UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA

RODRIGO SANTANA SOARES

KIT DIDÁTICO PARA AS DISCIPLINAS DE INSTRUMENTAÇÃO INDUSTRIAL I E II

Uberlândia - MG
2024

RODRIGO SANTANA SOARES

KIT DIDÁTICO PARA AS DISCIPLINAS DE INSTRUMENTAÇÃO INDUSTRIAL I E II

Projeto de Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Câmpus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica – FEELT

Orientador: Prof. Dr. Josué Silva de Morais

Uberlândia - MG

2024

Soares, Rodrigo Santana

Kit Didático Para as Disciplinas de Instrumentação Industrial I e II/ **Rodrigo Santana Soares**. – **UBERLÂNDIA**, **2023**- 102 p. : il. (algumas color) ; 30 cm.

Orientador: Prof. Dr. Josué Silva de Morais

Trabalho de Conclusão de Curso – Universidade Federal de Uberlândia – UFU Faculdade de Engenharia Elétrica. **2024**.
Inclui bibliografia

1. Instrumentação Industrial. 2. ESP32. 3. KIT Didático. I. Josué Silva de Morais. II. Universidade Federal de Uberlândia. III. Faculdade de Engenharia Elétrica. IV. Engenharia de Controle e Automação.

RODRIGO SANTANA SOARES

KIT DIDÁTICO PARA AS DISCIPLINAS DE INSTRUMENTAÇÃO INDUSTRIAL I E II

Projeto de Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Câmpus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Uberlândia (MG), 02 de maio de 2024.

Banca Examinadora:

Prof. Dr. Josué Silva de Morais

Orientador

Prof. Dr. Fábio Vincenzi Romualdo da Silva

Membro Avaliador

Mestre Luís Ricardo Cândido Côrtes

Membro Avaliador

Dedico este trabalho aos meus amados irmãos, Cláudio Soares da Silva Junior e Hygor Santana Soares, cujo apoio incansável e constante incentivo foram fontes de inspiração e força ao longo de toda minha jornada acadêmica.

AGRADECIMENTOS

Gostaria de expressar minha mais sincera gratidão a todas as pessoas que contribuíram na minha graduação e para a realização deste trabalho.

Primeiramente, dedico este momento à minha família, cujo amor incondicional e apoio foram fundamentais para alcançar este objetivo. Aos meus irmãos Cláudio Junior e Hygor Santana, aos meus pais Neida Raquel de Santana e Cláudio Soares da Silva, aos meus avós Maria Raquel, Sebastião Ferreira, Luzia Rosa e Sinair da Silva, também à minha madrasta Daniela e meu padrasto Augusto, além do meu tio Marcelo, tia Renata e primo Eduardo. O apoio de cada um de vocês foi essencial para meu crescimento pessoal e para a conquista deste importante marco em minha vida. Obrigado por estarem sempre ao meu lado e por serem parte tão significativa da minha trajetória.

Expresso meu sincero apreço ao meu estimado professor orientador, Josué Silva de Morais, pela orientação contínua, precisão e valiosas contribuições que enriqueceram este trabalho. Sua sabedoria e dedicação foram indispensáveis para seu desenvolvimento.

Aos demais professores do curso de Engenharia de Controle e Automação, manifesto minha gratidão pelas oportunidades de aprendizado e inspiração que proporcionaram, em especial ao professor Fábio Vincenzi Romualdo da Silva, cujo apoio foi essencial.

Não posso deixar de expressar minha profunda gratidão aos amigos que fiz durante a graduação, cujo apoio e encorajamento foram cruciais durante esta jornada. Alexandre Miranda, Giovana Rodrigues, Guilherme Almeida, Lucas Araújo, Pedro Cunha, Pedro Tizzo, Thiago Fernando e Vinícius Medeiros, vocês foram grandes amigos, e sua companhia e amizade foram fundamentais.

Por fim, agradeço a Deus por permitir que ultrapassasse os obstáculos encontrados ao longo deste trabalho. Sua presença e orientação foram fundamentais, iluminando meu caminho e fortalecendo minha fé. A todos que contribuíram de alguma forma, meu mais profundo agradecimento.

“Forget your lust for the rich man's gold, all that you need is in your soul.”

(Lynyrd Skynyrd, 1973)

RESUMO

Este projeto visa atender à necessidade de proporcionar aos estudantes de Engenharia de Controle e Automação uma experiência prática alinhada aos desafios reais da Instrumentação Industrial. O KIT didático, baseado no microcontrolador ESP32, oferece uma solução acessível e versátil para aprimorar as habilidades práticas dos alunos. A abordagem de custo reduzido busca democratizar o acesso a tecnologias de automação, enquanto a conectividade sem fio confere flexibilidade aos experimentos. O desenvolvimento de firmware específico para o ESP32 é central, garantindo o pleno funcionamento das funcionalidades propostas. O KIT, voltado para as disciplinas de Instrumentação Industrial I e II, promete ser economicamente viável e integrar-se de maneira portátil em diversas práticas educacionais, enriquecendo a formação dos estudantes.

Palavras-chave: Conectividade Sem Fio, Democratização Tecnológica, Engenharia de Controle e Automação, ESP32, Firmware Específico, Instrumentação Industrial, KIT Didático.

ABSTRACT

This project aims to meet the need to provide practical experience aligned with real challenges in Industrial Instrumentation for Control and Automation Engineering students. The didactic kit, based on the ESP32 microcontroller, offers an affordable and versatile solution to enhance students' practical skills. The cost-effective approach seeks to democratize access to automation technologies, while wireless connectivity provides flexibility for experiments. The development of specific firmware for the ESP32 is crucial to ensure the full functionality of the proposed features. The kit, designed for Industrial Instrumentation courses, promises to be economically viable and seamlessly integrate into various educational practices, enriching students' training.

Keywords: Wireless Connectivity, Technological Democratization, Control and Automation Engineering, ESP32, Specific Firmware, Industrial Instrumentation, Didactic KIT.

LISTA DE ILUSTRAÇÕES

Figura 1- Mapeamento do ESP32 utilizado no KIT. 23
Figura 2 - ESP32-DevKitC V4 com módulo ESP32-WROOM-32 soldado. 24
Figura 3 – Diagrama de blocos do ESP32. 28
Figura 4 - Sinal Hart sobreposto ao sinal 4-20 mA. 29
Figura 5 - Conexão de uma entrada a um instrumento HART. 30
Figura 6 - Conexão de uma saída HART. 31
Figura 7 - O Protocolo HART permite que dois dispositivos Mestre acessem informações de um mesmo equipamento de campo (escravo). 31
Figura 8 - Exemplo de aplicação MQTT. 35
Figura 9 - Exemplo de Fluxo Node-Red para aquisição de dados. 36
Figura 10 - Protótipo do painel do KIT didático. 38
Figura 11 - Fonte Hilink HLK-10M12. 39
Figura 12 - Conversor de corrente em Tensão HW-685. 40
Figura 13 - Gerador de sinal ajustável. 42
Figura 14 - Diagrama de conexão do módulo HW-685. 42
Figura 15 - Tensão de saída do módulo para 4mA. 43
Figura 16 - Tensão de saída do módulo para 20mA. 43
Figura 17 - Conversor de Tensão em Corrente. 44
Figura 18 - Amplificador operacional com feedback negativo. 46
Figura 19 - Diagrama de conexão do módulo conversor de tensão em corrente. 47
Figura 20 - Corrente de saída do módulo para 0V. 48
Figura 21 - Corrente de saída do módulo para 3,3V. 48
Figura 22 - Módulo Amplificador Operacional LM358. 49
Figura 23 - Diagrama de conexão do módulo amplificador LM358. 50
Figura 24 - Módulo MOSFET PWM de Alta Potência. 51
Figura 25 - Circuito do MOSFET PWM de Alta Potência. 52
Figura 26 - Diagrama de conexão do módulo MOSFET PWM. 53
Figura 27 – Exemplo de variação do Duty Cycle. 54
Figura 28 - Módulo Relé de Estado Sólido (SSR). 54
Figura 29 - Diagrama de conexão do módulo SSR. 56
Figura 30 - Módulo Conversor de Nível Lógico Digital Com Isolamento Óptico (IN:3V3 OUT:5V). 57

Figura 31 - Diagrama de Conexão do Módulo Conversor de Nível Lógico Digital Com Isolamento Óptico. 59
Figura 32 - Módulo Pré Amplificador passa-baixa Para Subwoofer..... 59
Figura 33 - Especificações do Modem HART DS8500 62
Figura 34 - Modem HART DS8500 63
Figura 35 - Um transmissor de processo inteligente apresenta o modem DS8500 HART comunicando-se com um microcontrolador de sistema. 64
Figura 36 - Forma de onda do modulador (DS8500). 65
Figura 37 - Forma de onda do demodulador (DS8500)..... 65
Figura 38 - Diagrama de conexão do Modem HART DS8500. 66
Figura 39 - Configuração WiFi. 68
Figura 40 - Conexão e Comunicação MQTT. 69
Figura 41 - Controle do Conversor de Tensão para Corrente (DAC). 71
Figura 42 - Controle do Conversor de Corrente para Tensão..... 71
Figura 43 - Controle do Módulo Amplificador de Ganho de Sinal para Geração de Funções. 73
Figura 44 - Controle do Módulo MOSFET PWM 74
Figura 45 - Controle do Módulo SSR..... 75
Figura 46 - Código teste para o modem HART DS8500. 76
Figura 47 - Arquitetura do projeto..... 77
Figura 48 - Fluxo do Node-RED para o projeto. 78
Figura 49 - UI da tela inicial do painel de monitoramento..... 79
Figura 50 - UI da tela de monitoramento do módulo conversor 4-20mA para 0-3,3V. 80
Figura 51 - UI da tela de monitoramento do módulo conversor 0-3,3V para 4-20mA. 82
Figura 52 - UI da tela de monitoramento do módulo Amplificador LM358..... 83
Figura 53 - Ondas resultantes do módulo amplificador: A)Senoidal; B)Triangular; C)Dente-de-serra; D)Quadrada. 83
Figura 54 - UI da tela de monitoramento do módulo MOSFET PWM. 84
Figura 55 - Teste do Módulo MOSFET PWM no osciloscópio com 50% de Duty Cycle. 85
Figura 56 - UI da tela de monitoramento do módulo SSR. 85
Figura 57 - Teste do Módulo Conversor de Nível Lógico Digital. 86
Figura 58 - Teste do Módulo Pré Amplificador passa-baixa..... 87
Figura 59 - Cabo Conversor USB para TTL FT232RL 6 Vias. 87
Figura 60 - Tela do PACTware: Parâmetros da porta serial..... 88
Figura 61 - UI da tela de monitoramento Modem HART. 89

LISTA DE TABELAS

Tabela 1 - Descrição Funcional da ESP32. 24
Tabela 2 - Pinos de Entrada Analógica da ESP32. 26
Tabela 3 - Lista parcial de comandos HART. 33
Tabela 4 - Configuração de intervalo de tensão no HW-685. 41
Tabela 5 - Terminal e definição dos pinos do módulo. 58
Tabela 6 - Descrição dos pinos do Modem HART DS8500 62
Tabela 7 - Possíveis configurações de frequências PWM na ESP32. 74
Tabela 8 - Resultados do conversor de corrente para tensão. 80
Tabela 9 - Resultados do conversor de tensão em corrente. 81

LISTA DE ABREVIATURAS E SIGLAS

IoT	Internet of Things
WiFi	Wireless Fidelity
PWM	Pulse-Width Modulation
SoC	System on Chip
Hz	Hertz - Unidade de medida de frequência
GPIO	General Purpose Input Output
MB	MegaByte - Unidade de medida de armazenamento de um hardware.
mA	Miliampere - Unidade de medida de corrente
V	Volts - Unidade de medida de tensão
SSR	Solid State Relay
LED	Light-Emiting Diode
AC	Corrente alternada
DC	Corrente contínua
D/A	Digital / Analógico
VCC	Tensão em corrente contínua
GND	Graduated neutral density filter
W	Watts - Unidade de medida de potência
CI	Circuito Integrado
SCADA	Supervisory Control and Data Acquisition
HART	Highway Addressable Remote Transducer
UART	Universal Asynchronous Receiver / Transmitter
UI	User Interface

SUMÁRIO

1 INTRODUÇÃO 15
1.1 Justificativas 16
1.2 Objetivos 17
1.2.1 Objetivos específicos 17
2 REFERENCIAL TEÓRICO 18
2.1 Considerações Iniciais 18
2.2 Industria 4.0 18
2.2.1 Instrumentação Industrial 19
2.3 Levantamento de Tecnologias 20
2.3.1 Microcontroladores 21
2.3.2 ESP32 22
2.3.3 Protocolo HART 28
2.3.4 Protocolo MQTT e Ambiente Node-RED 33
3 METODOLOGIA 37
3.1 Considerações Iniciais 37
3.2 Estudo de Tecnologias 37
3.3 Desenvolvimento do Kit Didático 38
3.3.1 Mini Fonte Hi-link HLK-10M12 39
3.3.2 Módulo conversor de corrente para tensão (4-20mA para 0-3,3V) 40
3.3.3 Módulo conversor de tensão para corrente (0-3,3V para 4-20mA) 44
3.3.4 Módulo amplificador de ganho de sinal LM358 49
3.3.5 Módulo MOSFET PWM de Alta Potência 51
3.3.6 Módulo Relé de Estado Sólido (SSR) 54
3.3.7 Módulo Conversor de Nível Lógico Digital Com Isolamento Óptico 57
3.3.8 Módulo Pré Amplificador passa-baixa Para Subwoofer 59
3.3.9 Modem HART DS8500 62
3.4 Desenvolvimento do Firmware 67
3.4.1 Desenvolvimento de Código no Ambiente Visual Studio 67
3.4.2 Configuração e Implementação do Node-RED 77
4 RESULTADOS E DISCUSSÕES 80
4.1 Considerações Iniciais 80
4.2 Avaliação e Análise dos Módulos Implementados 80

4.3 Avaliação e Análise do Modem HART 87
5 CONCLUSÃO 90
REFERÊNCIAS BIBLIOGRÁFICAS 91
APÊNDICE A – CÓDIGO COMPLETO DA ESP32 PARA OS MÓDULOS. 95
APÊNDICE B – CÓDIGO DA ESP32 PARA O MODEM HART DS8500. 100

1 INTRODUÇÃO

O progresso tecnológico na Instrumentação Industrial desafia constantemente os profissionais a adotarem inovações em seus métodos de trabalho. Os dispositivos de IoT têm se destacado como soluções eficazes para aprimorar a eficiência e a precisão nos sistemas industriais. No entanto, a rápida evolução tecnológica muitas vezes traz desafios relacionados à acessibilidade e viabilidade financeira. Softwares e hardwares de ponta frequentemente demandam investimentos substanciais, tornando-se um obstáculo para instituições e profissionais interessados em práticas mais avançadas. É essencial equilibrar a sofisticação tecnológica com a viabilidade financeira, buscando alternativas acessíveis que democratizem o acesso ao conhecimento em Instrumentação Industrial.

Nesse sentido, para impulsionar ainda mais o desenvolvimento da IoT e ampliar suas aplicações, são necessárias soluções poderosas, de baixo custo e baixo consumo de energia para os dispositivos IoT. Outro requisito fundamental para um dispositivo IoT é possuir um formato compacto; quanto menor o tamanho e peso do dispositivo, maior será a versatilidade de suas aplicações. Cada unidade IoT é composta por um microcontrolador e um módulo de comutação sem fio (geralmente WiFi), ou uma combinação de ambos. Já existem no mercado uma ampla variedade de módulos e microcontroladores utilizados para o design e desenvolvimento de dispositivos IoT, tais como Xbee, WhizFi, certas placas Arduino, entre outros.

Contudo, a maioria dos dispositivos atualmente disponíveis é relativamente cara e apresenta tamanho e peso consideráveis. Além disso, poucos desses módulos são de código aberto e não possuem restrições quanto à finalidade de operação. Um novo dispositivo denominado ESP32, no formato QFN48, foi lançado pela Espressif Systems em setembro de 2016 para substituir o anterior microcontrolador ESP8266. O ESP32 é um microcontrolador poderoso que integra Wi-Fi e Bluetooth, sendo projetado como uma solução ideal para dispositivos IoT (MAIER, 2017).

Preparar profissionais capacitados para atuar com excelência nas diversas áreas da Engenharia demanda cursos que ofereçam um conjunto sólido de conhecimentos. Disciplinas teóricas bem estruturadas, estágios no mercado de trabalho e aulas práticas são fundamentais para alcançar esses objetivos (BAZZO & PEREIRA, 2006).

Fomentando Competências Multidisciplinares para o Futuro - Desenvolvimento de um conjunto didático com o microcontrolador ESP32. Atualmente, conjuntos de desenvolvimento, como o Arduino UNO, fundamentados no microcontrolador ATmega328P,

são amplamente empregados na prática (SANTOS, 2023). Contudo, observa-se o surgimento de microcontroladores mais avançados, como o ESP32, que, além de apresentar vantagens econômicas e eficiência energética, incorpora funcionalidades de Wi-Fi e Bluetooth (ESPRESSIF, 2024), proporcionando espaço para inovações. Nesse contexto, propõe-se a elaboração de um conjunto didático baseado nesse microcontrolador, visando atender às demandas contemporâneas de formação técnica.

1.1 Justificativas

É fundamental destacar que, de acordo com o Mapa do Ensino Superior no Brasil do Instituto Semesp, mais da metade dos alunos que ingressam em faculdades no país acabam desistindo antes de completar o curso (CNN Brasil, 2023). Além disso, segundo uma pesquisa conduzida por Gama em 2018, uma variável significativa associada à evasão universitária foi a qualidade do corpo docente da instituição, com um coeficiente de relevância de 0,791. Essa constatação reflete na decisão dos alunos de abandonar a Universidade Federal de Uberlândia (UFU), indicando que quanto melhor for a percepção dos alunos em relação ao corpo docente, menor será a probabilidade de evasão (GAMA, 2018).

Embora as disciplinas de instrumentação industrial I e II sejam do quinto e sexto período do curso de Engenharia de Controle e Automação na Universidade Federal de Uberlândia, e o índice de evasão esteja frequentemente associado aos primeiros períodos, é crucial desenvolver novos métodos didáticos de ensino para reduzir esse coeficiente de evasão na faculdade.

Sendo assim, este projeto surge da necessidade de proporcionar aos estudantes de Engenharia de Controle e Automação uma experiência prática e didática mais alinhada com os desafios reais em Instrumentação Industrial. O KIT didático, fundamentado no ESP32, é apresentado como uma solução acessível e versátil para aprimorar as habilidades práticas dos alunos.

Adicionalmente, a abordagem de custo reduzido visa tornar o KIT acessível a instituições com recursos limitados, promovendo, assim, a democratização do acesso a tecnologias de automação. A inclusão da conectividade sem fio confere maior flexibilidade aos experimentos, enriquecendo ainda mais a formação prática dos alunos. Dessa forma, a justificativa para este projeto reside na sua capacidade de oferecer uma ferramenta didática eficaz e inovadora, alinhada às demandas do mercado, para proporcionar uma formação mais completa e qualificada aos futuros profissionais na área.

1.2 Objetivos

Este projeto visa desenvolver um KIT didático para as disciplinas de Instrumentação Industrial I e II, utilizando um ESP32 como controlador principal. O KIT oferecerá uma plataforma versátil e prática para estudantes de Engenharia de Controle e Automação, permitindo interação com diversos componentes e tecnologias relevantes para o campo. Além disso, propõe-se a implementação de um Painel de Monitoramento Remoto utilizando Node-RED, proporcionando uma experiência integrada de aprendizado e aplicação prática em automação e instrumentação industrial.

1.2.1 Objetivos específicos

- Explorar as capacidades de desenvolvimento e integração do microcontrolador ESP32, com ênfase no estudo e utilização de saídas DAC, PWM e comunicação Serial.
- Desenvolver um KIT que integre o ESP32 com os módulos e ferramentas pertinentes, possibilitando a aplicação prática dos conceitos estudados nas disciplinas de Instrumentação Industrial I e II.
- Criar firmware personalizado para o ESP32, assegurando o pleno funcionamento e a integração das funcionalidades previstas no KIT.
- Implementar um sistema no Node-RED que se comunique via Wi-Fi através do protocolo MQTT com o ESP32, permitindo o monitoramento e controle remoto das funções do KIT de forma intuitiva e acessível.

2 REFERENCIAL TEÓRICO

2.1 Considerações Iniciais

Neste capítulo, serão abordados os fundamentos essenciais para o entendimento do trabalho, juntamente com a análise de tecnologias correlatas já estabelecidas no mercado.

2.2 Industria 4.0

O conceito de Indústria 4.0, também conhecido como quarta revolução industrial, emergiu de iniciativas estratégicas do governo da Alemanha, visando solidificar o país como líder em tecnologia e reforçar sua competitividade global. Este termo foi oficialmente lançado em abril de 2013, durante a maior feira de tecnologia industrial, a "Feira de Hannover", como parte do projeto Industrie 4.0, que apresentou as primeiras recomendações para sua implementação. Kagermann descreveu a Indústria 4.0 como uma realidade na qual as empresas estabelecem redes globais por meio de Sistemas Físico Cibernéticos (CPS – Cyber-Physical Systems). Estes sistemas incorporam máquinas, sistemas de armazenagem e instalações de produção, capacitados para trocar informações e cooperar de forma autônoma através da Internet das Coisas (IoT - Internet of Things), desencadeando ações e controlando uns aos outros de maneira independente (KAGERMANN, 2013).

Além disso, a Indústria 4.0 não se limita apenas aos aspectos de produção, mas também impacta áreas como logística, manutenção, gestão de qualidade e desenvolvimento de produtos. Essa abordagem holística transforma não apenas os processos industriais, mas também os modelos de negócios, promovendo a inovação e a competitividade das organizações em um cenário global cada vez mais dinâmico e exigente. Embora muitas indústrias brasileiras já tenham automatizado seus processos, ainda não alcançamos a plena realização da manufatura digital. A Indústria 4.0 é caracterizada por duas vertentes essenciais: processos integrados que asseguram a produção personalizada e a criação de produtos inovadores. O Brasil ainda tem um longo caminho a percorrer nessas duas áreas.

Essa realidade é tangível em países como Alemanha e Estados Unidos, onde existem grandes projetos e iniciativas com participação tanto do governo quanto da iniciativa privada. Nos Estados Unidos, por exemplo, foi estabelecida a Smart Manufacturing Leadership Coalition, uma organização sem fins lucrativos dedicada a evidenciar, por meio de pesquisas, os benefícios da manufatura avançada e a facilitar sua adoção (SANTOS; MANHÃES; LIMA, 2018). Investir em inovação e educação é fundamental para reverter o atual cenário

brasileiro, inclusive para aumentar a compreensão do que envolve a digitalização. Embora já existam instituições, empresas e universidades trabalhando na área da Indústria 4.0, é preciso mais do que isso para gerar uma mão de obra qualificada e atender às demandas do mercado.

Em última análise, a revolução da Indústria 4.0 não apenas redefine os processos de produção, mas também redefine a maneira como entendemos e implementamos a instrumentação industrial. Ao abraçar os princípios da conectividade, automação inteligente e análise de dados em tempo real, a instrumentação industrial na era da Indústria 4.0 torna-se mais do que apenas uma ferramenta para medição e controle; ela se torna um componente essencial para a operação eficiente e otimizada das fábricas inteligentes.

2.2.1 Instrumentação Industrial

Com o advento da Indústria 4.0, a automação e a instrumentação industrial enfrentam novos desafios, tornando-se componentes essenciais dessa nova tendência e precisando se alinhar tecnologicamente. As tecnologias de instrumentação e controle sempre foram fundamentais para a produção industrial. No entanto, atualmente, não apenas completam o ciclo produtivo, mas também se tornam inteligentes o suficiente para alimentar os sistemas de gerenciamento de ativos. Isso significa uma transição de modernização para eficiência, garantindo não apenas a modernização dos processos, mas também a sua otimização (FONSECA, 2017).

À medida que os processos industriais se tornam cada vez mais complexos, o controle abrangente de todas as operações torna-se um fator fundamental para o sucesso de toda corporação. Nesse contexto, a instrumentação industrial e a automação emergem como elementos essenciais, permeando diversas áreas operacionais. A instrumentação industrial desempenha um papel crucial no controle de qualidade, na segurança dos trabalhadores e na otimização das operações. Seu impacto se reflete diretamente na eficiência dos processos produtivos e na qualidade do produto final, delineando os rumos da competitividade da empresa no mercado.

A instrumentação industrial é uma disciplina científica dedicada ao uso de instrumentos de medição para aprimorar o desempenho dos processos industriais. Ela se concentra em monitorar e controlar variáveis essenciais, como temperatura, tempo de produção e outros, para garantir a eficiência e a qualidade dos processos produtivos. Originada na década de 1940, a necessidade de automatizar o controle de válvulas pneumáticas impulsionou o desenvolvimento da instrumentação industrial. Ao longo dos anos, essas tecnologias evoluíram, proporcionando economia, eficiência e segurança aos

processos industriais. Atualmente, o monitoramento em tempo real dos equipamentos permite a realização de manutenção preventiva e corretiva, garantindo a segurança dos colaboradores e reduzindo custos com reposição de peças desnecessárias. Isso resulta em uma produção mais precisa e de alta qualidade, contribuindo para o sucesso das operações industriais (COUTINHO, 2021).

Sendo assim, a instrumentação industrial desempenha um papel crucial na era da Indústria 4.0, impulsionando o avanço tecnológico que permite o desenvolvimento de sistemas autônomos capazes de tomar decisões com base em dados sensoriais precisos.

Com a evolução contínua da tecnologia, a Indústria 4.0 está redefinindo os paradigmas da automação e instrumentação industrial. Destacam-se as máquinas inteligentes, dotadas de autonomia aprimorada, e a crescente importância do Big Data, aliados à integração total com tecnologias de ponta, como a computação em nuvem, que estão ampliando os limites do possível. Um exemplo notável dessa transformação é a Internet das Coisas Industrial (IIoT), que conecta equipamentos industriais em uma rede local, possibilitando a comunicação máquina a máquina (M2M) em tempo real e fornecendo um fluxo contínuo de dados dos sensores para análise e tomada de decisões.

Nesse contexto, a instrumentação industrial desempenha um papel vital. Quanto mais precisos e abrangentes forem os dados coletados, mais sofisticados podem ser os sistemas implementados. A capacidade de medir uma ampla gama de informações é fundamental para a criação de sistemas cada vez mais complexos e eficientes, uma vez que os dispositivos IIoT utilizam esses dados para otimizar as operações industriais.

2.3 Levantamento de Tecnologias

Antes de discutir sobre microcontroladores, é essencial compreender a definição de sistemas embarcados e sua relevância na sociedade moderna. Um sistema embarcado é caracterizado por integrar capacidade computacional dentro de um circuito integrado, equipamento ou sistema, sendo mais do que um simples computador. Este sistema completo e independente é projetado para realizar uma tarefa específica, sem proporcionar ao usuário final acesso ao programa embutido no dispositivo. No entanto, a interação com o equipamento é possível por meio de interfaces como teclados e displays, desde que o sistema seja projetado para tal interatividade (DA SILVA & ARAUJO, 2019).

Ao contrário de computadores que executam sistemas operacionais para suportar a instalação de diversos aplicativos, cada um destinado a uma aplicação diferente, os sistemas embarcados são concebidos para executar apenas uma tarefa predeterminada.

Frequentemente, eles carecem da flexibilidade, tanto de software quanto de hardware, para desempenhar funções diversas das originalmente projetadas e desenvolvidas. A flexibilidade permitida geralmente se resume a upgrades de novas versões, realizados pelos fabricantes, resultando em sistemas reprogramados com correções ou funcionalidades aprimoradas. Nesse contexto, um "cérebro" torna-se crucial para gerenciar o funcionamento do sistema. Um microprocessador ou microcontrolador é a escolha ideal para essa função, pois ambos possuem a capacidade de ler sinais externos, executar programas com tarefas específicas, processar os sinais e enviar os resultados esperados para os atuadores.

A programação desses sistemas é realizada através de uma codificação chamada firmware, armazenada em memória ROM ou memória FLASH, com atenção especial dada à estabilidade, uma vez que a atualização do firmware após a gravação é geralmente inviável para o usuário final.

E é cada vez mais comum o surgimento de chips que representam um sistema completo em uma única pastilha, conhecidos como SoC (Systems on Chip). Isso inclui microcontroladores que incorporam sensores (temperatura, pressão, etc.), transmissores (RF), interfaces gráficas para displays, e outras funcionalidades diretamente na pastilha. Esse avanço destaca a evolução dos microcontroladores para sistemas cada vez mais integrados e eficientes (CUNHA, 2007).

