

PLC500 EDGE DEVICE

PLC500ED

Nota de Aplicação



Nota de Aplicação

PLC500ED

Documento: 10011077119

Revisão: 00

Data de publicação: 07/2023

SUMÁRIO DAS REVISÕES

A informação abaixo descreve as revisões ocorridas neste manual.

Versão	Revisão	Descrição
-	R00	Primeira edição.

1	INTRODUÇÃO	1-1
1.1	ABREVIACÕES E DEFINIÇÕES	1-1
1.2	SOBRE O PLC500ED	1-2
2	CONFIGURAÇÃO PELA WEBPAGE	2-1
2.1	ACESSO AO DISPOSITIVO	2-1
2.1.1	Ethernet ou USB	2-1
2.1.2	Requisitos da Nova Senha	2-2
2.2	PÁGINA DE ESTADO	2-3
2.2.1	Painel de Informações do Sistema	2-4
2.2.2	Painel de Conexão MQTT	2-4
2.2.3	Painel de Informações do Docker	2-4
2.3	PÁGINA DE CONFIGURAÇÃO	2-5
2.3.1	Painel de Integração com a Nuvem	2-5
2.3.1.1	WEGnology	2-5
2.3.1.2	WEG Smart Machine	2-7
2.3.2	Painel de Interfaces de Rede	2-8
2.3.3	Painel da Interface Serial	2-9
2.3.4	Painel Docker	2-10
2.4	PÁGINA DE ADMINISTRAÇÃO	2-11
2.4.1	Painel de Ações de Aplicação	2-11
2.4.2	Painel de Testes de Comunicação	2-13
2.4.2.1	Teste da Comunicação Serial	2-13
2.4.2.2	Teste da Comunicação TCP/IP	2-14
2.4.2.3	Testes de Conectividade do Dispositivo	2-15
2.4.3	Painel de Gerenciamento do Sistema	2-16
3	PLATAFORMA WEGNOLOGY	3-1
3.1	CONFIGURAÇÃO INICIAL	3-1
3.1.1	Criação da Conta e da Aplicação	3-1
3.1.2	Adicionar uma Chave de Acesso	3-2
3.1.3	API Token	3-2
3.1.4	Application ID	3-3
3.1.5	Conectando o Dispositivo	3-3
3.1.6	Adicionando Atributos	3-3
4	CODESYS - BIBLIOTECA WEGNOLOGY	4-1
4.1	RECOMENDAÇÕES	4-1
4.2	BIBLIOTECA CODESYS IIOT LIBRARIES SL	4-2
4.3	SINTAXE DOS OBJETOS JSON PUBLICADOS	4-3
4.3.1	WEGnology	4-3
4.3.2	Edge Computing	4-3
4.4	BLOCOS DE FUNÇÕES E MÉTODOS MQTT	4-4
4.5	EXEMPLOS DE APLICAÇÕES	4-5
4.5.1	Publicação de payloads para a plataforma WEGnology	4-5
4.5.2	Subscrição em tópicos da plataforma WEGnology	4-8
4.5.3	Publicação de payloads para o Edge Agent	4-11
4.5.4	Subscrição em tópicos do Edge Agent	4-15
5	GUIA DE INÍCIO RÁPIDO	5-1
5.1	ESTABELEECER A CONEXÃO COM O PRODUTO	5-1
5.2	CONECTAR O PRODUTO NA INTERNET VIA CODESYS OU WEBPAGE	5-1
5.3	CONFIGURAR A APLICAÇÃO IOT PELA WEBPAGE	5-2
5.4	HABILITAR UMA IMAGEM DOCKER	5-2
5.5	CRIAÇÃO DA APLICAÇÃO	5-3

1 INTRODUÇÃO

Esta Nota de Aplicação apresenta as principais características e informações necessárias para a configuração e utilização do PLC500ED.

É apresentada a biblioteca WEGnology, desenvolvida pela WEG para simplificar o uso das funcionalidades de conectividade do dispositivo. Exemplos de aplicações básicas são mostradas para ilustrar algumas possibilidades do produto.

Para mais informações a respeito do hardware do produto, interfaces e protocolos de comunicação, consulte o Manual do Usuário do PLC500, disponível em www.weg.net.

1.1 ABREVIACIONES E DEFINIÇÕES

Broker: Servidor que gerencia o recebimento de mensagens enviadas pelos clientes publisher, as enviando para os clientes subscriber através do protocolo MQTT.

Codesys: Plataforma de programação que permite desenvolver, configurar e monitorar soluções para automação industrial e integração de sistemas.

Container: Instância de execução de uma imagem docker contendo todos os recursos necessários para executar uma aplicação.

Coreapp: Software de gerenciamento de serviços e recursos do PLC.

DNS: Sistema responsável pela tradução de endereços IP em nome de domínios, e vice-versa (Domain Name System).

Docker: Serviço de software que estabelece uma camada de abstração para virtualização de sistemas operacionais Windows/Linux entregando pacotes chamados containers.

Edge Agent: Container previamente instalado no produto que permite a execução local de Edge Workflows.

Edge Workflow: Lógicas programadas via plataformas IoT da WEG e que são executadas localmente nos dispositivos com capacidade de processamento de borda.

Ethernet: Arquitetura de interconexão para redes locais (IEEE 802.3).

Gateway: Dispositivo de hardware que permite o fluxo de dados entre diversas redes de comunicação.

Imagem Docker: Pacote de software utilizado como template na geração de containers.

IoT: Sigla que refere-se às tecnologias que facilitam a comunicação e a troca de dados entre dispositivos e a nuvem, bem como entre os próprios dispositivos (Internet of Things).

IP: Protocolo utilizado na internet para encaminhamento de datagramas entre dispositivos em rede (Internet Protocol).

MQTT: Protocolo de transporte que utiliza a topologia publicação/subscrição para transferência de mensagens leves entre dispositivos (Message Queuing Telemetry Transport).

Payload: Conteúdo da mensagem MQTT.

Plataforma cloud: Plataforma que oferece um conjunto de serviços de nuvem, como acesso remoto a softwares e armazenamento e processamento de dados por meio da internet.

Processamento de Borda: Processamento de dados realizado próximo ao usuário ou fonte de dados (Edge Computing). Utiliza o container do Edge Agent para a execução dos comandos.

QoS: Parâmetro utilizado para determinar o nível de qualidade de serviço em troca de mensagens utilizando o protocolo MQTT (Quality of Service).

WEGnology: Plataforma de serviço de nuvem utilizada nas aplicações de IoT da WEG. Nome da biblioteca desenvolvida pela WEG para uso das funcionalidades de conectividade do PLC500ED no CODESYS.

INTRODUÇÃO

1.2 SOBRE O PLC500ED

O PLC500 Edge Device (PLC500ED) é um Controlador Lógico Programável com capacidade de Processamento de Borda, pelo qual é possível conectar equipamentos industriais às plataformas de nuvem da WEG para utilização em soluções digitais.

É desenvolvido para atender aplicações de médio e grande porte. Possui alta velocidade de processamento devido a sua CPU composta por um processador Dual-core ARM Cortex-A7 rodando a 1 GHz, um coprocessador Real-time ARM Cortex-M4 de 200 MHz, memória RAM de 1 GB e Flash de 4 GB.

Possui um total de 8 saídas digitais, sendo 3 destas com funcionalidade PWM até 300 kHz, e 8 entradas digitais, das quais 4 podem operar até 150 kHz.

Como interfaces de comunicação, estão disponíveis duas portas Ethernet independentes, porta CAN, serial RS485, USB OTG, USB device e Micro SD Card.

São utilizados supercapacitores internos para o Relógio de Tempo Real (RTC) e também para salvar dados retentivos em memória Flash durante o Power Off, dispensando assim o uso de baterias.

O PLC500ED permite a conexão de cartões de expansão de entradas e saídas digitais, analógicas, termopar, PT100, PT1000, célula de carga, relés, etc., dando mais flexibilidade às aplicações. Possui conectores plug-in e a fixação pode ser feita em trilho DIN 35 ou diretamente no painel.

A programação do PLC500ED é realizada pelo software CODESYS, amplamente difundido no meio industrial, possibilitando a utilização de uma infinidade de aplicações e funções já desenvolvidas no mercado, bem como a importação de aplicações de outros produtos.

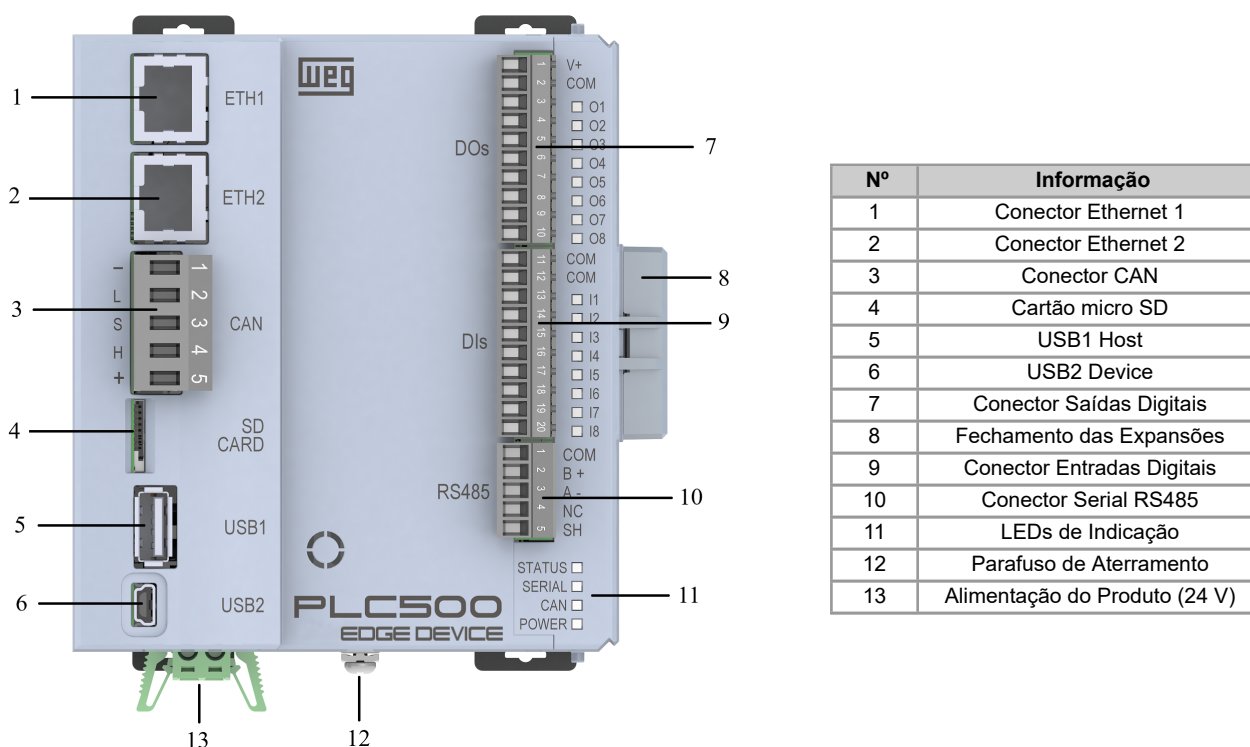


Figura 1.1: PLC500 Edge Device.

2 CONFIGURAÇÃO PELA WEBPAGE

O PLC500ED possui um servidor web local disponível para comunicação. É possível acessá-lo através das conexões ETH1, ETH2 e USB2. Para isso, deve-se realizar os seguintes passos:

2.1 ACESSO AO DISPOSITIVO

2.1.1 Ethernet ou USB

1. Conectar um cabo Ethernet ou mini USB entre o computador e a respectiva porta do PLC. As portas são localizadas na frente do produto, conforme visto na Figura 1.1.
2. O Computador deve ser configurado com um IP estático dentro da mesma rede utilizada pela conexão do PLC500ED. Os endereços de IP padrão do PLC500ED são mostrados na Tabela 2.1.

Tabela 2.1: Endereços de IP padrão do PLC500ED.

Conexão	Endereço de IP padrão
ETH1	192.168.1.10
ETH2	192.168.2.10
USB2	192.168.234.234

Exemplo de configuração (considerando os IPs padrão): configure o endereço de IP do Computador para 192.168.1.X (ETH1), 192.168.2.X (ETH2) ou 192.168.234.Y (USB2), onde X é qualquer valor inteiro de 1 a 254, exceto 10, e Y é qualquer valor inteiro de 1 a 254, exceto 234. Deixe a máscara de rede em seu valor padrão (255.255.255.0).



NOTA!

Os endereços de IP das portas ETH1 e ETH2 podem ser modificados através do **Codesys** ou no menu **Configuration** da Webpage do produto.



NOTA!

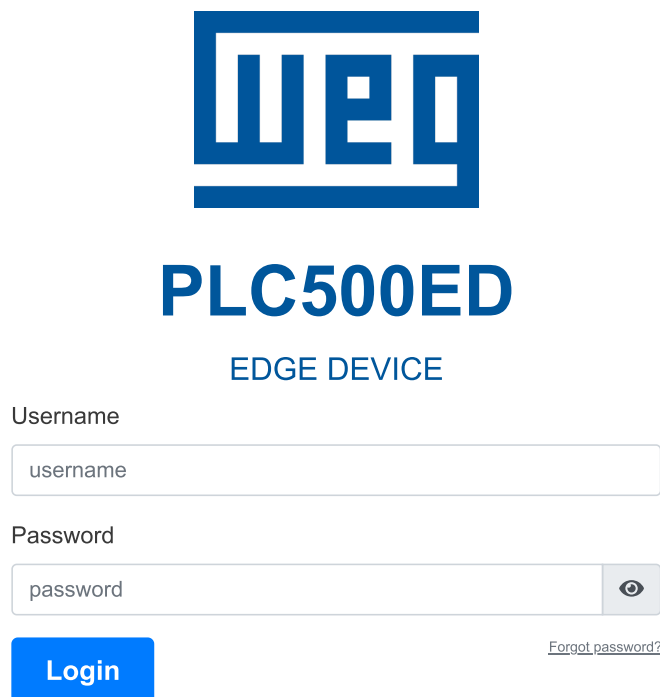
O IP do USB2 é fixo e não pode ser modificado.



