Universidade de Aveiro
2023

Luís Paulo Mendes Dias

Monitorização e controlo de uma máquina de lançamento de bolas de futebol

Monitoring and control of a football ball launching machine

Universidade de Aveiro
2023

Luís Paulo Mendes Dias

Monitorização e controlo de uma máquina de lançamento de bolas de futebol

Projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Automação Industrial, realizada sob a orientação científica de Jorge Augusto Fernandes Ferreira, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro, e de António Manuel de Monteiro Ramos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Este projeto teve o apoio dos projetos UIDB/00481/2020 e UIDP/00481/2020 - Fundação para a Ciência e a Tecnologia; e CENTRO-01-0145 FEDER-022083 - Programa Operacional Regional do Centro (Centro2020), através do Portugal 2020 e do Fundo Europeu de Desenvolvimento Regional.

Dedico este trabalho à minha namorada e família, pelo constante apoio e força que me concederam nesta jornada.

o júri

presidente

Prof. Doutor Rui Manuel Escadas Ramos Martins

professor auxiliar da Universidade de Aveiro

vogais

Prof. Doutor José António Barros Vieira

professor adjunto do Instituto Politécnico de Castelo Branco

Prof. Doutor Jorge Augusto Fernandes Ferreira

professor associado da Universidade de Aveiro

agradecimentos

Deixo agradecimentos aos meus orientadores, Professores Doutor Jorge Ferreira e António Ramos pela orientação, pela inteira disponibilidade, dedicação, incentivo e preocupação manifestada durante a realização do projeto. Agradeço a todos os participantes que colaboraram na realização deste projeto.

Aos meus pais que sempre me ajudaram ao longo da vida e garantiram a melhor educação, preocuparam-se com o meu bem-estar e crescimento pessoal, sempre me apoiaram e incentivaram a alcançar os meus objetivos.

À minha namorada que esteve ao meu lado nos tempos mais difíceis da minha vida, e sempre me apoiou e levantou para o caminho que devo traçar.

Por fim, a todas as pessoas que não foram aqui mencionadas, mas que me ajudaram dia-a-dia nesta gloriosa jornada universitária e que marcaram esta etapa da minha vida, a todos muito obrigado.

palavras-chave

Lançador de bolas, tecnologia, software, eletrónica, aplicação web, automação, controlo, responsividade.

resumo

O presente projeto foi elaborado no âmbito do Projeto Final do Mestrado em Engenharia de Automação Industrial, com o objetivo de dar continuidade ao desenvolvimento do trabalho apresentado na Dissertação intitulada "Desenvolvimento de um sistema de monitorização e controlo para máquina de lançamento de bolas de futebol" realizado por Diogo Filipe de Almeida Nunes em 2020. O projeto anterior deixou uma base de trabalho que requer melhorias, montagem final, testes funcionais e desenvolvimento de software para dispositivos móveis, visando o controlo e monitorização da máquina de lançamento de bolas.

Com este projeto foi possível melhorar o sistema previamente implementado, bem como desenvolver uma aplicação web responsiva e simples de utilizar, que é capaz de se adaptar a diferentes dispositivos. Para alcançar este objetivo, foram integradas tecnologias avançadas de automação e controlo, refeita a arquitetura de hardware e software, tornando assim o lançador de bolas uma máquina capaz de executar treinos e lances em modo automático, alem de suportar uma base de software altamente escalável que pode integrar sistemas de visão computacional e inteligência artificial. Com este projeto é possível proporcionar aos treinadores uma janela de oportunidade para aprimorar os seus treinos.

keywords

Ball launcher, technology, software, electronics, web aplication, automation, control, responsiveness.

abstract

The present project has been developed within the scope of the Final Project for the Master's Degree in Industrial Automation Engineering, with the objective of continuing the development of the work presented in the Dissertation entitled "Development of a Monitoring and Control System for a Football Ball Launching Machine" carried out by Diogo Filipe de Almeida Nunes in 2020. The previous project has laid a foundation that requires improvements, final assembly, functional testing, and development of software for mobile devices, aiming to control and monitor the ball launching machine.

With this project, it has been possible to enhance the previously implemented system and develop a responsive and user-friendly web application that can adapt to different devices. To achieve this objective, advanced automation and control technologies have been integrated, the hardware and software architecture have been redesigned, thus making the ball launcher a machine capable of performing training sessions and automatic throws. Additionally, it supports a highly scalable software foundation that can integrate computer vision systems and artificial intelligence. This project provides coaches with an opportunity to enhance their training sessions.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AI- Artificial Intelligence

IoT - Internet of Things

RAM - Random Access Memory

VAR – Vídeo árbitro

GRBL - G-code Real-time Boot Loader

AJAX - Asynchronous JavaScript and XML

XML - eXtensible Markup Language

HTTP - Hypertext Transfer Protocol

HTML - Hypertext Markup Language

CSS - Cascading Style Sheets

CNC - Controlo Numérico Computacional

HTTPS - Hypertext Transfer Protocol Secure

SSL - Secure Sockets Layer

TLS - Transport Layer Security

WSS - WebSocket Secure

SBC – Single board computer

MTV (Model-Template-View)

DOM - Document Object Model

JSON - JavaScript Object Notation

PWM – Pulse Width Modulation

Capítulo 1

1. Introdução

O projeto exposto neste documento foi desenvolvido no âmbito do Mestrado em Engenharia de Automação Industrial pela Universidade de Aveiro, com o objetivo de obter o grau de mestre. O foco deste trabalho é a automação de um lançador de bolas, um projeto que teve início na tese de Mestrado de Diogo Filipe de Almeida Nunes. Pretende-se que seja uma solução inovadora onde serão exploradas diversas áreas da Engenharia de Automação Industrial, desde a arquitetura de hardware e software até a integração de tecnologias avançadas.

1.1 Enquadramento

A humanidade sempre procura disputas, é algo que adquirimos ao longo da nossa evolução, nos tempos modernos estas disputas podem ser intelectuais, tecnológicas, territoriais, desportivas entre outras. O Futebol é uma disputa saudável que ao longo dos tempos ganhou adesão pela sociedade e tornou-se cada vez mais competitiva, hoje é considerado por muitos o desporto rei, palco de grandes confrontos entre países e clubes. Este nível de competitividade leva ao surgimento de novas tecnologias para aumentar a performance dos jogadores, como por exemplo a monitorização do desempenho, auxílio no treino, aquisição de dados dos jogadores para posterior análise, etc.

Os lançadores de bolas são uma mais-valia no contexto dos treinos de guarda-redes, aceleram a evolução e proporcionam um treino com ritmo elevado. No entanto, os lançadores de bolas presentes no mercado estão aquém das expectativas tecnológicas atuais, não são programáveis e limitam-se simplesmente a lançar bolas, sem oferecer suporte tecnológico que proporcione mais qualidade aos treinos.

O lançador de bolas desenvolvido neste projeto tem como objetivo superar estas limitações, incorporando um conjunto de tecnologias que permitem ir além do convencional.

Neste documento será apresentado o desenvolvimento e integração eletrónica do lançador de bolas, e de uma aplicação para controlo e monitorização.

1.2 Motivação e objetivos

O desenvolvimento de uma máquina é uma tarefa complexa que requer um conhecimento sólido em várias áreas. Nesse sentido, o projeto foi abraçado com a finalidade de trabalhar em algo real, que permitisse a aquisição e consolidação de conhecimentos, bem como explorar oportunidades tecnológicas que permitissem aplicar soluções inovadoras ao lançador de bolas.

O objetivo principal deste projeto consiste em dotar o lançador de bolas num sistema de controlo e gestão treinos avançados, preenchendo uma lacuna existente no mercado, aprimorando tanto a arquitetura de hardware quanto a de software.

Outro objetivo importante é o desenvolvimento de uma aplicação web responsiva e intuitiva, que permitirá o controlo do lançador de bolas por meio de diferentes dispositivos.

Para atingir estes objetivos e ao mesmo tempo testar as funcionalidades, foi necessário a criação de uma maquete de testes para validar o correto funcionamento das funções desenvolvidas, permitindo a correção de possíveis falhas no sistema. Além disso, pretende-se criar um menu de configuração rápido, que possibilita o comissionamento do lançador de bolas sem a necessidade de compilar o código fonte.

A possibilidade de contribuir para o avanço da engenharia de automação em máquinas de lançamento de bolas, promovendo o uso de sistemas inteligentes e automatizados no campo do desporto, foi a maior motivação que levou ao empenho constante durante o desenvolvimento deste projeto. Todo o projeto foi pensado para fornecer aos treinadores e jogadores uma ferramenta poderosa para auxilio no treino de guardas-redes.

1.3 Organização

O presente trabalho está dividido em 7 capítulos. No capítulo atual é apresentada uma breve introdução ao trabalho realizado neste projeto, bem como o enquadramento, motivações e objetivos que levaram a escolher este projeto.

O segundo capítulo aborda o estado da arte e da tecnologia dos lançadores de bolas, apresentando uma visão histórica desde as primeiras versões registadas, até as versões comerciais mais recentes.

O terceiro capítulo descreve a solução proposta para o projeto, incluindo uma lista das tecnologias que serão implementadas para alcançar os objetivos estabelecidos.

O quarto capítulo contextualiza as tecnologias que são implementadas no projeto, apresentando uma descrição e explicação de cada uma delas.

O quinto capítulo aborda a arquitetura de hardware do lançador de bolas, destacando a lógica e os critérios utilizados na escolha dos componentes utilizados.

O sexto capítulo concentra-se na arquitetura de software do projeto, descrevendo cada componente da interface gráfica, a sua interação com o utilizador e os diferentes dispositivos.

No sétimo e último capítulo são apresentadas as conclusões obtidas a partir do trabalho realizado, incluindo os principais resultados e contribuições, além de sugerir possíveis trabalhos futuros que possam vir a ser desenvolvidos.

Capítulo 2

2. Revisão do estado da arte e da tecnologia

2.1 Desporto e a sociedade

Certamente a tecnologia nunca ira substituir o homem, mas é um facto que o homem já não vive sem ela, diariamente ajuda-nos a resolver vários problemas e auxilia-nos em várias tarefas que sem a tecnologia seria difícil ou até quase impossível de as realizar.

A tecnologia é de facto um trunfo importante adquirido na sociedade, mas a intervenção humana será sempre necessária.

No futebol, a tecnologia está cada vez mais presente, auxiliando nos treinos, nas decisões dos treinadores, na contratação de jogadores, e até mesmo na aplicação precisa das regras durante uma partida. No entanto, a interpretação humana continua sendo essencial, e o objetivo principal, é aprender a usar a tecnologia para minimizar risco de erro e auxiliar-nos em algumas tarefas.

2.2 Tecnologia no futebol

Futebol, apelidado de “Desporto-Rei”, é um desporto que movimenta milhões de Euros o que leva ao aumento de investimento em tecnologia para melhorar tanto a performance dos jogadores como a correta aplicação das regras de jogo.

Através da implementação de tecnologias cada vez mais avançadas, é possível aprimorar a experiência dos adeptos e entusiastas de futebol, garantindo jogos mais competitivos e reduzindo a ocorrência de decisões errôneas na aplicação das regras. A tecnologia aplicada ao contexto futebolístico abrange uma ampla gama de elementos, como por exemplo sistemas para registar e monitorizar o desempenho dos jogadores, sistemas para monitorizar treinos e tecnologias de apoio aos árbitros, como por exemplo o vídeo arbitro (VAR).

As tecnologias de assistência aos árbitros desempenham um papel crucial no jogo, permitindo compreender e expor a importância que a tecnologia proporciona ao espetáculo que é o futebol.

2.3 Tecnologia de sistemas lança bolas atuais

Na generalidade quando se pensa em lança bolas vem sempre à mente um sistema complexo, com motores e eixos precisos equipados com uma certa automação, mas, no entanto, pode ser um pequeno engenho pala lançar bolas como o que mostra a Figura 1. Não é certo quando surgiram os primeiros sistemas de lança bolas tecnologicamente avançados, mas com o passar dos anos, à medida que novos desportos que envolvem bolas foram sendo incorporados, a evolução tecnológica impulsionou o surgimento dos primeiros lança bolas tecnologicamente sofisticados.

Figura 1 - Lança bolas manual [3]

O famoso René Lacoste inventou um dos primeiros lançadores de bola que há registo (Figura 2), este sistema mecânico era composto por uma alavanca que, ao ser girada, comprime uma mola, no momento de máxima força, a bola é lançada em alta velocidade.

Figura 2 - René com o seu lançador de bolas de ténis [4]

Nos anos seguintes esta ideia foi integrada em sistemas cada vez mais complexos, como é o caso do lançador de bolas de Robert H. Mclure, que em 1968 contruiu um sistema de propulsão a mola com um motor elétrico, o mecanismo de lançamento consistia em comprimir uma mola até atingir sua máxima energia potencial cinética e, em seguida, libertá-la instantaneamente, resultando no lançamento da bola. Este novo equipamento tem também um armazém de bolas, o que permite lançamentos com intervalos de tempo constantes (Figura 3).

Figura 3 - Lançador de bolas de Robert H. McClure [5]

Com o passar do tempo o conceito ficou cada vez melhor delineado, o mundo do desporto nesta época está a emergir e novos modelos com sistemas de propulsão diferentes foram surgindo. O sistema de propulsão através de ar comprimido foi um deles (Figura 4), este conceito envolve o armazenamento de ar comprimido num reservatório, no qual a energia potencial acumulada é convertida em energia cinética para o lançamento das bolas. A pressão dentro do reservatório, no momento da liberação, determina a velocidade de projeção da bola. Esse modelo de lançamento pode ser incorporado a uma válvula reguladora de pressão na saída, permitindo a alteração da pressão de lançamento num dos lados da bola. Isso resulta numa rotação no centro de massa da bola e alteração de sua trajetória durante o lançamento. Os sistemas pneumáticos são amplamente utilizados em lançadores de bolas de tênis devido à sua leveza e durabilidade, uma vez que não há desgaste mecânico. No entanto, quando aplicados ao futebol, esses sistemas apresentam desafios adicionais devido ao peso da bola e à imprevisibilidade do controlo da rotação.

Figura 4 - Lançador de bolas pneumático [6]

Outro método usado para lançar bolas é o de discos ou rolos rotativos (Figura 5), este mecanismo consiste em colocar a rolar a alta velocidade dois rolos com uma massa significativamente mais alta que a massa da bola. No processo de

lançamento, a bola é forçada a passar pelos dois rolos, onde o espaçamento entre eles é menor que o diâmetro da bola, resultando em um atrito elevado entre a bola e os rolos. Quando a bola entra em contato com os rolos, inicia um período de compressão e aceleração, após ser lançada, a bola inicia um voo impulsionada pela velocidade linear dos rolos.

Com este método de lançamento, é possível variar a velocidades individual dos rolos, resultando em rotação na bola. Esta rotação, por sua vez, influencia a trajetória da bola ao longo do lançamento, permitindo desviar a trajetória da bola ao longo do lançamento. Este método é amplamente usado pelos lançadores de bolas atuais por ser fiável, versátil e de fácil controlo [17].

Figura 5 - Lançador de bolas através de rolos rotativos [7].

2.4 Mercado de lançadores de bolas atual

Atualmente existem inúmeros lançadores de bolas no mercado que vão desde os mais básicos até aos mais profissionais. É possível comparar estes equipamentos por preço, especificações de performance, entre outras. Os lançadores mais profissionais destacam-se por oferecer um treino de maior qualidade, pois são capazes de lançar bolas com mais velocidade, repetibilidade e efeitos.

A Tabela 1 apresenta alguns modelos à venda no mercado, onde são listadas as especificações técnicas dos equipamentos, permitindo comparar a performance entre eles em relação ao preço sugerido. Neste contexto, será incluído o lançador de bolas desenvolvido no âmbito do presente projeto, permitindo uma análise comparativa com os demais modelos disponíveis no mercado.

Tabela 1 - Comparação das especificações técnicas dos lançadores existentes no mercado (2022)[1][2].

	EUROGOAL 600	EUROGOAL 1500	PRO TRAINER FIRST TOUH	THE BALL LAUNCHER	STRIKE ATTACK	TOCA MACHINE	POWAPASS BALL LAUNCHER	Lançador de bolas UA
Imagem								
Preço	3893 €	7500 €	1750 €	3600 €	3700 €	50 € por sessão	6000 €	-
Velocidade de lançamento	110km/h	140 km/h	100 km/h	130 km /h	120km/h	80km/h	140 km/h	120 km/h
Capacidade máxima	1 bola	1 bola	5 bolas	1 bola	1 bola	15+ bolas	5 bolas	5 + bolas
Alcance máximo	60 m	100 m	38 m	50m	64 m	64 m	100 m	64 m
Regulação vertical	Sim	Sim	Sim (0,90º)	Sim	Sim	Sim(0/45º)	Sim	Sim
Regulação horizontal	Sim	Sim	Não	Sim	Não	Sim	Sim	Sim
Cadencia	60 bolas/min	60 bolas/min	12 bolas/min	30 bolas/min	60 bolas/min	30 bolas/min	60 bolas/min	30 bolas/min
Autonomia	5 horas	8h	2-3 h	4 h	4 h	5 h	8 h	2 h
Controlo de velocidade	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Ajuste automatico	Não	Sim (5 modos)	Sim	Não	Não	Sim	Sim	Sim
Controlo Remoto	Não	Não	Sim	Não	Não	Sim Bluetooth	Não	Sim
Peso	25 kg	30 kg	31 kg	42 kg	70 kg	30 kg	100 kg	~ 50 kg
Efeito na Bola	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim

O lançador de bolas apresentado neste projeto equipara-se aos restantes modelos do mercado em termos de funcionalidades. Possui a capacidade de lançar bolas com três graus de liberdade e controlar individualmente a velocidade dos rolos, permitindo criar lances com efeitos personalizados, possui tambem um armazém com capacidade para mais de cinco bolas que reduz a necessidade de reposição a cada lançamento. A sua estrutura em alumínio torna-o leve e a potência dos rolos e graus de liberdade permitem lançar bolas precisas até aproximadamente 60 metros de distância. As características chave que o diferencia dos demais são as tecnológicas que foram implementadas, como monitorização, automação dos lançamentos, treinos personalizados e a aplicação responsiva que se adapta aos diferentes dispositivos.

Capítulo 3

3. Proposta de solução e conceito

Antes de prosseguirmos, é importante mencionar que este projeto baseia-se no desenvolvimento inicial realizado por Diogo Nunes em sua dissertação de Mestrado. Com grande mérito, ele concebeu o projeto e o transformou numa máquina real, focando-se principalmente no desenvolvimento mecânico. A versão final deste projeto pode ser visualizada na Figura 6.

Figura 6 - Renderização da solução final do lançador de bolas apresentada por Diogo Nunes [1].

O presente capítulo tem como objetivo apresentar o trabalho desenvolvido no âmbito deste projeto listando as tecnologias que vão ser usadas e uma abordagem à arquitetura de hardware e software (Figura 7).

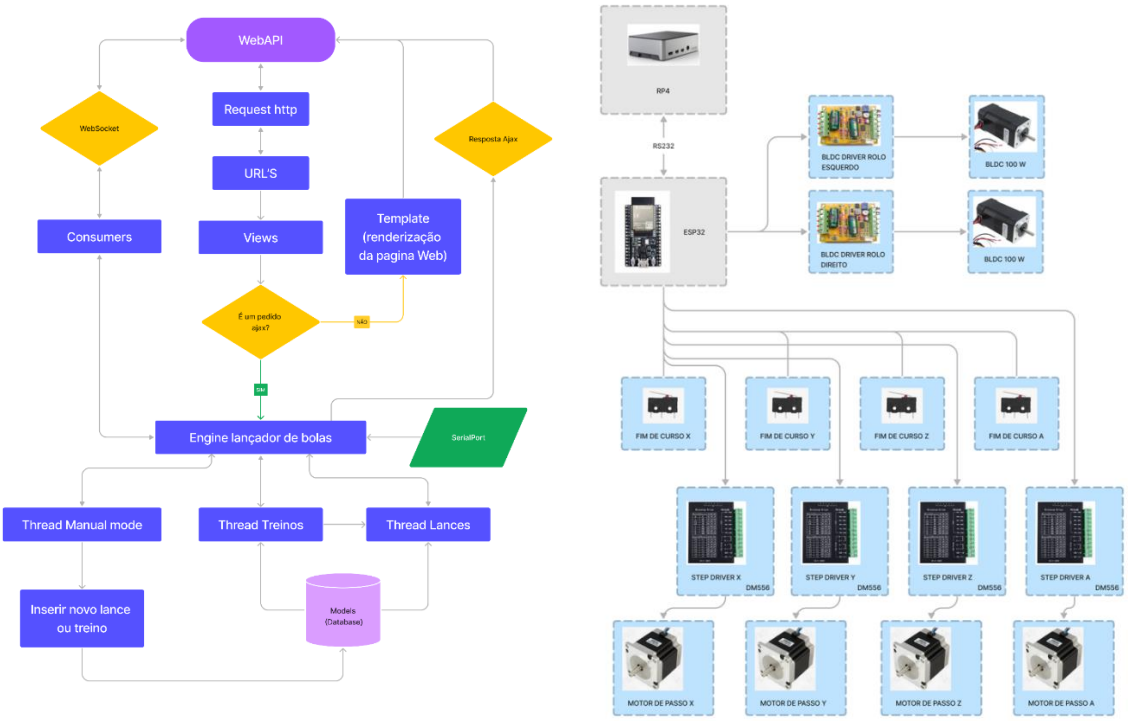


Figura 7 – Visão geral da arquitetura de software e hardware

Na arquitetura de *hardware* podemos observar que o raspberry pi 4 ocupa o topo do diagrama, é ele o centro de todo o controlo, comunica com o ESP32 por RS232 e lança um servidor *web* para controlo e monitorização do lançador de bolas. A arquitetura de software que comanda o fluxo de informação baseia-se na estrutura de comunicação do Django, na qual, foi adicionado um modulo denominado “Engine lançador de bolas”, e threads de controlo. Estas arquiteturas vão ser apresentadas detalhadamente no capítulo 5 e 6.

