Escola de Minas
CECAU - Colegiado do Curso de
Engenharia de Controle e Automação

Ricardo Caldeira Campos Linhares de Carvalho

Sistema de Controle Remoto de Abertura de Válvula Industrial e Sensoriamento IoT

Monografia de Graduação

Ouro Preto, 2025

Ricardo Caldeira Campos Linhares de Carvalho

Sistema de Controle Remoto de Abertura de Válvula Industrial e Sensoriamento IoT

Trabalho apresentado ao Colegiado do Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheira(o) de Controle e Automação.

Orientador: Prof. Edson Alves Figueira Junior
Coorientador: Prof. Bruno Nazário Coelho

Ouro Preto
2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Ricardo Caldeira Campos Linhares de Carvalho

Sistema de controle remoto de abertura de válvula industrial e sensoriamento IoT

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Engenheiro de Controle e Automação

Aprovada em 08 de agosto de 2025

Membros da banca

[Dr.] Edson Alves Figueira Junior - Orientador (Universidade Federal de Ouro Preto)
[Dr.] - Bruno Nazário Coelho - Coorientador (Universidade Federal de Ouro Preto)
Diogenes Viegas Mendes Ferreira - (Universidade Federal de Ouro Preto)
[Dra.] - Paulo Henrique Vieira Magalhães - (Universidade Federal de Ouro Preto)

Bruno Nazário Coelho, coorientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 28/08/2025

Documento assinado eletronicamente por **Bruno Nazario Coelho**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 29/08/2025, às 15:10, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0969684** e o código CRC **6B6F1AA5**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591533 - www.ufop.br

Agradecimentos

Este trabalho representa a culminação de anos de dedicação e esforço, e sua realização não seria possível sem o apoio incondicional de pessoas e instituições que foram pilares em minha jornada. Primeiramente, expresso minha profunda gratidão à minha família, fonte inesgotável de amor, incentivo e compreensão. O apoio de vocês, ao longo de toda a minha graduação, foi a força motriz que me impulsionou a superar desafios e a persistir na busca pelos meus objetivos. Cada conquista é, em grande parte, reflexo da dedicação e sacrifício de vocês.

Aos estimados professores do curso de Engenharia de Controle e Automação da Escola de Minas da Universidade Federal de Ouro Preto (UFOP), meu sincero agradecimento por todo o conhecimento compartilhado. As aulas, os desafios propostos e a orientação de vocês foram fundamentais para a minha formação acadêmica e profissional, moldando o engenheiro que sou hoje.

Em especial, dirijo meus agradecimentos ao Professor Doutor Edson Figueira. Sua confiança em mim, ao me conceder o desafio de desenvolver e implementar este sistema de controle em sua bancada de doutorado, foi um privilégio e uma oportunidade ímpar. A possibilidade de trabalhar em um projeto tão relevante, que não apenas aprimora uma pesquisa anterior, mas também será utilizado em aulas práticas de laboratório, é uma honra. Poder contribuir para a formação acadêmica de seus alunos, futuros engenheiros da Escola de Minas, é uma recompensa que transcende o próprio trabalho, e sou imensamente grato por essa valiosa experiência.

"A automação não é sobre substituir pessoas, é sobre economizar tempo, para que as pessoas possam focar no que realmente importa." (Sundar Pichai, CEO do Google)

"Você só tem uma chance, não a deixe escapar. Certas oportunidades só surgem uma vez na vida." (Eminem)

Resumo

Este trabalho detalha o desenvolvimento de um sistema para controle remoto de abertura de válvula industrial e monitoramento de pressão, governado por um microcontrolador ESP32. A metodologia adotada foi a de implementação e teste individual de cada subsistema, permitindo que, a cada melhoria implementada, novas oportunidades de aprimoramento fossem identificadas e exploradas. Dessa forma, cada funcionalidade foi desenvolvida e validada de maneira isolada, garantindo sua robustez antes da integração. Com a montagem final da bancada, o trabalho principal consistiu em validar cada melhoria implementada para uma futura integração em um único sistema coeso. O sistema permite o controle preciso da abertura da válvula através de um motor de passo, o monitoramento da vazão e pressão antes e depois da bomba por meio de sensores, e o controle do bombeamento de fluido abrasivo através de um inversor de frequência. Todas essas funcionalidades são consolidadas em um supervisório online na plataforma Adafruit IO, que possibilita o monitoramento e controle remoto a partir de qualquer dispositivo com acesso à internet. O desempenho do sistema é avaliado pela sua operação remota por meio do supervisório. O projeto, que evoluiu de um conceito inicial com Arduino Mega e ESP8266 para uma solução otimizada com apenas o ESP32, demonstra um sistema mais compacto, eficiente e de fácil implementação.

Palavras-chaves: Controle; Microcontroladores; ESP32; Pressão; Válvula; Motor; Supervisório; Online; Inversor de Frequência.

Abstract

This project details the development of a system for remote control of an industrial valve opening and pressure monitoring, governed by an ESP32 microcontroller. The adopted methodology was the individual implementation and testing of each subsystem, which allowed new improvement opportunities to be identified and explored with each implemented enhancement. This way, each functionality was developed and validated in isolation, ensuring its robustness before integration. With the final assembly of the test bench, the main work consisted of validating each implemented improvement for future integration into a single cohesive system. The system allows for precise control of the valve opening via a stepper motor, monitoring of flow and pressure before and after the pump through sensors, and control of abrasive fluid pumping using a frequency inverter. All these functionalities are consolidated in an online supervisory system implemented on the Adafruit IO platform, which enables remote monitoring and control from any device with internet access. The system's performance is evaluated by its remote operation through the supervisory system. The project, which evolved from an initial concept using an Arduino Mega with an ESP8266 to an optimized solution with only the ESP32, demonstrates a more compact, efficient, and easy-to-implement system.

Key-words: Control; Microcontrollers; ESP32; Pressure; Valve; Motor; Supervisory; Online; Frequency Inverter.

Lista de ilustrações

Lista de abreviaturas e siglas

ADC	Conversor Analógico-Digital (Analog-to-Digital Converter)
ESP32	Microcontrolador com WiFi e Bluetooth integrados
ESP8266	Microcontrolador com WiFi integrado
I2C	Protocolo de comunicação serial (Inter-Integrated Circuit)
IoT	Internet das Coisas (Internet of Things)
MQTT	Message Queuing Telemetry Transport
OLED	Diodo Orgânico Emissor de Luz (Organic Light-Emitting Diode)
SDA	Linha de Dados Serial (Serial Data Line)
SCL	Linha de Clock Serial (Serial Clock Line)
SSD1306	Controlador de display OLED
WiFi	Tecnologia de comunicação sem fio

Sumário

1 Introdução

1.1 Revisão da Literatura

As válvulas de controle representam componentes fundamentais em sistemas industriais, correspondendo a aproximadamente 5% dos custos totais na implementação de tubulações industriais e perdendo apenas para as conexões em termos de quantidade de unidades instaladas (BOJORGE; ALVES; DREUX, 2017). Este setor constitui um mercado estável avaliado em bilhões de dólares anualmente (ALLIED MARKET RESEARCH, 2020), o que evidencia sua relevância econômica e tecnológica.

Em ambientes industriais modernos, o controle preciso de processos tornou-se essencial para garantir eficiência operacional, qualidade de produtos e segurança. Neste contexto, a automação de válvulas e seu monitoramento remoto representam avanços significativos, permitindo ajustes precisos e em tempo real de variáveis críticas como pressão, temperatura e vazão (OGATA, 2010). A integração destas funcionalidades com tecnologias de Internet das Coisas (IoT) possibilita o controle e monitoramento à distância, reduzindo a necessidade de intervenção humana direta e minimizando riscos em ambientes perigosos (EMERSON AUTOMATION SOLUTIONS, 2020).

A implementação de sistemas de controle remoto para válvulas industriais representa uma tendência crescente na indústria 4.0, onde a conectividade e o monitoramento em tempo real são requisitos essenciais para a otimização de processos (MATHIAS; RAMOS; BORTONI, 2014). A capacidade de monitorar simultaneamente a pressão em diferentes pontos do sistema permite uma análise mais completa do comportamento do fluido, do desempenho da válvula e também da exigência e desgaste da bomba, contribuindo para a tomada de decisões mais precisas e para a implementação de estratégias de controle e manutenções mais eficientes.

Este estudo aborda o desenvolvimento de um sistema de controle remoto de abertura de válvula industrial e monitoramento de pressão utilizando o microcontrolador ESP32. O sistema desenvolvido permite a abertura e o fechamento de uma válvula industrial de forma precisa, além de possibilitar o monitoramento da pressão em tempo real por meio de um supervisório online, o que permite a configuração remota do sistema. O projeto também inclui o controle de um inversor de frequência para acionamento da bomba de fluido abrasivo presente na mesma bancada, integrando todas estas funcionalidades em uma única plataforma.

O sistema proposto tem potencial para ser aplicado em diversas áreas da indústria, como no controle de processos de tratamento de água, controle de processos químicos e

petroquímicos, e na automação de processos de produção de alimentos e bebidas. A precisão e o controle oferecidos pelo sistema podem melhorar significativamente a eficiência e a qualidade dos processos industriais, tornando-os mais seguros e confiáveis.

1.2 Justificativas e Relevância

O desenvolvimento deste sistema é justificado por diversos fatores técnicos, econômicos e acadêmicos:

- **Evolução tecnológica**: A integração de tecnologias IoT com sistemas de controle industrial representa uma evolução natural dos processos de automação, alinhando-se com os conceitos da Indústria 4.0 (ACOSTA; TODOROVICH; VÁZQUEZ, 2012).
- **Eficiência operacional**: O controle remoto e o monitoramento em tempo real permitem ajustes mais rápidos e precisos, reduzindo tempos de resposta e minimizando perdas de produção (RIBEIRO; SILVA, 1999).
- **Segurança industrial**: A operação remota reduz a exposição de operadores a ambientes potencialmente perigosos, como áreas com altas temperaturas, pressões elevadas ou substâncias tóxicas (DRISKELL, 1983).
- **Otimização de recursos**: O monitoramento simultâneo de pressão e temperatura permite uma análise mais completa do comportamento do sistema, facilitando a identificação de ineficiências e oportunidades de melhoria (WHITE, 2018).
- **Redução de custos operacionais**: A automação e o monitoramento remoto reduzem a necessidade de inspeções presenciais frequentes, diminuindo custos com mão de obra e deslocamentos (SINES; AKHTAR, 2018).
- **Manutenção preditiva**: Os dados coletados pelos sensores podem ser utilizados para identificar tendências e prever falhas antes que ocorram, reduzindo paradas não programadas (EMERSON AUTOMATION SOLUTIONS, 2020).
- **Flexibilidade de controle**: A implementação em ESP32 permite a adaptação do sistema para diferentes tipos de válvulas e aplicações, aumentando sua versatilidade (ESPRESSIF SYSTEMS, 2021).
- **Contribuição acadêmica**: O desenvolvimento e documentação deste sistema contribuem para o avanço do conhecimento na área de automação industrial e controle de processos, servindo como referência para futuros trabalhos.

Além destes fatores, o sistema desenvolvido pode ser utilizado como plataforma educacional para o ensino de conceitos de automação, controle, instrumentação e IoT,

contribuindo para a formação de profissionais mais bem preparados para os desafios da indústria.

O desenvolvimento deste sistema também é justificado por diversos fatores:

- Dar continuidade e aplicar melhorias na bancada desenvolvida por outros projetos, contribuindo para o avanço do conhecimento na área

  A pesquisa e o desenvolvimento em engenharia frequentemente se beneficiam da continuidade e aprimoramento de projetos preexistentes, permitindo a validação de conceitos anteriores e a introdução de inovações incrementais que impulsionam o avanço do conhecimento na área. Esta abordagem colaborativa e iterativa é fundamental para a construção de soluções mais robustas e eficientes, consolidando o aprendizado e a experiência acumulados em trabalhos anteriores (SILVA; SANTOS, 2019).

- Desenvolver um sistema de controle de válvula utilizando ESP32, explorando as capacidades deste microcontrolador para aplicações industriais

- Implementar um sistema de monitoramento de pressão para avaliar o desempenho da bomba em diferentes condições de operação

- Integrar um supervisório online para monitorar e controlar remotamente a abertura da válvula e o acionamento da bomba

- Validar experimentalmente o sistema de controle, verificando sua eficiência e precisão, garantindo a constante segurança das pessoas e do sistema durante as operações

- Atender à necessidade de garantir uma operação eficiente e segura de sistemas de transporte e distribuição de fluidos em diversas aplicações industriais e comerciais

  A segurança e a eficiência operacional são pilares essenciais na gestão de sistemas de transporte e distribuição de fluidos, especialmente em ambientes industriais. A implementação de tecnologias de controle e monitoramento avançadas é crucial para mitigar riscos, prevenir falhas e otimizar o desempenho, garantindo a integridade dos processos e a proteção dos operadores. Normas e diretrizes internacionais frequentemente regulamentam estas práticas, visando a conformidade e a excelência operacional (INTERNATIONAL SOCIETY OF AUTOMATION, 2016).