ATENÇÃO!

Ao baixar uma aplicação pelo **Codesys**, a configuração do **Setup** é gravada no produto, sobrescrevendo qualquer configuração realizada pela **Webpage**.

3. Uma vez que a configuração estiver completa, é necessário utilizar um navegador web recente (preferencialmente Firefox ou Chrome) e digitar o IP da porta do dispositivo no campo de URL. Após realizar a conexão via Ethernet ou USB, uma página com a tela de login deve aparecer no navegador, conforme mostra a Figura 2.1.



Username

Password

[Forgot password?](#)

Login

Figura 2.1: Tela de login do PLC500ED.

4. Inicialmente, o dispositivo vem configurado com o login e senha padrão da WEG, sendo necessário sua atualização no primeiro acesso. O login e senha padrão são:

- Usuário: weg
- Senha: weg

2.1.2 Requisitos da Nova Senha

Na primeira tentativa de acesso será exigida a modificação de usuário e senha. A nova senha deve respeitar os seguintes critérios de segurança:

- Possuir no mínimo 14 caracteres
- Possuir no mínimo 1 caractere maiúsculo
- Possuir no mínimo 1 caractere numérico
- Possuir no mínimo 1 caractere especial (símbolo)

Após preencher os campos da janela de geração de novo login e clicar no botão **Submit**, uma mensagem de sucesso irá aparecer se os critérios descritos anteriormente forem respeitados. Após este procedimento, é possível fazer o login normalmente digitando as novas credenciais.



ATENÇÃO!

Recomenda-se **salvar** o novo **login** e **senha** em local seguro, pois a recuperação de login e/ou senha somente serão possíveis via **suporte técnico!**

2.2 PÁGINA DE ESTADO

Uma vez logado, o usuário terá acesso à Página de Estado do produto. Dentro desta página é possível visualizar informações relativas ao funcionamento do sistema, conexão MQTT e aplicações Docker através de três painéis, sendo eles:

- Painel de Informações de Sistema
- Painel de Conexão MQTT
- Painel de Informações do Docker

A Figura 2.2 mostra a Página de Estado do PLC500ED.

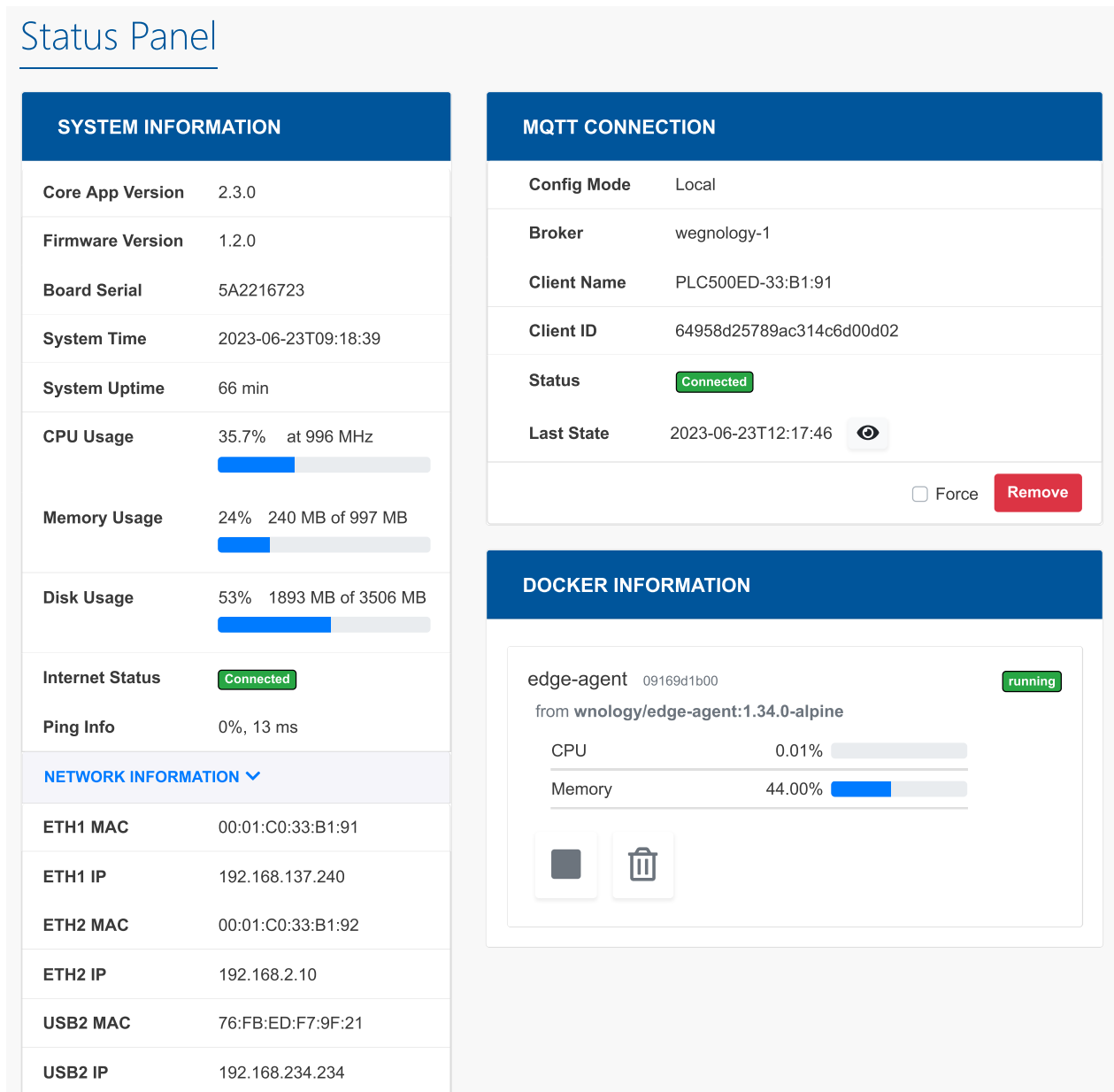


Figura 2.2: Página de Estado.

2.2.1 Painel de Informações do Sistema

No Painel de Informações do Sistema é possível visualizar informações sobre versões, modelo, configurações, estados, utilização de recursos, entre outros. A lista a seguir apresenta a descrição detalhada de cada campo contido no painel.

- **Core app Version:** Versão do software relacionado ao Core app no PLC500ED.
- **Codesys Version:** Versão do software relacionado ao Codesys no PLC500ED.
- **Board Serial:** Identificador único atribuído ao circuito eletrônico.
- **System Time:** Data e hora do sistema.
- **System Uptime:** Tempo de operação do sistema desde a sua inicialização (expresso em minutos).
- **CPU Usage:** Utilização do CPU e frequência atual do processador (expressas em % e MHz).
- **Memory Usage:** Quantidade de memória RAM sendo utilizada no momento (expressa em % e MB).
- **Disk Usage:** Quantidade de espaço em disco ocupado no momento (expressa em % e MB).
- **Internet Status:** Estado da conexão à internet (Conectado ou Desconectado).
- **Ping Info:** Taxa de perda de pacotes registrados (expressa em %) e Round-trip time - Tempo entre a requisição e o recebimento de um pacote registrado (expressa em milissegundos).

Informações de rede:

- **ETH1 MAC:** Endereço MAC da interface Ethernet ETH1.
- **ETH1 IP:** Endereço IP da interface Ethernet ETH1.
- **ETH2 MAC:** Endereço MAC da interface Ethernet ETH2.
- **ETH2 IP:** Endereço IP da interface Ethernet ETH2.
- **USB2 MAC:** Endereço MAC da interface USB.
- **USB2 IP:** Endereço IP da interface USB.

2.2.2 Painel de Conexão MQTT

Uma vez que o PLC500ED esteja conectado à plataforma WEGnology, é possível visualizar o Painel de Conexão MQTT com as seguintes informações:

- **Config Mode:** Modo de configuração: Local ou Remoto.
- **Broker:** Nome do broker MQTT.
- **Client Name:** Nome do cliente MQTT.
- **Client ID:** Identificador do cliente MQTT.
- **Status:** Estado da conexão MQTT (Conectado ou Desconectado).
- **Last State:** Informação de timestamp do último payload de estado enviado.
- **Force (caixa de seleção):** Flag que força a remoção da conexão MQTT.
- **Remove (botão):** tem a função de remover a conexão MQTT entre o cliente e o broker. Esta funcionalidade está disponível apenas quando o integrador WEGnology for selecionado na Página de Configuração.



NOTA!

Para visualizar as informações do **Painel de Conexão MQTT** é necessária a configuração da aplicação no **Painel de Integração com a Nuvem**, localizado na **Página de Configuração**.

2.2.3 Painel de Informações do Docker

No Painel de Informações do Docker são apresentadas as seguintes informações:

- **Identificação do container:** Nome do container, seu respectivo short ID e a imagem base usada para a montagem do container, localizados no canto superior esquerdo.
- **Estado do container:** Estado atual do container, localizado no canto superior direito (inicializando, rodando ou parado).
- **CPU e Memory:** Porcentagem da utilização de CPU e de memória RAM relacionadas ao container.
- Os botões identificados por um quadrado e uma lixeira possuem as funções de parar e remover o container, respectivamente.



NOTA!

Para visualizar as informações do **Painel de Informações do Docker** é necessário habilitar a imagem do container específico no **Painel do Docker**, localizado na **Página de Configuração**.

2.3 PÁGINA DE CONFIGURAÇÃO

A Página de Configuração permite a configuração de parâmetros relacionados à rede, plataforma de nuvem, interface serial e imagens Docker. Dentro desta página é possível acessar os seguintes painéis:

- Painel de Integração com a Nuvem
- Painel de Interfaces de Rede
- Painel de Interface Serial
- Painel Docker

A Figura 2.3 mostra a Página de Configuração do PLC500ED.

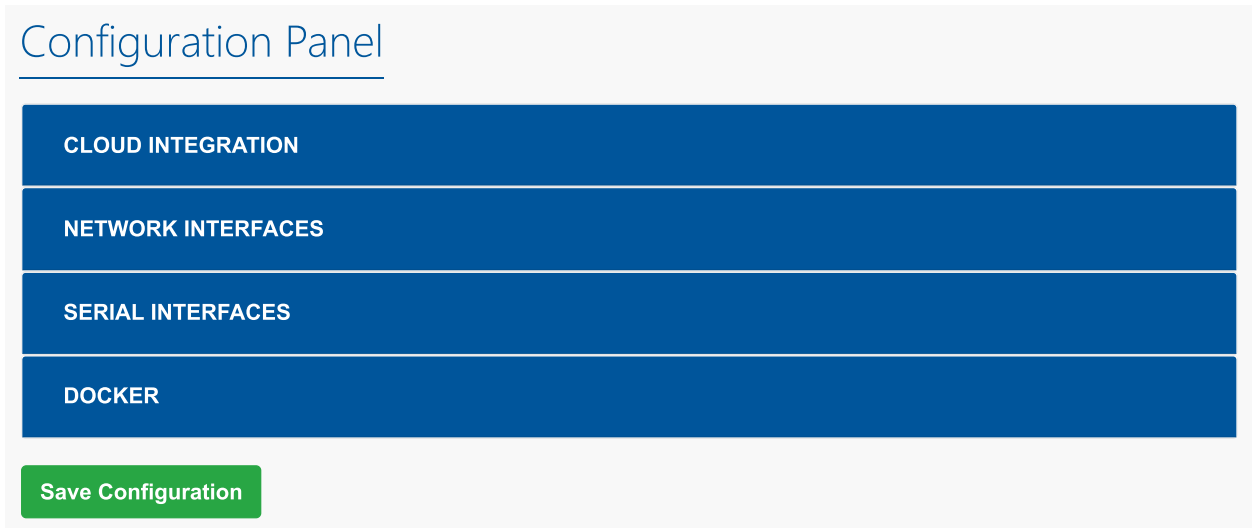


Figura 2.3: Página de Configuração.

2.3.1 Painel de Integração com a Nuvem

No Painel de Integração com a Nuvem é possível selecionar dois integradores diferentes: **WEGnology** e **WEG Smart Machine**. Os demais parâmetros do painel dependem do integrador selecionado.

2.3.1.1 WEGnology

A Figura 2.4 mostra os parâmetros de configuração disponíveis para o integrador WEGnology.

Configuration Panel

CLOUD INTEGRATION

Integrator: MQTT Configuration Status: **Locally Configured**

Application ID: API Token: Token file: (optional) Nenhum arq... escolhido

Access Key: Secret: Access Keys file: (optional) Nenhum arq... escolhido

NETWORK INTERFACES

SERIAL INTERFACES

DOCKER

Figura 2.4: Página de Configuração - Integração com a Nuvem - WEGnology.



NOTA!

- Antes de configurar a conexão com alguma plataforma cloud da WEG na webpage do PLC500ED é necessário realizar os procedimentos de geração da aplicação e de credenciais na plataforma respectiva.
- Para mais informações sobre estes procedimentos, consultar a documentação referente à plataforma no site da WEG ou consultar o suporte técnico.

- **Integrator:** Serviço de integração de nuvem (Selecionar opção: **WEGnology**).
- **Application ID:** ID da aplicação existente na plataforma da nuvem.
- **API Token:** Token para permissão de gerenciamento da aplicação na plataforma da nuvem.
- **Token file:** Opção de importar arquivo contendo as informações de API Token. Caso esta opção seja usada, o campo anterior não precisa ser preenchido.
- **Access Key:** Chave de acesso para permitir que um dispositivo se conecte à plataforma da nuvem.
- **Secret:** Senha de acesso para permitir que um dispositivo se conecte à plataforma da nuvem.
- **Access Keys file:** Opção de importar arquivo contendo as informações de Access Key e Secret. Caso esta opção seja usada, os dois campos anteriores não precisam ser preenchidos.