2.3.1 Microcontroladores

Em sua essência, um microcontrolador representa a convergência de um computador completo em um único chip. Este chip é composto por um processador (Unidade Lógica e Aritmética – ULA), memória, periféricos de entrada e saída, temporizadores, dispositivos de comunicação serial, entre outros elementos. Os microcontroladores surgiram como uma progressão natural dos circuitos digitais, impulsionados pelo aumento da complexidade desses sistemas. Em determinado ponto, torna-se mais eficiente, econômico e compacto substituir a lógica das portas digitais por um conjunto de processador e software.

O primeiro microcontrolador foi lançado pela empresa Intel em 1977 e recebeu a sigla “8048”. Com a sua posterior evolução, deu origem à família “8051”. Esse chip é programado em linguagem Assembly e possui um poderoso conjunto de instruções. Por ser um dos precursores, é utilizado em muitas aplicações de automação em diversas áreas do mundo (PENIDO & TRINDADE, 2006).

Dentro do contexto de sistemas embarcados, o Arduino e o ESP32 são exemplos notáveis de microcontroladores que desempenham papéis essenciais na construção de dispositivos inteligentes e projetos eletrônicos.

O Arduino, por exemplo, é conhecido por sua simplicidade e flexibilidade. Ele é amplamente utilizado em projetos de prototipagem rápida, permitindo que os desenvolvedores incorporem facilmente sensores, atuadores e outros componentes para criar dispositivos interativos. A programação do Arduino é feita através da linguagem C/C++, tornando-a acessível mesmo para aqueles que não têm uma vasta experiência em programação (DA SILVA & ARAUJO, 2019).

Já o ESP32 é uma evolução, oferecendo recursos avançados, especialmente em termos de conectividade sem fio. Com suporte para Wi-Fi e Bluetooth, o ESP32 é frequentemente escolhido em projetos que exigem comunicação sem fio, como dispositivos IoT (Internet das Coisas). Sua capacidade de se conectar à internet e interagir com outros dispositivos torna-o uma escolha popular para aplicações mais avançadas.

Ambos os microcontroladores, Arduino e ESP32, têm uma comunidade ativa de desenvolvedores e uma vasta gama de bibliotecas e recursos disponíveis. Isso facilita a implementação de projetos, desde simples automações residenciais até complexos dispositivos IoT. Além disso, a natureza de código aberto dessas plataformas incentiva a colaboração e a partilha de conhecimento, contribuindo para o constante avanço e aprimoramento das tecnologias embarcadas.

2.3.2 ESP32

O ESP32, desenvolvido pela Espressif Systems e lançado em 2016, representa uma evolução significativa em relação ao seu antecessor, o ESP8266. Este microcontrolador destaca-se pela integração de conectividade Wi-Fi e Bluetooth, além de oferecer maior poder de processamento e recursos aprimorados. Essas características fazem do ESP32 uma escolha amplamente adotada em projetos de IoT e eletrônicos.

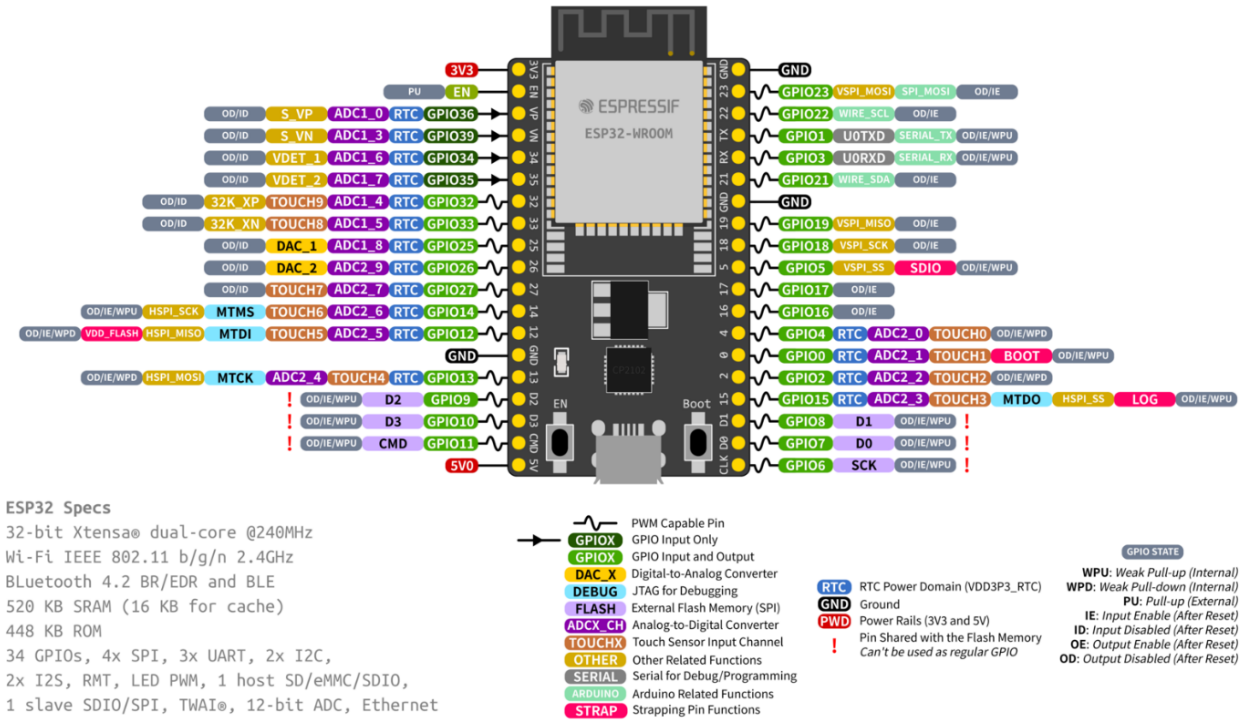
Classificado como um microcontrolador SoC (System on Chip) poderoso, o ESP32 incorpora Wi-Fi 802.11 b/g/n integrado, Bluetooth dual-mode na versão 4.2 e uma ampla gama de periféricos (ESPRESSIF, 2024). Destaca-se como sucessor avançado do chip ESP8266, especialmente devido à implementação de dois núcleos com clock em diferentes versões, podendo atingir até 240 MHz.

A missão Apollo 11, que levou o homem à lua, utilizou o computador de bordo AGC (Apollo Guidance Computer) com uma velocidade de processamento de apenas 2MHz. Em

comparação, como falamos anteriormente, a ESP32 possui uma velocidade de processamento de 240MHZ, mais de cem vezes a velocidade do AGC. Essa diferença ressalta não apenas a evolução da tecnologia desde então, mas também a incrível capacidade de processamento que temos à disposição atualmente.

Além dos recursos mencionados, o ESP32 expande significativamente o número de pinos GPIO de 17 para 36, os canais PWM para 16, e é equipado com generosos 4 MB de memória flash. A Figura 1 a seguir apresenta o mapeamento do microcontrolador utilizado do projeto.

Figura 1- Mapeamento do ESP32 utilizado no KIT.

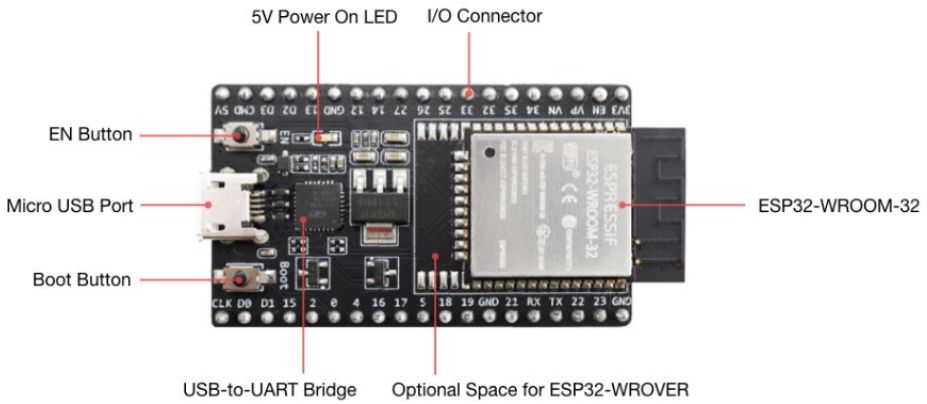


Fonte: Adaptado de ESPRESSIF, 2024.

Desde seu lançamento, o ESP32 conquistou grande popularidade na comunidade de desenvolvedores, graças à sua versatilidade e aos recursos avançados oferecidos. Seu uso comum abrange desde sistemas de monitoramento remoto até dispositivos de automação residencial, destacando-se como uma escolha robusta para uma variedade de aplicações eletrônicas. (BABIUCH, 2019).

A Figura 2 abaixo e a Tabela 1 subsequente apresentam os principais componentes, interfaces e controles da placa ESP32-DevKitC V4, a qual foi selecionada como a placa base para a implementação deste projeto.

Figura 2 - ESP32-DevKitC V4 com módulo ESP32-WROOM-32 soldado.



Fonte: Adaptado de ESPRESSIF, 2024.

Tabela 1 - Descrição Funcional da ESP32.

Componente	Descrição
ESP32-WROOM-32	Um módulo com ESP32 em seu núcleo
EN	Botão de reset
Boot	Botão de download. Manter pressionado Boot e pressionar EN inicia o modo de download de firmware para baixar firmware através da porta serial.
USB-to-UART Bridge	O chip ponte USB-UART único fornece taxas de transferência de até 3 Mbps.
Micro USB Port	Interface USB. Fonte de alimentação da placa bem como interface de comunicação entre um computador e o módulo ESP32-WROOM-32.
5V Power On LED	Acende quando o USB ou uma fonte de alimentação externa de 5V está conectada à placa.
I/O	A maioria dos pinos do módulo ESP está dividida nos conectores de pinos da placa. Você pode programar ESP32 para habilitar múltiplas funções como PWM, ADC, DAC, I2C, I2S, SPI, etc.

Fonte: Adaptado de ESPRESSIF, 2024.

Conforme dito antes, a ESP32 possui muitos recursos, sendo eles:

- 18 canais conversor analógico-digital (ADC);
- 2 conversores digital para analógico (DAC);
- 10 GPIOS (General Porpose Input/Output) de detecção capacitiva;
- 3 interfaces UART (Receptor/Transmissor Universal Assíncrono);
- 3 interfaces SPI (Serial Protocolo de Interface Periférica);
- 2 interfaces I2C (Circuito Inter-Intergrado para dados);
- 16 canais de saída PWM;
- 2 interfaces I2S (Circuito Inter-Integrado para Áudio).

Para começar, vamos abordar os 25 GPIOs disponíveis. Alguns desses pinos são destinados exclusivamente à entrada, sendo eles: GPIOs 34, 35, 36 e 39 (DIAS, 2022).

É importante notar que nem todos os pinos possuem pull-up interno para entrada, necessitando, portanto, de um pull-up externo nessas situações. Os pinos que contam com pull-up interno são os seguintes GPIOs: 14, 16, 17, 18, 19, 21, 22 e 23. Por outro lado, os pinos que não apresentam pull-up interno são os GPIOs: 13, 25, 26, 27, 32 e 33.

A ESP32 oferece 18 canais analógicos de 12 bits, ou seja, pinos capazes de converter sinais analógicos em digitais.

O ADC (Analog Digital Converter), é o circuito capaz de converter uma grandeza analógica de entrada (tensão) em uma representação digital (valor inteiro). No ESP32 existem 2 conversores AD, ADC1 com 8 canais e o ADC2 com 10 canais. Ambos os conversores possuem resolução de 12 bits alcançando 4096 valores distintos. A faixa de entrada pode ser alternada opcionalmente para níveis entre 0-1V, 0-1.34V, 0-2V ou 0-3.6V. Dessa forma, ao aplicar uma tensão de 0 volts, o valor digital será 0, enquanto a tensão máxima resultará em um valor digital de 4095. Os valores digitais entre esses extremos serão proporcionais às faixas de tensão correspondentes.

A Tabela 2 apresenta os pinos disponíveis da ESP32 como canais analógicos de 12 bits.

Tabela 2 - Pinos de Entrada Analógica da ESP32.

Canal	GPIO
ADC2_CH1	0
ADC2_CH2	2
ADC2_CHO	4
ADC2_CH5	12
ADC2_CH4	13
ADC2_CH6	14
ADC2_CH3	15
ADC2_CH8	25
ADC2_CH9	26
ADC1_CH7	35
ADC1_CHO	36
ADC1_CH1	37
ADC2_CH7	27
ADC1_CH4	32
ADC1_CH5	33
ADC1_CH6	34
ADC1_CH2	38
ADC1_CH3	39

Fonte: Adaptado de ESPRESSIF, 2024.

A ESP32 também apresenta 10 sensores de toque capacitivos, projetados para detectar variações em objetos que possuam carga elétrica, como a pele humana. Esses sensores funcionam de maneira semelhante ao recurso "touch" da tela de um smartphone. Eles podem ser facilmente integrados em blocos capacitivos, substituindo assim os botões mecânicos convencionais. Além disso, os pinos de toque capacitivo têm a capacidade de despertar a ESP32 do estado de "deep sleep", conferindo-lhes uma funcionalidade adicional importante.

Como dito anteriormente, discutimos sobre os canais ADC. No entanto, para realizar a conversão inversa, ou seja, digital para analógico, a ESP32 oferece dois canais de 8 bits. Esses canais, DAC1 (GPIO25) e DAC2 (GPIO26), são responsáveis por converter sinais digitais em saídas de sinal analógico. É importante ressaltar que esses pinos serão fundamentais para a realização do projeto, especialmente se houver a necessidade de gerar sinais analógicos para controlar dispositivos externos ou realizar outras tarefas que exijam essa funcionalidade.

Semelhante ao Arduino, a ESP32 também oferece pinos PWM. A placa disponibiliza 16 canais independentes, permitindo a configuração e geração de sinais com características distintas. Todos os pinos que têm a capacidade de operar como saída podem ser utilizados para gerar sinais PWM, exceto os pinos de entrada GPIOs 34 a 39, que não suportam essa funcionalidade. Para definir um sinal PWM, é necessário especificar os seguintes parâmetros no código: a frequência do sinal, o ciclo de trabalho, o canal PWM e o GPIO no qual o sinal será enviado. Esses parâmetros são cruciais para controlar com precisão o comportamento do sinal PWM.

Além das funcionalidades já mencionadas, a ESP32 também incorpora um transdutor em seu conjunto de características. O transdutor trata-se de um sensor capaz de detectar variações na tensão de saída quando submetido a um campo magnético. O princípio do Efeito Hall é fundamental para o funcionamento desse transdutor, onde a corrente elétrica sofre deflexão de sua trajetória em resposta à influência do campo magnético. Este sensor é integrado ao interior do chip do microcontrolador. Para gerar o campo magnético necessário para sua operação, basta posicionar um ímã próximo a ele. Essa capacidade de detecção magnética da ESP32 é uma ferramenta valiosa que pode ser utilizada em diversas aplicações, desde detecção de proximidade até controle de posição em sistemas embarcados.

A ESP32 também é equipada com três portas seriais essenciais para diversas funcionalidades. A primeira (RX0, TX0) é comumente utilizada para programação, permitindo a comunicação bidirecional de dados entre a ESP32 e dispositivos externos, como um computador. Os pinos associados a essa porta são GPIO3 (U0RXD) e GPIO1 (U0TXD). Além disso, outra porta serial está disponível, composta pelos pinos GPIO16 (U2RXD) e GPIO17 (U2TXD). Esses pinos possibilitam a comunicação serial adicional, ampliando as capacidades de comunicação da ESP32 com outros dispositivos ou microcontroladores. Estas portas UART da ESP32 serão essenciais para o desenvolvimento e funcionamento do projeto, facilitando a integração do modem HART que iremos utilizar.

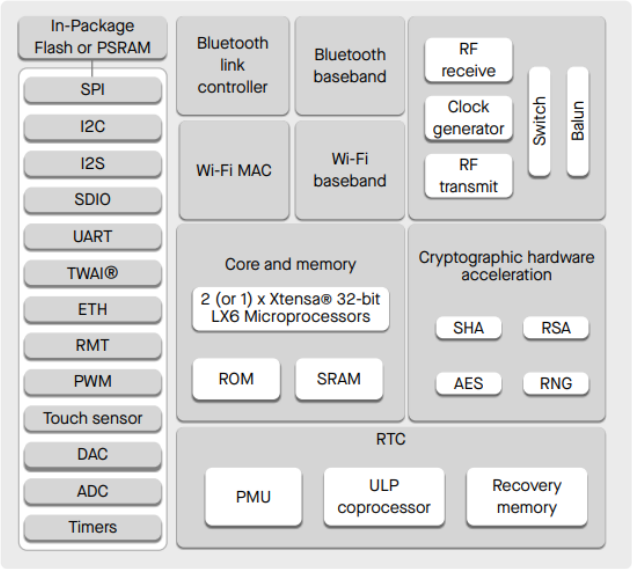
A ESP32 permite que todos os seus GPIOs sejam configurados para operar como interrupções. Isso significa que eles podem ser programados para detectar mudanças em sinais específicos, como bordas de subida ou descida, e responder imediatamente, interrompendo a execução normal do programa para lidar com a situação.

O pino de habilitação (EN) é utilizado para ativar o regulador de 3,3V na ESP32. Ele possui um pull-up interno, o que significa que, por padrão, o regulador está ativado. Para desativá-lo, é necessário conectar o pino EN ao GND. Uma aplicação comum desse recurso é utilizar um botão conectado a esse pino para reiniciar o ESP32 de forma conveniente.

E de acordo com as especificações do datasheet da placa, a corrente máxima absoluta que pode ser fornecida por cada GPIO é de 40mA, enquanto o máximo que pode ser consumido por eles é de 28mA. É importante observar esses limites para garantir o funcionamento correto e seguro da ESP32, conforme as condições de operação recomendadas.

Para concluir, temos um diagrama apresentado na Figura 3 abaixo que resume todas as características e funções da ESP32. Esse diagrama expõe que o ESP32 tem dual core, tem uma área do chip que controla o WiFi, e outra que controla o Bluetooth. Também tem aceleração de hardware para criptografia, que permite a conexão com LoRa, rede de longa distância que permite uma conexão até 15km, isso com uso de uma antena. Ainda observamos o clock generator, real time clock, e outros pontos que tratam, por exemplo, de PWM, ADC, DAC, UART, SDIO, SPI, entre outros, que fazem esse dispositivo bastante completo e funcional.

Figura 3 – Diagrama de blocos do ESP32.



Fonte: ESPRESSIF, 2024.

2.3.3 Protocolo HART

O Protocolo HART, lançado pela Fisher Rosemount em 1980, tem sido um marco na indústria. HART, que significa "Highway Addressable Remote Transducer", inicialmente desenvolvido como uma solução proprietária, foi aberto à comunidade em 1990, quando um grupo de usuários foi implementado (SEIXAS FILHO, 2007).

Uma das características mais marcantes desse protocolo é sua capacidade de permitir o uso de instrumentos inteligentes sobre os cabos tradicionais 4-20mA. Essa adaptação é

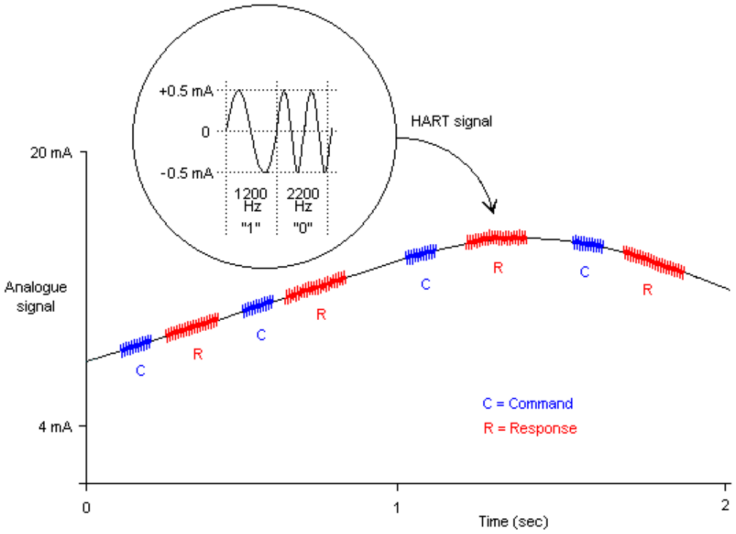
possível devido à sua baixa velocidade de transmissão, o que significa que os cabos padrão usados na instrumentação podem ser desligados sem alterações significativas.

Os dispositivos habilitados para essa comunicação híbrida são conhecidos como "smart", tornando possível uma gama mais ampla de funcionalidades e controle em sistemas industriais e de automação. Essa tecnologia de inteligência em dispositivos conectados via HART representa um avanço significativo na eficiência e na capacidade de monitoramento e controle dos processos industriais.

Sendo assim o sinal de Hart é usado na indústria para comunicação entre dispositivos como sensores e sistemas de controle. Ele usa duas frequências para representar dados digitais: 1200 Hz para o bit 1 e 2400 Hz para o bit 0. Esse sinal digital é enviado junto com um sinal analógico de corrente de 4-20mA. Para enviar dados digitais, o dispositivo varia a corrente de 4-20mA conforme necessário e superpõe os sinais de 1200 Hz ou 2400 Hz para representar os bits.

A comunicação é bidirecional, permitindo tanto a transmissão de dados dos sensores para o sistema de controle quanto o envio de comandos e configurações do sistema de controle para os dispositivos de campo. Essa combinação de sinais permite uma comunicação eficaz e confiável em ambientes industriais, facilitando o monitoramento e controle dos processos. A Figura 4 ilustra bem o funcionamento dessa comunicação.

Figura 4 - Sinal Hart sobreposto ao sinal 4-20 mA.



Fonte: Adaptado de SEIXAS FILHO, 2007.

O HART fornece dois canais de comunicação simultâneos, um analógico e outro digital: Um sinal de 4-20 mA comunica o valor medido primário (PV) como um valor

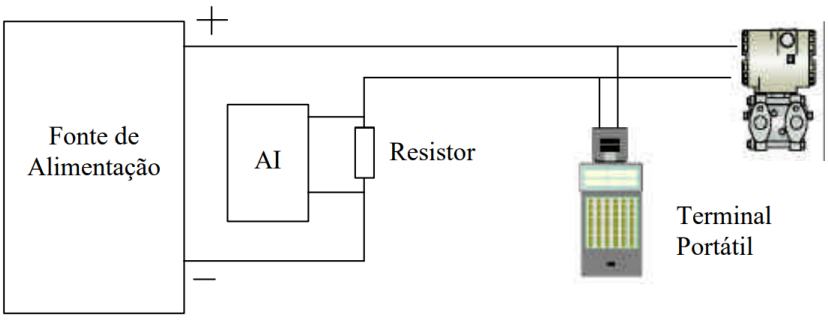
analógico de corrente usando a fiação que fornece energia ao instrumento. O sistema host então converte o valor atual em um valor físico de acordo com os parâmetros definidos pelo software HART. Por exemplo, 7 mA = 80 graus F.

A transmissão de informações de dispositivos digitais é realizada por meio da codificação de um sinal digital usando a técnica Frequency Shift Keying (FSK) através da mesma fiação utilizada para comunicações analógicas de 4-20 mA. Esse sinal digital inclui dados do dispositivo, como valor da variável de processo (PV), status do dispositivo, diagnósticos e outras informações relevantes. O uso do FSK garante que o sinal digital seja contínuo em fase e não cause interferência no sinal analógico presente na mesma linha, mantendo a integridade das comunicações em ambientes industriais. O protocolo segue o padrão Bell 202 Frequency Shift Keying, o que garante uma padronização reconhecida e confiável para esse tipo de comunicação (FIELDCOMM GROUP, 2024).

A rede pode ser configurada em topologia ponto a ponto ou multidrop, oferecendo flexibilidade na implantação e interconexão dos dispositivos. A arquitetura suporta até dois mestres: um mestre primário, que pode ser um computador, um controlador lógico programável (CLP) ou um multiplexador; e um mestre secundário, geralmente representado por terminais portáteis de configuração e calibração.

Para garantir o funcionamento adequado da rede, é necessário incluir uma resistência de pelo menos 230 ohms entre a fonte de alimentação e o instrumento. A configuração correta do terminal portátil de configuração deve ser entre o resistor e o dispositivo de campo, conforme indicado na Figura 5 do sistema.

Figura 5 - Conexão de uma entrada a um instrumento HART.

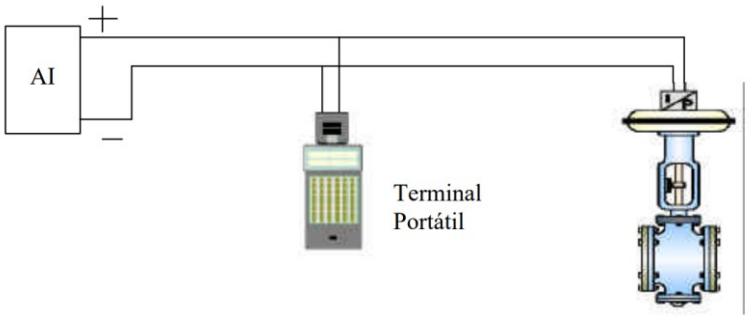


Fonte: Adaptado de SEIXAS FILHO, 2007.

O resistor em série geralmente já faz parte integrante dos cartões de entrada de controladores de loop único e de cartões de entrada remota, portanto, não há necessidade de adicioná-lo. Outros dispositivos de medição são conectados em série no loop de corrente, o

que causa uma queda de tensão em cada dispositivo. Para conectar dispositivos de saída a uma saída analógica, não é necessário usar um resistor shunt, assim como é mostrado na Figura 6.

Figura 6 - Conexão de uma saída HART.

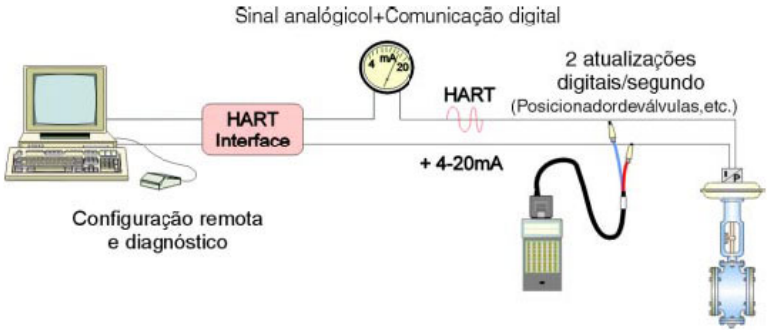


Fonte: Adaptado de SEIXAS FILHO, 2007.

O protocolo HART é do tipo mestre/escravo, o que significa que um instrumento de campo (escravo) só responde quando é solicitado por um mestre. Na rede HART, dois mestres (primário e secundário) podem se comunicar com um instrumento escravo. Os mestres secundários, como terminais de configuração portáteis, podem ser conectados em qualquer ponto da rede e se comunicar com os instrumentos de campo sem interferir na comunicação com o mestre primário. O mestre primário é normalmente um DCS (Sistema de Controle Digital Distribuído), um PLC (Controlador Lógico Programável), um sistema central baseado em computador ou sistema de monitoramento (FIELDCOMM GROUP, 2024).

Uma instalação típica com dois mestres é ilustrada na Figura 7.

Figura 7 - O Protocolo HART permite que dois dispositivos Mestre acessem informações de um mesmo equipamento de campo (escravo).



Fonte: Adaptado de FIELDCOMM GROUP, 2024.

O Protocolo HART é utilizado de diversas maneiras para trocar informações entre instrumentos de campo inteligentes e sistemas de controle central ou equipamentos de monitoração. A comunicação mestre/escravo digital, simultânea com o sinal analógico de 4-

20mA, é a mais comum. Nesse modo, a informação digital proveniente do instrumento escravo é atualizada duas vezes por segundo no mestre, enquanto o sinal de 4-20mA carrega a variável primária para controle de forma contínua.

A distância máxima do sinal HART é cerca de 3000 m com cabo de par trançado blindado e 1500 m com cabo múltiplo de blindagem simples. Existem barreiras de segurança intrínseca especiais que permitem o tráfego do sinal HART. O fator mais limitante do comprimento do cabo é sua capacitância, onde uma maior capacitância e um número maior de dispositivos conectados resultam em uma distância máxima menor (SEIXAS FILHO, 2007).

