MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CÂMPUS ITUMBIARA
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

GUSTAVO DOURADO SILVA

ARQUITETURA DE CONTROLE PARA LIMITAÇÃO DE EXPORTAÇÃO DE POTÊNCIA EM INVERSORES FOTOVOLTAICOS UTILIZANDO AUTOMAÇÃO ABERTA EM MICRORREDES

Itumbiara

2026

INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[X] TCC - Graduação	[ ] Trabalho Apresentado em Evento
[ ] Produto Técnico e Educacional - Tipo: _____	

Nome Completo do Autor: Gustavo Dourado Silva
Matrícula: 20211040100089
Título do Trabalho: Arquitetura de Controle para Limitação de Exportação de Potência em Inversores Fotovoltaicos Utilizando Automação Aberta em Microrredes

Autorização - Marque uma das opções

1. (X) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. ( ) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ___/___/_____ (Embargo);
3. ( ) Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

( ) O documento está sujeito a registro de patente.
( ) O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
( ) Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Itumbiara-GO, 28/04/2026.

Documento assinado digitalmente
gov.br
GUSTAVO DOURADO SILVA
Data: 28/04/2026 13:09:45-0300
Verifique em https://validar.iti.gov.br

_____
Assinatura do Autor e/ou Detentor dos Direitos Autorais

GUSTAVO DOURADO SILVA

ARQUITETURA DE CONTROLE PARA LIMITAÇÃO DE EXPORTAÇÃO DE POTÊNCIA EM INVERSORES FOTOVOLTAICOS UTILIZANDO AUTOMAÇÃO ABERTA EM MICRORREDES

Trabalho de conclusão de curso apresentado à banca examinadora, em cumprimento às exigências legais como requisito parcial à obtenção do grau de Bacharel em Engenharia de Controle e Automação no Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Câmpus Itumbiara.

Prof. Dr. Ghunter Paulo Viajante, (IFG) – Orientador

Itumbiara

2026

Dados Internacionais de Catalogação na Publicação (CIP)

S586a Silva, Gustavo Dourado

Arquitetura de controle para limitação de exportação de potência em inversores fotovoltaicos utilizando automação aberta em microrredes. / Gustavo Dourado Silva. – Itumbiara, 2026.

111 p.: il.

Trabalho de conclusão do curso (Bacharelado em Engenharia de Controle e Automação,) - Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Itumbiara, 2026.

Orientador: Prof. Dr. Ghunter Paulo Viajante.

1. Automação. 2. Sistemas fotovoltaicos. 3. Fator de potência. I. Título. II. Viajante, Ghunter Paulo. III. Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

CDD 629.895

Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Biblioteca Maria Gabriela Pacheco Pardey / Câmpus Itumbiara
Bibliotecário: Rosiane Gonçalves de Lima CRB1/1684

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CÂMPUS ITUMBIARA

Termo de Aprovação

Gustavo Dourado Silva

Arquitetura de Controle para Limitação de Exportação de Potência em Inversores Fotovoltaicos Utilizando Automação Aberta em Microrredes

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Campus Itumbiara, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Controle e Automação.

Trabalho de Conclusão de Curso defendido e aprovado, em 10 de fevereiro de 2026, pela banca examinadora constituída pelos seguintes membros:

BANCA EXAMINADORA

Prof. Dr. Ghunter Paulo Viajante - orientador

Instituto Federal de Goiás, Campus Itumbiara.

(Assinado Eletronicamente).

Prof. Dr. Eric Nery Chaves - membro interno

Instituto Federal de Goiás, Campus Itumbiara.

(Assinado Eletronicamente).

Prof. Dr. Marcelo Escobar de Oliveira - membro interno

Instituto Federal de Goiás, Campus Itumbiara.

(Assinado Eletronicamente).

Itumbiara – GO

Documento assinado eletronicamente por:

- **Marcelo Escobar de Oliveira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 16/04/2026 17:05:20.
- **Eric Nery Chaves**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 16/04/2026 16:13:33.
- **Ghunter Paulo Viajante**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 16/04/2026 16:11:38.

Este documento foi emitido pelo SUAP em 16/04/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifg.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 768305
Código de Autenticação: f4625364e9

Resumo

Este trabalho apresenta o desenvolvimento e a validação experimental de uma arquitetura de controle ativo de potência para microrredes fotovoltaicas, com o objetivo de implementar a operação *Grid Zero* (exportação nula) em inversores conectados à rede. A motivação inicial fundamenta-se na necessidade de adequação de sistemas fotovoltaicos em instalações que possuem limites máximos de exportação de energia, devido à saturação da capacidade da infraestrutura elétrica local, cenário característico do IFG Campus Itumbiara. Para solucionar este problema, foi desenvolvida uma malha de controle baseada em uma arquitetura de automação open source (código aberto) orquestradas em contêineres Docker: o *Node-RED* atuou como *gateway* de comunicação (convertendo Modbus TCP para MQTT), enquanto o *Home Assistant* executou a lógica de controle e a interface de operação. Os testes realizados na microrrede, composta por inversores Fronius e Victron, demonstraram que o sistema foi capaz de elevar o índice de autoconsumo para patamares superiores a 94%, anulando a exportação contínua de energia. Os resultados validam a arquitetura proposta como uma solução de baixo custo e replicável para o gerenciamento energético eficiente sob restrições de injeção.

Palavras-chave: Microrrede. Controle de Potência. Grid Zero. Automação Open Source.

Abstract

This work presents the development and experimental validation of an active power control architecture for photovoltaic microgrids, aiming to implement *Grid Zero* (zero export) operation in grid-tied inverters. The initial motivation is based on the need to adapt photovoltaic systems in facilities with maximum energy export limits due to the saturation of the local electrical infrastructure capacity, a characteristic scenario of the IFG Itumbiara Campus. To solve this problem, a control loop was developed based on an open-source automation architecture orchestrated in Docker containers: *Node-RED* acted as a communication *gateway* (converting Modbus TCP to MQTT), while *Home Assistant* executed the control logic and the operation interface. Tests performed on the microgrid, composed of Fronius and Victron inverters, demonstrated that the system was able to raise the self-consumption rate to levels above 94%, eliminating the continuous export of energy. The results validate the proposed architecture as a low-cost and replicable solution for efficient energy management under injection constraints.

Keywords: Microgrid. Power Control. Grid Zero. Open Source Automation.

Lista de ilustrações

Lista de tabelas

Lista de Códigos

Lista de abreviaturas e siglas

ANEEL Agência Nacional de Energia Elétrica

GD Geração Distribuída

IFG Instituto Federal de Goiás

TCC Trabalho de Conclusão de Curso

IHM Interface Homem-Máquina

IoT Internet das Coisas

BMS Sistema de Gerenciamento de Baterias

EMS Sistema de Gerenciamento de Energia

TSDB Banco de Dados de Séries Temporais

MQTT *Message Queuing Telemetry Transport*

SQL *Structured Query Language*

MetricsQL *Metrics Query Language*

TCP/IP *Transmission Control Protocol / Internet Protocol*

MPPT *Maximum Power Point Tracking*

ZHA *Zigbee Home Automation*

HACS *Home Assistant Community Store*

Sumário

1 Introdução

1.1 Contextualização

O panorama energético mundial tem passado por uma transformação significativa, impulsionada pela necessidade de diversificar a matriz energética e mitigar os impactos ambientais decorrentes do uso de fontes fósseis. Nesse contexto, a energia solar fotovoltaica destaca-se como uma das soluções mais promissoras, especialmente em países de alta incidência solar, como o Brasil. O avanço tecnológico e a redução dos custos dos equipamentos têm contribuído para a expansão da Geração Distribuída (GD), permitindo que consumidores residenciais, comerciais e institucionais se tornem também produtores de energia elétrica.

No Brasil, por exemplo, a Agência Nacional de Energia Elétrica (ANEEL) registrou um acréscimo de aproximadamente 8,84 GW de potência instalada em sistemas de micro e minigeração distribuída apenas em 2024 (ANEEL, 2025), o que demonstra o avanço expressivo desse modelo de produção energética descentralizada.

Tradicionalmente, a GD opera sob o regime de compensação de energia elétrica, conhecido como *net-metering*, no qual o excedente gerado é injetado na rede da concessionária, gerando créditos de energia para o consumidor. Contudo, alterações regulatórias recentes, somadas a limitações técnicas das redes de distribuição, têm incentivado uma nova abordagem: a maximização do autoconsumo, ou seja, o aproveitamento local e imediato da energia gerada, reduzindo ao mínimo a dependência da rede elétrica.

Esse novo paradigma impulsiona o desenvolvimento de sistemas de controle mais inteligentes, capazes de gerenciar dinamicamente o fluxo de potência entre geração e consumo. Surge, assim, o conceito de *Grid Zero* — ou sistema de energia autossuficiente — no qual a instalação é configurada para nunca exportar energia excedente para a rede, ajustando continuamente sua geração para igualar-se à demanda local (LOIOLA, 2024). O presente trabalho adota esse conceito como referência teórica e busca reproduzir seu comportamento por meio de uma lógica de limitação de potência, implementada externamente ao inversor. Dessa forma, o sistema proposto visa alcançar um efeito

operacional equivalente ao de um *Grid Zero*.

1.2 Problemática

A maioria dos inversores fotovoltaicos conectados à rede (*on-grid*), como o modelo Fronius utilizado neste trabalho, é projetada para operar em sua potência máxima disponível, realizando o *Maximum Power Point Tracking* (MPPT), independentemente do perfil de consumo instantâneo da unidade consumidora. Em sistemas tradicionais, esse comportamento não representa um problema, uma vez que o excedente é automaticamente injetado na rede e convertido em créditos energéticos.

Entretanto, em cenários onde há restrições contratuais ou técnicas à exportação de energia, essa característica torna-se um desafio. Sem um mecanismo de controle ativo, o excedente de geração causa a injeção de potência na rede, podendo acionar proteções, gerar instabilidade e até provocar o desligamento automático do inversor. Como consequência, há desperdício de energia renovável e redução da eficiência do sistema.

A problemática central deste trabalho consiste, portanto, em desenvolver um sistema de controle em malha fechada, de baixo custo e alta confiabilidade, capaz de modular ativamente a potência de saída de um inversor *on-grid* padrão. O objetivo é garantir que a geração fotovoltaica não ultrapasse o consumo das cargas locais, reduzindo a exportação de energia à rede e mantendo a operação contínua do sistema. O controle proposto busca, assim, produzir o comportamento característico de um sistema *Grid Zero*.

1.3 Justificativa

A relevância deste projeto é particularmente evidente em instalações com alta capacidade de geração fotovoltaica, como é o caso do Instituto Federal de Goiás (IFG) — Campus Itumbiara, local onde este estudo foi desenvolvido. Em instalações desse porte, é comum que a infraestrutura da concessionária local, Equatorial Goiás, imponha limitações contratuais ou técnicas quanto à quantidade de energia que pode ser exportada para a rede pública.

Quando a geração solar excede o consumo interno e atinge esse limite de exportação, o sistema fotovoltaico convencional é forçado a reduzir sua operação ou até mesmo desligar-se por completo, resultando na subutilização de um ativo de alto valor e em perdas de eficiência energética. Nesse contexto, a implementação de um sistema de controle dinâmico, como o proposto neste Trabalho de Conclusão de Curso (TCC), oferece uma solução direta. O controle desenvolvido permite que o inversor continue operando de forma contínua, ajustando automaticamente sua potência de saída conforme o consumo instantâneo das cargas locais, maximizando o autoconsumo e respeitando as restrições impostas pela concessionária.

A escolha por ferramentas de código aberto, como `Home Assistant`, `Node-RED` e `VictoriaMetrics`, todas orquestradas em contêineres `Docker`, justifica-se pela busca de uma solução de baixo custo, alta flexibilidade e que assegure a privacidade e soberania dos dados, uma vez que opera integralmente em ambiente local, sem depender de serviços de nuvem de terceiros.

Sob a perspectiva acadêmica, este trabalho também se destaca como um estudo de caso prático sobre a integração entre protocolos industriais consolidados, como o `Modbus`, e plataformas modernas de Internet das Coisas (IoT). Dessa forma, contribui para o avanço de soluções aplicadas à gestão energética inteligente e à otimização de microrredes fotovoltaicas no contexto da Engenharia de Controle e Automação.

1.4 Objetivos

1.4.1 Objetivo Geral

Desenvolver e implementar um sistema de controle de limitação de potência inspirado na filosofia *Grid Zero*, capaz de ajustar dinamicamente a geração de um inversor fotovoltaico *on-grid*, reduzindo ou eliminando a exportação de energia para a rede elétrica e mantendo o equilíbrio entre geração e consumo local.

1.4.2 Objetivos Específicos

- Analisar a arquitetura de comunicação dos inversores Victron e Fronius, identificando os registradores `Modbus` relevantes ao controle de potência;
- Projetar e implementar uma automação em `Node-RED` e `Home Assistant` para realizar a leitura, cálculo e envio de comandos de limitação de potência;
- Realizar a integração das plataformas em ambiente `Docker`, garantindo modularidade e isolamento entre os serviços;
- Monitorar o comportamento do sistema com e sem o controle ativo, avaliando o desempenho e a estabilidade da solução;
- Validar a eficácia do controle na redução da exportação de energia e na maximização do autoconsumo.

1.5 Estrutura do Trabalho

Este trabalho está organizado em nove capítulos.

O **Capítulo 1** apresenta o contexto do tema, a problemática, a justificativa e os objetivos do projeto. O **Capítulo 2** descreve a microrrede utilizada nos testes experimentais, detalhando sua configuração, componentes e condições de operação. O **Capítulo 3** apresenta a arquitetura geral do sistema proposto, abordando a interconexão entre os inversores, o sistema de controle e as plataformas de automação. O **Capítulo 4** descreve as ferramentas de software empregadas, incluindo `Node-RED`, `Home Assistant` e `Docker`, que compõem a infraestrutura de automação. O **Capítulo 5** aborda os protocolos de comunicação utilizados, como `Modbus` e MQTT, destacando suas funções na troca de dados entre os dispositivos. O **Capítulo 6** discute a estrutura de armazenamento e monitoramento de dados, incluindo as integrações com `Prometheus`, `VictoriaMetrics` e `Grafana`. O **Capítulo 7** apresenta o processo de implementação prática do sistema e a configuração das automações. O **Capítulo 8** reúne os resultados obtidos e as análises realizadas a partir dos testes experimentais. Por fim, o **Capítulo 9** consolida as conclusões do trabalho e propõe direções para pesquisas e melhorias futuras.

2 Microrrede do Sistema instalado no Lab Fontes

2.1 Descrição Geral

O sistema experimental utilizado neste trabalho é a microrrede instalada no Laboratório de Fontes Renováveis de Energia do IFG Campus Itumbiara. Essa plataforma, com capacidade aproximada de 3 kW, foi originalmente detalhada e empregada em estudos de supervisão e controle por Lopes (2025, Seção 2.5), servindo como base para o desenvolvimento do presente projeto.

Conforme descrito pelo autor, o arranjo é composto por um sistema fotovoltaico conectado a um inversor *on-grid* Fronius Primo 3.0-1, um inversor híbrido Victron Multiplus II 48/3000/35-32, um banco de baterias de tecnologia lítio-ferro-fosfato ($LiFePO_4$) e um conjunto de cargas formado pela iluminação dos laboratórios IFMaker e Fontes Renováveis.

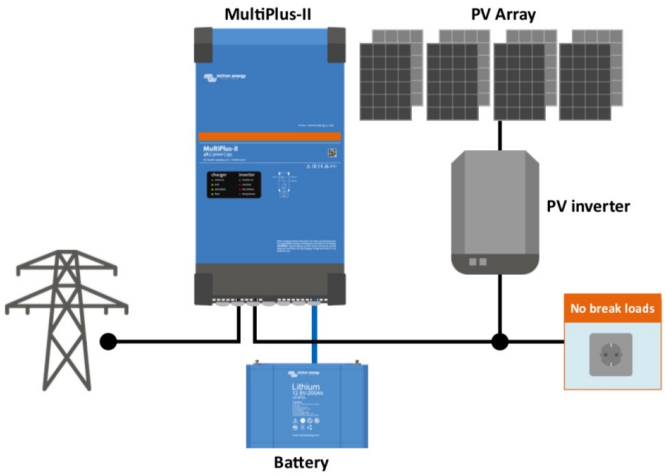
Essa configuração permite a operação tanto conectada à rede pública quanto em modo isolado, constituindo uma infraestrutura adequada para estudos de controle, eficiência energética e integração entre geração distribuída e sistemas de armazenamento. Na Figura 1 é apresentada uma foto do conjunto de equipamentos que compõem a microrrede.

Figura 1 – Microrrede do Laboratório de Fontes Renováveis do IFG Campus Itumbiara.

Fonte: Lopes (2025, Figura 6)

Para detalhar a interligação elétrica desses componentes, na Figura 2 é apresentado o diagrama esquemático da rede. A topologia adotada para este projeto é o acoplamento em corrente alternada (*AC-Coupling*). Nesta configuração, o inversor fotovoltaico (*PV Inverter*), representado neste projeto pelo inversor Fronius Primo, é conectado diretamente à saída de corrente alternada do inversor híbrido (Victron MultiPlus-II), compartilhando o barramento com as cargas críticas (*No break loads*). O inversor híbrido atua, assim, como o gerenciador central da microrrede, controlando o fluxo bidirecional de energia entre a rede elétrica da concessionária, o banco de baterias e a geração solar local.

Figura 2 – Diagrama elétrico do sistema ilustrando a topologia *AC-Coupling*.



Fonte: Victron Energy (2022).

2.2 Composição Física do Sistema

A microrrede é composta por elementos responsáveis pela geração, conversão, armazenamento e consumo de energia elétrica. Esses componentes formam a infraestrutura física necessária para a implementação e a avaliação do sistema de controle proposto neste trabalho.

Na Tabela 1 são apresentados os principais dispositivos que compõem a microrrede e suas respectivas potências nominais. Parte dessas informações refere-se à configuração original do sistema descrita por Lopes (2025); entretanto, os dados foram reorganizados e complementados para refletir a configuração efetivamente utilizada nos experimentos, incluindo a adição de cargas auxiliares controladas.

Tabela 1 – Componentes da microrrede do IFG Campus Itumbiara.

Componente	**Modelo**	**Potência**
Inversor fotovoltaico	Fronius Primo 3.0-1	3 kVA
Módulos fotovoltaicos	BYD 335PHK-36 (335 Wp) — 12 unidades	4,02 kWp
Inversor híbrido	Victron Multiplus II 48/3000/35-32	3 kVA
Banco de baterias	BYD B-Plus 2.5 (2,5 kWh/unidade) — 2 unidades	5 kWh
Cargas principais	Iluminação LED dos laboratórios	0,48 kW
Carga auxiliar	Refletor halógeno	0,20 kW

Fonte: Adaptado de Lopes (2025, Tabela 1) e complementado pelo autor.

Nas Figuras de 3 a 7 são ilustrados alguns dos principais elementos físicos da microrrede, abrangendo os dispositivos de geração, conversão, armazenamento e consumo de energia utilizados nos experimentos.

Figura 3 – Módulos fotovoltaicos instalados na microrrede.

Fonte: Autoria própria (2025).

Figura 4 – Inversor fotovoltaico Fronius Primo 3.0-1.

Fonte: Autoria própria (2025).

Figura 5 – Inversor híbrido Victron Multiplus II.

Fonte: Autoria própria (2025).

Figura 6 – Banco de baterias $LiFePO_4$ do sistema.

Fonte: Autoria própria (2025).

Figura 7 – Carga auxiliar utilizada nos experimentos da microrrede.

Fonte: Autoria própria (2025).

2.3 Cargas e Condições de Operação

As cargas utilizadas nos experimentos realizados neste trabalho são classificadas em dois grupos distintos: cargas principais e carga auxiliar controlada. Essa distinção permite avaliar o comportamento do sistema de controle tanto em condições de operação típicas quanto sob variações abruptas e controladas de consumo.

As cargas principais correspondem à iluminação dos laboratórios IFMaker e Fontes Renováveis, ambos integrantes do laboratório iTec-Lab, representando o perfil de consumo típico da microrrede durante sua operação normal. O consumo combinado dessas cargas constitui a base de referência para a análise dos resultados apresentados no Capítulo 8.

O acionamento e o desligamento da iluminação dos laboratórios provocam variações abruptas de potência, caracterizando degraus de carga naturais da microrrede. Esses eventos foram utilizados como estímulo para a avaliação do comportamento dinâmico do sistema de controle desenvolvido no projeto.

Além das cargas principais, foi incorporada à microrrede uma carga auxiliar controlada, conectada ao borne de carga não essencial do inversor Victron. Essa carga consiste em um refletor halógeno com potência nominal de 200 W, correspondente a aproximadamente 42% da potência total das cargas principais (480 W). utilizado especificamente para provocar variações abruptas e controladas de consumo, permitindo a avaliação da velocidade de resposta e da estabilidade do sistema de controle de potência implementado.

O acionamento da carga auxiliar é realizado por meio de um adaptador inteligente Zigbee (TNCE Tuya Brazil Plug 20A Zigbee), integrado ao sistema de automação por meio do *Home Assistant*. Esse arranjo possibilita o controle remoto e programado da carga, bem como sua utilização como elemento de teste repetível e reprodutível, ampliando a flexibilidade experimental da microrrede.

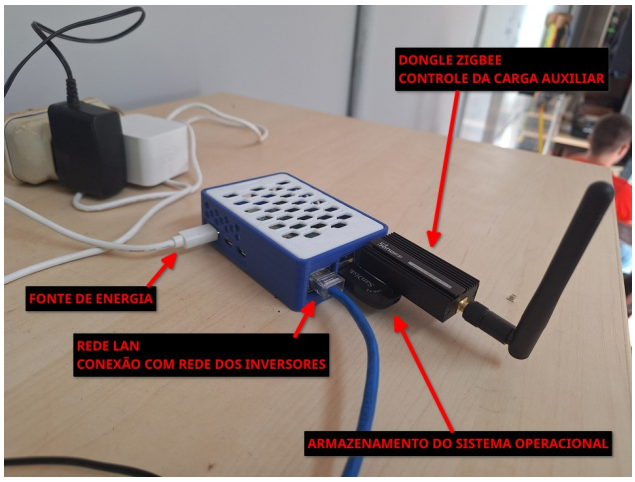
A microrrede operou predominantemente em modo conectado à rede (*grid-tied*), permitindo a análise da interação entre o inversor Fronius, responsável pela conversão da energia fotovoltaica, e o inversor Victron, que atua como ponto de acoplamento entre as cargas, o banco de baterias e a rede elétrica. O inversor Victron realiza o monitoramento

bidirecional do fluxo de energia, possibilitando a importação de potência da rede elétrica durante períodos de baixa ou inexistente geração solar e contribuindo para o equilíbrio energético do sistema.

2.4 Infraestrutura de Comunicação e Processamento

O sistema de supervisão e controle da microrrede é centralizado em um Raspberry Pi, que atua como unidade computacional principal do sistema. Nesse dispositivo encontram-se em execução os principais *softwares* utilizados neste projeto, a saber: *Home Assistant*, *Node-RED* e *VictoriaMetrics*. O equipamento e seus elementos associados podem ser observados na Figura 8.

Figura 8 – Raspberry Pi utilizado no sistema, com *dongle* Zigbee e interfaces de rede.



Fonte: Autoria própria (2026).

Além do adaptador Zigbee SONOFF Zigbee *Dongle*-E 3.0, utilizado para a comunicação com os dispositivos Zigbee responsáveis pelo controle da carga auxiliar, o Raspberry Pi possui conexão Ethernet cabeada e um dispositivo de armazenamento USB, no qual estão instalados o sistema operacional e os serviços de automação.

A microrrede opera sobre uma rede local dedicada do laboratório (LabFontes), sem acesso direto à internet. Os inversores Fronius e Victron, assim como o Raspberry

Pi, estão conectados a essa rede por meio de conexão Ethernet, garantindo baixa latência e maior confiabilidade na comunicação.

O acesso externo ao sistema é realizado por meio de uma segunda interface de rede do Raspberry Pi, conectada via Wi-Fi à rede institucional iTec-lab. Essa arquitetura permite a separação entre a rede operacional da microrrede e a rede com acesso à internet, preservando a autonomia e a segurança do sistema.

2.5 Objetivo Experimental da Microrrede

O objetivo experimental da microrrede do IFG Campus Itumbiara é disponibilizar uma infraestrutura real e instrumentada para a realização de ensaios aplicados em engenharia elétrica, controle e automação, permitindo a observação, a medição e a análise do comportamento energético do sistema em condições reais de operação.

No contexto deste trabalho, a microrrede foi utilizada especificamente para a implementação e a validação de uma estratégia de controle de potência inspirada na filosofia *Grid Zero*, baseada na limitação dinâmica da potência de saída de um inversor fotovoltaico *on-grid*. A arquitetura híbrida do sistema, composta por geração solar fotovoltaica, armazenamento em baterias e cargas reais, possibilitou a avaliação do desempenho do controle frente a variações abruptas de carga e às flutuações naturais da geração solar, típicas de sistemas fotovoltaicos conectados à rede.

Dessa forma, a microrrede serviu como um ambiente experimental representativo para a análise do tempo de resposta, da estabilidade e da eficácia do controle proposto na redução da exportação de energia para a rede elétrica e na maximização do autoconsumo, conforme apresentado e discutido nos capítulos subsequentes.

3 Arquitetura Geral do Sistema

O sistema foi projetado para operar de forma modular, na qual cada componente desempenha uma função específica e se comunica por meio de protocolos padronizados, garantindo flexibilidade, escalabilidade e facilidade de manutenção. A arquitetura integra elementos de hardware, como inversores e servidor de automação, e componentes de software, como *Node-RED*, *Home Assistant* e um *broker* MQTT, formando um fluxo de dados coeso desde a etapa de medição até a execução do controle.

O fluxo de informações do sistema pode ser descrito conforme as etapas a seguir.

1. **Medição e Leitura:** O *Node-RED* realiza a leitura dos dados de consumo das cargas por meio do inversor híbrido Victron, bem como da potência de geração do inversor Fronius, utilizando o protocolo Modbus TCP. Essas medições constituem a base para a tomada de decisão do sistema de controle.

2. **Processamento e Publicação:** Os dados brutos obtidos via Modbus são processados no *Node-RED*, passando por etapas de formatação, escalonamento e validação. Em seguida, as informações tratadas são publicadas em um *broker* MQTT local, por meio de tópicos específicos, possibilitando sua utilização por outros serviços do sistema, em especial o *Home Assistant*.

3. **Supervisão e Interação:** A plataforma *Home Assistant*, atuando como IHM, inscreve-se nos tópicos MQTT para receber, em tempo quase real, os dados provenientes dos inversores do sistema. Essas informações são apresentadas ao usuário por meio de painéis gráficos e indicadores de estado. Além disso, o *Home Assistant* realiza a leitura das grandezas elétricas associadas à carga auxiliar controlada via Zigbee, permitindo sua supervisão integrada ao restante do sistema.

4. **Definição dos Comandos de Controle:** A definição do limite de potência aplicado ao inversor Fronius pode ser realizada por meio da interface do *Home Assistant*, de duas formas distintas:

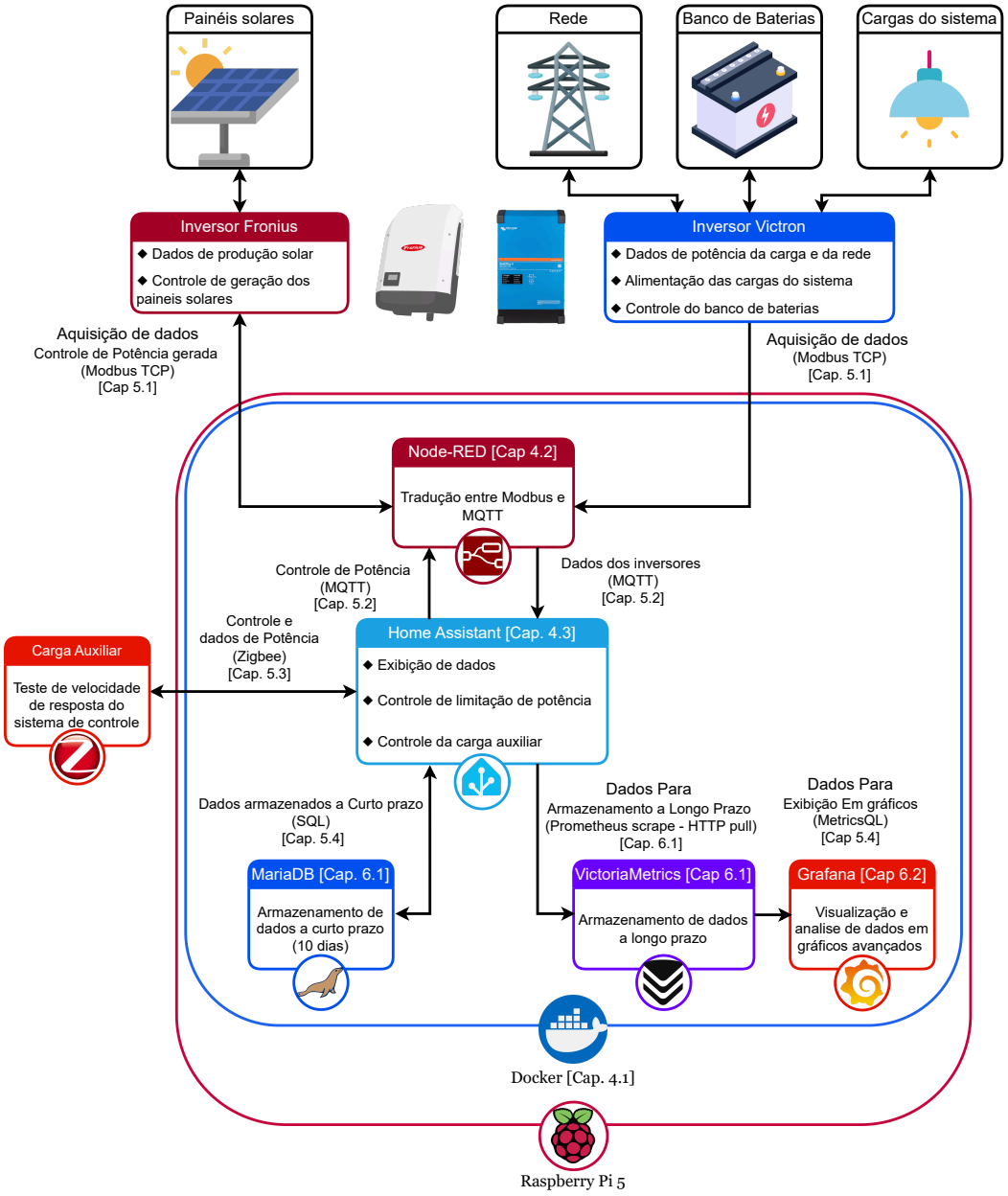
- **Controle manual:** o usuário define um valor fixo de potência, que atua como limite máximo para a geração do inversor fotovoltaico.
- **Controle automático:** o sistema ajusta dinamicamente o limite de potência do inversor de forma que a geração fotovoltaica acompanhe a potência instantaneamente consumida pelas cargas, com o objetivo de maximizar o autoconsumo e minimizar a exportação de energia para a rede.