2.3.1.2 WEG Smart Machine

A Figura 2.5 mostra os parâmetros de configuração disponíveis para o integrador WEG Smart Machine.

Figura 2.5: Página de Configuração - Integração com a Nuvem - WEG Smart Machine.

- **Integrator:** Serviço de integração de nuvem (Selecionar opção: **WEG Smart Machine**).
- **Application ID:** ID da aplicação existente na plataforma da nuvem.
- **API Token:** Token para permissão de gerenciamento da aplicação na plataforma da nuvem.
- **Device ID:** ID do dispositivo cadastrado na plataforma da nuvem.
- **Access Key:** Chave de acesso para permitir que um dispositivo se conecte à plataforma da nuvem.
- **Secret:** Senha de acesso para permitir que um dispositivo se conecte à plataforma da nuvem.
- **Credentials file:** Opção de importar arquivo contendo todas as informações descritas acima. Caso esta opção seja usada, os cinco campos anteriores não precisam ser preenchidos.

2.3.2 Painel de Interfaces de Rede

No Painel de Interfaces de Rede é possível configurar as interfaces ETH1 e ETH2, como pode ser visto na Figura 2.6. Cada interface pode ser habilitada ou desabilitada através dos botões On/Off no canto direito do painel, alinhados com o nome de cada interface.

The screenshot shows a web-based configuration panel titled "Configuration Panel". It has a blue header with "CLOUD INTEGRATION" and "NETWORK INTERFACES" sections. Below "NETWORK INTERFACES", there are three interface configuration sections: "Interface ETH1", "Interface ETH2", and "Interface USB2". Each section has a status toggle (On/Off) on the right. The "Interface ETH1" section has "Use DHCP" set to "Yes" and "Default Route" set to "Yes". The "Interface ETH2" section has "Use DHCP" set to "No" and "Default Route" set to "No". The "Interface USB2" section has "IP Address" set to "192.168.234.234" and "Network Mask" set to "255.255.255.0". The "Interface ETH1" section has "IP Address" set to "IP", "Network Mask" set to "Mask", and "Gateway" set to "Gateway". The "Interface ETH2" section has "IP Address" set to "192.168.2.10", "Network Mask" set to "255.255.255.0", and "Gateway" set to "Gateway". At the bottom of the panel, there is a green "Save Configuration" button.

Figura 2.6: Página de Configuração - Interfaces de Rede.

A lista a seguir apresenta a descrição de cada campo a ser configurado pelo usuário dentro do painel.

Interfaces ETH1 e ETH2

- **Use DHCP:** Habilita/Desabilita a utilização de DHCP.
- **Default Route:** Habilita/Desabilita a rota padrão de rede para o endereço de destino dos pacotes IP.
- **IP Address:** Endereço IP da interface Ethernet.
- **Network Mask:** Máscara de rede referente ao endereço IP da interface Ethernet.
- **Gateway:** Endereço IP do Gateway da rede.



NOTA!

Não configure as interfaces ETH1 e ETH2 com IPs de uma mesma rede. Ex: 192.168.1.10 e 192.168.1.20. Caso tenha feito, pode-se entrar na página web via USB2 e trocar os IPs novamente para os valores corretos ou realizar o **Factory Reset**.



NOTA!

Caso a configuração do **acesso à internet** seja realizada por meio de uma interface Ethernet, lembre-se de ativar a opção **Default Route**.

Interface USB2

No painel da interface USB2 são mostrados os respectivos endereço de IP e máscara de rede. Contudo, estes dados constam na Página de Configuração a título de informação apenas, uma vez que seus campos não podem ser alterados.

2.3.3 Painel da Interface Serial

Neste painel é possível habilitar ou desabilitar a interface serial via RS485 para ser utilizada através dos containers Docker. A Figura 2.7 mostra o Painel da Interface Serial.

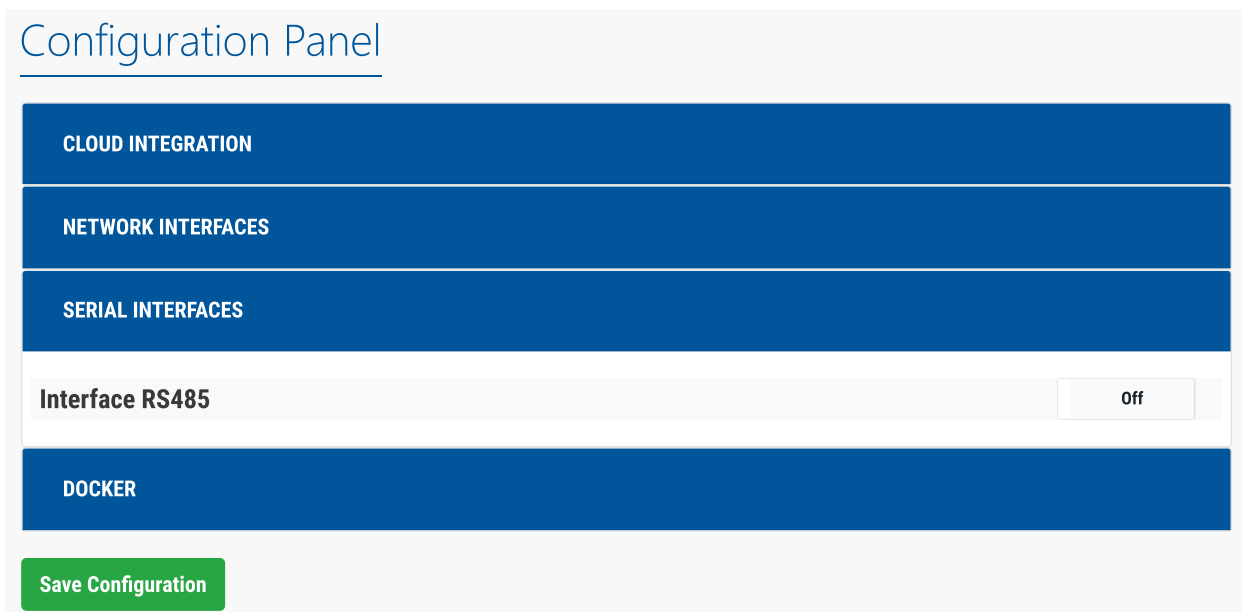


Figura 2.7: Página de Configuração - Painel da Interface Serial.



ATENÇÃO!

Não é recomendada a utilização concomitante da RS485 via plataformas WEG e pelo Codesys, uma vez que erros poderão ser gerados e a comunicação poderá ser interrompida.

2.3.4 Painel Docker

No Painel Docker, visto na Figura 2.8, é possível visualizar as seguintes informações:

Image Name	ID	Size	Created at	Status
wnology/edge-agent:1.25.0-alpine	4e41711b3	525	2021-11-24T13:56:35	Enabled <input type="button" value="Off"/>

Figura 2.8: Página de Configuração - Painel Docker.

- **Image Name:** Nome das imagens Docker usadas para a construção dos containers.
- **ID:** Short ID das imagens Docker.
- **Size:** Tamanho das imagens Docker em MB.
- **Created at:** Data e hora da criação das imagens Docker.
- **Status:** Estado da imagem Docker - Identifica se há algum container criado a partir da imagem. É possível habilitar ou desabilitar containers a partir das imagens através dos botões On/Off.



NOTA!

Para que o **Painel de Informações do Docker** seja visualizado na **Página de Estado**, deve-se habilitar a imagem Docker no campo **Status**.

2.4 PÁGINA DE ADMINISTRAÇÃO

A Página de Administração também pode ser acessada pelo usuário após o login. Dentro desta página é possível visualizar os seguintes painéis:

- Painel de Ações de Aplicação
- Painel de Testes de Comunicação
- Painel de Gerenciamento do Sistema

A Figura 2.9 mostra a Página de Administração do PLC500ED.

Figura 2.9: Página de Administração.

2.4.1 Painel de Ações de Aplicação

No Painel de Ações de Aplicação existem cinco botões que permitem ao usuário: reinicializar a aplicação, restaurar configurações de fábrica, modificar o login da página web, atualizar o firmware do produto e baixar os logs do sistema. Mais detalhes podem ser vistos abaixo.

- **Restart App:** Esta opção apenas reinicia as aplicações Edge do produto, sem reiniciar o PLC. Logo, a aplicação do Codesys não é reiniciada.



NOTA!

Apesar da aplicação do Codesys não ser reiniciada, a utilização do **Restart App** pode impactar a aplicação em execução, uma vez que os serviços ligados as plataformas WEG serão reiniciados.

CONFIGURAÇÃO PELA WEBPAGE

- **Factory Reset:** Ao clicar neste botão, as seguintes ações serão tomadas, seguidas da reinicialização do produto:
 - Modificação de login e senha da página web para o padrão de fábrica (login: weg, senha: weg).
 - Retomada de todas as configurações realizadas na Página de Configuração para o padrão de fábrica (cloud, redes, Docker).
 - Restauração das configurações de IP das redes para os valores padrão.
 - Exclusão da aplicação do Codesys.
- **Change Login:** Essa opção reseta o usuário e senha para o padrão de fábrica (login: weg, senha: weg).
- **Software Update:** Ao pressionar o botão **Software Update**, uma janela se abrirá mostrando as opções de **Atualização de Firmware** do produto, com o **Update Mode** inicialmente no modo **Local**, conforme mostra a Figura 2.10(a). Contudo, a opção **Local** pode ser alterada para **Remote**, de acordo com a Figura 2.10(b).

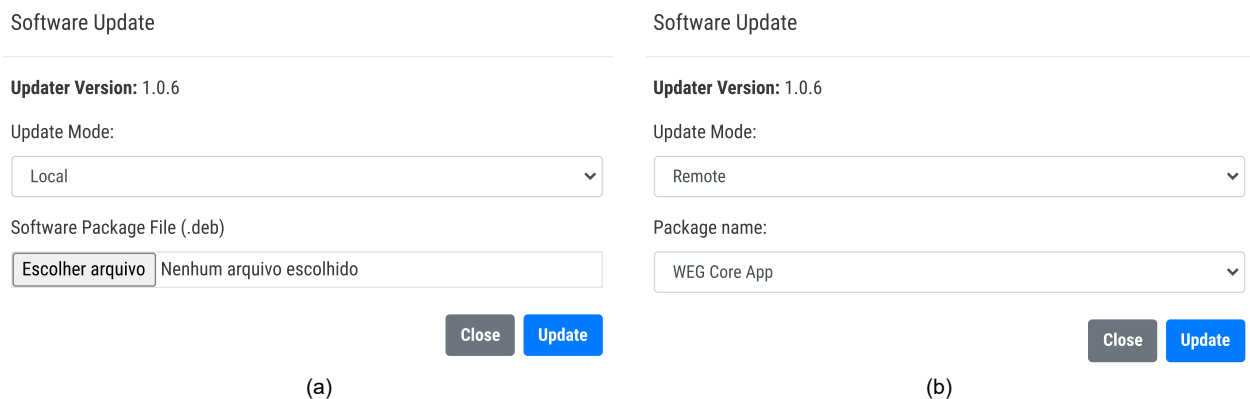


Figura 2.10: Painel de Atualização de Firmware: (a) Atualização Local. (b) Atualização Remota.

- **Modo Local:** A atualização de firmware em modo local permite que as atualizações possam ser realizadas mesmo que o PLC500ED não esteja conectado à internet. Para que isso seja possível, é necessário que o pacote de firmware seja previamente baixado no computador ou dispositivo móvel por onde a página web está sendo acessada. Em seguida, basta fazer o upload do pacote por meio do botão **Choose File** e pressionar o botão **Update** na sequência.



NOTA!

Na atualização de firmware em **Modo Local**, podem ser escolhidos tanto o arquivo **WEG Core App** quanto o **PLC500ED Firmware**.

- **Modo Remoto:** A atualização de firmware em modo remoto permite que a atualização possa ser realizada sem que o pacote de firmware seja previamente baixado. Na opção **Package name** podem ser escolhidos os pacotes **WEG Core App** (questões relacionadas ao gerenciamento de serviços) e **PLC500ED Firmware** (questões relacionadas basicamente ao Codesys).

Com estas condições satisfeitas, basta pressionar o botão **Update** para iniciar a atualização. Para isso, é essencial que o PLC500ED esteja com **acesso à internet**.



NOTA!

Na opção **PLC500ED Firmware**, o produto será reiniciado. Na opção **WEG Core App**, apenas as aplicações ligadas às plataformas WEG serão reiniciadas.

- **System Log Files:** Para que seja possível baixar os logs do sistema é exigido o preenchimento de uma chave de autenticação antes de fazer o download. Caso esta opção seja necessária, entrar em contato com a WEG para obter esta chave de autenticação.

2.4.2 Painel de Testes de Comunicação

No Painel de Testes de Comunicação é possível realizar os seguintes testes:

- Teste da Comunicação Serial
- Teste da Comunicação TCP/IP
- Testes de Conectividade do Dispositivo

2.4.2.1 Teste da Comunicação Serial

Ao pressionar o botão **Start**, a janela apresentada na Figura 2.11 é aberta. Como pode ser visto, é possível configurar os parâmetros necessários para estabelecer a comunicação Modbus RTU via RS485 e realizar a operação de **Read Holding Registers**. A lista a seguir apresenta a descrição de cada campo a ser configurado pelo usuário.

Test Protocol: Modbus RTU

Communication Parameters

Unit ID:

Speed:

Bits:

Parity:

Stop bits:

Holding Registers Read Information


Start Address: Size: 

Figura 2.11: Teste da Comunicação Serial.