Para atender aos requisitos deste projeto foi proposta uma solução de software que não apenas se alinhasse com o lançador de bolas projetado, mas também que se adaptasse a lançadores de bolas com diferentes características. Para atingir esse objetivo, o software foi concebido de forma totalmente configurável, eliminando a necessidade de compilar código.

Para suportar o desenvolvimento do projeto com estas especificações, houve um estudo inicial que deu origem a um “ecossistema” de tecnologias e linguagens de programação, que vão ser a base de todo projeto e tornaram o desenvolvimento mais rápido e completo. Estas tecnologias e linguagens propostas são:

- Django: *Framework web* para *backend*, além de ferramentas para lançar um servidor *web*, tem uma estrutura base que suporta a escalabilidade da aplicação;
- AJAX e *WebSocket*: permitem a comunicação assíncrona e em tempo real entre o servidor e o cliente;
- GRBL (*G-code Real-time Boot Loader*): utilizado para o controlo dos motores de passo e o posicionamento preciso da máquina;
- HTML (*Hypertext Markup Language*): linguagem de marcação utilizada para estruturar e exibir o conteúdo do *frontend*;
- CSS (*Cascading Style Sheets*); utilizado para definir a aparência e o estilo visual do *frontend*, incluindo cores, fontes, *layouts* e animações;
- JSON (*JavaScript Object Notation*): formato de dados de fácil leitura utilizado para a troca de informações entre o servidor e o cliente;
- jQuery: biblioteca JavaScript que simplifica a manipulação do HTML, eventos e animações;
- Bootstrap: *framework* CSS que fornece estilos e componentes predefinidos para uma interface responsiva e visualmente atraente;
- Base de dados relacional: utilizado para armazenar os dados do sistema;
- *Templates* Django: utilizados para a criação de páginas dinâmicas com conteúdo reutilizável.
- HTML Canvas: para renderização e manipulação de gráficos e animações em tempo real;
- Python: linguagem de programação utilizada para programação do sistema de controlo do lança bolas juntamente com o servidor web;
- *Javascript*: linguagem de programação utilizada para adicionar interatividade e dinamismo às páginas *web*;
- C++: linguagem de programação utilizada para o desenvolvimento do GRBL, responsável pelo controlo dos motores de passo e posicionamento preciso da máquina.

Considerando que já existe um sistema com duas baterias de 12V em série, foi proposto um sistema de carregamento que permite selecionar a bateria a ser carregada. Dessa forma, o carregamento pode ser realizado em simultâneo com a utilização do lançador, desde que seja garantida a percentagem mínima de bateria necessária.

O objetivo final é criar um sistema eficiente, robusto e de fácil compreensão, com capacidade de expansão, permitindo a implementação de novas funcionalidades com base no desenvolvimento atual, incluindo a criação de páginas responsivas e um software configurável e altamente escalável.

Capítulo 4

4. Materiais e métodos

Este capítulo tem como objetivo apresentar uma visão geral das tecnologias e componentes utilizados no projeto. Serão abordados os conceitos fundamentais de cada um deles, destacando suas vantagens. Com esta contextualização, procura-se criar uma introdução abrangente ao tema, que será explorado com mais detalhes nos próximos capítulos.

4.1 HTTP

O HTTP (*Hypertext Transfer Protocol*) é o protocolo de comunicação utilizado na *World Wide Web* para transferir dados entre computadores. Ele é o responsável por permitir a comunicação entre clientes (navegadores *web)* e servidores *web*. O protocolo define como as mensagens serão estruturadas e transmitidas, e é utilizado por todos os navegadores *web* para comunicar com o servidor.

4.2 *Bootstrap*

Bootstrap é um *framework* de CSS amplamente utilizado para o desenvolvimento de interfaces web responsivas. Oferece classes e componentes predefinidos, facilitando a criação de páginas web adaptáveis a diferentes dispositivos. No *boostrap* existem inúmeros recursos que são amplamente utilizados em páginas responsivas, como por exemplo o *flex grid,* que organiza os componentes em relação ao tamanho de visualização do dispositivo. As vantagens de o utilizar passam por ser uma *framework* gratuita, altamente escalável e bem documentada.

4.3 Ajax

Se considerarmos a situação em que, na web, cada vez que alguém precisa fazer uma pesquisa no Google, arrastar o mapa no *Google Maps* ou visualizar um tópico

no Twitter, toda a página precisa de ser carregada, tornaria as aplicações extremamente lentas e obsoletas por carregarem uma enorme quantidade de dados repetidamente. Para resolver este problema, existe a solução de atualizar a página *web* através de chamadas assíncronas, onde apenas uma pequena parte da página web é renderizada. Isso permite que apenas os elementos necessários sejam carregados e exibidos, tornando o processo mais eficiente. O AJAX (*Asynchronous JavaScript and XML)*, é uma tecnologia que pode solucionar o problema descrito. O termo AJAX foi mencionado por Jesse James Garrett em seu artigo "*Ajax: A New Approach to Web Applications*" (Uma nova abordagem para aplicações web), onde descreveu esta técnica que já estava a ser utilizada anteriormente na programação *web*. Desde então, o AJAX tornou-se uma tecnologia popular na programação *web*, melhorando a experiência do utilizador e aumentando a eficiência no processo de troca de informações entre o servidor e cliente [8].

4.4 *Long Polling*

Long polling é uma técnica utilizada com AJAX para solicitar informações do servidor de uma maneira mais eficiente. Diferente do *polling* tradicional, em que o cliente solicita informações ao servidor de forma periódica, o *long polling* mantém a conexão aberta entre o cliente e o servidor até que novas informações estejam disponíveis.

A técnica funciona da seguinte maneira: quando o cliente solicita informações do servidor, o servidor não fecha imediatamente a conexão e mantem-na aberta até que haja novas informações para enviar. Assim que o servidor tenha uma mensagem para enviar, ele responde à solicitação do cliente com essa mensagem, e imediatamente abre uma nova conexão para manter a comunicação aberta.

Com isso, o cliente web pode receber as informações instantaneamente, sem a necessidade de recarregar a página inteira. Além disso, é fácil de ser implementado e entrega as mensagens sem atrasos (Figura 8) [9].

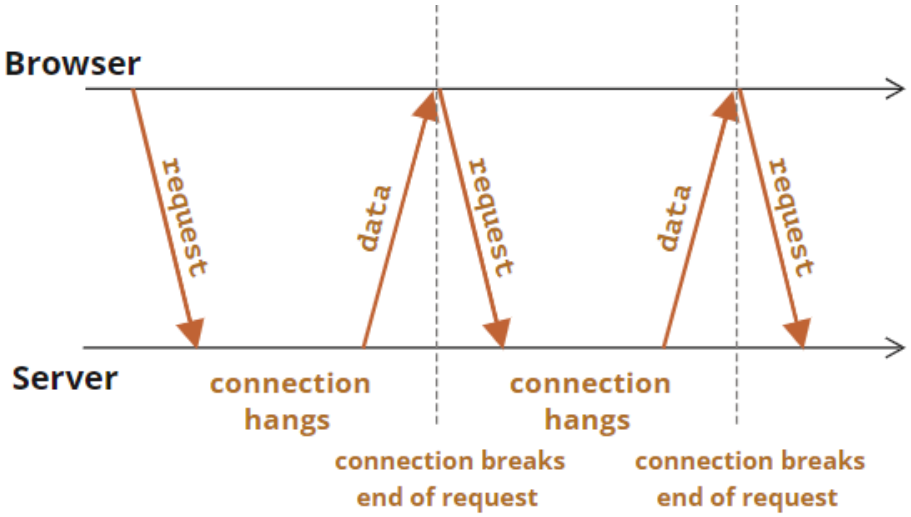


Figura 8 - Fluxo de informação através de long polling [9]

4.5 *Websockets*

Websockets é um protocolo de comunicação *full-duplex* que permite a transmissão bidirecional de dados entre um servidor e um cliente *web*. A primeira proposta de arquitetura "RFC" (*Request For Comments*) para o *Websocket* foi finalizada em dezembro de 2011, mas apenas em 2013 o projeto foi concluído e implementado em servidores web como o Apache e o NGINX. Atualmente, todos os navegadores modernos oferecem suporte ao *Websocket*.

Ao contrário do HTTP tradicional, que é baseado em solicitações e respostas, o *Websocket* permite uma comunicação mais eficiente e contínua entre o servidor e o cliente. Estabelece uma conexão persistente e bidirecional entre o navegador e o servidor, o que significa que o servidor pode enviar dados para o cliente a qualquer momento, sem que o cliente solicite uma ligação.

Na atualidade existe uma implementação muito solida deste protocolo nos navegadores, o que permite aos projetistas de *software* não ponderarem se existe a possibilidade da aplicação não funcionar em certos dispositivos.

A Tabela 2 indica quais os navegadores que suportam o protocolo, e as suas respetivas versões de suporte. É importante realçar que o *internet explorer*, um navegador descontinuado, já implementava este protocolo nas suas duas últimas versões [10][11].

Tabela 2 - Suporte de versão WebSocket por diferentes browsers [11].

Chrome	Edge	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS	Samsung Internet	Opera Mini	Opera Mobile	UC Browser for Android	Android Browser	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
		3.1-4	2-3.6													
4-14		5-5.1	4-5	10.1			3.2-4.1									
15		6-6.1	6-10	11.5	6-9		4.2-5.1			12		2.1-4.3				
16-104	12-104	7-15.6	11-104	12.1-90	10		6-15.6	4-17.0		12.1		4.4-4.4.4				
105	105	16.0	105	91	11	105	16.0	18.0	all	64	13.4	105	104	10.4	13.18	2.5
106-108		16.1-TP	106-107				16.1									

O *Websocket* não é limitado a ser usado apenas para comunicação entre navegadores e servidores *web*. Este também pode ser utilizado para criar canais de comunicação entre diferentes dispositivos, tudo o que é necessário é um servidor que implemente o protocolo *Websocket*. Esta flexibilidade torna-o uma tecnologia ideal para diversas aplicações, como jogos online, gráficos em tempo real, *chat's* em tempo real, etc. Também é possível usar o *WebSocket* em soluções IoT (*Internet of Things*), como por exemplo a abertura de portas remotas devido à sua capacidade de manter uma conexão persistente, e permitir a transmissão de dados em tempo real. A seguinte figura mostra como é efetuada uma conexão *websocket*.

Figura 9 - Esquema simplificado de uma conexão WebSocket [10].

4.6 Protocolos de Segurança

4.6.1 HTTPS

O HTTPS (*Hypertext Transfer Protocol Secure*) é uma versão mais segura do HTTP, o protocolo padrão para a transferência de dados na internet. O HTTPS usa criptografia SSL (*Secure Sockets Layer*) ou TLS (*Transport Layer Security*) para proteger as informações transmitidas entre o cliente e o servidor. Isso significa que todos os dados são criptografados, tornando-os ilegíveis para qualquer pessoa que intercete a comunicação.

O HTTPS é amplamente utilizado para proteger informações confidenciais, como informações bancárias e dados pessoais. Sites que utilizam HTTPS exibem um cadeado verde na barra de endereço do navegador, indicando que a conexão é segura. [12]

4.6.2 WSS

O WSS (*WebSocket Secure*) é uma versão segura do *WebSocket*. O WSS usa a criptografia TLS para proteger as informações transmitidas, o que significa que todos os dados são criptografados, tornando-os ilegíveis para qualquer pessoa que intercete a comunicação.[13]

4.7 Controlador GRBL com ESP32

O controlo dos motores é realizado pelo ESP32 (Figura 10), utilizando o software de CNC (Controlo numérico computacional) GRBL previamente instalado e configurado para o lançador de bolas. Isso permite o comando em tempo real dos motores. O GRBL já possui características essenciais para este projeto, como o armazenamento de posições, o controlo dos fins de curso para referenciamento dos eixos e funções de velocidade para o controlo dos rolos que realizarão os lançamentos.

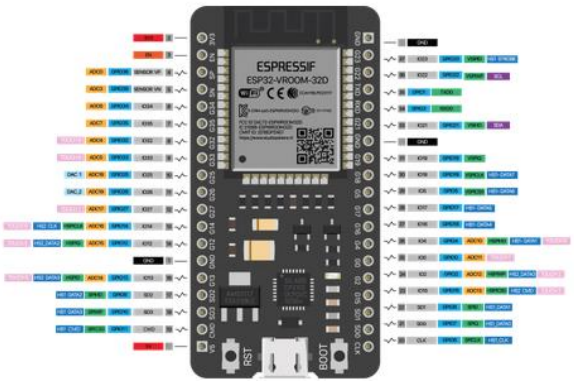


Figura 10 - Controlador ESP32 que opera o GRBL [14]

4.8 *Single board computer* (SBC)

Um dos componentes-chave no contexto de utilização do lançador de bolas é o *Single Board Computer* (SBC), ou computador de placa única. Um SBC é um computador completo integrado em uma única placa, contendo todos os principais componentes de hardware, como processador, memória, armazenamento e interfaces de entrada/saída.

No caso do lançador de bolas, este desempenha um papel crucial no controlo e gestão do sistema, atua como o cérebro do lançador, processando os comandos enviados pela aplicação web e coordenando todas as atividades do lançador. Além disso, o SBC oferece a capacidade de se conectar a diferentes dispositivos e sensores sem comprometer o desempenho.

Um aspeto importante do seu uso no lançador de bolas, é a capacidade de ser compacto e de baixo consumo de energia, o que permite maior autonomia em funcionamento com baterias.

Existem diversos SBC disponíveis no mercado, com diferentes especificidades. Para o presente projeto foi necessário um controlador com elevado processamento, que tivesse portas USB e saída de vídeo. De encontro com estes requisitos foi escolhido o Raspberry pi 4 por ser de baixo custo e atender às necessidades do projeto (Figura 11).

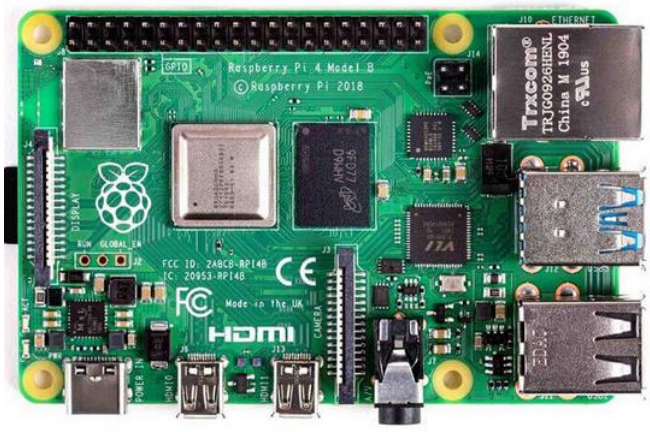


Figura 11 - Raspberry pi 4

Na seguinte tabela são apresentadas as principais características do modelo de (SBC) escolhido para este projeto.

Tabela 3 - Características do SBC (raspberry pi 4)

Raspberry PI 4	
Processador	Processador Broadcom 2711 Quad-core Cortex-A72 64-bit SoC @ 1,5 GHz
Memória	8GB de memória RAM
Conexão	WiFI 2,4 GHz / 5,0 GHz IEEE 802.11.b/g/n/ac | Bluetooth 5.0
Acesso	GPIO com 40 pinos
Alimentação	5 V / 3 A via conector USB tipo C
Ligações para programção	2 portas micro HDMI, vídeo de 4k | 4 portas USB
Extras	Interface para display (DSI) + Interface para câmera (CSI)

Capítulo 5

5. Arquitetura de Hardware

No presente capítulo será discutida a arquitetura de hardware do lançador de bolas, fornecendo uma visão detalhada das especificações que motivaram à escolha desta arquitetura, e clarificação dos pontos chave que levaram à escolha dos componentes.

A arquitetura de hardware do lançador de bolas adota um modelo de comunicação mestre-escravo, no qual o Raspberry Pi 4 atua como mestre e o ESP32 como escravo. A Figura 12 representa um modelo simplificado da arquitetura de hardware.

A estrutura do presente projeto baseia-se na comunicação entre o Raspberry Pi 4, que atua como Mestre, e o ESP32 como escravo. O Raspberry pi 4 ciclicamente envia instruções para receber o estado atual da máquina, de forma a ter a gestão total do lançador de bolas.

O ESP32, utiliza o software de controlo de comando numérico GRBL, por meio do qual recebe comandos do mestre no formato G-CODE, por este meio executa as instruções de controlo e fornece *feedback* sobre a execução.

Além disso, o ESP32 é capaz de enviar o estado atual da máquina, incluindo a posição dos motores, sempre que solicitado. É importante ressaltar que o ESP32 é o único componente do sistema que possui acesso direto aos atuadores e sensores e, portanto, desempenha a função de gestão dos movimentos e estados do lançador de bolas.

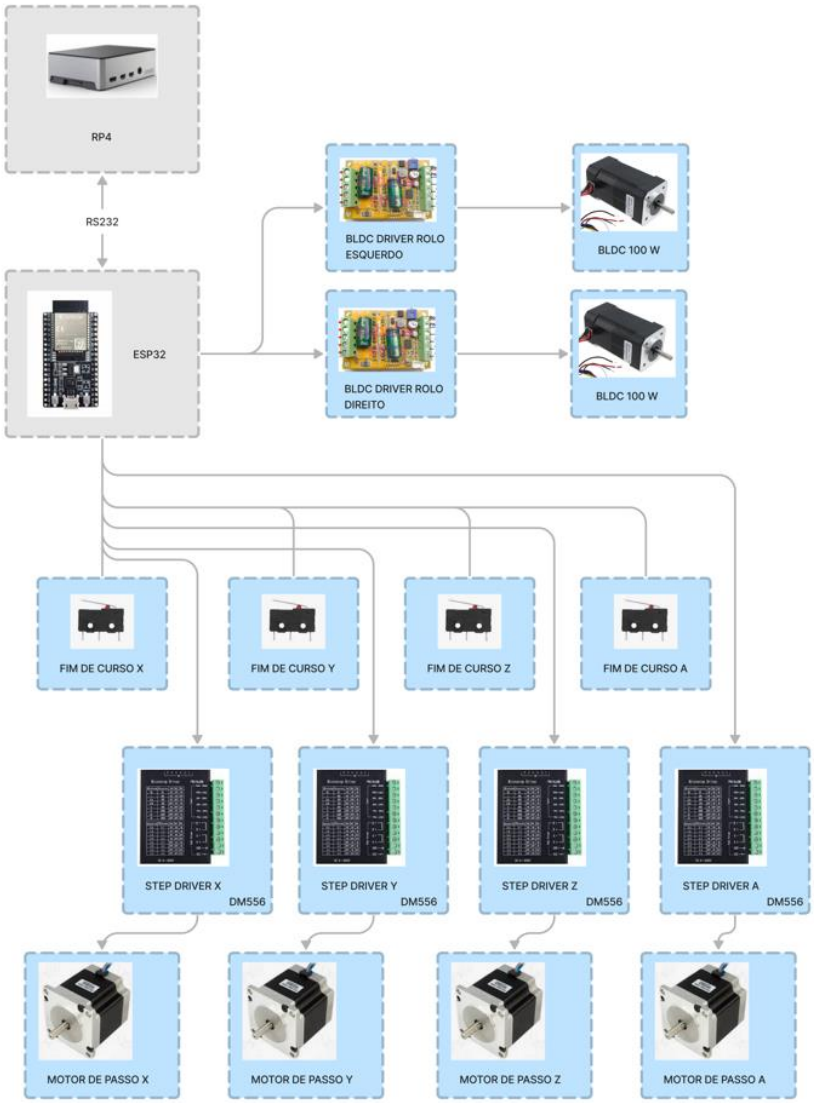


Figura 12 - Arquitetura simplificada de hardware

Neste capítulo seja discutico o conversor de sinal 3.3v para 5.5v que permitiu ligar o ESP32 aos *drives* dos motores, bem como a escolha da tipologia de controlo da placa. Serão destacadas as principais vantagens desse circuito e apresentados os cálculos para o seu dimensionamento.

Serão abordados também aspetos relacionados ao controlo do hardware, incluindo o acionamento dos motores passo-a-passo e dos motores BLDC (*brushless direct current*), detalhados os drivers utilizados e a configuração necessária para o seu correto funcionamento.

For fim no processo de desenvolvimento do lançador de bolas, foi elaborado um esquema elétrico completo, nele é possível retirar toda a informação necessária

para compreender melhor as conexões elétricas e entender como os componentes estão interligados. O esquema elétrico pode ser visualizado nos anexos deste documento. Será mencionada a gestão de carga da bateria e discutido o esquema elétrico que gere a sequência de carga.

5.1 Seleção dos drivers para os motores, configuração e funcionamento

Quando este projeto foi iniciado os *drivers* de motores já tinham sido selecionados e adquiridos previamente, o que levou a avaliar apenas se atendiam às necessidades para o lançador de bolas. Os *drivers* já escolhidos foram os DM556, capazes de fornecer até 5.6 A por motor, adequados para a aplicação em questão. Os *drivers* também possuem um sistema de *microstepping*, permitindo um controlo preciso dos motores.

Em relação aos drivers dos motores dos rolos (Figura 13), foi necessário substituí-los, pois um deles queimou devido a uma sobrecarga durante o início do projeto.

Figura 13 - Representação dos rolos com uma bola em fase de lançamento [1]

Com esta informação, foi decidido encomendar drivers com maior capacidade. Optando por adquirir drivers de 380W de 3 fases (Figura 14) que garantem uma capacidade de alimentação robusta para os motores dos rolos. Esta escolha visou evitar problemas futuros e assegurar a estabilidade e o desempenho do sistema.

Figura 14 - 380W 3 Phase BLDC Brushless Motor [15]

5.2 Conversor de sinal 3.3v para 5.5v

Para garantir o controlo adequado dos drivers DM556 através do ESP32, foi necessário desenvolver uma placa eletrônica que permitisse o controlo através das saídas a 3.3V do microcontrolador ESP32, com os drivers DM556 que requerem uma tensão mínima de controlo de 5V.

A escolha da tipologia de controlo da placa foi influenciada pelo diagrama de controlo presente no manual dos drivers, conforme pode ser observado na Figura 15. Desta forma, optou-se por utilizar a conexão "*connection to open-collector signal*", que consiste na utilização de um transístor NPN para fechar o circuito com o GND.