Em ambos os modos de operação, o valor do limite de potência definido é publicado pelo *Home Assistant* em um tópico MQTT dedicado ao controle.

5. **Execução do Controle:** O *Node-RED*, inscrito no tópico MQTT de controle, recebe o valor atualizado do limite de potência, converte essa informação em um comando Modbus TCP compatível com o inversor Fronius e o envia ao equipamento. A partir desse comando, o inversor ajusta sua potência de saída conforme o valor estabelecido, fechando a malha de controle do sistema.

Para detalhar a interligação desses componentes, na Figura 9 é apresentado o diagrama de blocos da rede. Nesta ilustração é demonstrado a arquitetura implementada, destacando os principais elementos do sistema e os fluxos de comunicação estabelecidos entre eles.

Figura 9 – Diagrama de blocos da arquitetura do sistema.



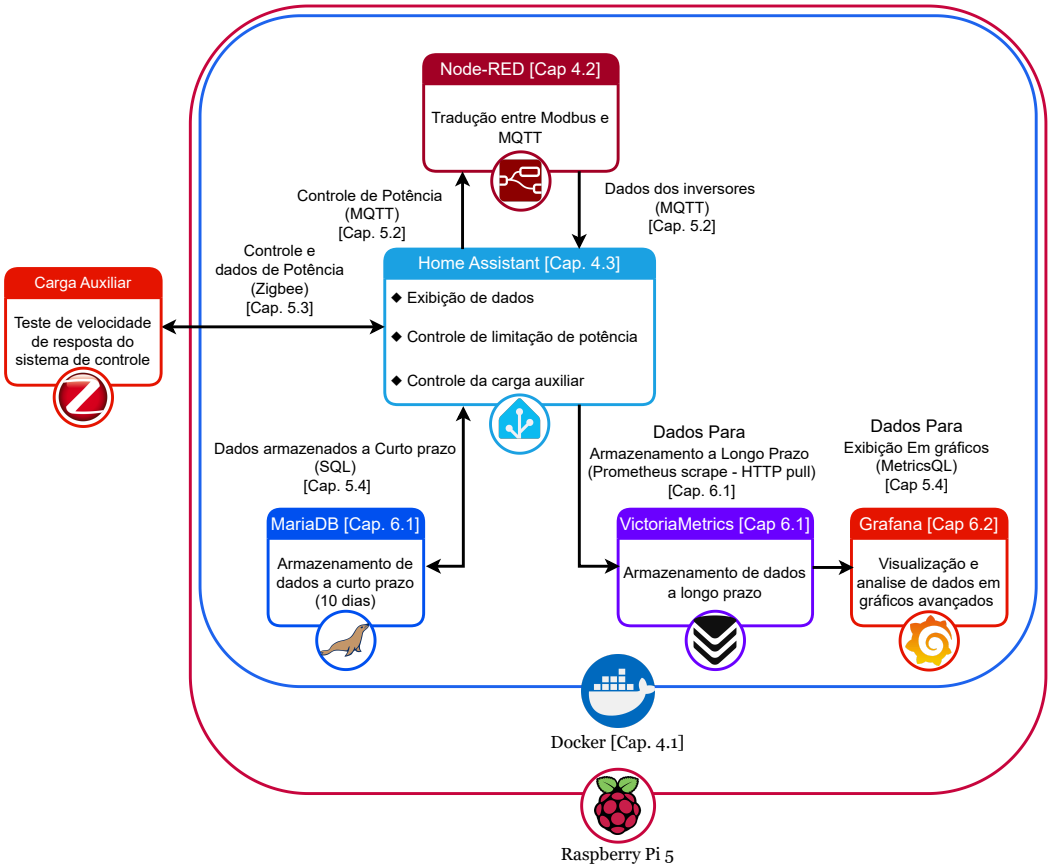
Fonte: Autoria própria (2026).

4 Plataforma de Orquestração e Ferramentas

Para a implementação da arquitetura proposta, foi selecionado um conjunto de ferramentas de *software* de código aberto, executadas de forma coesa em uma plataforma de orquestração de contêineres. Este capítulo descreve a infraestrutura de *software* base do projeto.

Para ilustrar a interligação desses componentes, a Figura 10 apresenta a arquitetura de *software* embarcada no *Raspberry Pi*. O esquema detalha o fluxo de informações, os protocolos de comunicação empregados e os serviços gerenciados via Docker utilizados, indicando em quais capítulos deste documento onde o funcionamento de cada ferramenta é aprofundado.

Figura 10 – Diagrama de blocos de protocolos e ferramentas.



Fonte: Autoria própria (2026).

4.1 Docker: Contêineres como Infraestrutura

Toda a infraestrutura de *software* deste projeto é gerenciada por meio da plataforma Docker. O Docker atua como uma plataforma aberta que fornece a capacidade de empacotar e executar aplicações em ambientes levemente isolados, denominados contêineres. Essa abordagem permite separar a aplicação da infraestrutura subjacente, garantindo que o sistema seja executado de forma rápida e consistente em qualquer ambiente computacional, eliminando conflitos de dependências entre as fases de desenvolvimento e operação (Docker Inc., 2026).

A adoção de contêineres Docker constituiu uma decisão de projeto fundamental, uma vez que essa abordagem apresenta vantagens relevantes para sistemas distribuídos e modulares, como o desenvolvido neste trabalho. Entre os principais benefícios, destacam-se:

- **Isolamento:** cada serviço do sistema opera em um ambiente independente, com suas próprias bibliotecas e dependências, evitando conflitos entre aplicações distintas;
- **Portabilidade:** a infraestrutura completa pode ser replicada em outros sistemas que disponham do Docker, mantendo o mesmo comportamento operacional e reduzindo problemas relacionados à configuração do ambiente;
- **Facilidade de gerenciamento:** a utilização de contêineres simplifica os processos de instalação, atualização, reinicialização e monitoramento dos serviços, favorecendo a manutenção e a escalabilidade do sistema.

No contexto deste trabalho, os principais componentes de *software* da arquitetura foram implementados como contêineres Docker, incluindo o *Node-RED*, o *Home Assistant*, o *broker* Mosquitto MQTT, o banco de dados relacional MariaDB, o banco de dados de séries temporais VictoriaMetrics e a plataforma de visualização Grafana. Essa organização contribuiu para a modularidade da solução e para a integração eficiente entre os diferentes serviços que compõem o sistema de supervisão e controle da microrrede. A função específica de cada um desses componentes no contexto de armazenamento de curto e longo prazo é detalhada no Capítulo 6.

4.2 Node-RED

O *Node-RED* é uma ferramenta de programação baseada em fluxos visuais (*low-code*) que atua como o núcleo lógico da arquitetura. No escopo deste projeto, o servidor *Node-RED* é executado em um contêiner Docker e atua como o principal tradutor de protocolos.

A decisão de utilizar o *Node-RED* para centralizar a comunicação Modbus, em detrimento das integrações nativas do *Home Assistant*, fundamenta-se em critérios

técnicos de flexibilidade e manutenibilidade. Durante as etapas iniciais, constatou-se que realizar a leitura direta pelo *Home Assistant* apresentava alta complexidade de configuração devido às limitações existentes nessa integração. Isso ocorre porque o inversor Fronius utiliza o padrão SunSpec Modbus, no qual diversas grandezas não são representadas diretamente como valores em ponto flutuante. Em muitos casos, os valores são codificados como inteiros escalonados, exigindo a leitura de registradores adicionais para aplicação de fatores de escala. Além disso, o mapeamento contempla registradores enumerados e grandezas de 32 e 64 bits, que requerem a combinação de múltiplos registradores de 16 bits.

A implementação desse tipo de tratamento diretamente no *Home Assistant* demandaria a criação de extensos modelos em *YAML*, com lógica aritmética e manipulação de tipos, resultando em uma solução pouco flexível e de difícil manutenção.

Em contrapartida, o *Node-RED* permite o uso de funções em *JavaScript*. Essa capacidade possibilita processar os vetores de dados brutos, aplicar os fatores de escala e estruturar os objetos JSON de forma programática antes do envio para a camada de supervisão.

Dessa forma, suas responsabilidades consolidadas neste projeto incluem:

- **Leitura e Processamento Modbus:** Atuar como cliente Modbus TCP, consultando os registradores e aplicando via código os fatores de escala para conversão em unidades de engenharia.
- **Tradução para MQTT:** Formatar e publicar os dados processados no *broker* MQTT, permitindo a leitura das variáveis pelo *Home Assistant*.
- **Tradução de Comandos (MQTT para Modbus):** Inscrever-se em tópicos de controle e traduzir as solicitações da IHM em escritas Modbus válidas para o inversor.

A implementação detalhada dos fluxos e dos algoritmos em *JavaScript* é apresentada no Capítulo 7.

4.3 Home Assistant: IHM e Automação

O *Home Assistant* é uma plataforma de automação residencial de código aberto, desenvolvida para atuar como um sistema central de controle e supervisão de dispositivos inteligentes. A plataforma funciona como um *hub* unificado capaz de integrar dispositivos de diferentes fabricantes e protocolos, operando predominantemente de forma local, sem dependência obrigatória de serviços em nuvem, o que contribui para maior privacidade e confiabilidade operacional (DAVIDSON, 2023).

A escolha do *Home Assistant* para este projeto fundamenta-se em quatro aspectos principais: controle totalmente local, preservação da privacidade dos dados, caráter de código aberto e ampla compatibilidade com diferentes protocolos e fabricantes. Diferentemente de soluções proprietárias de automação, a plataforma permite que todo o processamento, armazenamento e tomada de decisão ocorram no ambiente local, mantendo os dados restritos à infraestrutura da microrrede.

Além dessas características técnicas, o *Home Assistant* é amplamente reconhecido como uma das principais plataformas de automação disponíveis atualmente. Avaliações especializadas destacam sua elevada compatibilidade com dispositivos de terceiros, o suporte a um grande número de integrações e a existência de uma comunidade ativa de desenvolvedores e usuários, fatores que contribuem para sua maturidade e constante evolução (DIAZ, 2025).

No contexto deste trabalho, o *Home Assistant* foi empregado como a principal Interface Homem-Máquina (IHM) do sistema, sendo executado em um contêiner dedicado no ambiente Docker. A plataforma foi responsável pela supervisão das variáveis elétricas da microrrede, pela interação com o usuário e pela definição dos parâmetros de controle, como os limites de potência aplicados ao inversor fotovoltaico.

A integração com os demais componentes do sistema ocorre por meio do suporte nativo ao protocolo MQTT, que permite ao *Home Assistant* conectar-se ao *broker* local e instanciar automaticamente entidades do tipo sensores, atuadores e variáveis de controle a partir dos tópicos publicados na rede. Essa abordagem viabiliza uma integração flexível e escalável, facilitando a inclusão de novos dispositivos e medições sem a necessidade de reconfigurações extensas na interface.

Além da função de supervisão visual, o *Home Assistant* também desempenha o papel de gerenciador de dados, realizando a interface com o *backend* de armazenamento para o registro dos estados e históricos das variáveis monitoradas. Esses dados são posteriormente utilizados para análise e visualização, conforme detalhado no Capítulo 6.

4.3.1 Integrações utilizadas no Home Assistant

Para enriquecer o sistema de monitoramento e controle, foram empregadas integrações específicas do *Home Assistant* para coletar dados tanto dos inversores quanto de fontes externas, bem como para o envio de métricas ao sistema de armazenamento e visualização.

- ***Home Assistant Community Store* (HACS):** O HACS foi utilizado para a instalação e o gerenciamento de integrações e elementos de interface não incluídos no repositório oficial do *Home Assistant*. Trata-se de uma integração mantida pela comunidade que fornece uma interface gráfica para a descoberta, instalação, atualização e remoção de componentes personalizados, como integrações adicionais e *lovelaces*.[1].

  No contexto deste projeto, o HACS foi empregado para facilitar a incorporação de extensões hospedadas em repositórios públicos do GitHub, contribuindo para a padronização do processo de instalação e para a manutenção do ambiente.

- **Forecast.Solar:** Esta integração fornece previsões de produção de energia fotovoltaica. Com base nos dados de localização do sistema e nas especificações do arranjo de painéis, a plataforma estima a potência que será gerada ao longo do dia[2].

- **Electricity Maps:** Esta integração disponibiliza dados em tempo real sobre a intensidade de carbono da matriz elétrica local (em gCO_2eq/kWh) e a composição energética da rede, indicando a fração proveniente de fontes fósseis e renováveis, com base na localização geográfica do sistema[3]. A correlação desses dados com

[1] Disponível em: <https://hacs.xyz>
[2] Disponível em: <https://www.home-assistant.io/integrations/forecast_solar/>
[3] Disponível em: <https://www.home-assistant.io/integrations/co2signal/>

a geração fotovoltaica permite quantificar de forma objetiva a redução potencial de emissões de carbono e o índice de energia limpa associado ao consumo da instalação.

- **Victron (hass-victron):** A integração oficial da Victron foi utilizada para obter dados complementares do inversor híbrido, como o estado de operação do sistema e informações de *firmware*, que não são críticos para o funcionamento do laço de controle. Entretanto, seu intervalo de atualização padrão de 60 s foi considerado insuficiente para o controle de potência em tempo real. Em razão disso, foi implementado um ciclo de leitura Modbus personalizado utilizando o *Node-RED*[4].

- **SunSpec (ha-sunspec):** Esta integração, que implementa o protocolo Modbus SunSpec, foi utilizada na fase inicial do projeto para validar os dados lidos do inversor Fronius. Contudo, ela foi posteriormente substituída pela implementação Modbus direta no *Node-RED* por duas razões principais: a integração opera apenas em modo de leitura (*read-only*), o que inviabiliza a execução do controle de limitação de potência; e a implementação própria permite a utilização de intervalos de atualização mais reduzidos (15 s), adequados à dinâmica do sistema[5].

- **Zigbee Home Automation (ZHA):** A integração *Zigbee Home Automation* (ZHA) foi utilizada para a comunicação sem fio com dispositivos baseados no protocolo Zigbee, permitindo sua conexão direta ao *Home Assistant* por meio de um coordenador Zigbee. Trata-se de uma implementação de *gateway* Zigbee independente de fabricante, baseada na biblioteca de código aberto *zigpy*, capaz de substituir controladores proprietários, conforme descrito na documentação oficial da plataforma (HOME ASSISTANT, 2026)[6].

  No contexto deste projeto, a integração ZHA foi empregada para o controle da carga não essencial da microrrede, a qual utiliza um *plug* Zigbee. O controle dessa carga é realizado por meio de um *dongle* Zigbee conectado ao Raspberry Pi, possibilitando o acionamento remoto e programado da carga auxiliar diretamente pelo *Home Assistant*.

[4] Disponível em: <https://github.com/sfstar/hass-victron>
[5] Disponível em: <https://github.com/CJNE/ha-sunspec>
[6] Disponível em: <https://www.home-assistant.io/integrations/zha/>

- **Prometheus Exporter:** Esta integração foi utilizada para expor, por meio de um *endpoint* HTTP, as variáveis monitoradas pelo *Home Assistant* — como tensões, correntes, potências e estados de controle — no formato de métricas de séries temporais compatível com o ecossistema Prometheus. Essas métricas são coletadas periodicamente por um sistema externo de armazenamento de séries temporais, neste caso o *VictoriaMetrics*, e posteriormente utilizadas como fonte de dados pela plataforma *Grafana* para a construção de painéis analíticos e validação experimental do sistema[7].

4.3.2 Componentes de Interface (Lovelace Cards)

Além dos painéis nativos do *Home Assistant*, a IHM foi construída com o auxílio de componentes de *frontend* customizados (*Lovelace Cards*), que permitem visualizações de dados mais ricas e específicas para o monitoramento de energia:

- **Power Flow Card Plus:** Utilizado para prover uma forma fácil e clara de visualizar a distribuição de potência instantânea entre as fontes do sistema, como geração solar, rede e consumo das cargas[8].
- **Energy Flow Card Plus:** Inspirado no cartão oficial de distribuição de energia do *Home Assistant*, este componente foi usado por se integrar ao design do painel de energia, ao mesmo tempo que oferece mais funcionalidades, como a exibição de dispositivos individuais[9].
- **ApexCharts Card:** Baseado na biblioteca `ApexCharts.js`, este cartão é um componente utilizado para a criação de gráficos customizáveis. Inicialmente, foi empregado para a geração de gráficos de múltiplas linhas com o objetivo de comparar a produção solar e o consumo das cargas. No entanto, devido ao elevado processamento exigido, a execução desses gráficos diretamente no navegador de internet causava lentidão na interface do *Home Assistant*. Em razão desse impacto no desempenho, optou-se pela utilização da plataforma *Grafana* para a visualização

[7] Disponível em: <https://www.home-assistant.io/integrations/prometheus/>
[8] Disponível em: <https://github.com/flixlix/power-flow-card-plus>
[9] Disponível em: <https://github.com/flixlix/energy-flow-card-plus>

dessas métricas históricas de longa duração, mantendo o uso deste cartão apenas para representações simplificadas de curto prazo[10].

- **Sunsynk Power Flow Card:** Embora nomeado em referência a outra marca, este cartão é compatível com diversos inversores. Foi utilizado para criar um diagrama de fluxo animado, similar ao encontrado na tela física de inversores, que exibe em tempo real a potência fluindo entre cada componente do sistema (geração, rede e cargas)[11].

[10] Disponível em: <https://github.com/RomRider/apexcharts-card>
[11] Disponível em: <https://github.com/slipx06/sunsynk-power-flow-card>

5 Protocolos de Comunicação e Consulta

A interoperabilidade entre os diversos componentes de *hardware* e *software* do sistema é assegurada por um conjunto de protocolos de comunicação e de consulta padronizados. Esses protocolos permitem a integração entre inversores, sistemas de automação, interfaces de supervisão e plataformas de armazenamento de dados, viabilizando tanto o controle dinâmico da operação, com tempos de resposta na escala de segundos, quanto a análise histórica do desempenho da microrrede.

Para sintetizar o papel de cada tecnologia na arquitetura estrutural do projeto, na Figura 11 é apresentado um resumo visual dos protocolos e linguagens de consulta utilizados no projeto.

Figura 11 – Resumo visual dos protocolos de comunicação e linguagens de consulta utilizados no projeto.

Fonte: Autoria própria (2026).

5.1 Protocolo Modbus

A comunicação direta com os inversores constitui a base do sistema de controle, sendo o protocolo Modbus TCP o meio adotado para essa finalidade, em razão de sua ampla utilização em ambientes industriais e da compatibilidade nativa com os equipamentos empregados no projeto.

Nesta arquitetura, o contêiner do *Node-RED* atua como cliente Modbus, realizando requisições de leitura e escrita aos inversores, que operam como servidores Modbus e disponibilizam seus registradores por meio da rede local. Diferentemente do padrão Modbus RTU, o Modbus TCP não requer a configuração explícita de mestre e escravo, uma vez que a comunicação ocorre sobre o protocolo *Transmission Control*

Protocol / Internet Protocol (TCP/IP), sendo suficiente o endereçamento por endereço IP e porta de comunicação, usualmente a porta 502.

Para a correta implementação do controle, foi realizado o mapeamento dos registradores de interesse de cada inversor, identificando-se os endereços utilizados para a leitura das grandezas elétricas e para a escrita do limite de potência aplicado ao inversor fotovoltaico.

Os registradores Modbus utilizados para a leitura e o controle do inversor Fronius são apresentados na Tabela 5, disponibilizada no Apêndice A deste trabalho, conforme o mapeamento oficial fornecido pela fabricante (FRONIUS, s.d.). De forma análoga, os registradores empregados para a aquisição de dados do inversor Victron estão detalhados na Tabela 6, presente no Apêndice B, também estruturados com base na documentação oficial do equipamento (MVADER, 2025).

5.2 Protocolo MQTT

O protocolo MQTT foi empregado como a camada de comunicação entre a lógica de controle implementada no *Node-RED* e a interface de supervisão baseada no *Home Assistant*. A adoção desse protocolo se justifica por sua leveza, eficiência e pelo modelo de comunicação publicar/inscrever (*publish/subscribe*), que favorece o desacoplamento entre os diferentes componentes do sistema.

No projeto, um *broker* MQTT Mosquitto, executado em um contêiner *Docker* no servidor local, é responsável pelo gerenciamento da troca de mensagens. As medições processadas pelo *Node-RED* são publicadas em tópicos específicos, aos quais o *Home Assistant* se inscreve para fins de supervisão. De forma análoga, os comandos de controle definidos na interface do usuário são publicados em tópicos dedicados e consumidos pelo *Node-RED*, que os converte em comandos Modbus destinados ao inversor Fronius.

Um fator determinante para a escolha dessa arquitetura foi a utilização do recurso *MQTT Discovery* do *Home Assistant*. Por meio dessa funcionalidade, o *Node-RED* envia mensagens de configuração contendo metadados, como unidades de medida, ícones e nomes descritivos, permitindo que o *Home Assistant* crie automaticamente sensores e atuadores na interface de supervisão.

Essa abordagem reduz significativamente o esforço de configuração quando comparada a métodos alternativos, como o uso da biblioteca *node-red-contrib-home-assistant-websocket*. Nesse caso, a criação de cada entidade exigiria a configuração manual de nós específicos dentro dos fluxos, o que dificultaria a manutenção do sistema diante da elevada quantidade de registradores monitorados nos dois inversores.

Dessa forma, a utilização do MQTT associada ao mecanismo de *Discovery* assegura a troca de informações em tempo quase real, preserva a modularidade da arquitetura e facilita a expansão futura do sistema, uma vez que novos sensores adicionados aos fluxos de leitura passam a ser automaticamente reconhecidos pela interface de supervisão.

5.3 Protocolo Zigbee

O protocolo Zigbee foi adotado no projeto como a infraestrutura de comunicação sem fio para o acionamento de cargas e integração de dispositivos periféricos. A escolha por este padrão, em detrimento de dispositivos baseados em tecnologia Wi-Fi, fundamentou-se na premissa de garantir um controle 100% local, independente de serviços de nuvem proprietários e com latência determinística.

Dispositivos de automação Wi-Fi de baixo custo, amplamente disponíveis no mercado (como relés inteligentes baseados na plataforma Tuya), frequentemente dependem de conexão constante com servidores externos para autenticação e comando. Embora existam soluções de *software* desenvolvidas pela comunidade, como as integrações *Local Tuya* ou *Tuya Local*, que permitem o controle desses dispositivos dentro do *Home Assistant* sem a utilização de uma nuvem externa, essas abordagens muitas vezes exigem contornos técnicos complexos, dependem da obtenção de chaves de API proprietárias e podem perder funcionalidade com atualizações de *firmware* dos fabricantes.

Em contrapartida, o protocolo Zigbee oferece uma operação nativamente local. Ao utilizar um coordenador (o *dongle* conectado ao Raspberry Pi) e o *Home Assistant*, todo o tráfego de dados permanece restrito à rede interna do laboratório. Essa característica não apenas elimina a latência introduzida pelo tráfego de internet, mas também mitiga riscos relacionados à privacidade e soberania dos dados. Conforme apontam Brands e Kitchen (2021), dispositivos de IoT vinculados a servidores estrangeiros podem estar sujeitos a legislações que obrigam o compartilhamento de dados com governos

locais, representando um risco potencial à confidencialidade das informações coletadas.

No contexto deste trabalho, o Zigbee foi empregado para:

1. **Carga Auxiliar Controlada:** O acionamento remoto de um refletor halógeno de 200 W, conectado a um *smart plug* Zigbee. Esse dispositivo foi adicionado utilizando a integração do *Home Assistant* ZHA, permitindo a introdução de variações de carga para testes de resposta dinâmica do controle.
2. **Dispositivos Periféricos:** O aproveitamento da malha Zigbee para o acionamento de um relé inteligente conectado a uma televisão do laboratório. Embora este equipamento seja alimentado pela rede elétrica convencional do prédio, não fazendo parte do circuito elétrico da microrrede nem do algoritmo de controle de potência, sua integração lógica demonstra a versatilidade da arquitetura em centralizar a automação do ambiente, gerenciando múltiplos dispositivos sem depender de serviços em nuvem de terceiros.

Dessa forma, a utilização do Zigbee assegura que a infraestrutura de automação opere de forma autônoma, resiliente a falhas de conexão com a internet e alinhada aos requisitos de segurança de dados do projeto.

5.4 Protocolos de Consulta a Banco de Dados

Embora o armazenamento dos dados seja realizado por diferentes mecanismos, o acesso e a análise dessas informações ocorrem por meio de linguagens de consulta específicas, adequadas ao modelo de cada banco de dados empregado no sistema. Neste projeto, foram utilizadas duas linguagens distintas:

- ***Structured Query Language* (SQL):** Linguagem padrão para consulta a bancos de dados relacionais. No projeto, é utilizada internamente pelo *Home Assistant* para operações de registro, leitura e agregação de dados de curto prazo armazenados no banco de dados *MariaDB*, por meio do componente `recorder`.
- ***Metrics Query Language* (MetricsQL):** Linguagem de consulta utilizada pelo banco de dados de séries temporais *VictoriaMetrics*, compatível com o ecossistema

Prometheus. O *Grafana* emprega a MetricsQL para consultar, agregar e filtrar séries temporais de longo prazo, permitindo a análise histórica detalhada e a construção de painéis analíticos avançados.

6 Arquitetura de Armazenamento e Visualização

Para uma análise robusta do desempenho do sistema e para a criação de gráficos históricos detalhados, foi implementada uma arquitetura de armazenamento de dados em duas camadas, complementada por uma plataforma de visualização dedicada. Todos os componentes desta arquitetura são executados em contêineres Docker.

6.1 Estratégia de Armazenamento Duplo

O sistema adota uma estratégia de armazenamento duplo para equilibrar desempenho e retenção de dados a longo prazo. O MariaDB e o VictoriaMetrics operam de forma paralela, mas com papéis distintos: o primeiro mantém dados recentes e agregados necessários para operação e visualização imediata, enquanto o segundo armazena o histórico completo de medições para análise aprofundada e visualização avançada no Grafana.

- **Armazenamento de Curto Prazo (MariaDB):** Responsável por registrar eventos e estados detalhados do Home Assistant, utilizados nas visualizações internas, como histórico, logbook e estatísticas.
- **Armazenamento de Longo Prazo (VictoriaMetrics):** Destinado a manter um registro contínuo de todas as métricas exportadas pelo Home Assistant, possibilitando análises históricas extensas e correlações temporais detalhadas.

6.1.1 MariaDB: Banco de Dados Relacional (Curto Prazo)

O Home Assistant utiliza o componente nativo `recorder` para persistir informações de estados e eventos. Por padrão, o sistema emprega um arquivo SQLite, mas nesta implementação o `recorder` foi configurado para utilizar um servidor MariaDB externo, executado em seu próprio contêiner Docker. Essa abordagem oferece melhor

desempenho em consultas, maior robustez e facilita operações de backup e manutenção (ABEDINPOUR, 2024).

O banco de dados MariaDB foi configurado com o parâmetro `purge_keep_days = 10`, o que significa que o Home Assistant realiza uma purga automática dos dados brutos (*raw data*) mais antigos que dez dias. Esses registros são permanentemente excluídos e não arquivados. Entretanto, antes da exclusão, o Home Assistant executa um processo interno de agregação, preservando estatísticas de longo prazo — como médias, mínimos, máximos e somatórios — em tabelas específicas (`statistics` e `statistics_short_term`). Assim, mesmo após a purga dos dados detalhados, informações consolidadas continuam disponíveis para gráficos e dashboards de longo prazo, como o painel de energia.

6.1.2 VictoriaMetrics: Banco de Dados de Série Temporal (Longo Prazo)

Paralelamente ao registro interno no MariaDB, as métricas expostas pelo Home Assistant por meio da integração Prometheus são coletadas pelo VictoriaMetrics. Este é um Banco de Dados de Série Temporal (Banco de Dados de Séries Temporais (TSDB)) de alto desempenho, projetado para ingestão contínua e consultas eficientes de grandes volumes de dados provenientes de sistemas IoT e telemetria.

Diferentemente do MariaDB, o VictoriaMetrics mantém os dados brutos de forma integral e contínua, respeitando exclusivamente a política de retenção definida em sua configuração (por exemplo, meses ou anos). Não há agregação automática dos dados, o que assegura a preservação completa das séries temporais dentro do horizonte de retenção estabelecido, permitindo análises técnicas detalhadas sobre o comportamento do sistema em qualquer ponto do passado.

6.2 Grafana: Visualização Avançada

Embora o Home Assistant forneça gráficos adequados para a interação diária, análises técnicas mais aprofundadas são realizadas no Grafana, executado em um contêiner Docker. O Grafana é a plataforma de código aberto líder para visualização e análise

de métricas.

Neste projeto, o Grafana é utilizado para se conectar diretamente ao banco de dados de longo prazo (VictoriaMetrics). Utilizando a linguagem MetricsQL (detalhada na Seção 5.4), o Grafana permite a criação de paineis de visualização complexos, essenciais para a validação do sistema de controle e para a análise de desempenho da geração fotovoltaica ao longo do tempo.

7 Implementação do Sistema de Controle

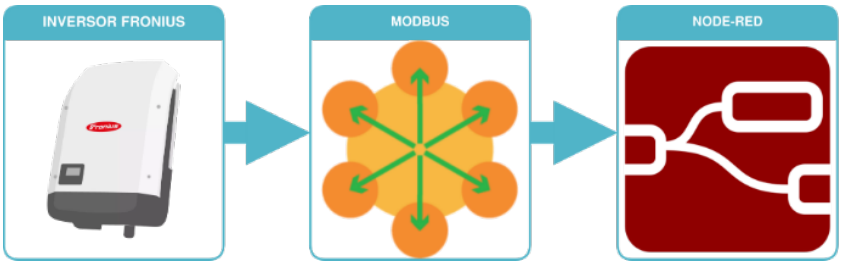
Este capítulo detalha a implementação prática da arquitetura, abrangendo os fluxos de dados no *Node-RED* e a configuração da interface e da lógica de automação no *Home Assistant*.