- **Unit ID:** Número identificador do dispositivo.
- **Speed:** Taxa de transferência ou baud rate da comunicação serial.
- **Bits:** Bits de dados.
- **Parity:** Bits de paridade.
- **Stop Bits:** Bits de parada.
- **Start Address:** Endereço do primeiro registrador a ser lido.
- **Size:** Número de registradores a serem lidos a partir do endereço inicial.

O botão **Add Registers** permite que um novo conjunto de campos **Start Address** e **Size** possa ser preenchido para incluir mais intervalos de registradores a serem lidos. O botão com símbolo de lixeira apaga os respectivos campos Start Address e Size criados. Após preencher todos os campos, basta pressionar o botão **Execute** para realizar o teste.



NOTA!

Não é recomendado utilizar o **Teste da Comunicação Serial** quando utilizando a rede **RS485 via Codesys**, uma vez que erros podem ser gerados e a comunicação poderá ser interrompida.

CONFIGURAÇÃO PELA WEBPAGE

2.4.2.2 Teste da Comunicação TCP/IP

Esta funcionalidade permite que testes de conectividade de redes TCP/IP possam ser realizadas entre o produto e os demais destinos. Ao pressionar o botão **Start**, a janela da Figura 2.12 é aberta.



NOTA!

O **Teste da Comunicação TCP/IP** testa a conectividade das redes TCP/IP. Não é possível ler ou escrever em endereços de memória através do protocolo Modbus.

Communication Parameters

Host/IP Address:

Port:

Timeout:

Network Communication Tests

Ping Test



Open Port Test



Close

Figura 2.12: Teste da Comunicação TCP/IP.

É possível realizar dois tipos de testes, os quais podem ser utilizados para a verificação de redes locais (LAN) e redes conectadas usando a internet (WAN):

- **Ping Test:** o teste de ping utiliza o protocolo ICMP para verificar se o produto consegue alcançar o destino especificado no campo Host/IP Address. Neste campo, é permitido especificar o domínio ou o IP do destino. Caso o destino esteja acessível, um símbolo verde de “check” irá aparecer ao lado do botão do teste. Caso contrário, um símbolo vermelho de “X” indicará que o destino não está acessível.



NOTA!

É importante lembrar que, em alguns casos, mesmo o destino estando acessível na rede, o protocolo ICMP pode ser bloqueado por um firewall. Portanto, é importante também realizar o teste **Open Port Test** para garantir o resultado.

- **Open Port Test:** o teste de porta aberta realiza uma conexão de socket entre o produto e o destino especificado pelos campos Host/IP Address + Port. Caso o socket seja estabelecido, um símbolo verde de “check” irá aparecer ao lado do botão do teste. Caso contrário, um símbolo vermelho de “X” indicará que o socket não pode ser estabelecido. Neste segundo caso, caso o teste de ping tenha tido sucesso, é provável que a porta no lado do destino não esteja liberada para a conexão.

Os campos que devem ser configurados pelo usuário são os seguintes:

- **Host/IP Address:** Endereço de IP de destino.
- **Port:** Porta de comunicação.
- **Timeout:** Timeout em segundos. É configurável para ambos os testes.

2.4.2.3 Testes de Conectividade do Dispositivo

Esta funcionalidade permite que sejam verificados se os principais destinos na internet, necessários para o correto funcionamento do PLC500ED, estão acessíveis com as atuais configurações de rede.

Todos os serviços especificados abaixo do campo **Mandatory remote services** devem apresentar um símbolo verde de “check”, indicando que o serviço está acessível. Caso algum campo apresente um símbolo vermelho de “X”, alguma providência deve ser tomada para corrigir o problema. A Figura 2.13 mostra a janela aberta ao pressionar o botão **Start**, onde nota-se que todos os campos estão acessíveis.

Mandatory remote services

- DNS name resolution ✓

- WEGnology Web Server ✓

- WEGnology MQTT Broker ✓

- WEGnology REST API Server ✓

- Docker Container Repository ✓

- WEG Software Update Repository ✓

Close

Check

Figura 2.13: Testes de Conectividade do Dispositivo.

Quando algum destino falha no teste, pode ser que exista algum firewall na rede bloqueando o acesso ao domínio especificado. Para maiores detalhes sobre os destinos e domínios que devem estar acessíveis e liberados para que o produto utilize na internet, verifique a Tabela 2.3.

Tabela 2.3: Destinos e domínios necessários para liberação de firewall.

Service name	Function	Description	Protocol	Port	Destination	Destination IP
WEGnology MQTT Broker	Communication with MQTT Broker	Used to transmit measurements and commands from the Edge device to the MFM platform	TCP / MQTT	8883	broker.app.wnology.io	3227206235 3.234.136.81 52.22.246.163
WEGnology REST API Server	WEGnology applications	Services of the application in which the device is connected and also of the WEGnology platform	TCP / HTTPS	443	*.wnology.io api.app.wnology.io	3.227.206.235 3.234.136.81 52.22.246.163
WEGnology Web Server	Internet connection status	Used for checking internet status and plant configuration via WEG Scan webpage	TCP / ICMP	443	console.app.wnology.io	65.8.248.99 65.8.248.109 65.8.248.31 65.8.248.46
Docker Container Repository	Communication with the Drive Scan application repository	Drive Scan application repository	TCP / HTTPS	443	hub.docker.com	Dynamic IP address
WEG Software Update Repository	Communication for Drive Scan Firmware Update	Drive Scan Firmware Update	TCP / HTTPS	443	nexus3.weg.net	Dynamic IP address
DNS name resolution	Lookup of IP addresses by DNS	Required when the customer system could not resolve the domain names	TCP / UDP	53	-	8.8.8.8 8.8.4.4
-	Internet connection status	Used for checking internet status and plant configuration via WEG Scan webpage	ICMP	53	google.com	Dynamic IP address

2.4.3 Painel de Gerenciamento do Sistema

No Painel de Administração do Sistema são encontradas as funções **VPN Client Service** e **SSH Server**, que devem ser utilizadas apenas pelo suporte técnico da WEG.

3 PLATAFORMA WEGNOLOGY

Este capítulo apresenta os passos básicos para a configuração do usuário e da aplicação na plataforma WEGnology. Para maiores detalhes, favor consultar a documentação específica em docs.app.wnology.io.

Conforme mostrado na Figura 2.4 do capítulo anterior, para a integração com a nuvem, são necessários os seguintes dados para serem inseridos na página web do produto:

- Access Key
- Secret
- API Token
- Application ID

Esses dados podem ser inseridos manualmente, ou através do upload de arquivos específicos que contenham essas informações. Na próxima seção são mostrados os passos básicos para a configuração inicial da plataforma e obtenção destes dados.

3.1 CONFIGURAÇÃO INICIAL

3.1.1 Criação da Conta e da Aplicação

Primeiramente, deve-se entrar na página accounts.app.wnology.io e criar uma conta, inserindo os dados necessários. Na sequência, é pedido para dar um nome à aplicação e, opcionalmente, inserir uma descrição. A Figura 3.1 mostra as telas de criação de conta (Figura 3.1(a)) e criação da aplicação (Figura 3.1(b)).

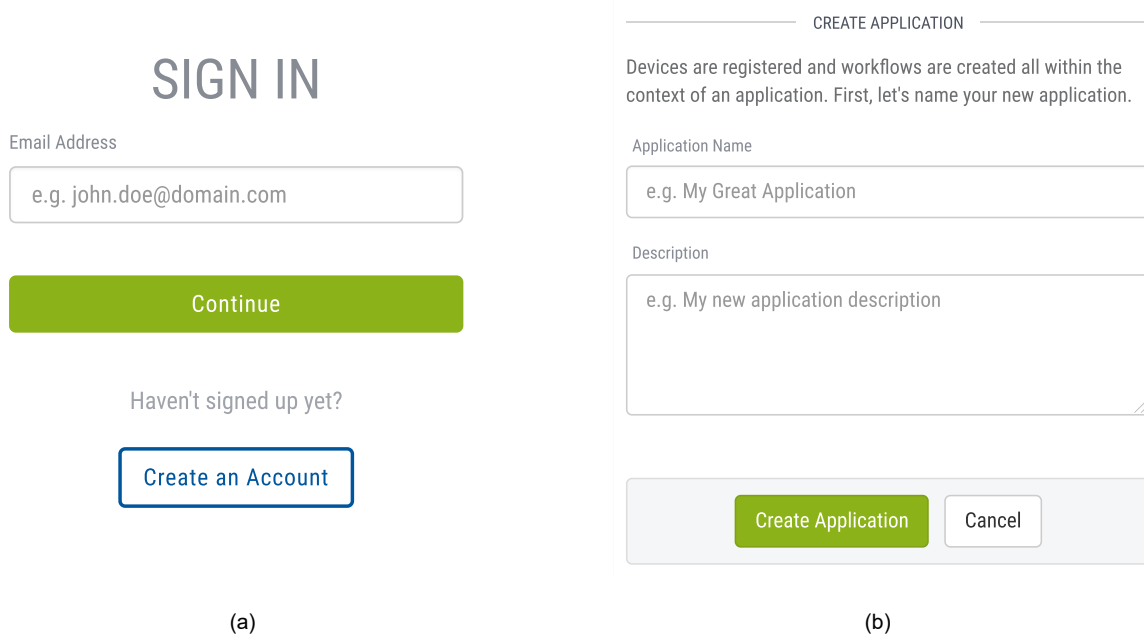


Figura 3.1: Criação da Conta e da Aplicação na plataforma WEGnology. (a) Página de login. (b) Página de criação da aplicação.

Após criar a aplicação, é aberta a tela principal da aplicação WEGnology, mostrada na Figura 3.2.

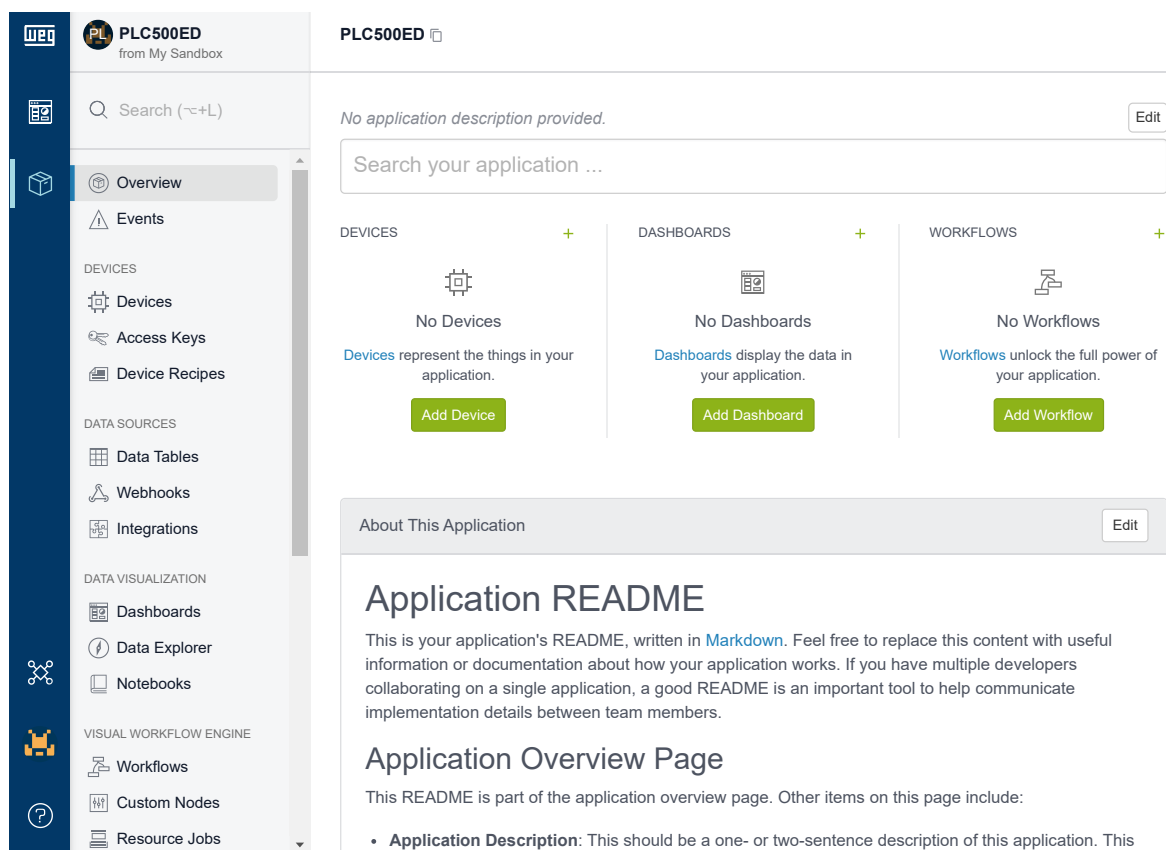


Figura 3.2: Tela principal da aplicação WEGnology.

3.1.2 Adicionar uma Chave de Acesso

Na tela principal, selecione **Access Keys > Add Access Key > Create Access Key**. Serão criados a **Access Key** e o **Access Secret**, que devem ser anotados ou salvos através do arquivo gerado pelo **Download File**. A Figura 3.3 mostra um exemplo de chave e segredo criados.

