LINHA EFM100

Manual do Usuário

Página intencionalmente deixada em branco

SUMÁRIO

1. SOBRE A DOCUMENTAÇÃO

1.1. Histórico de Revisões

Data	Revisões
27/02/2026	Lançamento da versão 1.1. Nova funcionalidade MQTT “Publish at change of state”. Mudança de ordem dos dados das respostas JSON em MQTT. MQTT: possui campo “device id” para aplicações. MQTT: usuário e senha aumentados tamanho de caracteres de 32 para 64. Buffer de publish MQTT opcional, para até 250 mensagens. Implementado monitor de carga de CPU. MQTT: nova funcionalidade publish tasks. Configurador melhorado para versão C15. MQTT comando “track” marcado para obsolescência, sendo substituído pela nova funcionalidade “Publish at change of state”.
01/10/2025	Atualização de imagens.
09/09/2025	Adicionados “timestamps” nas respostas JSON MQTT. Disponível novo comando “track” para serviços MQTT.
29/08/2025	Versão inicial

1.2. Representação dos Avisos no Manual

Avisos advertindo o usuário sobre riscos a pessoas, acidentes e ao ambiente.

Avisos advertindo o usuário sobre possível danos ou mal funcionamento.

Dicas de uso do dispositivo são identificadas com este aviso.

2. VISÃO GERAL DA LINHA EFM100

O EFM100 é uma solução em forma de hardware para conectar aplicações digitais às variáveis físicas do mundo real.

A comunicação com o EFM100 se dá através de Ethernet com protocolos abertos e requisições simplificadas.

A interface física conta com a robustez dos padrões industriais para sinais.

2.1. Módulos de Funções

Módulos são uma distribuição do produto pronto para uso em aplicações. São distribuídos em formatos construtivos com configurações pré-definidas para as aplicações mais típicas.

Módulo Completo	Art. No.	Descrição
EFM100-XR-B1	100305	Módulo de funções em Ethernet modelo EFM100-XR-B1, montagem em profundidade, formato construtivo B1. Principais características: 4 entradas digitais 24 Vcc opticamente isoladas, 4 saídas a relé contatos NA capacidade 0,5 A, uma porta Ethernet IEEE 802.3. Protocolos: RESTful API e MQTT. Configuração via browser. Alimentação 24Vcc. Tecnologia de conexão: terminal de engate rápido.

Lançamentos de novos módulos podem ocorrer sem aviso prévio.

Observe as normas de segurança e as normas técnicas que envolvam a aplicação final. O propósito do produto faz com que não seja indicado para uso em segurança de pessoas e controles de processos críticos.

2.2. Canais

O conceito de canais na linha de produtos EFM100 serve para identificar facilmente as funções dos módulos para a aplicação do usuário, sem necessitar de programações ou configurações adicionais.

Os canais são somados sequencialmente pelas funções contidas em cada módulo, no sentido crescente da esquerda para a direita. A contagem para qualquer canal se inicia em zero (0).

2.3. Ethernet

2.3.1. Referências

Os modelos de comunicação suportados neste dispositivo estão em conformidade com os respectivos IETF-RFC's de cada tecnologia em Ethernet empregada, inclusive sujeito às mesmas limitações técnicas e operacionais em que cada arquitetura de cabeamento e/ou protocolo esteja submetido.

Referência	Título
RFC 791	Internet Protocol (IP)
RFC 826	Ethernet Address Resolution Protocol (ARP)
RFC 1112	Host extensions for IP Multicasting (IGMPv1)
RFC 2236	Internet Group Management Protocol, Version 2
RFC 768	User Datagram Protocol (UDP)
RFC 793	Transmission Control Protocol (TCP)
RFC 3629	UTF-8, a transformation formato of ISO 10646
RFC 9110	HTTP Semantics
RFC 9111	HTTP Caching
RFC 9112	HTTP/1.1
RFC 7807	Problem Details for HTTPs APIs
RFC 8259	JavaScript Object Notation
ISO/IEC 20922	Information Technology – Message Queuing Telemetry Transport (MQTT) v3.1.1.
OASIS	Organization for the Advanced of Structured Information Standards

2.3.2. Notas de Desempenho

Comunicações em Ethernet são rápidas e confiáveis e valem ressaltar relevâncias técnicas que podem ser críticas na arquitetura de um projeto:

- Comunicações baseadas em TCP possuem um header maior, controle de handshaking e gerenciamento de perda e sequência de pacotes. Naturalmente a latência (tempo) de serviços baseados em TCP tende a ser maior que UDP. São comunicações mais seguras pelo controle de envio e recebimento.
- Comunicações baseadas em UDP possuem um header enxuto, porém sem controle de handshaking, sem confirmação de entrega, com verificação de pacotes mais simples (baseados em checksum). Tendem a ter uma latência ligeiramente menor com o custo de menor segurança de entrega de dados.
- Redes wireless devem ser evitadas em sistemas de automação onde o tempo de escrita e leitura (latência) e a periodicidade tendem a ser mais críticos. Este tempo de resposta tende a aumentar com o número de dispositivos e roteadores com Wifi ligados em um ambiente, mesmo que em redes diferentes.

- A quantidade de dispositivos integrados em uma subrede, bem como a quantidade de switches é crítica para sistemas de baixa latência. Sempre que possível é uma boa prática optar em isolamento de redes.
- Cabos de rede devem ser comprados ou construídos de forma a atender a categoria recomendada (neste caso: CAT5e).
- Ambientes com elevada radiação eletromagnética ou em ambientes fabris requerem uso de cabos blindados e com maior robustez.

O propósito do produto faz com que não seja indicado para uso em segurança de pessoas e controles de processos críticos.

3. INSTRUÇÕES PARA MÓDULOS

3.1. Instruções Gerais

3.1.1. Fixação

Para os módulos, a fixação do módulo depende do formato construtivo do dispositivo. Verifique o capítulo sobre o formato construtivo para obter as dimensões do módulo e a posição da furação para fixação.

3.1.2. Distâncias de Resfriamento

Para resfriamento do dispositivo é recomendado um espaçamento mínimo nas paredes do invólucro. É altamente recomendado seguir as referências a seguir:

Lado	Distância Mínima
Superior	20 mm
Inferior	20 mm
Esquerdo / Direito	0 mm

3.1.3. Cabeamento Ethernet

O cabeamento Ethernet recomendado segue as instruções:

Parâmetro	Valor
Cabeamento mínimo	CAT5e
Padrão de cores	T568A ou T568B
Distância máxima	100 metros
Blindagem	Necessária em ambientes industriais ou com alta interferência eletromagnática

Em ambientes próximos a alta exposição a interferências eletromagnéticas ou ambientes industriais é altamente recomendado o uso de cabos de rede com blindagem FTP ou STP.

3.2. EFM100-XR-B1

O módulo EFM100-XR-B1 é ideal para aplicações com baixa densidade de entradas e saídas. O módulo possui quatro entradas digitais e quatro saídas a relé, com identificação de estado via LEDs, sendo alimentado em 24 Vcc com proteção contra surtos e inversão de polaridade. Conta com uma porta Ethernet IEEE 802.3 para configuração e comunicação em RESful-API e MQTT.

3.2.1. Características

Característica	Valores
Código de produto	EFM100-XR-B1
Número de artigo (ART no.)	100305
Tensão de alimentação	24Vcc
Porta de comunicação	1x Ethernet 802.3 RJ45
Protocolos Ethernet	RESTful API MQTT
Portas de interface IO	4x entradas digitais 4x saídas a relé

3.2.2. Limites Operacionais

Os limites operacionais estabelecem valores no qual o dispositivo não deve operar. A operação fora dos limites causa danos ao produto e consequente perda de garantia.

Característica	Valores
Máxima tensão direta	+28,1 V
Máxima tensão reversa	-39,0 V
Mínima temperatura de operação	-5,0 °C
Máxima temperatura de operação	60,0 °C
Umidade relativa máxima, sem condensação	95 %
Vibração máxima, qualquer eixo	- m/s²
Choque máximo, qualquer eixo	- m/s²

3.2.3. Dados Técnicos Elétricos

Característica	Valores
Tensão nominal de alimentação	24,0 Vcc
Mínima tensão operacional de alimentação[1]	21,6 Vcc
Máxima tensão operacional de alimentação[1]	26,4 Vcc
Corrente nominal (à 24,0 Vcc)	80 mA
Máxima corrente de consumo	200 mA
Fusível especificado	250 mA

1) Operar o dispositivo fora da faixa operacional não garante o pleno funcionamento, podendo causar interrupções e instabilidade.

3.2.4. Dados Técnicos Mecânicos

Característica	Valores
Formato construtivo	B1,
Posição de fixação	Fundo de painel
Modo de fixação	4x parafusos M5
Dimensões do corpo	128 x 100 x 25 mm
Dimensões do perímetro externo	148 x 100 x 25 mm
Peso	120 g
Material	ABS
Grau de proteção	IP20
Sistema de ventilação	Ar, convecção natural

3.2.5. Dados Técnicos da Porta Ethernet

Característica	Valores
Tecnologia de conexão	RJ45
Quantidade de portas	1
Padrão	IEEE 802.3
Cabo recomendado	CAT5e (TIA T568A ou T568B)
Comprimento máximo do cabo	100 metros, a 100 Mbps
Velocidades de comunicação	10 Mbps / 100 Mbps
Direções suportadas	Half e full duplex
Suporte a autonegociação	Sim
DHCP	Sim
Protocolos de comunicação	RESTful API, MQTT
DNS	Sim, para serviço MQTT

Tipo de endereçamento	IPv4

3.2.6. Dados Técnicos das Entradas Digitais

Característica	Valores
Quantidade de entradas	4
Tensão de referência lógica	24 Vcc
Corrente nominal, por entrada	10 mA
Tensão mínima para nível alto	> +7,00 V
Tensão máxima para nível baixo	< +5,00 V
Tensão limite máxima reversa[1]	-50 V
LED identificação	Verde, IEC 61131-1
Tecnologia de conexão	Terminais de engate rápido, borne aparafusável
Fiação recomendada	0,75 mm² (18 AWG)
Fiação suportada	0,35 a 2,50 mm² (21 a 14 AWG)

1) Sujeito a danos irreversíveis.

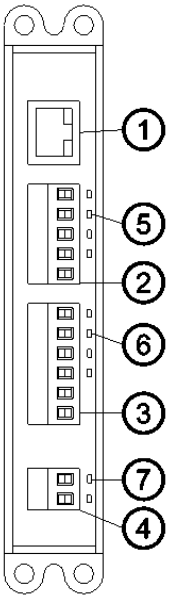
3.2.7. Dados Técnicos das Saídas a Relé

Observe a capacidade de carga de cada saída a relé para evitar danos e preservar sua vida útil.

Característica	Valores
Quantidade de saídas	4
Tipo de contato	Normalmente aberto (N.A.), comum compartilhado
Corrente nominal de chaveamento	0,5 A à 220 Vca
Potência máxima de chaveamento	125 W, cos(φ) = 1
Expectativa de vida dos relés	> 100.000 ciclos
Tempo de acoplamento	10 ms
Tempo de desacoplamento	5 ms
LED identificação	Vermelho, IEC 61131-1
Tecnologia de conexão	Terminais de engate rápido, borne aparafusável
Fiação recomendada	0,75 mm² (18 AWG)
Fiação suportada	0,35 a 2,50 mm² (21 a 14 AWG)

3.2.8. Visão Geral

Vista frontal do módulo de funções Ethernet EFM100-XR-B1:

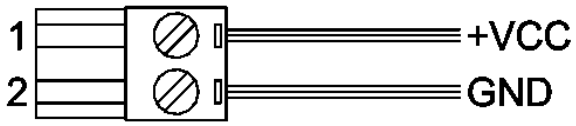


1. Porta Ethernet (RJ45).
2. Conector para entradas digitais.
3. Conector para saídas a relé.
4. Conector para alimentação.
5. LEDs de estado das entradas digitais.
6. LEDs de estado das saídas digitais.
7. LEDs de estado da CPU.

3.2.9. Instalação Elétrica

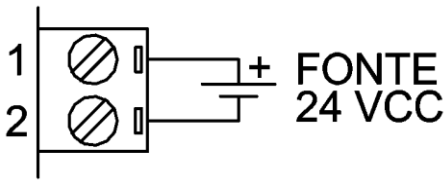
3.2.9.1. Conector de alimentação

Use este conector para alimentar o dispositivo com uma fonte de alimentação 24 Vcc.



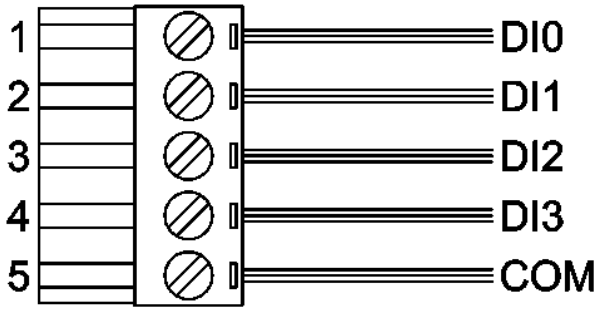
Terminal	Designação	Comentários
1	+VCC	Positivo da fonte de alimentação (+24 V).
2	GND	Negativo da fonte de alimentação (0 V).

O polo positivo da fonte de alimentação deve ser ligado no terminal 1 e o polo negativo da fonte no terminal 2.



3.2.9.2. Conector de Entradas Digitais

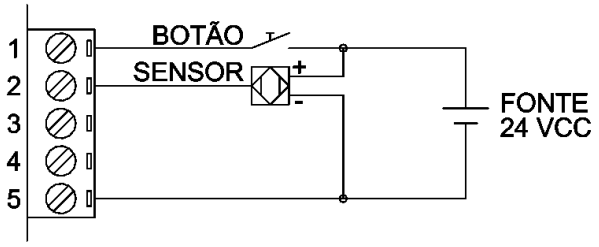
Um conector de engate rápido de cinco vias é usado para fazer a interface das entradas digitais da aplicação com o dispositivo EFM100.



Todos os terminais de entrada digital (DIx) deste dispositivo estão referenciados ao mesmo terminal comum (COM).

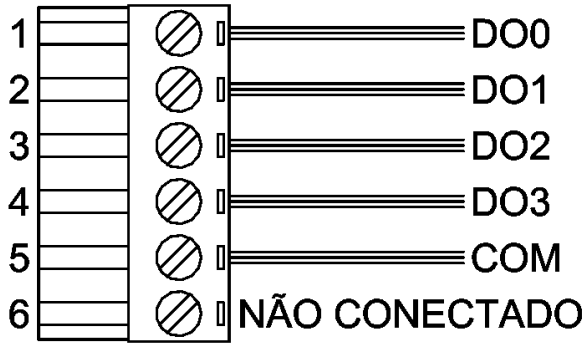
Terminal	Designação	Comentários
1	DI0	Entrada digital, terminal DI0.
2	DI1	Entrada digital, terminal DI1.
3	DI2	Entrada digital, terminal DI2.
4	DI3	Entrada digital, terminal DI3.
5	COM	Terminal comum para as entradas digitais DI0 a DI3.

Os terminais de entrada digitais devem ser ligados aplicando tensão positiva nos terminais DIx em relação ao terminal COM. Vide exemplo abaixo demonstrando uma ligação de um botão no terminal DI0 e um sensor no terminal DI1:



3.2.9.3. Conector de Saídas a Relé

Um conector de engate rápido de seis vias é usado para fazer a interface das saídas à relés da aplicação com o dispositivo EFM100.

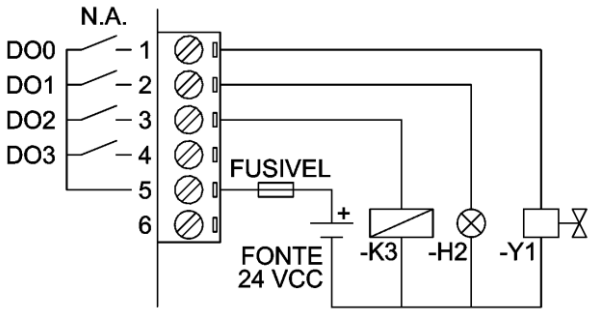


Todos os contatos normalmente abertos (N.A.) de saída à relé (DOx) deste dispositivo estão conectados ao mesmo terminal comum (COM).

Terminal	Designação	Comentários
1	DO0	Saída à relé, terminal DO0.
2	DO1	Saída à relé, terminal DO1.
3	DO2	Saída à relé, terminal DO2.
4	DO3	Saída à relé, terminal DO3.
5	COM	Terminal comum para as saídas à relé DO0 a DO3.
6	N.C.	Não conectado (sem uso).

Geralmente, o terminal 5 (comum) é usado para alimentar os contatos normalmente abertos (N.A.) dos relés internos. Os terminais DO0 a DO3 poderão ser usados para chavear a tensão do terminal 5 para as cargas.

O desenho a seguir ilustra uma ligação exemplo para uma válvula, uma lâmpada e um contator, todos alimentados por 24 Vcc.



Recomenda-se o uso de um fusível externo para o terminal 5.

4. CONFIGURAÇÃO

4.1. Introdução ao EFM100 Configurator

Toda a configuração do EFM100 é feita através de sua interface Ethernet. Acessando o ambiente de configuração chamado “EFM100 Configurator” é possível monitorar e configurar o dispositivo para qualquer aplicação. A tecnologia do EFM Configurator permite:

- Alta portabilidade: qualquer sistema operacional, qualquer navegador.
- Agilidade: acesse e configure no mesmo instante.
- Praticidade: sem necessidade de instalar softwares, sem cabos especiais.
- Visibilidade: domínio total do que está acontecendo no dispositivo.
- Intuitivo: ambiente otimizado e orientado à aplicação do usuário.

4.1.1. Itens Necessários

Para configurar o EFM100 são necessários os seguintes itens:

- O dispositivo EFM100 a ser configurado.
- Um cabo Ethernet.
- Um computador com porta Ethernet e navegador de Internet.

Qualquer navegador de Internet (browser) é apto a acessar o menu de configurações do EFM100.

4.1.2. Conexão

Ligar o computador e o EFM100.

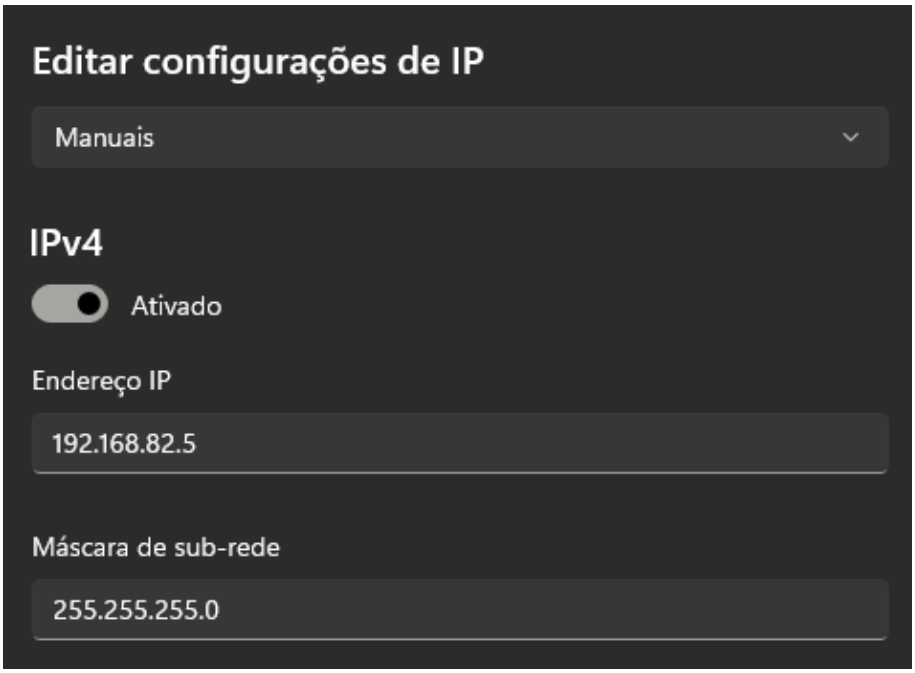
Conectar o cabo de rede entre o EFM100 e o computador. O uso de switches e roteadores, desde que dentro da mesma sub rede, é permitido.

Endereços IP duplicados dentro de uma sub rede não são permitidos. Sempre configure dispositivos individualmente numa rede.

Configurar o computador com a mesma faixa de rede do EFM100. Caso o EFM100 tiver sido ligado pela primeira vez ou recebido um reset de fábrica seu IP será 192.168.82.72.

Exemplo: se o EFM100 estiver no endereço 192.168.82.72, o computador deverá ser configurado com um endereço como 192.168.82.5, por exemplo.

Para configurar o IP estático (manual) de seu computador, acesse os menus de configuração dos dispositivos de rede. Desabilite a opção DHCP e preencha os endereços IP manualmente.



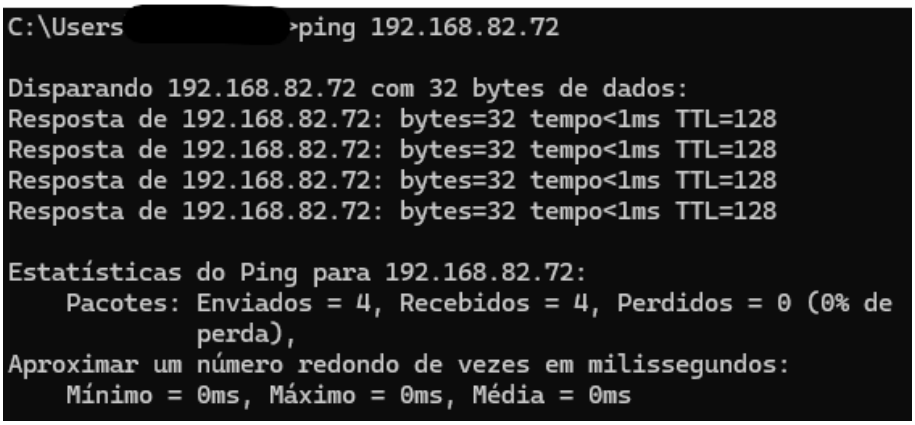
Ao mudar a configuração poderá haver problemas de comunicação se a porta de rede do computador estiver sendo usada para se comunicar com a Internet ou outros dispositivos em rede.

Após mudar a faixa de endereço IP, sugere-se um teste de ping com o EFM100. Abra o Powershell ou cmd (se Windows) ou o terminal (se Linux) e realize um teste de ping do computador para o EFM100.

Comando:

```
ping 192.168.82.72
```

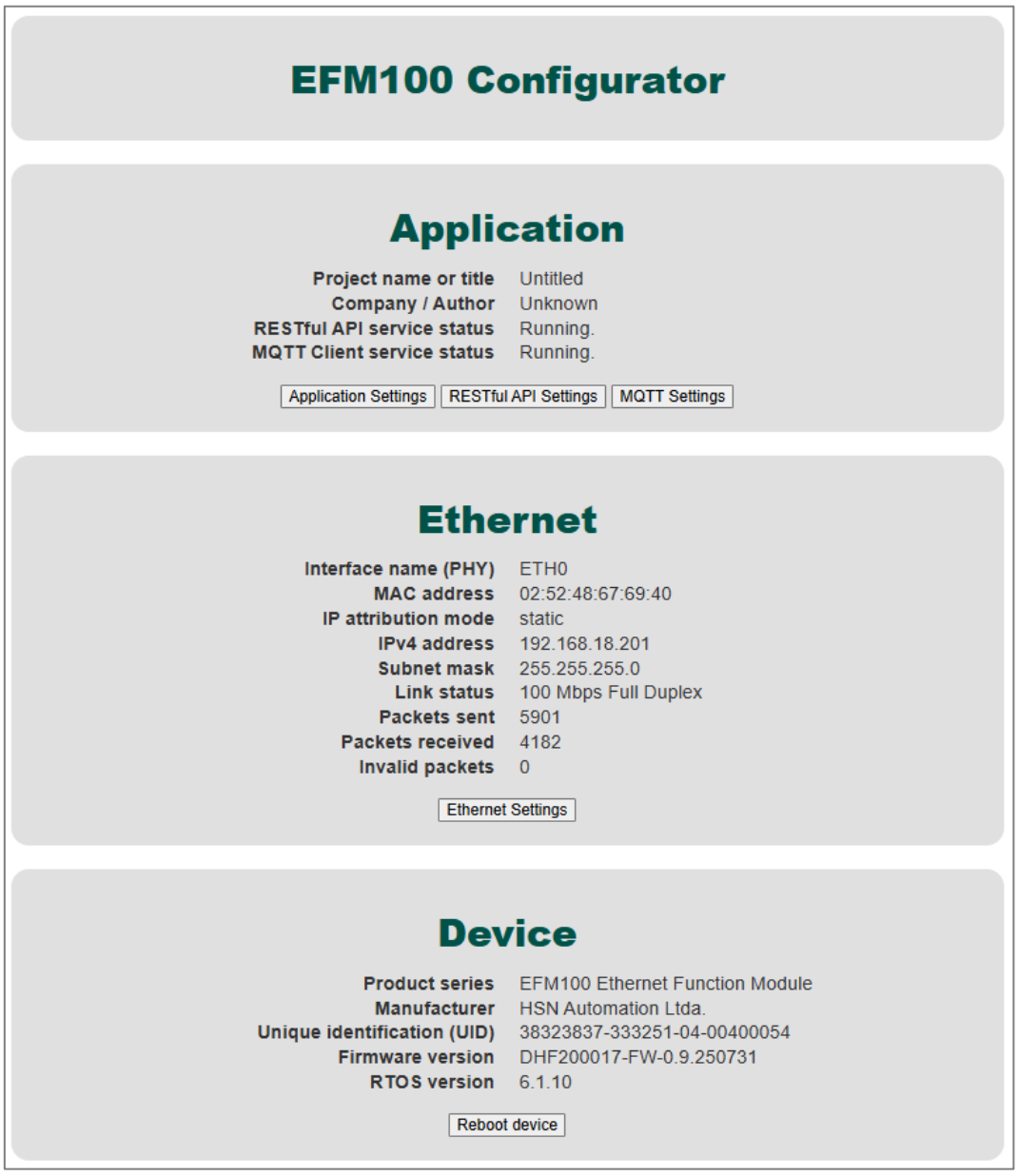
Onde 192.168.82.72 é o endereço IP do EFM100.



O teste deve passar em 100%. Confira o cabo, as configurações de endereço IP do computador e do EFM100, caso não for possível estabelecer uma conexão.

Se o EFM100 tiver sido configurado para obter um endereço IP através de DHCP será necessário obter o IP atribuído a ele através do Servidor DHCP.

Abra o navegador no computador e digite o endereço IP do EFM100.



O navegador abrirá a tela de configuração.

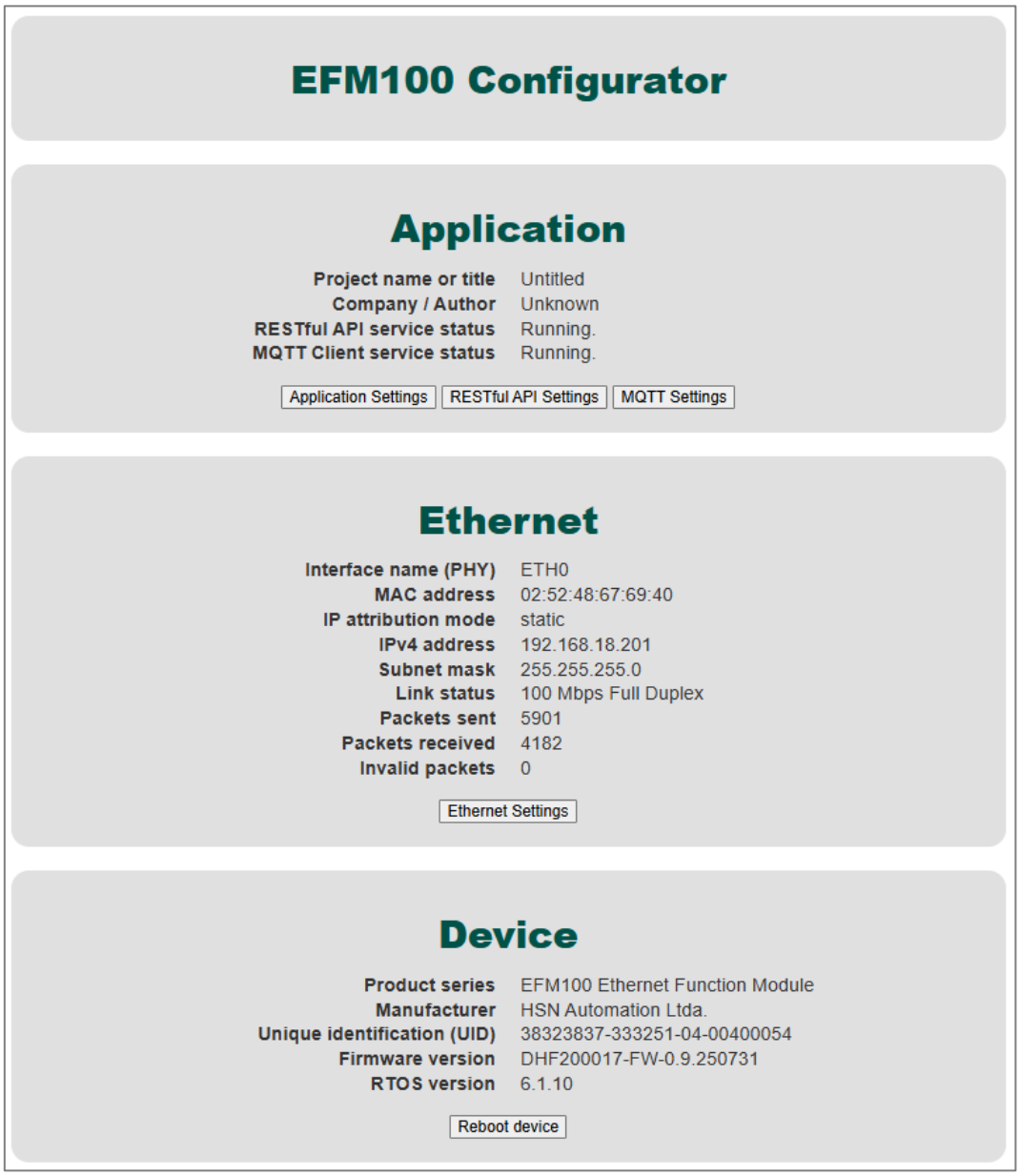
A versão do EFM100 Configurator aparecerá no rodapé.

4.2. Navegação do EFM100 Configurador

O menu principal é acessado normalmente quando digitado o IP do dispositivo no browser ou após realizar alguma ação em uma página de configuração.

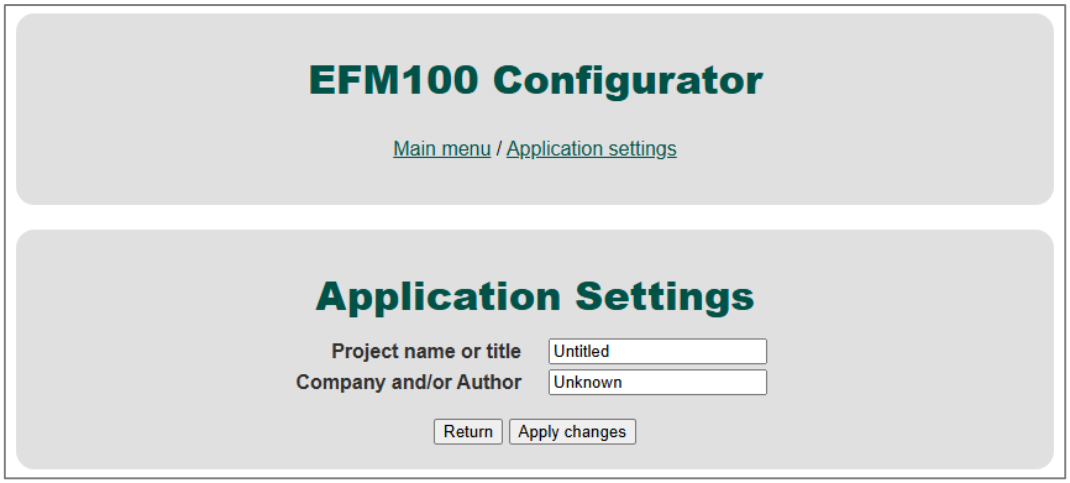
Neste menu o usuário consegue monitorar as principais informações para sua aplicação, como a identificação do projeto, estado dos serviços. Os principais dados da interface Ethernet e da CPU também podem ser monitorados.

O menu principal serve como acesso para outras páginas de monitoramento ou configuração. Para isso, veja os botões distribuídos nesta tela.



4.3. Configuração da Aplicação

Na tela de configuração da aplicação o usuário pode-se entrar detalhes de projeto e autoria para melhor identificar o dispositivo.



4.3.1. Parâmetros de Aplicação

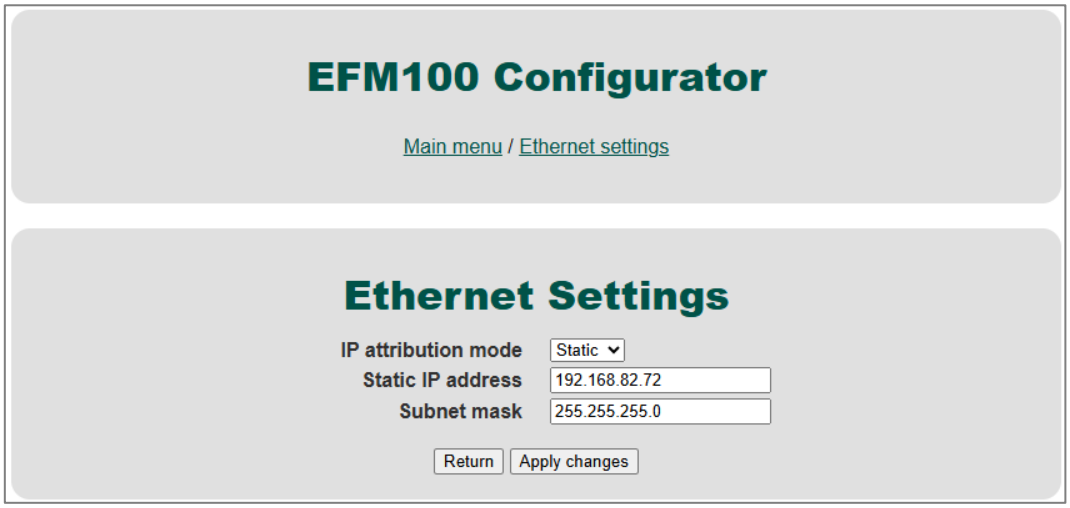
Parâmetro	Descrição
Project name or title	Para fins de identificação. Nome do projeto.
Company and/or author	Para fins de identificação. Nome da empresa ou autor.

Os parâmetros são salvos ao clicar em “Apply Changes”.

Estes parâmetros não necessitam de reinicialização para surtirem efeito.

4.4. Configuração Ethernet

Na tela de configuração de Ethernet é possível atribuir um novo endereço IP para o dispositivo.



4.4.1. Parâmetros de Ethernet

Parâmetro	Descrição
IP attribution mode	Tipo de atribuição de IP. Com “Static” o IP é dado manualmente. Com “DHCP” o IP é dado automaticamente através de um servidor DHCP.
Static IP address	Endereço IP atribuído manualmente.
Subnet mask	Máscara de sub-rede atribuída manualmente.

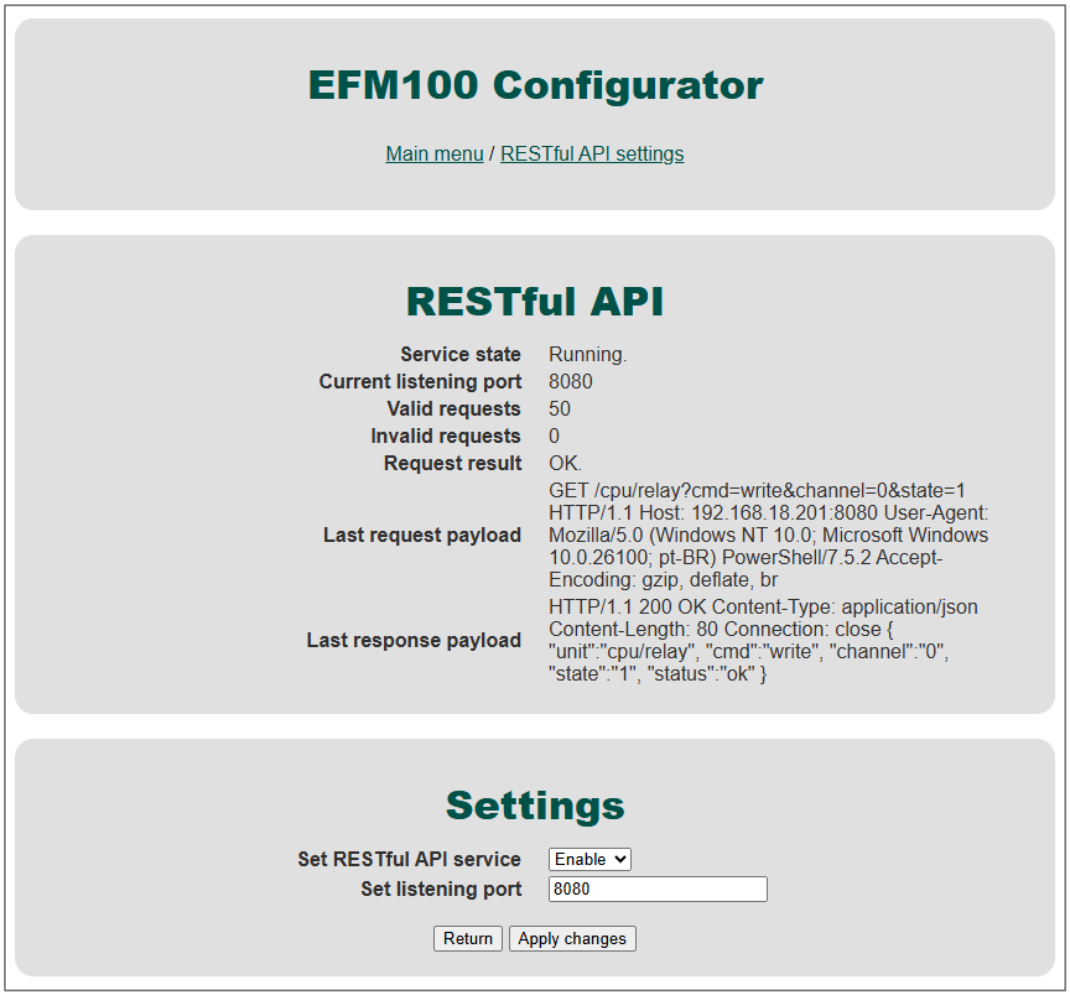
Clicar em “Apply changes” para gravar as alterações.

É necessário que o dispositivo esteja dentro de uma rede gerenciada por servidor DHCP para obter um IP automaticamente pelo modo “DHCP”.

Se o EFM100 tiver sido configurado para obter um endereço IP através de DHCP será necessário obter o IP atribuído a ele através do Servidor DHCP.

4.5. Configuração RESTful API

Permite configurar e monitorar o serviço de RESTful API.



4.5.1. Monitoração do Serviço RESTful API

Parâmetro	Descrição
Service state	Estado atual do serviço.
Current listening port	A porta TCP em que o serviço está operando.
Valid requests	Quantidade de requisições válidas recebidas.
Invalid requests	Quantidade de requisições inválidas recebidas.
Request result	Resultado da última requisição recebida.
Last request payload	A última mensagem de requisição recebida.
Last response payload	A última mensagem de resposta enviada.

4.5.2. Parâmetros de RESTful API

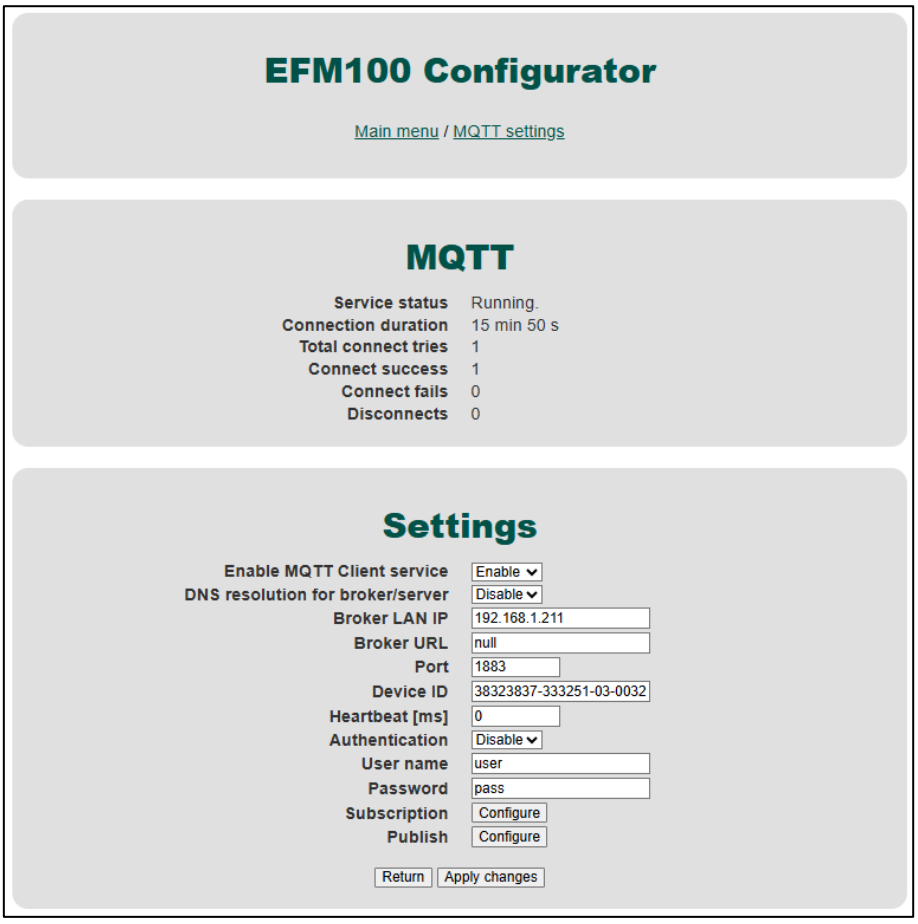
Parâmetro	Descrição
Set RESTful API service	Habilita ou não o serviço de RESTful API para o dispositivo. Basicamente para que a interface REST funcione no dispositivo esta opção deve ser configurada como “Enable”.

Set listening port	A porta utilizada pela API RESTful é de atribuição livre no servidor e não há diretriz da IANA. Todavia, deve-se evitar usar portas reservadas para outros serviços para evitar problemas de funcionamento e conformidade. Na linha EFM100 a porta padrão para API RESTful é 8080.

Configurações são válidas na próxima reinicialização (reenergização).

4.6. Configuração de MQTT

Permite configurar todos os parâmetros para o EFM100 operar no protocolo MQTT.



4.6.1. Monitoração do Serviço MQTT

Parâmetro	Descrição
Service status	Indica qual o estado do serviço MQTT.
Connection duration	Contador de tempo em que a conexão com o broker está ativa na seção atual. É zerado toda vez de uma conexão é interrompida.
Total connect tries	Contagem de tentativas de conexões com o broker.

Connect success	Contagem de conexões com sucesso.
Connect fails	Contagem de conexões falhas.
Disconnects	Contagem de desconexões.

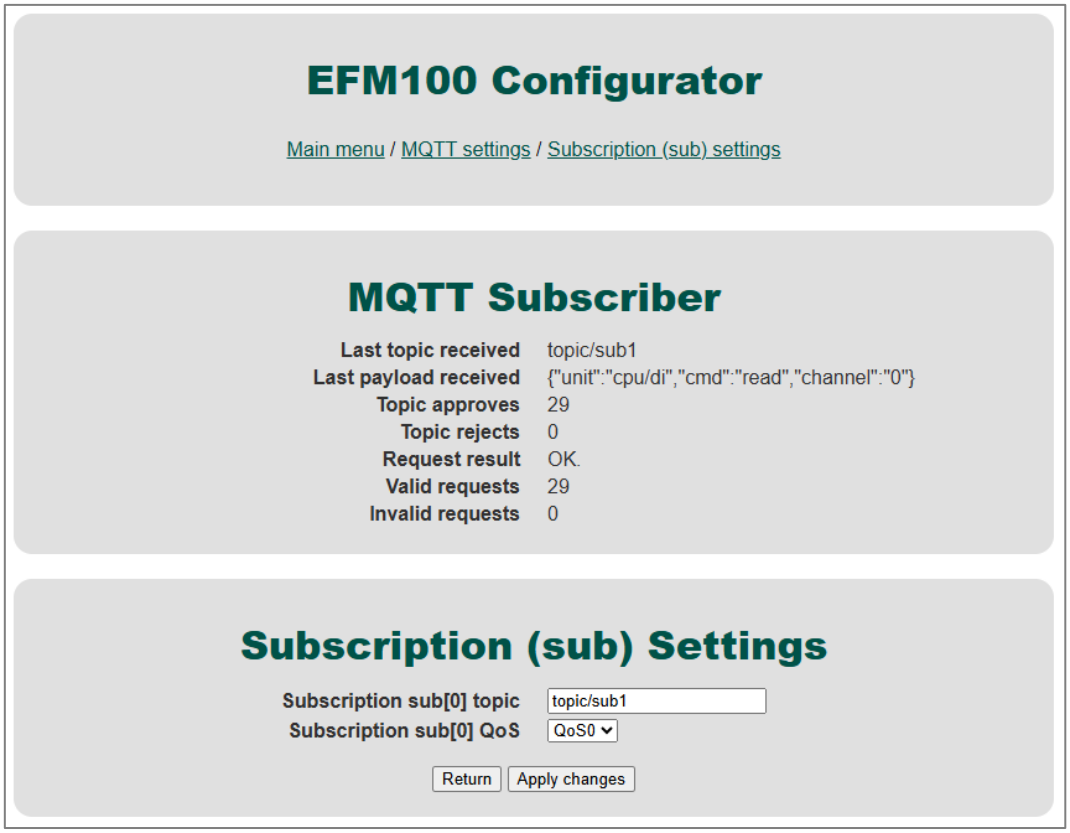
4.6.2. Parâmetros de MQTT

Parâmetro	Descrição
Enable MQTT Client service	Habilita ou não o serviço de cliente MQTT para o dispositivo. Basicamente para que o MQTT funcione no dispositivo esta opção deve ser configurada como “Enable”.
DNS resolution for broker/server	Caso o broker/servidor se encontre dentro de uma URL ou um IP não possível de ser alcançado usando o IPv4 simples (tal como uma aplicação em LAN), será necessário ativar a resolução em DNS para que uma URL seja usada para acessar o dispositivo. Usado para aplicações remotas, WAN ou em nuvem.
Broker LAN IP	Quando a resolução de DNS estiver “Disable” esta opção será usada para se conectar ao broker. Entre com o endereço IPv4 do broker neste campo. Nota: pode ser necessário que seu broker deva estar dentro da mesma sub rede caso o roteador de rede não seja capaz de rotear IPs em faixas diferentes ou realizar operação em NAT.
Broker URL	Quando a resolução de DNS estiver “Enable” esta opção será usada para se conectar ao broker remoto. Um URL será usado para se conectar ao broker.
Port	Número da porta em que o broker está apto a se comunicar. Por definição da IANA, a porta 1883 está alocada para conexões MQTT sem criptografia e a porta 8883 para conexões criptografadas em TLS/SSL. Recomenda-se usar estes valores padrão de portas de acordo com o caso de cada aplicação.
Device ID	A identificação única do dispositivo para conexão ao broker. Configurar conforme a aplicação do usuário. O valor de “Device UID” é carregado como padrão de fábrica.
Heartbeat [ms]	Heartbeat é uma mensagem broadcast periódica que o EFM100 publicará ao broker sinalizando sua existência e conexão. Quando o broker não confirma o recebimento de uma mensagem de heartbeat o EFM100 se desconectará e tentará uma nova conexão em intervalos de tempo. Para desativar o heartbeat selecione valor zero ‘0’ (valor padrão). O valor de heartbeat é o intervalo em milissegundos entre cada mensagem.
Authentication	Habilita ou desabilita a autenticação do Cliente junto ao broker/server. Quando “Enable” é necessário entrar com um usuário e senha que esteja devidamente configurado ao broker.
User name	Quando autenticação estiver ativa, entre com o nome de usuário para logar no broker.
Password	Quando autenticação estiver ativa, entre com a senha para logar no broker.

Após alteração, clique em “Apply changes” para gravar os parâmetros no dispositivo.

Configurações são válidas na próxima reinicialização (reenergização).

Ao clicar “Configure” para “Subscription” será apresentada a página:



4.6.3. Monitoração do Subscriber MQTT

Parâmetro	Descrição
Last topic received	Indica o último tópico recebido.
Last payload received	Último payload recebido.
Topic approves	Quantidade de tópicos recebidos que foram aprovados. Tópicos aprovados são aqueles que são configurados para recebimento de dados.
Topic rejects	Quantidade de tópicos recebidos que foram rejeitados.
Request result	Resultado da última requisição (apenas para tópicos aprovados).
Valid requests	Contagem de requisições válidas.
Invalid requests	Contagem de requisições inválidas.

4.6.4. Parâmetros de Subscriber MQTT

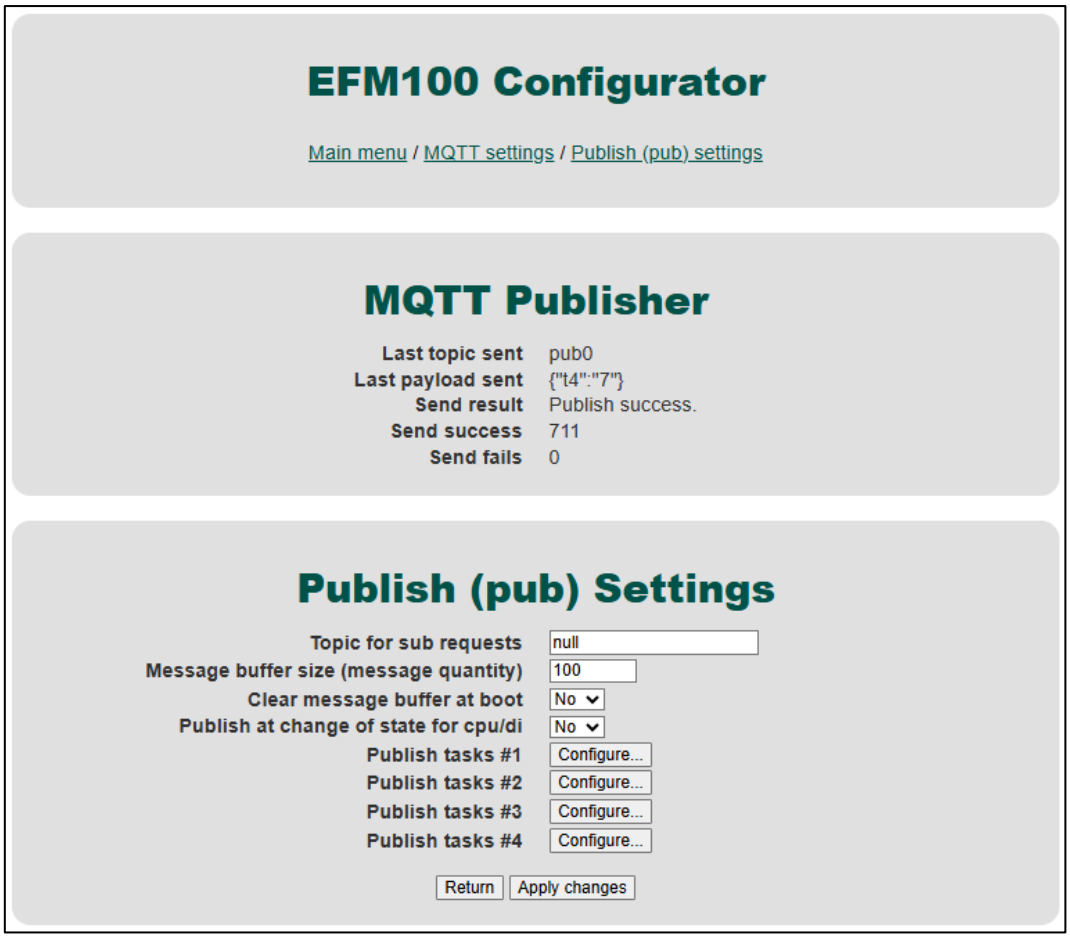
Parâmetro	Descrição

Subscription sub[0] topic	O tópico no qual o dispositivo fará a assinatura para receber dados. Todos os dados vindos do broker ao dispositivo usarão este tópico.
Subscription sub[0] QoS	O QoS (Quality of Service) que o tópico sub[0] usará.

Após alteração, clique em “Apply changes” para gravar os parâmetros no dispositivo.

Configurações são válidas na próxima reinicialização (reenergização).

No menu de configuração geral MQTT, ao clicar “Configure” para “Publish” será apresentada a página:



4.6.5. Monitoração do Publisher MQTT

Parâmetro	Descrição
Last topic sent	Indica o último tópico enviado.
Last payload sent	Último payload enviado.
Send result	Resultado do último envio.
Send success	Quantidade de envios com sucesso.

Send fails	Quantidade de envios falhos.

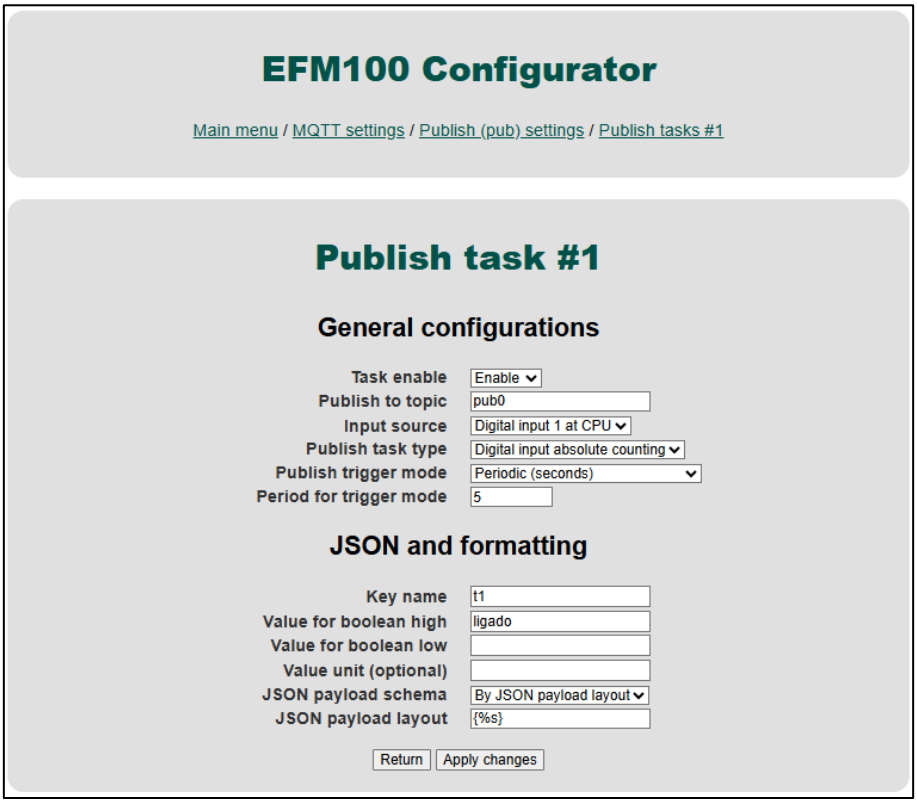
4.6.6. Parâmetros de Publisher MQTT

Parâmetro	Descrição
Publish pub[0] topic	O tópico no qual o dispositivo fará a assinatura para enviar os dados. Todos os dados enviados ao broker pelo dispositivo usarão este tópico.
Publish pub[0] Publish at change of state: cpu/di	Quando ativa fará com que o EFM100 publique uma mensagem informando o estado das entradas digitais da unit cpu/di.
Message buffer size (message quantity)	Se diferente de zero, o buffer de publicação de mensagens será ativo. As mensagens não entregues ao broker por desconexão são armazenadas em uma memória não-volátil e descarregadas na sequência que foram armazenadas ao retorno da conexão com o broker.
Clear message buffer at boot	Se habilitado, o buffer de mensagens a publicar é limpo na reinicialização do dispositivo.
Publish tasks	Configura as tarefas programadas de publicações ao broker através de funções configuráveis.

Buffer de mensagens: deve-se administrar o tamanho do buffer e a taxa de envios para não causar “starvation” na pilha de comunicações e causar falha de buffer cheio (buffer full). É altamente recomendável não ativar o buffer em aplicações que requerem uma alta taxa de envio de mensagens (<1s por mensagem).

4.6.7. Tarefas Programadas de Publicações (Publish Tasks)

Existem quatro instâncias de tarefas programadas que podem ser configuradas pelo usuário para enviar mensagens ao broker dado algum evento de disparo.

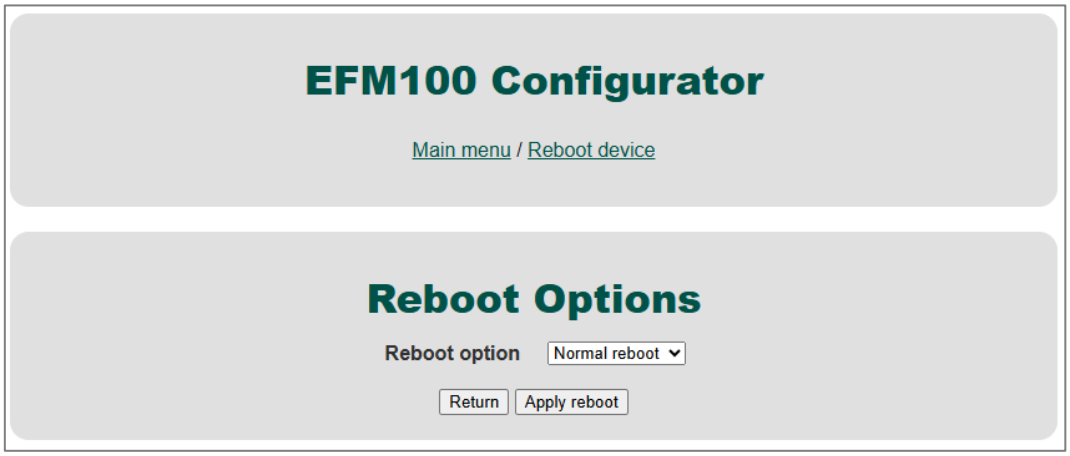


Parâmetro	Descrição
Task enable	Habilita a tarefa
Publish to topic	Topic que será enviada a mensagem
Input source	A entrada física que será usada
Publish task type	Digital input discrete event: usado para registrar o valor da entrada digital para quando estiver ativada ou desativada, usado em combinação com os campos “value for boolean high” e “value for boolean low”. Digital input absolute counting: usado para registrar o valor absoluto de contagem da entrada digital na sessão atual do dispositivo. Digital input relative counting: usado para registrar o valor de contagem desde a mensagem anterior.
Publish trigger mode	Selecione a forma de disparo da mensagem. Pode-se usar as bordas de sinais da entrada digital ou intervalos de tempo (ms, s, min ou h).
Period for trigger mode	O valor de tempo (ms, s, min ou h), quando o modo de disparo da mensagem for por tempo.
Key name	O nome da “key” para o par JSON key/value da mensagem a ser enviada.
Value for boolean high	O valor de “value” para o par JSON key/value quando o tipo de publicação for “discrete event” e a entrada digital estiver ativada.
Value for boolean low	O valor de “value” para o par JSON key/value quando o tipo de publicação for “discrete event” e a entrada digital estiver desativada.
Vlue unit (optional)	Pode-se colocar opcionalmente uma unidade para “value”, por exemplo “peças”. A mensagem possuirá “value”, por exemplo: “123 peças”

JSON payload schema	O esquema de mensagens de payload.
JSON payload layout	Usado para formatar a mensagem a ser enviada. Use o placeholder “%s” para considerar a inserção do par JSON key/value. Exemplo: {“dados”:{%s}} publicará mensagens como: {“dados”:{“contagem1”:”1234”}

4.7. Reinicialização

Nesta tela é possível reiniciar o dispositivo ou carregar os valores de fábrica.



Campos da tela:

- Reboot option: “Normal reboot” para efetuar uma reinicialização simples, “Factory default” para carregar os parâmetros de fábrica no dispositivo.
- Confirm reset factory default: selecione “Yes” para confirmar o carregamento do padrão de fábrica (esta opção aparecerá quando “Reboot option = Yes”).

A opção “Factory default” fará com que todos os parâmetros sejam substituídos pelo padrão de fábrica. A operação não pode ser desfeita.

4.8. Padrão de Fábrica

A configuração de fábrica é um estado de parâmetros no fornecimento do dispositivo ou quando um reset de padrão de fábrica for realizado. Todas as configurações de serviços e aplicações serão restabelecidas para seu valor inicial, incluindo as configurações de IP do dispositivo:

- Atribuição de IP: estático.
- Endereço IPv4 estático: 192.168.82.72.
- Máscara de sub rede: 255.255.255.0.

A configuração de fábrica pode ser carregada de duas formas:

- Através do botão de função no dispositivo.
- Através do configurador do dispositivo (browser).

4.8.1. Carregando Padrão de Fábrica via Botão de Função do Dispositivo

> Antes de realizar a operação de carregamento de padrão de fábrica é recomendável anotar os parâmetros atuais.

Para carregar o padrão de fábrica via botão de função do dispositivo siga o procedimento:

- Desligue o dispositivo, removendo a alimentação.
- Com o dispositivo desligado, mantenha pressionado o botão de função.
- Ligue o dispositivo, com o botão de função pressionado.
- Aguarde até os leds de estado vermelho e verde piscarem bem rápido.
- Libere o botão de função, o dispositivo estará inoperante.
- Desligue o dispositivo.
- Religue o dispositivo, os parâmetros de fábrica serão carregados.

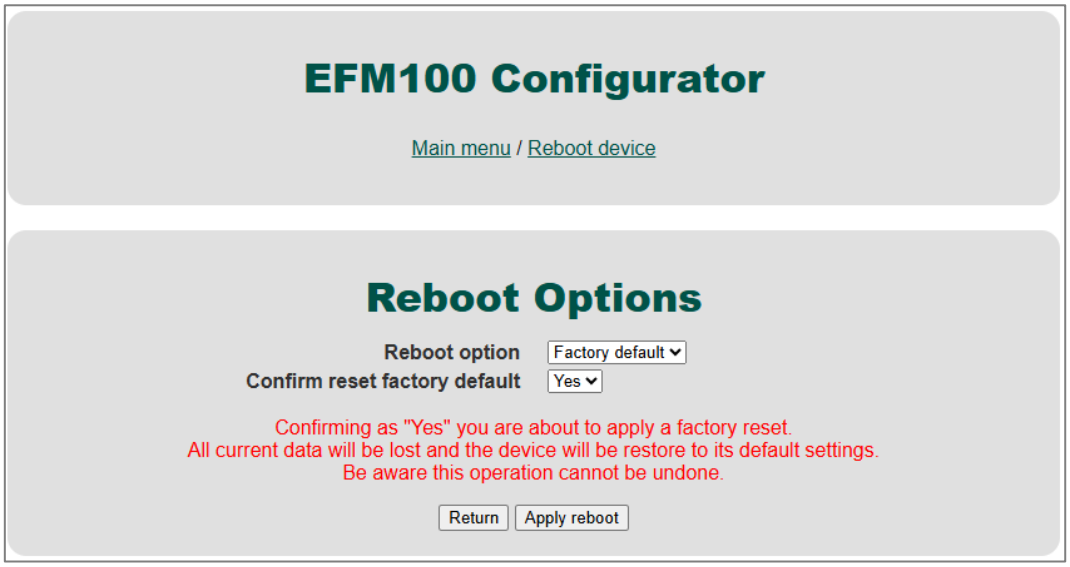
> Todos os parâmetros serão substituídos pelo padrão de fábrica. A operação não pode ser desfeita.

4.8.2. Carregando Padrão de Fábrica via Configurador

> Antes de realizar a operação de carregamento de padrão de fábrica é recomendável anotar os parâmetros atuais.

Para carregar o padrão de fábrica via Configurador do EFM100, acesse o menu de reinicialização: “Main menu / Device reboot”, localizado no submenu “CPU”.

A seguinte tela é exibida:



- Selecione a opção “Reboot option” para “Factory default”.
- Selecione a opção “Confirm reset factory default” para “Yes”.
- Pressione “Confirm and reboot”.
- O dispositivo se reiniciará com os parâmetros de fábrica carregados.

Todos os parâmetros serão substituídos pelo padrão de fábrica. A operação não pode ser desfeita.

5. OPERAÇÃO

5.1. RESTful API

5.1.1. Características

Característica	Descrição
Modelo de comunicação	Servidor
Sessões simultâneas	1
Métodos suportados	GET

5.1.2. Introdução

RESTful API (ou simplesmente API REST) é uma forma de comunicação entre sistemas que segue os princípios do REST (*Representational State Transfer*), um estilo de arquitetura para aplicações web. Ela permite que diferentes sistemas troquem dados e comandos através de requisições HTTP (como `GET`, `POST`, `PUT`, `DELETE`), geralmente usando URLs legíveis e simples.

Uma API REST funciona como uma ponte entre um cliente (por exemplo, um navegador, aplicativo ou sistema de automação) e um servidor (como um dispositivo embarcado, banco de dados ou *backend*). Usa o protocolo HTTP sobre TCP/IP e segue alguns princípios:

- Cliente-servidor: o cliente faz requisições, o servidor responde.
- Sem estado (*stateless*): cada requisição deve conter todas as informações necessárias; o servidor não armazena o estado da sessão entre requisições.
- Interface uniforme: URLs bem definidas e consistentes (ex: /api/dispositivos/1).
- Uso de métodos HTTP:
  - GET: ler dados
  - POST: criar ou executar uma ação
  - PUT: atualizar
  - DELETE: remover

A estrutura de URL para requisições REST são simples, como o exemplo a seguir:

```
http://192.168.52.48:8080/mod/di?cmd=read&channel=3
```

Podem ser divididas em cinco partes, como ilustrado na tabela:

Parte da URL	Significado
http://	Protocolo usado (neste caso, HTTP sobre TCP)
192.168.52.48	Endereço IP do dispositivo (servidor RESTful API)
:8080	Porta TCP onde a API está escutando

mod/di	Caminho do recurso / unit (endpoint)
?cmd=read&channel=3	Parâmetros da requisição (query string)

URL é *Uniform Resource Locator*, ou Localizador Uniforme de Recursos. É o endereço completo que é entrado em um navegador ou usado por um sistema para acessar um recurso na internet ou em uma rede local.

Os módulos de processamento da série EFM100 possuem interface nativa para RESTful API. Com ela é possível obter escrever e ler os dados de processo dos módulos, realizar leituras de estado e configurações do próprio dispositivo.

O EFM100 desempenha o papel de servidor RESTful API, enquanto a aplicação do usuário desempenha o papel de cliente.

A aplicação do usuário envia métodos HTTP do tipo GET ou POST para realizar a leitura ou escrita, respectivamente, no dispositivo e seus módulos.

As mensagens URL em RESTful API para o EFM100, em geral, seguem o seguinte corpo de mensagem de exemplo:

```
http://<ip>:<porta>/<endpoint>/<func>?<parametros>
```

Onde:

- <ip>: IP do EFM100 (recomenda-se usar IP estático).
- <porta>: Porta da aplicação RESTful API, padrão 8080.
- <endpoint>: Endpoint ou unidade destino.
- <func>: Função a ser executada.
- <parametros>: De acordo com a função.

Sugere-se conferir os capítulos das funções de aplicação para mais detalhes do corpo de mensagem.

URLs são case sensitive por padrão, ou seja, há diferenciação de caracteres minúsculos e maiúsculos.

Presença de espaços “ “ na URL de instruções não são tolerados.

Use sempre caracteres minúsculos nas aplicações para garantir compatibilidade, a menos que especificado o contrário.

5.1.3. Configuração

As configurações são feitas através do Configurador do EFM100, acessando via web browser. Veja o capítulo “Configurações” neste manual para maiores detalhes.

5.1.4. Comandos para Entradas Digitais

5.1.4.1. Leitura de Estado Lógico

Efetua a leitura simples dos estados de um canal de entrada digital em uma CPU ou módulo.

- Método: GET
- Endpoint: cpu/di | mod/di
- Parâmetros: cmd, channel
    - cmd: “read”.
    - channel: número inteiro positivo, o primeiro canal é ‘0’.

Corpo da URL para comando em uma CPU:

```
http://<ip>:<porta>/cpu/di?cmd=read&channel=#
```

Corpo da URL para comando em um módulo:

```
http://<ip>:<porta>/mod/di?cmd=read&channel=#
```

Exemplo: leitura do primeiro canal de entrada digital em um módulo, considerando que o EFM100 está no IP 192.168.1.20:

http://192.168.1.20:8080/mod/di?cmd=read&channel=0

A resposta com sucesso da requisição:

{
“unit”:”cpu/di”,
“cmd”:”read”,
“channel”:”0”,
“state”:”3”,
“status”:”ok”
}

As entradas digitais 1 e 2 do primeiro canal estão em nível lógico alto, pois state:3 corresponde à 0b00000011.

5.1.5. Comandos para Saída a Relé

5.1.5.1. Leitura de Estado Lógico

Efetua a leitura simples dos estados lógicos de um canal de saída à relé.

- Método: GET
- Endpoint: cpu/relay | mod/relay
- Parâmetros: cmd, channel
  - cmd: “read”.
  - channel: número inteiro positivo, o primeiro canal é ‘0’.

Corpo da URL para comando em uma CPU:

```
http://<ip>:<porta>/cpu/relay?cmd=read&channel=#
```

Corpo da URL para comando em um módulo:

```
http://<ip>:<porta>/mod/relay?cmd=read&channel=#
```

Exemplo: leitura do segundo canal de saída à relé em uma série de módulos:

http://192.168.1.20:8080/mod/relay?cmd=read&channel=1

A resposta com sucesso da requisição:

```
{
 “unit”:”mod/relay”,
 “cmd”:”read”,
 “channel”:”1”,
 “state”:”123”,
 “status”:”ok”
}
```

No segundo canal de saídas à relé, as saídas 1, 2, 4, 5, 6, e 7 estão ligadas. Ou seja, “state”:”123” corresponde à 0b01111011.

5.1.5.2. Escrita de Estado Lógico

Efetua a escrita de estados lógicos para todas as saídas em um canal de saída a relé.

- Método: GET
- Endpoint: cpu/relay | mod/relay
- Parâmetros: cmd, channel, state
  - cmd: “write”
  - channel: número inteiro, onde o primeiro canal é ‘0’.
  - state: número inteiro em formato binário, hex ou decimal, de 0 a 255.

Corpo da URL para comando em uma CPU:

`http://<ip>:<porta>/cpu/relay?cmd=write&channel=#&state=#`

Corpo da URL para comando em um módulo:

`http://<ip>:<porta>/mod/relay?cmd=write&channel=#&state=#`

Exemplo: ligar a saída a relé 2 e desligar as demais, no terceiro canal em módulos:

http://192.168.1.20:8080/mod/relay?cmd=write&channel=2&state=2

A resposta com sucesso da requisição:

```
{
 “unit”:”mod/relay”,
 “cmd”:”write”,
 “channel”:”2”,
 “state”:”2”,
 “status”:”ok”,
}
```

Operações de escrita tem a maior prioridade na atualização dos estados das saídas à relé. Se uma operação de pulso estiver sendo processada ela será cancelada e substituída.

5.1.5.3. Ligar uma Seleção de Saídas

Seta os estados lógicos de uma seleção de saídas à relé, não alterando o estado lógico das outras. As saídas selecionadas com o valor ‘1’ são alteradas para ON.

- Método: GET
- Endpoint: cpu/relay | mod/relay
- Parâmetros: cmd, channel, mask
  - cmd: “turn_on”.
  - channel: número inteiro, onde o primeiro canal é ‘0’.
  - mask: número em formato binário, hex ou decimal, de 0 a 255, representando apenas as saídas que serão ligadas.

Corpo da URL para comando em uma CPU:

`http://<ip>:<porta>/cpu/relay?cmd=turn_on&channel=#&mask=#`

Corpo da URL para comando em um módulo:

`http://<ip>:<porta>/mod/relay?cmd=turn_on&channel=#&mask=#`

Exemplo: ligar apenas a saída 4 do primeiro canal de saída a relé de um módulo, não alterando as demais.

http://192.168.1.20:8080/mod/relay?cmd=turn_on&channel=0&mask=8

A resposta com sucesso da requisição:

{
 “unit”:”mod/relay”,
 “cmd”:”turn_on”,
 “channel”:”0”,
 “mask”:”8”,
 “status”:”ok”
}

5.1.5.4. Desligar uma Seleção de Saídas

Reseta os estados lógicos de uma seleção de saídas à relé, não alterando o estado lógico das outras. As saídas selecionadas com o valor ‘1’ são alteradas para OFF.

- Método: GET
- Endpoint: cpu/relay | mod/relay
- Parâmetros: cmd, channel, mask
  - cmd: “turn_off”.
  - channel: número inteiro, onde o primeiro canal é ‘0’.
  - mask: número em formato binário, hex ou decimal, de 0 a 255, representando apenas as saídas que serão desligadas.

Corpo da URL para comando em uma CPU:

`http://<ip>:<porta>/cpu/relay?cmd=turn_off&channel=#&mask=#`

Corpo da URL para comando em um módulo:

`http://<ip>:<porta>/mod/relay?cmd=turn_off&channel=#&mask=#`

Exemplo: desligar apenas a saída 4 do primeiro canal de saída a relé de uma CPU, não alterando as demais.

http://192.168.1.20:8080/cpu/relay?cmd=turn_off&channel=0&mask=8

A resposta com sucesso da requisição:

{
 “unit”:”cpu/relay”,
 “cmd”:”turn_off”,
 “channel”:”0”,
 “mask”:”8”,
 “status”=ok
}

5.1.5.5. Pulsar ON uma Seleção de Saídas

Pulsa OFF-ON-OFF uma seleção de saídas à relé, não alterando o estado lógico das outras. As saídas selecionadas com o valor ‘1’ serão pulsadas.

- Método: GET
- Endpoint: cpu/relay | mod/relay
- Parâmetros: cmd, channel, mask, duration
  - cmd: “pulse_on”.
  - channel: número inteiro, onde o primeiro canal é ‘0’.
  - mask: número em formato binário, hex ou decimal, de 0 a 255, representando apenas as saídas que serão pulsadas.
  - duration: tempo em milissegundos que o pulso ocorrerá.

Corpo da URL para comando em uma CPU:

```
http://<ip>:<porta>/cpu/relay?cmd=pulse_on&channel=#&mask=#&duration=#
```

Corpo da URL para comando em um módulo:

```
http://<ip>:<porta>/mod/relay?cmd=pulse_on&channel=#&mask=#&duration=#
```

Exemplo: pulsar ON apenas a saída 4 do primeiro canal de saída a relé numa CPU, durante um segundo, não alterando as demais saídas.

http://192.168.1.20:8080/cpu/relay?cmd=pulse_on&channel=0&mask=8&duration=1000

A resposta com sucesso da requisição:

```
{
 “unit”:”cpu/relay”,
 “cmd”:”pulse_on”,
 “channel”:”0”,
 “mask”:”8”,
 “duration”:”1000”,
 “status”:”ok”
}
```

Se um comando de pulso estiver em processo ele será substituído pelo novo.

5.1.5.6. Pulsar OFF uma Seleção de Saídas

Pulsa ON-OFF-ON uma seleção de saídas à relé, não alterando o estado lógico das outras. As saídas selecionadas com o valor ‘1’ serão pulsadas.

- Método: GET

- Endpoint: cpu/relay | mod/relay
- Parâmetros: cmd, channel, mask, duration
    - cmd: “pulse_off”.
    - channel: número inteiro, onde o primeiro canal é ‘0’.
    - mask: número em formato binário, hex ou decimal, de 0 a 255, representando apenas as saídas que serão pulsadas.
    - duration: tempo em milissegundos que o pulso ocorrerá.

Corpo da URL para comando em uma CPU:

```
http://<ip>:<porta>/cpu/relay?cmd=pulse_off&channel=#&mask=#&duration=#
```

Corpo da URL para comando em um módulo:

```
http://<ip>:<porta>/mod/relay?cmd=pulse_off&channel=#&mask=#&duration=#
```

Exemplo: pulsar OFF apenas a saída 4 do primeiro canal de saída a relé da CPU, durante um segundo, não alterando as demais saídas.

http://192.168.1.20:8080/cpu/relay?cmd=pulse_off&channel=0&mask=8&duration=1000

A resposta com sucesso da requisição:

{
“unit”:”cpu/relay”,
“cmd”:”pulse_off”,
“channel”:”0”,
“mask”:”8”,
“duration”:”1000”,
“status”:”ok”
}

Se um comando de pulso estiver em processo ele será substituído pelo novo.

5.2. MQTT

5.2.1. Características

Característica	Descrição
Modelo de comunicação	MQTT Device
Sessões simultâneas	1
Subscribers	1
Publishers	1
Quality of Service	QoS1 e QoS2

Autenticação	Suportado
Criptografia	Não suportado

5.2.2. Introdução

MQTT (Message Queuing Telemetry Transport) é um protocolo de comunicação leve e eficiente, ideal para a troca de mensagens entre dispositivos em redes com pouca ou restrição de largura de banda — como em automação industrial, IoT (Internet das Coisas), sensores remotos, etc.

Característica	Descrição
Leve	Baixo uso de banda e energia. Ideal para dispositivos embarcados.
Assíncrono	Usa o modelo publish/subscribe. Desacopla remetente e receptor.
Suporte a QoS	Três níveis de qualidade de serviço (0, 1 e 2).
Retenção	Broker pode reter a última mensagem de um tópico.
Autenticação	Pode usar usuários/senhas.
Segurança	Pode usar criptografia por TLS.

Os serviços em MQTT no EFM100 foram concebidos para uma implementação enxuta, robusta, mantendo o conceito de implementação portável, ágil e simples.

A autenticação no MQTT é o processo pelo qual o broker (servidor) verifica se um cliente (publicador ou assinante) tem permissão para se conectar e trocar mensagens. Existem dois tipos de autenticação:

- 1. Autenticação com usuário e senha (mais comum)
  - O cliente envia username e password na conexão.
  - O broker verifica e autoriza ou recusa a conexão.
- 2. Autenticação por certificado digital (TLS/SSL)
  - O cliente se autentica usando um certificado X.509.
  - Alta segurança, usada em aplicações bancárias.

A função de autenticação no EFM100, assim como no padrão MQTT, é opcional.

Na versão atual não há suporte para autenticação por certificado digital (TLS/SSL).

O QoS no MQTT serve para equilibrar entre confiabilidade e desempenho, dependendo das necessidades da aplicação. Em redes instáveis ou com requisitos críticos, o QoS garante que as mensagens não sejam perdidas, duplicadas ou ignoradas.

Nível	Nome técnico	Garantia de entrega	Uso típico
0	At most once	A mensagem é enviada uma vez sem confirmação. Pode se perder.	Aplicações onde perder dados não é crítico (ex: temperatura em tempo real).

1	At least once	A mensagem é entregue pelo menos uma vez. Pode ser duplicada.	Quando a entrega é importante, mas duplicação pode ser tolerada.
2	Exactly once	A mensagem é entregue uma única vez e sem duplicação. Usa handshake.	Aplicações críticas, como controle financeiro, comandos de máquina.

O EFM100 suporta QoS 0 e 1.

5.2.3. Configuração

As configurações são feitas através do Configurador do EFM100, acessando via web browser. Veja o capítulo “Configurações” neste manual para maiores detalhes.

5.2.4. Mensagens JSON

No EFM100 mensagens em MQTT são recebidas e enviadas em formato JSON onde há um tópico para recebimento de dados (sub) e de envio de dados (pub), que são salvos nos parâmetros do dispositivo para o funcionamento de escrita e leitura dos dados de I/O.

A aplicação do usuário formatará os dados a serem publicados ao broker no qual os dispositivos EFM100 se encontram. Com o formato JSON a implementação da escrita e leitura dos dados nos dispositivos é potencialmente simplificada e padronizada.

A posição de cada elemento key/value no JSON enviado é importante. Os elementos key/value do objeto JSON são case sensitive.

Duas chaves (keys) são mandatórias: “unit” e “cmd”.

Sugere-se atenção ao preencher os dados de objeto JSON: usar minúsculas, as chaves e valores (keys e values) sempre dentro de chaves (“ “) e presença de espaços indesejados. Mensagens fora do formato JSON poderão ser silenciosamente descartadas.

5.2.4.1. Key: “unit”

A chave “unit” dentro de um objeto JSON aponta para qual unidade o comando está sendo direcionado. Basicamente existem duas units principais, que são os hardwares em si (cpu “cpu” ou módulo “mod”) seguido por “/” a função que se deseja operar, podendo ser várias (exemplo: entradas digitais “di”, saída a relé “relay”).

Exemplos	Descrição
“cpu/di”	Entrada digital localizada na CPU
“cpu/relay”	Saída a relé localizada na CPU
“mod/di”	Entrada digital localizada num módulo de função.
“mod/do”	Saída digital localizada num módulo de função.

“mod/ai”	Entrada analógica localizada num módulo de função.
“mod/ao”	Saída analógica localizada num módulo de função.
“mod/relay”	Saída a relé localizada num módulo de função

5.2.4.2. Key: “cmd”

A chave “cmd” em um objeto JSON indica qual a operação que se deseja realizar na “unit”.

Exemplos	Descrição
“read”	Realiza a leitura de dados de um módulo ou unidade de processamento
“write”	Realiza a leitura de dados ou estados de um módulo ou CPU.
“turn_on”	Liga uma seleção de saídas ou relés
“turn_off”	Desliga uma seleção de saídas ou relés.
“pulse_on”	Pulsa off-on-off uma seleção de saídas ou relés.
“pulse_off”	Pulsa on-off-on uma seleção de saídas ou relés.
“track”	Cria uma função de rastreamento de sinais.

É necessário que o “cmd” seja compatível com a respectiva “unit” para que o comando seja cumprido com sucesso. Confira os capítulos de operação de sinais de entradas e saídas para obter as combinações possíveis.

5.2.4.3. Key: “channel”

Os canais são usados para identificar a posição de uma função dos módulos ou unidade de processamento.

Podem agrupar sinais de entradas digitais, representar uma entrada de temperatura ou uma saída analógica. Para mais informações, leia o Item “Canais” neste manual e o respectivo módulo de funções que se deseja operar.

Usa-se um valor numérico inteiro positivo para representar a posição do canal, iniciando-se sempre em zero (0) para o primeiro canal.

A tentativa de acessar um canal inexistente retornará um erro.

5.2.4.4. Key: “state”

A chave “state” representa o valor dos estados booleanos para entradas e saídas digitais ou relés. O número em formato decimal convertido para booleano representa a forma com que os sinais físicos estão dispostos em um canal de entrada ou saída.

O “state” geralmente sobrescreve todos os valores de um canal, para um comando de escrita, por exemplo.

A seguir, uma tabela exemplificando o uso da chave “state”:

“state” decimal	“state” binário	Descrição
0	0b00000000	Nenhum bit está ativo. Nenhum terminal do canal de entrada ou saída está ativo
1	0b00000001	Apenas um bit está ativo (bit 0). O primeiro terminal do canal de entrada (DI1) ou saída (DO1, Relay1) está ativo.
3	0b00000011	Dois bits ativos. A entrada digital (DI1 e DI2) ou saídas (DO1 e DO2) estão ativas.
16	0b00010000	Somente a entrada digital DI5 ativa, ou somente a saída digital DO5 ativa.
17	0b00010001	As entradas DI1 e DI5 estão ativas, ou as saídas DO1 e DO5 estão ativas.
255	0b11111111	Todos os bits ativos. Todas as entradas digitais ou saídas digitais ativas.

5.2.4.5. Key: “mask”

A chave “mask” representa estados booleanos para entradas e saídas digitais ou relés, de forma similar a “state”. O número em formato decimal convertido para booleano representa a forma com que os sinais físicos estão dispostos em um canal de entrada ou saída.

A diferença principal entre “mask” sobre “state” é que ela é usada apenas para algumas funções de comandos de escrita e não sobrescreve todos os booleanos de um canal, apenas os selecionados.

A seguir, uma tabela exemplificando o uso da chave “mask”:

“state” decimal	“state” binário	Descrição
0	0b00000000	Nenhum booleano do canal será alterado.
1	0b00000001	Apenas um terminal (primeiro) do canal selecionado será alterado.
3	0b00000011	O primeiro e segundo terminal serão alterados.
16	0b00010000	Somente o quinto terminal do canal selecionado será alterado.
17	0b00010001	O primeiro e quinto terminal serão alterados.
255	0b11111111	Todos os terminais serão alterados.

5.2.4.6. Key: “timestamp”

Também conhecido como estampa de tempo, é um valor em milissegundos desde o início da sessão atual do dispositivo. Por exemplo:

“timestamp”:”12345”

Significa que a mensagem foi processada no instante 12,345 segundos desde o início da sessão atual.

5.2.4.7. Key: “function”

> Marcado descontinuidade a partir da versão 1.1.

Usado em conjunto com o comando “cmd”:”track”. Descreve qual a função que o comando track deverá desempenhar.

Function	Descrição
change of state	Uma publicação será enviada toda vez que a unit e channel tiverem uma mudança de seu estado lógico.

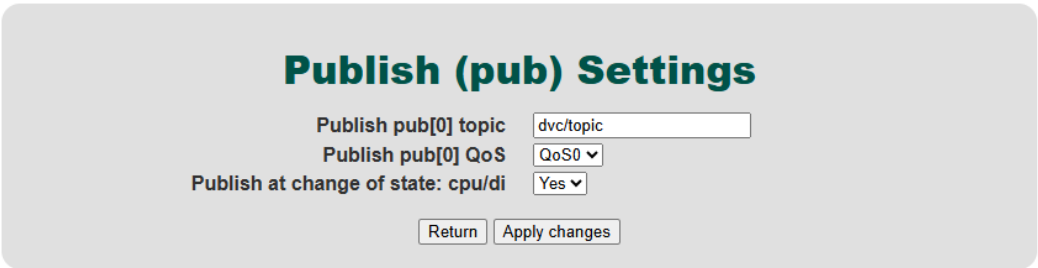
5.2.5. Publicações Automáticas

5.2.5.1. Publish at Change of State

> Disponível a partir da versão 1.1.

Esta função fará uma publicação automática no tópico configurado toda vez que um ou mais terminais de entrada digital de um canal mudarem de estado.

Através do EFM100 Configurator é possível habilitar a função de publicação a troca de estado de um canal de entradas digitais.



Toda vez que mudar o estado lógico do canal ativo, a resposta do dispositivo será, por exemplo:

```
{
  “device id”:”name”,
```

```
  “timestamp”:”12345”,
  “unit”:”cpu/di”,
  “cmd”:”change of state”,
  “channel”:”0”,
  “state”:”5”,
  “status”:”ok”
}
```

Onde “state” representa, em formato decimal, o valor das entradas digitais.

5.2.5.2. Heartbeat

É possível fazer com que o EFM100 publique periodicamente mensagem de sua existência e conexão com o broker com uma mensagem de heartbeat. Esta opção pode ser habilitada na página de configuração de MQTT.

Quando um valor diferente de zero é configurado, o EFM100 publicará mensagens no tópico `efm100/broadcast` com o seguinte formato JSON:

```
{
  “device id”:”name”,
  “timestamp”:”12345”,
  “heartbeat”:”2000”
}
```

- `device_id:` Device ID, valor editado no menu de configuração MQTT.
- `timestamp:` tempo da sessão atual, em milissegundos, e que a mensagem foi enviada;
- `heartbeat:` indica o período, em milissegundos, que cada heartbeat é enviado.

A aplicação nativa do EFM100 silenciosamente descarta o tópico efm100/broadcast. O recebimento deste tópico não é computado nas estatísticas de Subscriber do MQTT.

As informações publicadas não precisam ser, necessariamente, utilizadas. O uso das informações de heartbeat é de responsabilidade da aplicação do usuário.

Falha de resposta do broadcast fará com que o EFM100 tente uma nova conexão junto ao broker, periodicamente.

5.2.6. Comandos para Entradas Digitais

5.2.6.1. Leitura de Estado Lógico

Efetua a leitura simples dos estados de um canal de entrada digital através de um comando “read”. É possível ler sinais de entradas digitais tanto de módulos ou como de uma unidade de processamento.

Uma mensagem JSON é enviada ao dispositivo através do tópico sub. A resposta é enviada pelo dispositivo por uma mensagem JSON no tópico pub.

- Unit: cpu/di | mod/di.
- Parâmetros: cmd, channel.
    - cmd: “read”.
    - channel: número inteiro positivo, o primeiro canal é ‘0’.

Por exemplo, para fazer a leitura do primeiro canal (=0) de entrada digital em uma série de módulos do dispositido “machine_1”:

```
{
  “device id”:”machine_1”,
  “unit”:”mod/di”,
  “cmd”:”read”,
  “channel”:”0”
}
```

Mensagem respondida pelo dispositivo:

```
{
  “device”:”machine_1”,
  “timestamp”:”568112”,
  “unit”:”mod/di”,
  “cmd”:”read”,
  “channel”:”0”,
  “state”:”2”,
  “status”:”ok”
}
```

No exemplo acima, a resposta indica que a entrada digital #2 do primeiro módulo está ativa.

5.2.7. Comandos para Saída a Relé

5.2.7.1. Leitura de Estado Lógico

Efetua a leitura simples de um canal de saída à relé. Uma mensagem JSON é enviada ao dispositivo através do tópico sub. A resposta é enviada pelo dispositivo por uma mensagem JSON no tópico pub.

- Unit: cpu/relay | mod/relay.
- Parâmetros: cmd, channel.
    - cmd: “read”.
    - channel: número inteiro positivo, o primeiro canal é ‘0’.

A seguir um exemplo de leitura do segundo canal (=1) de saída a relé em uma série de módulos. O dispositivo responde indicando que apenas a DI3 está acionada (4 em decimal):

```
{
  “device id”:”pack_machine”,
  “unit”:”mod/relay”,
  “cmd”:”read”,
  “channel”:”1”
}
```

Mensagem respondida pelo dispositivo:

```
{
  “device”:”machine_1”,
  “timestamp”:”3789544”,
  “unit”:”mod/di”,
  “cmd”:”read”,
  “channel”:”1”,
  “state”:”4”,
  “status”:”ok”
}
```

-

Exemplo:

JSON enviado ao dispositivo:

{

JSON respondido pelo dispositivo:

{

5.2.7.2. Escrita de Estado Lógico

Efetua a escrita de estados lógicos para todas as saídas em um canal de saída a relé.

- Unit: cpu/relay | mod/relay.
- Parâmetros: cmd, channel, state
  - cmd: “write”
  - channel: número inteiro positivo, o primeiro canal é ‘0’.
  - state: número inteiro positivo, de 0 a 255.

Exemplo: ligar a saída a relé 3 e desligar as demais, no terceiro canal de uma série de módulos:

JSON enviado ao dispositivo:

```
{
  “unit”:”mod/relay”,
  “cmd”:”write”,
  “channel”:”2”,
  “state”:”4”
}
```

Operações de escrita tem a maior prioridade na atualização dos estados das saídas à relé. Se uma operação de pulso estiver sendo processada ela será cancelada e substituída.

5.2.7.3. Ligar uma Seleção de Saídas”

Seta os estados lógicos de uma seleção de saídas à relé, não alterando o estado lógico das outras. Somente as saídas selecionadas com o valor ‘1’ em “mask” são alteradas para ON.

- Unit: cpu/relay | mod/relay
- Parâmetros: cmd, channel, mask
  - cmd: “turn_on”.
  - channel: número inteiro positivo, onde o primeiro canal é ‘0’.
  - mask: número inteiro positivo, de 0 a 255, representando apenas as saídas que serão ligadas.

Exemplo: ligar apenas a saída 4 do canal de saída a relé da unidade de processamento, não alterando as demais.

```
{
  “unit”:”mod/relay”,
```

```
  “cmd”:”turn_on”,
  “channel”:”0”,
  “mask”:”8”
}
```

5.2.7.4. Desligar uma Seleção de Saídas

Reseta os estados lógicos de uma seleção de saídas à relé, não alterando o estado lógico das outras. Somente as saídas selecionadas com o valor ‘1’ em “maks” são alteradas para OFF.

- Unit: cpu/relay | mod/relay
- Parâmetros: cmd, channel, mask
  - cmd: “turn_off”.
  - channel: número inteiro positivo, onde o primeiro canal é ‘0’.
  - mask: número inteiro positivo, de 0 a 255, representando apenas as saídas que serão desligadas.

Exemplo: desligar apenas a saída 4 do primeiro canal de saída a relé em uma série de módulos, não alterando as demais.

```
{
  “unit”:”mod/relay”,
  “cmd”:”turn_off”,
  “channel”:”0”,
  “mask”:”8”
}
```

5.2.7.5. Pulsar ON uma Seleção de Saídas

Pulsa OFF-ON-OFF uma seleção de saídas à relé, não alterando o estado lógico das outras. As saídas selecionadas com o valor ‘1’ em “mask” serão pulsadas.

- Unit: cpu/relay | mod/relay
- Parâmetros: cmd, channel, mask, duration.
  - cmd: “pulse_on”.
  - channel: número inteiro positivo, onde o primeiro canal é ‘0’.
  - mask: número inteiro positivo, de 0 a 255, representando apenas as saídas que serão pulsadas.
  - duration: número inteiro positivo, de 0 a 65535, representando o tempo em milissegundos que o pulso ocorrerá.

Exemplo: pulsar ON apenas a saída 4 (mask = 8) do primeiro canal de saída a relé da unidade de processamento, durante um segundo (1000 ms), não alterando as demais saídas.

JSON enviado ao dispositivo:

```
{
  “unit”:”mod/relay”,
  “cmd”:”pulse_on”,
  “channel”:”0”,
  “mask”:”8”,
  “duration”:”1000”
}
```

Se um prévio comando de pulso no respectivo canal estiver em execução ele será substituído pelo novo.

5.2.7.6. Rastrear Sinais de Uma Unidade

Esta função está descontinuada a partir da versão 1.1. Para novas aplicações veja a função “publish at change of state”.

Criar um rastreamento de sinais fará com que o EFM100 publique uma mensagem quando um critério dado em “function” for atingido.

Quando um rastreamento é configurado para “change of state”, toda vez que a mudança de estado lógico acontecer causará o EFM100 realizar uma publicação no tópico previamente configurado.

Os rastreios são salvos na memória e mantida a configuração mesmo após desligado. A única forma de cancelar um comando de rastreio é enviando a função “off” para a unidade e canal que se deseja desligar.

- Unit: cpu/di | mod/di | cpu/do | mod/do | cpu/relay | mod/relay
- Parâmetros: cmd, channel, function
  - cmd: “track”.
  - channel: número inteiro positivo, onde o primeiro canal é ‘0’.
  - function: “off” | “change of state”

Exemplo: configurar o canal ‘0’ da CPU para publicar toda vez que houver mudança de estado em uma entrada digital:

JSON enviado ao dispositivo:

```
{
  “unit”:”cpu/di”,
  “cmd”:”track”,
  “function”:”change of state”,
  “channel”:”0”
}
```

Toda vez que uma ou mais terminais da entrada digital mudarem de estado lógico será publicado:


```
{
  “unit”:”cpu/di”,
  “cmd”:”track”,
  “function”:”change of state”,
  “channel”:”0”,
  “state”:”5”,
  “timestamp”:”12345”,
  “status”:”ok”
}
```


O registro do comando para rastreio é mantido mesmo após a reinicialização.

6. DIAGNÓSTICO

6.1. Interface LEDs

O cartão de processamento EFM100 possui dois LEDs para indicar o estado de funcionamento.

- LED ST (verde): indica o estado de operação.
- LED ER (vermelho): indica a presença de algum alerta ou erro.

LED	Comportamento	Significado
ST + ER	Pisca junto rápido	Carregado padrão de fábrica
ST + ER	Acende brevemente ao energizar	Sistema iniciando
ST	Piscando lento (a cada 4 s)	Ethernet desconectada
ST	Piscando lento (a cada 2 s)	Ethernet conectada, estabelecendo conexão (DHCP)
ST	Piscando médio (a cada 1 s)	Conexão estabelecida
ST	Piscando rápido	Serviços de comunicação em operação normal.
ER	Aceso por um segundo	Algum alerta foi detectado
ER	Aceso por três segundos	Um erro foi detectado
ER	Aceso permanente	Erro fatal

Durante a energização os LEDs verde e vermelhos piscarão brevemente. Este evento é normal e não indica presença de erros.

6.2. Ferramentas de Debug

6.2.1. Windows Powershell

A versão padrão do Powershell que acompanha o Windows é suficiente para realizar depurações mais simples. As versões posteriores ao Powershell 7.0 possuem mais recursos e podem ser baixadas gratuitamente na Microsoft Store.

Verifique a versão do Powershell usando o comando a seguir:

```
PowerShell 7.5.2
PS C:\Windows\System32> $PSVersionTable.PSVersion

Major  Minor  Patch  PreReleaseLabel BuildLabel
-----  -----  -----  --------------- ----------
7      5      2

PS C:\Windows\System32>
```

Com o serviço de RESTful API habilitado, pode-se enviar comandos para o EFM100, como por exemplo:

```
PS C:\Windows\System32> $response = Invoke-WebRequest -Uri "http://192.168.18.201:8080/cpu/di?cmd=read&channel=0" -Method GET
PS C:\Windows\System32> $response

StatusCode        : 200
StatusDescription : OK
Content           : {
                    "unit":"cpu/di",
                    "cmd":"read",
                    "channel":"0",
                    "state":"15",
                    "status":"ok"
                    }
RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Type: application/json
                    Content-Length: 77

                    {
                    "unit":"cpu/di",
                    "cmd":"read",
                    "channel":"0",
                    "state":"15",
                    "status":"ok"
                    }
Headers           : {[Connection, System.String[]], [Content-Type, System.String[]], [Content-Length, System.String[]]}
Images            : {}
InputFields       : {}
Links             : {}
RawContentLength  : 77
RelationLink      : {}
```

O exemplo a seguir pode ser usado para testar continuamente a escrita da saída digital no cartão da CPU:

> Recomenda-se desconectar os terminais de conexão dos relés.

```
PS C:\Windows\System32> while ($true) {
>>     $r = Invoke-WebRequest -Uri "http://192.168.18.201:8080/cpu/relay?cmd=write&channel=0&state=$a" -Method GET
>>
>>     if ($a -eq 15) {
>>         $a = 0
>>     }
>>     else {
>>         $a = $a + 1
>>     }
>>     Start-Sleep -Milliseconds 100
>> }
```

7. FORMATOS CONSTRUTIVOS

7.1. B1