7.1 Fluxo de Leitura do Inversor Fronius

A integração com o inversor fotovoltaico Fronius via protocolo Modbus TCP tem como objetivo monitorar as grandezas associadas à geração de energia solar e ao estado operacional do equipamento. Na microrrede estudada, este inversor é o elemento responsável pela conversão da energia proveniente do arranjo fotovoltaico, fornecendo dados essenciais tanto para a supervisão do sistema quanto para a implementação da estratégia de controle do tipo *Grid Zero*.

Do ponto de vista estrutural, a aquisição de dados segue o modelo cliente/servidor baseado em requisição e resposta. Nesse arranjo, o inversor Fronius atua como servidor Modbus TCP, disponibilizando os registradores de medição, enquanto o *Node-RED* opera como cliente, responsável por enviar as requisições de leitura e processar as respostas recebidas. O protocolo Modbus TCP estabelece a comunicação padronizada entre os dispositivos, conforme ilustrado no diagrama na Figura 12.

Figura 12 – Diagrama estrutural do fluxo de leitura do inversor Fronius.



Fonte: Autoria própria (2026).

Além do monitoramento passivo, o equipamento desempenha um papel ativo na malha de controle proposta. Seus parâmetros operacionais, especificamente os limites de potência ativa, podem ser ajustados dinamicamente por meio de comandos de escrita Modbus, permitindo a modulação da geração de acordo com a demanda das cargas.

O mapeamento de registradores do equipamento segue estritamente o padrão industrial *SunSpec Modbus*. Nesse padrão, as grandezas elétricas não são apresentadas de forma direta, mas sim codificadas através de estratégias complexas que envolvem fatores de escala dinâmicos e mapas de memória segmentados. Valores físicos frequentemente ocupam múltiplos registradores de 16 bits ou dependem de um segundo registrador para determinar sua escala. Consequentemente, os dados brutos disponibilizados pelo inversor exigem um tratamento aritmético e semântico rigoroso antes que possam ser utilizados pela camada de supervisão.

Diante dessas características, a definição da estratégia de leitura constituiu uma etapa crítica do projeto. As subseções a seguir descrevem a evolução dessa implementação, partindo das tentativas preliminares até a consolidação da arquitetura final adotada para a aquisição e o processamento dos dados.

7.1.1 Tentativas Iniciais de Leitura do Inversor Fronius

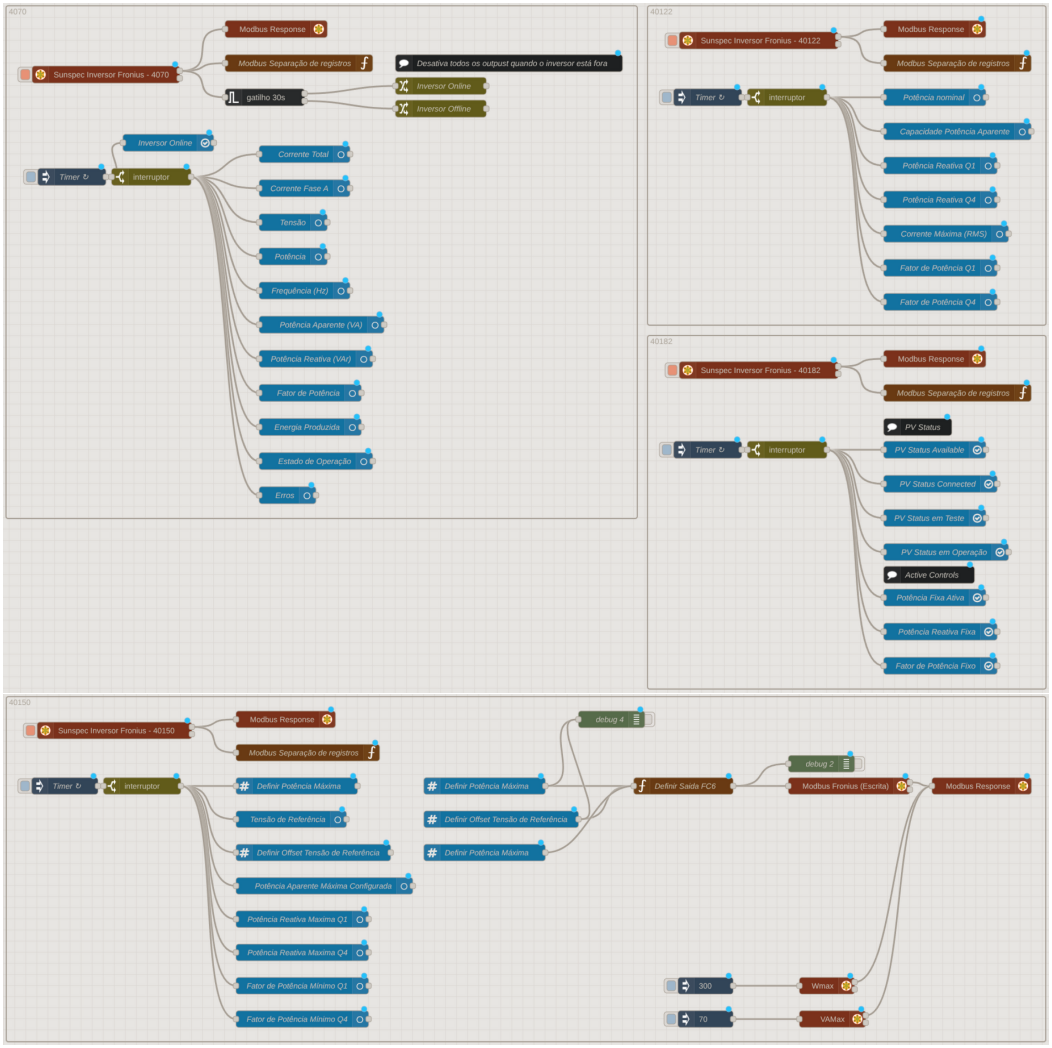
Antes da implementação definitiva da leitura e do controle do inversor Fronius por meio do *Node-RED*, foi avaliada a utilização da integração Modbus nativa do *Home Assistant*. Embora essa abordagem permitisse a comunicação direta com os registradores, tentar realizar todo o tratamento matemático exigido pelo equipamento exclusivamente via arquivos YAML resultou em uma arquitetura rígida e operacionalmente inviável, justificando a necessidade de um ambiente de processamento mais robusto, conforme já discutido na Seção 4.2.

Após essa etapa inicial, optou-se por migrar a leitura dos dados para o *Node-RED*, mantendo a comunicação com o *Home Assistant*. A análise da tabela de mapeamento de registradores, com base no padrão *SunSpec Modbus* (vide Tabela 5, presente no Apêndice A), evidenciou que as grandezas elétricas não são disponibilizadas de forma homogênea, estando organizadas em diferentes blocos e codificadas segundo múltiplas estratégias de representação.

Observou-se que diversas variáveis utilizam técnicas de *integer scaling*, nas quais valores com precisão decimal são representados como inteiros de 16 bits, exigindo a leitura de registradores adicionais para a aplicação de fatores de escala. Adicionalmente, o mapeamento contempla registradores do tipo enumerado, bem como grandezas representadas em 32 e 64 bits, que demandam a junção de múltiplos registradores de 16 bits para a correta reconstrução do valor original.

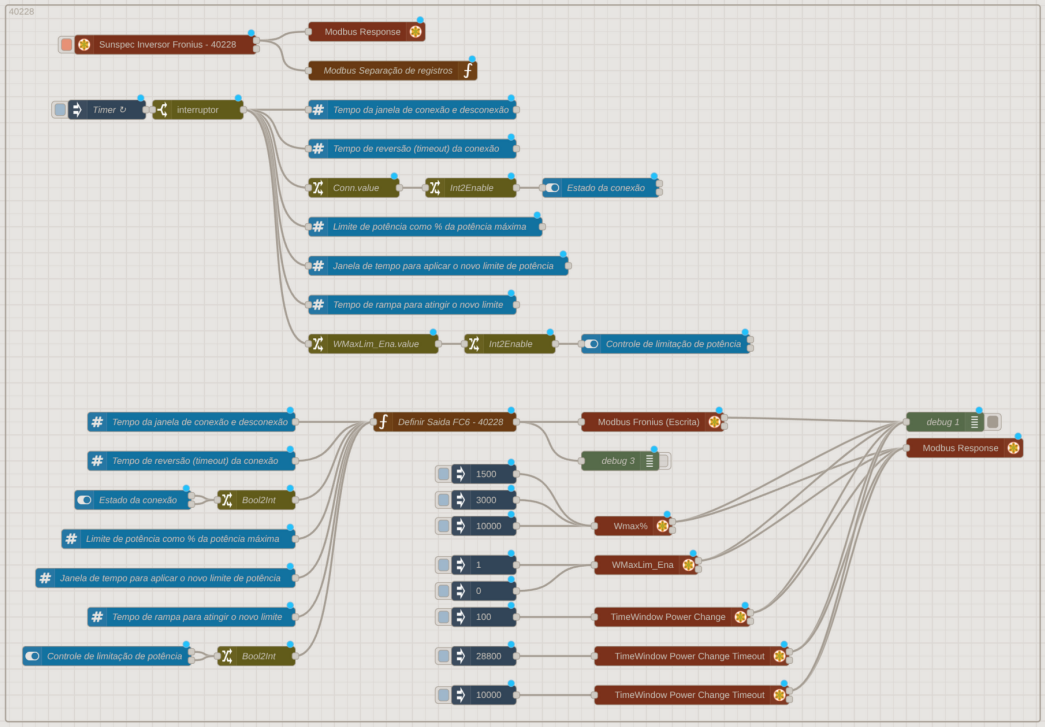
Em função dessa organização, foi inicialmente implementada uma estratégia de leitura segmentada, na qual cada bloco de registradores era consultado separadamente e processado por um código de interpretação específico. Nessa abordagem preliminar, cada conjunto de dados era tratado individualmente e enviado ao *Home Assistant* por meio da extensão *node-red-contrib-home-assistant-websocket*, responsável pela criação e atualização direta das entidades correspondentes; essa implementação pode ser observada na Figura 13.

Figura 13 – Fluxo de leitura e escrita inicial do inversor Fronius no Node-RED.



Fonte: Autoria própria (2026).

Figura 13 – Fluxo de leitura e escrita inicial do inversor Fronius no Node-RED (Continuação).



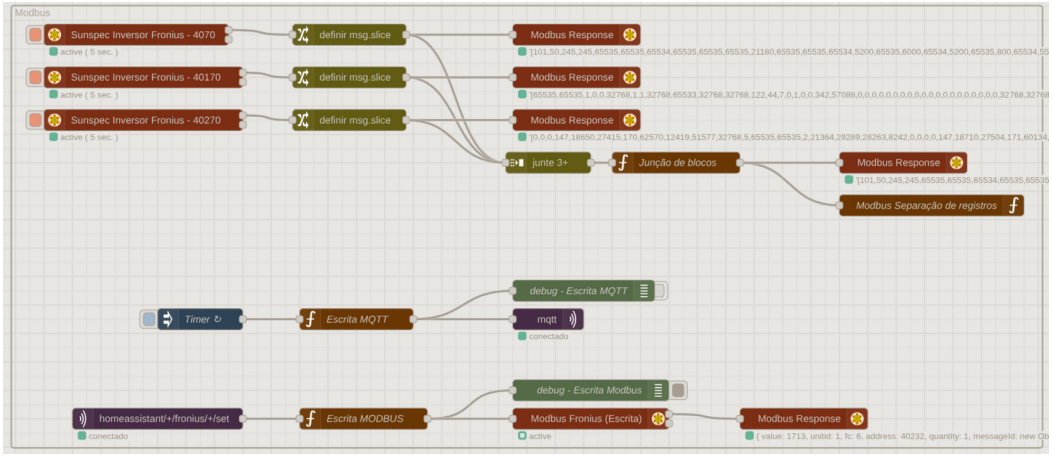
Fonte: Autoria própria (2026).

Embora funcional, essa estratégia mostrou-se pouco escalável e de difícil manutenção. Cada variável precisava ser criada e gerenciada individualmente, e funções auxiliares recorrentes, como a conversão de valores com sinal e a aplicação de fatores de escala, precisavam ser replicadas em diversos trechos de código. Em caso de alterações, era necessário modificar cada código individualmente. Esse cenário aumentava a complexidade do sistema e dificultava a realização de expansões futuras.

Em razão dessas limitações, a estratégia inicial foi substituída por uma abordagem centralizada, na qual múltiplos blocos de registradores são lidos de forma independente, agregados em uma única estrutura de dados e posteriormente processados por um único algoritmo de interpretação. Essa solução permitiu a centralização das rotinas de conversão, reduziu significativamente a redundância de código e simplificou a manutenção do sistema, resultando na arquitetura final descrita a seguir. Está solução final completa

pode ser observado na Figura 14.

Figura 14 – Fluxos de leitura e controle implementados no Node-RED para o inversor Fronius.

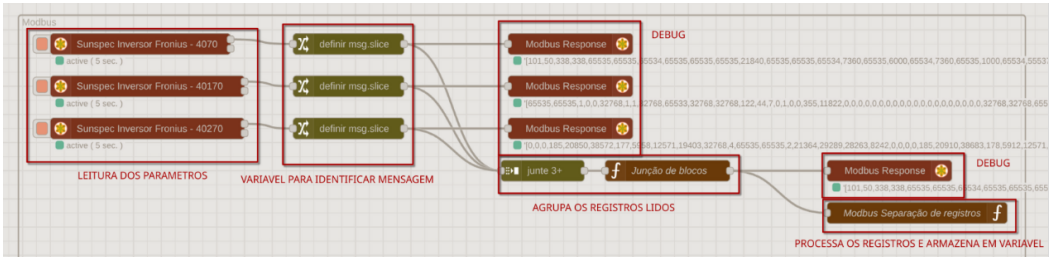


Fonte: Autoria própria (2025).

7.1.2 Fluxo de Leitura Final do Inversor Fronius

Conforme ilustrado na Figura 15, este fluxo é responsável pela leitura periódica dos dados do inversor Fronius. A consulta aos registradores Modbus é realizada através de três nós do tipo `Modbus-Read` (em laranja).

Figura 15 – Fluxos de leitura implementados no Node-RED para o inversor Fronius.



Fonte: Autoria própria (2026).

Esta divisão da leitura é necessária devido à limitação do protocolo Modbus, que permite a leitura de um número finito de registradores por transação (geralmente 125). Desta forma, o processo foi segmentado da seguinte maneira:

- **Primeira Leitura:** Consulta os 100 registradores no intervalo de `40070` a `40169`.
- **Segunda Leitura:** Consulta os 100 registradores no intervalo de `40170` a `40269`.
- **Terceira Leitura:** Consulta os 31 registradores restantes, no intervalo de `40270` a `40300`.

Após as três leituras serem executadas, as mensagens resultantes são agregadas pelo nó `join` (em verde-escuro), configurado para aguardar a chegada das três mensagens e combiná-las em uma única mensagem contendo um vetor com os dados completos. Os nós `change` (em verde-claro) auxiliam neste processo, preparando as mensagens para a junção.

Finalmente, a mensagem unificada é processada pelo nó de função `Modbus Separação de Registros`. Este script em *JavaScript* desempenha um papel fundamental no sistema, executando as seguintes tarefas:

- **Interpretação Padrão SunSpec:** O código decodifica o vetor de dados seguindo a estrutura do padrão industrial **SunSpec Modbus**, organizando as informações em modelos lógicos.
- **Aplicação de Fatores de Escala:** O script localiza e lê os registradores de "Fator de Escala"(*Scale Factors*) e os aplica aos valores inteiros brutos, convertendo-os em unidades de engenharia (e.g., Volts, Amperes, Watts).
- **Tratamento de Tipos de Dados:** Utiliza funções auxiliares para manipular diferentes tipos de dados, como inteiros de 32 bits e valores com sinal, garantindo a integridade da informação.
- **Criação de Objeto Estruturado:** Ao final, gera um único objeto *JavaScript*, onde todos os dados do inversor estão organizados, nomeados e prontos para uso.

Este objeto estruturado é então armazenado em uma variável de contexto global do *Node-RED*, tornando os dados do inversor acessíveis para outros fluxos. O código-fonte completo da função encontra-se disponível no Apêndice C, estando apresentado no Código C.2.

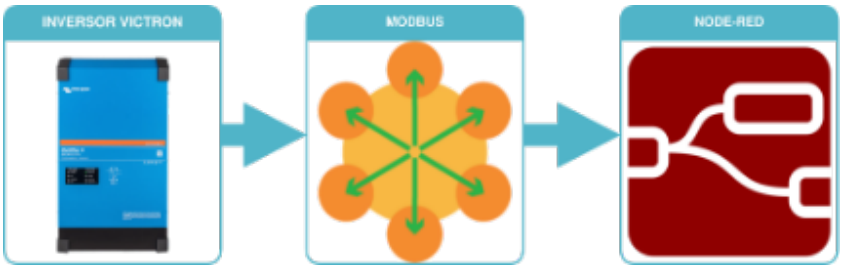
Neste projeto, esses dados são consumidos pelo bloco de função `Escrita MQTT`, cujo funcionamento é detalhado na Seção 7.3.

7.2 Estratégias de Leitura do Inversor Híbrido Victron

A integração com o inversor híbrido Victron Multiplus II via protocolo Modbus TCP tem como objetivo monitorar um conjunto abrangente de variáveis essenciais para o balanço energético da microrrede. Diferente do inversor Fronius, que fornece apenas dados de geração solar, o equipamento da Victron atua como o centralizador das medidas elétricas, disponibilizando dados sobre o consumo das cargas, o fluxo de potência com a rede elétrica (*Grid*), a tensão e corrente do banco de baterias, além do estado dos relés programáveis.

Do ponto de vista estrutural, a aquisição de dados segue o modelo cliente/servidor baseado em requisição e resposta. Nesse arranjo, o inversor híbrido Victron atua como servidor Modbus TCP, disponibilizando os registradores referentes às medições elétricas e ao estado operacional do sistema, enquanto o *Node-RED* opera como cliente, responsável por enviar as requisições de leitura e processar as respostas recebidas. O protocolo Modbus TCP estabelece a comunicação padronizada entre os dispositivos, conforme ilustrado no diagrama da Figura 16.

Figura 16 – Diagrama estrutural do fluxo de leitura do inversor híbrido Victron.



Fonte: Autoria própria (2026).

7.2.1 Desafios de Mapeamento e Limitações Iniciais

Para a leitura do inversor híbrido Victron Multiplus II, buscou-se inicialmente replicar a estratégia de leitura em blocos contínuos utilizada com sucesso no inversor Fronius. Entretanto, testes práticos revelaram uma diferença fundamental no comportamento do servidor Modbus dos dois equipamentos.

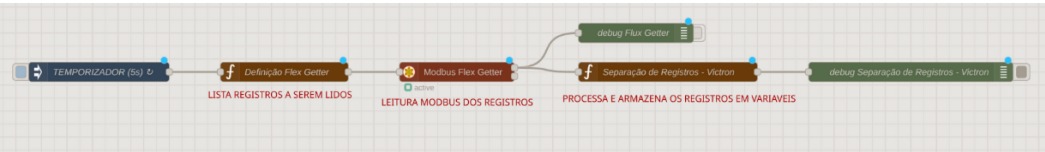
Enquanto o inversor Fronius permite a leitura de intervalos de endereços que contenham registradores não alocados (retornando valores nulos ou o valor máximo de 16 bits, 65.535), o equipamento da Victron não tolera "lacunas"na requisição. Caso uma solicitação de leitura abranja um endereço de memória não definido no mapa de registradores, o inversor interrompe a transação e retorna um erro de exceção Modbus, invalidando todo o bloco de dados solicitado.

Essa característica inviabilizou a estratégia de leitura em grandes blocos sequenciais adotada para o inversor Fronius, uma vez que o mapa de memória do Victron possui endereços dispersos. A alternativa imediata, criar um nó de leitura individual para cada registrador ou pequeno grupo, remeteria aos mesmos problemas de escalabilidade e manutenção descritos na Seção 7.1.1, resultando em um fluxo visualmente poluído e de difícil gestão.

7.2.2 Implementação via Modbus Flex Getter

Para solucionar o problema da descontinuidade dos registradores sem comprometer a organização do fluxo, adotou-se a utilização do nó `Modbus Flex Getter`. Diferente do nó de leitura padrão, que possui parâmetros estáticos, o `Flex Getter` permite que os parâmetros da requisição (endereço inicial, quantidade, ID da unidade) sejam injetados dinamicamente via mensagem. Este fluxo de leitura é apresentado na Figura 17.

Figura 17 – Fluxo de leitura do inversor Victron utilizando o método Flex Getter.



Fonte: Autoria própria (2026).

Dessa forma, foi desenvolvido um algoritmo em *JavaScript* dentro de um nó de função, responsável por orquestrar a leitura. O código define uma lista de objetos, onde cada objeto representa um bloco de endereços válidos e contínuos, "pulando"os endereços vazios que causariam erros.

O Código 7.1 apresenta a implementação dessa lógica. O script itera sobre a lista de blocos definidos e envia as requisições sequencialmente para o nó Modbus. Foi introduzido um atraso controlado de 200 ms entre cada envio (utilizando a função `setTimeout`) para evitar a saturação do *buffer* de comunicação do inversor e garantir a estabilidade da leitura.

Código 7.1 – Função Node-RED para leitura do inversor Victron utilizando o método Modbus Flex Getter.

```
// === Definição Flex Getter (Node-RED Function) ===
// Envia blocos válidos (fc, unitid, address, quantity) para o nó Modbus (Flex Getter).
const blocks = [
    { fc: 3, unitid: 100, address: 800,  quantity: 27 }, // 800..826 (System, AC L1, Grid L1)
    { fc: 3, unitid: 100, address: 830,  quantity: 4  }, // 830..833 (System Time / misc)
    { fc: 3, unitid: 100, address: 840,  quantity: 7  }, // 840..846 (Battery)
    { fc: 3, unitid: 100, address: 850,  quantity: 2  }, // 850..851 (PV DC-coupled)
    { fc: 3, unitid: 100, address: 855,  quantity: 1  }, // 855 (Charger Power)
    { fc: 3, unitid: 100, address: 860,  quantity: 1  }, // 860 (DC System Power)
    { fc: 3, unitid: 100, address: 865,  quantity: 2  }, // 865..866 (VE.Bus)
    { fc: 3, unitid: 100, address: 1026, quantity: 6  }, // 1026..1031 (AC Output L1)
];

// Envia um msg por bloco para o nó Modbus Flex Getter.
// O payload segue o formato esperado: { fc, unitid, address, quantity }
blocks.forEach((block, index) => {
    setTimeout(() => {
        node.send({
            topic: "modbus/request",
            payload: {
                fc: block.fc,
                unitid: block.unitid,
                address: block.address,
                quantity: block.quantity
            }
        });
    }, index * 200); // 200 ms entre requisições para não saturar o buffer
});

return null;
```

Após a execução das leituras, o nó retorna múltiplos vetores (*arrays*) contendo os dados brutos de cada bloco solicitado, conforme ilustrado na Figura 18.

Figura 18 – Mensagens recebidas no bloco de debug do bloco `Modbus Flex Getter`.

Fonte: Autoria própria (2026).

7.2.3 Processamento e Estruturação dos Dados

Uma vez obtidos os vetores de dados brutos apresentados na Figura 18, o fluxo encaminha as mensagens para o nó de função responsável pela interpretação dos valores. Este script desempenha um papel fundamental na normalização dos dados, convertendo o fluxo binário do protocolo Modbus em informações estruturadas e legíveis. O código-fonte completo desta função encontra-se disponível no Apêndice D.

O funcionamento do script baseia-se na correlação entre a posição do dado no vetor recebido e o seu endereço real de memória. O script itera sobre cada elemento do *array* (`msg.payload`) e calcula seu endereço somando o índice atual ao endereço inicial da requisição (obtido via `msg.modbusRequest.address`).

A lógica do algoritmo sustenta-se em três pilares principais:

1. **Mapeamento Semântico:** O código-fonte utiliza um objeto dicionário (`registerMap`) que associa cada endereço calculado a uma chave textual (ex: endereço `820` corresponde a `grid_power_l1`). Além do nome, esse mapa define metadados essenciais, como a unidade de medida (*W, V, A, %*), o tipo de dado e a categoria funcional do sensor.

2. **Tratamento de Tipos e Escalas:** Diferentemente de outros equipamentos, o inversor Victron utiliza extensivamente inteiros com sinal (`int16`) para representar o fluxo de potência, onde valores positivos indicam consumo e negativos indicam injeção na rede. O script implementa funções auxiliares para decodificar corretamente esses valores, além de aplicar os divisores de escala necessários (ex: dividir a tensão por 10 para converter o valor bruto `524` em `52,4 V`).

3. **Estruturação Global:** Ao final do processamento, os dados são armazenados na variável de contexto global `global.set('victron', ...)`. Os dados são organizados em categorias lógicas como `solar_ac`, `battery` e `ac_grid`.

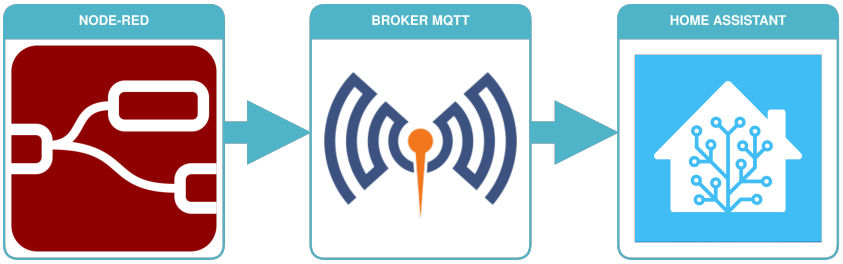
Essa estratégia de armazenamento global é crucial para a arquitetura do sistema, pois desacopla a etapa de leitura da etapa de publicação. Os dados estruturados e armazenados neste ponto serão posteriormente consumidos pelo script `Escrita MQTT`,

responsável pela integração com o *Home Assistant*, cujo funcionamento detalhado é apresentado na Seção 7.3.

7.3 Publicação de Dados via MQTT Discovery

A publicação de dados e a criação de entidades no *Home Assistant* ocorrem de forma automatizada por meio da implementação do protocolo *MQTT Discovery*. Do ponto de vista estrutural, a comunicação segue o modelo publicador/assinante, no qual o *Node-RED* processa e publica os dados, o *Broker* MQTT atua como o roteador intermediário, e o *Home Assistant* consome as informações, conforme ilustrado no diagrama na Figura 19.

Figura 19 – Diagrama estrutural do fluxo de dados via MQTT.



Fonte: Autoria própria (2026).

Na implementação prática, conforme ilustrado pelos fluxos desenvolvidos no *Node-RED* (Figura 20), o processo segue uma sequência lógica iniciada por nós do tipo *Timer* (destacados em azul), responsáveis por acionar a rotina de publicação periodicamente a cada 5 segundos.

Em seguida, ambos os fluxos (Fronius e Victron) encaminham o pulso para um nó de função, que atua como o organizador da lógica. Esta etapa é crucial para a arquitetura, pois o *script* interno converte os objetos de dados estruturados, armazenados nas variáveis globais, nos formatos *payload* e tópico exigidos para a criação de entidades acionáveis na plataforma de supervisão. Essa abordagem permite ao *Node-RED* instruir o *Home Assistant* a criar e configurar componentes automaticamente, eliminando a necessidade de edição manual de arquivos YAML. Os *scripts* embutidos nesses nós (detalhados no Apêndice E para o inversor Fronius e no Apêndice F para o inversor Victron) iteram sobre os dados processados e geram as configurações baseadas nos metadados de cada variável.

Por fim, após a formatação, os dados são enviados ao nó MQTT (bloco em roxo). Este é o componente final responsável por realizar a conexão com o *broker* MQTT e publicar as mensagens efetivamente na rede, tornando-as visíveis e utilizáveis pelo *Home Assistant*.

Figura 20 – Fluxos de escrita MQTT dos inversores do projeto no Node-RED.

(a) Fluxo do inversor Fronius.

(b) Fluxo do inversor Victron.

Fonte: Autoria própria (2026).

Para realizar essa integração e formatação prévia, o algoritmo inserido no nó de função executa as seguintes lógicas para cada ponto de dado:

1. **Definição do Tipo de Entidade:** O código verifica as propriedades da variável para definir seu domínio no *Home Assistant*:
    - **Sensor (`sensor`):** Para variáveis de apenas leitura (ex: Potência, Tensão).
    - **Interruptor (`switch`):** Para variáveis binárias de escrita (ex: `WMaxLim_Ena` do Fronius ou Relés do Victron).
    - **Número (`number`):** Para variáveis analógicas de escrita (ex: Limite percentual `WMaxLimPct`).

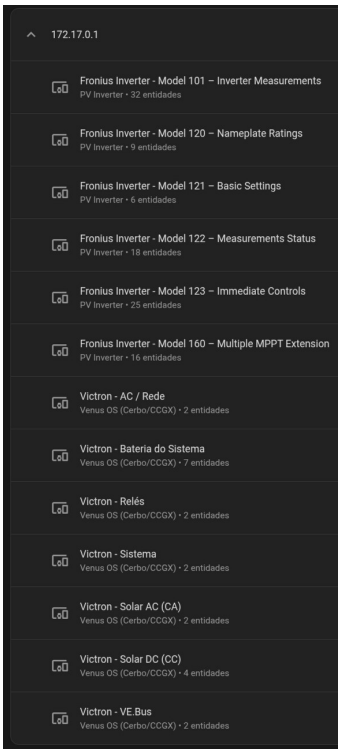
2. **Agrupamento de Dispositivos:** Para evitar a poluição visual na interface e organizar a grande quantidade de sensores, as entidades são agrupadas logicamente em dispositivos virtuais:

    - No **Fronius**, o agrupamento reflete a estrutura dos Modelos SunSpec:
        - **Model 101 (Inverter Measurements):** Medições do inversor (Potência AC, Corrente, Tensão).
        - **Model 120 (Nameplate Ratings):** Dados de placa (Potência e correntes nominais).
        - **Model 121 (Basic Settings):** Configurações básicas (Limites padrão de operação).
        - **Model 122 (Measurements Status):** Status das medições (Estados de conexão PV e Rede).
        - **Model 123 (Immediate Controls):** Controles imediatos (Ajuste de limite de potência e fator de potência).
        - **Model 124 (Basic Storage Controls):** Controles de armazenamento (Não utilizado neste projeto).
        - **Model 160 (Multiple MPPT Extension):** Extensão MPPT (Dados individuais das *strings* DC).
    - No **Victron**, o agrupamento é funcional, segmentado em sete dispositivos:
        - **Solar AC (CA):** Produção fotovoltaica acoplada em corrente alternada.
        - **Solar DC (CC):** Produção fotovoltaica acoplada em corrente contínua (controladores de carga).
        - **AC / Rede:** Monitoramento da conexão com a concessionária (Importação/Exportação).
        - **Bateria do Sistema:** Estado do banco de baterias (Tensão, Corrente, SOC).
        - **Relés:** Controle e estado dos relés programáveis do equipamento.
        - **VE.Bus:** Dados internos do Sistema de Gerenciamento de Baterias (BMS) do inversor Victron.
        - **Sistema:** Informações gerais de status e fonte ativa de energia.