O HART é um protocolo de comunicação de solicitação-resposta, o que significa que durante a operação normal, cada comunicação é iniciada por uma solicitação (ou comando) do dispositivo de comunicação host. O host geralmente é um sistema de controle distribuído (DCS), um controlador lógico programável (PLC) ou um sistema de gerenciamento de ativos baseado em PC. O dispositivo é tipicamente um dispositivo de medição de campo, como um transmissor de pressão, nível, temperatura, fluxo ou outro.

Para garantir a interoperabilidade entre dispositivos HART de diferentes fabricantes, os comandos são definidos nas especificações HART e implementados em todos os dispositivos compatíveis com HART. Os usuários não precisam se preocupar com os detalhes desses comandos, pois eles são gerenciados pelas funções do host. Os recursos específicos de um dispositivo são disponibilizados para o host por meio da Descrição do Dispositivo (DD) associada ao dispositivo.

As respostas de comunicação incluem indicações definidas do status do dispositivo, que são interpretadas pelo host para fornecer informações básicas de diagnóstico. O conjunto de comandos HART garante uma comunicação uniforme e consistente para todos os dispositivos de campo, divididos em três classes: Universal, Prática Comum e Específico do Dispositivo.

- **Comandos Universais:** acessam informações básicas como leitura de variáveis primárias e unidades.
- **Comandos de Prática Comum:** oferecem funções implementadas por muitos dispositivos HART.
- **Comandos Específicos do Dispositivo:** são exclusivos de cada dispositivo e acessam informações de configuração, calibração e construção. Informações sobre comandos específicos do dispositivo estão disponíveis nos fabricantes de dispositivos.

Uma lista parcial dos tipos de comandos HART é mostrada na Tabela 3.

Tabela 3 - Lista parcial de comandos HART.

Comandos Universais	**Comandos de prática comum**	**Comandos específicos do dispositivo**
Ler fabricante e tipo de dispositivo	Ler seleção de até quatro variáveis dinâmicas	Ler ou escrever limite de fluxo baixo
Ler variável primária (PV) e unidades	Escrever constante de tempo de amortecimento	Iniciar, parar ou limpar totalizador
Ler saída atual e percentual da faixa	Escrever valores de faixa do dispositivo	Ler ou escrever fator de calibração de densidade
Ler até quatro variáveis dinâmicas pré-definidas	Calibrar (definir zero, definir alcance)	Escolher PV (massa, fluxo ou densidade)
Ler ou escrever tag de oito caracteres, descritor de 16	Definir corrente de saída fixa	Ler ou escrever informações de materiais ou construção
caracteres, data	Realizar auto-teste	Ajustar calibração do sensor
Ler ou escrever mensagem de 32 caracteres	Realizar reset mestre	Habilitar PID
Ler valores de faixa do dispositivo, unidades e constante de tempo de	Ajustar zero da variável primária (PV)	Escrever ponto de ajuste do PID
amortecimento	Escrever unidade da variável primária (PV)	Caracterização da válvula
Ler ou escrever número de montagem final	Ajustar zero e ganho do DAC	Ponto de ajuste da válvula
Escrever endereço de sondagem	Escrever função de transferência (raiz quadrada/linear)	Limites de deslocamento
Ler ou escrever mensagem de 32 caracteres	Escrever número de série do sensor	Unidades do usuário
	Ler ou escrever atribuições de variáveis dinâmicas	Informações de exibição local

Fonte: Adaptado de FIELDCOMM GROUP, 2024.

2.3.4 Protocolo MQTT e Ambiente Node-RED

O protocolo Message Queue Telemetry Transport (MQTT) é uma tecnologia usada para comunicação em ambientes de Internet das Coisas (IoT), operando sobre o protocolo Transport Control Protocol (TCP). Inicialmente desenvolvido pela IBM, o MQTT foi concebido como um método eficiente e leve de comunicação máquina a máquina. Ele foi posteriormente padronizado pela ISO/IEC 20922 e integrado ao OASIS.

O MQTT utiliza o modelo de comunicação de publicação-assinatura, no qual os clientes não precisam enviar solicitações de atualização ativamente. Isso reduz o uso de recursos, tornando o MQTT ideal para ambientes com restrições de largura de banda.

Neste protocolo, os dispositivos IoT se conectam a um servidor central, conhecido como "corretor", que gerencia o envio e recebimento de mensagens entre os clientes MQTT. Os clientes não se comunicam diretamente entre si; todas as mensagens são roteadas pelo corretor. Cada mensagem MQTT possui um tópico, organizado em uma estrutura de árvore, que os clientes podem assinar ou publicar. Quando um cliente publica uma mensagem em um tópico, o corretor a distribui para todos os clientes que estão inscritos naquele tópico específico.

O MQTT foi projetado para comunicação assíncrona, permitindo que as assinaturas e publicações de diferentes entidades ocorram de forma paralela. Além disso, o protocolo oferece três níveis de Qualidade de Serviço (QoS), que garantem diferentes níveis de confiabilidade na entrega das mensagens. Comparado a outros protocolos como HTTP, o MQTT requer menos recursos computacionais e de largura de banda, tornando-o especialmente adequado para dispositivos com recursos limitados.

No entanto, é importante observar que nem todos os brokers MQTT oferecem o mesmo nível de recursos em termos de autenticação e criptografia. Por exemplo, o aplicativo de código aberto Eclipse Mosquitto implementa muitos recursos padronizados do MQTT, como SSL/TLS e suporte a certificados de cliente. No entanto, por padrão, o Mosquitto não fornece segurança robusta para o esquema de mensagens, e informações de autenticação podem ser transmitidas em texto simples. Portanto, é necessário implementar mecanismos de segurança adicionais para proteger as informações transmitidas (DINCULEANĂ, 2019).

O protocolo MQTT facilita a comunicação entre dispositivos IoT usando o modelo de publicação/assinatura. Os dispositivos que usam MQTT são clientes MQTT, que podem enviar (publicar) ou receber (assinar) mensagens. O agente MQTT, também chamado de broker, coordena essas mensagens entre os clientes.

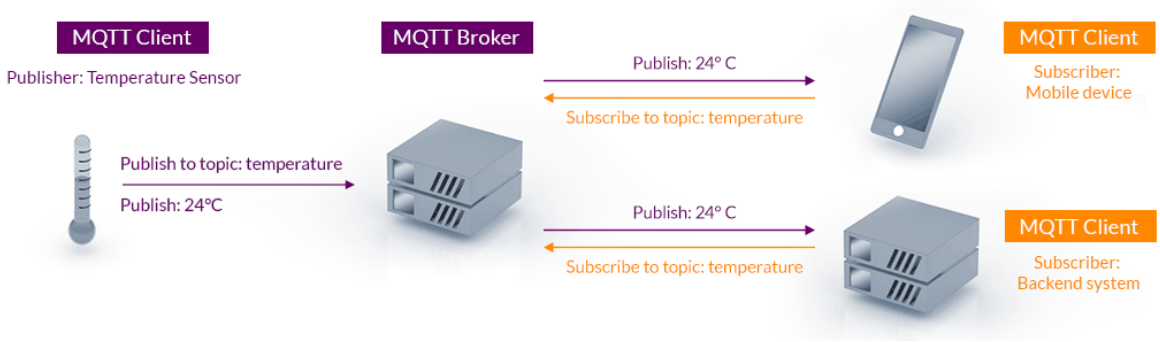
Os clientes MQTT se conectam ao agente MQTT usando uma conexão MQTT, onde o cliente envia uma mensagem CONNECT e o agente responde com CONNACK para confirmar a conexão. Os clientes e agentes dependem da pilha TCP/IP para comunicação, e os clientes nunca se conectam diretamente entre si.

No MQTT, os clientes podem publicar mensagens contendo um tópico e dados associados, organizados hierarquicamente. Por exemplo, uma lâmpada pode publicar uma mensagem "on" no tópico "livingroom/light" para indicar que está ligada.

Os clientes também podem assinar tópicos específicos para receber mensagens relevantes. Por exemplo, um aplicativo de casa inteligente pode assinar o tópico "light" para contar quantas luzes estão ligadas e atualizar um contador conforme recebe mensagens de status.

Em resumo, o MQTT permite uma comunicação eficiente em ambientes de IoT, com clientes que publicam e assinam mensagens e um agente (broker) que gerencia e roteia essas mensagens entre os dispositivos conectados. A Figura 8 exemplifica bem o funcionamento do protocolo MQTT.

Figura 8 - Exemplo de aplicação MQTT.



Fonte: MQTT ORG, 2024.

Outro aspecto fundamental a ser explorado é o Node-RED que se trata de uma plataforma de código aberto que simplifica a programação em aplicações de Internet das Coisas (IoT) por meio de uma abordagem visual, permitindo aos usuários conectar blocos de código denominados "nós" para executar atividades específicas. Quando esses nós são conectados para realizar uma sequência de atividades, eles formam um fluxo.

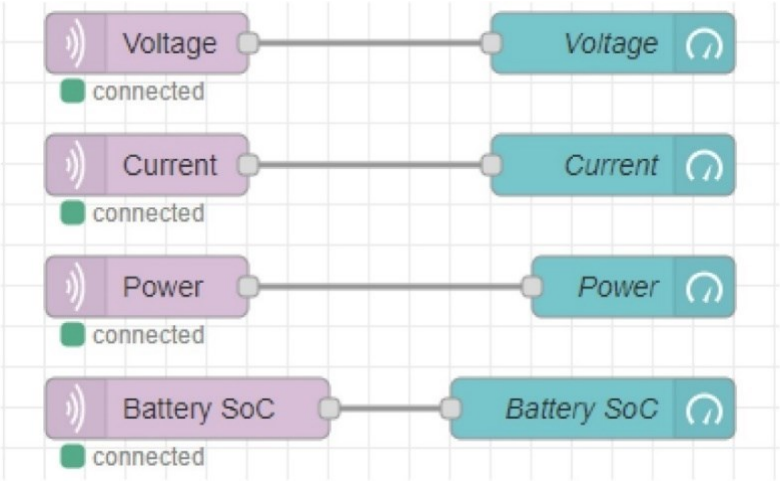
No contexto da comunicação MQTT, o Node-RED pode ser implantado no dispositivo que atua como broker (como um computador ou uma ESP32), facilitando a comunicação entre esse dispositivo e os demais dispositivos da rede.

Além disso, o Node-RED oferece recursos para criar interfaces gráficas que mediam a interação entre o usuário e o sistema, bem como para armazenar e coletar dados de bancos de dados, criar eventos baseados em tempo, e realizar coleta de dados da web, entre outras funcionalidades.

A Figura 9, por exemplo, ilustra um fluxo desenvolvido por Baig (2017) utilizando o ambiente de programação visual Node-RED. Nesse contexto, os dados de tensão, corrente, potência e estado de carga da bateria são transmitidos por meio do protocolo MQTT a partir dos componentes de hardware do sistema e podem ser facilmente visualizados na interface do

usuário do Node-RED. O servidor Node-RED está instalado localmente em uma máquina dedicada para processamento e exibição dos dados coletados.

Figura 9 - Exemplo de Fluxo Node-Red para aquisição de dados.



Fonte: BAIG, 2021.

3 METODOLOGIA

3.1 Considerações Iniciais

Neste capítulo, abordaremos a análise das tecnologias empregadas no projeto, detalhando o desenvolvimento tanto do KIT didático quanto do firmware.

E para alcançar os objetivos do trabalho, o desenvolvimento do projeto foi dividido em etapas distintas: (a) Estudo de tecnologias, (b) Desenvolvimento do KIT didático e (c) Desenvolvimento do Firmware.

Ao abordar cada uma dessas etapas de forma sistemática, buscamos garantir o sucesso e a qualidade do projeto, demonstrando sua viabilidade e utilidade em aplicações reais.

3.2 Estudo de Tecnologias

Na fase inicial, iniciamos com uma análise abrangente dos componentes que serão integrados à ESP32, a peça central deste projeto.

Os módulos propostos para o KIT abrangem um conversor de nível com isolamento óptico, um MOSFET PWM, um amplificador operacional de ganho LM358, um conversor de corrente para tensão, um conversor de tensão para corrente, um filtro passa-baixa (não diretamente conectado à ESP32, mas disponível no painel do KIT), um relé de estado sólido (SSR), um modem HART e uma mini fonte de alimentação Hi-link HLK-10M12 para os módulos. Contudo, mantemos em aberto a possibilidade de integrar mais elementos ao KIT no futuro.

Para criar os esquemas do projeto, utilizamos o software Fritzing, uma ferramenta de hardware open-source desenvolvida na Universidade Aplicada de Postdam, Alemanha. O Fritzing permite criar rapidamente diagramas eletrônicos e layouts de placas de circuito impresso (PCB). Com o Fritzing, conseguimos desenvolver nossos diagramas eletrônicos de forma rápida e precisa, economizando tempo e aumentando a produtividade. Além disso, o software gera automaticamente diagramas elétricos e layouts de PCB a partir dos modelos criados no protoboard, proporcionando um resultado profissional.

Com relação ao modem HART, foi essencial realizar um estudo aprofundado sobre seu funcionamento, os dispositivos associados e as possibilidades de comandos. Para conduzir os testes com dispositivos HART, utilizamos o software PACTware, uma interface aberta e independente amplamente reconhecida para operar dispositivos, sistemas e interfaces de

comunicação. Essa escolha permitiu uma integração eficaz e uma análise detalhada das funcionalidades dos dispositivos HART em nosso projeto.

E para visualização e envio de informações dos módulos de forma prática e profissional, optamos pelo uso do Node-RED com comunicação MQTT. O Node-RED é uma ferramenta de desenvolvimento de baixo código baseada em fluxo, projetada para conectar dispositivos de hardware, APIs e serviços online, especialmente como parte da Internet das Coisas. Essa escolha oferece uma abordagem intuitiva e visual para programação, permitindo uma integração eficiente entre os diversos componentes do sistema, facilitando o monitoramento e controle das informações de forma acessível e poderosa.

E por fim, para o desenvolvimento do firmware da ESP32, optamos por utilizar o IDE Visual Studio com PlataformIO, uma plataforma de desenvolvimento integrada que simplifica a criação e gerenciamento de projetos para sistemas embarcados.

3.3 Desenvolvimento do Kit Didático

Nesta etapa do projeto, apresentamos de forma detalhada a montagem de cada módulo conectado à ESP32. Isso inclui esquemáticos e diagramas eletrônicos feito no FRITZING, explicação sobre os módulos e sua aplicação específica no contexto do projeto.

A Figura 10 abaixo apresenta um protótipo do painel físico do KIT didático contendo todos os elementos mencionados anteriormente. É importante ressaltar que este protótipo não representa a versão final, uma vez que o desenvolvimento ainda está em andamento.

Figura 10 - Protótipo do painel do KIT didático.

Fonte: Autoria Própria, 2024.

3.3.1 Mini Fonte Hi-link HLK-10M12

Para alimentação elétrica da maioria dos módulos utilizados no KIT deste projeto, será empregada a Mini Fonte Hi-link HLK-10M12, ilustrada na Figura 11 abaixo:

Figura 11 - Fonte Hilink HLK-10M12.

Fonte: SHENZHEN HI-LINK ELECTRONIC, 2024.

A Fonte Hilink HLK-10M12 desempenha um papel essencial como uma fonte step down AC-DC, convertendo uma tensão de entrada AC na faixa de 100 a 240V para uma saída DC de 12V, capaz de fornecer até 833mA de corrente com uma potência máxima de 10W e uma eficiência de aproximadamente 78% (SHENZHEN HI-LINK ELECTRONIC, 2024).

Este dispositivo oferece versatilidade com sua entrada de tensão bivolt, funcionando de forma eficiente independentemente da rede elétrica ser de 110V ou 220V AC.

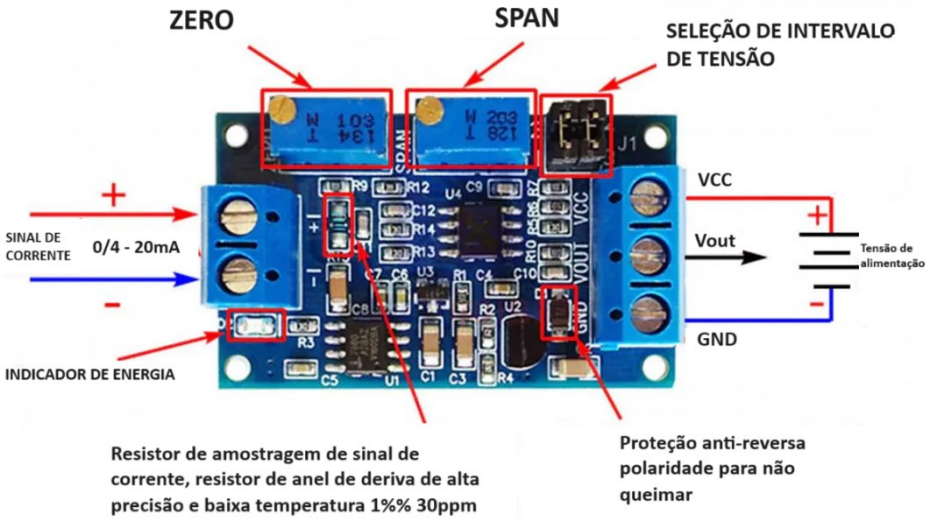
A Fonte Hilink HLK-10M12 é projetada com múltiplos mecanismos de segurança, incluindo proteção contra superaquecimento, sobrecarga de corrente, curto-circuito e isolamento entre a saída e a entrada de tensão. Seu encapsulamento selado de quatro pinos torna-a adequada para espaços limitados, como painéis residenciais e tomadas elétricas, facilitando sua instalação em diversos projetos. Este componente é notável por sua saída de tensão estável e baixo consumo de energia, sendo amplamente utilizado em projetos de robótica, automação, domótica e IoT, em conjunto com uma variedade de microcontroladores disponíveis no mercado.

Portanto essa fonte se torna uma solução confiável e com excelente custo-benefício para converter tensão AC em DC em espaços reduzidos, a Fonte Hilink HLK-10M12 é uma escolha ideal para este projeto.

3.3.2 Módulo conversor de corrente para tensão (4-20mA para 0-3,3V)

O Conversor de Corrente em Tensão HW-685 apresentado na Figura 12, é um módulo projetado para facilitar a comunicação entre sensores e atuadores industriais que utilizam sinais de corrente na saída e microcontroladores como Arduino, PIC, ESP, e outros que operam com leitura de sinais de tensão.

Figura 12 - Conversor de corrente em Tensão HW-685.



Fonte: Adaptado de CASA DA ROBÓTICA, 2024.

Este dispositivo converte sinais de corrente variando de 4mA (valor mínimo) a 20mA (valor máximo) em sinais de tensão que variam de 0V a 10V. Isso permite ao desenvolvedor configurar o módulo ajustando o Trimpot ZERO para definir a tensão de saída correspondente a 0V na corrente mínima (por exemplo, 4mA) e o Trimpot SPAN para ajustar a tensão de saída conforme a corrente máxima (por exemplo, 20mA). Dessa forma, é possível adaptar o intervalo de saída para compatibilidade com diferentes microcontroladores, visto que alguns leem sinais de 0 a 3,3V e outros de 0 a 5V.

Para facilitar a conexão, o módulo possui um par de bornes a parafuso para entrada de corrente (+/-) e um trio de bornes (GND e VCC para alimentação da placa, VOUT para tensão de saída), além de 4 perfurações para fixação. A tensão de alimentação deve ser DC na faixa de 7V~12V e é necessário alimentar com tensão maior do que a tensão desejada na saída.

O módulo possui 4 pinos conectados por meio de jumpers para ajustar o intervalo de tensão de saída. A configuração de jumpers pode ser observada na Tabela 4.

Tabela 4 - Configuração de intervalo de tensão no HW-685.

Intervalo de tensão	**Jumpers J1**	
	1-2	3-4
0 – 2,5 V	Desconectado	Conectado
0 – 3,3 V	Desconectado	Desconectado
0 – 5 V	Conectado	Conectado
0 – 10 V	Conectado	Desconectado

Fonte: Adaptado de TURAIS TECH, 2024.

O loop de corrente de 4–20mA é um método popular de sinalização analógica para enviar sinais de instrumentação. Tipicamente, a variável física medida tem um valor mínimo de 4 mA e um valor máximo de 20mA. Um valor mínimo de corrente não nulo ajuda na detecção de possíveis problemas de transmissão, como cabos quebrados. Para converter a variação de corrente de 4–20mA em uma faixa de tensão, há um resistor de precisão conectado aos terminais de entrada

A Equação (1) é utilizada para calcular o valor da tensão de saída que corresponde ao valor da corrente de entrada.

$$P_v = \frac{P_{vhigh} - P_{vlow}}{I_{high} - I_{low}} \times (I - I_{low}) + P_{vlow} \tag{1}$$

Nesse contexto, onde Pv representa o valor físico medido, $Pvlow$ é o valor mínimo do intervalo de medição, $Pvhigh$ é o valor máximo do intervalo de medição, $Ilow$ é o valor mínimo da corrente correspondente, $Ihigh$ é o valor máximo da corrente correspondente, e I é a corrente que corresponde ao valor medido (ALKASAP, 2021).

Para o projeto configuramos o módulo no intervalo de tensão de 0-3,3V. Isso foi feito retirando os jumpers, como explicado na Tabela 4, e ajustando os trimpots zero e span para conversão ser no intervalo de 4-20mA para 0-3,3V.

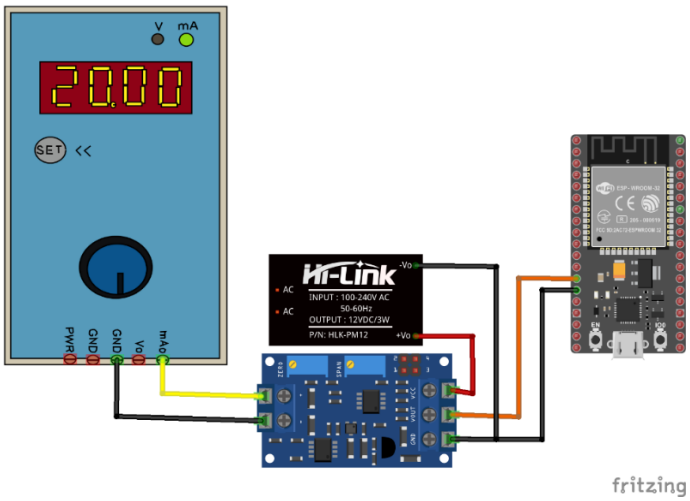
E isso só foi possível com o auxílio de um gerador de sinal ajustável ONEBUSI que é ideal para aplicações que requerem sinais analógicos precisos, como testes de equipamentos eletrônicos, automação industrial e controle de processos. O Gerador é alimentado por uma fonte de energia de 15Vcc a 27Vcc ou através de uma conexão micro USB. Com uma corrente de saída ajustável de 0 a 20mA ou 4 a 20mA, e uma tensão de saída ajustável de 0 a 10Vcc ou 2 a 10Vcc. O gerador de sinal ajustável pode ser visto na Figura 13.

Figura 13 - Gerador de sinal ajustável.

Fonte: Adaptado de SMARTKITS, 2024.

Como mencionado anteriormente, a ESP32 oferece 18 canais analógicos de 12 bits, ou seja, pinos capazes de converter uma grandeza analógica de entrada (tensão) em uma representação digital (valor inteiro). Os conversores analógico-digitais (ADCs) da ESP32 possuem uma resolução de 12 bits, o que significa que podem distinguir até 4096 valores distintos. Portanto, este módulo foi conectado a uma entrada analógica ADC para enviar o valor da tensão convertida para a ESP32, com base na corrente gerada pelo gerador de sinal ajustável. O diagrama mostrando essa conexão está representado na Figura 14.

Figura 14 - Diagrama de conexão do módulo HW-685.

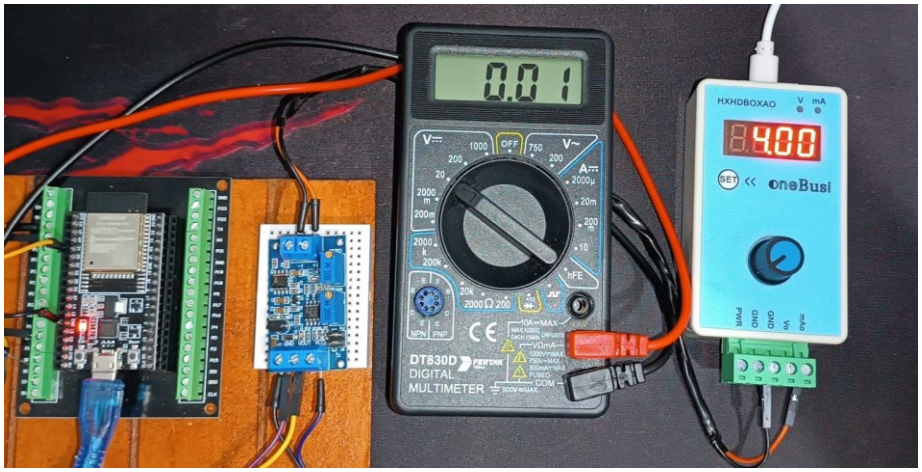


Fonte: Autoria Própria, 2024.

O gerador de sinal foi alimentado por meio de uma conexão micro USB, e sua configuração foi ajustada para gerar um sinal de corrente no intervalo de 4-20mA. O módulo foi alimentado por uma fonte AC/DC com entrada de 110/220V e saída de 12V/10W. Conforme observado na Figura 14, as saídas do gerador de sinal foram conectadas ao par de bornes a parafuso para entrada de corrente (+/-), enquanto a tensão de saída do módulo foi conectada diretamente à entrada analógica GPIO 12 da ESP32.

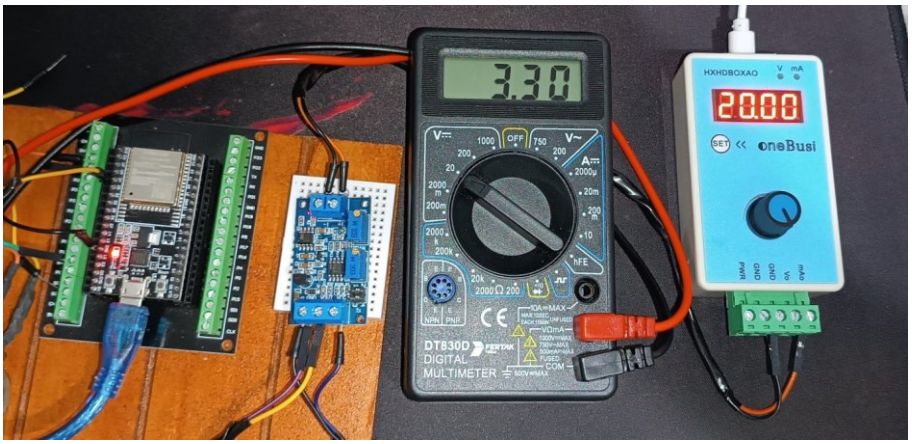
Para ajustar e testar o módulo de conversão HW-685, utilizamos um multímetro digital. Nas Figuras 15 e 16, é possível observar a montagem física do circuito conforme representado no Fritzing, juntamente com os valores de tensão de saída do módulo, tanto para o valor mínimo de 4mA quanto para o valor máximo de 20mA, respectivamente. O objetivo era obter 0V para 4mA e 3,3V para 20mA.

Figura 15 - Tensão de saída do módulo para 4mA.



Fonte: Autoria Própria, 2024.

Figura 16 - Tensão de saída do módulo para 20mA.



Fonte: Autoria Própria, 2024.

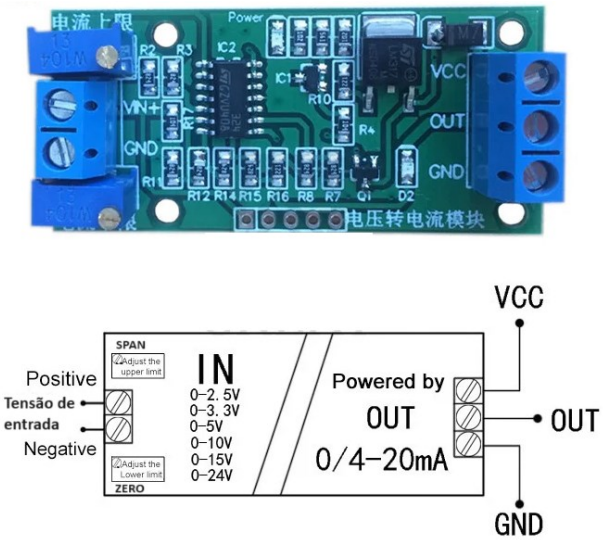
Consequentemente, os valores obtidos nos testes estão alinhados com as expectativas dentro desse intervalo. Esses dados estão sendo recebidos pela ESP32 e podem ser monitorados e empregados em saídas analógicas, como parte das aplicações na disciplina de Instrumentação Industrial. Os resultados foram registrados na seção de Resultados e Discussões.

