INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* SABARÁ
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Larissa da Silva Moreira

AUTOMAÇÃO DE TESTES DE CONTINUIDADE EM CABOS ELÉTRICOS

Sabará
2025

LARISSA DA SILVA MOREIRA

AUTOMAÇÃO DE TESTES DE CONTINUIDADE EM CABOS ELÉTRICOS

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Sabará, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Rodrigo Hiroshi Murofushi

Sabará
2025

Moreira, Larissa da Silva

M838a Automação de testes de continuidade em cabos elétricos [manuscrito]. / Larissa da Silva Moreira. - 2025.

77 f. : il.

Orientação: Prof. Dr. Rodrigo Hiroshi Murofushi.

Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) – Instituto Federal de Minas Gerais, *Campus* Sabará.

1. Cabos elétricos – Testes de continuidade. – Monografia. 2. Correntes contínuas. – Monografia. – Monografia. 3. Servidores da Web. – Monografia. 4. Microcontroladores. – Monografia. 5. Automação - Testes. – Monografia. 6. Controle de Qualidade. – Monografia. I. Murofushi, Rodrigo Hiroshi. II. Instituto Federal de Minas Gerais, *Campus* Sabará. III. Bacharelado em Engenharia de Controle e Automação. IV. Título.

CDU 681.5:37

César dos Santos Moreira / CRB6-2229
Biblioteca do IFMG *Campus* Sabará

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Sabará
Diretoria de Ensino, Pesquisa e Extensão
Coordenação do Curso de Engenharia de Controle e Automação
Rodovia MGC 262, Km 10 - Bairro Sobradinho - CEP 34590-390 - Sabará - MG
- www.ifmg.edu.br

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Ao dia 28 do mês de março do ano de 2025, às 18:30 horas, sob a presidência de Rodrigo Hiroshi Murofushi, a discente **Larissa da Silva Moreira** do curso de Engenharia de Controle e Automação, R.A nº 0042699 do IFMG *campus* Sabará, defendeu o Trabalho de Conclusão de Curso (TCC) intitulado "**Automação de Testes de Continuidade em Cabos Elétricos**", foi **aprovado** e avaliado com a nota final média de **87,5 pontos**, que está condicionada ao cumprimento dos procedimentos pós-defesa do TCC.

Compuseram a Banca Examinadora:
Membro 1: Dr. Rodrigo Hiroshi Murofushi - IFMG *campus* Sabará (orientador) ,
Membro 2: Dra. Mariella Maia Quadros - IFMG *campus* Sabará,
Membro 3: Me. Moisés Martins Gonçalves - IFMG *campus* Sabará.

A discente deverá apresentar o trabalho com as devidas modificações em formato pdf, até 14/04/2025 ao professor orientador e fazer o depósito no repositório institucional de TCC do IFMG. O não cumprimento dos procedimentos pós-defesa de TCC até a data estipulada implica no não cumprimento das horas referentes ao componente curricular de Trabalho de Conclusão de Curso.

Sabará, 28 de março de 2025.

Documento assinado eletronicamente por **Rodrigo Hiroshi Murofushi**, **Professor EBTT**, em 28/03/2025, às 21:45, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Moisés Martins Gonçalves**, **Técnico de Laboratório**, em 29/03/2025, às 10:01, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Mariella Maia Quadros**, **Professora EBTT**, em 31/03/2025, às 15:19, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **2248091** e o código CRC **F88C6C6B**.

À Estela, minha professora, que pediu aos meus pais que investissem nos meus estudos, pois acreditava que eu chegaria longe.

AGRADECIMENTOS

À Deus, que me dá forças para levantar todos os dias e para lutar para alcançar todos os meus sonhos.

Aos meus pais, Adriano e Julieta, por me ensinarem, desde criança, que os estudos me fariam alcançar todos os meus sonhos. Cada ensinamento e cada palavra de incentivo foram fundamentais para que eu chegasse até aqui.

À minha irmã, Letícia, que por muitos anos foi minha parceira de estudos e hoje é meu exemplo de dedicação e esforço.

Ao meu companheiro e amor da minha vida, Diego, por estar comigo durante esta caminhada e por todo o incentivo. Obrigada por acreditar em mim, e por fazer que eu acredite em mim mesma.

Aos meus professores, em especial o Rodrigo Hiroshi, meu orientador, que foi fundamental na minha formação, e nesse trabalho. Sem você acho que eu ainda estaria um pouco perdida.

À MOBA do Brasil, que me permitiu realizar esse trabalho e acreditou no meu potencial. Vocês foram fundamentais no meu desenvolvimento acadêmico e profissional.

E um agradecimento especial, à Larissa do passado, que sempre sonhou e se esforçou muito para chegar até aqui.

"A única coisa que separa as mulheres negras de qualquer outra pessoa é a oportunidade."Viola Davis

RESUMO

A verificação de continuidade elétrica é um dos processos fundamentais no controle de qualidade de circuitos e dispositivos eletrônicos, sendo essencial para garantir a integridade dos sistemas. No entanto, os métodos tradicionais de teste geralmente envolvem inspeção manual, tornando o processo sujeito a erros humanos, demorado e pouco eficiente em larga escala. Isso pode resultar em falhas não detectadas, retrabalho e prejuízos financeiros para a empresa. Diante desse problema, este trabalho desenvolve um dispositivo automatizado para a realização de testes de continuidade elétrica em cabos MOBA, utilizados principalmente em máquinas de grande porte, como tratores, escavadeiras e outros veículos voltados para construção e mineração. Esses cabos permitem a transmissão de dados e de energia e possuem diferentes padrões de ligação e exigem precisão na verificação de suas vias de conexão. O sistema é baseado no microcontrolador ESP32, que gerencia os testes e transmite os resultados para uma interface web acessível em computadores, tablets e celulares. A solução proposta reduz o tempo de execução dos testes, aumenta a confiabilidade do processo e elimina a necessidade de inspeção manual, garantindo maior precisão na identificação de falhas como rompimentos, curtos-circuitos e inversão de vias. Além disso, a automação possibilita a padronização dos testes, tornando-os mais rápidos e consistentes. O sistema desenvolvido se apresenta como uma ferramenta viável para empresas que buscam otimizar seus processos de controle de qualidade, oferecendo um método eficiente e preciso para a verificação de continuidade elétrica.

Palavras-chave: Automação; ESP32; Servidor Web; Continuidade Elétrica; Controle de Qualidade.

ABSTRACT

Electrical continuity verification is one of the fundamental processes in the quality control of circuits and electronic devices, being essential to ensure system integrity. However, traditional testing methods often rely on manual inspection, making the process prone to human errors, time-consuming, and inefficient on a large scale. This can result in undetected failures, rework, and financial losses for the company. To address this issue, this study develops an automated device for performing electrical continuity tests on MOBA cables, which are mainly used in heavy machinery such as tractors, excavators, and other vehicles designed for construction and mining. These cables allow the transmission of both data and power, have different wiring patterns, and require precision when verifying their connection paths. The system is based on the ESP32 microcontroller, which manages the tests and transmits the results to a web interface accessible from computers, tablets, and smartphones. The proposed solution reduces test execution time, increases process reliability, and eliminates the need for manual inspection, ensuring greater accuracy in identifying faults such as open circuits, short circuits, and incorrect wiring. Additionally, automation enables standardized testing, making the process faster and more consistent. The developed system presents itself as a viable tool for companies seeking to optimize their quality control processes, offering an efficient and precise method for electrical continuity verification.

Keywords: Automation; ESP32; Web Server; Electrical Continuity; Quality Control.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AP	Ponto de Acesso
CI	Circuito Integrado
CLP	Controlador Lógico Programável
CSS	Folha de Estilo em Cascata
GPIO	Portas Programáveis de Entrada e Saída de Dados
HTTP	Protocolo de Transferência de Hipertexto
HTTPS	Protocolo de Transferência de Hipertexto Seguro
HTML	Linguagem de Marcação e Hipertexto
IFMG	Instituto Federal de Minas Gerais
IHM	Interface Homem-Máquina
IOT	Internet das Coisas
LED	Diodo Emissor de Luz
LCD	Display de Cristal Líquido
PHP	Pré-Processador de Hipertexto
SPI	Interface Periférica Serial
SPIFFS	Sistema de Arquivo Flash de Interface Periférica Serial
USART	Receptor/Transmissor Assíncrono Umiversal
V	Volts
W3C	Consórcio do Mundo da Web

SUMÁRIO

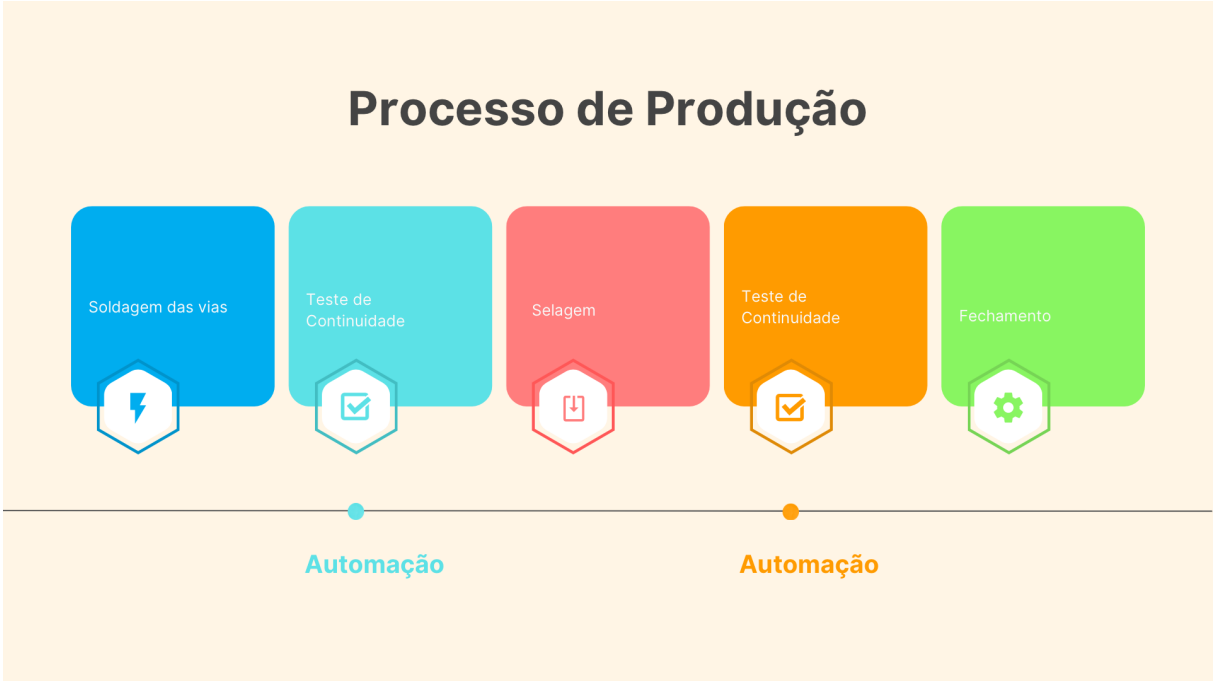
1 INTRODUÇÃO

Em ambientes industriais e operacionais, a qualidade dos processos e equipamentos é fundamental para garantir a eficiência e a segurança das operações. Sistemas e máquinas, sejam eles de produção, controle ou monitoramento, precisam funcionar de maneira precisa e contínua para evitar paradas indesejadas, minimizar desperdícios e garantir a segurança dos trabalhadores.

Um dos componentes críticos nesses sistemas são os cabos, responsáveis pela transmissão de sinais e energia. A falha em cabos pode resultar em sérios problemas, incluindo interrupção de serviços, danos a equipamentos e até riscos à segurança. Por isso, a realização de testes rigorosos em cabos é uma prática essencial para garantir sua integridade e funcionamento adequado.

Na empresa MOBA existem variados modelos de cabos elétricos, destinados à alimentação e à conexão dos sistemas eletrônicos. A fabricação desses cabos é realizada semanalmente, com uma média de 90 unidades produzidas por semana, seguindo um processo de produção em etapas, conforme ilustrado na Figura 1.

Figura 1 – Processo de Produção dos Cabos MOBA.



Fonte: Do próprio autor, 2024.

Na etapa inicial do processo produtivo, realiza-se a soldagem das vias, em que cada via é posicionada e soldada sobre o seu respectivo conector, conforme ilustrado na Figura 2.

Figura 2 – Etapa 1 - Soldagem das vias.

Fonte: Do próprio autor, 2024.

A etapa seguinte compreende o primeiro teste de continuidade, cuja finalidade é verificar a integridade das conexões antes que o cabo seja submetido ao processo de selagem. É imprescindível que todas as vias estejam corretamente soldadas, pois, após a aplicação do selante, qualquer necessidade de retrabalho exigirá a remoção do material, onde o cabo será reenviado à linha de produção e só será finalizado no próximo lote.

O teste de continuidade é realizado com o auxílio de um multímetro. Para sua execução, um operador posiciona uma das pontas de prova do instrumento em uma extremidade do cabo, enquanto outro operador manipula a segunda ponta de prova na extremidade oposta. O procedimento consiste na verificação individual de cada via, assegurando a existência de continuidade elétrica, bem como na conferência entre vias distintas para garantir a ausência de curto-circuito.

Uma vez aprovado no teste inicial, o cabo passa pelo processo de selagem. Este consiste na aplicação de um composto químico com tempo de cura definido, que endurece ao secar, protegendo as soldas contra agentes externos como umidade e poeira.

Durante esse processo, pode ocorrer o rompimento de algumas vias devido ao torque aplicado ao rosquear a tampa de apoio ao selante. Por esse motivo, é realizado um segundo teste de continuidade após a cura do selante, assegurando a funcionalidade do cabo.

Por fim, o cabo é fechado e finalizado, ficando pronto para entrega.

Duas das etapas do processo de produção são os testes de continuidade e, tradicionalmente, são realizados de forma manual, com o auxílio de um multímetro, o que pode ser demorado, sujeito a erros humanos e limitado em termos de produtividade e consistência.

Vale destacar que a execução dos testes em um lote completo de cabos leva, em média, dois dias, e pode apresentar falhas recorrentes, como vias invertidas, vias não soldadas ou vias com curto-circuito — o que reforça a importância da padronização e automação do processo.

A automação de testes de qualidade em ambientes operacionais oferece uma solução para esses desafios, permitindo a execução sistemática e contínua de verificações, a coleta de dados em tempo real e a identificação precoce de falhas.

Neste contexto, este Trabalho de Conclusão de Curso tem como objetivo desenvolver um dispositivo capaz de automatizar os testes de continuidade realizados no processo de produção dos cabos MOBA. O dispositivo proposto será capaz de avaliar cinco modelos de cabos MOBA e será construído em uma caixa plástica, contendo Diodos Emissores de Luz (LEDs, do inglês *Light Emitting Diode*) como indicativo de possíveis falhas de funcionamento ou pleno funcionamento. A automatização se dará através do uso de um microcontrolador e uma interface *Web* que permitirá que o operador selecione o cabo a ser testado e obtenha o resultado do teste.

1.1 Objetivos

A crescente demanda por qualidade e eficiência nos processos industriais tem exigido soluções cada vez mais precisas e automatizadas, especialmente em atividades que envolvem testes e inspeções. No contexto da fabricação de cabos elétricos, garantir a continuidade elétrica é essencial para assegurar o funcionamento correto de sistemas eletrônicos. Assim, este trabalho propõe o desenvolvimento de um dispositivo automatizado capaz de realizar testes de continuidade de forma confiável, com o intuito de padronizar o processo, minimizar falhas humanas e otimizar o controle de qualidade na produção.

1.1.1 *Objetivo geral*

O objetivo geral é desenvolver um dispositivo para testar a continuidade de cabos elétricos de forma eficaz para a detecção de defeitos, como curtos-circuitos, rupturas internas e variações indesejadas de resistência. Além disso, pretende-se facilitar o processo de inspeção e manutenção, permitindo que problemas sejam identificados e corrigidos antes de ocorrerem falhas mais graves, com foco na melhoria da eficiência, redução de tempos de inatividade e aumento da qualidade e confiabilidade dos produtos fornecidos.

1.1.2 *Objetivos específicos*

Para alcançar o objetivo desse trabalho, será necessário cumprir os objetivos específicos a seguir.

- Selecionar modelos de cabos mais fabricados e vendidos;
- Fazer o levantamento de *datasheets* dos cabos a serem testados;
- Projetar e construir o *Hardware*;
- Desenvolver o *Software* do microcontrolador;
- Desenvolver o Servidor *web*.

1.2 Justificativa

Durante o processo de produção dos cabos MOBA ilustrado na Figura 1, as fases mais críticas são os testes de continuidade das vias, que exigem a participação de dois funcionários e consomem uma quantidade significativa de tempo. Em algumas situações, quando não há disponibilidade de tempo ou de dois funcionários para realizar o teste, essas etapas são ignoradas, comprometendo a confiabilidade do produto final.

