PROTÓTIPO DE UM CONVERSOR BIDIRECIONAL DE ESCALA CONFIGURÁVEL PARA SENSORES ANALÓGICOS INDUSTRIAIS[1]

João Victor Oliveira Cordeiro de Souza[2]
Carlos Toshyuki Matsumi[3]
José Flávio Dums[4]

Resumo

Este trabalho apresenta o desenvolvimento de um protótipo de conversor de sinais de entrada e saída de escala configurável para sensores analógicos industriais. O método empregado envolveu a concepção e implementação de um dispositivo modular, composto por estágios de entrada para condicionamento de sinais (tensão e corrente), um módulo de interface e processamento baseado no microcontrolador ATmega328P (Arduino) para cálculos e controle via PWM, e módulos de saída para geração de sinais em corrente (Fonte de Howland) e em tensão (Conversor Buck). Os resultados obtidos demonstraram fidelidade e linearidade do protótipo entre os valores teóricos e medidos em diversas escalas de conversão testadas (ex: 0-10V para 4-20mA), validando sua viabilidade funcional. A conclusão aponta que o dispositivo atende aos requisitos fundamentais de projeto, oferecendo uma solução versátil e configurável para a compatibilidade de sinais em ambientes industriais.

Palavras-Chave: Sensores Analógicos, Protótipo, Microcontrolador, Conversor de Sinais, Eletrônica.

PROTOTYPE OF A CONFIGURABLE SCALE INPUT-OUTPUT CONVERTER FOR INDUSTRIAL ANALOG SENSORS

Abstract

This work presents the development of a configurable input-output scale converter prototype for industrial analog sensors. The method employed involved the design and implementation of a modular device, consisting of input stages for signal conditioning (voltage and current), an interface and processing module based on the ATmega328P microcontroller (Arduino) for calculations and PWM control, and output modules for generating current signals (Howland Source) and voltage signals (Buck Converter). The results obtained demonstrated high fidelity and linearity of the prototype between theoretical and measured values across various tested conversion scales (e.g., 0-10V to 4-20mA), validating its functional viability. The conclusion indicates that the device meets fundamental design requirements, offering a versatile and configurable solution for signal compatibility in industrial environments.
Keywords: Analog Sensors, Prototype, Microcontroller, Signal Converter, Electronics.

---

[1] Artigo apresentado ao curso de Engenharia Elétrica do Instituto Federal de Santa Catarina, para obtenção do título de bacharel em Engenharia Elétrica. 2025.

[2] Acadêmico do curso de Engenharia Elétrica do Instituto Federal de Santa Catarina. joao.v07@aluno.ifsc.edu.br

[3] Orientador, professor Doutor do curso Bacharelado em Engenharia Elétrica do Instituto Federal de Santa Catarina. ifscmatsumi@gmail.com

[4] Co-orientador, professor Doutor do curso Bacharelado em Engenharia Elétrica do Instituto Federal de Santa Catarina. joseflavio@ifsc.edu.br

1 INTRODUÇÃO

A instrumentação é parte imprescindível em qualquer processo industrial. É por meio desta que podemos realizar as leituras necessárias para estabelecer a comparação entre diferentes metodologias. Isto é possível por conta do uso de sensores e instrumentos capazes de medir as diferentes grandezas físicas presentes no processo.

Para fornecer informações acerca de grandezas físicas, um sistema de medição pode ser composto por alguns elementos, entre eles: sensores, circuitos, visores, equações e programas de computadores, que atuam no processo de medição, segundo Aguirre (2013).

O termo 'sensor' refere-se a dispositivos que detectam diferentes tipos de energia, como luz, calor ou movimento, e têm como objetivo fornecer informações sobre diversas grandezas a serem medidas, como temperatura e pressão (THOMAZINI; ALBUQUERQUE, 2020).

Na indústria encontramos diversos tipos de sensores, que podem ser digitais, analógicos e, os mais atuais, sensores inteligentes. Os sensores digitais, ao longo do tempo, podem assumir dois estados, ligado ou desligado (zero ou um). Não há grandezas físicas expressas dessa maneira, mas ao passar por um sistema de controle, entregam ao controlador esse status. Os sensores analógicos, por sua vez, ao longo do tempo, podem assumir qualquer valor numérico disponível em sua escala de operação. Este último é importante para o monitoramento em tempo real em uma aplicação ou controle de algum sistema eletrônico.

A gama de produtos no mercado atual é diversificada quando se fala em Controlador Lógico Programável (CLP). Existem diversas marcas e modelos, com diferentes usabilidades, e de maneira análoga, os produtos possuem entradas e saídas analógicas em diferentes escalas, podendo ou não ser configuráveis.

No monitoramento de sinais analógicos, tanto os sensores quanto os controladores podem assumir duas grandezas: tensão ou corrente. Para a escala em tensão, é muito comum a faixa de operação ser 0-10V, mas também pode ser encontrado de 1-5V ou 0-5V para o caso de microcontroladores. Já na escala em corrente, encontramos tanto faixa de operação em 0-20mA quanto em 4-20mA.

É necessário que haja uma certa flexibilidade nos modos de operações dos controladores e/ou sensores, em razão das dinamicidades ou até mesmo da escassez de produtos no mercado, pois na hipótese de se estar usando sensor saída em tensão e este queimar, sua reposição pode não ser 100% compatível e será preciso continuar rodando a aplicação. Assim, faz-se necessário algum conversor para que condicione a saída do sensor para o sinal de entrada que o controlador espera receber.

A incompatibilidade entre as escalas dos sinais fornecidos pelos sensores e os sinais exigidos pelos controladores lógicos é um desafio comum na automação industrial. Essa discrepância pode causar falhas nos sistemas automatizados, levando a erros de medição e controle.

Como solução para esse problema, a construção de um dispositivo capaz de converter os diferentes tipos de sinais de entrada, no tipo de sinal de saída que melhor for conveniente, foi o que gerou a necessidade de estudo e elaboração desse protótipo e artigo.

Hoje no mercado, é possível encontrar dispositivos capazes de realizar essas conversões, mas de maneira estática, não possibilitando a inversão de grandezas.

O diferencial deste projeto é a possibilidade de escolhas entre as escalas das grandezas e obter-se tudo em um único dispositivo as conversões bidirecionais: Tensão → Corrente e Corrente → Tensão.

O protótipo desenvolvido pode ser dividido em três principais módulos de estudo:

- Módulo de Entrada: Responsável por realizar a aquisição e o condicionamento dos dados dos sensores analógicos que estão presentes nas seguintes escalas: 0-10V, 0-5V, 1-5V, 0-20mA e 4-20mA;
- Módulo de Interface e Processamento: Responsável por realizar a interação entre o usuário e o protótipo, onde será possível fazer a seleção de escalas e visualização da grandeza convertida, além de realizar todo o processamento microprocessado para realização das conversões necessárias;
- Módulo de Saída: Responsável por gerar o sinal desejado pelo usuário, seja ele por corrente ou tensão, nas seguintes escalas: 0-10V, 0-5V, 1-5V, 0-20mA e 4-20mA;

1.1 Sinais Analógicos na indústria

Na automação industrial, a medição de grandezas físicas como pressão, temperatura, nível e vazão, é a base para o controle de processos. Essa medição é realizada por sensores e transdutores que, em sua maioria, representam a grandeza medida através de um sinal elétrico analógico. Os padrões mais consolidados para essa comunicação são o sinal em tensão (tipicamente 0-10 V) e o sinal em corrente (tipicamente 4-20 mA). A escolha entre eles depende de fatores como a distância de transmissão, a imunidade a ruído e a compatibilidade com o dispositivo de aquisição, como um CLP.

Segundo Fialho (2010), sinais analógicos são todos aqueles que podem assumir qualquer valor dentro de determinados limites e levam a informação na sua amplitude, enquanto os sinais digitais são aqueles que estabelecem um número finito de estados entre os valores máximo e mínimo do sinal em estudo.

A compreensão dessa natureza contínua é o ponto de partida para o problema central deste trabalho: como manipular a amplitude de um sinal para que ele se adeque a diferentes padrões de leitura.

Em aplicações industriais, o uso de sinais analógicos em tensão geralmente é escolhido devido a sua facilidade de medição, visto que é possível fazer a medição sem que o circuito seja aberto, apenas colocando o medidor de tensão em paralelo com a carga.

Entretanto, o sinal em tensão é mais suscetível a ruídos eletromagnéticos que são amplamente encontrados na indústria, devido a partida de motores, inversores de frequências e entre outros motivos. Além disso, em grandes distâncias, é comum que haja queda de tensão no sinal, o que pode gerar uma falsa leitura no sinal do sensor.

A fim de mitigar as inconsistências nas medições e resolver os problemas que um sinal a tensão pode ter, são utilizados sinais em corrente, geralmente 4-20 mA. Um circuito em corrente é beneficiado devido a sua baixa impedância de entrada que, por sua vez, não sofre tanto com os ruídos eletromagnéticos, uma vez que é difícil que sejam induzidas correntes significantes a ponto de atrapalhar o sinal.

1.2 Condicionamento do sinal

O sinal bruto gerado por um sensor, ou o sinal a ser entregue a um atuador, raramente está em um formato ideal para transmissão ou processamento. Ele pode ser suscetível a ruídos eletromagnéticos, sofrer atenuação em longos cabos ou apresentar níveis de impedância incompatíveis com o circuito seguinte. A partir dessa necessidade, surge a etapa de condicionamento, cujo objetivo é "limpar" e adequar o sinal.

Como destacam Junior e Silva (2015), "um circuito de condicionamento deve atender tanto as características do dispositivo sensor quanto as necessidades do controlador, que precisa ter uma grandeza elétrica em determinadas condições de amplitude e frequência para que possam desempenhar adequadamente suas funções". Portanto, o condicionamento não é apenas uma etapa preliminar, mas um requisito fundamental para a precisão de qualquer sistema de medição e controle.

A necessidade do condicionamento do sinal neste caso é devido a limitação de hardware do microcontrolador escolhido, o ATmega328P tem como VCC o valor 5V, significando que a tensão máxima nos pinos de entrada são de 5V, com uma tolerância de +0,5V.

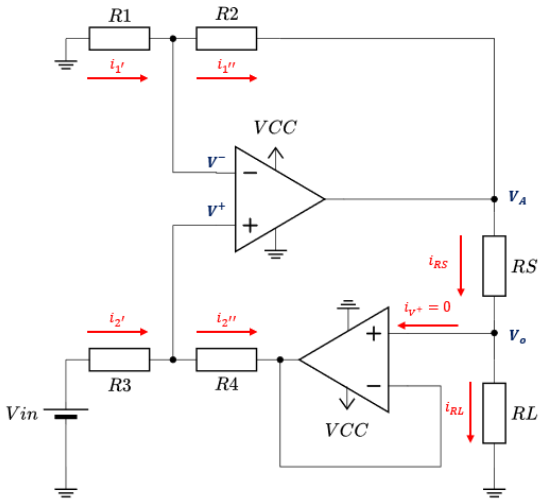
1.3 Saída em corrente: Fonte de Howland

Para implementar a funcionalidade de saída em corrente, como o padrão 4-20 mA, não basta simplesmente limitar a corrente de uma fonte de tensão. É preciso um circuito que forneça uma corrente de saída precisa e estável, independentemente das variações na impedância da carga (seja um longo cabo ou a entrada do CLP). Uma das soluções que podem ser empregadas para este desafio é a Fonte de Corrente Controlada por Tensão (FCCT).

Conforme aponta Jiang (2018), "as fontes de corrente controladas por tensão (FCCT) são amplamente utilizadas em muitas áreas, como máquinas médicas e automação industrial. A precisão CC, o desempenho CA e a capacidade de acionamento de uma (FCCT) são extremamente importantes nessas aplicações".

A escolha da topologia da Fonte de Howland (Figura 1) para o protótipo se justifica por sua alta impedância de saída e precisão, características que a tornam ideal para emular a saída de transdutores industriais e garantir que a corrente entregue ao controlador seja fiel ao valor processado.

Figura 1 – Fonte de Howland melhorada



Fonte: Autor (2025).

A corrente de interesse é a que incide no resistor RS, que pela lei de Kirchhoff das correntes será a mesma fornecida à carga (RL), ou seja, o circuito equivalente do controlador. Para achar a corrente i_{RS} é necessária a análise completa do circuito, primeiramente encontrando as equações para as correntes, conforme a figura 1.

Para satisfazer a condição de uma fonte de corrente ideal, o casamento entre os resistores R1, R2, R3 e R4 deve ser preciso, desta forma assumimos um valor de R.

Aplicando a lei de Kirchhoff das correntes (LKC) para i_1 temos que:

$$i_{1'} = i_{1''}$$

$$\frac{0 - V^-}{R} = \frac{V^- - V_A}{R}$$

$$V^- = \frac{V_A}{2}$$

Agora para i_2, temos que:

$$i_{2'} = i_{2''}$$

$$\frac{V_{in} - V^+}{R} = \frac{V^+ - V_o}{R}$$

$$V^+ = \frac{V_{in} + V_o}{2}$$

Utilizando o curto-circuito virtual, que diz que $\mathrm{V}^+ = \mathrm{V}^-$, temos que:

$$V^+ = V^-$$

$$\frac{V_{in} + V_o}{2} = \frac{V_A}{2}$$

$$V_o = V_A - V_{in}$$

Para achar a corrente i_{RS} devemos aplicar LKC ao nó de saída:

$$i_{RS} = i_{RL}$$

$$i_{RS} = \frac{V_A - V_o}{RS}$$

$$i_{RS} = \frac{V_A - V_A + V_{in}}{RS}$$

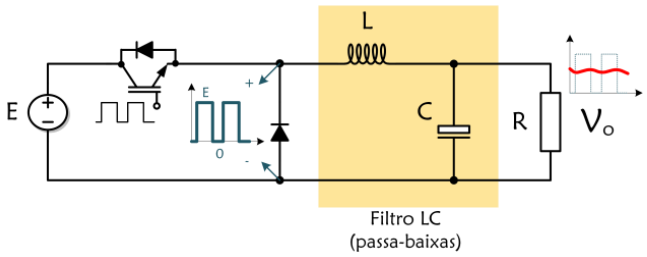
$$i_{RS} = i_{RL} = \frac{V_{in}}{RS}$$

1.4 Saída em tensão: Conversor não-isolado Buck

De forma análoga, para a geração de um sinal de tensão ajustável e estável, como o padrão 0-10 V, a eficiência energética é um fator crucial, especialmente em sistemas embarcados. Utilizar reguladores lineares para reduzir uma tensão de alimentação maior (ex: 24 V), para a faixa de saída desejada, resultaria em grande dissipação de potência na forma de calor. Por isso, uma possível abordagem é o uso de fontes chaveadas (SMPS - Switched-Mode Power Supply).