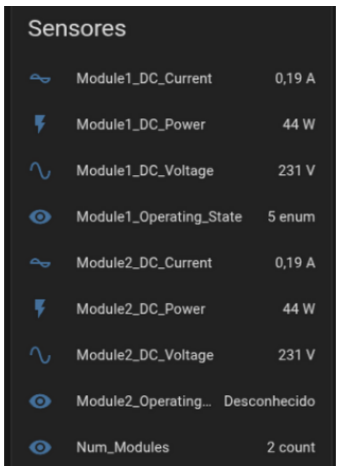
3. **Classificação Semântica:** Com base na unidade de medida (W, V, A, kWh), o script atribui automaticamente as propriedades `device_class` e `state_class`, permitindo que o *Home Assistant* gere gráficos estatísticos e trate a acumulação de energia corretamente.

A partir da integração via protocolo MQTT *Discovery*, foi possível registrar automaticamente as diversas variáveis dos inversores. Na Figura 21 é apresentado o conjunto de dispositivos e entidades gerados no *Home Assistant*.

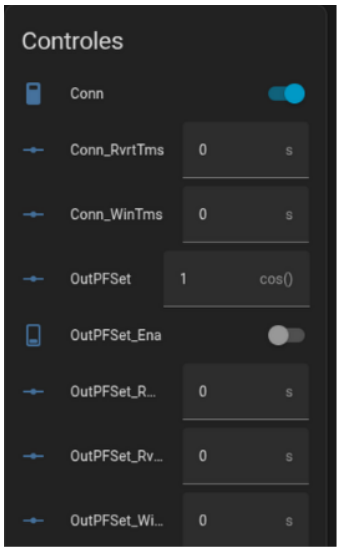
Figura 21 – Configurações das entidades MQTT no Home Assistant, mostrando a detecção de dispositivos e a criação automática de sensores.



(a) Dispositivos MQTT gerados.



(b) Entidades MQTT para monitoramento do inversor Fronius.



(c) Entidades MQTT para o controle do inversor Fronius.

Fonte: Autoria própria (2026).

Para cada entidade, são publicados até três tópicos MQTT distintos, todos con-

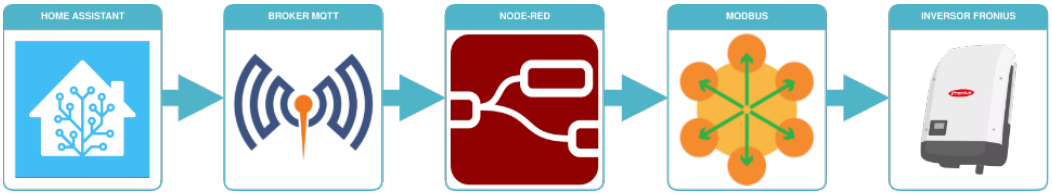
figurados com a flag `retain: true` para garantir a persistência dos dados em caso de reinicialização do servidor:

- **Tópico de Configuração (**`.../config`**):** Contém o *payload* JSON de descoberta, definindo nome, ID único e capacidades da entidade.
- **Tópico de Estado (**`.../state`**):** Transporta o valor atual da medição ou o estado do interruptor.
- **Tópico de Atributos (**`.../attributes`**):** (Específico da implementação Victron) Publica metadados adicionais, como o endereço Modbus original e a descrição técnica do registrador, facilitando a depuração do sistema.

7.4 Fluxo de Subscrição e Controle Modbus

O fluxo de controle, responsável por enviar comandos do usuário de volta ao inversor Fronius, obedece a um caminho estrutural reverso ao da leitura de dados. Nesse modelo, o *Home Assistant* atua como a interface de emissão do comando, o *Broker* MQTT roteia a mensagem, o *Node-RED* recebe e converte a instrução, e o protocolo Modbus TCP atua como o meio de transmissão final até o inversor Fronius. O diagrama que representa está arquitetura é apresentado na Figura 22.

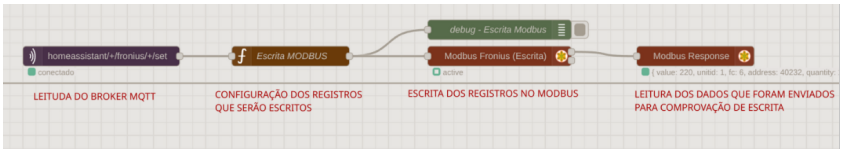
Figura 22 – Arquitetura geral do fluxo de controle entre Home Assistant, Broker MQTT, Node-RED e inversor Fronius via Modbus TCP.



Fonte: Autoria própria (2026).

De forma mais detalhada,a implementação desse fluxo lógico no *Node-RED* é apresentada na Figura 23.

Figura 23 – Fluxos de escrita Modbus do inversor Fronius no Node-RED.



Fonte: Autoria própria (2026).

O processo é iniciado por um nó `mqtt in` (em roxo), que se inscreve em um tópico MQTT dinâmico para receber comandos oriundos do *Home Assistant*.

Este nó utiliza a subscrição no tópico `homeassistant/+/fronius/+/set`, onde o caractere + atua como um *wildcard* (curinga). Essa configuração permite que um único

nó de entrada escute simultaneamente os canais de comando de todas as entidades, sejam elas do tipo interruptor (`switch`) ou numérico (`number`).

É fundamental destacar a convenção de tópicos utilizada para evitar loops de controle:

- **Tópicos `/state`:** Usados pelo *Node-RED* para **publicar** o estado atual lido do dispositivo.
- **Tópicos `/set`:** Usados pelo *Node-RED* para **receber** comandos, publicados pelo *Home Assistant* quando o usuário interage com a IHM ou quando uma automação é disparada.

Ao receber uma mensagem em um tópico `/set`, o fluxo a encaminha para o nó de função `Escrita Modbus`. Este script desempenha o papel de tradutor entre o protocolo de mensageria e o protocolo industrial, executando quatro etapas lógicas essenciais, conforme detalhado no Código disponível no Apêndice G:

1. **Decomposição do Tópico (Regex):** O script utiliza expressões regulares para analisar a string do tópico recebido (ex: `.../fronius/model123_WMaxLimPct/set`) e extrair o identificador único da variável alvo.
2. **Resgate de Metadados:** Com base no identificador extraído, o código consulta a variável global `fronius` (populada anteriormente pelo fluxo de leitura). Isso permite recuperar dinamicamente o endereço de memória do registrador e, crucialmente, o seu Fator de Escala, sem a necessidade de duplicar essas informações ("hardcoding") no fluxo de escrita.
3. **Cálculo Inverso de Escala:** Para variáveis numéricas, o script deve reverter a lógica aplicada na leitura. Se o inversor espera um valor inteiro bruto, o valor recebido da interface (em engenharia) deve ser multiplicado pela potência inversa do fator de escala, conforme a Equação 7.1:

$$Valor_{Modbus} = Valor_{IHM} \times 10^{-SF} \tag{7.1}$$

   Considere, por exemplo, o ajuste da potência do inversor para 1500 W (50% da capacidade nominal de 3000 W). O registrador responsável por definir esse limite

é o `WMaxLimPct` (endereço 40232), que utiliza como referência o fator de escala `WMaxLimPct_SF` (endereço 40251), cujo valor constante é -2. Aplicando a fórmula, o cálculo realizado é $50 \times 10^{-(-2)} = 50 \times 100 = 5000$. Desta forma, o sistema envia o valor inteiro 5000 para comandar o limite de 50%.

4. **Formatação do Payload:** Por fim, o script constrói o objeto de comando exigido pelo nó `Modbus-Flex-Write`, definindo o código de função `fc: 6` (Escrita de Registrador Único) e ajustando o endereço de memória (subtraindo 1 para adequar-se à indexação base zero do *Node-RED*).

Após esse processamento, o comando formatado é enviado ao nó `Modbus Fronius (Escrita)`, que é do tipo `Modbus-Flex-Write`. Diferente do nó de escrita padrão, que opera com endereços fixos, este componente permite que todos os parâmetros da transação Modbus (endereço, valor e ID da unidade) sejam definidos dinamicamente pela mensagem de entrada. Conectado à saída deste nó, encontra-se o nó `Modbus-response`, utilizado exclusivamente para fins de depuração, permitindo a verificação do sucesso dos comandos enviados e a identificação de eventuais erros.

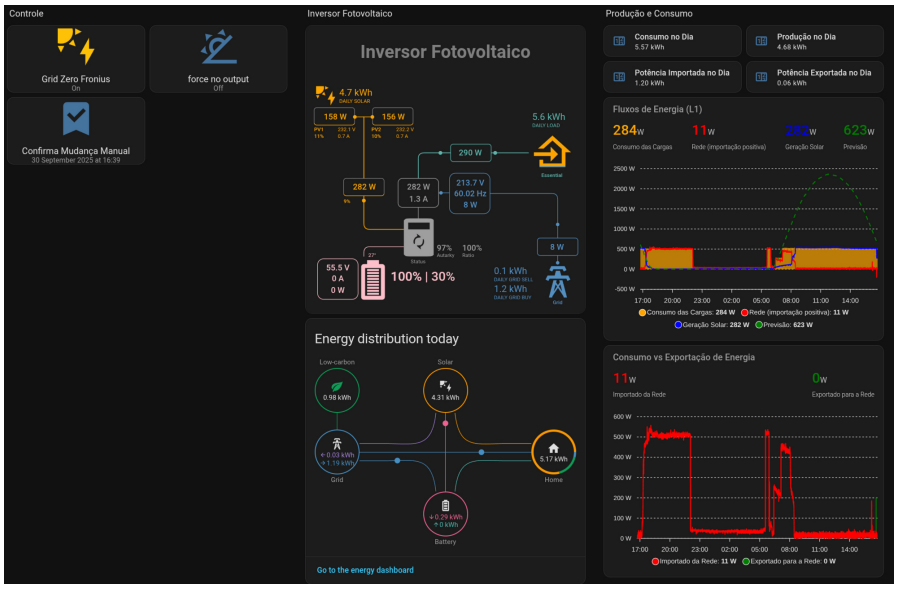
7.5 Interface Homem-Máquina no Home Assistant

Uma vez que os dados dos inversores são processados no *Node-RED* e publicados no barramento MQTT, o *Home Assistant* detecta e registra automaticamente os dispositivos conforme as configurações de descoberta enviadas. A partir desse ponto, a plataforma assume o papel de central de supervisão, permitindo o monitoramento em tempo real e a atuação sobre o sistema.

Na interface do *Home Assistant*, é possível visualizar variáveis críticas como potência consumida, potência exportada, energia acumulada (importada e exportada), bem como a potência instantânea da rede e dos geradores. Além das grandezas elétricas da microrrede, o *dashboard* integra dispositivos controlados via protocolo *Zigbee*. Entre estes, destacam-se a carga auxiliar (refletor halógeno), operada por meio de um *smart plug*, e um aparelho de televisão localizado no corredor do laboratório iTec-lab, controlado através de um relé inteligente.

A inclusão da TV no sistema justifica-se pelo aproveitamento da infraestrutura de automação existente no laboratório, permitindo a centralização de dispositivos periféricos em uma única interface. Por meio do *Home Assistant*, foi implementada uma automação para gerenciar o horário de funcionamento desse equipamento. Embora a TV não faça parte do balanço energético estrito da microrrede, sua integração demonstra a versatilidade da plataforma em unificar diferentes ecossistemas. Na Figura 24 é ilustrado o painel de controle e monitoramento no Home Assistant exibindo os sensores e suas leituras em tempo real, além dos botões para acionamento manual da carga auxiliar.

Figura 24 – painel de controle e monitoramento desenvolvno Home Assistant

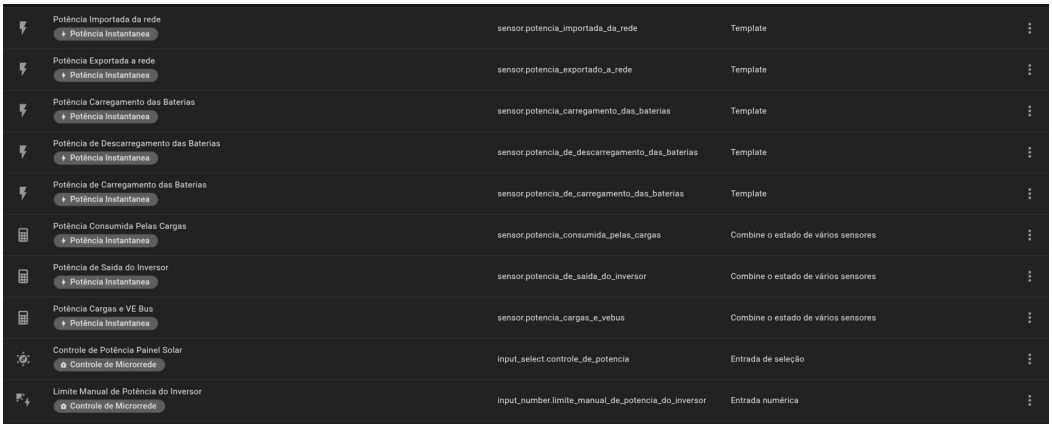


Fonte: Autoria própria (2025).

7.5.1 Processamento de Métricas e Ajudantes do Home Assistant

Alguns dados fundamentais, como o fluxo de potência da rede e da bateria, são registrados pelo inversor como grandezas bidirecionais com sinal: valores positivos indicam importação da rede ou carga da bateria, enquanto valores negativos indicam exportação ou descarga. Para a correta contabilização de energia, foi necessária a criação de Entidades Ajudantes (*Helpers*) do tipo *Template Sensor* para segregar esses valores em entidades distintas (apenas importação ou apenas exportação). Esses ajudantes podem ser observados na parte superior da Figura 25 e na parte superior esquerda da Figura 27.

Figura 25 – Entidades Ajudantes: Separação de potência e variáveis de controle.

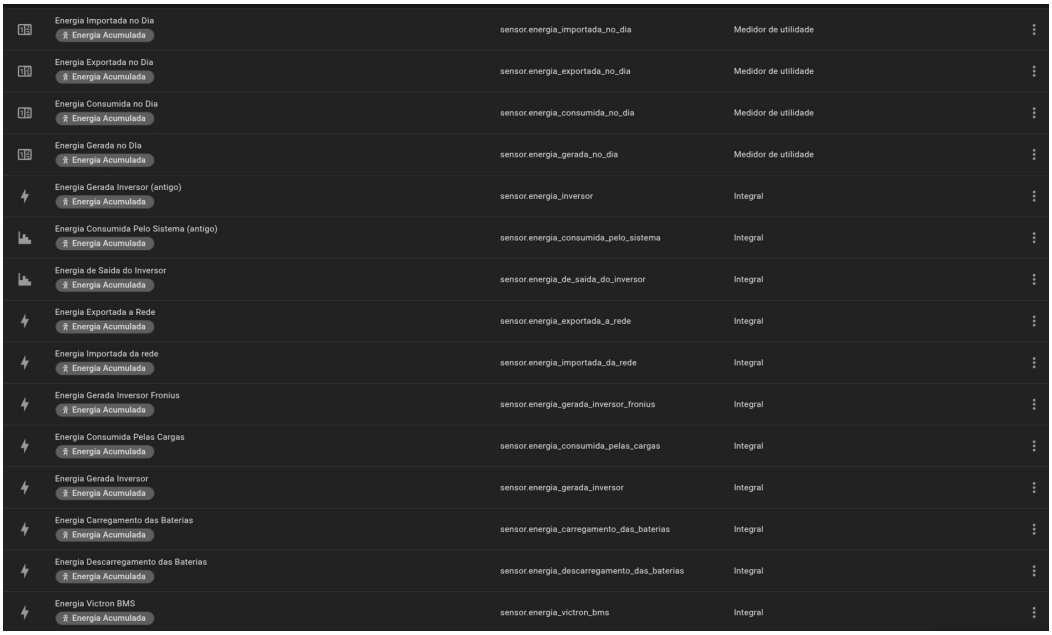


Fonte: Autoria própria (2025).

Adicionalmente, para obter uma visão consolidada do consumo total do sistema, criou-se a entidade `Potência Cargas e VE Bus`, que realiza a soma aritmética do consumo das cargas essenciais com o consumo interno do BMS e do inversor Victron.

Para viabilizar o painel de gerenciamento de energia, foi necessário processar as grandezas de potência instantânea (W) para obter valores de energia acumulada (kWh). Para isso, utilizaram-se sensores de integração baseados na Soma de Riemann à esquerda (*Integration - Riemann sum left*). Esses sensores realizam a integração temporal da potência lida, permitindo contabilizar a produção e o consumo ao longo do tempo. Tais entidades podem ser observadas no centro e na parte inferior da Figura 26.

Figura 26 – Entidades Ajudantes: Integração de energia acumulada e contadores diários.

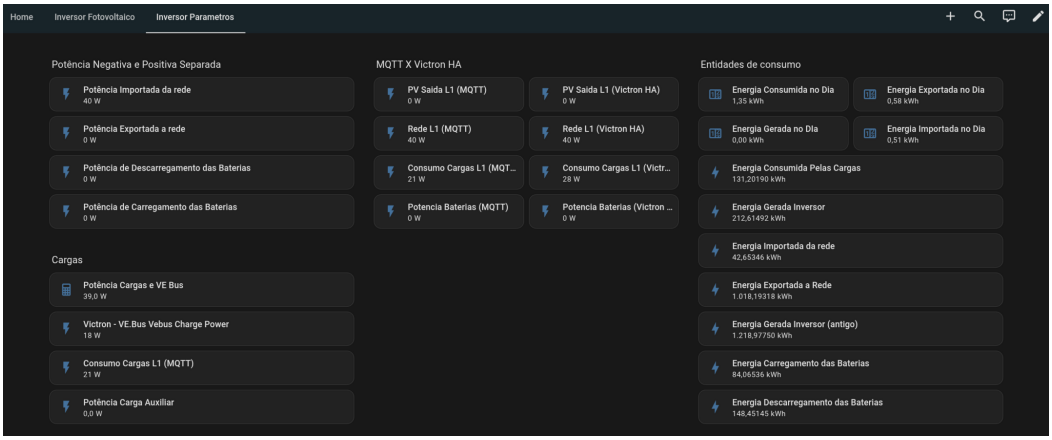


Fonte: Autoria própria (2026).

Complementarmente, foram implementados Contadores de Utilidade (*Utility Meters*), responsáveis por segmentar esses dados acumulados em ciclos diários. Conforme observado na parte superior da Figura 26, essas ferramentas fornecem as métricas necessárias para a aba de energia, apresentada na Figura 28, que consolida o balanço energético do sistema.

A interface serviu também como ferramenta de validação para a arquitetura de comunicação. No centro da Figura 27, é apresentada uma comparação entre os dados registrados via MQTT (provenientes do *Node-RED*) e os valores lidos pela integração nativa *hass-victron*. Esta análise confirmou a necessidade da implementação via Modbus no *Node-RED*, uma vez que esta apresentou uma frequência de atualização superior (3 s), essencial para a estabilidade do laço de controle, em contraste com o intervalo de 60 s da integração padrão.

Figura 27 – Parâmetros de potência e energia do sistema (Comparativo de atualização).



Fonte: Autoria própria (2026).

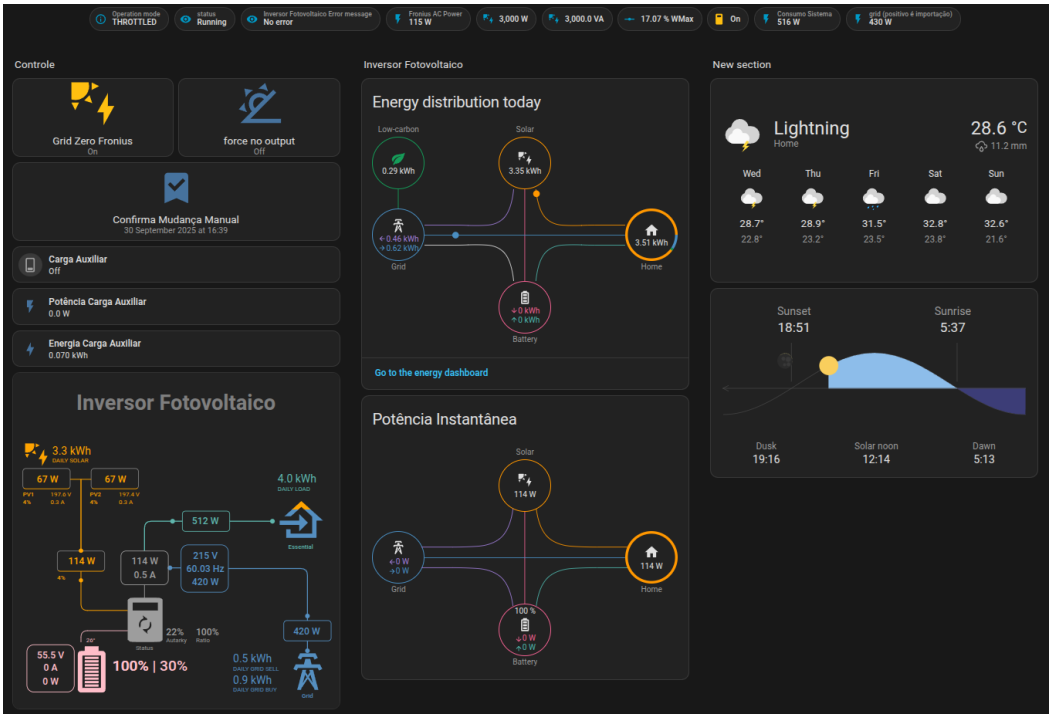
Figura 28 – Painel de monitoramento de energia desenvolvido no Home Assistant, consolidando consumo, produção e balanço energético.

Fonte: Autoria própria (2026).

7.5.2 Painel de Controle Centralizado (Dashboard)

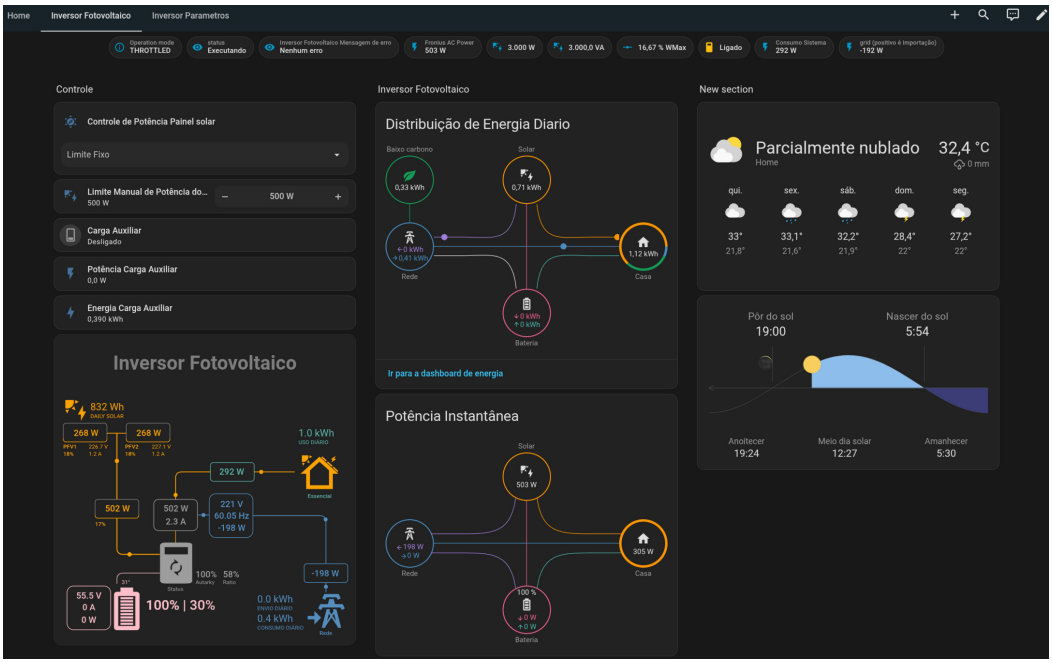
A culminação da integração entre o sistema de hardware e as ferramentas de software é apresentada no *dashboard* centralizado do *Home Assistant*, ilustrado na Figura 29 e na Figura 30. Este painel foi projetado para fornecer ao operador tanto uma visão macroscópica do balanço energético quanto o controle detalhado dos parâmetros de operação da microrrede.

Figura 29 – IHM para supervisão e controle da microrrede operando em *Grid Zero*.



Fonte: Autoria própria (2026).

Figura 30 – IHM para supervisão atualizado operando em modo limite fixo de potência.



Fonte: Autoria própria (2026).

A interface é organizada em quatro zonas funcionais principais, descritas a seguir:

1. **Barra Superior de Status:** Exibe o estado operacional do inversor fotovoltaico Fronius em tempo quase real. No exemplo apresentado na Figura 29, observa-se o estado *Throttled*, indicando que a potência de saída do inversor está sendo limitada pela lógica de controle. O valor de 17,07% refere-se à porcentagem da potência nominal do inversor (3000 W), o que corresponde a aproximadamente 512 W de potência máxima permitida naquele instante. Esse valor é coerente com o consumo total do sistema, registrado em 516 W, evidenciando o funcionamento correto da estratégia de controle do tipo *Grid Zero*. Entretanto, devido às condições ambientais desfavoráveis — caracterizadas por clima chuvoso e baixa irradiância solar no horário do registro — a geração efetiva do sistema fotovoltaico encontra-se limitada a aproximadamente 115 W. Também são apresentados os limites nominais do inversor (3000 W/3000 VA), o estado do controle *Grid Zero* (ativo) e o fluxo

da rede elétrica, no qual o indicador de *Grid Positivo* representa a importação de energia da concessionária para suprir a demanda excedente das cargas.

2. **Painel Lateral de Controle e Cargas:** Localizado à esquerda da interface, este painel concentra os principais comandos de operação manual do sistema. Nele é possível habilitar ou desabilitar o modo de controle *Grid Zero*, bem como acionar a função *Force no output*, responsável por manter a saída do inversor Fronius em zero para fins de segurança ou testes experimentais. Adicionalmente, esta seção permite o controle da carga auxiliar, exibindo seu estado (ligado ou desligado), a potência instantânea em watts e o consumo acumulado em kWh. Após a atualização do controle de potência este painel foi modificado para se ter um seletor do controle de potência, como pode ser observado na Figura 30.

3. **Diagrama da Microrrede e Estado dos Equipamentos:** A seção inferior esquerda apresenta uma representação gráfica da microrrede utilizando o cartão *Sunsynk Power Flow*. Este diagrama ilustra a organização do sistema, incluindo a potência individual dos dois rastreadores MPPT do inversor Fronius, a potência de saída do inversor híbrido Victron, o consumo das cargas essenciais, o fluxo de energia proveniente da rede elétrica e o estado do banco de baterias. São indicados, ainda, o sentido do fluxo energético (carga ou descarga) e o estado de carga do banco de baterias em termos percentuais.

4. **Fluxo de Energia e Contexto Ambiental:** Na região central do painel, são apresentados dois diagramas complementares. O primeiro, implementado por meio do *Energy Flow Card*, exibe a distribuição da energia acumulada ao longo do dia, evidenciando a produção solar, o consumo das cargas e a fração de energia proveniente de fontes renováveis. O segundo diagrama, baseado no *Power Flow Card*, representa o fluxo de potência instantânea no momento da visualização. A informação referente à participação de fontes renováveis no consumo é obtida por meio da integração *Electricity Maps*. À direita do painel, são exibidos dados meteorológicos e um gráfico indicando o horário de nascer do sol (*sunrise*) e pôr do sol (*sunset*), fornecendo subsídios para a previsão qualitativa da disponibilidade de geração fotovoltaica ao longo do dia.

A organização do *dashboard* permite uma supervisão clara e integrada da microrrede, possibilitando a verificação do correto funcionamento do sistema de controle inspirado na filosofia *Grid Zero*. A coerência entre os valores de consumo, limitação de potência e geração efetiva confirma a eficácia da comunicação entre o *Node-RED*, responsável pela aquisição e tratamento dos dados, e o *Home Assistant*, encarregado da visualização e interação com o operador.

7.6 Controle de Potência do Inversor Fronius

Para a implementação do controle de potência de saída do inversor fotovoltaico Fronius, foram utilizados os registradores `WMaxLimPct` (endereço Modbus 40233) e `WMaxLim_Ena` (endereço Modbus 40237). O registrador `WMaxLimPct` define o limite de potência ativa do inversor como uma porcentagem de sua potência nominal — neste caso, 3000 W — enquanto o registrador `WMaxLim_Ena` habilita ou desabilita o modo de limitação de potência.

Conforme descrito no manual Modbus do inversor Fronius (FRONIUS, 2024, p. 85), qualquer alteração no valor do limite de potência durante a operação requer a reativação explícita do modo de controle. Em razão disso, toda atualização do valor de `WMaxLimPct` é precedida pela escrita do valor lógico 1 no registrador `WMaxLim_Ena`, garantindo a aplicação do novo limite.

Para conferir flexibilidade ao sistema e permitir a realização de diferentes experimentos, a lógica de controle foi evoluída para uma máquina de estados implementada no *Home Assistant*. Em vez de um simples acionamento binário (em que pode ser observado na Figura 29), utilizou-se um auxiliar de seleção (*Input Select*), denominado `input_select.controle_de_potencia` (o qual pode ser observado na Figurafig:dashboard_principal_novo), que permite ao operador alternar entre quatro modos distintos de funcionamento:

- **Normal:** Envia o comando para desabilitar a limitação de potência (escreve 0 em `WMaxLim_Ena`), permitindo que o inversor opere livremente em sua capacidade máxima de geração ou conforme a disponibilidade solar.