O uso do módulo para conectar sensores de corrente (4-20 mA) em processos industriais oferece vantagens significativas para monitoramento e controle. Esses sensores, que monitoram variáveis como temperatura, pressão, vazão e nível, são essenciais para garantir a operação eficiente e segura dos equipamentos (ALAVALA, 2009). Ao converter o sinal de corrente em tensão, e usando a ESP32, é possível implementar sistemas de controle mais precisos e responsivos. Isso possibilita ajustes em tempo real com base nas leituras dos sensores, permitindo uma resposta rápida a variações nas condições do processo.

3.3.3 Módulo conversor de tensão para corrente (0-3,3V para 4-20mA)

Assim como o módulo anterior, porém de forma inversa, conforme ilustrado na Figura 17, temos o módulo conversor de tensão para corrente. Este dispositivo eletrônico altamente funcional foi projetado para simplificar a comunicação entre sensores e atuadores industriais que operam com sinais de tensão na saída, e microcontroladores como Arduino, PIC e outros que são capazes de ler sinais de corrente.

Figura 17 - Conversor de Tensão em Corrente.



Fonte: Adaptado de CASA DA ROBÓTICA, 2024.

Este conversor recebe sinais de tensão variando de 0V (valor mínimo) a 24V (valor máximo) e os converte em sinais de corrente na faixa de 0/4 a 20mA. Isso permite ao usuário ajustar a corrente de saída para 4mA na tensão inicial desejada (por exemplo, 0V) usando o potenciômetro ZERO, e posteriormente ajustar a corrente de saída conforme a tensão final (por exemplo, 3,3V) usando o trimpot SPAN. Essa funcionalidade de ajuste fino dos potenciômetros, assim como do módulo anterior, torna este dispositivo altamente adaptável a diferentes condições de operação e requisitos específicos de aplicação industrial.

Para simplificar a conexão, o Conversor está equipado com um par de bornes a parafuso para entrada de tensão (VIN/GND) e um trio de bornes (GND e VCC para alimentação da placa, e VOUT para a corrente de saída). A instalação deste conversor é conveniente, pois possui 4 perfurações na placa que permitem a fixação com parafusos (parafusos não inclusos), por exemplo, podendo ser montado em cases ou outros recipientes conforme necessários.

O processo de conversão de tensão para corrente no módulo é realizado por meio de um circuito que incorpora o amplificador operacional LM324 e outros elementos. O LM324 amplifica e condiciona o sinal de entrada de tensão, transformando-o em uma corrente de saída na faixa de 4 a 20mA. Os potenciômetros de ajuste de zero e span são fundamentais para calibrar o conversor, determinando a corrente de saída quando a tensão de entrada é mínima (zero) e máxima (span), respectivamente. Esses ajustes são essenciais para estabelecer a relação linear entre a tensão de entrada e a corrente de saída desejada. O circuito pode incluir ainda mecanismos de feedback e controle, contribuindo para a estabilidade e precisão da conversão em diversas condições de operação, garantindo assim o funcionamento confiável do conversor.

Gerar corrente precisa em um circuito de sinal é essencial para utilizar a corrente como representação analógica de uma grandeza física. Uma solução eficaz é o uso de um amplificador operacional com feedback negativo, projetado para manter a corrente em um valor específico ajustando a tensão aplicada ao circuito de carga. Essa abordagem, conhecida como fonte de corrente, permite controlar com precisão a corrente independentemente das variações na carga ou resistência do circuito.

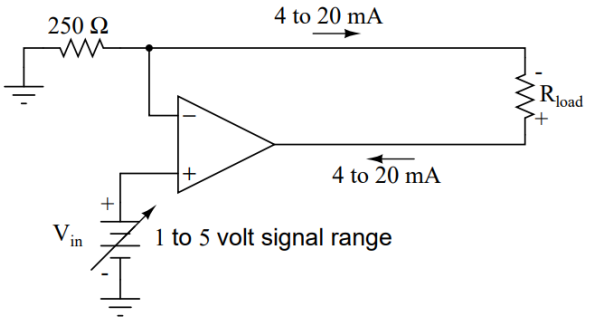
A Equação (2) é a utilizada para calcular o valor da corrente de saída que corresponde ao valor da tensão de entrada.

$$I = I_{low} + (P_v - P_{vlow}) \times \frac{I_{high} - I_{low}}{P_{vhigh} - P_{vlow}} \tag{2}$$

Nesse contexto, I representa o valor físico medido, $Ilow$ é o valor mínimo do intervalo de medição, $Ihigh$ é o valor máximo do intervalo de medição, $Pvlow$ é o valor mínimo da tensão correspondente, $Pvhigh$ é o valor máximo da tensão correspondente, e Pv é a tensão que corresponde ao valor medido (ALKASAP, 2021).

O autor Kuphaldt (2009) propõe um circuito onde a tensão de entrada é proveniente de um transdutor/amplificador calibrado para gerar 1 volt a 0% e 5 volts a 100% da medição física. Neste circuito, uma faixa padrão de sinal de corrente analógica de 4 mA a 20 mA é utilizada, representando 0% a 100% da faixa de medição, respectivamente. Como mostrado na Figura 18, essa implementação utiliza um resistor de 250 Ω de precisão para converter a tensão de entrada em corrente de saída. Com uma entrada de 5 volts, o resistor de 250 Ω gera uma corrente de 20 mA, independente da resistência da carga (R) ou da resistência do fio no loop, contanto que o amplificador operacional tenha uma tensão de alimentação adequada para produzir a voltagem necessária para atingir os 20 mA na carga R.

Figura 18 - Amplificador operacional com feedback negativo.

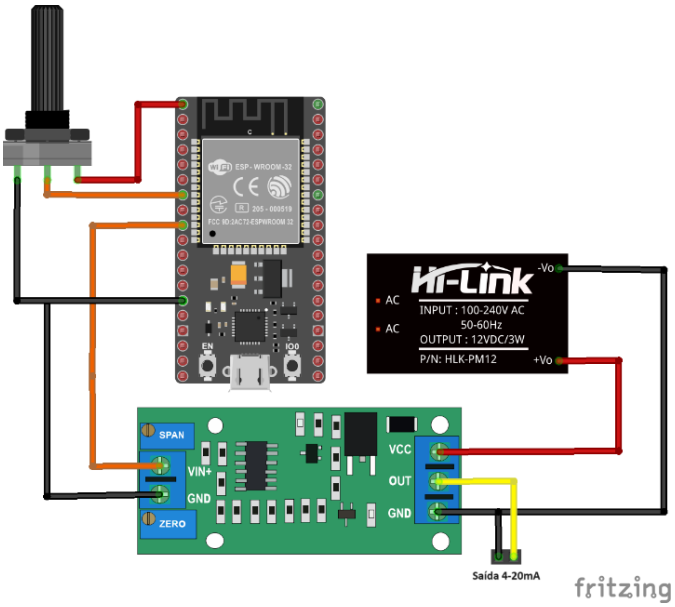


Fonte: KUPHALDT, 2009.

Essa configuração estabelece uma relação linear entre a tensão de entrada (1-5 V) e a corrente de saída (4-20 mA). Esse circuito também é conhecido como amplificador de transcondutância, onde a transcondutância, uma medida da razão entre a mudança de corrente e a mudança de tensão (ΔI / Δ V), é estabelecida pelo valor do resistor de 250 Ω.

Para este projeto, configuramos o módulo conversor de tensão para corrente ajustando os trimpots zero e span para realizar a conversão no intervalo de 0-3,3V para 4-20mA. O diagrama de conexão deste módulo está representado na Figura 19 abaixo.

Figura 19 - Diagrama de conexão do módulo conversor de tensão em corrente.



Fonte: Autoria Própria, 2024.

Conforme demonstrado no diagrama, para este módulo foi empregado um potenciômetro alimentado pela própria ESP32 e conectado a um canal ADC da ESP32, permitindo a variação da tensão no intervalo de 0 a 3,3V. Além disso, este módulo, assim como o anterior, está sendo alimentado pela mesma fonte AC/DC com entrada de 110/220V e saída de 12V/10W.

Como mencionado anteriormente, a ESP32 possui dois canais DAC (conversor digital para analógico) de 8 bits, conectados respectivamente ao GPIO25 (Canal 1) e GPIO26 (Canal 2). Cada canal DAC pode converter um valor digital de 0 a 255 em uma tensão analógica de 0 a Vref.

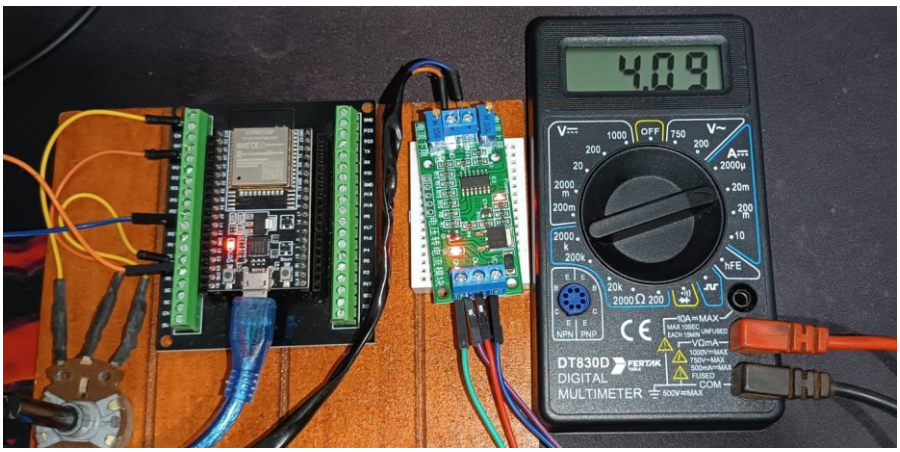
Dessa forma, enquanto o potenciômetro varia a tensão entre 0 e 3,3V para a ESP32, esses valores são enviados para o Canal 2 (GPIO26) do DAC, que está diretamente ligado à entrada de tensão do módulo conversor. Com o auxílio do potenciômetro, estamos variando a tensão de 0 a 3,3V na entrada do módulo e convertendo-a em corrente no intervalo de 4 a 20mA, onde 0V representa 4mA e 3,3V representa 20mA.

Um problema conhecido com o ESP32 DAC é que ele não atinge exatamente 0 volts quando a entrada é definida como 0 e pode não alcançar exatamente 3,3 volts quando a entrada é 255. É importante considerar essa limitação ao utilizá-lo para controlar dispositivos que requerem precisão absoluta em níveis de 0 ou 3,3 volts.

Para ajustar e testar este módulo de conversão, também utilizamos um multímetro digital. Nas Figuras 20 e 21, é possível observar a montagem física do circuito conforme

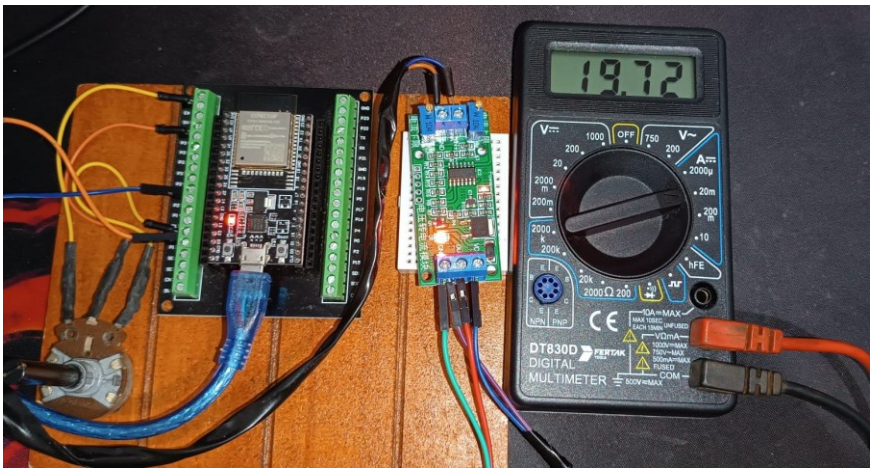
representado no Fritzing, juntamente com os valores de corrente de saída do módulo, tanto para o valor mínimo de 0V quanto para o valor máximo de 3,3V, respectivamente. O objetivo era obter 4mA para 0V e 20mA para 3,3V.

Figura 20 - Corrente de saída do módulo para 0V.



Fonte: Autoria Própria, 2024.

Figura 21 - Corrente de saída do módulo para 3,3V.



Fonte: Autoria Própria, 2024.

Assim, os valores obtidos nos testes estão de acordo com as expectativas dentro desse intervalo. Todos os resultados de testes foram registrados na seção de Resultados e Discussões do trabalho.

O módulo conversor de tensão para corrente (0-3,3V para 4-20mA) é interessante na indústria porque facilita conectar sensores que usam corrente de 4-20mA a dispositivos digitais como a ESP32. Isso é importante para monitorar e controlar processos industriais, como temperatura e pressão. Além disso, a ESP32 tem essa função útil de "Saída de Tensão Direta", onde você pode simplesmente enviar um valor de 0 a 255 no canal DAC da ESP32, e

a tensão determinada correspondente pode ser utilizada neste módulo. Essa tensão permanecerá lá até que o DAC seja chamado novamente (Nesse caso, variando pelo potenciômetro), o que permite um controle direto e fácil sobre a saída de tensão conforme necessário nos sistemas industriais. Essa versatilidade faz com que o módulo seja uma excelente escolha tanto para aplicações práticas na indústria quanto para fins educacionais.

3.3.4 Módulo amplificador de ganho de sinal LM358

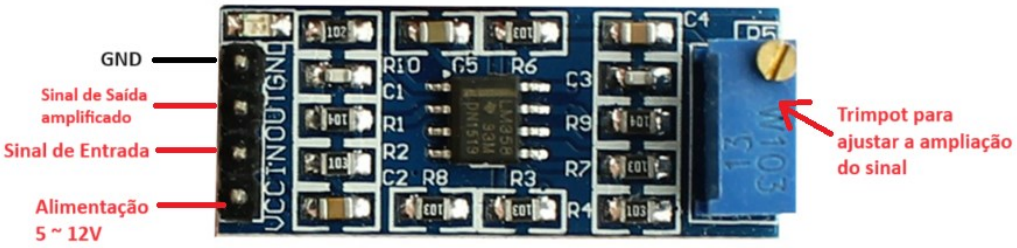
O amplificador operacional (op-amp) é um circuito integrado utilizado para amplificar sinais elétricos ou tensão. O ganho em um circuito de amplificador operacional é determinado pelos valores das resistências na rede associada. Essas resistências incluem o resistor de realimentação (RF) e o resistor de entrada (RE). Existem dois tipos principais de circuitos de amplificadores operacionais: os amplificadores inversores e os amplificadores não inversores.

Em um amplificador inversor, a tensão de saída possui uma fase oposta à da tensão de entrada aplicada ao amplificador operacional. Por outro lado, em um amplificador não inversor, a tensão de saída possui a mesma fase que a tensão de entrada. Um exemplo popular de amplificador operacional é o LM358, amplamente utilizado devido à sua capacidade de funcionar com fontes de alimentação simétricas ou assimétricas (PUTRANTO,2021).

Uma característica distintiva do Módulo Amplificador Operacional LM358 é sua integração em uma placa, formando um módulo completo que inclui um trimpot para ajustar a intensidade da amplificação. O LM358 é conhecido por seu tamanho compacto e suas características únicas, permitindo amplificar sinais com ganho de até 100 vezes.

Para funcionar, o Amplificador Operacional LM358, apresentado na Figura 22, requer uma tensão de entrada entre 5 e 12V DC e possui quatro pinos para conexão (VCC, IN, OUT, GND). Ele é ideal para amplificar o sinal de saída de sensores que produzem um sinal de baixa amplitude, melhorando a precisão e a sensibilidade em aplicações eletrônicas.

Figura 22 - Módulo Amplificador Operacional LM358.



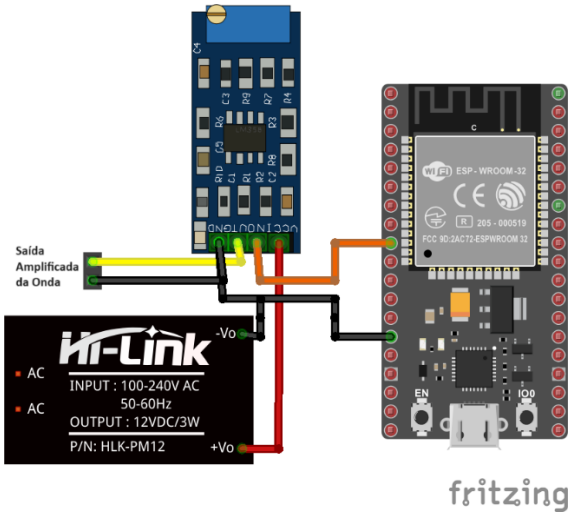
Fonte: Adaptado de SMART PROJECTS, 2024.

A saída DAC (Conversor Digital-Analógico) do ESP32 é limitada pela própria tensão de alimentação do ESP32, que é no máximo 3,3V. No entanto, é possível amplificar essa saída utilizando um módulo de amplificação, dependendo da fonte de alimentação do módulo utilizado.

Um uso comum do DAC é a geração de formas de onda, o que se encaixa bem com o ESP32 DAC. Embora não seja capaz de produzir formas de onda com a precisão de um equipamento de laboratório, pode ser bastante útil para gerar ondas simples e criar sons.

Neste projeto, iremos utilizar o módulo amplificador para aumentar a amplitude das ondas geradas e enviadas para o Canal 1 (GPIO25) do DAC. Vamos gerar diferentes tipos de ondas, como senoidais, quadradas, triangulares e em forma de dente de serra. O diagrama de conexão deste módulo está representado na Figura 23 abaixo.

Figura 23 - Diagrama de conexão do módulo amplificador LM358.



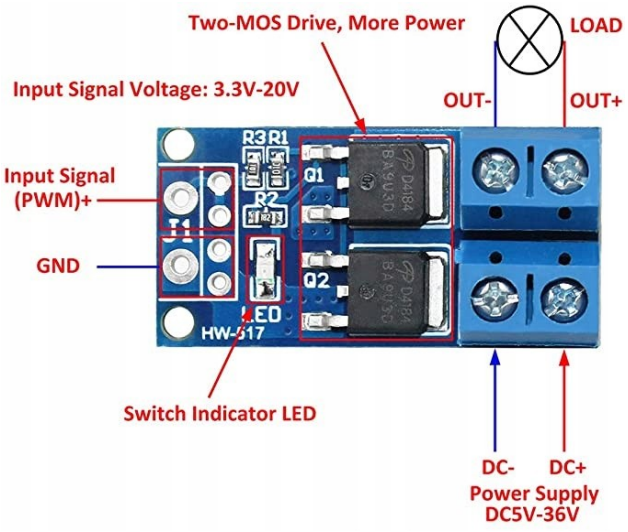
Fonte: Autoria Própria, 2024.

Conforme demonstrado no diagrama, este módulo sendo alimentado pela mesma fonte AC/DC com entrada de 110/220V e saída de 12V/10W. E está conectado a um canal ADC da ESP32.

O tipo de onda será selecionado ao clicar em um botão no painel de controle desenvolvido no Node-RED. E os testes foram realizados usando um osciloscópio, e os resultados foram registrados na seção de Resultados e Discussões do trabalho.

3.3.5 Módulo MOSFET PWM de Alta Potência

Figura 24 - Módulo MOSFET PWM de Alta Potência.



Fonte: Adaptado de USAINFO, 2024.

O Módulo Controlador PWM N-MOS MOSFET 30A 36V apresentado na Figura 24 foi projetado para proporcionar controle preciso sobre dispositivos de alta potência, como motores elétricos e cargas resistivas. Este dispositivo utiliza a técnica de modulação por largura de pulso (PWM) em conjunto com um transistor MOSFET de canal-N para ajustar a potência entregue à carga conforme necessário.

A frequência de operação do PWM deste controlador é ajustável na faixa de 0 a 20 kHz, o que oferece uma flexibilidade significativa para se adaptar às demandas específicas da aplicação. A capacidade de variar a frequência permite otimizar o desempenho do sistema em diferentes cenários, garantindo eficiência energética e controle suave sobre a carga.

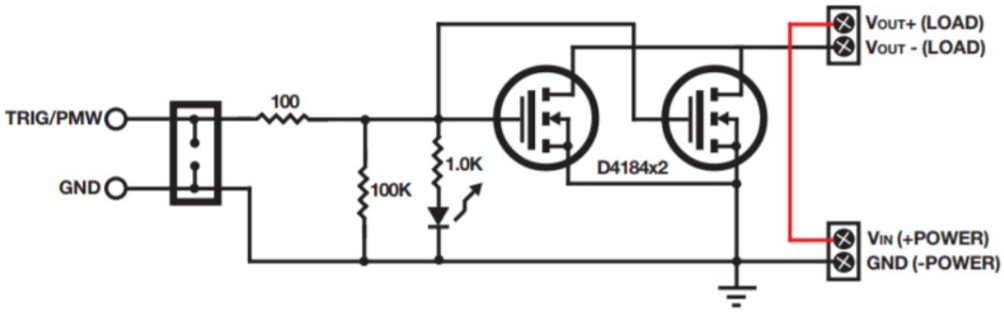
O MOSFET utilizado neste controlador é capaz de lidar com correntes de até 30A e tensões de até 36V, tornando-o adequado para uma ampla gama de aplicações industriais. Ele pode fornecer uma corrente estável de até 30A com o uso de um sistema de refrigeração do módulo ou 15A sem o uso de refrigeração. Com essa capacidade, o controlador PWM pode controlar motores DC de grande porte, sistemas de iluminação e outras cargas que demandam alta corrente. E a potência máxima de saída é de 400W.

Uma das características distintivas deste controlador é sua compatibilidade com diversas plataformas microcontroladas, como Arduino, ESP8266, ESP32 e Raspberry Pi, facilitando a integração em projetos eletrônicos complexos. Isso permite que o controlador

seja facilmente incorporado a sistemas de automação residencial, robótica, controle de motores, entre outros.

Além disso, o Controlador PWM N-MOS MOSFET 30A 36V possui características de proteção importantes, incluindo proteção contra sobretensão, curto-circuito e proteção térmica. Esses recursos garantem a segurança e a confiabilidade do sistema durante operações de alta potência. O diagrama do circuito deste módulo pode ser observado na Figura 25.

Figura 25 - Circuito do MOSFET PWM de Alta Potência



Fonte: Adaptado de PROTOSUPPLIES, 2024.

O módulo requer 3 conexões. Um sinal lógico para ligar/desligar os MOSFETS; uma fonte de alimentação DC para alimentar o dispositivo (carga) que está sendo controlado e, finalmente, a própria carga. Conforme ilustrado na Figura 24.

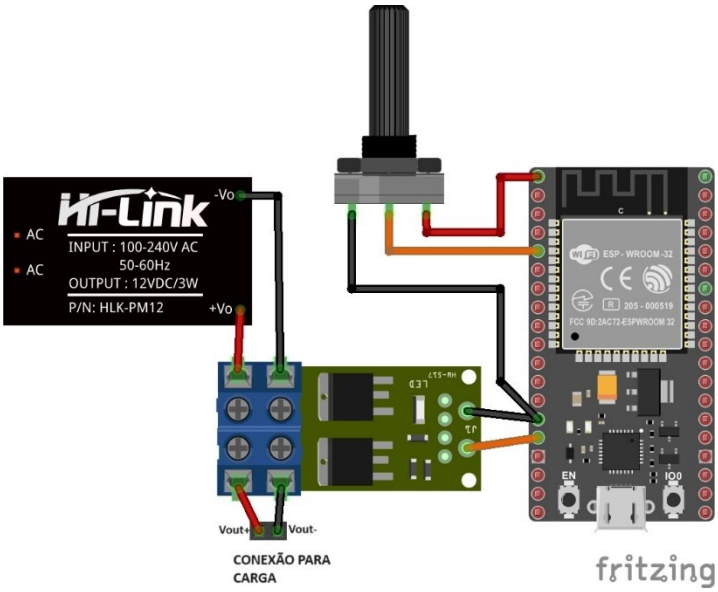
O conector J1 é para fazer a conexão do sinal lógico. Possui dois furos que suportam um terminal de parafuso de 2 posições, bem como quatro furos menores que podem suportar um cabeçote macho ou fêmea. Os pinos são identificados como TRIG/PWM na parte traseira da placa para o sinal e GND para o aterramento do sinal.

A fonte de alimentação de carga CC é conectada aos terminais de parafuso marcados com VIN+ / VIN- na parte traseira do módulo. O terminal positivo da fonte de alimentação se conecta ao VIN+ e a conexão de aterramento é conectada ao VIN-. As conexões de aterramento do sinal e de aterramento VIN são conectadas juntas no módulo.

A carga que está sendo acionada é conectada aos terminais de parafuso marcados OUT+/OUT- na parte traseira do módulo. O terminal positivo se conecta a OUT+ e o terminal negativo se conecta a OUT-.

Neste projeto, utilizaremos o Módulo MOSFET PWM conectado a ESP32 para estabelecer a conexão do sinal lógico no conector J1 do módulo. Um potenciômetro será empregado nesta configuração para variar o duty cycle gerado em uma saída PWM da ESP32. O diagrama de conexão deste módulo está representado na Figura 26 abaixo.

Figura 26 - Diagrama de conexão do módulo MOSFET PWM.



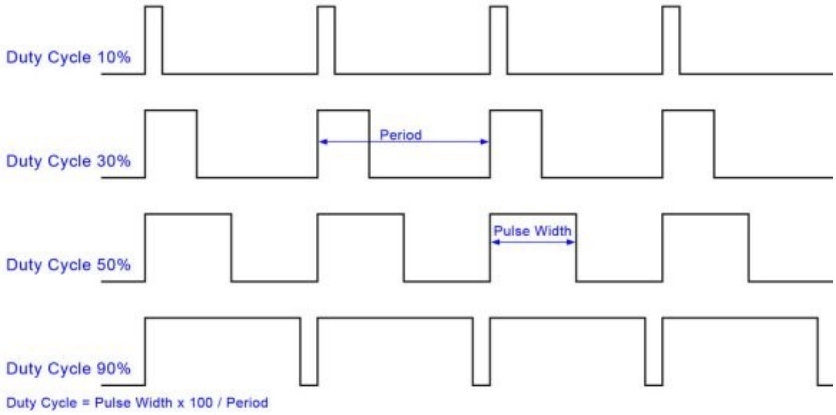
Fonte: Autoria Própria, 2024.

Conforme ilustrado no diagrama, este módulo utiliza um potenciômetro alimentado pela própria ESP32 e conectado a um canal ADC da ESP32, permitindo a variação do ciclo de trabalho em um canal PWM. Além disso, assim como os módulos anteriores, este também é alimentado pela mesma fonte AC/DC com entrada de 110/220V e saída de 12V/10W. A saída do módulo está disponível para conectar uma carga, na qual o Duty Cycle será variado com uma frequência específica.

A modulação por largura de pulso (PWM) é uma técnica amplamente utilizada em diversos meios, principalmente para controle de potência e velocidade. Com o PWM, podemos controlar a tensão média em um componente, e por consequência a potência, como um LED ou a velocidade de um motor DC.

A variação do Duty Cycle observada na Figura 27 é realizada ajustando a largura dos pulsos de forma a controlar a potência entregue à carga. Muitos microcontroladores, como o ESP32, possuem circuitos PWM integrados no chip, permitindo que seja especificada a frequência e a porcentagem do Duty Cycle diretamente na programação. Nesses casos, o PWM opera de maneira automática (OLIVEIRA, 2019).

Figura 27 – Exemplo de variação do Duty Cycle.

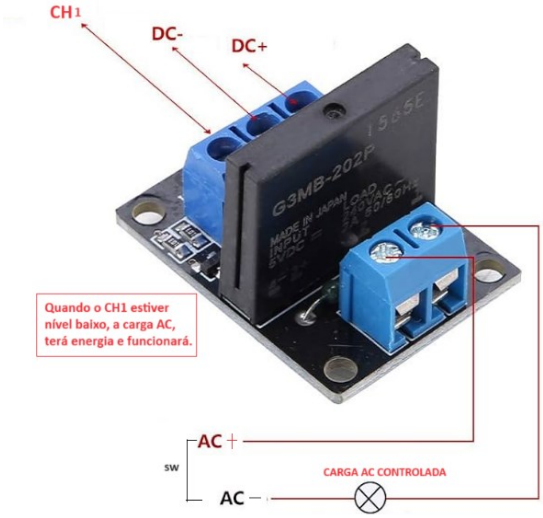


Fonte: MORAIS, 2017.