Para este projeto, onde a tensão de saída será sempre menor que a de entrada, a topologia mais indicada é o conversor Buck (figura 2). Como descrito por Ejury (2013), "um conversor buck é a topologia SMPS mais básica. É amplamente utilizado em toda a indústria para converter uma tensão de entrada mais alta em uma tensão de saída mais baixa. O conversor buck é um conversor não isolado, portanto, não há isolamento galvânico entre a entrada e a saída".

Figura 2 - Circuito básico Conversor não-isolado Buck



Fonte: Adaptado de Oliveira (2024).

No contexto deste trabalho, o conversor Buck será responsável por gerar a tensão de saída, cujo valor será controlado pelo módulo de processamento para representar a grandeza medida na escala desejada.

O funcionamento do conversor Buck baseia-se na comutação de alta frequência de um interruptor eletrônico, geralmente um transistor MOSFET, que controla o fluxo de energia para um circuito LC (Indutor-Capacitor) passa-baixas.

O princípio do conversor Buck está na forma como a tensão de saída é regulada. Isso é feito através da Modulação por Largura de Pulso (PWM), que controla o ciclo de trabalho ($\boldsymbol{Duty\ Cycle, D}$) do MOSFET. O ciclo de trabalho é a razão entre o tempo em que a chave fica ligada ($\boldsymbol{T_{on}}$) e o período total de comutação ($\boldsymbol{T}$).

A tensão de saída (V_o) é, idealmente, proporcional à tensão de entrada multiplicada pelo ciclo de trabalho ($V_o = E * D$), considerando para conversores em condução contínua.

Para gerar uma saída de 0 a 10 V a partir de uma entrada de 24 V, por exemplo, o ciclo de trabalho iria variar de 0% a aproximadamente 42%.

1.5 Processamento e Interface

Este módulo representa o cérebro do protótipo, sendo responsável por orquestrar a conversão do sinal. Sua função é ler a grandeza analógica na entrada, processá-la de acordo com a escala e o tipo de conversão desejados, e gerar os sinais de controle precisos para os módulos de saída em tensão (Conversor Buck) e corrente (Fonte de Howland). Para cumprir essa tarefa, o módulo foi subdividido em três pilares funcionais: o microcontrolador, o sistema de aquisição de dados e a interface homem-máquina (IHM).

1.5.1 O Microcontrolador como Unidade Central

Para a tarefa de processamento, foi selecionada a plataforma Arduino, especificamente utilizando um microcontrolador da família AVR (como o ATmega328P). A escolha se justifica pela sua robustez, vasta documentação, baixo custo e, crucialmente, pela flexibilidade de programação em baixo nível. Neste projeto, o microcontrolador executa quatro tarefas primordiais:

1. Aquisição: Realiza a leitura do sinal de entrada condicionado através de seu conversor analógico-digital (ADC).
2. Cálculo: Aplica a lógica de escalonamento (mapeamento de uma faixa de entrada para uma faixa de saída, ex: 0-10V para 4-20mA).
3. Controle: Gera os sinais de modulação por largura de pulso (PWM) que controlam os circuitos de saída.
4. Interface: Gerencia a exibição de informações e a recepção de comandos do usuário através do display OLED.

1.5.2 Aquisição do Sinal e Geração do Controle (ADC e PWM)

O sinal de entrada já condicionado pelo módulo de entrada, é lido pelo Conversor Analógico-Digital interno. Este componente quantiza a tensão contínua em um valor digital discreto (para um ADC de 10 bits, a faixa é de 0 a 1023), que serve como base para todo o processamento lógico subsequente.

Um diferencial técnico deste projeto reside na geração do sinal de controle. Em vez de utilizar a função analogWrite() padrão do Arduino, que opera em frequências baixas (tipicamente ~490 Hz ou ~980 Hz), optou-se pela manipulação direta dos registradores do microcontrolador para configurar o hardware do Timer/Counter, essa abordagem permitiu elevar a frequência do sinal PWM para 100 kHz.

A utilização de uma frequência elevada é fundamental para a eficiência dos estágios de saída:

- No Conversor Buck, uma frequência de chaveamento alta permite o uso de indutores e capacitores menores, otimizando o tamanho físico do circuito e melhorando a resposta a transientes.

- Na Fonte de Howland, um sinal PWM de alta frequência, após filtragem por um filtro passa-baixas RC, resulta em uma tensão de referência (Vref) muito mais estável e com menor *ripple*, garantindo a precisão da corrente de saída.

O *duty cycle* (ciclo de trabalho) deste sinal PWM de 100 kHz, ajustado via software com base nos cálculos de escalonamento, torna-se a variável de controle que define a tensão ou a corrente final na saída do protótipo.

1.5.3 Interface Homem-Máquina (IHM) com Display OLED

Para garantir a usabilidade e o monitoramento em tempo real do conversor, foi integrada uma Interface Homem-Máquina (IHM) simples e eficaz. A escolha recaiu sobre um display OLED (Organic Light Emitting Diode) de 0.96 polegadas com interface I2C.

As vantagens dessa escolha são múltiplas:

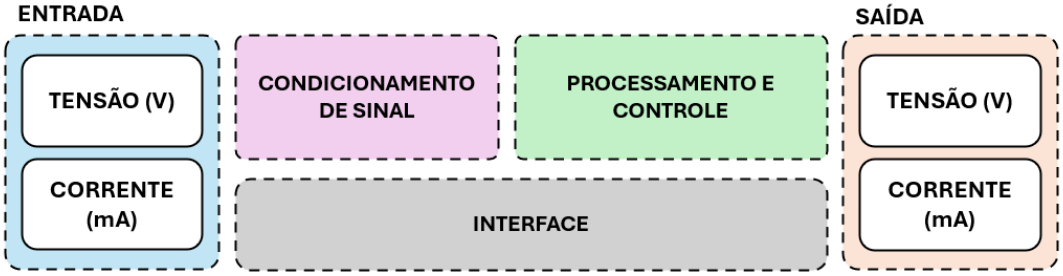
- Comunicação I2C: Utiliza apenas dois pinos do microcontrolador (SDA e SCL), economizando portas para outras funções.
- Alta Visibilidade: A tecnologia OLED oferece alto contraste e dispensa *backlight*, resultando em excelente legibilidade e baixo consumo de energia.
- Funcionalidade: O display é utilizado para exibir informações cruciais como o valor analógico lido na entrada, o valor correspondente na saída e o modo de operação atual. Em futuras iterações, esta interface pode ser expandida com botões para permitir a configuração dos parâmetros de conversão diretamente no dispositivo.

2 DESENVOLVIMENTO

2.1 Visão geral do protótipo

A figura 3 representa a visão geral em blocos do protótipo, que abrange a parte de entrada em tensão e corrente, passa pelo condicionamento e processamento dos sinais, enquanto interface interage com o usuário até que a o tipo de saída seja escolhido.

Figura 3 - Topologia do Protótipo



Fonte: Autor (2025).

2.2 Projeto e dimensionamento do hardware

2.2.1 Estágio de entrada e condicionamento

Dadas as diferentes naturezas dos sinais de entrada (0-10V, 0-5V, 1-5V, 0-20mA e 4-20mA), as estratégias de condicionamento precisam ser específicas e precisas. Para os sinais de tensão que excedem o limite do microcontrolador, como o padrão 0-10V, a abordagem mais comum é o uso de um divisor de tensão.

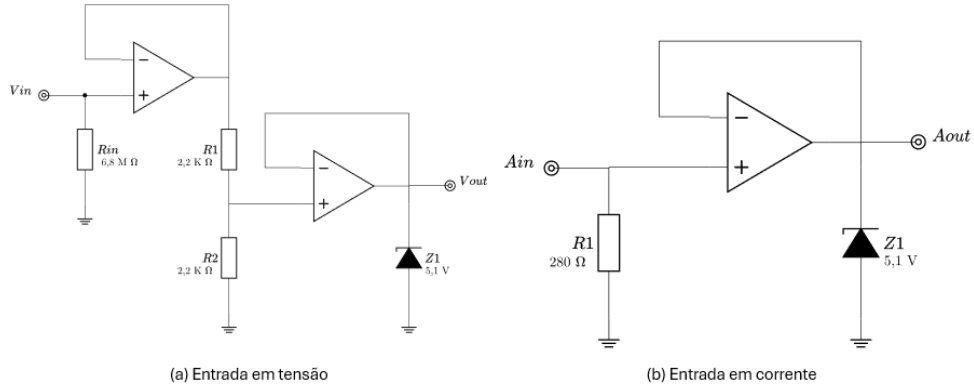
A alta impedância de entrada faz com que o circuito de medição atue como um "observador passivo", lendo a tensão sem interferir ou "carregar" o circuito que a está gerando. Essa é a razão pela qual entradas analógicas de tensão em CLPs e microcontroladores são projetadas com impedâncias na casa dos Mega-ohms (MΩ), garantindo a integridade do sinal e a precisão da medição em toda a sua planta industrial.

Já para os sinais de corrente, como 0-20mA e 4-20mA, o desafio é convertê-los em um sinal de tensão. Isso é alcançado utilizando um resistor shunt de precisão. Conforme a Lei de Ohm ($\boldsymbol{V = I * R}$), ao passar a corrente do processo por este resistor, gera-se uma queda de tensão proporcional que pode ser lida pelo pino analógico do ATmega328P.

Idealmente, para converter um máximo de 20mA em 5V, um resistor de 250Ω é a escolha padrão da indústria para esta aplicação, porém, não havia resistor comercial com esse valor, desta forma foi escolhido um resistor de 280Ω.

Os circuitos abaixo, na figura 4, retratam as duas condições citadas acima.

Figura 4 - Circuito de entrada em tensão e corrente



(a) Entrada em tensão (b) Entrada em corrente

Fonte: Autor (2025).

Além da adequação de nível e da conversão de corrente para tensão, o estágio de condicionamento desempenha a função de padronizar todos os sinais de entrada para uma única faixa de tensão (0-5V). Essa padronização não apenas viabiliza a leitura pelo hardware, mas também simplifica o software do microcontrolador. Com todos os sinais normalizados, o mesmo código de leitura e conversão analógico-digital (ADC) pode ser reutilizado, aplicando-se apenas diferentes fatores de escala para reconstruir o valor original da grandeza medida (seja ela tensão ou corrente).

Adicionalmente, este estágio pode incorporar elementos de proteção, como diodos Zener para limitar picos de sobretensão, e circuitos buffer com amplificadores operacionais, que isolam o sensor do circuito de leitura, apresentando uma alta impedância de entrada.

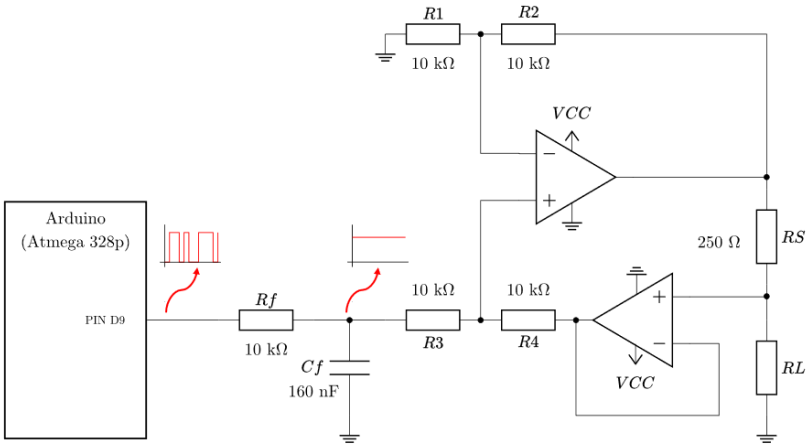
2.2.2 Saída em corrente

O circuito prático, cujo esquemático é apresentado na Figura 5, foi dimensionado com base no equacionamento desenvolvido anteriormente. Os valores dos resistores ($\boldsymbol{R1}, \boldsymbol{R2}, \boldsymbol{R3}, \boldsymbol{R4}, \boldsymbol{R_S}$), cruciais para o desempenho do circuito, foram calculados para garantir não apenas a relação correta entre a tensão de entrada e a corrente de saída, mas também a condição de casamento que maximiza a estabilidade da fonte frente a variações na impedância da carga, foram utilizados resistores de precisão com tolerância de 1%.

O controle da corrente de saída é gerenciado digitalmente pelo microcontrolador Arduino, em malha aberta. Para converter o comando digital em um sinal analógico preciso, o Arduino gera um sinal de Modulação por Largura de Pulso (PWM). Este sinal PWM passa por um filtro passa-baixa RC de primeira ordem, que atenua as altas frequências e extrai seu valor médio, resultando em uma tensão DC estável, conforme figura 9.

Esta tensão DC, serve como a tensão de referência ($\boldsymbol{V_{in}}$), para a Fonte de Howland. Dessa forma, ao ajustar o duty cycle do PWM (de 0 a 255 no Arduino), é possível modular $\boldsymbol{V_{in}}$ de forma linear e, consequentemente, controlar a corrente de saída $\boldsymbol{I_s}$ que será injetada à carga, podendo variar de 0 a 20 mA.

Figura 5 - Circuito da fonte de Howland incorporado ao arduino



Fonte: Autor (2025).

2.2.3 Saída em tensão

Além da capacidade de emular sinais de corrente no padrão 4-20 mA, o protótipo conta com um circuito de saída a tensão. Sinais em tensão são amplamente utilizados para comandar inversores de frequência, válvulas proporcionais e outros atuadores, bem como, para simular sensores que operam nesse padrão. Visando atender a essa demanda, o projeto incorpora um estágio de saída de tensão baseado em um conversor DC-DC chaveado.