- **Limite Fixo:** Habilita a limitação e define um valor percentual estático, baseado em um número definido manualmente pelo usuário através do auxiliar `limite_manual_de_potencia_do_inversor`.
- **Grid Zero:** Modo de controle dinâmico. O sistema calcula continuamente a potência necessária para suprir apenas as cargas locais (baseado na leitura do Victron) e atualiza o limite do inversor Fronius.
- **Saída Zero:** Força o limite de potência para 0%, interrompendo a geração ativa sem desconectar o inversor da rede, útil para testes de segurança e manutenção.

A automação é executada periodicamente a cada 15 s. A estrutura condicional verifica o estado do seletor e executa as ações correspondentes via serviços do *Home Assistant*, conforme apresentado no Apêndice H.

Quando o modo **Grid Zero** é selecionado, utilizou-se como variável de referência a potência consumida pelas cargas locais, medida pelo inversor híbrido Victron por meio do registrador `AC Consumption L1`. O sistema executa os seguintes passos a cada ciclo de controle:

1. Reativa o modo de limitação de potência do inversor, escrevendo o valor lógico 1 no registrador `WMaxLim_Ena`;
2. Calcula o novo limite percentual de potência com base na potência consumida pelas cargas, conforme a Equação 7.2:

$$P_{\text{limite}} = \frac{P_{\text{carga}}}{P_{\text{max}}} \times 100 \tag{7.2}$$

onde $P_{\text{max}} = 3000\,\text{W}$ representa a potência nominal do inversor Fronius.

Dessa forma, o inversor fotovoltaico fornece apenas a potência necessária para suprir a demanda instantânea das cargas locais, evitando a exportação de energia excedente para a rede elétrica e apresentando um comportamento operacional equivalente ao de um sistema *Grid Zero*.

7.7 Automação de Cargas Periféricas e Dispositivos Auxiliares

Além do controle de potência do inversor fotovoltaico, o sistema implementado contempla a automação de cargas periféricas do laboratório por meio da integração ZHA (*Zigbee Home Automation*). Essas automações não integram a malha de controle do inversor, mas desempenham papel relevante no suporte experimental e na operação do ambiente.

Entre os dispositivos automatizados, destaca-se a carga não essencial, constituída por um refletor halógeno conectado a um *smart plug* Zigbee. O acionamento controlado dessa carga foi utilizado como ferramenta experimental para a aplicação de degraus de carga bem definidos, permitindo avaliar a resposta dinâmica e a estabilidade do sistema de controle de potência descrito na Seção 7.6. O controle e a supervisão dessa carga são realizados por meio do *Home Assistant*, que disponibiliza informações de estado, potência instantânea e energia acumulada.

Adicionalmente, uma televisão localizada no laboratório foi integrada ao sistema de automação utilizando a mesma infraestrutura Zigbee. Embora esse dispositivo não faça parte do balanço energético da microrrede nem do sistema de controle proposto, sua inclusão demonstra a capacidade da plataforma em unificar o gerenciamento de dispositivos heterogêneos em uma única Interface Homem-Máquina, reforçando a versatilidade da arquitetura adotada.

8 Resultados e Discussão

Este capítulo apresenta e discute os resultados obtidos com a implementação do sistema de controle de limitação de potência inspirado no conceito *Grid Zero*. A análise foi conduzida por meio da comparação do desempenho da microrrede em dois cenários distintos: operação com o controle de potência desativado (condição de referência) e operação com o controle de potência ativo.

Os dados foram coletados e visualizados por meio da plataforma *Home Assistant*, utilizando o painel de energia integrado ao *Grafana*. As medições em tempo real dos inversores Fronius e Victron foram registradas no banco de dados *VictoriaMetrics*, possibilitando a análise detalhada do comportamento do sistema ao longo do período de operação por meio dos gráficos gerados na plataforma *Grafana*.

8.1 Descrição dos Gráficos

As figuras apresentadas a seguir foram geradas pelas plataformas *Home Assistant* e *Grafana*, ambas configuradas para monitorar o fluxo e a distribuição diária de energia da microrrede. Para garantir a clareza na interpretação dos resultados e evitar a repetição exaustiva de legendas, os dados estão organizados segundo as convenções visuais descritas abaixo.

Painéis do Home Assistant

Nos gráficos extraídos do painel de energia nativo do *Home Assistant* (a exemplo da Figura 31), a leitura obedece ao seguinte padrão de cores e agrupamentos:

- **Gráfico superior – Uso de energia:**
  - **Colunas laranjas:** potência solar instantaneamente utilizada pelas cargas (autoconsumo);
  - **Colunas azuis:** potência importada da rede elétrica para suprir as cargas do sistema;

  - **Colunas roxas:** potência excedente exportada para a rede elétrica;
  - **Colunas rosas:** potência destinada ao carregamento das baterias.
  - **Colunas verde claro:** potência consumida das baterias para suprir as cargas do sistema.

- **Gráfico inferior – Produção solar:**
  - **Colunas laranjas:** geração solar instantânea total medida no inversor Fronius;
  - **Linha pontilhada:** curva prevista de geração solar diária, fornecida pela integração `Forecast.Solar`.

Adicionalmente, os medidores circulares (*gauges*) laterais do *dashboard* apresentam os seguintes indicadores consolidados de desempenho:

- **Retorno líquido à rede:** balanço final entre a energia importada e a exportada;
- **Energia solar autoconsumida:** percentual da geração fotovoltaica consumido localmente;
- **Autossuficiência:** fração do consumo total da instalação que foi suprida pela geração própria;
- **Energia de baixo carbono consumida:** proporção da energia total proveniente de fontes renováveis, calculada com base nos dados da integração *Electricity Maps*.

Painéis do Grafana

Para as análises temporais de alta resolução geradas no *Grafana* (como a ilustrada na Figura 32), a representação visual adota a seguinte convenção estrutural no **Gráfico de Fluxo de Energia**:

- **Linha pontilhada verde:** curva prevista de geração solar diária fornecida pela integração `Forecast.Solar`;
- **Linha amarela:** produção solar registrada;

- **Linha azul:** potência de intercâmbio com a rede elétrica (valores positivos indicam importação, enquanto valores negativos indicam exportação de excedente);
- **Linha laranja:** consumo de energia elétrica das cargas.

8.2 Sistema sem Controle de Potência (Linha de Base)

Na Figura 31 é apresentado o comportamento da microrrede no dia 4 de novembro de 2025, operando com o sistema de limitação de potência desativado, caracterizando a condição de linha de base para comparação com os demais cenários. Para uma análise mais aprofundada, na Figura 32 é detalhado essa mesma dinâmica operacional por meio de gráficos gerados no Grafana.

Figura 31 – Fluxo diário de energia em 4 de novembro de 2025, sem aplicação de controle de potência.

Fonte: Autoria própria (2026).

Figura 32 – Gráfico gerado no grafana para o dia 4 de novembro de 2025, com o sistema operando sem controle de potência ativo.

Fonte: Autoria própria (2026).

Observa-se que, apesar da elevada geração solar ao longo do período diurno (representada pelas colunas laranja no gráfico inferior), o consumo das cargas locais manteve-se relativamente constante e significativamente inferior à potência gerada durante os horários de pico de irradiação.

Como consequência direta desse desbalanceamento entre geração e consumo, ocorreu a injeção de energia excedente na rede elétrica, evidenciada pelas colunas roxas nos gráficos de uso de energia. Esse comportamento é característico de sistemas fotovoltaicos *on-grid* convencionais, nos quais a geração não é adaptada dinamicamente ao perfil de carga local.

A análise quantitativa dos indicadores apresentados no painel demonstra que apenas **19% da energia solar gerada foi autoconsumida**, enquanto aproximadamente **10,06 kWh foram exportados para a rede elétrica**. Esse resultado evidencia a baixa taxa de aproveitamento local da energia produzida e reforça a motivação para a implementação de estratégias de controle ativo de potência, visando maximizar o autoconsumo e reduzir a dependência da rede.

8.3 Sistema com Controle Ativo

A operação da microrrede com o controle de potência ativo foi analisada em quatro dias distintos: 24/10/2025, 31/10/2025, 18/12/2025 e 19/12/2025. Esses dias foram selecionados por apresentarem diferentes perfis de carga e condições de geração solar, permitindo avaliar o desempenho do controle em cenários variados de operação.

Em todos os cenários, observou-se a atuação efetiva do sistema na limitação da potência de saída do inversor Fronius, ajustando-se dinamicamente conforme o consumo instantâneo das cargas locais e direcionando a geração fotovoltaica prioritariamente para o autoconsumo.

8.3.1 Análise de Desempenho e Autoconsumo

No dia **31 de outubro de 2025**, caracterizado por um perfil de carga relativamente estável, o sistema atingiu um índice de **97% de energia solar autoconsumida**, conforme indicado no painel lateral do dashboard de energia. Os resultados deste dia são apresentados nas Figuras 33 e 34.

Figura 33 – Dashboard do Grafana em 31 de outubro de 2025 com controle ativo.

Fonte: Autoria própria (2025).

Figura 34 – Fluxo de energia em 31 de outubro de 2025 com controle ativo e carga estável.

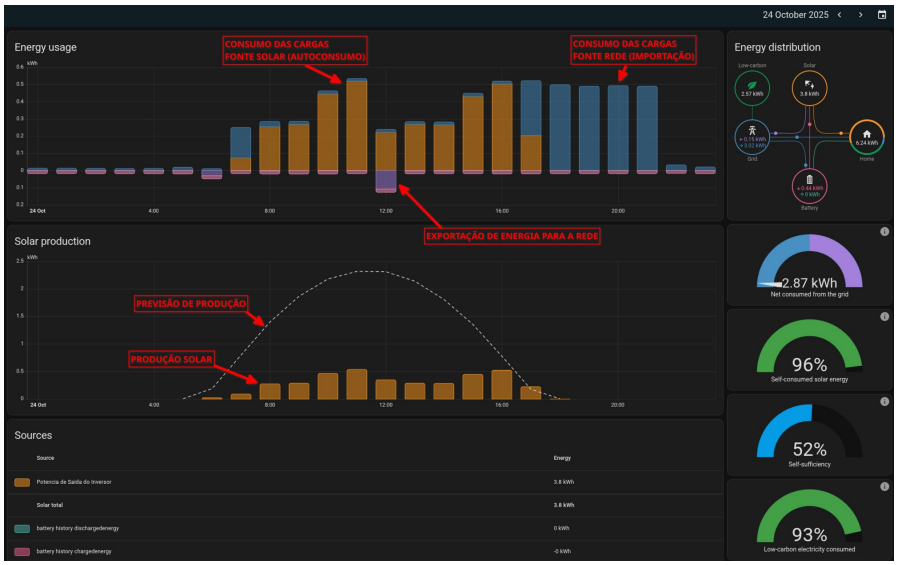
Fonte: Autoria própria (2025).

Já no dia **24 de outubro de 2025**, o sistema foi submetido a variações mais acentuadas de carga ao longo do dia. Mesmo sob essas condições dinâmicas, o controle manteve desempenho consistente, resultando em **96% de autoconsumo da energia solar gerada**. Esse resultado, ilustrado nas Figuras 35 e 36, evidencia a capacidade do sistema de adaptar o limite de potência do inversor às flutuações de demanda.

Figura 35 – Dashboard do Grafana em 24 de outubro de 2025 com controle ativo.

Fonte: Autoria própria (2025).

Figura 36 – Fluxo de energia em 24 de outubro de 2025 com controle ativo e variação de carga.

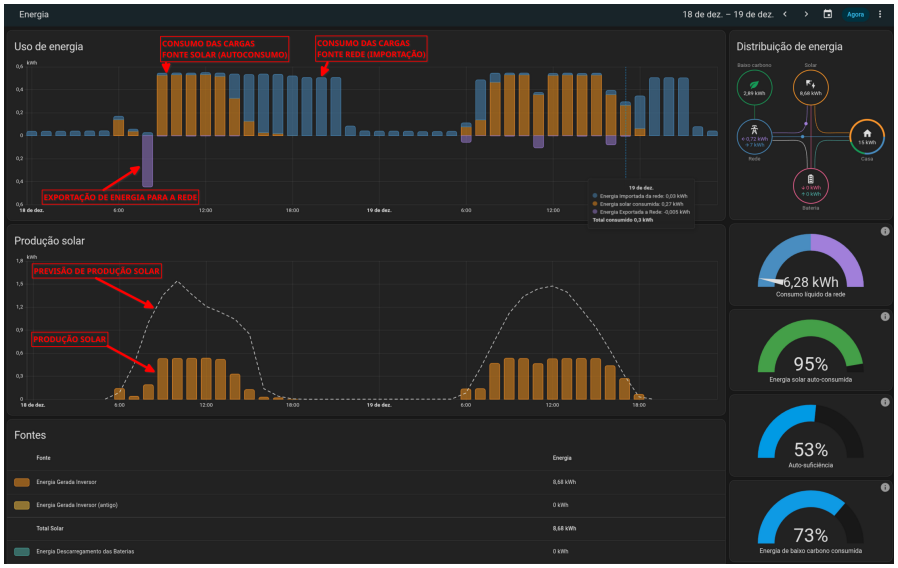


Fonte: Autoria própria (2025).

Para validar a estabilidade do controle em um período prolongado, analisou-se a operação contínua durante os dias **18 e 19 de dezembro de 2025** (Figura 37). Nesses dias, o sistema manteve um elevado aproveitamento da geração local, com **95% de**

energia solar autoconsumida, indicando que a quase totalidade da energia produzida foi utilizada diretamente pelas cargas, com exportações mínimas para a rede elétrica.

Figura 37 – Fluxo de energia estável durante os dias 18 e 19 de dezembro de 2025 no dashboard de energia do Home Assistant.



Fonte: Autoria própria (2025).

8.3.2 Dinâmica de Controle e Regime Transitório

Embora o desempenho global apresente altos índices de autoconsumo, observam-se pequenas quantidades de energia exportada nos gráficos (visíveis como barras roxas residuais). Esses eventos correspondem a regimes transitórios de curta duração, associados à resposta dinâmica do sistema durante mudanças abruptas de carga.

A análise detalhada dos gráficos de fluxo de energia permite compreender esse comportamento. Em eventos de desligamento repentino de cargas — da ordem de aproximadamente 550 W — verifica-se uma redução imediata no consumo, enquanto a potência de saída do inversor Fronius permanece momentaneamente no valor anterior. Isso gera um excedente temporário que é injetado na rede.

Esse fenômeno, caracterizado como um *overshoot* transitório típico de sistemas digitais, ocorre devido à natureza discreta da malha de controle, cujo ciclo de atualização

é executado a cada 15 s no *Home Assistant*. Nesse intervalo entre a detecção da variação de carga e a aplicação do novo comando Modbus de limitação, ocorre a exportação residual.

Após a execução do próximo ciclo de controle, o valor de limitação é atualizado e o sistema restabelece o equilíbrio entre geração e consumo, eliminando a exportação. Portanto, tais eventos não decorrem de falhas no algoritmo, mas sim de limitações inerentes ao tempo de amostragem e comunicação da arquitetura proposta. Conclui-se que o controle apresenta tempo de resposta adequado e estabilidade satisfatória para operação em tempo quase real.

8.4 Teste com Gerenciamento de Baterias do Victron

Acessando diretamente o endereço IP do inversor Victron por meio de um navegador de internet, é possível realizar diversas configurações do sistema. A maior parte dos testes foi conduzida com o sistema de baterias operando no modo `KEEP_BATTERIES_CHARGED`, no qual o inversor mantém o banco de baterias carregado, utilizando-o apenas em situações de falta de energia da concessionária.

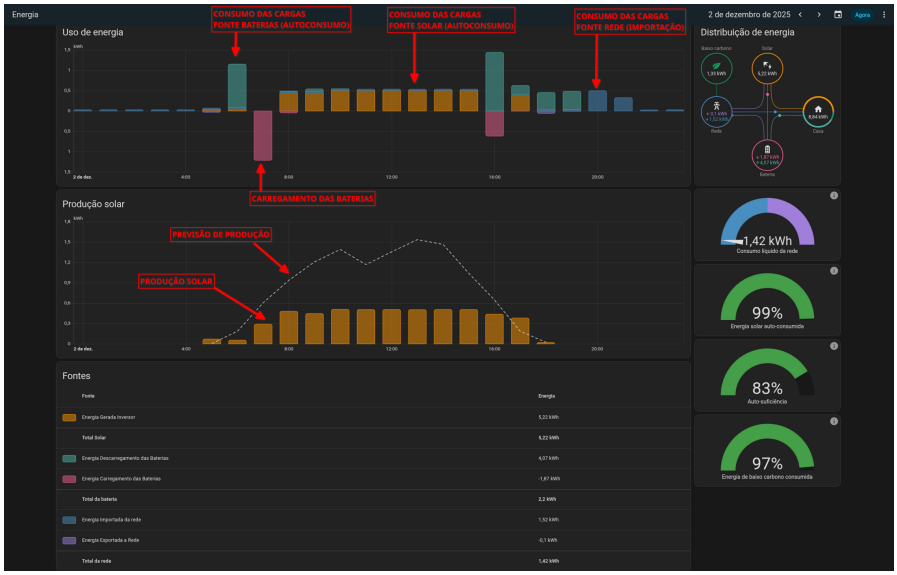
Adicionalmente, foi realizado um teste com o sistema de baterias configurado no modo `SELF_CONSUMPTION`. Nesse modo de operação, o inversor utiliza o banco de baterias para maximizar a autossuficiência do sistema, ou seja, reduzir ao máximo o consumo de energia da concessionária. Para isso, as baterias passam a suprir as cargas quando não há produção solar e são recarregadas quando existe excedente de geração fotovoltaica ou quando o estado de carga atinge o nível mínimo configurado, neste caso, 50%.

Na Figura 38 é apresentado o funcionamento do sistema no dia 02/12/2025. O gráfico segue o esquema de cores definido na Seção 8.1, com a adição das colunas em verde, que indicam a energia consumida a partir das baterias, e em roxo, que representam a energia utilizada para o carregamento do banco de baterias. Nesse dia, o sistema operou com a estratégia de *Grid Zero* implementada e com o gerenciamento de baterias configurado em modo de consumo próprio.

O resultado observado foi um nível de autoconsumo superior ao obtido nos

demais testes, atingindo 99%, com o banco de baterias sendo utilizado para suprir as cargas do sistema até por volta das 19:00. Contudo, o sistema de gerenciamento de baterias do inversor Victron não reconhece que a limitação de potência do sistema fotovoltaico é imposta externamente pela estratégia de controle implementada. Como consequência, o inversor aguarda momentos de produção solar excedente para realizar o carregamento das baterias. Quando esse excedente não ocorre, torna-se necessária a importação de energia da concessionária para atender às cargas e recarregar o banco de baterias.

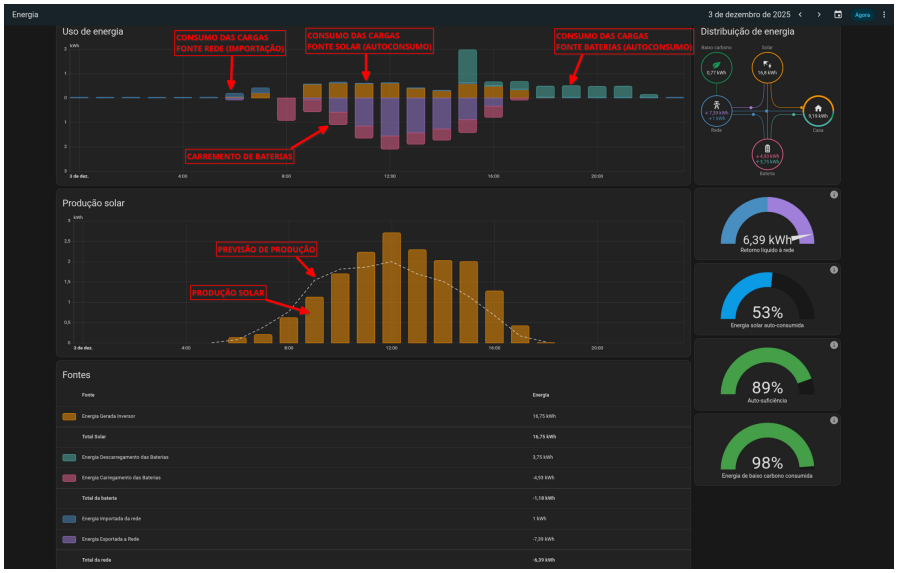
Figura 38 – Dashboard de energia do Home Assistant operando com a estratégia *Grid Zero* e o sistema de baterias configurado em modo de consumo próprio.



Fonte: Autoria própria (2026).

Na Figura 39 é ilustrado o funcionamento do sistema no dia seguinte, 03/12/2025. Nessa data, o sistema operou sem a limitação de potência imposta ao inversor fotovoltaico, permitindo que o excedente de produção solar fosse utilizado para o carregamento do banco de baterias.

Figura 39 – Dashboard de energia do Home Assistant operando com produção solar sem limitação de potência e o sistema de baterias em modo de consumo próprio.



Fonte: Autoria própria (2026).

Os resultados obtidos indicam que a estratégia de limitação de potência não apresentou comportamento adequado quando combinada com o gerenciamento de baterias nativo do inversor Victron. Isso ocorre porque o inversor não distingue se a redução da potência disponível é causada por baixa irradiância solar ou por uma limitação artificial imposta externamente pelo sistema de controle.

Esse problema pode ser mitigado por diferentes abordagens. O inversor Victron permite a utilização de um controlador externo para o gerenciamento do banco de baterias, o que possibilitaria o desenvolvimento de um gerenciador dedicado para integrar de forma coordenada os dois sistemas. Alternativamente, pode-se adotar uma margem de produção solar superior ao consumo instantâneo das cargas. Por exemplo, ao detectar que o estado de carga das baterias está abaixo de 80%, o sistema poderia elevar temporariamente o alvo de produção fotovoltaica para um valor acima do consumo, destinando essa diferença ao carregamento do banco de baterias.

Após a conclusão dos testes, o inversor Victron foi retornado ao modo de operação `KEEP_BATTERIES_CHARGED`, uma vez que, no escopo deste trabalho, sua principal função

é atuar como monitor das cargas e do fluxo de potência do sistema.

8.5 Teste em Limite de potência Fixo

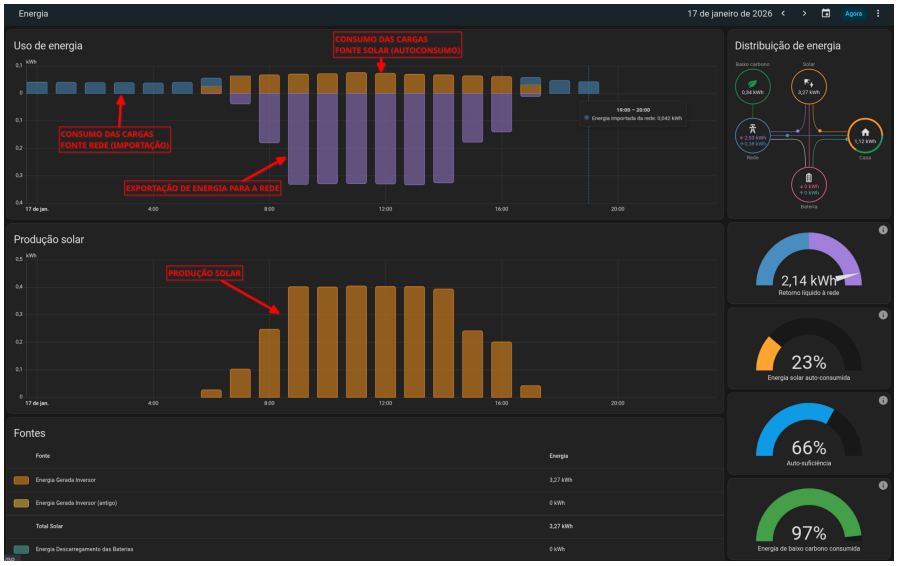
Além do modo dinâmico *Grid Zero*, o sistema de controle desenvolvido no *Home Assistant*, descrito na Seção 7.6, permite a operação do inversor fotovoltaico em regime de **Limite Fixo de Potência**. Nessa configuração, a potência ativa de saída do inversor é restringida a um valor constante definido manualmente pelo operador, independentemente do consumo instantâneo das cargas.

A seleção deste modo de operação é realizada por meio da IHM, utilizando um seletor de estados (*input_select*) para ativar a opção *Limite Fixo*. O valor da potência máxima permitida é definido por meio de um controle numérico deslizante (*slider*), expresso diretamente em watts, conforme ilustrado no canto superior esquerdo da Figura 30. Internamente, esse valor é convertido para percentual da potência nominal do inversor, de modo a atender aos requisitos do protocolo de controle do equipamento.

Uma vez configurado, o sistema mantém o limite de potência de saída constante, desde que haja disponibilidade de irradiância solar suficiente para atender ao valor estabelecido. Caso contrário, a potência gerada passa a ser limitada exclusivamente pelas condições ambientais, seguindo a curva natural de geração fotovoltaica.

Nas Figuras 40 e 41 são apresentados os resultados experimentais obtidos no dia 17/01/2026. Inicialmente, foi configurado um limite de potência de 400 W. Durante o período da manhã, observa-se que a geração acompanha a curva crescente de irradiância solar até atingir esse valor, instante a partir do qual ocorre o ceifamento (*clipping*) da potência, mantendo a saída do inversor estabilizada no patamar definido.

Figura 40 – Painel de supervisão do Home Assistant mostrando a operação do sistema fotovoltaico em modo de limite fixo de potência no dia 17/01/2026.



Fonte: Autoria própria (2026).

Figura 41 – Visualização temporal da potência gerada pelo sistema fotovoltaico em modo de limite fixo, obtida por meio do Grafana, no dia 17/01/2026.

Fonte: Autoria própria (2026).

Às 15:13, foi realizado um ajuste manual via interface, reduzindo o limite de potência para 200 W. O sistema respondeu reconfigurando o controle e estabilizando a potência de saída nesse novo valor. Esse regime permaneceu estável até 16:58, quando a irradiância solar disponível tornou-se insuficiente para sustentar a geração de 200 W,

fazendo com que a potência produzida passasse novamente a seguir a curva natural de disponibilidade solar.

Os resultados obtidos demonstram que o sistema de controle operou corretamente no modo de Limite Fixo, respeitando o valor configurado pelo operador sempre que as condições de geração permitiram, e transitando de forma suave para o regime limitado por irradiância quando necessário. Esse comportamento confirma a correta implementação da lógica de controle e a adequada integração entre a interface de supervisão, o *Node-RED* e o inversor fotovoltaico.

8.6 Teste utilizando a Carga Auxiliar

A carga auxiliar foi empregada como ferramenta experimental para avaliar a resposta dinâmica do sistema de controle quando submetido a variações abruptas de demanda operando no modo *Grid Zero*. Esses testes permitem caracterizar os efeitos das latências de medição, processamento e atuação do controle, que resultam em breves desequilíbrios energéticos durante eventos de comutação de carga.

Na Figura 42, obtida por meio da plataforma *Grafana*, é apresentado a resposta global do sistema aos eventos de ligação e desligamento da carga auxiliar, evidenciando os transitórios de potência associados.

Figura 42 – Resposta do sistema à comutação da carga auxiliar em modo *Grid Zero*.

Fonte: Autoria própria (2026).

8.6.1 Resposta à Ligação da Carga Auxiliar

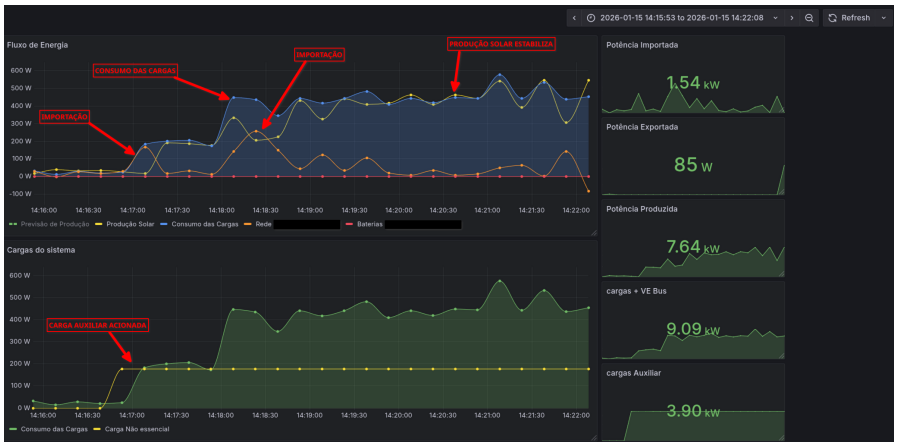
A resposta do sistema ao acionamento da carga auxiliar é detalhada na Figura 43 e na Tabela 2. Observa-se que, no instante 14:17:03, a carga passa a consumir aproximadamente 177 W. Contudo, esse acréscimo só é refletido no consumo total das cargas no registro seguinte, às 14:17:18, quando a variável totalizadora salta de 24 W para 184 W.

Esse atraso de 15 s decorre da taxa de amostragem do banco de dados *Victoria-Metrics* e do atraso de 3 s presente no ciclo de leitura criado no *Node-RED*, configurado para registrar os estados dos sensores do *Home Assistant* nesse intervalo. Nesse momento, a produção solar ainda se mantém reduzida (19 W), exigindo a importação de 167 W da rede elétrica para suprir a demanda.

A atuação efetiva do controle ocorre às 14:17:33, cerca de 30 s após o aciona-

mento da carga, quando a produção solar é elevada para 191 W. Como consequência, a importação da rede é reduzida para 17 W, restabelecendo o equilíbrio energético do sistema.

Figura 43 – Resposta do sistema ao acionamento da carga auxiliar.



Fonte: Autoria própria (2026).

Tabela 2 – Dados registrados durante o acionamento da carga auxiliar.

Horário	Consumo Cargas	Carga Não Essencial	Previsão Produção	Produção Solar	Rede (Import.)	Baterias (Carreg.)
14:16:18	14 W	0 W	1.410 kW	40.0 W	-02.0 W	0.0 W
14:16:33	29 W	0 W	1.410 kW	32.0 W	27.0 W	0.0 W
14:16:48	21 W	0 W	1.410 kW	35.0 W	17.0 W	0.0 W
14:17:03	24 W	177 W	1.410 kW	28.0 W	29.0 W	0.0 W
14:17:18	184 W	178 W	1.410 kW	19.0 W	167.0 W	0.0 W
14:17:33	201 W	178 W	1.410 kW	191.0 W	17.0 W	0.0 W
14:17:48	207 W	177 W	1.410 kW	188.0 W	33.0 W	0.0 W
14:18:03	174 W	177 W	1.410 kW	178.0 W	12.0 W	0.0 W

Fonte: Autoria própria (2026).

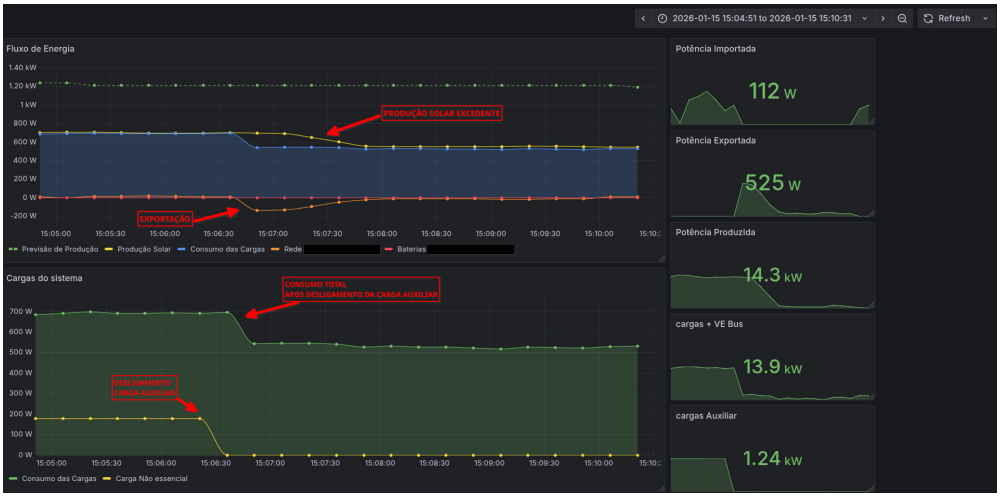
8.6.2 Resposta ao Desligamento da Carga Auxiliar

O comportamento inverso é observado durante o desligamento da carga auxiliar, conforme apresentado na Figura 44 e na Tabela 3. Nesse caso, evidencia-se a ocorrência de um *overshoot* temporário de geração.

No instante 15:06:26, a carga auxiliar é desligada, reduzindo seu consumo a 0 W. Entretanto, devido à latência de leitura, o consumo total ainda é registrado como elevado (696 W), mantendo a produção solar próxima de 700 W. A redução efetiva do consumo é detectada apenas às 15:06:41, quando a demanda total cai para 543 W, enquanto a produção permanece elevada, resultando em exportação momentânea de energia para a rede, registrada em -137 W.

A resposta do controlador inicia-se às 15:07:11, aproximadamente 45 s após o desligamento físico da carga, promovendo uma redução progressiva da potência gerada. O sistema atinge novamente uma condição próxima ao ponto de operação *Grid Zero* às 15:07:56, cerca de 90 s após o desligamento da carga, quando a produção solar (552 W) se aproxima novamente da demanda das cargas (532 W), reduzindo a troca com a rede elétrica a valores desprezíveis.

Figura 44 – Resposta do sistema ao desligamento da carga auxiliar.



Fonte: Autoria própria (2026).

Os resultados obtidos indicam que, embora o sistema seja capaz de perseguir o ponto de operação *Grid Zero*, ocorrem janelas temporais da ordem de 30 a 45 s nas quais há importação ou exportação transitória de energia. Esse comportamento decorre da soma dos tempos de ciclo do *Node-RED* (3 s), da automação do *Home Assistant* (15 s) e da taxa de amostragem do sistema de armazenamento de dados.

A exportação momentânea de energia pode ser validada no painel de energia do *Home Assistant*, conforme ilustrado na Figura 45. As barras na cor roxa indicam os intervalos de injeção de energia na rede, coincidindo com os eventos de comutação da carga auxiliar.

Figura 45 – Painel de energia do Home Assistant evidenciando exportações momentâneas durante a comutação da carga auxiliar (15/01/2026).

Fonte: Autoria própria (2026).

A Figura 46 apresenta a distribuição consolidada de consumo energético do dia 15/01/2026, permitindo identificar a parcela de energia atribuída especificamente à carga auxiliar em relação ao consumo total da microrrede.

Figura 46 – Distribuição do consumo de energia por fonte e por dispositivo no dia 15/01/2026.

Fonte: Autoria própria (2026).

Tabela 3 – Dados registrados durante o desligamento da carga auxiliar.

Horário	Consumo Cargas	Carga Não Essencial	Previsão Produção	Produção Solar	Rede (Import.)	Baterias (Carreg.)
15:04:56	693.0 W	177.0 W	1.240 kW	710.0 W	1.0 W	0.0 W
15:05:11	699.0 W	177.0 W	1.210 kW	706.0 W	14.0 W	0.0 W
15:05:26	693.0 W	177.0 W	1.210 kW	700.0 W	16.0 W	0.0 W
15:05:41	693.0 W	178.0 W	1.210 kW	695.0 W	19.0 W	0.0 W
15:05:56	694.0 W	178.0 W	1.210 kW	698.0 W	14.0 W	0.0 W
15:06:11	693.0 W	178.0 W	1.210 kW	698.0 W	8.0 W	0.0 W
15:06:26	696.0 W	0.0 W	1.210 kW	700.0 W	11.0 W	0.0 W
15:06:41	543.0 W	0.0 W	1.210 kW	698.0 W	-137.0 W	0.0 W
15:06:56	547.0 W	0.0 W	1.210 kW	694.0 W	-130.0 W	0.0 W
15:07:11	545.0 W	0.0 W	1.210 kW	650.0 W	-94.0 W	0.0 W
15:07:26	542.0 W	0.0 W	1.210 kW	602.0 W	-48.0 W	0.0 W
15:07:41	527.0 W	0.0 W	1.210 kW	556.0 W	-19.0 W	0.0 W
15:07:56	532.0 W	0.0 W	1.210 kW	552.0 W	-11.0 W	0.0 W
15:08:11	528.0 W	0.0 W	1.210 kW	550.0 W	-13.0 W	0.0 W
15:08:26	526.0 W	0.0 W	1.210 kW	550.0 W	-10.0 W	0.0 W
15:08:41	523.0 W	0.0 W	1.210 kW	550.0 W	-10.0 W	0.0 W
15:08:56	518.0 W	0.0 W	1.210 kW	550.0 W	-16.0 W	0.0 W
15:09:11	528.0 W	0.0 W	1.210 kW	558.0 W	-15.0 W	0.0 W

Fonte: Autoria própria (2026).

8.7 Síntese dos Resultados

Na Tabela 4 é apresentado um resumo comparativo dos principais indicadores obtidos ao longo dos testes realizados, contemplando diferentes estratégias de controle de potência e condições de operação da microrrede.

Tabela 4 – Resumo comparativo dos indicadores de desempenho da microrrede.

Data	Modo	AC (%)	AS (%)	Saldo rede* (kWh)	Comentário	Figura
21/10/2025	Grid Zero	97	55	−2,63	Teste painel de energia	Fig. 28
24/10/2025	Grid Zero	96	52	−2,87	Teste Grid Zero	Fig. 36
31/10/2025	Grid Zero	97	60	−3,02	Teste Grid Zero	Fig. 34
04/11/2025	Normal	19	50	10,06	Teste sem controle de potência	Fig. 31
02/12/2025	Grid Zero	99	83	−1,42	Teste com sistema de baterias	Fig. 38
03/12/2025	Normal	53	89	6,39	Teste com sistema de baterias	Fig. 39
18/12/2025	Grid Zero	94	42	−3,75	Teste Grid Zero	Fig. 37
19/12/2025	Grid Zero	95	64	−2,52	Teste Grid Zero	Fig. 37
15/01/2026	Grid Zero	96	70	−0,64	Teste com carga auxiliar	Fig. 46
17/01/2026	Limite Fixo	23	66	2,14	Teste com limite fixo de potência	Fig. 40

*Convenção adotada: valores positivos indicam exportação líquida de energia para a rede elétrica; valores negativos indicam importação líquida de energia da rede.
Siglas utilizadas: AC - Autoconsumo; AS - Autossuficiência.
Fonte: Autoria própria (2026).

Os resultados sintetizados na Tabela 4 permitem uma avaliação abrangente do desempenho do sistema sob diferentes estratégias de controle de potência e condições operacionais. De forma consistente, observa-se que a ativação do controle do tipo *Grid Zero* promove um aumento expressivo do índice de autoconsumo, definido como o percentual da geração solar aproveitado instantaneamente pelas cargas locais, quando comparado à operação convencional sem limitação de potência.

Nos cenários em que o sistema operou em modo *Normal*, com destaque para o dia 04/11/2025, verificou-se baixo aproveitamento da geração fotovoltaica, com apenas 19 % de autoconsumo e elevada exportação líquida de energia para a rede elétrica (10,06 kWh). Esse comportamento é característico de sistemas fotovoltaicos *on-grid* desprovidos de estratégias de gerenciamento ativo. Em contraste, nos testes realizados com o controle *Grid Zero* habilitado, os índices de autoconsumo mantiveram-se predominantemente acima de 94 %, atingindo valores de até 99 % em condições favoráveis de carga e geração.

A análise do saldo energético com a rede evidencia que o controle implementado foi eficaz na redução da exportação líquida de energia, mantendo o balanço energético próximo de zero ao longo do período de operação. As pequenas parcelas residuais de exportação ou importação observadas decorrem, principalmente, da natureza discreta da malha de controle, associada ao período de atualização da automação (15 s), ao ciclo de

leitura do *Node-RED* (3 s), às latências de comunicação e ao tempo de resposta dinâmica do inversor para o ajuste do *setpoint* de potência.

Os testes envolvendo o sistema de baterias do inversor Victron evidenciaram impacto significativo na autossuficiência do sistema, entendida como a fração do consumo total suprida por energia solar. No dia 02/12/2025, a autossuficiência atingiu 83 %, o maior valor registrado sob controle *Grid Zero*. Contudo, observou-se que a limitação artificial da potência pode restringir o carregamento do banco de baterias, uma vez que o inversor Victron não distingue a limitação imposta pelo controle externo de uma redução efetiva na irradiância solar.

Adicionalmente, os ensaios com carga auxiliar e com o modo de *Limite Fixo* demonstraram a flexibilidade da arquitetura proposta, possibilitando a operação do sistema em diferentes regimes de controle, seja para maximização do autoconsumo, limitação manual da potência ou imposição de saída nula para fins experimentais e de segurança.

De modo geral, os resultados obtidos validam o objetivo central deste trabalho: o desenvolvimento de uma solução aberta, de baixo custo e baseada em ferramentas de automação livre, capaz de reproduzir de forma eficaz o comportamento de um sistema *Grid Zero*. A arquitetura proposta mostrou-se funcional, estável e adaptável, configurando-se como uma alternativa viável para aplicações experimentais, residenciais e acadêmicas em microrredes fotovoltaicas.

9 Conclusões e Trabalhos Futuros

Este capítulo consolida os resultados obtidos ao longo do desenvolvimento do trabalho, avalia o cumprimento dos objetivos propostos e aponta possíveis caminhos para a evolução do sistema implementado.

9.1 Conclusões

A motivação deste trabalho surgiu da necessidade prática de implementar um controle ativo de potência em um inversor fotovoltaico conectado à rede (*on-grid*), capaz de restringir a exportação de energia (*Grid Zero*) utilizando uma infraestrutura de comunicação acessível e baseada em software livre.

A análise inicial apresentada na Seção 8.2 confirmou que a operação padrão do inversor é ineficiente em cenários nos quais há restrição de injeção de energia na rede elétrica. Sem a atuação de um controle ativo, o sistema apresentou um índice de autoconsumo de apenas 19 %, direcionando a maior parte da geração solar para a rede elétrica, comportamento típico de sistemas fotovoltaicos *on-grid* convencionais.

Para solucionar esse problema, foi desenvolvida e validada uma arquitetura de controle em malha fechada baseada integralmente em ferramentas de código aberto. A orquestração via `Docker` garantiu a integração estável entre os diferentes componentes do sistema, incluindo o `Node-RED`, responsável pela tradução e tratamento dos protocolos Modbus e MQTT, o banco de dados temporal *VictoriaMetrics* e o *Home Assistant*, que atuou como controlador central, concentrando a lógica de controle e a interface homem-máquina (IHM).

Os resultados experimentais validaram a robustez e a eficácia da solução proposta. A ativação do controle elevou o índice de autoconsumo para patamares superiores a 94 %, mantendo o saldo energético com a rede elétrica próximo de zero ao longo do período de operação. A análise da resposta dinâmica demonstrou que as pequenas exportações residuais observadas (*overshoots*) são consequências físicas inevitáveis da soma das latências do sistema, incluindo o ciclo de atualização da automação (15 s),

o processamento no `Node-RED`, as latências de comunicação e o tempo de rampa do próprio inversor para ajuste do *setpoint* de potência. Mesmo diante dessas limitações inerentes a sistemas de controle discretos, o sistema cumpriu adequadamente sua função de evitar a injeção contínua de energia na rede.

Conclui-se, portanto, que os objetivos propostos foram atingidos. O projeto demonstrou ser uma alternativa viável, de baixo custo e agnóstica a fabricantes, capaz de transformar um inversor fotovoltaico convencional em um sistema inteligente de limitação de potência. A solução não apenas resolveu o problema de exportação de energia no IFG Campus Itumbiara, como também evidenciou o potencial de integração entre protocolos industriais consolidados, como o Modbus, e plataformas modernas de IoT, como o *Home Assistant*, para a implementação de microrredes fotovoltaicas eficientes.

9.2 Trabalhos Futuros

Embora o sistema atual forneça uma base sólida e funcional, sua lógica de atuação ainda é predominantemente baseada na restrição de potência (*curtailment*). Para evoluir de um limitador de potência para um Sistema de Gerenciamento de Energia (EMS) mais abrangente, são propostas as seguintes extensões.

9.2.1 Coordenação Ativa com Banco de Baterias

Atualmente, o controle atua reduzindo a potência de saída do inversor Fronius quando há excedente de geração, comportamento que o inversor Victron — responsável pelo gerenciamento das baterias — interpreta como uma condição de baixa irradiância solar. Como consequência, o sistema de armazenamento permanece subutilizado em determinados cenários.

Como evolução natural, propõe-se o estabelecimento de uma comunicação direta entre a malha de controle e o carregador de baterias. A lógica de operação deve ser invertida: antes de limitar a potência do inversor fotovoltaico, o sistema deve priorizar o aumento da corrente de carga do banco de baterias. A estratégia de *Grid Zero* passaria a atuar apenas como último recurso, sendo acionada quando o banco de baterias

atingir níveis elevados de estado de carga (por exemplo, acima de 90 %), aumentando a autossuficiência da microrrede como etapa evolutiva do sistema.

9.2.2 Gerenciamento de Cargas de Desvio

Como forma de evitar o desperdício de energia renovável em situações nas quais o banco de baterias encontra-se totalmente carregado, propõe-se a automação de cargas controláveis do tipo resistivo ou indutivo, tais como aquecedores de água, bombas elétricas ou carregadores de veículos elétricos. Essas cargas podem atuar como *dump loads*, absorvendo o excedente de geração fotovoltaica que, de outra forma, seria limitado pelo inversor.

A partir das medições em tempo real disponibilizadas pelo *Home Assistant*, é possível implementar gatilhos de controle que acionem essas cargas de forma proporcional ao excedente solar disponível. Dessa maneira, a energia que seria bloqueada pela estratégia de limitação de potência pode ser convertida em trabalho útil, aumentando a eficiência global do sistema e o aproveitamento da geração local.

Os testes realizados com a carga auxiliar constituída por uma lâmpada halógena demonstram a viabilidade prática dessa abordagem. Embora se trate de uma carga simples e de baixa potência, o experimento comprova que é possível integrar, acionar e supervisionar cargas não essenciais de forma coordenada com o sistema de controle, abrindo caminho para estratégias mais avançadas de gerenciamento de cargas de desvio.

9.2.3 Refinamento da Malha de Controle

O intervalo de atualização de 15 s adotado na malha de controle mostrou-se adequado para o balanço energético em regime permanente, porém relativamente lento para a resposta a transientes rápidos de carga. Essa limitação torna-se evidente em eventos abruptos de variação de consumo, nos quais podem ocorrer exportações residuais de curta duração antes da atualização do comando de limitação de potência.

Uma possibilidade de aprimoramento consiste na migração da lógica de controle para um script dedicado, desenvolvido em linguagens como Python ou C++, executado como um serviço no próprio *Home Assistant* ou de forma externa, comunicando-se

via MQTT. Essa abordagem permitiria reduzir o ciclo de leitura e escrita do controle, aumentando a responsividade do sistema frente a variações dinâmicas de carga.

Além do ganho em desempenho dinâmico, essa migração contribuiria significativamente para a replicabilidade da solução. Na implementação atual, a reprodução do sistema requer a configuração manual de fluxos específicos no *Node-RED*, bem como a criação de automações e *helpers* no *Home Assistant*. A centralização da lógica em um serviço dedicado reduziria essa dependência de configurações distribuídas.

Como evolução final dessa proposta, a criação de uma integração nativa para o *Home Assistant* permitiria encapsular toda a lógica de aquisição, controle e escrita de comandos em um único componente instalável. Dessa forma, a replicação do sistema por outros usuários seria significativamente simplificada, exigindo apenas a instalação da integração, sem a necessidade de configuração manual de fluxos ou automações, aumentando a portabilidade e a escalabilidade da solução.

Apesar das vantagens operacionais dessas abordagens, sua implementação não foi realizada neste trabalho em razão das limitações de prazo e da complexidade envolvida no desenvolvimento, testes e validação de um *software* nativo estável.

A utilização do *Node-RED* mostrou-se adequada para viabilizar a prova de conceito e validar a arquitetura proposta dentro do cronograma estabelecido, permitindo demonstrar o funcionamento do sistema sem comprometer a robustez da solução. O desenvolvimento de uma aplicação nativa, com maior nível de otimização e encapsulamento, permanece como possibilidade para trabalhos futuros.

9.3 Considerações Finais

Este trabalho reafirma que soluções eficazes em Engenharia de Controle e Automação não dependem exclusivamente de hardware proprietário ou de alto custo. A combinação de software livre, protocolos padronizados e hardware amplamente disponível possibilitou a criação de uma microrrede fotovoltaica funcional, estável e replicável, servindo como referência para futuras aplicações residenciais, experimentais e acadêmicas voltadas à eficiência energética.

Referências

ABEDINPOUR, P. *MariaDB vs MySQL vs PostgreSQL vs SQLite: A Comprehensive Comparison for Web Applications*. 2024. Disponível em: <https://medium.com/@peymaan.abedinpour/mariadb-vs-mysql-vs-postgresql-vs-sqlite-a-comprehensive-comparison-for-web-applications-0523cc3bc9d8>. Acesso em: 23 out. 2025.

ANEEL. *Micro e minigeração distribuída de energia elétrica cresceu 8,85 GW em 2024*. 2025. Disponível em: <https://www.gov.br/aneel/pt-br/assuntos/noticias/2025/micro-e-minigeracao-distribuida-de-energia-eletrica-cresceu-8-84-gw-em-2024>. Acesso em: 10 nov. 2025.

BRANDS, H.; KITCHEN, K. *Tuya may be the China threat that beats Russia's ransomware attacks*. 2021. Disponível em: <https://thehill.com/opinion/cybersecurity/564962-tuya-may-be-the-china-threat-that-beats-russias-ransomware-attacks/>. Acesso em: 11 jan. 2025.

DAVIDSON, A. *What is Home Assistant, how does it work, and what do you need to get started?* 2023. Disponível em: <https://www.pocket-lint.com/what-is-home-assistant-how-does-it-work/>. Acesso em: 12 jan. 2026.

DIAZ, M. *The best home automation systems: How Google Home, Amazon Alexa, and more compare*. 2025. Disponível em: <https://www.zdnet.com/home-and-office/smart-home/best-home-automation-system/>. Acesso em: 12 jan. 2026.

Docker Inc. *What is Docker?* 2026. Disponível em: <https://docs.docker.com/get-started/docker-overview/>. Acesso em: 18 fev. 2026.

FRONIUS. *Operating Instructions: Fronius GEN24 Modbus TCP RTU*. 031. ed. Pettenbach, Áustria, 2024. 85 p. Nota da página 85. Disponível em: <https://www.fronius.com/~/downloads/Solar%20Energy/Operating%20Instructions/42,0410,2649.pdf>. Acesso em: 23 out. 2025.

FRONIUS. *Fronius Inverter Modbus Register Map*. s.d. <https://www.forum-fronius.pl/wp-content/uploads/asgarosforum/2812/Inverter_register_map.pdf>. Acesso em: 21 out. 2025.

HOME ASSISTANT. *Zigbee Home Automation*. 2026. Disponível em: <https://www.home-assistant.io/integrations/zha/>. Acesso em: 12 jan. 2026.

LOIOLA, V. *Grid zero: entenda o que é, como funciona e por que homologar*. 2024. <https://www.portalsolar.com.br/grid-zero>. Acesso em: 10 nov. 2025.

LOPES, K. A. *Sistema IoT para supervisão e controle de inversores com aplicação em microrredes*. Monografia (Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica)) — Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Itumbiara, GO, 2025.

MVADER. *GX Modbus-TCP Manual*. 2025. <https://www.victronenergy.com/live/ccgx:modbustcp_faq>. Acesso em: 28 out. 2025.

VICTRON ENERGY. *AC-coupling and the Factor 1.0 rule*. 2022. Disponível em: <https://www.victronenergy.com/live/ac_coupling:start>. Acesso em: 5 fev. 2026.

APÊNDICE A – Mapeamento de Registradores do Inversor Fronius (SunSpec)

Tabela 5 – Mapeamento dos Registradores do Inversor Fronius baseado na especificação SunSpec.

POS	INICIO	FIM	SIZE	RW	FUNÇÃO MODBUS	NOME	DESCRIÇÃO	TIPO	UNIDADE	FATOR DE ESCALA	FAIXA
0	40070	40070	1	R	0x03	ID	Uniquely identifies this as a SunSpec Inverter Modbus Map; 101: single	uint16			101, 102, 103
1	40071	40071	1	R	0x03	L	Length of inverter model block	uint16	Registers		50
2	40072	40072	1	R	0x03	A	AC Total Current value	uint16	A	A_SF	
3	40073	40073	1	R	0x03	AphA	AC Phase-A Current value	uint16	A	A_SF	
4	40074	40074	1	R	0x03	AphB	AC Phase-B Current value	uint16	A	A_SF	
5	40075	40075	1	R	0x03	AphC	AC Phase-C Current value	uint16	A	A_SF	
6	40076	40076	1	R	0x03	A_SF	AC Current Scale factor	sunssf			
7	40077	40077	1	R	0x03	PPVphAB	AC Voltage Phase-AB value	uint16	V	V_SF	
8	40078	40078	1	R	0x03	PPVphBC	AC Voltage Phase-BC value	uint16	V	V_SF	
9	40079	40079	1	R	0x03	PPVphCA	AC Voltage Phase-CA value	uint16	V	V_SF	
10	40080	40080	1	R	0x03	PhVphA	AC Voltage Phase-A-to-neutral value	uint16	V	V_SF	
11	40081	40081	1	R	0x03	PhVphB	AC Voltage Phase-B-to-neutral value	uint16	V	V_SF	
12	40082	40082	1	R	0x03	PhVphC	AC Voltage Phase-C-to-neutral value	uint16	V	V_SF	
13	40083	40083	1	R	0x03	V_SF	AC Voltage Scale factor	sunssf			
14	40084	40084	1	R	0x03	W	AC Power value	int16	W	W_SF	
15	40085	40085	1	R	0x03	W_SF	AC Power Scale factor	sunssf			
16	40086	40086	1	R	0x03	Hz	AC Frequency value	uint16	Hz	Hz_SF	
17	40087	40087	1	R	0x03	Hz_SF	Scale factor	sunssf			
18	40088	40088	1	R	0x03	VA	Apparent Power	int16	VA	VA_SF	
19	40089	40089	1	R	0x03	VA_SF	Scale factor	sunssf			
20	40090	40090	1	R	0x03	VAr	Reactive Power	int16	VAr	VAr_SF	
21	40091	40091	1	R	0x03	VAr_SF	Scale factor	sunssf			
22	40092	40092	1	R	0x03	PF	Power Factor	int16	%	PF_SF	
23	40093	40093	1	R	0x03	PF_SF	Scale factor	sunssf			
24	40094	40095	2	R	0x03	WH	AC Lifetime Energy production	uint32	Wh	WH_SF	may be imprecise
26	40096	40096	1	R	0x03	WH_SF	AC Lifetime Energy production scale factor	sunssf			
31	40101	40101	1	R	0x03	DCW	DC Power value	int16	W	DCW_SF	Total power of all DC inputs
32	40102	40102	1	R	0x03	DCW_SF	Scale factor	sunssf			
38	40108	40108	1	R	0x03	St	Operating State	enum16	Enumerated	N/A	
39	40109	40109	1	R	0x03	StVnd	Vendor Defined Operating State	enum16	Enumerated	N/A	
40	40110	40111	2	R	0x03	Evt1	Event Flags (bits 0-31)	uint32	Bitfield	N/A	
42	40112	40113	2	R	0x03	Evt2	Event Flags (bits 32-63)	uint32	Bitfield	N/A	
44	40114	40115	2	R	0x03	EvtVnd1	Vendor Defined Event Flags (bits 0-31)	uint32	Bitfield	N/A	
46	40116	40117	2	R	0x03	EvtVnd2	Vendor Defined Event Flags (bits 32-63)	uint32	Bitfield	N/A	
48	40118	40119	2	R	0x03	EvtVnd3	Vendor Defined Event Flags (bits 64-95)	uint32	Bitfield	N/A	
50	40120	40121	2	R	0x03	EvtVnd4	Vendor Defined Event Flags (bits 96-127)	uint32	Bitfield	N/A	
52	40122	40122	1	R	0x03	ID	A well-known value 120. Uniquely identifies this as a SunSpec Namepla	uint16			120
53	40123	40123	1	R	0x03	L	Length of Nameplate Model	uint16	Registers		26
54	40124	40124	1	R	0x03	DERTyp	Type of DER device. Default value is 4 to indicate PV device.	enum16			4
55	40125	40125	1	R	0x03	WRtg	Continuous power output capability of the inverter.	uint16	W	WRtg_SF	
56	40126	40126	1	R	0x03	WRtg_SF	Scale factor	sunssf			0
57	40127	40127	1	R	0x03	VARtg	Continuous Volt-Ampere capability of the inverter.	uint16	VA	VARtg_SF	
58	40128	40128	1	R	0x03	VARtg_SF	Scale factor	sunssf			0
59	40129	40129	1	R	0x03	VArRtgQ1	Continuous VAR capability of the inverter in quadrant 1.	int16	var	VArRtg_SF	
62	40132	40132	1	R	0x03	VArRtgQ4	Continuous VAR capability of the inverter in quadrant 4.	int16	var	VArRtg_SF	
63	40133	40133	1	R	0x03	VArRtg_SF	Scale factor	sunssf			1
64	40134	40134	1	R	0x03	ARtg	Maximum RMS AC current level capability of the inverter.	uint16	A	ARtg_SF	
65	40135	40135	1	R	0x03	ARtg_SF	Scale factor	sunssf			-2
66	40136	40136	1	R	0x03	PFRtgQ1	Minimum power factor capability of the inverter in quadrant 1.	int16	cos()	PFRtg_SF	
69	40139	40139	1	R	0x03	PFRtgQ4	Minimum power factor capability of the inverter in quadrant 4.	int16	cos()	PFRtg_SF	
70	40140	40140	1	R	0x03	PFRtg_SF	Scale factor	sunssf			-3

71	40141	40141	1 R	0x03	WHRtg	Nominal energy rating of storage device.		uint16	Wh	WHRtg_SF			
72	40142	40142	1 R	0x03	WHRtg_SF	Scale factor.		sunssf					0
75	40145	40145	1 R	0x03	MaxChaRte	Maximum rate of energy transfer into the storage device.		uint16	W	MaxChaRte_SF			
76	40146	40146	1 R	0x03	MaxChaRte_SF	Scale factor.		sunssf					0
77	40147	40147	1 R	0x03	MaxDisChaRte	Maximum rate of energy transfer out of the storage device.		uint16	W	MaxDisChaRte_SF			
78	40148	40148	1 R	0x03	MaxDisChaRte_SF	Scale factor.		sunssf					0
79	40149	40149	1 R	0x03	Pad	Pad register.							
80	40150	40150	1 R	0x03	ID	A well-known value 121. Uniquely identifies this as a SunSpec Basic Set uint16							121
81	40151	40151	1 R	0x03	L	Length of Basic Settings Model		uint16	Registers				30
82	40152	40152	1 R	0x03 0x06 0x10	WMax	Setting for maximum power output. Default to I_WRtg.		uint16	W	Wmax_SF			
83	40153	40153	1 R	0x03	VRef	Voltage at the PCC.		uint16	V	Vref_SF			
84	40154	40154	1 R	0x03 0x06 0x10	VRefOfs	Offset from PCC to inverter.		int16	V	VRefOfs_SF			
87	40157	40157	1 RW	0x03	VAMax	Setpoint for maximum apparent power. Default to I_VARtg.		uint16	VA	VAMax_SF			
88	40158	40158	1 R	0x03	VARMaxQ1	Setting for maximum reactive power in quadrant 1. Default to VArRtgQ1. int16		var		VARMax_SF			
91	40161	40161	1 R	0x03	VARMaxQ4	Setting for maximum reactive power in quadrant 4 Default to VArRtgQ4. int16		var		VARMax_SF			
93	40163	40163	1 R	0x03	PFMinQ1	Setpoint for minimum power factor value in quadrant 1. Default to PFRtg int16		cos()		PFMin_SF			
96	40166	40166	1 R	0x03	PFMinQ4	Setpoint for minimum power factor value in quadrant 4. Default to PFRtg int16		cos()		PFMin_SF			
102	40172	40172	1 R	0x03	WMax_SF	Scale factor for maximum power output.		sunssf					1
103	40173	40173	1 R	0x03	VRef_SF	Scale factor for voltage at the PCC.		sunssf					0
104	40174	40174	1 R	0x03	VRefOfs_SF	Scale factor for offset voltage.		sunssf					0
106	40176	40176	1 R	0x03	VAMax_SF	Scale factor for voltage at the PCC.		sunssf					1
107	40177	40177	1 R	0x03	VARMax_SF	Scale factor for reactive power.		sunssf					1
109	40179	40179	1 R	0x03	PFMin_SF	Scale factor for minimum power factor.		sunssf					-3
112	40182	40182	1 R	0x03	ID	A well-known value 122. Uniquely identifies this as a SunSpec Measure uint16							122
113	40183	40183	1 R	0x03	L	Length of Measurements_Status Model		uint16	Registers				44
114	40184	40184	1 R	0x03	PVConn	PV inverter present/available status. Enumerated value.		bitfield16		Bit 0: ConnectedBit 1: AvailableBit 2: OperatingBit 3: Test			
115	40185	40185	1 R	0x03	StoConn	Storage inverter present/available status. Enumerated value.		bitfield16		bit 0: CONNECTEDbit 1: AVAILABLEbit 2: OPERATINGbit 3: TEST			
116	40186	40186	1 R	0x03	ECPConn	ECP connection status: disconnected=0, connected=1.		bitfield16		0: Disconnected1: Connected			
117	40187	40190	4 R	0x03	ActWh	AC lifetime active (real) energy output.		acc64	Wh				
147	40217	40218	2 R	0x03	StActCtl	Bit Mask indicating which inverter controls are currently active.		bitfield32		Bit 0: FixedWBit 1: FixedVARBit 2: FixedPF			
149	40219	40222	4 R	0x03	TmSrc	Source of time synchronization.		String8		RTC			
153	40223	40224	2 R	0x03	Tms	Seconds since 01-01-2000 00:00 UTC		uint32	Secs				
158	40228	40228	1 R	0x03	ID	A well-known value 123. Uniquely identifies this as a SunSpec Immedia uint16							123
159	40229	40229	1 R	0x03	L	Length of Immediate Controls Model		uint16	Registers				24
160	40230	40230	1 RW	0x03 0x06 0x10	Conn_WinTms	Time window for connect/disconnect.		uint16	Secs				
161	40231	40231	1 RW	0x03 0x06 0x10	Conn_RvrtTms	Timeout period for connect/disconnect.		uint16	Secs				
162	40232	40232	1 RW	0x03 0x06 0x10	Conn	Enumerated value. Connection control.		bitfield16		0: Disconnected1: Connected			
163	40233	40233	1 RW	0x03 0x06 0x10	WMaxLimPct	Set power output to specified level.		uint16	%WMax	WMaxLimPct_SF			
164	40234	40234	1 RW	0x03 0x06 0x10	WMaxLimPct_WinTms	Time window for power limit change.		uint16	Secs				
165	40235	40235	1 RW	0x03 0x06 0x10	WMaxLimPct_RvrtTms	Timeout period for power limit.		uint16	Secs				
166	40236	40236	1 RW	0x03	WMaxLimPct_RmpTms	Ramp time for moving from current setpoint to new setpoint.		uint16	Secs				
167	40237	40237	1 RW	0x03 0x06 0x10	WMaxLim_Ena	Enumerated value. Throttle enable/disable control.		enum16		0: Disabled1: Enabled			
168	40238	40238	1 RW	0x03 0x06 0x10	OutPFSet	Set power factor to specific value - cosine of angle.		int16	cos()	OutPFSet_SF			0.8 to 1.0 and -0.8 to -0.999
169	40239	40239	1 RW	0x03 0x06 0x10	OutPFSet_WinTms	Time window for power factor change.		uint16	Secs				0 - 300
170	40240	40240	1 RW	0x03 0x06 0x10	OutPFSet_RvrtTms	Timeout period for power factor.		uint16	Secs				0 - 28800
171	40241	40241	1 RW	0x03 0x06 0x10	OutPFSet_RmpTms	Ramp time for moving from current setpoint to new setpoint.		uint16	Secs				0 - 65534 (0xFFFF has the same effect as 0x0000)
172	40242	40242	1 RW	0x03 0x06 0x10	OutPFSet_Ena	Enumerated value. Fixed power factor enable/disable control.		enum16		0: Disabled1: Enabled			
174	40244	40244	1 RW	0x03 0x06 0x10	VArMaxPct	Reactive power in percent of I_VArMax.		int16	%VArMax	VArPct_SF			
176	40246	40246	1 RW	0x03 0x06 0x10	VArPct_WinTms	Time window for VAR limit change.		uint16	Secs				0 - 300
177	40247	40247	1 RW	0x03 0x06 0x10	VArPct_RvrtTms	Timeout period for VAR limit.		uint16	Secs				0 - 28800
178	40248	40248	1 RW	0x03 0x06 0x10	VArPct_RmpTms	Ramp time for moving from current setpoint to new setpoint.		uint16	Secs				0 - 65534 (0xFFFF has the same effect as 0x0000)
179	40249	40249	1 R	0x03	VArPct_Mod	Enumerated value. VAR limit mode.		enum16		2: VAR limit as a % of VArMax			
180	40250	40250	1 RW	0x03 0x06 0x10	VArPct_Ena	Enumerated value. Fixed VAR enable/disable control.		enum16		0: Disabled1: Enabled			

181	40251	40251	1 R	0x03	WMaxLimPct_SF	Scale factor for power output percent.		sunssf			-2
182	40252	40252	1 R	0x03	OutPFSet_SF	Scale factor for power factor.		sunssf			-3
183	40253	40253	1 R	0x03	VArPct_SF	Scale factor for reactive power.		sunssf			0
184	40254	40254	1 R	0x03	ID	A well-known value 160. Uniquely identifies this as a SunSpec Multiple f unit16					160
185	40255	40255	1 R	0x03	L	Length of Multiple MPPT Inverter Extension Model		uint16			48
186	40256	40256	1 R	0x03	DCA_SF	Current Scale Factor		sunssf			
187	40257	40257	1 R	0x03	DCV_SF	Voltage Scale Factor		sunssf			
188	40258	40258	1 R	0x03	DCW_SF	Power Scale Factor		sunssf			
189	40259	40259	1 R	0x03	DCWH_SF	Energy Scale Factor		sunssf		Not supported for Fronius Hybrid inverters	
190	40260	40261	2 R	0x03	Evt	Global Events		bitfield32			
192	40262	40262	1 R	0x03	N	Number of Modules		uint16			2
194	40264	40264	1 R	0x03	1_ID	Input ID		uint16			1
195	40265	40272	8 R	0x03	1_IDStr	Input ID String		String16		"String 1"	
203	40273	40273	1 R	0x03	1_DCA	DC Current		uint16	A	DCA_SF	
204	40274	40274	1 R	0x03	1_DCV	DC Voltage		uint16	V	DCV_SF	
205	40275	40275	1 R	0x03	1_DCW	DC Power		uint16	W	DCW_SF	
206	40276	40277	2 R	0x03	1_DCWH	Lifetime Energy		acc32	Wh	DCWH_SF	Not supported for Fronius Hybrid inverters	
208	40278	40279	2 R	0x03	1_Tms	Timestamp		uint32	Secs		
211	40281	40281	1 R	0x03	1_DCSt	Operating State		enum16			
214	40284	40284	1 R	0x03	2_ID	Input ID		uint16			2
215	40285	40292	8 R	0x03	2_IDStr	Input ID Sting		String16		"String 2" or "Not supported"	
223	40293	40293	1 R	0x03	2_DCA	DC Current		uint16	A	DCA_SF	Not supported if only one DC input.
224	40294	40294	1 R	0x03	2_DCV	DC Voltage		uint16	V	DCV_SF	Not supported if only one DC input.
225	40295	40295	1 R	0x03	2_DCW	DC Power		uint16	W	DCW_SF	Not supported if only one DC input.
226	40296	40297	2 R	0x03	2_DCWH	Lifetime Energy		acc32	Wh	DCWH_SF	Not supported if only one DC input.Not supported for Fronius Hybrid inverters
228	40298	40299	2 R	0x03	2_Tms	Timestamp		uint32	Secs		Not supported if only one DC input.
231	40301	40301	1 R	0x03	2_DCSt	Operating State		enum16			Not supported if only one DC input.

APÊNDICE B – Mapeamento de Registradores Modbus do Inversor Victron

Tabela 6 – Mapeamento dos Registradores do Inversor Victron.

NOTE: Use unit-id 100 for the com.victronenergy.system data, for more information see FAQ.					ModbusTCP Register list					
dbus-service-name	description	pos	Address	Type	Scale	Range	dbus-obj-path	writable	dbus-unit	Remarks
com.victronenergy.system	Serial (System)		800	string[6]	1	12 characters	/Serial	no		System value -> MAC address of CCGX (represented as string)
com.victronenergy.system	CCGX Relay 1 state	0	806	uint16	1	0 to 1	/Relay/0/State	yes	0=Open; 1=Closed	
com.victronenergy.system	CCGX Relay 2 state	1	807	uint16	1	0 to 1	/Relay/1/State	yes	0=Open; 1=Closed	Relay 1 is available on Venus GX only.
com.victronenergy.system	PV - AC-coupled on output L1	2	808	uint16	1	0 to 65536	/Ac/PvOnOutput/L1/Power	no	W	Summation of all AC-Coupled PV Inverters on the output
com.victronenergy.system	PV - AC-coupled on input L1	5	811	uint16	1	0 to 65536	/Ac/PvOnGrid/L1/Power	no	W	Summation of all AC-Coupled PV Inverters on the input
com.victronenergy.system	AC Consumption L1	11	817	uint16	1	0 to 65536	/Ac/Consumption/L1/Power	no	W	Power supplied by system to loads.
com.victronenergy.system	Grid L1	14	820	int16	1	-32768 to 32767	/Ac/Grid/L1/Power	no	W	Power supplied by Grid to system.
com.victronenergy.system	Active input source	20	826	int16	1	0 to 32768	/Ac/ActiveIn/Source	no	0=Unknown; 1=Grid; 2=Generator; 3=Shore power; 240=Not connected	0 indicates that there is an active input, but it is not configured under Settings → System setup.
com.victronenergy.system	Battery Voltage (System)	34	840	uint16	10	0 to 6553.5	/Dc/Battery/Voltage	no	V DC	Battery Voltage determined from different measurements. In order of preference: BMV-voltage (V), Multi-DC-Voltage (CV), MPPT-DC-Voltage (ScV), Charger voltage
com.victronenergy.system	Battery Current (System)	35	841	int16	10	-3276.8 to 3276.7	/Dc/Battery/Current	no	A DC	Postive: battery begin charged. Negative: battery being discharged
com.victronenergy.system	Battery Power (System)	36	842	int16	1	-32768 to 32767	/Dc/Battery/Power	no	W	Postive: battery begin charged. Negative: battery being discharged
com.victronenergy.system	Battery State of Charge (System)	37	843	uint16	1	0 to 100	/Dc/Battery/Soc	no	%	Best battery state of charge, determined from different measurements.
com.victronenergy.system	Battery state (System)	38	844	uint16	1	0 to 65536	/Dc/Battery/State	no	0=idle; 1=charging; 2=discharging	
com.victronenergy.system	Battery Consumed Amphours (System)	39	845	uint16	-10	0 to -6553.6	/Dc/Battery/ConsumedAmphours	no	Ah	
com.victronenergy.system	Battery Time to Go (System)	40	846	uint16	0,01	0 to 6553600	/Dc/Battery/TimeToGo	no	s	Special value: 0 = charging
com.victronenergy.system	PV - DC-coupled power	44	850	uint16	1	0 to 65536	/Dc/Pv/Power	no	W	Summation of output power of all connected Solar Chargers
com.victronenergy.system	PV - DC-coupled current	45	851	int16	10	-3276.8 to 3276.7	/Dc/Pv/Current	no	A DC	Summation of output current of all connected Solar Chargers
com.victronenergy.system	Charger power	49	855	uint16	1	0 to 65536	/Dc/Charger/Power	no	W	
com.victronenergy.system	DC System Power	54	860	int16	1	-32768 to 32767	/Dc/System/Power	no	W	Power supplied by Battery to system.
com.victronenergy.system	VE.Bus charge current (System)	59	865	int16	10	-3276.8 to 3276.7	/Dc/Vebus/Current	no	A DC	Current flowing from the Multi to the dc system. Negative: the other way around.
com.victronenergy.system	VE.Bus charge power (System)	60	866	int16	1	-32768 to 32767	/Dc/Vebus/Power	no	W	System value etc. AND: Positive: power flowing from the Multi to the dc system. Negative: the other way around.
com.victronenergy.pvinverter	Position		1026	uint16	1	0 to 65535	/Position	no	0=AC input 1; 1=AC output; 2=AC input 2	
com.victronenergy.pvinverter	L1 Voltage		1027	uint16	10	0 to 6553.5	/Ac/L1/Voltage	no	V AC	
com.victronenergy.pvinverter	L1 Current		1028	int16	10	-3276.8 to 3276.7	/Ac/L1/Current	no	A AC	
com.victronenergy.pvinverter	L1 Power		1029	uint16	1	0 to 65535	/Ac/L1/Power	no	W	

APÊNDICE C – Função de Separação de Registros do Inversor Fronius

O código a seguir, implementado usando a linguagem JavaScript no nó de função do Node-RED `Modbus Separação de Registros - Fronius`, é responsável por processar o vetor de dados lidos do inversor Fronius e organizá-lo em um objeto dentro de uma variável de contexto global.

Código C.2 – Código-fonte da função `Modbus Separação de Registros - Fronius`.

```
// ================================================
// === FUNÇÕES AJUDANTES
// ================================================
function toInt16(val) { return val > 0x7FFF ? val - 0x10000 : val; }
function toUint32(lo, hi) { return (hi << 16) + lo; }
function toAcc64(w0, w1, w2, w3) {
  return (BigInt(w3) << 48n) + (BigInt(w2) << 32n) + (BigInt(w1) << 16n) +
  ↪  BigInt(w0);
}

// ================================================
// === PROCESSAMENTO DOS REGISTRADORES FRONIUS
// ================================================

// 1. Inicializa objeto limpo
const froniusData = {};
const values = msg.payload; // registros brutos começando em 40070

// ================================================
// MODEL 101/102/103: Inverter Measurements - Medidas do Inversor
// ================================================
froniusData.model101 = {};

// Scale Factors
```

```
const A_SF   = toInt16(values[6]);   // 40076
const V_SF   = toInt16(values[13]);  // 40083
const W_SF   = toInt16(values[15]);  // 40085
const Hz_SF  = toInt16(values[17]);  // 40087
const VA_SF  = toInt16(values[19]);  // 40089
const VAr_SF = toInt16(values[21]);  // 40091
const PF_SF  = toInt16(values[23]);  // 40093
const WH_SF  = toInt16(values[26]);  // 40096

// --- Identificação ---
froniusData.model101.ID = { value: values[0], unit: "enum", address: 40070,
↪  description: "SunSpec Model ID" };
froniusData.model101.L = { value: values[1], unit: "registers", address: 40071,
↪  description: "Model Length" };

// --- AC Current ---
froniusData.model101.AC_Current_Total = { value: values[2] * Math.pow(10, A_SF),
↪  unit: "A", address: 40072, scale_factor: A_SF };
froniusData.model101.AC_Current_PhaseA = { value: values[3] * Math.pow(10,
↪  A_SF), unit: "A", address: 40073, scale_factor: A_SF };
//froniusData.model101.AC_Current_PhaseB = { value: values[4] * Math.pow(10,
↪  A_SF), unit: "A", address: 40074, scale_factor: A_SF };
//froniusData.model101.AC_Current_PhaseC = { value: values[5] * Math.pow(10,
↪  A_SF), unit: "A", address: 40075, scale_factor: A_SF };
froniusData.model101.AC_Current_SF = { value: A_SF, unit: "sunssf", address:
↪  40076 };

// --- AC Voltage ---
//froniusData.model101.AC_Voltage_AB = { value: values[7] * Math.pow(10, V_SF),
↪  unit: "V", address: 40077, scale_factor: V_SF };
//froniusData.model101.AC_Voltage_BC = { value: values[8] * Math.pow(10, V_SF),
↪  unit: "V", address: 40078, scale_factor: V_SF };
//froniusData.model101.AC_Voltage_CA = { value: values[9] * Math.pow(10, V_SF),
↪  unit: "V", address: 40079, scale_factor: V_SF };
froniusData.model101.AC_Voltage_AN = { value: values[10] * Math.pow(10, V_SF),
↪  unit: "V", address: 40080, scale_factor: V_SF };
//froniusData.model101.AC_Voltage_BN = { value: values[11] * Math.pow(10, V_SF),
↪  unit: "V", address: 40081, scale_factor: V_SF };
//froniusData.model101.AC_Voltage_CN = { value: values[12] * Math.pow(10, V_SF),
↪  unit: "V", address: 40082, scale_factor: V_SF };
```

```
51 froniusData.model101.AC_Voltage_SF = { value: V_SF, unit: "sunssf", address:
   ↪ 40083 };
52
53 // --- Power ---
54 froniusData.model101.AC_Power = { value: toInt16(values[14]) * Math.pow(10,
   ↪ W_SF), unit: "W", address: 40084, scale_factor: W_SF };
55 froniusData.model101.AC_Power_SF = { value: W_SF, unit: "sunssf", address:
   ↪ 40085 };
56 froniusData.model101.AC_Frequency = { value: values[16] * Math.pow(10, Hz_SF),
   ↪ unit: "Hz", address: 40086, scale_factor: Hz_SF };
57 froniusData.model101.AC_Frequency_SF = { value: Hz_SF, unit: "sunssf", address:
   ↪ 40087 };
58 froniusData.model101.AC_VA = { value: toInt16(values[18]) * Math.pow(10, VA_SF),
   ↪ unit: "VA", address: 40088, scale_factor: VA_SF };
59 froniusData.model101.AC_VA_SF = { value: VA_SF, unit: "sunssf", address: 40089
   ↪ };
60 froniusData.model101.AC_VAr = { value: toInt16(values[20]) * Math.pow(10,
   ↪ VAr_SF), unit: "VAr", address: 40090, scale_factor: VAr_SF };
61 froniusData.model101.AC_VAr_SF = { value: VAr_SF, unit: "sunssf", address:
   ↪ 40091 };
62 froniusData.model101.AC_PowerFactor = { value: toInt16(values[22]) *
   ↪ Math.pow(10, PF_SF), unit: "%", address: 40092, scale_factor: PF_SF };
63 froniusData.model101.AC_PowerFactor_SF = { value: PF_SF, unit: "sunssf",
   ↪ address: 40093 };
64
65 // --- Energy ---
66 froniusData.model101.AC_Energy_WH = { value: toUint32(values[24], values[25]) *
   ↪ Math.pow(10, WH_SF), unit: "Wh", address: 40094, scale_factor: WH_SF };
67 froniusData.model101.AC_Energy_SF = { value: WH_SF, unit: "sunssf", address:
   ↪ 40096 };
68
69 // --- DC Measurements ---
70 const DCA_SF = toInt16(values[28]);  // 40098
71 const DCV_SF = toInt16(values[30]);  // 40100
72 const DCW_SF = toInt16(values[32]);  // 40102
73
74 froniusData.model101.DC_Current = { value: values[27] * Math.pow(10, DCA_SF),
   ↪ unit: "A", address: 40097, scale_factor: DCA_SF };
75 froniusData.model101.DC_Current_SF = { value: DCA_SF, unit: "sunssf", address:
   ↪ 40098 };
```

```
76 froniusData.model101.DC_Voltage = { value: values[29] * Math.pow(10, DCV_SF),
   ↪ unit: "V", address: 40099, scale_factor: DCV_SF };
77 froniusData.model101.DC_Voltage_SF = { value: DCV_SF, unit: "sunssf", address:
   ↪ 40100 };
78 froniusData.model101.DC_Power = { value: toInt16(values[31]) * Math.pow(10,
   ↪ DCW_SF), unit: "W", address: 40101, scale_factor: DCW_SF };
79 froniusData.model101.DC_Power_SF = { value: DCW_SF, unit: "sunssf", address:
   ↪ 40102 };
80
81 // --- Status ---
82 froniusData.model101.Operating_State = { value: values[38], unit: "enum",
   ↪ address: 40108 };
83 froniusData.model101.Vendor_State = { value: values[39], unit: "enum", address:
   ↪ 40109 };
84
85 // --- Events ---
86 froniusData.model101.Events1 = { value: toUint32(values[40], values[41]), unit:
   ↪ "bitfield", address: 40110 };
87 froniusData.model101.Events2 = { value: toUint32(values[42], values[43]), unit:
   ↪ "bitfield", address: 40112 };
88 froniusData.model101.Vendor_Events1 = { value: toUint32(values[44], values[45]),
   ↪ unit: "bitfield", address: 40114 };
89 froniusData.model101.Vendor_Events2 = { value: toUint32(values[46], values[47]),
   ↪ unit: "bitfield", address: 40116 };
90 froniusData.model101.Vendor_Events3 = { value: toUint32(values[48], values[49]),
   ↪ unit: "bitfield", address: 40118 };
91
92 // ================================================
93 // MODEL 120: Nameplate Ratings - Classificação Indicativa
94 // ================================================
95 froniusData.model120 = {};
96
97 const WRtg_SF = toInt16(values[56]);   // 40126
98 const VARtg_SF = toInt16(values[58]);  // 40128
99 const VArRtgQ_SF = toInt16(values[63]); // 40133
100 const ARtg_SF = toInt16(values[65]);   // 40135
101 const PFRtg_SF = toInt16(values[70]);  // 40140
102
103 froniusData.model120.ID = { value: values[52], unit: "enum", address: 40122,
   ↪ description: "Nameplate Model ID", ignore: true };
```

```
froniusData.model120.L = { value: values[53], unit: "registers", address: 40123, description: "Model Length", ignore: true };
froniusData.model120.DeviceType = { value: values[54], unit: "enum", address: 40124, description: "Type of DER device" };

froniusData.model120.WRtg = { value: values[55] * Math.pow(10, WRtg_SF), unit: "W", address: 40125, scale_factor: WRtg_SF };
froniusData.model120.WRtg_SF = { value: WRtg_SF, unit: "sunssf", address: 40126, ignore: true };
froniusData.model120.VARtg = { value: values[57] * Math.pow(10, VARtg_SF), unit: "VA", address: 40127, scale_factor: VARtg_SF };
froniusData.model120.VARtg_SF = { value: VARtg_SF, unit: "sunssf", address: 40128, ignore: true };
froniusData.model120.ARtg = { value: values[64] * Math.pow(10, ARtg_SF), unit: "A", address: 40134, scale_factor: ARtg_SF };
froniusData.model120.ARtg_SF = { value: ARtg_SF, unit: "sunssf", address: 40135, ignore: true };

// ================================================
// MODEL 121: Basic Settings - Configurações Basicas
// ================================================
froniusData.model121 = {};

const WMax_SF = toInt16(values[102]);  // 40172
const VRef_SF = toInt16(values[103]);  // 40173
const VRefOfs_SF = toInt16(values[104]); // 40174
const VAMax_SF = toInt16(values[106]); // 40176
const VARMax_SF = toInt16(values[107]); // 40177

froniusData.model121.ID = { value: values[80], unit: "enum", address: 40150, description: "Basic Settings Model ID", ignore: true };
froniusData.model121.L = { value: values[81], unit: "registers", address: 40151, description: "Model Length", ignore: true };

froniusData.model121.WMax = { value: values[82] * Math.pow(10, WMax_SF), unit: "W", address: 40152, scale_factor: WMax_SF, writable: true };
froniusData.model121.VRef = { value: values[83] * Math.pow(10, VRef_SF), unit: "V", address: 40153, scale_factor: VRef_SF, writable: true };
```

```
130 froniusData.model121.VRefOfs = { value: toInt16(values[84]) * Math.pow(10,
    ↪ VRefOfs_SF), unit: "V", address: 40154, scale_factor: VRefOfs_SF, writable:
    ↪ true };
131 froniusData.model121.VAMax = { value: values[87] * Math.pow(10, VAMax_SF), unit:
    ↪ "VA", address: 40157, scale_factor: VAMax_SF, writable: true };
132
133 // ================================================
134 // MODEL 122: Measurements Status - Status de medição
135 // ================================================
136 froniusData.model122 = {};
137
138 froniusData.model122.ID = { value: values[112], unit: "enum", address: 40182,
    ↪ description: "Measurements Status Model ID", ignore: true };
139 froniusData.model122.L = { value: values[113], unit: "registers", address:
    ↪ 40183, description: "Model Length", ignore: true };
140 froniusData.model122.Status = { value: values[114], unit: "bitfield", address:
    ↪ 40184, description: "PV Inverter Present/Available Status" };
141 froniusData.model122.PCC_Connection = { value: values[116], unit: "enum",
    ↪ address: 40186, description: "PCC Connection Status" };
142
143 // Energy totals (lifetime)
144 froniusData.model122.AC_Energy_WH_Total = { value: toUint32(values[117],
    ↪ values[118]), unit: "Wh", address: 40187 };
145 froniusData.model122.AC_Energy_VAh_Total = { value: toUint32(values[121],
    ↪ values[122]), unit: "VAh", address: 40191 };
146 froniusData.model122.AC_Energy_VArh_Q1 = { value: toUint32(values[125],
    ↪ values[126]), unit: "VArh", address: 40195 };
147 froniusData.model122.AC_Energy_VArh_Q2 = { value: toUint32(values[129],
    ↪ values[130]), unit: "VArh", address: 40199 };
148 froniusData.model122.AC_Energy_VArh_Q3 = { value: toUint32(values[133],
    ↪ values[134]), unit: "VArh", address: 40203 };
149 froniusData.model122.AC_Energy_VArh_Q4 = { value: toUint32(values[137],
    ↪ values[138]), unit: "VArh", address: 40207 };
150
151 // Available power
152 const VArAval_SF = toInt16(values[142]); // 40212
153 const WAval_SF = toInt16(values[144]);   // 40214
154
155 froniusData.model122.VArAvailable = { value: toInt16(values[141]) * Math.pow(10,
    ↪ VArAval_SF), unit: "VAr", address: 40211, scale_factor: VArAval_SF };
```

```
156 froniusData.model122.VArAvailable_SF = { value: VArAval_SF, unit: "sunssf",
    ↪  address: 40212, ignore: true };
157 froniusData.model122.WAvailable = { value: values[143] * Math.pow(10, WAval_SF),
    ↪  unit: "W", address: 40213, scale_factor: WAval_SF};
158 froniusData.model122.WAvailable_SF = { value: WAval_SF, unit: "sunssf", address:
    ↪  40214, ignore: true };
159 froniusData.model122.StActCtl = {value: values[147], unit: "bitfield", address:
    ↪  40217, description: "Bit Mask indicating which inverter controls are
    ↪  currently active."}
160
161 // Bit 0 → FixedW
162 froniusData.model122.StActCtl_FixedW = {value: (values[147] & (1 << 0)) ? 1 : 0,
    ↪  unit: "binary",address: 40217, description: "Active Control:
    ↪  FixedW",writable: false,isSwitch: false};
163 // Bit 1 → FixedVAR
164 froniusData.model122.StActCtl_FixedVAR = {value: (values[147] & (1 << 1)) ? 1 :
    ↪  0, unit: "binary", address: 40217, description: "Active Control: FixedVAR",
    ↪  writable: false, isSwitch: false};
165 // Bit 2 → FixedPF
166 froniusData.model122.StActCtl_FixedPF = {value: (values[147] & (1 << 2)) ? 1 :
    ↪  0, unit: "binary", address: 40217, description: "Active Control: FixedPF",
    ↪  writable: false, isSwitch: false};
167
168
169
170 // ================================================
171 // MODEL 123: Immediate Controls - Controles Imediatos
172 // ================================================
173 froniusData.model123 = {};
174
175 // Scale Factors
176 const WMaxLimPct_SF = toInt16(values[181]); // 40251
177 const OutPFSet_SF = toInt16(values[182]); // 40252
178 const VArPct_SF = toInt16(values[183]); // 40253
179
180 // Identificação
181 froniusData.model123.ID = { value: values[158], unit: "enum", address: 40228,
    ↪  description: "Immediate Controls Model ID", ignore: true };
182 froniusData.model123.L = { value: values[159], unit: "registers", address:
    ↪  40229, description: "Model Length", ignore: true };
```

```
183
184 // Conexão
185 froniusData.model123.Conn_WinTms = { value: values[160], unit: "s", address:
    ↪ 40230, description: "Janela de tempo para conectar/desconectar", writable:
    ↪ true };
186 froniusData.model123.Conn_RvrtTms = { value: values[161], unit: "s", address:
    ↪ 40231, description: "Timeout para conexão/desconexão", writable: true };
187 froniusData.model123.Conn = { value: values[162], unit: "enum", address: 40232,
    ↪ description: "Controle de conexão (0=Desconectado, 1=Conectado)", writable:
    ↪ true, isSwitch: true };
188
189 // Potência ativa
190 froniusData.model123.WMaxLimPct = { value: values[163] * Math.pow(10,
    ↪ WMaxLimPct_SF), unit: "% WMax", address: 40233, scale_factor: WMaxLimPct_SF,
    ↪ writable: true };
191 froniusData.model123.WMaxLimPct_WinTms = { value: values[164], unit: "s",
    ↪ address: 40234, description: "Janela de tempo para mudança de limite de
    ↪ potência", writable: true };
192 froniusData.model123.WMaxLimPct_RvrtTms = { value: values[165], unit: "s",
    ↪ address: 40235, description: "Timeout do limite de potência", writable:
    ↪ true };
193 froniusData.model123.WMaxLimPct_RmpTms = { value: values[166], unit: "s",
    ↪ address: 40236, description: "Tempo de rampa para mudar setpoint de
    ↪ potência", writable: true };
194 froniusData.model123.WMaxLim_Ena = { value: values[167], unit: "enum", address:
    ↪ 40237, description: "Habilita/Desabilita limite de potência", writable:
    ↪ true, isSwitch: true };
195
196 // Fator de potência
197 froniusData.model123.OutPFSet = { value: toInt16(values[168]) * Math.pow(10,
    ↪ OutPFSet_SF), unit: "cos()", address: 40238, scale_factor: OutPFSet_SF,
    ↪ writable: true };
198 froniusData.model123.OutPFSet_WinTms = { value: values[169], unit: "s", address:
    ↪ 40239, description: "Janela de tempo para mudança do fator de potência",
    ↪ writable: true };
199 froniusData.model123.OutPFSet_RvrtTms = { value: values[170], unit: "s",
    ↪ address: 40240, description: "Timeout para fator de potência", writable:
    ↪ true };
```

```
200 froniusData.model123.OutPFSet_RmpTms = { value: values[171], unit: "s", address:
    ↪ 40241, description: "Tempo de rampa para mudar fator de potência", writable:
    ↪ true };
201 froniusData.model123.OutPFSet_Ena = { value: values[172], unit: "enum", address:
    ↪ 40242, description: "Habilita/Desabilita fator de potência fixo", writable:
    ↪ true, isSwitch: true };
202
203 // Potência reativa
204 froniusData.model123.VArWMaxPct = { value: toInt16(values[173]) * Math.pow(10,
    ↪ VArPct_SF), unit: "% WMax", address: 40243, scale_factor: VArPct_SF,
    ↪ description: "Potência reativa em % de WMax (reflete VarMaxPct)", writable:
    ↪ false };
205 froniusData.model123.VArMaxPct = { value: toInt16(values[174]) * Math.pow(10,
    ↪ VArPct_SF), unit: "% VArMax", address: 40244, scale_factor: VArPct_SF,
    ↪ writable: true };
206 // VArAvalPct não suportado → ignorado
207 froniusData.model123.VArPct_WinTms = { value: values[176], unit: "s", address:
    ↪ 40246, description: "Janela de tempo para mudança de limite de VAR",
    ↪ writable: true };
208 froniusData.model123.VArPct_RvrtTms = { value: values[177], unit: "s", address:
    ↪ 40247, description: "Timeout para limite de VAR", writable: true };
209 froniusData.model123.VArPct_RmpTms = { value: values[178], unit: "s", address:
    ↪ 40248, description: "Tempo de rampa para mudar limite de VAR", writable:
    ↪ true };
210 froniusData.model123.VArPct_Mod = { value: values[179], unit: "enum", address:
    ↪ 40249, description: "Modo do limite de VAR (2 = % de VArMax)"};
211 froniusData.model123.VArPct_Ena = { value: values[180], unit: "enum", address:
    ↪ 40250, description: "Habilita/Desabilita controle fixo de VAR", writable:
    ↪ true, isSwitch: true };
212
213 // Scale Factors
214 froniusData.model123.WMaxLimPct_SF = { value: WMaxLimPct_SF, unit: "sunssf",
    ↪ address: 40251, ignore: true };
215 froniusData.model123.OutPFSet_SF = { value: OutPFSet_SF, unit: "sunssf",
    ↪ address: 40252, ignore: true };
216 froniusData.model123.VArPct_SF = { value: VArPct_SF, unit: "sunssf", address:
    ↪ 40253, ignore: true };
217
218
219 // ================================================
```

```
220 // MODEL 160: Multiple MPPT Extension - Extensão para multiplos MPPT
221 // ================================================
222 froniusData.model160 = {};
223
224 const DCA_SF_160 = toInt16(values[186]); // 40256
225 const DCV_SF_160 = toInt16(values[187]); // 40257
226 const DCW_SF_160 = toInt16(values[188]); // 40258
227
228 froniusData.model160.ID = { value: values[184], unit: "enum", address: 40254,
    ↪  description: "Multiple MPPT Model ID", ignore: true };
229 froniusData.model160.L = { value: values[185], unit: "registers", address:
    ↪  40255, description: "Model Length", ignore: true };
230
231 froniusData.model160.DC_Current_SF = { value: DCA_SF_160, unit: "sunssf",
    ↪  address: 40256, ignore: true };
232 froniusData.model160.DC_Voltage_SF = { value: DCV_SF_160, unit: "sunssf",
    ↪  address: 40257, ignore: true };
233 froniusData.model160.DC_Power_SF = { value: DCW_SF_160, unit: "sunssf", address:
    ↪  40258, ignore: true };
234
235 froniusData.model160.Num_Modules = { value: values[192], unit: "count", address:
    ↪  40262, description: "Number of MPPT Modules" };
236
237 // Module 1
238 froniusData.model160.Module1_ID = { value: values[194], unit: "enum", address:
    ↪  40264, ignore: true };
239 froniusData.model160.Module1_DC_Current = { value: values[203] * Math.pow(10,
    ↪  DCA_SF_160), unit: "A", address: 40273, scale_factor: DCA_SF_160 };
240 froniusData.model160.Module1_DC_Voltage = { value: values[204] * Math.pow(10,
    ↪  DCV_SF_160), unit: "V", address: 40274, scale_factor: DCV_SF_160 };
241 froniusData.model160.Module1_DC_Power = { value: values[205] * Math.pow(10,
    ↪  DCW_SF_160), unit: "W", address: 40275, scale_factor: DCW_SF_160 };
242 froniusData.model160.Module1_Operating_State = { value: values[211], unit:
    ↪  "enum", address: 40281 };
243
244 // Module 2 (if supported)
245 if (values[192] > 1) {
246     froniusData.model160.Module2_ID = { value: values[214], unit: "enum",
        ↪  address: 40284, ignore: true };
```

```
247     froniusData.model160.Module2_DC_Current = { value: values[223] *
        ↪ Math.pow(10, DCA_SF_160), unit: "A", address: 40293, scale_factor:
        ↪ DCA_SF_160 };
248     froniusData.model160.Module2_DC_Voltage = { value: values[224] *
        ↪ Math.pow(10, DCV_SF_160), unit: "V", address: 40294, scale_factor:
        ↪ DCV_SF_160 };
249     froniusData.model160.Module2_DC_Power = { value: values[225] * Math.pow(10,
        ↪ DCW_SF_160), unit: "W", address: 40295, scale_factor: DCW_SF_160 };
250     froniusData.model160.Module2_Operating_State = { value: values[231], unit:
        ↪ "enum", address: 40301 };
251 }
252
253 // ================================================
254 // FINALIZAÇÃO
255 // ================================================
256
257 // Salva no contexto global
258 global.set('fronius', froniusData);
259
260 // Retorna no payload
261 msg.payload = froniusData;
262
263 return msg;
```