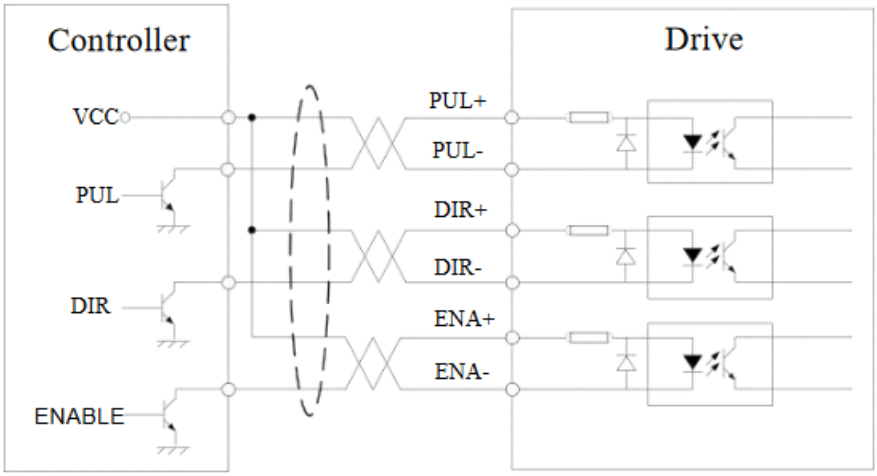


Figura 15 – Ligação do tipo "open-collector signal" (Fonte: stepperonline.com).

Uma das principais vantagens deste circuito é que caso seja necessário aumentar a tensão de controlo dos drivers para até 12V ou 24V, a estrutura da placa permanece inalterada. Além disso, caso se opte por utilizar o controlo PNP, a complexidade da placa seria significativamente maior.

No desenvolvimento da circuito foi necessário avaliar quais as possibilidades de falha que poderiam surgir na replicação dos sinais de comando. Para tal realizou-se um estudo do tipo de transístores que iriam ser utilizados, e a necessidade de incluir um circuito de *boost* na transição ascendente do sinal, de forma a aumentar a velocidade de comutação.

Primeiramente foi observado com o osciloscópio o sinal de controlo do GRBL com o ESP32, no qual foi possível verificar que a velocidade de comutação pode alcançar 300 kHz, sendo que, os pulsos de controlo gerados pelo GRBL possuem um *Duty Cycle* baixo, ou seja, a maior parte do tempo está no nível baixo. Com esta informação, optou-se por usar 10 vezes a velocidade máxima observada como referência para o circuito a ser projetado. Sendo assim, o circuito projetado deverá conseguir comandar sinais que possam ter uma frequência de até 3 MHz.

Após uma análise cuidada foi escolhido o transístor 2N222, que é um transístor BJT de alta frequência, juntamente com um circuito de *boost* que fica em paralelo com a resistência de polarização da *gate*. Este circuito é composto por um condensador em série com uma resistência, esta configuração permite encurtar o tempo de transição do sinal.

5.2.1.1 Cálculo do circuito *boost*

O calculo tem por base a frequência escolhida para o circuito 3Mhz, para tal é calculado o periodo e dividido por dois para retirarmos somente o tempo ascendente da transição:

$$t = \frac{1/3Mhz}{2} = 166.5\,ns$$

É definido uma resistência de 50 ohms para limitar a corrente maxima de pico a 50 mA:

$R = \frac{(3.3-0.7)}{50mA} = 52\ ohms$

Para calcular o tempo de carga do condensador, podemos usar a equação:

$$t = R * C$$

$Onde$:
$t =\ tempo\ de\ carregamento\ (\ minimo\ \rightarrow\ 166.5\,ns\ )$
$R = resistência\ em\ ohms\ (52\ ohms)$
$C = Capacidade\ do\ condensador\ em\ farads$

$$Substituindo\ os\ valores, obtemos:$$
$$C = \frac{166.5\,ns}{52\ ohms} = 3.2\,nF$$

No circuito final, foi utilizado um condensador de 4.7 nF e uma resistência de 50 ohms. Com este circuito, a *gate* do transístor é polarizada durante a transição utilizando uma resistência de 50 ohms em série com o condensador. Esta configuração permite uma polarização com corrente elevada, resultando em um aumento na velocidade de transição. Após o condensador estar na sua carga máxima, a resistência de 1 Kohm passa a polarizar a *gate*. O circuito final pode ser observado na Figura 16.

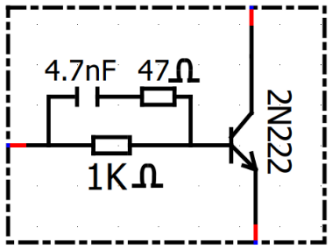


Figura 16 - Circuito com transístor NPN para controlo dos motores de passo

Na Figura 17 podemos observar que sem o circuito de boost o tempo de subida do sinal é de 111.2 ns, após a implementação do circuito a velocidade de comutação é reduziu para 73.75 ns. Esta redução traduz-se num aumento de 33.65% de velocidade de comutação face à velocidade sem o circuito *boost*. Finalizando a frequência máxima que o circuito consegue comandar é de aproximadamente 6.67MHz.

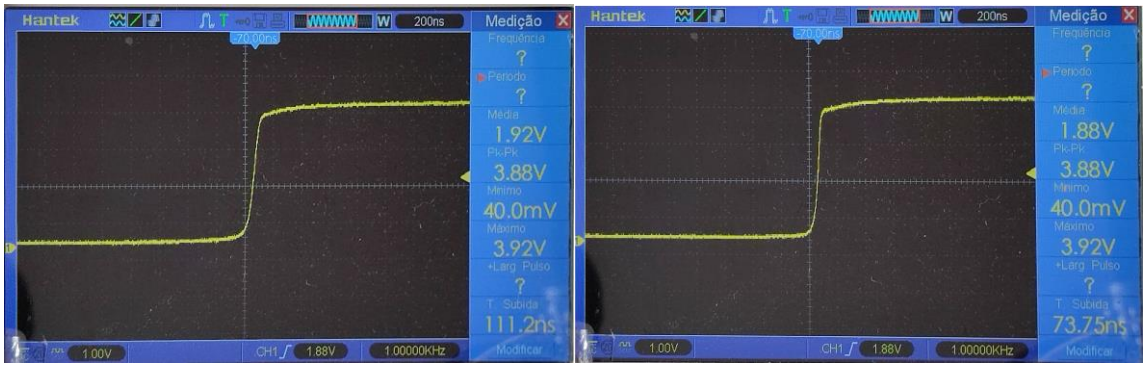


Figura 17 - Comparação entre circuito NPN de controlo sem e com boost à gate

Na Figura 18 é possível visualizar a placa de controlo após ser soldada e estar pronta para funcionamento.

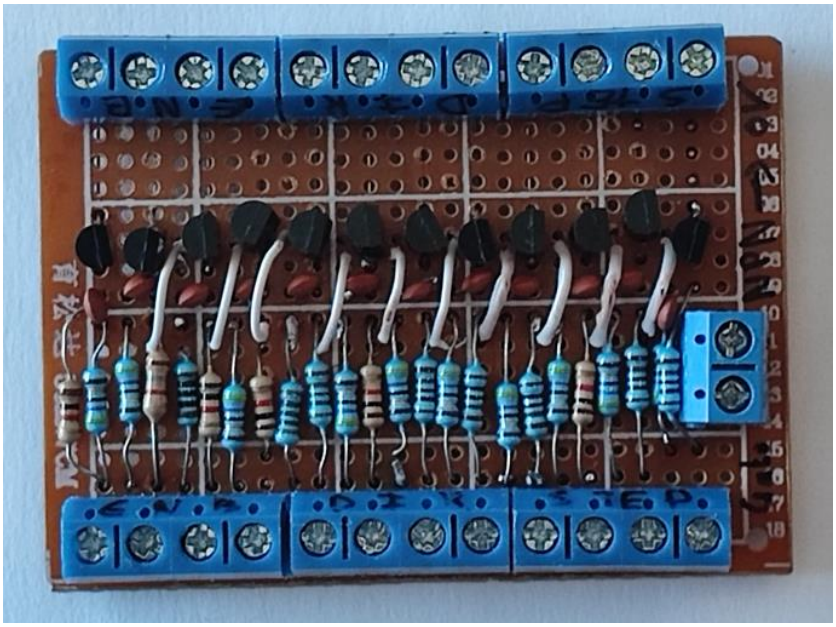


Figura 18 - Placa eletrónica desenvolvida para comandar os controladores dos motores de passo

5.3 Controlo do hardware

5.3.1 Acionamento dos motores passo-a-passo

Os motores de passo são acionados enviando sinais elétricos sequenciais às bobinas do motor. Para facilitar este processo e garantir precisão no controlo, é possivel utilizar drivers de motor de passo que são amplamente utilizados no mercado. Um deles, o DM556, possui características que tornam o seu comando fácil e eficiente. Alem de ser facil de comandar, o DM556 possui recursos avançados, como por exemplo o sistema de *microsteps*, que permite um controlo ainda mais preciso e suave do motor.

O motores de passo que são utilizados neste projeto são do tipo NEMA23 com ligação a 4 fios, no qual o primeiro pár de fios (A+, A-) devolve a ligação de uma bobine, e o segundo pár de fios (B+, B-) devolve a ligação à segunda bobine (Figura 19).

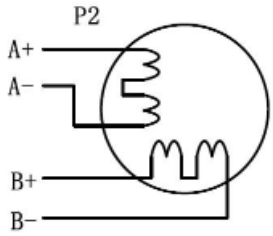


Figura 19 - Esquema dos enrolamentos da bobine de um motor de passo a 4 fios
(Fonte: Biblioteca de componentes do software QeletroTech)

Os *drivers* dos motores de passo possuem uma sequência de controlo simples que permite o acionamento do motores. A Figura 20 representa a ligação do driver de passo a passo do eixo “X” ao conector que posteriormente vai ser ligado ao motor.

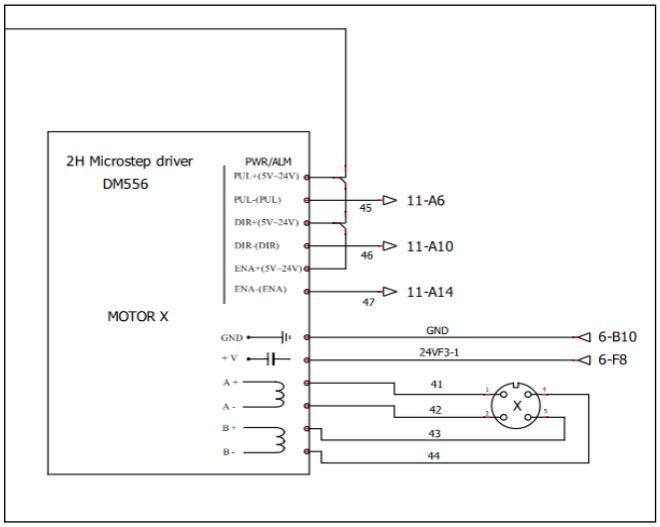


Figura 20 - Esquema elétrico de conexão do driver DM556
(Fonte: próprio autor).

A sequencia de controlo é a seguinte:

- O sinal "*ENA*" é responsável por habilitar a potência do driver. Quando ativado, fornece energia ao motor de passo, permitindo o seu funcionamento.
- Já o sinal "*DIR*" é utilizado para controlar a direção de rotação do motor. Quando esse sinal está em nível baixo (0), o motor gira em uma direção específica, e quando está em nível alto (1), a direção de rotação é invertida.

- Com o sinal "*STEP*" a cada pulso recebido o motor dá um passo na sequência de movimentos.

Antes de começar a operar o motor, é necessário realizar a configuração dos "*switch's*" que determinam a corrente aplicada ao motor e o número de "*microsteps*" a serem utilizados para o controlo. O fabricante do driver disponibiliza duas tabelas que fornecem as correspondências entre os estados dos pinos, e as configurações desejadas. É importante realizar essa configuração corretamente, pois uma configuração inadequada pode resultar em baixo desempenho do motor ou até mesmo sobreaquecimento, o que reduz a vida útil do mesmo.

O primeiro, segundo e terceiro "*switch*" na posição ON/OFF correspondem à corrente aplicada ao motor, cujas configurações podem ser consultadas na Tabela 4.

Tabela 4 - correspondência do "switch" 1 a 3 em relação à corrente do driver.

(Fonte: stepperonline.com)

Peak Current	RMS Current	SW1	SW2	SW3
1.8A	1.3A	ON	ON	ON
2.1A	1.5A	OFF	ON	ON
2.7A	1.9A	ON	OFF	ON
3.2A	2.3A	OFF	OFF	ON
3.8A	2.7A	ON	ON	OFF
4.3A	3.1A	OFF	ON	OFF
4.9A	3.5A	ON	OFF	OFF
5.6A	4.0A	OFF	OFF	OFF

O quarto "switch" na posição OFF limita a corrente a 50% da configuração definida, na posição ON a corrente configurada entre os Switches 1 a 3 é liberada integralmente.

Os restantes switch's de 5 a 8 correspondem à configuração dos microsteps, a configuração pode ser visualizada na Tabela 5.

Tabela 5 - correspondência de configuração dos "microsteps" por "switch's"
(Fonte: stepperonline.com)

Microstep	Steps/rev.(for 1.8°motor)	SW5	SW6	SW7	SW8
2	400	OFF	ON	ON	ON
4	800	ON	OFF	ON	ON
8	1600	OFF	OFF	ON	ON
16	3200	ON	ON	OFF	ON
32	6400	OFF	ON	OFF	ON
64	12800	ON	OFF	OFF	ON
128	25600	OFF	OFF	OFF	ON
5	1000	ON	ON	ON	OFF
10	2000	OFF	ON	ON	OFF
20	4000	ON	OFF	ON	OFF
25	5000	OFF	OFF	ON	OFF
40	8000	ON	ON	OFF	OFF
50	10000	OFF	ON	OFF	OFF
100	20000	ON	OFF	OFF	OFF
125	25000	OFF	OFF	OFF	OFF

No lançador de bolas projetado, os motores foram configurados com 16 *microsteps*. Levando em consideração que os motores possuem um ângulo de passo de 1.8º, e se fossem utilizados passos completos (*full step*), seriam necessários 200 passos para completar uma volta. No entanto, com os 16 *microsteps*, são necessários 16 vezes mais passos para completar uma volta, ou seja, 3200 passos. Estas configurações para controlo podem ser ajustadas no GRBL, sendo que, por padrão, está configurado para 16 *microsteps*.

5.4 Gestão de carga da bateria

O lançador de bolas é uma máquina que precisa funcionar sem estar ligada diretamente à rede elétrica. Para solucionar este problema, foi instalado um conjunto de duas baterias, com 12 volts cada, em série e desenvolvido um sistema para carregá-las. O sistema permite que o utilizador verifique a carga da bateria, e escolha qual delas deseja carregar por meio de um seletor localizado no quadro elétrico. O esquema de carregamento pode ser visualizado na Figura 21.

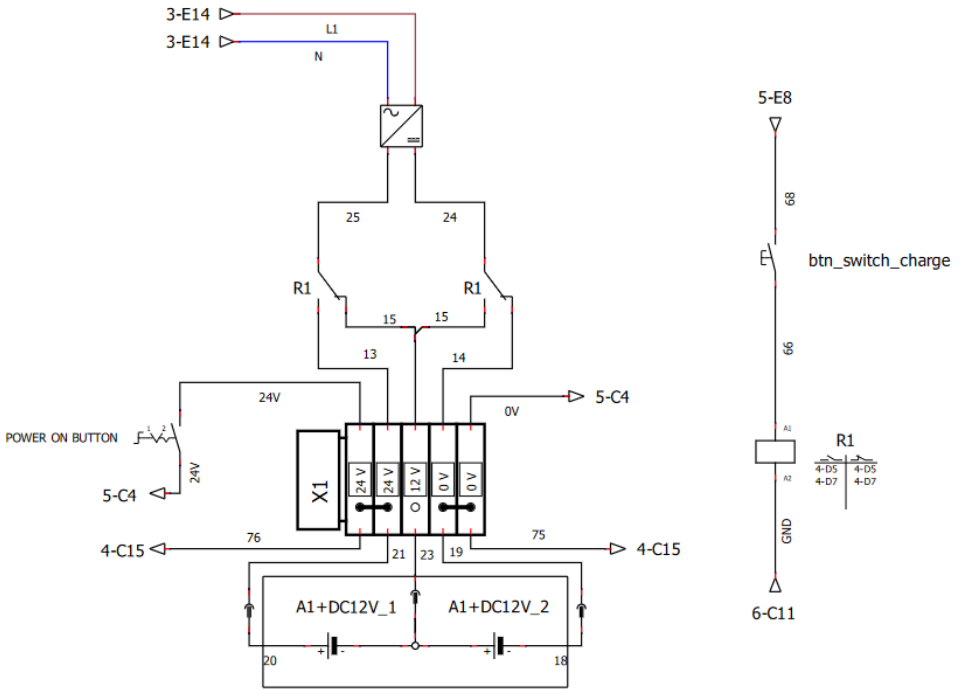


Figura 21 - Esquema elétrico do carregamento das baterias com seletor de carga

Uma das vantagens deste sistema é que o lançador de bolas pode estar a funcionar enquanto as baterias estão a ser carregadas. No entanto, é importante lembrar que o utilizador nunca deve ligar o lançador de bolas quando a carga das baterias está no mínimo.

Durante o desenvolvimento da aplicação para o lançador de bolas, foi criada uma maquete que simula fielmente o ambiente real. Esta maquete, composta por motores de passo de menor dimensão e uma estrutura impressa em 3D para os sensores de fim de curso, foi capaz de reproduzir todos os movimentos e sinais necessários. Nesta maquete existe um ESP32 conectado a pequenos drivers, que por sua vez comandam os motores dos rolos e motores de passo. Esta abordagem permitiu realizar testes e ajustes na aplicação antes que o lançador de bolas estivesse pronto para operar efetivamente. Na Figura 22 é possível visualizar a maquete desenvolvida para simulação.

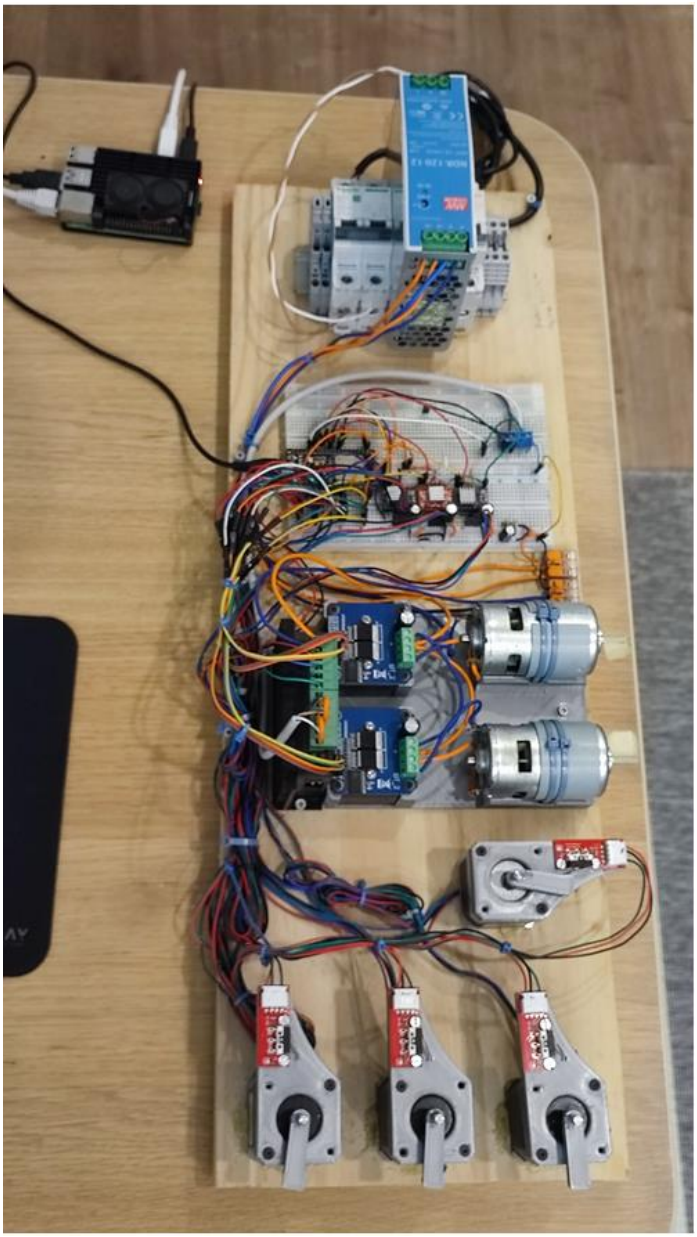


Figura 22 - Maquete desenvolvida para simulação do lançador de bolas

Capítulo 6

6. Software de controlo e monitorização

Para o controlo foi adotada uma arquitetura que prioriza a escalabilidade da aplicação, juntamente com uma interface de utilizador altamente responsiva e fácil de usar. Para alcançar esses objetivos, foram utilizados *frameworks*, Django para o *backend* e *Bootstrap* para o *frontend*, proporcionando uma redução significativa no tempo de desenvolvimento.

6.1 Arquitetura de software

A estrutura de software do lançador de bolas é baseada na arquitetura MTV (*Model-Template-View*) do Django, que é amplamente utilizada em aplicações *web*. Essa arquitetura proporciona uma organização eficiente do código e facilita o desenvolvimento da aplicação. A Figura 23 apresenta uma representação visual da arquitetura de software simplificada do lançador de bolas, mostrando como os diferentes componentes se relacionam e interagem entre si.

Figura 23 - Arquitetura de software do lançador de bolas

Ao sistema base *Django* foram adicionados *threads* de controlo para os modos (manual, automático e lances) de forma a estruturar o processo de controlo em módulos funcionais.

Explicando com tudo se interliga, quando um utilizador se conecta ao serviço web do lançador de bolas, a primeira iteração é devolver uma página web que contém html, css e *javascript*. Essas tecnologias permitem a criação de uma interface dinâmica e responsiva ao utilizador. O *javascript* presente na página web é responsável por comunicar com o *backend* e atualizar o DOM (*Document Object Model*) dinamicamente com as respostas.