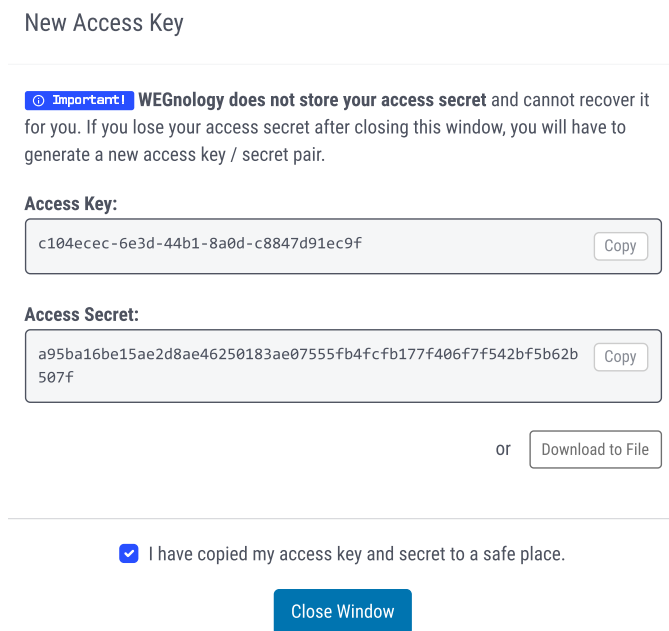


Figura 3.3: Criando uma nova Chave de Acesso.

3.1.3 API Token

Para a obtenção do **Token**, selecionar **API Tokens > Add Application Token > Create Application Token**. Será criado um novo **Application API Token**, que deve também ser anotado ou salvo através do arquivo gerado pelo **Download File**. A Figura 3.4 mostra um exemplo de Application API Token criado.

New Application Token

Important! WEGnology does not store your application token and cannot recover it for you. If you lose your application token after closing this window, you will have to generate a new application token.

Application API Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI2M2M1OWVhMmJhMG3NjEiLCJzY29wZSI6WyJhbGwuQXBwbiJlYXRpb24iXSwiaWF0IjoxNjc2ODk1NTg2LCJpc3MiOiJhcHAud25vbG9neS5pbyJ9.0SBVtkb61QAEbVh8_g2XmwAbFhRCJ4KjQfBP8kyQ
```

or

I have copied my application token to a safe place.

Figura 3.4: Criando um novo API Token.

3.1.4 Application ID

O **Application ID** pode ser encontrado em **Overview** e depois clicando no ícone de copiar ao lado do nome da aplicação. Também consta ao final da URL da página ao clicar em **Overview**, conforme mostra a Figura 3.5.

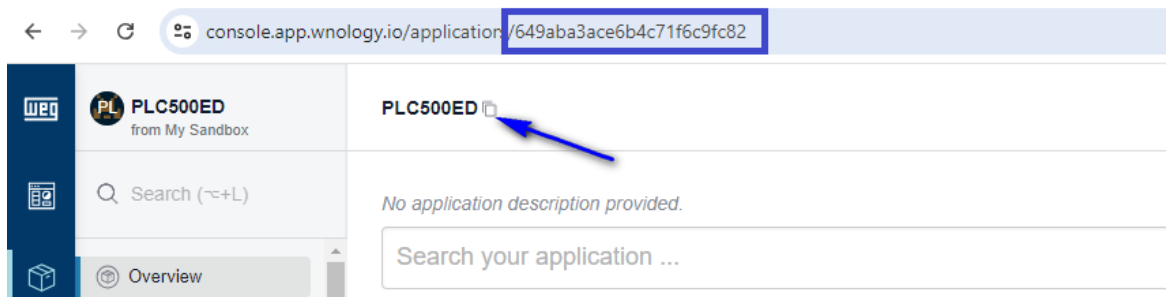


Figura 3.5: Obtendo o Application ID.

3.1.5 Conectando o Dispositivo

Com os dados obtidos acima em mãos, pode-se agora acessar a página web do produto e inseri-los no **Painel de Integração com a Nuvem**, na Página de Configuração, e salvar a configuração. Se os dados corretos forem inseridos, a Página de Estado mostrará o **Painel de Conexão MQTT** com os dados da aplicação WEGnology.

Após estes passos, o dispositivo deve ser conectado automaticamente à plataforma WEGnology. Para conferir, deve-se clicar em **Devices** e conferir se o dispositivo foi adicionado com sucesso. O ícone de um plug verde indica que o dispositivo está conectado. A Figura 3.6 mostra o dispositivo conectado à plataforma WEGnology.

Connection Status	Name	Device Class	Last Updated
Any Status	Filter by Name...	Any Class	Select...
Connected since 17 de jan de 2023 10:08:36	PLC500ED-21:B0:1F ID: 63c69dabec4a0f660941d292 PLC500ED - 21:B0:1F	Edge Compute	17 de jan de 2023 10:07

Figura 3.6: Dispositivo conectado à plataforma WEGnology.

3.1.6 Adicionando Atributos

Os dados de telemetria do dispositivo são armazenados em forma de **Atributos**. Para criar novos atributos e assim serem utilizados nas aplicações IoT, deve-se clicar em **Devices > PLC500ED-XX:YY:ZZ > Attributes > Add**. Assim que for definido o nome do atributo e o tipo de dado, clique em **Create Attribute**. O atributo está pronto para ser utilizado nas aplicações da plataforma.

4 CODESYS - BIBLIOTECA WEGNOLOGY

A biblioteca WEGnology foi desenvolvida para contribuir com a conectividade entre o software **Codesys**, plataforma **WEGnology** e container **Edge Agent**, responsável pelo processamento de borda do PLC500ED. Através dos blocos de funções e métodos disponíveis, a troca de informações entre estas interfaces é realizada de forma simples, permitindo a rápida configuração e utilização das funcionalidades disponíveis. A Figura 4.1 mostra a biblioteca WEGnology no Codesys.

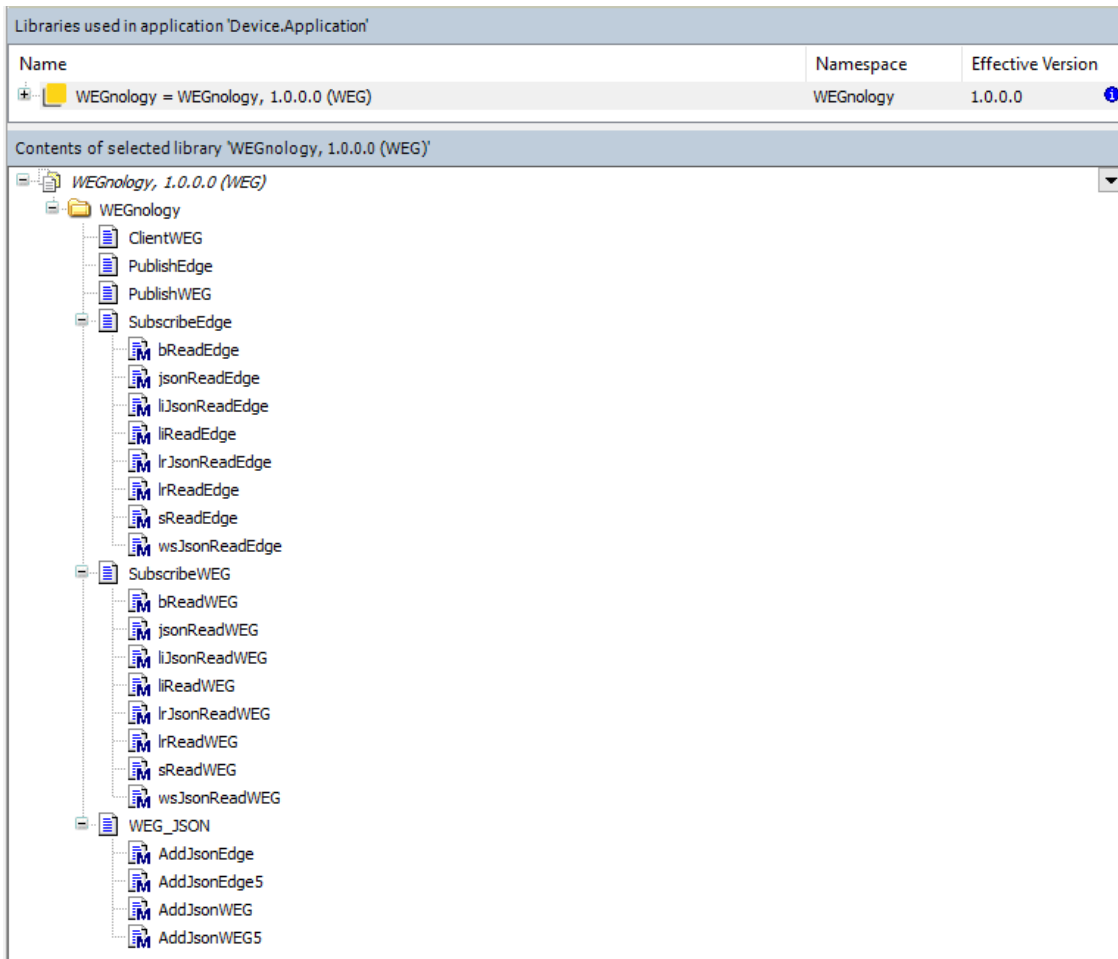


Figura 4.1: Biblioteca WEGnology.



NOTA!

A biblioteca **WEGnology** é instalada juntamente com o **WEG package**. Consulte o manual do PLC500 para mais informações.



NOTA!

Para que a biblioteca **WEGnology** seja utilizada no **Codesys**, é necessário primeiramente a configuração do **Painel de Integração com a Nuvem**, localizado na página web do produto.

4.1 RECOMENDAÇÕES

No Codesys, recomendam-se os seguintes cuidados quando implementando aplicações IoT:

- Utilização de tarefas específicas para publicação e subscrição de payloads MQTT, quando possível, e com baixa prioridade (20 - 31), de forma a evitar interferências em tarefas críticas. Obs: para o Codesys, a tarefa de maior prioridade é a 0 (zero).
- Não operar cargas críticas através de plataformas IoT, pois as mesmas podem possuir uma latência elevada, além da dependência de uma conexão estável com a internet.
- Para maiores informações sobre os blocos de funções e métodos da biblioteca WEGnology, consultar a ajuda da própria biblioteca no Codesys.

4.2 BIBLIOTECA CODESYS IIOT LIBRARIES SL

Para o funcionamento da biblioteca **WEGnology** é necessário a instalação prévia da biblioteca **CODESYS IIoT Libraries SL**, a qual inclui diversas outras bibliotecas e protocolos, tais como:

- MQTT Client (MQTT)
- Web Client (http, https)
- AWS IoT Core Client (MQTT)
- Azure IoT Hub Client (MQTT, https)
- Mail Service (POP3, SMTP)
- SNTP Service (SNTP)
- SNMP Library (SNMP)
- SMS Service (SMS)
- JSON Utilities
- XML Utility

Para a instalação desta biblioteca, abra o Codesys e acesse **Tools > CODESYS Installer**. Na nova janela aberta, clique em **Browse** e busque por “**IIoT**”. Selecione a biblioteca **IIoT Libraries SL** e clique em **Install**. Será necessário fechar a aplicação do Codesys antes de finalizar a instalação. A Figura 4.2 mostra a janela do CODESYS installer, a partir de onde a biblioteca CODESYS IIoT Libraries SL pode ser instalada.

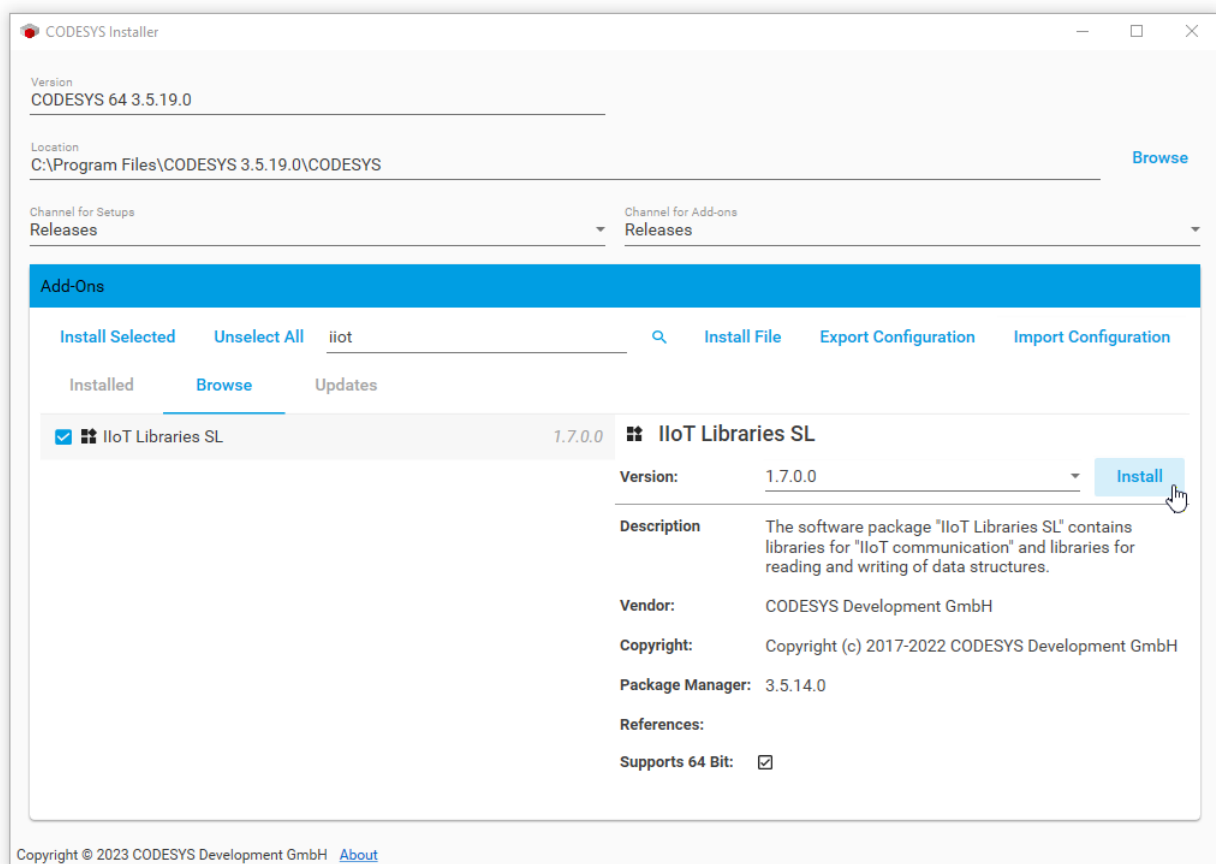


Figura 4.2: Instalação da biblioteca CODESYS IIoT Libraries SL.

4.3 SINTAXE DOS OBJETOS JSON PUBLICADOS

Os payloads MQTT publicados pelo Codesys são organizados na forma de objetos JSON. São utilizados padrões específicos de objetos JSON de acordo com a forma de utilização dos dados, pois podem ser utilizados diretamente pela plataforma WEGnology ou pelo processamento de borda, onde diversos dispositivos podem ser conectados ao PLC500ED. Seguem os padrões de objetos JSON.

4.3.1 WEGnology

A plataforma WEGnology recebe os dados publicados no seguinte formato:

