INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAJAZEIRAS

CARLOS HENRIQUE SIBALDO TORRES DE LIRA
JOÃO PAULO SIBALDO TORRES DE LIRA

SISTEMA DE CONTROLE DE VELOCIDADE PARA MOTOR CC COM SUPERVISÓRIOS LOCAL VIA MQTT E REMOTO EM NUVEM: ANÁLISE DE MÉTODOS DE SINTONIA

Cajazeiras-PB
2025

CARLOS HENRIQUE SIBALDO TORRES DE LIRA
JOÃO PAULO SIBALDO TORRES DE LIRA

SISTEMA DE CONTROLE DE VELOCIDADE PARA MOTOR CC COM SUPERVISÓRIOS LOCAL VIA MQTT E REMOTO EM NUVEM: ANÁLISE DE MÉTODOS DE SINTONIA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Bacharelado em Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba-*Campus* Cajazeiras, como parte dos requisitos para a obtenção do Título de Bacharel em Engenharia de Controle e Automação, sob Orientação do Prof. Emanoel Raimundo Queiroz Chaves Júnior e Coorientação do Prof. Gerônimo Barbosa Alexandre.

Cajazeiras-PB
2025

IFPB / Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Cícero Luciano Félix CRB-15/750

L768s Lira, Carlos Henrique Sibaldo Torres de.
Sistema de controle de velocidade para motor CC com supervisórios local via MQTT remoto em nuvem : análise de métodos de sintonia / Carlos Henrique Sibaldo Torres de Lira, João Paulo Sibaldo Torres de Lira. – Cajazeiras, 2025.
102f. : il.

Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2025.

Orientador: Prof. Emanoel Raimundo Queiroz Chaves Júnior.
Coorientador: Prof. Gerônimo Barbosa Alexandre.

1. Engenharia de controle e automação. 2. Sistemas de controle. 3. Motor de corrente contínua. 4. Internet das coisas. I. Lira, João Paulo Sibaldo Torres de. II. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. III. Título.

IFPB/CZ CDU: 681.511.26(043.2)

CARLOS HENRIQUE SIBALDO TORRES DE LIRA
JOÃO PAULO SIBALDO TORRES DE LIRA

SISTEMA DE CONTROLE DE VELOCIDADE PARA MOTOR CC COM SUPERVISÓRIOS LOCAL VIA MQTT E REMOTO EM NUVEM: ANÁLISE DE MÉTODOS DE SINTONIA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Bacharelado em Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, *Campus* Cajazeiras, como parte dos requisitos para a obtenção do Título de Bacharel em Engenharia de Controle e Automação.

Aprovado em _____ de _____ de _____.

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br
EMANOEL RAIMUNDO QUEIROZ CHAVES JUNIO
Data: 09/09/2025 11:21:50-0300
Verifique em https://validar.iti.gov.br

_____
Emanoel Raimundo Queiroz Chaves Júnior – IFPB-*Campus* Cajazeiras
Orientador

Documento assinado digitalmente
gov.br
Geronimo Barbosa Alexandre
Data: 09/09/2025 14:50:00-0300
Verifique em https://validar.iti.gov.br

_____
Gerônimo Barbosa Alexandre – IFPB-Campus Cajazeiras
Coorientador

Documento assinado digitalmente
gov.br
JAILTON FERREIRA MOREIRA
Data: 09/09/2025 15:02:18-0300
Verifique em https://validar.iti.gov.br

_____
Jaílton Ferreira Moreira – IFPB-*Campus* Cajazeiras
Examinador 1

Documento assinado digitalmente
gov.br
HEITOR MEDEIROS FLORENCIO
Data: 09/09/2025 12:43:45-0300
Verifique em https://validar.iti.gov.br

_____
Heitor Medeiros Florêncio – IMD/UFRN
Examinador 2

Dedicamos este trabalho a Deus, por sua graça, e às nossas famílias, por serem nosso alicerce em cada etapa desta conquista.

AGRADECIMENTOS

Por João Paulo Sibaldo Torres de Lira

Primeiramente, agradeço a Deus, ser supremo e criador de todas as coisas, por me conceder forças nos momentos de dificuldade, iluminar meu caminho e me abençoar com a persistência necessária para enfrentar os desafios ao longo desta caminhada.

À minha mãe, Irisvânia Bispo de Lira, meu pilar e principal incentivadora. Seu amor, dedicação e apoio incondicional foram fundamentais desde o início da minha jornada acadêmica, especialmente por todo o suporte oferecido mesmo à distância, desde os tempos do curso técnico em Eletrotécnica. Seus gestos de carinho e acolhimento, mesmo nas noites em que estive exausto de tanto estudar, jamais serão esquecidos.

À minha tia, Betânia Maria Farias de Lira, que representou uma verdadeira segunda mãe em minha vida. Seu apoio, cuidado e presença constante foram essenciais para que eu pudesse seguir firme neste percurso.

Aos meus avós, Job Bispo de Lira e Áurea Maria de Lira, por fazerem parte dessa rede de afeto e suporte que tanto me fortaleceu. Cada palavra de incentivo e cada gesto de carinho contribuíram significativamente para que eu chegasse até aqui.

Ao meu pai, José Edijânio Sibaldo Torres, cuja trajetória me inspira e cujo apoio foi imprescindível. Ao longo de todo esse processo, compartilhamos aprendizados e superações que marcaram profundamente minha formação pessoal e acadêmica.

Ao meu irmão, Carlos Henrique Sibaldo Torres de Lira, por estar sempre ao meu lado, acompanhando de perto essa trajetória e sendo presença constante nos momentos bons e difíceis.

Aos professores Emanoel Raimundo Queiroz Chaves Júnior e Gerônimo Barbosa Alexandre, pela orientação atenciosa e valiosa prestada durante o desenvolvimento deste trabalho. Sua dedicação e conhecimento foram fundamentais para a concretização deste projeto.

Aos demais professores do curso de Engenharia de Controle e Automação do IFPB – Campus Cajazeiras, pela excelência no ensino, pelos conhecimentos compartilhados e pelo comprometimento com a formação de cada aluno. Carregarei comigo cada ensinamento.

E, por fim, aos amigos que fiz ao longo dessa jornada no IFPB, pelo companheirismo, pelas risadas, pelos desafios enfrentados juntos e pelos momentos inesquecíveis que tornaram essa caminhada mais leve e significativa.

A todos, minha eterna gratidão.

AGRADECIMENTOS

Por Carlos Henrique Sibaldo Torres de Lira

Agradeço a Deus por renovar minhas forças em cada etapa desta caminhada acadêmica e iluminar meu caminho nos momentos de maior dúvida.

Agradeço à minha namorada, Ayrla Barbosa de Lira, pelo apoio emocional constante, pelo carinho e pela companhia nas horas de estudo que tornaram cada desafio mais leve e cada conquista mais significativa.

Minha mãe, Irisvânia Bispo de Lira, merece minha eterna gratidão por acreditar no meu potencial e por incentivar minha decisão de estudar longe de casa, tanto no curso técnico em Eletrotécnica quanto no bacharelado em Engenharia de Controle e Automação. Suas mensagens carinhosas e seus gestos de cuidado à distância foram alicerces para minha confiança.

Minha tia, Betânia Maria Farias de Lira, atuou como segunda mãe, sempre pronta a oferecer orientação afetuosa, motivação e apoio nos momentos mais desafiadores. Com sua presença, senti-me sempre amparado.

Agradeço aos meus avós, Job Bispo de Lira e Áurea Maria de Lira, pelo amor generoso, pelas histórias inspiradoras e pelos conselhos que fortaleceram minha determinação.

Meu irmão, João Paulo Sibaldo Torres de Lira, esteve ao meu lado em todas as turmas e etapas desta trajetória, compartilhando estudos, revisões e momentos de descontração que tornaram esta jornada mais leve e produtiva.

Sou grato ao meu pai, José Edijânio Sibaldo Torres, por ter sido exemplo profissional desde os primeiros passos no curso técnico em Eletrotécnica até a elaboração deste trabalho. Seu conhecimento prático orientou minhas escolhas e inspirou meu crescimento na área de controle e automação.

Aos professores Emanoel Raimundo Queiroz Chaves Júnior e Gerônimo Barbosa Alexandre, agradeço pelas orientações atenciosas e pelas sugestões que enriqueceram este projeto.

Registro minha gratidão ao corpo docente do IFPB Campus Cajazeiras, curso de Engenharia de Controle e Automação, pelas aulas práticas, pelos laboratórios desafiadores e pelo apoio constante.

Aos colegas e amigos de turma, agradeço pelas discussões produtivas, pelas horas de estudo em grupo e pelas risadas que tornaram cada dia especial. E a todos que, de perto ou de longe, contribuíram para minha formação pessoal e acadêmica, deixo minha sincera gratidão.

RESUMO

Este trabalho tem como objetivo comparar o desempenho de diferentes métodos de sintonia de controladores PID aplicados ao controle de velocidade de um motor de corrente contínua (CC). Para viabilizar essa análise, foi desenvolvido um sistema de controle digital implementado na plataforma ESP32 e integrado a duas interfaces de supervisão: uma local, construída com Node-RED e broker MQTT, e outra remota, desenvolvida por meio da plataforma *Firebase Studio* e hospedada na Vercel. O modelo matemático da planta (motor) foi identificado experimentalmente a partir da resposta ao degrau, permitindo sua representação como um sistema de primeira ordem com atraso. A partir dessa modelagem, aplicaram-se os métodos de Ziegler-Nichols, Cohen-Coon e Controle por Modelo Interno (IMC), cujos parâmetros foram inicialmente testados em simulações no ambiente Scilab/Xcos e, em seguida, validados em experimentos práticos. A análise considerou indicadores de desempenho como tempo de subida, tempo de acomodação e sobressinal, possibilitando avaliar de forma comparativa a eficácia de cada técnica de sintonia. Os resultados evidenciaram diferenças significativas no comportamento do sistema conforme o método adotado, destacando as vantagens e limitações de cada abordagem. Além disso, o sistema de controle desenvolvido demonstrou ser funcional, de baixo custo e aplicável a contextos de automação industrial, permitindo operação local e remota de forma integrada e em tempo real.

Palavras-chave: controle PID; métodos de sintonia; ESP32; supervisórios local e remoto; MQTT.

ABSTRACT

This work aims to compare the performance of different PID controller tuning methods applied to the speed control of a direct current (DC) motor. To enable this analysis, a digital control system was developed, implemented on the ESP32 platform and integrated with two supervisory interfaces: a local one, built with Node-RED and an MQTT broker, and a remote one, developed using the Firebase Studio platform and hosted by Vercel. The mathematical model of the plant (motor) was experimentally identified based on the step response, allowing its representation as a first-order system with delay. Based on this modeling, the Ziegler-Nichols, Cohen-Coon, and Internal Model Control (IMC) methods were applied, whose parameters were initially tested in simulations in the Scilab/Xcos environment and subsequently validated in practical experiments. The analysis considered performance indicators such as rise time, settling time, and overshoot, allowing a comparative evaluation of the effectiveness of each tuning technique. The results revealed significant differences in system behavior depending on the method adopted, highlighting the advantages and limitations of each approach. Furthermore, the developed control system proved to be functional, low-cost, and applicable to industrial automation contexts, enabling integrated local and remote operation in real time.

Keywords: PID control; tuning methods; ESP32; local and remote supervision; MQTT.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AD – Analógico-Digital.

API – *Application Programming Interface.*

CC – Corrente Contínua.

DA – Digital-Analógico.

DUP – Duplicata.

GPIO – *General Purpose Input/Output.*

HTTP – *Hypertext Transfer Protocol.*

IA – Inteligência Artificial.

IBM – *International Business Machines Corporation.*

iOS – *iPhone Operating System.*

IoT – *Internet of Things.*

JSON – *JavaScript Object Notation.*

LED – *Light Emitting Diode.*

MQTT – *Message Queuing Telemetry Transport.*

NoSQL – *Not Only SQL.*

PID – Proporcional, Integral e Derivativo.

Pub – *Publisher.*

PWM – *Pulse Width Modulation.*

QoS – *Quality of Service.*

RAM – *Random Access Memory.*

ROM – *Random Only Memory.*

Sub – *Subscriber.*

TCP – *Transmission Control Protocol.*

TCC – Trabalho de Conclusão de Curso.

USB – *Universal Serial Bus*.

LISTA DE SÍMBOLOS

K_p – Ganho proporcional.

T_i – Tempo integral.

T_d – Tempo derivativo.

E – Diferença entre o valor de referência e a variável controlada.

α – Fator do filtro derivativo.

K_i – Ganho integral.

K_d – Ganho derivativo.

T_t – Constante de tempo do anti-*reset wind-up*.

U_d – Sinal de controle desejado.

U – Sinal de controle saturado.

E_s – Erro de saturação.

T – Período de amostragem.

k – Tempo discreto.

n – Amostras.

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da automação industrial, o controle preciso de atuadores, como motores elétricos, tornou-se essencial em diversas aplicações industriais e acadêmicas. Dentre os diversos métodos de controle existentes, o controlador PID (Proporcional, Integral e Derivativo) destaca-se por sua simplicidade, robustez e aplicabilidade em sistemas de tempo continuo e discretos (Ogata, 2010).

Diante do crescimento da globalização, diversos setores da indústria passaram a demandar soluções mais eficientes para a integração entre dispositivos, com o objetivo de acompanhar o progresso tecnológico e aprimorar a gestão dos processos produtivos. Nesse contexto, emergiu o conceito de IoT (*Internet of Things* - Internet das Coisas), que propõe a conexão de sensores, dispositivos e unidades inteligentes capazes de interagir entre si de forma autônoma, ou seja, sem a necessidade de intervenção humana. Essa tecnologia possibilita que objetos heterogêneos sejam monitorados, controlados e compartilhem informações em tempo real, viabilizando a coleta, o processamento e a análise de dados gerados por sensores embutidos nos dispositivos, os quais são transmitidos por meio de uma rede de comunicação (Aguilar *et al.*, 2021).

Nesse cenário, torna-se fundamental a existência de protocolos padronizados de comunicação, capazes de viabilizar o intercâmbio de informações entre diferentes dispositivos conectados, assegurando a interoperabilidade do sistema (Freitas, 2017). Com esse propósito, surgiram protocolos específicos voltados para aplicações em Internet das Coisas, projetados para operar com eficiência em redes com largura de banda limitada e dispositivos com baixo poder de processamento. Entre esses protocolos destaca-se o MQTT (*Message Queuing Telemetry Transport* - Transporte de Telemetria para Enfileiramento de Mensagens), amplamente utilizado devido à sua leveza, simplicidade e eficiência na troca de mensagens entre dispositivos distribuídos.

O protocolo MQTT desenvolvido originalmente pela IBM (*International Business Machines* Corporation - Máquinas de Negócios Internacionais), adota uma arquitetura baseada no modelo publicador/assinante (*publish*/*subscribe*). Essa abordagem permite uma comunicação eficiente e desacoplada entre os dispositivos, tornando o protocolo especialmente adequado para ambientes com recursos limitados, como dispositivos com baixa capacidade de memória, processamento reduzido, largura de banda restrita e alta latência de rede (Aguilar et al., 2021). Graças à sua simplicidade, leveza e confiabilidade, o MQTT consolidou-se como uma das principais opções para aplicações em Internet das Coisas, com potencial para se tornar

um padrão de fato nesse contexto.

Neste trabalho, foi desenvolvido e analisado um sistema de controle PID digital aplicado ao controle de velocidade de um motor de Corrente Contínua (CC), utilizando a plataforma embarcada ESP32. A implementação do controlador foi realizada por meio da discretização do modelo contínuo, utilizando o método de Tustin, também conhecido como transformação bilinear, que garante uma boa aproximação da resposta dinâmica do sistema original no domínio discreto (Nise, 2012).

A identificação do modelo matemático da planta do sistema baseia-se no método da curva de reação de Ziegler-Nichols, utilizando a aplicação de um sinal em degrau no motor e o registro da resposta de velocidade ao longo do tempo. A partir dessa resposta, adota-se uma aproximação da dinâmica do motor como um modelo de primeira ordem com atraso puro, permitindo a aplicação das técnicas de sintonia, conforme descrito por Åström e Hägglund (1995). Esse modelo simplificado fornece parâmetros iniciais adequados para o projeto de controladores PID em sistemas industriais.

Com o modelo matemático da planta identificado, o projeto considera a determinação dos parâmetros do controlador PID a partir de três estratégias distintas: o método de Ziegler-Nichols, o método de Cohen-Coon e a técnica IMC (*Internal Model Control* – Controle por Modelo Interno). Esses parâmetros são utilizados em uma etapa de simulação computacional no ambiente Scilab/Xcos, permitindo a análise do comportamento dinâmico do sistema em termos de tempo de subida, tempo de acomodação e sobressinal. A implementação dos controladores ocorre em uma plataforma embarcada ESP32, que viabiliza a execução prática dos testes de controle. A comparação entre as respostas obtidas nas simulações e nos testes experimentais fornece subsídios para uma análise mais aprofundada do desempenho associado a cada técnica de sintonia adotada.

A comunicação entre o sistema embarcado e o ambiente de monitoramento adota uma arquitetura híbrida, composta por um supervisório local, desenvolvido no Node-RED, e um supervisório remoto, baseado em recursos de inteligência artificial na plataforma *Firebase Studio*, com hospedagem na Vercel. O supervisório local estabelece comunicação direta com a ESP32 por meio do protocolo MQTT, além de realizar a atualização contínua de um banco de dados em nuvem. O supervisório remoto, por sua vez, utiliza esse banco de dados para viabilizar o monitoramento e controle em tempo real, por meio de uma interface acessível via web. Essa estrutura possibilita o acompanhamento e a operação do sistema tanto em ambiente local quanto remoto, promovendo maior flexibilidade, integração e acessibilidade.

Dessa forma, este trabalho buscou não apenas demonstrar a implementação prática e eficiente de um controlador PID digital em uma plataforma de baixo custo, mas também explorou técnicas de identificação e sintonização de sistemas, além da integração com ferramentas modernas de conectividade, alinhando teoria e prática em um contexto atual de automação e IoT.

2 OBJETIVOS

Neste capítulo são apresentados os objetivos do Trabalho de Conclusão do Curso (TCC).

2.1 OBJETIVO GERAL

Desenvolver um sistema de controle de velocidade para motor CC com supervisórios local e remoto, utilizando comunicação MQTT e banco de dados em nuvem, e analisar o desempenho de diferentes métodos de sintonia.

2.2 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral, foram delineados os seguintes objetivos específicos:

- implementar o controle de velocidade para motor de corrente contínua utilizando a plataforma embarcada ESP32.
- desenvolver um supervisório local no Node-RED que se comunique com a ESP32 via protocolo MQTT e atualize o banco de dados em nuvem (*Realtime Database*).
- construir um supervisório remoto utilizando a plataforma *Firebase Studio* que monitore os dados e controle o motor via atualização do banco de dados em nuvem (*Realtime Database*).
- identificar o modelo matemático da planta (motor) a partir da curva de reação.
- realizar a sintonia dos parâmetros de controle utilizando os métodos de Ziegler-Nichols, Cohen-Coon e Modelo Interno (IMC).
- comparar e analisar o desempenho dos métodos de sintonia aplicados ao sistema de controle de velocidade.

3 REVISÃO DE LITERATURA

O avanço das tecnologias embarcadas e dos protocolos de comunicação tem possibilitado o desenvolvimento de sistemas de controle cada vez mais integrados e eficientes. Neste contexto, esta seção reúne os fundamentos teóricos essenciais ao projeto, com destaque para os métodos de sintonia de controladores PID, a técnica de discretização adotada, e os principais recursos empregados, como o microcontrolador ESP32, o ambiente Node-RED, a plataforma *Firebase*, o protocolo MQTT, o acionamento via PWM e o encoder.

3.1 ESP32

O ESP32 foi lançado em 2016 pela *Espressif Systems*, empresa dedicada ao desenvolvimento tecnológico. Este microcontrolador foi concebido para oferecer alta velocidade de processamento, destacando-se principalmente pela sua capacidade integrada de conectividade com redes Wi-Fi (Kolban, 2018 *apud* Soares, 2023).

3.1.1 Microcontrolador

O microcontrolador ESP32-D0WDQ6 contido no módulo ESP-WROOM-32, é um dispositivo programável de dimensão reduzida, baixo consumo energético e custo acessível. Sua estrutura básica compreende microprocessador, memórias RAM e ROM, pinagens de entrada e saída, *clock* e conversores Digital-Analógico (DA) e Analógico-Digital (AD). Estas características tornam os microcontroladores amplamente empregados em sistemas de controle remoto e aplicações embarcadas (Santos, 2019 *apud* Soares, 2023).

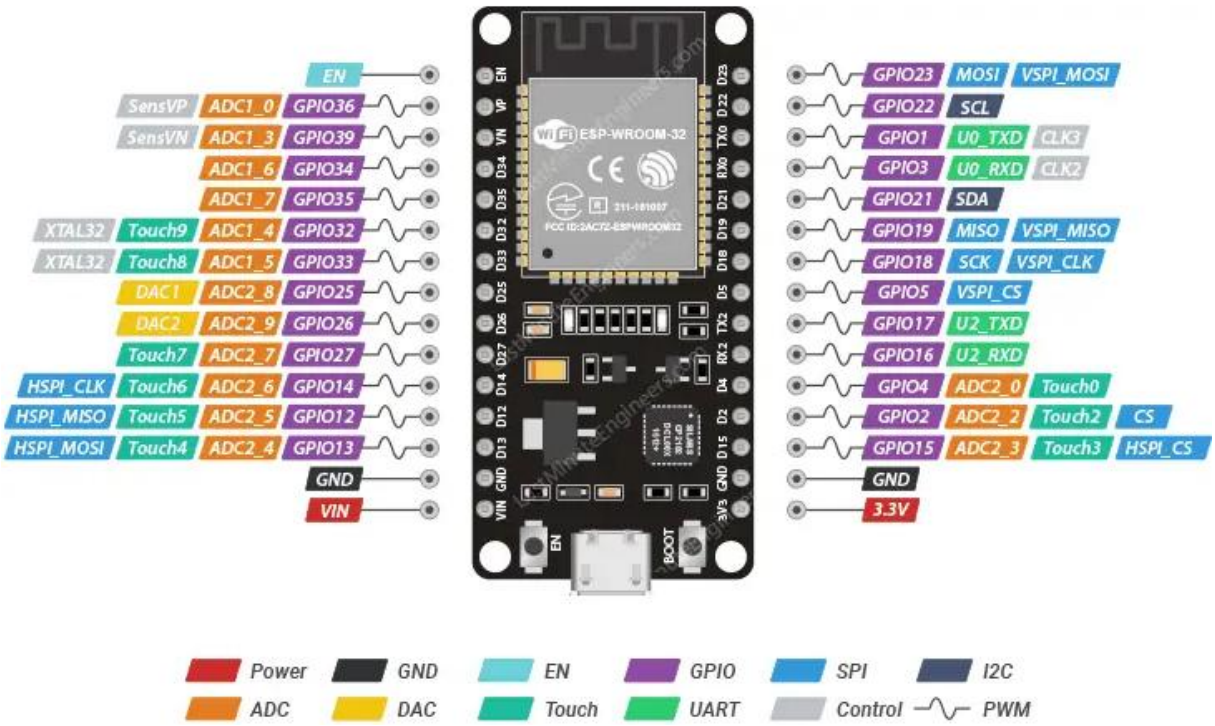
Com o avanço tecnológico dos dispositivos, surgiu a capacidade de conexão à internet, possibilitando comunicação e controle entre equipamentos diversos. Essa evolução permite o monitoramento remoto de dispositivos conectados que transmitem dados para determinados receptores. Tal funcionalidade oferece a possibilidade de realizar ajustes ou tomar decisões mais rapidamente com base na aquisição desses dados (Santos, 2019 *apud* Soares, 2023).

3.1.2 Pinagem do ESP32 DeviKit V1

Essa placa de desenvolvimento do ESP32 possui em sua totalidade 30 pinos como mostra a Figura 1, no entanto, são 25 pinos GPIO (*General Purpose Input/Output*) que podem ser configurados para diversas funções através de programação em registros específicos. Eles se dividem em categorias como somente digitais, analógicos e as que funcionam como sensores

capacitivos, com os dois últimos também funcionando como digitais quando configurados. A maioria destes pinos suporta configurações de *pull-up*, *pull-down* ou alta impedância, oferecendo flexibilidade para diferentes aplicações de hardware (Last Minute Engineers, s.d.).

Figura 1 – Pinagem do ESP32 com suas funcionalidades.



Fonte: Last Minute Engineers (s.d.).

3.2 NODE-RED

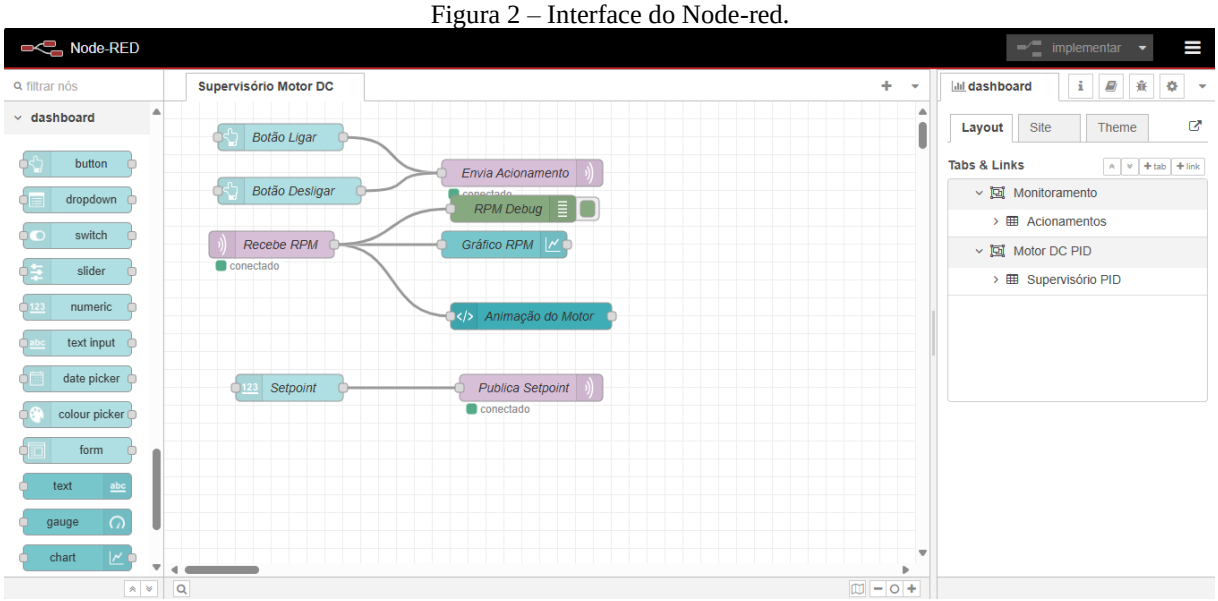
Desenvolver soluções IoT requer conhecimentos multidisciplinares em eletrônica, redes e programação. É um desafio realizar o gerenciamento de diversas conexões entre dispositivos embarcados, servidores e *smartphones* através de muitos serviços online. Junto a isso, é necessário criar aplicações para transmissão, controle e análise de dados coletados. Para facilitar este processo e atender às demandas específicas deste segmento tecnológico, foram desenvolvidas várias ferramentas especializadas, destancando-se entre elas o Node-RED, uma plataforma de programação visual que simplifica significativamente a integração dos componentes IoT e o processamento de seus fluxos de dados (Passe *et al.*, 2017).

De acordo com Passe *et al.* (2017), o Node-RED é uma ferramenta computacional de código aberto desenvolvida pela IBM, que permite a programação baseada em fluxos através de uma interface gráfica acessível diretamente pelo navegador. A plataforma tem vários nós com funções diferentes que podem ser conectados de forma simples, permitindo o fluxo de informações e a criação de aplicações. Cada nó tem uma função bem definida e, na maioria das

vezes, já cuida dos detalhes técnicos, facilitando a vida do programador. O Node-RED oferece muitas possibilidades de aplicação. É possível trabalhar com diferentes protocolos de rede, como HTTP, MQTT e TCP, além de estabelecer conexões locais via serial e integrar com bancos de dados. A plataforma também permite criar painéis de visualização de dados, com recursos como gráficos, tabelas e botões interativos, facilitando o monitoramento e o controle de sistemas. Além disso, é possível implementar funções personalizadas usando linguagens como *Python* e *JavaScript*, tudo diretamente pelo navegador.

No contexto deste trabalho, o Node-RED é utilizado como uma plataforma de supervisão para o controle de um motor CC. Através do seu *dashboard*, é possível monitorar a velocidade do motor em tempo real, ajustar o *setpoint* do controle PID e acionar ou desligar o motor diretamente pela interface. A comunicação entre o Node-RED e o motor é realizada via MQTT, um protocolo leve e eficiente, ideal para aplicações em IoT. Na Figura 2 mostra a interface do Node-RED com um exemplo de aplicação utilizando nós de *dashboards* para monitoração e controle de motor CC.

Figura 2 – Interface do Node-red.



Fonte: Autoria própria (2025).

Além disso, os fluxos criados na plataforma são armazenados em formato JSON, permitindo sua importação e exportação de forma simples. Isso possibilita compartilhar configurações entre projetos, realizar *backups* e migrar fluxos para outros dispositivos de maneira prática (Soares, 2024).

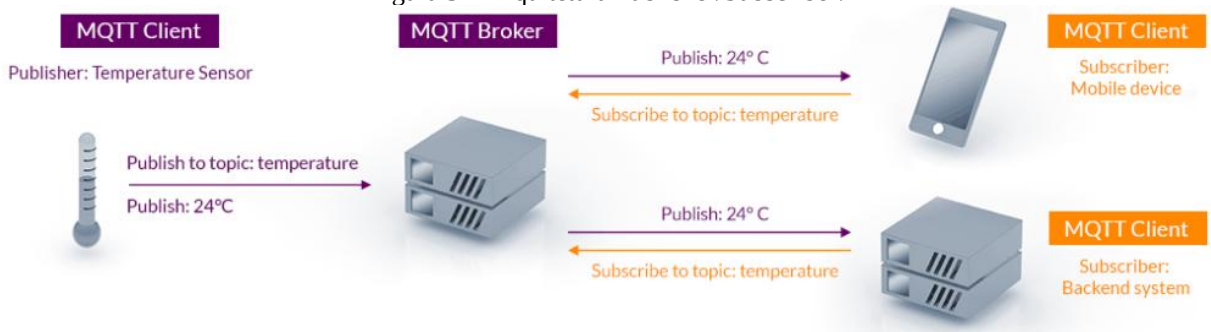
3.3 PROTOCOLO DE COMUNICAÇÃO MQTT

Em 1999, Andy Stanford-Clark, da IBM, e Arlen Nipper, da Arcom (atualmente Cirrus Link), desenvolveram o protocolo MQTT com o objetivo de minimizar o consumo de bateria e o uso de largura de banda durante a comunicação via satélite com oleodutos (Hivemq, 2024).

O funcionamento deste protocolo se baseia em um modelo orientado a eventos, seguindo o padrão *Publisher/Subscriber* (Pub/Sub). Neste modelo, o publicador é responsável por enviar informações, enquanto o assinante escolhe os tópicos que deseja receber. A comunicação é bidirecional, permitindo que diferentes dispositivos publiquem e se inscrevam em diferentes tópicos, e é intermediada por um *broker* MQTT, que gerencia as mensagens, filtrando-as e distribuindo-as corretamente. Entre os *brokers* mais utilizados, destacam-se o Mosquitto e o HiveMQ, ambos disponíveis em versões gratuitas para testes e aplicações reais (Soares, 2024).

Esta estrutura possibilita desacoplar o remetente do cliente, e dessa forma, basta apenas conhecer o endereço do broker para concretizar a comunicação entre os dispositivos. Na Figura 3 traz detalhes sobre a arquitetura MQTT.

Figura 3 – Arquitetura Publisher/Subscriber.



Fonte: Mqtt (2024).

Os tópicos são sequências de texto organizadas de forma hierárquica, semelhantes a caminhos de arquivos, onde as mensagens são publicadas. Dispositivos que desejam receber informações se inscrevem nesses tópicos como assinantes. A hierarquia dos tópicos permite organizar a comunicação de maneira clara e eficiente, pois é possível definir diferentes níveis de informação, separados por barras (/). Isso facilita o gerenciamento e o controle dos dados trocados entre os dispositivos conectados. Um exemplo de um tópico é: *myhome/kitchen/smartdishwasher* (Hivemq, 2024).