- Contribuir para o aprimoramento da gestão de recursos e processos em diversas áreas, o que pode levar a uma redução de custos, aumento da produtividade e minimização de erros humanos

- Explorar o uso de dispositivos eletrônicos e programação em aplicações de engenharia, desenvolvendo competências essenciais para os profissionais da área

Além de todos esses fatores, a bancada completa desse sistema será utilizada pelo professor orientador para a realização de aulas práticas, contribuindo para a formação de futuros engenheiros da Escola de Minas.

1.3 Metodologia

O trabalho apresentado aqui insere-se em uma linha de pesquisa estabelecida e desenvolvida no Laboratório de Fenômenos de Transporte da Universidade Federal de Uberlândia e, posteriormente, na Universidade Federal de Ouro Preto, dando continuidade e aprimorando o projeto de automação de uma válvula globo. A bancada experimental utilizada, que serve como base para esta pesquisa, foi originalmente desenvolvida como parte do trabalho de doutorado do Professor Orientador Edson Figueira e, posteriormente, utilizada e aprimorada na monografia de Rafael Cardoso (CARDOSO, 2021). Essa trajetória prévia forneceu uma base sólida para o desenvolvimento atual, permitindo focar nas inovações e otimizações do sistema.

Embora o trabalho de Cardoso tenha sido fundamental como ponto de partida, a abordagem de controle e automação foi integralmente redesenhada. Optou-se por uma reestruturação completa do código, distanciando-se das implementações anteriores para garantir maior flexibilidade, robustez e escalabilidade. As melhorias implementadas são significativas e abrangem diversos aspectos do sistema. Primeiramente, houve um upgrade do hardware de controle, migrando do Arduino Mega, utilizado por Cardoso, para o ESP32. Essa transição não apenas confere maior capacidade de processamento e memória, mas também integra funcionalidades de conectividade Wi-Fi e Bluetooth nativamente, simplificando a arquitetura do sistema.

No que diz respeito ao controle da válvula, a metodologia foi refinada para se basear exclusivamente na contagem de passos do motor de passo em relação à amplitude de abertura desejada da válvula. Essa abordagem simplifica o controle de posição e aumenta a precisão da abertura. Adicionalmente, a capacidade de conexão à internet do ESP32 foi explorada para implementar um sistema de controle e monitoramento online. Isso permite que o sistema seja operado e seus dados sejam visualizados remotamente através de uma interface supervisória baseada na web, complementando o controle tradicional via Serial Monitor da Arduino IDE. Por fim, uma importante adição foi a integração com o inversor de frequência, possibilitando o acionamento e desligamento remoto da bancada experimental, conferindo maior autonomia e segurança à operação.

O sistema desenvolvido utiliza um motor de passo (modelo Wotiom Nema WS23-24kgf junto do Driver WD-2404) fixado no volante da válvula para realizar a abertura e fechamento da mesma. O motor de passo é controlado pelo ESP32, que ajusta a abertura da válvula de acordo com os comandos recebidos do supervisório online.

A bancada de testes utilizada para a caracterização da válvula, composta por elementos como reservatório de água, motobomba e medidores de vazão e pressão, assemelha-se à configuração descrita por (CARDOSO, 2021) na seção 4.4. A seleção de componentes como o motor de passo NEMA WS23 e o controlador WD-2440, conforme listado na seção 4.1 da monografia de Cardoso, também serviu como referência para a escolha dos materiais em nosso projeto.

Figura 1 – Válvula adaptada com o motor montada na bancada. Fonte: (CARDOSO, 2021)

Para o monitoramento do sistema, são utilizados dois sensores de pressão, um posicionado antes e outro depois da bomba, permitindo a análise da queda e aumento de pressão em diferentes níveis de abertura. O sistema também inclui o controle de um inversor de frequência Weg CFW10 Easydrive para acionamento da bomba de fluido abrasivo e um sensor de vazão posicionado pós-bomba.

A metodologia envolve a implementação do sistema em um único microcontrolador ESP32, a calibração dos sensores de pressão e vazão, a configuração do supervisório online na plataforma Adafruit IO e a integração com o inversor de frequência. O desempenho do sistema é avaliado por meio de testes experimentais que consistem na medição da pressão em diferentes aberturas da válvula e no controle remoto da mesma por meio do supervisório online, além do constante monitoramento da vazão.

1.3.1 Materiais utilizados

Os seguintes materiais foram utilizados no desenvolvimento deste projeto:

- Válvula industrial do tipo Globo
- Motor de passo Wotiom Nema WS23-24kgf
- Driver WD-2404
- Microcontrolador ESP32 WROOM-32
- Torre com trilho de fixação do motor
- Sensor de fim de curso para o trilho da torre
- Display OLED SSD1306
- Sensores transmissor de pressão ASTA TP110
- Sensor de vazão MAG600
- Inversor de frequência Weg CFW10 Easydrive
- Protoboard
- Fios e cabos (quantidade variável)
- Conversor AC/DC
- Bancada de fluido abrasivo

1.4 Cronograma

O desenvolvimento deste projeto seguiu o seguinte cronograma:

- Migração do sistema de Arduino Mega com ESP8266 para ESP32
- Desenvolvimento do supervisório online e integração com o sistema
- Implementação do controle do inversor de frequência
- Montagem da bancada e fixação dos componentes
- Realização de testes finais para validação de cada sistema
- Análise e interpretação dos resultados
- Redação do trabalho de conclusão de curso
- Revisão e submissão do trabalho

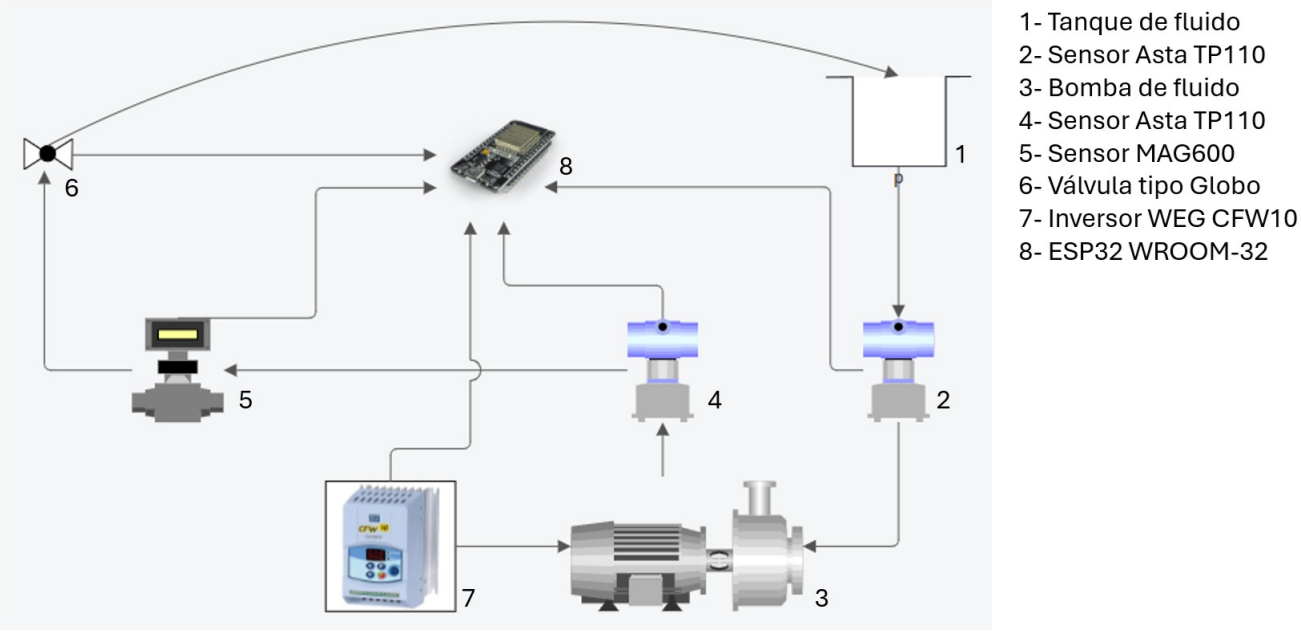

Figura 2 – Representação da montagem da bancada por um diagrama. Fonte: O Autor

Figura 3 – Bancada montada no laboratório. Fonte: O Autor

2 Migração de Arduino Mega com ESP8266 para ESP32

O desenvolvimento deste projeto passou por uma importante evolução em termos de hardware, migrando de uma configuração inicial que utilizava Arduino Mega integrado com módulo ESP8266 para uma solução mais compacta e eficiente baseada exclusivamente no microcontrolador ESP32. Esta seção detalha as motivações, desafios e benefícios dessa migração, bem como as adaptações necessárias para implementar todas as funcionalidades do sistema em uma única plataforma.

Figura 4 – Arduino Mega WiFi - ESP 8266. Fonte: (BAÚ DA ELETRÔNICA, 2025)

2.1 Configuração inicial: Arduino Mega com ESP8266

Na concepção inicial do projeto, a arquitetura de hardware foi planejada utilizando um Arduino Mega como unidade central de processamento, controlador este que já estava sendo utilizado nas primeiras versões do projeto, e que era responsável pelo controle do motor de passo, enquanto um módulo ESP8266, adicionado posteriormente, seria utilizado exclusivamente para prover conectividade WiFi e comunicação com o serviço Adafruit IO para o supervisório online.

Esta abordagem apresentava algumas vantagens iniciais:

- Separação clara de responsabilidades: o Arduino Mega gerenciaria toda a lógica de controle e o ESP8266 seria dedicado à comunicação WiFi
- Maior número de portas digitais e analógicas disponíveis no Arduino Mega
- Familiaridade com a plataforma Arduino para desenvolvimento do código principal
- Manutenção do hardware já disponível nas primeiras versões do projeto

No entanto, esta configuração também apresentava desafios significativos:

- Necessidade de comunicação serial entre o Arduino Mega e o ESP8266
- Complexidade na sincronização de dados entre os dois dispositivos
- Maior consumo de energia com dois microcontroladores
- Maior espaço físico necessário para acomodar ambos os dispositivos
- Duplicidade de código e maior complexidade de manutenção

2.2 Motivações para a migração para ESP32

A evolução dos microcontroladores para aplicações de Internet das Coisas (IoT) tem sido marcada pelo surgimento de plataformas como o ESP32, que se destacam por sua capacidade de processamento dual-core, maior quantidade de memória e integração de módulos Wi-Fi e Bluetooth em um único chip. Tais características conferem ao ESP32 uma vantagem significativa em termos de desempenho e eficiência energética em comparação com soluções que combinam microcontroladores mais antigos, como o Arduino Mega, com módulos de comunicação externos, como o ESP8266. Essa consolidação de funcionalidades em um único dispositivo simplifica a arquitetura do sistema, reduz o consumo de energia e otimiza o espaço físico, tornando-o ideal para aplicações embarcadas complexas (GUPTA; KUMAR; SINGH, 2019; PATEL; SHAH; MEHTA, 2020).

A decisão de migrar para o ESP32 foi motivada por diversos fatores técnicos e práticos que contribuíram para uma implementação mais eficiente e robusta do sistema. Entre as principais motivações, foram destacados:

2.2.1 Maior facilidade de implementação

O ESP32 permite a integração de todas as funcionalidades em um único dispositivo, o que elimina a necessidade de comunicação entre múltiplos microcontroladores. Isso simplifica significativamente a arquitetura do sistema e reduz pontos potenciais de falha. Com um único código executando em um único dispositivo, a implementação torna-se mais direta e menos propensa a erros de sincronização e comunicação.

2.2.2 Disponibilidade de portas analógicas

Uma das limitações do ESP8266 era a disponibilidade de apenas uma porta analógica, o que seria insuficiente para o projeto que necessita de múltiplas entradas analógicas para os sensores de pressão e vazão. O ESP32, por outro lado, possui um conversor analógico-digital (ADC) de 12 bits com múltiplos canais, permitindo a leitura simultânea de vários sensores analógicos com maior precisão, possibilitando também futuras melhorias, como a adição de novos sensores para monitoramento do sistema como um todo.

2.2.3 Possibilidade de trabalhar com um único código

A migração para o ESP32 eliminou a necessidade de desenvolver e manter dois códigos separados (um para o Arduino Mega e outro para o ESP8266), simplificando o desenvolvimento e a manutenção do software. Isso também facilita futuras atualizações e melhorias no sistema, pois todas as modificações podem ser feitas em um único arquivo.

2.2.4 Maior potência e capacidade de processamento

O ESP32 possui um processador dual-core de 32 bits operando a até 240MHz, significativamente mais potente que o Arduino Mega (16MHz) e o ESP8266 (80MHz). Esta maior capacidade de processamento permite a execução simultânea de múltiplas tarefas, como controle do motor de passo, leitura de sensores, atualização do display OLED e comunicação WiFi, sem comprometer o desempenho do sistema (ESPRESSIF SYSTEMS, 2021).

2.2.5 Maior compacidade do sistema

A utilização de um único microcontrolador reduz significativamente o tamanho físico do sistema, tornando-o mais compacto e facilitando sua instalação em espaços restritos. Isso também reduz a quantidade de cabos e conexões necessárias, aumentando a confiabilidade do sistema.