```
{
  "data" :
  {
    "time" : "value",
    "attribute1" : "value1",
    "attribute2" : "value2",
    :
    "attributeN" : "valueN"
  }
}
```

No Codesys, o usuário precisa definir apenas os nomes dos atributos e as variáveis das quais seus valores são obtidos.

4.3.2 Edge Computing

O objeto JSON utilizado pelo processamento de borda possui um elemento específico para identificar o dispositivo de onde os dados são originados. No exemplo abaixo, tem-se um objeto JSON no padrão utilizado pelo processamento de borda, sendo o "device1" o nome do dispositivo:

```
{
  "data" :
  {
    "time" : "value",
    "device1" :
    {
      "attribute1" : "value1",
      "attribute2" : "value2",
      :
      "attributeN" : "valueN"
    }
  }
}
```

Para o processamento de borda, o usuário precisa definir no Codesys apenas o nome do dispositivo, nomes dos atributos e as variáveis das quais seus valores são obtidos.

Tópico de publicação: plc500ed/toAgent/state/<device_name>



NOTA!

Os nomes dos atributos publicados devem ser exatamente iguais aos cadastrados na plataforma WEGnology para que sejam reconhecidos adequadamente.



NOTA!

O bloco de função **WEG_JSON**, juntamente com seus métodos, organizam de forma automática os objetos JSON a serem publicados de acordo com a sintaxe correta.



NOTA!

Os payloads destinados ao processamento de borda devem ser tratados na plataforma WEGnology para identificar o dispositivo de onde os dados são originados. Ex.: device1, device2, etc.

4.4 BLOCOS DE FUNÇÕES E MÉTODOS MQTT

Os blocos de funções e métodos que compõem a biblioteca WEGnology são divididos de acordo com a forma de utilização dos dados. Eles podem ser utilizados diretamente pela plataforma WEGnology – sufixo **WEG** – ou pelo processamento de borda (Edge Computing) – sufixo **Edge**. Segue abaixo a descrição dos blocos de função e métodos disponíveis. Para mais detalhes, consultar a documentação da biblioteca WEGnology diretamente no Codesys.

- **ClientWEG**: Conecta o dispositivo ao broker interno, que por sua vez se conecta à plataforma WEGnology. Toda a aplicação IoT da biblioteca WEGnology necessita de um método ClientWEG, incluindo as aplicações Edge. Apenas uma instância do ClientWEG deve ser utilizada para vários blocos de funções de Publicar ou Subscriver.



NOTA!

Recomenda-se a utilização de tarefas com intervalos de tempo menores que **500 ms** para a execução do bloco **ClientWEG**.

- **PublishWEG**: Publica no tópico MQTT padrão da plataforma WEGnology. É publicado o JSON criado pelos métodos encontrados no bloco de função WEG_JSON, sufixo WEG.
- **PublishEdge**: Publica em um tópico MQTT customizado para ser utilizado pelo processamento de borda. É publicado o JSON criado pelos métodos encontrados no bloco de função WEG_JSON, sufixo Edge. Deve ser declarado apenas um PublishEdge para cada dispositivo conectado ao PLC500ED.
- **SubscribeWEG**: Subscrive ao tópico MQTT padrão da plataforma WEGnology. Apenas um SubscribeWEG deve ser declarado. Métodos específicos contidos no SubscribeWEG devem ser utilizados para cada tipo de variável. A Tabela 4.1 apresenta os métodos disponíveis no bloco de função SubscribeWEG.

Método	Descrição
bReadWEG	Recebe um comando do tipo BOOL da plataforma e atualiza a variável de interesse.
liReadWEG	Recebe um comando do tipo LINT da plataforma e atualiza a variável de interesse.
lrReadWEG	Recebe um comando do tipo LREAL da plataforma e atualiza a variável de interesse.
sReadWEG	Recebe um comando do tipo STRING da plataforma e atualiza a variável de interesse.
liJsonReadWEG	Recebe um objeto JSON da plataforma e atualiza uma variável do tipo LINT.
lrJsonReadWEG	Recebe um objeto JSON da plataforma e atualiza uma variável do tipo LREAL.
wsJsonReadWEG	Recebe um objeto JSON da plataforma e atualiza uma variável do tipo WSTRING.
jsonReadWEG	Recebe um objeto JSON da plataforma e procura por um Comando específico. Este método deve ser utilizado quando o JSON recebido possui uma hierarquia de dados com vários níveis. O objeto JSON recebido deve ser ainda tratado para isolar as variáveis de interesse.

Tabela 4.1: Métodos disponíveis no bloco de função SubscribeWEG.

- **SubscribeEdge**: Subscrive a um tópico MQTT customizado para ser utilizado pelo processamento de borda. Apenas um SubscribeEdge deve ser declarado para cada dispositivo conectado ao PLC500ED. Métodos específicos contidos no SubscribeEdge devem ser utilizados para cada tipo de variável. A Tabela 4.2 mostra os métodos disponíveis no bloco de função SubscribeEdge.

Método	Descrição
bReadEdge	Recebe um comando do tipo BOOL e atualiza a variável de interesse.
liReadEdge	Recebe um comando do tipo LINT e atualiza a variável de interesse.
lrReadEdge	Recebe um comando do tipo LREAL e atualiza a variável de interesse.
sReadEdge	Recebe um comando do tipo STRING e atualiza a variável de interesse.
liJsonReadEdge	Recebe um objeto JSON e atualiza uma variável do tipo LINT.
lrJsonReadEdge	Recebe um objeto JSON e atualiza uma variável do tipo LREAL.
wsJsonReadEdge	Recebe um objeto JSON e atualiza uma variável do tipo WSTRING.
jsonReadEdge	Recebe um objeto JSON e procura por um Comando específico. Este método deve ser utilizado quando o JSON recebido possui uma hierarquia de dados com vários níveis. O objeto JSON recebido deve ser ainda tratado para isolar as variáveis de interesse.

Tabela 4.2: Métodos disponíveis no bloco de função SubscribeEdge.

- **WEG_JSON**: Bloco de função que contém métodos para adicionar e atualizar atributos e valores ao objeto JSON esperado pela plataforma WEGnology ou pelo Edge Agent. Os métodos deste bloco de função aceitam como entradas os seguintes tipos de variáveis: BOOL, BYTE, WORD, SINT, INT, UINT, DINT, UDINT, LINT, REAL e LREAL. Os dados são convertidos para WSTRING. Estes métodos podem ser declarados diversas vezes, acrescentando novas variáveis ao JSON criado ou atualizando os valores das variáveis existentes. A Tabela 4.3 mostra os métodos disponíveis no bloco de funções SubscribeEdge.

Método	Descrição
AddJsonWEG	Método para criação de um JSON, adição de um atributo e atualização das variáveis referentes à plataforma WEGnology.
AddJsonWEG5	Método para criação de um JSON, adição de cinco atributos e atualização de variáveis referentes à plataforma WEGnology.
AddJsonEdge	Método para criação de um JSON, adição de atributos e atualização de variáveis referentes ao processamento de borda.
AddJsonEdge5	Método para criação de um JSON, adição de cinco atributos e atualização de variáveis referentes ao processamento de borda.

Tabela 4.3: Métodos disponíveis no bloco de funções WEG_JSON.

4.5 EXEMPLOS DE APLICAÇÕES

Nesta seção são apresentados exemplos de aplicações no Codesys – em texto estruturado (ST) e em Ladder – e na plataforma WEGnology para demonstrar algumas funcionalidades IoT do PLC500ED. O intervalo de tempo das tarefas é de 100 ms. São mostrados os seguintes exemplos:

- Publicação de payloads para a plataforma WEGnology
- Subscrição em tópicos da plataforma WEGnology
- Publicação de payloads para o Edge Agent
- Subscrição em tópicos do Edge Agent

4.5.1 Publicação de payloads para a plataforma WEGnology

O ClientWEG é conectado ao broker interno do PLC500ED, que por sua vez se conecta à plataforma WEGnology. Os valores das variáveis do tipo BOOL, UINT e REAL são alteradas a cada ciclo, e a cada 10 ciclos, seus valores são publicados na plataforma sob os atributos "temperature", "power" e "flag", respectivamente. Na Figura 4.3 é mostrada a declaração das variáveis e na Figura 4.4 o payload recebido pela plataforma. O programa completo em ST é apresentado na Figura 4.5, e em Ladder na Figura 4.6.

```

PublishWEG_PRG
PROGRAM PublishWEG_PRG
VAR
    // CLIENT
    WEG_CLIENT : WEGnology.ClientWEG;
    xErrorClient : BOOL;
    eErrorClient : MQTT.MQTT_ERROR;
    xConnectedToBroker : BOOL;

    // PUBLISH WEG
    WEG_PUBWEG : WEGnology.PublishWEG;
    eErrorPubWEG : MQTT.MQTT_ERROR;
    xDonePubWEG, xErrorPubWEG, xBusyPubWEG : BOOL;

    // WEG_JSON
    WEG_JSON : WEGnology.WEG_JSON;
    xErrorAdd : BOOL;
    xDoneAdd1, xDoneAdd2, xDoneAdd3 : BOOL;

    // PRG
    rTemperature : REAL;
    uiPower, uiCount : UINT;
    bFlag, uiPub : BOOL;
    RS_1 : RS;
    TON_1 : TON;
END_VAR
    
```

Figura 4.3: Declaração de variáveis PublishWEG.

Device PLC500ED-21:B0:1F reported its state
MQTT wnology/63c82eb9cad5c854a76acd/state
qui 19 de jan de 2023 12:36:29.752 GMT-03:00

Received Payload
▼ (root) {} 1 key
▼ "data": {} 4 keys
"flag": "0"
"power": "662"
"temperature": "6.620057"
"time": "1674142589178"

Device PLC500ED-21:B0:1F reported its state
MQTT wnology/63c82eb9cad5c854a76acd/state
qui 19 de jan de 2023 12:36:27.851 GMT-03:00

Received Payload
▼ (root) {} 1 key
▼ "data": {} 4 keys
"flag": "1"
"power": "643"
"temperature": "6.430053"
"time": "1674142587278"

Device PLC500ED-21:B0:1F reported its state
MQTT wnology/63c82eb9cad5c854a76acd/state
qui 19 de jan de 2023 12:36:25.951 GMT-03:00

Figura 4.4: Payload PublishWEG recebido pela plataforma.

PublishWEG_PRG - Structured text (ST)

```

// Set Reset
RS_1(SET := NOT xErrorClient, RESET1 := xErrorClient);

// Timer
TON_1(IN := RS_1.Q1, PT := T#1S);

// FB ClientWEG
WEG_CLIENT(
  xEnable := TON_1.Q,
  xConnectedToBroker => xConnectedToBroker,
  xError => xErrorClient,
  eErrorType => eErrorClient);

// REAL variable
rTemperature := rTemperature + 0.01;
IF rTemperature > 100 THEN rTemperature := -100; END_IF;

// UINT variable
uiPower := uiPower + 1;
IF uiPower > 10000 THEN uiPower := 0; END_IF;

// BOOL variable
bFlag := NOT bFlag;

// Publish the payload each 10 x Task interval
IF uiPub = 0 THEN
  uiCount := uiCount + 1;
  IF uiCount > 10 THEN uiCount := 0; uiPub := 1; END_IF;
ELSE
  // Add the bFlag value to the "flag" attribute
  WEG_JSON.AddJsonWEG(
    xExecute := NOT xDoneAdd1,
    wsVarName := "flag", anyVar := bFlag,
    AddJsonWEG => xDoneAdd1);

  // Add the rTemperature value to the "temperature" attribute
  WEG_JSON.AddJsonWEG(
    xExecute := NOT xDoneAdd2,
    wsVarName := "temperature", anyVar := rTemperature,
    AddJsonWEG => xDoneAdd2);

  // Add the uiPower value to the "power" attribute
  WEG_JSON.AddJsonWEG(
    xExecute := NOT xDoneAdd3,
    wsVarName := "power", anyVar := uiPower,
    AddJsonWEG => xDoneAdd3);

  // Publish the created JSON object
  WEG_PUBWEG(
    xExecute := ((NOT xErrorPubWEG) AND (NOT xDonePubWEG) AND (xConnectedToBroker)),
    xDone => xDonePubWEG,
    xBusy => xBusyPubWEG,
    xError => xErrorPubWEG,
    eErrorType => eErrorPubWEG);