3.3.1 Níveis de qualidade de serviço (QoS)

De acordo com Freitas (2017), o protocolo MQTT possui três níveis de qualidade de serviço, que definem o grau de garantia na entrega das mensagens entre o remetente e o destinatário:

a) QoS 0:

-a mensagem é enviada sem confirmação de entrega, garantindo maior velocidade, mas sem certeza de que o destinatário recebeu.

b) QoS 1:

-a mensagem é entregue com garantia de recebimento, mas o destinatário pode recebê-la mais de uma vez;

c) QoS 2:

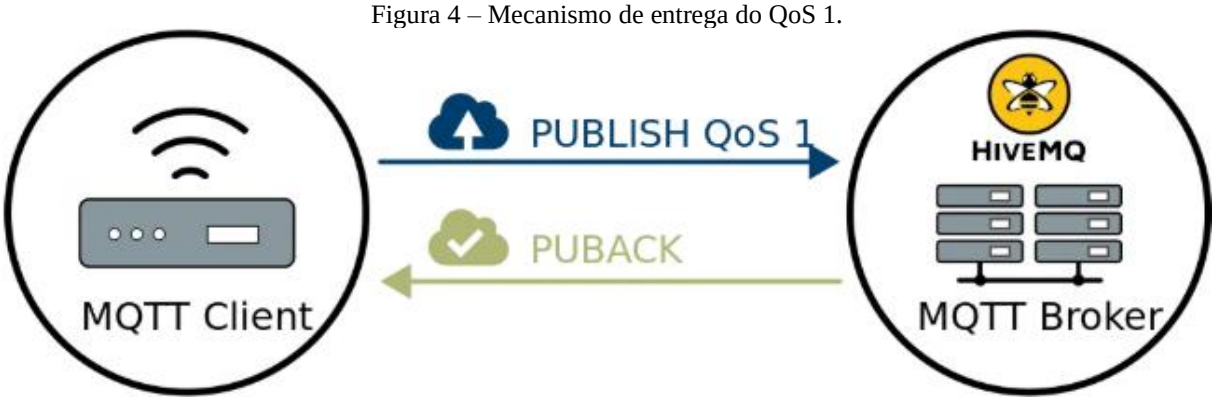
-o mais alto nível de garantia onde a mensagem é entregue uma única vez, sem duplicidade, embora seja o mais lento dos três.

Hivemq (2025) fornece detalhes sobre o mecanismo de entrega das mensagens no MQTT:

-o nível mais baixo oferece o melhor esforço na entrega da mensagem, no entanto, o destinatário não recebe garantia que a mensagem foi recebida e o remetente não a armazena ou retransmite.

-no nível 1, o publicador envia um pacote *Publish* e aguarda um *Puback* do receptor como confirmação de recebimento. Caso o *Puback* não chegue no tempo esperado, o publicador retransmite o mesmo pacote *Publish*, ajustando o sinalizador DUP para indicar duplicata. Esse mecanismo garante que a mensagem chegue ao *broker* pelo menos uma vez, embora possa gerar entregas duplicatas que devem ser tratadas pelo receptor. A Figura 4 ilustra esse mecanismo de entrega da mensagem.

Figura 4 – Mecanismo de entrega do QoS 1.

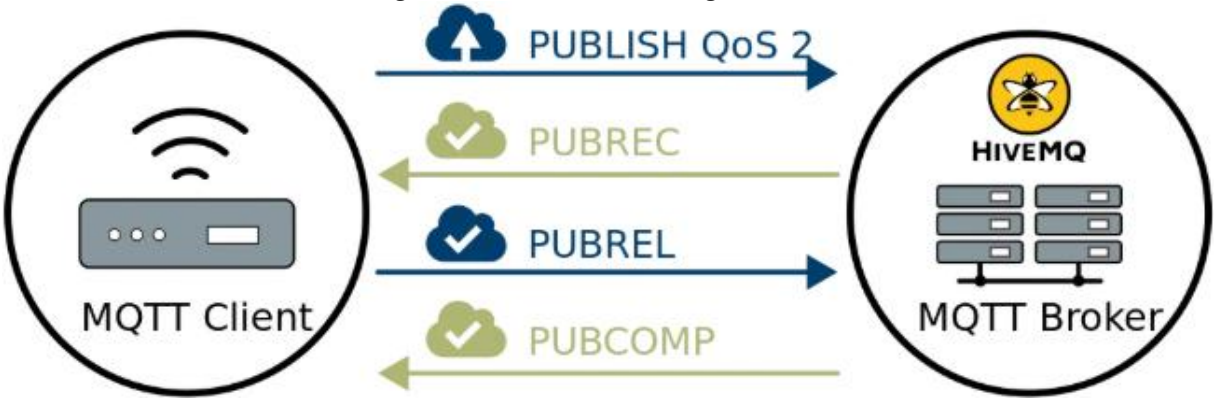


Fonte: Hivemq (2025).

-o QoS 2 apresenta um *handshake* de quatro etapas para assegurar a entrega da mensagem, como mostra na Figura 5. Incialmente o publicador envia o *Publish*, ao receber, o receptor envia um *Pubrec*, reconhecendo provisoriamente o pacote. Ao receber o *Pubrec*, o publicador enviar o *Pubrel*, indicando que pode liberar o pacote do seu *buffer*. O receptor, ao receber o *Pubrel*, processa a mensagem e retorna um *Pubcomp*, confirmando a conclusão do

handshake. Por fim, somente após o *Pubcomp* ambos liberam os seus estados, garantindo que a mensagem foi entregue exatamente uma vez.

Figura 5 – Mecanismo de entrega do QoS 2.



Fonte: Hivemq (2025).

3.3.2 Pacotes de mensagem MQTT

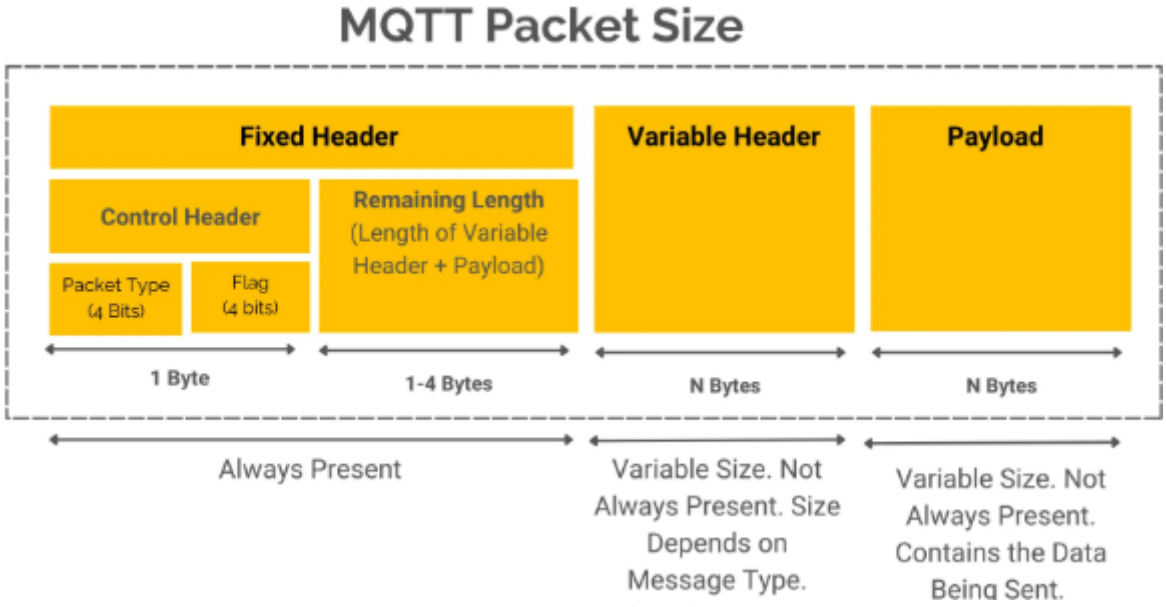
Segundo Hivemq (2024), no protocolo MQTT, os pacotes de mensagem são organizados de forma semelhante a um serviço postal inteligente, conforme ilustrado na Figura 6. Esses pacotes são compostos por três partes principais: o cabeçalho fixo, o cabeçalho variável e a carga útil.

O cabeçalho fixo (*Fixed headrer*) está sempre presente em todos os pacotes e contém informações essenciais, como o tipo e o tamanho da mensagem. Ele funciona como um envelope que identifica o tipo de mensagem e sua configuração básica.

O cabeçalho variável (*Variable header*) é opcional e aparece apenas quando necessário, oferecendo detalhes adicionais, como informações de roteamento e tratamento específico da mensagem. Sua presença depende do tipo de comunicação realizada.

Por fim, a carga útil (*Payload*) representa os dados reais que estão sendo transmitidos. É como o conteúdo de uma carta dentro do envelope, sendo a informação que o remetente deseja ao destinatário.

Figura 6 – Estrutura do pacote de dados MQTT.



Fonte: Hivemq (2024).

De acordo com o trabalho de Dias (2017), o cabeçalho fixo está presente em todos os pacotes e contém informações essenciais sobre o tipo e o controle do pacote. No caso específico do pacote *Publish*, ele possui três marcadores importantes: o DUP, que indica se o pacote é uma retransmissão; o QoS, que define o nível de qualidade da entrega (0, 1 ou 2); e o *RETAIN*, que determina se a mensagem será armazenada pelo *broker* para novos assinantes. A Figura 7 mostra a estrutura desse pacote, nela observa-se que os marcadores DUP, QoS e *RETAIN* ocupam os bits 3, 2 ,1 e 0, respectivamente.

Figura 7 – Formato de um cabeçalho fixo para um caso de pacote *Publish*.

Bit	7	6	5	4	3	2	1	0
Byte 1	Tipo de pacote MQTT				DUP	Nível QoS		RETAIN
Byte 2	Largura restante							

Fonte: Dias (2017).

3.4 FIREBASE

O *Firebase* é uma plataforma completa de desenvolvimento de aplicações, disponibilizada pela Google, que oferece um conjunto integrado de ferramentas e serviços voltados à criação, implantação e escalabilidade de aplicativos móveis, tanto para Android quanto iOS, além de aplicações *web*. Seu principal objetivo é simplificar o desenvolvimento, permitindo que os desenvolvedores foquem na experiência do usuário, enquanto a gestão da infraestrutura é controlada pela plataforma. Dentre seus principais recursos, destacam-se dois modelos de banco de dados na nuvem: o *Realtime Database*, que armazena dados em formato de árvore JSON e permite sincronização em tempo real entre usuários e dispositivos, e o

Firestore Database, que utiliza uma estrutura de documentos e coleções, oferecendo maior flexibilidade e escalabilidade. Além disso, a plataforma conta com serviços como autenticação de usuários, armazenamento em nuvem, hospedagem de aplicações e funções em nuvem que permitem a execução de códigos no lado do servidor sem necessidade de infraestrutura própria. Esses serviços atuam de forma integrada, proporcionando sincronização em tempo real, segurança robusta e alta escalabilidade, características essenciais para aplicações modernas (Geeksforgeeks, 2024).

3.4.1 Firebase Realtime Database

O *Firebase Realtime Database* é a base de dados utilizada para este trabalho, onde os dados são armazenados em nuvem e estruturado no formato JSON, cuja principal característica é a capacidade de sincronização em tempo real entre os dispositivos conectados a uma mesma instância da aplicação. Dessa forma, qualquer modificação nos dados é automaticamente propagada para todos os clientes vinculados, independentemente da plataforma utilizada, seja Android, iOS ou aplicações *web*. Esse modelo elimina a necessidade de requisições constantes aos servidores, otimizando o desempenho e garantindo a consistência das informações de forma contínua e eficiente (Firebase, 2025).

Essa ferramenta se trata de um banco de dados NoSQL em nuvem que oferece sincronização de dados em tempo real entre usuários e dispositivos. Esse serviço permite que as informações permaneçam disponíveis localmente, mesmo sem conexão com a internet, e sejam automaticamente sincronizadas assim que o dispositivo volta a estar online. Além disso, possui um sistema robusto de regras de segurança e controle de acesso, funcionando de forma integrada ao serviço de autenticação do *Firebase*. Por ser uma base NoSQL, sua modelagem de dados deve ser cuidadosamente estruturada, considerando como os usuários acessarão as informações, para garantir desempenho e escalabilidade (Firebase, 2025).

Características Principais:

Banco de Dados em Tempo Real: Sincronização instantânea dos dados entre todos os dispositivos conectados.

Funcionamento Offline: Os dados são armazenados localmente, permitindo que o aplicativo continue funcionando sem internet. As alterações são sincronizadas automaticamente ao restabelecer a conexão.

Resolução Automática de Conflitos: Durante a sincronização, o sistema combina alterações locais e remotas de forma segura.

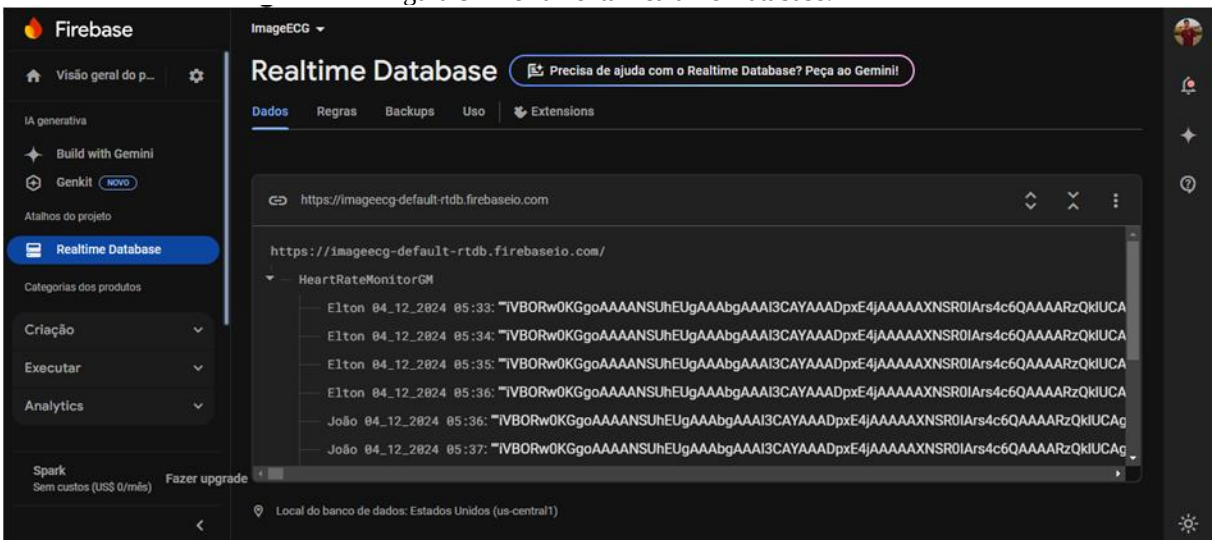
Regras de Segurança: Permite definir regras baseadas em expressões condicionais, controlando quem pode ler ou gravar dados, especialmente quando integrado ao *Firebase Authentication.*

Alta Escalabilidade: Otimizado para lidar com milhões de usuários simultaneamente, desde que a modelagem dos dados esteja alinhada com os padrões de acesso da aplicação.

Modelo NoSQL: Estrutura de dados em formato de árvore JSON, ideal para aplicações que exigem rapidez e atualização contínua, como mensagens, painéis de monitoramento, sistemas de automação, entre outros.

A Figura 8 apresenta a ferramenta *Realtime Database* aplicada em um projeto de Trabalho de Conclusão de Curso desenvolvido por Ramos (2025).

Figura 8 – Ferramenta Realtime Database.

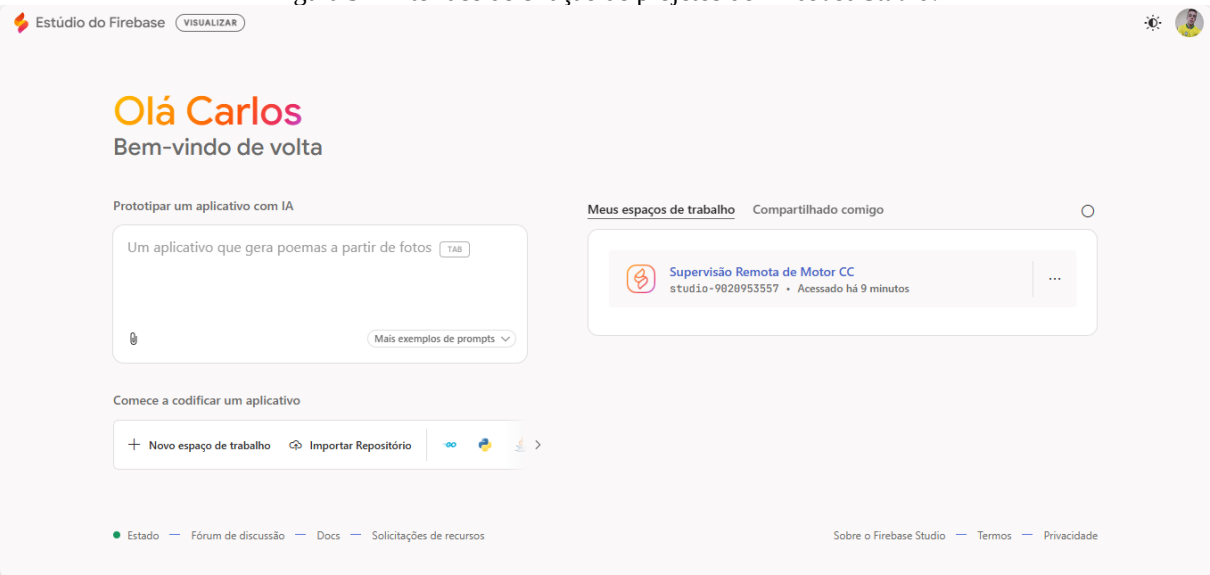


Fonte: Ramos (2025).

3.4.2 Firebase Studio

O *Firebase Studio* é uma plataforma de desenvolvimento baseada em nuvem que integra recursos de inteligência artificial conforme mostra na Figura 9, visando auxiliar na criação de aplicações completas, abrangendo desde APIs e *back-end* até *front-end* e aplicações móveis. A ferramenta combina o ambiente *Project IDX* com agentes de IA e o suporte do *Gemini*, proporcionando um espaço de trabalho colaborativo, acessível remotamente, e que centraliza todos os recursos necessários para o desenvolvimento de sistemas. Além disso, permite tanto a criação de novos projetos, a partir de modelos compatíveis com diversas linguagens e frameworks, quanto a importação de projetos já existentes, otimizando o fluxo de trabalho dos desenvolvedores (Firebase, 2025).

Figura 9 – Interface de criação de projetos do *Firebase Studio*.



Fonte: Autoria própria (2025).

No desenvolvimento deste trabalho, o *Firebase Studio* foi utilizado como ambiente de desenvolvimento para a criação da interface supervisória em plataforma web. Sua integração nativa com serviços como o *Realtime Database* facilitou a implementação da comunicação entre o sistema supervisório e o microcontrolador ESP32 via protocolo MQTT, além de permitir a construção de um sistema acessível remotamente e com gerenciamento simplificado de dados.

3.5 CONTROLADOR PID

O controlador PID avalia constantemente a diferença entre o valor desejado e o real, aplicando as três correções: proporcional, integral e derivativa para calcular a ação de controle necessária. Esta combinação permite uma resposta estável e eficiente, que se ajusta às necessidades particulares de cada aplicação. A Equação 1 apresenta o modelo ideal do controlador PID.

$$u(t) = K_p e(t) + K_p \frac{1}{T_i} \int e(t) + K_p T_d \frac{de(t)}{dt} \tag{1}$$

Pode ser representada no domínio da frequência conforme a Equação 2, aplicando a transformada de Laplace considerando as condições iniciais nulas.

$$U(s) = K_p E(s) + \frac{K_p}{T_i s} E(s) + K_p T_d s E(s) \tag{2}$$

A Alpes automação (2025), uma fornecedora de soluções de automação para indústria e agronegócio, especifica os termos do controlador PID:

a) termo proporcional(P):

   -atua diretamente conforme a intensidade do desvio (distância entre o valor atual e o desejado). Quando o desvio se amplia, a ação proporcional aumenta correspondentemente, diminuindo com rapidez a separação inicial e aproximando o sistema do ponto ideal de operação.

b) termo integral(I):

   -esta função soma os desvios durante a operação, modificando a resposta para neutralizar diferenças persistentes. Representa um papel fundamental na eliminação de erros constantes, assegurando que o sistema mantenha exatidão contínua.

c) termo derivativo(D):

   -analisa a velocidade de alteração do desvio, contribuindo para antecipar e reduzir variações futuras. Este elemento é determinante para a estabilização do processo, reduzindo movimentos oscilatórios e proporcionando uma transição gradual durante alterações no sistema.

3.5.1 Aspectos de implementação prática do controlador PID

Na implementação prática, o termo derivativo não é aplicado em sua forma teórica pura devido à sua tendência de amplificar ruídos do sistema, além de apresentar limitações físicas de implementação. Este componente, em sua concepção original, não é realizável na prática, pois resultaria em uma função de transferência com o grau do numerador superior ao do denominador, violando princípios de causalidade. Para contornar estas restrições, tem-se como solução a incorporação de um filtro na ação derivativa, conforme demonstrado matematicamente na Equação 4 (Pinto, 2014).

$$C(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_p}{T_i s} + K_p T_d s \tag{3}$$

$$C(s) = K_p + \frac{K_p}{T_i s} + \frac{K_p T_d s}{\alpha T_d s + 1} \tag{4}$$

O fator α, segundo Pinto (2014), geralmente estar em torno de $\frac{1}{8}$.

Segundo Jhonson e Moradi (2005, *apud* Pinto, 2014), o PID paralelo tambem pode ser chamado de forma desacoplada do PID e possui uma relação direta com a forma padrão ou ideal, representada por:

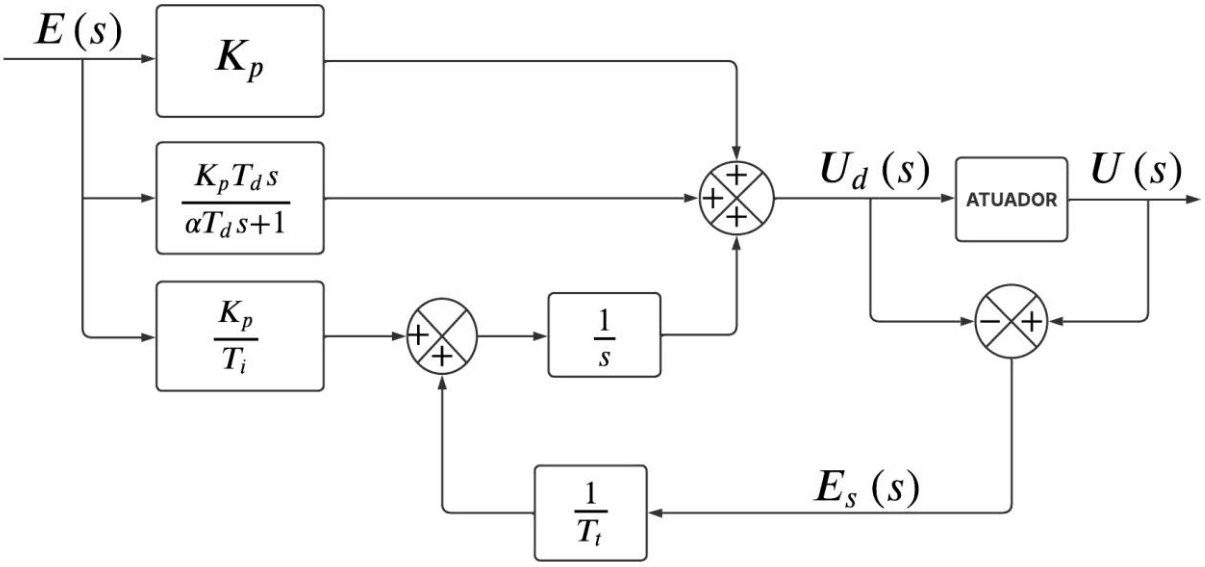
$$K_p = K_{p,ideal},\ K_i = \frac{K_{p,ideal}}{T_{i,ideal}} \text{ e } K_d = K_{p,ideal}T_{d,ideal}$$

$$C(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{\alpha T_d s + 1} \tag{5}$$

Na realidade operacional, todos os atuadores possuem limitações físicas. Quando a variável de controle alcança o limite superior ou inferior do dispositivo, ocorre a saturação do sinal de controle. Este fenômeno compromete a integridade da malha de retroalimentação, pois o atuador permanece fixo em seu valor extremo, tornando-se temporariamente insensível às variações na saída do processo. Em sistemas que empregam controladores com componente integral, este cenário torna-se particularmente problemático. Durante a saturação, o erro continua sendo acumulado pelo integrador, que progressivamente assume valores excessivamente elevados, fenômeno conhecido tecnicamente como "*wind-up*". Para que o sistema retorne à operação normal na região linear, torna-se necessária a "descarga" deste acúmulo no termo integral. Este processo de recuperação somente ocorre quando o sinal de erro inverte sua polaridade e permanece com sinal contrário por um período considerável, permitindo a gradual redução do valor acumulado. Como consequência direta deste mecanismo, o comportamento transitório do sistema apresenta características indesejáveis para aplicações industriais: resposta lenta e comportamento oscilatório acentuado (Silva, 2000).

De acordo com Silva (2000), existem varias formas de evitar o *wind-up* da ação integral. Entretanto, para o trabalho optou-se por utilizar o anti-*reset wind-up* do tipo *Back-Calculation*. Com o *Back-Calculation*, no momento que a saida do atuador satura, Silva (2000) explica que o sistema automaticamente recalcula o termo integral, ajustando-o para manter a saída exatamente no valor limite do atuador. Esta recalibração não ocorre de forma instantânea, mas segue um processo dinâmico controlado por uma constante de tempo específica T_t, proporcionando uma transição mais suave e controlada. Na Figura 10 é apresentado o diagrama de blocos do controlador PID com o anti-*reset* do tipo *Back-Calculation*.

Figura 10 – Diagrama de blocos do controlador com filtro e anti-*reset wind-up*.



Fonte: Autoria própria (2025).

Observe que o sistema apresenta uma malha interna de realimentação adicional, onde a discrepância entre o valor de entrada e saída do atuador constitui um erro E_s, que é reintroduzido na entrada do integrador com um ganho de $\frac{1}{T_t}$.

É importante observar que durante operação normal (sem saturação), este erro E_s, permanece em zero, tornando este circuito adicional transparente quando o controlador opera em regime linear, ou seja, com saída não saturada. Entretanto, quando ocorre saturação, E_s assume valor diferente de zero, e o sinal aplicado na entrada do integrador passa a ser:

$$\frac{1}{T_t}E_s(s) + \frac{K_p}{T_i}E(s) \tag{6}$$

Com isso, tem-se em regime permanente:

$$E_s(s) = -\frac{K_p T_t}{T_i}E(s) \tag{7}$$

Isto significa que a entrada do integrador tenderá a zero, prevenindo o acúmulo excessivo do termo integral. O período necessário para zerar a entrada do integrador é determinado pelo ganho $\frac{1}{T_t}$, sendo que T_t representa a constante de tempo que define a velocidade com que a entrada do integrador será neutralizada.

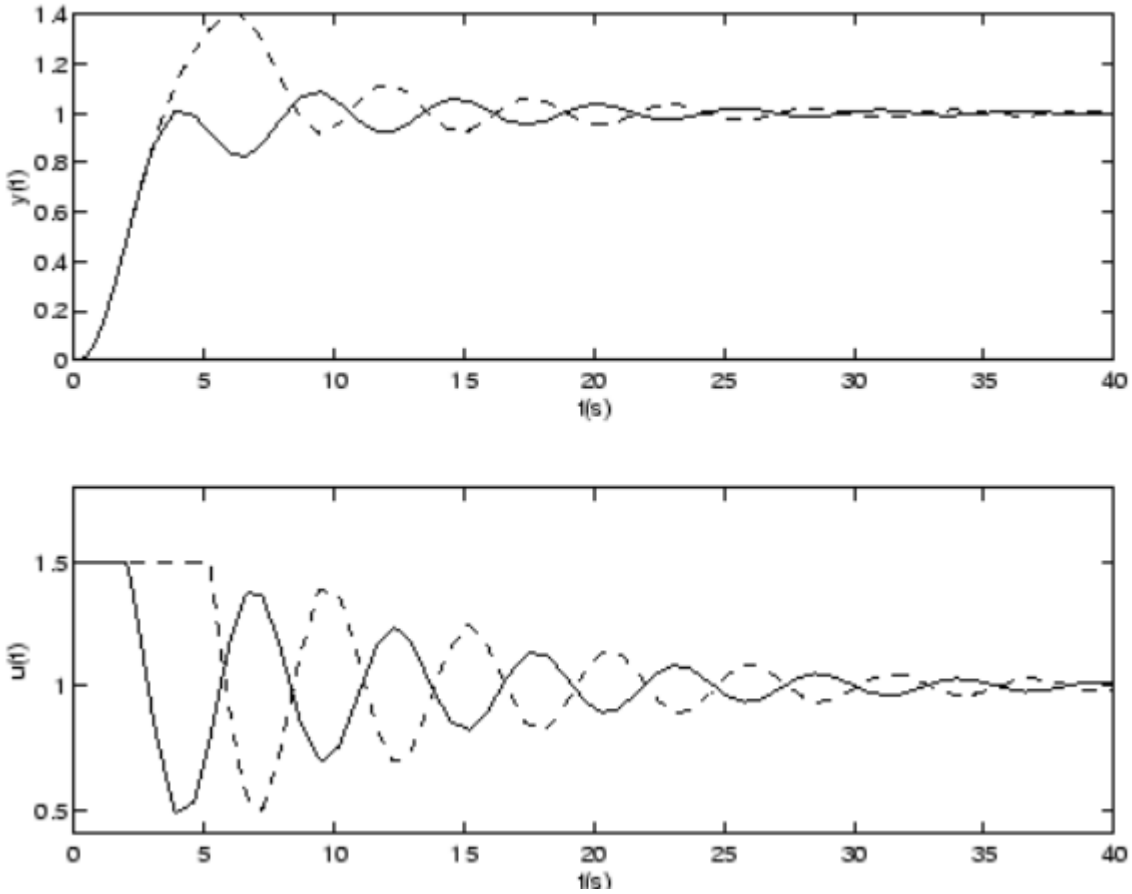
Embora valores reduzidos para T_t pareçam inicialmente vantajosos, recomenda-se cautela na parametrização, especialmente em sistemas que empregam ação derivativa. Ruídos

espúrios podem provocar saturações momentâneas no controlador, ativando prematuramente o mecanismo de anti *wind-up* e zerando indevidamente a entrada do integrador. Na prática, recomenda-se configurar T_t maior que T_d e menor que T_i. Uma regra empírica frequentemente aplicada sugere:

$$T_t = \sqrt{T_i T_d} \tag{8}$$

Conforme apresentado por Silva (2000), a Figura 11 ilustra a resposta de um sistema sob ação de um controlador Proporcional-Integral (PI), comparando os cenários com e sem a aplicação da compensação de *wind-up*.

Figura 11 – Com anti wind-up(linha continua) e sem anti wind-up(linha tracejada).



Fonte: UFRGS (2000).

Observa-se que o sistema, quando operando sem o mecanismo de anti *wind-up*, apresenta um sobressinal mais elevado e um tempo de estabilização maior.

Isto posto, pode-se obter o controlador com o anti *wind-up* conforme a Equação 9.

$$U_d(s) = K_p E(s) + \left(\frac{K_p}{T_i} E(s) + \frac{1}{T_t} E_s(s)\right)\frac{1}{s} + \frac{K_p T_d s}{\alpha T_d s + 1} E(s) \tag{9}$$

Ou conforme a Equação 10 na forma desacoplada.

$$U_d(s) = K_p E(s) + \left(K_i E(s) + \frac{1}{T_t} E_s(s)\right)\frac{1}{s} + \frac{K_d s}{\alpha T_d s + 1} E(s) \quad (10)$$

3.6 DISCRETIZAÇÃO PELO METODO DE TUSTIN

De acordo com Tavares (2017), nos sistemas de tempo discreto, a modelagem não se dá por meio de equações diferenciais, como nos sistemas contínuos, mas sim por equações de diferenças. Nessa abordagem, a transformada de Laplace é substituída pela transformada Z, ferramenta mais apropriada para a análise do comportamento dinâmico de sistemas discretos. Os métodos de discretização têm como principal finalidade possibilitar a obtenção de uma representação em tempo discreto, geralmente na forma de uma função de transferência na variável z, a partir de um sistema originalmente contínuo. Para isso, é estabelecida uma relação de mapeamento entre as variáveis s, da transformada de Laplace, e z, da transformada Z.

Uma forma de realizar esse mapeamento, é utilizar o metodo de Tustin. Este metódo, é uma transformação bilinear que pode ser aplicada por meio de cálculos manuais e permite obter uma função de transferência digital cuja resposta, nos instantes de amostragem, se aproxima da resposta do sistema analógico correspondente (Nise, 2012). Por tanto, tal mapeamento, utilizando a transformada Z de Tustin é dado por:

$$s = \frac{2}{T}\frac{z-1}{z+1} = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \quad (11)$$

E a inversa é dada por:

$$z = \frac{-\left(s+\frac{2}{T}\right)}{\left(s-\frac{2}{T}\right)} = \frac{1+\frac{T}{2}s}{1-\frac{T}{2}s} \quad (12)$$

Ainda segundo Nise (2012), a medida que se utiliza um periodo de amostragem(T) menor, a saida do controlador digital projetado se aproxima mais da saida controlador analógico.