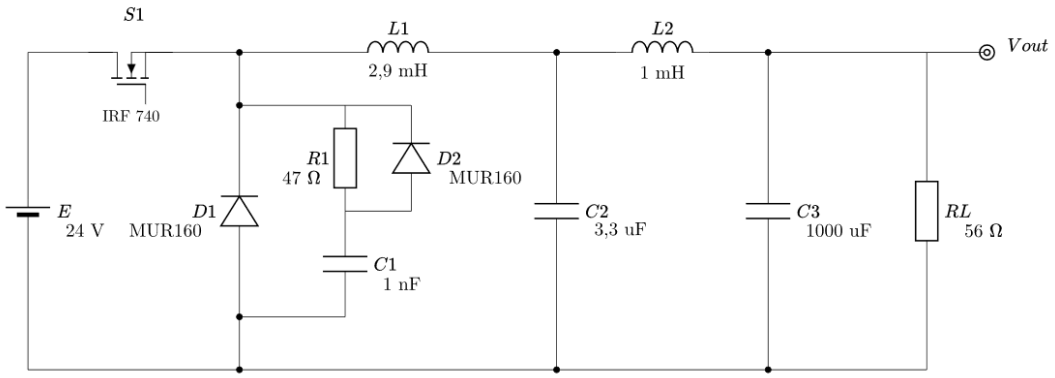
A topologia escolhida foi a do conversor Buck (step-down), ideal para reduzir a tensão de alimentação de 24 V, comum em painéis industriais, para o nível de controle desejado.

O projeto foi guiado pelas seguintes especificações: tensão de entrada (E) de 24 V, tensão de saída máxima (Vo) de 10 V, e uma frequência de comutação (f) de 100 kHz. Para garantir a fidelidade do sinal, a ondulação de tensão na saída (ΔVo) foi restringida a um máximo de 0,1%, e a ondulação de corrente no indutor (ΔiL) a 10% de seu valor médio. Tais critérios são fundamentais para que a saída de tensão seja confiável e não introduza ruídos no sistema controlado.

O circuito, projetado para alimentar uma carga nominal de 56 Ω (R) com potência próxima de 2 W, é representado pelo esquemático na Figura 6.

A condução contínua é garantida através dos parâmetros de projetos escolhidos.

Figura 6 - Circuito do conversor Buck implementado



Fonte: Autor (2025).

O indutor do conversor, peça central para o armazenamento de energia, não é um componente comercial. Ele foi confeccionado manualmente, seguindo a metodologia e os cálculos apresentados no Memorial de Cálculo. Essa abordagem permitiu um controle sobre o valor da indutância, resultando em um componente otimizado e dedicado à aplicação da frequência de operação de 100 kHz.

Durante a comutação, a recuperação reversa do diodo combinada com indutâncias parasitas do layout pode gerar picos de tensão indesejados. Para suprimir esse efeito, foi implementado um circuito snubber RCD em paralelo com o diodo MUR160. Ele atua como um amortecedor, dissipando a energia do transiente, protegendo o MOSFET e reduzindo a emissão de ruído eletromagnético (EMI).

Para garantir uma tensão de saída com o mínimo de ruído possível, um filtro LC passivo de segunda ordem foi adicionado após o capacitor principal do conversor. Enquanto o filtro Buck primário atenua a ondulação na frequência de chaveamento, este filtro adicional visa mitigar ruídos e harmônicos de alta frequência, entregando uma tensão de 10V.

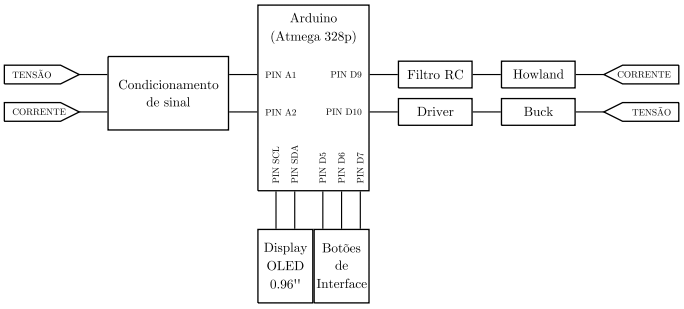
Assim como na saída em corrente, o Arduino é o responsável pela modulação PWM que através de um driver de controle, varia o duty-cycle no MOSFET, e consequentemente varia a tensão saída, tendo assim a variação necessária para o protótipo.

O controle da saída em tensão é feito em malha aberta, ou seja, não há uma realimentação do sinal de saída ao controlador para que ele possa ajustar e garantir a tensão.

2.2.4 Interface

O Arduino como meio de interface de todos os módulos possui a seguinte topologia e ligações, conforme figura 7.

Figura 7 – Topologia de ligações entre módulos e arduino



Fonte: Autor (2025).

Os sinais dos sensores são lidos por sua entrada analógica, e a saída PWM faz o controle dos módulos de saída, controla a interface OLED via protocolo I2C, ligados aos pinos SDA e SCL, e juntamente com os três botões de ação (Up, Down e Select) navega através do menu criado para a aplicação.

No display, as telas de navegação ficaram conforme figura 8:

Figura 8 – Telas desenvolvidas

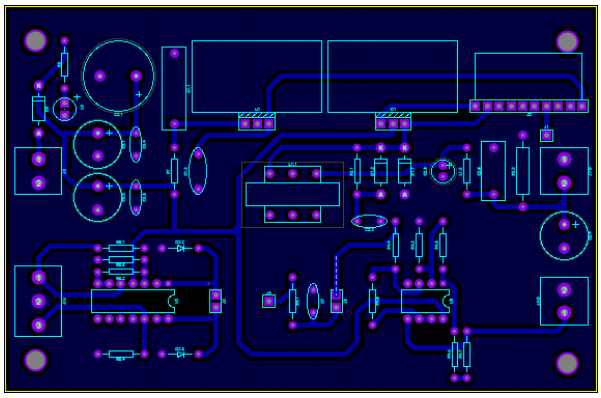
Fonte: Autor (2025).

2.2.5 Montagem do protótipo

A primeira fase da montagem consistiu na implementação de alguns circuitos do protótipo em uma protoboard. Essa abordagem permitiu a validação funcional das topologias, verificando a lógica do circuito e a realização de testes preliminares de integração.

Após a validação em protoboard, o próximo passo foi projetar uma Placa de Circuito Impresso (PCB) dedicada, visando aumentar a robustez do protótipo, a confiabilidade das conexões e a compactação do hardware. O layout da placa foi desenvolvido em um software de design de PCB, otimizando o roteamento das trilhas e o posicionamento dos componentes, conforme apresentado na Figura 9.

Figura 9 – Projeto Layout PCB

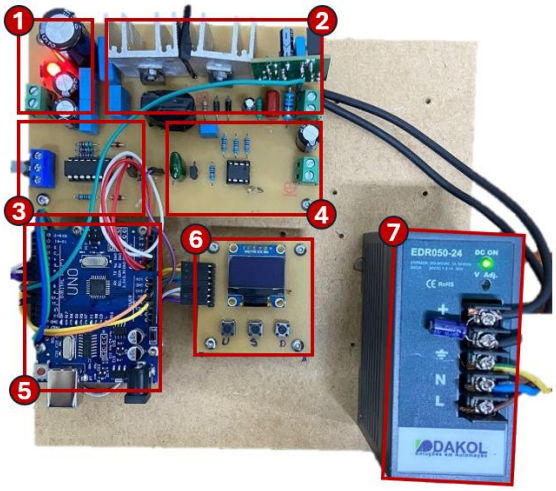


Fonte: Autor (2025).

Para a fabricação da placa, optou-se pelo método de fresagem por CNC. Essa técnica consiste na remoção do cobre de uma placa de fenolite (ou fibra de vidro) cobreada para formar as trilhas e ilhas de solda. Este método foi escolhido por sua precisão e agilidade na produção do protótipo, eliminando a necessidade de processos químicos corrosivos.

Na figura 10 são mostrados em detalhes as partes que compõe o protótipo: (1) Filtro de alimentação de entrada, (2) Conversor Buck, (3) Circuito de Entrada, (4) Fonte de Howland, (5) Arduino, (6) Interface do usuário e (7) Fonte de alimentação.

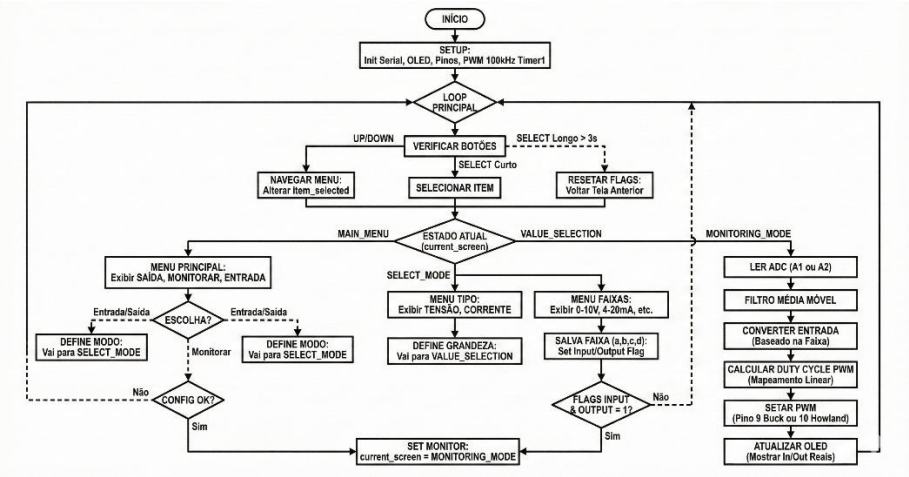
Figura 10 – Protótipo total montado



Fonte: Autor (2025).

2.3 Desenvolvimento do firmware

Figura 11 – Fluxograma do firmware



Fonte: Autor (2025).

O firmware, desenvolvido em ambiente Arduino para o microcontrolador ATmega328P, atua como o elemento central do sistema. Conforme demonstrado no fluxograma (figura 11), o código opera em um ciclo contínuo que gerencia a interação com o usuário através de menus e, uma vez configurado, assume o controle dos processos de aquisição e atuação.

A lógica inicia-se com a leitura das grandezas analógicas. O microcontrolador adquire os sinais de tensão ou corrente através de seu Conversor Analógico-Digital (ADC) interno. Para mitigar ruídos e garantir a estabilidade do sistema, as leituras brutas do ADC não são utilizadas diretamente; elas são submetidas a um filtro passa-baixas digital (implementado via média móvel com fator de suavização ALPHA).

O fluxo segue para a conversão de dados. Funções específicas, denominadas converterTensao() e converterCorrente(), são responsáveis por traduzir os valores filtrados do ADC para as unidades físicas correspondentes (V ou mA). A função converterCorrente(), por exemplo, ajusta a leitura baseada nas faixas industriais (0-20mA ou 4-20mA) e na tensão de referência.

A etapa de processamento conecta a entrada à saída utilizando a função mapfloat(). Esta função realiza uma interpolação linear, transformando proporcionalmente um valor de uma faixa de entrada para uma faixa de saída. Dentro das funções de controle, como a voltage_output() (para o conversor Buck) e current_output() (para a fonte de Howland), o mapeamento desempenha um papel duplo e simultâneo:

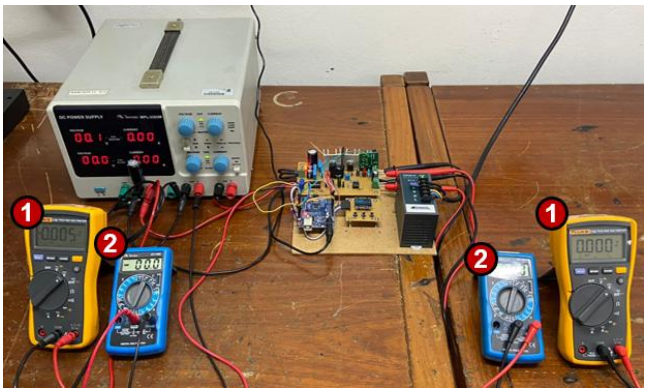
1. Controle de Hardware: Mapeia a leitura da entrada para o ciclo de trabalho (*duty cycle*) do sinal PWM de 100 kHz , que regula a potência entregue aos conversores.
2. Interface de Usuário: Calcula separadamente a variável saida_conv, que representa o valor final (em V ou mA) a ser apresentado visualmente no display OLED.

Dessa forma, o firmware assegura que o sinal de controle enviado ao hardware e a informação exibida ao usuário sejam processados de forma síncrona e coerente com as escalas definidas.

2.4 Procedimento de teste

Para a validação funcional, foram realizadas medições de tensão e corrente, tanto na entrada quanto na saída do conversor. O instrumental empregado (figura 12) consistiu em dois multímetros digitais True-RMS modelo Fluke 115 (1), para aferição de tensão, e dois multímetros digitais modelo Minipa ET-1002 (2), para medição de corrente. A utilização de vários multímetros permitiu o monitoramento simultâneo entre entrada e saída.

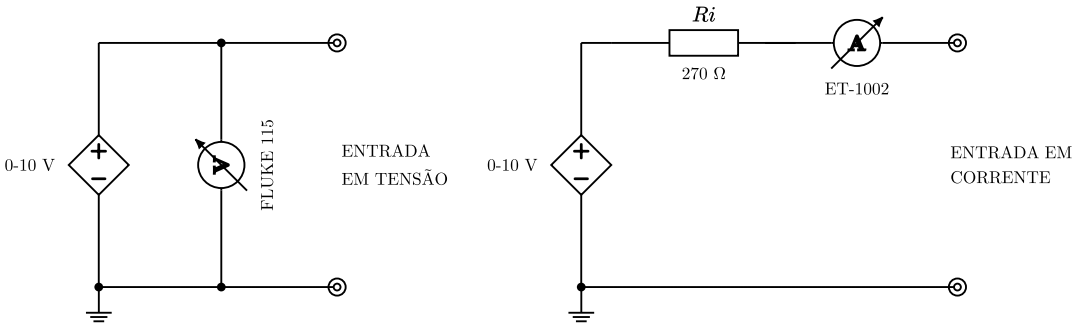
Figura 12 – Procedimento de testes



Fonte: Autor (2025).

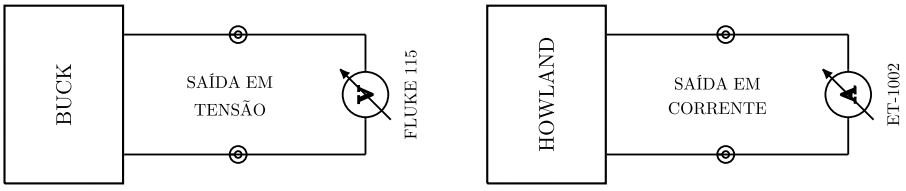
Para simular os sinais que o conversor deve processar, foi utilizado um circuito composto por uma fonte de alimentação e resistores. Conforme figura 13, este conjunto permitiu gerar sinais de entrada e medi-los, e simular as saídas de sensores industriais e permitindo testar a performance do protótipo em cada uma de suas configurações, enquanto as medições nas saídas foram realizadas conforme figura 14.

