000.            {
1001.                stepperHorizontal.step(ANTIHORARIO);  // ESQ
1002.                set_global_direcao(ESQ);
1003.            }
1004.            else if (global_direcao == DIR)
1005.            {
1006.                stepperHorizontal.step(ANTIHORARIO);  // Retorna ao centro
1007.                delay(500);
1008.                stepperHorizontal.step(ANTIHORARIO);  // Posição ESQUERDA
1009.                set_global_direcao(ESQ);
1010.            }
1011.            else if (global_direcao == ESQ)
1012.            {
1013.                set_global_direcao(ESQ);
1014.            }
1015.            break;
1016.
1017.        case CENTRO:
1018.            if(global_direcao == CENTRO)
1019.            {
1020.                set_global_direcao(CENTRO);
1021.            }
1022.            else if (global_direcao == ESQ)
1023.            {
1024.                stepperHorizontal.step(HORARIO);  // Retorna ao centro
1025.                set_global_direcao(CENTRO);
1026.            }
1027.            else if (global_direcao == DIR)
1028.            {
1029.                stepperHorizontal.step(ANTIHORARIO);  // Retorna ao centro
1030.                set_global_direcao(CENTRO);
1031.            }
1032.            break;
1033.
1034.        default:
1035.            if(global_direcao == DIR)
1036.            {
1037.                stepperHorizontal.step(ANTIHORARIO);
1038.            }
1039.            else if(global_direcao == ESQ)
1040.            {
1041.                stepperVertical.step(HORARIO);
1042.            }
1043.            break;
1044.    }
1045. }
```

APÊNDICE D – LÓGICA DE PROGRAMAÇÃO HMTL/CSS

HTML

```
<!DOCTYPE html>
<html lang="pt">
<head>
    <title>Ball Launcher UI</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="style.css">

</head>
<body>

    <div class="header">
        <a href="#default" class="logo">Ball Launcher X2000</a>
    </div>

    <h2>Selecione as opções de treino:</h2>

    <p>É necessário selecionar uma opção para todos os campos.</p>

    <label for="freq">Frequência de lançamento:</label>
    <select name="freq" id="freq">
        <option value="NULL"> </option>
        <option value="0">Alta</option>
        <option value="1">Média</option>
        <option value="2">Baixa</option>
    </select>

    <br>
    <br>


    <label for="topspin">Tipo e velocidade de spin:</label>
    <select name="topspin" id="topspin">
        <option value="NULL"> </option>
        <optgroup label="Topspin">
            <option value="3">Alta</option>
            <option value="4">Média</option>
            <option value="5">Baixa</option>
            <option value="6">Zero</option>
        </optgroup>
        <optgroup label="Underspin">
            <option value="7">Alta</option>
            <option value="8">Média</option>
            <option value="9">Baixa</option>
            <option value="6">Zero</option>
          </optgroup>
    </select>

    <br>
    <br>

    <label for="potencia">Força do lançamento:</label>
    <select name="potencia" id="potencia">
        <option value="NULL"> </option>
        <option value="11">Alta</option>
        <option value="12">Média</option>
        <option value="13">Baixa</option>
    </select>

    <br>
```

```
61.     <br>
62.
63.     <label for="direcao">Direção de lançamento:</label>
64.     <select name="direcao" id="direcao">
65.         <option value="NULL"> </option>
66.         <option value="14">Esquerda</option>
67.         <option value="15">Direita</option>
68.         <option value="16">Centro</option>
69.     </select>
70.
71.     <br>
72.     <br>
73.     <br>
74.
75.     <a href="#" id="myConfig1" name="myConfig_1"><button onclick="Iniciar()"
    class="button">Iniciar</button></a>
76.
77.     <br>
78.     <br>
79.
80.     <a href="#" id="stop" name="Stop"><button onclick="Parar()"
    class="button">Parar</button></a>
81.
82.     <br>
83.     <br>
84.
85.     <a href="#" id="off" name="Off"><button onclick="Desligar()"
    class="button">Desligar</button></a>
86.
87.     <br>
88.     <br>
89.
90.
91.     <script>
92.         function Iniciar() {
93.             let msg_freq = document.getElementById("freq").value;
94.
95.             let msg_topspin = document.getElementById("topspin").value;
96.
97.             let msg_potencia = document.getElementById("potencia").value;
98.
99.             let msg_direcao = document.getElementById("direcao").value;
100.
101.             var msg_esp32 = "get?" + "myConfig1=" + msg_freq + msg_topspin +
    msg_potencia + msg_direcao;
102.
103.             alert(msg_esp32);
104.
105.             location.href = msg_esp32;
106.
107.         }
108.     </script>
109.
110.     <script>
111.         function Parar() {
112.             var msg_esp32 = "get?" + "stop=0";
113.
114.             alert(msg_esp32);
115.
116.             location.href = msg_esp32;
117.         }
118.     </script>
119.
120.     <script>
121.         function Desligar() {
122.             var msg_esp32 = "get?" + "off=0";
```

```
123.
124.             alert(msg_esp32);
125.
126.             location.href = msg_esp32;
127.         }
128.     </script>
129. </body>
130. </html>
```

CSS

```
1.          body {
2.              text-align: center;
3.          }
4.          h1 {
5.              display: inline-block;
6.              background-color: rgb(50, 113, 150);
7.          }
8.          h2 {
9.              font-size: 20px;
10.             color: rgb(0, 0, 0);
11.         }
12.         p {
13.             font-size: 12px;
14.             color: rgb(163, 31, 31);
15.         }
16.
17.         label {
18.             width:200px;
19.             display: inline-block;
20.         }
21.
22.         select {
23.             width:80px;
24.         }
25.
26.         /* Style the header with a grey background and some padding */
27.         .header {
28.             overflow: hidden;
29.             background-color: #f1f1f1;
30.             padding: 20px 10px;
31.         }
32.
33.         /* Change the background color on mouse-over */
34.         .header a:hover {
35.             background-color: #ddd;
36.             color: black;
37.         }
38.
39.         /* Style the header links */
40.         .header a {
41.             float: center;
42.             color: black;
43.             text-align: center;
44.             padding: 12px;
45.             text-decoration: none;
46.             font-size: 18px;
47.             line-height: 25px;
48.             border-radius: 4px;
49.         }
50.
51.         /* Style the logo link (notice that we set the same value of line-height and
    font-size to prevent the header to increase when the font gets bigger */
52.         .header a.logo {
53.             font-size: 25px;
```

```
            font-weight: bold;
        }

        /* Change the background color on mouse-over */
        .header a:hover {
            background-color: #ddd;
            color: black;
        }

        /* Loader style */
        .loader {
            border: 2px solid #f3f3f3;
            border-radius: 50%;
            border-top: 2px solid #3498db;
            width: 16px;
            height: 16px;
            -webkit-animation: spin 2s linear infinite; /* Safari */
            animation: spin 2s linear infinite;
          }

          /* Safari */
        @-webkit-keyframes spin {
            0% { -webkit-transform: rotate(0deg); }
            100% { -webkit-transform: rotate(360deg); }
        }

        @keyframes spin {
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
        }

        .flex-container {
            display: flex;
            align-items: center;
            justify-content: center;
        }

        .flex-child {
            flex: 1;
        }

        .flex-child:first-child {
            margin-right: 8px;
        }
```