APÊNDICE D – Função de Separação de Registros do Inversor Victron

O código a seguir, implementado usando a linguagem JavaScript no nó de função do Node-RED `Modbus Separação de Registros - Victron`, é responsável por processar o vetor de dados lidos do inversor Victron e organizá-lo em um objeto dentro de uma variável de contexto global.

Código D.2 – Código-fonte da função `Modbus Separação de Registros - Victron`.

```
// === Separação de Registros - Victron (Node-RED Function) ===
// Recebe: msg.payload = array de uint16 (registros), msg.modbusRequest.address = startAddress
// Saída: grava global.set('victron', victronData)
// NÃO publica MQTT/Discovery aqui.

const registerMap = {
    // SYSTEM & AC L1 (800+)
    800: { key:"system_serial", display:"System Serial", unit:null, type:"string", len:12, category:"system" },
    806: { key:"relay1_state", display:"Relay 1 State", unit:null, type:"enum", scale:1, writable:true, category:"relays" },
    807: { key:"relay2_state", display:"Relay 2 State", unit:null, type:"enum", scale:1, writable:true, category:"relays" },
    808: { key:"pv_ac_output_l1", display:"PV AC Output L1", unit:"W", type:"uint16", scale:1, category:"solar_ac" },
    // 809,810 (L2/L3) OMITIDOS ON PURPOSE
    811: { key:"pv_ac_grid_l1", display:"PV AC Grid L1", unit:"W", type:"uint16", scale:1, category:"solar_ac" },
    // 812,813 (L2/L3) OMITIDOS
    817: { key:"ac_consumption_l1", display:"AC Consumption L1", unit:"W", type:"uint16", scale:1, category:"ac_grid" },
    // 818,819 omitted
```

```
    820: { key:"grid_power_l1", display:"Grid Power L1", unit:"W", type:"int16", scale:1, category:"ac_grid" },
    // 821,822 omitted
    826: { key:"active_input_source", display:"Active Input Source", unit:null, type:"enum", scale:1, category:"system" },

    // BATTERY (840+)
    840: { key:"battery_voltage", display:"Battery Voltage", unit:"V", type:"uint16", scale:10, category:"battery" },
    841: { key:"battery_current", display:"Battery Current", unit:"A", type:"int16", scale:10, category:"battery" },
    842: { key:"battery_power", display:"Battery Power", unit:"W", type:"int16", scale:1, category:"battery" },
    843: { key:"battery_soc", display:"Battery SOC", unit:"%", type:"uint16", scale:1, category:"battery" },
    844: { key:"battery_state", display:"Battery State", unit:null, type:"enum", scale:1, category:"battery" },
    845: { key:"battery_consumed_ah", display:"Battery Consumed Ah", unit:"Ah", type:"uint16", scale:10, category:"battery" },
    846: { key:"battery_time_to_go", display:"Battery Time to Go", unit:"s", type:"uint16", scale:0.01, category:"battery" },

    // PV DC (850+)
    850: { key:"pv_dc_power", display:"PV DC Coupled Power", unit:"W", type:"uint16", scale:1, category:"solar_dc" },
    851: { key:"pv_dc_current", display:"PV DC Coupled Current", unit:"A", type:"int16", scale:10, category:"solar_dc" },
    855: { key:"charger_power", display:"Charger Power", unit:"W", type:"uint16", scale:1, category:"solar_dc" },
    860: { key:"dc_system_power", display:"DC System Power", unit:"W", type:"int16", scale:1, category:"solar_dc" },

    // VE.Bus (865+)
    865: { key:"vebus_charge_current", display:"VE.Bus Charge Current", unit:"A", type:"int16", scale:10, category:"vebus" },
    866: { key:"vebus_charge_power", display:"System Bus Charge Power", unit:"W", type:"int16", scale:1, category:"vebus" },
    // VE.Bus AC OUT (1026+)
    1026: { key:"vebus_out_l1_voltage", display:"VE.Bus AC Out L1 Voltage", unit:"V", type:"uint16", scale:10, category:"vebus" },
```

```
    1028: { key:"vebus_out_l1_current", display:"VE.Bus AC Out L1 Current", unit:"A", type:"int16", scale:10, category:"vebus" },
    1029: { key:"vebus_out_l1_power",   display:"VE.Bus AC Out L1 Power", unit:"W", type:"uint16", scale:1,  category:"vebus" }
};

// HELPERS
function toInt16(val) {
    let uint16 = val & 0xFFFF;
    return (uint16 & 0x8000) ? uint16 - 0x10000 : uint16;
}
function toInt32(lo, hi) {
    const val = (hi << 16) | lo;
    return (val > 0x7FFFFFFF) ? val - 0x100000000 : val;
}
function decodeString(buffer) {
    let str = "";
    for (let i = 0; i < buffer.length; i++) {
        str += String.fromCharCode((buffer[i] >> 8) & 0xFF);
        str += String.fromCharCode(buffer[i] & 0xFF);
    }
    return str.replace(/\0/g, '');
}

// Safe id helper
function safeKey(k){ return k.toLowerCase().replace(/[^a-z0-9]/g,'_'); }

// START
if (!msg.modbusRequest || !Array.isArray(msg.payload)) return null;

const startAddress = msg.modbusRequest.address;
const regs = msg.payload;
let victronData = global.get('victron') || {
    solar_ac:{}, solar_dc:{}, ac_grid:{}, battery:{}, relays:{}, vebus:{}, system:{}
};

for (let i=0; i<regs.length; i++){
    const addr = startAddress + i;
```

```
77      const def = registerMap[addr];
78      if (!def) continue;
79
80      let raw = regs[i];
81      let value = null;
82      let skip = 0;
83
84      switch(def.type){
85          case "int16":
86              value = toInt16(raw);
87              break;
88          case "uint16":
89          case "enum":
90              value = raw;
91              break;
92          case "int32":
93              if (i+1 < regs.length) {
94                  // Victron order can vary; try (low, high) then (high, low) if
                    ↪  values look wrong in practice
95                  value = toInt32(regs[i], regs[i+1]);
96                  skip = 1;
97              }
98              break;
99          case "string":
100             const nRegs = Math.ceil(def.len / 2);
101             if (i + nRegs <= regs.length) {
102                 value = decodeString(regs.slice(i, i + nRegs));
103                 skip = nRegs - 1;
104             }
105             break;
106         default:
107             value = raw;
108     }
109
110     if (value !== null && typeof value === 'number' && def.scale && def.scale
        ↪  !== 1) {
111         value = value / def.scale;
112         value = Math.round(value * 100) / 100;
113     }
```

```
114
115     if (value !== null) {
116         const cat = def.category || "system";
117         victronData[cat] = victronData[cat] || {};
118         victronData[cat][def.key] = {
119             name: def.display || def.key,
120             value: value,
121             unit: def.unit || null,
122             address: addr,
123             writable: !!def.writable,
124             description: def.display || ""
125         };
126     }
127
128     i += skip;
129 }
130
131 // Save global
132 global.set('victron', victronData);
133
134 return null;
```

APÊNDICE E – Código da Função de Publicação MQTT Inversor Fronius

O código JavaScript apresentado a seguir, pertencente ao nó de função `Escrita MQTT`, é responsável por formatar e publicar os dados do inversor Fronius para o broker MQTT. Sua principal característica é a implementação do protocolo **Home Assistant MQTT Discovery**, que automatiza a criação de entidades na plataforma de automação.

Código E.2 – Código-fonte da função `Escrita MQTT - Fronius` para integração via MQTT Discovery.

```
// ================================================
// FUNÇÃO: PUBLICAR DADOS NO MQTT PARA HOME ASSISTANT
// ================================================

// 1. Recupera o objeto fronius salvo no contexto global
const froniusData = global.get('fronius');
if (!froniusData) {
    node.warn("Nenhum dado Fronius encontrado no contexto global.");
    return null;
}

// 2. Função auxiliar para nomes amigáveis de modelos
function modelFriendlyName(model) {
    const map = {
        model101: "Model 101 - Inverter Measurements",
        model120: "Model 120 - Nameplate Ratings",
        model121: "Model 121 - Basic Settings",
        model122: "Model 122 - Measurements Status",
        model123: "Model 123 - Immediate Controls",
        model124: "Model 124 - Basic Storage Controls",
        model160: "Model 160 - Multiple MPPT Extension"
    };
    return map[model] || model;
}
```

```
25
26 // 3. Função auxiliar para gerar payload de discovery
27 function makeDiscoveryPayload(model, point, id, name, unit, writable = false,
   ↪  isSwitch = false) {
28     let baseDevice = {
29         identifiers: [`fronius_${model}`],
30         name: `Fronius Inverter - ${modelFriendlyName(model)}`,
31         manufacturer: "Fronius",
32         model: "PV Inverter"
33     };
34
35     let payload;
36     if (writable && isSwitch) {
37         // Para registros de escrita que são switches
38         payload = {
39             name: name,
40             command_topic: `homeassistant/switch/fronius/${id}/set`,
41             state_topic: `homeassistant/switch/fronius/${id}/state`,
42             unique_id: `fronius_${id}`,
43             device: baseDevice,
44             payload_on: "1",
45             payload_off: "0"
46         };
47     } else if (writable) {
48         // Para outros registros de escrita (number)
49         payload = {
50             name: name,
51             command_topic: `homeassistant/number/fronius/${id}/set`,
52             state_topic: `homeassistant/number/fronius/${id}/state`,
53             unique_id: `fronius_${id}`,
54             device: baseDevice,
55             unit_of_measurement: unit,
56             min: 0,
57             max: 99999
58         };
59     } else {
60         // Para registros somente leitura (sensor)
61         payload = {
62             name: name,
```

```
            state_topic: `homeassistant/sensor/fronius/${id}/state`,
            unique_id: `fronius_${id}`,
            device: baseDevice
        };
        if (unit) {
            payload.unit_of_measurement = unit;
                // --- Atribui device_class automaticamente ---
            if (unit === "W") {
                payload.device_class = "power";
                payload.state_class = "measurement";
            }
            else if (unit === "A") {
                payload.device_class = "current";
                payload.state_class = "measurement";
            }
            else if (unit === "V") {
                payload.device_class = "voltage";
                payload.state_class = "measurement";
            }
            else if (unit === "Wh" || unit === "kWh") {
                payload.device_class = "energy";
                payload.state_class = "total_increasing";
            }
        }
    }
    return payload;
}

// 4. Varre recursivamente todos os modelos e pontos
let mqttMsgs = [];

// Lista de registradores que devem ser tratados como switches
const switchRegisters = ['Conn', 'WMaxLim_Ena', 'OutPFSet_Ena', 'VArPct_Ena'];

for (let model in froniusData) {
    for (let point in froniusData[model]) {


```

```
102
103
104         const entry = froniusData[model][point];
105
106         if (entry.ignore === true) { //remove valores como o ID de serem
            ↪  enviados ao MQTT
107             continue;
108         }
109
110         const id = `${model}_${point}`;
111         const name = `${point}`; // Nome da entidade dentro do modelo
112         const unit = entry.unit || null;
113         const writable = entry.writable || false;
114         const isSwitch = writable && switchRegisters.includes(point); // Define
            ↪  isSwitch
115
116         // 4a. Discovery payload
117         let discoveryTopic;
118         if (isSwitch) {
119             discoveryTopic = `homeassistant/switch/fronius/${id}/config`;
120         } else if (writable) {
121             discoveryTopic = `homeassistant/number/fronius/${id}/config`;
122         } else {
123             discoveryTopic = `homeassistant/sensor/fronius/${id}/config`;
124         }
125
126         const discoveryPayload = makeDiscoveryPayload(model, point, id, name,
            ↪  unit, writable, isSwitch);
127
128         mqttMsgs.push({
129             topic: discoveryTopic,
130             payload: JSON.stringify(discoveryPayload),
131             qos: 1,
132             retain: true
133         });
134
135         // 4b. Valor atual (state)
136         let stateTopic;
137         let statePayload;
```

```
        if (isSwitch) {
            stateTopic = `homeassistant/switch/fronius/${id}/state`;
            if (entry.value === 1) {
                statePayload = "1";
            } else {
                statePayload = "0";
            }
        } else if (writable) {
            stateTopic = `homeassistant/number/fronius/${id}/state`;
            statePayload = String(entry.value);
        } else {
            stateTopic = `homeassistant/sensor/fronius/${id}/state`;
            statePayload = String(entry.value);
        }

        mqttMsgs.push({
            topic: stateTopic,
            payload: statePayload,
            qos: 0,
            retain: true
        });
    }
}

// 5. Retorna mensagens para nó MQTT out
return [mqttMsgs];
```

APÊNDICE F – Código da Função de Publicação MQTT Inversor Victron

O código JavaScript apresentado a seguir, pertencente ao nó de função `Escrita MQTT (Victron)`, é responsável por formatar e publicar os dados do inversor **Victron** para o broker MQTT. Sua principal característica é a implementação do protocolo **Home Assistant MQTT Discovery**, que automatiza a criação de entidades na plataforma de automação.

Código F.2 – Código-fonte da função `Escrita MQTT - Victron` para integração via MQTT Discovery.

```
// === Escrita MQTT - Victron (Node-RED Function) ===
// Publica discovery + estados para Home Assistant
// Agrupamento em 7 devices:
// solar_ac, solar_dc, ac_grid, battery, relays, vebus, system

const victronData = global.get("victron");
if (!victronData) return null;

// ================================
// DEVICE DEFINITIONS
// ================================
const devices = {
    solar_ac: { id: "victron_solar_ac", name: "Victron - Solar AC (CA)" },
    solar_dc: { id: "victron_solar_dc", name: "Victron - Solar DC (CC)" },
    ac_grid:  { id: "victron_ac_grid",  name: "Victron - AC / Rede" },
    battery:  { id: "victron_battery",  name: "Victron - Bateria do Sistema" },
    relays:   { id: "victron_relays",   name: "Victron - Relés" },
    vebus:    { id: "victron_vebus",    name: "Victron - VE.Bus" },
    system:   { id: "victron_system",   name: "Victron - Sistema" }
};

// ================================
// DEVICE INFO BASE
```

```
// ================================
function makeDeviceBase(deviceId, deviceName) {
    return {
        identifiers: [deviceId],
        name: deviceName,
        manufacturer: "Victron Energy",
        model: "Venus OS (Cerbo/CCGX)"
    };
}

// ================================
// DEVICE CLASS + STATE CLASS
// ================================
function getDeviceClassAndStateClass(unit) {
    if (!unit) return { state_class: "measurement" };

    const u = String(unit);

    switch (u) {
        case "W":
            return { device_class: "power", state_class: "measurement",
            ↪  unit_of_measurement: "W" };
        case "V":
            return { device_class: "voltage", state_class: "measurement",
            ↪  unit_of_measurement: "V" };
        case "A":
            return { device_class: "current", state_class: "measurement",
            ↪  unit_of_measurement: "A" };
        case "%":
            return { device_class: "battery", state_class: "measurement",
            ↪  unit_of_measurement: "%" };
        case "Wh":
            return { device_class: "energy", state_class: "total_increasing",
            ↪  unit_of_measurement: "Wh" };
        case "kWh":
            return { device_class: "energy", state_class: "total_increasing",
            ↪  unit_of_measurement: "kWh" };
        default:
            return {
                unit_of_measurement: u,
```

```
                state_class: "measurement"
            };
    }
}

// ================================
// KEYS QUE SÃO SWITCHES
// ================================
const switchKeys = ["relay1_state", "relay2_state"];

// ================================
// FUNÇÃO PARA GERAR UM NOME BONITO
// ================================
function niceName(str) {
    return str
        .replace(/_/g, " ")
        .replace(/\b\w/g, c => c.toUpperCase());
}

// ================================
// BUILD MQTT MESSAGES
// ================================
let mqttMsgs = [];

for (let catKey in devices) {

    const dev = devices[catKey];
    const deviceBase = makeDeviceBase(dev.id, dev.name);

    const categoryObj = victronData[catKey];
    if (!categoryObj) continue;

    for (let pointKey in categoryObj) {
        const entry = categoryObj[pointKey];
        if (!entry) continue;

        // IGNORAR valores inválidos Victron (evita quebras)
        if (entry.value === null || entry.value === undefined) continue;
        if (entry.value === -1 || entry.value === "-1") continue;
```

```
97
98          const safePointId = pointKey.toLowerCase().replace(/[^a-z0-9]/g, "_");
99          const isSwitch = switchKeys.includes(pointKey);
100         const entityType = isSwitch ? "switch" : "sensor";
101         const uniqueId = `${dev.id}_${safePointId}`;
102
103         const discoveryTopic =
            ↪  `homeassistant/${entityType}/${dev.id}/${safePointId}/config`;
104         const stateTopic     =
            ↪  `homeassistant/${entityType}/${dev.id}/${safePointId}/state`;
105         const attributesTopic =
            ↪  `homeassistant/${entityType}/${dev.id}/${safePointId}/attributes`;
106
107         // ================
108         // DISCOVERY PAYLOAD
109         // ================
110         let payload = {
111             name: niceName(pointKey),
112             unique_id: uniqueId,
113             device: deviceBase,
114             state_topic: stateTopic,
115             expire_after: 120
116         };
117
118         if (isSwitch) {
119             payload.command_topic =
                ↪  `homeassistant/switch/${dev.id}/${safePointId}/set`;
120             payload.payload_on  = "1";
121             payload.payload_off = "0";
122         } else {
123             const classes = getDeviceClassAndStateClass(entry.unit);
124
125             if (classes.unit_of_measurement) payload.unit_of_measurement =
                ↪  classes.unit_of_measurement;
126             if (classes.device_class) payload.device_class =
                ↪  classes.device_class;
127             if (classes.state_class) payload.state_class = classes.state_class;
128
129             // Coloca sensores não-energéticos como DIAGNÓSTICO
130             if (!["W","Wh","kWh"].includes(entry.unit)) {
```

```
                payload.entity_category = "diagnostic";
            }
        }

        // ================
        // ATRIBUTOS OPCIONAIS
        // ================
        const attributes = {
            address: entry.address,
            description: entry.description || "",
            unit: entry.unit,
            raw_key: pointKey,
            device_group: catKey
        };

        // Atributos (retain)
        mqttMsgs.push({
            topic: attributesTopic,
            payload: JSON.stringify(attributes),
            qos: 1,
            retain: true
        });

        // Discovery (retain)
        mqttMsgs.push({
            topic: discoveryTopic,
            payload: JSON.stringify(payload),
            qos: 1,
            retain: true
        });

        // Estado atual
        mqttMsgs.push({
            topic: stateTopic,
            payload: String(entry.value),
            qos: 0,
            retain: true
        });
    }
```

```
170 }
171
172 return [mqttMsgs];
```

APÊNDICE G – Código da Função de Escrita Modbus

O código JavaScript a seguir pertence ao nó de função `Escrita Modbus`. Sua finalidade é receber comandos do *Home Assistant* via MQTT e traduzi-lo em uma mensagem específica para o nó `Modbus-Flex-Write`, que efetuará a alteração no registrador correspondente do inversor Fronius.

Código G.2 – Código-fonte da função `Escrita Modbus` para envio de comandos ao inversor Fronius.

```
// ================================================
// FUNÇÃO: ESCREVER MUDANÇAS DO HOME ASSISTANT → MODBUS
// ================================================

// 1. Recupera o objeto fronius do contexto global
const froniusData = global.get('fronius');
if (!froniusData) {
    node.warn("Nenhum dado Fronius encontrado no contexto global.");
    return null;
}

// 2. Regex para extrair informações do tópico (agora captura switch E number)
const topicRegex = /homeassistant\/(switch|number)\/fronius\/([^/]+)\/set/;
const match = msg.topic.match(topicRegex);

if (!match) {
    node.warn(`Tópico não corresponde ao esperado: ${msg.topic}`);
    return null;
}

const entityType = match[1];  // 'switch' ou 'number'
const fullId = match[2];      // ex: "model123_Conn"

// 3. Separa modelo e ponto (antes e depois do "_")
```

```
25 const [model, ...rest] = fullId.split("_");
26 const point = rest.join("_"); // cobre nomes compostos
27
28 // 4. Verifica se modelo existe
29 if (!froniusData[model]) {
30     node.warn(`Modelo ${model} não encontrado em froniusData`);
31     return null;
32 }
33
34 // 5. Busca chave no objeto de forma case-insensitive
35 const keys = Object.keys(froniusData[model]);
36 const realKey = keys.find(k => k.toLowerCase() === point.toLowerCase());
37
38 if (!realKey) {
39     node.warn(`Ponto ${model}.${point} não encontrado em froniusData`);
40     return null;
41 }
42
43 const entry = froniusData[model][realKey];
44
45 // 6. Valida se é writable
46 if (!entry.writable) {
47     node.warn(`Ponto ${model}.${realKey} não é writable`);
48     return null;
49 }
50
51 // 7. Converte valor recebido baseado no tipo de entidade
52 let rawValue;
53
54 if (entityType === 'switch') {
55     // // Para switches: converte "1"/"0" para 1/0
56     // if (msg.payload === "1") {
57     //     rawValue = 1;
58     // } else if (msg.payload === "0") {
59     //     rawValue = 0;
60     // } else {
61     //     node.warn(`Valor inválido para switch ${model}.${realKey}:
    ↪  ${msg.payload}. Esperado "1" ou "0".`);
62     //     return null;
```

```
    // }
    // node.log(`Switch ${model}.${realKey} definido para: ${rawValue}`);
    rawValue = msg.payload

} else {
    // Para numbers: converte para número e aplica scale_factor
    rawValue = Number(msg.payload);
    if (isNaN(rawValue)) {
        node.warn(`Valor inválido recebido para ${model}.${realKey}:
        ↪ ${msg.payload}`);
        return null;
    }

    // Ajusta com scale_factor (se existir)
    if (entry.scale_factor !== undefined) {
        rawValue = Math.round(rawValue * Math.pow(10, -entry.scale_factor));
        node.log(`Valor ajustado com scale_factor ${entry.scale_factor}:
        ↪ ${rawValue}`);
    }
}


// 9. Adiciona metadados para debug e contexto
msg.registerData = {
    name: realKey,
    model: model,
    entityType: entityType,
    originalValue: msg.payload
}

// 8. Monta mensagem Modbus com informações adicionais
msg.payload = {
    value: rawValue,
    fc: 6, // write single register
    unitid: 1,
    address: entry.address -1,
    quantity: 1
};

```

```
100 // msg.registerName = realKey;
101 // msg.model = model;
102 // msg.entityType = entityType;
103 // msg.originalValue = msg.payload; // valor original recebido do HA
104
105 // Se quiser enviar todos os dados do registro, descomente a linha abaixo:
106 // msg.registerData = entry;
107
108 node.log(`Enviando para Modbus: ${model}.${realKey} = ${rawValue} (endereço:
    ↪  ${entry.address})`);
109
110 return msg;
```

APÊNDICE H – Automação de Controle Grid Zero (Home Assistant)

O código apresentado nesta seção corresponde à automação desenvolvida no *Home Assistant*, escrita em linguagem YAML. Esta rotina atua como o controlador central do sistema, sendo responsável por:

1. Ler os sensores de potência da rede e do inversor em tempo real;
2. Executar o algoritmo de cálculo do saldo energético;
3. Determinar o novo limite de potência (*setpoint*);
4. Publicar o comando de atuação via MQTT para o *Node-RED*.

Código H.2 – Código-fonte da automação (YAML) para o controle Grid Zero no Home Assistant.

```yaml
alias: Gerenciador de Potência do Inversor Fronius
description: >
  Gerencia o modo de controle de potência do inversor Fronius (Normal, Limite
  Fixo, Grid Zero e Saída Zero)
triggers:
  - seconds: /15
    trigger: time_pattern
actions:
  - choose:
      - conditions:
          - condition: state
            entity_id: input_select.controle_de_potencia
            state: Normal
        sequence:
          - target:
              entity_id: switch.fronius_inverter_model_123_immediate_controls_
              ↪ wmaxlim_ena
```

```
17            action: switch.turn_off
18      - conditions:
19          - condition: state
20            entity_id: input_select.controle_de_potencia
21            state: Limite Fixo
22        sequence:
23          - target:
24              entity_id: switch.fronius_inverter_model_123_immediate_controls_
                ↪ wmaxlim_ena
25            action: switch.turn_on
26          - target:
27              entity_id: number.fronius_inverter_model_123_immediate_controls_
                ↪ wmaxlimpct
28            data:
29              value: >
30                {{ (states('input_number.limite_manual_de_potencia_do_inversor
                  ↪ ')
31                | float(0) / 3000) * 100 }}
32            action: number.set_value
33      - conditions:
34          - condition: state
35            entity_id: input_select.controle_de_potencia
36            state: Grid Zero
37        sequence:
38          - target:
39              entity_id: switch.fronius_inverter_model_123_immediate_controls_
                ↪ wmaxlim_ena
40            action: switch.turn_on
41          - target:
42              entity_id: number.fronius_inverter_model_123_immediate_controls_
                ↪ wmaxlimpct
43            data:
44              value: >
45                {{ (states('sensor.victron_ac_rede_ac_consumption_l1') |
46                float(0) / 3000) * 100 }}
47            action: number.set_value
48      - conditions:
49          - condition: state
50            entity_id: input_select.controle_de_potencia
```

```
            state: Saida Zero
        sequence:
          - target:
              entity_id: switch.fronius_inverter_model_123_immediate_controls_
              ↪ wmaxlim_ena
            action: switch.turn_on
          - target:
              entity_id: number.fronius_inverter_model_123_immediate_controls_
              ↪ wmaxlimpct
            data:
              value: 0
            action: number.set_value
mode: restart
```