Figura 13 – Procedimento de testes – Medição de entrada



Fonte: Autor (2025).

Figura 14 – Procedimento de testes – Medição de saída



Fonte: Autor (2025).

3 RESULTADOS E DISCUSSÕES

3.1 Coleta de dados práticos

O processo de validação centrou-se na análise comparativa entre os valores teóricos esperados e os valores efetivamente medidos para cada uma das configurações de entrada e saída testadas. Esta coleta sistemática de dados permitiu quantificar a exatidão e a linearidade do conversor em suas diferentes faixas de operação, confirmando a conformidade do hardware com os requisitos de projeto.

Tabela 1 – Entrada em tensão 0-10V para saída em corrente

	SAÍDA (0-20mA)			SAÍDA (4-20mA)		
ENTRADA (0-10V)	TEÓRICO	MEDIDO	ERRO (%)	TEÓRICO	MEDIDO	ERRO (%)
0,00	0,00	0,00	0,00%	4,00	4,40	10,00%
1,00	2,00	1,98	-1,00%	5,60	6,00	7,14%
2,00	4,00	4,10	2,50%	7,20	7,60	5,56%
3,00	6,00	6,30	5,00%	8,80	9,30	5,68%
4,00	8,00	8,30	3,75%	10,40	10,80	3,85%
5,00	10,00	10,34	3,40%	12,00	12,60	5,00%
6,00	12,00	12,54	4,50%	13,60	14,20	4,41%
7,00	14,00	14,56	4,00%	15,20	15,80	3,95%
8,00	16,00	16,80	5,00%	16,80	17,70	5,36%
9,00	18,00	18,80	4,44%	18,40	19,30	4,89%
10,00	20,00	20,40	2,00%	20,00	20,70	3,50%

Fonte: Autor (2025).

Tabela 2 – Entrada em tensão 0-5V para saída em corrente

	SAÍDA (0-20mA)			SAÍDA (4-20mA)		
ENTRADA (0-5V)	TEÓRICO	MEDIDO	ERRO (%)	TEÓRICO	MEDIDO	ERRO (%)
0,00	0,00	0,00	0,00%	4,00	4,50	12,50%
0,50	2,00	1,80	-10,00%	5,60	6,05	8,04%
1,00	4,00	3,96	-1,00%	7,20	7,70	6,94%
1,50	6,00	5,95	-0,83%	8,80	9,35	6,25%
2,00	8,00	8,06	0,75%	10,40	11,00	5,77%
2,50	10,00	10,04	0,40%	12,00	12,68	5,67%
3,00	12,00	12,10	0,83%	13,60	14,36	5,59%
3,50	14,00	14,02	0,14%	15,20	16,05	5,59%
4,00	16,00	16,30	1,88%	16,80	17,60	4,76%
4,50	18,00	18,20	1,11%	18,40	19,30	4,89%
5,00	20,00	20,20	1,00%	20,00	20,70	3,50%

Fonte: Autor (2025).

Tabela 3 – Entrada em tensão 1-5V para saída em corrente

	SAÍDA (0-20mA)			SAÍDA (4-20mA)		
ENTRADA (1-5V)	TEÓRICO	MEDIDO	ERRO (%)	TEÓRICO	MEDIDO	ERRO (%)
1,00	0,00	0,00	0,00%	4,00	4,40	10,00%
1,40	2,00	1,15	-42,50%	5,60	5,20	-7,14%
1,80	4,00	3,30	-17,50%	7,20	6,80	-5,56%
2,20	6,00	5,50	-8,33%	8,80	8,50	-3,41%
2,60	8,00	7,50	-6,25%	10,40	10,10	-2,88%
3,00	10,00	9,85	-1,50%	12,00	11,80	-1,67%
3,40	12,00	11,60	-3,33%	13,60	13,30	-2,21%
3,80	14,00	13,65	-2,50%	15,20	14,80	-2,63%
4,20	16,00	15,86	-0,88%	16,80	16,30	-2,98%
4,60	18,00	18,20	1,11%	18,40	17,90	-2,72%
5,00	20,00	20,10	0,50%	20,00	19,80	-1,00%

Fonte: Autor (2025).

Tabela 4 – Entrada em corrente 0-20mA para saída em tensão

	SAÍDA (0-10V)			SAÍDA (0-5V)			SAÍDA (1-5V)		
ENTRADA (0-20mA)	TEÓRICO	MEDIDO	ERRO (%)	TEÓRICO	MEDIDO	ERRO (%)	TEÓRICO	MEDIDO	ERRO (%)
0,00	0,00	0,00	0,00%	0,00	0,00	0,00%	1,00	1,20	20,00%
2,00	1,00	1,21	21,00%	0,50	0,83	65,00%	1,40	1,60	14,29%
4,00	2,00	2,07	3,50%	1,00	1,20	20,00%	1,80	1,88	4,44%
6,00	3,00	3,14	4,67%	1,50	1,60	6,67%	2,20	2,31	5,00%
8,00	4,00	3,99	-0,25%	2,00	2,03	1,50%	2,60	2,62	0,77%
10,00	5,00	4,99	-0,20%	2,50	2,45	-2,00%	3,00	3,05	1,67%
12,00	6,00	6,08	1,33%	3,00	2,90	-3,33%	3,40	3,35	-1,47%
14,00	7,00	6,93	-1,00%	3,50	3,50	0,00%	3,80	3,80	0,00%
16,00	8,00	7,88	-1,50%	4,00	3,95	-1,25%	4,20	4,11	-2,14%
18,00	9,00	8,92	-0,89%	4,50	4,42	-1,78%	4,60	4,57	-0,65%
20,00	10,00	9,92	-0,80%	5,00	4,82	-3,60%	5,00	4,93	-1,40%

Fonte: Autor (2025).

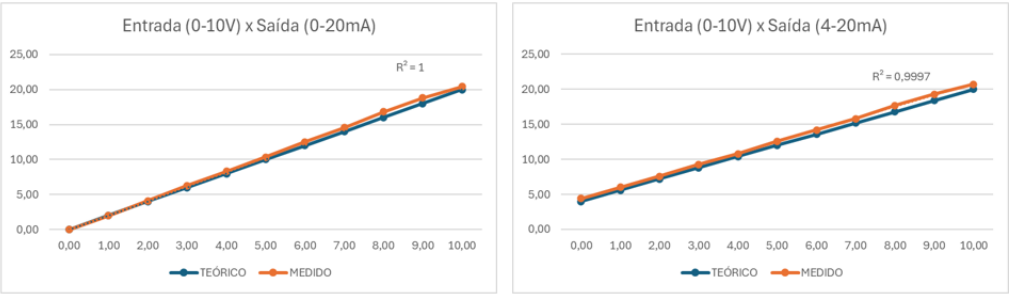
Tabela 5 – Entrada em corrente 4-20mA para saída em tensão

	SAÍDA (0-10V)			SAÍDA (0-5V)			SAÍDA (1-5V)		
ENTRADA (4-20mA)	TEÓRICO	MEDIDO	ERRO (%)	TEÓRICO	MEDIDO	ERRO (%)	TEÓRICO	MEDIDO	ERRO (%)
4,00	0,00	0,00	0,00%	0,00	0,00	0,00%	1,00	1,20	20,00%
5,60	1,00	1,07	7,00%	0,50	0,83	65,40%	1,40	1,60	14,29%
7,20	2,00	2,01	0,50%	1,00	1,20	20,00%	1,80	1,88	4,44%
8,80	3,00	2,92	-2,67%	1,50	1,60	6,67%	2,20	2,18	-0,91%
10,40	4,00	3,84	-4,00%	2,00	2,02	1,00%	2,60	2,60	0,00%
12,00	5,00	4,91	-1,80%	2,50	2,46	-1,60%	3,00	3,05	1,67%
13,60	6,00	5,82	-3,00%	3,00	2,90	-3,33%	3,40	3,35	-1,47%
15,20	7,00	6,77	-3,29%	3,50	3,35	-4,29%	3,80	3,65	-3,95%
16,80	8,00	7,72	-3,50%	4,00	3,80	-5,00%	4,20	4,11	-2,14%
18,40	9,00	8,82	-2,00%	4,50	4,39	-2,44%	4,60	4,54	-1,30%
20,00	10,00	10,01	0,10%	5,00	4,95	-1,00%	5,00	4,88	-2,40%

Fonte: Autor (2025).

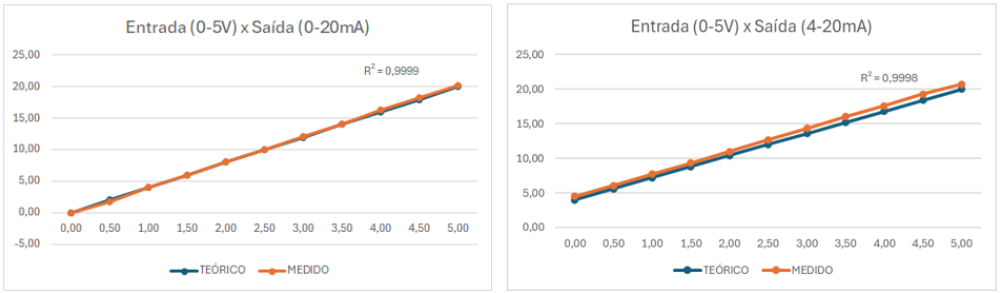
3.2 Gráficos de Linearidade

Figura 15 – Gráfico de Linearidade 0-10V x 0-20mA/4-20mA



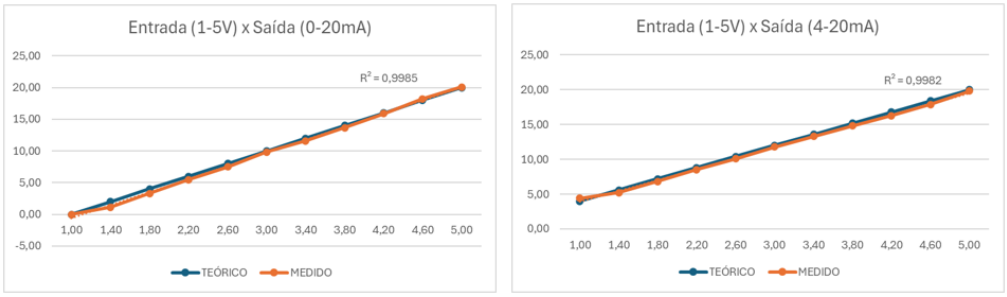
Fonte: Autor (2025).

Figura 16 – Gráfico de Linearidade 0-5V x 0-20mA/4-20mA



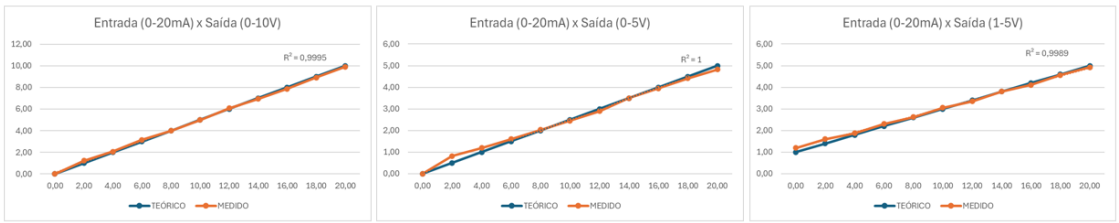
Fonte: Autor (2025).

Figura 17 – Gráfico de Linearidade 1-5V x 0-20mA/4-20mA



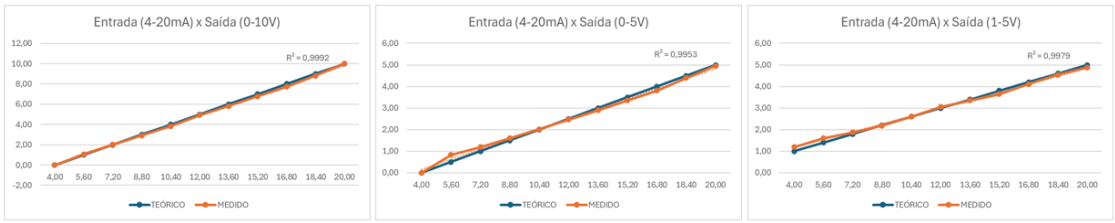
Fonte: Autor (2025).

Figura 18 - Gráfico de Linearidade 0-20mA x 0-10V/0-5V/1-5V



Fonte: Autor (2025).

Figura 19 - Gráfico de Linearidade 4-20mA x 0-10V/0-5V/1-5V



Fonte: Autor (2025).

A análise dos dados coletados revela uma forte correlação entre a resposta medida do protótipo e os valores teóricos para todas as escalas de conversão testadas (ex: 0-10V para 4-20mA, 1-5V para 0-20mA, etc.). Observa-se um padrão consistente de linearidade, onde o aumento progressivo do sinal de entrada resulta em um aumento proporcional do sinal de saída.

Para avaliar essa característica, os dados foram plotados em gráficos (Figuras 15 a 19), comparando as saídas medidas com as esperadas. Visualmente, os pontos alinham-se de forma consistente ao longo da reta de transferência ideal. Ao analisar quantitativamente o erro percentual, notam-se desvios pontuais, especialmente nas regiões de início de escala (offsets), onde a sensibilidade matemática do cálculo percentual é maior. No entanto, esses erros tendem a estabilizar ou diminuir proporcionalmente conforme a grandeza aumenta, demonstrando que o comportamento do sistema não é aleatório, mas sim sistemático.

Essa constatação é fundamental, pois indica que, embora existam discrepâncias numéricas, a linearidade positiva do sistema permanece intacta. Isso significa que o erro não compromete a viabilidade do projeto: como a resposta do conversor mantém uma proporção previsível entre entrada e saída.

Portanto, as discrepâncias observadas são esperadas em implementações analógicas e não invalidam a funcionalidade do protótipo, cuja curva de transferência comprovou ser crescente e linear.

4 CONSIDERAÇÕES FINAIS

Este trabalho se propôs a desenvolver e validar um protótipo de conversor de sinais analógicos configurável, uma ferramenta de grande relevância no cenário da automação industrial. A análise dos dados coletados confirma que o protótipo é funcionalmente viável e atende aos requisitos fundamentais de projeto.

O dispositivo demonstrou linearidade em suas diferentes faixas de operação. As imprecisões, embora aceitáveis para muitas aplicações industriais, foram atribuídas a fatores práticos e previsíveis, como a tolerância intrínseca dos componentes eletrônicos.