  IF (xDonePubWEG = 1 OR xErrorPubWEG = 1) THEN uiPub := 0; END_IF;
END_IF;

```

Figura 4.5: Programa PublishWEG em texto estruturado.

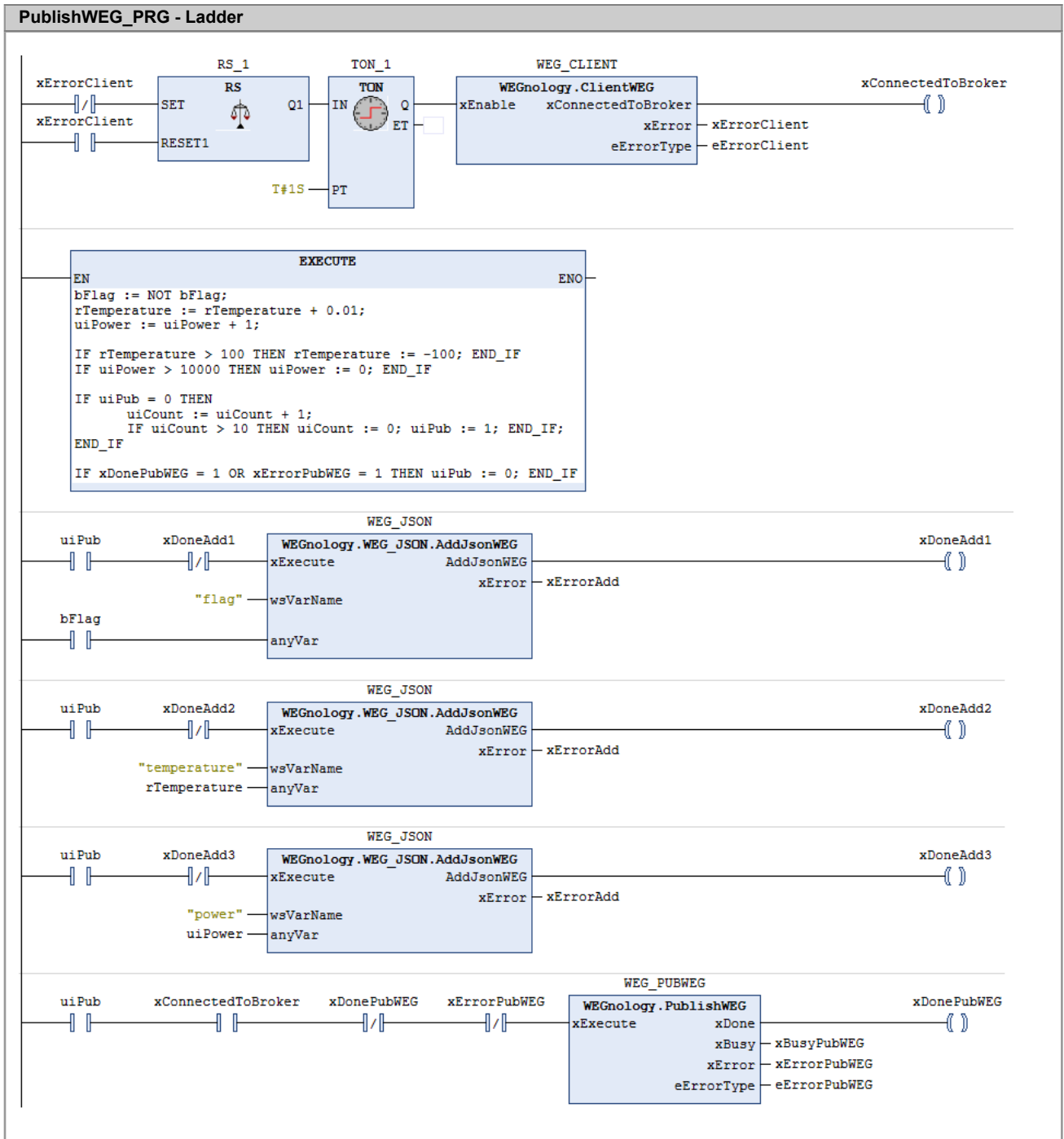


Figura 4.6: Programa PublishWEG em ladder.

4.5.2 Subscrição em tópicos da plataforma WEGnology

Primeiramente, é criada uma aplicação na plataforma WEGnology para a publicação dos payloads que serão recebidos pelo Codesys através do bloco SubscribeWEG e de seus métodos. Para isso, no painel principal da plataforma, ir em **Workflows > Application Workflows** e criar a aplicação. Na Figura 4.7 é mostrado o workflow implementado para este teste, o qual utiliza os nodes **Virtual Button** e **Device: Command**.

Quando um botão virtual é pressionado, é publicado um comando MQTT com um **Command Name** específico, juntamente com o **Command Payload** correspondente. Como exemplo, na Figura 4.7 são mostrados comandos que enviam um payload do tipo inteiro com o valor de 500, e um que envia um objeto JSON.

Na aplicação do Codesys, o ClientWEG é conectado ao broker interno do PLC500ED, que por sua vez se conecta à plataforma WEGnology. Quando a flag **xSubscribeActive** está em estado alto, o bloco SubscribeWEG está ativo e aguardando um payload. Variáveis do tipo LINT e LREAL são incrementadas a cada ciclo. Métodos do blobo SubscribeWEG aguardam payloads com atributos e tipos específicos. Quando a plataforma envia algum comando com uma variável válida, seu valor é imediatamente atualizado. A declaração das variáveis utilizadas no Codesys é mostrada na Figura 4.8. Na Figura 4.9 tem-se o programa em texto estruturado, e na Figura 4.10, o programa em Ladder.

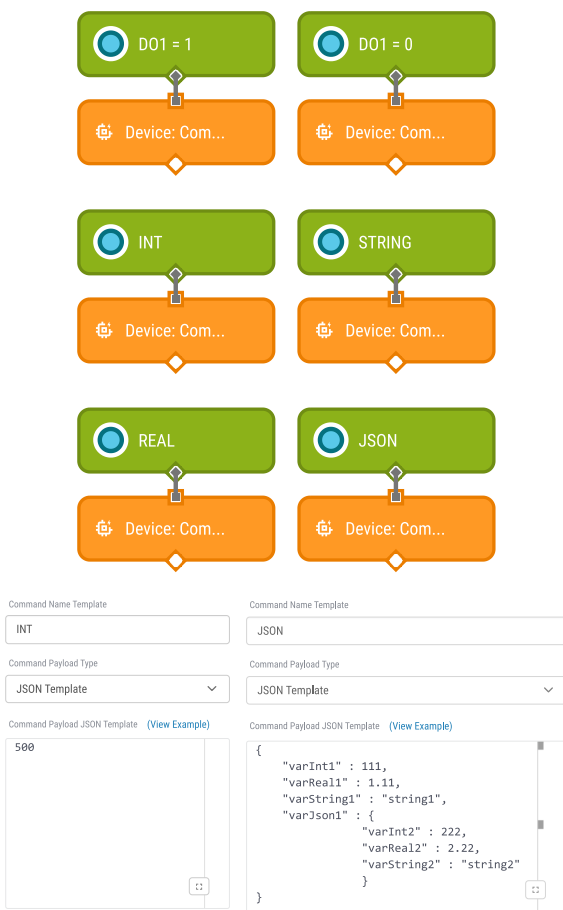


Figura 4.7: Workflow para envio de comandos SubscribeWEG.

```

SubscribeWEG_PRG

PROGRAM SubscribeWEG_PRG
VAR
  // CLIENT
  WEG_CLIENT : WEGnology.ClientWEG;
  xErrorClient : BOOL;
  eErrorClient : MQTT.MQTT_ERROR;
  xConnectedToBroker : BOOL;

  // SUBSCRIBE WEG
  WEG_SUBWEG : WEGnology.SubscribeWEG;
  eErrorSubWEG : MQTT.MQTT_ERROR;
  xErrorSubWEG, xBusySubWEG, xSubActiveWEG : BOOL;
  xReceivedSubWEG, xDonebSubWEG, xDoneliSubWEG : BOOL;
  xDoneIrSubWEG, xDonesSubWEG : BOOL;
  xDoneliJsonSubWEG, xDoneIrJsonSubWEG : BOOL;
  xDonesJsonSubWEG, xDoneJsonSubWEG : BOOL;

  // PRG
  bSubVar : BOOL;
  liSubVar, liJsonSubVar : LINT;
  lrSubVar, lrJsonSubVar : LREAL;
  sSubVar : STRING(JSON.GParams.g_diMaxStringSize);
  wsJsonSubVar : WSTRING(JSON.GParams.g_diMaxStringSize);
  wsPayloadJson : WSTRING(2048);
  udiSizeJson : UDINT;
  JSONDataJson : JSON.JSONData;
  RS_1 : RS;
  TON_1 : TON;
END_VAR
    
```

Figura 4.8: Declaração de variáveis SubscribeWEG.

SubscribeWEG_PRG - Structured text (ST)

```

// Set Reset
RS_1(SET := NOT xErrorClient, RESET1 := xErrorClient);

// Timer
TON_1(IN := RS_1.Q1, PT := T#1S);

// FB ClientWEG
WEG_CLIENT(
  xEnable := TON_1.Q,
  xConnectedToBroker => xConnectedToBroker,
  xError => xErrorClient,
  eErrorType => eErrorClient);

// LINT and LREAL variables
liSubVar := liSubVar + 1;
lrSubVar := lrSubVar + 0.01;
liJsonSubVar := liJsonSubVar + 1;
lrJsonSubVar := lrJsonSubVar + 0.01;

IF liSubVar > 1000 THEN liSubVar := 0; END_IF
IF liJsonSubVar > 1000 THEN liJsonSubVar := 0; END_IF
IF lrSubVar > 1000 THEN lrSubVar := -1000; END_IF
IF lrJsonSubVar > 1000 THEN lrJsonSubVar := -1000; END_IF

// Subscribe to the WEGnology platform
WEG_SUBWEG(
  xEnable := (NOT xErrorSubWEG) AND (NOT xReceivedSubWEG),
  xReceived => xReceivedSubWEG,
  xBusy => xBusySubWEG,
  xError => xErrorSubWEG,
  eErrorType => eErrorSubWEG,
  xSubscribeActive => xSubActiveWEG);

// Methods to receive a Command Name called "sVarName" with different types of payloads (BOOL, LINT, LREAL or STRING)
WEG_SUBWEG.bReadWEG(
  xExecute := NOT xDonebSubWEG, sVarName := 'DO1', bVar := bSubVar, bReadWEG => xDonebSubWEG);

WEG_SUBWEG.liReadWEG(
  xExecute := NOT xDoneliSubWEG, sVarName := 'INT', liVar := liSubVar, liReadWEG => xDoneliSubWEG);

WEG_SUBWEG.lrReadWEG(
  xExecute := NOT xDonelrSubWEG, sVarName := 'REAL', lrVar := lrSubVar, lrReadWEG => xDonelrSubWEG);

WEG_SUBWEG.sReadWEG(
  xExecute := NOT xDonesSubWEG, sVarName := 'STRING', sVar := sSubVar, sReadWEG => xDonesSubWEG);

// Methods to receive an object JSON, which Command Name is called "JSON", and it searches for a wsKey of different types (LINT, LREAL or WSTRING)
WEG_SUBWEG.liJsonReadWEG(xExecute := NOT xDoneliJsonSubWEG,
  wsCommandName := "JSON", wsKey := "varInt1", liVar := liJsonSubVar, liJsonReadWEG => xDoneliJsonSubWEG);

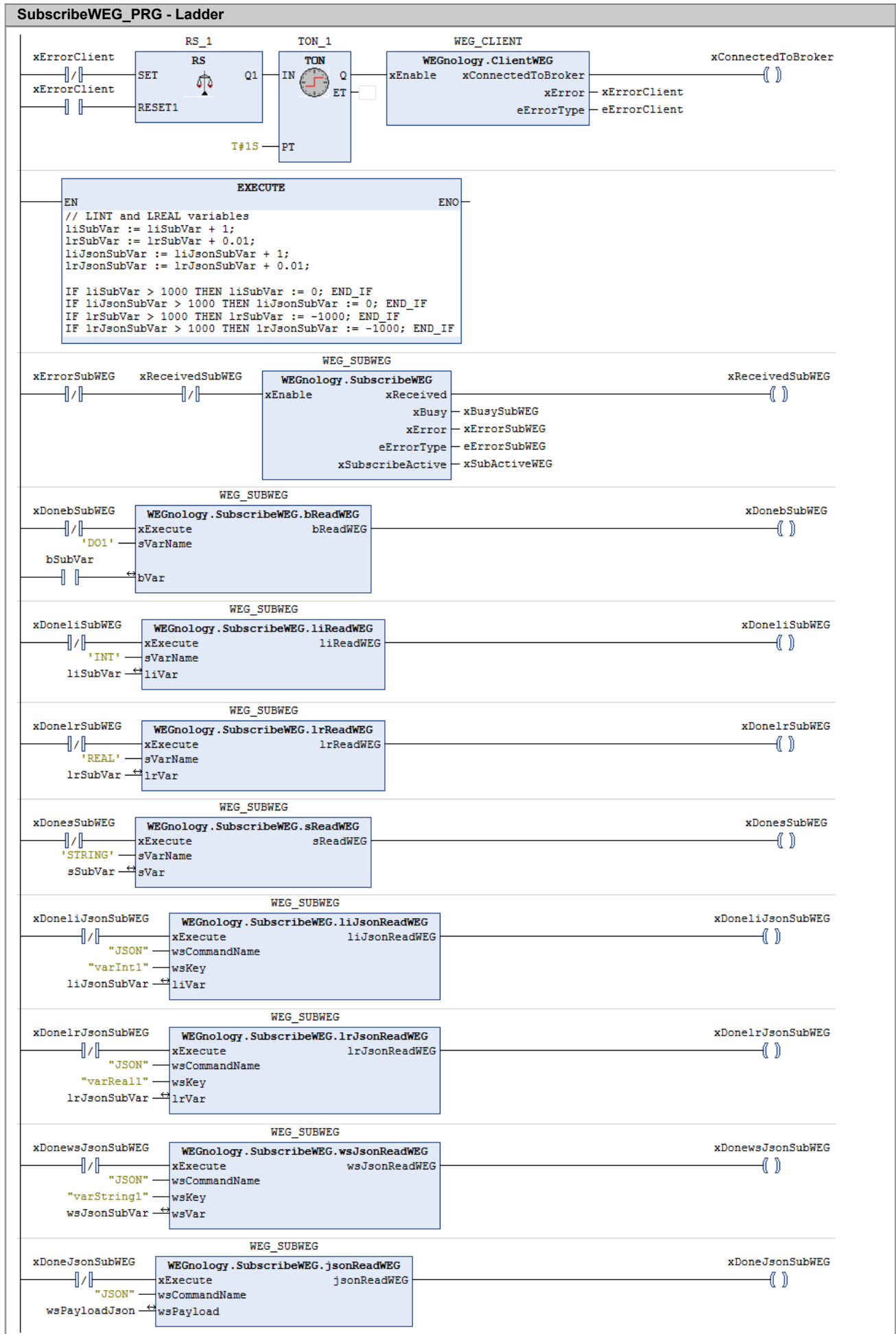
WEG_SUBWEG.lrJsonReadWEG(xExecute := NOT xDonelrJsonSubWEG,
  wsCommandName := "JSON", wsKey := "varReal1", lrVar := lrJsonSubVar, lrJsonReadWEG => xDonelrJsonSubWEG);