As próximas iterações com o *backend* são sempre assíncronas, usando recursos como o AJAX e *WebSockets*. Sempre que o utilizador faz um pedido de uma ação que vá realizar movimentos no lançador, o pedido é encaminhado para o "Engine lançador de bolas", que é um módulo em Python desenvolvido para gerenciar os pedidos de movimento e encaminhá-los para as *threads* de controlo.

Todas a estrutura “Engine lançador de bolas” foi desenvolvida de forma que seja altamente escalável e abstrata para os níveis mais altos de controlo. Esta forma de programar permite criar aplicações escaláveis, cada vez mais evoluídas e sem que o nível de complexidade da programação aumente.

6.2 *Templates* HTML/CSS do lançador de bolas

Quando um utilizador se conecta à aplicação *web* ou navega entre os menus, são retornados *templates* do *backend* para a aplicação. Os *templates* são componentes do Django essenciais no desenvolvimento das aplicações. Resumidamente os *templates* são uma estrutura do Django para retornar páginas estáticas que contêm HTML, CSS e *JavaScript*, com a diferença de serem previamente renderizadas segundo uma lógica de dados no *backend*.

Utilizar *templates* traz vários benefícios. Em primeiro lugar promove a separação clara entre a lógica da aplicação *backend* e a apresentação visual *frontend*. Além disso, facilitam a manutenção e reutilização de código, pois existe a possibilidade de criar *templates* base que podem ser estendidos por outros *templates*, o que torna possível criar *layouts* consistentes em toda a aplicação, evitando duplicação desnecessária de código.

Em segundo lugar, os *templates*, oferecem suporte a recursos avançados, como a incorporação da linguagem Python na construção de HTML, o que permite exibir dinamicamente dados armazenados na base de dados.

Com todas essas vantagens em mente, é apresentada a descrição detalhada dos *templates* que compõem esta aplicação.

6.2.1 *Template* “Home”

No *template* "Home", o utilizador é recebido com um menu de boas-vindas que oferece opções de *login* e referência do lançador. Este é o primeiro *template* apresentado quando o usuário acede ao servidor.

É também a primeira página da aplicação e permite que o utilizador faça a referência dos eixos do lançador de bolas, e aceda ao menu de login. O *template* é totalmente responsivo, ou seja, adapta-se dinamicamente a dispositivos móveis, *desktops* e *tablets*. A Figura 24 ilustra o *template* no formato *desktop*, enquanto a Figura 25 mostra como ele se adapta para uma visualização em dispositivos móveis, com os componentes redimensionados para uma experiência otimizada. É também possível observar o menu fica basculante quando apresentado em formato *web mobile*, este menu é transversal a todos os *templates*.

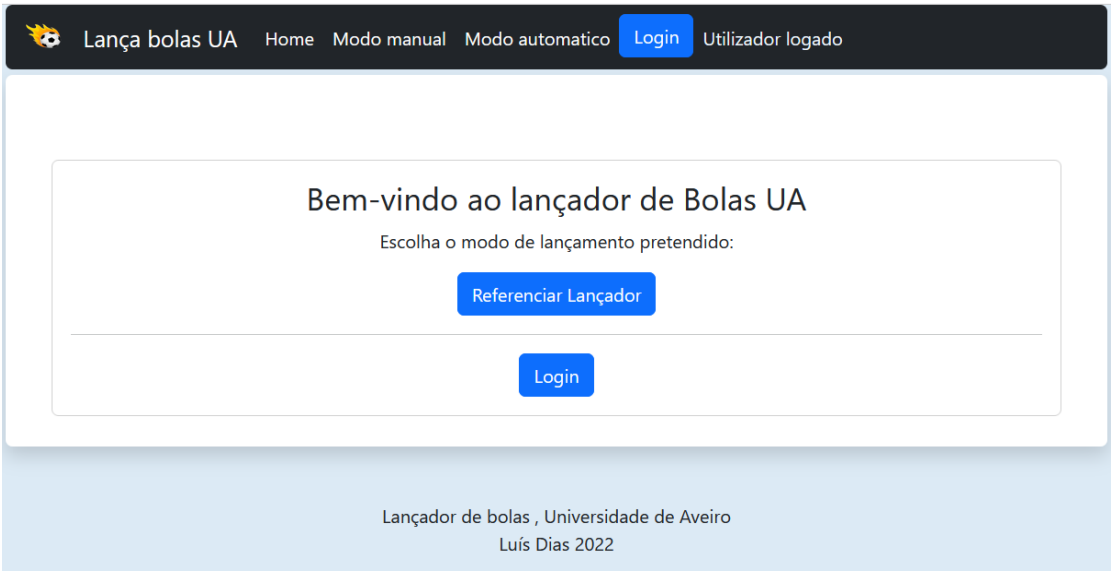


Figura 24 - Template "home" em formato desktop

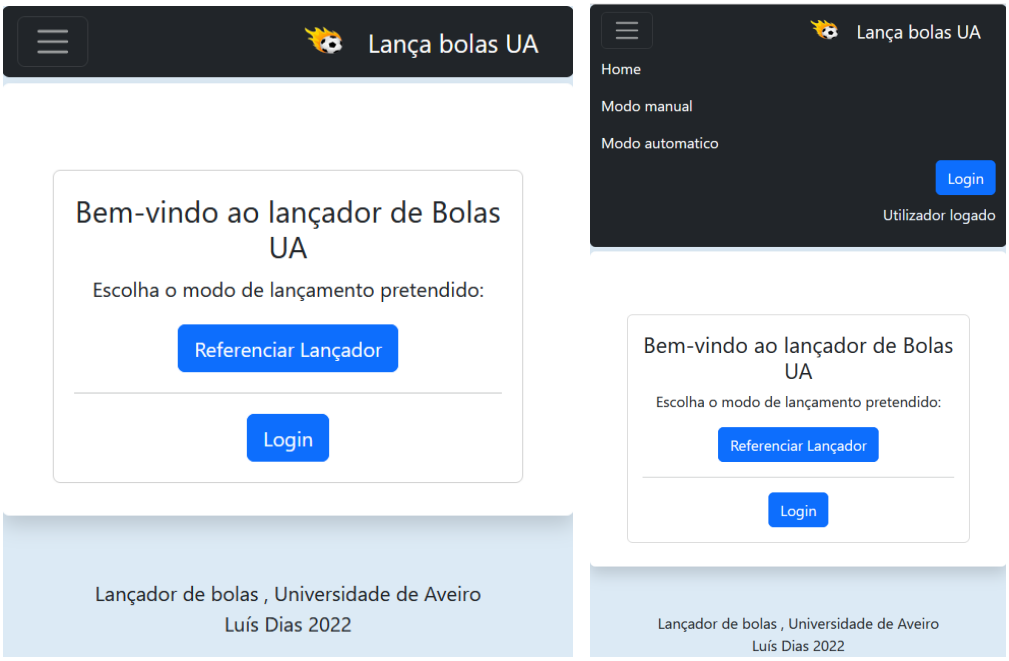


Figura 25 - Template "home" em formato tablet e smartphone

6.2.1.1 *Template* "Modo Manual"

Em seguida, temos o *template* "Modo Manual" (Figura 26), que é acedido através do menu superior da aplicação, nele é possível encontrar um conjunto de ferramentas de comando para movimentar o lançador e lançar bolas.

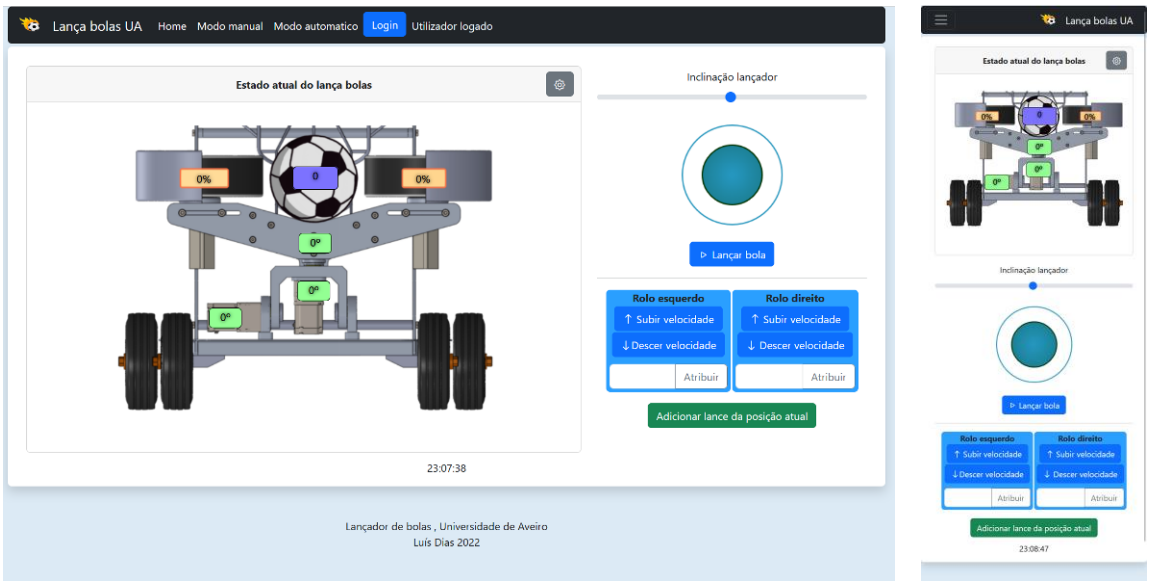


Figura 26 - Template do "Modo manual" em formato desktop e mobile

Existe duas zonas definidas para este menu, zona de visualização e zona de comando. Na zona de visualização podemos ver a representação do lançador de bolas com a indicação em tempo real da posição dos eixos, a velocidade dos rolos e quantidade de bolas que foram lançadas no modo manual.

Existe também um pequeno icon no canto superior da janela da zona de visualização, que é apresentado apenas quando um utilizador está autenticado. Com este icon é possível aceder a um *template* de comissionamento que vai ser descrito mais à frente.

Na zona de controlo estão todas as ferramentas necessárias para controlar e para lançar bolas numa interface de controlo em tempo real.

Por ordem de visualização é possível ver um *slide bar* para posicionar em ângulo o eixo de inclinação, um *Joystick* desenvolvido em *canvas* que permite ao utilizador movimentar o eixo x e y. Logo após existem duas janelas para controlo dos rolos. Cada janela tem dois botões e uma área de inserção e atribuição de velocidade dos rolos. Com estes controlos é possível definir e atribuir velocidade aos rolos, com os botões ajustar a velocidade fina.

Por último existe um botão para adicionar um lance da posição atual em que se encontra a máquina. Este botão abre uma Janela do tipo modal (Figura 27), que apresenta um formulário com os valores que estão a ser utilizados para lançar a bola, o utilizador necessita obrigatoriamente de inserir o nome do lance e, se preferir, personalizar o lance atual antes de o inserir na base de dados.

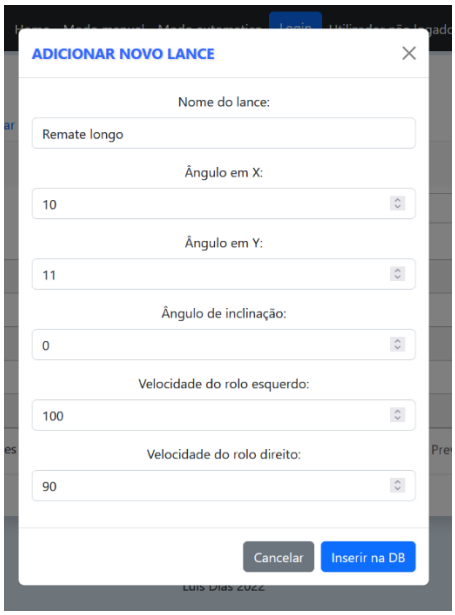


Figura 27 - Janela do tipo modal para inserção do lance atual na base de dados

Se todos os valores forem validados é apresentado um alerta (Figura 28) com a mensagem “lance inserido com sucesso!”, caso contrário existe um sistema de verificação dados que lançam um alerta de erro.

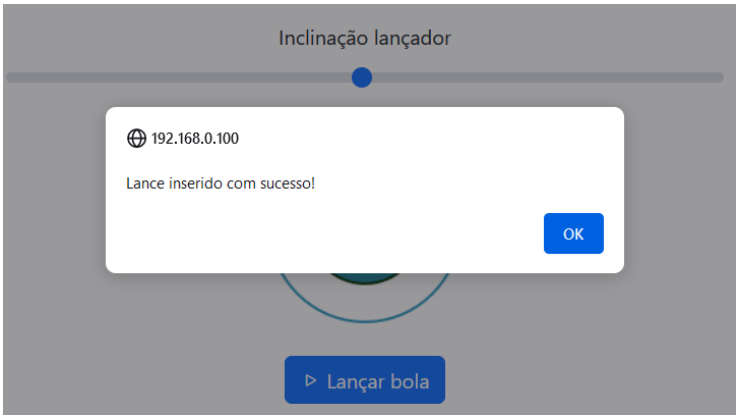


Figura 28 - Mensagem do tipo "alert" que indica se foi adicionado o lance com sucesso

6.2.1.2 *Template* “Menu de configurações”

Outro *template* disponível na aplicação é o "menu de configurações", que só pode ser acedido após efetuar a autenticação na página. Este menu foi desenvolvido com o propósito de permitir instalar a aplicação em diferentes lançadores de bolas, possibilitando um comissionamento rápido sem a necessidade de realizar alterações e compilação de código no comissionamento. Esta abordagem torna a aplicação do lançador de bolas facilmente portável para outros sistemas físicos, sem a exigência de compreender o código-fonte subjacente.

A Figura 29 ilustra a página que é exibida ao aceder o menu de configurações, nela é possivel observar que existe dois submenus, “Configs JSON” e o “Configs ESP32-GRBL”.

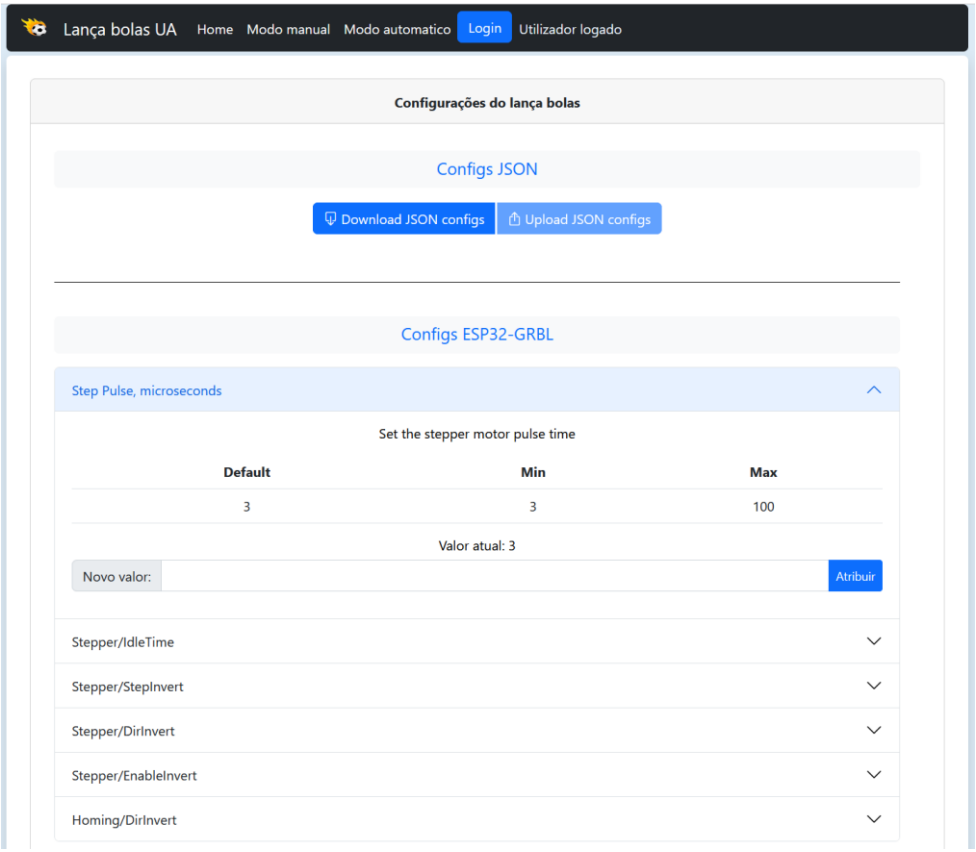


Figura 29 - Página inicial do template de configurações do lança bolas

No submenu “Configs JSON” existem dois botões disponíveis para baixar um arquivo JSON (Figura 30) de configuração que está armazenado no *backend*, e transportá-lo para o *frontend* em forma de formulário, conforme ilustrado na Figura 31, o segundo botão fica acessível após o download do JSON e é usado para enviar as alterações de volta para o *backend*. É importante ressaltar que o formulário é gerado dinamicamente, e uma vez que as configurações são enviadas para a aplicação, tornam-se efetivas durante a execução, e em tempo real. Significa, portanto, que as alterações feitas no formulário têm impacto imediato no funcionamento do sistema de controlo do lançador de bolas.

```json
{
  "configs_lb": {
    "maximo": {
      "X": 25,
      "Y": 25,
      "Z": 25,
      "A": 15
    },
    "angulo": {
      "min_X": -45,
      "max_X": 45,
      "min_Y": -3,
      "max_Y": 20,
      "min_Z": -20,
      "max_Z": 18,
      "min_A": 0,
      "max_A": 90
    },
    "graus_desl_a": {
      "retemBola": 15,
      "lancaBola": 45
    },
    "velocidadeAvancoGate": "6000",
    "velocidadeZeroMaquina": 3000
  }
}
```

Figura 30 - Ficheiro JSON que dá origem ao formulário

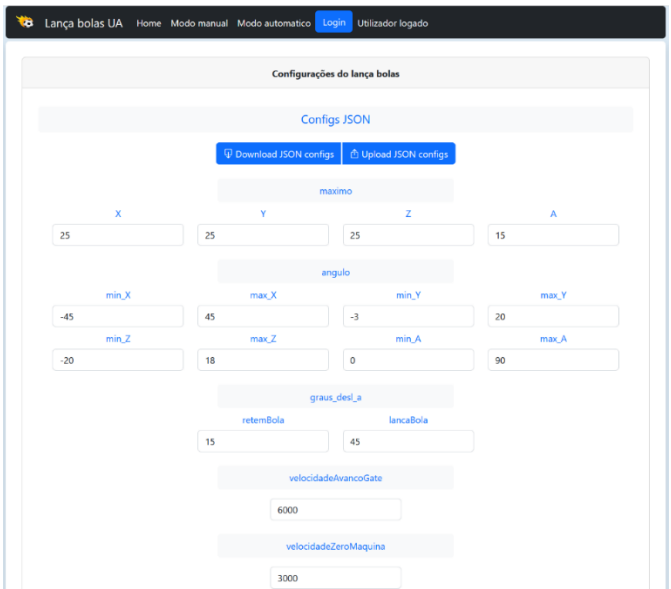


Figura 31 - Formulário JSON gerado em tempo de execução através das configurações do ficheiro JSON.

O segundo submenu denominado "Config ESP32-GRBL", é um menu escalável onde podem ser adicionados menus de configurações através da base de dados. Para configurar este submenu, é recomendado consultar a documentação disponível do GRBL [https://github.com/bdring/Grbl_Esp32/wiki/Settings].É importante destacar que o GRBL possui uma ampla variedade de parâmetros, com mais de 100 opções, mas nem todos eles são relevantes para um controlador de bolas. Alguns parâmetros são específicos para máquinas CNC que envolvem ferramentas de corte e movimentos complexos. Portanto, é necessário identificar os parâmetros adequados para o controlador de bolas em questão.

O mesmo princípio de responsividade aplicado aos outros *templates* é também válido para este como ilustra a Figura 32. Assim é possível que as configurações possam ser efetuadas com *smartphone*, *tablet* ou *desktop*, o que permite configurar o sistema independentemente do dispositivo utilizado.

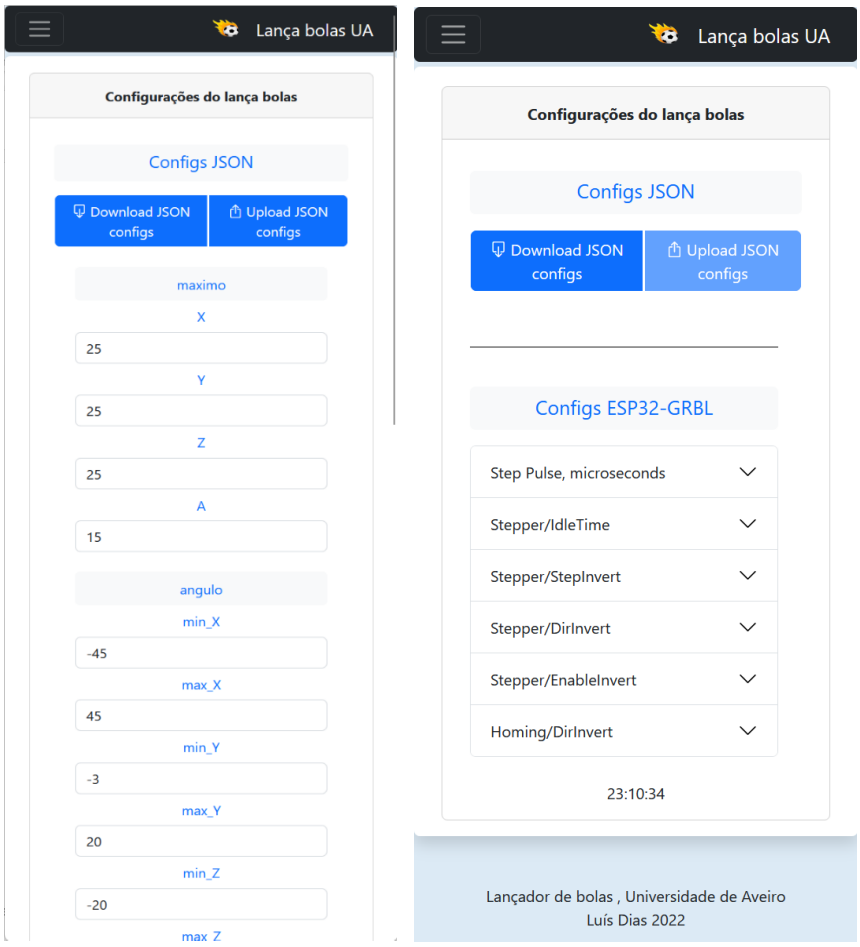


Figura 32 - Template "Menu de configurações" em versão web mobile totalmente responsivo

6.2.1.3 *Template* "Modo Automático"

Outra opção disponível é o *template* "Modo Automático", que oferece acesso aos lances e treinos previamente armazenados na base de dados, juntamente com as ferramentas necessárias para executá-los.