A conclusão deste projeto abre um leque de oportunidades para pesquisas e desenvolvimentos futuros, visando aprimorar ainda mais o dispositivo e expandir sua aplicabilidade:

- Realizar testes de bancadas reais: Fazer a leitura em um CLP, do valor convertido, ou acionar alguma carga com controle em sinal analógico como por exemplo um inversor de frequência;
- Aprimoramento do Sinal de Saída: Para mitigar o ruído de alta frequência inerente à geração de tensão via PWM filtrado, pesquisas futuras podem explorar o uso de um chip conversor digital-analógico (DAC) dedicado.
- Miniaturização com Tecnologia SMD: A transição do design atual, com componentes PTH (Pin Through-Hole), para a tecnologia de montagem em superfície (SMD - Surface-Mount Device) permitiria uma compactação da placa de circuito impresso.
- Integração do Sistema: Para criar um produto independente e robusto, o microcontrolador poderia ser embarcado na mesma placa de circuito impresso do conversor. Isso eliminaria a necessidade de módulos externos, reduziria pontos de falha (conectores e cabos) e simplificaria a instalação em campo.
- Implementação de Controle em Malha Fechada: Para atingir o mais alto nível de precisão e confiabilidade, as saídas poderiam ser controladas em malha fechada. Isso envolveria adicionar um circuito de medição que realimentasse o valor real da saída (tensão ou corrente) para o microcontrolador.
- Em suma, este trabalho não apenas entregou um protótipo funcional, mas também permite futuras melhorias. As análises realizadas e as futuras linhas de pesquisa aqui traçadas demonstram o potencial do conversor.

REFERÊNCIAS

AGUIRRE, L. A. **Fundamentos de instrumentação**. São Paulo: Pearson Education do Brasil, 2013.

BUȘCHIN, Lenart. **A quick design guide for a buck converter**. Milpitas: Analog Devices, 2016. Disponível em: www.mouser.lt. Acesso em: 21 out. 2025.

FIALHO, Arivelto B. **Instrumentação industrial: conceitos, aplicações e análises.** 7. ed. Rio de Janeiro: Érica, 2010. E-book. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788536505190/. Acesso em: 10 nov. 2025.

FUJISAWA, Cassio H. *et al*. **Instrumentação e automação industrial**. Porto Alegre: SAGAH, 2022. E-book. Disponível em: app.minhabiblioteca.com.br. Acesso em: 08 out. 2025.

JIANG, Nick. **A large current source with high accuracy and fast settling.** Analog Dialogue, [S. l.], v. 52, n. 10, p. 1-4, out. 2018.

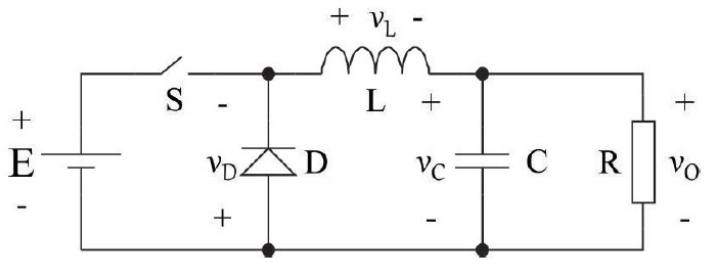
JÚNIOR, Sérgio Luiz S.; SILVA, Rodrigo A. **Automação e instrumentação industrial com Arduino: teoria e projetos.** Rio de Janeiro: Érica, 2015. E-book. Disponível em: app.minhabiblioteca.com.br. Acesso em: 19 out. 2025.

OLIVEIRA, Thiago. **Conversor Buck: da teoria ao projeto**. Elt Geral, 25 maio 2024. Disponível em: https://eltgeral.com.br/conversor-buck-da-teoria-ao-projeto/. Acesso em: 31 out. 2025.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro Urbano Braga de. **Sensores industriais**. 9. ed. Rio de Janeiro: Érica, 2020. E-book. Disponível em: app.minhabiblioteca.com.br. Acesso em: 08 out. 2025.

Apêndice A

Memorial de cálculo Conversor Buck - MCC



Parâmetros de projeto:

$E := 24\ V$ $V_o := 10\ V$ $P := 2\ W$ $R := 56\ \Omega$

$f := 100\ kHz$ $\Delta V_o := 0.1\%$ $\Delta i_L = 10\%\ i_{Lméd}$

Determinando os elementos passivos do conversor

Com base nos equacionanmentos obtidos na seção 2, podemos projetar alguns dos elementos que compõem o conversor Buck-Boost:

$P := \frac{V_o^{\,2}}{R} = 1.786\ W$ *Resistor*

$D := \frac{V_o}{E} = 0.417$ *Razão Ciclíca*

$i_{Lméd} := \frac{P}{V_o} = 178.571\ mA$ *Corrente média no indutor*

$i_{oméd} := \frac{P}{V_o} = 178.571\ mA$ *Corrente média na carga*

$\Delta i_L := 10\% \cdot i_{Lméd} = 17.857\ mA$ *Variação da corrente no indutor*

$\Delta V_C := \Delta V_o \cdot V_o = 10\ mV$ *Variação da tensão no capacitor*

$T := \frac{1}{f} = 10\ \mu s$ *Período em função da frequência*

$L := \frac{\left(\left(E - V_o\right) \cdot D \cdot T\right)}{\Delta i_L} = 3.267\ mH$ *Indutor*

$C := \frac{T \cdot \Delta i_L}{8 \cdot \Delta V_C} = 2.232\ \mu F$ *Capacitor*

Cálculo das correntes

Indutor

$$i_{Lméd} := \frac{1}{T}\int_{0}^{D\cdot T}\frac{1}{L}\cdot\left(E - V_o\right)\cdot t + 0.95\cdot\left(\frac{V_o}{R}\right)\mathrm{d}t + \frac{1}{T}\int_{D\cdot T}^{T}\frac{1}{L}\cdot\left(-V_o\right)\cdot\left(t - D\cdot T\right) + 1.05\cdot\left(\frac{V_o}{R}\right)\mathrm{d}t$$

$i_{Lméd} = 178.571\ mA$ *Corrente média do indutor*

$$i_{Lef} := \sqrt{\frac{1}{T}\left(\int_{0}^{D\cdot T}\left(\frac{1}{L}\cdot\left(E - V_o\right)\cdot t + 0.95\cdot\left(\frac{V_o}{R}\right)\right)^2\mathrm{d}t + \int_{D\cdot T}^{T}\left(\frac{1}{L}\cdot\left(-V_o\right)\cdot\left(t - D\cdot T\right) + 1.05\cdot\left(\frac{V_o}{R}\right)\right)^2\mathrm{d}t\right)}$$

$i_{Lef} = 178.646\ mA$ *Corrente eficaz do indutor*

$i_{Lmáx} := i_{Lméd} + \frac{\Delta i_L}{2} = 0.188\ A$ *Corrente máxima do indutor*

$i_{Lmín} := i_{Lméd} - \frac{\Delta i_L}{2} = 0.17\ A$ *Corrente miníma do indutor*

Chave

$$i_{Sméd} := \frac{1}{T}\int_{0}^{D\cdot T}\frac{1}{L}\cdot\left(E - V_o\right)\cdot t + 0.95\cdot\left(\frac{V_o}{R}\right)\mathrm{d}t = 74.405\ mA$$

Corrente média na chave

$$i_{Sef} := \sqrt{\frac{1}{T}\left(\int_{0}^{D\cdot T}\left(\frac{1}{L}\cdot\left(E - V_o\right)\cdot t + 0.95\cdot\left(\frac{V_o}{R}\right)\right)^2\mathrm{d}t\right)} = 115.315\ mA$$

Corrente eficaz na chave

Diodo

$$i_{Dméd} := \frac{1}{T}\int_{D\cdot T}^{T}\frac{1}{L}\cdot\left(-V_o\right)\cdot\left(t - D\cdot T\right) + 1.05\cdot\left(\frac{V_o}{R}\right)\mathrm{d}t = 104.167\ mA$$

Corrente média no diodo

$$i_{Def} := \sqrt{\frac{1}{T}\int_{D\cdot T}^{T}\left(\frac{1}{L}\cdot\left(-V_o\right)\cdot\left(t - D\cdot T\right) + 1.05\cdot\left(\frac{V_o}{R}\right)\right)^2\mathrm{d}t} = 136.443\ mA$$

Corrente eficaz no diodo

Cálculo das tensões

Indutor

$V_{Lmáx} := E - V_o = 14\ V$ *Tensão máxima no indutor*

$V_{Lmin} := -V_o = -10\ V$ *Tensão miníma no indutor*

Capacitor

$V_{Cmáx} := V_o + \frac{\Delta V_C}{2} = 10.005\ V$ *Tensão máxima no capacitor*

$V_{Cmin} := V_o - \frac{\Delta V_C}{2} = 9.995\ V$ *Tensão miníma no capacitor*

Chave

$V_{Smáx} := E = 24\ V$ *Tensão máxima na chave*

$V_{Smin} := 0\ V$ *Tensão miníma na chave*

Diodo

$V_{Dmáx} := E = 24\ V$ *Tensão máxima no diodo*

$V_{Smin} := 0\ V$ *Tensão miníma no diodo*

Dimensionamento térmico para o Interruptor

Dados do datasheet:

$r_{DS} := 0.46\ \Omega$ $t_r := 80\ ns$ $t_f := 85\ ns$ $T_J := 150\ W$ $T_A := 25\ W$

$R_{tjc} := 0.93$ $R_{tcd} := 0.5$ $R_t := 62.5$

Dados calculados:

$i_{Sef} = 115.315\ mA$ $i_{Smáx} := i_{Lmáx} = 187.5\ mA$ $V_{Smáx} = 24\ V$ $T = 10\ \mu s$

Perdas por condução: Perdas por comutação:

$P_{cond} := r_{DS} \cdot i_{Sméd}^{\ 2} = 0.003\ W$ $P_{comut} := \frac{\left(t_r + t_f\right) \cdot i_{Smáx} \cdot V_{Smáx}}{2 \cdot T} = 0.037\ W$

Perdas totais:

$P_{total} := P_{cond} + P_{comut} = 0.04\ W$

Dissipação necessária:

$R_{tda} := \frac{\left(T_J - T_A\right)}{P_{total}} = 3.151 \cdot 10^3$ $R_{tda} := R_{tda} - R_{tjc} - R_{tcd} = 3.149 \cdot 10^3$

Em teoria, como $R_{tda} < R_t$ não se faz necessário o uso de um dissipador, entretanto por ter dado um valor bem próximo e na atividade prática ter percebido um aquecimento do mosfet foi colocado um dissipador para que o interruptor não fosse afetado por perdas de calor.

Projeto físico do indutor

Definições iniciais:

$f = 100\ kHz$ — *Frequência de chaveamento*

$L = 3.267\ mH$ — *Indutância*

$i_{Lmáx} = 187.5\ mA$ — *Corrente máxima do indutor*

$i_{Lef} = 178.646\ mA$ — *Corrente eficaz no indutor*

$k_W := 70\%$ — *Fator de utilização da janela do núcleo*

$J_{máx} := 450\ \frac{A}{cm^2}$ — *Máxima densidade de corrente admitida nos condutores*

$B_{máx} := 0.35\ T$ — *Máxima densidade de Fluxo Magnético admitida no núcleo*

Determinando o núcleo

$$AeAw := \frac{L \cdot i_{Lmáx} \cdot i_{Lef}}{k_W \cdot B_{máx} \cdot J_{máx}} = 0.01\ cm^4$$

Núcleo escolhido: E 20/10/5

$A_E := 0.31\ cm^2$ — *Área da perna central do núcleo*

$A_W := 0.26\ cm^2$ — *Área da janela do núcleo*

$A_E A_W := A_E \cdot A_W = 0.081\ cm^4$

$l_e := 4.3\ cm$ — *Comprimento médio de uma espira*

$l_t := 3.8\ cm$ — *Área da janela do núcleo*

$v_e := 1.34\ cm^3$ — *Volume de ferrite no núcleo*

Determinando o número de espiras

$N_L := \text{ceil}\left(\frac{L \cdot i_{Lmáx}}{A_E \cdot B_{máx}}\right) = 57$ *Número de espiras*

Determinando o tamanho do entre ferro

$\mu_o := 4 \cdot \pi \cdot 10^{-7} \ \frac{H}{m}$ *Permeabilidade dar*

$\delta := \frac{\mu_o \cdot N_L \cdot A_E}{2 \cdot L} = \left(3.399 \cdot 10^{-4}\right) \ mm$ *Entreferro a ser adicionado em cada lado*

Selecionando o condutor

$\Delta_P := \frac{7.5}{\sqrt{f}} \cdot \frac{cm}{\sqrt{s}} = 0.024 \ cm$ *Profundidade de pentração*

$d_{cond} := 2 \cdot \Delta_P = 0.474 \ mm$ *Diâmetro do condutor a ser utilizado*

Condutor escolhido: 25 AWG

$d_c := 0.361 \ mm$ *Diâmetro do condutor escolhido*

$A_c := \pi \cdot \left(\frac{d_c}{2}\right)^2 = 0.102 \ mm^2$ *Área de cobre do condutor escolhido*

$d_{cisol} := 0.410 \ mm$ *Diâmetro do condutor escolhido com isolação em verniz*

$A_{cisol} := \pi \cdot \left(\frac{d_{cisol}}{2}\right)^2 = 0.132 \ mm^2$ *Área de cobre do condutor escolhido com isolação em verniz*

$S_{CU} := \frac{i_{Lef}}{J_{máx}} = 0.04 \ mm^2$ *Área de cobre necessária para a corrente*

$N_{cond} := \left(\frac{S_{CU}}{A_c}\right) = 0.388$ *Números de condutores em paralelo*

$A_{Wnec} := \frac{A_{cisol} \cdot N_{cond} \cdot N_L}{k_W} = 0.042 \ cm^2$ *Área de janela necessária para enrolar o indutor*