2.2.6 Conexão direta ao WiFi sem necessidade de intermediador

O ESP32 possui módulo WiFi integrado de alta performance, eliminando a necessidade de um dispositivo intermediário para comunicação com a internet. Isso não apenas simplifica a arquitetura do sistema, mas também melhora a confiabilidade e a velocidade da comunicação com o serviço Adafruit IO.

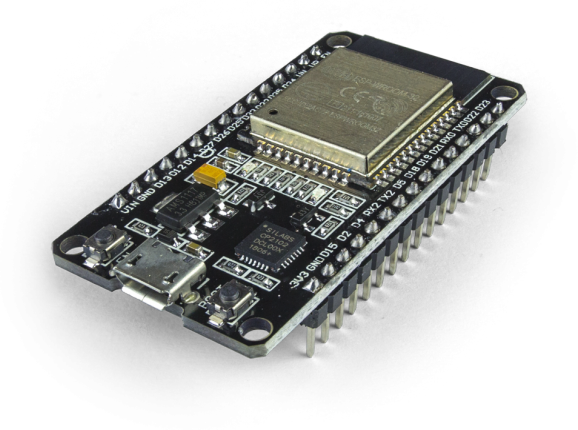

Figura 5 – Modelo de controlador ESP32 utlizado no projeto. Fonte: (ROBOCORE, 2025)

2.3 Desafios da migração

A migração de uma arquitetura baseada em Arduino Mega com ESP8266 para ESP32 apresentou alguns desafios que precisaram ser superados:

2.3.1 Adaptação do código

Foi necessário adaptar o código existente para a nova plataforma, considerando as diferenças entre as bibliotecas e APIs do Arduino Mega e do ESP32. Isso incluiu a adaptação das funções de controle do motor de passo, leitura dos sensores e comunicação com o display OLED.

2.3.2 Remapeamento de pinos

O ESP32 possui uma pinagem diferente do Arduino Mega, o que exigiu um retrabalho completo em cima das conexões físicas. Foi necessário considerar cuidadosamente quais pinos do ESP32 seriam utilizados para cada função, levando em conta as limitações e características específicas de cada pino.

2.3.3 Ajustes na comunicação WiFi

Embora o ESP32 possua WiFi integrado, as bibliotecas e métodos de conexão são diferentes dos utilizados no ESP8266. Foi necessário adaptar o código de comunicação WiFi e integração com o Adafruit IO para a nova plataforma.

2.4 Implementação com ESP32

A implementação final do sistema utilizando o ESP32 resultou em um código mais integrado e eficiente. As principais características desta implementação incluem:

2.4.1 Configuração de pinos e periféricos

O ESP32 foi configurado para controlar todos os periféricos do sistema:

- Pinos 12 (STEP) e 14 (DIR) para controle do motor de passo através do driver WD-2404
- Pino 27 para o sensor de fim de curso
- Pino 34 para o sensor de vazão
- Pinos 36 e 39 para leitura dos sensores de pressão (ADC1_CH0 e ADC1_CH3)
- Pinos 26 e 25 para controle e monitoramento do inversor de frequência
- Pinos 21 (SDA) e 22 (SCL) para comunicação I2C com o display OLED

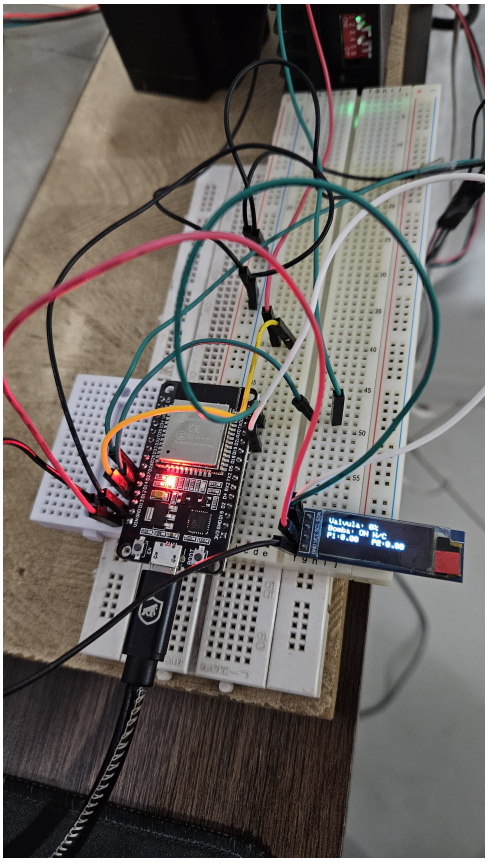

Figura 6 – Sistema montado na protoboard utilizando a ESP32. Fonte: O Autor

2.4.2 Gerenciamento de tarefas

O ESP32 possui capacidade de processamento suficiente para executar todas as tarefas necessárias de forma eficiente:

- Controle do motor de passo para ajuste da abertura da válvula
- Leitura e processamento dos sinais dos sensores de pressão e vazão
- Controle do inversor de frequência para acionamento da bomba
- Atualização do display OLED com informações do sistema
- Comunicação WiFi com o serviço Adafruit IO para o supervisório online
- Processamento de comandos recebidos do supervisório
- Envio de dados de monitoramento para o supervisório

2.4.3 Benefícios da implementação com ESP32

A implementação final com ESP32 trouxe diversos benefícios para o projeto:

- Redução da complexidade do sistema
- Maior confiabilidade devido à eliminação de comunicação entre dispositivos
- Menor consumo de energia
- Menor tamanho físico
- Maior facilidade de manutenção e atualização
- Melhor desempenho geral do sistema

2.5 Considerações sobre a migração

A migração de Arduino Mega com ESP8266 para ESP32 representou uma evolução significativa, resultando em um sistema mais eficiente, compacto e confiável. Esta experiência demonstra a importância de avaliar cuidadosamente as opções de hardware disponíveis e estar aberto a mudanças durante o desenvolvimento de um projeto, buscando sempre a solução mais adequada para os requisitos específicos.

3 Sistema de Monitoramento de Pressão

Sensores de pressão são dispositivos cruciais em sistemas de monitoramento industrial, convertendo pressão em um sinal elétrico mensurável. Embora as especificações exatas variem entre fabricantes e modelos, a maioria dos sensores analógicos de pressão para aplicações gerais opera com uma faixa de medição definida (e.g., 0-100 kPa), fornecendo uma saída de tensão proporcional à pressão aplicada (e.g., 0.5-4.5V). A precisão, tempo de resposta e faixa de temperatura de operação são parâmetros críticos que devem ser considerados na seleção, sendo tipicamente detalhados em seus respectivos *datasheets* (HONEYWELL SENSING AND PRODUCTIVITY SOLUTIONS, 2018). O modelo ASTA TP110, um transdutor de pressão industrial, opera com uma saída de corrente de 4 a 20 mA, o que exige uma conversão para tensão antes da leitura pelo Conversor Analógico-Digital (ADC) do microcontrolador (ASTA INSTRUMENTAÇÃO CONTROLE, 2021).

Figura 7 – Sensor de Pressão ASTA TP110. Fonte: (ASTA INDÚSTRIA E COMÉRCIO DE INSTRUMENTAÇÃO E CONTROLE LTDA., 2008)

O sistema de monitoramento de pressão permite a análise do comportamento da bomba em diferentes condições de operação e fornece dados essenciais para o controle e otimização do sistema. Esta seção detalha a implementação deste sistema, incluindo os sensores utilizados, a calibração, o processamento dos sinais e a integração com o ESP32 e o supervisório online.

3.1 Sensores de pressão

Para o monitoramento da pressão no sistema, foram utilizados dois sensores de pressão modelo ASTA TP110, estrategicamente posicionados antes e depois da bomba. Estes

sensores permitem a medição da pressão em tempo real e a análise do comportamento do sistema como um todo.

3.1.1 Características técnicas dos sensores

O transmissor de pressão ASTA TP110 é um dispositivo robusto, projetado para aplicações industriais que exigem durabilidade, precisão e qualidade. Suas principais características técnicas são:

- Modelo: TP 110
- Faixa de pressão: -1 a 1000 bar (equivalente a -100 kPa a 100.000 kPa)
- Tipo de sensor: Piezorresistivo
- Sinal de Saída: 4 a 20 mA a 2 fios
- Alimentação: 15 a 30 Vcc
- Precisão: 0,5% F.E. (Fundo de Escala)
- Grau de proteção do invólucro: IP-65
- Material do invólucro: Aço inoxidável AISI 304
- Conexão: 1/2”NPT/BSP

3.1.2 Posicionamento dos sensores

Os sensores foram posicionados estrategicamente no sistema:

- Sensor 1 (pressão de entrada): Instalado na tubulação antes da bomba, medindo a pressão de entrada.
- Sensor 2 (pressão de saída): Instalado na tubulação após a bomba, medindo a pressão de saída.

Este posicionamento permite a análise da variação da pressão através da bomba, um parâmetro importante para avaliar o desempenho do sistema em diferentes condições de operação.

3.2 Integração com o ESP32

A integração dos sensores de pressão ASTA TP110 com o ESP32 foi realizada através das entradas analógicas do microcontrolador, necessitando de um circuito de condicionamento de sinal para converter a saída de corrente 4-20mA em uma tensão legível pelo ESP32.

3.2.1 Circuito de Condicionamento de Sinal

Para converter o sinal de corrente de 4-20mA em uma tensão que o ESP32 possa ler (0-3.3V), utiliza-se um resistor shunt de 150 Ω em série com o sensor. A queda de tensão nesse resistor será proporcional à corrente. Para 4mA, a tensão será de 0.6V, e para 20mA, será de 3.0V. Essa faixa de tensão é compatível com o ADC de 12 bits do ESP32, que opera de 0 a 3.3V. É crucial que o GND do sensor e do ESP32 estejam interligados e que o sensor seja alimentado por uma fonte externa de 15-30Vcc.

3.2.2 Conexão física

Os sensores de pressão serão conectados ao ESP32 da seguinte forma:

- Pino VCC dos sensores: Conectado à fonte de alimentação externa (15-30Vcc).
- Pino GND dos sensores: Conectado ao pino GND do ESP32 e ao GND da fonte externa.
- Saída de sinal do sensor (após o resistor shunt): Conectado ao pino 36 (ADC1_CH0) do ESP32 para o Sensor 1.
- Saída de sinal do sensor (após o resistor shunt): Conectado ao pino 39 (ADC1_CH3) do ESP32 para o Sensor 2.

3.2.3 Configuração no código

A configuração dos sensores de pressão no código é realizada através das seguintes definições e inicializações:

```
// Configurações dos sensores de pressão
#define PRESSURE1_SENSOR_PIN 36 // Pino para o sensor de pressão 1 (pré-válvula) AS
#define PRESSURE2_SENSOR_PIN 39 // Pino para o sensor de pressão 2 (pós-válvula) AS

// No setup()
// Configuração do ADC
analogReadResolution(12);  // Resolução de 12 bits (0-4095)
analogSetAttenuation(ADC_11db); // Atenuação para permitir leitura de tensões até 3
```

3.3 Calibração dos sensores

A calibração dos sensores de pressão é essencial para garantir a precisão das medições. O processo de calibração envolve a determinação da relação entre o valor lido pelo ADC do

ESP32 e a pressão real no sistema, considerando a conversão de corrente para tensão e a faixa de operação do sensor.

3.3.1 Equação de calibração

A relação entre o valor lido pelo ADC e a pressão real é determinada pela função de conversão de 4-20mA para o valor físico, conforme descrito no código:

```
// Constantes de calibração para sensores 4-20mA
const float ADC_MAX_VALUE = 4095.0;
const float ADC_VOLTAGE_MAX = 3.3;
const float R_SHUNT = 150.0; // Ohms

// Faixa do sensor de pressão ASTA TP110: -1 a 1000 bar (convertido para kPa)
const float PRESSURE_MIN_BAR = -1.0; // bar
const float PRESSURE_MAX_BAR = 1000.0; // bar
const float PRESSURE_MIN_KPA = PRESSURE_MIN_BAR * 100.0; // kPa
const float PRESSURE_MAX_KPA = PRESSURE_MAX_BAR * 100.0; // kPa

// Função para converter leitura ADC para valor físico (pressão)
float convert4_20mA_to_Physical(int rawADC, float minPhysical, float maxPhysical) {
  float voltage = (rawADC / ADC_MAX_VALUE) * ADC_VOLTAGE_MAX;
  float current_mA = (voltage / R_SHUNT) * 1000.0;
  current_mA = constrain(current_mA, 4.0, 20.0);
  float physicalValue = map(current_mA * 100, 400, 2000, minPhysical * 100, maxPhysical
  return physicalValue;
}
```

3.4 Leitura e processamento dos sinais

A leitura e processamento dos sinais dos sensores de pressão são realizados periodicamente no loop principal do programa.