A configuração do código para este módulo está detalhada na seção Desenvolvimento de Firmware. O percentual de variação do Duty Cycle gerado no trigger PWM deste módulo poderá ser visualizado no painel de monitoramento desenvolvido no Node-RED. E os testes foram conduzidos utilizando um osciloscópio, e os resultados foram registrados na seção Resultados e Discussões do trabalho.

3.3.6 Módulo Relé de Estado Sólido (SSR)

Figura 28 - Módulo Relé de Estado Sólido (SSR)



Fonte: Adaptado de SOUZA, 2023.

Na Figura 28, apresentamos o Módulo Relé de Estado Sólido, uma solução versátil equipada com um relé de acionamento eletrônico capaz de controlar cargas de corrente alternada de 100V a 250V com até 2A.

Comparado aos relés eletromecânicos convencionais, o Relé de Estado Sólido SSR realiza o acionamento eletronicamente por meio de transistores, oferecendo maior velocidade de resposta, operação silenciosa e uma vida útil mais longa e eficiente (BRAGA, 2017). Sendo assim, a principal vantagem do relé de estado sólido em relação aos relés convencionais (eletromecânicos) é a ausência de componentes mecânicos em seu interior. Isso resulta em uma vida útil prolongada, menor tempo de chaveamento, ausência de problemas relacionados a interferências eletromagnéticas, além de não produzir ruídos durante o chaveamento.

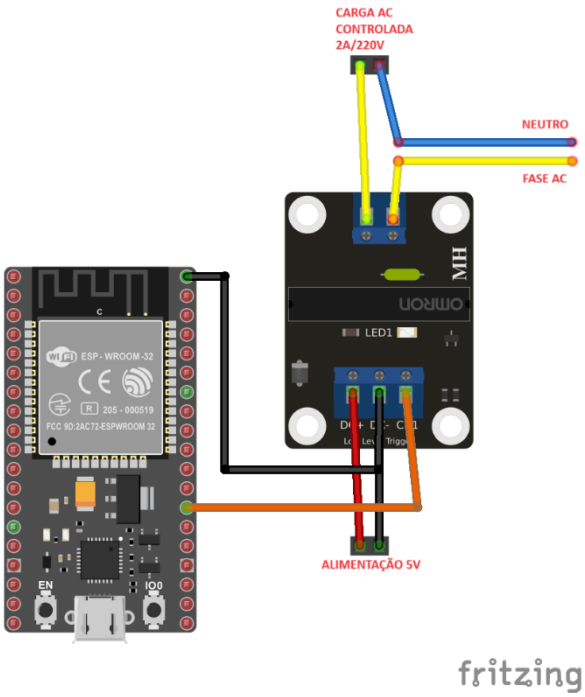
Funcionando como um interruptor controlado por microcontroladores, o Relé de Estado Sólido permite o controle remoto de praticamente qualquer dispositivo elétrico, respeitando os limites de operação do relé. Ele pode ser utilizado para ligar ou desligar eletrodomésticos, iluminação, motores e uma variedade de aparelhos elétricos.

Este módulo é particularmente útil em projetos de automação residencial e industrial, oferecendo simplicidade de integração com plataformas como Arduino, ESP8266 e ESP32 (SOUZA, 2023). Quanto à pinagem, o módulo de relé de estado sólido possui dois bornes: um com três pinos para alimentação 5V (DC+ e DC-) e comunicação com o microcontrolador (CH1), e outro com dois pinos para conexão da carga AC (SW1). Os detalhes dos pinos do módulo estão descritos na Figura 28. O LED indicador permite monitorar facilmente a atividade do relé. O módulo relé de estado sólido opera com nível lógico invertido, logo, para acioná-lo será necessário definir o pino CH1 como nível lógico baixo (0V – 1,5V = relé ligado) e para desligá-lo definir o pino CH1 como nível lógico alto (2,5V – 5V = relé desligado).

O Módulo Relé de Estado Sólido é uma ferramenta eficaz para incorporar funcionalidades de controle remoto e automação em diversos projetos eletrônicos, oferecendo confiabilidade e facilidade de uso para uma ampla gama de aplicações.

No contexto do KIT, este módulo funcionará de uma forma muito simples. O relé será acionado remotamente pelo ESP32 via MQTT Node-RED para controlar cargas de até 2A/220V. O diagrama de conexão deste módulo está representado na Figura 29 abaixo.

Figura 29 - Diagrama de conexão do módulo SSR.



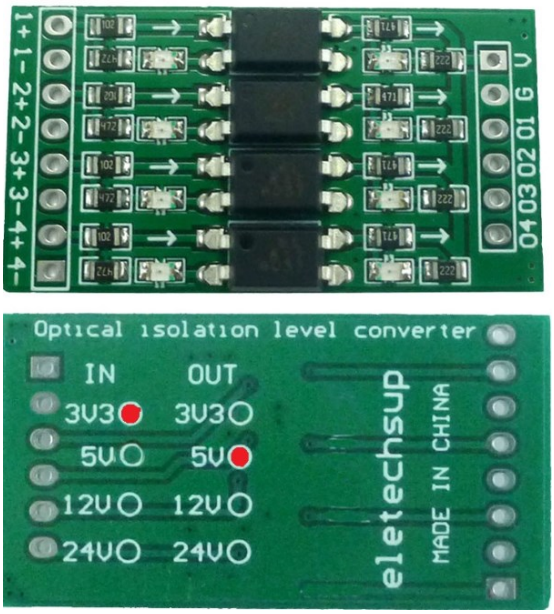
Fonte: Autoria Própria, 2024.

Conforme ilustrado no diagrama, de um lado foi conectado o circuito de controle, em uma saída digital. Na entrada DC alimentação 5V, o GND no GND da ESP32 e no pino CH1 um pino digital da ESP32, que irá controlar a carga. Do outro lado será conectado uma carga AC a ser controlada, como uma lâmpada por exemplo. É importante lembrar que este módulo funciona com nível lógico invertido, portanto se inserir nível lógico baixo (LOW) o relé irá ativar, e usando nível lógico alto (HIGH), o relé vai desativar.

Como dito antes, o acionamento da carga será feito através do painel de monitoramento desenvolvido no Node-RED. E os resultados dos testes foram registrados na seção Resultados e Discussões do trabalho.

3.3.7 Módulo Conversor de Nível Lógico Digital Com Isolamento Óptico

Figura 30 - Módulo Conversor de Nível Lógico Digital Com Isolamento Óptico (IN:3V3 OUT:5V)



Fonte: Adaptado de ELETECHSUP, 2024.

A Figura 30, apresenta o componente Módulo Conversor de Nível com Isolamento Óptico. Esses dispositivos são projetados para realizar duas funções principais:

- **Isolamento Óptico:** Utiliza acopladores ópticos (como LEDs e fotodetectores) para isolar eletricamente duas partes de um sistema. Isso é particularmente importante em ambientes industriais e em sistemas nos quais a separação elétrica é necessária para proteger componentes sensíveis contra interferências e surtos elétricos.
- **Conversão de Nível Lógico:** Convertem os sinais lógicos de um nível para outro. Por exemplo, podem converter sinais de 3,3V para 5V. Isso é útil quando você precisa integrar dispositivos ou componentes que operam em diferentes níveis de tensão.

Esse conversor de nível com isolamento óptico é capaz de trabalhar com saídas compatíveis tanto com transistores PNP quanto NPN e podem ser usados em situações em que é necessário proteger eletricamente circuitos, enquanto ao mesmo tempo se adaptam a diferentes níveis de tensão lógica. Aplicações comuns incluem ambientes industriais, sistemas de automação e qualquer lugar onde a interferência elétrica ou a variação nos níveis de tensão possam ser preocupações. Sua forma de operação é capaz de isolar dois circuitos eletrônicos

com total segurança por canal, mantendo um controle de comunicação entre ambas as partes, já que internamente não possui um contato elétrico e nem mesmo mecânico, apenas contato luminoso.

Em resumo, a utilização do Conversor de Nível Lógico é muito simples, podendo ser utilizado para realizar a interligação entre diferentes canais que trabalham com tensões diferentes. Entretanto, o conversor não é capaz de funcionar com sinais analógicos. E na Tabela 5 abaixo é descrito a pinagem deste módulo.

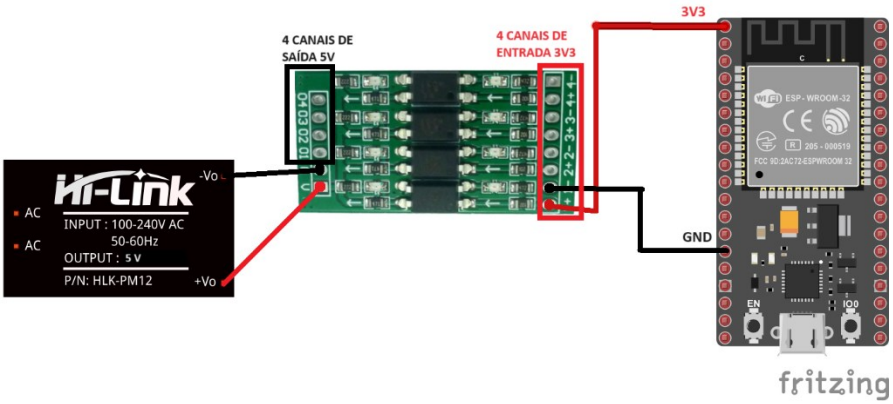
Tabela 5 - Terminal e definição dos pinos do módulo.

Função	**Rótulo**	**Instrução**
Entrada do sinal	1+	Primeira entrada do sinal ânodo
	1-	Primeira entrada do sinal cátodo
	2+	Segunda entrada do sinal ânodo
	2-	Segunda entrada do sinal cátodo
	3+	Terceira entrada do sinal ânodo
	3-	Terceira entrada do sinal cátodo
	4+	Quarta entrada do sinal ânodo
	4-	Quarta entrada do sinal cátodo
Alimentação	VCC	Alimentação VCC 5V
	GND	GND
Saída do sinal	O1	A primeira saída do sinal
	O2	A segunda saída do sinal
	O3	A terceira saída do sinal
	O4	A quarta saída do sinal

Fonte: Adaptado de ELETECHSUP, 2024.

No contexto do KIT, a utilização deste módulo será bastante simples e direta. Como a ESP32 gera sinais digitais de 3,3V e muitas aplicações requerem sinais de 5V, este módulo será empregado para realizar essa conversão. O diagrama de conexão deste módulo está representado na Figura 31 abaixo.

Figura 31 - Diagrama de Conexão do Módulo Conversor de Nível Lógico Digital Com Isolamento Óptico.



Fonte: Autoria Própria, 2024.

Conforme ilustrado no diagrama, a fonte de alimentação do módulo é de 5V, que deve ser igual à tensão de saída desejada. Este módulo realiza a conversão de 3,3V para 5V. Portanto, o VCC (alimentação) do módulo está conectado a 5V. O canal de entrada do módulo será diretamente conectado ao pino de saída de 3,3V da ESP32. É importante lembrar que este conversor não trabalha com sinais analógicos. Portanto, no código da ESP32, não haverá uma implementação lógica específica para este módulo, pois ele opera apenas com a conversão digital. No entanto, este módulo pode ser muito útil, podendo até mesmo servir como fonte de alimentação para o módulo SSR descrito anteriormente ou para a própria ESP32. Os resultados dos testes realizados com este módulo estão registrados na seção de Resultados e Discussões do trabalho.

3.3.8 Módulo Pré Amplificador passa-baixa Para Subwoofer.

Figura 32 - Módulo Pré Amplificador passa-baixa Para Subwoofer.

Fonte: Adaptado de STRX, 2024.

Na Figura 32, podemos observar um pré amplificador de filtro passa-baixa com ganho, que estará disponível no painel do KIT, embora não seja conectado ao ESP32.

Este dispositivo é alimentado por uma única fonte de alimentação de 10-24V DC e utiliza um circuito integrado (CI) para gerar um sinal de áudio de baixa frequência a partir de um sinal de áudio de entrada. O CI atua como um filtro passa-baixa, permitindo que apenas as frequências mais baixas do sinal de áudio passem, enquanto as frequências mais altas são filtradas e removidas. A placa também inclui dois potenciômetros: um para ajustar a frequência de corte do filtro e outro para ajustar o ganho. A frequência de corte é o ponto em que o filtro começa a remover as frequências mais altas. E faixa de frequência que este dispositivo trabalha é de 22 a 300 Hz.

Este pré-amplificador foi projetado para subwoofer, um tipo de alto-falante especializado na reprodução de frequências baixas, responsáveis pelos sons graves. Um pré-amplificador é um dispositivo que amplifica um sinal eletrônico para posterior amplificação ou processamento. Quando o sinal de entrada é forte o suficiente, o pré-amplificador requer pouco trabalho; no entanto, em casos de sinais fracos, como nas saídas de microfones, o pré-amplificador se torna essencial. Além de elevar o sinal a níveis adequados para processamento, os pré-amplificadores podem incluir funções adicionais, como equalização. Neste caso específico, eles também controlam o volume dos dois sinais de entrada do sistema (CAVALLI, 2018).

Este pré amplificador é ideal para o reforço de graves e melhorar a qualidade dos áudios. Algumas de das vantagens desse pré amplificador são:

- **Monitoramento em Tempo Real:** O painel frontal do subwoofer possui indicadores LED que permitem monitorar o status da energia em tempo real.
- **Amplificador Operacional Substituível:** A base do CI permite a substituição a qualquer momento, possibilitando experimentar os efeitos de diferentes amplificadores operacionais e melhorar a qualidade sonora.
- **Alta Capacidade Anti-interferência:** A placa demonstra uma forte capacidade de resistência à interferência, garantindo um funcionamento livre de ruídos próximos ao alto-falante mesmo na ausência de sinal de entrada.
- **Prevenção de Interferências:** A placa apresenta uma robusta capacidade de resistência à interferência, minimizando ruídos próximos ao alto-falante na ausência de sinal de entrada.

Sabendo que o pré-amplificador de filtro passa-baixa em questão foi projetado para amplificar e filtrar sinais de áudio em geral, oferecendo a capacidade de ajustar a frequência de corte do filtro e o ganho do sinal dentro de uma faixa de operação típica entre 22 a 300 Hz, suas propriedades podem ser aplicadas de maneira educativa em diversos domínios, incluindo a instrumentação industrial.

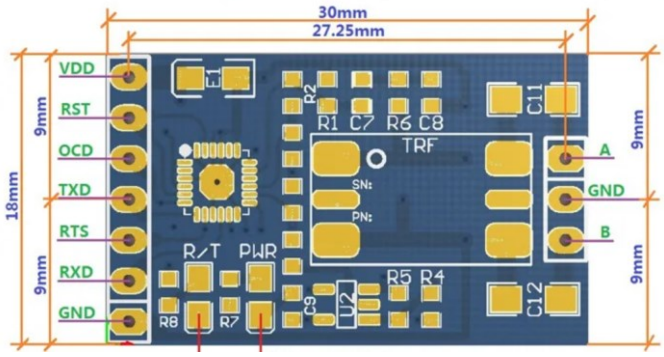
No contexto do desenvolvimento do KIT para a disciplina de Instrumentação Industrial, o pré-amplificador pode ser utilizado para introduzir e explorar as seguintes aplicações:

- **Filtragem de Sinais:** O pré-amplificador atua como um filtro passa-baixa, permitindo apenas a passagem de frequências abaixo de uma determinada frequência de corte. Isso é fundamental para demonstrar técnicas de filtragem de ruído e de sinais indesejados em sistemas industriais, onde a precisão e a qualidade dos dados são essenciais.
- **Aliasing e Nyquist:** Ao ajustar a frequência de corte do filtro, podemos abordar o conceito de aliasing, que ocorre quando sinais de alta frequência são erroneamente interpretados como frequências mais baixas devido à amostragem inadequada. O uso do pré-amplificador para ajustar a frequência de corte permite ilustrar como evitar problemas de aliasing e respeitar o teorema de Nyquist.
- **Amplificação e Condicionamento de Sinais:** O pré-amplificador também demonstra o papel essencial da amplificação na instrumentação industrial, especialmente quando lidamos com sinais de baixa amplitude que requerem amplificação antes do processamento ou da aquisição de dados.

Os resultados dos testes realizados com este módulo estão registrados na seção de Resultados e Discussões do trabalho.

3.3.9 Modem HART DS8500

Figura 33 - Especificações do Modem HART DS8500



Fonte: GARCIA, 2019.

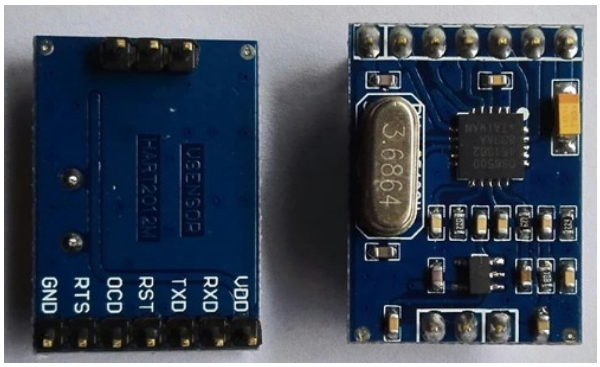
Na Tabela 6 é descrito o tipo e a função de cada pino do Modem HART apresentado na Figura 33.

Tabela 6 - Descrição dos pinos do Modem HART DS8500.

Pino	Tipo	Função
RXD	Entrada Digital	Entrada de dados série digital. Vindo da saída do demodulador (terminal d_out ds8500).
TXD	Saída Digital	Saída de dados série digital. Entrada para o modulador (terminal ds8500 d_in).
RST	Entrada/Saída	Reset ativo para baixo. É retirado como saída quando ocorre uma condição de reinicialização interna.
OCD	Saída	Detecção de portadora. Um status lógico alto indica uma detecção de portadora válida em fsk_in.
RTS	Entrada	Requisito para enviar, quando definido para alto, o dispositivo está colocado no modo demodulador. Um estado logic low coloca o dispositivo no modulador.
VDD	Alimentação	Alimentação de tensão digital 3,3 Vcc.
GND	Terra Digital	A 0V.
FSK_IN	Entrada Hart	Entrada analógica fsk, recebe sinal do loop de corrente de 4-20mA para o demodulador HART.
FSK_OUT	Saída Hart	Saída analógica fsk do modulador HART (frequências de 1200hz e 220hz) para o loop de corrente 4-20mA.

Fonte: Adaptado de GARCIA, 2019.

Figura 34 - Modem HART DS8500



Fonte: XINZHU TECH, 2024.

O DS8500 apresentado na Figura 34 é um modem HART que fornece modulação e demodulação FSK contínua em fase para aplicações de controle de processo. Este dispositivo é um modem de baixo consumo de energia rico em recursos que atende às especificações da camada física definidas pela HART Communication Foundation. O DS8500 possui muitos recursos que permitem ao usuário projetar de maneira fácil e eficaz um sistema de controle de processo que requer um modem HART (ANALOG DEVICES, 2010). Alguns dos recursos são:

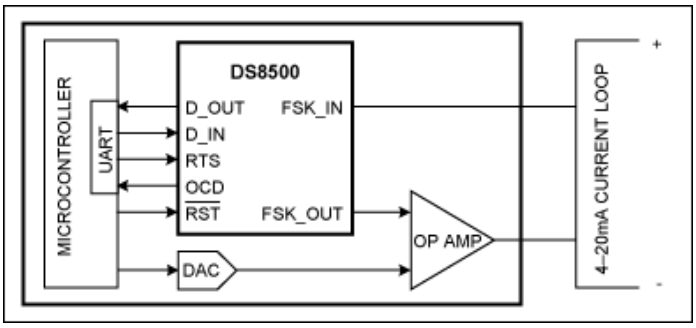
- Detecção de sinal confiável
- Poucos componentes externos
- Sinal de saída sinusoidal
- Baixo consumo de energia
- Cristal padrão de 3,6864 MHz
- Uma técnica avançada de processamento de sinal digital possibilita uma detecção confiável do sinal FSK_IN com pouca necessidade de componentes externos para separar um sinal HART do ruído. O sinal FSK_OUT é uma forma de onda senoidal que minimiza a distorção harmônica no sistema.

Além de tudo, o modelo utilizado neste projeto possui isolamento eletromagnético. Um chip esse isolamento possui recursos projetados para proteger o circuito interno contra interferências externas, incluindo ruídos elétricos e campos magnéticos. Isso é alcançado por meio de técnicas como barreiras físicas, materiais dielétricos e o uso de componentes isolados, como optoacopladores ou transformadores. Essas medidas ajudam a garantir a integridade dos sinais, melhorar a precisão das operações e garantir a conformidade com

padrões de emissão eletromagnética, especialmente em ambientes onde a imunidade a interferências é crucial, como em aplicações de alta tensão ou ambientes industriais.

O protocolo HART requer que os sinais sejam comunicados em um formato UART específico de 11 bits: um bit de início, 8 bits de dados, um bit de paridade e um bit de parada. Os blocos, modulador e demodulador do DS8500 precisam se comunicar com um microcontrolador UART para atender aos requisitos do protocolo. A Figura 35 mostra um diagrama de blocos de nível superior do DS8500 em um transmissor de processo inteligente. O design destaca a interface entre o modem HART e outros ICs externos.

Figura 35 - Um transmissor de processo inteligente apresenta o modem DS8500 HART comunicando-se com um microcontrolador de sistema.



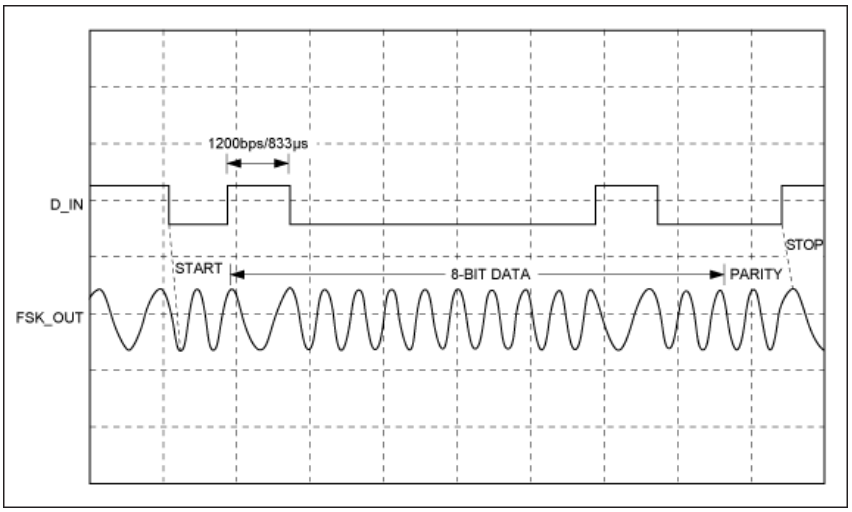
Fonte: ANALOG DEVICES 2010.

No modo demodulador, o DS8500 aguarda um sinal de início UART válido para sincronizar a comunicação de dados. A interface entre o modem HART e o microcontrolador também é mostrada na Figura 36. D_IN é a entrada de dados do sinal digital para o DS8500 que será modulado em um sinal FSK_OUT. D_OUT é a saída de dados do sinal digital do DS8500 que foi demodulado de um sinal FSK_IN. RTS recebe a solicitação do microcontrolador para iniciar o modo de demodulação (Rx) ou modulação (Tx) do modem.

O RST ativo-baixo fornece uma reinicialização ao DS8500 e garante que todos os registros internos e filtros iniciem a partir de um valor padrão conhecido. OCD é um sinal de detecção de portadora que determina um sinal FSK com amplitude válida na entrada do demodulador. Uma lógica alta no OCD indica que a amplitude do sinal FSK_IN é maior que 120mV; um valor lógico baixo indica que a amplitude do sinal FSK_IN é menor que 80mV ou que não há sinal de portadora. Opcionalmente, o microcontrolador pode fornecer um clock de 3,6864 MHz para o DS8500 (ANALOG DEVICES 2010).

A Figura 36 mostra o DS8500 no modo modulado onde D_IN é a entrada do modem e FSK_OUT é a saída modulada. Os dados são fornecidos em formato UART de 11 bits.

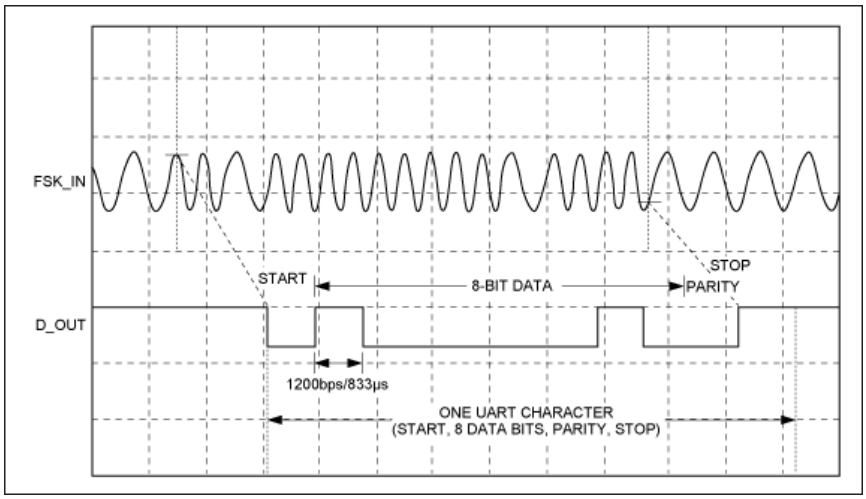
Figura 36 - Forma de onda do modulador (DS8500).



Fonte: ANALOG DEVICES 2010.

A Figura 37 mostra o DS8500 no modo demodulado onde FSK_IN é a entrada para o modem e D_OUT é a saída para o UART.

Figura 37 - Forma de onda do demodulador (DS8500).

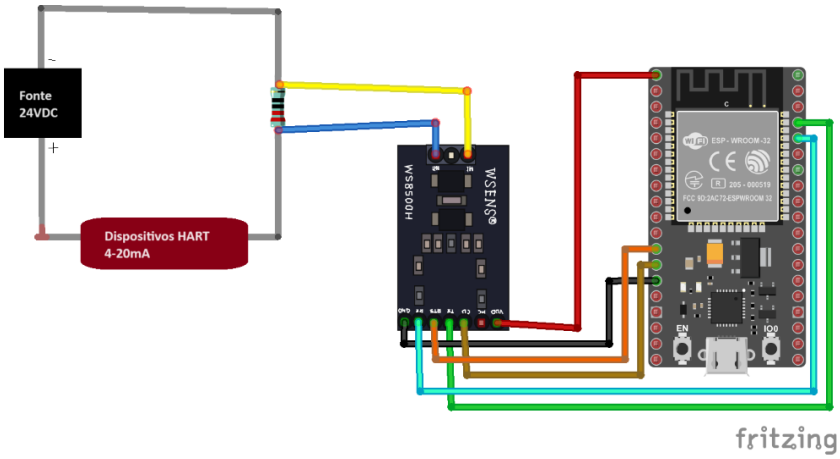


Fonte: ANALOG DEVICES 2010.

Neste projeto, para verificar a forma de comunicação do protocolo HART, foi desenvolvido um sistema que integra o HART DS8500, que atua como DCE (Equipamento de Comunicação de Dados), e o ESP32, utilizado como DTE (Equipamento Terminal de Dados).

O diagrama projetado para este sistema está ilustrado na Figura 38.

Figura 38 - Diagrama de conexão do Modem HART DS8500.



Fonte: Autoria Própria, 2024.

Na Tabela 6 foi descrito, cada um dos pinos do modem Modem HART DS8500. O dispositivo possui os seguintes terminais conectados:

- **VDD:** Tensão de alimentação contínua de 3,3V fornecida pelo ESP32.GND: Referência de tensão a 0V DC proveniente do ESP32.
- **TXD:** Entrada de dados em formato serial para o Modulador HART DS8500. Este terminal se conecta diretamente ao terminal TX do ESP32 no pino 1. A saída é em formato serial com velocidade de transmissão configurada para 1200 BPS para o protocolo HART.
- **RXD:** Saída de dados em formato serial do Demodulador HART DS8500. Este terminal se conecta diretamente ao terminal RX do ESP32 no pino 0.
- **OCD:** Indica detecção de portadora com dados direcionados para D_OUT. Este pino em dispositivos HART escravos permanece em nível baixo enquanto não há solicitação de informações por parte de um dispositivo Master HART. Quando uma mudança de nível baixo é detectada em OCD, os dados seriais provenientes do demodulador DS8500 são lidos no terminal D_OUT e encaminhados para o RX do ESP32.
- **RTS**: Requisito para envio de dados seriais para D_IN.
- **FSK_IN:** Entrada de dados analógicos FSK para o demodulador RX a partir do laço de corrente de 4-20mA.
- **FSK_OUT:** Saída do modulador TX para o laço de corrente de 4-20 mA.