$k_{exec} := \frac{A_{Wnec}}{A_W} = 0.16$ *Possibilidade de execução*

$l_{cond} := l_t \cdot N_L = 2.166 \ m$ *Comprimento total de cada condutor*

APÊNDICE B - CÓDIGO FONTE CONVERSOR BIDIRECIONAL

```
// ========================================================================
// PROJETO: PROTÓTIPO DE UM CONVERSOR DE ENTRADA E SAÍDA DE ESCALA
// CONFIGURÁVEL PARA SENSORES ANALÓGICOS INDUSTRIAIS
//
// DESCRIÇÃO: Este firmware controla um protótipo de conversor de sinais
//            analógicos, permitindo ao usuário configurar a escala de
//            entrada e saída (tensão ou corrente) e monitorar os valores.
//            Ele utiliza um display OLED para interface e botões para
//            navegação, além de gerar sinais PWM para controle dos
//            módulos de saída (Buck e Howland).
//
// MICROCONTROLADOR: ATmega328P (Plataforma Arduino)
//
// DATA: 2025 (Referência do artigo)
// ========================================================================

// ========================================================================
// INCLUSÃO DE BIBLIOTECAS
// ========================================================================
#include "U8glib.h"     // Biblioteca para controle do display OLED
#include <stdlib.h>     // Necessária para a função dtostrf (float para string)

// ========================================================================
// DEFINIÇÕES E VARIÁVEIS GLOBAIS PARA DISPLAY OLED
// ========================================================================

// Buffers para conversão de float para string para exibição no display
char buff[10];
char buff1[10];

// Objeto U8glib para o display OLED SSD1306 via I2C (sem ACK)
// Para SPI, usaria: U8GLIB_SSD1306_128X64 u8g(12, 11, 8, 9, 10);
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NO_ACK);

// ========================================================================
// DEFINIÇÕES DE ÍCONES (Armazenados em PROGMEM para economizar RAM)
// ========================================================================
// Ícone de entrada (16x16px)
const unsigned char icon_input[] PROGMEM = {
  0x00, 0x00, 0x00, 0x80, 0x00, 0xc0, 0x00, 0xe0, 0x00, 0xf0, 0x00, 0xf8, 0x7f,
0xfc, 0x7f, 0xfe,
  0x7f, 0xfe, 0x7f, 0xfc, 0x00, 0xf8, 0x00, 0xf0, 0x00, 0xe0, 0x00, 0xc0, 0x00,
0x80, 0x00, 0x00
};
// Ícone de saída (16x16px)
const unsigned char icon_output[] PROGMEM = {
  0x00, 0x00, 0x01, 0x00, 0x03, 0x00, 0x07, 0x00, 0x0f, 0x00, 0x1f, 0x00, 0x3f,
0xfe, 0x7f, 0xfe,
  0x7f, 0xfe, 0x3f, 0xfe, 0x1f, 0x00, 0x0f, 0x00, 0x07, 0x00, 0x03, 0x00, 0x01,
0x00, 0x00, 0x00
};
// Ícone de monitor (16x16px)
```

```
const unsigned char icon_monitor[] PROGMEM = {
  0x00, 0x00, 0x40, 0x00, 0xe0, 0x0e, 0x40, 0x06, 0x40, 0x0a, 0x40, 0x10, 0x42,
0x20, 0x45, 0x40,
  0x48, 0x80, 0x50, 0x00, 0x60, 0x00, 0x40, 0x00, 0x40, 0x02, 0x7f, 0xff, 0x40,
0x02, 0x00, 0x00
};
// Ícone de tensão (16x16px)
const unsigned char icon_voltage[] PROGMEM = {
  0x00, 0x00, 0x03, 0xc0, 0x0c, 0x30, 0x10, 0x08, 0x28, 0x04, 0x24, 0x04, 0x43,
0x82, 0x42, 0x42,
  0x42, 0x42, 0x01, 0x80, 0x00, 0x00, 0x04, 0x20, 0x04, 0x20, 0x02, 0x40, 0x01,
0x80, 0x00, 0x00
};
// Ícone de corrente (16x16px)
const unsigned char icon_current[] PROGMEM = {
  0x00, 0x00, 0x03, 0xc0, 0x0c, 0x30, 0x10, 0x08, 0x28, 0x04, 0x24, 0x04, 0x43,
0x82, 0x42, 0x42,
  0x42, 0x42, 0x01, 0x80, 0x00, 0x00, 0x03, 0xc0, 0x01, 0x80, 0x01, 0x80, 0x03,
0xc0, 0x00, 0x00
};
// Ícone de voltar (16x16px)
const unsigned char icon_back[] PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf8, 0x00, 0x04, 0x00, 0x02, 0x00,
0x02, 0x00, 0x02,
  0x10, 0x02, 0x30, 0x04, 0x7f, 0xf8, 0x30, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00
};

// Array de todos os bitmaps para conveniência.
// Organizado por telas e tipo de ícone.
const unsigned char* bitmap_icons[2][3] = {
  { icon_output, icon_monitor, icon_input },     // Ícones para MAIN_MENU:
SAIDA, MONITORAR, ENTRADA
  { icon_voltage, icon_current, icon_back }      // Ícones para
SELECT_MODE/VALUE_SELECTION: TENSAO, CORRENTE, VOLTAR (ou tipo específico)
};

// =====================================================================
// ELEMENTOS GRÁFICOS DO MENU
// =====================================================================
// Scrollbar background (8x64px)
const unsigned char bitmap_scrollbar_background[] PROGMEM = {
  0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00,
0x02, 0x00, 0x02,
  0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00,
0x02, 0x00, 0x02,
  0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00,
0x02, 0x00, 0x02,
  0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00,
0x02, 0x00, 0x00
};
// Contorno do item selecionado (128x21px)
const unsigned char bitmap_item_sel_outline[] PROGMEM = {
  0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
```

```
0xff, 0xff, 0xc0,
  0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x20,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x30,
  0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xe0,
  0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xc0
};

// ======================================================================
// DEFINIÇÕES DO MENU
// ======================================================================

// Menu principal (Tela 0)
const int NUM_ITEMS_SCR0 = 3;
const int MAX_ITEM_LEN_SCR0 = 10;
```

```
char menu_itens_SCR0[NUM_ITEMS_SCR0][MAX_ITEM_LEN_SCR0] = {
  { "SAIDA" },
  { "MONITORAR" },
  { "ENTRADA" },
};

// Menu de seleção de tipo de grandeza (Tensão/Corrente) (Tela 1)
const int NUM_ITEMS_SCR1 = 3;
const int MAX_ITEM_LEN_SCR1 = 10;
char menu_itens_SCR1[NUM_ITEMS_SCR1][MAX_ITEM_LEN_SCR1] = {
  { "TENSAO" },
  { "CORRENTE" },
  { "VOLTAR" },
};

// Menu de seleção de faixas de valor (Tela 2)
// CLASS = 2 indica 2 grupos de faixas (Tensão e Corrente)
const int CLASS = 2;
const int NUM_ITEMS_SCR2 = 3;
const int MAX_ITEM_SCR = 10;
char menu_itens_SCR2[CLASS][NUM_ITEMS_SCR2][MAX_ITEM_SCR] = {
  { { "0 - 10 V" }, { "0 - 5 V" }, { "1 - 5 V" } },    // Opções de faixas para
Tensão
  { { "0 - 20 mA" }, { "4 - 20 mA" }, { "VOLTAR" } }   // Opções de faixas para
Corrente
};

// ======================================================================
// DEFINIÇÃO DE PINOS E CONSTANTES DO SISTEMA
// ======================================================================
#define BUTTON_UP_PIN 7        // Pino do botão UP
#define BUTTON_SELECT_PIN 6    // Pino do botão SELECT
#define BUTTON_DOWN_PIN 5      // Pino do botão DOWN

#define INPUT_FLAG 4           // Pino de saída para indicar estado de entrada
configurado
#define OUTPUT_FLAG 3          // Pino de saída para indicar estado de saída
configurado

#define VoltageGain 2          // Ganho do circuito de entrada de tensão (se
aplicável)
#define VoltageMax 4.5         // Tensão máxima esperada após condicionamento
(se aplicável)
#define VoltageMax1 3.8        // Outro valor de tensão máxima (uso verificar,
pode ser redundante)
#define VoltageIN 24           // Tensão de entrada para o conversor Buck (24V
industrial)
#define setFreq 159            // Valor para ICR1, define a frequência do PWM
(100kHz)
#define ResIn 250              // Valor do resistor shunt de entrada para
corrente (Ohm)
#define mA 0.001               // Fator de conversão para miliampères
#define resBit 8               // Resolução de bits (provavelmente relacionado
ao PWM, mas o uso direto precisa ser revisado)
```

```
const int ledPin = 13;         // LED interno do Arduino

// ========================================================================
// VARIÁVEIS DE ESTADO DO SISTEMA
// ========================================================================

// Estado dos botões
int button_up_clicked = 0;
int button_select_clicked = 0;
int button_down_clicked = 0;
int button_last_state = 0;

// Variáveis para detecção de "long press" no botão SELECT
unsigned long select_press_start = 0;
bool select_held = false;
bool press = 0; // Sinaliza que o botão SELECT foi pressionado (curto ou longo)

// Flags de configuração de entrada/saída
bool input_flag = 0;  // True se a entrada foi configurada
bool output_flag = 0; // True se a saída foi configurada

// Variáveis de estado do menu
int mode = 0;          // 0 = Configuração de Entrada, 1 = Configuração de Saída
int item_selected = 0; // Índice do item atualmente selecionado no menu
int current_screen = 0; // Tela atual (MAIN_MENU, SELECT_MODE, etc.)

// Variáveis de controle de exibição de ícones
int bitmaplock_i = 0;
int bitmaplock_j = 0;

// Variáveis de configuração das escalas (min e max)
// 'a' e 'b' para a faixa de entrada (min, max)
// 'c' e 'd' para a faixa de saída (min, max)
int a = 0;
int b = 0;
int c = 0;
int d = 0;

// Tipo de grandeza (0 = Tensão, 1 = Corrente)
int grandeza = 0; // Usado na seleção de escalas da tela 2
int entrada = 0;  // Tipo de grandeza de entrada selecionada (0=Tensão,
1=Corrente)
int saida = 0;    // Tipo de grandeza de saída selecionada (0=Tensão,
1=Corrente)

// Variáveis para filtro de média móvel (low-pass filter)
float valorFiltradoEntradaA1 = 0.0; // Estado do filtro para o sinal de entrada
no pino A1
float valorFiltradoEntradaA2 = 0.0; // Estado do filtro para o sinal de entrada
no pino A2
const float ALPHA = 0.1;           // Fator de suavização (0.0 a 1.0), menor
valor = mais suavização
```

```
// ========================================================================
// ENUMERAÇÃO DAS TELAS DO MENU
// ========================================================================
enum Screens {
  MAIN_MENU,         // Tela principal (Saída, Monitorar, Entrada)
  SELECT_MODE,       // Seleção de grandeza (Tensão, Corrente)
  VALUE_SELECTION,   // Seleção da faixa de valores (0-10V, 4-20mA, etc.)
  MONITORING_MODE    // Tela de monitoramento em tempo real
};

// ========================================================================
// PROTÓTIPOS DE FUNÇÕES (Declarações antecipadas)
// ========================================================================
void setupPWM100khz();
void set_dutycycle(char pino, int valor);
float converterTensao(int leituraADC, float faixaMinSaida, float faixaMaxSaida);
float converterCorrente(int leituraADC, float correnteMin, float correnteMax);
float mapfloat(float x, float in_min, float in_max, float out_min, float
out_max);
float voltage_output(int inType, float* in, float inMin, float inMax, float
outMin, float outMax);
float current_output(int inType, float* in, float inMin, float inMax, float
outMin, float outMax);

// ========================================================================
// FUNÇÃO: isButtonPressed
// DESCRIÇÃO: Verifica o estado de um botão (HIGH = pressionado)
// PARÂMETROS:
//   buttonPin: Pino digital do botão.
// RETORNO: true se o botão estiver pressionado, false caso contrário.
// ========================================================================
bool isButtonPressed(int buttonPin) {
  if (digitalRead(buttonPin) == HIGH) {
    digitalWrite(ledPin, HIGH); // Acende o LED onboard quando um botão é
pressionado
  } else {
    digitalWrite(ledPin, LOW);  // Apaga o LED onboard
  }
  return (digitalRead(buttonPin) == HIGH);
}

// ========================================================================
// FUNÇÃO: checkSelectButton
// DESCRIÇÃO: Gerencia o estado do botão SELECT, incluindo detecção de
//            "long press" para voltar à tela anterior.
// ========================================================================
void checkSelectButton() {
  if (isButtonPressed(BUTTON_SELECT_PIN)) {
    if (!button_select_clicked) { // Primeiro pressionamento
      button_select_clicked = 1;
      press = 1;
      select_press_start = millis(); // Registra o tempo de início do
pressionamento
      //Serial.println("PRESS"); // Debug
```

```
    } else if (!select_held && (millis() - select_press_start >= 3000)) { //
Long press (3 segundos)
      //Serial.println("Botão segurado - Voltando para a tela anterior"); //
Debug
      select_held = true; // Marca que o botão foi segurado
      input_flag = 0;     // Reseta flags de configuração ao voltar
      output_flag = 0;

      // Navega para a tela anterior
      if (current_screen == SELECT_MODE || current_screen == VALUE_SELECTION) {
        current_screen--;
      } else { // Se estiver na tela de monitoramento ou menu principal, volta
para o menu principal
        current_screen = MAIN_MENU;
      }
      item_selected = 0; // Reseta a seleção de item ao mudar de tela
    }
  } else { // Botão liberado
    if (button_select_clicked) {
      button_select_clicked = 0;
      press = 0;
      select_held = false; // Reseta flag de "long press"
      //Serial.println("UNPRESS"); // Debug
    }
  }
}

// ========================================================================
// FUNÇÃO: drawMenuScreen
// DESCRIÇÃO: Desenha uma tela de menu no display OLED com base nos itens
//            fornecidos e na seleção atual. Inclui scrollbar e ícones.
// PARÂMETROS:
//   menu_items: Array bidimensional com os nomes dos itens do menu.
//   num_items: Número total de itens na lista.
//   screen_index: Índice da tela atual, usado para selecionar o conjunto de
ícones.
// ========================================================================
void drawMenuScreen(char menu_items[][10], int num_items, int screen_index) {
  // Ajusta a exibição para a tela de seleção de valores (Tela 2)
  // que usa um conjunto diferente de ícones ou uma lógica bitmaplock
específica.
  if (screen_index != VALUE_SELECTION) {
    // Calcula os índices do item anterior, atual e próximo para rolagem visual
    int item_sel_previous = (item_selected - 1 + num_items) % num_items;
    int item_sel_next = (item_selected + 1) % num_items;