3.4.1 Leitura dos sensores

A leitura dos sensores é realizada na função `readSensors()`:

```
// Função para ler os sensores
void readSensors() {
  // Ler pressão 1 (pré-válvula)
```

```
  int pressure1Raw = analogRead(PRESSURE1_SENSOR_PIN);
  pressure1 = convert4_20mA_to_Physical(pressure1Raw, PRESSURE_MIN_KPA, PRESSURE_MA

  // Ler pressão 2 (pós-válvula)
  int pressure2Raw = analogRead(PRESSURE2_SENSOR_PIN);
  pressure2 = convert4_20mA_to_Physical(pressure2Raw, PRESSURE_MIN_KPA, PRESSURE_MA

  // ... (outras leituras de sensores, se houver)
}
```

3.4.2 Análise da pressão

A diferença entre as leituras dos dois sensores de pressão (pressão 1 - pressão 2) representa a variação de pressão através da bomba. Este parâmetro é fundamental para avaliar o desempenho do sistema e pode ser utilizado para:

- Avaliar a eficiência do sistema em diferentes níveis de abertura da válvula.
- Detectar obstruções ou problemas na válvula.
- Otimizar o controle do sistema para minimizar perdas de energia.
- Avaliar o esforço e desgaste da bomba em diferentes cenários.

A análise da pressão em função da abertura da válvula permite estabelecer uma relação entre estes parâmetros, fornecendo informações valiosas para o controle e otimização do sistema.

3.5 Considerações sobre o Sistema de Monitoramento de Pressão

O sistema de monitoramento de pressão implementado neste projeto, utilizando os sensores ASTA TP110, representa uma solução robusta e precisa para a medição e análise da pressão em sistemas de controle de válvulas industriais. A utilização de dois sensores de pressão, estrategicamente posicionados, permite uma análise completa do comportamento do sistema em diferentes condições de operação.

A integração com o ESP32 e o supervisório online proporciona flexibilidade e facilidade de monitoramento, tanto local quanto remoto, contribuindo significativamente para a eficiência e confiabilidade do sistema como um todo.

4 Sistema de Monitoramento de Vazão

A medição de vazão é um parâmetro essencial em diversos processos industriais, permitindo o controle preciso de fluidos e a otimização de sistemas. Medidores de vazão eletromagnéticos, como o Siemens SITRANS FM MAG 6000, são amplamente utilizados devido à sua alta precisão, confiabilidade e capacidade de medir a vazão de líquidos condutivos sem obstrução ao fluxo (SIEMENS, 2021). A integração desses dispositivos em sistemas de controle automatizados, como o desenvolvido neste trabalho, é crucial para o monitoramento em tempo real e a tomada de decisões baseada em dados.

O sistema de monitoramento de vazão implementado neste projeto visa fornecer dados precisos sobre o fluxo de fluido na bancada, complementando as informações de pressão para uma análise abrangente do desempenho do sistema. Esta seção detalha o sensor de vazão utilizado, suas características, o método de integração com o ESP32 e o processamento do sinal.

4.1 Sensor de Vazão MAG600

Para o monitoramento da vazão no sistema, foi utilizado um sensor de vazão do tipo MAG600 (referindo-se a um medidor de vazão eletromagnético da série MAG da Siemens), que oferece medições precisas e confiáveis em ambientes industriais.

Figura 8 – Sensor de Vazão MAG600. Fonte: (DAYLER, 2025)

4.1.1 Características técnicas do sensor

Os medidores de vazão eletromagnéticos da série MAG da Siemens, como o MAG 6000, compartilham características importantes para aplicações de controle e monitoramento:

- Tipo: Medidor de vazão eletromagnético

- Faixa de medição: Variável, dependendo do diâmetro da tubulação e modelo específico (ex: 0 a 100 L/min para aplicações de bancada)
- Sinal de Saída: 4 a 20 mA a 2 fios
- Alimentação: Geralmente 15 a 30 Vcc
- Precisão: Alta, tipicamente ±0,2% do valor medido
- Material do invólucro: Aço inoxidável ou similar, para ambientes industriais
- Conexão: Flangeada ou roscada, dependendo do modelo

4.1.2 Posicionamento do sensor

O sensor de vazão foi instalado em linha com a tubulação principal do sistema, em um trecho reto e sem obstruções, para garantir uma medição precisa e sem turbulências. O posicionamento ideal segue as recomendações do fabricante para medidores eletromagnéticos, que geralmente exigem trechos retos de tubulação antes e depois do sensor para um perfil de fluxo laminar adequado.

4.2 Integração com o ESP32

A integração do sensor de vazão MAG600 com o ESP32 segue o mesmo princípio dos sensores de pressão, utilizando um circuito de condicionamento de sinal para converter a saída de corrente 4-20mA em uma tensão legível pelo microcontrolador.

4.2.1 Circuito de Condicionamento de Sinal

Assim como os sensores de pressão, o sensor de vazão MAG600 com saída 4-20mA requer um resistor shunt de 150 Ω em série para converter a corrente em tensão (0.6V a 3.0V para 4-20mA, respectivamente). Essa tensão é então lida por uma entrada analógica do ESP32. A alimentação do sensor deve ser externa (15-30Vcc) e o GND do sensor e do ESP32 devem ser comuns.

4.2.2 Conexão física

O sensor de vazão será conectado ao ESP32 da seguinte forma:

- Pino VCC do sensor: Conectado à fonte de alimentação externa (15-30Vcc).
- Pino GND do sensor: Conectado ao pino GND do ESP32 e ao GND da fonte externa.
- Saída de sinal do sensor (após o resistor shunt): Conectado ao pino 34 (ADC1_CH6) do ESP32.

4.2.3 Configuração no código

A configuração do sensor de vazão no código é realizada através das seguintes definições e inicializações:

```
// Configurações do sensor de vazão
#define FLOW_SENSOR_PIN 34 // Pino para o sensor de vazão MAG600 (ADC1_CH6)

// No setup()
// Configuração do ADC (já configurado para os sensores de pressão)
// analogReadResolution(12);
// analogSetAttenuation(ADC_11db);
```

4.3 Ajustes e conversão do sensor

O ajuste e conversão do sensor de vazão é crucial para garantir a precisão das medições. O processo envolve a determinação da relação entre o valor lido pelo ADC do ESP32 e a vazão real no sistema, considerando a conversão de corrente para tensão e a faixa de operação do sensor.

4.3.1 Equação de calibração

A relação entre o valor lido pelo ADC e a vazão real é determinada pela função de conversão de 4-20mA para o valor físico, conforme descrito no código:

```
// Constantes de conversão para sensores 4-20mA (já definidas para pressão)
// const float ADC_MAX_VALUE = 4095.0;
// const float ADC_VOLTAGE_MAX = 3.3;
// const float R_SHUNT = 150.0; // Ohms

// Faixa do sensor de vazão (ajuste conforme a especificação real do seu sensor)
const float FLOW_MIN_LPM = 0.0;   // Litros por minuto
const float FLOW_MAX_LPM = 100.0; // Litros por minuto

// Função para converter leitura ADC para valor físico (vazão)
// float convert4_20mA_to_Physical(int rawADC, float minPhysical, float maxPhysical) {
```

4.4 Leitura e processamento do sinal

A leitura e processamento do sinal do sensor de vazão são realizados periodicamente no loop principal do programa.

4.4.1 Leitura do sensor

A leitura do sensor é realizada na função `readSensors()`:

```
// Função para ler os sensores
void readSensors() {
  // ... (leituras de pressão)

  // Ler vazão
  int flowRaw = analogRead(FLOW_SENSOR_PIN);
  flowRate = convert4_20mA_to_Physical(flowRaw, FLOW_MIN_LPM, FLOW_MAX_LPM);

  // ... (envio de dados para Adafruit IO)
}
```

4.5 Considerações sobre o Sistema de Monitoramento de Vazão

O sistema de monitoramento de vazão, utilizando o sensor MAG600, é um componente vital para a compreensão e controle do fluxo de fluido na bancada. A precisão e a confiabilidade das medições de vazão, combinadas com as informações de pressão, fornecem um panorama completo do comportamento hidrodinâmico do sistema. A integração com o ESP32 e o supervisório online permite o monitoramento contínuo e remoto, facilitando a análise de dados e a otimização do processo.

5 Sistema de Controle do Inversor de Frequência

O sistema de controle desenvolvido neste projeto não se limita apenas à abertura e fechamento da válvula, mas também incorpora o controle do bombeamento de fluido abrasivo através de um inversor de frequência, o que possibilita ligar, desligar e monitorar toda a bancada remotamente. Esta seção detalha a implementação do controle do inversor de frequência Weg CFW10 Easydrive, sua integração com o ESP32 e a interface com o supervisório online.

5.1 Inversor de Frequência Weg CFW10 Easydrive

Inversores de frequência, como o modelo CFW10 Easydrive da WEG, são amplamente empregados na indústria para o controle preciso da velocidade de motores elétricos de indução. Estes dispositivos oferecem funcionalidades essenciais como controle de rampa de aceleração e desaceleração, proteções integradas contra sobrecorrente e sobretensão, e diversas interfaces de controle, permitindo a otimização do consumo de energia e a adaptação do motor às demandas específicas do processo. As especificações detalhadas de operação e configuração são fornecidas nos manuais técnicos do fabricante (WEG S.A., 2020).

No contexto deste projeto, o inversor é utilizado para controlar a bomba que impulsiona o fluido abrasivo através do sistema.

5.1.1 Características técnicas do inversor

O Weg CFW10 Easydrive apresenta as seguintes características técnicas relevantes para este projeto:

- Tensão de alimentação: 220-240V monofásico
- Potência: Compatível com a bomba utilizada no sistema
- Interface de controle: Entradas digitais para controle START/STOP
- Saída de status: Sinal digital READY indicando o estado de prontidão do inversor
- Método de controle: V/F (tensão/frequência) linear
- Frequência de saída: 0 a 300 Hz

Figura 9 – Inversor CFW10 utilizado na bancada. Fonte: O Autor

- Proteções integradas: Sobrecorrente, sobretensão, subtensão, sobretemperatura, entre outras

5.1.2 Configuração do inversor

Para a operação neste projeto, o inversor deve estar configurado para operar no modo de controle remoto, permitindo o acionamento da bomba através de sinais digitais provenientes do ESP32. Os principais parâmetros configurados no inversor incluem:

- Modo de controle: Remoto via entradas digitais
- Rampa de aceleração: 5 segundos (para evitar picos de corrente durante a partida)
- Rampa de desaceleração: 5 segundos (para evitar sobretensão durante a parada)
- Frequência mínima: 0 Hz
- Frequência máxima: Ajustada de acordo com a vazão máxima desejada
- Proteção de sobrecarga: Configurada de acordo com as características da bomba

5.2 Integração com o ESP32

A integração do inversor de frequência com o ESP32 foi realizada através de conexões digitais que permitem o controle e monitoramento do estado do inversor.

5.2.1 Conexão física

O inversor de frequência foi conectado ao ESP32 através de dois pinos digitais:

- Pino 26 (INVERTER_START_PIN): Saída digital do ESP32 conectada à entrada START/STOP do inversor
- Pino 25 (INVERTER_READY_PIN): Entrada digital do ESP32 conectada à saída READY do inversor

Para garantir a isolação elétrica entre o ESP32 e o inversor de frequência, é necessário o uso de optoacopladores nas conexões, protegendo o microcontrolador contra possíveis surtos de tensão provenientes do inversor.

5.2.2 Configuração dos pinos no ESP32

Os pinos do ESP32 foram configurados da seguinte forma:

```
// Configuração do inversor de frequência
#define INVERTER_START_PIN 26  // Pino para controle START/STOP do inversor
#define INVERTER_READY_PIN 25  // Pino para monitorar o sinal READY do inversor

// No setup()
pinMode(INVERTER_START_PIN, OUTPUT);
pinMode(INVERTER_READY_PIN, INPUT);
digitalWrite(INVERTER_START_PIN, LOW); // Iniciar com o inversor desligado
```

5.3 Detecção e Verificação de Conexão com o Inversor

Uma característica importante do sistema é a capacidade de detectar automaticamente se o inversor de frequência está conectado. Isso é realizado através da função `checkInverterConnection()`, que verifica se há resposta do inversor aos comandos enviados.

5.4 Controle do Inversor via Supervisório Online

O controle do inversor de frequência (e consequentemente da bomba) é realizado através do supervisório online implementado na plataforma Adafruit IO. Um feed específico (`pump-control`) foi criado para permitir o acionamento remoto da bomba.

5.4.1 Configuração do feed no Adafruit IO

O feed `pump-control` foi configurado como um feed do tipo booleano (ON/OFF), permitindo o controle simples do estado da bomba.

5.4.2 Função de controle da bomba

A função `handlePumpControl()` é responsável por processar os comandos recebidos do supervisório online e controlar o inversor de frequência.

5.5 Monitoramento do Estado do Inversor

O sistema monitora continuamente o estado do inversor de frequência através do sinal READY, que indica se o inversor está pronto para operar ou se está em estado de falha.

5.6 Segurança e Proteção

O sistema implementa diversas medidas de segurança relacionadas ao controle do inversor de frequência:

- Verificação automática da conexão com o inversor
- Inicialização com a bomba desligada
- Monitoramento contínuo do sinal READY
- Ignorar comandos quando o inversor não está conectado
- Exibição clara do estado do inversor no display OLED

Estas medidas garantem a operação segura do sistema, evitando acionamentos indevidos da bomba e permitindo a rápida identificação de problemas.

5.7 Aplicações e Benefícios

O controle remoto do inversor de frequência através do ESP32 e do supervisório online oferece diversos benefícios:

- Operação remota da bomba, permitindo o controle do sistema a partir de qualquer local com acesso à internet
- Integração com o controle da válvula, possibilitando estratégias coordenadas de controle
- Monitoramento em tempo real do estado do inversor e da bomba
- Maior segurança operacional através das verificações automáticas
- Flexibilidade para adaptar o sistema a diferentes requisitos de processo

5.8 Considerações sobre o Sistema de Controle do Inversor

O sistema de controle do inversor de frequência implementado neste projeto representa uma solução robusta e flexível para o acionamento remoto da bomba de fluido abrasivo. A integração com o ESP32 e o supervisório online proporciona facilidade de operação e monitoramento, enquanto as medidas de segurança implementadas garantem a operação confiável do sistema.

A capacidade de detectar automaticamente a presença do inversor e adaptar o comportamento do sistema de acordo é uma característica importante que aumenta a versatilidade e robustez da solução.

6 Interface com o Usuário: Display OLED

Displays OLED (Organic Light-Emitting Diode) baseados no controlador SSD1306 são escolhas populares para projetos embarcados devido ao seu baixo consumo de energia e alto contraste. Comumente disponíveis em resoluções como 128x32 pixels e interface de comunicação I2C, esses displays são ideais para fornecer feedback visual conciso e claro em aplicações onde o espaço e a eficiência energética são críticos. As especificações técnicas detalhadas, incluindo diagramas de conexão e protocolos de comunicação, são fornecidas no *datasheet* do controlador (ADAFRUIT INDUSTRIES, 2014).

A interface com o usuário é um componente fundamental em sistemas de controle industrial, permitindo a visualização de informações críticas e o monitoramento do estado do sistema em tempo real. Neste projeto, foi implementada uma interface local utilizando um display OLED, que fornece informações essenciais sobre o funcionamento do sistema diretamente no local de instalação. Esta seção detalha a implementação desta interface, suas funcionalidades e sua importância para a operação do sistema.

6.1 Display OLED SSD1306

Este display foi escolhido por suas características favoráveis para aplicações embarcadas:

- Tamanho compacto, ideal para integração em painéis de controle
- Baixo consumo de energia, importante para sistemas embarcados
- Alto contraste, permitindo boa visibilidade mesmo em ambientes com iluminação variável
- Interface I2C, que requer apenas dois pinos de comunicação (SDA e SCL)
- Biblioteca de suporte robusta (Adafruit_SSD1306) com ampla documentação

6.1.1 Especificações técnicas

- Resolução: 128x32 pixels
- Interface de comunicação: I2C
- Tensão de operação: 3,3V - 5V
- Dimensões: 0,91 polegadas na diagonal

- Cor: Monocromático (branco)
- Ângulo de visão: >160°

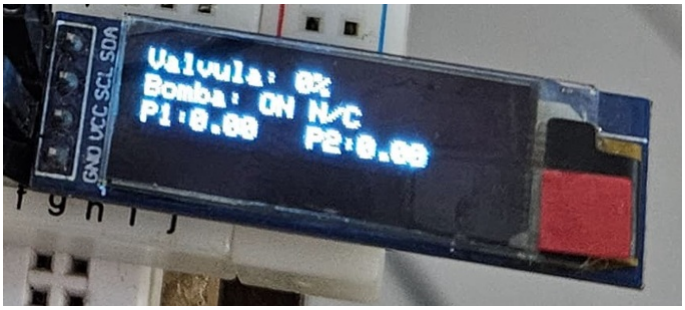

Figura 10 – Display Oled utilizado. Fonte: O Autor

6.2 Integração com o ESP32

A integração do display OLED com o ESP32 foi realizada através da interface I2C, utilizando os pinos dedicados do ESP32 para essa comunicação.

6.2.1 Conexão física

O display OLED foi conectado ao ESP32 da seguinte forma:

- Pino VCC do display: Conectado ao pino 3,3V do ESP32
- Pino GND do display: Conectado ao pino GND do ESP32
- Pino SDA do display: Conectado ao pino 21 (SDA) do ESP32
- Pino SCL do display: Conectado ao pino 22 (SCL) do ESP32

6.3 Funcionalidades da Interface

A interface implementada no display OLED fornece diversas informações importantes sobre o estado do sistema, permitindo o monitoramento local sem a necessidade de acesso ao supervisório online.

6.3.1 Informações exibidas

O display OLED exibe as seguintes informações:

- Porcentagem de abertura da válvula

- Estado da bomba (ON/OFF)
- Estado do inversor de frequência (READY/WAIT/N/C)
- Leituras dos sensores de pressão e vazão (P1, P2 e F)

Estas informações são atualizadas periodicamente, garantindo que o operador tenha acesso aos dados mais recentes sobre o funcionamento do sistema.

6.4 Benefícios da Interface Local

A implementação de uma interface local através do display OLED trouxe diversos benefícios para o sistema:

- Acesso imediato a informações críticas, sem necessidade de conexão à internet
- Feedback visual direto sobre o estado do sistema
- Operação independente do supervisório online, garantindo monitoramento mesmo em caso de falha na conexão
- Facilidade de diagnóstico e manutenção no local
- Confirmação visual de comandos enviados remotamente

6.5 Considerações sobre a Interface com o Usuário

A interface com o usuário implementada através do display OLED representa um componente essencial do sistema, proporcionando monitoramento local e feedback visual sobre o estado e operação. A combinação desta interface local com o supervisório online oferece uma solução completa para o controle e monitoramento do sistema, atendendo às necessidades tanto de operação local quanto remota.

A escolha do display OLED SSD1306 mostrou-se adequada para esta aplicação, oferecendo boa visibilidade, baixo consumo de energia e fácil integração com o ESP32. O layout da interface foi projetado para maximizar a legibilidade e fornecer as informações mais relevantes de forma clara e organizada, contribuindo para a usabilidade e eficiência do sistema como um todo.

7 Supervisório Online com Adafruit IO

Um dos aspectos mais inovadores deste projeto é a implementação de um supervisório online utilizando a plataforma Adafruit IO, que permite o monitoramento e controle remoto do sistema a partir de qualquer dispositivo com acesso à internet. Esta seção detalha a arquitetura, implementação e funcionalidades deste supervisório online, destacando sua importância para a operação e monitoramento do sistema.

Figura 11 – Supervisório online - Dash desenvolvida para o projeto. Fonte: O Autor

7.1 Plataforma Adafruit IO

Plataformas de Internet das Coisas (IoT) baseadas em nuvem, como o Adafruit IO, oferecem uma infraestrutura robusta para a coleta, visualização e controle de dados provenientes de dispositivos conectados. Através de interfaces intuitivas e suporte a protocolos como MQTT, essas plataformas permitem a criação de *dashboards* personalizados, gerenciamento de *feeds* de dados e integração com uma vasta gama de hardware, facilitando o desenvolvimento e a implantação de soluções IoT para monitoramento e controle remoto. A documentação oficial da plataforma detalha suas funcionalidades e APIs para integração (ADAFRUIT INDUSTRIES, 2023).

Esta plataforma foi escolhida para este projeto devido às suas características favoráveis:

- Interface amigável e intuitiva
- Suporte nativo para ESP32 através de bibliotecas dedicadas
- Capacidade de criar dashboards personalizados
- Suporte para diferentes tipos de widgets (gráficos, botões, sliders, etc.)
- Sistema de feeds para organização de dados
- Comunicação bidirecional (envio e recebimento de dados)
- API REST para integração com outros sistemas
- Segurança através de chaves de API
- Possibilidade de integração entre dispositivos mobile em conjunto de computadores tradicionais

7.2 Arquitetura do Supervisório

A arquitetura do supervisório online implementado neste projeto segue o modelo cliente-servidor, onde o ESP32 atua como cliente, enviando e recebendo dados da plataforma Adafruit IO (servidor). A comunicação é realizada através do protocolo MQTT (Message Queuing Telemetry Transport), um protocolo leve e eficiente para comunicação em aplicações IoT.

7.2.1 Feeds do Adafruit IO

Os feeds são os canais de comunicação utilizados para troca de dados entre o ESP32 e a plataforma Adafruit IO. Neste projeto, foram configurados os seguintes feeds:

- `valve-position`: Feed para controle e monitoramento da posição da válvula (0-100%)
- `pump-control`: Feed para controle do estado da bomba (ON/OFF)
- `pressure`: Feed para monitoramento da pressão antes da válvula
- `pressure2`: Feed para monitoramento da pressão depois da válvula
- `vazaoat`: Feed para monitoramento da vazao

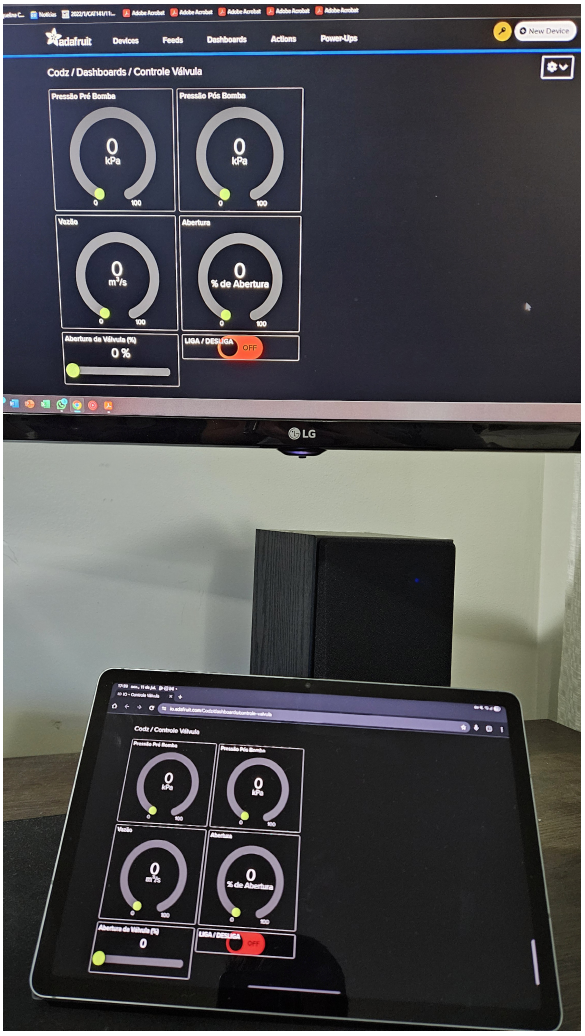

Figura 12 – Dash (painel de controle) aberta em um computador e em um dispositivo móvel. Fonte: O Autor

7.2.2 Dashboard do Adafruit IO

O dashboard é a interface gráfica do supervisório online, onde os dados são visualizados e os comandos são enviados. O dashboard foi configurado com os seguintes widgets:

- Slider para controle da posição da válvula (0-100%)
- Botão para controle da bomba (ON/OFF)
- Indicadores numéricos para exibição das pressões e da vazão
- Indicador de status da conexão

O layout do dashboard foi projetado para ser intuitivo e funcional, agrupando os controles e indicadores de forma lógica e facilitando a operação do sistema.

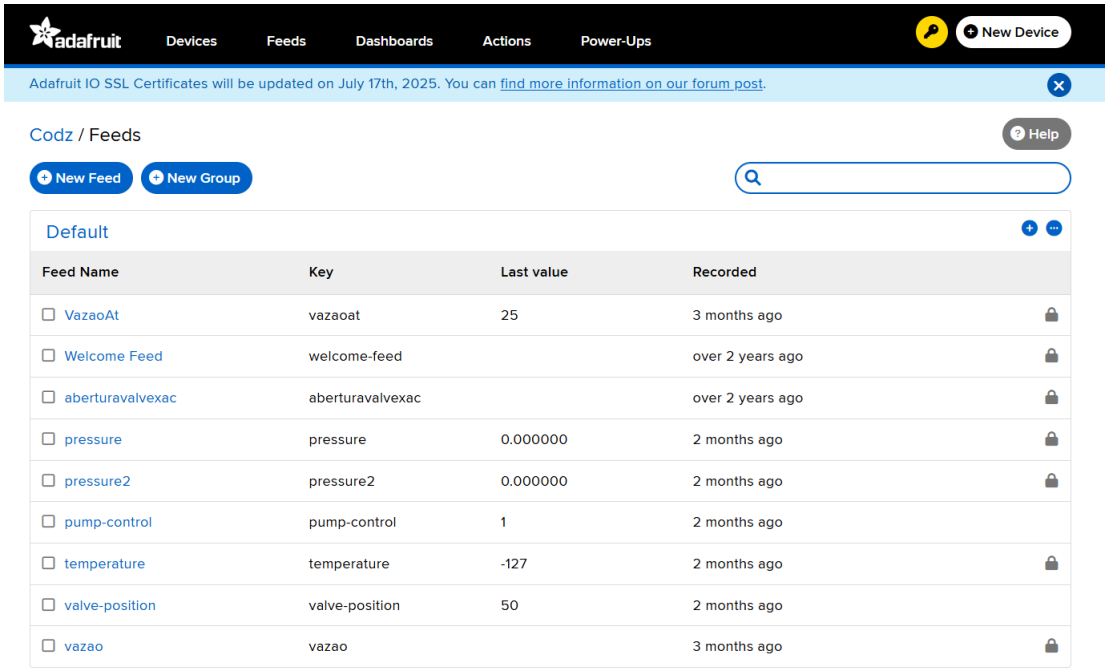

Figura 13 – Área de criação das feeds (fontes de dados) na plataforma Adasruit IO. Fonte: O Autor

7.3 Implementação no ESP32

A implementação do supervisório online no ESP32 foi realizada utilizando a biblioteca AdafruitIO_WiFi, que simplifica a comunicação com a plataforma Adafruit IO.

7.4 Mecanismos de Robustez

Para garantir a operação confiável do supervisório online, foram implementados diversos mecanismos de robustez:

- Operação offline quando necessário
- Reconexão automática em caso de perda de conexão
- Validação de dados recebidos
- Tratamento de erros e exceções

7.5 Benefícios do Supervisório Online

A implementação do supervisório online com Adafruit IO trouxe diversos benefícios para o projeto:

- Controle remoto do sistema a partir de qualquer dispositivo com acesso à internet

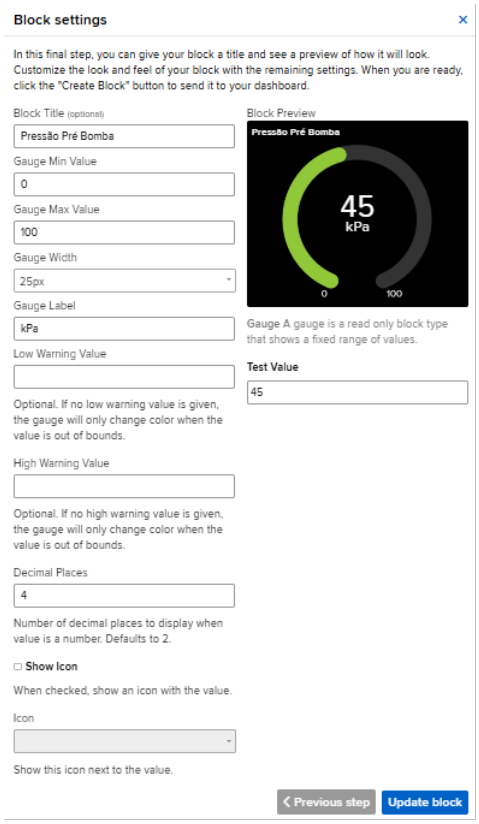

Figura 14 – Área de configuração dos widgets na plataforma Adafruit IO. Fonte: O Autor

- Monitoramento em tempo real das condições de operação
- Registro histórico de dados para análise posterior
- Interface gráfica intuitiva e personalizável
- Possibilidade de configurar alertas e notificações
- Acesso simultâneo por múltiplos usuários e dispositivos
- Integração com outros sistemas através da API REST
- Possibilidade de integração com dispositivos mobile

7.6 Considerações sobre o Supervisório Online

O supervisório online implementado com a plataforma Adafruit IO representa um componente essencial do sistema, proporcionando controle e monitoramento remoto de forma eficiente e intuitiva. A combinação de uma interface local (display OLED) com o supervisório online oferece uma solução completa para a operação do sistema, atendendo às necessidades tanto de operação local quanto remota.

A escolha da plataforma Adafruit IO mostrou-se adequada para esta aplicação, oferecendo todas as funcionalidades necessárias com uma implementação relativamente simples. Os mecanismos de robustez implementados garantem a operação confiável do sistema mesmo em condições adversas, como instabilidades na conexão com a Internet.

O supervisório online não apenas facilita a operação do sistema, mas também agrega valor ao projeto como um todo, demonstrando a aplicação de conceitos modernos de Internet das Coisas (IoT) em um contexto de controle industrial.

8 Sistema de Controle do Motor de Passo e Abertura da Válvula

Um dos componentes centrais deste projeto é o sistema de controle do motor de passo responsável pela abertura e fechamento da válvula industrial. Esta seção detalha a implementação deste sistema, incluindo o hardware utilizado, a configuração do motor de passo, o algoritmo de controle e a integração com o supervisório online.

No desenvolvimento do sistema de automação da válvula, a abordagem para o controle de posição do motor de passo, fundamental para a abertura e fechamento precisos, baseou-se em princípios semelhantes aos explorados por (CARDOSO, 2021). A contagem de passos do motor, que permite o conhecimento da posição do eixo sem a necessidade de realimentação, é um método eficaz para o controle de deslocamentos discretos, conforme detalhado na seção 3.4 da referida monografia.

8.1 Motor de Passo e Driver

Para o controle preciso da abertura da válvula, foi utilizado um motor de passo de alto torque, capaz de movimentar o eixo da válvula mesmo sob condições de alta pressão do fluido.

8.1.1 Motor de passo Wotiom Nema WS23-24kgf

O motor de passo escolhido para este projeto foi o Wotiom Nema WS23-24kgf, que apresenta as seguintes características técnicas:

- Tipo: Motor de passo híbrido bipolar
- Ângulo de passo: 1,8° (200 passos por revolução)
- Torque de retenção: 24 kgf · cm (aproximadamente 2,35 N · m)
- Corrente nominal: 2,8A por fase
- Resistência por fase: 0,9 Ω
- Indutância por fase: 2,5 mH
- Número de fios: 4 (bipolar)
- Dimensões: NEMA 23 (57 mm x 57 mm)

- Peso: Aproximadamente 700g

Figura 15 – Motor de passo Wotiom Nema WS23-24kgf. Fonte: (SOLDAFRIA COMPONENTES ELETRÔNICOS, 2025)

8.1.2 Driver WD-2404

Para controlar o motor de passo, foi utilizado o driver WD-2404, que oferece as seguintes características:

- Tensão de alimentação: 24-50V DC
- Corrente de saída: Ajustável até 4,5A por fase
- Microstepping: Configurável (1, 2, 4, 8, 16, 32 passos)
- Proteções: Sobrecorrente, sobretemperatura, subtensão
- Interface de controle: STEP/DIR (pulso/direção)
- Frequência máxima de pulso: 200 kHz
- Dimensões: 118 mm x 75 mm x 33 mm

8.2 Integração com o ESP32

A integração do motor de passo com o ESP32 foi realizada através do driver WD-2404, utilizando a interface STEP/DIR.

Figura 16 – Driver WD-2404. Fonte: (ARMAZÉM DA ELÉTRICA, 2025)

8.3 Algoritmo de Controle da Válvula

O algoritmo de controle da válvula foi implementado para permitir o ajuste preciso da abertura da válvula de acordo com os comandos recebidos do supervisório online.

8.4 Calibração e Referenciamento

Os procedimentos para o levantamento da característica de vazão da válvula, incluindo a medição do diferencial de pressão e da vazão para diversas aberturas, seguem a metodologia empregada por (CARDOSO, 2021). A análise do comportamento da válvula, que pode apresentar relações não lineares entre o coeficiente de vazão e a abertura, como observado por Cardoso na seção 4.4, é crucial para a compreensão do desempenho do sistema.

Para garantir a precisão do controle da válvula, foi implementado um procedimento de calibração e referenciamento que é executado na inicialização do sistema. Este procedimento é baseado no uso de um sensor de fim de curso posicionado no limite de fechamento da válvula, local este em que ela se encontra completamente fechada. Assim, ao iniciar o sistema, a válvula começa a fechar, pouco a pouco, até que o fim de curso seja acionado, indicando a posição inicial para controle de abertura. Após esta calibração prévia, foi observado que o motor de passo necessita de 700 passos para abrir completamente a válvula, uma vez que a válvula em questão precisa girar 1260° para ir de totalmente fechada para totalmente aberta, e como o motor trabalha com 1.8° por passo, temos então que são necessários 700 passos para percorrer toda a amplitude de abertura, informação esta usada como referência para o controle baseado na porcentagem de abertura desejada pelo operador.

8.5 Controle da Válvula via Supervisório Online

O controle da abertura da válvula é realizado através do supervisório online implementado na plataforma Adafruit IO, além do controle local, através do Monitor Serial presente no Arduino IDE. Um feed específico (`valve-position`) foi criado para permitir o ajuste remoto da abertura da válvula.

8.6 Considerações sobre o Sistema de Controle do Motor de Passo e Abertura da Válvula

O sistema de controle do motor de passo e abertura da válvula implementado neste projeto representa uma solução robusta e precisa para o controle de válvulas industriais. A utilização de um motor de passo de alto torque, combinado com um driver adequado e um algoritmo de controle eficiente, permite o ajuste preciso da abertura da válvula, contribuindo significativamente para a eficiência e confiabilidade do sistema como um todo.

9 Conclusão

O desenvolvimento deste sistema de controle remoto de abertura de válvula industrial e monitoramento de pressão demonstrou a viabilidade e as vantagens da aplicação de tecnologias modernas em processos industriais.

A implementação do supervisório online utilizando a plataforma Adafruit IO proporcionou uma interface intuitiva e acessível para o monitoramento e controle remoto do sistema, contribuindo para a tendência da Indústria 4.0 de conectividade e acesso remoto a processos industriais.

Os resultados alcançados neste projeto podem servir como base para o desenvolvimento de soluções similares em outros contextos industriais, demonstrando como tecnologias acessíveis podem ser aplicadas para resolver problemas reais de automação industrial.

9.1 Resultados Alcançados

O projeto alcançou com sucesso todos os objetivos propostos, resultando em um sistema completo e pronto para a integração de todas as suas funcionalidades. Os principais resultados alcançados foram:

- Implementação de um sistema de controle preciso da abertura da válvula utilizando motor de passo
- Desenvolvimento de um sistema de monitoramento de vazão e pressão antes e depois da bomba
- Integração do controle do inversor de frequência para acionamento da bomba
- Implementação de interface local através de display OLED
- Desenvolvimento de supervisório online utilizando a plataforma Adafruit IO
- Migração bem-sucedida da arquitetura original (Arduino Mega + ESP8266) para uma solução integrada com ESP32

9.2 Contribuições do Projeto

Este projeto contribui para a área de automação industrial de diversas formas:

- Demonstração de como tecnologias acessíveis e de baixo custo, como o ESP32, podem ser utilizadas para desenvolver soluções robustas para automação industrial

- Integração bem-sucedida de diferentes componentes (motor de passo, sensores, inversor de frequência, display OLED, plataforma IoT)
- Implementação de supervisório online demonstrando as possibilidades de monitoramento e controle remoto de processos industriais
- Metodologia de desenvolvimento que pode servir como referência para projetos similares

9.3 Limitações e Desafios

Durante o desenvolvimento do projeto, foram encontrados diversos desafios e identificadas algumas limitações:

- Limitações da plataforma Adafruit IO na versão gratuita (taxa de dados limitada)
- Desafios na calibração dos sensores de pressão
- Integração mecânica do motor de passo à válvula industrial
- Garantia de robustez da conexão WiFi em ambientes industriais

9.4 Trabalhos Futuros

Com base nos resultados alcançados e nas limitações identificadas, são sugeridos os seguintes trabalhos futuros:

- Implementação de controle PID para regular automaticamente a abertura da válvula com base na pressão desejada
- Expansão do sistema para controlar múltiplas válvulas em um processo industrial mais complexo
- Implementação de algoritmos de detecção de falhas na válvula ou na bomba
- Migração para plataformas IoT mais robustas e voltadas para aplicações industriais

Referências

ACOSTA, Gabriel; TODOROVICH, Elías; VÁZQUEZ, Martín. Sistemas de Control para Válvulas Industriales: Una Revisión. *Revista Iberoamericana de Automática e Informática Industrial*, v. 9, n. 1, p. 24–36, 2012. Citado 1 vez na página 13.

ADAFRUIT INDUSTRIES. *Adafruit IO Documentation*. 2023. Disponível em: https://io.adafruit.com/api/docs/. Citado 1 vez na página 42.

ADAFRUIT INDUSTRIES. *SSD1306 Datasheet*. New York, 2014. Disponível em: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf. Citado 1 vez na página 39.

ALLIED MARKET RESEARCH. *Industrial Valves Market by Material Type, Valve Type, and Industry: Global Opportunity Analysis and Industry Forecast, 2020-2027*. Portland, 2020. Citado 1 vez na página 12.

ARMAZÉM DA ELÉTRICA. *Driver para Motor de Passo 4A WD-2404 Wotiom*. 2025. https://www.armazemdaeletrica.com.br/driver-para-motor-de-passo-4a-wd-2404-wotiom. Acessado em 11 de julho de 2025. Citado 0 vez na página 50.

ASTA INDÚSTRIA E COMÉRCIO DE INSTRUMENTAÇÃO E CONTROLE LTDA. *Instrumentação & Controle: Catálogo Geral de Produtos*. São Paulo, SP, 2008. Edição: OUT/2008. Citado 0 vez na página 25.