Caso os cabos não sejam testados adequadamente, há um risco de que apresentem falhas após a conclusão da montagem. Nesse caso, o cabo precisa ser reaberto para correção, isso pode gerar várias consequências negativas:

- **Perda de Material**

  Os conectores utilizados na montagem dos cabos não podem ser reutilizadas após a abertura. Se o cabo precisar ser reaberto, essas peças serão quebradas e descartadas, gerando desperdício de material.

- **Atraso na Entrega**

  O selante utilizado na vedação dos cabos é um produto químico que, após aberto, deve ser completamente utilizado ou descartado, não sendo passível de armazenamento. Caso o cabo precise ser reaberto, o selante é removido, e só poderá ser selado novamente quando um novo lote de selante for aberto, o que geralmente ocorre apenas quando há uma nova demanda de produção.

- **Entrega em Desconformidade**

  Os sistemas eletrônicos são comercializados junto com o serviço de instalação. Se o defeito for detectado apenas durante a instalação, o técnico responsável terá que corrigir o problema no local, o que pode resultar na entrega do cabo sem o selante adequado, comprometendo sua integridade.

- **Perda de Credibilidade**

  Caso o cabo seja enviado ao cliente com defeito, tal situação poderá ocasionar transtornos e comprometer a credibilidade da empresa. Problemas no funcionamento do produto são prejudiciais à imagem da organização, impactando negativamente a confiança do cliente nos serviços prestados.

O desenvolvimento do projeto envolveu a aplicação prática de diversos conhecimentos adquiridos ao longo do curso de Engenharia de Controle e Automação, como o desenvolvimento de sistemas embarcados, automação de processos industriais, controle de qualidade e integração homem-máquina por meio de interfaces web. Durante o curso, foram abordados conteúdos fundamentais que sustentam este projeto, como eletrônica digital, instrumentação, microcontroladores, sistemas embarcados, redes industriais, programação de sistemas e desenvolvimento de interfaces.

Por outro lado, alguns conhecimentos específicos aplicados no projeto, como o uso avançado do ESP32, desenvolvimento de servidores web com HTML/CSS/JavaScript em ambientes embarcados e integração via Sistema de Arquivo Flash de Interface Periférica Serial (SPIFFS, do inglês *SPI Flash File System*), foram adquiridos de forma autodidata e por meio de cursos complementares.

O dispositivo proposto neste trabalho contribui diretamente para resolver os principais problemas enfrentados na fase de testes de continuidade dos cabos MOBA. Ao automatizar este processo, elimina-se a dependência de dois operadores, reduzindo significativamente o tempo necessário para realização dos testes — que atualmente pode levar até dois dias por lote de produção, com uma média de 90 cabos produzidos semanalmente. Além disso, o sistema automatizado proporciona maior precisão na detecção de falhas, como inversão de vias, vias não soldadas e curtos-circuitos, minimizando erros humanos e evitando retrabalhos.

Outros ganhos relevantes incluem o aumento da produtividade, pois a automação permite a execução simultânea de múltiplos testes e reduz o tempo de análise por cabo; a redução do tempo de produção, com entregas mais rápidas e confiáveis; e a eliminação de tarefas repetitivas, permitindo que os funcionários se concentrem em atividades de maior valor. A padronização e confiabilidade dos testes também é um benefício importante, garantindo uniformidade nos critérios de qualidade.

Dessa forma, este trabalho não apenas soluciona um problema prático enfrentado na linha de produção da empresa, como também aplica, de maneira efetiva, os conhecimentos adquiridos

ao longo da graduação em Engenharia de Controle e Automação, promovendo a inovação e contribuindo com melhorias reais nos processos industriais.

2 REVISÃO BIBLIOGRÁFICA

Entre os desafios de desenvolver um dispositivo para automação de testes de continuidade em cabos elétricos, destaca-se o fato de não haver um equipamento semelhante disponível no mercado, além da escassez de referências específicas sobre esse tipo de desenvolvimento. As poucas referências encontradas e citadas a seguir foram utilizadas em etapas específicas do projeto.

O trabalho de Mueller (2021) apresentou a solução mais próxima ao dispositivo proposto. Nesse estudo, foi desenvolvida uma placa eletrônica para realizar testes de continuidade elétrica nos cabos da empresa Imply, utilizando um microcontrolador, Circuitos Integrados (CIs) e um Display de Cristal Líquido (LCD, do inglês *Liquid Crystal Display*) para exibição das informações. Contudo, essa abordagem não foi considerada viável para o presente projeto, devido ao tamanho e à robustez dos conectores dos cabos da MOBA, além das limitações do display LCD, que não seria capaz de exibir todas as informações desejadas. No entanto, a utilização de um microcontrolador e a automação dos testes foram conceitos fundamentais que foram incorporados no desenvolvimento do nosso dispositivo.

O trabalho de Santos e Junior (2019) apresentou a implementação de um sistema de automação residencial utilizando o microcontrolador ESP32 e um servidor web para o controle remoto de dispositivos elétricos. A proposta é semelhante ao projeto desenvolvido neste trabalho, pois ambos buscam a automatização de processos utilizando tecnologias de comunicação sem fio e microcontroladores. No entanto, enquanto o estudo de Santos e Junior (2019) foca na automação de residências para maior acessibilidade e comodidade, o presente trabalho se concentra na automação de testes de continuidade elétrica em cabos MOBA, otimizando processos industriais e garantindo maior precisão nos diagnósticos.

No trabalho de Andia, Leite *et al.* (2014) foi desenvolvida uma bancada automatizada para testes de continuidade de aterramento e de tensão suportável em motores. Utilizando um Controlador Lógico Programável (CLP) e uma Interface Homem-Máquina (IHM) para interação com o operador, o sistema não apenas executa os testes, mas também registra os resultados, indicando se os últimos testes foram aprovados ou rejeitados. Esse trabalho reforçou a importância da automação para garantir maior eficiência e rastreabilidade nos testes. Da mesma forma, o presente projeto implementa uma interface web para que o operador visualize os resultados dos testes, garantindo que todas as medições sejam registradas e acessíveis.

No mesmo contexto, Bortoluz (2015) realizou um estudo detalhado sobre ensaios de flexão de cabos elétricos. A automação dos testes permitiu a coleta de dados precisos, possibilitando uma análise aprofundada sobre as variáveis que mais influenciam a vida útil dos cabos. Esse estudo foi relevante para o desenvolvimento do presente trabalho, pois reforça como a automação pode contribuir para a padronização dos testes e para a obtenção de resultados confiáveis. No projeto atual, os testes de continuidade foram programados para garantir uma verificação precisa e automatizada dos cabos MOBA, reduzindo erros humanos e aumentando a eficiência do processo.

Além da aplicação na automação de testes de qualidade, dispositivos como CLP e microcontroladores são amplamente usados em outras áreas industriais e de monitoramento. Por exemplo, Graça (2017) utilizou o microcontrolador NodeMCU e um Servidor *Web* para monitorar e adquirir dados sobre luminosidade, temperatura e umidade em ambientes de trabalho. Esse sistema exemplifica a versatilidade dos microcontroladores, integrando o monitoramento de variáveis ambientais aos processos industriais. O presente projeto seguiu essa abordagem ao utilizar um ESP32 para capturar e processar os dados dos testes de continuidade, além de enviar essas informações para uma interface web.

Carvalho *et al.* (2020) também utilizaram o NodeMCU para desenvolver um sistema de monitoramento e controle de irrigação de plantas, com a particularidade de utilizar o aplicativo *Pushbullet* como interface com o operador. O *Pushbullet*, um aplicativo gratuito e configurável, permite a execução de ações automáticas, demonstrando como soluções de automação podem ser aplicadas de maneira eficaz e simples em diferentes setores. Esse trabalho foi relevante para o desenvolvimento do presente projeto ao demonstrar como a comunicação entre dispositivos e interfaces web pode ser utilizada para melhorar a experiência do usuário e tornar os sistemas mais acessíveis. Inspirado nisso, o dispositivo desenvolvido emprega uma interface web responsiva, permitindo que os testes de continuidade sejam monitorados de qualquer dispositivo com acesso à rede local.

Esses trabalhos ilustram como a automação, aplicada de forma inovadora, pode otimizar processos industriais e de monitoramento, destacando o papel central de tecnologias como CLP, CIs e microcontroladores na melhoria da eficiência, da precisão e da confiabilidade dos processos de controle de qualidade. O presente projeto foi baseado nessas abordagens adaptando-as para o contexto da verificação de continuidade elétrica em cabos MOBA, criando um dispositivo automatizado que garante maior confiabilidade e praticidade na realização dos testes.

3 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica deste trabalho apresenta os conceitos essenciais para o desenvolvimento do dispositivo proposto. São abordados temas relacionados aos princípios de funcionamento dos testes de continuidade, às tecnologias utilizadas no projeto e aos componentes envolvidos na sua implementação. Esses conceitos são fundamentais para compreender as decisões tomadas ao longo do desenvolvimento do hardware e software, garantindo a eficiência e precisão do sistema.

3.1 Cabos MOBA

Os cabos MOBA (Figura 3) são componentes elétricos projetados para aplicações industriais severas, especialmente em máquinas de grande porte como tratores, escavadeiras e veículos utilizados nos setores de construção e mineração. Desenvolvidos para suportar condições adversas, esses cabos oferecem elevada resistência mecânica e térmica, garantindo confiabilidade na transmissão de sinais e energia, mesmo sob vibração constante, exposição a poeira, umidade e variações extremas de temperatura.

Fabricados com materiais robustos e conexões de alta precisão, esses cabos são essenciais para o funcionamento seguro e eficiente dos sistemas eletrônicos embarcados, permitindo o controle preciso de movimentos e o monitoramento em tempo real das operações. No Brasil, são amplamente utilizados em equipamentos que exigem desempenho consistente e longa vida útil, devido à sua capacidade de manter a integridade elétrica e estrutural mesmo em ambientes hostis.

Os cabos MOBA apresentam variações de comprimento, que podem variar entre 3 e 12 metros, e possuem diferentes quantidades de vias internas, geralmente entre 4 e 12 vias. O modelo mais comum no processo produtivo é o cabo com 7 vias, utilizado em grande parte das aplicações industriais. Essas características são definidas conforme a necessidade da máquina ou do sistema em que o cabo será instalado, o que reforça a importância da correta identificação e verificação de cada modelo antes da aplicação.

Apesar da aparência visual semelhante entre os modelos, essas variações tornam fundamental a padronização e a verificação rigorosa de sua continuidade elétrica antes da instalação. Sua função vai além da simples condução elétrica, sendo elementos críticos para a comunicação e operação coordenada dos sistemas de controle em campo.

Figura 3 – Cabos MOBA.

Fonte: Do próprio autor, 2021.

3.2 Continuidade Elétrica

A continuidade elétrica é um conceito fundamental na eletricidade e eletrônica, referindo-se à capacidade de uma corrente elétrica fluir de maneira ininterrupta por um condutor ou circuito. Para que um sistema elétrico funcione corretamente, é essencial que os caminhos pelos quais a corrente circula estejam íntegros, sem falhas ou interrupções.

A continuidade elétrica de um sistema é sua capacidade de conduzir corrente elétrica. Cada sistema caracteriza-se pela sua resistência R. Se R = 0 ohm, o sistema é um condutor perfeito. Se R é infinita, o sistema é um isolante perfeito. Quanto menor a resistência de um sistema, melhor será sua continuidade elétrica (CABLOFIL, 2011).

A verificação da continuidade elétrica é amplamente utilizada na manutenção e teste de circuitos, garantindo que fios, cabos, conexões e componentes estejam devidamente conectados. Esse teste é essencial para identificar possíveis problemas, como condutores rompidos, conexões defeituosas ou soldas frias, que podem comprometer o desempenho de um sistema elétrico ou

eletrônico.

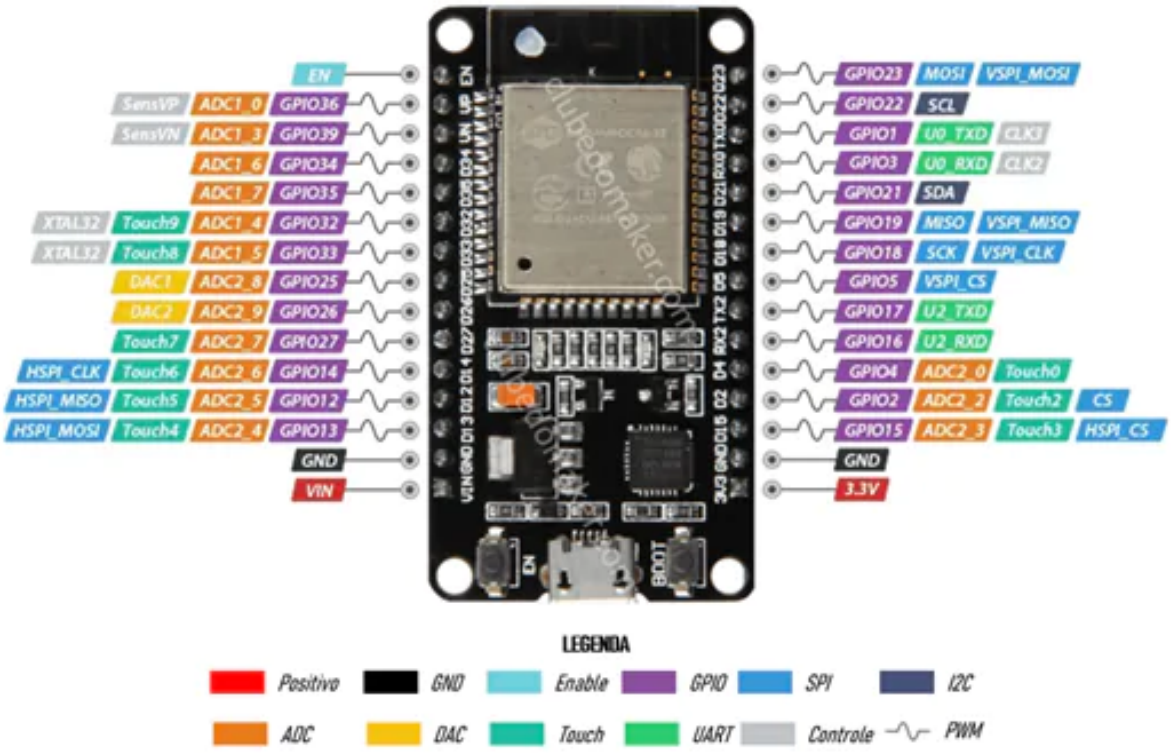
Na prática, a continuidade elétrica pode ser testada utilizando um multímetro ou outros dispositivos específicos. Quando há uma conexão adequada entre dois pontos, o equipamento indica a passagem de corrente, confirmando que o circuito está fechado. Caso contrário, a ausência de continuidade sugere que há uma interrupção no condutor ou no próprio circuito.

3.3 Microcontrolador ESP32

O ESP32, ilustrado na Figura 4, é um microcontrolador versátil e de alto desempenho desenvolvido pela fabricante chinesa Espressif Systems, lançado em 2016. Ele é a evolução do ESP8266 e oferece conectividade Wi-Fi e Bluetooth integradas, bem como maior poder de processamento e mais recursos, tornando-o amplamente utilizado em projetos de Internet das Coisas (IoT, do inglês Internet of Things) e eletrônicos. Desde seu lançamento, o ESP32 ganhou popularidade na comunidade de desenvolvedores devido à sua versatilidade e recursos avançados.

A série ESP32 emprega um microprocessador Tensilica Xtensa LX6 em variações de núcleo duplo e núcleo único e inclui interruptores de antena integrados, amplificador de potência, amplificador de recepção de baixo ruído, filtros e módulos de gerenciamento de energia (HENDRY, 2019).

Figura 4 – Microcontrolador ESP32.



Fonte: Clube do Maker, 2024.

No projeto em questão, foram aproveitadas diversas de suas funcionalidades para automatizar os testes de continuidade dos cabos MOBA. A conectividade Wi-Fi integrada foi utilizada para hospedar uma interface Web, permitindo que o operador selecione o modelo do cabo a ser testado e visualize os resultados remotamente por meio de computadores, tablets ou smartphones. A presença de múltiplos pinos GPIO possibilitou a conexão com os cabos em teste e com os LEDs indicadores de falha ou funcionamento adequado, essenciais para a sinalização visual dos resultados.

Além disso, a memória SRAM de 520 KB e o processador dual-core Tensilica Xtensa LX6, operando entre 160 MHz e 240 MHz, forneceram desempenho suficiente para gerenciar simultaneamente a lógica de testes, a comunicação Wi-Fi e o controle da interface. A comunicação serial UART foi utilizada durante o desenvolvimento e depuração do sistema, permitindo o envio de mensagens de status e resultados diretamente ao terminal serial. Também foi empregada a funcionalidade de temporizador para controle do tempo de teste e de exibição dos LEDs.