    // Desenha o contorno do item selecionado
    u8g.drawBitmapP(0, 22, 128 / 8, 21, bitmap_item_sel_outline);

    // Desenha o item anterior (fonte normal)
    u8g.setFont(u8g_font_7x14);
    u8g.drawStr(25, 15, menu_items[item_sel_previous]);
    u8g.drawBitmapP(4, 2, 16 / 8, 16,
bitmap_icons[screen_index][item_sel_previous]);
```

```
    // Desenha o item selecionado (fonte em negrito)
    u8g.setFont(u8g_font_7x14B);
    u8g.drawStr(25, 37, menu_items[item_selected]);
    u8g.drawBitmapP(4, 24, 16 / 8, 16,
bitmap_icons[screen_index][item_selected]);

    // Desenha o próximo item (fonte normal)
    u8g.setFont(u8g_font_7x14);
    u8g.drawStr(25, 59, menu_items[item_sel_next]);
    u8g.drawBitmapP(4, 46, 16 / 8, 16,
bitmap_icons[screen_index][item_sel_next]);

    // Desenha a scrollbar
    u8g.drawBitmapP(120, 0, 8 / 8, 64, bitmap_scrollbar_background);
    u8g.drawBox(125, 64 / num_items * item_selected, 3, 64 / num_items);
  } else { // Lógica específica para a tela de seleção de valores (Tela 2)
    // Calcula os índices do item anterior, atual e próximo para rolagem visual
    int item_sel_previous = (item_selected - 1 + num_items) % num_items;
    int item_sel_next = (item_selected + 1) % num_items;

    // Desenha o contorno do item selecionado
    u8g.drawBitmapP(0, 22, 128 / 8, 21, bitmap_item_sel_outline);

    // Desenha o item anterior (usa ícones "travados" por bitmaplock_i/j)
    u8g.setFont(u8g_font_7x14);
    u8g.drawStr(25, 15, menu_items[item_sel_previous]);
    u8g.drawBitmapP(4, 2, 16 / 8, 16, bitmap_icons[bitmaplock_i][bitmaplock_j]);

    // Desenha o item selecionado (fonte em negrito, usa ícones "travados")
    u8g.setFont(u8g_font_7x14B);
    u8g.drawStr(25, 37, menu_items[item_selected]);
    u8g.drawBitmapP(4, 24, 16 / 8, 16,
bitmap_icons[bitmaplock_i][bitmaplock_j]);

    // Desenha o próximo item (fonte normal, usa ícones "travados")
    u8g.setFont(u8g_font_7x14);
    u8g.drawStr(25, 59, menu_items[item_sel_next]);
    u8g.drawBitmapP(4, 46, 16 / 8, 16,
bitmap_icons[bitmaplock_i][bitmaplock_j]);

    // Desenha a scrollbar
    u8g.drawBitmapP(120, 0, 8 / 8, 64, bitmap_scrollbar_background);
    u8g.drawBox(125, 64 / num_items * item_selected, 3, 64 / num_items);
  }
}

// ======================================================================
// FUNÇÃO: navigateMenu
// DESCRIÇÃO: Controla a navegação no menu usando os botões UP e DOWN.
//            Atualiza 'item_selected' para o item visualmente centralizado.
// ======================================================================
void navigateMenu() {
  int num_items_current_screen;
```

```
  // Define o número de itens da tela atual
  if (current_screen == MAIN_MENU) num_items_current_screen = NUM_ITEMS_SCR0;
  else if (current_screen == SELECT_MODE) num_items_current_screen =
NUM_ITEMS_SCR1;
  else if (current_screen == VALUE_SELECTION) num_items_current_screen =
NUM_ITEMS_SCR2;
  else return; // Não navegar no modo de monitoramento

  if (isButtonPressed(BUTTON_UP_PIN) && !button_up_clicked) {
    item_selected = (item_selected > 0) ? item_selected - 1 :
num_items_current_screen - 1;
    button_up_clicked = 1;
    //Serial.println("UP"); // Debug
  }
  if (isButtonPressed(BUTTON_DOWN_PIN) && !button_down_clicked) {
    item_selected = (item_selected < num_items_current_screen - 1) ?
item_selected + 1 : 0;
    button_down_clicked = 1;
    //Serial.println("DOWN"); // Debug
  }

  // Reseta os flags de clique dos botões UP/DOWN quando liberados
  if (!isButtonPressed(BUTTON_UP_PIN)) button_up_clicked = 0;
  if (!isButtonPressed(BUTTON_DOWN_PIN)) button_down_clicked = 0;
}

// ========================================================================
// FUNÇÃO: handleMenuSelection
// DESCRIÇÃO: Processa a seleção do item atual do menu quando o botão
//            SELECT é pressionado (clique curto).
//            Gerencia a transição entre as telas e a configuração
//            das faixas de entrada/saída.
// ========================================================================
void handleMenuSelection() {
  // Verifica se o botão SELECT foi clicado (e não segurado)
  if (button_select_clicked && !select_held) {
    switch (current_screen) {
      case MAIN_MENU:
        switch (item_selected) {
          case 0: // Selecionou "SAIDA"
            //Serial.println("saída");
            mode = 1;             // Define modo de configuração de SAÍDA
            current_screen = SELECT_MODE; // Vai para a tela de seleção de
grandeza
            item_selected = 0;    // Reseta a seleção para a nova tela
            break;
          case 1: // Selecionou "MONITORAR"
            //Serial.println("monitorar");
            current_screen = MONITORING_MODE; // Vai para a tela de
monitoramento (se configurado)
            item_selected = 0;    // Reseta a seleção
            break;
          case 2: // Selecionou "ENTRADA"
            //Serial.println("entrada");
```

```
            mode = 0;             // Define modo de configuração de ENTRADA
            current_screen = SELECT_MODE; // Vai para a tela de seleção de
grandeza
            item_selected = 0;    // Reseta a seleção para a nova tela
            break;
        }
        break;

      case SELECT_MODE:
        switch (item_selected) {
          case 0: // Selecionou "TENSAO"
            //Serial.println("Tensão selecionada");
            grandeza = 0;                 // 0 = Tensão
            current_screen = VALUE_SELECTION; // Vai para a tela de seleção de
faixa de valores
            bitmaplock_i = 1;             // Define o conjunto de ícones para
Tensão (ícones de medidas)
            bitmaplock_j = 0;             // Define o ícone específico de Tensão
            item_selected = 0;            // Reseta a seleção para a nova tela
            break;
          case 1: // Selecionou "CORRENTE"
            //Serial.println("Corrente selecionada");
            grandeza = 1;                 // 1 = Corrente
            current_screen = VALUE_SELECTION; // Vai para a tela de seleção de
faixa de valores
            bitmaplock_i = 1;             // Define o conjunto de ícones para
Corrente
            bitmaplock_j = 1;             // Define o ícone específico de
Corrente
            item_selected = 0;            // Reseta a seleção para a nova tela
            break;
          case 2: // Selecionou "VOLTAR"
            //Serial.println("voltar");
            current_screen = MAIN_MENU; // Volta para o menu principal
            item_selected = 0;          // Reseta a seleção para a nova tela
            break;
        }
        break;

      case VALUE_SELECTION:
        // Verifica se é configuração de Tensão ou Corrente
        if (grandeza == 0) { // Configuração de Tensão
          switch (item_selected) {
            case 0: // Selecionou "0 - 10 V"
              //Serial.println("Tensão: 0-10V");
              if (mode == 0) { // Modo ENTRADA
                a = 0; b = 10;     // Define faixa de entrada 0-10V
                input_flag = 1;    // Sinaliza que a entrada foi configurada
                // mode = 1;          // Parece ser um erro lógico aqui, deveria
permanecer mode=0 para entrada
                entrada = 0;       // Tipo de entrada = Tensão
                current_screen = SELECT_MODE; // Volta para seleção de tipo
(Entrada/Saída)
              } else { // Modo SAIDA
```

```
              c = 0; d = 10;     // Define faixa de saída 0-10V
              output_flag = 1;   // Sinaliza que a saída foi configurada
              saida = 0;         // Tipo de saída = Tensão
            }
            break;
          case 1: // Selecionou "0 - 5 V"
            //Serial.println("Tensão: 0-5V");
            if (mode == 0) { // Modo ENTRADA
              a = 0; b = 5;
              input_flag = 1;
              // mode = 1;          // Possível erro lógico
              current_screen = SELECT_MODE;
              entrada = 0;
            } else { // Modo SAIDA
              c = 0; d = 5;
              output_flag = 1;
            }
            break;
          case 2: // Selecionou "1 - 5 V"
            //Serial.println("Tensão: 1-5V");
            if (mode == 0) { // Modo ENTRADA
              a = 1; b = 5;
              input_flag = 1;
              // mode = 1;          // Possível erro lógico
              current_screen = SELECT_MODE;
              entrada = 0;
            } else { // Modo SAIDA
              c = 1; d = 5;
              output_flag = 1;
              saida = 0;
            }
            break;
        }
        //button_select_clicked = 0; // Desativa o flag de clique para evitar
múltiplos processamentos
      } else if (grandeza == 1) { // Configuração de Corrente
        switch (item_selected) {
          case 0: // Selecionou "0 - 20 mA"
            //Serial.println("Tensão: 0-20mA"); // Erro de string, deveria ser
Corrente
            if (mode == 0) { // Modo ENTRADA
              a = 0; b = 20;
              input_flag = 1;
              // mode = 1;          // Possível erro lógico
              current_screen = SELECT_MODE;
              entrada = 1;       // Tipo de entrada = Corrente
            } else { // Modo SAIDA
              c = 0; d = 20;
              output_flag = 1;
              saida = 1;         // Tipo de saída = Corrente
            }
            break;
          case 1: // Selecionou "4 - 20 mA"
            //Serial.println("Tensão: 4-20mA"); // Erro de string
```

```
              if (mode == 0) { // Modo ENTRADA
                a = 4; b = 20;
                input_flag = 1;
                // mode = 1;          // Possível erro lógico
                current_screen = SELECT_MODE;
                entrada = 1;
              } else { // Modo SAIDA
                c = 4; d = 20;
                output_flag = 1;
                saida = 1;
              }
              break;
            case 2: // Selecionou "VOLTAR"
              current_screen = SELECT_MODE; // Volta para seleção de tipo de
grandeza (Tensão/Corrente)
              item_selected = 0;          // Reseta a seleção
              break;
          }
        }
        break;
    }
    // O flag button_select_clicked é resetado em checkSelectButton()
    // após o loop para garantir que o clique foi processado.
  }
}

// ========================================================================
// FUNÇÃO: isInMenu
// DESCRIÇÃO: Verifica se o sistema está em alguma tela de menu (não
monitoramento).
// RETORNO: true se estiver em uma tela de menu, false caso contrário.
// ========================================================================
bool isInMenu() {
  return (current_screen == MAIN_MENU || current_screen == SELECT_MODE ||
current_screen == VALUE_SELECTION);
}

// ========================================================================
// FUNÇÃO: setup
// DESCRIÇÃO: Configurações iniciais do Arduino e periféricos.
//            Executada uma única vez ao ligar ou resetar.
// ========================================================================
void setup() {
  u8g.setColorIndex(255); // Define cor para o display (branco)
  pinMode(BUTTON_UP_PIN, INPUT_PULLUP);    // Configura botões como entrada com
pull-up interno
  pinMode(BUTTON_SELECT_PIN, INPUT_PULLUP);
  pinMode(BUTTON_DOWN_PIN, INPUT_PULLUP);
  pinMode(INPUT_FLAG, OUTPUT);             // Configura pinos de flag como saída
  pinMode(OUTPUT_FLAG, OUTPUT);
  pinMode(ledPin, OUTPUT);                 // Configura LED onboard como saída
  setupPWM100khz();                        // Configura Timer1 para PWM de
100kHz
```

```
  // Inicializa os estados dos filtros com as primeiras leituras dos pinos
analógicos
  valorFiltradoEntradaA1 = analogRead(A1);
  valorFiltradoEntradaA2 = analogRead(A2);

  Serial.begin(9600); // Inicializa comunicação serial para depuração
}

// ========================================================================
// FUNÇÃO: loop
// DESCRIÇÃO: Loop principal do programa. Executado repetidamente.
//            Gerencia o display, a interação do usuário e o processamento
//            e saída dos sinais.
// ========================================================================
void loop() {
  // Se estiver em modo de menu (não monitoramento), processa a navegação e
seleção
  if (isInMenu()) {
    navigateMenu();
    checkSelectButton();
    handleMenuSelection();
    // Se tanto a entrada quanto a saída foram configuradas, transiciona para o
modo de monitoramento
    if (input_flag && output_flag) {
      current_screen = MONITORING_MODE;
      item_selected = 0; // Reseta a seleção de item ao mudar de tela
    }
  }

  // Inicia o processo de desenho na tela OLED
  u8g.firstPage();
  do {
    switch (current_screen) {
      case MAIN_MENU:
        drawMenuScreen(menu_itens_SCR0, NUM_ITEMS_SCR0, 0); // Desenha o menu
principal
        break;
      case SELECT_MODE:
        drawMenuScreen(menu_itens_SCR1, NUM_ITEMS_SCR1, 1); // Desenha o menu de
seleção de grandeza
        break;
      case VALUE_SELECTION:
        drawMenuScreen(menu_itens_SCR2[grandeza], NUM_ITEMS_SCR2, 2); // Desenha
o menu de seleção de faixa
        break;
      case MONITORING_MODE:
        {  // Usa chaves para criar um escopo local para as variáveis desta case
          float entrada_conv; // Valor da entrada convertida
          float saida_conv;   // Valor da saída convertida

          // Processamento da saída (chama a função de saída correta)
          if (saida == 0) {  // Se a saída configurada é Tensão
            saida_conv = voltage_output(entrada, &entrada_conv, a, b, c, d);
          } else {  // Se a saída configurada é Corrente
```

```
            saida_conv = current_output(entrada, &entrada_conv, a, b, c, d);