ASTA INSTRUMENTAÇÃO CONTROLE. *Transmissor de Pressão TP 110*. 2021. https://www.astainstrumentos.com.br/produtos/transmissores-de-pressao/tp110/. Citado 1 vez na página 25.

BAÚ DA ELETRÔNICA. *Arduino Mega 2560 com WiFi ESP8266 Integrado*. 2025. https://www.baudaeletronica.com.br/produto/arduino-mega-2560-com-wifi-esp8266-integrado.html. Acessado em 11 de julho de 2025. Citado 0 vez na página 19.

BOJORGE, Neilo; ALVES, Gutemberg; DREUX, Marcia. Controle de Processos Industriais. *Revista de Engenharia Química e Biotecnologia*, v. 12, n. 3, p. 45–62, 2017. Citado 1 vez na página 12.

CARDOSO, Rafael Batista. *Automação e levantamento da característica de vazão de uma válvula globo*. 2021. Monografia de Graduação – Universidade Federal de Uberlândia, Uberlândia, MG. Citado 4 vezes nas páginas 15, 16, 48, 50.

DAYLER. *Medidores de Vazão*. 2025. https://dayler.com/produtos/medidores-de-vazao/. Acessado em: 11 de julho de 2025. Citado 0 vez na página 30.

DRISKELL, Leslie R. *Control Valve Selection and Sizing*. Research Triangle Park: ISA, 1983. Citado 1 vez na página 13.

EMERSON AUTOMATION SOLUTIONS. *Control Valve Handbook*. 5. ed. Marshalltown, 2020. Citado 2 vezes nas páginas 12, 13.

ESPRESSIF SYSTEMS. *ESP32 Technical Reference Manual*. Shanghai, 2021. Citado 2 vezes nas páginas 13, 21.

GUPTA, S.; KUMAR, A.; SINGH, S. A Comparative Study of ESP32 and ESP8266 Microcontrollers for IoT Applications. *International Journal of Engineering Research Technology (IJERT)*, v. 8, n. 09, p. 101–105, 2019. Citado 1 vez na página 20.

HONEYWELL SENSING AND PRODUCTIVITY SOLUTIONS. *Pressure Sensors: Basic Guide*. Charlotte, 2018. Citado 1 vez na página 25.

INTERNATIONAL SOCIETY OF AUTOMATION. *Safety Instrumented Systems: A Life-Cycle Approach*. Research Triangle Park: ISA, 2016. Citado 1 vez na página 14.

MATHIAS, Artur C.; RAMOS, Carlos L.; BORTONI, Edson C. Análise de Desempenho de Válvulas de Controle em Sistemas Industriais. *Revista Brasileira de Automação Industrial*, v. 8, n. 2, p. 78–95, 2014. Citado 1 vez na página 12.

OGATA, Katsuhiko. *Engenharia de Controle Moderno*. 5. ed. São Paulo: Pearson, 2010. Citado 1 vez na página 12.

PATEL, D.; SHAH, R.; MEHTA, S. Performance Analysis of ESP32 and Arduino for Real-Time IoT Applications. *Journal of Engineering Research and Applications*, v. 10, n. 5, p. 1–6, 2020. Citado 1 vez na página 20.

RIBEIRO, Marco A.; SILVA, Marcos G. Válvulas de Controle & Curvas Características. *Revista InTech Brasil*, v. 2, n. 14, p. 52–68, 1999. Citado 1 vez na página 13.

ROBOCORE. *ESP32 WiFi Bluetooth*. 2025. https://www.robocore.net/wifi/esp32-wifi-bluetooth. Acessado em 11 de julho de 2025. Citado 0 vez na página 22.

SIEMENS. *SITRANS FM MAG 6000 electromagnetic flow transmitter*. 2021. https://www.siemens.com/us/en/products/automation/process-instrumentation/flow-measurement/electromagnetic/sitrans-f-m-mag-6000.html. Citado 1 vez na página 30.

SILVA, João; SANTOS, Maria. Desenvolvimento de Bancada Experimental para Controle de Processos Industriais. *Anais do Congresso Brasileiro de Automação*, p. 123–130, 2019. Citado 1 vez na página 14.

SINES, Jeffrey H.; AKHTAR, Mahmood. *Energy Efficiency in Industrial Valve Applications*. Washington, DC, 2018. Citado 1 vez na página 13.

SOLDAFRIA COMPONENTES ELETRÔNICOS. *Motor de Passo NEMA 23 - 24 Kgf.cm - 2A*. 2025. https://www.soldafria.com.br/motor-de-passo-nema-23-24-kgf-cm-2a. Acessado em 11 de julho de 2025. Citado 0 vez na página 49.

WEG S.A. *CFW10 - Inversor de Frequência: Manual do Usuário*. Jaraguá do Sul, 2020. Citado 1 vez na página 34.

WHITE, Frank M. *Mecânica dos Fluidos*. 8. ed. Porto Alegre: AMGH, 2018. Citado 1 vez na página 13.