A propriedade de translação da transformada Z é essencial para compreender como ocorre o deslocamento temporal de um sinal, seja em forma de atraso ou adiantamento. Esse deslocamento pode ser classificado em dois casos: translação para a direita, que representa um atraso no tempo, e translação para a esquerda, que corresponde a um adiantamento (Oliveira, 2025).

a) Translação à direita:

$$Z\{x(k-n)\} = z^{-n}X(z), \text{ tempo discreto } k \text{ e } n \text{ amostras.} \quad (13)$$

b) Translação à esquerda:

$$Z\{x(k+n)\} = z^{n}X(z) - \sum_{k=0}^{n-1} x(k)z^{n-k}, \text{ tempo discreto } k \text{ e } n \text{ amostras.} \quad (14)$$

De acordo com Oliveira (2025), a transformada Z inversa é uma técnica importante na análise de sistemas discretos, pois permite recuperar o sinal original no tempo discreto $x(k)$ a partir de sua forma transformada no domínio z. Por tanto, pode-se utilizar a propriedade de translação para obter a transformada inversa.

Para uma melhor compreensão, a Equação 10, que representa o controlador PID na forma contínua e desacoplada, será decomposta em suas três componentes: proporcional, integral e derivativa. O método de Tustin é aplicado separadamente a cada uma dessas partes, com o objetivo de obter a forma discreta do controlador:

a) Discretizando a componente proporcional:

$$U_p(z) = K_p E(z) \quad (15)$$

Transformada Z inversa:

$$U_p(k) = K_p E(k) \quad (16)$$

b) Discretizando a componente integral:

$$U_i(z) = \left(K_i E(z) + \frac{1}{T_t} E_s(z)\right) \frac{T}{2} \frac{1+z^{-1}}{1-z^{-1}} \quad (17)$$

$$U_i(z)(1-z^{-1}) = \frac{T}{2}(1+z^{-1})\left(K_i E(z) + \frac{1}{T_t} E_s(z)\right) \quad (18)$$

Transformada Z inversa:

$$U_i(k) - U_i(k-1) = \frac{T}{2}\left\{K_i[E(k) + E(k-1)] + \frac{1}{T_t}[E_s(k) + E_s(k-1)]\right\} \quad (19)$$

$$U_i(k) = U_i(k-1) + \frac{T}{2} K_i[E(k) + E(k-1)] + \frac{T}{2T_t}[E_s(k) + E_s(k-1)] \quad (20)$$

c) Discretizando a componente derivativa:

$$U_f(z) = \frac{K_d \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}}{\alpha T_d \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} + 1} E(z) \tag{21}$$

Multiplicando o numerador e o denominador por $(1+z^{-1})$:

$$U_f(z) = \frac{K_d \frac{2}{T}(1-z^{-1})}{\alpha T_d \frac{2}{T}(1-z^{-1}) + (1+z^{-1})} E(z) \tag{22}$$

Dividindo o numerador e o denominador por $\frac{2}{T}$:

$$U_f(z) = \frac{K_d(1-z^{-1})}{\alpha T_d(1-z^{-1}) + \frac{T}{2}(1+z^{-1})} E(z) \tag{23}$$

$$U_f(z)\alpha T_d(1-z^{-1}) + U_f(z)\frac{T}{2}(1+z^{-1}) = K_d(1-z^{-1})E(z) \tag{24}$$

Transformada Z inversa:

$$\alpha T_d\left[U_f(k) - U_f(k-1)\right] + \frac{T}{2}\left[U_f(k) + U_f(k-1)\right] = K_d[E(k) - E(k-1)] \tag{25}$$

$$\alpha T_d U_f(k) - \alpha T_d U_f(k-1) + \frac{T}{2}U_f(k) + \frac{T}{2}U_f(k-1) = K_d[E(k) - E(k-1)] \tag{26}$$

$$U_f(k)\left(\alpha T_d + \frac{T}{2}\right) + U_f(k-1)\left(\frac{T}{2} - \alpha T_d\right) = K_d[E(k) - E(k-1)] \tag{27}$$

$$U_f(k)\left(\alpha T_d + \frac{T}{2}\right) = -U_f(k-1)\left(\frac{T}{2} - \alpha T_d\right) + K_d[E(k) - E(k-1)] \tag{28}$$

$$U_f(k) = -\frac{\left(\frac{T}{2} - \alpha T_d\right)}{\left(\alpha T_d + \frac{T}{2}\right)} U_f(k-1) + \frac{K_d}{\left(\alpha T_d + \frac{T}{2}\right)}[E(k) - E(k-1)] \tag{29}$$

3.7 ACIONAMENTO POR PWM

Para Guse (2013), o PWM (Modulação por Largura de Pulso - *Pulse Width Modulation*) é uma técnica amplamente utilizada para controlar a potência fornecida a dispositivos eletrônicos, como motores e LEDs. Seu funcionamento baseia-se na comutação rápida de um sinal digital entre os estados ligado (*ON*) e desligado (*OFF*), dentro de um período fixo. A razão entre o tempo em que o sinal permanece ativado e a duração total do ciclo, conhecida como *duty cycle*, determina a média de potência entregue ao dispositivo.

Alterando o tempo em que o sinal permanece no nível alto durante cada ciclo de PWM, consegue-se variar a tensão média aplicada ao dispositivo. Essa modulação influencia

diretamente o desempenho do componente controlado, permitindo, por exemplo, ajustar a velocidade de um motor CC. O princípio de operação do PWM está relacionado ao chamado ciclo de trabalho (*duty cycle*), que representa a fração do tempo em que o sinal permanece no nível lógico alto(T_{on}) durante um ciclo completo de operação($T_{on} + T_{off}$), em que T_{off} é o tempo que o sinal permanece em nivel lógico baixo. O ciclo de trabalho é expresso em porcentagem e dado pela Equação 30.

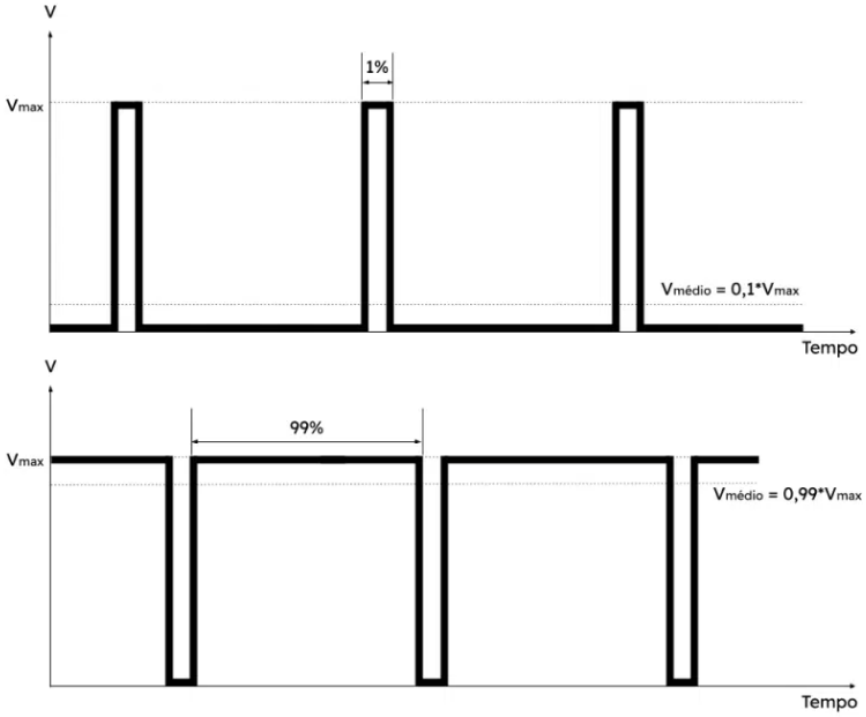
$$D = \frac{T_{on}}{T_{on} + T_{off}} \tag{30}$$

À medida que o tempo em que o sinal permanece no estado ligado (*ON*) aumenta, eleva-se também a tensão média fornecida ao dispositivo, resultando em uma maior potência efetiva aplicada à carga. Desta forma, a Equação para tensão média aplicada à carga é dada conforme a Equação 31.

$$V_{médio} = DV_{max} \tag{31}$$

Guse (2013) mostra dois exemplos de aplicação da Equação 31, pode-se vizualiza-los na Figura 12.

Figura 12 – Exemplos de aplicação da Equação 31.



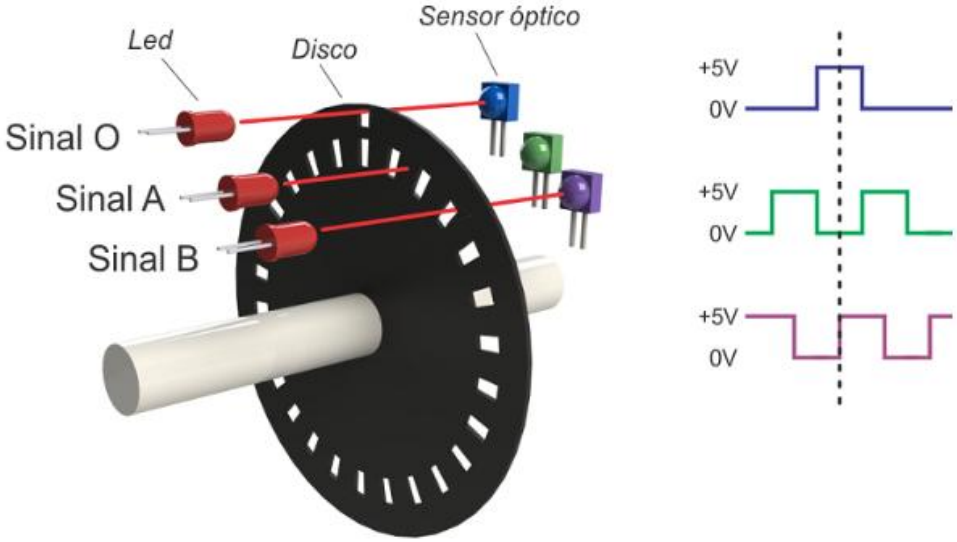
Fonte: Makerhero (2013).

3.8 ENCODER

Encoders são sensores eletromecânicos utilizados para converter informações de posição ou movimento em sinais elétricos digitais. Por meio desses dispositivos, torna-se possível quantificar deslocamentos, controlar a velocidade de sistemas, medir ângulos e rotações, além de viabilizar aplicações como o posicionamento preciso e o controle de movimento de braços robóticos, entre outras (Almeida, 2017).

Para Almeida (2017), os *encoders* ópticos são compostos por um disco com marcações, um emissor de luz (normalmente um LED) e um receptor (como um fotodetector). À medida que o disco gira, ele alterna entre bloquear e permitir a passagem da luz, gerando pulsos em forma de onda quadrada nas saídas do *encoder*. A quantidade desses pulsos por volta determina a resolução do *encoder*. O modelo incremental, o mais utilizado, possui três sinais de saída: A, B e O. O sinal A fornece os pulsos, o sinal B é igual ao sinal A, porém defasado em + ou – 90º para indicar o sentido de rotação, e o sinal O marca o ponto de referência de cada volta, conforme pode-se visualizar na Figura 13.

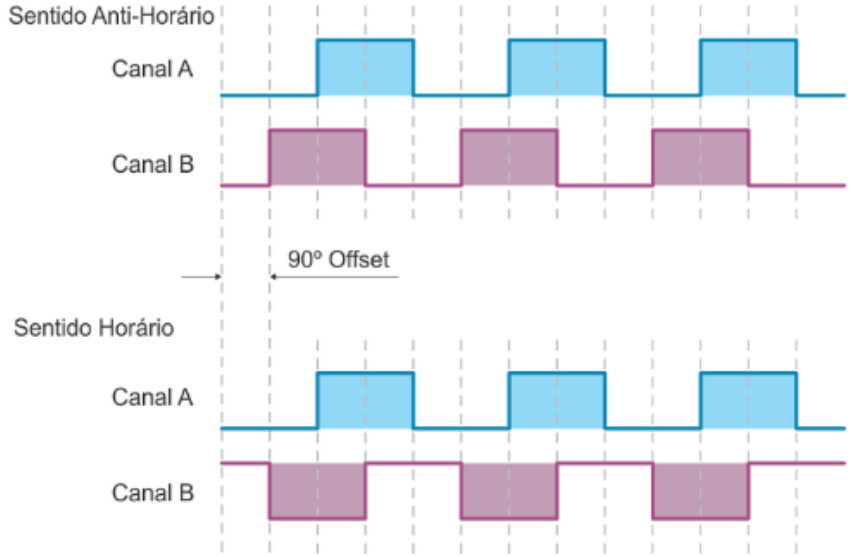
Figura 13 – Sinais de saída do encoder.



Fonte: Hitecnologia (2017).

A direção do giro é determinada pela ordem de chegada dos sinais A e B. Se B estiver adiantado 90º em relação a A, o movimento é anti-horário, se estiver atrasado, o giro é horário. Essa análise é feita observando o estado de B durante as bordas de subida do sinal A, a Figura 14 ilustra esse comportamento. Já com relação ao sinal O, Z ou I do *encoder* serve como referência para a posição zero, permitindo contar as voltas completas do eixo.

Figura 14 – Direção de giro determinada pelos canais A e B.



Fonte: Hitecnologia (2017).

3.9 MÉTODOS DE SINTONIA DE CONTROLADORES PID

Controladores PID exigem uma sintonia precisa de seus parâmetros para que os objetivos de controle sejam alcançados, assegurando uma resposta adequada da variável controlada. Por tanto, de acordo com VanDore (2006, *apud* Pinto, 2014), sintonizar uma malha de controle envolve analisar como o processo responde às ações do controlador e, a partir dos critérios de desempenho estabelecidos, ajustar a dinâmica do algoritmo PID para reduzir os erros e melhorar a resposta do sistema.

Independentemente da metodologia adotada, os procedimentos de identificação da dinâmica do processo e de sintonia de controladores PID costumam seguir uma sequência comum. Inicialmente, o sistema é submetido a variações no sinal de controle, com o objetivo de provocar uma resposta dinâmica. Em seguida, essa resposta é observada e quantificada. Por fim, com base nessa análise e nas especificações de desempenho desejadas, realiza-se o ajuste dos parâmetros do controlador PID (Johnson e Moradi, 2005, *apud* Pinto, 2014).

Dentre os métodos de sintonia de controladores PID, estão os métodos de Ziegler-Nichols, Cohen e Coon e o método de Controle por Modelo Interno (IMC).

3.9.1 Método de Ziegler-Nichols

Ziegler e Nichols estão entre os pioneiros no desenvolvimento de um método de sintonia que se destaca pela simplicidade e objetividade. Eles propuseram dois métodos de sintonia para controladores PID, baseados na análise do comportamento do processo. O primeiro método,

aplicado em malha aberta, o segundo método, em malha fechada. A partir dessas informações, os parâmetros do controlador K_p, T_i e T_d são definidos por meio de expressões matemáticas simples. As regras propostas buscavam limitar o sobressinal da resposta a aproximadamente 25% (Fermino, 2014).

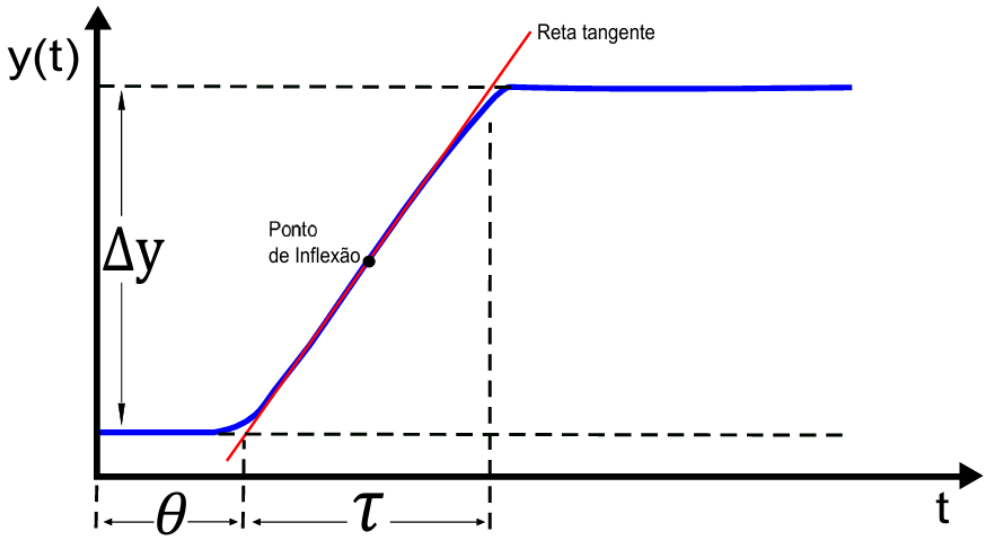
Neste trabalho, foi adotado o primeiro método, baseado na resposta ao degrau do sistema em malha aberta. Esse método é aplicável a sistemas estáveis cuja saída apresenta um comportamento em forma de "S" (Fermino, 2014). Por tanto, são para sistemas, cuja resposta ao degrau apresenta um conportamento monotônico após um tempo incial. Desta forma, ele pode ser aproximado por uma função de transferência de primeira ordem com atraso na resposta, como apresentado na Equação 32.

$$G_p(s) = \frac{k}{\tau s + 1} e^{-\theta s} \tag{32}$$

Em que k representa o ganho estático do processo, definido pela razão entre a variação da saída (ΔY) e a variação da entrada (ΔU), ou seja, $k = \frac{\Delta Y}{\Delta U}$.

A forma da resposta ao degrau pode ser descrita por dois parâmetros fundamentais: o tempo de atraso (θ) e a constante de tempo (τ). Para estimá-los, traça-se uma linha tangente ao ponto de maior inclinação da curva. O ponto onde essa tangente cruza o eixo do tempo fornece uma estimativa de θ, enquanto o ponto que ela atinge o valor final da saída, $Y(\infty)$, permite determinar τ. Esse procedimento está representado na Figura 15 (Pinto, 2014).

Figura 15 – Resposta em malha aberta de um processo de primeira ordem.



Fonte: Pinto (2014).

Após determinar os valores de k, θ e τ recorre-se à Tabela 1, elaborada por Ziegler e Nichols (1943, *apud* Pinto, 2014), a qual apresenta diretrizes para o ajuste dos ganhos do controlador PID, tomando como base um modelo de sistema de primeira ordem com atraso, conforme representado na Equação 32.

Tabela 1 – Sintonia pelo método de Ziegler-Nichols.

Controlador	K_p	T_i	T_d
P	$\frac{\tau}{k\theta}$	-	-
PI	$0{,}9\left(\frac{\tau}{k\theta}\right)$	$3{,}33\theta$	-
PID	$1{,}2\left(\frac{\tau}{k\theta}\right)$	2θ	$0{,}5\theta$

Fonte: Adaptado de Pinto (2014).

Considerações a cerca do método de Ziegler-Nichols, segundo Pinto(2014):

a) Desenvolvida com base nos controladores disponíveis na época, a proposta de Ziegler e Nichols apresentada na Tabela 1 ainda gera discussões quanto à estrutura utilizada, série ou paralela. Apesar dessa incerteza, sabe-se que o ganho proporcional interfere nos componentes integral e derivativo.

b) O fator de incontrolabilidade, expresso pela razão $\frac{\theta}{\tau}$, é um parâmetro essencial a ser considerado na análise do sistema. De acordo com Corripio (1996, *apud* Pinto, 2014), valores desse fator entre 0,1 e 0,3 tendem a resultar em boas sintonias. Já Rivera et al. (1986, *apud* Pinto, 2014) indicam que sintonias aceitáveis podem ser alcançadas quando esse valor está entre 0,2 e 1,4. No entanto, caso o fator ultrapasse 4, há uma forte tendência de que o sistema se torne instável.

c) Diante das incertezas inerentes à modelagem do processo que podem variar entre 5% e 20%, Campos e Teixeira (2006) recomendam a aplicação de um fator de segurança, também conhecido como "*detuning*", ao utilizar as sintonias PID propostas por Ziegler e Nichols. Essa estratégia consiste em ajustar apenas os ganhos proporcional e integral, conforme demonstrado nas Equações 33 e 34:

$$K_p = \frac{K_{pzn}}{1{,}25} \tag{33}$$

$$T_i = 2{,}5T_{izn} \tag{34}$$

3.9.2 Método de Cohen e Coon

Como a sintonia de controladores para sistemas de primeira ordem com atraso pode variar devido à presença de um número infinito de modos harmônicos, soluções associadas à equação característica fundamental, Cohen e Coon propuseram uma abordagem teórica focada no modo harmônico fundamental, que possui a menor frequência e maior amplitude. O objetivo também incluiu a definição de parâmetros de controle adequados para processos com atraso mais acentuado, ou seja, com fator de incontrolabilidade $\frac{\theta}{\tau}$ superior a 0,3. A Tabela 2 apresenta os valores recomendados para a sintonia PID com base no método de Cohen e Coon, considerando o uso do controlador PID na forma ideal no estudo original (Campos e Teixeira, 2006). Por sua vez, o método de Cohen e Coon, de acordo com Rivera et al. (1886, *apud* Campos e Teixeira, 2006), oferece um desempenho satisfatório quando o fator de incontrolabilidade se encontra na faixa de 0,6 a 4,5. Em contrapartida, a estabilidade do sistema é reduzida quando o fator de incontrolabilidade é inferior a 2 (Pinto, 2014).

Tabela 2 – Sintonia pelo método de Cohen e Coon.

Controlador	K_p	T_i	T_d
P	$\left(1{,}03 + 0{,}35\frac{\theta}{\tau}\right)\frac{\tau}{k\theta}$	-	-
PI	$\left(0{,}9 + 0{,}083\frac{\theta}{\tau}\right)\frac{\tau}{k\theta}$	$\frac{\left(0{,}9 + 0{,}083\frac{\theta}{\tau}\right)}{\left(1{,}27 + 0{,}6\frac{\theta}{\tau}\right)}\theta$	-
PID	$\left(1{,}35 + 0{,}25\frac{\theta}{\tau}\right)\frac{\tau}{k\theta}$	$\frac{\left(1{,}35 + 0{,}25\frac{\theta}{\tau}\right)}{\left(0{,}54 + 0{,}33\frac{\theta}{\tau}\right)}\theta$	$\frac{0{,}5\theta}{\left(1{,}35 + 0{,}25\frac{\theta}{\tau}\right)}$

Fonte: Adaptado de Campos e Teixeira (2006).

3.9.3 Método de Controle por Modelo Interno (IMC)

O controle por modelo interno (IMC – *Internal Model Control*) busca projetar um controlador com base no modelo matemático do processo e nos requisitos de desempenho desejados. Nessa abordagem, o controlador incorpora uma representação interna da planta, a qual pode ser utilizada exclusivamente durante o projeto ou ainda integrada ao funcionamento contínuo do sistema (Campos e Teixeira, 2006).

O controlador utilizado seguiu a estrutura PID ideal na forma paralela. As diretrizes de ajuste do método IMC são recomendadas para casos em que o fator de incontrolabilidade seja superior a 0,125, conforme indicado por Rivera et al. (1986, *apud* Pinto, 2014). Os autores

consideraram diversas configurações dinâmicas para o processo e, com base nelas, definiram os parâmetros PID a partir do valor do parâmetro de desempenho (Pinto, 2014).

Quando a planta pode ser bem representada por um modelo de primeira ordem com atraso conforme a Equação 32, a sintonia PID resultante é apresentada na Tabela 3 (Campos e Teixeira, 2006).

No método de sintonia IMC, o parâmetro λ atua como critério de desempenho, determinando a velocidade com que a saída do processo deve acompanhar o valor de referência (*setpoint*). A escolha deste parâmetro deve respeitar as restrições dinâmicas do processo, por tanto, não deve ser menor que o atraso do processo, uma vez que levaria a uma sintonia agressiva (Campos e Teixeira, 2006).

Tabela 3 – Sintonia pelo método IMC.

Controlador	K_p	T_i	T_d	**Sugestão para o desempenho**
PI	$\frac{2\tau+\theta}{k2\lambda}$	$\tau+\frac{\theta}{2}$	-	$\frac{\lambda}{\theta} > 1{,}7$
PID	$\frac{2\tau+\theta}{k(2\lambda+\theta)}$	$\tau+\frac{\theta}{2}$	$\frac{\tau\theta}{2\tau+\theta}$	$\frac{\lambda}{\theta} > 0{,}8$

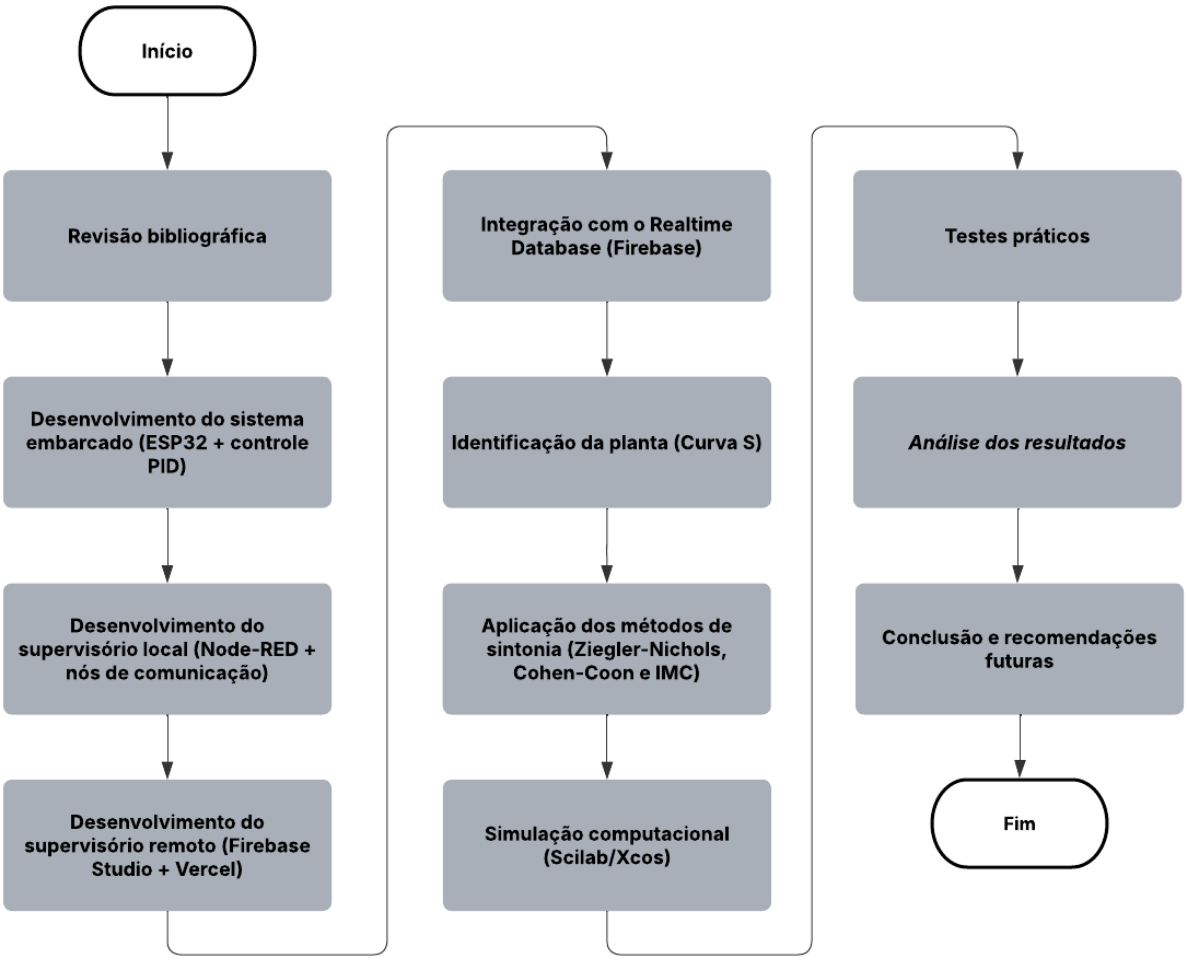
Fonte: Adaptado de Campos e Teixeira (2006).

Como regra geral, e para uma operação mais conservadora, sugere-se que λ seja igual à constante de tempo dominante do processo(maior constante de tempo) ($\lambda \cong \tau_{dominante}$). Para garantir a robustez e a estabilidade de um sistema de controle, a sintonia deve ser realizada de maneira mais conservadora (utilizando um λ maior) em algumas situações. Isso é especialmente relevante quando o sistema exibe não-linearidades pronunciadas, como histereses, bandas mortas ou saturações, ou quando a modelagem do processo apresenta imprecisões. Além disso, um tempo morto (atraso de transporte) considerável no processo é um fator que complica o controle e, portanto, requer que o parâmetro λ seja ajustado para um valor mais elevado (Campos e Teixeira, 2006).

4 METODOLOGIA

Este trabalho adota uma abordagem quantitativa, pois envolve a coleta e análise de dados obtidos por sensores, supervisórios e banco de dados, para avaliar o desempenho de diferentes métodos de sintonia aplicados ao controle de velocidade de um motor de corrente contínua. Trata-se de uma pesquisa aplicada, voltada à solução de um problema real no contexto da automação industrial. Quanto aos objetivos, caracteriza-se como uma pesquisa exploratória e explicativa, uma vez que busca tanto entender o comportamento do sistema quanto explicar os efeitos de cada técnica de sintonia aplicada. O percurso metodológico foi realizado conforme o Fluxograma 1.

Fluxograma 1 – Percurso Metodológico do trabalho.



Fonte: Autoria própria (2025).

a) Revisão bibliográfica:

Inicialmente, foi realizada uma revisão bibliográfica com base dos principais conceitos relacionados ao controle de velocidade de motores CC, protocolo de comunicação industrial MQTT, microcontrolador da ESP32, e ferramentas de supervisão, como o Node-RED e a

plataforma *Firebase studio*. Também foram estudadas técnicas de sintonia de controladores PID, como Ziegler-Nichols, Cohen-Coon e Modelo Interno (IMC). O objetivo desta etapa foi contextualizar o projeto com soluções existentes, compreendendo os fundamentos necessários para a implementação do sistema de controle proposto.

b) Implementação do sistema embarcado e supervisórios:

Com base nos conhecimentos adquiridos, foi desenvolvido o sistema de controle de velocidade de um motor CC de 6V, utilizando o microcontrolador ESP32, com os parâmetros de controle obtidos através dos métodos de sintonia. O sistema de acionamento emprega uma ponte H L298N, com o sinal de PWM gerado pela ESP32. A velocidade do motor é monitorada em tempo real por um *encoder* incremental, cujos dados são utilizados para calcular a velocidade angular em RPM.

Foi implementado um supervisório local no Node-RED, com comunicação via MQTT (utilizando o broker Mosquitto), que permite a visualização e controle do sistema em tempo real. Além disso, foi criado um supervisório remoto em nuvem, desenvolvido na plataforma *Firebase studio* e hospedado no vercel, utilizando o *Realtime Database* como meio de integração entre os dados do Node-RED e a interface web.

c) Identificação do modelo matemático da planta e aplicação dos métodos de sintonia

Para permitir a aplicação das técnicas de controle, foi realizada a identificação do modelo matemático da planta do sistema, que se refere ao motor utilizado no trabalho, por meio da curva S obtida da resposta ao degrau no regime de malha aberta. Em seguida, foram implementados os métodos de sintonia de controladores PID baseados em Ziegler-Nichols, Cohen-Coon e IMC. Cada conjunto de parâmetros foi aplicado ao sistema, em malha fechada, para fins de comparação.

d) Simulação computacional para validação e comparação dos resultados

Como etapa complementar à análise prática do sistema de controle, foi realizada uma simulação computacional no ambiente Scilab/Xcos. Esta etapa teve como objetivo validar os parâmetros dos controladores PID obtidos pelos métodos de sintonia aplicados, tais como Ziegler-Nichols, Cohen-Coon e IMC, bem como permitir uma comparação mais precisa entre os comportamentos simulados e os comportamentos reais observados experimentalmente.

e) Análise dos métodos

A análise dos resultados teve como objetivo avaliar o desempenho de cada método de sintonia frente a um mesmo conjunto de testes. Foram extraídos indicadores de desempenho como tempo de acomodação, tempo de subida e sobressinal (*overshoot*), durante o

funcionamento do sistema, por meio dos gráficos de velocidade gerados no Excel *Data Streamer* em testes práticos com o sistema físico e em simulação computacional. A interpretação dos resultados foi realizada com base em critérios de desempenho padronizados, possibilitando uma comparação objetiva entre os métodos analisados.

f) Apresentação dos resultados

A última etapa consistiu na redação da versão final do TCC, contemplando todas as etapas do desenvolvimento, os resultados obtidos e as conclusões sobre o comportamento do sistema sob diferentes métodos de sintonia. O documento inclui gráficos, tabelas e imagens dos sistemas implementados, além de recomendações para trabalhos futuros.

4.1 ARQUITETURA DO SISTEMA

O sistema desenvolvido possui uma estrutura distribuída composta por três camadas principais: a camada de controle embarcado, a camada de comunicação e a camada de supervisão. A camada de controle embarcado é responsável pela execução do algoritmo de controle em tempo real no ESP32, utilizando os parâmetros ajustados para cada método de sintonia. A camada de comunicação utiliza o protocolo MQTT para intercâmbio de dados entre o ESP32 e os supervisórios. Por fim, a camada de supervisão permite o monitoramento e a atuação sobre o sistema, tanto localmente quanto remotamente. A Figura 16 apresenta a estrutura do sistema desenvolvido.

Figura 16 – Arquitetura do sistema.

Fonte: Autoria própria (2025).