3.4 Desenvolvimento do Firmware

Após a conclusão do desenvolvimento do kit didático, o próximo passo consiste na elaboração do firmware para a ESP32. Para essa etapa, utilizamos a plataforma Visual Studio Code com a extensão PlatformIO, que oferece um ambiente de desenvolvimento integrado (IDE) poderoso e flexível para programar e gerenciar projetos em microcontroladores. O uso do PlatformIO simplifica o processo de desenvolvimento de firmware para a ESP32, fornecendo ferramentas abrangentes de compilação, depuração e gerenciamento de bibliotecas. Essa abordagem permite uma programação eficiente e organizada, facilitando a implementação das funcionalidades necessárias para o projeto, como controle de sensores, comunicação Wi-Fi e interação com periféricos externos, como DACs e conversores analógico-digitais. O código completo está no Apêndice A deste documento.

Por fim, será detalhada a configuração realizada no Node-RED, onde foi desenvolvido um dashboard funcionando como um sistema SCADA. Essa integração permite a comunicação com a ESP32 por meio do protocolo MQTT, possibilitando visualizar e enviar informações aos módulos conectados.

3.4.1 Desenvolvimento de Código no Ambiente Visual Studio

Inicialmente, a ESP32 é configurada para se conectar à rede Wi-Fi local, o que é fundamental para possibilitar a comunicação sem fio com outros dispositivos na mesma rede. O código começa importando a biblioteca *WiFi.h* e define as credenciais da rede Wi-Fi, incluindo o nome (SSID) e a senha de acesso. Em seguida, a função *setup_wifi()* é definida para estabelecer a conexão Wi-Fi. Durante a execução dessa função, a ESP32 tenta se conectar à rede utilizando as credenciais fornecidas. O status da conexão é monitorado continuamente até que a conexão seja estabelecida com sucesso. Após a conexão bem-sucedida, o endereço IP local da ESP32 é impresso no console serial para fins de diagnóstico e monitoramento.

Essa etapa de conexão Wi-Fi é crucial para permitir a comunicação remota da ESP32 com um servidor MQTT e para o controle dos elementos por meio do dashboard Node-RED. Essa funcionalidade é essencial em aplicativos IoT que demandam interação e controle remoto. A configuração descrita pode ser visualizada na Figura 39, que exemplifica essa parte fundamental do processo de comunicação e configuração da ESP32.

Figura 39 - Configuração WiFi.

```
#include <Arduino.h>
#include <WiFi.h>

const char* ssid = "NomeDaRedeWiFi";
const char* password = "SenhaRedeWiFi";

void setup_wifi() {
  Serial.println();
  Serial.print("Conectando à rede ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  // Aguarda a conexão ser estabelecida
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Endereço IP: ");
  Serial.println(WiFi.localIP());
}

void setup() {
  Serial.begin(115200);
  setup_wifi();
}
```

Fonte: Autoria Própria, 2024.

A ESP32 foi configurada para estabelecer uma conexão com um servidor MQTT externo, como o test.mosquitto.org, utilizando o protocolo MQTT para comunicação bidirecional. Para facilitar essa comunicação MQTT, foi importada a biblioteca *PubSubClient.h*, conforme observado na Figura 40.

Figura 40 - Conexão e Comunicação MQTT.

```
#include <PubSubClient.h>
#include <WiFi.h>

const char* mqtt_server = "test.mosquitto.org";
const int mqtt_port = 1883;
WiFiClient espClient;
PubSubClient client(espClient);

void reconnect() {
  // Reconecta ao servidor MQTT em caso de desconexão
  while (!client.connected()) {
    Serial.print("Tentando se reconectar ao MQTT Broker...");
    if (client.connect("ESP32Client")) {
      Serial.println("Conectado");
      client.subscribe("tcc/#");  // Subscreve a todos os tópicos
relacionados ao TCC
    } else {
      Serial.print("Falhou, rc=");
      Serial.print(client.state());
      Serial.println("Tentando novamente em 5 segundos");
      delay(5000);
    }
  }
}
void callback(char* topic, byte* payload, unsigned int length) {
  // Função de callback para tratar mensagens MQTT recebidas
  Serial.print("Mensagem recebida [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}
void setup() {
  Serial.begin(115200);
  setup_wifi();

  // Configura o servidor MQTT e a função de callback
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}
```

Fonte: Autoria Própria, 2024.

A função *reconnect()* foi implementada para lidar com a reconexão automática ao servidor MQTT em caso de desconexão. Dentro desta função, a ESP32 tenta se reconectar de forma contínua até estabelecer uma conexão bem-sucedida. Uma vez conectada, a ESP32 se inscreve em um tópico MQTT geral (tcc/#) para receber mensagens de qualquer tópico relacionado ao projeto. Além disso, a função *callback()* é definida para lidar com mensagens MQTT recebidas. Quando uma mensagem é recebida, esta função é acionada para processar e responder à mensagem conforme necessário.

Essa comunicação MQTT desempenha um papel fundamental no sistema IoT implementado na ESP32, permitindo o controle remoto e o monitoramento eficaz do dispositivo. Essa funcionalidade possibilita uma interação dinâmica e em tempo real com a ESP32 por meio do servidor MQTT, agregando flexibilidade e eficiência ao sistema.

Na parte de controle do módulo conversor de tensão para corrente no código, a ESP32 gerencia a saída do DAC para produzir uma tensão que será convertida em corrente pelo módulo externo mencionado anteriormente. Um pino analógico conectado a um potenciômetro (potPin) é usado para ler um valor de tensão variável. Essa leitura de tensão é então convertida em um valor de 8 bits (0-255) adequado para o DAC e enviado ao módulo externo.

O módulo de conversão de tensão para corrente transforma essa entrada de tensão em uma corrente correspondente (4-20mA), que pode ser controlada pela ESP32 ajustando o valor enviado ao DAC. Esse controle possibilita ajustar dinamicamente a corrente de saída com base na leitura do potenciômetro, permitindo um controle preciso do sistema. O valor de tensão de saída é publicado via MQTT para fins de monitoramento remoto. A configuração do código pode ser observada na Figura 41.

Figura 41 - Controle do Conversor de Tensão para Corrente (DAC).

```
const int potPin = 39;   // Pino analógico conectado ao potenciômetro
const int modulo1 = 26;  // Pino DAC conectado ao conversor

void loop() {
  int potValue = analogRead(potPin); // Leitura do valor do potenciômetro
  float potVoltage = (3.3 / 4095.0) * potValue;  // Converte valor para tensão
(0-3.3V)

  client.publish("tcc/potenciometro", String(potVoltage).c_str());
  client.publish("tcc/potenciometro", String(potVoltage).c_str());

  dacWrite(modulo1, potVoltage / 3.3 * 255);  // Converte tensão para valor DAC
(0-255)

  delay(100);  // Pequeno atraso para estabilidade
}
```

Fonte: Autoria Própria, 2024.

Na figura 42 é mostrado a parte do código em que a ESP32 lê a tensão de saída do conversor de corrente para tensão e a publica via MQTT para fins de monitoramento remoto. Um pino analógico (modulo2) está conectado ao conversor de corrente para tensão, e a leitura desse pino é convertida em uma tensão correspondente (0-3.3V) utilizando uma escala de 12 bits. A ESP32 publica o valor da tensão lida no tópico MQTT "tcc/modulo2" para que possa ser monitorado e visualizado em tempo real por outros dispositivos conectados à mesma rede. Esse processo de leitura e publicação de dados é fundamental para obter informações sobre o estado do sistema e facilitar a tomada de decisões remotas.

Figura 42 - Controle do Conversor de Corrente para Tensão.

```
const int modulo2 = 32;  // Pino analógico conectado ao conversor

void loop() {
  // Leitura da tensão do módulo que converte corrente em tensão
  int voltage_value = analogRead(modulo2);
  float voltage_modulo2 = (3.3 / 4095.0) * voltage_value;  // Converte valor
para tensão (0-3.3V)

  // Publica o valor lido via MQTT no tópico "tcc/modulo2"
  client.publish("tcc/modulo2", String(voltage_modulo2).c_str());

  delay(100);  // Pequeno atraso para estabilidade
}
```

Fonte: Autoria Própria, 2024.

Na parte responsável pela geração das ondas no código, a ESP32 utiliza o DAC para produzir diferentes formas de onda. O objetivo é controlar a saída de um pino (no caso, o pino conectado ao módulo amplificador LM358, definido como onda) para gerar uma forma de onda específica.

O código utiliza uma tabela de formas de onda predefinidas (*WaveFormTable*) que são armazenadas como arrays de bytes. Cada forma de onda é representada por um conjunto de valores de amostra (de 0 a 255) que correspondem à amplitude da forma de onda em cada ponto. As formas de onda disponíveis incluem:

- **Onda Senoidal:** Representada por uma série de valores que formam uma curva suave e contínua, típica de uma onda senoidal.
- **Onda Triangular:** Caracterizada por uma série de valores que formam uma curva triangular ascendente e descendente.
- **Onda Dente de Serra:** Representada por uma sequência de valores que formam uma linha reta ascendente seguida por uma descida abrupta.
- **Onda Quadrada:** Consiste em uma sequência de valores que representam uma forma de onda retangular com bordas nítidas, alternando rapidamente entre os níveis máximo e mínimo.

A seleção da forma de onda é feita com base em mensagens MQTT recebidas pelo dispositivo. Quando uma mensagem é recebida indicando o tipo de onda desejado ("senoidal", "triangular", "dente-de-serra" ou "quadrada"), o código atribui o índice correspondente à tabela *WaveFormTable* à variável *wave_type*.

Dentro do loop principal (*loop()*), a saída do DAC é atualizada continuamente com os valores da forma de onda selecionada, avançando através da tabela de amostras para gerar a forma de onda desejada. Isso permite a geração dinâmica e controle em tempo real das formas de onda com base nos comandos recebidos via MQTT.

Esse mecanismo de controle da saída do DAC com formas de onda pré-definidas proporciona flexibilidade e capacidade de adaptação do dispositivo ESP32 para gerar diferentes sinais analógicos conforme necessário para aplicações de controle e instrumentação. Parte deste código está apresentado na Figura 43, porém o código completo está no Apêndice A.

Figura 43 - Controle do Módulo Amplificador de Ganho de Sinal para Geração de Funções.

```
// Tabela de ondas
#define Num_Samples  112   //  numero de amostras pra onda
#define MaxWaveTypes 4   // 4 tipos de onda
byte wave_type = 0; // Onda inicial: Seno
int i = 0;

dacWrite(onda, WaveFormTable[wave_type][i]);   // Saída da forma de onda
i++;
if (i >= Num_Samples) i = 0;
```

Fonte: Autoria Própria, 2024.

No controle do Módulo MOSFET PWM no ESP32, conforme mostrado na Figura 44, um potenciômetro é utilizado para ajustar o duty cycle do sinal PWM. O duty cycle determina a proporção do tempo em que o sinal está em nível alto durante cada período. Essa função é essencial para modular a potência entregue a um dispositivo ou circuito controlado, sendo uma parte importante da aplicação deste KIT didático.

No código abaixo, *output_pin* é o pino PWM da ESP32 que está conectado ao MOSFET responsável por controlar o sinal de saída. Já *potentiometer_pin* é o pino conectado ao potenciômetro, permitindo ao usuário ajustar o duty cycle do PWM manualmente. A leitura analógica do potenciômetro é realizada com analogRead(), retornando um valor entre 0 e 4095 correspondente à posição do potenciômetro. Esse valor é então mapeado para o intervalo desejado de duty cycle (0 a 1024) usando *map(),* o que determina a intensidade do sinal PWM.

Em seguida, o canal PWM é configurado com *ledcSetup()*, especificando a frequência do PWM (5000Hz neste caso) e a resolução de 10 bits. O valor calculado de duty cycle é então escrito no canal PWM com *ledcWrite(),* ajustando a saída do sinal de acordo com a posição do potenciômetro. E por fim, os valores do percentual do duty cycle e da frequência PWM são publicados via MQTT, permitindo a monitorização e o controlo remoto desses parâmetros.

Figura 44 - Controle do Módulo MOSFET PWM

```
const int output_pin = 14;          // Pino PWM da ESP32 conectado ao MOSFET
const int potentiometer_pin = 34;  // Pino conectado ao potenciômetro para
controlar o duty cycle

// Leitura do Potenciômetro e Configuração do PWM
int potValuePWM = analogRead(potentiometer_pin);
int dutyCycle = map(potValuePWM, 0, 4095, 0, 1024);

ledcAttachPin(output_pin, 0);  // Associa o canal PWM ao pino de saída
ledcSetup(0, 5000, 10);  // Configura o canal PWM com frequência de 5000Hz e
resolução de 10 bits
ledcWrite(0, dutyCycle);  // Define o valor do duty cycle

float percentage = (float)dutyCycle)/1024.0*100.0; //Percentual do duty cycle

client.publish("tcc/duty", String(percentage).c_str());
client.publish("tcc/freq", String(5000).c_str());
```

Fonte: Autoria Própria, 2024.

Na configuração acima, a frequência utilizada é de 5 kHz. É importante destacar que a frequência pode ser configurada em uma faixa que vai até 40 MHz. No entanto, ao optar por frequências mais altas, a resolução é limitada, podendo alcançar até 2 bits, como detalhado na Tabela 7 abaixo.

Tabela 7 - Possíveis configurações de frequências PWM na ESP32.

Frequência	**Resolução (máxima)**	**Duty Cycle**
1 Hz até 1 kHz	20 bits	0 a 100%
1 kHz a 100 kHz	13 bits	0 a 100%
100 kHz a 1 MHz	7 bits	0 a 100%
1 MHz a 20 MHz	3 bits	25%, 50%, 75% ou 100%
20 MHz a 40MHz	1 bit	50%

Fonte: Adaptado de OLIVEIRA, 2019.

A parte do código responsável pelo controle do módulo SSR está apresentado na Figura 45 e permite que o ESP32 acione ou desligue alguma carga AC de no máximo 2A/220V conectada ao relé SSR de forma remota via MQTT.

Primeiramente, é configurado o número do pino digital ao qual o módulo SSR está conectado, utilizando a variável *pinRele*. Em seguida, na função *setup(),* esse pino é

configurado como saída digital usando *pinMode(pinRele, OUTPUT),* indicando que será utilizado para enviar sinais de controle ao relé. No callback MQTT, implementado na função *callback(),* o código verifica se a mensagem recebida está no tópico "*tcc/carga*". Se essa condição for atendida, o código analisa o conteúdo da mensagem *(payload[0]).* Se o primeiro byte do payload for '0', significa que a carga deve ser ativada, e o relé é acionado através de *digitalWrite(pinRele, LOW),* considerando a lógica típica de ativação do SSR. Por outro lado, se o payload for '1', o código desliga a carga ao configurar o pino *pinRele* como HIGH através de *digitalWrite(pinRele, HIGH).* Após ativar ou desativar o relé SSR, uma mensagem de status é publicada de volta via MQTT no tópico "*status/carga*", informando se a carga foi acionada ou desativada.

Figura 45 - Controle do Módulo SSR.

```
const int pinRele = 5; // Pino de controle do relé SSR

void setup() {
    pinMode(pinRele, OUTPUT); // Configura o pino de saída para o relé SSR
}
void callback(char* topic, byte* payload, unsigned int length) {
    // Verifica se a mensagem recebida é relacionada ao controle do relé
    if (strcmp(topic, "tcc/carga") == 0) {
        // Liga ou desliga o relé baseado no payload recebido
        if (payload[0] == '0') {
            digitalWrite(pinRele, LOW); // Liga o relé
            client.publish("status/carga", "Carga acionada");
        } else if (payload[0] == '1') {
            digitalWrite(pinRele, HIGH); // Desliga o relé
            client.publish("status/carga", "Carga desativada");
        }
    }
}
```

Fonte: Autoria Própria, 2024.

Para o modem HART DS8500, foi desenvolvido um código de teste separado dos módulos anteriores devido à sua complexidade e ao diferente baudrate, visando verificar a comunicação serial. O principal objetivo desse código é validar a comunicação entre a placa ESP32 e o dispositivo HART. Ele implementa a comunicação serial entre a ESP32 e o modem DS8500 utilizando o protocolo HART. O código correspondente pode ser visualizado na Figura 46 abaixo.

Figura 46 - Código teste para o modem HART DS8500.

```
include <HardwareSerial.h>

#define UART_RX_PIN 3 // Pino RX da ESP32 conectado ao TX do DS8500
#define UART_TX_PIN 1 // Pino TX da ESP32 conectado ao RX do DS8500
#define RTS_PIN 14     // Pino RTS da ESP32 conectado ao RTS do DS8500
#define CD_PIN 12      // Pino CD do DS8500 conectado a um pino GPIO da ESP32
#define HART_BAUD_RATE 1200 // Baud rate do protocolo HART

HardwareSerial uart(1);

void setup() {
  Serial.begin(115200); // Inicia a comunicação serial para depuração
  uart.begin(HART_BAUD_RATE, SERIAL_8N1, UART_RX_PIN, UART_TX_PIN); //
Configura a UART da ESP32
  pinMode(RTS_PIN, OUTPUT); // Define o pino RTS como saída
  pinMode(CD_PIN, INPUT);   // Define o pino CD como entrada
}
void loop() {
  digitalWrite(RTS_PIN, HIGH); // Entra no modo de demodulação
  delay(100); // Tempo suficiente para o modem DS8500 se estabilizar

  // Exemplo de envio de dados HART
  uart.write(0x00); // Envia um byte de dados HART
  delay(1000); // Aguarda 1 segundo
  // Exemplo de recebimento de dados HART
  if (uart.available()) {
    int receivedByte = uart.read(); // Lê um byte recebido pela UART
    Serial.print("Byte recebido: ");
    Serial.println(receivedByte, HEX);
  }
  if (digitalRead(CD_PIN) == HIGH) {
    Serial.println("Sinal de portadora detectado.");
  } else {
    Serial.println("Nenhum sinal de portadora detectado.");
  }
  digitalWrite(RTS_PIN, LOW); // Sai do modo de demodulação
  delay(100); // Tempo suficiente para o modem DS8500 se estabilizar
}
```

Fonte: Autoria Própria, 2024.

Os pinos de comunicação, incluindo RX, TX, RTS e CD, são definidos para estabelecer a conexão física entre os dispositivos. A porta serial da ESP32 é configurada com uma taxa de transmissão (baud rate) de 1200 bps, conforme exigido pelo protocolo HART.

No método *setup(),* a comunicação serial é inicializada para depuração, e os pinos RTS e CD são configurados como saída e entrada, respectivamente, para controle e detecção de portadora.

No método *loop(),* o código entra no modo de demodulação ao configurar o pino RTS como alto, permitindo a comunicação com o modem DS8500. Em seguida, envia um byte de dados HART pela UART para o modem e aguarda um segundo antes de verificar se há dados recebidos disponíveis na porta serial. Se houver dados recebidos, eles são lidos e impressos no monitor serial. Além disso, o código verifica o estado do pino CD para detectar se há sinal de portadora do modem. Por fim, o código sai do modo de demodulação ao configurar o pino RTS como baixo.

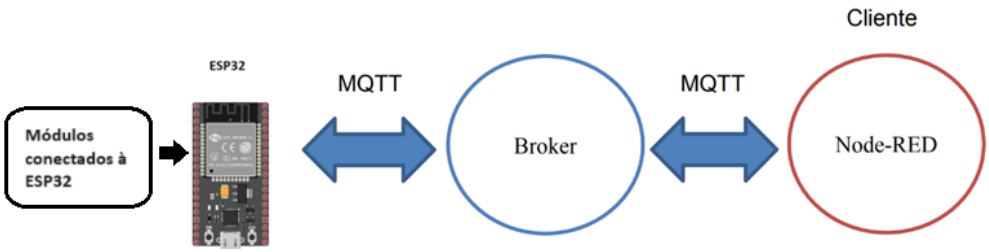
Essa abordagem possibilita o teste da comunicação serial com o modem DS8500 por meio da ESP32. Uma versão mais avançada deste código foi desenvolvida, permitindo o envio de frames via MQTT através da comunicação serial. Este código mais avançado pode ser encontrado no Apêndice B do documento.

3.4.2 Configuração e Implementação do Node-RED

A arquitetura do projeto é detalhada na Figura 47. Os sistemas operam como clientes, assim como o Node-RED, enquanto o Mosquitto Broker atua como intermediário na comunicação entre estes clientes, seguindo o modelo de publicação/assinatura do protocolo MQTT.

Optou-se pelo nível de Qualidade de Serviço (QoS) 2 para todas as mensagens MQTT transmitidas, considerando que serviços de automação residencial frequentemente operam em redes com pouca confiabilidade. Este sistema não foi concebido para lidar com mensagens duplicadas, e, portanto, o QoS 2 é essencial para assegurar que as mensagens sejam entregues corretamente e apenas uma vez, contribuindo para o desempenho consistente de todo o sistema (MARTINS, 2019).

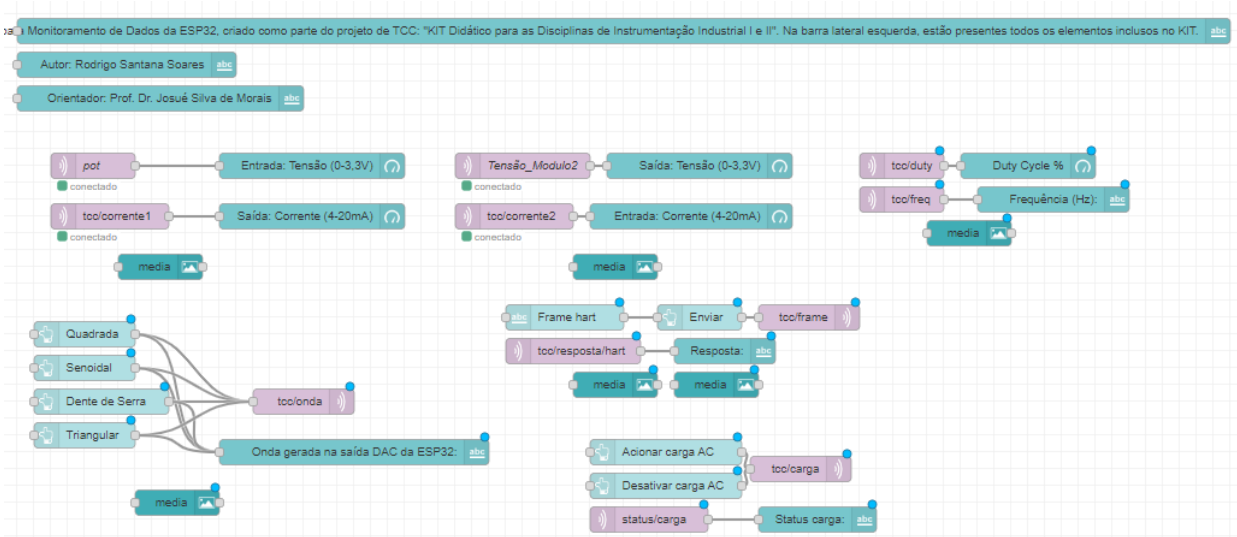
Figura 47 - Arquitetura do projeto



Fonte: Adaptado de MARTINS, 2019.

Após carregar o programa no ESP32, o próximo passo foi configurar o broker e a Interface de Usuário (UI). Com o Node-RED e o Mosquitto Broker em execução no sistema, utilizando os comandos "node-red" e "mosquitto -v" no prompt de comando, o Node-RED se conecta automaticamente ao broker. A UI do Node-RED pode ser acessada por um navegador web no endereço local do computador. Para acessar esta UI, foi instalado o node-red-dashboard no Node-RED, que serve como interface para visualizar os dados do sistema em um dashboard e interagir com ele. Isso é feito de acordo com a configuração dos fluxos previamente criados e carregados no Node-RED.

Figura 48 - Fluxo do Node-RED para o projeto.



Fonte: Autoria Própria, 2024.

Após isso, foi desenvolvido um fluxo utilizando Node-RED, como demonstrado na Figura 48, para gerenciar os diversos componentes do sistema. Quatro gráficos medidores foram configurados para receber dados de quatro nós de entrada MQTT, os quais se conectam aos módulos de conversão de tensão em corrente e corrente em tensão. Além disso, foram integrados quatro botões que representam diferentes tipos de onda, conectados a um nó de saída MQTT com o tópico "tcc/onda". O propósito é permitir que cada botão selecione um tipo de onda para a saída DAC da ESP32, conectada à entrada do módulo amplificador operacional. Um nó "output text" foi adicionado para exibir qual onda está sendo gerada.

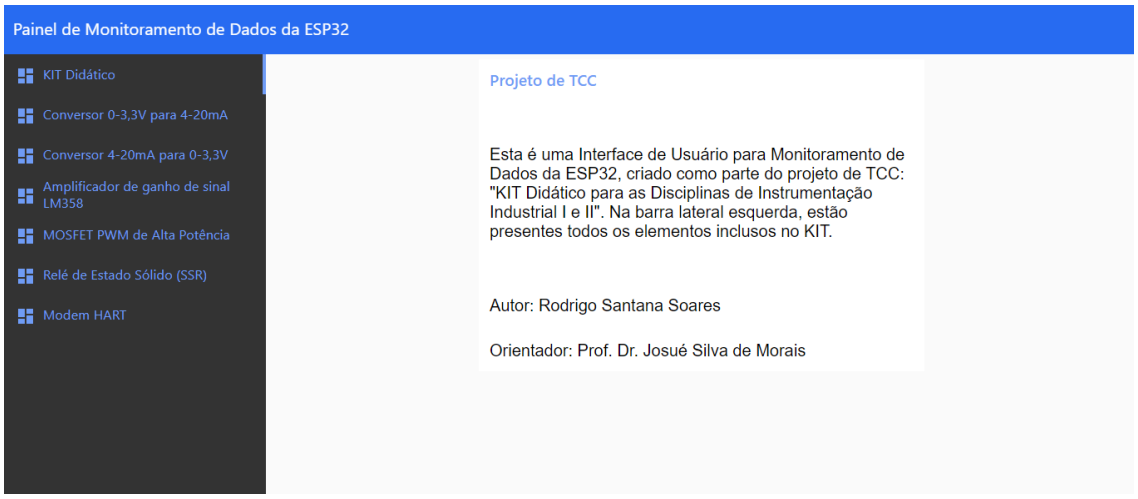
Para o módulo MOSFET PWM, um gráfico medidor conectado a um nó de entrada MQTT monitora a porcentagem do ciclo de trabalho (Duty Cycle) da saída PWM, enquanto um "output text" monitora a frequência do PWM.

No caso do relé de estado sólido, foram adicionados dois botões: um para ligar e outro para desligar a carga AC. Esses botões estão conectados a um nó de saída MQTT associado ao tópico "tcc/carga". Também foi incluído um "output text" para monitorar o status da carga, indicando se está ligada ou desligada.

Para o modem HART, foi adicionado um campo de texto (“text input”) conectado a um nó de saída MQTT para enviar frames de comando, e um output text conectado a um nó de entrada MQTT para receber a resposta do modem HART. Essa configuração permite controlar e monitorar o sistema de forma interativa por meio da interface do Node-RED.

A tela inicial do painel de monitoramento de dados da ESP32 criada com este fluxo está ilustrada na Figura 49.

Figura 49 - UI da tela inicial do painel de monitoramento.



Fonte: Autoria Própria, 2024.

Na barra lateral, é possível visualizar as páginas de monitoramento de cada módulo integrado com a ESP32, as quais estão registradas no tópico "Resultados e Discussões".

4 RESULTADOS E DISCUSSÕES

4.1 Considerações Iniciais

Nesta seção, apresentam-se os resultados dos testes realizados durante a pesquisa, seguidos por uma discussão sobre as descobertas e suas implicações.