3.4 Servidor *Web*

Um servidor *Web* é um sistema computacional que hospeda e fornece acesso aos conteúdos e aplicações através da internet. Geralmente contratado como um serviço, esse servidor recebe e processa as solicitações feitas por navegadores através de protocolos de rede como o Protocolo de Transferência de Hipertexto (HTTP, do inglês *Hipertext Transfer Protocol*) ou Protocolo de Transferência de Hipertexto Seguro (HTTPS, do inglês *Hipertext Transfer Protocol Secure*) (CONTROLENET, 2022).

Esse tipo de sistema não exige que a sua execução seja realizada mediante sua prévia instalação local, assim como é feito com aplicações *desktop*. A execução da grande maioria de aplicações *Web* acontece somente com o uso de um navegador *Web* como, por exemplo, Google Chrome (JUNIOR; FORTES, 2010).

O funcionamento de um servidor web ocorre em um ciclo simples: o usuário envia uma requisição através do navegador, o servidor processa essa solicitação, recupera os dados necessários (como arquivos, imagens ou *scripts*) e retorna uma resposta ao cliente, que então exibe a página requisitada.

3.5 Linguagens de Programação

Para se implementar um algoritmo em um computador, é necessário descrevê-lo de uma forma que o computador esteja apto a executá-lo. Essa descrição é feita por intermédio de uma linguagem de programação. O próprio conjunto de instruções de um processador pode ser entendido como uma linguagem de programação. Entretanto, essa linguagem normalmente não é a mais adequada para a descrição de um programa, uma vez que os algoritmos necessários podem ser sofisticados, e essa linguagem primitiva, também chamada de linguagem de máquina,

não é amigável ao programador, demandando um esforço grande na elaboração de programas mais complexos. Sendo assim, foram desenvolvidas, ao longo da história da computação, diversas linguagens de programação, cada qual, a seu tempo, introduzindo facilidades e recursos que foram tornando a tarefa de programar mais fácil e menos susceptível a erros (GUDWIN, 1997).

Uma linguagem de programação é um método padronizado utilizado para expressar as instruções de um programa a um computador programável. Ela segue um conjunto de regras sintáticas e semânticas para definir um programa de computador. Regras sintáticas dizem respeito à forma de escrita e regras semânticas ao conteúdo.

Através da especificação de uma linguagem de programação pode-se especificar quais dados um computador vai usar; como estes dados serão tratados, armazenados, transmitidos; quais ações devem ser tomadas em determinadas circunstâncias. (GOTARDO, 2015)

As linguagens de programação podem ser classificadas de diversas formas, sendo uma das principais divisões entre linguagens de baixo nível e linguagens de alto nível.

Linguagens de baixo nível são aquelas mais próximas da linguagem de máquina, como o Assembly, oferecendo maior controle sobre o hardware, mas exigindo um maior conhecimento técnico. Linguagens de alto nível são mais próximas da linguagem humana, possuindo sintaxe mais intuitiva e facilitando o desenvolvimento de *software*. Exemplos incluem *Python*, Java, C, *JavaScript* e Pré-Processador de Hipertexto (PHP, do inglês Hypertext Preprocessor).

Cada linguagem tem suas vantagens e aplicações específicas. Algumas são mais usadas para desenvolvimento web, como *JavaScript* e PHP, enquanto outras são populares para desenvolvimento de sistemas embarcados, como C e C++, devido à sua eficiência e controle sobre o *hardware*. Já linguagens como *Python* se destacam em áreas como inteligência artificial, automação e análise de dados.

Com a evolução da computação, novas linguagens e paradigmas continuam surgindo, adaptando-se às necessidades do mercado e às inovações tecnológicas. Independentemente da escolha, a compreensão das linguagens de programação é essencial para qualquer profissional que deseja atuar no desenvolvimento de *software* e sistemas computacionais.

3.5.1 *Linguagem C++*

O C++ é uma linguagem de programação de propósito geral, desenvolvida por Bjarne Stroustrup, nos laboratórios da AT&T Bell, no início dos anos 80, como uma evolução do C, incorporando, dentre outras, as seguintes extensões: suporte para a criação e uso de tipos de dados abstratos, suporte ao paradigma de programação orientada a objeto, além de diversas outras pequenas melhorias nas construções existentes no C. Algumas de suas características são o uso de tipos estáticos, a definição de classes, funções virtuais e overload de operadores para o suporte à programação orientada a objeto, o uso de templates para programação genérica, além de prover facilidades de programação de baixo nível (a exemplo do C).

A linguagem C++ é uma extensão da linguagem de programação C. A motivação para o desenvolvimento de C++ foi a complexidade. Grandes sistemas implementados com a linguagem C, são difíceis de controlar ou mesmo entender sua totalidade. O C++ surgiu para permitir que essa barreira seja criada. O objetivo do C++ é permitir que programadores possam gerenciar e compreender programas maiores e mais complexos (RICARTE, 1995).

Uma das grandes vantagens do C++ é sua flexibilidade. Ele suporta múltiplos paradigmas de programação, incluindo:

- Programação procedural, herdada do C, que permite um controle preciso sobre o fluxo do programa.

- Programação orientada a objetos, que organiza o código em classes e objetos, facilitando a reutilização e a manutenção.

- Programação genérica, através do uso de templates, permitindo a criação de código reutilizável e otimizado. Programação funcional, permitindo abordagens modernas inspiradas em linguagens como *Haskell* e *Lisp*.

O C++ também é conhecido por seu gerenciamento eficiente de memória, permitindo o uso de ponteiros e alocação dinâmica. Isso o torna ideal para sistemas embarcados e aplicações que exigem alto desempenho e controle sobre os recursos do *hardware*.

3.5.2 *Linguagem HTML*

A Linguagem de Marcação e Hipertexto (HTML, do inglês *Hypertex Markup Language*) é a linguagem padrão utilizada para o acesso e exibição de páginas *Web*. As linhas de código HTML são interpretadas pelo *browser* que mostra o resultado final ao utilizador. Genericamente, a linguagem HTML é constituída de textos e de códigos especiais denominados marcas ou *tags* (COSTA, 2007).

É imprescindível deixar claro que o HTML não é uma linguagem de programação, mas uma linguagem de marcação. Considerando um combo de códigos que possibilitam definir como e onde cada elemento deve aparecer na página, por exemplo, o tamanho da letra, cor, fonte e etc. (TORRES, 2018). O que significa que ele não executa lógicas ou comandos diretamente, mas sim estrutura o conteúdo para ser interpretado pelos navegadores. Ele é utilizado em conjunto com outras tecnologias, como:

- **CSS**, responsável pelo design e pela personalização visual das páginas.

- ***JavaScript***, que adiciona interatividade e funcionalidades dinâmicas às páginas web.

3.5.3 *Linguagem JavaScript*

O *JavaScript* é uma linguagem de programação amplamente utilizada para tornar páginas *web* interativas e dinâmicas. Criada em 1995 por Brendan Eich, a linguagem foi inicialmente desenvolvida para rodar em navegadores e melhorar a experiência do usuário.

JavaScript é a linguagem de programação da *Web*. A ampla maioria dos sites modernos usa *JavaScript* e todos os navegadores modernos incluem interpretadores *JavaScript*, tornando-a a linguagem de programação mais onipresente da história. *JavaScript* faz parte da tríade de tecnologias que todos os desenvolvedores *Web* devem conhecer: HTML, para especificar o conteúdo de páginas *Web*; CSS, para especificar a apresentação dessas páginas; e *JavaScript*, para especificar o comportamento delas (FLANAGAN, 2012).

O JavaScript se destaca por ser:

- Linguagem interpretada: O código JavaScript é executado diretamente pelo navegador, sem necessidade de compilação.
- Multiparadigma: Suporta programação orientada a objetos, funcional e procedural.
- Assíncrona e baseada em eventos: Possui recursos como Promises e async/await, permitindo execução de código sem bloquear a interface do usuário.
- Independente de plataforma: Pode ser executado em qualquer dispositivo que tenha um navegador moderno.

3.5.4 *Linguagem CSS*

A Linguagem de Folha de Estilo em Cascata (CSS, do inglês *Cascading Style Sheet* é uma linguagem voltada para a criação de folhas de estilo em páginas *Web*. O termo folha de estilo significa a descrição de um conjunto de regras que permitem definir a aparência de um *website*. Quando falamos em aparência estamos nos referindo às diversas características existentes numa página, como por exemplo a fonte, os espaçamentos, as cores dos elementos, etc. (SCHEIDT, 2015).

Criado em 1996 pelo W3C (Consórcio do Mundo da Web, do inglês *World Wide Web Consortium*), o CSS surgiu como uma solução para separar a estrutura do conteúdo (HTML) da sua apresentação visual. Essa separação torna o desenvolvimento *web* mais organizado, permitindo a reutilização de estilos e facilitando a manutenção de sites.

O CSS possui diversas funcionalidades que permitem personalizar páginas *web* de maneira flexível e eficiente. Uma de suas principais vantagens é a separação entre o conteúdo e o estilo, o que mantém o código HTML mais limpo e organizado. Além disso, proporciona facilidade na

manutenção, pois as alterações de design podem ser realizadas rapidamente sem necessidade de modificar diversas páginas.

Outra característica importante do CSS é a reutilização de estilos, já que um único arquivo pode ser aplicado a várias páginas, garantindo uniformidade na apresentação do site. O CSS também possibilita a criação de *layouts* responsivos, permitindo que as páginas se adaptem a diferentes tamanhos de tela, o que é essencial para dispositivos móveis.

Além disso, ele viabiliza a implementação de animações e transições, tornando possível a criação de efeitos visuais sofisticados sem a necessidade de utilizar *JavaScript*. Essas características fazem do CSS uma ferramenta fundamental no desenvolvimento web moderno.

4 METODOLOGIA

Na proposta deste projeto, aborda-se a necessidade de um dispositivo automatizado para a realização de testes de continuidade elétrica em cabos. Atualmente, os métodos tradicionais de teste são, muitas vezes, manuais e suscetíveis a falhas humanas, tornando-se ineficientes em aplicações que exigem precisão e agilidade. Assim, este trabalho visa desenvolver um sistema capaz de automatizar esse processo, garantindo maior confiabilidade e eficiência na verificação da integridade dos cabos.

A primeira fase do desenvolvimento consistiu na seleção dos modelos de cabos que seriam testados pelo dispositivo. Para isso, foi realizada uma análise dos cabos mais fabricados e comercializados no mercado, garantindo que o projeto atendesse às principais demandas da indústria. Com base nessa seleção, foram definidos os conectores que seriam utilizados, influenciando diretamente o projeto mecânico do dispositivo.

Na segunda etapa, foi elaborado o projeto do *Hardware*, dividido em três partes principais: mecânica, elétrica e construção do dispositivo. O projeto mecânico definiu a estrutura física do equipamento, incluindo a disposição dos conectores e indicadores visuais, como LEDs para sinalização dos testes. Já o projeto elétrico determinou a forma como os cabos seriam conectados ao microcontrolador, garantindo que todas as vias fossem corretamente testadas por meio da aplicação de sinais elétricos. Por fim, a construção do dispositivo consolidou essas definições, resultando em um equipamento funcional e estruturado para os testes.

A terceira etapa corresponde ao desenvolvimento do *Software*, dividido entre a programação do microcontrolador e a criação da interface para o usuário. A programação do microcontrolador, realizada em C++ na plataforma Arduino IDE e *Visual Studio Code*, foi estruturada para gerenciar o envio e a recepção de sinais elétricos, analisando a continuidade de cada cabo testado. Além disso, o código foi desenvolvido de acordo com um fluxo lógico.

Paralelamente, foi desenvolvida uma interface web utilizando HTML e *JavaScript*, permitindo que o usuário selecione o modelo de cabo e visualize os resultados dos testes de forma clara e intuitiva. Essa interface foi projetada para ser acessível em diferentes dispositivos, como computadores, *tablets* e *smartphones*, garantindo maior flexibilidade no uso do equipamento.

Por fim, como última fase do projeto, o dispositivo foi submetido a testes para verificar sua funcionalidade e precisão. Os testes foram realizados com os cinco modelos de cabos selecionados, analisando a confiabilidade dos resultados obtidos e a capacidade do sistema de identificar falhas, como rompimentos e conexões incorretas. Os resultados indicaram que o sistema desenvolvido atende aos requisitos propostos, demonstrando sua eficácia na automação de testes de continuidade elétrica.

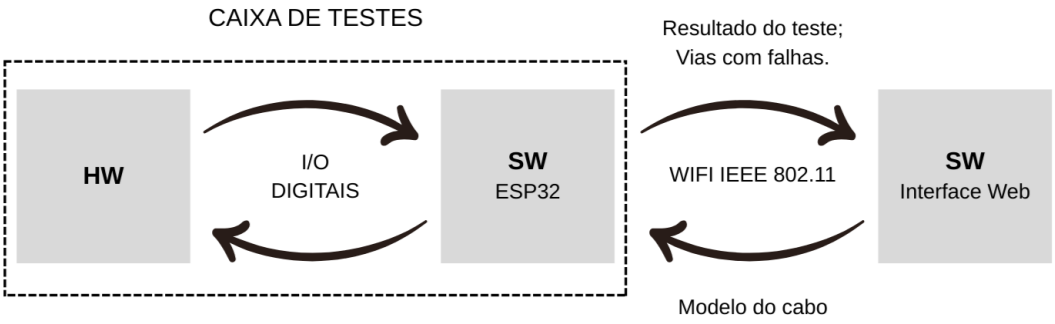
5 DESENVOLVIMENTO E RESULTADOS

O desenvolvimento deste projeto foi realizado em etapas bem definidas para garantir uma abordagem estruturada e eficiente. Antes da implementação do *Hardware* e do *Software*, foi necessário realizar um estudo detalhado para selecionar os modelos de cabos que o dispositivo seria capaz de testar. Essa seleção foi baseada na análise dos cabos mais fabricados e vendidos, garantindo que o dispositivo atendesse às demandas do mercado e tivesse uma ampla aplicabilidade.

Após essa fase inicial, o desenvolvimento foi dividido em duas etapas principais: *Hardware* e *Software*. Essa separação permitiu um planejamento mais organizado, facilitando a construção do dispositivo e a implementação do sistema de automação dos testes de continuidade em cabos elétricos.

No diagrama, ilustrado na Figura 5, é possível analisar como as etapas foram definidas e como elas se relacionam entre si.

Figura 5 – Diagrama Geral.



Fonte: Do própio autor, 2025.

5.1 Seleção dos Modelos de Cabos

Antes de iniciar o desenvolvimento do *Hardware* e *Software*, foi realizada uma análise dos tipos de cabos mais fabricados e vendidos pela empresa MOBA. Esse levantamento permitiu identificar os padrões mais utilizados, levando em consideração o número de vias, os tipos de conectores, os esquemas de ligação e os principais defeitos que poderiam ser detectados durante os testes.

Dentre os modelos analisados, foram selecionados cinco modelos de cabos MOBA, que são amplamente utilizados na indústria. O esquema elétrico dos cabos estão disponíveis no Apêndice A.

- **55-04-02-02620:** Cabo de 7 vias com conectores de 7 pinos / 7 soquetes
- **55-04-02-02621:** Cabo de 7 vias com conectores de 7 pinos / 7 soquetes
- **55-04-02-02624:** Cabo de 7 vias com conectores de 7 pinos / 7 soquetes
- **55-04-02-02561:** Cabo de 4 vias com conectores de 10 pinos / 12 pinos
- **55-04-02-02566:** Cabo de 4 vias com conectores de 10 pinos / 12 pinos

Esses cabos foram escolhidos por sua alta demanda no mercado, cada um com características e aplicações específicas, o que exigiu um sistema de testes adaptável para garantir a precisão na validação de sua continuidade elétrica. A variedade nas configurações de vias e conectores desses modelos permitiu testar a flexibilidade e a eficiência do dispositivo em diferentes condições de uso.

Com base nessa seleção, foram estabelecidos os requisitos para o projeto elétrico e a programação do microcontrolador, garantindo que o dispositivo fosse capaz de testar corretamente cada modelo de cabo. Além disso, essa definição também impactou diretamente o projeto mecânico, pois determinou os tipos de conectores que seriam utilizados no dispositivo. A estrutura física do equipamento foi projetada para acomodar esses conectores de forma organizada e acessível, permitindo que a conexão dos cabos ocorresse de maneira intuitiva e eficiente. Dessa forma, o design mecânico foi otimizado para facilitar o processo de teste, garantindo compatibilidade com os modelos escolhidos e proporcionando um manuseio prático durante as verificações de continuidade elétrica.

5.2 *Hardware*

A etapa de *Hardware* foi subdividida em três fases: projeto mecânico, projeto elétrico e construção do dispositivo. Cada uma dessas etapas foi essencial para garantir que o dispositivo tivesse um funcionamento adequado e robusto.