APÊNDICE A – Código Fonte Completo

O código fonte completo do sistema de controle remoto de abertura de válvula industrial e monitoramento de pressão está disponível abaixo, ilustrando as principais funcionalidades implementadas:

```
#include <Arduino.h>
#include <WiFi.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include <AccelStepper.h>
#include "AdafruitIO_WiFi.h"

// Configurações do WiFi
#define WIFI_SSID ""
#define WIFI_PASS ""

// Configurações do Adafruit IO
#define IO_USERNAME "Codz"
#define IO_KEY "aio_FNcp28QhSjM7WM17O3dJ1wNlq6yN"

// Configurações do display OLED
#define SCREEN_WIDTH 128  // Largura do display OLED em pixels
#define SCREEN_HEIGHT 32  // Altura do display OLED em pixels
#define OLED_RESET -1     // Pino de reset (ou -1 se compartilhar o pino de reset d
#define OLED_SDA 21       // Pino SDA para ESP32
#define OLED_SCL 22       // Pino SCL para ESP32

// Configurações do motor de passo
#define STEP_PIN 12       // Pino STEP para o driver do motor
#define DIR_PIN 14        // Pino DIR para o driver do motor
#define LIMIT_SWITCH 27   // Pino do sensor de fim de curso

// Configurações dos sensores (Pinos ADC do ESP32)
```

```
// Assumindo que os circuitos de condicionamento de sinal 4-20mA para tensão estão conec
#define FLOW_SENSOR_PIN 34      // Pino para o sensor de vazão MAG600 (ADC1_CH6)
#define PRESSURE1_SENSOR_PIN 36 // Pino para o sensor de pressão 1 (pré-válvula) ASTA TE
#define PRESSURE2_SENSOR_PIN 39 // Pino para o sensor de pressão 2 (pós-válvula) ASTA TE

// Configuração do inversor de frequência
#define INVERTER_START_PIN 26  // Pino para controle START/STOP do inversor
#define INVERTER_READY_PIN 25  // Pino para monitorar o sinal READY do inversor

// Configurações de timeout e modo offline
#define WIFI_TIMEOUT_MS 10000  // Timeout para conexão WiFi (10 segundos)
#define IO_TIMEOUT_MS 8000     // Timeout para conexão Adafruit IO (8 segundos)
#define ADAFRUIT_SEND_INTERVAL 3000 // Intervalo para envio de dados ao Adafruit IO (3 s
#define WIFI_CONNECT_ATTEMPTS 3     // Número de tentativas de conexão WiFi

// Inicialização de objetos
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
AccelStepper stepper(1, STEP_PIN, DIR_PIN); // 1 = Driver interface
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);

// Configuração dos feeds do Adafruit IO
AdafruitIO_Feed *valvePositionFeed = io.feed("valve-position");
AdafruitIO_Feed *pumpControlFeed = io.feed("pump-control");
AdafruitIO_Feed *pressure1Feed = io.feed("pressure1"); // Feed para o sensor de pressão
AdafruitIO_Feed *pressure2Feed = io.feed("pressure2"); // Feed para o sensor de pressão
AdafruitIO_Feed *flowFeed = io.feed("flow");           // Feed para o sensor de vazão

// Variáveis globais
int currentPosition = 0;
int previousPosition = 0;
int valvePercentage = 0;
bool pumpStatus = false;
float pressure1 = 0.0;  // Variável para o sensor de pressão 1
float pressure2 = 0.0;  // Variável para o sensor de pressão 2
float flowRate = 0.0;   // Variável para o sensor de vazão
bool inverterReady = false;  // Estado do sinal READY do inversor
bool inverterConnected = false; // Flag para verificar se o inversor está conectado
bool wifiConnected = false;  // Flag para verificar se o WiFi está conectado
bool ioConnected = false;    // Flag para verificar se o Adafruit IO está conectado
```

```
unsigned long lastUpdateTime = 0;
unsigned long lastAdafruitSendTime = 0; // Último momento em que dados foram enviad
const int updateInterval = 5000; // Intervalo de atualização em ms

// Constantes de calibração para sensores 4-20mA
// Assumindo resistor shunt de 150 Ohm, resultando em 0.6V para 4mA e 3.0V para 20m
// ADC do ESP32 é de 12 bits (0-4095) para 0-3.3V
const float ADC_MAX_VALUE = 4095.0;
const float ADC_VOLTAGE_MAX = 3.3;
const float R_SHUNT = 150.0; // Ohms

// Faixas dos sensores (ajuste conforme a especificação real dos seus sensores)
// Exemplo: Sensor de Pressão ASTA TP110: -1 a 1000 bar (convertido para kPa)
// 1 bar = 100 kPa, então -100 kPa a 100000 kPa
const float PRESSURE_MIN_BAR = -1.0; // bar
const float PRESSURE_MAX_BAR = 1000.0; // bar
const float PRESSURE_MIN_KPA = PRESSURE_MIN_BAR * 100.0; // kPa
const float PRESSURE_MAX_KPA = PRESSURE_MAX_BAR * 100.0; // kPa

// Exemplo: Sensor de Vazão MAG600 (ajuste a faixa de vazão conforme seu modelo)
const float FLOW_MIN_LPM = 0.0;   // Litros por minuto
const float FLOW_MAX_LPM = 100.0; // Litros por minuto

// Função para converter leitura ADC para valor físico (pressão ou vazão)
float convert4_20mA_to_Physical(int rawADC, float minPhysical, float maxPhysical) {
  // Converte leitura ADC para tensão (0-3.3V)
  float voltage = (rawADC / ADC_MAX_VALUE) * ADC_VOLTAGE_MAX;

  // Converte tensão para corrente (mA)
  // V = I * R_SHUNT => I = V / R_SHUNT
  float current_mA = (voltage / R_SHUNT) * 1000.0; // Multiplica por 1000 para conv

  // Mapeia a corrente (4-20mA) para a faixa física do sensor
  // Garante que a corrente esteja dentro da faixa esperada (4-20mA)
  current_mA = constrain(current_mA, 4.0, 20.0);

  float physicalValue = map(current_mA * 100, 400, 2000, minPhysical * 100, maxPhys
  return physicalValue;
}
```

```
// Configuração do ADC do ESP32
void configureADC() {
  // Configurar resolução do ADC para 12 bits (0-4095)
  analogReadResolution(12);

  // Configurar atenuação para permitir leitura de tensões até 3.3V
  analogSetAttenuation(ADC_11db);
}

// Função para controlar a posição da válvula
void handleValvePosition(AdafruitIO_Data *data) {
  valvePercentage = data->toInt();

  // Verificar se o valor está dentro do intervalo válido
  if (valvePercentage < 0 || valvePercentage > 100) {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 0);
    display.println(F("Abertura:"));
    display.setCursor(0, 15);
    display.print("VALOR INVALIDO");
    display.display();
    delay(3000);

    // Restaurar display com o valor atual
    updateDisplay();
    return;
  }

  // Calcular a nova posição
  int targetPosition = (700 * valvePercentage / 100);
  int stepsToMove = targetPosition - previousPosition;

  Serial.print("Movendo válvula para: ");
  Serial.print(valvePercentage);
  Serial.println("%");
```

```
  // Mover o motor
  moveSteps(stepsToMove);

  // Atualizar o display
  updateDisplay();
}

// Função para controlar o inversor de frequência
void handlePumpControl(AdafruitIO_Data *data) {
  pumpStatus = data->toBool();

  Serial.print("Bomba: ");
  Serial.println(pumpStatus ? "LIGADA" : "DESLIGADA");

  // Controlar o sinal START/STOP do inversor apenas se estiver conectado
  if (inverterConnected) {
    digitalWrite(INVERTER_START_PIN, pumpStatus ? HIGH : LOW);
  } else {
    Serial.println("Inversor não conectado. Comando ignorado.");
  }

  // Atualizar o display
  updateDisplay();
}

// Função para mover o motor de passo
void moveSteps(int steps) {
  Serial.print("Movendo ");
  Serial.print(steps);
  Serial.println(" passos");

  stepper.move(steps);
  stepper.runToPosition();

  previousPosition += steps;
}

// Função para atualizar o display OLED
void updateDisplay() {
```

```
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  // Mostrar a posição da válvula
  display.setCursor(0, 0);
  display.print("Valvula: ");
  display.print(valvePercentage);
  display.println("%");

  // Mostrar o status da bomba e do inversor
  display.setCursor(0, 10);
  display.print("Bomba: ");
  display.print(pumpStatus ? "ON" : "OFF");
  display.print(" ");

  // Mostrar status do inversor apenas se estiver conectado
  if (inverterConnected) {
    display.print(inverterReady ? "READY" : "WAIT");
  } else {
    display.print("N/C");  // Não conectado
  }

  // Mostrar as pressões e vazão
  display.setCursor(0, 20);
  display.print("P1:");
  display.print(pressure1, 1); // 1 casa decimal
  display.print(" P2:");
  display.print(pressure2, 1); // 1 casa decimal
  display.print(" F:");
  display.print(flowRate, 1);  // 1 casa decimal

  display.display();
}

// Função para ler os sensores
void readSensors() {
  // Ler pressão 1 (pré-válvula)
  int pressure1Raw = analogRead(PRESSURE1_SENSOR_PIN);
```

```
  pressure1 = convert4_20mA_to_Physical(pressure1Raw, PRESSURE_MIN_KPA, PRESSURE_MA

  // Ler pressão 2 (pós-válvula)
  int pressure2Raw = analogRead(PRESSURE2_SENSOR_PIN);
  pressure2 = convert4_20mA_to_Physical(pressure2Raw, PRESSURE_MIN_KPA, PRESSURE_MA

  // Ler vazão
  int flowRaw = analogRead(FLOW_SENSOR_PIN);
  flowRate = convert4_20mA_to_Physical(flowRaw, FLOW_MIN_LPM, FLOW_MAX_LPM);

  // Ler estado do sinal READY do inversor apenas se estiver conectado
  if (inverterConnected) {
    inverterReady = digitalRead(INVERTER_READY_PIN);
  }

  // Enviar dados para o Adafruit IO apenas se estiver conectado e respeitando o in
  unsigned long currentTime = millis();
  if (ioConnected && (currentTime - lastAdafruitSendTime >= ADAFRUIT_SEND_INTERVAL)
    pressure1Feed->save(pressure1);
    pressure2Feed->save(pressure2);
    flowFeed->save(flowRate);
    lastAdafruitSendTime = currentTime;
    Serial.println("Dados enviados para o Adafruit IO");
  }

  Serial.print("Pressão 1 Pré: ");
  Serial.print(pressure1, 1);
  Serial.print(" kPa, Pressão 2 Pós: ");
  Serial.print(pressure2, 1);
  Serial.print(" kPa, Vazão: ");
  Serial.print(flowRate, 1);

  if (inverterConnected) {
    Serial.print(" L/min, Inversor: ");
    Serial.println(inverterReady ? "READY" : "NOT READY");
  } else {
    Serial.println(" L/min, Inversor: NÃO CONECTADO");
  }
}
```

```
// Função para fechar a válvula completamente
void closeValve() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println(F("INICIALIZANDO"));
  display.setCursor(0, 15);
  display.println(F("Fechando a valvula"));
  display.display();

  // Fechar a válvula até atingir o fim de curso
  while (digitalRead(LIMIT_SWITCH) == HIGH) {
    moveSteps(-25);
    if (digitalRead(LIMIT_SWITCH) == LOW) {
      Serial.println("Válvula fechada");
      break;
    }
  }

  // Resetar a posição
  previousPosition = 0;
  valvePercentage = 0;

  // Atualizar o display
  updateDisplay();

  // Enviar posição atual para o Adafruit IO apenas se estiver conectado
  if (ioConnected) {
    valvePositionFeed->save(valvePercentage);
  }
}

// Função para verificar a conexão com o inversor
void checkInverterConnection() {
  // Ler o estado inicial do pino READY
  int initialState = digitalRead(INVERTER_READY_PIN);
```

```
  // Alternar o estado do pino START para verificar se há resposta no pino READY
  digitalWrite(INVERTER_START_PIN, HIGH);
  delay(100);
  int stateAfterHigh = digitalRead(INVERTER_READY_PIN);

  digitalWrite(INVERTER_START_PIN, LOW);
  delay(100);
  int stateAfterLow = digitalRead(INVERTER_READY_PIN);

  // Se houver mudança no estado do pino READY, o inversor está conectado
  if (initialState != stateAfterHigh || stateAfterHigh != stateAfterLow) {
    inverterConnected = true;
    Serial.println("Inversor detectado e conectado!");
  } else {
    inverterConnected = false;
    Serial.println("Inversor não detectado. Operando sem controle de bomba.");
  }
}

// Função para conectar ao WiFi com otimização para conexão mais rápida
bool connectToWiFi() {
  Serial.print("Conectando ao WiFi");
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println(F("Conectando ao WiFi"));
  display.setCursor(0, 10);
  display.print(F("SSID: "));
  display.println(WIFI_SSID);
  display.setCursor(0, 20);
  display.println(F("Aguarde..."));
  display.display();

  // Configurações para conexão mais rápida
  WiFi.mode(WIFI_STA);
  WiFi.setAutoReconnect(true);
  WiFi.persistent(true);
```

```
// Desconectar de qualquer conexão anterior
WiFi.disconnect();
delay(100);

// Múltiplas tentativas de conexão
for (int attempt = 0; attempt < WIFI_CONNECT_ATTEMPTS; attempt++) {
  Serial.printf("Tentativa %d de %d\n", attempt + 1, WIFI_CONNECT_ATTEMPTS);

  WiFi.begin(WIFI_SSID, WIFI_PASS);

  unsigned long startAttemptTime = millis();

  // Tentar conectar com timeout
  while (WiFi.status() != WL_CONNECTED &&
         millis() - startAttemptTime < WIFI_TIMEOUT_MS) {
    Serial.print(".");
    display.print(".");
    display.display();
    delay(100); // Reduzido para 100ms para verificar mais frequentemente
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println();
    Serial.println("WiFi conectado!");
    Serial.print("Endereço IP: ");
    Serial.println(WiFi.localIP());

    display.clearDisplay();
    display.setCursor(0, 0);
    display.println(F("WiFi conectado!"));
    display.setCursor(0, 10);
    display.print(F("IP: "));
    display.println(WiFi.localIP());
    display.display();
    delay(500); // Reduzido para 500ms

    return true;
  }
```

```
    // Se não conectou, espera um pouco antes da próxima tentativa
    if (attempt < WIFI_CONNECT_ATTEMPTS - 1) {
      Serial.println("\nFalha na tentativa. Tentando novamente...");
      delay(500);
    }
  }

  Serial.println();
  Serial.println("Falha na conexão WiFi após múltiplas tentativas!");

  display.clearDisplay();
  display.setCursor(0, 0);
  display.println(F("Falha na conexão WiFi"));
  display.setCursor(0, 10);
  display.println(F("Operando offline"));
  display.display();
  delay(1000); // Reduzido para 1000ms

  return false;
}

// Função para conectar ao Adafruit IO com timeout
bool connectToAdafruitIO() {
  if (!wifiConnected) {
    return false;
  }

  Serial.print("Conectando ao Adafruit IO");
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println(F("Conectando ao"));
  display.setCursor(0, 10);
  display.println(F("Adafruit IO..."));
  display.display();

  io.connect();

  unsigned long startAttemptTime = millis();
```

```
// Tentar conectar com timeout
while (io.status() < AIO_CONNECTED &&
       millis() - startAttemptTime < IO_TIMEOUT_MS) {
  Serial.print(".");
  display.print(".");
  display.display();
  delay(100); // Reduzido para 100ms para verificar mais frequentemente
}

if (io.status() == AIO_CONNECTED) {
  Serial.println();
  Serial.println("Conectado ao Adafruit IO!");

  display.clearDisplay();
  display.setCursor(0, 0);
  display.println(F("Adafruit IO"));
  display.setCursor(0, 10);
  display.println(F("Conectado!"));
  display.display();
  delay(500); // Reduzido para 500ms

  // Configurar handlers para os feeds
  valvePositionFeed->onMessage(handleValvePosition);
  pumpControlFeed->onMessage(handlePumpControl);

  // Inscrever-se nos feeds
  valvePositionFeed->get();
  pumpControlFeed->get();

  return true;
} else {
  Serial.println();
  Serial.println("Falha na conexão com Adafruit IO!");

  display.clearDisplay();
  display.setCursor(0, 0);
  display.println(F("Falha na conexão"));
  display.setCursor(0, 10);
  display.println(F("com Adafruit IO"));
```

```
    display.setCursor(0, 20);
    display.println(F("Operando offline"));
    display.display();
    delay(1000); // Reduzido para 1000ms

    return false;
  }
}

void setup() {
  // Inicializar comunicação serial
  Serial.begin(115200);
  Serial.println("Inicializando sistema de controle de válvula...");

  // Configurar pinos
  pinMode(LIMIT_SWITCH, INPUT_PULLUP);
  pinMode(INVERTER_START_PIN, OUTPUT);
  pinMode(INVERTER_READY_PIN, INPUT);
  digitalWrite(INVERTER_START_PIN, LOW); // Iniciar com o inversor desligado

  // Configurar ADC
  configureADC();

  // Inicializar o display OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("Falha na inicialização do SSD1306"));
    while (true);
  }

  // Configurar o motor de passo
  stepper.setMaxSpeed(500);
  stepper.setAcceleration(300);

  // Mensagem inicial no display
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
```

```
  display.println(F("Iniciando sistema"));
  display.setCursor(0, 10);
  display.println(F("Conectando WiFi..."));
  display.display();

  // PRIMEIRO: Conectar ao WiFi com otimização para conexão mais rápida
  wifiConnected = connectToWiFi();

  // SEGUNDO: Conectar ao Adafruit IO apenas se WiFi estiver conectado
  if (wifiConnected) {
    ioConnected = connectToAdafruitIO();
  }

  // TERCEIRO: Verificar conexão com o inversor
  checkInverterConnection();

  // QUARTO: Fechar a válvula completamente na inicialização (apenas após tentar conexão
  closeValve();

  Serial.println("Sistema inicializado e pronto!");

  // Exibir status final
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println(F("Sistema Pronto!"));
  display.setCursor(0, 10);
  display.print(F("WiFi: "));
  display.println(wifiConnected ? "Conectado" : "Offline");
  display.setCursor(0, 20);
  display.print(F("Inversor: "));
  display.println(inverterConnected ? "Conectado" : "N/C");
  display.display();
  delay(1000); // Reduzido para 1000ms

  // Atualizar display com informações iniciais
  updateDisplay();
}

void loop() {
```

```
// Manter a conexão com o Adafruit IO apenas se estiver conectado
if (ioConnected) {
  io.run();
} else if (wifiConnected) {
  // Tentar reconectar ao Adafruit IO periodicamente
  unsigned long currentTime = millis();
  static unsigned long lastReconnectAttempt = 0;
  if (currentTime - lastReconnectAttempt > 60000) { // Tentar a cada 1 minuto
    lastReconnectAttempt = currentTime;
    ioConnected = connectToAdafruitIO();
  }
} else {
  // Tentar reconectar ao WiFi periodicamente
  unsigned long currentTime = millis();
  static unsigned long lastReconnectAttempt = 0;
  if (currentTime - lastReconnectAttempt > 60000) { // Tentar a cada 1 minuto
    lastReconnectAttempt = currentTime;
    wifiConnected = connectToWiFi();
    if (wifiConnected) {
      ioConnected = connectToAdafruitIO();
    }
  }
}

// Atualizar leituras dos sensores a cada intervalo
unsigned long currentTime = millis();
if (currentTime - lastUpdateTime >= updateInterval) {
  readSensors();
  updateDisplay();
  lastUpdateTime = currentTime;
}

// Verificar comandos da serial (para debug)
if (Serial.available() > 0) {
  Serial.println("Entre com a abertura da válvula em %");
  int inputPercentage = Serial.parseInt();

  if (inputPercentage >= 0 && inputPercentage <= 100) {
    valvePercentage = inputPercentage;
```

```
      // Calcular a nova posição
      int targetPosition = (700 * valvePercentage / 100);
      int stepsToMove = targetPosition - previousPosition;

      // Mover o motor
      moveSteps(stepsToMove);

      // Atualizar o display
      updateDisplay();

      // Enviar para o Adafruit IO apenas se estiver conectado
      if (ioConnected) {
        valvePositionFeed->save(valvePercentage);
      }
    } else {
      Serial.println("Valor inválido! Use um número entre 0 e 100.");
    }

    // Limpar o buffer serial
    while (Serial.available() > 0) {
      Serial.read();
    }
  }
}
```

ANEXO A – Datasheet dos Componentes

Os datasheets dos principais componentes utilizados neste projeto estão disponíveis nos links abaixo:

- ESP32: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- Motor de passo Wotiom Nema WS23-24kgf: Disponível com o fabricante
- Driver WD-2404: Disponível com o fabricante
- Display OLED SSD1306: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
- Inversor de frequência Weg CFW10 Easydrive: https://static.weg.net/medias/downloadcenter/h6e/h1f/WEG-cfw-10-manual-do-usuario-0899.5832-3.1x-manual-port pdf