Este menu (conforme ilustrado na Figura 33) possui duas abas que permitem alternar entre o modo de execução.

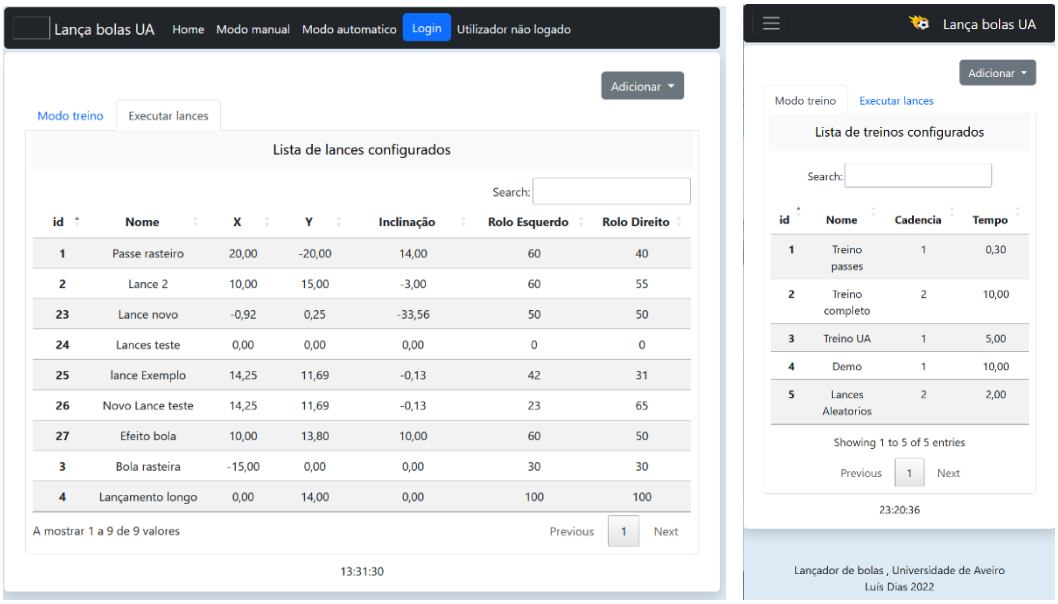


Figura 33 - Template "Modo automático" na versão desktop e web mobile.

Em ambas as abas, é exibida uma tabela responsiva que apresenta a lista de lances e treinos armazenados na base de dados, com o respetivo id do lance. A tabela possui recursos avançados, como a capacidade de ordenar os campos com um único clique no cabeçalho da coluna, uma ferramenta de pesquisa que permite que o utilizador localize rapidamente lances e treinos específicos e os exiba no máximo em 10 linhas de treinos ou lances. Caso haja mais linhas, o utilizador pode navegar entre as tabelas usando um menu localizado no canto inferior direito da página. Estes recursos visam facilitar a consulta e visualização dos dados armazenados no sistema.

Neste *template* os utilizadores têm a capacidade de executar lances e treinos. Ao clicarem na linha correspondente ao lance ou treino desejado, um modal é aberto, estabelecendo uma comunicação AJAX que por sua vez recebe da base de dados a informação do componente a executar. Esta interação através de JavaScript

permite a configuração do treino ou lance a ser executado (Figura 34). Após a configuração dinâmica do *modal* o utilizador terá acesso a informações relevantes e botões de controlo para iniciar os ciclos.

No *modal* (Janela sobreposta que bloqueia a principal) de execução do lance, o utilizador deve inserir a cadência do lance e a quantidade de bolas a serem executadas antes de poder executá-lo. Após fornecer os dados, o utilizador pode dar início à execução do lance.

Já no *modal* de execução de treino, as informações provenientes da base de dados são suficientes para dar início ao treino. Portanto, não é necessário inserir nenhuma informação adicional antes de iniciar o treino.

Os menus foram projetados com o objetivo de oferecer uma interface limpa, intuitiva e responsiva para operar máquina, o que permite aos utilizadores poderem facilmente aceder às funcionalidades desejadas e configurar os ciclos automáticos de treino de acordo com suas preferências.

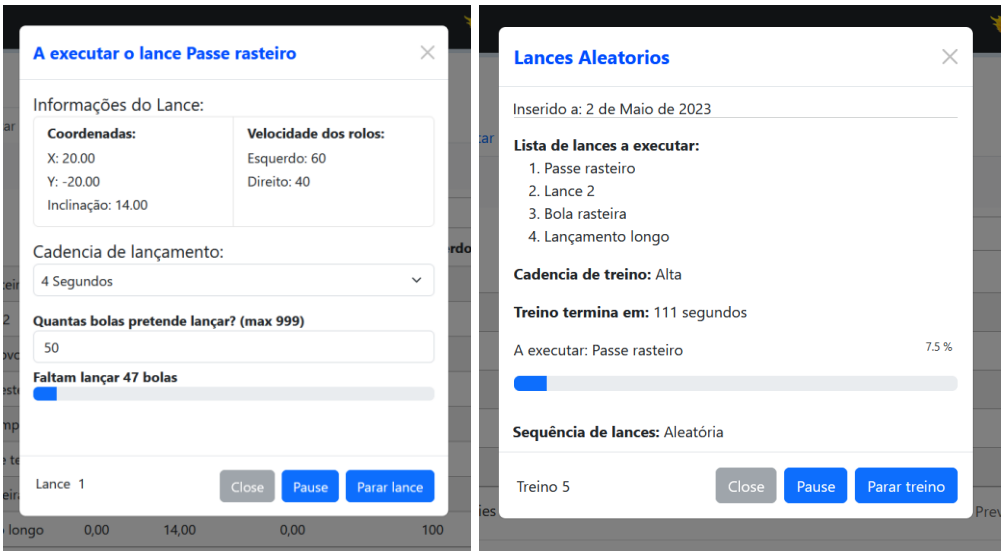


Figura 34 - Menu modal para controlo e execução de treinos e lances.

Ainda no menu de “Modo Automático”, existe no canto superior direito um botão do tipo "*dropdown*" que permite adicionar novos lances ou treinos. Ao clicar na inserção um submenu do tipo *modal* é exibido (Figura 35), apresentando um formulário que possibilita o preenchimento dos dados necessários para a inserção na base de dados. Estes formulários facilitam a adição de novos registos.

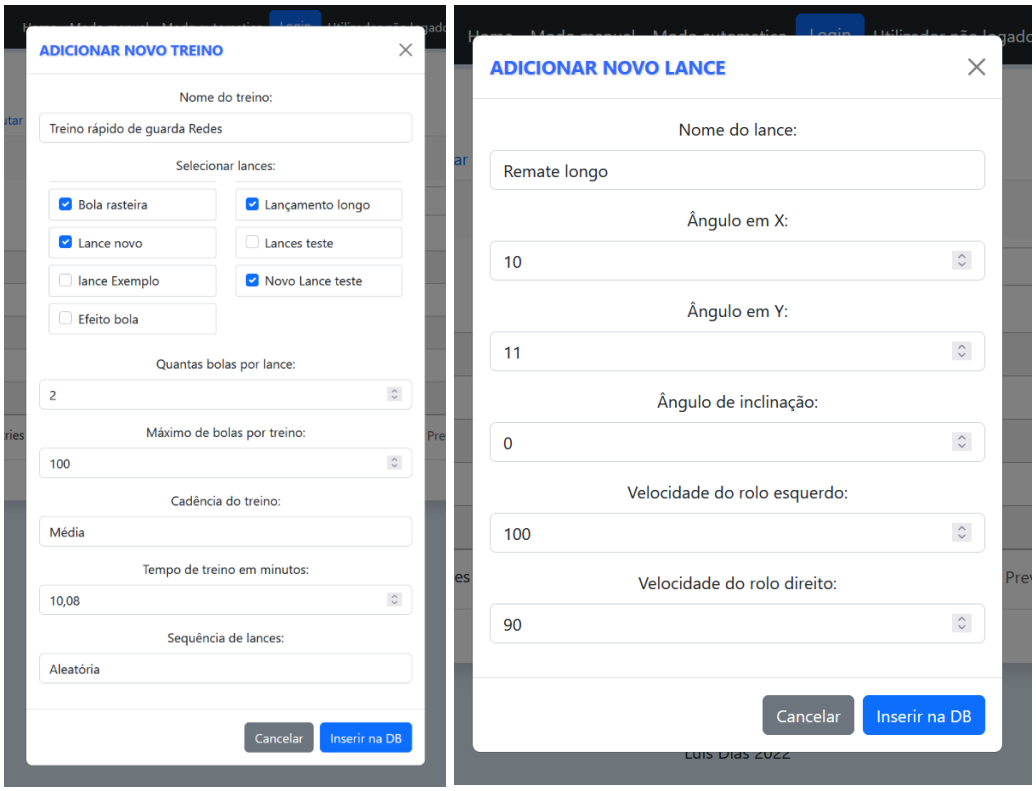


Figura 35 - Menus do tipo modal para inserção de novo lance ou novo treino na base de dados do lançador de bolas.

Por último existe um menu de administração que controla a acesso aos utilizadores e também permite aceder, alterar e apagar lances ou treinos da base de dados. Quando um utilizador carrega num dos botões de login disponíveis da aplicação web é redirecionada para a página de autenticação (Figura 36).

Figura 36 - Página de autenticação personalizada.

Quando é efetuado o login com sucesso o utilizador é redirecionado para a página de administração (Figura 37). Esta página, que também é responsiva, tem um menu do lado direito com o histórico de ações recentes para possibilitar a visualização do histórico de interações com a base de dados. Têm também um menu do lado esquerdo que permite todas as operações com a base de dados dos treinos e lances, e um menu que permite configurar e ajustar as opções relacionadas ao GRBL. Estas configurações podem ser facilmente acedidas e modificadas por meio de um menu dinâmico acessível no *template* de configurações.

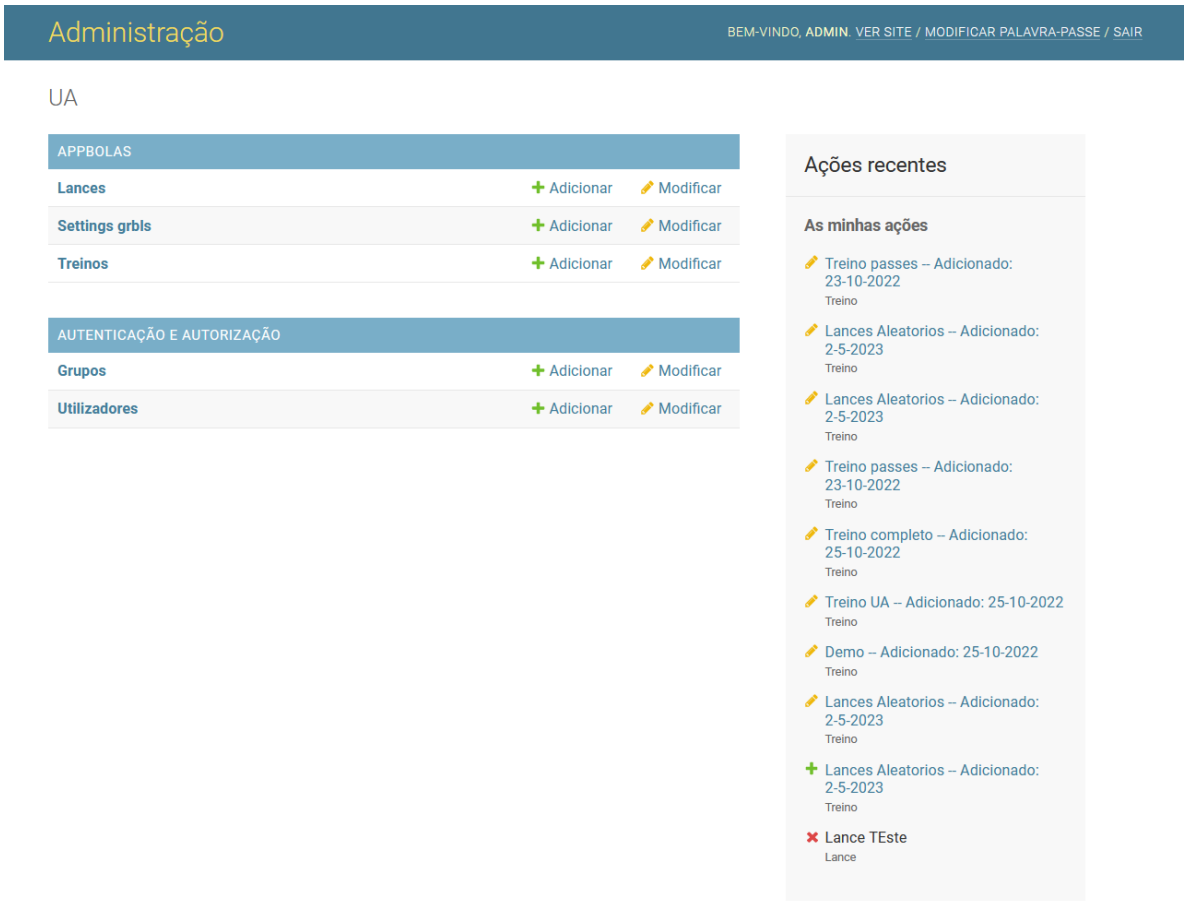


Figura 37 - Página de administração do lançador de bolas

6.3 Base de dados

No projeto, foi incluída uma base de dados para melhorar a experiência do utilizador. O objetivo é permitir que o utilizador possa planear, armazenar e aceder a lances individuais, bem como criar treinos com base nesses lances. Para isso, os parâmetros de lançamento são armazenados na base de dados do controlador e disponibilizados para acesso através da aplicação.

Para atingir estes objetivos, foram desenvolvidas duas bases de dados distintas, uma para treinos e outra para lances individuais. Estas bases de dados contêm todas as informações necessárias para que a aplicação possa reproduzir os lances ou conjuntos de lances.

Além disso, foi implementada uma terceira tabela na base de dados, que permite a criação dinâmica de menus que contêm configurações específicas para o controlador GRBL. A seguinte tabela representa a estrutura criada na base de dados para alojar a informação.

Tabela 6 - Estrutura da base de dados do lançador de bolas

Base de dados		
Lances	Treinos	SettingsGRBL
Id: int	**nomeTreino**: CharField	**Título**: CharField
anguloX: DecimalField	**Lances**: ManyToManyField(lance)	**textoInformação**: CharField
anguloY: DecimalField	Qt_**bolas**_lance: IntegerField	**valorDefault**: IntegerField
AnguloInclinacao: DecimalField	**maxBolasTreino**: IntegerField	**valorMin**: IntegerField
velocidadeRoloEsq: SmallIntegerField	**Cadencia**: IntegerChoices	**valorMax**: IntegerField
velocidadeRoloDir: SmallIntegerField	**cadenciaTreino**: IntegerField	**comandoGRBL**: CharField
	dataCriacao: DateTimeField	
	tempoTreino: DecimalField	
	Seqlances: IntegerChoices	
	SequenciaLances: IntegerField	

Trabalhar com bases de dados no Django é uma tarefa intuitiva devido à estrutura *models*. Os *models* do Django fornecem métodos que permitem ler e escrever dados, sem a necessidade de executar uma *Query* SQL diretamente na base de dados. Com esta estrutura do Django, o acesso aos dados é obtido com o retorno de objetos que contêm campos correspondentes à informação correspondente na base de dados, e métodos para consultar os dados de forma estruturada.

A base de dados padrão fornecida pelo Django é o SQLite, que já está pronta para ser utilizada no instanciamento da aplicação. Como o SQLite oferece todas as funcionalidades necessárias para este projeto, não houve a necessidade de migrar para outro sistema de base de dados.

6.4 Configurações esp32/GRBL

A máquina é controlada utilizando um ESP32 com o *firmware* GRBL, que foi personalizado para atender às especificações do lançador de bolas.

É importante destacar que existem três tipos de alterações/configurações possíveis no GRBL:

1º - Alterações no código fonte do GRBL;

2º - Configuração das opções de compilação do GRBL;

3º - Configurações do GRBL através de comunicação RS232, que por sua vez são armazenadas na EPROM.

O GRBL já contém uma estrutura que permite a adaptação do algoritmo de controlo a diferentes situações. No entanto, houve uma limitação na versão original do GRBL, que suporta apenas o controlo dos motores através de PWM (*Pulse Width Modulation*). Essa limitação não atendia aos requisitos da máquina, que exigia o controlo de modulação por frequência. Por esse motivo, foi necessário modificar código fonte do GRBL, que se traduz na alteração de configurações nos três níveis possíveis de personalização do GRBL.

As principais modificações no GRBL foram realizadas nas configurações de compilação. Essas configurações abrangem a definição dos pinos de entrada e saída, a quantidade de eixos e a ativação das duas saídas analógicas para o controlo dos motores dos rolos.

Após o envio do *firmware* para o controlador ESP32, as configurações que podem ser reescritas e armazenadas na EPROM. Na aplicação do lançador de bolas estas configurações estão acessíveis através do *template* de configurações, nesse menu

é possível inserir, consoante as necessidades, uma lista detalhada de comandos para serem alterados em tempo de execução, como descrito na secção anterior.

6.4.1 Alterações ao GRBL

Dentro do diretório do GRBL, encontra-se uma pasta chamada "*Machines*", onde estão armazenadas as configurações para diferentes tipos de máquinas. Nessa pasta, foi criada uma configuração específica para o lançador de bolas. O arquivo de configuração utilizado é o "3axis_v4.h", no qual foi possível personalizar e pinos responsáveis pelo controlo dos inputs e dos outputs do lançador de bolas. A Tabela 7 apresenta as configurações efetuadas no ficheiro.

Tabela 7 - Mapa de Inputs/Outputs definido no GRBL para controlo do lançador de bolas.

#define MACHINE_NAME	"Lanca bolas UA"
#define STEPPERS_DISABLE_PIN	GPIO_NUM_13
#define DEFAULT_INVERT_ST_ENABLE	0
#define X_STEP_PIN	GPIO_NUM_12
#define X_DIRECTION_PIN	GPIO_NUM_14
#define Y_STEP_PIN	GPIO_NUM_26
#define Y_DIRECTION_PIN	GPIO_NUM_15
#define Z_STEP_PIN	GPIO_NUM_27
#define Z_DIRECTION_PIN	GPIO_NUM_33
#define A_STEP_PIN	GPIO_NUM_25
#define A_DIRECTION_PIN	GPIO_NUM_32
#define X_LIMIT_PIN	GPIO_NUM_17
#define Y_LIMIT_PIN	GPIO_NUM_4
#define Z_LIMIT_PIN	GPIO_NUM_16
#define A_LIMIT_PIN	GPIO_NUM_19
#define SPINDLE_TYPE	SpindleType::PWM
#define SPINDLE_ENABLE_PIN	GPIO_NUM_22
#define USER_ANALOG_PIN_0	GPIO_NUM_5
#define USER_ANALOG_PIN_1	GPIO_NUM_18

Além das configurações de compilação, outra área de modificação importante no GRBL foi o código fonte. Com o objetivo de habilitar o controlo da velocidade dos

motores dos rolos por frequência, foram necessárias alterações significativas e a criação de novas funções.

A primeira etapa desse processo envolveu a modificação do arquivo GCode.cpp, responsável pela interpretação e execução do código G. Nesse arquivo, foi necessário criar novas funções e redirecionar funções existentes de forma a incorporar a lógica de controlo da velocidade dos motores. A Figura 38 exibe uma árvore de decisão em forma de estrutura de controlo "*case*", na qual são armazenados os valores de velocidade compreendidos entre 0 e 100%. Estes valores são posteriormente utilizados como argumento para a função de envio de comandos aos motores.

```
case 'P':
    axis_word_bit     = GCodeWord::P;
    gc_block.values.p = value;
    break;
case 'Q':
    axis_word_bit     = GCodeWord::Q;
    //Alteração para o APP_BOLAS, de duty para Freq, caso estaja definido USER_PWM_OUTPUT
    gc_block.values.q = value;
    //Visualizar quanto estou a receber aqui
    //grbl_msg_sendf(CLIENT_SERIAL, MsgLevel::Info, "Q %2.2f", gc_block.values.q);
    break;
case 'R':
    axis_word_bit     = GCodeWord::R;
    gc_block.values.r = value;
    break;
case 'S':
    axis_word_bit     = GCodeWord::S;
    gc_block.values.s = value;
```

Figura 38 - Valor recebido do comando GRBL M67 E(X) Q(X)

Após decifrar as mensagens de gcode existe funções que executam o controlo da máquina segundo os inputs, na Figura 39 podemos observar que a tomada de decisão para o controlo por PWM ou por frequência, depende na definição de #USER_PWM_OUTPUT.

Esta constante quando definida passa a poder receber um valor de “gc_block.values.q” que antes era de 0 a 100 % de PWM para um valor que pode variar entre entre 0 a 9600 Hz. Apos o valor de “q” ser normalizado, passa por outra árvore de decisão que permite chamar a função “sys_set_freq(uint8_t io_num, float freq)”. Está função recebe o parametro “e”, que corresponde ao motor acionado e o valor de velocidade “q” (PWM ou Frequência).

```
if ((gc_block.modal.io_control == IoControl::SetAnalogSync) || (gc_block.modal.io_control == IoControl::SetAnalogImmediate)) {
    if (gc_block.values.e < MaxUserDigitalPin) {

        // Transforma esta variavel num valor valido de 0 a 100 APP_BOLAS
        #ifdef USER_PWM_OUTPUT
            gc_block.values.q = constrain(gc_block.values.q, 0.0, 20000.0);  // force into valid range
        #else
            gc_block.values.q = constrain(gc_block.values.q, 0.0, 100.0);  // force into valid range
        #endif

        if (gc_block.modal.io_control == IoControl::SetAnalogSync) {
            protocol_buffer_synchronize();
        }

        //if (!sys_set_analog((int)gc_block.values.e, gc_block.values.q)) {
        //    FAIL(Error::PParamMaxExceeded);
        //}

        // APP_BOLAS ATUALIZAÇÃO da FUNçÃO

        //APP_BOLAS
        #ifdef USER_PWM_OUTPUT
            if (!sys_set_freq((int)gc_block.values.e, gc_block.values.q)) {
            FAIL(Error::PParamMaxExceeded);
        }
        #else
            if (!sys_set_analog((int)gc_block.values.e, gc_block.values.q)) {
            FAIL(Error::PParamMaxExceeded);
        }
        #endif


    } else {
        FAIL(Error::PParamMaxExceeded);
    }
}
```

Figura 39 - Alteração da função de chamada de execução do comando M67

A nova função “sys_set_freq(uint8_t io_num, float freq)”, foi referenciada no ficheiro “system.h” e definida no “system.cpp”, está função retorna o canal PWM disponivel no firmware para controlo, com myAnalogOutputs[io_num] e executa a função para definir a frequência de controlo (Figura 40).

```
298     // io_num is the virtual analog pin#
299     bool sys_set_analog(uint8_t io_num, float percent) {
300         auto     analog    = myAnalogOutputs[io_num];
301         uint32_t numerator = percent / 100.0 * analog->denominator();
302         return analog->set_level(numerator);
303     }
304
305     // io_num is the virtual analog pin#            //APP_BOLAS
306     bool sys_set_freq(uint8_t io_num, float freq) {
307         auto     analog    = myAnalogOutputs[io_num];
308         //uint32_t numerator = freq / 100.0 * analog->denominator();
309         return analog->set_freq(freq);
310     }
311
```

Figura 40 - Funções de controlo por PWM e frequência

Por fim é criada uma nova função que atualiza o valor no ESP32 denominada “set_freq(float frequencia)” (Figura 41), referenciada em “UserOutputs.h” e definida em “userOutputs.cpp”. Esta implementação atualiza a frequência do canal PWM e coloca o (Duty Cicle) a 50%.

```
100     // Função criada para o APP_BOLAS, altera a frequencia, frequencia maxima 9600 hz
101     bool AnalogOutput::set_freq(float frequencia) {
102         // look for errors, but ignore if turning off to prevent mask turn off from generating errors
103         if (_pin == UNDEFINED_PIN) {
104             return false;
105         }
106 
107         if (_pwm_channel == -1) {
108             grbl_msg_sendf(CLIENT_SERIAL, MsgLevel::Info, "M67 PWM channel error");
109             return false;
110         }
111 
112         if (_pwm_frequency == frequencia) {
113             return true;
114         }
115 
116         _pwm_frequency = frequencia;
117         ledcSetup(_pwm_channel, _pwm_frequency, _resolution_bits);
118         ledcAttachPin(_pin, _pwm_channel);
119         uint32_t duty = 50 / 100.0 * denominator();
120         ledcWrite(_pwm_channel, duty);
121 
122 
123         return true;
124     }
```

Figura 41 - Função de envio da frequência de controlo para os canais PWM do esp32

Estas alterações foram cuidadosamente planeadas e implementadas de forma cirúrgica, evitando quaisquer possíveis efeitos indesejados no software GRBL que é conhecido por ser robusto e passar por inumeros anos de desenvolvimento e testes.

É possível concluir que, com a possibilidade de ter acesso ao código fonte, é possivel criar funções que não estavam originalmente projetadas, o que proporciona uma flexibilidade significativa durante o processo de desenvolvimento.

Capítulo 7

7. Conclusões e trabalhos futuros

Após esta longa jornada foi possível concluir que a aplicação está totalmente funcional. Possui uma base de dados robusta que permite a configuração do lançador através da aplicação web, bem como rotinas de lançamento e uma estrutura escalável para futuros desenvolvimentos.

As várias tecnologias que foram utilizadas no âmbito deste projeto foram fundamentais para desenvolver um projeto desta complexidade, a combinação delas possibilitou uma integração rápida dos componentes da aplicação web, resultando em um controlo aprimorado e personalizável do lançador de bolas. Foi possível elevar a qualidade da solução final e reduzir o tempo necessário de desenvolvimento, tudo isto devido à utilização de funções e menus funcionais amplamente testados.

Uma das conclusões principais que foram retiradas deste projeto, foi a necessidade de inserir um microcontrolador dedicado ao controlo dos motores. No início do projeto foi iniciado o desenvolvimento de uma biblioteca para controlo dos motores de passo, mas logo foi abortada esta abordagem. Foi constatado que com o *raspberry pi* não é possível ter controlo total do tempo em que os sinais são enviados para os drivers. Este facto deve-se ao tipo de arquitetura de controlo dos pinos auxiliares de saída do *raspberry pi 4*, os comandos de ativação passam sempre pelo *Kernel* do *linux*, o que leva a entrarem em concorrência direta com outros processos em execução no sistema operativo. Quando uma aplicação é executada e requer mais processamento é notório que os sinais de comando começam a perder os tempos de pulso, que se traduz num controlo defeituoso e mau funcionamento dos motores.

Ao longo do projeto, utilizou-se jQuery para interações com o DOM e chamadas assíncronas. A meio do projeto, após algumas pesquisas, verificou-se que o jQuery não é mais necessário, e está a perder terreno para recentes atualizações do *JavaScript*, funções essas que no início era de uma extrema complexidade implementá-las com *JavaScript*, e agora a própria linguagem já incorpora métodos para uma simples implementação.

Uma vez concluída a montagem do lançador de bolas, é possível fazer a transição da maquete de simulação para o lançador de bolas projetado. O software foi desenvolvido de forma a permitir uma fácil adaptação a qualquer lançador de bolas, através da edição de parâmetros acessíveis na aplicação web.

Existem vários trabalhos futuros que podem ser implementadas para melhorar a experiência do utilizador e elevar ainda mais o nível desta máquina. Algumas delas incluem:

- Implementação de uma câmara de vídeo que permita através de visão computacional, calcular a posição do lançador no campo em relação à baliza, possibilitando corrigir os ângulos dos lances;
- Interação com o utilizador por meio da câmara de vídeo, possibilitando o posicionamento dos motores com apenas um toque no ecrã que indicaria para onde o utilizador pretendia lançar a bola;
- Adição de um recurso de localização GPS para obter informações de geolocalização do lançador de bolas;
- Posicionamento no campo por meio de triangulação com as bandeiras de canto, utilizando uma webcam;
- Desenvolvimento de um sistema automático de monitorização e carregamento das baterias;
- Implementação de um modo de lançamento e posicionamento automático com visão computacional e inteligência artificial, permitindo ao lançador detetar os movimentos do guarda-redes e lançar a bola para o ponto mais distante, elevando o nível do treino;
- Transporte da aplicação web para aplicativos nativos de Android e iOS.

Durante o desenvolvimento, foi identificado um defeito em replicar treinos sem que o utilizador tenha o cuidado de posicionar o lançador na mesma direção e distância em relação à baliza. Como objetivo futuro, seria interessante incorporar um sistema de localização no lançador, permitindo que ele se posicione automaticamente no campo e faça compensações nas coordenadas de lançamento.

8. Referências

[1] Nunes, Diogo. (2021, 26 de dezembro), Desenvolvimento de um sistema de monitorização e controlo para máquinas de lançamento de bolas de futebol (Tese de Mestrado), 21,44,45.

[2] Almamedical (2022, 03 de março) https://www.almamedical.es/en/productos/globus-eurogoal-600-ball-shooting-machine-soccer.html

[3] Goldpet (2022, 03 de março) Lança bolas https://goldpet.pt/bolas/11520-lanca-bolas-65cm-4018653845714.html

[4] Ertheo. (2022, 3 de Janeiro) How a Tennis Ball Machine Could Take Your
[5] Game to the Next Level. https://www.ertheo.com/blog/en/tennis-ball-machine/

[6] AELS (2022, 25 de fevereiro) Exemplo de um lançador de bolas de ténis pneumático http://blog.kotarak.net/2012/04/pneumatic-antenna-launchers-for-sale.html

[7] TOPGIM (2022, 10 de Abrir) Venda de equipamento para auxílio no desporto https://topgim.com/2060-maquina-lancadora-de-bolas-para-futebol-eurogoal-600/

[8] AJAX (2023, 20 de janeiro) Asynchronous Javascript and XML https://pt.wikipedia.org/wiki/Ajax_(programa%C3%A7%C3%A3o)

[9] Long Polling (2023, 20 Janeiro) Long polling vs Regular polling with ajax https://javascript.info/long-polling

[10] WebSockets (2023, 25 janeiro) Websocket: O que é e para que serve https://www.novageo.pt/novageo/displayArticles?numero=38362&protocolo_websocket_que__para_que_serve

[11] WebSockets (2023, 25 de janeiro) Bidirectional communication technology for web apps https://caniuse.com/websockets

[12] HTTPS (2023, 2 de fevereiro) Hyper Text Transfer Protocol Secure https://pt.wikipedia.org/wiki/Hyper_Text_Transfer_Protocol_Secure

[13] WSS (2023, 3 de fevereiro) WebSockets Living Standard https://websockets.spec.whatwg.org/

[14] ESP32 (2023, 5 de fevereiro) Imagem ESP32 WROOM https://iztuts.com/lap-trinh-cho-esp32-wroom/

[15] BLDC CONTROLER (2023, 20 de março) Imagem de controlador BLDC https://www.botnroll.com/en/controllers/3975-380w-3-phase-bldc-brushless-motor-controller-pwm-plc-driver-module-dc-6-50v.html

[16] História do Futebol em Portugal (2022, 10 de dezembro) https://portugalstore.fpf.pt/pt/editorial/historia-do-futebol-em-portugal-portugal-store-fpf

[17] Tecnologia e o estado da arte do futebol (2022, 12 de dezembro) https://medium.com/@LuizAlbertoRodrigues/tecnologia-e-o-estado-da-arte-do-futebol-4f7018071b54

[18] CAFÉ, Lucas, Futebol, Poder e Política. Bahia: 2010

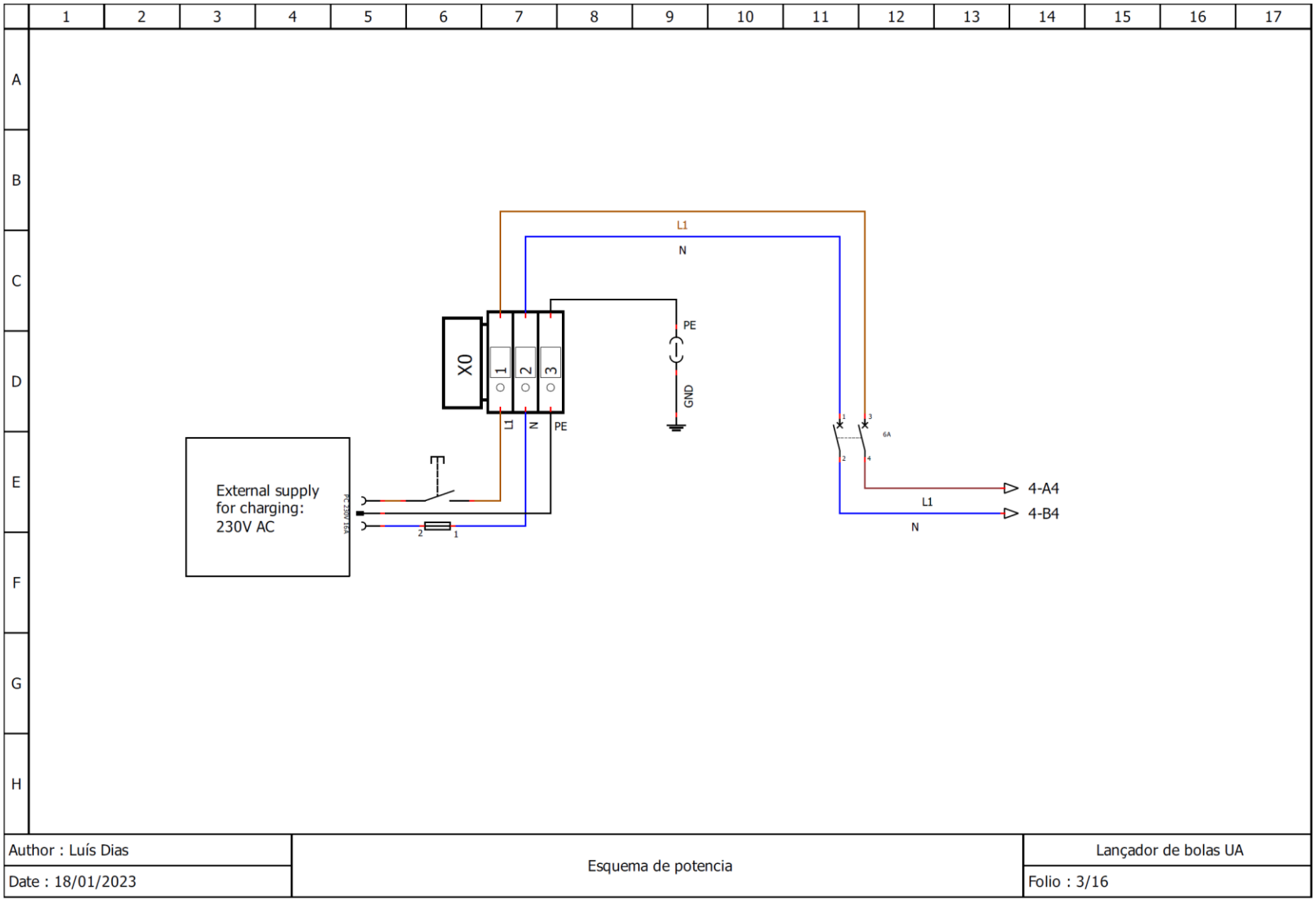
9.Anexos

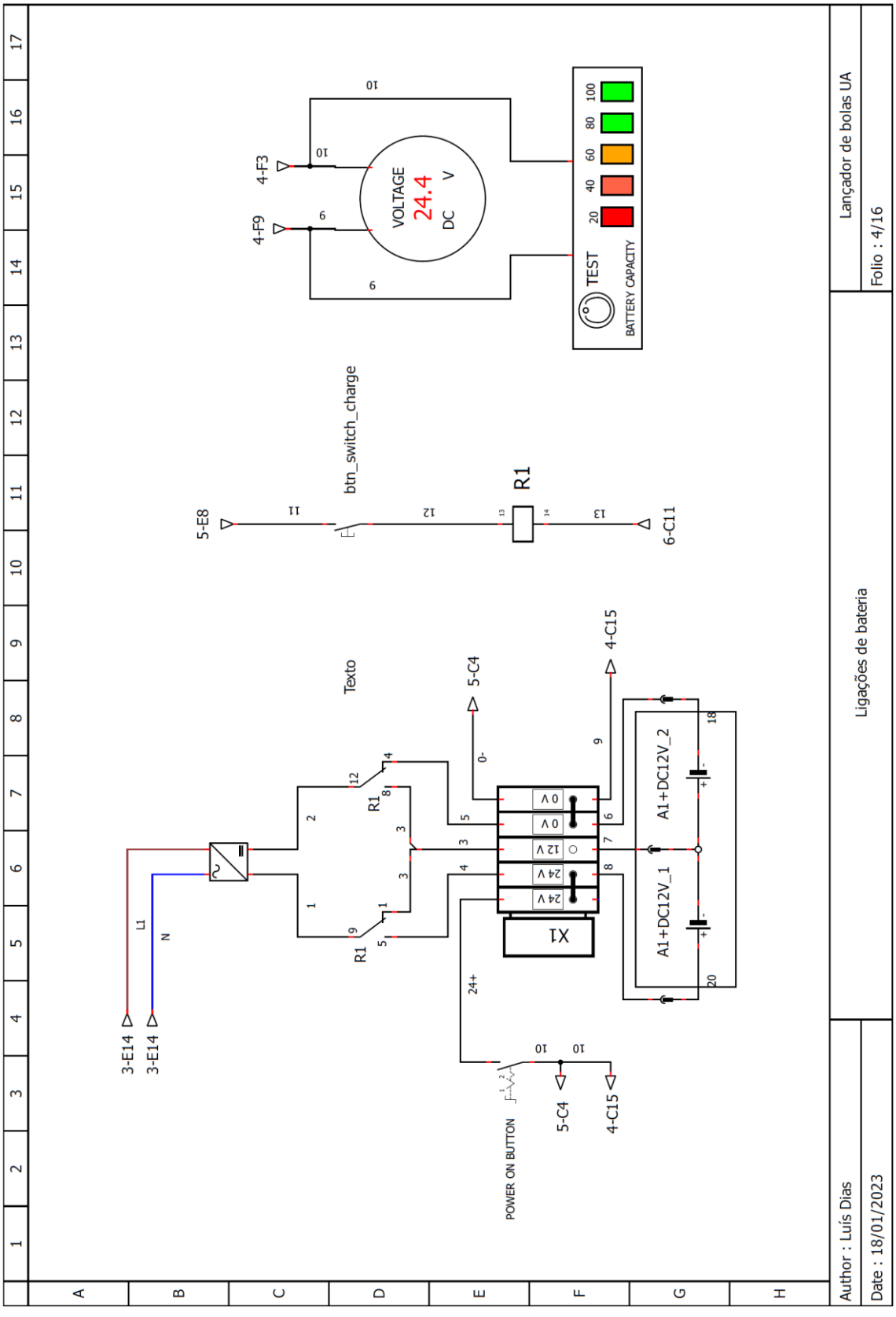
9.1 Esquema elétrico

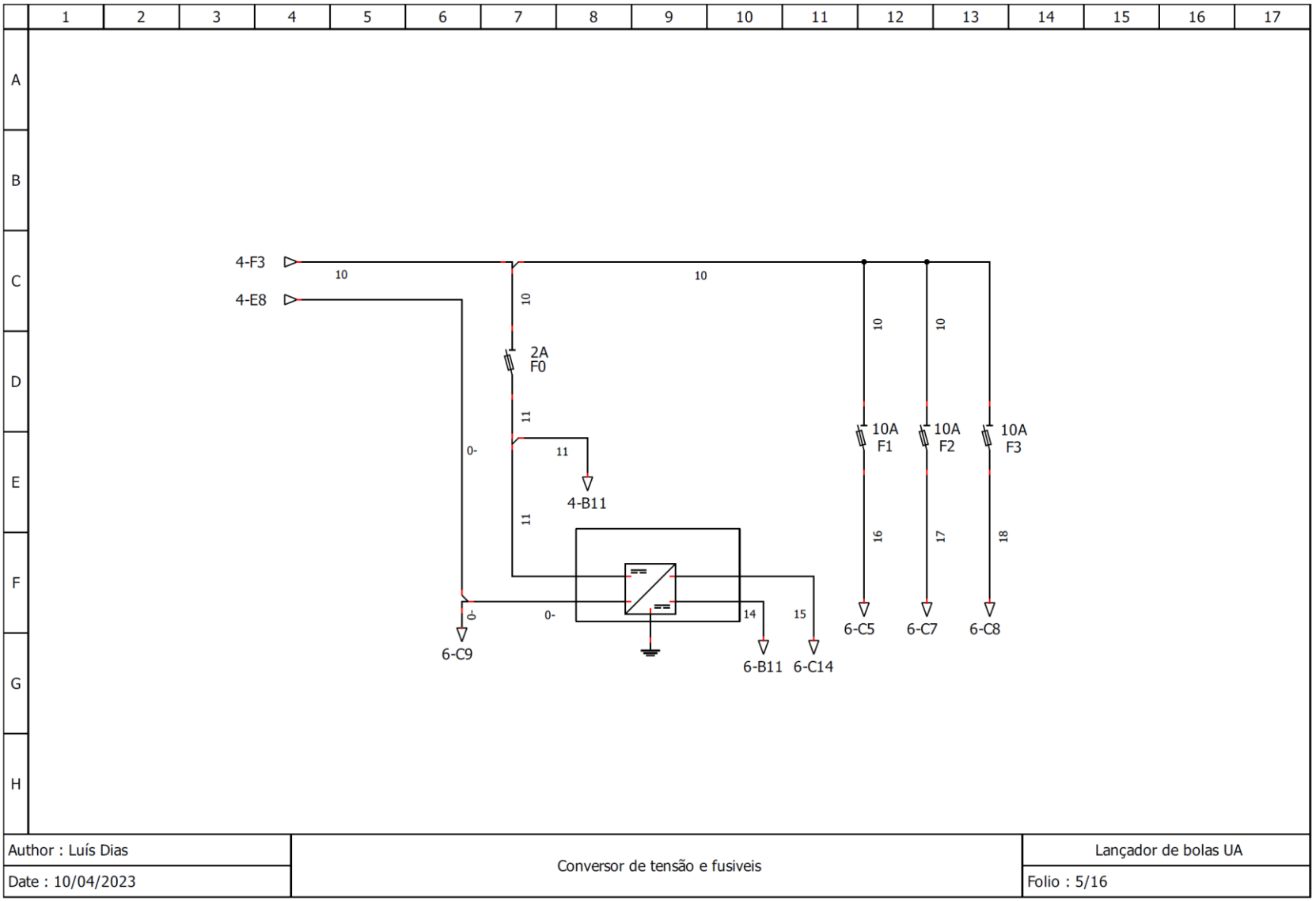
CORES UTILIZADAS PARA OS CABOS

TENSÃO	DESIGNAÇÃO	COR
220 VAC	Fase	Castanho
	Neutro	Azul
	PE	Verde/amarelo
24 V	Positivo	Vermelho
	Negativo	Preto
5 V	Positivo	Orange
	Negativo	Branco
3.3 V	Positivo	Blank
	Negativo	Branco