4.2 COMPONENTES UTILIZADOS

Esta seção apresenta os principais materiais e componentes físicos empregados na montagem do circuito do sistema de controle de velocidade do motor de corrente contínua. O objetivo é descrever os dispositivos que compõem a estrutura física do projeto, detalhando sua função dentro da arquitetura geral do sistema. A montagem foi realizada de forma a garantir a integração entre o controle embarcado, o acionamento do motor e a aquisição dos sinais de realimentação, essenciais para o funcionamento da malha fechada.

a) Microcontrolador

O circuito de controle foi baseado na placa ESP32 DevKit V1 (Figura 17), que integra o microcontrolador ESP32-D0WDQ6, o qual possui conectividade Wi-Fi e Bluetooth, amplamente utilizado em projetos embarcados. Sua função no sistema foi realizar a leitura dos sinais do *encoder*, aplicar o controle PID e gerar o sinal PWM responsável pela variação da velocidade do motor. Além disso, o ESP32 também realizou a comunicação com os supervisórios local e remoto por meio do protocolo MQTT.

Figura 17 – Plataforma embarcada ESP32.

Fonte: Thingsboard (s.d.).

b) Motor de corrente contínua com *encoder*

O sistema de controle foi desenvolvido utilizando um motor de corrente contínua (CC) de 6 V com redutor acoplado. Esse atuador foi escolhido por permitir o controle preciso da velocidade angular do eixo de saída, aspecto fundamental para os objetivos do projeto, que envolvem a análise do comportamento dinâmico do sistema em malha fechada.

Acoplado ao motor, encontra-se um *encoder* óptico incremental com dois canais defasados, responsáveis por fornecer os pulsos digitais utilizados no cálculo da velocidade. Esses sinais são lidos pelo microcontrolador e utilizados como realimentação pelo algoritmo de controle PID implementado. Com isso, é possível ajustar dinamicamente o sinal de acionamento do motor, mantendo a rotação próxima ao valor de referência estabelecido.

A Figura 18 ilustra o motorredutor com *encoder* utilizado.

Figura 18 – Motorredutor com *encoder*.

Fonte: Autoria própria (2025).

A Tabela 4 apresenta as principais características técnicas do motor com *encoder*.

Tabela 4 – Características técnicas do motor com *encoder*.

Parâmetro	**Valor**	**Unidade**	**Tolerância**
Tensão nominal	6,0	V	-
Corrente nominal (com carga)	0,45	A	-
Corrente nominal (stall)	1,3	A	-
Redução mecânica	45: 1	–	-
Velocidade nominal (sem carga)	130	RPM	$\pm 10\%$
Velocidade nominal (com carga)	100	RPM	$\pm 10\%$
Torque nominal (com carga)	0,49	$kgf \times cm$	-
Torque de partida (stall)	8,2	$kgf \times cm$	-

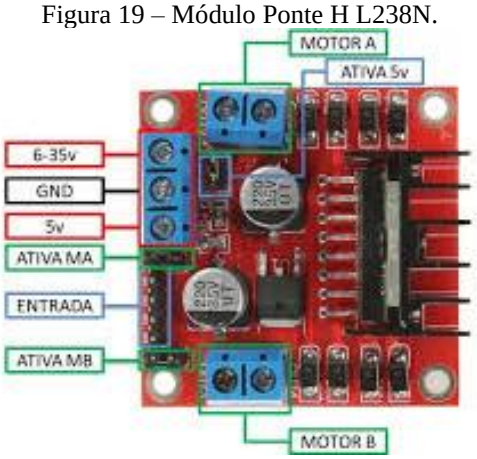
Fonte: Adaptado de Xytmotors (s.d).

c) Ponte H

Para o acionamento do motor CC, foi utilizado um módulo ponte H do tipo L298N, capaz de controlar motores em ambos os sentidos de rotação. Esse componente permitiu que o sinal PWM gerado pela ESP32 fosse aplicado ao motor, funcionando como uma interface de potência, além de possibilitar o controle do sentido de giro por meio de duas entradas digitais. O uso da ponte H foi essencial para fornecer corrente adequada ao motor e garantir a flexibilidade de operação exigida pelo sistema.

A Figura 19 mostra o módulo ponte H utilizado no sistema.

Figura 19 – Módulo Ponte H L238N.



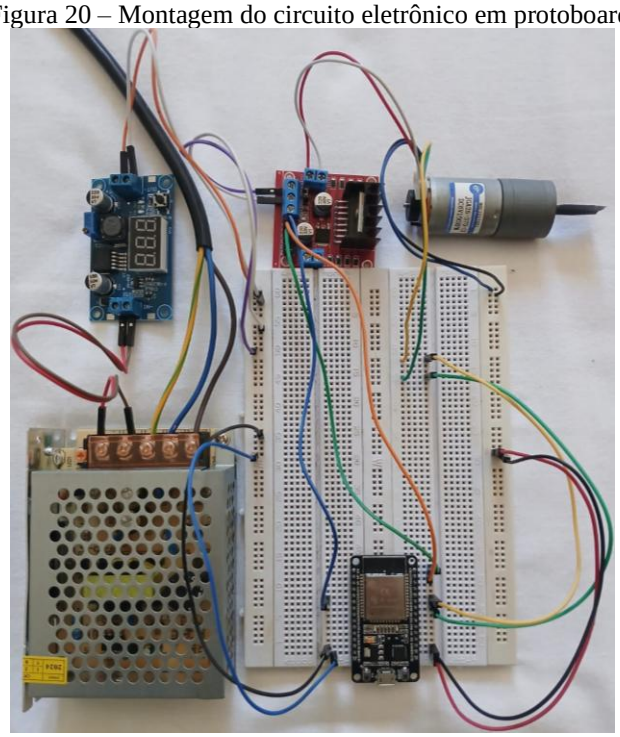
Fonte: Makerhero (2025).

d) Conexões e montagem

Os componentes foram interligados por meio de uma *protoboard*, utilizando fios do tipo jumper e cabos de conexão macho-macho e macho-fêmea. Esse arranjo permitiu a montagem rápida do circuito e facilitou testes e ajustes durante o desenvolvimento. Para garantir uma alimentação adequada ao motor, foi utilizada uma fonte externa de 12V interligada a um regulador de tensão reduzindo a tensão para 6V, com capacidade de corrente suficiente para o funcionamento do conjunto, evitando quedas de tensão ou superaquecimento.

A Figura 20 apresenta a montagem do circuito com os principais componentes eletrônicos do sistema, incluindo a ESP32, o módulo ponte H, o motorredutor com *encoder* e a alimentação externa conectados na protoboard.

Figura 20 – Montagem do circuito eletrônico em protoboard.



Fonte: Autoria própria (2025).

O esquema de ligação do circuito apresentado na Figura 20 encontra-se no Apêndice A.

4.3 DESENVOLVIMENTO DO SISTEMA EMBARCADO

O sistema embarcado foi desenvolvido com foco na comunicação entre o microcontrolador ESP32 e os supervisórios local e remoto, utilizando o protocolo MQTT como principal meio de troca de informações. A conexão à rede foi realizada via Wi-Fi, por meio da biblioteca WiFi.h, possibilitando que a ESP32 se conectasse a um roteador e passasse a operar como um nó da rede local.

Após estabelecer a conexão com a rede, a ESP32 conectou-se ao broker MQTT por meio da biblioteca PubSubClient.h, utilizando como identificador um nome exclusivo gerado a partir do endereço MAC do dispositivo. Durante a inicialização, o sistema se inscreveu nos tópicos IFPBCajazeiras/usuario01/acionamento e IFPBCajazeiras/usuario01/setpoint, responsáveis por receber, respectivamente, os comandos de ligar/desligar o motor e os valores de referência da velocidade desejada.

Além de receber comandos, o sistema também publica informações relevantes sobre a operação do motor. A cada 50 milissegundos, a ESP32 envia a velocidade atual do motor (em rotações por minuto) por meio do tópico IFPBCajazeiras/usuario01/velocidade, bem como o tempo acumulado de funcionamento no tópico IFPBCajazeiras/usuario01/Time. Para fins de análise e monitoramento da qualidade do controle, o valor do erro entre o *setpoint* e a velocidade real também é publicado, utilizando o tópico IFPBCajazeiras/usuario01/erro.

As mensagens recebidas nos tópicos de acionamento e *setpoint* são tratadas por uma função de *callback*, que interpreta os comandos recebidos e ajusta o estado interno da lógica embarcada. A ESP32 foi configurada para receber os comandos de acionamento dos supervisórios via MQTT, por meio dos nós de comunicação MQTT do Node-RED. Sempre que o motor é desligado, as variáveis do controlador PID são reinicializadas para garantir uma retomada estável e precisa.

Essa estrutura de comunicação permitiu uma integração eficiente entre o sistema embarcado e os supervisórios desenvolvidos, tanto no ambiente local (via Node-RED) quanto no ambiente remoto (via interface *web*), garantindo envio e recebimento de dados em tempo real com baixa latência e alta confiabilidade.

O controlador PID foi implementado no ESP32 utilizando as equações no tempo discreto obtidas na Seção 3.6, que representam as ações proporcional, integral e derivativa do modelo do controlador PID apresentado na Equação 10 da Seção 3.5.1. Essa formulação

corresponde à estrutura em forma paralela do PID, incorporando o filtro na ação derivativa e o mecanismo de anti *wind-up* do tipo *back-calculation*.

4.3.1 Principais trechos do código

Para ilustrar a implementação do sistema embarcado, são apresentados os principais trechos do código desenvolvidos para a ESP32, divididos em dois módulos:

a) Comunicação com o supervisório local

A comunicação ocorre por meio da publicação e assinatura nos tópicos MQTT correspondentes. O trecho do código responsável por estabelecer a conexão da ESP32 com o broker MQTT e realizar a assinatura dos tópicos necessários para comunicação é mostrado na Figura 21.

Figura 21 – Trecho do código da conexão com o broker e assinatura dos tópicos acionamento e *setpoint*.

```
201   while (!client.connected()) {                // Conecta ao broker MQTT
202     String client_id = "esp32-client-" + String(WiFi.macAddress());
203     Serial.print("Conectando ao broker MQTT... ");
204     if (client.connect(client_id.c_str())) {
205       Serial.println("Conectado ao broker!");
206       client.subscribe("IFPBCajazeiras/usuario01/acionamento");  // Assina
207       client.subscribe("IFPBCajazeiras/usuario01/setpoint");  // Assina o
208     } else {
209       Serial.print("Falha, código: ");
210       Serial.println(client.state());
211       delay(2000);  // Espera 2 segundos antes de tentar novamente
212     }
213   }
```

Fonte: Autoria própria (2025).

Quando a conexão com o broker é bem-sucedida, a função client.subscribe() é utilizada para que a ESP32 realize a assinatura dos tópicos "IFPBCajazeiras/usuario01/acionamento" e "IFPBCajazeiras/usuario01/setpoint", permitindo receber mensagens de controle do motor e atualizações de *setpoint*. Em caso de falha na conexão, o código exibe o erro retornado pelo broker e aguarda dois segundos antes de tentar novamente, garantindo que a ESP32 permaneça conectada e pronta para comunicação contínua.

A função callback_mqtt é chamada sempre que uma mensagem chega em um dos tópicos assinados. O trecho do código responsável por tratar as mensagens recebidas por meio dessa função é apresentado na Figura 22.

Figura 22 – Implementação responsável por tratar as mensagens recebidas pela ESP32.

```
// ========== FUNÇÃO DE CALLBACK PARA MENSAGENS MQTT ==========
void callback_mqtt(char *topico, byte *mensagem, unsigned int comprimento) {
  String conteudo = "";
  for (int i = 0; i < comprimento; i++) {
    conteudo += (char)mensagem[i];  // Converte a mensagem recebida para string
  }

  if (String(topico) == "IFPBCajazeiras/usuario01/acionamento") {
    // Controle de acionamento do motor via MQTT
    if (conteudo == "true") {
      estadoMQTT = true;  // Ativa o controle via MQTT
      Serial.println("Node-RED: LIGAR");
    } else if (conteudo == "false") {
      estadoMQTT = false;  // Desativa o controle via MQTT
      Serial.println("Node-RED: DESLIGAR");
    }
  } else if (String(topico) == "IFPBCajazeiras/usuario01/setpoint") {
    // Atualiza o setpoint com a mensagem recebida via MQTT
    setpoint = conteudo.toFloat();  // Converte a string recebida para um valor float
    Serial.print("Novo setpoint recebido: ");
    Serial.println(setpoint);  // Exibe o novo setpoint no Serial Monitor
  }
}
```

Fonte: Autoria própria (2025).

O conteúdo da mensagem recebida é convertido para *string* e, em seguida, tratado de acordo com o tópico. No tópico "IFPBCajazeiras/usuario01/acionamento", a mensagem define o estado do motor, ativando-o quando o valor recebido é "*true*" e desativando-o quando é "*false*". Já no tópico "IFPBCajazeiras/usuario01/setpoint", a mensagem recebida é convertida para o tipo *float* e atribuída à variável *setpoint*, atualizando o valor de referência utilizado no controle de velocidade.

A publicação das mensagens no broker MQTT é realizada por meio da função client.publish(), que envia dados em formato de *string* para os tópicos configurados. A Figura 23 apresenta a implementação dessa função no código desenvolvido.

Figura 23 – Implementação responsável por realizar publicação da mensagem de velocidade.

```
    // Publica MQTT
    String mensagem = String(rpm_saida);
    char msgChar[20];
    mensagem.toCharArray(msgChar, 20);
    client.publish("IFPBCajazeiras/usuario01/velocidade", msgChar);
```

Fonte: Autoria própria (2025).

Esse trecho do código mostra como a velocidade do motor, calculada e armazenada na variável rpm_saida, é preparada e enviada ao broker MQTT. Inicialmente o valor é convertido para o tipo s*tring* e em seguida transformado em um vetor de caracteres por meio da função toCharArray(), já que a função client.publish() exige esse formato para publicar a mensagem. Por fim, o dado é enviado ao tópico "IFPBCajazeiras/usuario01/velocidade". A publicação do

erro e do tempo acumulado segue exatamente a mesma lógica, diferenciando-se apenas pelo valor utilizado e pelo tópico de destino.

b) Implementação do controlador PID digital

O controlador PID com filtro derivativo e anti *wind-up* foi implementado digitalmente utilizando as Equações 16, 20 e 29 apresentadas na Seção 3.6 (páginas 35 e 36), que correspondem, respectivamente, às componentes proporcional, integral e derivativa em sua forma discreta.

Por tanto, resumidamente a implementação seguiu essas etapas:

- Cálculo do erro: A cada período de amostragem de 50ms, calcula-se a diferença entre o *setpoint* e a variável de processo (velocidade medida pelo *encoder*);

- Ação proporcional: Obtida a partir do produto direto do ganho proporcional com erro, como descrito na Equação 16;

- Ação integral com anti *wind-up*: A integral é atualizada recursivamente conforme a discretização obtida via método de Tustin, de acordo com a Equação 20. O termo de *back-calculation*, ponderado por T_t, é utilizado para corrigir a acumulação da integral quando ocorre saturação na saída;

- Ação derivativa filtrada: A derivada é implementada conforme a Equação 29, obtida a partir da discretização por Tustin com filtro de primeira ordem, reduzindo ruídos de alta frequência;

- Sinal de controle: O valor não saturado do controlador é obtido pela soma das três ações: proporcional, integral e derivativo;

- Saturação e anti *wind-up*: O sinal de controle é limitado ao intervalo de atuação do PWM (0 a 255). O valor saturado é então comparado com o sinal de controle e a diferença entre ambos é realimentada na equação da integral pelo método de *back-calculation*, evitando o acúmulo excessivo da ação integral durante a saturação.

O trecho do código referente a essa implementação está ilustrado na Figura 24.

Figura 24 – Implementação digital do controlador PID.

```
// Erro atual
float erro = setpoint - rpm_saida;

// 1. Termo proporcional
float P = Kp * erro;

// 2. Termo integral com anti-windup
float I = I_anterior + (T/2)* Ki * (erro + erro_anterior) + (T / (2*Tt)) * (es + es_anterior);

// 3. Termo derivativo com filtro
float d = A * (erro - erro_anterior) - B * d_anterior;

// 4. Soma dos termos para obter a saída do controlador
float u = P + I + d;

// 5. Implementação da saturação
float u_sat = u;
if (u > u_max) {
  u_sat = u_max;
} else if (u < u_min) {
  u_sat = u_min;
}

// 6. Cálculo do erro de saturação para anti-windup
es_anterior = es;
es = u_sat - u;

// 7. Atualização das variáveis de estado para a próxima iteração
erro_anterior = erro;
I_anterior = I;
d_anterior = d;
```

Fonte: Autoria própria (2025).

O pré-cálculo dos coeficientes do termo derivativo e das variáveis de estado do controlador PID é implementado na função iniciar_pid, conforme é apresentado na Figura 25.

Figura 25 – Função para inicialização do controlador PID.

```
void iniciar_pid() {
  // Pré-cálculo dos coeficientes para o termo derivativo
  A = Kd / (alpha * Td + T/2);
  B = (T/2 - alpha * Td) / (alpha * Td + T/2);

  // Inicializa variáveis de estado do PID
  erro_anterior = 0.0;
  I_anterior = 0.0;
  d_anterior = 0.0;
  es = 0.0;
  es_anterior = 0.0;
}
```

Fonte: Autoria própria (2025).

O código-fonte completo, incluindo as bibliotecas utilizadas e a lógica implementada, encontra-se disponível no Apêndice B ao final deste trabalho.

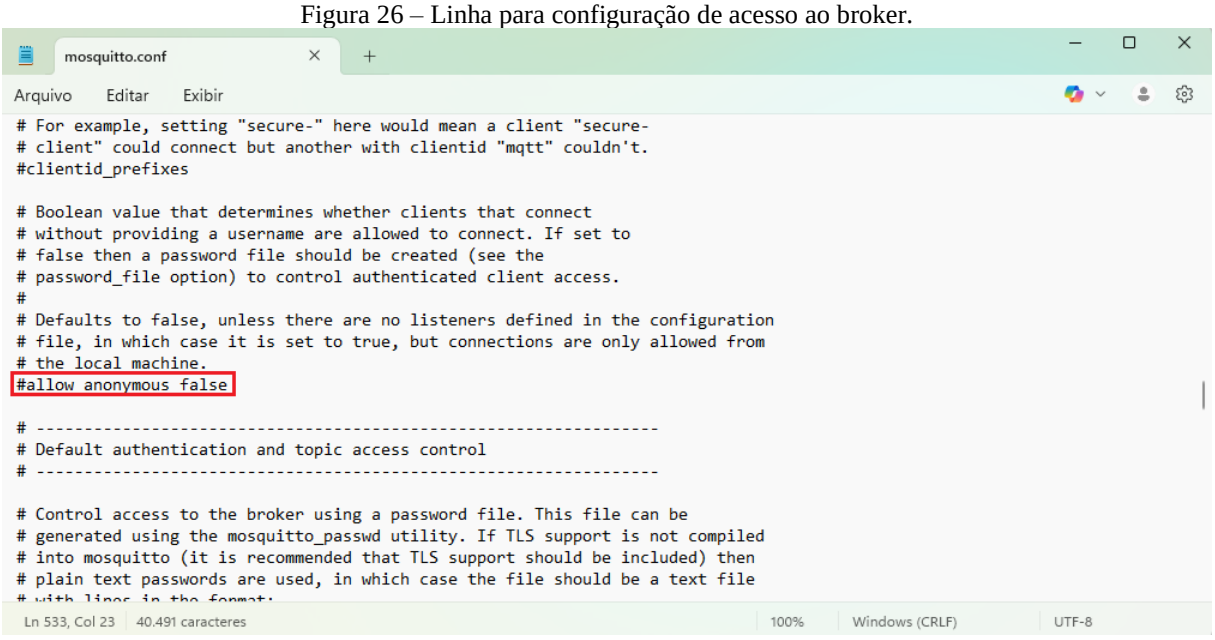
4.4 SUPERVISÓRIO LOCAL COM NODE-RED E BROKER MQTT

Com o intuito de realizar a supervisão do sistema de controle de velocidade de maneira local, foi implementado um supervisório utilizando a plataforma Node-RED, que opera em conjunto com um broker MQTT configurado no próprio notebook utilizado durante os testes. Para isso, empregou-se o Mosquitto, uma implementação leve e amplamente utilizada do protocolo MQTT, instalada e executada localmente.

O principal objetivo dessa etapa foi garantir que a comunicação entre o microcontrolador da ESP32 e o sistema supervisório ocorresse exclusivamente por meio do broker local, sem necessidade de conexão com a internet ou uso de serviços em nuvem. Dessa forma, o notebook atuou como um servidor MQTT local, assumindo o papel de intermediário na troca de mensagens entre os dispositivos da rede interna.

A configuração do Mosquitto no notebook foi necessária para este dispositivo se tornar um servidor local MQTT. Na pasta onde o Mosquitto foi instalado contém o arquivo *mosquito.conf*, ao abri-lo pelo bloco de notas como administrador alterou-se o texto na linha mostrada na Figura 26 para *allow_anonymous true*.

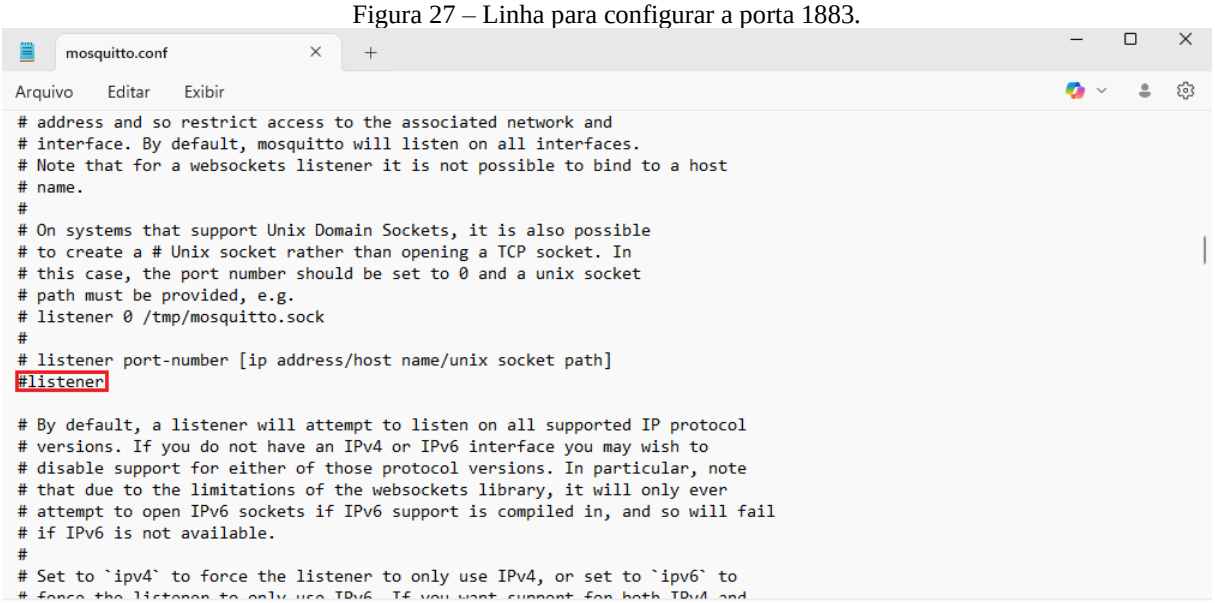
Figura 26 – Linha para configuração de acesso ao broker.



Fonte: Autoria própria (2025).

Da mesma forma, foi alterado o texto mostrado na Figura 27 para *listener 1883 0.0.0.0*. Isso irá possibilitar que a porta 1883 seja liberada para toda a internet.

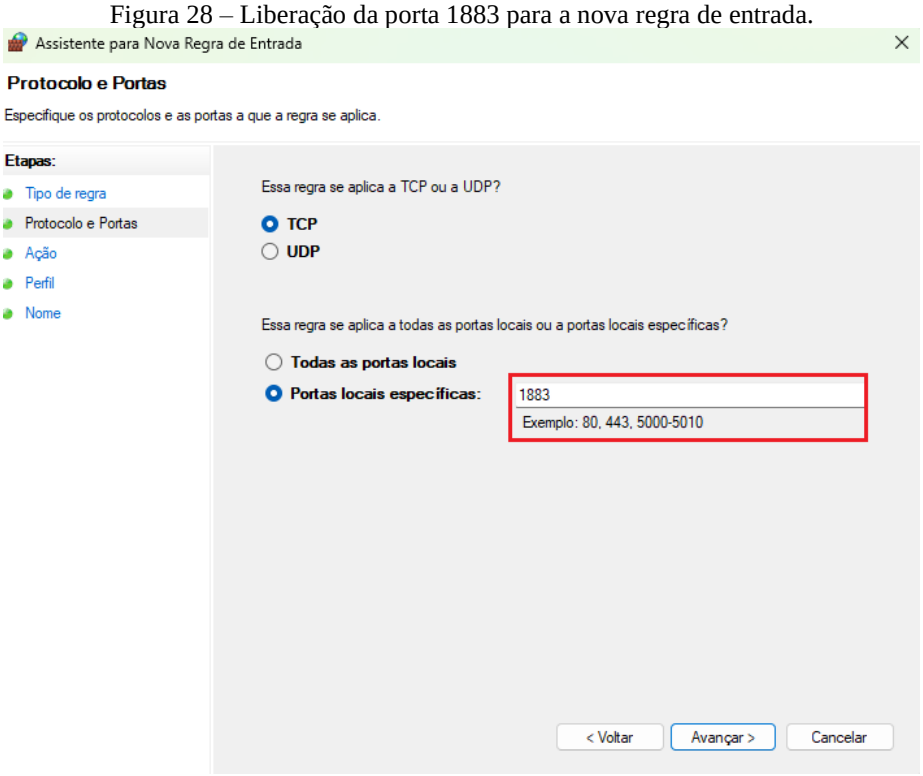
Figura 27 – Linha para configurar a porta 1883.



Fonte: Autoria própria (2025).

No entanto, somente essas configurações não é suficiente, pois é necessário fazer algumas alterações nas regras pré-definidas de entrada e saída no *Firewall* do *Windows Defender*, para permitir a conexão com a porta 1883. Para tanto, foram criadas uma regra de entrada e uma regra de saída, com a liberação da porta 1883, como exemplifica a Figura 28.

Figura 28 – Liberação da porta 1883 para a nova regra de entrada.



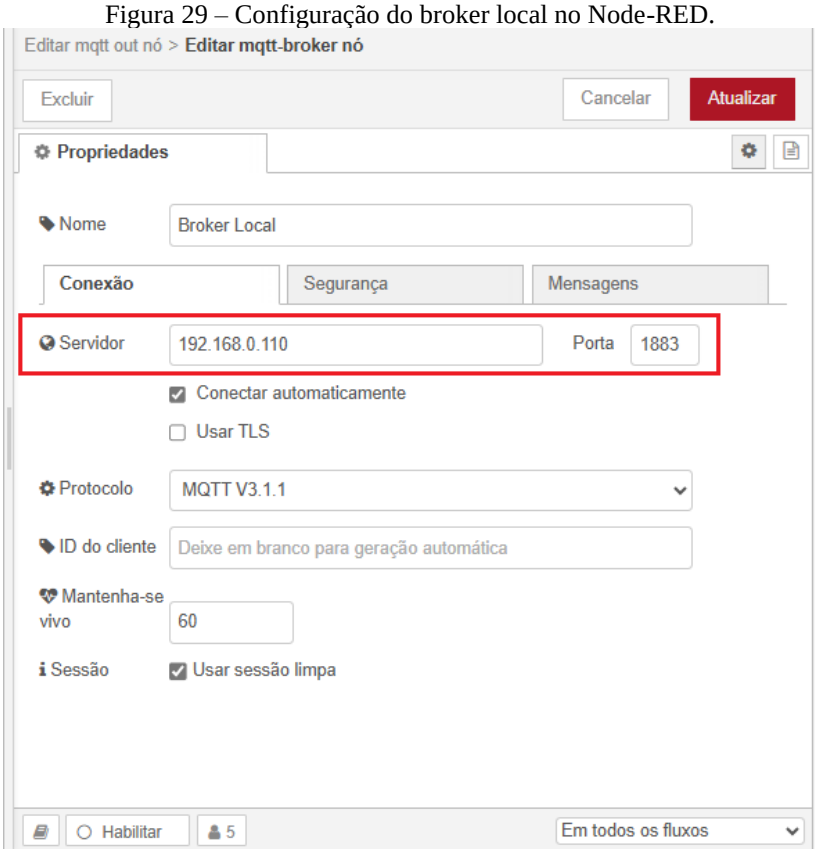
Fonte: Autoria própria (2025).

De forma semelhante, a nova regra de saída foi criada com a liberação da porta 1883.

O Node-RED, por sua vez, foi configurado para se conectar ao broker Mosquitto,

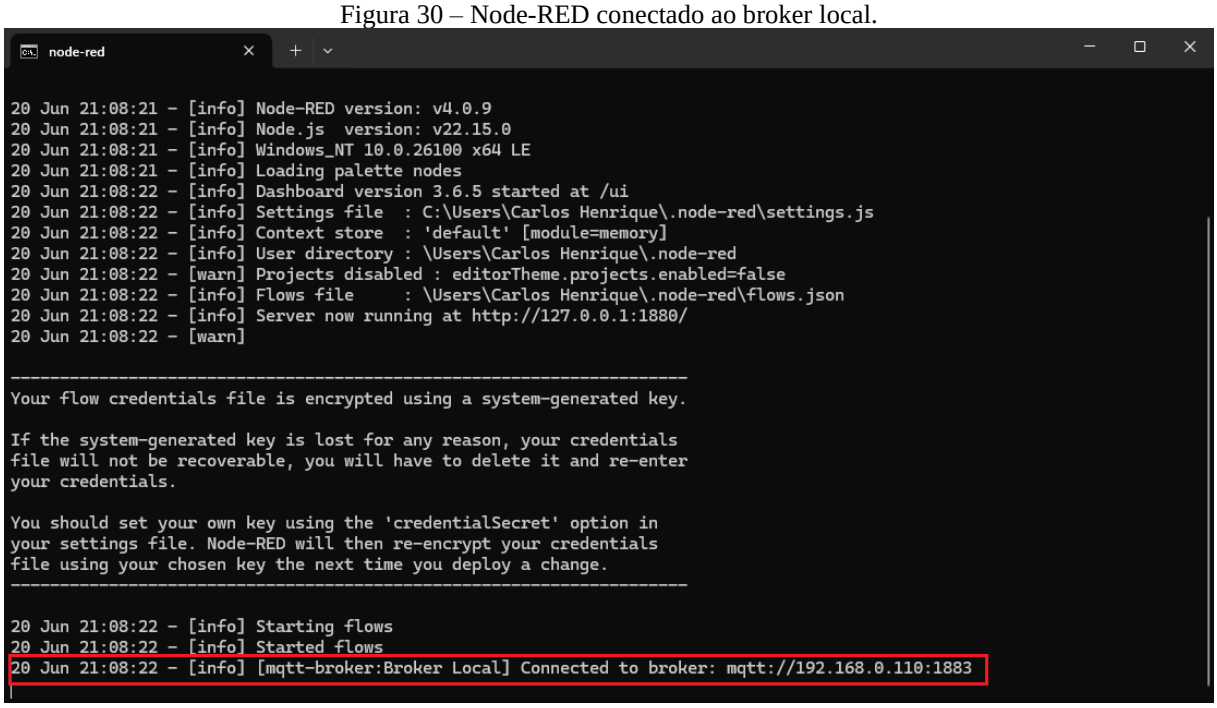
recebendo e publicando mensagens em tópicos específicos relacionados ao sistema de controle. A configuração do broker no Node-RED foi feita com a utilização de um dos nós de comunicação MQTT, inserindo neste nó, o endereço IP do notebook utilizado. A Figura 29 mostra o local onde é inserido o endereço IP do servidor no nó de rede *mqtt out*, mais abaixo na Figura 30 tem-se a indicação de que o broker local foi conectado com sucesso no Node-RED.

Figura 29 – Configuração do broker local no Node-RED.



Fonte: Autoria própria (2025).

Figura 30 – Node-RED conectado ao broker local.



Fonte: Autoria própria (2025).

Os nós de rede *mqtt in* e *mqtt out*, ilustrados na Figura 31, são responsáveis por realizar a comunicação com o ESP32. O nó *mqtt in* foi utilizado para receber os dados enviados pelo microcontrolador, como a velocidade do motor publicada em um tópico específico. Por outro lado, o nó *mqtt out* teve como função publicar os dados de escrita enviados ao ESP32, como os comandos de acionamento e alteração do *setpoint*, por meio dos respectivos tópicos.

Tópicos utilizados no projeto para a comunicação com o broker local:

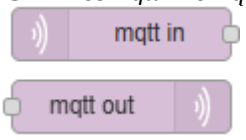
-IFPBCajazeiras/usuario01/acionamento.

-IFPBCajazeiras/usuario01/velocidade.

-IFPBCajazeiras/usuario01/setpoint.

-IFPBCajazeiras/usuario01/erro.

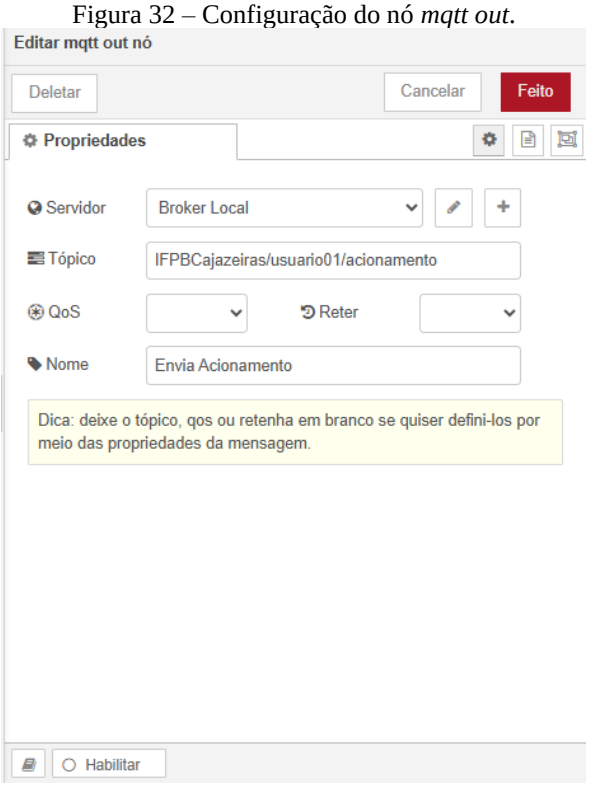
Figura 31 – Nós *mqtt in* e *mqtt out*.



Fonte: Autoria própria (2025).

Na Figura 32 observa-se a configuração realizada no nó *mqtt out* com o preenchimento dos campos Servidor e Tópico. Em Servidor inseriu-se o broker local já configurado como mostrado na Figura 29 e o campo Tópico foi preenchido com o tópico onde serão publicados os dados de acionamento do sistema.

Figura 32 – Configuração do nó *mqtt out*.



Fonte: Autoria própria (2025).

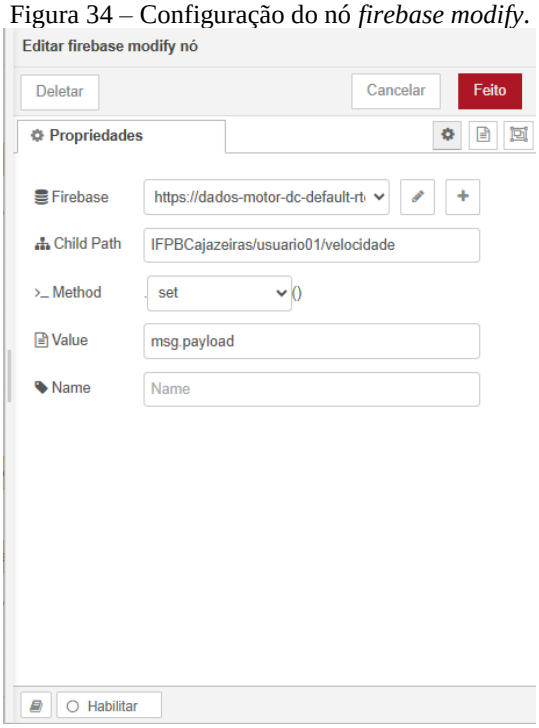
A integração do supervisório local com o *Realtime Database* da plataforma *Firebase* foi realizada utilizando os nós da biblioteca *node-red-contrib-firebase*, especialmente os nós *firebase modify* e *firebase.on()*, indicados na Figura 33. O nó *firebase modify* foi configurado com o método *.set()*, permitindo a escrita de dados no banco de dados em nuvem, incluindo comandos de acionamento, valores de *setpoint*, bem como dados de velocidade e erro coletados pela ESP32, garantindo a sincronização dessas informações com o supervisório remoto. Já o nó *firebase.on()* monitora em tempo real as alterações no banco, captando comandos e ajustes enviados pelo supervisório remoto, e encaminhando-os para os nós *mqtt out*, que publicam essas informações nos tópicos MQTT correspondentes (como IFPBCajazeiras/usuario01/acionamento), para que a ESP32 receba as atualizações. Dessa forma, o uso combinado desses nós permite a comunicação bidirecional eficiente entre supervisórios local e remoto, utilizando o banco *Firebase* como intermediário para troca e sincronização dos dados.

Figura 33 – Nós *firebase modify* e *firebase.on()*.

Fonte: Autoria própria (2025).

A Figura 34 exibe a configuração do nó *firebase modify*, utilizado para realizar a escrita de dados no banco. Neste exemplo, o caminho definido no campo *Child Path* foi IFPBCajazeiras/usuario01/velocidade, que corresponde ao local destinado ao armazenamento da velocidade atual do motor coletada pela ESP32. O método escolhido foi *.set()*, o qual sobrescreve o valor existente no banco. Em *Value*, foi utilizado *msg.payload*, de forma que o valor da mensagem recebida pelo nó seja escrito diretamente no banco de dados.

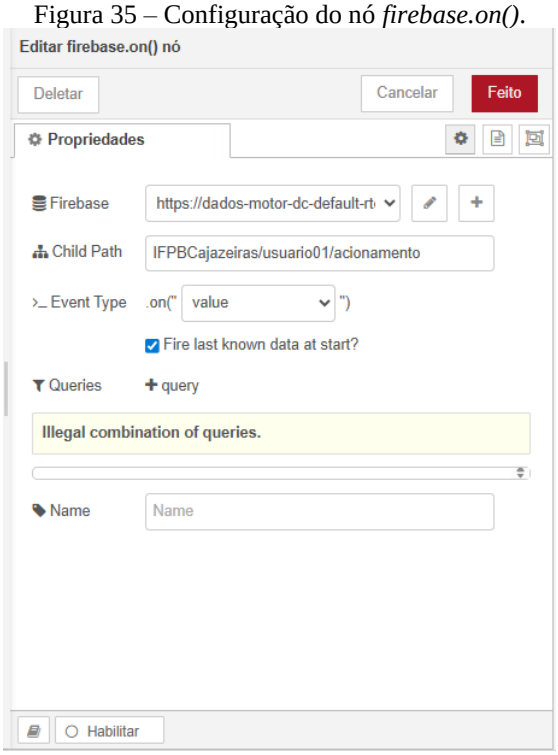
Figura 34 – Configuração do nó *firebase modify*.



Fonte: Autoria própria (2025).

A Figura 35 apresenta a interface de configuração do nó *firebase.on()*, responsável por escutar alterações no banco de dados. No campo *Firebase*, foi vinculada a URL do *Realtime Database* do projeto (https://dados-motor-dc-default-rtdb.firebaseio.com/). Em *Child Path*, foi definido o caminho IFPBCajazeiras/usuario01/acionamento, que corresponde ao local onde o supervisório remoto escreve o comando de acionamento. O campo *Event Type* foi configurado como *value*, o que indica que qualquer alteração no valor deste caminho acionará o nó. A opção *Fire last known data at start* foi marcada para que, ao iniciar o fluxo, o nó envie o valor atual presente no banco. Quando esse nó detecta uma alteração, ele emite uma mensagem com o valor recebido, que pode ser repassada ao tópico MQTT da ESP32.

Figura 35 – Configuração do nó *firebase.on()*.



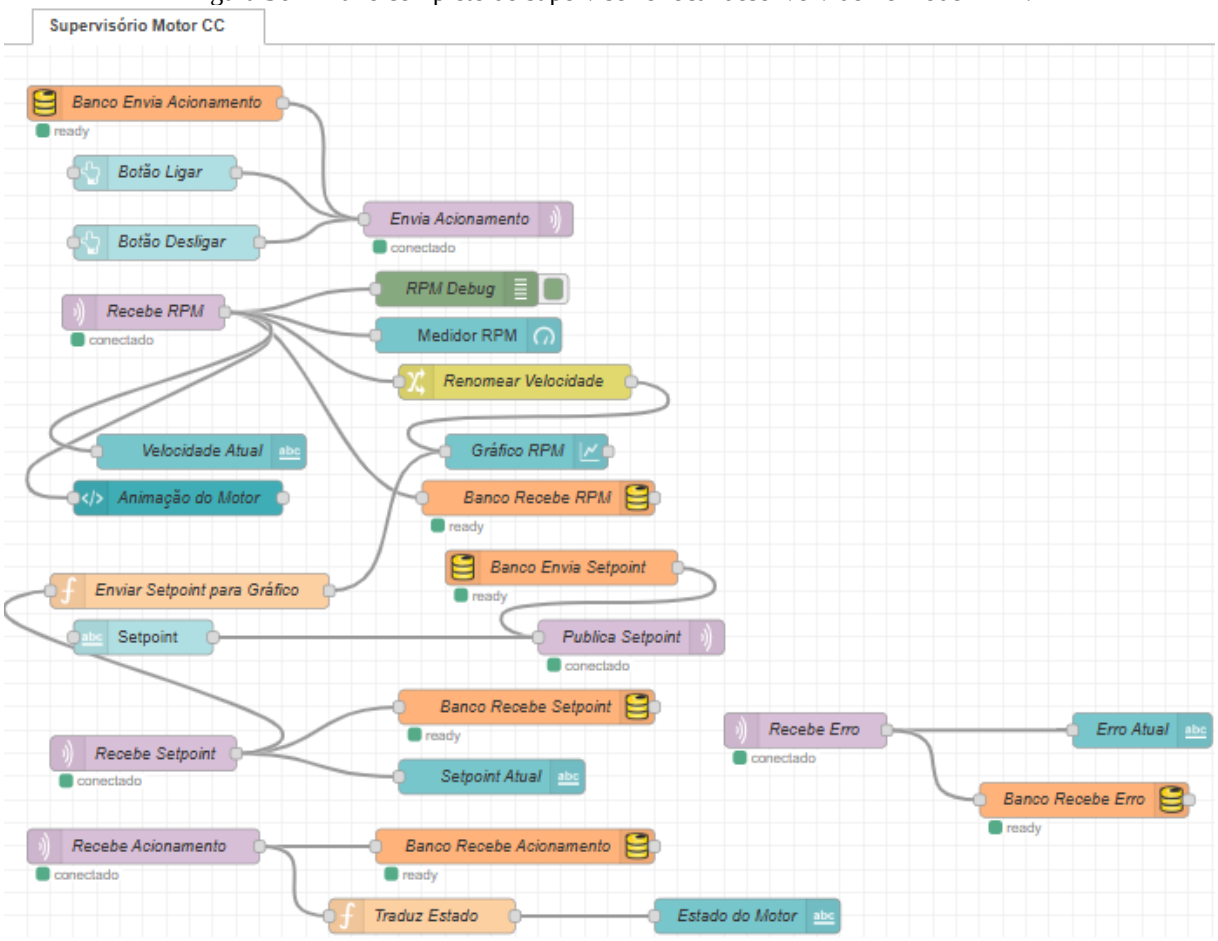
Fonte: Autoria própria (2025).

A utilização desses dois nós permite a comunicação bidirecional entre o supervisório local e o supervisório remoto, com o *Firebase* servindo como intermediário. Por meio do nó *firebase.modify*, o sistema local envia dados relevantes como a velocidade do motor. Já com o *firebase.on()*, o sistema capta comandos externos, como o acionamento enviado a partir do *website* remoto, garantindo a atualização em tempo real dos dados na ESP32.

Foram criados fluxos lógicos que permitiam a visualização em tempo real da velocidade do motor, bem como o acionamento do sistema, definição de *setpoints* para o controle PID e animações que indica o funcionamento do motor. A plataforma também foi utilizada para exibir, de forma gráfica, os dados de velocidade em RPM enviados pelo *encoder* acoplado ao motor.

A Figura 36 apresenta o fluxo completo desenvolvido no Node-RED, com todos os nós interconectados para garantir o funcionamento do sistema.

Figura 36 – Fluxo completo do supervisório local desenvolvido no Node-RED.



Fonte: Autoria própria (2025).

Essa estrutura local permitiu realizar testes de desempenho e validação do sistema de forma independente de serviços externos, promovendo maior confiabilidade e controle sobre o ambiente de execução. A configuração do broker Mosquitto para operação local, em comunicação com a ESP32 por meio de uma rede Wi-Fi dedicada, reflete práticas adotadas em ambientes industriais, nos quais muitas aplicações utilizam redes locais para assegurar maior estabilidade e simplicidade de comunicação. Entretanto, é importante destacar que o setor industrial também emprega redes padronizadas e conectadas, com arquiteturas de segurança em camadas, garantindo confiabilidade, escalabilidade e proteção dos dados.

A Figura 37 apresenta a interface completa do supervisório local exibida ao usuário no navegador.

Figura 37 – Interface do supervisório local desenvolvida com Node-RED Dashboard.

Fonte: Autoria própria (2025).

4.5 DESENVOLVIMENTO DOS CAMINHOS DO REALTIME DATABASE

Para a viabilização do supervisório remoto deste projeto, foi fundamental a integração com uma base de dados em nuvem. Dentre as alternativas disponíveis, optou-se pelo *Firebase Realtime Database*, uma ferramenta da Google que permite armazenar e sincronizar dados entre usuários em tempo real. A escolha por esse serviço foi motivada por sua compatibilidade com a plataforma web e pela facilidade de integração com aplicações desenvolvidas em JavaScript, além de sua estrutura baseada em formato JSON, que possibilita a organização hierárquica dos dados.

A primeira etapa desse desenvolvimento consistiu na configuração do projeto no *Firebase*, por meio da criação de uma aplicação na plataforma. Após essa criação, o aplicativo ”supervisório-web” é salvo nas configurações do projeto, e dessa forma o *Firebase* disponibiliza um conjunto de parâmetros de conexão, responsável por inicializar a aplicação e permitir o acesso aos serviços da nuvem. Essa configuração inclui informações como chave de autenticação, endereço do banco de dados, identificadores do projeto e outros dados essenciais para que a aplicação *web* se comunique corretamente com o *Realtime Database*. A Figura 38 apresenta a interface da plataforma *Firebase* na aba de configurações do projeto, destacando o aplicativo *web* criado para o supervisório remoto.

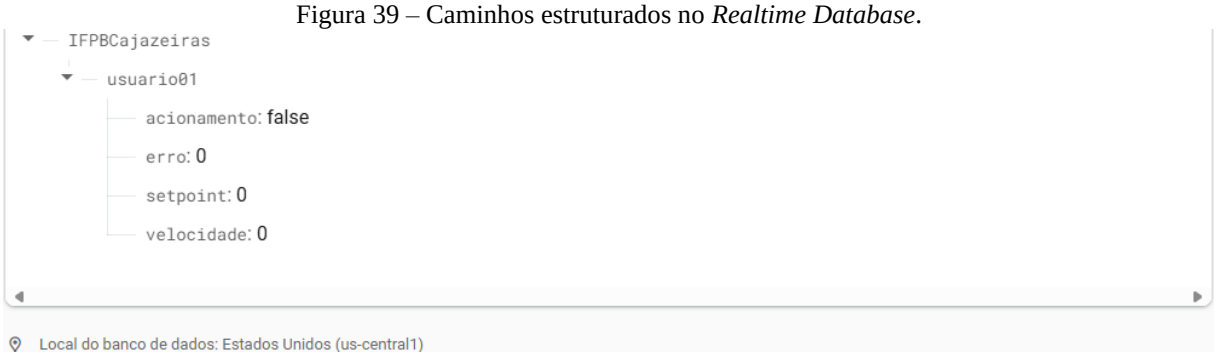
Figura 38 – Tela de configurações do projeto no Firebase com identificação do aplicativo “supervisorio-web”.

Fonte: Autoria própria (2025).

Esse conjunto de informações foi fornecido à inteligência artificial da plataforma *Firebase Studio*, permitindo que ela estabelecesse a integração automática com o banco de dados em nuvem, etapa fundamental para o funcionamento do supervisório remoto. A partir dessa integração, tornou-se possível a comunicação entre o sistema local (composto pela ESP32 e pelo Node-RED) e o ambiente remoto (hospedado na Vercel), assegurando que qualquer alteração no sistema físico fosse refletida instantaneamente na interface remota.

Após a integração, foram definidos os caminhos de dados no *Realtime Database*, representando a estrutura lógica de armazenamento e leitura das variáveis de controle. A Figura 39 apresenta a árvore de dados implementada no projeto:

Figura 39 – Caminhos estruturados no *Realtime Database*.



Fonte: Autoria própria (2025).

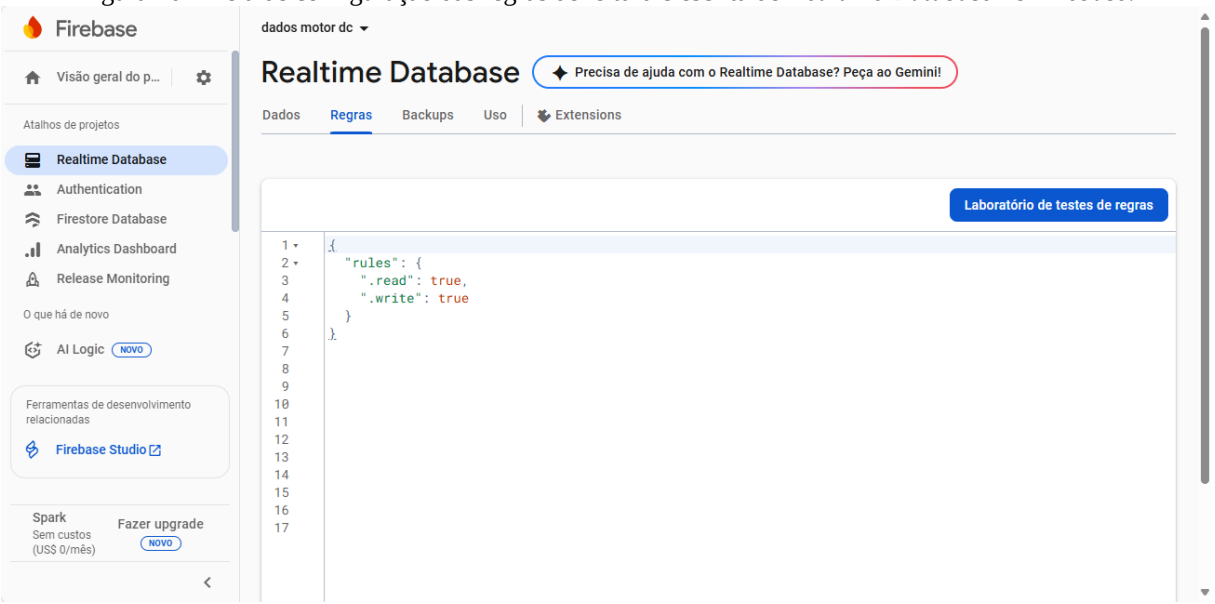
Como mostra a figura, os dados foram organizados a partir do nó principal “IFPBCajazeiras”, seguido pelo nó “usuario01”, que agrupa as variáveis necessárias para o monitoramento e controle do motor de corrente contínua. Nesse nível, estão incluídas as variáveis acionamento, um valor booleano que indica o estado do motor (ligado ou desligado); *setpoint*, um valor numérico que define a referência de velocidade desejada; erro, que representa

numericamente a diferença entre a velocidade real e o *setpoint*; e velocidade, expressa em rotações por minuto (RPM), correspondente à velocidade medida do motor.

Cada uma dessas variáveis pode ser lida ou modificada tanto pela ESP32 quanto pelo supervisório remoto, garantindo sincronização em tempo real e bidirecional.

Para assegurar a integridade e o funcionamento correto do sistema, também foram definidas regras de acesso ao banco de dados, responsáveis por autorizar ou restringir ações de leitura e escrita. Essas regras foram configuradas na aba “Regras” da plataforma *Firebase* e, durante a fase de testes e validações do projeto, foram ajustadas para permitir acesso livre. Essa flexibilização possibilitou que o sistema fosse testado em diferentes cenários e com acesso remoto pleno. Futuramente, recomenda-se a implementação de autenticação de usuários para reforçar a segurança do sistema. A Figura 40 apresenta as regras definidas no *Realtime Database*, permitindo leitura e escrita livre durante o desenvolvimento do projeto.

Figura 40 – Tela de configuração das regras de leitura e escrita do *Realtime Database* no *Firebase*.



Fonte: Autoria própria (2025).

Por fim, destaca-se que a configuração do projeto no *Firebase* foi essencial para que a inteligência artificial utilizada no *Firebase Studio* conseguisse gerar o supervisório remoto com sucesso. Com base nas informações fornecidas, a IA conseguiu autenticar a aplicação, identificar corretamente o banco de dados e estruturar os caminhos necessários para leitura e escrita de dados. Assim, essa configuração funcionou como ponte entre o sistema físico e a interface remota, permitindo o monitoramento e controle do motor em tempo real por meio da *web*.

4.6 DESENVOLVIMENTO DO SUPERVISÓRIO REMOTO

O desenvolvimento do supervisório remoto seguiu uma abordagem baseada em tecnologias *web* integradas a serviços em nuvem, com o objetivo de permitir o acesso ao sistema de controle do motor CC a partir de qualquer local com conexão à internet.

A metodologia de implementação seguiu as seguintes etapas:

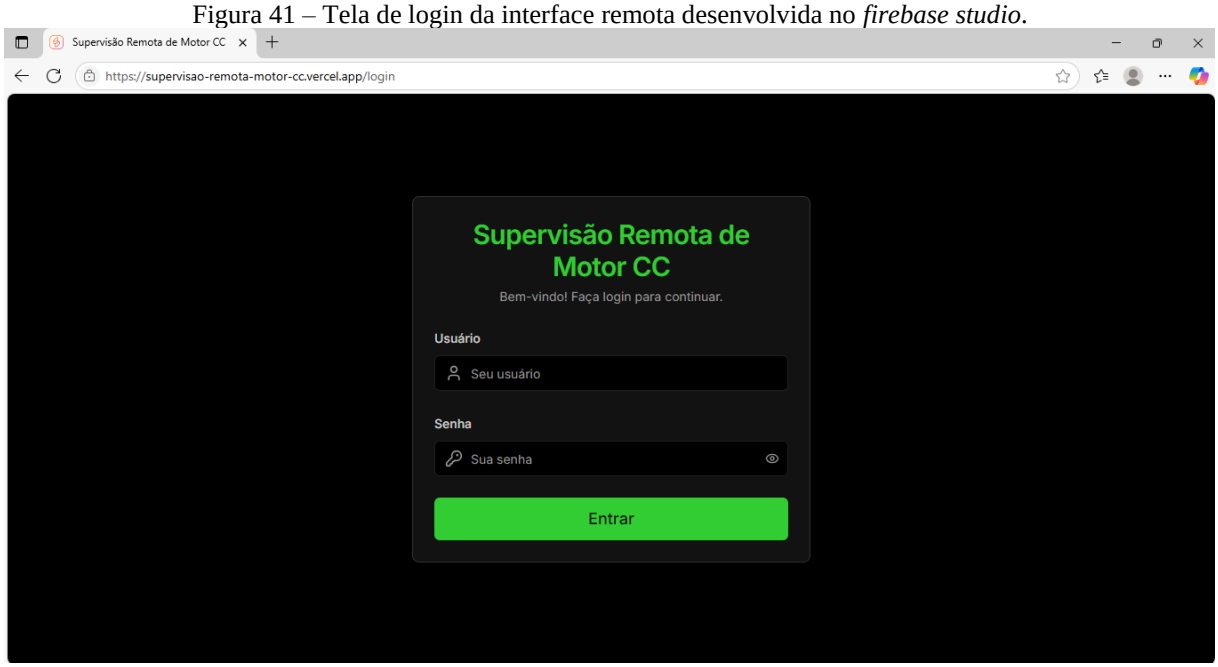
a) Criação da interface com *Firebase Studio*

A criação da interface do supervisório remoto foi realizada por meio da plataforma *Firebase Studio*, uma ferramenta experimental da Google baseada em inteligência artificial, voltada para o desenvolvimento acelerado de aplicações *web* integradas aos serviços do ecossistema *Firebase*, como o *Realtime Database*.

Durante o desenvolvimento, foram realizadas diversas interações e orientações à inteligência artificial do *Firebase Studio*, com o propósito de adequar o supervisório remoto às necessidades específicas do projeto. Essas interações permitiram ajustes precisos no *design*, na funcionalidade e na lógica de navegação, assegurando o atendimento aos requisitos de usabilidade e a integração eficiente com o banco de dados em tempo real.

A interface foi organizada em duas telas principais: uma tela inicial de autenticação e uma tela de supervisão, acessada mediante a validação das credenciais do usuário. A tela de login contempla campos para inserção de nome de usuário e senha. Neste trabalho, optou-se pela implementação de um controle de acesso baseado em credenciais fixas, restritas ao uso exclusivo da equipe responsável pelo projeto. Importa destacar que o sistema de autenticação foi concebido pela própria inteligência artificial do *Firebase Studio*, sem a utilização do serviço *Firebase Authentication*, adotando-se, portanto, um método simplificado de controle de acesso. A Figura 41 apresenta a tela de login desenvolvida.

Figura 41 – Tela de login da interface remota desenvolvida no *firebase studio*.



Fonte: Autoria própria (2025).

Na tela principal da aplicação foram inseridos os seguintes elementos:

Botões de acionamento e desligamento do motor, os quais ao serem acionados enviam valores lógicos (*true* ou *false*) para o caminho /IFPBCajazeiras/usuario01/acionamento do *Realtime Database*;

Campo numérico para inserção do *setpoint*, que grava o valor de referência desejado para a velocidade do motor no caminho /IFPBCajazeiras/usuario01/setpoint;

Gráfico dinâmico de linhas, que exibe em tempo real a comparação entre a velocidade medida e o *setpoint*, a partir da leitura dos caminhos /IFPBCajazeiras/usuario01/velocidade e /IFPBCajazeiras/usuario01/*setpoint*;

Indicador visual animado de um motor, representado por um desenho retangular com um eixo em um dos lados, que mostra uma hélice girando no eixo sempre que o motor entra em funcionamento. A mudança é determinada com base na leitura do valor atual no caminho de acionamento e no caminho de velocidade, ambos citados anteriormente;

Campo de visualização do erro de controle, que apresenta, em tempo real, o valor do erro obtido entre a velocidade medida e o *setpoint*. Esse dado é lido diretamente do caminho /IFPBCajazeiras/usuario01/erro;

As respectivas Figura 42, Figura 43 e Figura 44 ilustram a tela principal da interface com todos os elementos visuais implementados.

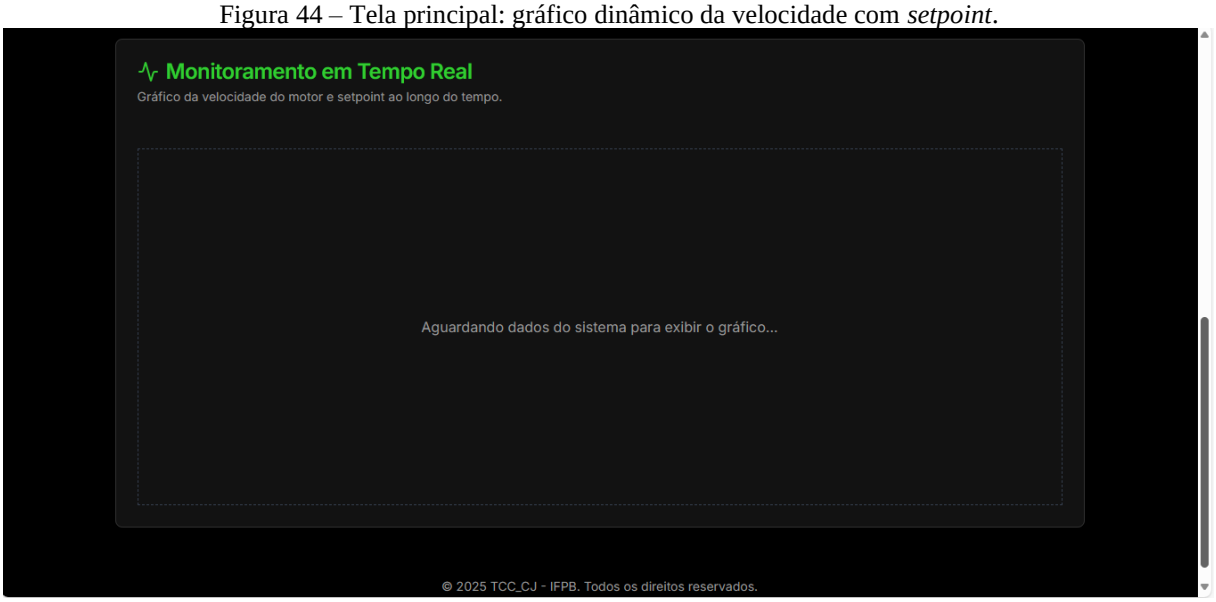
Figura 42 – Tela principal: grupos de acionamento e ajuste de *setpoint*.

Fonte: Autoria própria (2025).

Figura 43 – Tela principal: leitura da velocidade atual, *setpoint* e erro atual.

Fonte: Autoria própria (2025).

Figura 44 – Tela principal: gráfico dinâmico da velocidade com *setpoint*.



Fonte: Autoria própria (2025).

Esses componentes visuais foram configurados dentro da ferramenta Firebase Studio, com ajustes de layout, cores, tamanhos e animações, permitindo uma apresentação clara e objetiva das informações. A associação dos elementos visuais aos caminhos do *Realtime Database* foi feita por meio dos recursos nativos da ferramenta, sem necessidade de programação manual de código-fonte.

A integração com o banco de dados garante que toda interação do usuário, como acionamento do motor, alteração do *setpoint* e visualização de variáveis, ocorra de forma

síncrona e remota, refletindo em tempo real no sistema de controle físico, por meio do supervisório local e da ESP32.

Essa abordagem proporcionou uma solução visualmente atrativa, funcional e eficiente, com o uso de uma plataforma de desenvolvimento orientada por inteligência artificial, alinhada aos objetivos do projeto de controle remoto e supervisão distribuída.

b) Publicação do projeto no GitHub e *deploy* na Vercel

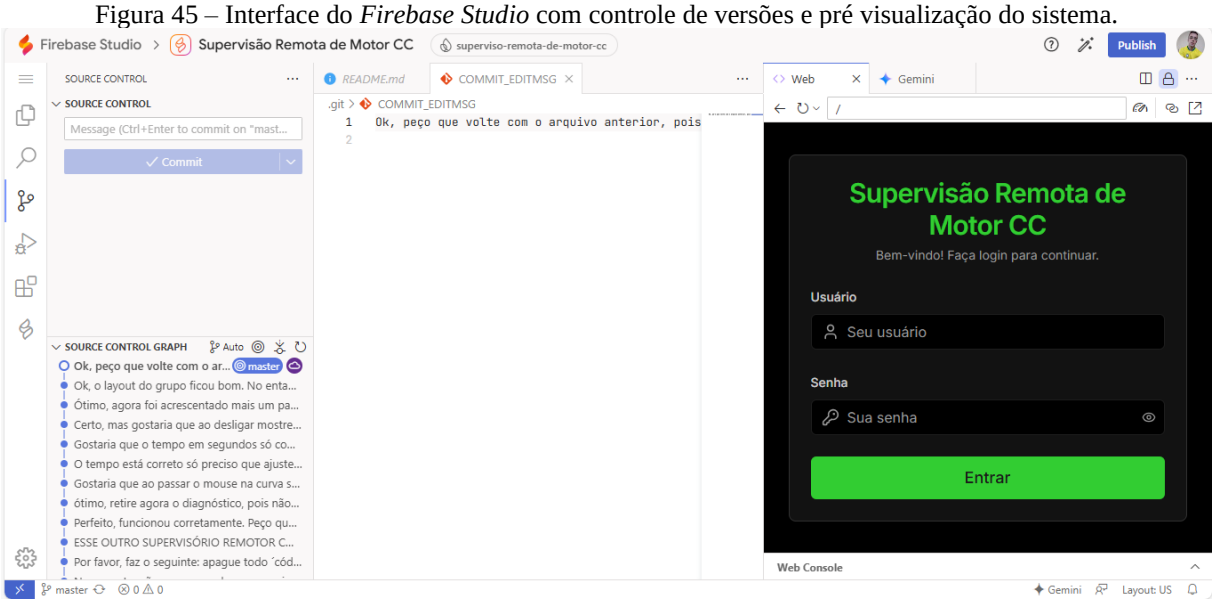
Após a finalização da interface do supervisório remoto na plataforma *Firebase Studio*, foi realizado o processo de publicação da aplicação, com o objetivo de disponibilizá-la para acesso remoto via navegador. As etapas envolveram o versionamento do projeto na plataforma GitHub e a hospedagem do sistema por meio da Vercel, permitindo que a solução fosse acessada pela internet em tempo real.

Com o projeto concluído, foi ativado o modo de edição avançada do *Firebase Studio* por meio da opção "*Switch to Code*", permitindo o gerenciamento de todos os arquivos do projeto e controle de versões. A partir disso, foi possível utilizar a aba "*Source Control*", que disponibiliza ferramentas para registrar alterações feitas na interface, descrever modificações por meio de mensagens e acompanhar o histórico visual de desenvolvimento.

Utilizando essa funcionalidade, o *Firebase Studio* fornece a opção para conexão com o GitHub, realizando a autenticação da conta e a criação de um repositório remoto. A partir desse momento, a plataforma do *Firebase* passou a sincronizar automaticamente as versões salvas com o repositório GitHub, garantindo organização, rastreabilidade e segurança ao projeto.