          }

          // Converte os valores float para string para exibição
          dtostrf(entrada_conv, 6, 2, buff); // 6 caracteres, 2 casas decimais
          dtostrf(saida_conv, 6, 2, buff1);

          // Define as unidades de exibição com base nos tipos de grandeza
configurados
          const char* unidade_entrada = (entrada == 0) ? " V" : "mA";
          const char* unidade_saida = (saida == 0) ? " V" : "mA";

          // Desenha na tela OLED
          u8g.setFont(u8g_font_7x14B);
          u8g.drawStr(20, 10, "MONITORAMENTO");

          u8g.drawStr(5, 30, "Entrada: ");
          u8g.drawStr(60, 30, buff);
          u8g.drawStr(110, 30, unidade_entrada);

          u8g.drawStr(5, 50, "Saida: ");
          u8g.drawStr(60, 50, buff1);
          u8g.drawStr(110, 50, unidade_saida);
        }
        break;
    }
  } while (u8g.nextPage()); // Continua desenhando até todas as páginas do
display serem atualizadas
}

// ========================================================================
// FUNÇÃO: setupPWM100khz
// DESCRIÇÃO: Configura o Timer/Counter 1 do ATmega328P para gerar sinais
//            PWM de 100 kHz nos pinos 9 (OC1A) e 10 (OC1B).
//            Utiliza o modo Fast PWM (Mode 14) com TOP definido por ICR1.
// ========================================================================
void setupPWM100khz() {
  TCCR1A = 0;             // Limpa os registradores de configuração do Timer 1
(TCCR1A e TCCR1B)
  TCCR1B = 0;
  TCNT1 = 0;              // Reseta o contador do Timer 1

  // Configura TCCR1A:
  // _BV(COM1A1) | _BV(COM1B1) : Modo non-inverting para OC1A (pino 9) e OC1B
(pino 10)
  // _BV(WGM11)                : Parte inferior para Modo 14 (Fast PWM)
  TCCR1A = _BV(COM1A1)    // Canal A no modo non-inverting
           | _BV(COM1B1)  // Canal B no modo non-inverting
           | _BV(WGM11);  // Mode 14: Fast PWM, TOP = ICR1

  // Configura TCCR1B:
  // _BV(WGM12) | _BV(WGM13) : Parte superior para Modo 14 (Fast PWM)
  // _BV(CS10)               : Prescaler = 1 (sem pré-escalonamento)
  TCCR1B = _BV(WGM12)     //
```

```
           | _BV(WGM13)   //
           | _BV(CS10);   // Prescaler = 1

  ICR1 = setFreq;         // Define o valor TOP (setFreq = 159 para 100kHz com
clock de 16MHz e prescaler 1)

  // Configura os pinos de saída PWM
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
}

// ========================================================================
// FUNÇÃO: set_dutycycle
// DESCRIÇÃO: Define o ciclo de trabalho (duty cycle) para um pino PWM
específico.
// PARÂMETROS:
//   pino: O pino PWM (9 ou 10).
//   valor: O valor do ciclo de trabalho (0 a ICR1, onde ICR1 é 159 para
100kHz).
// ========================================================================
void set_dutycycle(char pino, int valor) {
  switch (pino) {
    case 9:  // Pino OC1A
      OCR1A = valor;
      break;
    case 10: // Pino OC1B
      OCR1B = valor;
      break;
  }
}

// ========================================================================
// FUNÇÃO: converterTensao
// DESCRIÇÃO: Converte uma leitura ADC para um valor de tensão real.
//            Aplica um filtro de suavização.
// PARÂMETROS:
//   leituraADC: Valor bruto lido do Conversor Analógico-Digital (0-1023).
//   faixaMinSaida: Valor mínimo da faixa de tensão de ENTRADA (ex: 0, 1).
//   faixaMaxSaida: Valor máximo da faixa de tensão de ENTRADA (ex: 5, 10).
//                  NOTA: O nome do parâmetro é 'faixaMaxSaida', mas é usado
para
//                        escalar a entrada.
// RETORNO: O valor de tensão convertido e suavizado.
// ========================================================================
float converterTensao(int leituraADC, float faixaMinSaida, float faixaMaxSaida)
{
  float entrada_lida_estavel;
  // Aplica filtro passa-baixas à leitura do pino A1
  valorFiltradoEntradaA1 = (ALPHA * leituraADC) + ((1.0 - ALPHA) *
valorFiltradoEntradaA1);
  entrada_lida_estavel = valorFiltradaEntradaA1;

  // Converte a leitura filtrada (0-1023) para a tensão real do sensor
  // Considerando que a faixaMaxSaida é a tensão máxima que o sensor pode ter na
```

```
entrada do protótipo
  // e que o ADC do microcontrolador está configurado para uma referência de 5V.
  // Pode ser necessário ajustar a escala se o condicionamento de sinal anterior
  // ao ADC já atenuou para 0-5V.
  float tensaoLida = entrada_lida_estavel * (faixaMaxSaida / 1023.0);
  float tensaoConvertida = tensaoLida;
  return tensaoConvertida;
}

// ====================================================================
// FUNÇÃO: converterCorrente
// DESCRIÇÃO: Converte uma leitura ADC (de um resistor shunt) para um valor
//            de corrente real. Aplica filtro de suavização e ajusta para
//            faixas 0-20mA ou 4-20mA.
// PARÂMETROS:
//   leituraADC: Valor bruto lido do Conversor Analógico-Digital (0-1023).
//   correnteMin: Valor mínimo da faixa de corrente (ex: 0, 4).
//   correnteMax: Valor máximo da faixa de corrente (ex: 20).
// RETORNO: O valor de corrente convertido e suavizado.
// ====================================================================
float converterCorrente(int leituraADC, float correnteMin, float correnteMax) {
  //Serial.println(leituraADC); // Debug

  float entrada_lida_estavel;
  // Aplica filtro passa-baixas à leitura do pino A2
  valorFiltradoEntradaA2 = (ALPHA * leituraADC) + ((1.0 - ALPHA) *
valorFiltradoEntradaA2);
  entrada_lida_estavel = valorFiltradoEntradaA2;

  // Converte a leitura filtrada (0-1023) para a tensão real no pino A2
(Vref=5V)
  float tensaoLida = entrada_lida_estavel * (5.0 / 1023.0);

  float tensaoMin = 0;
  // Ajusta a tensão mínima de acordo com a faixa de corrente
  if (correnteMin == 4) { // Para faixa 4-20mA, o 4mA corresponde a 1V no shunt
de 250 Ohm
    tensaoMin = 1;
  } else { // Para faixa 0-20mA, o 0mA corresponde a 0V
    tensaoMin = 0;
  }

  // Converte a tensão lida para o valor de corrente usando mapeamento linear
  // 4.875 é a tensão máxima esperada para 20mA com o resistor shunt,
  // considerando uma referência de 5V e o ganho do condicionamento.
  float correnteConvertida = (tensaoLida - tensaoMin) * (correnteMax -
correnteMin) / (4.875 - tensaoMin) + correnteMin;

  //Serial.println(correnteConvertida); // Debug
  return correnteConvertida;
}

// ====================================================================
// FUNÇÃO: mapfloat
```

```
// DESCRIÇÃO: Mapeia um valor de uma faixa para outra, com suporte a floats.
//            Similar à função map() do Arduino, mas para números de ponto
flutuante.
// PARÂMETROS:
//   x: O valor a ser mapeado.
//   in_min: Valor mínimo da faixa de entrada.
//   in_max: Valor máximo da faixa de entrada.
//   out_min: Valor mínimo da faixa de saída.
//   out_max: Valor máximo da faixa de saída.
// RETORNO: O valor mapeado, limitado entre out_min e out_max.
// ========================================================================
float mapfloat(float x, float in_min, float in_max, float out_min, float
out_max) {
  // Fórmula de mapeamento linear: y = (x - x1) * (y2 - y1) / (x2 - x1) + y1
  float mapOut = ((x - in_min) * (out_max - out_min) / (in_max - in_min)) +
out_min;
  // Garante que o valor de saída esteja dentro da faixa especificada
(constrain)
  return constrain(mapOut, out_min, out_max);
}

// ========================================================================
// FUNÇÃO: voltage_output
// DESCRIÇÃO: Calcula e define o ciclo de trabalho do PWM para a saída de tensão
//            (Conversor Buck). Também retorna o valor de tensão de saída para
exibição.
// PARÂMETROS:
//   inType: Tipo de entrada (0=Tensão, 1=Corrente).
//   in: Ponteiro para armazenar o valor da entrada convertida.
//   inMin: Mínimo da faixa de entrada.
//   inMax: Máximo da faixa de entrada.
//   outMin: Mínimo da faixa de saída de tensão desejada.
//   outMax: Máximo da faixa de saída de tensão desejada.
// RETORNO: O valor de tensão de saída calculado.
// ========================================================================
float voltage_output(int inType, float* in, float inMin, float inMax, float
outMin, float outMax) {
  float scale_inMin = 0;  // Mínimo da escala de entrada (em ADC)
  float scale_inMax = 0;  // Máximo da escala de entrada (em ADC)
  float valueRead = 0;    // Leitura bruta do ADC da entrada

  switch (inType) {
    case 0:  // Entrada Tensão (pino A1)
      valueRead = analogRead(A1);
      *in = converterTensao(valueRead, inMin, inMax); // Converte e armazena o
valor da entrada
      // Calcula as escalas de entrada em valores ADC para o mapeamento
      // Os fatores (1.0/VoltageMax) e (1.0/VoltageGain) ajustam a leitura
      // para a faixa esperada do ADC antes do mapeamento.
      scale_inMin = (1023.0 * inMin) * (1.0 / VoltageMax) * (1.0 / VoltageGain);
      scale_inMax = (1023.0 * inMax) * (1.0 / VoltageMax) * (1.0 / VoltageGain);
      break;
    case 1:  // Entrada Corrente (pino A2)
      valueRead = analogRead(A2);
```

```
      //Serial.println(valueRead); // Debug
      *in = converterCorrente(valueRead, inMin, inMax); // Converte e armazena o
valor da entrada
      // Calcula as escalas de entrada em valores ADC para o mapeamento
      // inMin/Max * mA * ResIn = tensão correspondente no shunt.
      // (1.0/4.5) = Ajuste para faixa do ADC (considerando 4.5V como ref ou
limite)
      scale_inMin = (1023.0 * inMin * mA * ResIn) * (1.0 / 4.5);
      scale_inMax = (1023.0 * inMax * mA * ResIn) * (1.0 / 4.5);
      break;
    default:
      return -1; // Erro ou tipo de entrada inválido
  }

  // Calcula as escalas de saída para o PWM (ciclo de trabalho)
  // VoltageIN = 24V (tensão de entrada do Buck)
  // setFreq = 159 (valor TOP do PWM)
  scale_outMin = outMin * (1.0 / VoltageIN) * setFreq;
  scale_outMax = (outMax) * (1.0 / VoltageIN) * setFreq;

  // Mapeia a leitura bruta do ADC para o valor de controle do PWM
  float control = mapfloat(valueRead, scale_inMin, scale_inMax, scale_outMin,
scale_outMax);
  // Mapeia a leitura bruta do ADC para o valor de saída de tensão para exibição
  float saida_conv = mapfloat(valueRead, scale_inMin, scale_inMax, outMin,
outMax);

  set_dutycycle(9, control); // Define o duty cycle no pino 9 (OC1A) para o
Conversor Buck

  return saida_conv;
}

// ======================================================================
// FUNÇÃO: current_output
// DESCRIÇÃO: Calcula e define o ciclo de trabalho do PWM para a saída de
corrente
//            (Fonte de Howland). Também retorna o valor de corrente de saída
para exibição.
// PARÂMETROS:
//   inType: Tipo de entrada (0=Tensão, 1=Corrente).
//   in: Ponteiro para armazenar o valor da entrada convertida.
//   inMin: Mínimo da faixa de entrada.
//   inMax: Máximo da faixa de entrada.
//   outMin: Mínimo da faixa de saída de corrente desejada.
//   outMax: Máximo da faixa de saída de corrente desejada.
// RETORNO: O valor de corrente de saída calculado.
// ======================================================================
float current_output(int inType, float* in, float inMin, float inMax, float
outMin, float outMax) {
  float scale_inMin = 0;  // Mínimo da escala de entrada (em ADC)
  float scale_inMax = 0;  // Máximo da escala de entrada (em ADC)
  float valueRead = 0;    // Leitura bruta do ADC da entrada
```

```
  switch (inType) {
    case 0:  // Entrada Tensão (pino A1)
      valueRead = analogRead(A1);
      *in = converterTensao(valueRead, inMin, inMax);
      scale_inMin = (1023.0 * inMin) * (1.0 / VoltageMax) * (1.0 / VoltageGain);
      scale_inMax = (1023.0 * inMax) * (1.0 / VoltageMax) * (1.0 / VoltageGain);
      break;
    case 1:  // Entrada Corrente (pino A2)
      valueRead = analogRead(A2);
      *in = converterCorrente(valueRead, inMin, inMax);
      // Ajusta escalas de entrada para Corrente
      scale_inMin = (1023.0 * inMin * mA * ResIn) * (1.0 / VoltageMax); //
VoltageMax aqui parece ser a referência do ADC ou limite do condicionamento
      scale_inMax = (1023.0 * inMax * mA * ResIn) * (1.0 / VoltageMax);
      break;
    default:
      return -1;
  }

  // Calcula as escalas de saída para o PWM (ciclo de trabalho)
  // Os fatores (1.0/5), 135 e (1.0/resBit)*2 parecem ser calibrações
  // para mapear a corrente desejada para o valor de duty cycle correto
  // para a Fonte de Howland. O 135 pode ser o valor TOP do PWM para essa saída
específica.
  scale_outMin = outMin * (1.0 / 5) * 135 * (1.0 / resBit) * 2;
  scale_outMax = outMax * (1.0 / 5) * 135 * (1.0 / resBit) * 2;

  // Mapeia a leitura bruta do ADC para o valor de controle do PWM
  float control = mapfloat(valueRead, scale_inMin, scale_inMax, scale_outMin,
scale_outMax);
  // Mapeia a leitura bruta do ADC para o valor de saída de corrente para
exibição
  float saida_conv = mapfloat(valueRead, scale_inMin, scale_inMax, outMin,
outMax);

  set_dutycycle(10, control); // Define o duty cycle no pino 10 (OC1B) para a
Fonte de Howland

  return saida_conv;
}
```

INSTITUTO FEDERAL
Santa Catarina

Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
INSTITUTO FEDERAL DE SANTA CATARINA

JOÃO VICTOR OLIVEIRA CORDEIRO DE SOUZA

PROTÓTIPO DE UM CONVERSOR BIDIRECIONAL DE ESCALA CONFIGURÁVEL PARA SENSORES ANALÓGICOS INDUSTRIAIS

Este trabalho foi julgado adequado para obtenção do título em Bacharel em Engenharia Elétrica, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

Joinville, 01 de dezembro de 2025.

Prof. Carlos Toshiyuki Matsumi, Dr. Sc.
Orientador
Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

Prof. Janderson Duarte, Dr. Sc.
Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

Prof. Jeferson Luiz Curzel, Dr. Sc.
Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina