Etec Paulino Botelho

Habilitação Profissional Técnica de Nível Médio de Técnico em Eletrônica

Breno Lucas Chaves
Guilherme Henrique Scatolin de Oliveira
Isabelle de Lima Pacheco
João Victor Zago Garcia
Samuel Junior Casseta

Prensa eletropneumática com acionamento via sensor óptico

Breno Lucas Chaves
Guilherme Henrique Scatolin de Oliveira
Isabelle de Lima Pacheco
João Victor Zago Garcia
Samuel Junior Casseta

Prensa eletropneumática com acionamento via sensor óptico

Trabalho de Conclusão de Curso apresentado à Etec Paulino Botelho, como requisito parcial para a obtenção do título de Técnico em Eletrônica.

Orientador: Prof.Valter Cesar Govoni

São Carlos

2025

"O mar, uma vez que lança seu feitiço, aprisiona a pessoa em sua rede de maravilhas para sempre" - Jacques-Yves Cousteau

AGRADECIMENTOS

Agradecemos primeiramente a Deus, pela força, saúde e perseverança concedidas a cada um de nós durante todo o desenvolvimento deste trabalho.

Expressamos nossa sincera gratidão às nossas famílias, que nos apoiaram em todos os momentos, oferecendo compreensão, incentivo e acolhimento nos períodos mais desafiadores. Sem esse suporte, nada disso seria possível.

Agradecemos também ao nosso orientador Prof. Valter Cesar Govoni, que, com paciência e dedicação, contribuiu significativamente para a construção deste projeto, guiando-nos com suas orientações, conhecimentos e sugestões fundamentais.

CHAVES, Breno Lucas; OLIVEIRA, Guilherme Henrique Scatolin de; PACHECO, Isabelle de Lima; GARCIA, João Victor Zago; CASSETA, Samuel Junior.
Prensa eletropneumática com acionamento via sensor óptico. 2025. Trabalho de Conclusão de Curso (Técnico em Eletrônica) – Etec Paulino Botelho, São Calos, 2025.

RESUMO

O projeto apresenta o desenvolvimento de uma prensa eletropneumática automatizada com sensor óptico destinada ao processo de reciclagem de latas de alumínio, visando reduzir o esforço físico do operador, eliminar riscos ergonômicos e aumentar a eficiência produtiva. O estudo aprofunda-se nos princípios de sistemas pneumáticos, seguindo normas de segurança. Com o desenvolvimento do projeto, incluiu a seleção, o dimensionamento e a integração dos componentes mecânicos, pneumáticos e eletrônicos, além da elaboração de esboços, moldes e diagramas que orientaram a montagem do protótipo. O sistema foi programado utilizando o microcontrolador ESP32. Após a montagem do circuito pneumático e elétrico, realizaram-se testes para avaliar o funcionamento do ciclo automático, envolvendo detecção por sensor óptico. Os resultados demonstraram que o sistema respondeu adequadamente às rotinas de acionamento e controle, embora tenha apresentado insuficiência de força para realizar a prensagem completa da latinha real devido à limitação no fluxo de ar fornecido ao cilindro. A demonstração do ciclo foi realizada com papeis em formato de cilindro, comprovando a eficiência da lógica de comando, da integração e detecção. Conclui-se que o protótipo valida o funcionamento do sistema eletropneumático, mas requer redimensionamento dos componentes pneumáticos, especialmente cilindro, válvula e tubulação, para atingir a força necessária à compactação efetiva das latas.

Palavras-chave: Prensa eletropneumática. Automação Pneumática. Sensor óptico. Reciclagem de alumínio. ESP32.

CHAVES, Breno Lucas; OLIVEIRA, Guilherme Henrique Scatolin de; PACHECO, Isabelle de Lima; GARCIA, João Victor Zago; CASSETA, Samuel Junior. Electropneumatic press with optical sensor activation. 2025. Trabalho de Conclusão de Curso (Técnico em Eletrônica) – Etec Paulino Botelho, São Calos, 2025.

ABSTRACT

The project presents the development of an automated electro-pneumatic press with an optical sensor for the aluminum can recycling process, aiming to reduce the operator's physical effort, eliminate ergonomic risks, and increase production efficiency. The study delves into the principles of pneumatic systems, following safety standards. The development of the project included the selection, sizing, and integration of mechanical, pneumatic, and electronic components, as well as the preparation of sketches, molds, and diagrams that guided the assembly of the prototype. The system was programmed using the ESP32 microcontroller. After assembling the pneumatic and electrical circuit, tests were carried out to evaluate the operation of the automatic cycle, involving detection by an optical sensor. The results showed that the system responded adequately to the activation and control routines, although it lacked sufficient force to completely press the actual can due to the limitation in the air flow supplied to the cylinder. The cycle was demonstrated using cylinder-shaped paper, proving the efficiency of the control logic, integration, and detection. It was concluded that the prototype validates the operation of the electro-pneumatic system, but requires resizing of the pneumatic components, especially the cylinder, valve, and tubing, to achieve the force necessary for effective can compaction.

Key-words: Electropneumatic press. Pneumatic automation. Optical sensor. Aluminum recycling. ESP32.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS OU SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
NR 12	Norma Regulamentadora n° 12
NR 17	Norma Regulamentadora n° 17
Mm	Milimetro
FRL	*Filter, Regulador and Lubricator (filtro, regulador e lubrificante)*
PNP	Positive-Negative-Positive (Positivo-negativo-Positivo)
Wi-Fi	Wireless Fidelity
HTTP	HyperText Transfer protocol
I/O	Input/Output (Entrada/Saída)
ESP32	Embedded System Plataform 32 (microcontrolador de Espressif)
Vcc	Tensão contínua
Bar	Unidade de pressão pneumática
CPU	Central Processing Unit (Unidade Central de Processamento)
Hz	Hertz (frequência)
AC	Alternating Current (Corrente Alternada)
DC	Direct Current (Corrente Contínua)

SUMÁRIO

1 INTRODUÇÃO

Este projeto se aplica uma prensa eletropneumática com sensor óptico para automatizar o processo de reciclagem de latas de alumínio, melhorando particularmente a compactação dos materiais. A ideia não só melhora o fluxo, como também reduz drasticamente o esforço físico exigido ao trabalhador, eliminando praticamente os riscos ergonómicos e a fadiga causados por movimentos repetitivos. Naturalmente, como muitos campos dependem diretamente da eficiência do processo de estampagem, o uso da prensa pneumática reduz o tempo de ciclo da peça, melhorando não apenas os resultados do processo, mas também toda a cadeia de suprimentos como um todo. À vista disso, o projeto combina inovação tecnológica e automação de processos com produtividade e saúde ocupacional, oferecendo um ambiente industrial mais seguro, eficaz e sustentável.

1.1 Objetivo

Sendo amplamente utilizada em processos industriais como corte e montagem, seu diferencial está na integração de um sensor óptico, que garante que o acionamento ocorra apenas quando a peça estiver corretamente posicionada, aumentando a segurança do operador e reduzindo falhas no processo. Além de proporcionar maior confiabilidade, esse sistema contribui para a padronização da produção, assegurando repetibilidade e qualidade no produto final. O uso da automação também reduz o tempo de ciclo, melhora a eficiência energética e diminui a dependência de operações manuais. Dessa forma, a prensa eletropneumática com sensor óptico representa uma solução moderna e eficaz, alinhada às demandas da Indústria 4.0, que busca integrar tecnologia, segurança e produtividade em ambientes industriais cada vez mais inteligentes.

1.1.1 Objetivo especifico

O principal objetivo deste projeto, é proporcionar a automatização no processo de reciclagem de latas de alumínio, assim facilitando o processo de compactação dos materiais.

1.2 Justifivativa

No projeto, a implementação visa que haja menos esforço físico do colaborador na hora da execução, evitando assim o risco ergonômico e esforço repetitivo, pois tais fatores a longo prazo podem ser fundamentais para que ocorram lesões ao colaborador, prejudicando a sua saúde e estado físico. Tendo em vista que na maioria das vezes outros processos são dependentes do processo de estampagem, a prensa pneumática irá auxiliar na redução de tempo da peça a ser estampada, não só otimizando o processo em si, como outros processos da empresa que estão vinculados ao processo de fabricação e estampagem dos botões.

FUNDAMENTOS TEÓRICOS

2.1 Sistemas pneumáticos

Os sistemas pneumáticos usam ar comprimido como energia para fazer máquinas e processos industriais funcionarem. De acordo com a Nepin, esses sistemas são conhecidos por serem fáceis de usar e por terem um custo operacional relativamente baixo. Por isso, eles são uma escolha bastante popular na automação industrial (Nepin, 2025).

No funcionamento típico, um compressor funciona assim: ele suga o ar do ambiente, o comprime e armazena essa energia em uma rede de distribuição. Antes de chegar aos atuadores, esse ar comprimido passa por uma unidade de preparação chamada FRL, que é responsável por filtrar as impurezas, ajustar a pressão e, se necessário, lubrificar o ar para proteger os componentes pneumáticos (Nepin, 2025).

Depois desse processo de preparação, o ar pressurizado é conduzido por tubulações até válvulas e atuadores pneumáticos. Esses dispositivos transformam a energia do ar em movimentos mecânicos controlados. Esses sistemas continuam sendo bastante utilizados na indústria porque oferecem respostas rápidas, são relativamente fáceis de manter e se integram facilmente com outros tipos de controle, como os eletrônicos. Isso faz com que sejam versáteis para diferentes aplicações e tamanhos de produção (Nepin, 2025).

Figura 1: Sistema pneumático

Fonte: mtibrasil.com.br

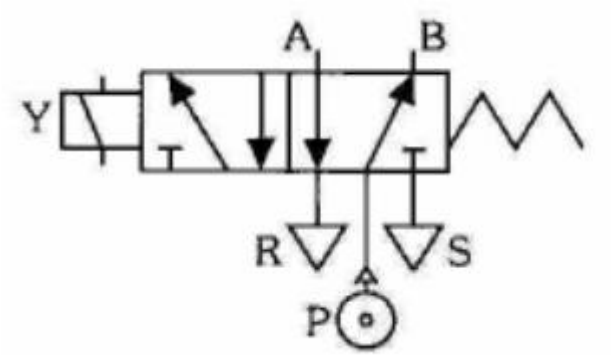
2.1.1 Válvulas Pneumáticas

Dispositivo empregado para regular o fluxo e a direção do ar comprimido em instalações industriais. Na prática, ela funciona como um pulso de decisão, um elemento de sistema que determina qual caminho deve tomar, viabilizando que atuadores, cilindros e outros possam efetuar movimentos mecânicos específicos.

Portanto, suas funções centrais são abrir, bloquear ou definir um novo rumo para o ar em questão. As válvulas pneumáticas são categorizadas pelo número de vias e de posições, podendo ser 2/2; 3/2; 5/2 ou 5/3. Cada sistema define quantos caminhos o ar, quantos estados a válvula pode assumir respectivamente.

As possibilidades de acionamento são manuais, mecânico, pneumático ou elétrico (com solenoide), sendo mais comum em instalações automatizadas, uma vez que possibilita rapidez no controle e conexão com sensoriamento e controladores mais avançados. Além disso, ela também pode regular a velocidade e a sequências dos movimentos da máquina, conferindo segurança e eficiência ao sistema. Sem ela, o ar comprimido teria pouca ou nenhuma direção útil, e os movimentos seriam completamente descoordenados.

Figura 2: Válvula 5/2 vias avanço por solenoide e retorno por mola

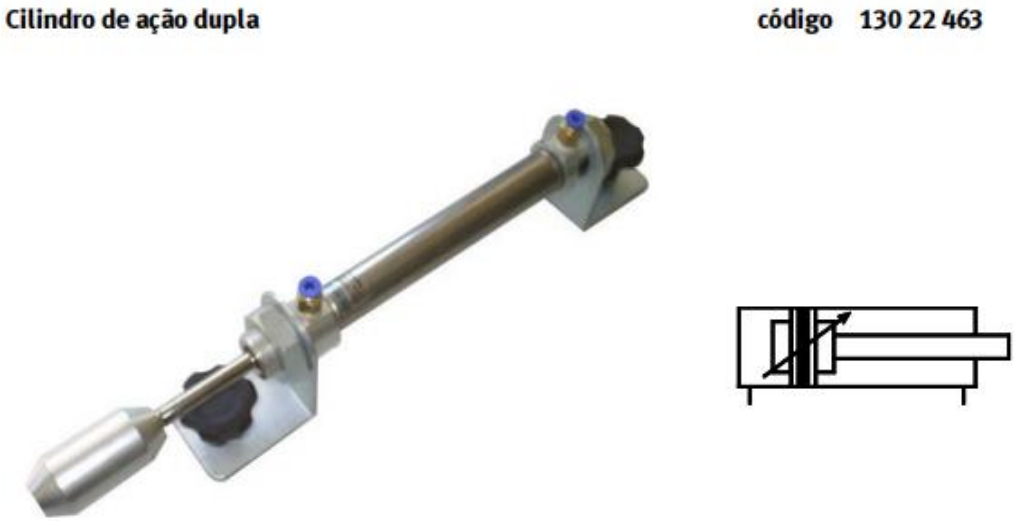


Fonte: archive.org

2.1.2 Cilindro de dupla ação

O cilindro pneumático de dupla ação é um tipo de atuador que performa movimentos linearmente controlados no avanço e retorno. Sustenta-se pela dualidade de duas câmaras internas, sendo estes espaços vedados e rebatidos individualmente com ar comprimido através de entrada de ar. Conforme o fluxo de ar através da alimentação da entrada de ar para a câmara traseira que o pistão se desloca para a frente, “empurrando” a haste para fora. No momento em que o fluxo de ar base para a câmara dianteira, o pistão é “puxado” para o mesmo lado “puxando” a haste interna.

Figura 3: Cilindro de dupla ação



Fonte: Manual bancada FESTO.

Dados técnicos:

- avanço e retorno pneumáticos;
- camisa de aço inoxidável e haste microrroletada;
- êmbolo magnético para detecção por sensores sem contato físico;
- came de acionamento em alumínio, montado na ponta da haste;
- amortecimento regulável nas posições finais de curso.
- Dimensões:

. diâmetro do êmbolo: 20 mm;

. diâmetro da haste: 8 mm;

. curso: 100 mm;

- pressão máxima de trabalho: 10 bar.

2.2 Sistema Eletropneumático

A eletropneumática é uma tecnologia que aplica o controle elétrico à engenharia pneumática e é utilizada em diversas áreas da indústria. De acordo com o estudo "*Design and Simulation of Electro-Pneumatic Sequential System Using Fluid Sim Software*", esse método se baseia no uso da energia do ar comprimido e dos comandos de controle da energia elétrica para obter sequências de movimentos confiáveis e repetitivas em máquinas e equipamentos industriais. Neste tipo de automação, a parte elétrica é responsável pelo comando: sensores, botões, chaves de fim de curso, relés e temporizadores enviam sinais que acionam as eletroválvulas. Essas válvulas, por sua vez, orientam o fluxo de ar para cilindros pneumáticos, resultando em movimento linear ou rotativo. O estudo destaca que tal padrão é empregado em grande parte devido à sua simplicidade, baixo custo e velocidade de resposta. O artigo também afirma que circuitos eletropneumáticos sequenciais podem ser construídos como no exemplo de dois cilindros desenvolvidos com a sequência A+ → B+ → B– → A–. Os autores afirmam que a lógica elétrica pode gerenciar as fases dos movimentos pneumáticos de maneira ordenada e eficaz, permitindo que as máquinas realizem tarefas repetitivas com precisão (ISHACK et al., 2024).

NR 12 – SEGURANÇA NO TRABALHO EM MÁQUINAS E EQUIPAMENTOS

Conforme o Anexo VIII da NR-12, os sistemas de segurança utilizados em máquinas — incluindo prensas pneumáticas — devem ser projetados para alcançar um nível adequado de desempenho, considerando uma integração entre diferentes tecnologias, como componentes mecânicos, hidráulicos, pneumáticos e eletrônicos. A norma também orienta que a seleção de cada elemento do sistema leve em conta a categoria correspondente e sua tecnologia, garantindo que a função de segurança permaneça confiável mesmo diante de falhas ou perda de desempenho de algum componente. Essa abordagem reforça a necessidade de combinar proteções físicas, dispositivos eletropneumáticos, sensores eletrônicos, sistemas lógicos de controle e métodos redundantes, assegurando que o operador não seja exposto à zona de risco da prensa em nenhuma etapa do ciclo de operação.

NR 17

A NR-17 estabelece diretrizes para adaptar as condições de trabalho às características psicofisiológicas dos trabalhadores, a fim de prevenir doenças ocupacionais, fadiga excessiva e risco associado à adoção de posturas inadequadas, esforço repetitivo ou condições ambientais desfavoráveis. A norma, portanto, estabelece parâmetros para organização do trabalho, levantamento e transporte manual de cargas, mobiliário, condições ambientais e até mesmo uso de tecnologias, determinando que o desempenho de quaisquer atividades deva ocorrer de acordo com o conforto, segura e eficientemente de maneira quantitativa ou qualitativa. Para máquinas e equipamentos, como mostra a NR-17 implica a concepção de sistemas de operação com esforço desnecessário, translação além do necessário ou movimentos em tempos redundantes. Isso implica, por exemplo, botões de acionamento e sensores posicionados corretamente, interfaces de controle e livre acesso somente quando o operador mantém a postura correta o tempo todo em um ciclo de trabalho apropriado. A norma também estabelece que o tempo do ciclo seja o mais curto possível, de preferência sem exposição direta do operador à operação, e que o tempo de exposição seja compatível com a capacidade humana, fornecendo intermitência de tarefas ou pausas, alterando atividades ou ritmo operacional.

2.3 Sensor óptico

Os sensores ópticos conhecidos também por sensores fotoelétricos, funcionam com base na emissão e recepção de luz para detectar objetos, medir distâncias ou monitorar movimentos no ambiente industrial. Eles são bastante utilizados porque são capazes de detectar sem contato físico, com alta precisão e velocidade. As tecnologias mais populares são luz vermelha, laser e infravermelho, cada uma com sua especificidade aplicável a objetos opacos, pequenos ou até mesmo transparentes.

Para o sensor óptico, existem três princípios básicos de operação: barreira, difuso e reflexivo. A barreira - o emissor e o receptor estão separados e um feixe de luz constante é formado, que é interrompido quando um objeto passa entre eles.

O tipo difuso possui emissor e receptor no mesmo corpo e utiliza-se da reflexão da luz no próprio objeto para realizar a detecção, sendo assim, ele é mais influenciado pelas características da superfície, principalmente cor e textura.

Por último, o reflexivo possui um refletor colocado de frente para o sensor para que o raio luminoso se reflita no refletor e volte ao receptor, dessa forma consegue-se maior distância e menor influência da cor do objeto.

Figura 4: **Sensor óptico**

Fonte: Manual bancada FESTO.

Dados técnicos:

- distância de sensorização: até 300 mm;
- tensão de alimentação: 10 a 30 Vcc;
- freqüência máxima: 100 Hz;
- sinal de saída: 24 Vcc PNP;
- LED indicador de operação;
- cabo elétrico equipado com pinos do tipo banana de 4 mm (incluso);
- positivo: Vermelho;
- negativo: azul;
- saída PNP: preto.

2.4 ESP32 controlador eletrônico do sistema

Este projeto utiliza o ESP32 Wroom, um microcontrolador moderno, programável e com recursos avançados de conectividade. O uso como dispositivo de controle obedece a uma tendência recente da da automação compacta, que traz maior flexibilidade, menor custo e baixo consumo de energia.

Segundo o blog técnico da Eletrogate(2024), o ESP32 é um microcontrolador de alto desempenho, obtendo WI-Fi, Bluetooth, processadores dual-core e diversos recursos de I/O. Ele permite a implementação do sistema embarcados complexos, entre eles: Leitura de sensores digitais e analógicos; processamento rápido de sinais; conexão com redes Wi-Fi para interação via WebSocket, MQTT e HTTP; Controle de atuadores e dispositivos externos (Eletrogate, 2018).

Figura 5: **Esp32**

Fonte: blog.eletrogate.com

Metodologia

Este capítulo apresenta o desenvolvimento do sistema de automação aplicado à prensa eletropneumática, estruturado a partir dos princípios teóricos discutidos anteriormente. São descritos os critérios adotados para a seleção e o dimensionamento dos componentes pneumáticos e eletrônicos, bem como a definição da lógica de comando responsável pela operação do conjunto. Nesse sentido, concentra-se na descrição detalhada do circuito pneumático, enfatizando particularmente a iniciação do processo, o acionamento dos dispositivos e seu desacionamento, retornando à posição inicial.

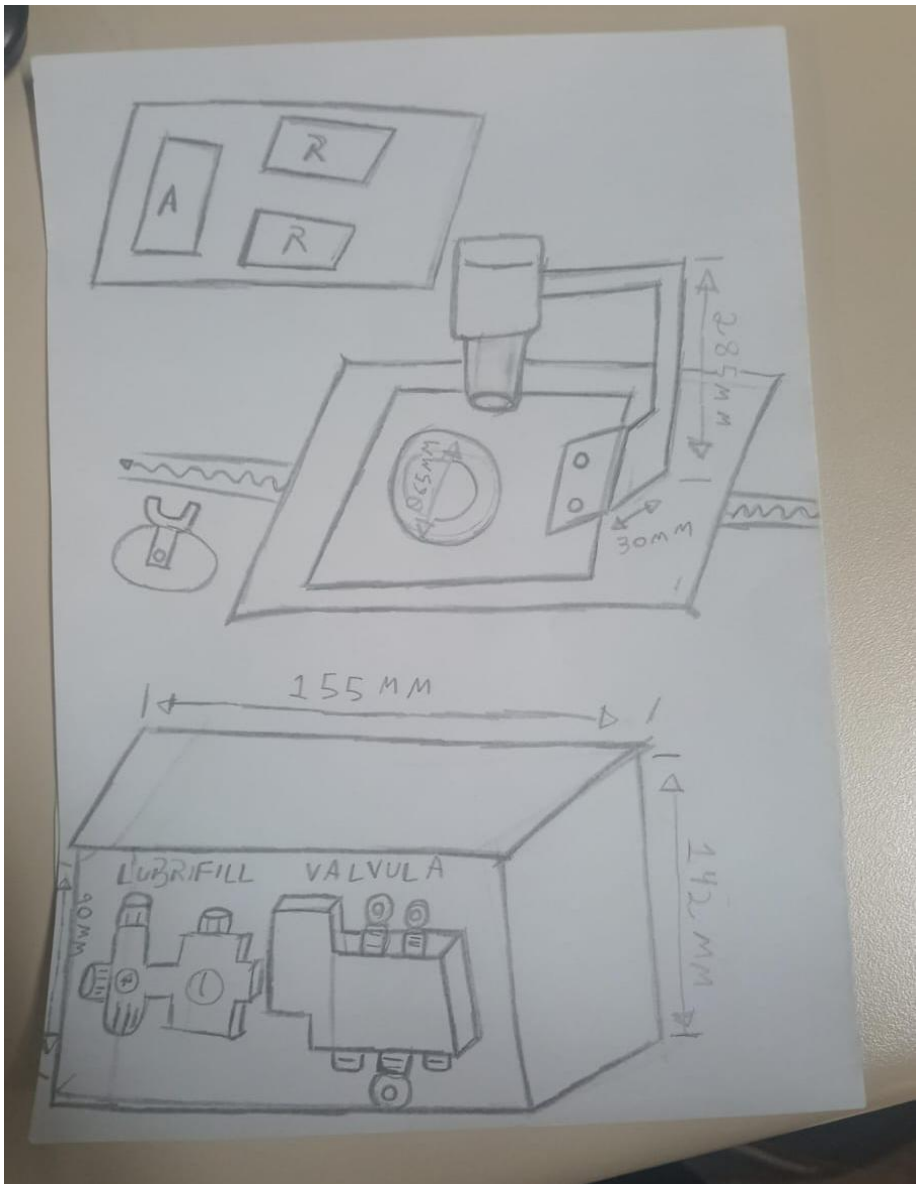
As etapas apresentadas neste capítulo formam a base para um funcionamento sólido, que atende às demandas do setor industrial e ao desempenho esperado do projeto.

3.1 Projeto do Circuito

Primeiramente, é fundamental garantir que o sistema opere com eficiência, segurança e confiabilidade. Cada dispositivo foi escolhido considerando suas características técnicas e a sua capacidade de atender às exigências do processo de prensagem.

Ao iniciar o projeto, utilizamos um papel como protótipo inicial para visualizar a expectativa do resultado final (figura 6). Esse esboço funcionou como um mapa orientador, pois nele representamos a disposição dos componentes, assim como a área destinada ao posicionamento da latinha, .

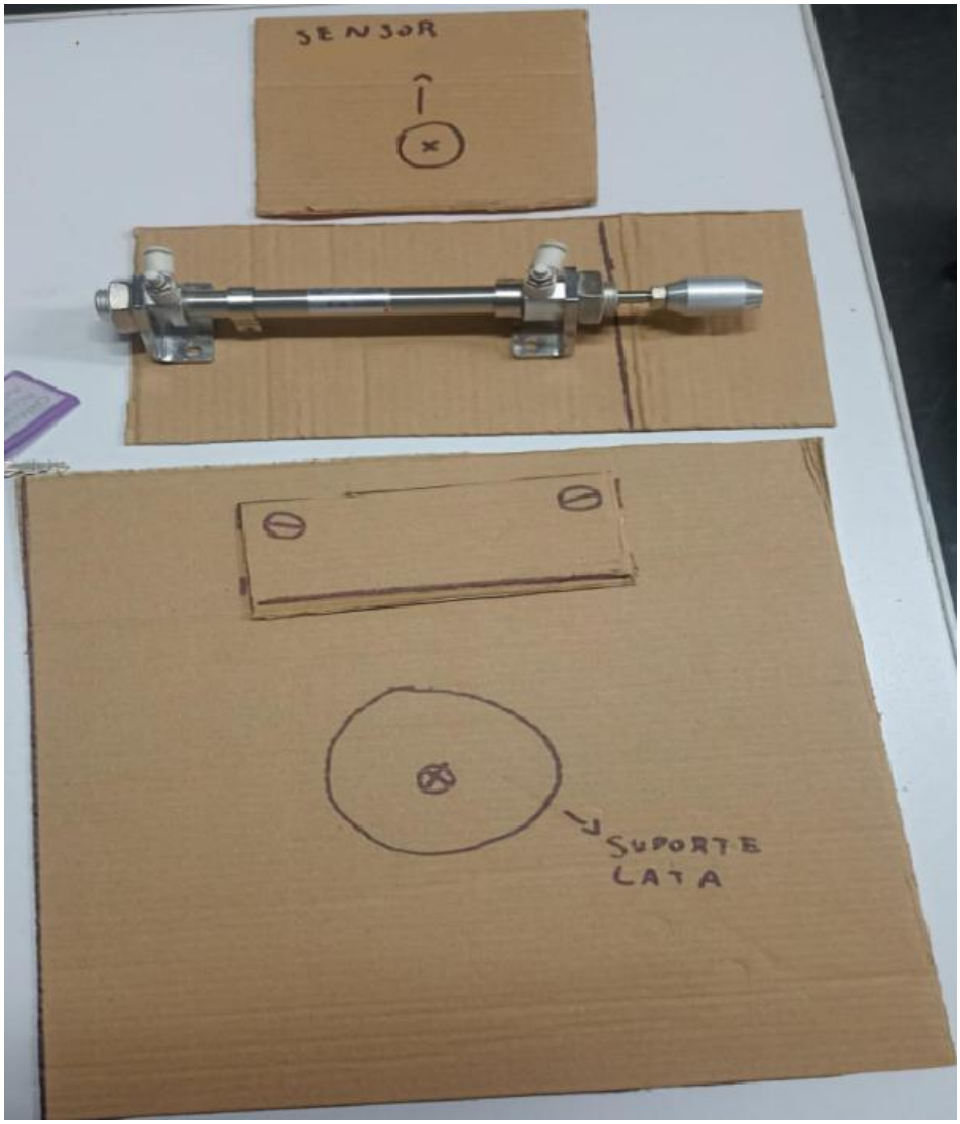
Figura 6: **esboço inicial no papel**



Fonte: Desenvolvido pelo autor.

Conforme a figura 7, foi realizada a medição do suporte utilizando um molde confeccionado em papelão, a fim de determinar com precisão o comprimento e a largura da chapa metálica destinada à fixação do cilindro pneumático e do sensor óptico. Essa etapa permitiu estabelecer as dimensões adequadas para o posicionamento dos componentes, garantindo alinhamento, estabilidade estrutural e correta instalação, demonstrado na figura abaixo.

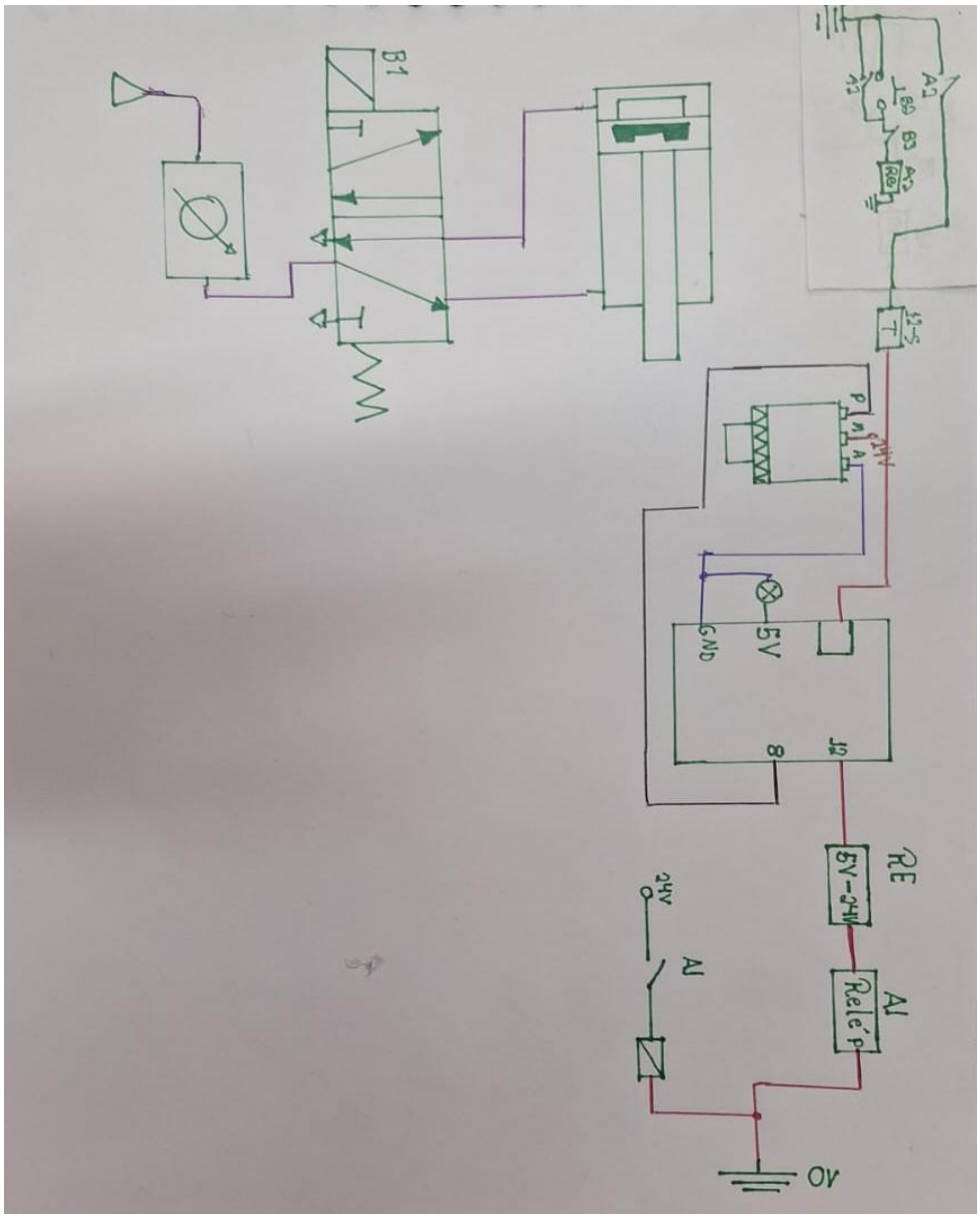
Figura 7: **Molde de papelão**



Fonte: Desenvolvido pelo autor.

Após a medição, foi criado um desenho preliminar em papel, mostrando a estrutura e a disposição dos componentes necessários para montar o sistema eletropneumático. Esse esboço ajudou a identificar as dimensões e os materiais, além de servir como uma referência inicial para planejar a instalação, como pode ser visto na figura abaixo.

Figura 8: **Diagrama pneumático**

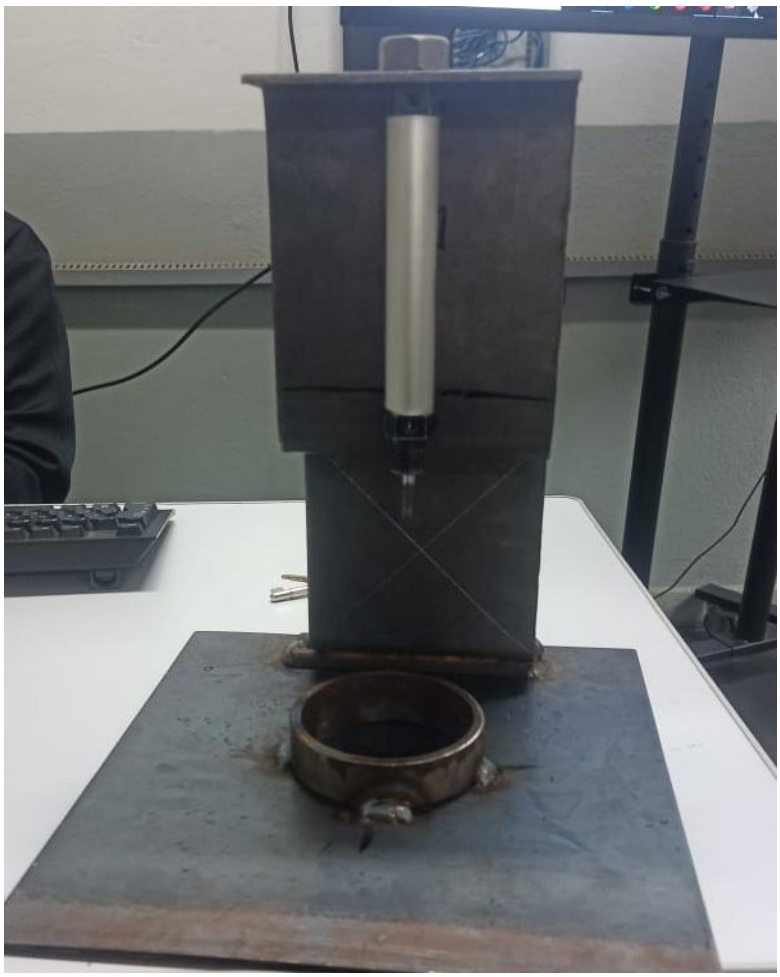


Fonte: Desenvolvido pelo autor.

O diagrama mostra de forma simples os componentes que acionam, controlam e fazem o retorno do cilindro pneumático usado na prensa. No lado esquerdo do desenho, evidencia-se o circuito pneumático, que inclui uma válvula direcional 5/2 vias com retorno por mola, o cilindro de dupla ação e as linhas que levam ar comprimido até eles. Também está indicado um sensor óptico, que identifica a peça e dá o comando para iniciar o ciclo de operação.

No lado direito da folha, você encontrará o diagrama elétrico que está ligado ao controle da válvula solenoide. Logo, estão destacados o relé que faz a comutação, a fonte de 24 V que alimenta a bobina da válvula, o interruptor que ativa o sistema, além do módulo de controle eletrônico que gera o sinal de acionamento. Além disso, também inclui as referências de tensão, o aterramento e as conexões entre os dispositivos, permitindo visualizar como o circuito elétrico interage com o circuito pneumático.

Figura 9: **chapa metálica**



Fonte: Desenvolvido pelo autor.

Ao final, foi realizado a chapa metálica como mostra a figura à cima.

3.2 Programação do sistema

Para que os dados fossem processados e transmitidos, utilizou-se o microcontrolador ESP32 VROOM, escolhido por sua elevada capacidade de processamento, conectividade Wi-Fi integrada. A comunicação entre os dispositivos foi estruturada a partir do protocolo MQTT, que, por suas características, não oferece comunicação direta com aplicações web. Diante dessa limitação, tornou-se necessária a implementação de um mecanismo de encapsulamento para a transferência das informações até a interface do usuário.

O arquivo webSocket.py foi responsável pela configuração do protocolo de comunicação, realizando o encapsulamento dos dados provenientes do MQTT para o formato WebSocket. Essa etapa permitiu estabelecer a ponte entre o servidor intermediário e a web, garantindo a transmissão contínua e organizada das informações. Ao mesmo tempo, o código ltcc_webSocket_v2.py executou a função de contador, recebendo os pulsos emitidos pelos sensores, processando-os e enviando os valores ao broker MQTT. Dessa forma, assegurou-se que os dados coletados fossem devidamente analisados e disponibilizados para as etapas posteriores.

Na camada de apresentação, o script script_v2.js realizou o recebimento dos dados encapsulados pelo WebSocket, estruturando-os em uma tabela dinâmica exibida na página index.html.

4 RESULTADOS

Os resultados obtidos no trajeto do projeto, provam que a prensa eletropneumática opera bem, demonstrando a eficiência dos componentes para automação. Depois dos testes, o seu funcionamento permitiu observar o comportamento do sistema, como ele interage em condições reais de operação a mecânica e eletropneumática.

A Figura 10 apresenta o resultado final do molde metálico, evidenciando a conformação da latinha após a atuação do cilindro pneumático.

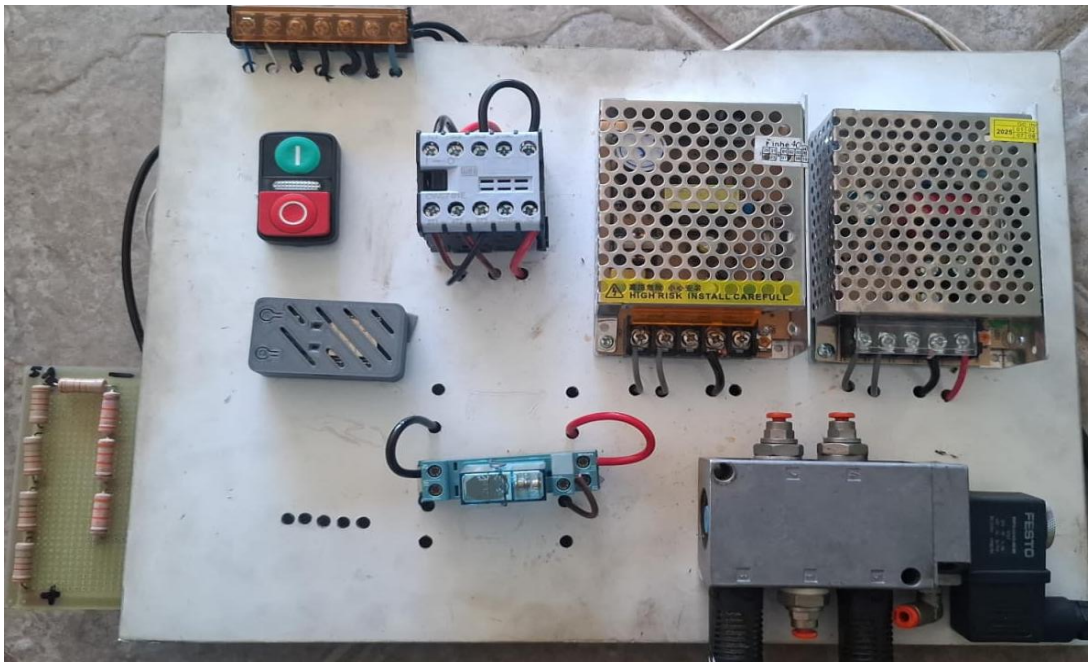
Em seguida, a Figura 11 mostra a montagem pronta e a organização da parte elétrica/eletropneumática.

Por fim, a Figura 12 apresenta a tabela com o custo final do projeto, reunindo os valores individuais dos componentes utilizados.

Figura 10: **chapa de metal com a latinha**

Fonte: Desenvolvido pelo autor.

Figura 11: **sistema eletropneumático**



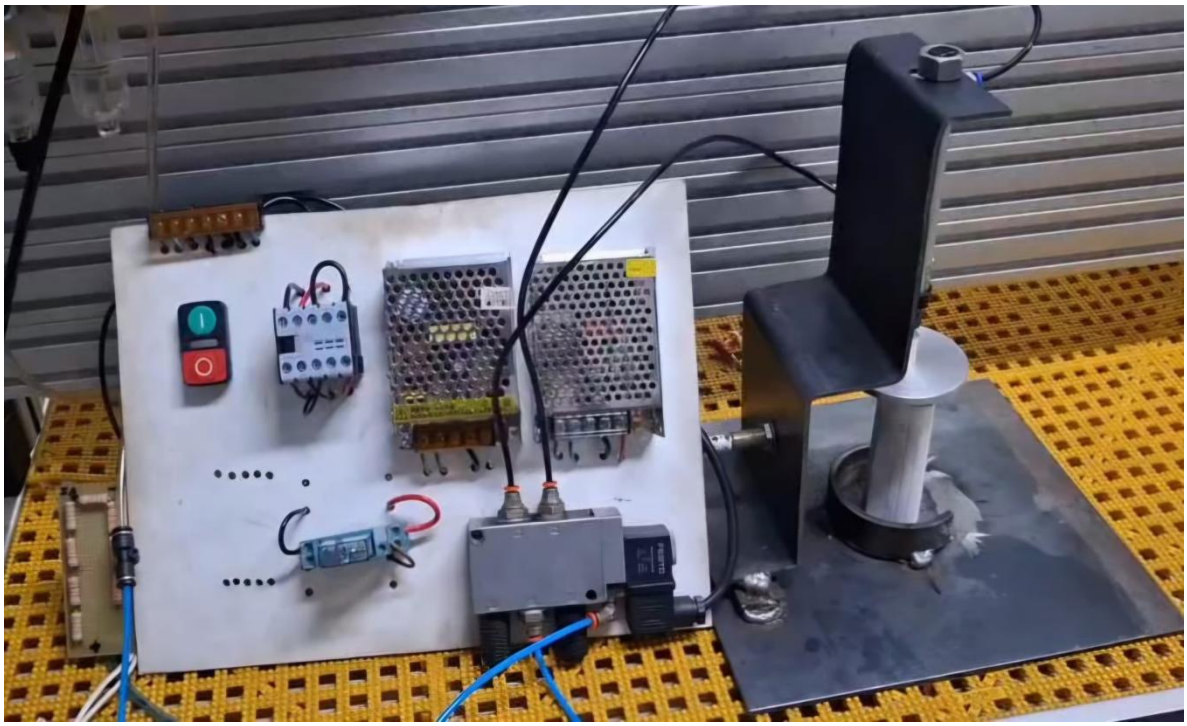
Fonte: Desenvolvido pelo autor.

Tabela 1: **Orçamento geral**

Nome do Componente	Quantidade	Preço
Fonte 12V chaveada 5A	1	40,00
Fonte 24V chaveada 5A	1	40,00
Relé Finder 40.31 + Acoplamento 95.63	1	87,50
ESP32 Wroom	1	55,90
Válvula 5/2 vias	1	177,81
Sensor Óptico	1	661,00
Mini Contator CW07 01E	1	64,99
Chapa de 3mm	2	70,00
3 metros de fio de 3mm de diâmetro	1	15,00
Conector Wago221	3	17,00
Resistor 33k	8	8,00
Placa de Cobre Perfurada	1	16,00
Botão	1	73,52
	TOTAL	**R$ 1.326,72**

Fonte: Dados da pesquisa, 2025.

Figura 12: **Projeto completo**



Fonte: Desenvolvido pelo autor.

Apesar dos testes individuais realizados, ao integrar todos os componentes e executar o sistema completo, percebe-se que o cilindro pneumático apresentou falha no processo de prensagem da latinha. Observou-se que o cabo de ar não passava a quantidade certa de ar que o cilindro precisava, comprometendo o desempenho do conjunto pneumático.

Embora a substituição desse cabo pudesse aumentar o fluxo de ar, essa alteração implicaria a necessidade de reajustes em outros componentes do circuito, como a válvula solenoide, o próprio cilindro e a chapa de metal para uma maior, a fim de suportar um mais robusto.

Dessa forma, a força disponível não era suficiente para realizar o esmagamento do material metálico. Entretanto, optou-se por demonstrar o funcionamento da prensa utilizando folhas de papel A4 (em formato de cilindro, com o propósito de se assemelhar a uma latinha), permitindo evidenciar o ciclo operacional do equipamento mesmo sem a força necessária para comprimir a latinha.

5. Conclusão

Ao longo do processo, foi possível averiguar a lógica de acionamento, o ciclo automático e a interação entre os componentes, demonstrando que o sistema responde de forma eficaz às condições programadas. Mesmo diante das limitações de força encontradas durante a execução, o protótipo cumpriu seu papel, evidenciando a eficiência do comando desenvolvido e a integração entre as etapas de detecção, controle e atuação.

Ademais, recomenda-se para trabalhos futuros reavaliar o dimensionamento dos componentes pneumáticos, especialmente o cilindro, de forma a aumentar a força disponível para a prensagem. A aplicação de uma válvula solenoide proporcional compatível com um cilindro, garantindo um ciclo mais eficiente. Além disso, é importante revisar o diâmetro dos cabos de ar para garantir que venha a quantidade certa e não prejudique quaisquer dos componentes por falta de ar.

REFERÊNCIAS

NEPIN, 2025. Disponível em: https://www.nepin.com.br/blog/solucoes-industriais/. Acesso em: 12 Out. 2025.

BONACORSO, Nelson Gauze; NOLL, Valdir. **Automação Eletropneumática**. 9°Edição. São Paulo: Erica, 2006.

MTIBRASIL, Disponível em: https://mtibrasil.com.br/blog/pneumatica-geral/pneumatica/. Acesso em: 17 Out. 2025.

ROGGIA, Leandro; FUENTES, Rodrigo Cardozo. **Automação Industrial**. Universidade Federal de Santa Maria, Colégio Técnico Industrial de Santa Maria, Rede e-Tec Brasil, 2016.

ELETROGATE, 2018 Disponível em: https://blog.eletrogate.com/conhecendo-o-esp32-introducao-1/ Acesso em: 21 Out. 2025.

INTERNATIONAL RESEARCH JOURNAL OF ENGINNERING AND TECHNOLOGY (IRJET). 2024.V.11. Disponível em: https://www.irjet.net/archives/V11/i3/IRJET-V11I321.pdf

BRASIL. Ministério do Trabalho e Emprego. Norma Regulamentadora NR-17 - Ergonomia. Brasília: MTE, 2022. Disponível em: https://www.gov.br/trabalho-e-emprego/pt-br/acesso-a-informacao/participacao-social/conselhos-e-orgaos-colegiados/comissao-tripartite-partitaria-permanente/normas-regulamentadora/normas-regulamentadoras-vigentes/nr-17-atualizada-2023.pdf. Acesso em: 25 Out. 2025.

BRASIL. Ministério do Trabalho e Emprego. Norma Regulamentadora NR-12 – Segurança no Trabalho em Máquinas e Equipamentos. Brasília: MTE, 2022. Disponível em:https://www.gov.br/trabalho-e-emprego/pt-br/acesso-a-informacao/participacao-social/conselhos-e-orgaos-colegiados/comissao-tripartite-partitaria-permanente/arquivos/normas-regulamentadoras/nr-12-atualizada-2022-1.pdf. Acesso em 26 Out. 2025.

APÊNDICES

Apêndice I Código utilizado no Esp32 Vroom

Código A.1 - ltcc_webSocket_v2.py

```
import network
import uasyncio as asyncio
import ujson
from machine import Pin
from utime import localtime
from time import sleep
import os

try:
    from webSocket import websocket_handshake, Websocket
except ImportError:
    print("Erro: Módulo 'webSocket' não encontrado. Certifique-se de que
está instalado.")
    raise

# ======== CONFIGURAÇÕES ========
SSID = "Galaxy A54 5G 832D"
PASSWORD = "casseta21"
PIN_ENTRADA = 21
ARQUIVO_PULSOS = "pulsos.json"

# ======== Gerenciamento de Arquivo JSON ========
def carregar_pulsos():
    """Carrega contagens salvas do arquivo JSON, ou retorna um dicionário
vazio."""
    try:
        with open(ARQUIVO_PULSOS, "r") as f:
            return ujson.load(f)
    except:
        return {}

def salvar_pulsos(dados):
    """Salva o dicionário de contagens no arquivo JSON."""
    try:
        with open(ARQUIVO_PULSOS, "w") as f:
            ujson.dump(dados, f)
    except Exception as e:
        print(f"Erro ao salvar pulsos: {e}")

# ======== Conectar ao Wi-Fi ========
def conectar_wifi(nome_rede, senha_rede):
    conexao_wifi = network.WLAN(network.STA_IF)
    conexao_wifi.active(True)
    if not conexao_wifi.isconnected():
        print("Conectando a rede Wi-Fi...")
        conexao_wifi.connect(nome_rede, senha_rede)
        timeout = 10
        start = utime.time()
        while not conexao_wifi.isconnected():
            if utime.time() - start > timeout:
                print("\nFalha ao conectar ao Wi-Fi!")
                return None
            print(".", end="")
            sleep(0.5)
   print("\nConectado ao Wi-Fi!")
```

```
    ip_esp32 = conexao_wifi.ifconfig()[0]
    print("IP do ESP32:", ip_esp32)
    return ip_esp32

# ======== Cria objeto com data/hora e contador ========
def criar_objeto(contador, data):
   objeto = {
        "contador": contador,
        "data": data,
    }
    return objeto

# ======== Função de contagem de pulsos ========
contador = 0
ultimo_estado = 0
historico_pulsos = carregar_pulsos()  # Carrega contagens salvas na
inicialização

def contar_pulso(pino):
    global contador, ultimo_estado, historico_pulsos
    estado_atual = pino.value()
    if estado_atual == 1 and ultimo_estado == 0:  # Detecta borda de subida
        contador += 1
        tempo = localtime()
        data_str = "{:02d}/{:02d}/{:04d}".format(tempo[2], tempo[1],
tempo[0])
        historico_pulsos[data_str] = contador
        salvar_pulsos(historico_pulsos)
        print(f"Contagem: {contador} (Data: {data_str})")
    ultimo_estado = estado_atual

# ======== Servidor WebSocket ========
async def atender_cliente(conexao_entrada, conexao_saida):
    sucesso_handshake = await websocket_handshake(conexao_entrada,
conexao_saida)
    if not sucesso_handshake:
        print("Falha no handshake com cliente WebSocket")
        return
    conexao_ws = Websocket(conexao_entrada, conexao_saida)
    print("Cliente WebSocket conectado!")

    try:
        await conexao_ws.send(ujson.dumps({"historico": historico_pulsos}))
        print("Histórico enviado:", historico_pulsos)
    except Exception as e:
        print(f"Erro ao enviar histórico: {e}")

    try:
        ultimo_contador = contador
        ultima_data = None
        while True:
            tempo = localtime()
            data_str = "{:02d}/{:02d}/{:04d}".format(tempo[2], tempo[1],
tempo[0])
            if contador != ultimo_contador:
                objeto = criar_objeto(contador, data_str)
                print("Enviando:", objeto)
                await conexao_ws.send(ujson.dumps(objeto))
                ultimo_contador = contador
                ultima_data = data_str
            await asyncio.sleep(0.2)
    except Exception as erro:
```

```
        print("Erro no WebSocket:", erro)
    finally:
        conexao_ws.close()
        print("Conexão WebSocket encerrada")

# ======== Função principal ========
async def main():
    global contador, historico_pulsos
    tempo = localtime()
    data_str = "{:02d}/{:02d}/{:04d}".format(tempo[2], tempo[1], tempo[0])
    contador = historico_pulsos.get(data_str, 0)
    print(f"Contagem inicial para {data_str}: {contador}")

    # Configurar pino com pull-down interno e interrupção
    pino_entrada = Pin(PIN_ENTRADA, Pin.IN, Pin.PULL_DOWN)
    try:
        pino_entrada.irq(trigger=Pin.IRQ_RISING | Pin.IRQ_FALLING,
handler=contar_pulso)
        print("Interrupção configurada com pull-down interno!")

   except Exception as e:
        print(f"Erro na configuração da interrupção: {e}")

   ip_esp = conectar_wifi(SSID, PASSWORD)

    if not ip_esp:
        print("Não foi possível iniciar o servidor devido à falha na conexão
Wi-Fi")
        return
    print("Servidor rodando... IP:", ip_esp)
    server = await asyncio.start_server(atender_cliente, "0.0.0.0", 8080)
    print("Aguardando clientes WebSocket...")
    while True:
        await asyncio.sleep(1)

# ======== Execução ========
asyncio.run(main())
```

Código A.2 - webSocket.py

```
import ure as re
import ustruct as struct
import urandom as random
import ubinascii
import uhashlib
from ucollections import namedtuple
from micropython import const

# Função simples de log para debug
def log_debug(*args):
    print("[DEBUG]", *args)

# Opcodes
OP_CONT = const(0x0)
OP_TEXT = const(0x1)
OP_BYTES = const(0x2)
OP_CLOSE = const(0x8)
OP_PING = const(0x9)
OP_PONG = const(0xa)

# Close codes
CLOSE_OK = const(1000)
```

```
CLOSE_GOING_AWAY = const(1001)
CLOSE_PROTOCOL_ERROR = const(1002)
CLOSE_DATA_NOT_SUPPORTED = const(1003)
CLOSE_BAD_DATA = const(1007)
CLOSE_POLICY_VIOLATION = const(1008)
CLOSE_TOO_BIG = const(1009)
CLOSE_MISSING_EXTN = const(1010)
CLOSE_BAD_CONDITION = const(1011)

URL_RE = re.compile(r'(wss|ws)://([A-Za-z0-9-\.]+)(?:\:([0-9]+))?(/.+)?')
URI = namedtuple('URI', ('protocol', 'hostname', 'port', 'path'))

class NoDataException(Exception):
    pass

class ConnectionClosed(Exception):
    pass

def urlparse(uri):
    """Parse ws:// URLs"""
    match = URL_RE.match(uri)
    if match:
        protocol = match.group(1)
        host = match.group(2)
        port = match.group(3)
        path = match.group(4)

        if protocol == 'wss':
            if port is None:
                port = 443
        elif protocol == 'ws':
            if port is None:
                port = 80
        else:
            raise ValueError('Scheme {} is invalid'.format(protocol))

        return URI(protocol, host, int(port), path)

async def websocket_handshake(client_reader, client_writer):
    try:
        request = (await client_reader.read(1024)).decode('utf-8')
        print("Requisição recebida:", request)  # Depuração
        headers = {}
        for line in request.split("\r\n")[1:]:
            if ": " in line:
                key, value = line.split(": ", 1)
                headers[key.lower()] = value

        key = headers.get("sec-websocket-key")
        if not key:
            print("Erro: sec-websocket-key não encontrado")
            return False

        GUID = "258EAFA5-E914-47DA-95CA-C5AB0DC85B11"
        accept = ubinascii.b2a_base64(
            uhashlib.sha1((key + GUID).encode()).digest()
        ).decode().strip()

        response = (
            "HTTP/1.1 101 Switching Protocols\r\n"
            "Upgrade: websocket\r\n"
            "Connection: Upgrade\r\n"
```

```
            f"Sec-WebSocket-Accept: {accept}\r\n\r\n"
        )
        client_writer.write(response.encode())
        await client_writer.drain()
        print("Handshake enviado com sucesso")
        return True
    except Exception as e:
        print("Erro no handshake:", e)
        return False

class Websocket:
    """Basis of the Websocket protocol for ESP32 with uasyncio streams"""

    is_client = False

    def __init__(self, reader, writer):
        self.reader = reader
        self.writer = writer
        self.open = True

    def __enter__(self):
        return self

    def __exit__(self, exc_type, exc, tb):
        self.close()

    def settimeout(self, timeout):
        pass

    async def recv_nowait(self):
        """Recebe mensagem se houver, senão retorna None"""
        if not self.open:
            return None
        try:
            return await self.recv()
        except Exception:
            return None

    async def read_frame(self, max_size=1024):
        """Read a frame from the stream"""
        try:
            two_bytes = await self.reader.readexactly(2)
            if not two_bytes:
                raise NoDataException
        except Exception:
            raise NoDataException

        byte1, byte2 = struct.unpack('!BB', two_bytes)

        fin = bool(byte1 & 0x80)
        opcode = byte1 & 0x0f

        mask = bool(byte2 & (1 << 7))
        length = byte2 & 0x7f

        if length == 126:
            length, = struct.unpack('!H', await self.reader.readexactly(2))
        elif length == 127:
            length, = struct.unpack('!Q', await self.reader.readexactly(8))

        if max_size is not None and length > max_size:
            log_debug("Frame too big, closing")
```

```
            self.close(code=CLOSE_TOO_BIG)
            return True, OP_CLOSE, None

        if mask:
            mask_bits = await self.reader.readexactly(4)
        try:
            data = await self.reader.readexactly(length)
        except MemoryError:
            log_debug("Frame too big, closing")
            self.close(code=CLOSE_TOO_BIG)
            return True, OP_CLOSE, None
        except Exception:
            raise NoDataException

        if mask:
            data = bytes(b ^ mask_bits[i % 4] for i, b in enumerate(data))

        return fin, opcode, data

    async def write_frame(self, opcode, data=b''):
        """Write a frame to the stream"""
        fin = True
        mask = self.is_client

        length = len(data)
        byte1 = 0x80 if fin else 0
        byte1 |= opcode
        byte2 = 0x80 if mask else 0

        if length < 126:
            byte2 |= length
            self.writer.write(struct.pack('!BB', byte1, byte2))
        elif length < (1 << 16):
            byte2 |= 126
            self.writer.write(struct.pack('!BBH', byte1, byte2, length))
        elif length < (1 << 64):
            byte2 |= 127
            self.writer.write(struct.pack('!BBQ', byte1, byte2, length))
        else:
            raise ValueError()

        if mask:
            mask_bits = struct.pack('!I', random.getrandbits(32))
            self.writer.write(mask_bits)
            data = bytes(b ^ mask_bits[i % 4] for i, b in enumerate(data))

        self.writer.write(data)
        await self.writer.drain()

    async def recv(self):
        """Receive data from the websocket"""
        assert self.open


        while self.open:
            try:
                fin, opcode, data = await self.read_frame()
            except NoDataException:
                return None
            except ValueError:
                log_debug("Failed to read frame. Stream dead.")
```

```
            self._close()
            raise ConnectionClosed()

        if not fin:
            raise NotImplementedError()

        if opcode == OP_TEXT:
            return data.decode('utf-8')
        elif opcode == OP_BYTES:
            return data
        elif opcode == OP_CLOSE:
            self._close()
            return
        elif opcode == OP_PONG:
            continue
        elif opcode == OP_PING:
            log_debug("Sending PONG")
            await self.write_frame(OP_PONG, data)
            continue
        elif opcode == OP_CONT:
            raise NotImplementedError(opcode)
        else:
            raise ValueError(opcode)

async def send(self, buf):
    """Send data to the websocket"""
    assert self.open

    if isinstance(buf, str):
        opcode = OP_TEXT
        buf = buf.encode('utf-8')
    elif isinstance(buf, bytes):
        opcode = OP_BYTES
    else:
        raise TypeError()

    await self.write_frame(opcode, buf)

def close(self, code=CLOSE_OK, reason=''):
    """Close the websocket"""
    if not self.open:
        return

    buf = struct.pack('!H', code) + reason.encode('utf-8')
    asyncio.create_task(self.write_frame(OP_CLOSE, buf))
    self._close()

def _close(self):
    log_debug("Connection closed")
    self.open = False
    asyncio.create_task(self._async_close_writer())

async def _async_close_writer(self):
    try:
        self.writer.close()
        await self.writer.wait_closed()
    except Exception:
        pass
```

Código A.3 - script_v2.js

```
function criarGrafico() {
  const canvas = document.getElementById('meuGrafico');

  if (canvas instanceof HTMLCanvasElement) {
    let contexto = canvas.getContext('2d');
    let grafico = new Chart(contexto, {
      type: 'bar',
      data: {
        labels: [], // Datas (ex.: "08/10/2025")
        datasets: [
          {
            label: 'Contador de Pulsos',
            data: [], // Contagens por data
            borderColor: '#34e758ff',
            backgroundColor: '#34e75880',
            borderWidth: 2,
            fill: true,
            tension: 0.1,
          },
        ],
      },
      options: {
        plugins: {
          legend: {
            labels: {
              font: { size: 20 },
              color: '#697b6dff',
            },
          },
        },
        scales: {
          x: {
            ticks: { font: { size: 18 }, color: '#697b6dff' },
            title: {
              display: true,
              text: 'Data',
              font: { size: 18 },
              color: '#697b6dff',
            },
          },
          y: {
            ticks: { font: { size: 18 }, color: '#697b6dff' },
            beginAtZero: true,
            title: {
              display: true,
              text: 'Contador',
              font: { size: 18 },
              color: '#697b6dff',
            },
          },
        },
      },
    });
    conectarWebSocket(grafico);
  } else {
    console.error('Elemento canvas não encontrado ou inválido');
  }
}

function conectarWebSocket(grafico) {
  let socket = new WebSocket('ws://192.168.251.68:8080');
```

```
  socket.onopen = () => {
    console.log('Conexão WebSocket estabelecida com sucesso');
  };
  socket.onmessage = (event) => {
    try {
      let dados = JSON.parse(event.data);
      processamentoMensagem(grafico, dados);
    } catch (erro) {
      console.error('Falha ao processar a mensagem:', erro);
    }
  };
  socket.onclose = () => {
   console.log('Conexão WebSocket fechada, reconectando...');
    setTimeout(() => conectarWebSocket(grafico), 2000);
  };
  socket.onerror = (erro) => {
    console.error('Erro no WebSocket:', erro);
  };
}

function processamentoMensagem(grafico, dados) {
  console.log('Dados recebidos:', dados);

  if (dados.historico) {
    // Processa o histórico completo
    const historico = dados.historico;
    grafico.data.labels = []; // Limpa labels existentes
    grafico.data.datasets[0].data = []; // Limpa dados existentes
    const datas = Object.keys(historico).sort(); // Ordena datas
    for (const data of datas) {
      if (grafico.data.labels.length < 30) { // Limita a 30 dias
        grafico.data.labels.push(data);
        grafico.data.datasets[0].data.push(historico[data]);
      }
    }
  } else if (dados.data && dados.contador !== undefined) {
    // Processa atualização em tempo real
    const data = dados.data;
    const contador = dados.contador;
    const index = grafico.data.labels.indexOf(data);

    if (index === -1) {
      // Nova data
      if (grafico.data.labels.length < 30) { // Limita a 30 dias
        grafico.data.labels.push(data);
        grafico.data.datasets[0].data.push(contador);
      } else {
        // Remove a data mais antiga e adiciona a nova
        grafico.data.labels.shift();
        grafico.data.datasets[0].data.shift();
        grafico.data.labels.push(data);
        grafico.data.datasets[0].data.push(contador);
      }
    } else {
      // Atualiza contagem para data existente
      grafico.data.datasets[0].data[index] = contador;
    }
  }

  grafico.update();
}

document.addEventListener('DOMContentLoaded', criarGrafico);
```

Código A.4 - index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>Gráfico de Pulsos</title>
    <link rel="stylesheet" href="style.css" />
   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script type="module" src="script_v2.js" defer></script>
    <style>
      .grafico {
        width: 800px;
        height: 400px;
        max-width: 100%;
        margin: 0 auto;
      }
    </style>
  </head>
  <body>
    <div class="grafico">
      <canvas id="meuGrafico" class="grafico"></canvas>
    </div>
  </body>
</html>
```