4.2 Avaliação e Análise dos Módulos Implementados

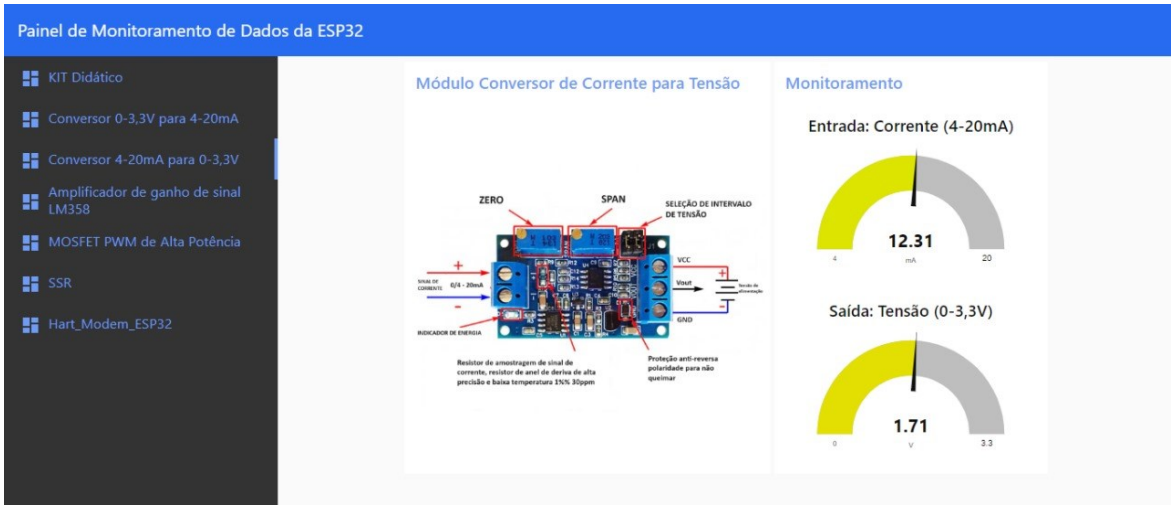
Para o módulo conversor de corrente para tensão (4-20mA para 0-3,3V), foram realizados testes de conversão utilizando os valores listados na Tabela 8 a seguir. Os resultados desses testes também são apresentados na tabela correspondente.

Tabela 8 - Resultados do conversor de corrente para tensão.

Sinal de Corrente	Tensão Obtida
4 mA	0 V
8 mA	0,65 V
12 mA	1,4 V
16 mA	2,2 V
20 mA	3,3 V

Fonte: Autoria Própria, 2024.

Figura 50 - UI da tela de monitoramento do módulo conversor 4-20mA para 0-3,3V.



Fonte: Autoria Própria, 2024.

Na Figura 50, é evidente o funcionamento da tela de monitoramento do módulo conversor 4-20mA para 0-3,3V. Observa-se que a comunicação via MQTT no ambiente

desenvolvido no Node-RED operou conforme o esperado: uma entrada de corrente de 12,31mA resultou em uma tensão de saída de 1,71V, alinhada com as expectativas. Ao modificar a corrente de entrada, a tensão correspondente também variou. Estes valores foram verificados com precisão por meio de um multímetro digital.

Os resultados obtidos com o módulo conversor de corrente em tensão HW-685 estão completamente alinhados com as expectativas estabelecidas, evidenciando uma configuração ideal do hardware e a eficácia do firmware desenvolvido.

Este módulo desempenha um papel essencial na instrumentação industrial ao converter sinais de corrente de 4-20mA em uma faixa de tensão de forma precisa e confiável. Sua aplicação em sistemas de automação permite a integração eficaz de dispositivos que operam com corrente em ambientes que exigem sinais de tensão, facilitando o controle e a medição de variáveis como temperatura, pressão, vazão e nível. A capacidade de comunicação entre dispositivos de diferentes tipos de saída é aprimorada pelo módulo, contribuindo para a implementação de sistemas de controle mais precisos e eficientes na indústria. Além disso, a integração com a ESP32 e a interface de usuário desenvolvida para visualização dos dados oferecem um valioso recurso de monitoramento para os processos industriais.

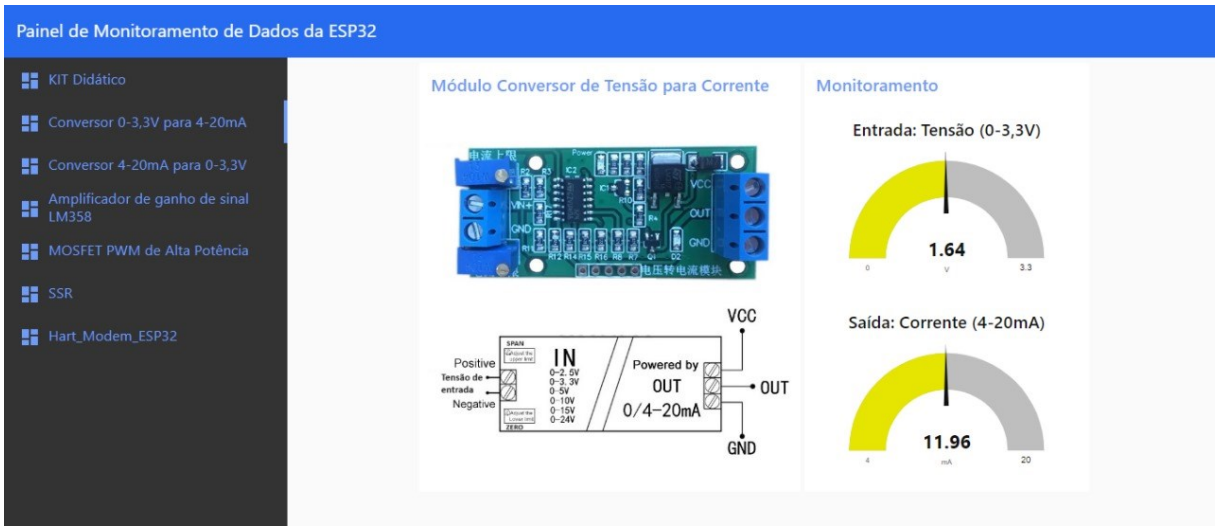
Com relação ao módulo conversor de tensão para corrente (0-3,3V para 4-20mA), é importante mencionar que a ESP32 enfrenta um problema conhecido em sua saída DAC. Quando o valor digital é 0, a tensão gerada pelo DAC não é exatamente 0V, e quando o valor digital é 255, a tensão gerada não atinge os 3,3V esperados. Os resultados dos testes realizados com este módulo estão apresentados na Tabela 9.

Tabela 9 - Resultados do conversor de tensão em corrente.

Valor DAC	**Tensão Esperada**	**Tensão Gerada**	**Corrente Obtida**
0	0 V	0,095 V	4,09 mA
64	0,825 V	0,871 V	8,22 mA
128	1,65 V	1,646 V	11,96 mA
192	2,475 V	2,421 V	15,70 mA
255	3,3 V	3,173 V	19,72 mA

Fonte: Autoria Própria, 2024.

Figura 51 - UI da tela de monitoramento do módulo conversor 0-3,3V para 4-20mA.



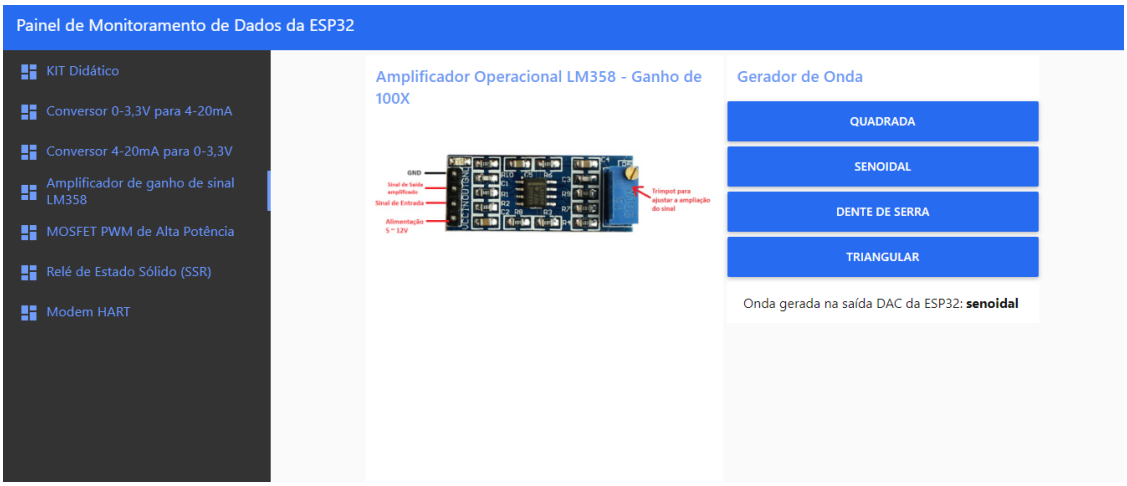
Fonte: Autoria Própria, 2024.

Na Figura 51, a interface da tela de monitoramento do módulo conversor 0-3,3V para 4-20mA é apresentada. A comunicação via MQTT no ambiente desenvolvido no Node-RED funcionou conforme esperado: uma entrada de tensão de 1,64V resultou em uma corrente de saída de 11,96mA, em conformidade com as expectativas. Modificando a tensão de entrada, a corrente correspondente variou de acordo. Esses valores foram precisamente confirmados por meio de um multímetro digital.

A consistência dos valores obtidos por meio deste módulo é fundamental para garantir resultados precisos e repetíveis em aplicações que demandam alta confiabilidade.

Ao utilizar o ESP32 DAC, é possível alcançar resultados consistentes, essenciais para projetos que exigem precisão na geração de sinais analógicos customizados na faixa de 0 a 3,3V. Isso é particularmente útil em aplicações como a geração de sinais de áudio de alta qualidade ou a produção de sinais analógicos de referência para calibração de sensores e circuitos externos. E graças à capacidade do módulo de converter tensão em corrente no padrão industrial de 4-20mA, ele pode ser utilizado em sistemas de automação e controle industrial para monitorar variáveis como temperatura, pressão e nível. Isso simplifica a comunicação entre dispositivos que operam com diferentes tipos de sinais, facilitando a integração e o desenvolvimento de sistemas mais eficientes e precisos na instrumentação industrial.

Figura 52 - UI da tela de monitoramento do módulo Amplificador LM358.

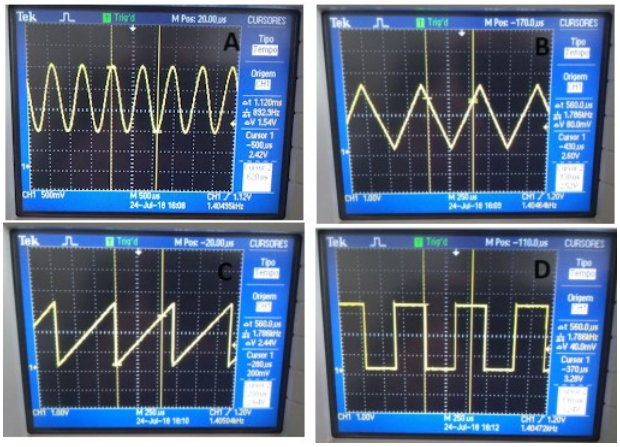


Fonte: Autoria Própria, 2024.

Na Figura 52, apresenta-se a interface da tela de monitoramento do módulo Amplificador Operacional LM35 com ganho de 100x. Observou-se que a comunicação MQTT de subscrição no ambiente Node-RED, onde o sistema envia informações para ESP32, foi um pouco mais lenta em comparação ao modo de publicação, no qual as informações são enviadas da ESP32 para o sistema. Essa lentidão pode ser atribuída ao uso de um servidor broker público, como o Mosquitto. Com um servidor MQTT dedicado, essa lentidão provavelmente não seria observada.

Conforme detalhado anteriormente neste trabalho, este módulo está conectado a outra saída DAC da ESP32. Todos os tipos de ondas foram verificados utilizando um osciloscópio. Na Figura 53, estão apresentadas as ondas resultantes da amplificação realizada por este módulo.

Figura 53 - Ondas resultantes do módulo amplificador: A)Senoidal; B)Triangular; C)Dente-de-serra; D)Quadrada.

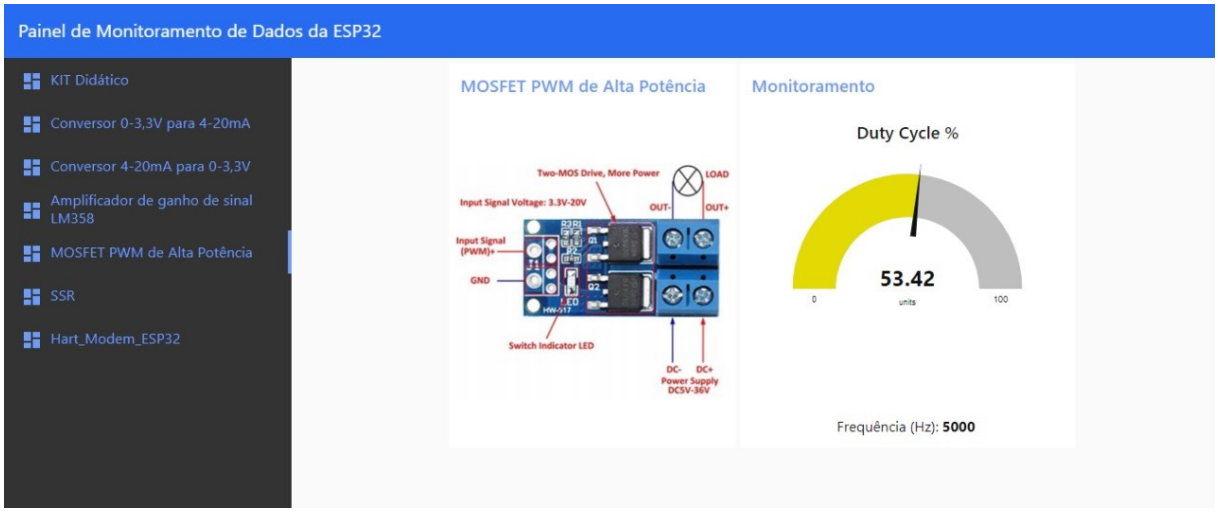


Fonte: Autoria Própria, 2024.

A interação entre o DAC da ESP32 e o Módulo Amplificador LM358 desempenha um papel muito interessante na criação de sinais de controle precisos e ajustáveis para acionamento de dispositivos ou sistemas. Por meio dessa configuração, é possível gerar sinais de onda com diversas amplitudes e frequências.

Sendo assim, o Módulo Amplificador LM358, quando em parceria com o DAC da ESP32, não apenas amplifica sinais de baixa amplitude para níveis adequados de processamento ou monitoramento, mas também possibilita a criação de formas de onda complexas e personalizadas. Essa capacidade é particularmente valiosa em aplicações eletrônicas que exigem controle detalhado sobre a forma e a amplitude dos sinais gerados.

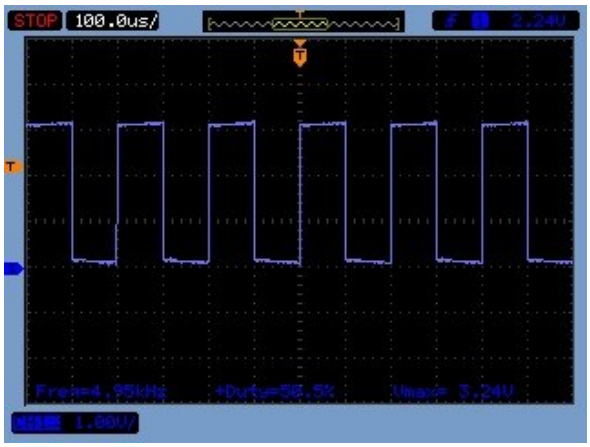
Figura 54 - UI da tela de monitoramento do módulo MOSFET PWM.



Fonte: Autoria Própria, 2024.

Na Figura 54, apresenta-se a interface da tela de monitoramento do módulo MOSFET PWM. Como mencionado anteriormente, este módulo está conectado a uma saída PWM da ESP32, e está sendo monitorado o ciclo de trabalho (Duty Cycle), o qual varia conforme o ajuste do potenciômetro e a frequência empregada. Para verificar a precisão dos resultados no sistema Node-RED, utilizou-se um osciloscópio. Na Figura 55, que corresponde à verificação no osciloscópio, é possível observar um Duty Cycle de 50% e uma frequência de 5 kHz gerados na saída PWM.

Figura 55 - Teste do Módulo MOSFET PWM no osciloscópio com 50% de Duty Cycle.



Fonte: Autoria Própria, 2024.

O uso de um módulo MOSFET PWM, presente no KIT didático, em um sistema de controle de potência oferece vantagens significativas devido ao seu controle preciso e eficiente. O MOSFET ajusta a potência média fornecida à carga variando a largura dos pulsos de sinal. Isso resulta em maior eficiência energética, reduzindo o desperdício de energia na forma de calor em comparação com métodos tradicionais. A capacidade de ajuste rápido e preciso torna o módulo MOSFET PWM ideais para aplicações com sistemas de motor.

Os resultados obtidos com este módulo confirmam a eficácia da configuração do KIT e do firmware desenvolvido, alinhando-se completamente com as expectativas estabelecidas. Este alinhamento demonstra como a implementação adequada deste sistema oferece uma solução eficaz para o controle de potência em várias aplicações, aproveitando a capacidade de alta potência desse módulo.

Figura 56 - UI da tela de monitoramento do módulo SSR.

Fonte: Autoria Própria, 2024.

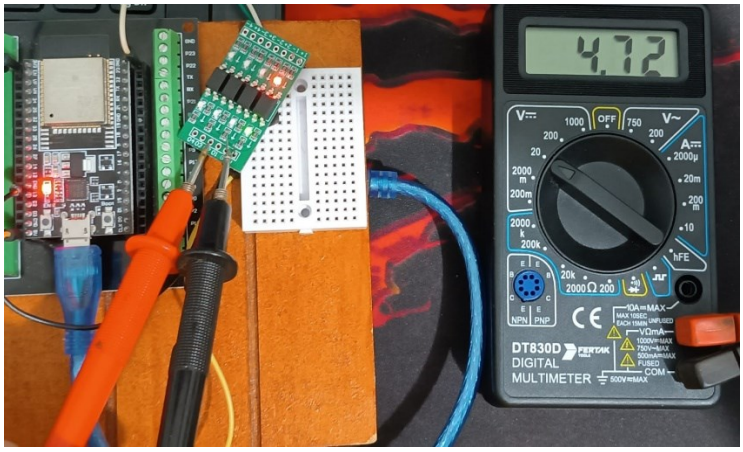
O módulo SSR foi configurado para ser acionado remotamente pelo ambiente criado no Node-RED, conforme apresentado na Figura 56. Enfrentamos um desafio semelhante ao do Amplificador LM358: a comunicação MQTT de subscrição no ambiente Node-RED, onde o sistema recebe informações da ESP32, é mais lenta devido ao uso de um servidor broker público. No entanto, apesar do tempo de espera prolongado, conseguimos ativar remotamente a carga, conforme evidenciado pelo LED indicativo de acionamento do relé.

Para otimizar a eficiência deste módulo no KIT, seria recomendável utilizar um botão físico para acionar e desativar a carga AC de até 2A/220V. Essa configuração não é difícil de implementar. Alternativamente, outra opção seria utilizar um servidor MQTT dedicado, o que poderia melhorar significativamente o desempenho e a responsividade do sistema.

Por fim, temos os módulos Conversor de Nível Lógico e o Pré Amplificador passa-baixa, que não estão integrados à lógica do código nem ao painel de monitoramento desenvolvido no Node-RED, mas fazem parte do KIT didático.

O Conversor de Nível Lógico está presente no KIT para converter o sinal de 3,3V da ESP32 para 5V, adequado para diversas aplicações. Na Figura 57, foi realizado um teste com um multímetro digital para validar essa conversão.

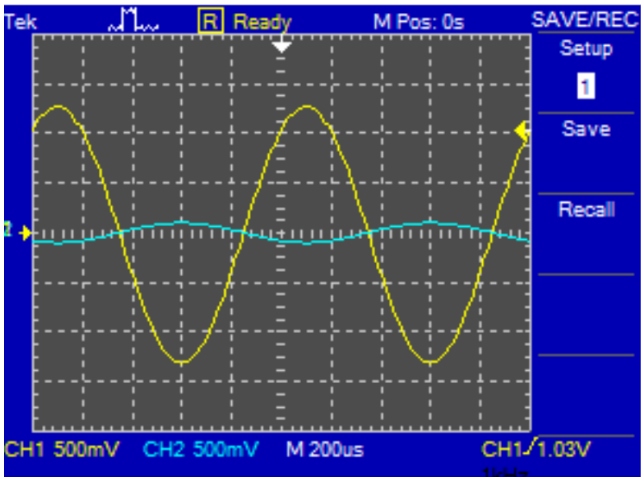
Figura 57 - Teste do Módulo Conversor de Nível Lógico Digital.



Fonte: Autoria Própria, 2024.

Quanto ao Pré Amplificador passa-baixa, embora não esteja conectado à ESP32, ele desempenha um papel importante no contexto do desenvolvimento do KIT para a disciplina de Instrumentação Industrial. E como já dito neste projeto, este pré-amplificador pode ser utilizado para explorar diversas aplicações, como Filtragem de Sinais, Aliasing, Amplificação e Condicionamento de Sinais, entre outras.

Figura 58 - Teste do Módulo Pré Amplificador passa-baixa.



Fonte: Autoria Própria, 2024.

Na Figura 58, podemos observar o pré amplificador em ação. Nele, aplicamos um sinal senoidal de entrada e podemos visualizar o sinal de saída com maior ganho para uma frequência de 1kHz. Este componente demonstra a sua utilidade prática ao facilitar a compreensão e o estudo de conceitos fundamentais em instrumentação e processamento de sinais.

4.3 Avaliação e Análise do Modem HART

Para avaliar o Modem HART, inicialmente foi utilizado o Cabo Conversor USB para TTL FT232RL de 6 vias ilustrado na Figura 59.

Figura 59 - Cabo Conversor USB para TTL FT232RL 6 Vias.

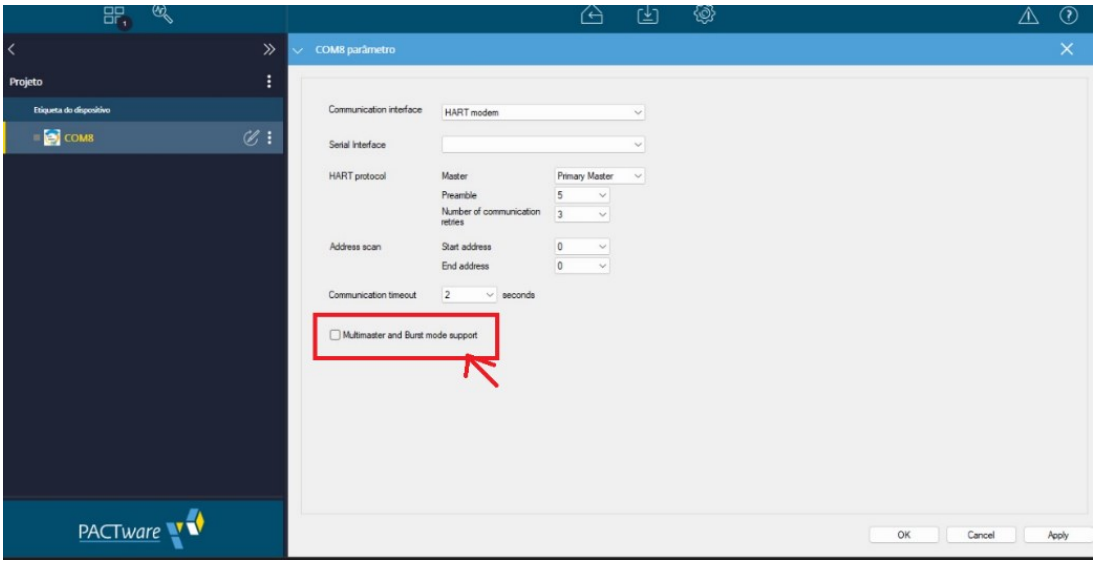
Fonte: Adaptado de SARAVATI, 2024.

O cabo serial USB para TTL, baseado no chipset FT232RL padrão, oferece uma solução eficiente para conectar dispositivos com interface serial de nível TTL ao USB. O cabo incorpora o FT232RQ USB da FTD para o dispositivo de interface Serial UART, gerenciando toda a sinalização e protocolos USB. Com uma placa de circuito eletrônico interna utilizando o FT232R, encapsulada na extremidade do conector USB, esses cabos oferecem versatilidade com uma variedade de conectores na outra extremidade para suportar diferentes aplicações.

O Modem HART foi testado e validado usando o cabo mencionado, por meio do software PACTware, uma interface aberta e independente para operação de dispositivos, sistemas e interfaces de comunicação. Este software é integrado à interface FDT (Field Device Tool), que é o padrão para unificação da interface entre dispositivo e usuário. A FDT permite uma rápida integração na operação do dispositivo, enquanto o PACTware facilita a configuração, operação e diagnóstico dos dispositivos de uma instalação de maneira simples e eficiente.

Durante o teste de comunicação do modem HART com um dispositivo HART utilizando o cabo serial USB para TTL, foi observado no PACTware que, para garantir a comunicação adequada, é crucial desmarcar a opção "Multimaster and Burst mode support" nos parâmetros da porta serial, como ilustrado na Figura 60. Sem desmarcar essa opção, a comunicação não é estabelecida.

Figura 60 - Tela do PACTware: Parâmetros da porta serial.

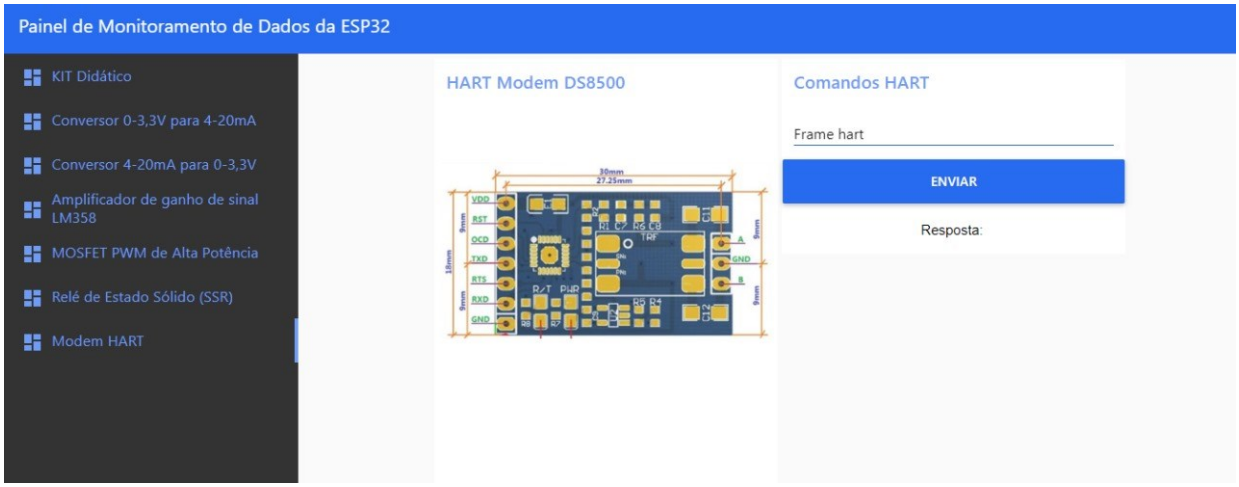


Fonte: Autoria Própria, 2024.

Essa observação sugere que o não funcionamento do código de teste para ESP32, ilustrado na Figura 46, com o Modem HART, pode ser atribuída a esta questão. Apesar de

terem sido conduzidos vários testes de comunicação entre a ESP32 e o Modem HART, utilizando diferentes configurações de código, não foi possível estabelecer qualquer comunicação entre a ESP32 e o Modem HART, nem por meio do PACTware, nem ao enviar quadros via MQTT pelo ambiente Node-RED, conforme delineado na Figura 61, como originalmente planejado para este projeto. É evidente que é necessário um período adicional de desenvolvimento para resolver as dificuldades de comunicação entre a ESP32 e o Modem HART.

Figura 61 - UI da tela de monitoramento Modem HART.



Fonte: Autoria Própria, 2024.

Assim, o cabo serial USB para TTL se destaca como a única solução viável no momento para a comunicação entre o modem e os dispositivos HART no KIT didático. Para projetos futuros, seria enriquecedor investigar mais profundamente a integração da ESP32 com este modem, permitindo o envio remoto de quadros de comando para os dispositivos HART. Essa ampliação poderia proporcionar uma maior flexibilidade e potencializar as capacidades do sistema.

5 CONCLUSÃO

Durante o desenvolvimento deste trabalho, concebeu-se e implementou-se um KIT didático destinado às disciplinas de Instrumentação Industrial I e II. A sinergia alcançada ao integrar a ESP32 com os módulos estudados, aliada à comunicação MQTT e à interface de usuário no Node-RED, permitiu a criação de um sistema repleto de funcionalidades inovadoras.