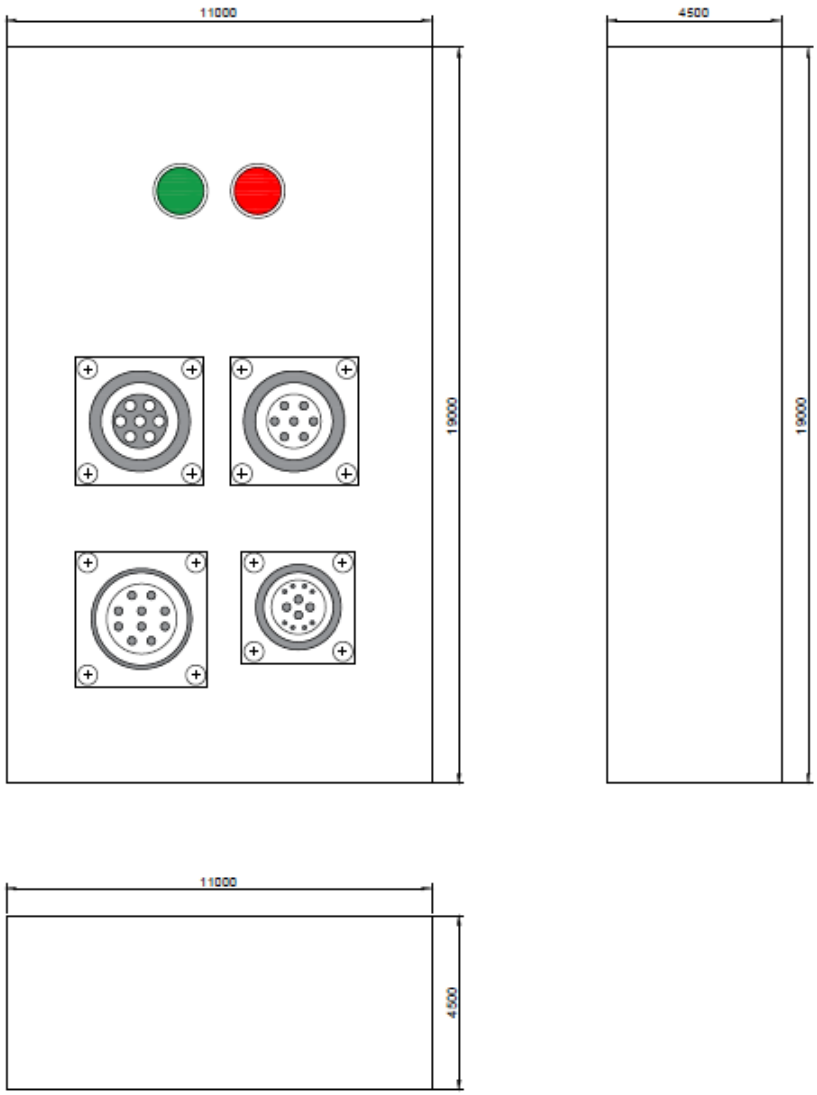
5.2.1 *Projeto Mecânico*

O projeto mecânico consistiu no planejamento da estrutura física do dispositivo, considerando a disposição dos componentes, a escolha dos materiais e a usabilidade. Definiu-se que a estrutura seria montada em uma caixa plástica resistente, garantindo proteção aos componentes internos e facilitando a manipulação dos cabos durante os testes. Na parte frontal do dispositivo, foram posicionados os conectores de teste, permitindo a conexão eficiente dos cabos. Na Figura 6 é possivel visualizar o projeto.

Os conectores utilizados foram selecionados de forma estratégica, garantindo que fosse possível realizar todos os testes necessários para os cinco modelos de cabos MOBA escolhidos.

Essa escolha foi fundamental para garantir a compatibilidade com os diferentes tipos de cabos e seus respectivos conectores. Além disso, foram adicionados indicadores visuais, como LEDs, para fornecer *feedback* instantâneo ao usuário sobre o status do teste.

Figura 6 – Projeto Mecânico.



Fonte: Do própio autor, 2025.

5.2.2 *Projeto Elétrico*

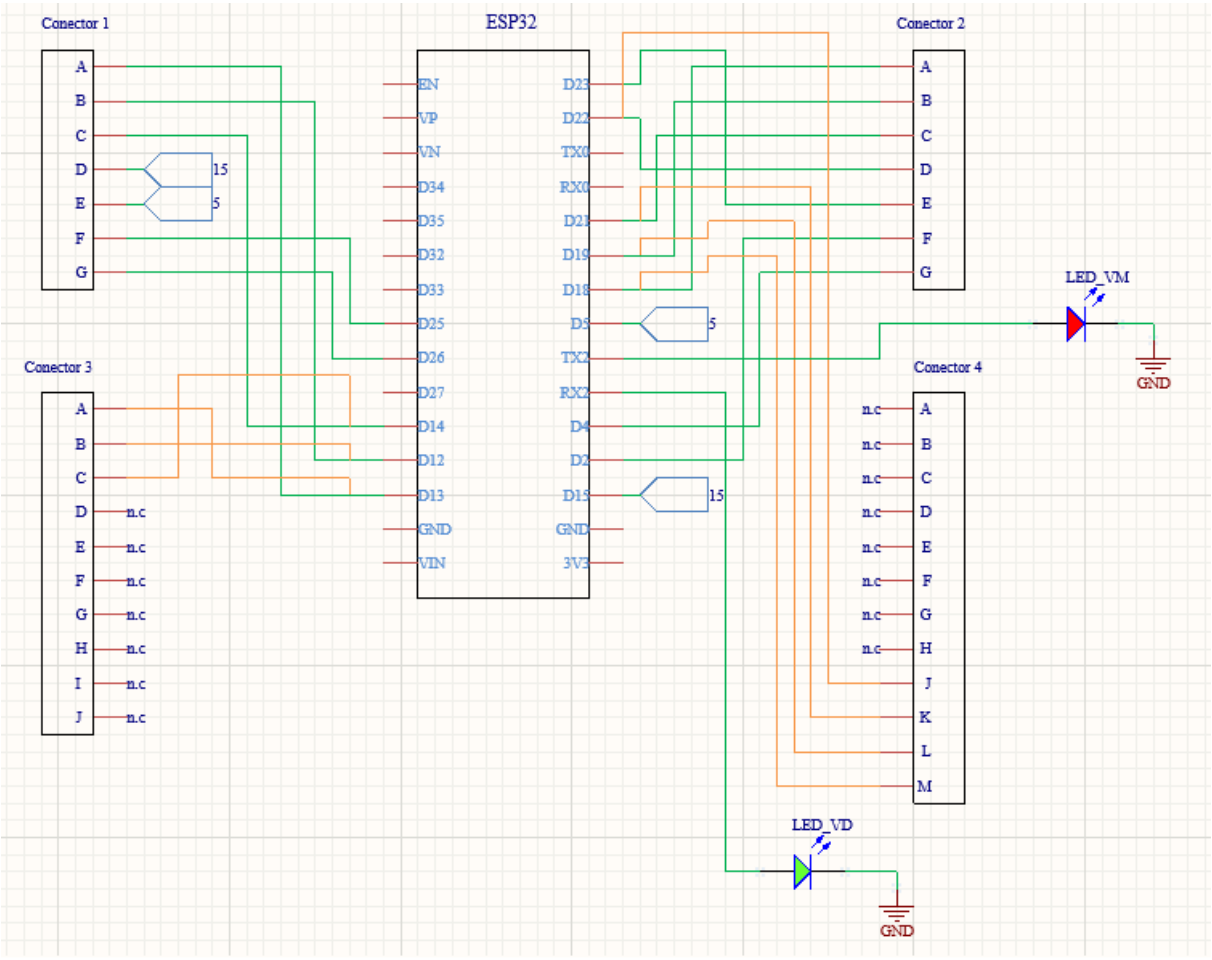
O projeto elétrico teve como foco o desenvolvimento do circuito responsável por interligar os cabos ao microcontrolador, garantindo a correta verificação da continuidade elétrica. Para isso, foi projetado um circuito elétrico, conforme demonstrado na Figura 7, no qual cada via do cabo testado fosse conectada a uma entrada ou saída digital do microcontrolador.

Para os cinco modelos de cabos MOBA, as vias de uma das extremidades do cabo são conectadas às saídas digitais do microcontrolador, permitindo que o microcontrolador envie

uma tensão pré-definida. Já as vias da outra extremidade são conectadas às entradas digitais, possibilitando a verificação da continuidade dos cabos por meio da tensão recebida. Dessa forma, foi possível identificar falhas como conexões interrompidas, curtos-circuitos e mau contato entre os condutores internos.

Além disso, para garantir um nível lógico estável e evitar leituras errôneas nas entradas digitais do microcontrolador, foram utilizados resistores pull-down internos do ESP32. Esses resistores mantêm as entradas em nível baixo (0V) quando não há sinal aplicado, prevenindo interferências e leituras indesejadas causadas por estados flutuantes. A inclusão desses resistores melhora a confiabilidade das medições, garantindo que as leituras dos sinais de continuidade sejam precisas e consistentes.

Figura 7 – Projeto Elétrico.



Fonte: Do própio autor, 2025.

5.2.3 *Construção do Dispositivo*

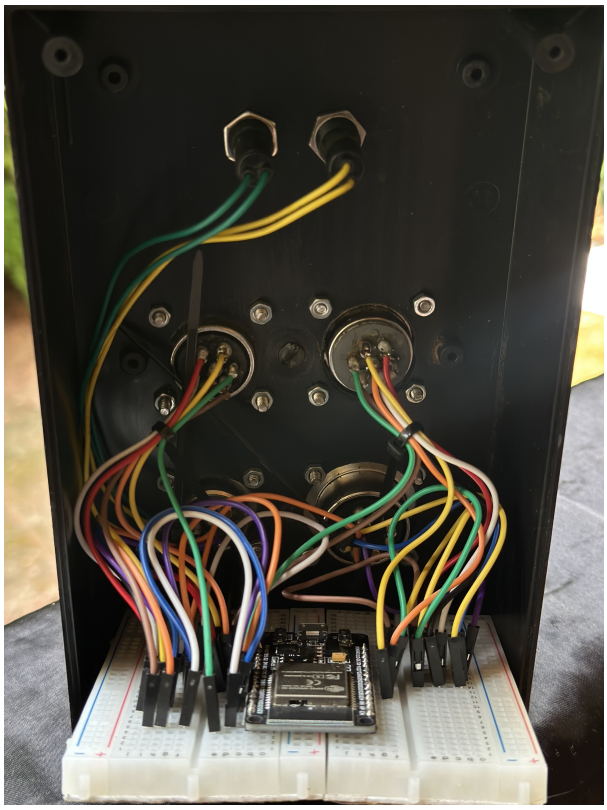
A construção do dispositivo consistiu na implementação prática dos projetos mecânico e elétrico. A montagem foi realizada utilizando uma caixa plástica previamente planejada, onde foram feitos furos e encaixes para a fixação dos conectores e LEDs. Internamente, foi instalado

um protoboard para organizar as conexões elétricas, de acordo com a Figura 7, e permitir ajustes durante a fase de testes. O microcontrolador foi fixado de forma segura e interligado aos demais componentes elétricos, garantindo um funcionamento estável do sistema. Nas Figuras 8 e 9, que apresentam a vista frontal e a visão interna do dispositivo, é possível visualizar o equipamento finalizado e pronto para operação.

Figura 8 – Visão Frontal do Dispositivo.

Fonte: Do própio autor, 2025.

Figura 9 – Visão Interna do Dispositivo.



Fonte: Do própio autor, 2025.

5.3 *Software*

A etapa de software foi responsável pelo desenvolvimento da programação do microcontrolador e da interface para o usuário, garantindo a automação dos testes e facilitando a interação com o dispositivo.

5.3.1 *Programação do Microcontrolador*

A programação do microcontrolador foi realizada utilizando a linguagem de programação C++, com a plataforma Arduino IDE. Esta escolha foi baseada na sua simplicidade, flexibilidade e ampla compatibilidade com o microcontrolador escolhido para o projeto, permitindo a rápida implementação e testes. O código foi estruturado para realizar a leitura das conexões elétricas dos cabos testados. Foram desenvolvidas rotinas para enviar sinais elétricos a determinadas vias e monitorar as respostas recebidas, verificando a continuidade e identificando possíveis falhas. Além disso, foram implementadas funções para processamento dos dados coletados, permitindo que o sistema determinasse automaticamente se o cabo estava dentro dos padrões esperados ou apresentava defeitos.

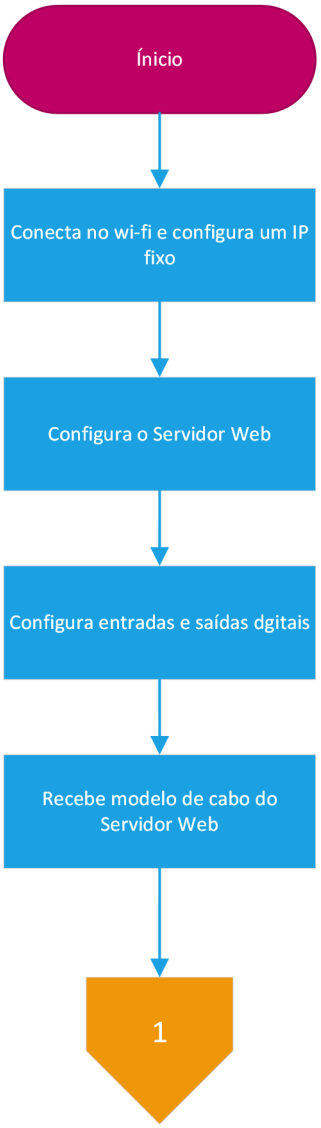
O código de programação inicia com uma série de definições para o ESP32, incluindo a criação de variáveis utilizadas na programação, a definição de entradas e saídas digitais do

microcontrolador, e funções responsáveis pela comunicação entre o controlador e o servidor *web*. Após essas definições iniciais, o sistema configura uma rede local Wi-Fi, permitindo o acesso à interface web através de um navegador. Esse processo de comunicação sem fio e a interface web tornam o uso do dispositivo mais acessível e facilitam a operação durante os testes.

O fluxograma, ilustrado nas Figuras 10, 11 e 12, detalha o funcionamento do código do microcontrolador, apresentando as etapas do processamento desde a inicialização do sistema até a exibição dos resultados dos testes. Esse diagrama proporciona uma visão clara da lógica implementada, auxiliando na compreensão do fluxo de operações e facilitando futuras modificações e otimizações.

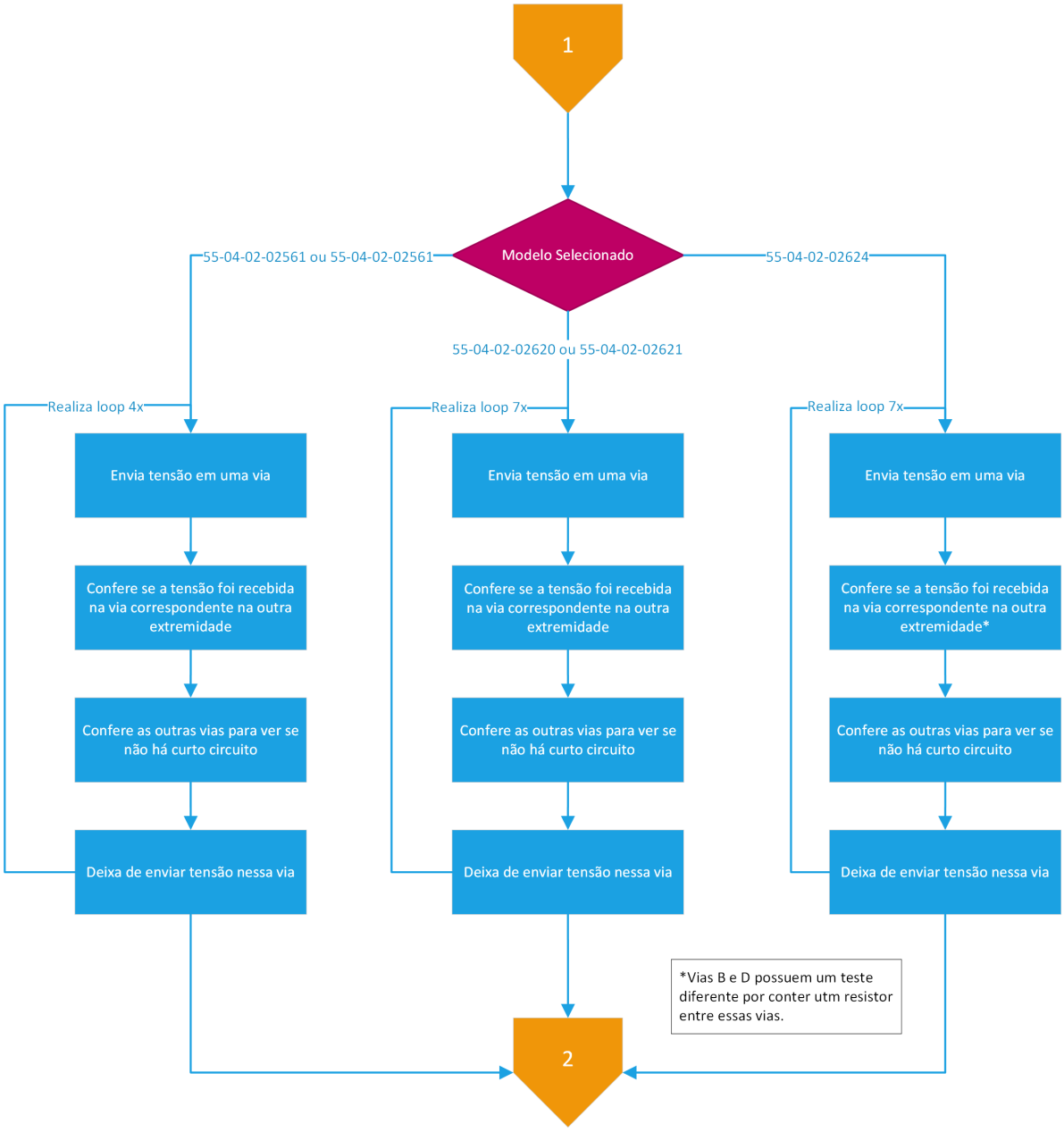
O código completo pode ser consultado no Apêndice B.

Figura 10 – Fluxograma.



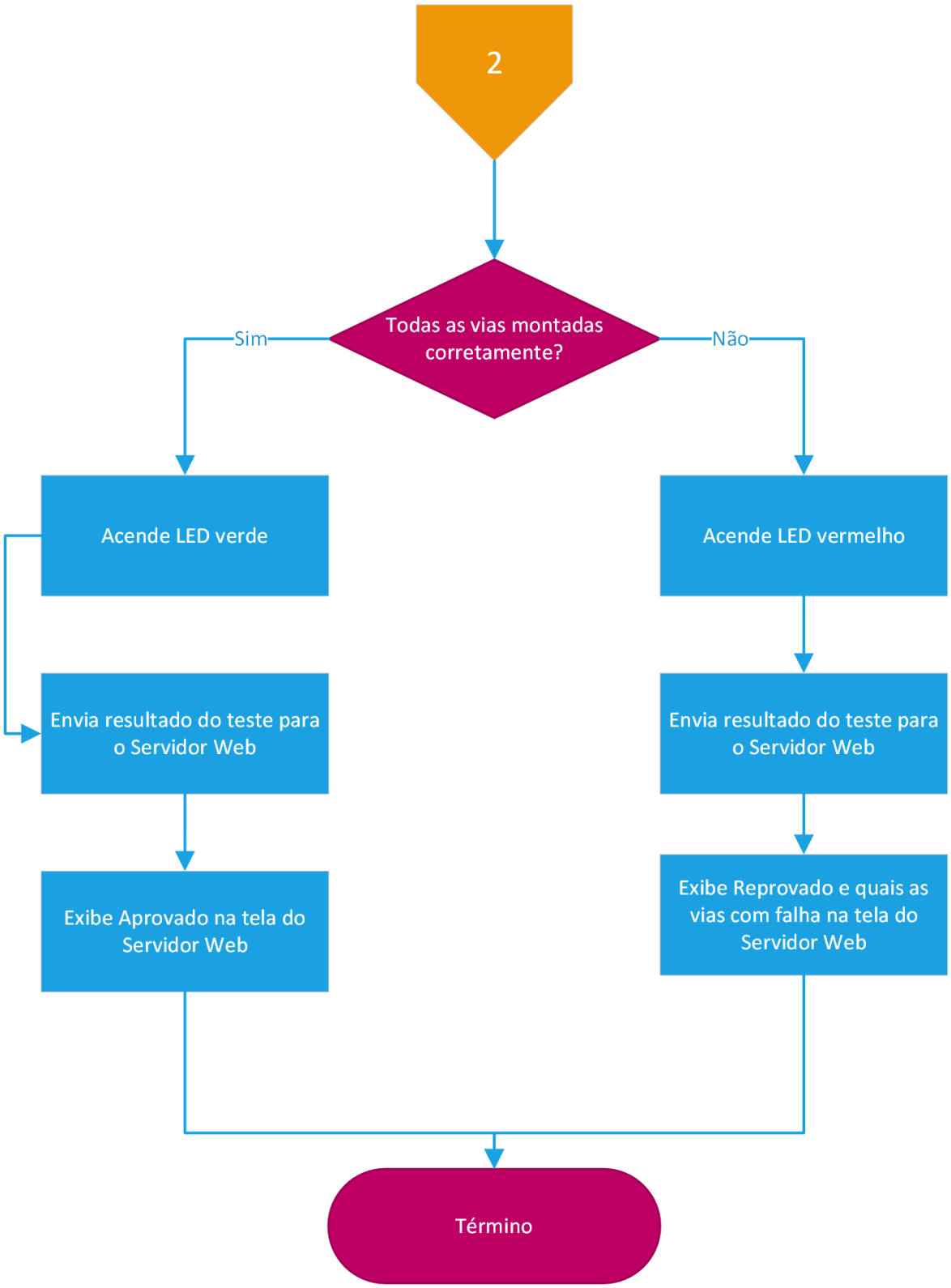
Fonte: Do própio autor, 2025.

Figura 11 – Fluxograma.



Fonte: Do própio autor, 2025.

Figura 12 – Fluxograma.



Fonte: Do própio autor, 2025.

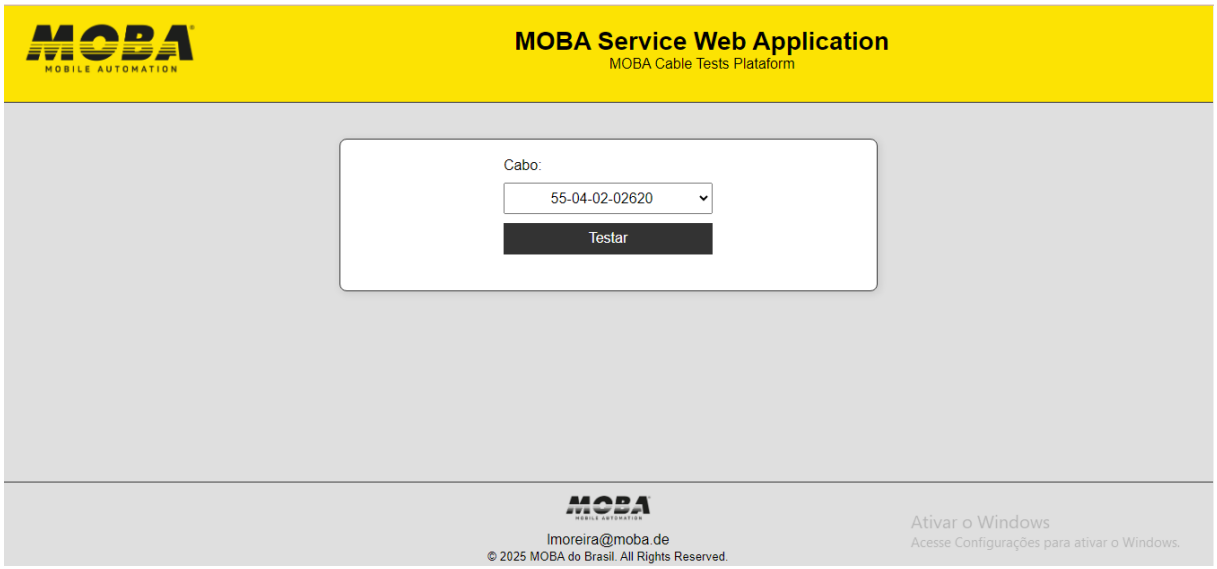
5.3.2 *Programação Servidor Web*

Além da automação dos testes, foi desenvolvida uma interface para o usuário utilizando HTML, *JavaScript* e CSS. A plataforma utilizada para o desenvolvimento da interface foi o *Visual Studio Code*, uma ferramenta poderosa e amplamente utilizada no desenvolvimento de aplicações *web*.

A interface foi projetada para selecionar o modelo de cabo a ser testado e para exibir os resultados dos testes de forma clara e intuitiva, garantindo que o operador consiga facilmente interpretar as informações fornecidas pelo sistema. Os resultados são apresentados com detalhes, incluindo o status do teste, como a aprovação ou reprovação do cabo e a identificação de falhas específicas.

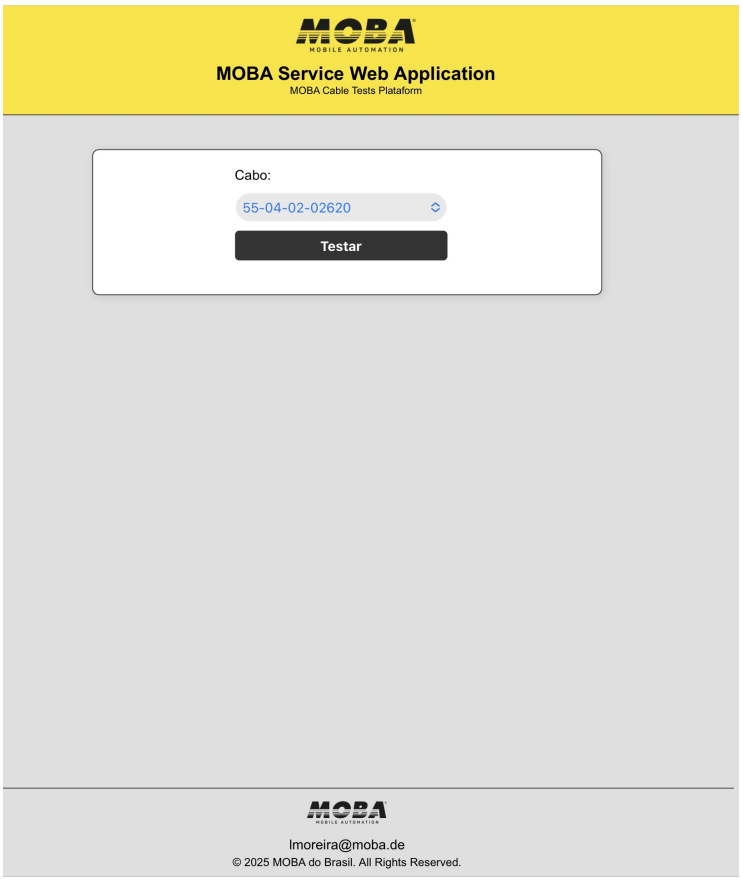
Além disso, a tela foi desenvolvida para ser compatível com computadores, tablets e celulares. Essa abordagem multiplataforma garante maior acessibilidade, permitindo que o dispositivo seja utilizado em diferentes tipos de dispositivos, dependendo da necessidade do usuário. Nas Figuras 13, 14 e 15, é possível visualizar as telas projetadas para cada plataforma, demonstrando a interface adaptada para computadores, *tablets* e celulares.

Figura 13 – Servidor *web* - Computador.



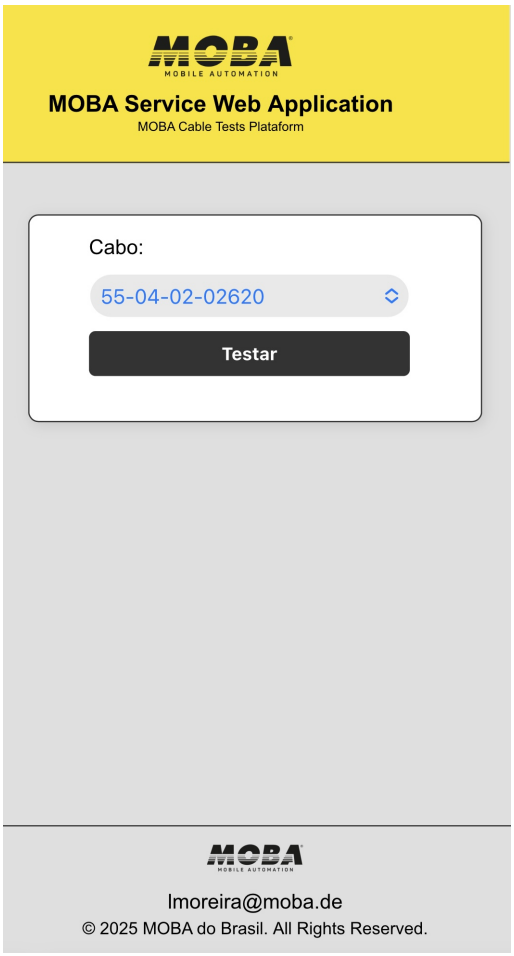
Fonte: Do própio autor, 2025.

Figura 14 – Servidor *web* - Tablet.



Fonte: Do própio autor, 2025.

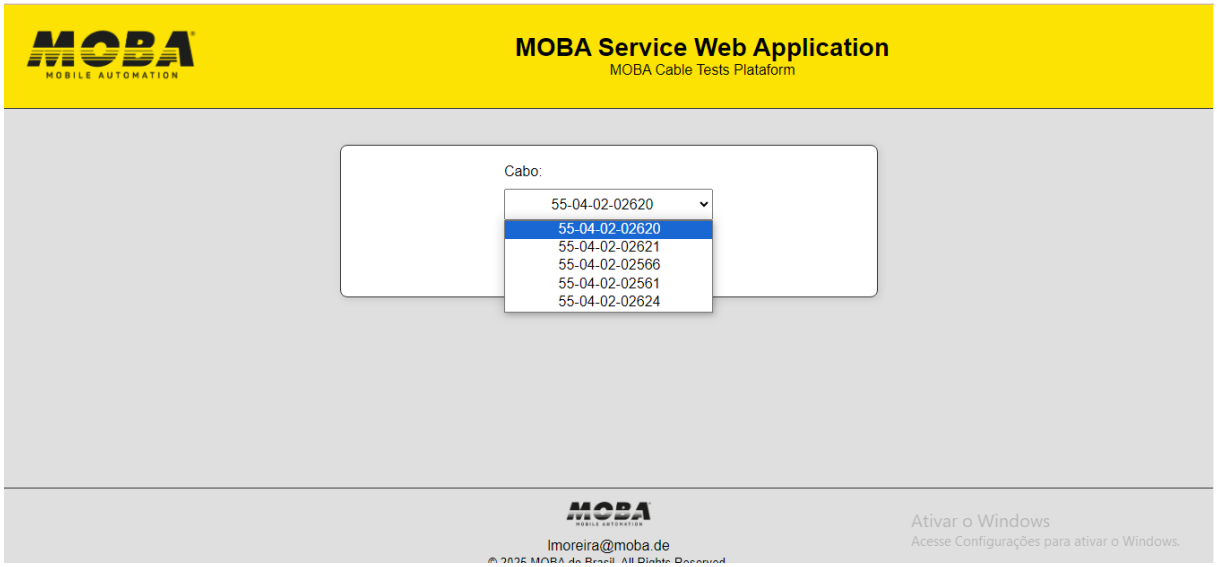
Figura 15 – Servidor *web* - Celular.



Fonte: Do própio autor, 2025.

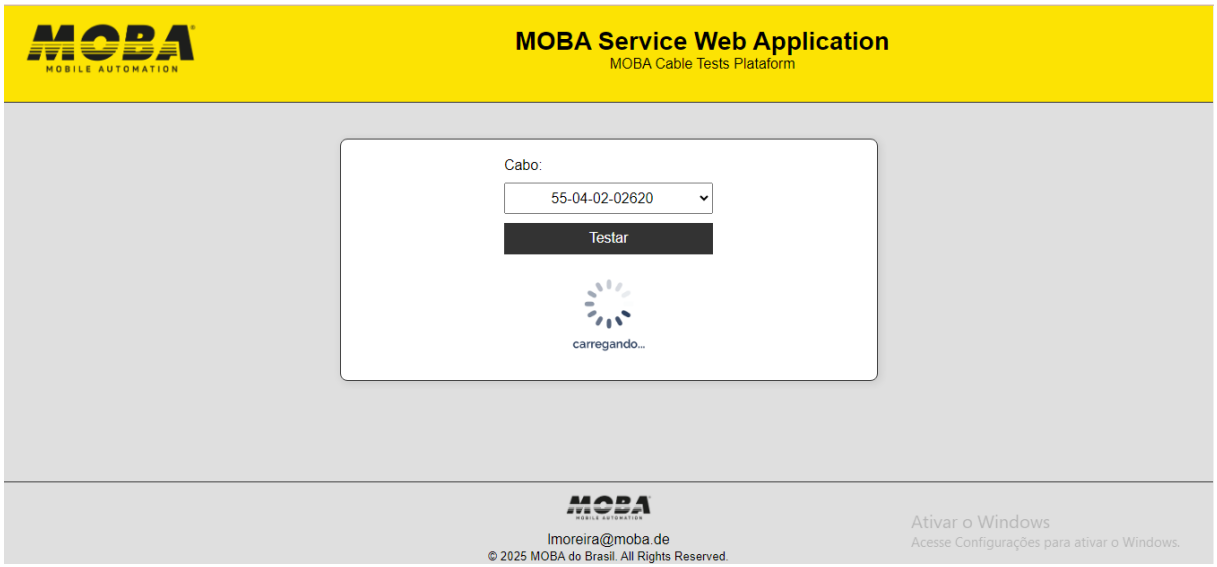
As imagens a seguir ilustram o funcionamento da interface do sistema, destacando suas principais etapas. Inicialmente, na Figura 16, é possível visualizar o menu de seleção dos cabos suspenso, permitindo que o usuário escolha o modelo de cabo a ser testado. Ao clicar em Testar, a tela exibe um símbolo de carregamento enquanto o teste de continuidade está em andamento, conforme demonstrado na Figura 17. Após a finalização do teste, a interface apresenta diferentes respostas de acordo com o resultado: uma tela indicando a aprovação do cabo (Figura 18) quando todas as vias estão em conformidade e outra informando a reprovação (Figura 19) caso sejam detectadas falhas, especificando as vias com problemas para que o operador possa identificar e corrigir os defeitos com precisão.

Figura 16 – Servidor *web* - Menu Suspenso.



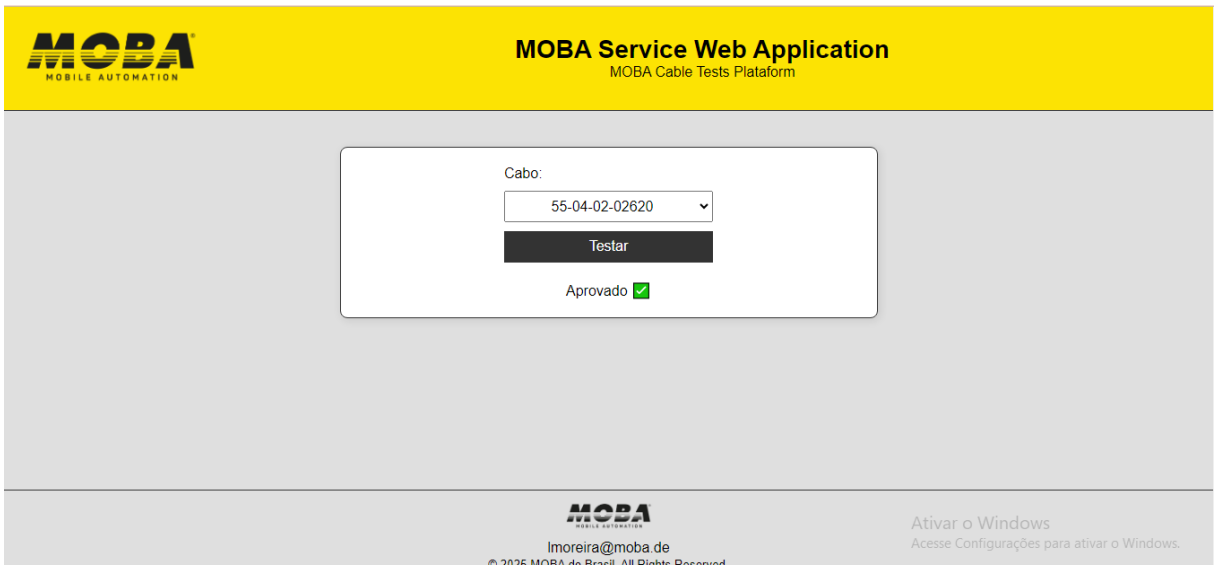
Fonte: Do própio autor, 2025.

Figura 17 – Servidor *web* - Carregando.



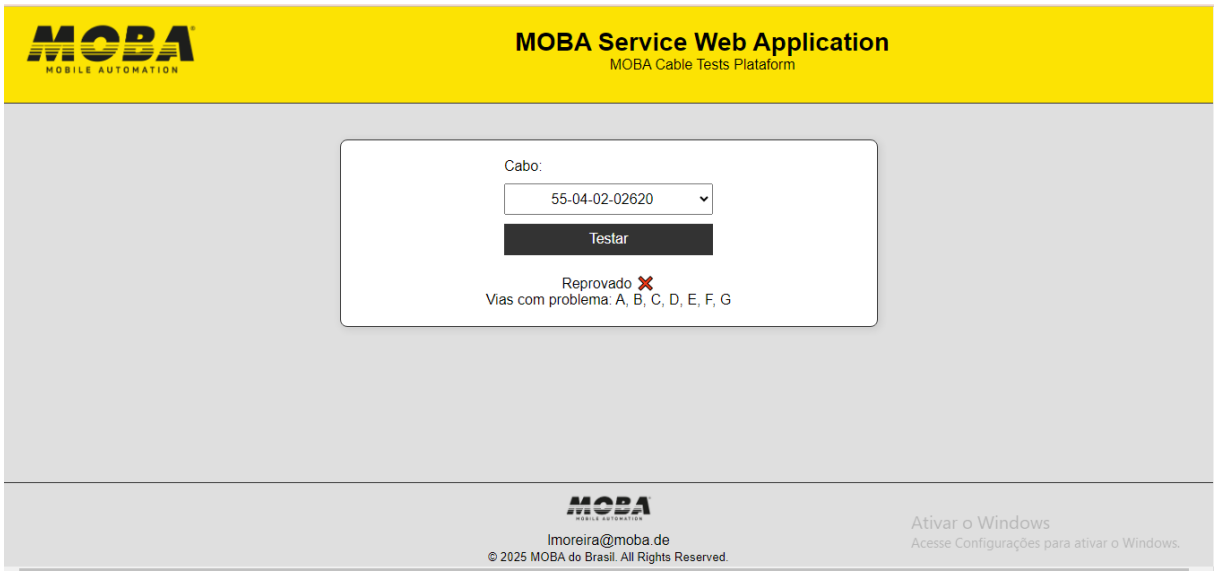
Fonte: Do própio autor, 2025.

Figura 18 – Servidor *web* - Teste Aprovado.



Fonte: Do própio autor, 2025.

Figura 19 – Servidor *web* - Teste Reprovado.



Fonte: Do própio autor, 2025.

O código completo pode ser consultado no Apêndice C.

6 CONCLUSÃO E TRABALHOS FUTUROS

O desenvolvimento deste projeto permitiu a criação de um dispositivo automatizado para testes de continuidade em cabos elétricos, proporcionando maior eficiência e precisão na verificação dos circuitos. A separação do desenvolvimento em etapas de hardware e software garantiu uma abordagem estruturada, facilitando a implementação e testes das funcionalidades. A seleção dos modelos de cabos testados orientou o projeto mecânico e elétrico, garantindo que o dispositivo fosse compatível com as principais necessidades da empresa.

A implementação do ESP32 como microcontrolador demonstrou ser uma solução viável devido à sua conectividade Wi-Fi e capacidade de processamento, permitindo a integração com uma interface web desenvolvida em HTML, CSS e JavaScript. Essa interface possibilitou uma experiência intuitiva para o usuário, permitindo a seleção do modelo de cabo e a visualização dos resultados dos testes de forma clara.

Com o funcionamento validado, conclui-se que o dispositivo atende ao objetivo de automatizar o processo de teste de continuidade, reduzindo o tempo necessário para a análise e minimizando erros humanos. O projeto pode ser utilizado em ambientes industriais e laboratoriais, contribuindo para um controle de qualidade mais eficiente na fabricação e manutenção de cabos elétricos.

6.1 Trabalhos Futuros

Embora o dispositivo tenha atingido seus objetivos, existem possibilidades de aprimoramento e expansão. Uma melhoria futura seria a implementação de um banco de dados para armazenar os resultados dos testes, permitindo um histórico detalhado das verificações realizadas.

Outra possível evolução seria a adaptação do dispositivo para testar outros tipos de cabos elétricos, além dos modelos MOBA inicialmente selecionados. Isso exigiria ajustes na interface de conexão e no software para reconhecer diferentes padrões de cabeamento.

Por fim, o desenvolvimento de um aplicativo móvel poderia facilitar ainda mais o acesso aos resultados dos testes, permitindo que os usuários consultassem as informações remotamente. Essas melhorias tornariam o dispositivo ainda mais versátil e útil para aplicações em larga escala.

REFERÊNCIAS

ANDIA, G. d. C.; LEITE, L. E. S. *et al.* **Automatização dos ensaios de continuidade de aterramento e de tensão suportável.** Tese (Doutorado) — UNIVERSIDADE DE SÃO PAULO, 2014. Citado na página 17.

BORTOLUZ, L. **Automação de ensaios por flexão de cabos elétricos e análise da vida útil dos condutores**. Tese (Doutorado) — Universidade de Caxias do Sul, 2015. Citado na página 17.

CABLOFIL. [S.l.], 2011. Citado na página 20.

CARVALHO, M. A.; ALMEIDA, G. K. F. C.; MAGALHÃES, Y. C.; ALMEIDA, W. R. M. Sistema de monitoramento e controle de irrigação de plantas utilizando nodemcu esp8266. **Revista Ceuma Perspectivas**, v. 34, n. 2, p. 36–43, 2020. Citado na página 18.

CONTROLENET. **Web Server: Saiba o que é e como funciona um servidor web**. 2022. Accessed on September 03, 2024. Disponível em: <https://www.controle.net/faq/web-server-o-que-e-como-funciona-um-servidor-web>. Citado na página 22.

COSTA, C. J. **Desenvolvimento para web**. [S.l.]: ITML press/Lusocredito, 2007. Citado na página 24.

FLANAGAN, D. **JavaScript: o guia definitivo**. [S.l.]: Bookman Editora, 2012. Citado na página 25.

GOTARDO, R. A. Linguagem de programação. **Rio de Janeiro: Seses**, v. 60, 2015. Citado na página 23.

GRAÇA, P. C. **Sistema de aquisição de dados utilizando o módulo ESP8266 NodeMCU**. Tese (Doutorado) — Universidade Estadual Paulista (Unesp), 2017. Citado na página 18.

GUDWIN, R. R. Linguagens de programação. **Campinas: DCA/FEEC/UNICAMP**, p. 24, 1997. Citado na página 23.

HENDRY, I. **ESP32 Development using the Arduino IDE**. [S.l.: s.n.], 2019. Citado na página 21.

JUNIOR, E. A. d. O.; FORTES, R. P. d. M. **Arquitetura de software na web atual: processamento no servidor**. Tese (Doutorado) — Universidade de São Paulo, 2010. Citado na página 22.

MUELLER, C. M. **Desenvolvimento de um testador de cabos Imply.** Tese (Doutorado) — Universidade de Santa Cruz do Sul, 2021. Citado na página 17.

RICARTE, I. L. M. Programação orientada a objetos com c++. **Notas de apoio a cursos-FEEC/UNICAMP**, 1995. Citado na página 24.

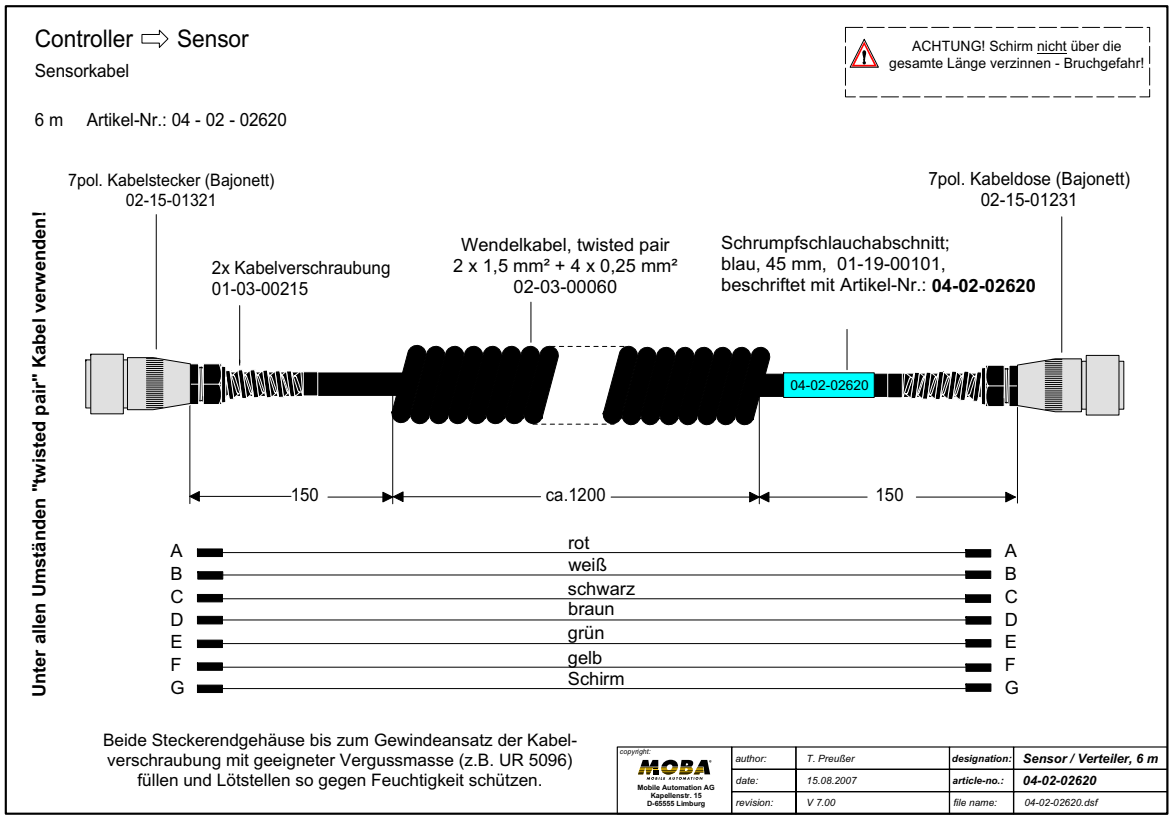
SANTOS, J. W.; JUNIOR, R. C. d. L. **Sistema de automatização residencial de baixo custo controlado pelo microcontrolador esp32 e monitorado via smartphone**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2019. Citado na página 17.

SCHEIDT, F. A. **Fundamentos de CSS: criando design para sistemas web**. [S.l.]: Outbox Livros Digitais, 2015. Citado na página 25.

TORRES, V. M. Html e seus componentes. **Revista Ada Lovelace**, v. 2, p. 99–101, 2018. Citado na página 24.

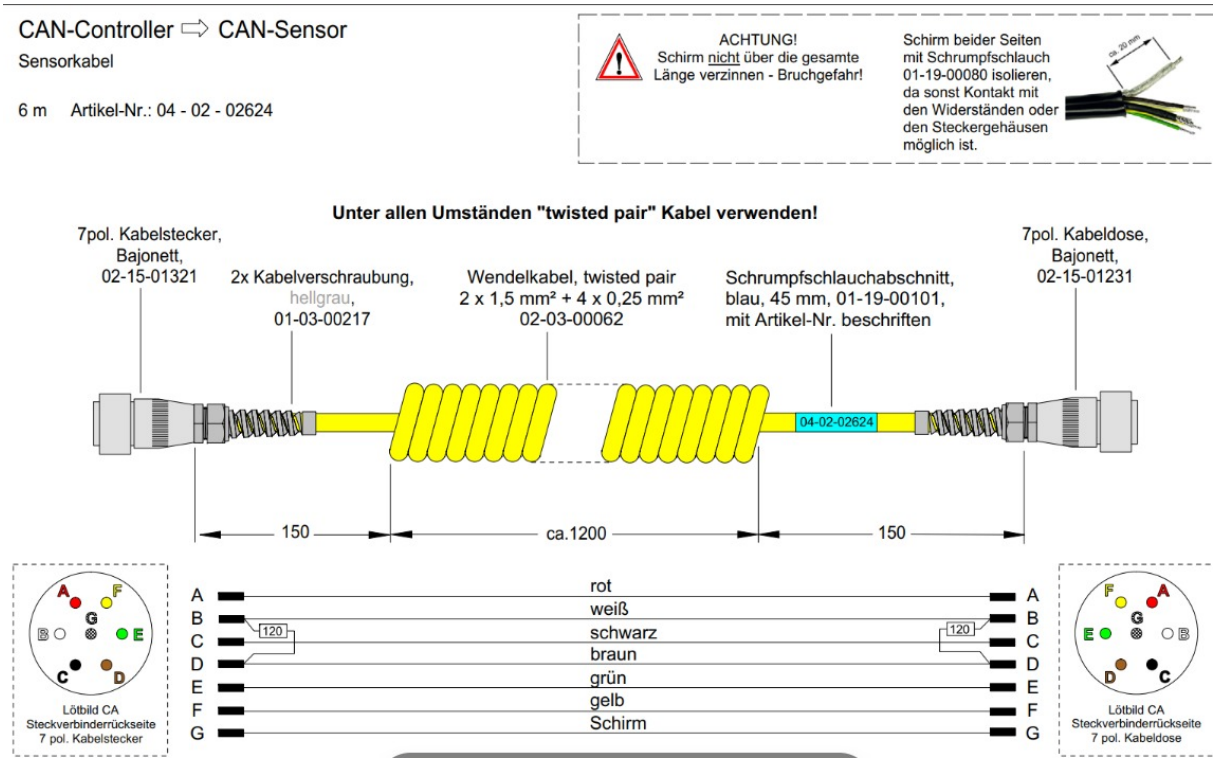
APÊNDICE A – ESQUEMA ELÉTRICO

Figura 20 – Cabo 55-04-02-02620.



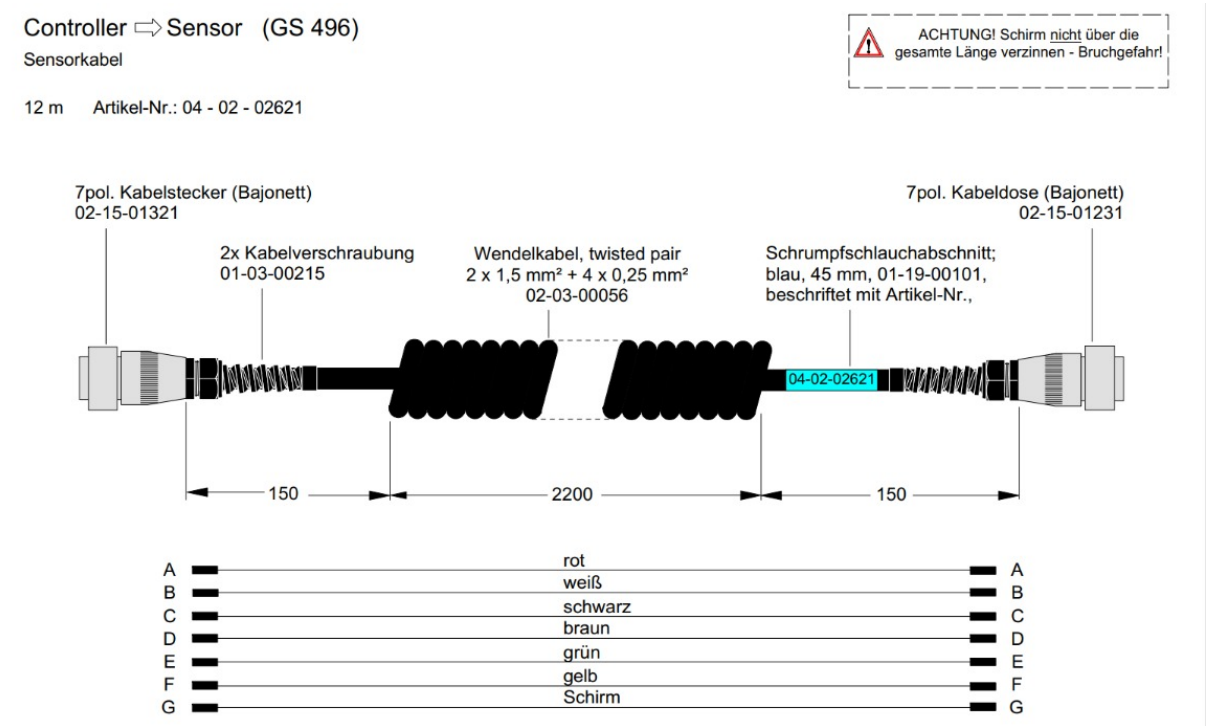
Fonte: Do próprio autor, 2024.

Figura 21 – Cabo 55-04-02-02624.



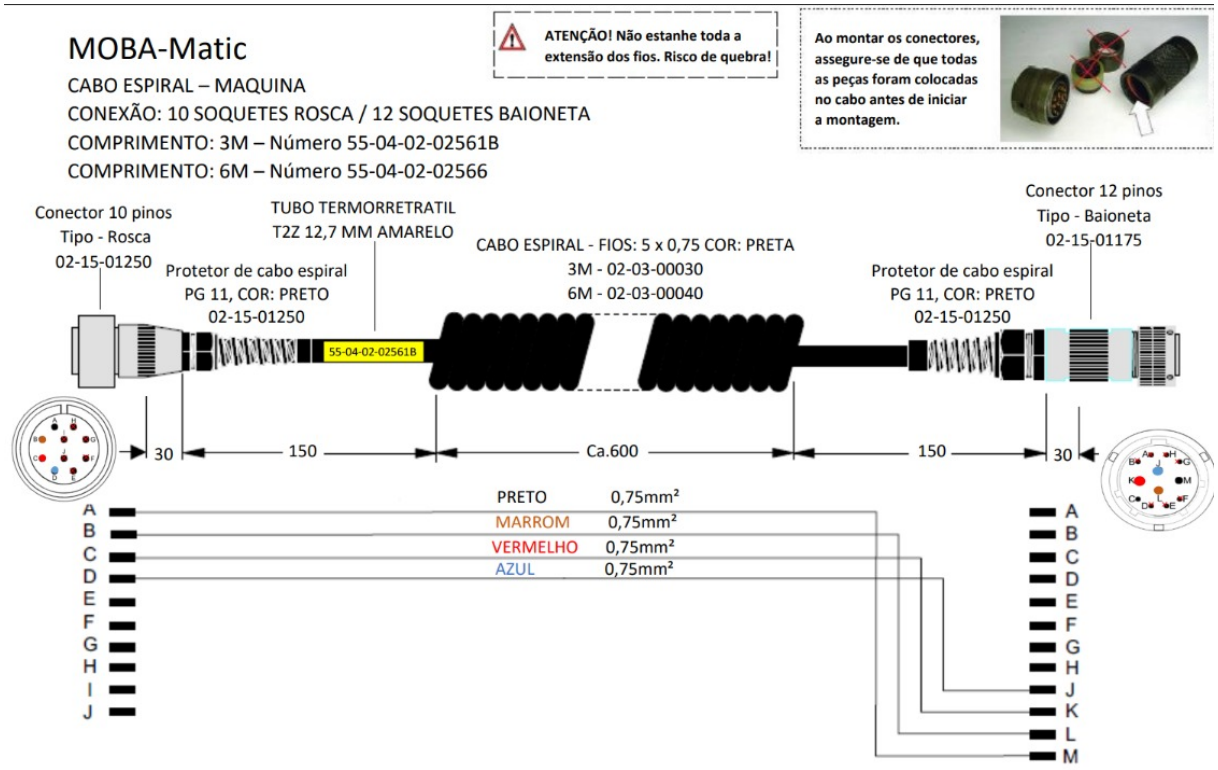
Fonte: Do próprio autor, 2024.

Figura 22 – Cabo 55-04-02-02621.



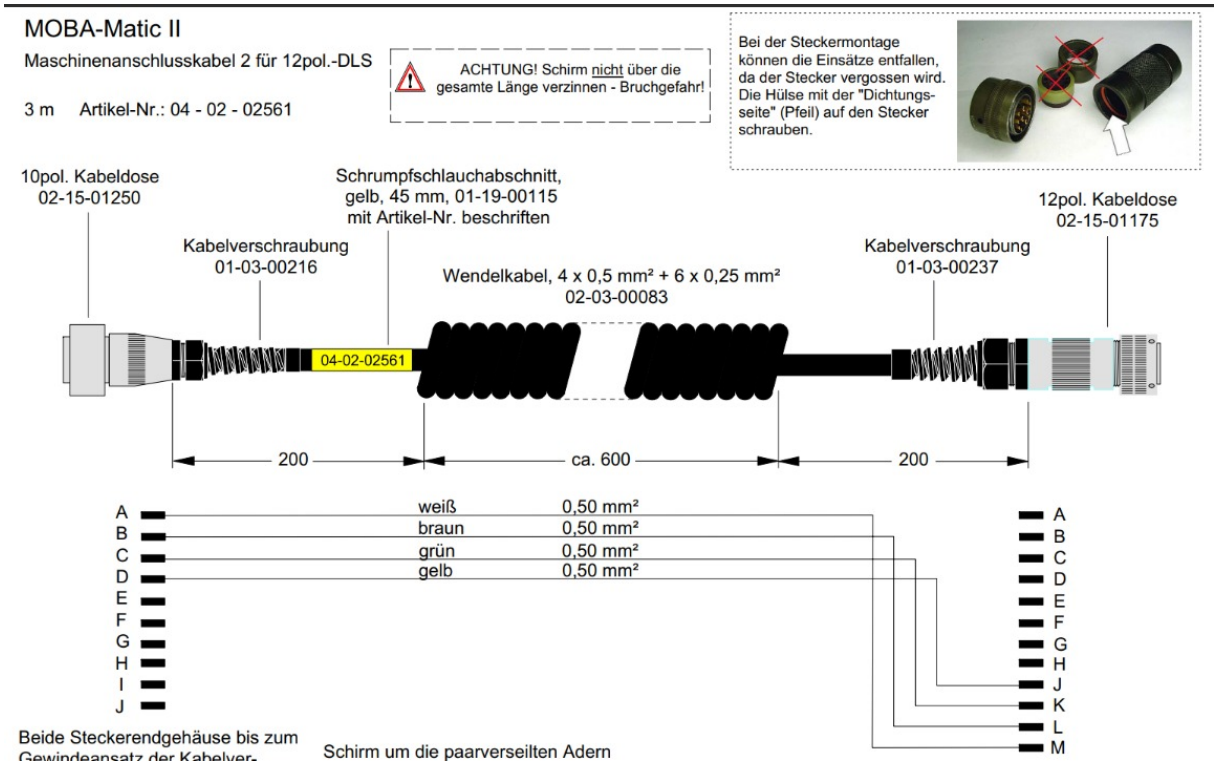
Fonte: Do próprio autor, 2024.

Figura 23 – Cabo 55-04-02-02566.



Fonte: Do próprio autor, 2024.

Figura 24 – Cabo 55-04-02-02561.



Fonte: Do próprio autor, 2024.

APÊNDICE B – CÓDIGO MICROCONTROLADOR.

```
#include <WiFi.h>
#include <SPIFFS.h>
#include <WebServer.h>

// LEDs
#define LED_VD          16
#define LED_VM          17

// Saídas
#define PIN_1           13
#define PIN_2           12
#define PIN_3           14
#define PIN_4           15
#define PIN_5           5
#define PIN_6           25
#define PIN_7           26

// Entradas
#define PIN_8           18
#define PIN_9           19
#define PIN_10          21
#define PIN_11          22
#define PIN_12          23
#define PIN_13          2
#define PIN_14          4

// Variáveis
int saidas[7] = {PIN_1, PIN_2, PIN_3, PIN_4, PIN_5, PIN_6, PIN_7};
int entradas[7] = {PIN_8, PIN_9, PIN_10, PIN_11, PIN_12, PIN_13, PIN_14};
bool vias[7] = {false, false, false, false, false, false, false};
```

```
bool CC[7] = {false, false, false, false, false, false, false};
String cabo;
int aux = 0;              // Contar curto circuitos em cada via
int auxV = 0;             // Contar vias sem defeitos
int auxCC = 0;            // Contar vias sem curto circuito
bool isOK = false;            // Indicar tudo OK no teste

WebServer server(80);

// Roda o HTML
void handleRoot() {
        File file = SPIFFS.open("/index.html", "r");
        if (!file) {
                server.send(500, "text/plain", "Erro ao carregar a página");
                return;
        }
        String html = file.readString();
        file.close();
        server.send(200, "text/html", html);
}

// Recebe o valor do cabo selecionado
void handleSelect() {
        if (server.method() == HTTP_POST && server.hasArg("cabo")) {
                cabo = server.arg("cabo");
                Serial.print("Cabo selecionado: ");
                Serial.println(cabo);
        }

        server.sendHeader("Location", "/");
        server.send(303);
```

```
}

// Envia o resultado do teste
void handleResultado() {
        String json = "{";
        json += "ïsOK:"+ String(isOK ? "true": "false") + ",";
        json += "ĊC:[";
        for (int i = 0; i < 7; i++) {
                json += String(CC[i] ? "true": "false");
                if (i < 6) json += ",";
        }
        json += "],";
        json += " "vias ":[";
        for (int i = 0; i < 7; i++) {
                json += String(vias[i] ? "true": "false");
                if (i < 6) json += ",";
        }
        json += "]}";
        server.send(200, "application/json", json);
}

// Função para testar as vias
void TestaVia(int pinos) {
        auxV = 0;            // Reseta contador de vias sem defeitos
        auxCC = 0;           // Reseta contador de vias sem curto circuito
        memset(vias, false, sizeof(vias));          // Reseta vias
        memset(CC, false, sizeof(CC));           // Reseta CC
        for (int i = 0; i < pinos; i++) {
                // Envia 5V de um lado do cabo
                digitalWrite(saidas[i], HIGH);
```

```
            // Confere se chega 5V no outro lado
            if (digitalRead(entradas[i]) == HIGH) {
                vias[i] = true;
                auxV++;
            }
            aux = 0;            // Reseta contador de curto circuitos em cada via

            // Confere se não há curto circuito
            for (int j = 0; j < pinos; j++) {
                if (entradas[j] != entradas[i]) {
                    if (digitalRead(entradas[j]) == LOW) {
                        aux++;
                    }
                }
            }

            // Conta quantos vias sem curto circuitos
            if (aux == (pinos - 1)) {
                CC[i] = true;
                auxCC++;
            }

            // Deixa de mandar os 5V
            digitalWrite(saidas[i], LOW);
        }

        // Verifica se todas as vias estão montadas corretamente
        if (auxV == pinos && auxCC == pinos) {
            isOK = true;
        }
        else {
```

```
                isOK = false;
        }

        // Caso cabo esteja OK, acende led verde
        if (isOK) {
                digitalWrite(LED_VM, LOW);
                digitalWrite(LED_VD, HIGH);
        }

        // Caso cabo não esteja OK, acende o led vermelho
        else {
                digitalWrite(LED_VD, LOW);
                digitalWrite(LED_VM, HIGH);
        }
        Serial.println("isOK: ");
        Serial.print(isOK);
        Serial.println("vias: ");
        for (int l = 0; l < 7; l++){
                Serial.print(vias[l]);
        }
}

void setup() {
        Serial.begin(115200);

        // Inicializa o SPIFFS
        if (!SPIFFS.begin(true)) {
                Serial.println("Erro ao montar SPIFFS");
                return;
        }
```

```
// Conecta ao Wi-Fi
WiFi.mode(WIFI_STA);
WiFi.begin("rede", "senha");
WiFi.config(IPAddress(192, 168, 15, 39), IPAddress(255, 255, 255, 0), IPAddress(192, 168, 15, 1));
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando ao WiFi...");
}
Serial.println("Conectado ao WiFi!");
Serial.print("IP: ");
Serial.println(WiFi.localIP());

// Configura o servidor web
server.on("/", handleRoot);
server.on("/select", HTTP_POST, handleSelect);
server.on("/resultado", HTTP_GET, handleResultado);
server.begin();
server.on("/style.css", HTTP_GET, []() {
    File file = SPIFFS.open("/style.css", "r");
    server.streamFile(file, "text/css");
    file.close();
});
server.on("/Moba-Logo-blk-_1_.webp", HTTP_GET, []() {
    File file = SPIFFS.open("/Moba-Logo-blk-_1_.webp", "r");
    server.streamFile(file, "image/webp");
    file.close();
});
server.on("/load.gif", HTTP_GET, []() {
    File file = SPIFFS.open("/load.gif", "r");
```

```
                server.streamFile(file, "image/gif");
                file.close();
        });

// Saídas
pinMode(PIN_1, OUTPUT);
pinMode(PIN_2, OUTPUT);
pinMode(PIN_3, OUTPUT);
pinMode(PIN_4, OUTPUT);
pinMode(PIN_5, OUTPUT);
pinMode(PIN_6, OUTPUT);
pinMode(PIN_7, OUTPUT);

// Entradas
pinMode(PIN_8, INPUT_PULLDOWN);
pinMode(PIN_9, INPUT_PULLDOWN);
pinMode(PIN_10, INPUT_PULLDOWN);
pinMode(PIN_11, INPUT_PULLDOWN);
pinMode(PIN_12, INPUT_PULLDOWN);
pinMode(PIN_13, INPUT_PULLDOWN);
pinMode(PIN_14, INPUT_PULLDOWN);

// Led Verde
pinMode(LED_VD, OUTPUT);

// Led Vermelho
pinMode(LED_VM, OUTPUT);
}
void loop() {
        server.handleClient();
        if (cabo == "55-04-02-02620| cabo == "55-04-02-02621") {
```

```
            TestaVia(7);            // Chama função testar vias
            delay(2000);
            cabo = "" ;             // Reseta variável
            digitalWrite(LED_VD, LOW);          //Reseta LED
            digitalWrite(LED_VM, LOW);          //Reseta LED
            server.send(200, "text/html", «script>atualizarResultado();</script>");
    }
    else if (cabo == "55-04-02-02566 | cabo == "55-04-02-02561") {
            TestaVia(4);            // Chama função testar vias
            delay(2000);
            cabo = "";              // Reseta variável
            digitalWrite(LED_VD, LOW);          //Reseta LED
            digitalWrite(LED_VM, LOW);          //Reseta LED
            server.send(200, "text/html", «script>atualizarResultado();</script>");
    }
    else if (cabo == "55-04-02-02624") {
            auxV = 0;
            auxCC = 0;

            // Via 1
            digitalWrite(PIN_1, HIGH);          //Envia 5V
            Serial.println("Saida 1");
            Serial.print(digitalRead(PIN_1) == HIGH ? "HIGH": "LOW");
            delay(800);

            // Confere se chega os 5V
            if (digitalRead(PIN_8) == HIGH) {
                    vias[0] = true;
                    auxV = auxV + auxV;
            }
```

```
                Serial.println("Entrada 1");
                Serial.print(digitalRead(PIN_8) == HIGH ? "HIGH": "LOW");
                delay(800);

                // Confere se há curto circuitos
                if (digitalRead(PIN_9) == LOW && digitalRead(PIN_10) == LOW &&
digitalRead(PIN_11) == LOW && digitalRead(PIN_12) == LOW && digitalRead(PIN_13) ==
LOW && digitalRead(PIN_14) == LOW) {
                        CC[0] = true;
                        auxCC = auxCC + auxCC;
                }

                digitalWrite(PIN_1, LOW);           // Deixa de enviar os 5V

                // Via 2
                digitalWrite(PIN_2, HIGH);           //Envia 5V
                Serial.println("Saida 2");
                Serial.print(digitalRead(PIN_2) == HIGH ? "HIGH": "LOW");
                delay(800);

                // Confere se chega os 5V
                if (digitalRead(PIN_9) == HIGH && digitalRead(PIN_11) == HIGH) {
                        vias[1] = true;
                        auxV = auxV + auxV;
                }
                Serial.println("Entrada 2");
                Serial.print(digitalRead(PIN_9) == HIGH ? "HIGH": "LOW");
                delay(800);
                Serial.print(digitalRead(PIN_11) == HIGH ? "HIGH": "LOW");
                delay(800);

                // Confere se há curto circuitos
```

```
                if (digitalRead(PIN_8) == LOW && digitalRead(PIN_10) == LOW &&
digitalRead(PIN_12) == LOW && digitalRead(PIN_13) == LOW && digitalRead(PIN_14) ==
LOW) {
                        CC[1] = true;
                        auxCC = auxCC + auxCC;
                }

                digitalWrite(PIN_2, LOW);           // Deixa de enviar os 5V

                // Via 3
                digitalWrite(PIN_3, HIGH);           //Envia 5V
                Serial.println("Saida 3");
                Serial.print(digitalRead(PIN_3) == HIGH ? "HIGH": "LOW");
                delay(800);

                // Confere se chega os 5V
                if (digitalRead(PIN_10) == HIGH) {
                        vias[2] = true;
                        auxV = auxV + auxV;
                }
                Serial.println("Entrada 3");
                Serial.print(digitalRead(PIN_10) == HIGH ? "HIGH": "LOW");
                delay(800);

                // Confere se há curto circuitos
                if (digitalRead(PIN_8) == LOW && digitalRead(PIN_9) == LOW &&
digitalRead(PIN_11) == LOW && digitalRead(PIN_12) == LOW && digitalRead(PIN_13) ==
LOW && digitalRead(PIN_14) == LOW) {
                        CC[2] = true;
                        auxCC = auxCC + auxCC;
                }
```

```
                digitalWrite(PIN_3, LOW);            // Deixa de enviar os 5V

                // Via 4
                digitalWrite(PIN_4, HIGH); //Envia 5V
                Serial.println("Saida 4");
                Serial.print(digitalRead(PIN_4) == HIGH ? "HIGH": "LOW");
                delay(800);

                // Confere se chega os 5V
                if (digitalRead(PIN_9) == HIGH && digitalRead(PIN_11) == HIGH) {
                        vias[3] = true;
                        auxV = auxV + auxV;
                }
                Serial.println("Entrada 4");
                Serial.print(digitalRead(PIN_9) == HIGH ? "HIGH": "LOW");
                delay(800);
                Serial.print(digitalRead(PIN_11) == HIGH ? "HIGH": "LOW");
                delay(800);

                // Confere se há curto circuitos
                if (digitalRead(PIN_8) == LOW && digitalRead(PIN_10) == LOW &&
digitalRead(PIN_12) == LOW && digitalRead(PIN_13) == LOW && digitalRead(PIN_14) ==
LOW) {
                        CC[3] = true;
                        auxCC = auxCC + auxCC;
                }

                digitalWrite(PIN_4, LOW);            // Deixa de enviar os 5V

                // Via 5
                digitalWrite(PIN_5, HIGH);           //Envia 5V
                Serial.println("Saida 5");
```

```
Serial.print(digitalRead(PIN_5) == HIGH ? "HIGH": "LOW");
delay(800);

// Confere se chega os 5V
if (digitalRead(PIN_12) == HIGH) {
        vias[4] = true;
        auxV = auxV + auxV;
}
Serial.println("Entrada 5");
Serial.print(digitalRead(PIN_12) == HIGH ? "HIGH": "LOW");
delay(800);

// Confere se há curto circuitos
if (digitalRead(PIN_8) == LOW && digitalRead(PIN_9) == LOW &&
digitalRead(PIN_10) == LOW && digitalRead(PIN_11) == LOW && digitalRead(PIN_13) ==
LOW && digitalRead(PIN_14) == LOW) {
        CC[4] = true;
        auxCC = auxCC + auxCC;
}

digitalWrite(PIN_5, LOW);          // Deixa de enviar os 5V

// Via 6
digitalWrite(PIN_6, HIGH);          //Envia 5V
Serial.println("Saida 6");
Serial.print(digitalRead(PIN_6) == HIGH ? "HIGH": "LOW");
delay(800);

// Confere se chega os 5V
        if (digitalRead(PIN_13) == HIGH) {
        vias[5] = true;
        auxV = auxV + auxV;
```

```
}
Serial.println("Entrada 6");
Serial.print(digitalRead(PIN_13) == HIGH ? "HIGH": "LOW");
delay(800);

// Confere se há curto circuitos
if (digitalRead(PIN_8) == LOW && digitalRead(PIN_9) == LOW &&
digitalRead(PIN_10) == LOW && digitalRead(PIN_11) == LOW && digitalRead(PIN_12) ==
LOW && digitalRead(PIN_14) == LOW) {
        CC[5] = true;
        auxCC = auxCC + auxCC;
}

digitalWrite(PIN_6, LOW);          // Deixa de enviar os 5V

// Via 7
digitalWrite(PIN_7, HIGH);          //Envia 5V
Serial.println("Saida 7");
Serial.print(digitalRead(PIN_7) == HIGH ? "HIGH": "LOW");
delay(800);

// Confere se chega os 5V
if (digitalRead(PIN_14) == HIGH) {
        vias[6] = true;
        auxV = auxV + auxV;
}
Serial.println("Entrada 7");
Serial.print(digitalRead(PIN_14) == HIGH ? "HIGH": "LOW");
delay(800);

// Confere se há curto circuitos
if (digitalRead(PIN_8) == LOW && digitalRead(PIN_9) == LOW &&
```

```
digitalRead(PIN_10) == LOW && digitalRead(PIN_11) == LOW && digitalRead(PIN_12) ==
LOW && digitalRead(PIN_13) == LOW) {
                CC[6] = true;
                auxCC = auxCC + auxCC;
        }

        digitalWrite(PIN_7, LOW);           // Deixa de enviar os 5V

        // Caso chegue 5V em todas as vias acende led verde
        if (auxV == 7 && auxCC == 7) {
                digitalWrite(LED_VM, LOW);
                digitalWrite(LED_VD, HIGH);
                isOK = true;
        }
        else {
                // Caso alguma via não tenha 5V, acende o led vermelho
                digitalWrite(LED_VD, LOW);
                digitalWrite(LED_VM, HIGH);
                isOK = false;
        }
        Serial.println("isOK: ");
        Serial.print(isOK);
        Serial.println("vias: ");
        for (int l = 0; l < 7; l++){
                Serial.print(vias[l]);
        }
        delay(2000);
        cabo = "";          // Reseta variável
        digitalWrite(LED_VD, LOW);          //Reseta LED
        digitalWrite(LED_VM, LOW);          //Reseta LED
        server.send(200, "text/html", «script>atualizarResultado();</script>");
```

```
            }
        }
```

APÊNDICE C – CÓDIGO SERVIDOR *WEB*.

C.1 HTML e JavaScript

```
<!DOCTYPE html>
<html lang="pt-br" >

<head>
<meta charset="UTF-8" >
<meta name="viewport"content="width=device-width, initial-scale=1.0" >
<link rel="stylesheet"href="style.css" >
<title>Cable Tests</title>
</head>
<body>
<div class="container" >
<nav>
<div class="cabecalho" >
<div class="logo" >
<img src="Moba-Logo-blk-_1_.webp"alt="Logo MOBA do Brasil" >
</div>

<div class="titulo" >
<h1>MOBA Service Web Application</h1>
<h2>MOBA Cable Tests Plataform</h2>
</div>

</div>

<div class="teste" >
<div class="escolha" >
<form action="/select"method="POST" >
<label for="cabo" >Cabo:</label>
```

```
<select name="cabo"id="cabo" >
<option value="55-04-02-02620»55-04-02-02620</option>
<option value="55-04-02-02621»55-04-02-02621</option>
<option value="55-04-02-02566»55-04-02-02566</option>
<option value="55-04-02-02561»55-04-02-02561</option>
<option value="55-04-02-02624»55-04-02-02624</option>
</select>

<input type="submit"value="Testar" >

</form>
</div>

<div class="resultado" >
<img id="loadingGif"src="load.gif"style="display: none;" >
<p id="status»</p>
<p id="vias»</p>
</div>
</div>

<script>

document.querySelector("form").addEventListener("submit", function(event) {
event.preventDefault(); // Impede o reload da página

let statusElem = document.getElementById("status");
let viasElem = document.getElementById("vias");
let loadingGif = document.getElementById("loadingGif");

// Apaga o resultado anterior
if (statusElem) statusElem.innerText = ¨̃¨;
if (viasElem) viasElem.innerText = ¨̃¨;
```

```
// Exibe o GIF de carregamento
if (loadingGif) loadingGif.style.display = "block";

// Envia o teste para o servidor
fetch("/select", {
method: "POST",
body: new FormData(this)
})
.then(response => {
if (!response.ok) {
throw new Error("Erro ao enviar teste.");
}
console.log("Teste enviado com sucesso. Aguardando resultado...");
setTimeout(atualizarResultado, 3000); // Aguarda 3s para buscar resultado
})
.catch(error => {
console.error("Erro ao enviar o teste:", error);
let loadingGif = document.getElementById("loadingGif");
if (loadingGif) loadingGif.style.display = "none";
});

});

function atualizarResultado() { document.getElementById("loadingGif").style.display =
"block";

fetch("/resultado")
.then(response => response.json())
.then(data => { document.getElementById("loadingGif").style.display = "none";
```

```
// Exibir status do teste document.getElementById("status").innerText = data.isOK ? "Aprovado ": "Reprovado ";

if (!data.isOK) {
// Obter o cabo selecionado
let caboSelecionado = document.getElementById("cabo").value;

// Mapas de conversão de vias para letras
const mapaVias = {
"55-04-02-02620": ["A", "B", "C", "D", "E", "F", "G"],
"55-04-02-02621": ["A", "B", "C", "D", "E", "F", "G"],
"55-04-02-02624": ["A", "B", "C", "D", "E", "F", "G"],
"55-04-02-02566": ["A ou M", "B ou L", "C ou K", "D ou J"],
"55-04-02-02561": ["A ou M", "B ou L", "C ou K", "D ou J"]
};

// Obter o mapa correto para o cabo atual
let mapaAtual = mapaVias[caboSelecionado] || [];

// Filtrar apenas os índices onde as vias e CC estão com problema
let viasComProblema = data.vias
.map((v, i) => v ? null : mapaAtual[i]) // Mapeia para a letra correspondente
.filter(v => v !== null);

let ccComProblema = data.CC
.map((c, i) => c ? null : mapaAtual[i]) // Mapeia para a letra correspondente
.filter(c => c !== null);

document.getElementById("vias").innerText = "Vias com problema: "+ (viasComProblema.length > 0 ? viasComProblema.join(", ") : "Nenhuma");

} else { document.getElementById("vias").innerText = ;
```

```
}
})
.catch(error => console.error("Erro ao atualizar resultado:", error));
}

</script>

</nav>
</div>
<footer class="footer-info" >
<a href='https://www.moba-automation.com.br'><img src="Moba-Logo-blk-_1_.webp"alt="MOBA"
logo"/></a>
<p>
<a href="mailto:lmoreira@moba.de»lmoreira@moba.de</a>
<br />
<small>&copy; 2025 MOBA do Brasil. All Rights Reserved.</small>
</p>
</footer>
</body>
</html>
```

C.2 CSS

```
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}

body {
background-color: rgb(223, 223, 223);
min-height: 100vh;
```

```
font-family: Arial, sans-serif;
display: flex;
flex-direction: column;
align-items: center;
}

/* Container principal */
.container {
width: 90
max-width: 1200px;
}

/* Cabeçalho */
.cabecalho {
display: flex;
align-items: center;
background-color: rgb(252, 227, 3);
border-bottom: 1px solid 333;
margin-left: -75px;
width: 99.5vw;
padding: 20px;
justify-content: space-between;
}

.cabecalho img {
width: 200px;
height: auto;
}

.titulo {
text-align: center;
```

```
flex-grow: 1; /* Faz o título ocupar espaço centralizado */
}

.titulo h1 {
font-size: 28px; /* Redução para melhorar a responsividade */
font-weight: bold;
}

.titulo h2 {
font-size: 16px;
font-weight: 100;
}

/* Teste */
.teste {
display: flex;
flex-direction: column;
align-items: center;
width: 100
max-width: 600px;
margin: 40px auto;
padding: 20px;
background-color: white;
border: 1px solid 333;
border-radius: 8px;
box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
}

.teste select,
.teste input {
margin-top: 10px;
```

```
width: 100
max-width: 250px;
height: 35px;
font-size: 16px;
text-align: center;
}

input[type="submit"] {
background-color: 333;
color: white;
border: none;
cursor: pointer;
transition: 0.3s;
}

input[type="submit"]:hover {
background-color: 555;
}

/* Resultado */
.resultado {
margin-top: 20px;
text-align: center;
}

.resultado img {
width: 150px;
display: none;
}

/* Rodapé */
```

```
footer.footer-info {
border-top: 1px solid 333;
width: 100
padding: 10px;
text-align: center;
background-color: rgb(223, 223, 223);
position: relative; /* Alterado de 'fixed' para 'relative' */
bottom: 0;
}

.footer-info img {
width: 100px;
margin-bottom: 5px;
}

loadingGif {
display: none;
}

.footer-info a {
text-decoration: none;
color: black;
}

/*  MEDIA QUERIES - Ajustes para tablets e celulares */
@media (max-width: 1024px) {
.cabecalho {
flex-direction: column;
align-items: center;
width: 900px;
}
```

```
.cabecalho img {
width: 150px;
}

.titulo h1 {
font-size: 22px;
}

.titulo h2 {
font-size: 12px;
}

.teste {
width: 900px;
}

.resultado img {
width: 140px;
}

footer.footer-info {
width: 910px;
}
}
/* Telas pequenas - Celulares */
@media (max-width: 600px) {
.cabecalho {
align-items: center;
width: 450px;
}
```

```
.cabecalho img {
width: 120px;
}

.titulo h1 {
font-size: 18px;
}

.titulo h2 {
font-size: 10px;
}

.teste {
width: 100
padding: 15px;
}

.resultado img {
width: 110px;
}

input[type="submit"] {
font-size: 14px;
}

.footer-info img {
width: 80px;
}

footer.footer-info {
width: 450px;
```

```
}
}
/* Ajuste do corpo para ocupar toda a altura da tela */
html, body {
height: 100
display: flex;
flex-direction: column;
}

/* O container principal cresce para empurrar o rodapé para baixo */
.container {
flex: 1;
}
```