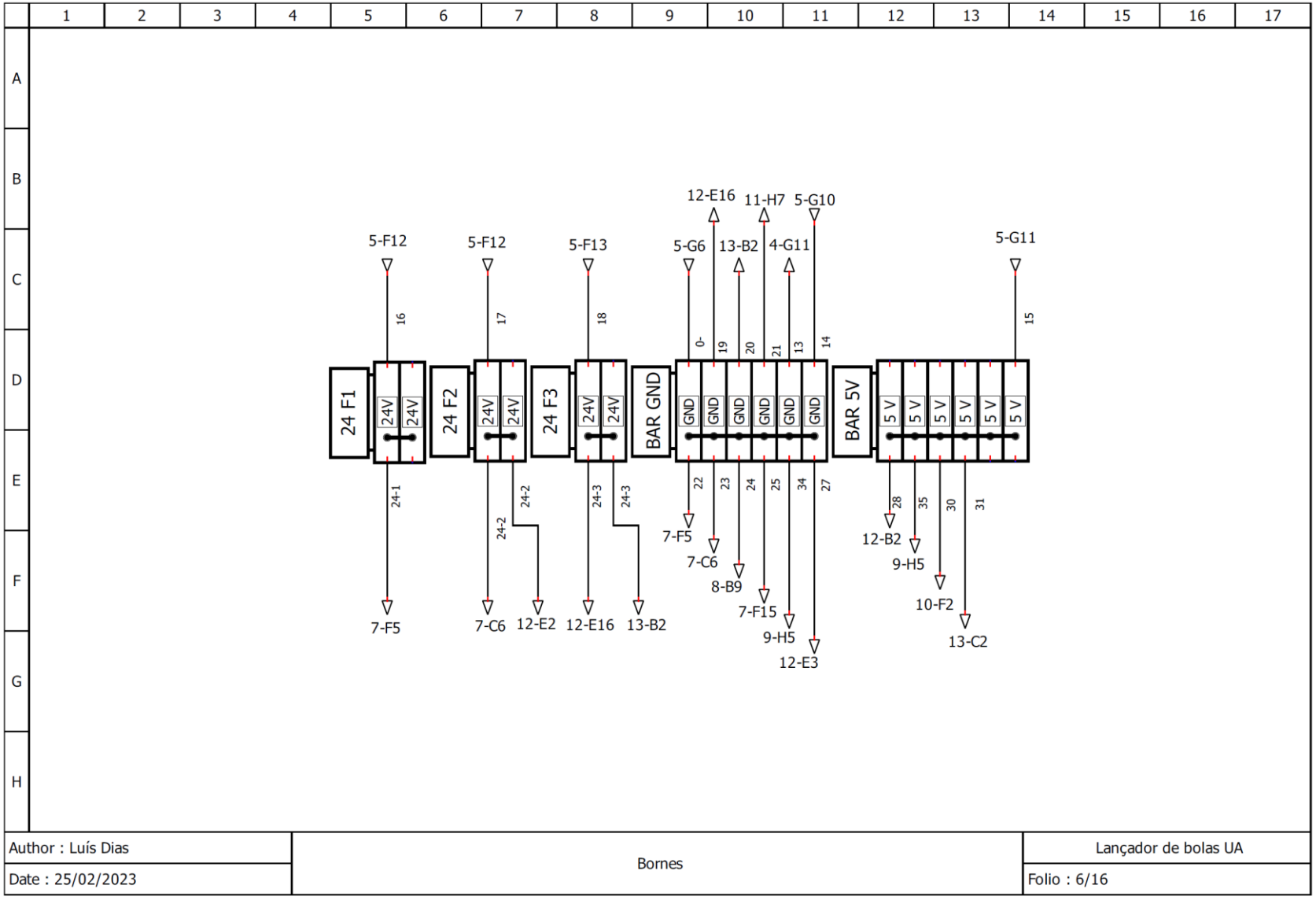
Author : Luís Dias	Cores dos cabos	Lançador de bolas UA
Date : 25/03/2023		Folio : 2/16


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
A B C D E F G H
L1
N
PE
X0
1
2
3
GND
L1
N
PE
External supply
for charging:
230V AC
PC 230V 16A
F
2
1
6A
4-A4
4-B4
L1
N
Author : Luís Dias
Date : 18/01/2023
Esquema de potencia
Lançador de bolas UA
Folio : 3/16

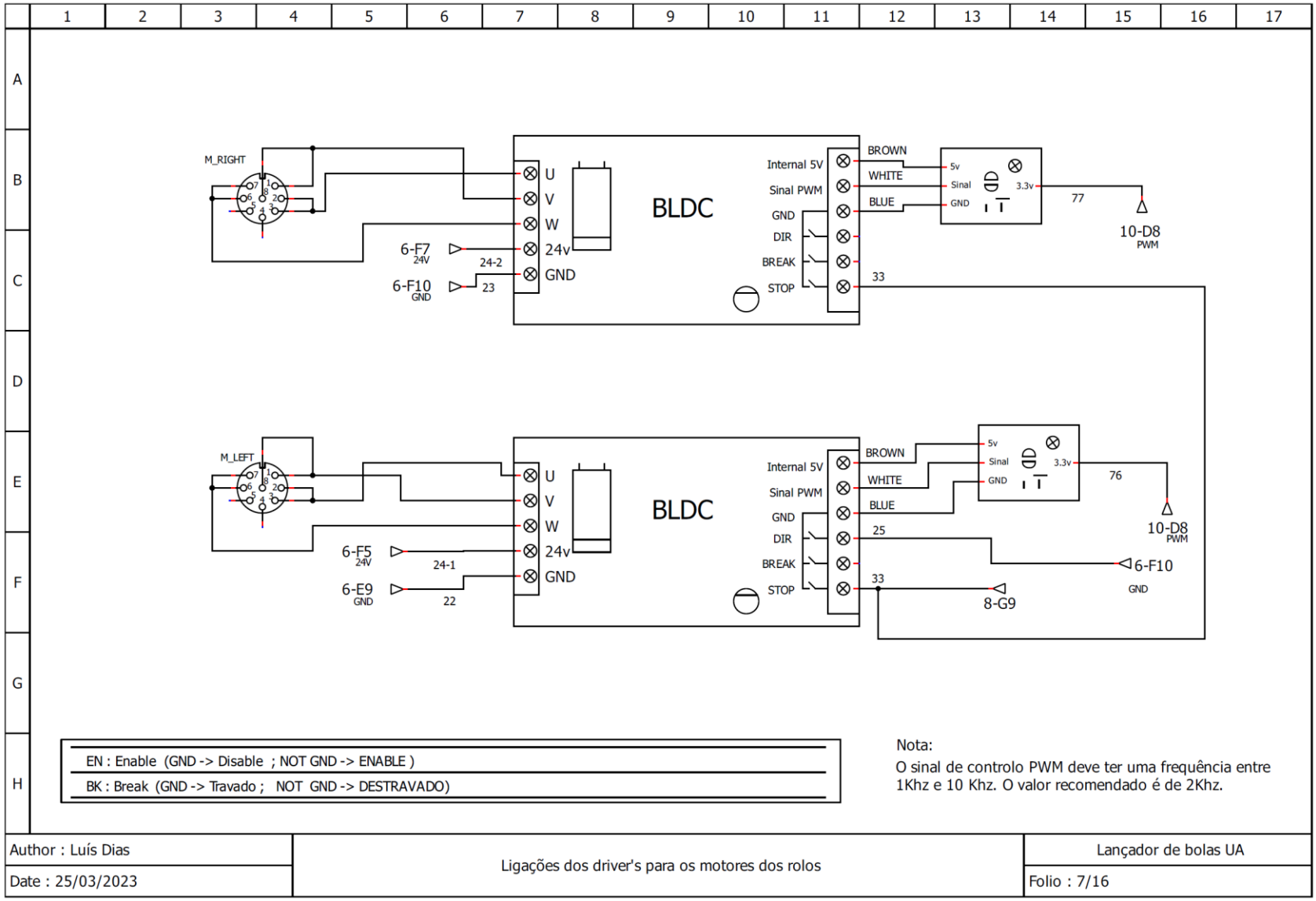

Lançador de bolas UA
Folio : 4/16
Ligações de bateria
Author : Luís Dias
Date : 18/01/2023
4-F3
4-F9
VOLTAGE
24.4
V
DC
TEST
BATTERY CAPACITY
100
80
60
40
20
btn_switch_charge
R1
5-E8
6-C11
Texto
5-C4
4-C15
0-
24+
X1
0 V
12 V
24 V
A1+DC12V_1
A1+DC12V_2
3-E14
L1
N
POWER ON BUTTON



Author : Luís Dias	Conversor de tensão e fusiveis	Lançador de bolas UA
Date : 10/04/2023		Folio : 5/16



Author : Luís Dias	Bornes	Lançador de bolas UA
Date : 25/02/2023		Folio : 6/16

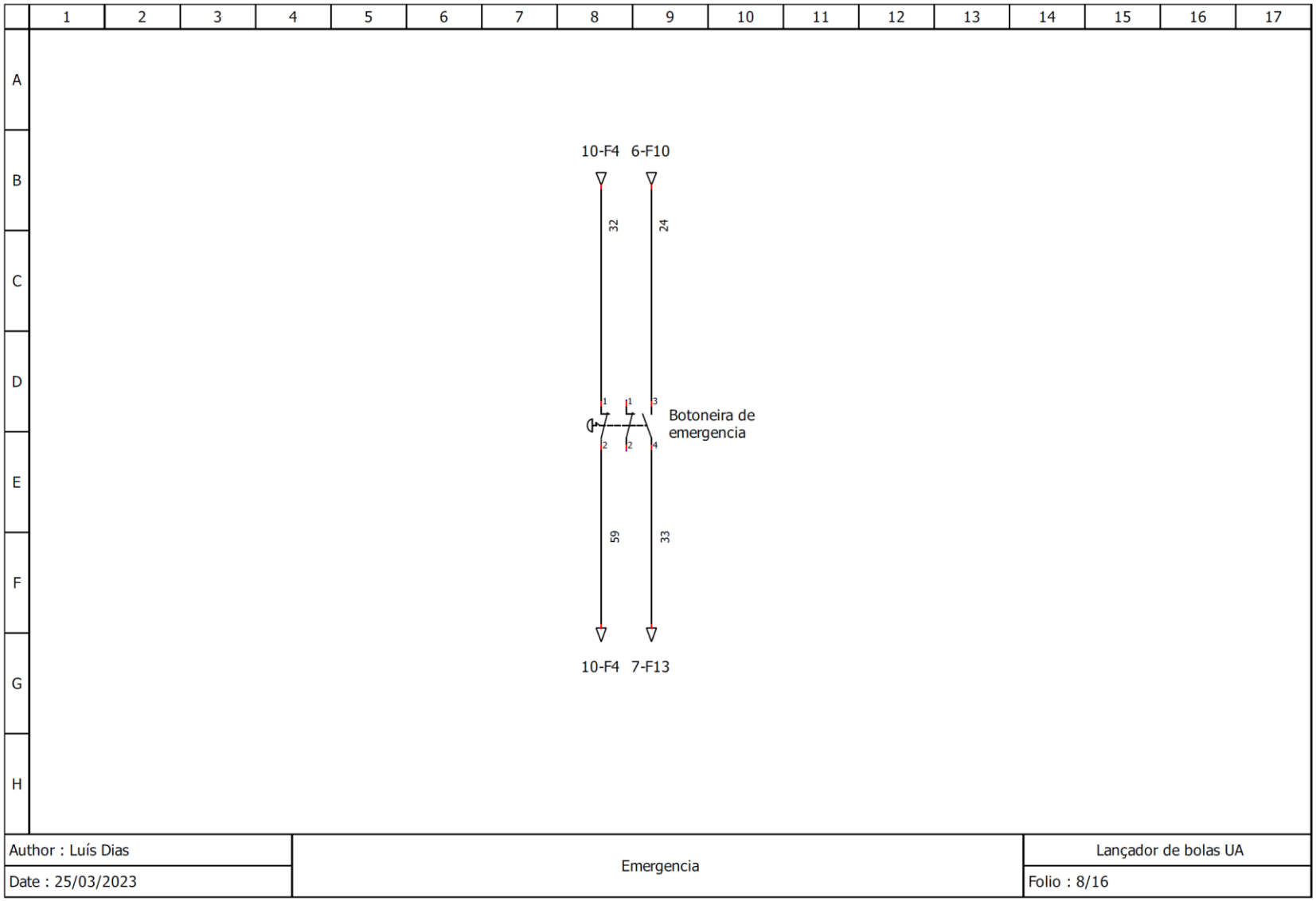


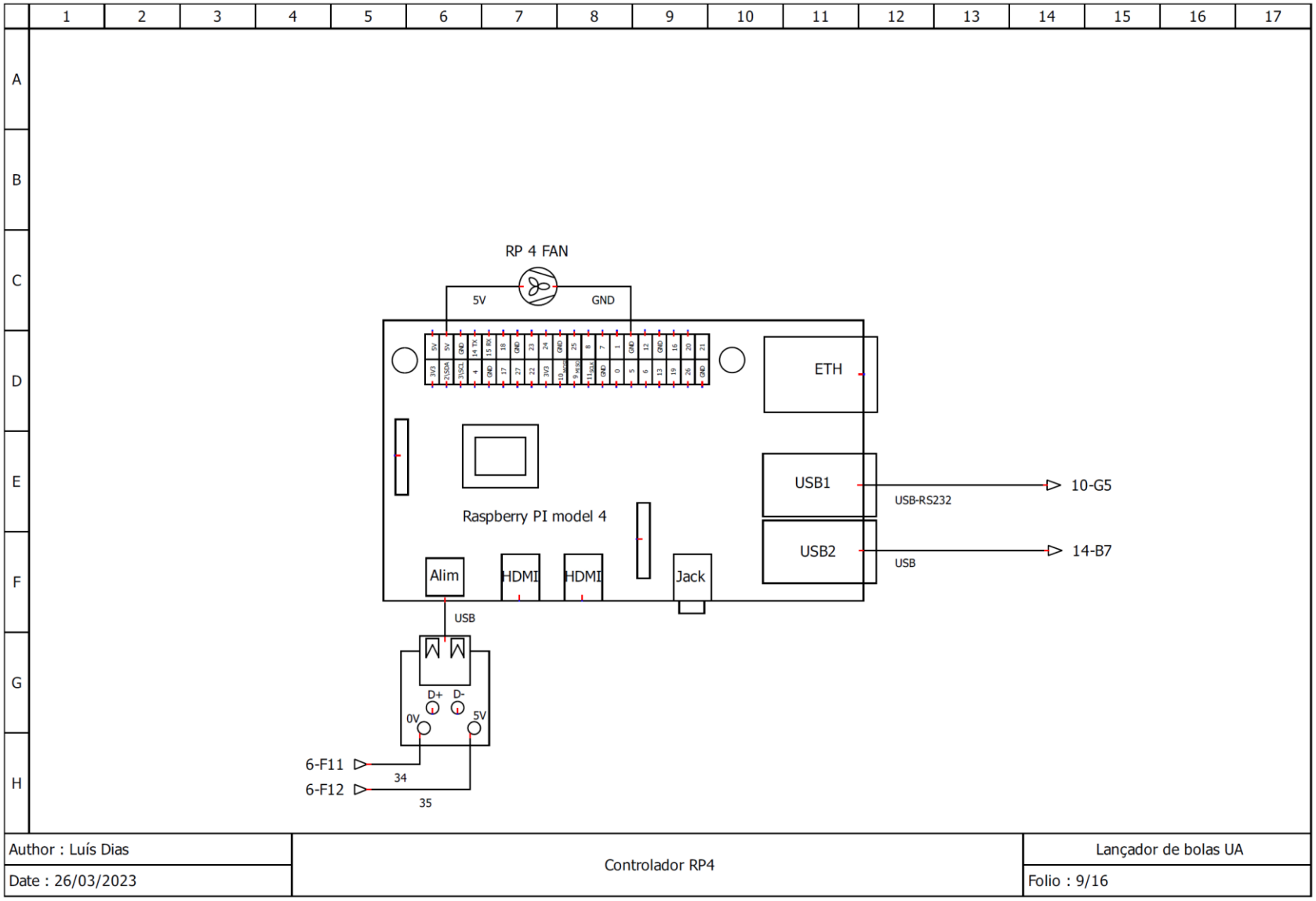
EN : Enable (GND -> Disable ; NOT GND -> ENABLE)

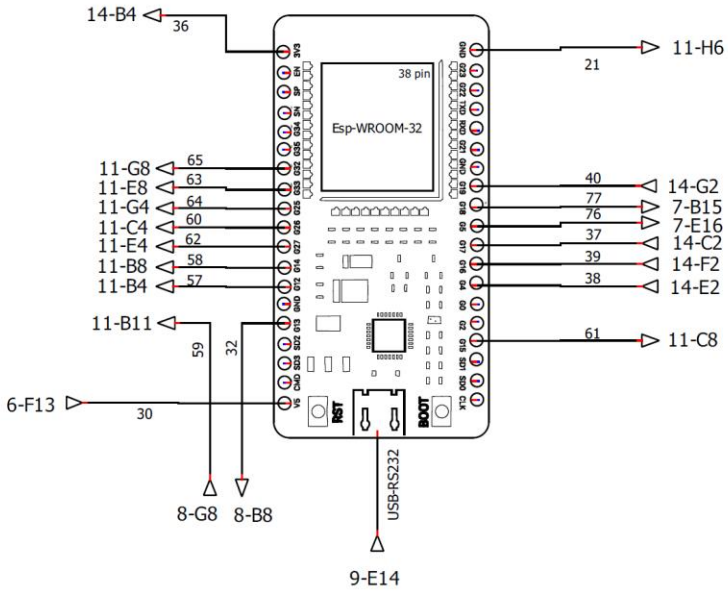
BK : Break (GND -> Travado ; NOT GND -> DESTRAVADO)

Nota:
O sinal de controlo PWM deve ter uma frequência entre 1Khz e 10 Khz. O valor recomendado é de 2Khz.

Author : Luís Dias	Ligações dos driver's para os motores dos rolos	Lançador de bolas UA
Date : 25/03/2023		Folio : 7/16

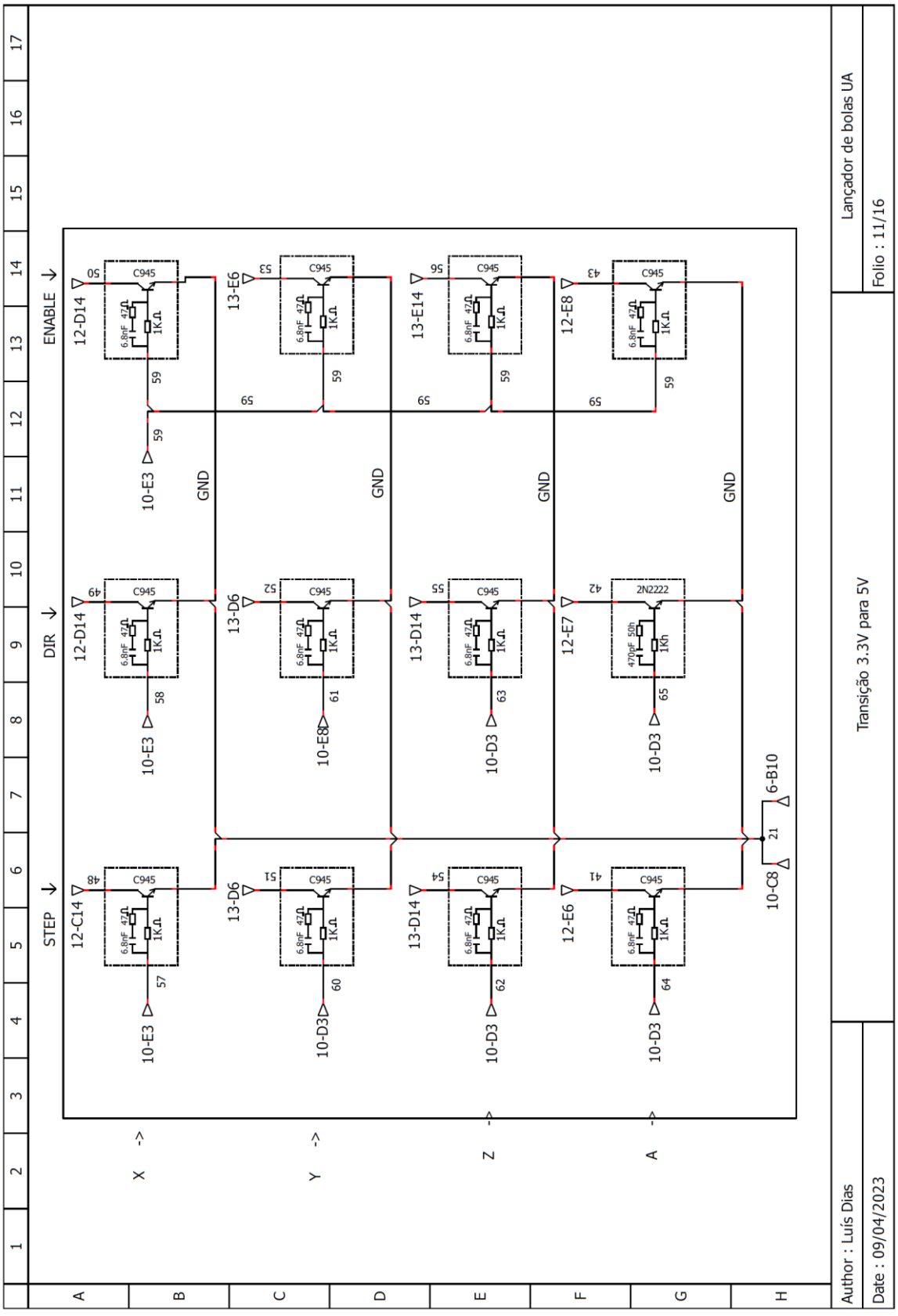

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
A B C D E F G H
10-F4 6-F10
32
24
1 1 3
Botoneira de emergencia
2 2 4
59
33
10-F4 7-F13
Author : Luís Dias
Date : 25/03/2023
Emergencia
Lançador de bolas UA
Folio : 8/16

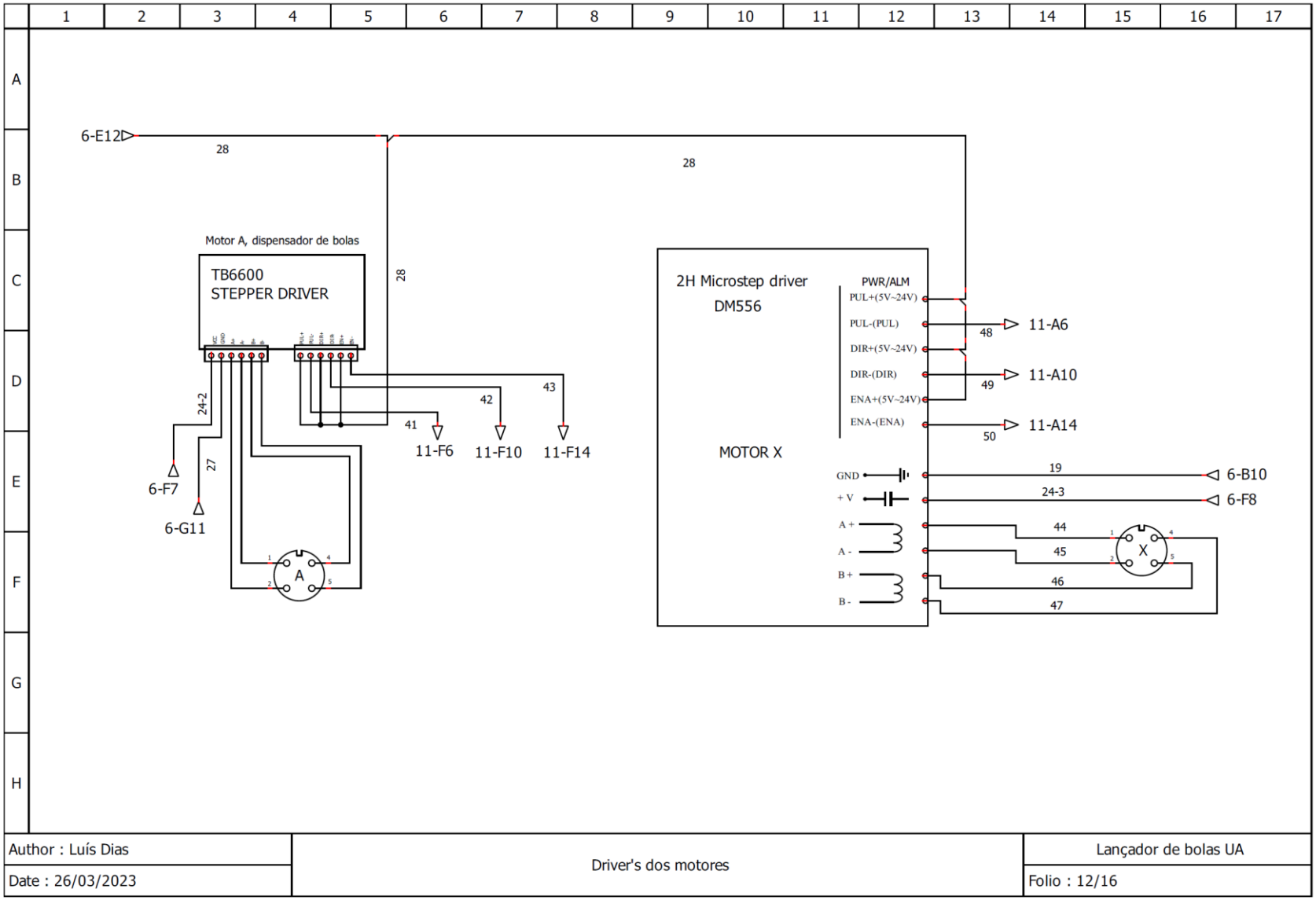

RP 4 FAN
5V
GND
ETH
USB1
10-G5
USB-RS232
Raspberry PI model 4
USB2
14-B7
USB
Alim
HDMI
HDMI
Jack
USB
D+
D-
0V
5V
6-F11
34
6-F12
35
Author : Luís Dias
Date : 26/03/2023
Controlador RP4
Lançador de bolas UA
Folio : 9/16



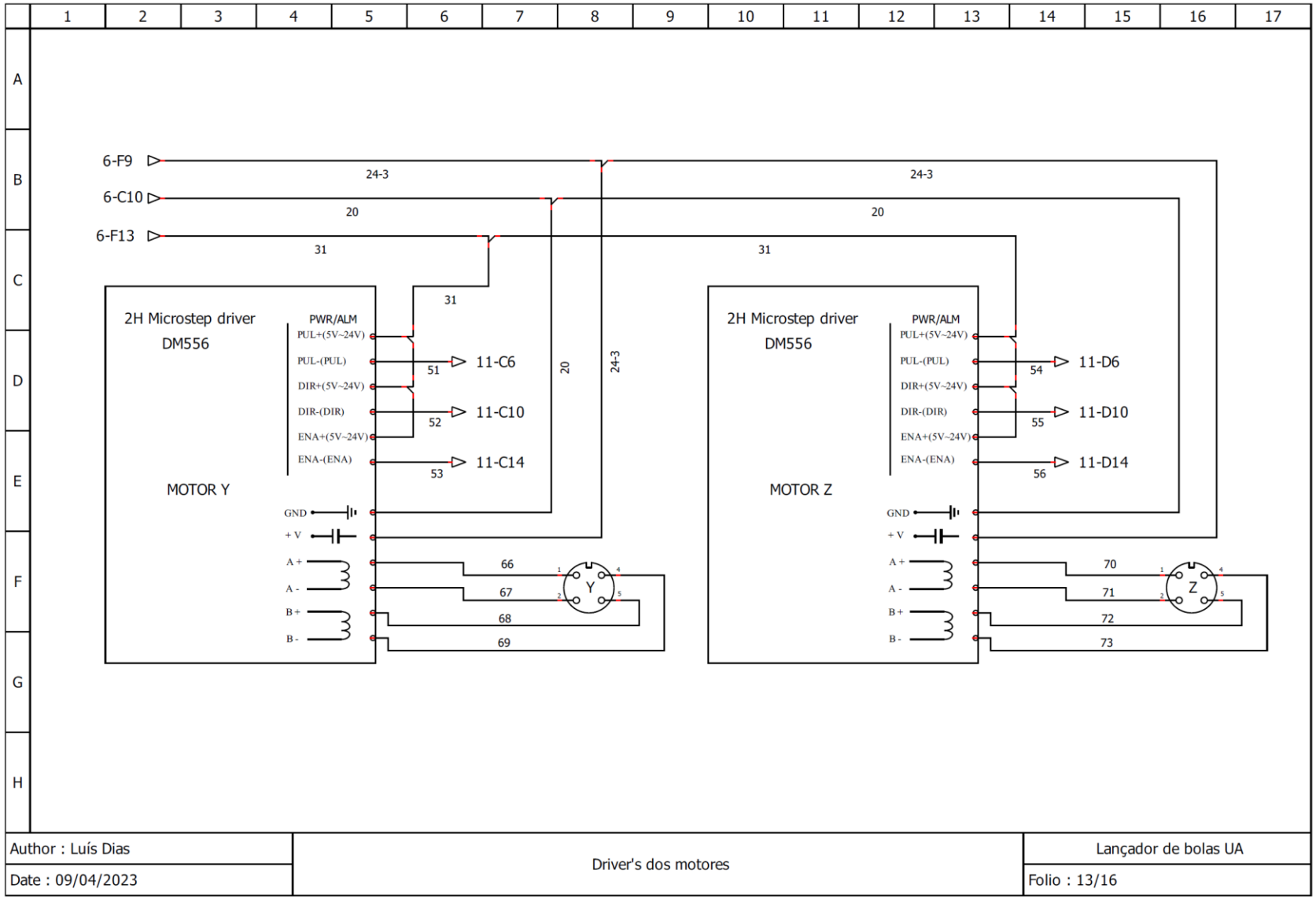

```
#define MACHINE_NAME                "Lanca bolas UA"
#define STEPPERS_DISABLE_PIN         GPIO_NUM_13
#define DEFAULT_INVERT_ST_ENABLE     0
#define X_STEP_PIN                   GPIO_NUM_12
#define X_DIRECTION_PIN              GPIO_NUM_14
#define Y_STEP_PIN                   GPIO_NUM_26
#define Y_DIRECTION_PIN              GPIO_NUM_15
#define Z_STEP_PIN                   GPIO_NUM_27
#define Z_DIRECTION_PIN              GPIO_NUM_33
#define A_STEP_PIN                   GPIO_NUM_25
#define A_DIRECTION_PIN              GPIO_NUM_32
#define X_LIMIT_PIN                  GPIO_NUM_17
#define Y_LIMIT_PIN                  GPIO_NUM_4
#define Z_LIMIT_PIN                  GPIO_NUM_16
#define A_LIMIT_PIN                  GPIO_NUM_19
#define SPINDLE_TYPE                 SpindleType::PWM
#define SPINDLE_ENABLE_PIN           GPIO_NUM_22
#define USER_ANALOG_PIN_0            GPIO_NUM_5
#define USER_ANALOG_PIN_1            GPIO_NUM_18
```

Author : Luís Dias	Controlador NC	Lançador de bolas UA
Date : 26/03/2023		Folio : 10/16

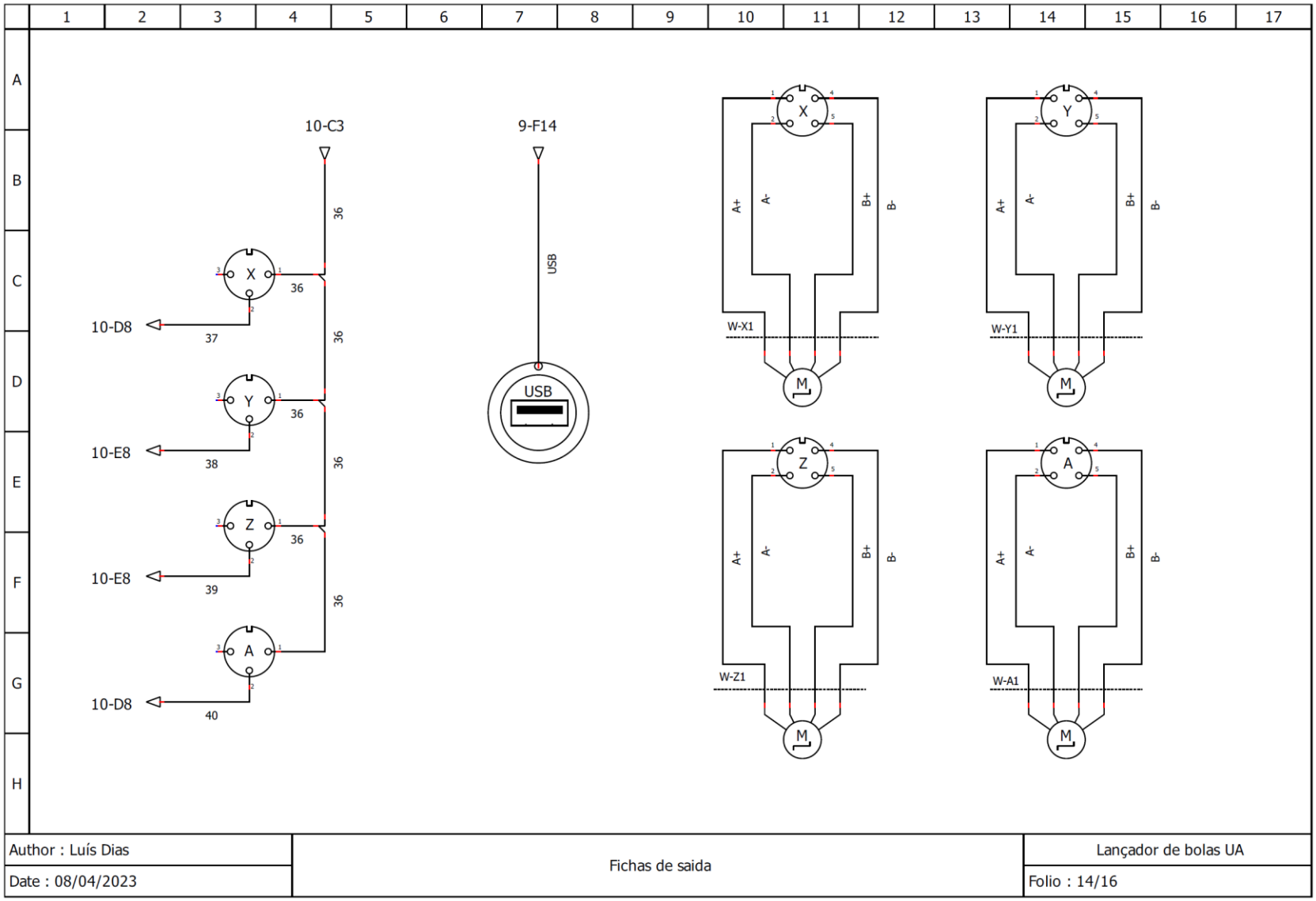

Lançador de bolas UA
Folio : 11/16
Transição 3.3V para 5V
Author : Luís Dias
Date : 09/04/2023
STEP
DIR
ENABLE
X ->
Y ->
Z ->
A ->
GND
C945
2N2222
6.8nF
47Ω
1KΩ
470nF
50h
1Kn
12-C14
13-D6
12-D14
13-D14
12-E6
13-E6
13-E14
12-E7
12-E8
10-E3
10-D3
10-E8
10-C8
6-B10

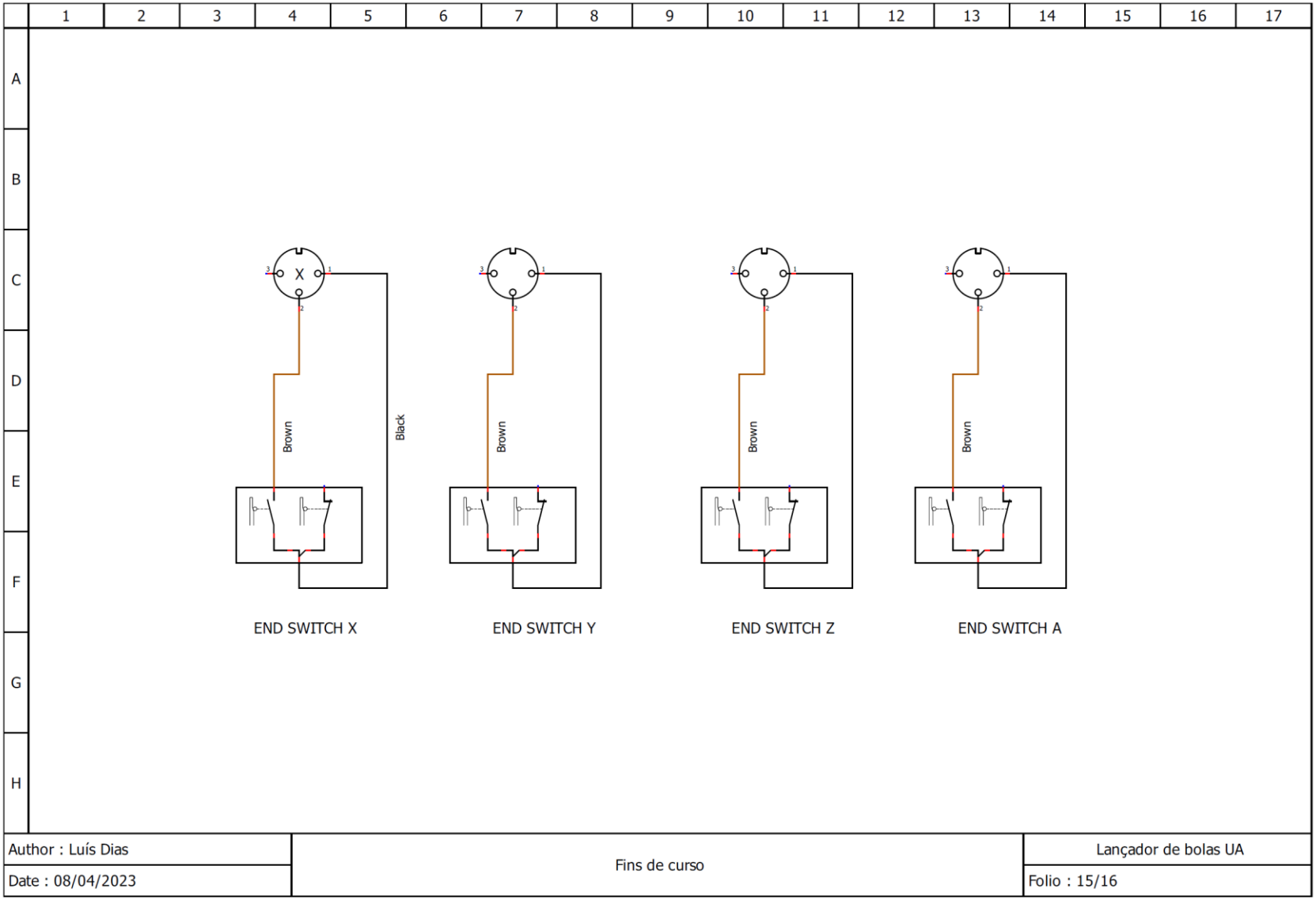


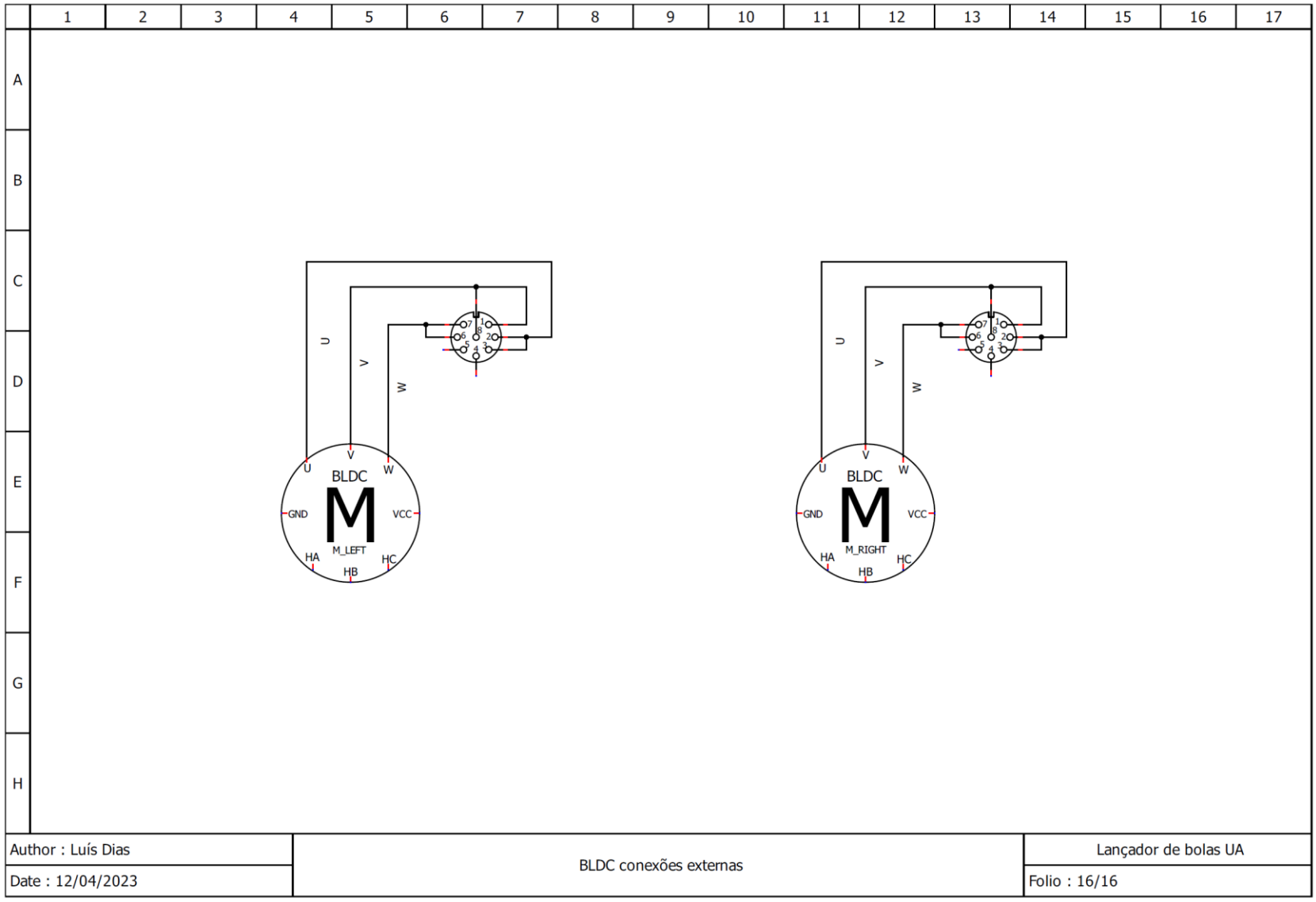
Author : Luís Dias	Driver's dos motores	Lançador de bolas UA
Date : 26/03/2023		Folio : 12/16



Author : Luís Dias	Driver's dos motores	Lançador de bolas UA
Date : 09/04/2023		Folio : 13/16


10-C3
9-F14
36
X
36
10-D8
37
Y
36
10-E8
38
Z
36
10-E8
39
A
36
10-D8
40
USB
USB
X
A+
A-
B+
B-
W-X1
M
Y
A+
A-
B+
B-
W-Y1
M
Z
A+
A-
B+
B-
W-Z1
M
A
A+
A-
B+
B-
W-A1
M
Author : Luís Dias
Date : 08/04/2023
Fichas de saida
Lançador de bolas UA
Folio : 14/16


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
A B C D E F G H
X
Brown
Black
Brown
Brown
Brown
END SWITCH X
END SWITCH Y
END SWITCH Z
END SWITCH A
Author : Luís Dias
Date : 08/04/2023
Fins de curso
Lançador de bolas UA
Folio : 15/16


1
2
3
4
5
6
7
8
A
B
C
D
E
F
G
H
U
V
W
BLDC
M
GND
VCC
HA
HB
HC
M_LEFT
M_RIGHT
Author : Luís Dias
Date : 12/04/2023
BLDC conexões externas
Lançador de bolas UA
Folio : 16/16