Os testes realizados evidenciaram resultados encorajadores, especialmente no que diz respeito ao desempenho da interface de monitoramento, que se mostrou eficaz na apresentação dos dados provenientes dos módulos. Contudo, identificaram-se desafios funcionais a serem superados, como a lentidão na transmissão de informações através do protocolo MQTT no ambiente Node-RED, atribuível ao uso de um servidor broker público. É plausível supor que com um servidor MQTT dedicado, tal lentidão não seria observada. Além disso, a integração do modem HART com a ESP32 não produziu os resultados esperados, apontando para a necessidade de pesquisas adicionais nessa área, dada a escassez de informações disponíveis. Dessa forma, o presente estudo abre novos caminhos para investigações futuras, destacando a importância de explorar alternativas para otimizar a comunicação MQTT e aprimorar a integração de dispositivos como o modem HART.

Apesar dos desafios enfrentados, a solução proposta continua representando uma alternativa de baixo custo e alta eficácia para o ensino educacional. Os dados coletados e armazenados pela ESP32 oferecem uma base sólida para análises subsequentes, permitindo a identificação de padrões, tendências e a otimização contínua de processos industriais. Além disso, a transmissão desses dados para plataformas externas viabiliza o monitoramento remoto e o controle em tempo real dos referidos processos.

No âmbito educacional, este projeto desempenha um papel crucial ao demonstrar os princípios fundamentais da instrumentação industrial. Ele proporciona aos alunos uma experiência prática valiosa na interpretação de sinais de sensores e na compreensão do papel desses dispositivos em ambientes industriais reais. No entanto, é importante ressaltar que ainda há espaço para implementações adicionais e aprimoramentos em futuros projetos, visando a contínua evolução e refinamento deste KIT didático.

REFERÊNCIAS BIBLIOGRÁFICAS

MAIER, Alexander; SHARP, Andrew; VAGAPOV, Yuriy. *Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things*. In: 2017 Internet Technologies and Applications (ITA). IEEE, 2017. p. 143-148. Citado na página 16.

BAZZO, Walter Anthony; PEREIRA, Luiz Teixeira do Vale. *Introdução à Engenharia*. 1ª ed. Florianópolis: Editora da UFSC, 2006. Citado na página 16.

SANTOS, Beatriz Correia et al. *Development of a didactic kit with the ESP32 microcontroller*. Seven Editora, 2023. Citado na página 17.

ESPRESSIF (SHANGHAI). ESP32. 2024. Disponível em: https://www.espressif.com/en/products/socs/esp32 .Acesso em: 7 jan 2024. Citado na página 17.

CNN BRASIL. *55,5% dos alunos desistem antes de completar o ensino superior, aponta relatório*. 2023, [S.l.], [s.d.]. Disponível em: https://www.cnnbrasil.com.br/nacional/555-dos-alunos-desistem-antes-de-completar-ensino-superior-aponta-relatorio/#:~:text=relat%C3%B3rio%20%7C%20CNN%20Brasil-,55%2C5%25%20dos%20alunos%20desistem%20antes%20de,completar%20ensino%20superior%2C%20aponta%20relat%C3%B3rio&text=O%20Mapa%20do%20Ensino%20Superior,antes%20de%20completar%20o%20curso. Citado na página 17.

GAMA, Bruna Borges de Oliveira. *Determinantes da evasão universitária e impacto no gasto público*. 2018. 137 f. Dissertação (Mestrado Profissional em Gestão Organizacional) - Universidade Federal de Uberlândia, Uberlândia, 2018. DOI http://dx.doi.org/10.14393/ufu.di.2018.579. Citado na página 17.

KAGERMANN, H., W. et al. (2013) *Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group*. Disponível em: http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderse iten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf. Citado na página 19.

SANTOS, M.; MANHÃES, A. M.; LIMA, A. R. *Indústria 4.0: Desafios e oportunidades para o Brasil*. In: SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO DE SERGIPE, 10., 2018, São Cristóvão, SE. Anais [...]. São Cristóvão, SE, 2018. p. 317 - 329. Citado na página 19.

FONSECA, Thiago dos Santos da. *Desenvolvimento e validação de métodos de sensoreamento visual aplicados à instrumentação de processos no contexto da indústria 4.0*. 2017. Citado na página 20.

COUTINHO, Thiago. *Entenda como a instrumentação industrial e Python atuam juntos na Indústria 4.0*. Voitto, 2021. Disponível em: https://www.voitto.com.br/blog/artigo/instrumentacao-industrial . Acesso em: 8 de abr 2024. Citado na página 21.

DA SILVA, Rogério Oliveira; ARAUJO, Warley Monteiro; CAVALCANTE, Maxwell Machado. *Visão geral sobre microcontroladores e prototipagem com arduino*. Tecnologias em Projeção, v. 10, n. 1, p. 36-46, 2019. Citado 2 vezes nas páginas 21 e 23.

CUNHA, Alessandro F. *O que são sistemas embarcados*. Saber Eletrônica, v. 43, n. 414, p. 1-6, 2007. Citado na página 22.

PENIDO, Édilus de Carvalho Castro; TRINDADE, Ronaldo Silva. *Microcontroladores*. Ouro Preto: Instituto Federal de Educação, Ciência e Tecnologia, 2013. Citado na página 22.

ESPRESSIF. ESP32-DevKitC V4 Getting Started Guide. *ESP-IDF Programming Guide latest documentation*, 2024. Disponível em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html#start-application-development. Acesso em: 9 abr. 2024. Citado na página 23.

BABIUCH, Marek; FOLTÝNEK, Petr; SMUTNÝ, Pavel. *Using the ESP32 microcontroller for data processing*. In: 2019 20th International Carpathian Control Conference (ICCC). IEEE, 2019. p. 1-6. Citado na página 24.

DIAS, M. (2022). *ESP32 Pinout: Saiba Tudo Sobre A ESP!* Disponível em: https://lobodarobotica.com/blog/esp32-pinout/ . Acesso em: 8 de abr 2024. Citado na página 26.

FIELDCOMM GROUP. *HART Technology Explained*. Disponível em: https://www.fieldcommgroup.org/technologies/hart/hart-technology-explained. Acesso em: 10 abr. 2024. Citado 2 vezes nas páginas 31 e 32.

SEIXAS FILHO, Constantino. *Introdução ao Protocolo HART*. 2007. Citado na página 33.

DINCULEANĂ, Dan; CHENG, Xiaochun. *Vulnerabilidades e limitações do protocolo MQTT usado entre dispositivos IoT*. Ciências Aplicadas , v. 9, n. 5, pág. 848, 2019. Citado na página 35.

MQTT ORG, MQTT - *The Standard for IoT Messaging*. Mqtt.org. Disponível em: https://mqtt.org/ . Acesso em: 10 abr. 2024. Nenhuma citação no texto.

BAIG, Mirza Jabbar Aziz et al. *Design and implementation of an open-Source IoT and blockchain-based peer-to-peer energy trading platform using ESP32-S2, Node-Red and, MQTT protocol*. Energy reports, v. 7, p. 5733-5746, 2021. Citado na página 37.

FRITZING ORG. Fritzinf. *Welcome to Fritzing*. Disponível em: https://fritzing.org/ . Acesso em: 11 abr. 2024. Nenhuma citação no texto.

PACTWARE. *Configuration software for field instruments | PACTware*. Disponível em: https://pactware.com/. Acesso em: 11 abr. 2024. Nenhuma citação no texto.

NODE-RED. Nodered.org. Disponível em: https://nodered.org/ . Acesso em: 10 abr. 2024. Nenhuma citação no texto.

SHENZHEN HI-LINK ELECTRONIC Co., Ltd. Hlktech.net. Disponível em: https://www.hlktech.net/. Acesso em: 22 abr. 2024. Citado na página 40.

CASA DA ROBÓTICA. *Módulo Conversor de Corrente para Tensão 4 a 20 mA para 0-3,3 V / 5 V / 10 V*. Casa da Robótica. Conversores. Disponível em: https://www.casadarobotica.com/sensores-modulos/modulos/conversores/modulo-conversor-de-corrente-para-tensao-4-a-20ma-para-0-3-3v-5v-10v . Acesso em: 11 abr. 2024. Nenhuma citação no texto.

ALKASAP, Aseel Y.; HAGEM, Rabee M. *Low Cost Portable System for Converting Mosul Electrical Substations to Smart One's*. Al-Rafidain Engineering Journal (AREJ), v. 26, n. 2, p. 323-339, 2021. Citado 2 vezes nas páginas 42 e 47.

TURAIS TECH. *HW-685 Sensor de Corrente 4-20mA*. Turais Tech Docs. Disponível em: https://docs.turais.de/docs/modules/hw685/. Acesso em: 11 abr. 2024. Nenhuma citação no texto.

SMARTKITS. *Gerador de Sinal Ajustável - 0 a 10V / 4 a 20mA*. Tray Tecnologia. Disponível em: https://www.smartkits.com.br/gerador-de-sinal-ajustavel-0-a-10v-4-a-20ma. Acesso em: 12 abr. 2024. Nenhuma citação no texto.

ALAVALA, Chennakesava R. *Principles of Industrial Instrumentation and Control Systems. Cengage Learning Services*, 2009. Citado na página 45.

CASA DA ROBÓTICA. *Módulo Conversor de Tensão 0 a 5V para Corrente 4 a 20mA.* Casa da Robótica - Loja de Robótica e Conteúdo Educacional. Disponível em: https://www.casadarobotica.com/sensores-modulos/modulos/conversores/modulo-conversor-de-tensao-0-a-5v-para-corrente-4-a-20ma?parceiro=3259 . Acesso em: 12 abr. 2024. Nenhuma citação no texto.

KUPHALDT, Tony. *Lessons in electric circuits*, volume iii–semiconductors. 2009. Citado na página 47.

SMART PROJECTS. *Módulo Amplificador Operacional LM358 Ganho 100X*. Disponível em: https://www.smartprojectsbrasil.com.br/modulo-amplificador-operacional-lm358-ganho-100x . Acesso em: 15 abr. 2024. Nenhuma citação no texto.

PUTRANTO, Ari Bawono et al. *Transforme todos os diodos de qualidade com ESP32 e Amplificador Op LM358 baseado em Android*. Ultima Computing: Jurnal Sistem Komputer, v. 1, pág. 22-29, 2021. Citado na página 50.

USAINFO. *Controlador PWM N-MOS Mosfet 30A 36V*. [S.l.], Usinainfo, [s.d.]. Disponível em: https://www.usinainfo.com.br/driver-para-motor/controlador-pwm-n-mos-mosfet-46a-30v-5506.html. Acesso em: 21 abr. 2024. Nenhuma citação no texto.

PROTOSUPPLIES. *High-Power Dual MOSFET Switch Module*. [S.l.], ProtoSupplies, [s.d.]. Disponível em: https://protosupplies.com/product/high-power-dual-mosfet-switch-module/. Acesso em: 22 abr. 2024. Nenhuma citação no texto.

OLIVEIRA, Jailson. *PWM – ESP32*. [S.l.], XProjetos, 2019 [s.d.]. Disponível em: https://xprojetos.net/pwm-esp32/. Acesso em: 22 abr. 2024. Citado na página 54.

MORAIS, José. *Controle de potência via PWM - ESP32* | Portal Vida de Silício, 2017. Portal Vida de Silício. Disponível em: https://portal.vidadesilicio.com.br/controle-de-potencia-via-pwm-esp32/. Acesso em: 22 abr. 2024. Nenhuma citação no texto.

SOUZA, Jonas. *Como utilizar o modulo relé/relay Estado Sólido 5v com Arduino – Blog da Robótica*, 2023. Disponível em: https://www.blogdarobotica.com/2023/01/10/como-utilizar-o-modulo-rele-relay-estado-solido-5v-com-arduino/ . Acesso em: 23 abr. 2024. Citado na página 56.

BRAGA, Newton C. *Relés: Circuitos e aplicações*. Editora Newton C. Braga, 2017. Citado na página 56.

ELETECHSUP. *Digital Switch Optical Isolation Module Logic Level Converter for PLC RS485 IO Communication [OP71A04_3V3_5V].* [S.l.], Eletechsup, [s.d.]. Disponível em: https://485io.com/io-isolation-protection-board-c-29/op71a04-33v-5v-npn-active-low-10khz-dido-digital-switch-optical-isolation-module-logic-level-converter-for-plc-rs485-io-communication-p-479.html . Acesso em: 22 abr. 2024. Nenhuma citação no texto.

WR KITS. *Pré Amplificador Para Subwoofer*, 2016. Disponível em: https://www.youtube.com/watch?v=0mPKgBMQXgY&ab_channel=WRKits. Acesso em: 22 abr. 2024. Nenhuma citação no texto.

ALIEXPRESS. *Subwoofer Front Stage Plate*, Single Power, 10-24V, Filtro Passa Baixa, Tone Board, Placa acabada. [S.l.], AliExpress, [s.d.]. Disponível em: https://pt.aliexpress.com/item/1005006160229191.html?spm=a2g0o.order_list.order_list_main.111.57d3caa4jg7O9C&gatewayAdapt=glo2bra. Acesso em: 22 abr. 2024. Nenhuma citação no texto.

CAVALLI, Marco Aurélio. *Sound System Public Addres Vertical Line Array*. 2018. 65 f. Trabalho final de graduação (Engenheiro Eletricista). Curso de Engenharia Elétrica. Universidade de Passo Fundo, Passo Fundo, RS, 2018. Citado na página 61.

GARCIA, Richard Felizandro. *Sistema de comunicación FSK y conectividad de instrumentos de campo con tecnología Hart para ampliar el conocimiento de protocolos industriales 2018*. 2019. Nenhuma citação no texto.

HART Module DS8500 or *2085 Chip HART Modem.* Modem HART Protocol Slave. AliExpress, [s.d.]. Disponível em: https://pt.aliexpress.com/i/1005004945489067.html. Acesso em: 16 abr. 2024. Nenhuma citação no texto.

ANALOG DEVICES. *Introduction to the DS8500 HART Modem*. Analog.com, [s.d.]. Disponível em: https://www.analog.com/en/resources/technical-articles/introduction-to-the-ds8500-hart-modem.html. Acesso em: 16 abr. 2024. Citado 2 vezes nas páginas 64 e 65.

MARTINS, Victor Ferreira. *Automação residencial usando protocolo MQTT, Node-RED e Mosquitto Broker com ESP32 e ESP8266*. 2019. 53 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Universidade Federal de Uberlândia. Uberlândia. 2020. Citado na página 78.

SARAVATI. *Cabo FTDI USB A para TTL RS232 FT232 6 Pinos*. Disponível em: https://www.saravati.com.br/cabo-ftdi-usb-a-para-ttl-rs232-ft232-6-pinos.html?gad_source=1&gclid=Cj0KCQjw_qexBhCoARIsAFgBletqNYUv25UtvPDn4KxCcd2b2cuBN3zPxGzzgQc4Vhi7tC8X0iSs2rUaAoaIEALw_wcB\. Acesso em: 26 abr. 2024. Nenhuma citação no texto.

APÊNDICE A – CÓDIGO COMPLETO DA ESP32 PARA OS MÓDULOS.

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>

// Definição das credenciais da rede Wi-Fi
const char* ssid = "NomeDaRede";
const char* password = "SenhaDaRede";

// Definição do endereço e porta do servidor MQTT
const char* mqtt_server = "test.mosquitto.org";
const int mqtt_port = 1883;

// Definição dos pinos utilizados

// Módulo Conversor tensão para corrente
const int potPin = 39; // Pino analógico conectado ao potenciômetro
const int modulo1 = 26; // Pino DAC conectado ao módulo de conversão de tensão
para corrente

// Módulo Conversor corrente para tensão
const int modulo2 = 32; // Pino analógico conectado ao módulo de conversão de
corrente para tensão

// Módulo amplificador
const int onda = 25;  // Pino DAC conectado ao módulo amplificador LM358

// Módulo MOSFET PWM
const int output_pin = 14; // Pino PWM da ESP32
const int potentiometer_pin = 34; // Pino do potenciômetro
const int freq = 5000; // Frequência PWM definida para 5000Hz

// Módulo SSR
const int pinRele = 5; // Pino de controle do relé

// Tabela de formas de onda
#define Num_Samples  112  // Número de amostras por forma de onda
#define MaxWaveTypes 4    // Total de tipos de formas de onda
byte wave_type = 0;       // Tipo de onda inicial (senoidal)
int i = 0;                // Índice para percorrer a tabela de formas de onda

// Tabela de amostras para diferentes formas de onda
static byte WaveFormTable[MaxWaveTypes][Num_Samples] = {
   // Onda senoidal
   {
    0x80, 0x83, 0x87, 0x8A, 0x8E, 0x91, 0x95, 0x98, 0x9B, 0x9E, 0xA2, 0xA5,
0xA7, 0xAA, 0xAD, 0xAF,
```

```
    0xB2, 0xB4, 0xB6, 0xB8, 0xB9, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF, 0xBF, 0xBF,
0xC0, 0xBF, 0xBF, 0xBF,
    0xBE, 0xBD, 0xBC, 0xBB, 0xB9, 0xB8, 0xB6, 0xB4, 0xB2, 0xAF, 0xAD, 0xAA,
0xA7, 0xA5, 0xA2, 0x9E,
    0x9B, 0x98, 0x95, 0x91, 0x8E, 0x8A, 0x87, 0x83, 0x80, 0x7C, 0x78, 0x75,
0x71, 0x6E, 0x6A, 0x67,
    0x64, 0x61, 0x5D, 0x5A, 0x58, 0x55, 0x52, 0x50, 0x4D, 0x4B, 0x49, 0x47,
0x46, 0x44, 0x43, 0x42,
    0x41, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x41, 0x42, 0x43, 0x44,
0x46, 0x47, 0x49, 0x4B,
    0x4D, 0x50, 0x52, 0x55, 0x58, 0x5A, 0x5D, 0x61, 0x64, 0x67, 0x6A, 0x6E,
0x71, 0x75, 0x78, 0x7C
   },
   // Onda triangular
   {
     0x80, 0x84, 0x89, 0x8D, 0x92, 0x96, 0x9B, 0x9F, 0xA4, 0xA8, 0xAD, 0xB2,
0xB6, 0xBB, 0xBF, 0xC4,
     0xC8, 0xCD, 0xD1, 0xD6, 0xDB, 0xDF, 0xE4, 0xE8, 0xED, 0xF1, 0xF6, 0xFA,
0xFF, 0xFA, 0xF6, 0xF1,
     0xED, 0xE8, 0xE4, 0xDF, 0xDB, 0xD6, 0xD1, 0xCD, 0xC8, 0xC4, 0xBF, 0xBB,
0xB6, 0xB2, 0xAD, 0xA8,
     0xA4, 0x9F, 0x9B, 0x96, 0x92, 0x8D, 0x89, 0x84, 0x7F, 0x7B, 0x76, 0x72,
0x6D, 0x69, 0x64, 0x60,
     0x5B, 0x57, 0x52, 0x4D, 0x49, 0x44, 0x40, 0x3B, 0x37, 0x32, 0x2E, 0x29,
0x24, 0x20, 0x1B, 0x17,
     0x12, 0x0E, 0x09, 0x05, 0x00, 0x05, 0x09, 0x0E, 0x12, 0x17, 0x1B, 0x20,
0x24, 0x29, 0x2E, 0x32,
     0x37, 0x3B, 0x40, 0x44, 0x49, 0x4D, 0x52, 0x57, 0x5B, 0x60, 0x64, 0x69,
0x6D, 0x72, 0x76, 0x7B
  },

   // Onda dente de serra
   {
     0x00, 0x02, 0x04, 0x06, 0x09, 0x0B, 0x0D, 0x10, 0x12, 0x14, 0x16, 0x19,
0x1B, 0x1D, 0x20, 0x22,
     0x24, 0x27, 0x29, 0x2B, 0x2D, 0x30, 0x32, 0x34, 0x37, 0x39, 0x3B, 0x3E,
0x40, 0x42, 0x44, 0x47,
     0x49, 0x4B, 0x4E, 0x50, 0x52, 0x54, 0x57, 0x59, 0x5B, 0x5E, 0x60, 0x62,
0x65, 0x67, 0x69, 0x6B,
     0x6E, 0x70, 0x72, 0x75, 0x77, 0x79, 0x7C, 0x7E, 0x80, 0x82, 0x85, 0x87,
0x89, 0x8C, 0x8E, 0x90,
     0x93, 0x95, 0x97, 0x99, 0x9C, 0x9E, 0xA0, 0xA3, 0xA5, 0xA7, 0xA9, 0xAC,
0xAE, 0xB0, 0xB3, 0xB5,
     0xB7, 0xBA, 0xBC, 0xBE, 0xC0, 0xC3, 0xC5, 0xC7, 0xCA, 0xCC, 0xCE, 0xD1,
0xD3, 0xD5, 0xD7, 0xDA,
     0xDC, 0xDE, 0xE1, 0xE3, 0xE5, 0xE8, 0xEA, 0xEC, 0xEE, 0xF1, 0xF3, 0xF5,
0xF8, 0xFA, 0xFC, 0xFE
   },
   // Onda quadrada
```

```
   {
     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00
   }
};
WiFiClient espClient;
PubSubClient client(espClient);

// Limites de corrente para conversão
const float MIN_CURRENT = 4.0;  // Corrente mínima em mA
const float MAX_CURRENT = 20.0; // Corrente máxima em mA

// Função para configurar a conexão Wi-Fi
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Conectando à rede ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi conectado");
    Serial.println("Endereço IP: ");
    Serial.println(WiFi.localIP());
}
// Função para reconectar ao servidor MQTT
void reconnect() {
    while (!client.connected()) {
        Serial.print("Tentando se reconectar ao MQTT Broker...");
        if (client.connect("ESP32Client")) {
            Serial.println("Conectado");
            client.subscribe("tcc/onda"); // Inscreve-se no tópico para
receber o tipo de onda
```

```
            client.subscribe("tcc/carga"); // Inscreve-se no tópico para
receber o comando pro SSR
        } else {
            Serial.print("Falhou, rc=");
            Serial.print(client.state());
            Serial.println(" Tentando novamente em 5 segundos");
            delay(5000);
        }
    }}
// Callback para tratamento de mensagens MQTT recebidas
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Mensagem recebida no tópico: ");
    Serial.println(topic);

    // Converte o payload para string
    String message;
    for (int i = 0; i < length; i++) {
        message += (char)payload[i];
    }
    // Determina o tipo de onda com base na mensagem recebida
    if (message.equals("senoidal")) {
        wave_type = 0;
    } else if (message.equals("triangular")) {
        wave_type = 1;
    } else if (message.equals("dente-de-serra")) {
        wave_type = 2;
    } else if (message.equals("quadrada")) {
        wave_type = 3;
    }
// Módulo SSR
    if (strcmp(topic, "tcc/carga") == 0) {
    if (payload[0] == '0') {
      digitalWrite(pinRele, LOW); // Liga a carga
      client.publish("status/carga", "Carga acionada");
    } else if (payload[0] == '1') {
      digitalWrite(pinRele, HIGH); // Desliga a carga
      client.publish("status/carga", "Carga desativada");
    }
  }
}
// Função para converter leitura analógica em corrente
float analogToCurrent(float value) {
    float current = MIN_CURRENT + ((MAX_CURRENT - MIN_CURRENT) / (3.3 - 0)) *
(value - 0);
    return current;
}
// Configurações iniciais do dispositivo
void setup() {
    Serial.begin(115200);
```

```
    setup_wifi();
    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
    pinMode(output_pin, OUTPUT); // Configura o pino de saída PWM
    pinMode(pinRele, OUTPUT); // Configura o pino de saída pro rele SSR
}
// Loop principal do programa
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    // Leitura e processamento dos módulos

    // Módulo Conversor tensão para corrente
    int potValue = analogRead(potPin);
    float potVoltage = (3.3 / 4095.0) * potValue;
    float current_modulo1 = analogToCurrent(potVoltage);
    dacWrite(modulo1, potVoltage / 3.3 * 255);
    client.publish("tcc/potenciometro", String(potVoltage).c_str());
    client.publish("tcc/corrente1", String(current_modulo1).c_str());

    // Módulo Conversor corrente para tensão
    int voltage_value = analogRead(modulo2);
    float voltage_modulo2 = (3.3 / 4095.0) * voltage_value;
    float current_modulo2 = analogToCurrent(voltage_modulo2);
    client.publish("tcc/modulo2", String(voltage_modulo2).c_str());
    client.publish("tcc/corrente2", String(current_modulo2).c_str());

    // Módulo Amplificador de Ganho de Sinal para Geração de Funções
    dacWrite(onda, WaveFormTable[wave_type][i]);
    i++;
    if (i >= Num_Samples) i = 0;

    // Módulo MOSFET PWM
    int potValuePWM = analogRead(potentiometer_pin);
    int dutyCycle = map(potValuePWM, 0, 4095, 0, 1024);

    ledcAttachPin(output_pin, 0); // Associa o canal PWM ao pino de saída
    ledcSetup(0, freq, 10);        // Configura o canal PWM
    ledcWrite(0, dutyCycle);      // Define o valor do duty cycle

    float percentage = (float)dutyCycle / 1024.0 * 100.0;
    client.publish("tcc/duty", String(percentage).c_str());
    client.publish("tcc/freq", String(freq).c_str());

    delay(100); // Aguarda um curto intervalo antes da próxima iteração
}
```

APÊNDICE B – CÓDIGO DA ESP32 PARA O MODEM HART DS8500.

```
#include <Arduino.h>
#include <HardwareSerial.h>
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "NomeDaRede";
const char* password = "Senha";
// Definição do endereço e porta do servidor MQTT
const char* mqtt_server = "test.mosquitto.org";
const int mqtt_port = 1883;

WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);

// Definição dos tópicos MQTT
const char* mqtt_topic_frame = "tcc/frame";
const char* mqtt_topic_response = "tcc/resposta/hart";

#define DS8500_RX 3 // Pino RX da ESP32 conectado ao pino RX do DS8500
#define DS8500_TX 1 // Pino TX da ESP32 conectado ao pino TX do DS8500
#define DS8500_RTS 12 // Pino RTS da ESP32 conectado ao pino RTS do DS8500
#define DS8500_CD 14 // Pino CD da ESP32 conectado ao pino CD do DS8500

// Inicializando a comunicação serial com o DS8500
HardwareSerial ds8500Serial(1); // Usando a UART1 da ESP32
#define BAUD_RATE 1200

void setup() {
  Serial.begin(12000);

  // Conectar-se à rede Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Conectando ao WiFi...");
  }
  Serial.println("Conectado ao WiFi com sucesso!");

  // Inicializar o cliente MQTT
  mqttClient.setServer(mqtt_server, mqtt_port);

  // Conectar-se ao broker MQTT
  connectToMQTT();
}
void loop() {
  if (!mqttClient.connected()) {
```

```
    connectToMQTT();
  }
  mqttClient.loop();
}
void connectToMQTT() {
  while (!mqttClient.connected()) {
    Serial.println("Conectando ao broker MQTT...");
    if (mqttClient.connect("ESP32Client"){
      Serial.println("Conectado ao broker MQTT!");
      mqttClient.subscribe(mqtt_topic_frame);
    } else {
      Serial.print("Falha ao conectar ao broker MQTT, rc=");
      Serial.println(mqttClient.state());
      delay(2000);
    }
  }}
void callback(char* topic, byte* payload, unsigned int length) {
  if (strcmp(topic, mqtt_topic_frame) == 0) {
    // Recebeu mensagem no tópico tcc/frame
    if (length > 0) {
      // Enviar o quadro HART para o DS8500
      sendHARTFrame(payload, length);
    }
  }}
void sendHARTFrame(byte* frameData, unsigned int frameLength) {
  // Enviar o quadro de dados HART via porta serial para o DS8500
  ds8500Serial.write(frameData, frameLength);

  // Imprimir o quadro enviado no monitor serial (para debug)
  Serial.println("Quadro HART enviado:");
  for (unsigned int i = 0; i < frameLength; i++) {
    Serial.print(frameData[i], HEX);
    Serial.print(" ");
  }
  Serial.println();
  // Aguardar a resposta do DS8500 (tempo suficiente para a resposta)
  delay(100); // Ajuste conforme necessário

  // Verificar se há dados disponíveis na porta serial do DS8500
  if (ds8500Serial.available()) {
    // Ler a resposta do DS8500
    byte response = ds8500Serial.read();

    // Enviar a resposta HART via MQTT
    mqttClient.publish(mqtt_topic_response, (const uint8_t*)&response, 1);
  } else {
    Serial.println("Timeout: Nenhuma resposta do DS8500");
  }}
```