A Figura 45 ilustra o ambiente do *Firebase Studio* com a visualização do controle de versões e a interface do sistema sendo exibida em tempo real. É possível observar a opção *Commit*, que é a responsável por salvar as modificações e publicar no GitHub logo em seguida.

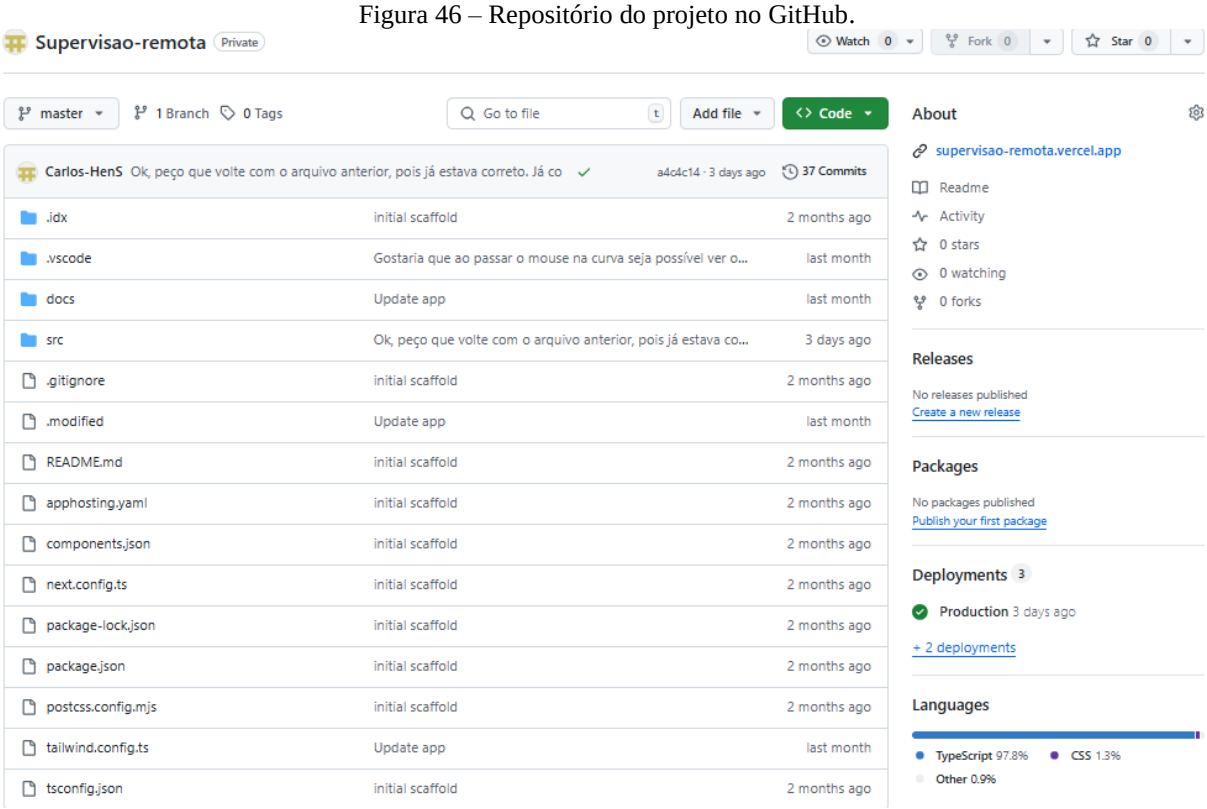
Figura 45 – Interface do *Firebase Studio* com controle de versões e pré visualização do sistema.



Fonte: Autoria própria (2025).

Por fim, com o projeto devidamente armazenado no GitHub conforme mostra a Figura 46, foi realizada a publicação da aplicação por meio da plataforma Vercel, que oferece suporte a aplicações *web* com integração direta a repositórios GitHub.

Figura 46 – Repositório do projeto no GitHub.



Fonte: Autoria própria (2025).

O processo de *deploy* foi feito em etapas simples: criação da conta na Vercel utilizando o login do GitHub, importação do repositório com o projeto do supervisório remoto, e configuração básica da aplicação para publicação. Após essa etapa, a Vercel executou

automaticamente a hospedagem da aplicação, tornando o sistema acessível em um endereço web público. A Figura 47 apresenta a tela de conclusão do *deploy* em ambiente de produção, confirmando que o supervisório remoto está acessível via navegador.

Figura 47 – Publicação do supervisório remoto na plataforma Vercel.

Fonte: Autoria própria (2025).

Uma das principais vantagens da Vercel é sua integração contínua, pois sempre que o projeto é atualizado e uma nova versão é registrada no GitHub, a Vercel realiza automaticamente a atualização da aplicação publicada, sem necessidade de ações manuais adicionais.

O endereço para acesso ao supervisório remoto pode ser consultado no Apêndice C.

4.7 IDENTIFICAÇÃO DO MODELO MATEMÁTICO DA PLANTA

A resposta de um motor CC a uma entrada do tipo degrau exibe, geralmente, o comportamento típico de um sistema de primeira ordem com atraso. Essa característica se reflete em uma curva com formato semelhante a um “S”, permitindo a aplicação do método de modelagem baseado na resposta ao degrau em malha aberta de Zigler-Nichols, conforme descrito na Equação 32 e discutido na Seção 3.9.1.

Para esse ensaio, foi aplicada uma entrada em degrau ao motor, expressa em volts, correspondente ao valor médio do sinal PWM gerado pelo ESP32. Esse sinal é entregue ao motor por meio de uma ponte H, que atua como driver de potência.

O canal PWM do ESP32 foi configurado com resolução de 8 bits, o que resulta em um intervalo de valores possível entre 0 e 255. Com base nisso, foi definido o valor 150 como nível do degrau aplicado, o que corresponde a um *duty cycle* de $150/255$ (aproximadamente

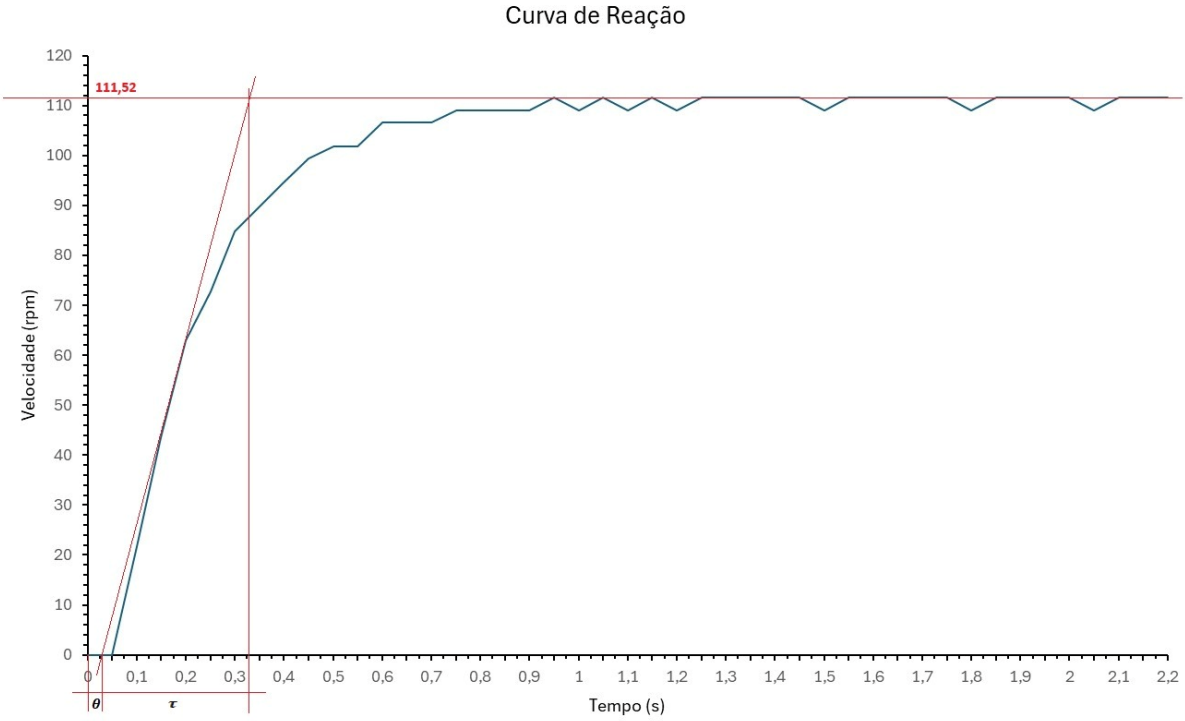
58,82%).

A escolha do valor 150 foi feita com base em dois critérios principais. Primeiro, para evitar *duty cycles* muito próximos de 100%, pois, nessa condição, o motor opera com a tensão máxima da fonte e já está trabalhando no seu limite. Além disso, foi observado que o *encoder* se tornava mais sensível a ruídos, dificultando a medição precisa da velocidade e comprometendo a análise da resposta ao degrau. Segundo, para manter uma amplitude suficientemente elevada que garanta uma boa relação sinal/ruído, evitando *duty cycles* muito baixos, que poderiam resultar em uma resposta fraca e dificultar a extração precisa dos parâmetros do sistema.

Dessa forma, a tensão média efetiva aplicada ao motor correspondente à entrada em degrau é dada pelo produto entre esse *duty cycle* $(150/255)$ e a tensão de alimentação do motor, de 6 V, conforme descrito na Equação 31. Portanto, a entrada aplicada ao motor pode ser interpretada como uma tensão média de aproximadamente 3,53 V. Entretanto, para garantir maior precisão, optou-se por medir diretamente a tensão média gerada pelo PWM com *duty cycle* de 58,82% utilizando um multímetro, devido às perdas inevitáveis causadas pela queda de tensão na ponte H. A medição resultou em um valor de 3,28 V.

Para a realização deste ensaio, utilizou-se o suplemento *Data Streamer* do Excel, com o objetivo de facilitar a formatação do gráfico e a extração dos parâmetros do processo, como o atraso, a constante de tempo e o ganho. O *Data Streamer* é um suplemento oficial da Microsoft que permite a comunicação bidirecional em tempo real entre o Excel e dispositivos externos, por meio da porta serial geralmente via conexão USB. Dessa forma, foi possível obter os resultados apresentados na Figura 48.

Figura 48 – Resposta do processo ao degrau de 3,28V em malha aberta.



Fonte: Autoria própria (2025).

Dessa forma, foram determinados a partir do gráfico o atraso θ, com valor aproximado de $0{,}026\text{s}$, e a constante de tempo τ, estimada em $0{,}301\text{s}$, resultante da diferença $0{,}327 - 0{,}026$.

O ganho do processo k é calculado como a razão entre a variação da saída e a variação da entrada, conforme apresentado na Seção 3.9.1. Assim, obtém-se:

$$k = \frac{111{,}52\ rpm}{3{,}28\ volts} \tag{35}$$

$$k = 34\frac{rpm}{volts} \tag{36}$$

Logo, a função de transferência aproximada do processo utilizando o método de Ziegler-Nichols da resposta ao degrau em malha aberta é dada por:

$$G_p(s) = \frac{34}{0{,}301s + 1}\mathrm{e}^{-0{,}026s} \tag{37}$$

4.8 APLICAÇÃO DAS TÉCNICAS DE SINTONIA

Com o modelo aproximado do processo devidamente identificado, foram aplicados os métodos de sintonia do controlador PID: Ziegler-Nichols, Cohen-Coon e IMC. Com o objetivo

de determinar os parâmetros de controle correspondentes. Essa etapa visa possibilitar uma análise comparativa de desempenho entre os métodos, considerando critérios como tempo de acomodação, tempo de subida e sobressinal.

a) Ziegler-Nichols

Ganho proporcional:

$$K_p = 1{,}2\left(\frac{0{,}301}{34 \times 0{,}026}\right) \tag{38}$$

$$K_p = 0{,}409 \tag{39}$$

Tempo integral:

$$T_i = 2 \times 0{,}026 \tag{40}$$

$$T_i = 0{,}052\ s \tag{41}$$

Tempo derivativo:

$$T_d = 0{,}5 \times 0{,}026 \tag{42}$$

$$T_d = 0{,}013\ s \tag{43}$$

Tabela 5 – Ajuste dos parâmetros de controle pelo método de Ziegler-Nichols.

Controlador	K_p	T_i	T_d
PID	0,409	0,052	0,013

Fonte: Autoria própria (2025).

Constante de tempo *wind-up*:

$$T_t = \sqrt{0{,}052 \times 0{,}013} \tag{44}$$

$$T_t = 0{,}026\ s \tag{45}$$

b) Cohen e Coon

Ganho proporcional:

$$K_p = \left(1{,}35 + 0{,}25\frac{0{,}026}{0{,}301}\right)\frac{0{,}301}{34 \times 0{,}026} \tag{46}$$

$$K_p = 0{,}467 \tag{47}$$

Tempo integral:

$$T_i = \frac{\left(1{,}35 + 0{,}25\frac{0{,}026}{0{,}301}\right)}{\left(0{,}54 + 0{,}33\frac{0{,}026}{0{,}301}\right)}0{,}026 \tag{48}$$

$$T_i = 0{,}0627\ s \tag{49}$$

Tempo derivativo:

$$T_d = \frac{0{,}5 \times 0{,}026}{\left(1{,}35 + 0{,}25\frac{0{,}026}{0{,}301}\right)} \tag{50}$$

$$T_d = 0{,}00948\ s \tag{51}$$

Tabela 6 – Ajuste dos parâmetros de controle pelo método de Cohen e Coon.

Controlador	K_p	T_i	T_d
PID	0,467	0,0627	0,00948

Fonte: Autoria própria (2025).

Constante de tempo *wind-up*:

$$T_t = \sqrt{0{,}0627 \times 0{,}00948} \tag{52}$$

$$T_t = 0{,}0244\ s \tag{53}$$

c) IMC

Conforme descrito na Seção 3.9.3, o parâmetro λ não deve ser inferior ao valor do atraso do processo, pois isso resultaria em uma sintonia excessivamente agressiva. Dessa forma, optou-se pela escolha de $\lambda = 0{,}03$, de modo a garantir um compromisso adequado entre desempenho e robustez.

Ganho proporcional:

$$K_p = \frac{2 \times 0{,}301 + 0{,}026}{34(2 \times 0{,}03 + 0{,}026)} \tag{54}$$

$$K_p = 0,215 \tag{55}$$

Tempo integral:

$$T_i = 0{,}301 + \frac{0{,}026}{2} \tag{56}$$

$$T_i = 0{,}314\ s \tag{57}$$

Tempo derivativo:

$$T_d = \frac{0{,}301 \times 0{,}026}{2 \times 0{,}301 + 0{,}026} \tag{58}$$

$$T_d = 0{,}0125\ s \tag{59}$$

Tabela 7 – Ajuste dos parâmetros de controle pelo método IMC.

Controlador	K_p	T_i	T_d
PID	0,215	0,314	0,0125

Fonte: Autoria própria (2025).

Constante de tempo *wind-up*:

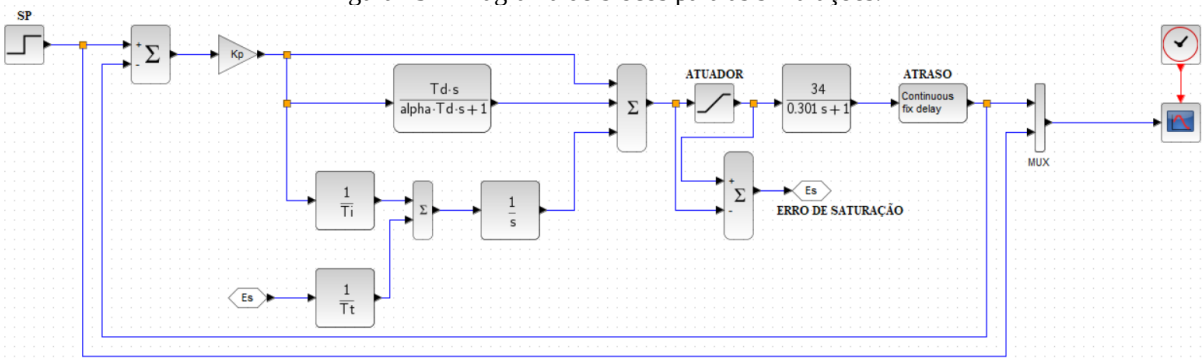
$$T_t = \sqrt{0{,}314 \times 0{,}0125} \tag{60}$$

$$T_t = 0{,}0627\ s \tag{61}$$

4.9 VALIDAÇÃO DAS TÉCNICAS DE CONTROLE

Para a validação preliminar das técnicas de controle, foram realizadas simulações no *software* Scilab, com o objetivo de observar o desempenho do sistema sob as sintonias obtidas pelos métodos de Ziegler-Nichols, Cohen-Coon e IMC, viabilizando uma posterior comparação com os resultados experimentais. A Figura 49 apresenta o diagrama de blocos utilizado nessas simulações.

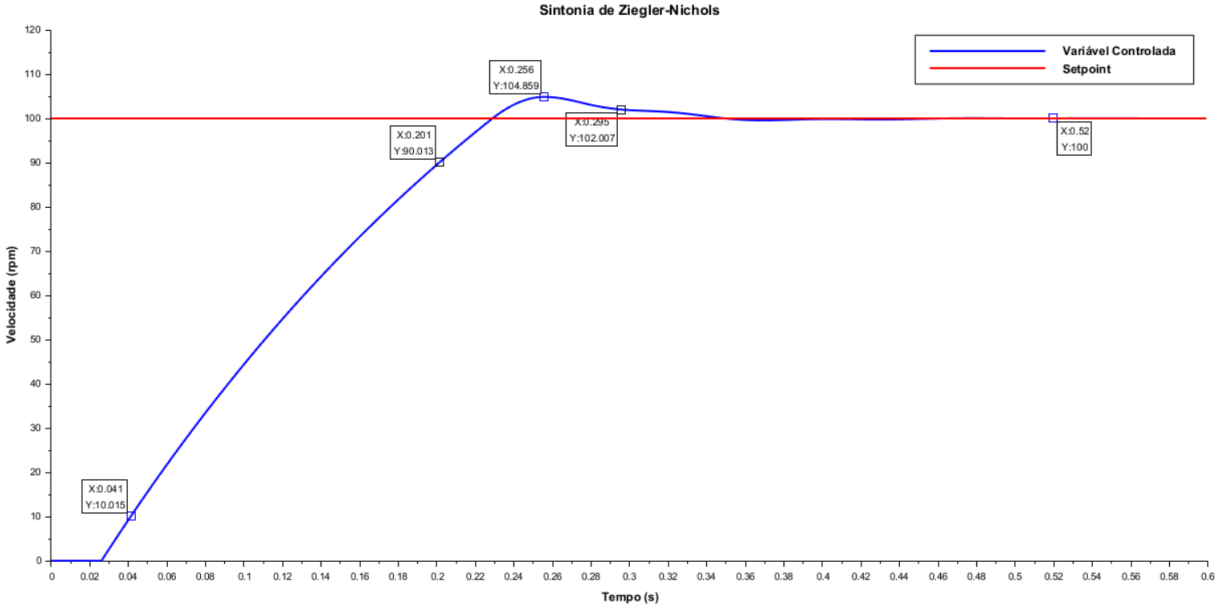
Figura 49 – Diagrama de blocos para as simulações.



Fonte: Autoria própria (2025).

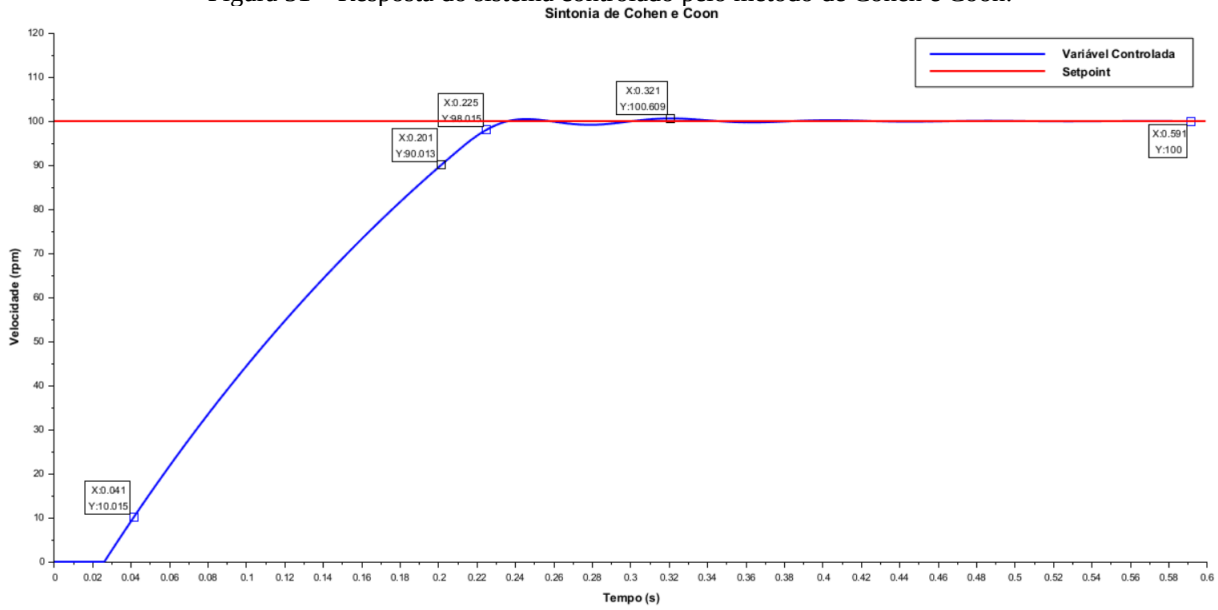
Como abordado previamente na Seção 3.5.1, o fator α(alpha) costuma assumir o valor aproximado de $^1/_8$. Com base nisso, foi adotado $\alpha = {}^1/_8 = 0{,}125$. As simulações das sintonias de controle foram então realizadas utilizando uma entrada degrau com amplitude de 100 rpm(*setpoint*):

Figura 50 – Resposta do sistema controlado pelo método de Ziegler-Nichols.



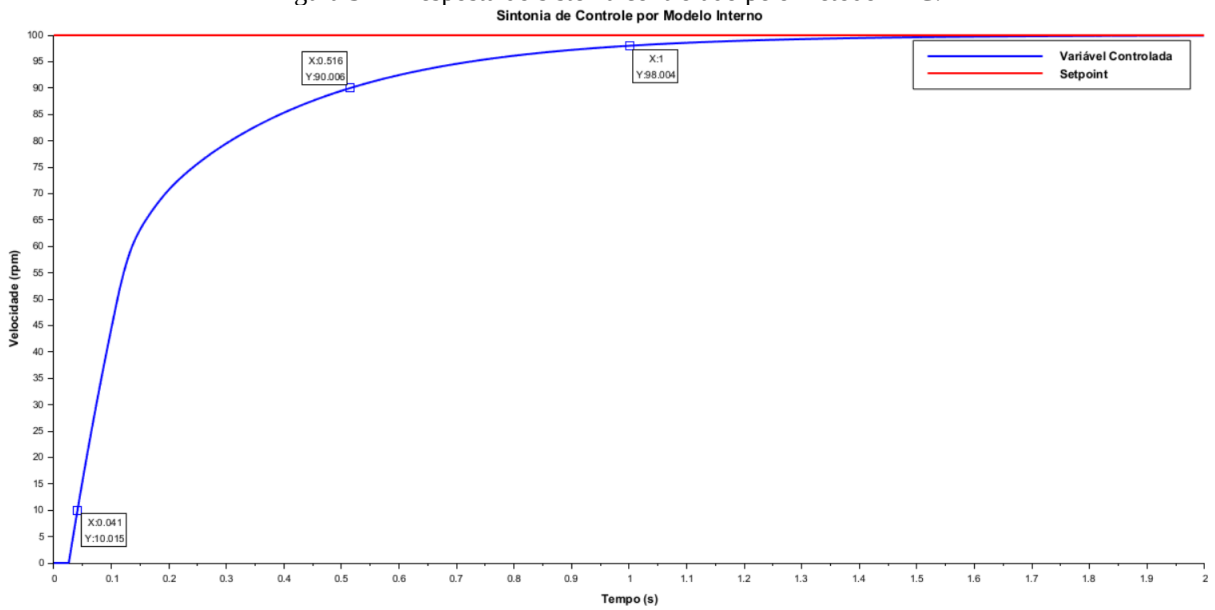
Fonte: Autoria própria (2025).

Figura 51 – Resposta do sistema controlado pelo método de Cohen e Coon.



Fonte: Autoria própria (2025).

Figura 52 – Resposta do sistema controlado pelo método IMC.



Fonte: Autoria própria (2025).

O tempo de acomodação T_s representa o intervalo necessário para que a resposta transitória atinja e permaneça dentro de uma faixa de $\pm 2\%$ em torno do valor final. Já o tempo de subida T_r corresponde ao tempo necessário para que a resposta se eleve de 10% a 90% do valor final. O sobressinal, ou ultrapassagem percentual, por sua vez, indica o quanto a resposta ultrapassa o valor de regime permanente no instante do pico máximo, sendo geralmente expresso como uma porcentagem do valor final. Esse parâmetro pode ser calculado por meio da Equação 62 (Nise, 2012).

$$\%UP = \frac{c_{max} - c_{final}}{c_{final}} \times 100 \tag{62}$$

$\%UP$: sobressinal ou ultrapassagem percentual;

c_{max}: valor no pico máximo;

c_{final}: valor final em regime permanente;

Tabela 8 – Comparativo do desempenho das sintonias de controle.

Método	$\%UP$	T_s	T_r
Ziegler-Nichols	4,859	0,295	0,16
Cohen e Coon	0,609	0,225	0,16
IMC	0	1	0,475

Fonte: Autoria própria (2025).

Os arquivos dessa simulação estão presentes no Apêndice B ao final deste trabalho.

5 RESULTADOS E ANÁLISES

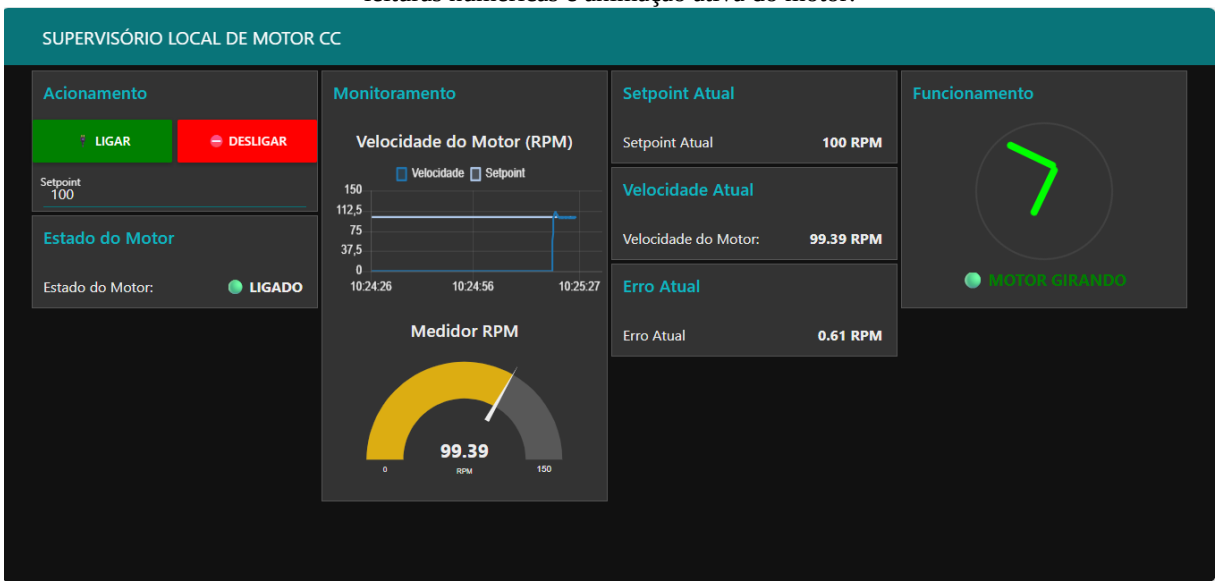
Neste capítulo, são apresentados e discutidos os resultados obtidos a partir do desenvolvimento e aplicação do sistema proposto. Inicialmente, analisa-se o funcionamento dos supervisórios local e remoto, com ênfase na resposta em tempo real das interfaces, nos elementos visuais e na confiabilidade da comunicação entre os dispositivos. Em seguida, são detalhados os dados registrados durante a aplicação dos diferentes métodos de sintonia do controlador PID, utilizados para o controle da velocidade do motor de corrente contínua. Para avaliar o desempenho de cada método, são considerados os seguintes parâmetros: tempo de subida, tempo de acomodação e sobressinal. Também são apresentados os resultados referentes ao banco de dados em nuvem, que demonstrou capacidade de atualizar continuamente as variáveis do sistema e transmitir essas informações imediatamente aos supervisórios. Por fim, realiza-se uma comparação entre os métodos aplicados, a fim de verificar qual abordagem oferece a melhor resposta dinâmica e maior eficiência no controle do sistema.

O Apêndice D, localizado ao final deste trabalho, contém um vídeo demonstrativo que ilustra o funcionamento completo do sistema. Nele, são evidenciadas as operações dos supervisórios no controle e monitoramento em tempo real, a atualização contínua do banco de dados e o funcionamento do motor no circuito físico.

5.1 FUNCIONAMENTO DO SUPERVISÓRIO LOCAL: MONITORAMENTO EM TEMPO REAL E VISUALIZAÇÃO DINÂMICA

A Figura 53 apresenta detalhadamente o painel local com gráfico de velocidade em tempo real comparando com o valor de referência, indicador de rotação por minuto, leituras numéricas de valor de referência, velocidade e erro, botões "LIGAR" e "DESLIGAR" e o indicador "MOTOR GIRANDO" iluminado. A atualização imediata desses componentes confirma a comunicação eficiente entre a ESP32 e o ambiente Node-RED, por meio do protocolo MQTT, conhecido por sua baixa latência e confiabilidade em aplicações de internet das coisas (Yu, 2024).

Figura 53 – Supervisório local em operação com gráfico de velocidade, indicador de RPM, botões de controle, leituras numéricas e animação ativa do motor.



Fonte: Autoria própria (2025).

Nota-se na Figura 53 a animação no grupo "Funcionamento" do Node-RED com iluminação verde, destacando o texto "MOTOR GIRANDO". A presença dessa animação evidencia que o estado físico do motor está sendo refletido com fidelidade na interface. Sistemas que utilizam MQTT e elementos visuais animados permitem ao operador acompanhar o desempenho do processo em tempo real, proporcionando uma supervisão contínua e eficiente das operações.

5.2 FUNCIONAMENTO DO SUPERVISÓRIO REMOTO: VISUALIZAÇÃO POR NAVEGADOR COM ELEMENTOS ATIVOS

Na Figura 54, observa-se o painel remoto com o gráfico dinâmico de velocidade, mudança de cores nos botões de controle, imagem animada do motor e leituras numéricas de valor de referência, velocidade e erro. Apesar da latência comum da comunicação em nuvem, os dados são exibidos de forma consistente, demonstrando a eficácia do banco de dados em tempo real da plataforma *Firebase*, cuja arquitetura permite sincronização quase imediata entre os dispositivos conectados (Firebase, 2025).

Figura 54 – Supervisório remoto com gráfico dinâmico, botões de controle, animação do motor e leituras numéricas em tempo real.

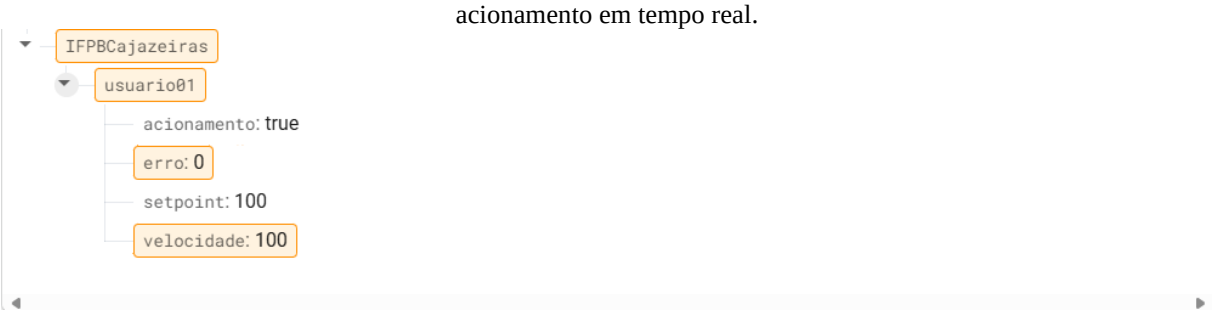
Fonte: Autoria própria (2025).

Observa-se o ícone do motor em tonalidade verde, indicando o acionamento do sistema. Esse resultado demonstra que o estado físico do equipamento foi corretamente transmitido e representado no ambiente remoto, validando a coerência entre o hardware e a interface visual, conforme documentado em estudos sobre o uso do *Firebase* na supervisão de dispositivos conectados (Ably, 2022).

5.3 BANCO DE DADOS EM TEMPO REAL: SINCRONIZAÇÃO SIMULTÂNEA DE DADOS

A Figura 55 apresenta o console da plataforma *Firebase* com os campos de velocidade, erro, valor de referência e acionamento sendo atualizados em tempo real durante a operação. Esse comportamento confirma a eficiência da arquitetura reativa do *Realtime Database*, que utiliza conexões persistentes para manter os dados sincronizados automaticamente, sem necessidade de recarregamento manual (Appmaster, 2023).

Figura 55 – Console do *Firebase* exibindo os dados atualizados de velocidade, erro, valor de referência e acionamento em tempo real.



Fonte: Autoria própria (2025).

Essa arquitetura possibilita que qualquer ação executada nos painéis local ou remoto seja automaticamente refletida em todos os pontos conectados (dispositivo, banco de dados e interfaces), promovendo uma integração eficaz e sem atrasos perceptíveis.

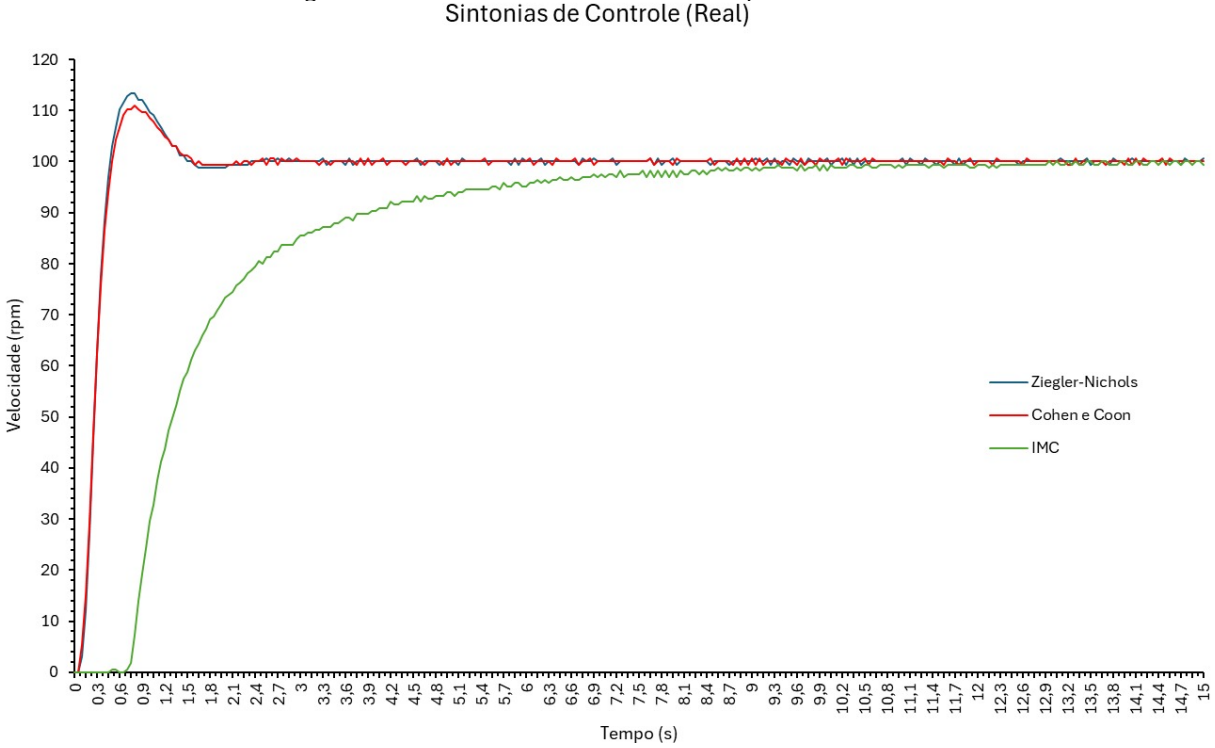
5.4 RESULTADOS PRÁTICOS OBTIDOS COM OS MÉTODOS DE SINTONIA

Com o objetivo de registrar e analisar os resultados experimentais das diferentes sintonias de controle, optou-se por utilizar novamente a ferramenta Excel *Data Streamer*, devido à sua praticidade na aquisição e visualização dos dados em tempo real. Essa abordagem permitiu centralizar as curvas de resposta de todas as sintonias em um único gráfico, facilitando uma comparação direta de seus desempenhos, conforme ilustrado na Figura 56.

De forma análoga, as curvas obtidas a partir das simulações também foram reunidas em um único gráfico (Figura 57), possibilitando uma análise comparativa entre os resultados experimentais e os simulados, para um valor de referência de 100 rpm. Essa estratégia foi fundamental para avaliar a aderência dos modelos utilizados e a efetividade de cada método de sintonia aplicado no controle da velocidade do motor CC. O acesso a essa simulação está disponível no Apêndice B.

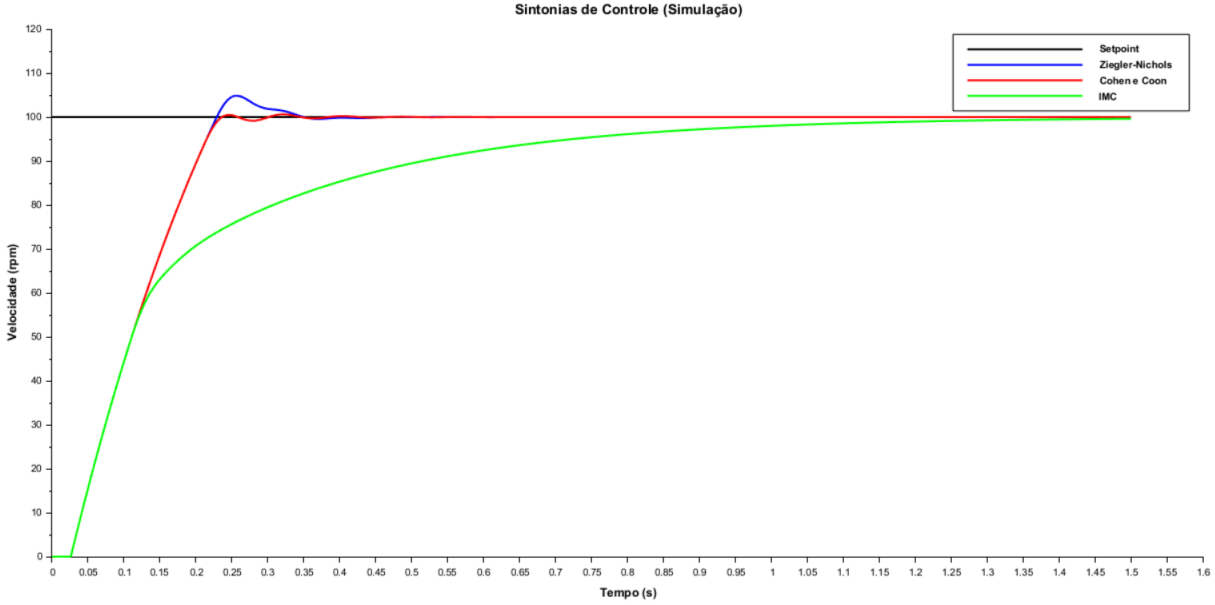
Complementarmente, a Tabela 9 apresenta uma comparação dos principais parâmetros de desempenho, permitindo uma análise quantitativa entre os resultados obtidos nas simulações e aqueles observados experimentalmente. Cabe ressaltar, contudo, que os valores referentes ao desempenho prático foram estimados com base na interpretação visual dos gráficos, o que pode introduzir pequenas imprecisões na análise.

Figura 56 – Curvas das sintonias obtidas experimentalmente.



Fonte: Autoria própria (2025).

Figura 57 – Curvas das sintonias obtidas por simulação.



Fonte: Autoria própria (2025).

Tabela 9 – Comparativo de desempenho simulação e prática

Método	**Simulação**			**Prática**		
	$\%UP$	T_s	T_r	$\%UP$	T_s	T_r
Ziegler-Nichols	4,859	0,295	0,16	12	1,387	0,3
Cohen e Coon	0,609	0,225	0,16	10,75	1,387	0,3

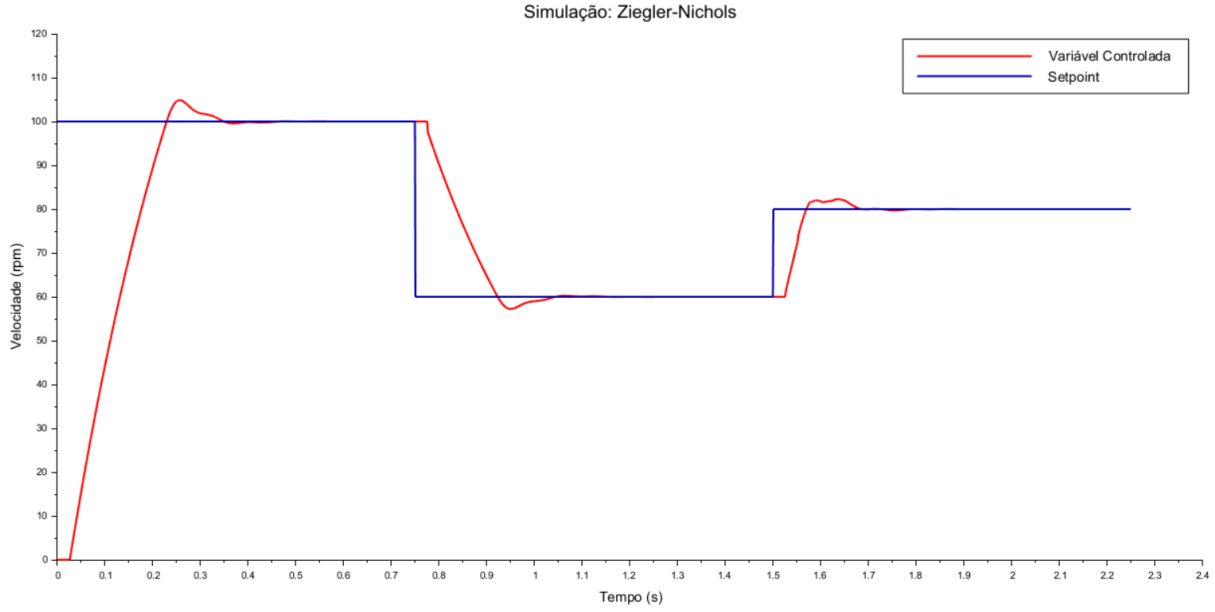
Método	Simulação			Prática		
	$\%UP$	T_s	T_r	$\%UP$	T_s	T_r
IMC	0	1	0,475	0	7,275	2,925

Fonte: Autoria própria (2025).

Adicionalmente, foi realizado um ensaio no qual o *setpoint* foi modificado de forma sequencial para os valores de 100 rpm, 60 rpm e 80 rpm, contemplando os três métodos de sintonia. Esse procedimento foi conduzido inicialmente em simulação, permitindo registrar a curva de resposta do sistema durante cada alteração de referência, bem como o sinal de controle em volts na saída do bloco atuador. Utilizando o mesmo diagrama de blocos da Figura 49, com alteração apenas no degrau aplicado, foi implementado o bloco *Signal Builder* para gerar diferentes *setpoints*. Em seguida, o mesmo teste foi repetido em aplicação prática, na qual, por meio da ferramenta Excel Data Streamer, foram extraídos tanto a curva de resposta do sistema quanto o sinal de controle correspondente à tensão média aplicada ao motor pela ponte H (proveniente do PWM). Os resultados foram organizados em gráficos específicos, permitindo identificar com clareza as transições de regime e o comportamento dinâmico do motor em cada variação de *setpoint*.

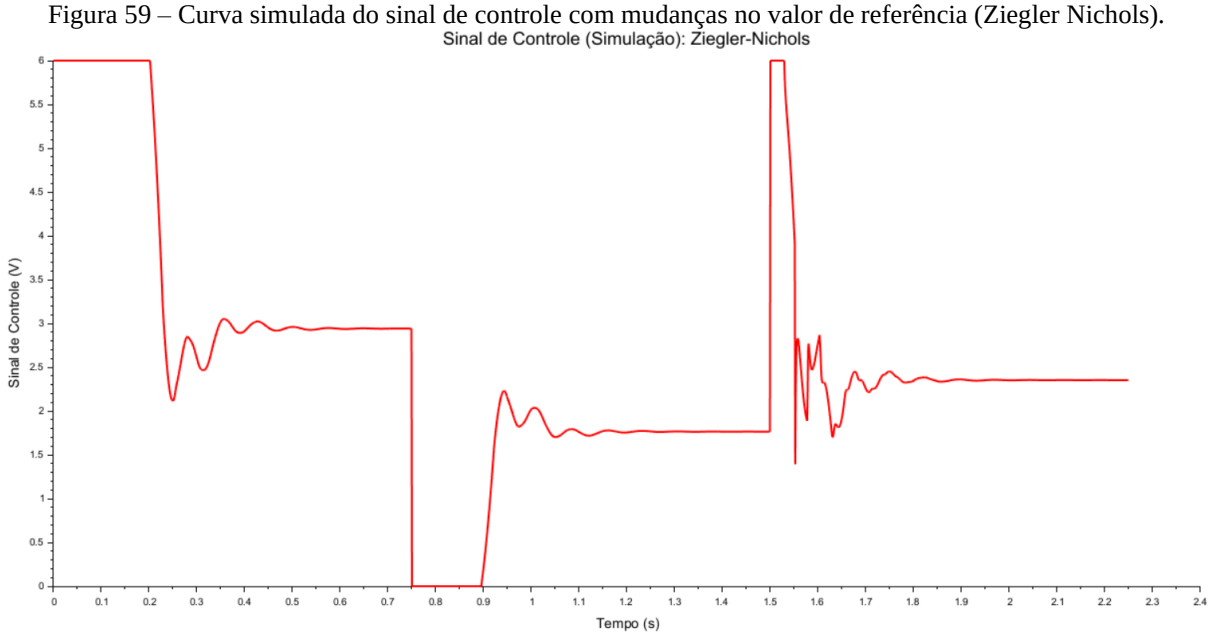
A Figura 58 apresenta a curva de resposta obtida em simulação no Scilab, utilizando o método de Ziegler-Nichols e considerando variações sequenciais de *setpoint*. A Figura 59 exibe o sinal de controle gerado na simulação. Já a Figura 60 e Figura 61 correspondem ao mesmo ensaio em aplicação prática, ilustrando, respectivamente, a resposta do sistema às alterações de referência e o sinal de controle associado.

Figura 58 – Curva de resposta simulada com mudanças no valor de referência (Ziegler-Nichols).



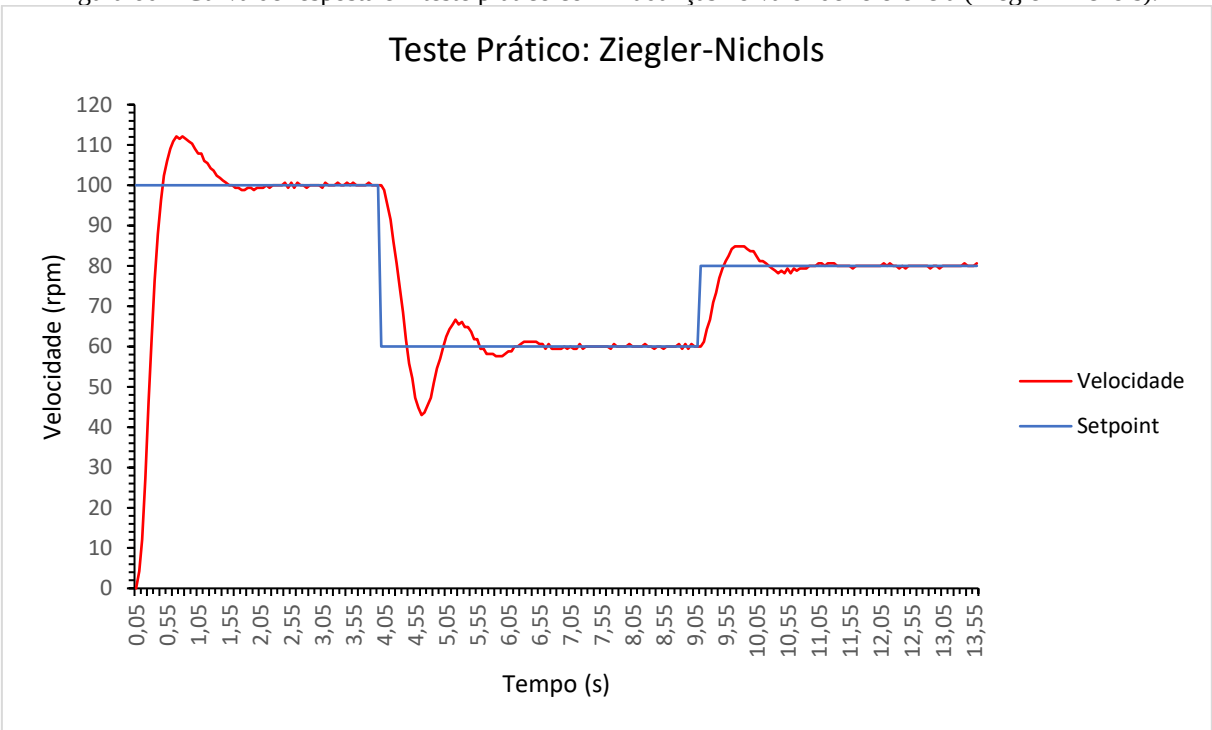
Fonte: Autoria própria (2025).

Figura 59 – Curva simulada do sinal de controle com mudanças no valor de referência (Ziegler Nichols).



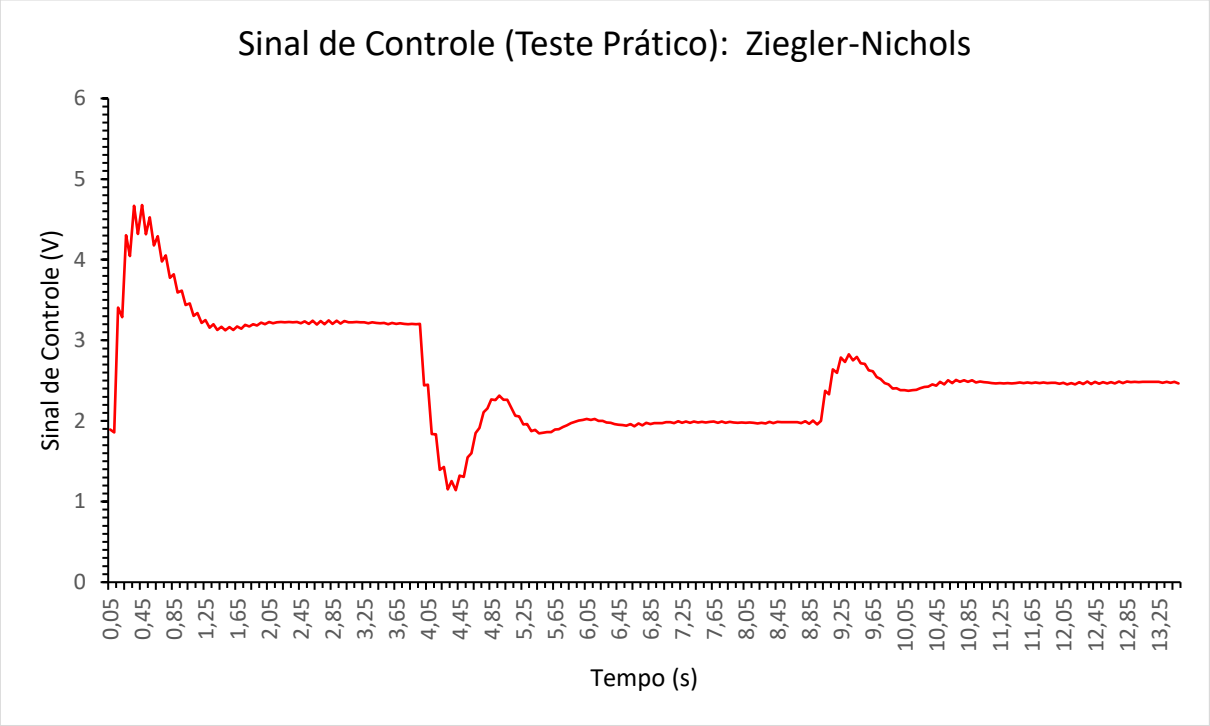
Fonte: Autoria própria (2025).

Figura 60 – Curva de resposta em teste prático com mudanças no valor de referência (Ziegler-Nichols).



Fonte: Autoria própria (2025).

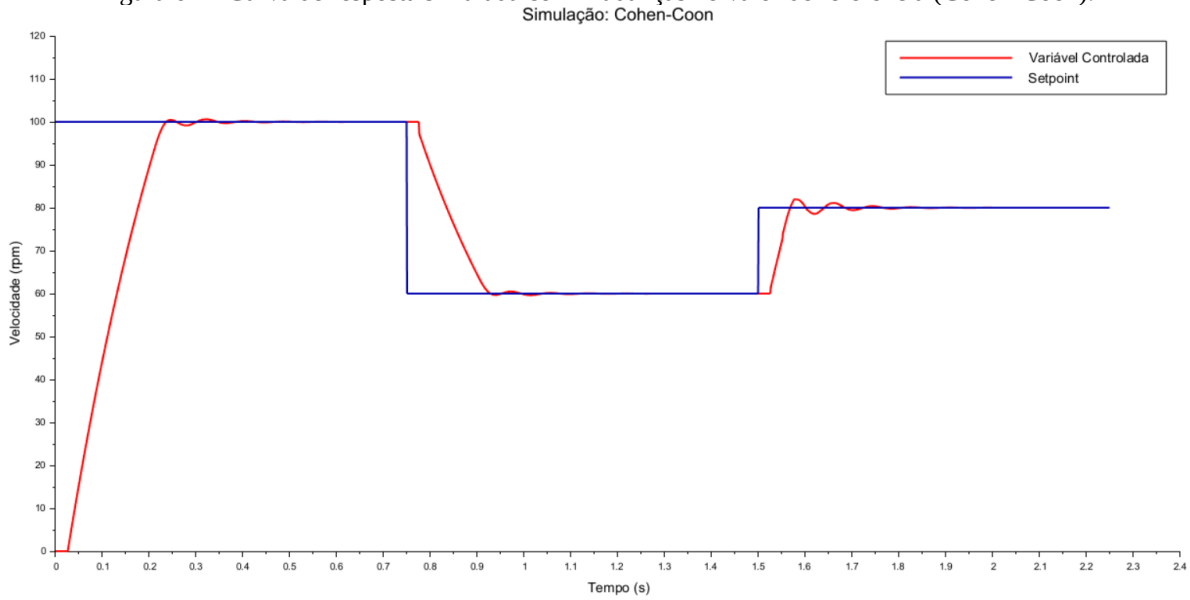
Figura 61 – Curva do sinal de controle em teste prático com mudanças no valor de referência (Ziegler Nichols).



Fonte: Autoria própria (2025).

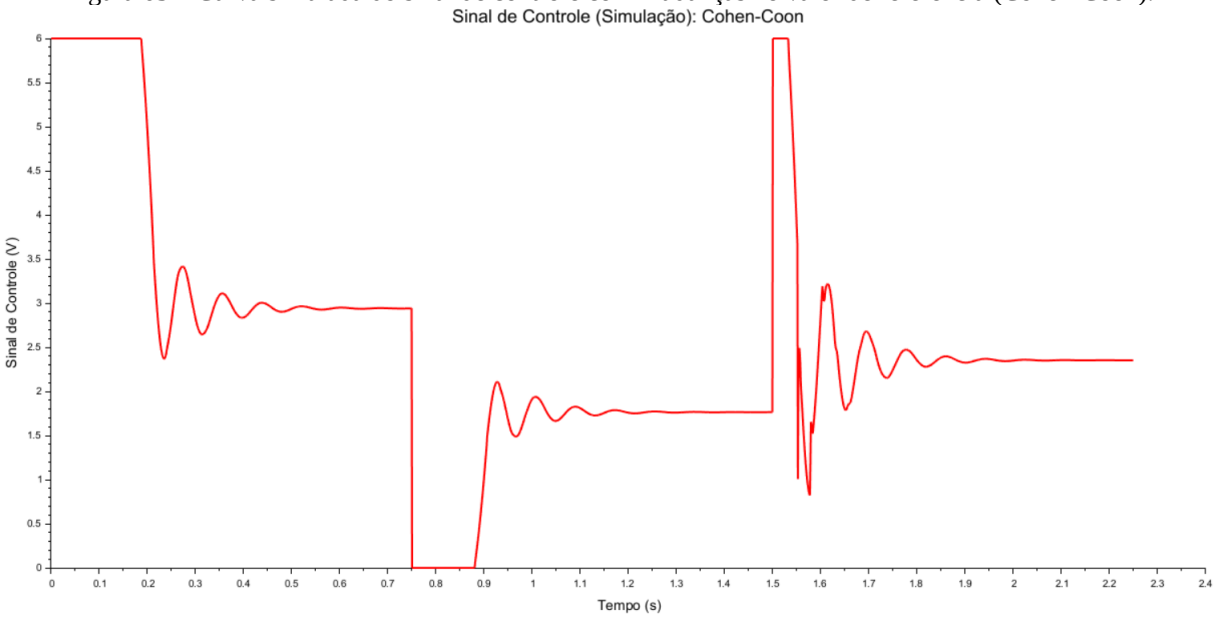
Na aplicação do método de Cohen-Coon, a Figura 62 apresenta a curva de resposta obtida em simulação, considerando as mesmas variações no valor de referência, enquanto a Figura 63 mostra o sinal de controle correspondente. Os resultados práticos do mesmo ensaio são exibidos na Figura 64 e Figura 65, relativos à curva de resposta e o sinal de controle, respectivamente.

Figura 62 – Curva de resposta simulada com mudanças no valor de referência (Cohen-Coon).



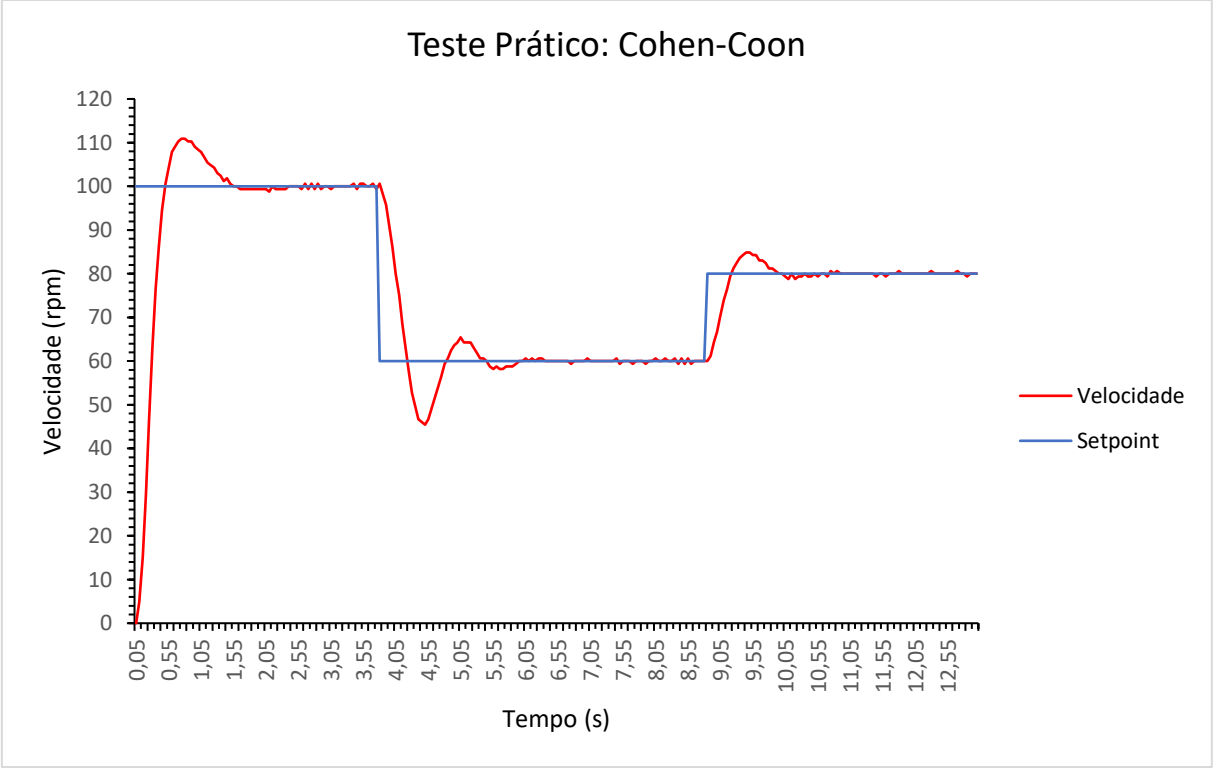
Fonte: Autoria própria (2025).

Figura 63 – Curva simulada do sinal de controle com mudanças no valor de referência (Cohen-Coon).



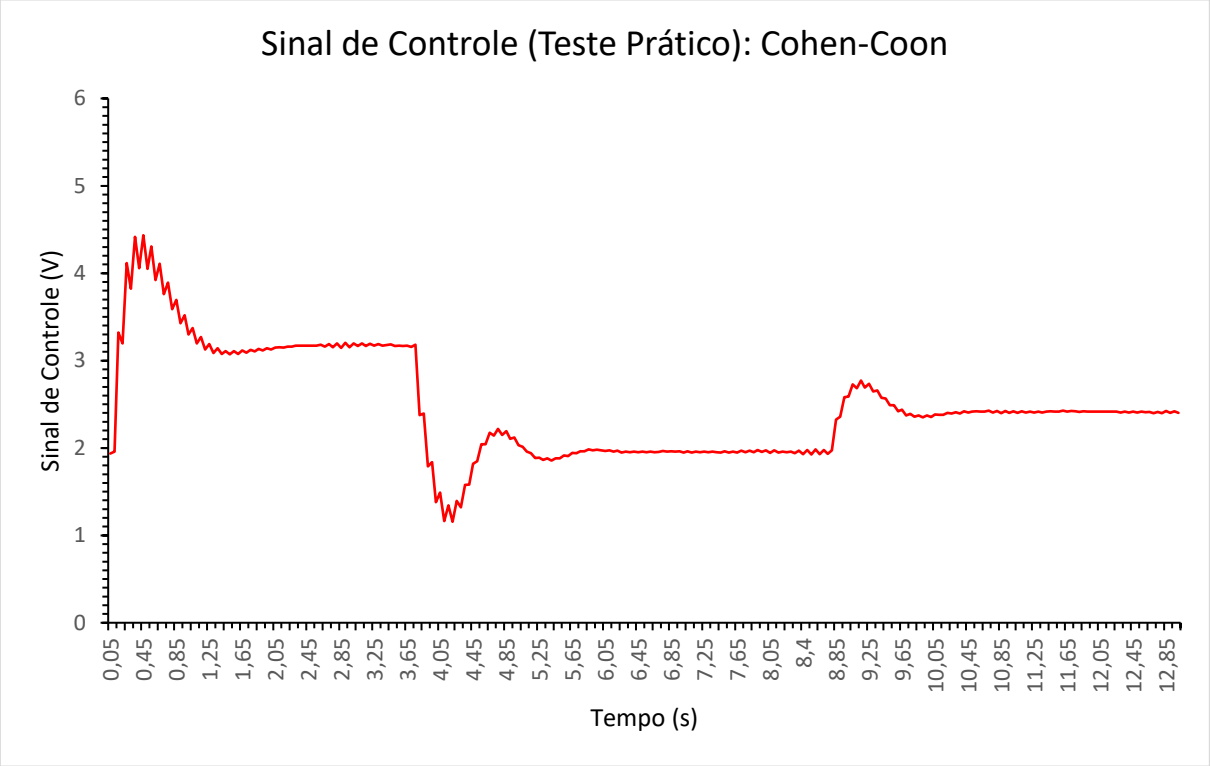
Fonte: Autoria própria (2025).

Figura 64 – Curva de resposta em teste prático com mudanças no valor de referência (Cohen-Coon).



Fonte: Autoria própria (2025).

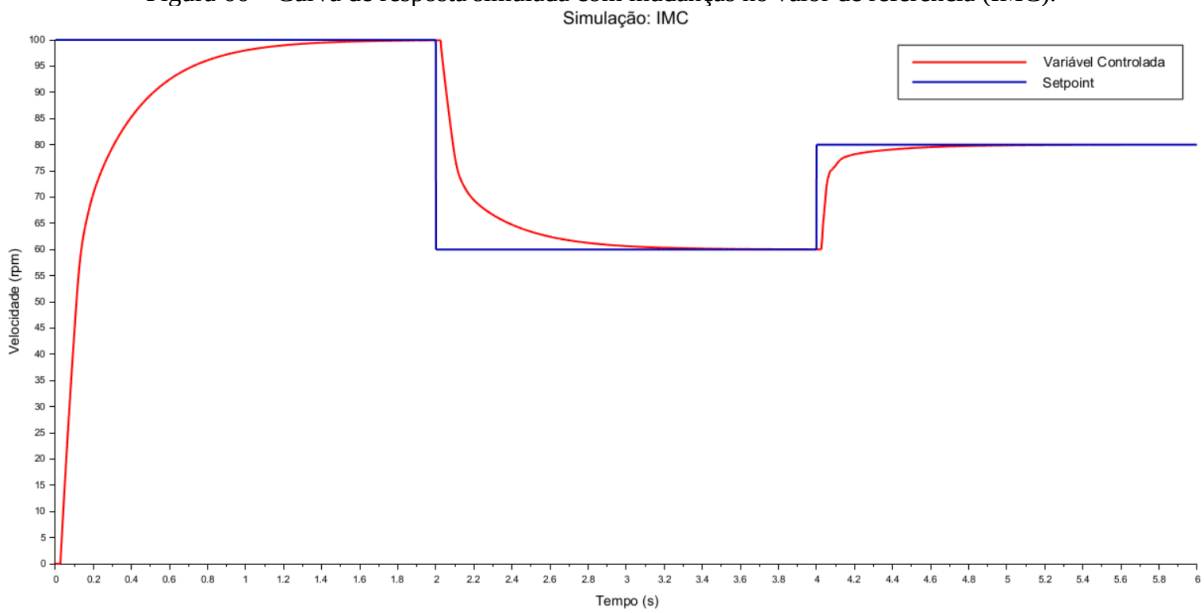
Figura 65 – Curva do sinal de controle em teste prático com mudanças no valor de referência (Cohen-Coon).



Fonte: Autoria própria (2025).

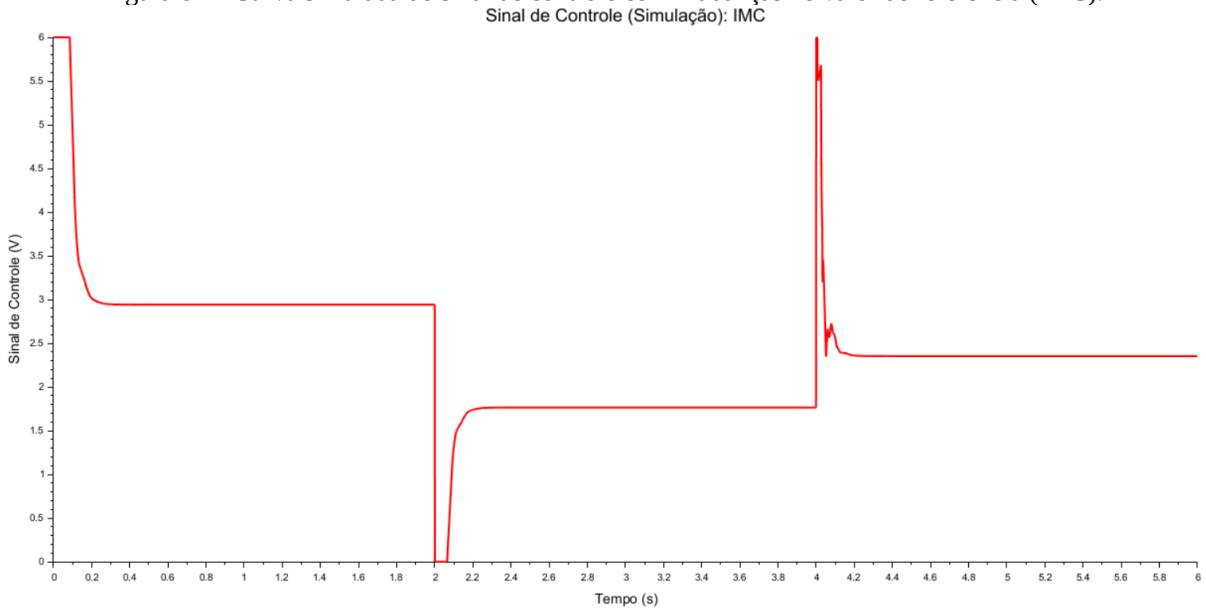
Para o método IMC, a Figura 66 mostra a curva de resposta em simulação e a Figura 67 o sinal de controle, enquanto a Figura 68 e a Figura 69 apresentam os resultados práticos correspondentes.

Figura 66 – Curva de resposta simulada com mudanças no valor de referência (IMC).



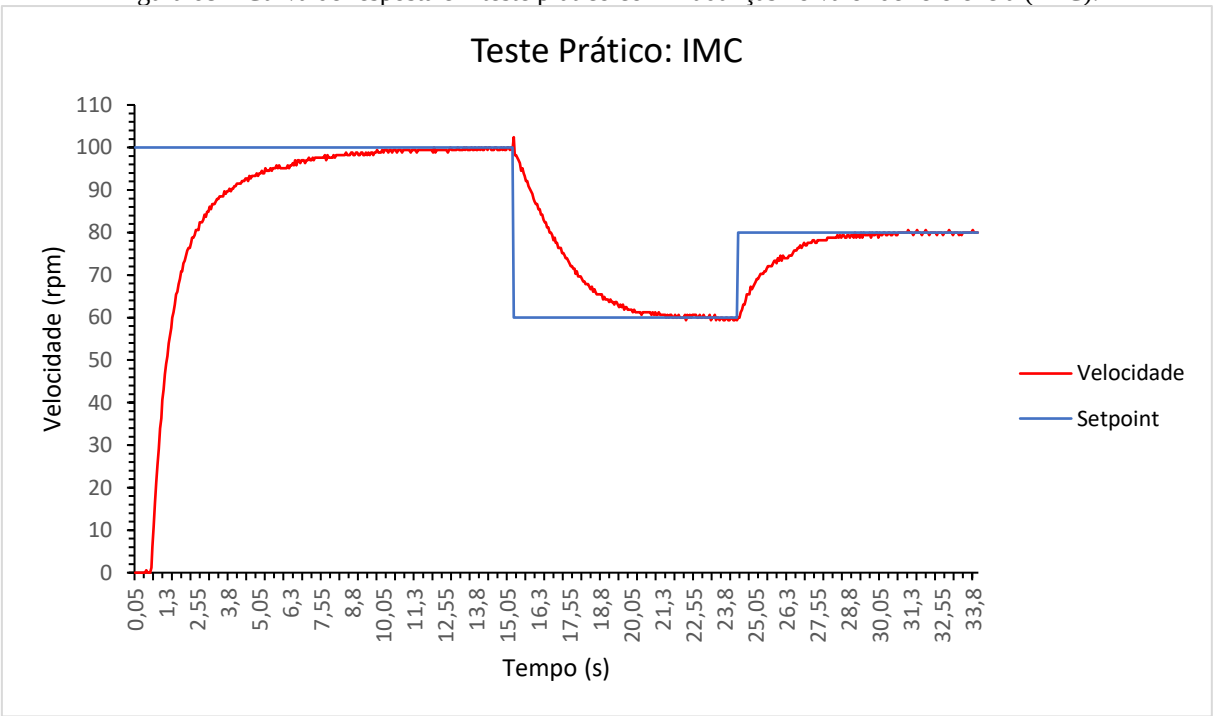
Fonte: Autoria própria (2025).

Figura 67 – Curva simulada do sinal de controle com mudanças no valor de referência (IMC).



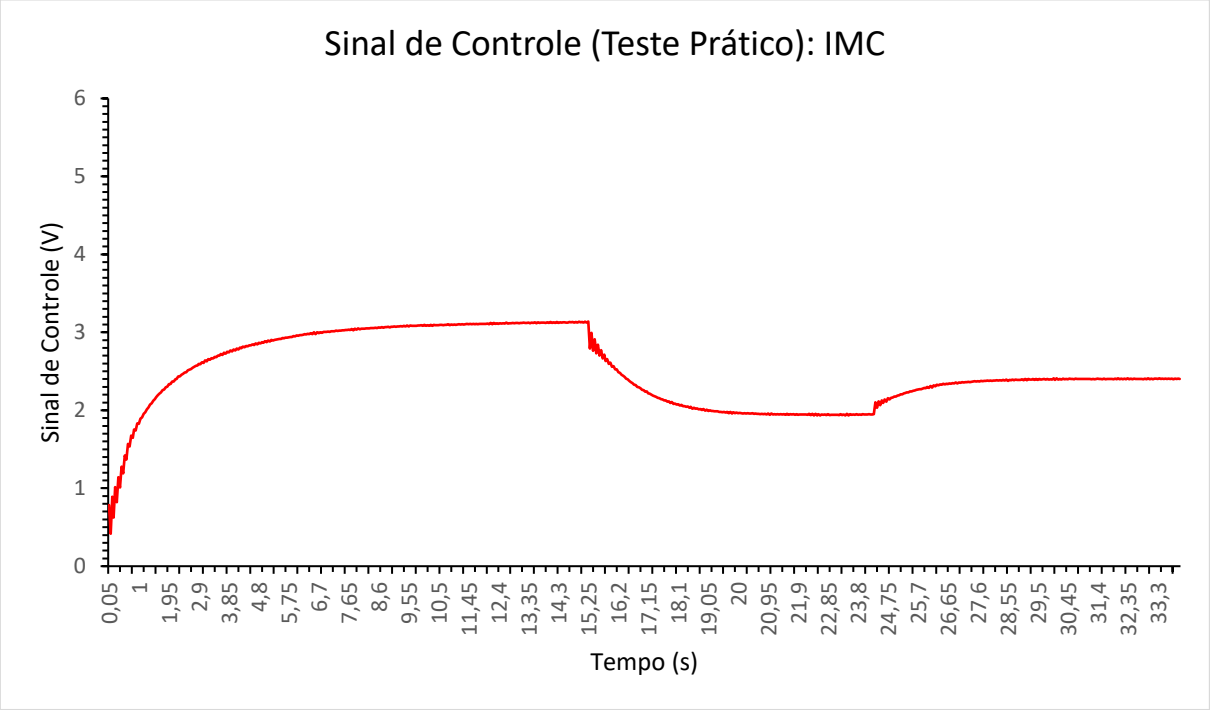
Fonte: Autoria própria (2025).

Figura 68 – Curva de resposta em teste prático com mudanças no valor de referência (IMC).



Fonte: Autoria própria (2025).

Figura 69 – Curva do sinal de controle em teste prático com mudanças no valor de referência (IMC).



Fonte: Autoria própria (2025).

5.4.1 Análise integrada do desempenho das técnicas de sintonia

A análise comparativa entre os métodos de Ziegler-Nichols, Cohen-Coon e IMC permite destacar o comprometimento entre desempenho dinâmico e robustez que cada técnica impõe ao sistema. Em termos gerais, os métodos de Ziegler-Nichols e Cohen-Coon revelaram-se eficazes na obtenção de respostas rápidas, sobretudo no ambiente simulado. No entanto, na prática, essas abordagens evidenciaram limitações consideráveis, como o aumento do sobressinal e maior sensibilidade a ruídos e incertezas da planta.

Tanto o método de Ziegler-Nichols quanto o de Cohen-Coon apresentaram tempos de subida e de acomodação semelhantes, tanto nas simulações quanto nos testes práticos. No entanto, os valores obtidos experimentalmente foram significativamente superiores aos simulados, evidenciando os efeitos de fatores como ruídos, atrasos e não linearidades presentes na planta real. Parte dessa divergência também pode ser atribuída à queda de tensão na ponte H, que compromete a tensão efetiva aplicada ao motor, exigindo maior esforço dos controladores para atingir o *setpoint* e, consequentemente, retardando a resposta do sistema.

Ambos os métodos são classificados como estratégias de sintonia agressiva, pois priorizam rapidez na resposta ao custo de maior sensibilidade a distúrbios externos e aumento do sobressinal. Esse comportamento ficou evidente nos testes práticos: o método de Ziegler-Nichols apresentou o maior sobressinal entre os métodos avaliados, resultando em uma resposta

mais oscilatória e com menor amortecimento. O método de Cohen-Coon, embora também agressivo, produziu um sobressinal ligeiramente menor, indicando uma resposta marginalmente mais estável, ainda que igualmente exposta às limitações da planta real.

De acordo com Campos e Teixeira (2006), os métodos de Ziegler-Nichols e Cohen-Coon tendem a gerar sintonias agressivas. Por esse motivo, recomenda-se que, na prática, sejam realizados ajustes manuais, reduzindo inicialmente os ganhos com o objetivo de suavizar a ação de controle. Tais ajustes geralmente envolvem a redução do ganho proporcional, o aumento do tempo integral e a diminuição do tempo derivativo. A partir dessas correções, os parâmetros devem ser ajustados gradualmente, com base na observação do comportamento dinâmico do sistema, até que se atinja um desempenho satisfatório em termos de estabilidade, rapidez e robustez.

Em contraste com os outros métodos, o método IMC apresentou desempenho significativamente superior no que diz respeito à estabilidade da resposta e à ausência de sobressinal, tanto nas simulações quanto nos testes práticos. Essa característica está diretamente relacionada à filosofia conservadora do método, que busca garantir robustez mesmo diante de incertezas e variações na planta. O tempo de acomodação mais elevado observado nos testes é uma consequência direta da escolha de um valor maior para o parâmetro λ, o qual atua reduzindo a agressividade do controlador e favorecendo um comportamento mais estável e amortecido.

A discrepância entre os tempos de resposta simulados e experimentais também foi observada no IMC, mas em maior proporção. Essa diferença pode ser atribuída, principalmente, à queda de tensão na ponte H, que limita a tensão efetiva aplicada ao motor, além dos efeitos de ruídos, atrasos e não linearidades da planta física, fatores que não são completamente capturados no modelo matemático utilizado nas simulações.

Como discutido na Seção 3.9.3, Campos e Teixeira (2006) enfatizam que o IMC adota uma abordagem mais prudente, priorizando a robustez e a estabilidade do sistema em malha fechada. Essa abordagem se mostra especialmente eficaz em processos com comportamento dinâmico complexo, presença de incertezas estruturais ou dificuldade de modelagem precisa. Nesses casos, a elevação do valor de λ é recomendada como forma de ampliar a margem de robustez do controlador, tornando o sistema mais tolerante a variações e a perturbações externas, ainda que à custa de uma resposta mais lenta. Essa troca entre velocidade e robustez é uma característica fundamental do IMC e reforça sua aplicabilidade em contextos onde a previsibilidade e a segurança operacional são prioritárias.

Um ponto crucial para justificar as divergências observadas em todos os resultados do trabalho reside na diferença fundamental entre os modelos de controle. Na simulação, o sistema opera em um cenário ideal com um controlador contínuo, representado no domínio de Laplace. Em contrapartida, a implementação prática utiliza um controlador digital, que requer a discretização dos parâmetros do PID pelo método de Tustin. Essa transição de um modelo ideal para um sistema digital com tempo de amostragem finito e processamento em etapas, somada às restrições físicas, é uma das principais causas das diferenças observadas nas respostas. Enquanto o modelo contínuo representa um comportamento teórico e sem limitações, o sistema prático reflete as não linearidades e atrasos inerentes à conversão e processamento de sinais, justificando por que os resultados simulados e práticos são distintos em todas as sintonias.

A análise da resposta do sistema com alterações do *setpoint* em tempo real revelou que as respostas simuladas, por operarem em um ambiente ideal, demonstraram uma capacidade de rastreamento de referência com transições rápidas. Em contraste, as respostas obtidas na prática foram mais lentas e graduais. Essa diferença é justificada pela análise do sinal de controle: na simulação, o controlador atinge o limite de 6V de forma instantânea para corrigir o erro. Na prática, a tensão útil entregue ao motor é reduzida por fatores como a queda de tensão da fonte sob carga e a perda de aproximadamente 1,5 a 2V na Ponte H, limitando o sinal de controle e resultando em uma resposta menos reativa do sistema.

Portanto, observa-se que nenhum método é universalmente superior. A escolha da estratégia de sintonia deve ser orientada pelos critérios específicos do projeto: se o requisito for resposta rápida com tolerância a variações de desempenho, Ziegler-Nichols ou Cohen-Coon podem ser adotados, desde que associados a compensações adequadas. Para sistemas em que a estabilidade e a previsibilidade operativa são indispensáveis, especialmente em aplicações industriais contínuas, a abordagem IMC revela-se mais apropriada.

6 CONCLUSÃO

Os resultados obtidos ao longo do desenvolvimento deste trabalho demonstraram, de forma clara, a eficiência da solução proposta. O sistema de controle de velocidade para motor de corrente contínua, baseado na plataforma ESP32, apresentou desempenho satisfatório tanto em simulação quanto na aplicação prática, evidenciando a eficácia da implementação do controlador PID digital.

As técnicas de sintonia aplicadas, Ziegler-Nichols, Cohen-Coon e Controle por Modelo Interno (IMC), permitiram avaliar o comportamento dinâmico do sistema sob diferentes parâmetros, sendo possível verificar, nos testes experimentais, que os métodos propostos atendem de maneira adequada aos critérios de desempenho esperados, como tempo de subida, tempo de acomodação e sobressinal. A aproximação da planta por modelo de primeira ordem com atraso, obtida por meio da curva de reação, mostrou-se suficiente para a aplicação dos métodos de sintonia, mesmo considerando as simplificações envolvidas.

A integração do controle embarcado com os supervisórios local e remoto consolidou uma arquitetura moderna, funcional e acessível. O supervisório local, desenvolvido no Node-RED, viabilizou a operação e o monitoramento em tempo real com confiabilidade. O supervisório remoto, implementado por meio da plataforma *Firebase Studio* e hospedado no Vercel, demonstrou-se eficiente na atualização e leitura dos dados do banco em nuvem, permitindo o acompanhamento remoto do sistema com resposta ágil e interface amigável.

A utilização do protocolo MQTT como meio de comunicação entre os elementos do sistema também se mostrou uma escolha acertada, garantindo leveza, baixo consumo de recursos e elevada confiabilidade na troca de dados.

Dessa forma, conclui-se que os objetivos gerais e específicos propostos foram integralmente alcançados, confirmando a viabilidade do sistema e sua aderência às demandas atuais da automação integrada com tecnologias de Internet das Coisas. O projeto demonstrou que é possível aliar baixo custo, conectividade e controle eficiente em um mesmo sistema, o que o torna uma base sólida para futuras aplicações acadêmicas e industriais.

6.1 SUGESTÕES PARA TRABALHOS FUTUROS

Com base na experiência adquirida neste projeto, foi possível identificar algumas melhorias e ideias que podem ser exploradas em trabalhos futuros, com o objetivo de tornar o sistema mais completo, seguro e próximo de aplicações reais:

a) Implementar autenticação com *Firebase Authentication*

Atualmente, o acesso ao supervisório remoto é feito por meio de um usuário fixo criado pela própria ferramenta *Firebase Studio*. Uma sugestão é usar o *Firebase Authentication* para permitir o login com e-mail e senha de forma segura, com gerenciamento de diferentes usuários e controle de permissões. Isso deixaria o sistema mais seguro e adequado para uso por várias pessoas.

b) Controlar mais de um motor ao mesmo tempo

O sistema pode ser expandido para controlar vários motores simultaneamente. Isso exigiria ajustes na programação, no supervisório e no banco de dados, mas permitiria aplicar o projeto em cenários mais próximos do ambiente industrial, onde normalmente se controla mais de um equipamento.

c) Testar outros tipos de controle além do PID

Outras técnicas de controle, como controle adaptativo e controle preditivo (MPC), podem ser aplicadas ao mesmo sistema. Assim, seria possível comparar os resultados com o PID e entender melhor qual método funciona melhor em diferentes situações, como mudanças de carga ou variações no sistema.

d) Criar um aplicativo para celular

Uma ideia interessante é desenvolver um aplicativo para Android ou iOS que permita monitorar e controlar o sistema pelo celular. Isso daria mais praticidade ao usuário, permitindo acesso remoto de forma simples e rápida, diretamente do telefone.

e) Usar métodos mais precisos para identificar o modelo matemático da planta (motor)

Neste trabalho, foi utilizado um método clássico e prático para identificar o comportamento do motor, baseado na resposta ao degrau. Como sugestão para trabalhos futuros, podem ser exploradas técnicas alternativas de identificação, como o método dos mínimos quadrados, que permite estimar os parâmetros de um modelo matemático diretamente a partir dos dados experimentais de entrada e saída. Essa abordagem é amplamente conhecida, de fácil implementação e pode resultar em modelos mais precisos, contribuindo para um controle mais eficiente do sistema.

f) Integrar uma câmera ao sistema para visualização do motor

Uma sugestão muito importante é a instalação de uma câmera próxima ao motor, com transmissão de vídeo integrada diretamente ao supervisório remoto. Isso permitiria ao operador acompanhar visualmente o funcionamento do motor à distância, oferecendo mais segurança e

confiabilidade durante a operação. Essa função seria restrita ao supervisório remoto, já que ele é quem tem maior necessidade de monitoramento visual em tempo real, principalmente em situações de supervisão remota onde não há ninguém fisicamente no local.

Essas ideias mostram que o projeto pode ser melhorado e adaptado para diferentes necessidades, com foco em segurança, flexibilidade e aplicação prática. Elas também abrem espaço para explorar novas tecnologias e ferramentas dentro da área de controle e automação.

REFERÊNCIAS

ABLY. **Firebase vs WebSocket: differences and how they work together.** Ably, 2022. Disponível em: https://ably.com/topic/firebase-vs-websocket. Acesso em: 30 jun. 2025.

APPMASTER. **Is firebase realtime database the right choice for real-time data sync?.** Appmaster, 2023. Disponível em: https://appmaster.io/blog/firebase-realtime-database-for-real-time-data-sync. Acesso em: 02 jul. 2025.

AGUILAR, D. D. R. de; GUIMARÃES, M. A.; MANIÇOBA, R. H. C.; SANTOS, A. F. dos. **Construção de uma estação meteorológica modular com o uso da plataforma arduino e protocolo de comunicação MQTT.** Brazilian journal of development, Curitiba – PR, v. 6, p. 36119-36127, abr., 2021.

ALPES AUTOMAÇÃO. **O que é controle PID?.** Alpes Automação, 2025. Disponível em: https://alpesautomacao.com.br/controladores/o-que-e-controle-pid/. Acesso em: 10 abr. 2025.

ASTRÖM, K.; HÄGGLUND, T. **PID Controllers:** Theory, design and tuning. 2 ed. North Carolina: Instrument Society of America, 1995.

ALMEIDA, F. **O que é encoder?.** Hitecnologia, 2017. Disponível em: https://materiais.hitecnologia.com.br/blog/o-que-%C3%A9-encoder-para-que-serve-como-escolher-como-interfacear/. Acesso em: 23 maio 2025.

CAMPOS, M. C. M. M. de; TEIXEIRA, H. C. G. **Controles típicos de equipamentos e processos industriais.** 1 ed. São Paulo: Editora Edgard Blucher LTDA, 2008.

DIAS, G. O. **Monitoramento reativo de protocolos em sistemas IoT sobre MQTT.** 2017. 90 p. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Universidade Estadual do Norte do Paraná, Bandeirantes – PR, 2017.

FERMINO, F. **Estudo comparativo de métodos de sintonia de controladores PID.** 2014. 90 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) – Escola de Engenharia de São Carlos, São Carlos – SP, 2014.

FIREBASE. **Firebase realtime database.** Firebase, 2025. Disponível em: https://firebase.google.com/docs/database?hl=pt-br. Acesso em: 09 maio 2025.

FIREBASE. **Firebase studio.** Firebase, 2025. Disponível em: https://firebase.google.com/docs/studio?hl=pt-br. Acesso em: 13 maio 2025.

FREITAS, A. S. **Estudo e implementação do protocolo de comunicação MQTT aplicado a um sistema de automação predial.** 2017. 56 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Campina Grande – PB, 2017.

GEEKSFORGEEKS. **Firebase tutorial.** Geeksforgeeks, 2024. Disponível em: https://www.geeksforgeeks.org/firebase/firebase-tutorial/. Acesso em: 03 maio 2025.

GUSE, R. **Controlando um motor DC com driver ponte H L298N.** Makerhero, 2013. Disponível em: https://www.makerhero.com/blog/motor-dc-arduino-ponte-h-l298n/?srsltid=AfmBOoq5Dl-DVx0BttEoUBvzstjnHPsh4fR08_QdsZZX3aokUU_9PnO. Acesso em: 09 maio 2025.

HIVEMQ. **Introducing the MQTT protocol – MQTT essentials: part 1.** HiveMQ, 2024. Disponível em: https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/#heading-exploring-mqtt-topics-subscriptions-qo-s-persistent-messaging-and-more. Acesso em: 17 abr. 2025.

HIVEMQ. **MQTT packets: a comprehensive guide.** HiveMQ, 2024. Disponível em: https://www.hivemq.com/blog/mqtt-packets-comprehensive-guide/. Acesso em: 30 abr. 2025.

HIVEMQ. **What is MQTT quality of service (QoS) 0,1, & 2? – MQTT essentials: part 6.** HiveMQ, 2025. Disponível em: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/#heading-benefits-of-mqtt-qo-s-for-reliable-io-t-communication. Acesso em: 19 abr. 2025.

LAST MINUTE ENGINEERS. **ESP32 pinout reference.** Last Minute Engineers, [*s.d.*]. Disponível em: https://lastminuteengineers.com/esp32-pinout-reference/. Acesso em: 07 abr. 2025.

MQTT. **MQTT: the standard for IoT messaging.** MQTT, 2024. Disponível em: https://mqtt.org/. Acesso em: 17 abr. 2025.

NISE, N. S. **Engenharia de sistemas de controle.** 6 ed. Rio de Janeiro: Livro Técnicos e Científicos Editora, 2012.

OGATA, K. **Engenharia de controle moderno.** 5 ed. São Paulo: Prentice Hall, 2010.

OLIVEIRA, E. A. de. **Aplicação da transformada z na análise de circuitos elétricos discretos.** 2025. Trabalho de Conclusão de Curso (Bacharelado em Ciência e Tecnologia) – Universidade Federal Rural do Semi-Árido, Caraúbas – RN, 2025.

PASSE, F. F.; VASCONCELOS, V. C. R.; CANESCHE. M.; FERREIRA, R. **Perspectivas do Node-RED no ensino de IoT.** Computer architecture education, Viçosa – MG, v. 6, p. 46-51, dez., 2017.

PINTO, J. E. M. G. **Aplicação prática do método de sintonia de controladores PID utilizando o método do relé com histerese.** 2014. 110 p. Dissertação (Mestrado em Engenharia Elétrica e de Computação) – Universidade Federal do Rio Grande do Norte, Natal – RN, 2014.

RAMOS, E. Y. T. **Projeto de um sistema holter com envio de dados para armazenamento remoto via app mobile.** 2025. 61 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica) – Universidade Tecnológica Federal do Paraná, Campo Mourão – PR, 2025.

SARRAF, G. **Node-RED IoT dashboard tutorial: build interactive real-time dashboards.** Thinkrobotics, 2025. Disponível em: https://thinkrobotics.com/blogs/learn/node-red-iot-dashboard-tutorial-build-interactive-real-time-dashboards. Acesso em: 29 jun. 2025.

SILVA, J. M. G. da. **Wind-up da ação integral.** UFRGS, 2000. Disponível em: https://www.ece.ufrgs.br/~jmgomes/pid/Apostila/apostila/node31.html. Acesso em: 13 Abr. 2025.

SOARES, C. S. H. **Arcabouço de integração do Node-RED com ambientes simulados industriais.** 2024. 43 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Campina Grande – PB, 2024.

SOARES, E. de S. **Desenvolvimento de medidor de energia de baixo custo aplicado a internet das coisas utilizando ESP-32.** 2021. 76 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) – Universidade Federal Rural do Semi-Árido, Caraúbas – RN, 2021.

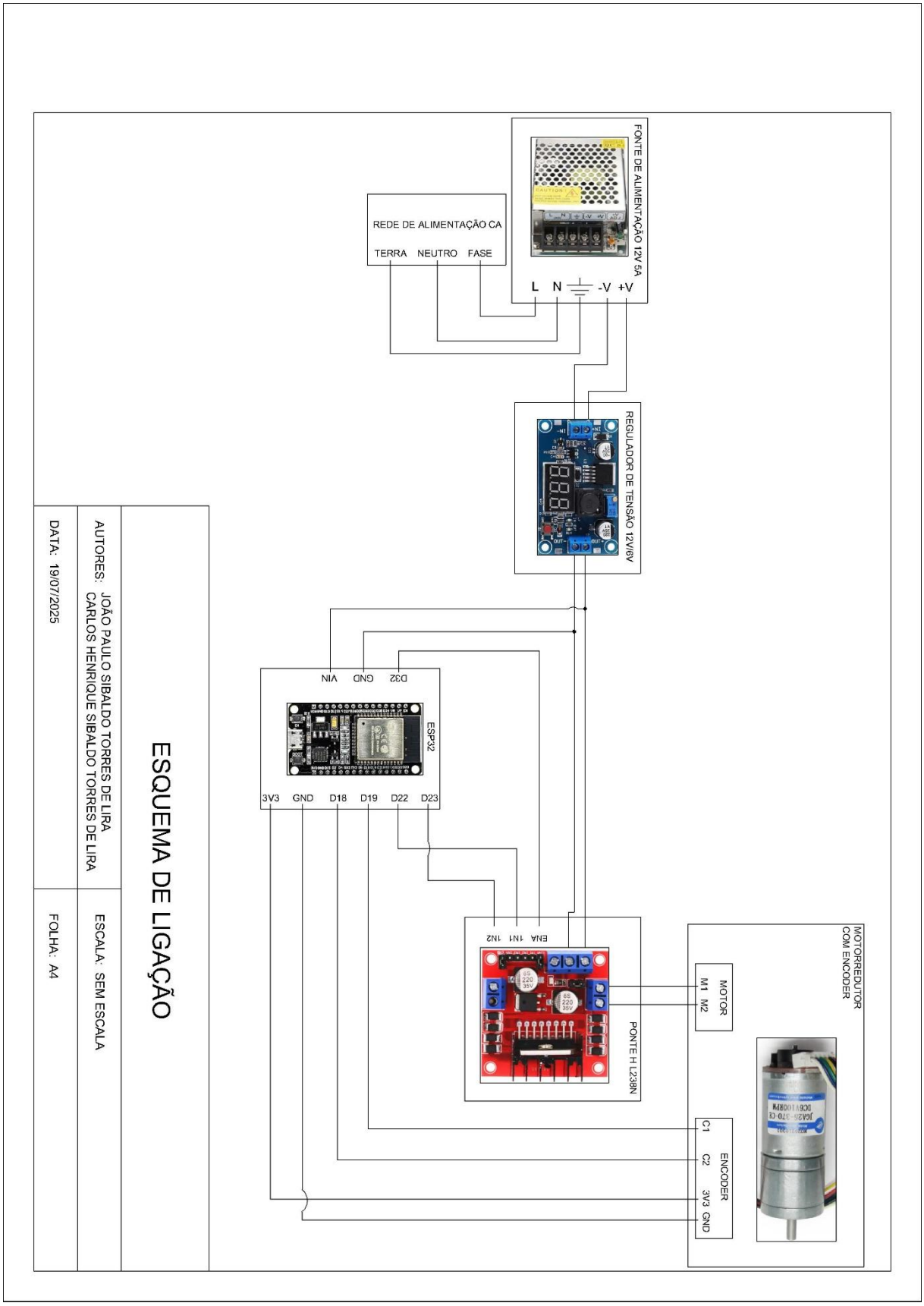
TAVARES, L. G. **Desenvolvimento de uma planta didática para estudo e demonstração do controle PID de velocidade de motores de corrente contínua.** 2017. 50 p. Monografia (Especialização em Automação Industrial) – Universidade Tecnológica Federal do Paraná, Curitiba – PR, 2017.

THINGSBOARD. **How to connect ESP32 Dev Kit V1 to thingsboard.** Thingsboard, [*s.d.*]. Disponível em: https://thingsboard.io/docs/devices-library/esp32-dev-kit-v1/. Acesso em: 11 maio 2025.

XYTMOTORS. **DC geared motor (encoder).** Xytmotors, [*s.d.*]. Disponível em: http://www.xytmotors.com/encoder/153.html. Acesso em: 21 maio 2025.

YU, S. **MQTT with Node-RED: a beginner's guide with examples.** Emqx, 2024. Disponível em: https://www.emqx.com/en/blog/using-node-red-to-process-mqtt-data. Acesso em: 28 jun. 2025.

APÊNDICE A – ESQUEMA DE LIGAÇÃO



APÊNDICE B – CÓDIGO FONTE E SIMULAÇÃO COMPUTACIONAL

Neste apêndice, encontra-se o link para o repositório no GitHub que contém o código-fonte do sistema embarcado desenvolvido e todos os arquivos de simulação do Scilab, juntamente com o QR *code* correspondente. O repositório permite acessar toda a implementação e simulação, facilitando sua reprodução, análise e futuras modificações no sistema.

https://github.com/Carlos-HenS/TCC-Carlos_Joao-BECA

APÊNDICE C – ACESSO AO SUPERVISÓRIO REMOTO

Este apêndice disponibiliza o link para acesso ao supervisório remoto desenvolvido, além do QR *code* correspondente. A plataforma permite o controle e monitoramento do sistema em tempo real por meio de uma interface gráfica interativa.

https://supervisao-remota-motor-cc.vercel.app/

APÊNDICE D – ACESSO AO VÍDEO DO FUNCIONAMENTO DO SISTEMA

Este apêndice disponibiliza o link para acesso ao vídeo demonstrativo do funcionamento do sistema, juntamente com o QR *code* correspondente. O vídeo apresenta os supervisórios local e remoto realizando o controle e monitoramento em tempo real, a atualização do banco de dados e o funcionamento prático do circuito com o motor.

https://drive.google.com/file/d/1E9YAWiiti5nlibqxQHQGJ14-8OKPHF2V/view?usp=sharing