WEG_SUBWEG.wsJsonReadWEG(xExecute := NOT xDonewsJsonSubWEG,
  wsCommandName := "JSON", wsKey := "varString1", wsVar := wsJsonSubVar, wsJsonReadWEG => xDonewsJsonSubWEG);

// Methods to receive an object JSON. Its Command Name is called "JSON" and its payload must be treated.
WEG_SUBWEG.jsonReadWEG(xExecute := NOT xDoneJsonSubWEG,
  wsCommandName := "JSON", wsPayload := wsPayloadJson, jsonReadWEG => xDoneJsonSubWEG);

```

Figura 4.9: Programa SubscribeWEG em texto estruturado.



4.5.3 Publicação de payloads para o Edge Agent

Com a utilização do bloco de função PublishEdge é possível publicar payloads de diferentes dispositivos conectados ao PLC500ED. Os payloads são publicados nos tópicos `plc500ed/toAgent/state/<device_name>`, onde `<device_name>` é o nome do dispositivo de qual as variáveis são provenientes. É necessário definir um ponteiro para JSON para cada dispositivo conectado ao PLC, como pode ser visto na Figura 4.11, que mostra a declaração das variáveis da aplicação PublishEdge no Codesys.

De forma semelhante às demais aplicações, o bloco ClientWEG é utilizado para conectar o PLC500ED ao broker interno. Variáveis do tipo REAL, UINT e BOOL são alteradas a cada ciclo, simulando uma aplicação complexa. É criado um primeiro objeto JSON, proveniente do **device1**, com as variáveis `rTemperature1`, `uiPower1` e `bFlag1`, que são publicadas sob os atributos "temperature", "power" e "flag". O segundo JSON é referente ao **device2**, e é composto pela variável `rTemperature2`, publicada também sob o atributo "temperature". O último JSON é formado pela variável `uiPower3`, enviada sob o atributo "power", para o **device3**.



NOTA!

Para a utilização das funcionalidades de processamento de borda, a imagem do **Edge Agent** deve ser habilitada na página web do produto, em **Página de Configuração > Docker > On**.



NOTA!

Recomenda-se utilizar um intertravamento entre o PublisherEdge e os métodos AddJsonEdge, pois isso impede que o objeto JSON seja modificado enquanto o publisher tenta enviá-lo.

PublishEdge_PRG

```
PROGRAM PublishEdge_PRG
```

```
VAR
```

```
  // CLIENT
```

```
  WEG_CLIENT : WEGnology.ClientWEG;  
  xErrorClient : BOOL;  
  eErrorClient : MQTT.MQTT_ERROR;  
  xConnectedToBroker : BOOL;
```

```
  // PUBLISH EDGE
```

```
  WEG_PUBEDGE1, WEG_PUBEDGE2, WEG_PUBEDGE3 : WEGnology.PublishEdge;  
  xDonePubEdge1, xErrorPubEdge1, xBusyPubEdge1 : BOOL;  
  xDonePubEdge2, xErrorPubEdge2, xBusyPubEdge2 : BOOL;  
  xDonePubEdge3, xErrorPubEdge3, xBusyPubEdge3 : BOOL;  
  eErrorPubEdge1, eErrorPubEdge2, eErrorPubEdge3 : MQTT.MQTT_ERROR;
```

```
  // POINTER TO JSON - SEVERAL DEVICES
```

```
  JSONDataFactory1, JSONDataFactory2, JSONDataFactory3 : JSON.JSONDataFactory;  
  eFactoryError1, eFactoryError2, eFactoryError3 : FBF.ERROR;  
  pJsonData1 : POINTER TO JSON.JSONData := JSONDataFactory1.Create(eError => eFactoryError1);  
  pJsonData2 : POINTER TO JSON.JSONData := JSONDataFactory2.Create(eError => eFactoryError2);  
  pJsonData3 : POINTER TO JSON.JSONData := JSONDataFactory3.Create(eError => eFactoryError3);
```

```
  // WEG_JSON
```

```
  WEG_JSON : WEGnology.WEG_JSON;  
  xDoneAdd1, xDoneAdd2, xDoneAdd3 : BOOL;  
  xDoneAdd4, xDoneAdd5, xErrorAdd : BOOL;
```

```
  // PRG
```

```
  rTemperature1, rTemperature2 : REAL;  
  uiPower1, uiPower3 : UINT;  
  bFlag1 : BOOL := 0;  
  uiCount : UINT := 0;  
  RS_1 : RS;  
  TON_1 : TON;  
  timer : BLINK; // Blinking signal (turning on and off for specific durations)
```

```
END_VAR
```

Figura 4.11: Declaração de variáveis PublishEdge.

É criada uma aplicação na plataforma WEGnology para receber payloads publicados nos tópicos "plc500ed/toAgent/state/device1", "plc500ed/toAgent/state/device2" e "plc500ed/toAgent/state/device3" através do node **trigger MQTT**. Quando algo é recebido em algum destes tópicos, o objeto JSON integrante do payload é decodificado através do node **JSON: Decode**, e o resultado é mostrado pelo node de **Debug**. Na Figura 4.12 é mostrado o workflow criado na plataforma WEGnology e na Figura 4.13 tem-se os payloads recebidos dos três dispositivos através desta aplicação exemplo.

O programa completo da aplicação PublishEdge do Codesys, em texto estruturado, é mostrado na Figura 4.14, e o respectivo programa em Ladder, é apresentado na Figura 4.15.

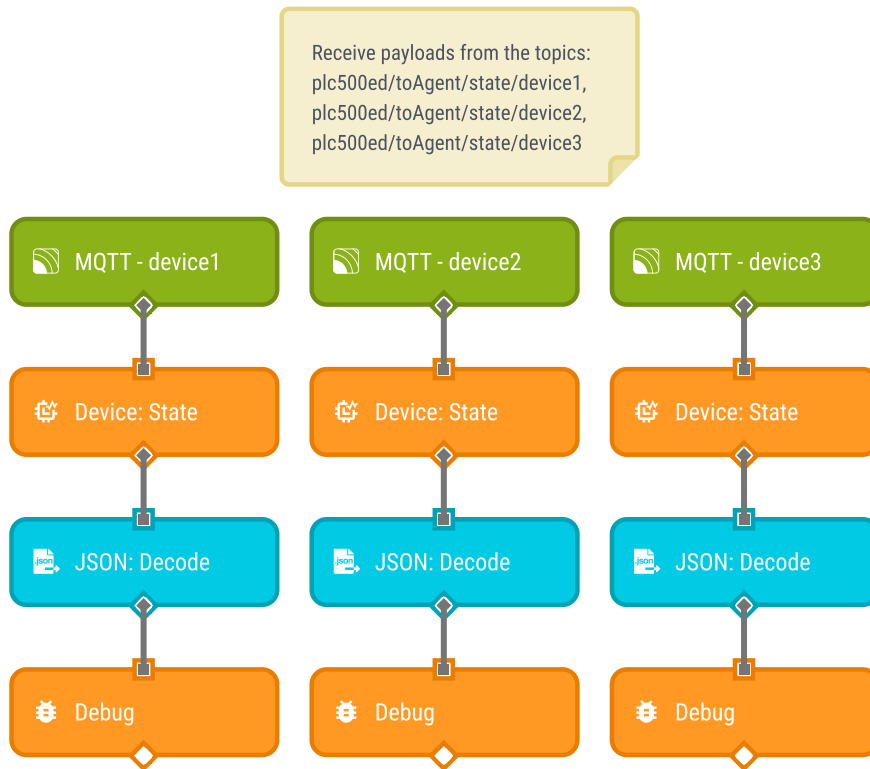


Figura 4.12: Workflow para recebimento de payloads PublishEdge.

PLC500ED-21:B0:1F / 2023-01-23T19-19-22 / Debug (12ms) Debug Node Output seg 23 de jan de 2023 16:20:46.743 GMT-03:00	PLC500ED-21:B0:1F / 2023-01-23T19-01-47 / Debug (31ms) Debug Node Output seg 23 de jan de 2023 16:14:56.927 GMT-03:00	PLC500ED-21:B0:1F / 2023-01-23T19-01-47 / Debug (15ms) Debug Node Output seg 23 de jan de 2023 16:14:56.931 GMT-03:00
<pre> Payload Timing ▼ data {} 1 key ▼ "data": {} 2 keys ▼ "device1": {} 3 keys "flag": "0" "power": "399" "temperature": "3.989997" "time": "1674501646667" </pre>	<pre> Payload Timing ▼ data {} 1 key ▼ "data": {} 2 keys ▼ "device2": {} 1 key "temperature": "21.02039" "time": "1674501293567" </pre>	<pre> Payload Timing ▼ data {} 1 key ▼ "data": {} 2 keys ▼ "device3": {} 1 key "power": "2102" "time": "1674501293568" </pre>

Figura 4.13: Payloads recebidos PublishEdge.

PublishEdge_PRG - Structured text (ST)

```

// Set Reset
RS_1(SET := NOT xErrorClient, RESET1 := xErrorClient);

// Timer
TON_1(IN := RS_1.Q1, PT := T#1S);

// FB ClientWEG
WEG_CLIENT( xEnable := TON_1.Q,
  xConnectedToBroker => xConnectedToBroker, xError => xErrorClient, eErrorType => eErrorClient);

// Increment of variables and limitations
rTemperature1 := rTemperature1 + 0.01; rTemperature2 := rTemperature2 + 0.01;
uiPower1 := uiPower1 + 1; uiPower3 := uiPower3 + 1; bFlag1 := NOT bFlag1;
IF rTemperature1 > 100 THEN rTemperature1 := -100; END_IF
IF rTemperature2 > 100 THEN rTemperature2 := -100; END_IF
IF uiPower1 > 10000 THEN uiPower1 := 0; END_IF
IF uiPower3 > 10000 THEN uiPower3 := 0; END_IF

// Boolean square wave (2s off, 2s on)
timer(ENABLE := 1, TIMELOW := T#2s, TIMEHIGH := T#2s);

// Add the bFlag1 value to the "flag" attribute of the "device1"
WEG_JSON.AddJsonEdge(
  xExecute := NOT xDoneAdd1, AddJsonEdge => xDoneAdd1, wsVarName := "flag", anyVar := bFlag1,
  wsDeviceName := "device1", pJsonData := pJsonData1);

// Add the rTemperature1 value to the "temperature" attribute of the "device1"
WEG_JSON.AddJsonEdge(
  xExecute := NOT xDoneAdd2, AddJsonEdge => xDoneAdd2, wsVarName := "temperature", anyVar := rTemperature1,
  wsDeviceName := "device1", pJsonData := pJsonData1);

// Add the uiPower1 value to the "power" attribute of the "device1"
WEG_JSON.AddJsonEdge(
  xExecute := NOT xDoneAdd3, AddJsonEdge => xDoneAdd3, wsVarName := "power", anyVar := uiPower1,
  wsDeviceName := "device1", pJsonData := pJsonData1);

// Add the rTemperature2 value to the "temperature" attribute of the "device2"
WEG_JSON.AddJsonEdge(
  xExecute := NOT xDoneAdd4, AddJsonEdge => xDoneAdd4, wsVarName := "temperature", anyVar := rTemperature2,
  wsDeviceName := "device2", pJsonData := pJsonData2);

// Add the uiPower3 value to the "power" attribute of the "device3"
WEG_JSON.AddJsonEdge(
  xExecute := NOT xDoneAdd5, AddJsonEdge => xDoneAdd5, wsVarName := "power", anyVar := uiPower3,
  wsDeviceName := "device3", pJsonData := pJsonData3);

// Publish the created JSON object of the "device1"
WEG_PUBEDGE1(
  xExecute := ((timer.OUT) AND (NOT xErrorPubEdge1) AND (NOT xDonePubEdge1) AND (xConnectedToBroker)),
  xDone => xDonePubEdge1, xBusy => xBusyPubEdge1, xError => xErrorPubEdge1, eErrorType => eErrorPubEdge1,
  pJsonData := pJsonData1);

// Publish the created JSON object of the "device2"
WEG_PUBEDGE2(
  xExecute := ((timer.OUT) AND (NOT xErrorPubEdge2) AND (NOT xDonePubEdge2) AND (xConnectedToBroker)),
  xDone => xDonePubEdge2, xBusy => xBusyPubEdge2, xError => xErrorPubEdge2, eErrorType => eErrorPubEdge2,
  pJsonData := pJsonData2);

// Publish the created JSON object of the "device3"
WEG_PUBEDGE3(
  xExecute := ((timer.OUT) AND (NOT xErrorPubEdge3) AND (NOT xDonePubEdge3) AND (xConnectedToBroker)),
  xDone => xDonePubEdge3, xBusy => xBusyPubEdge3, xError => xErrorPubEdge3, eErrorType => eErrorPubEdge3,
  pJsonData := pJsonData3);

```

Figura 4.14: Programa PublishEdge em texto estruturado.

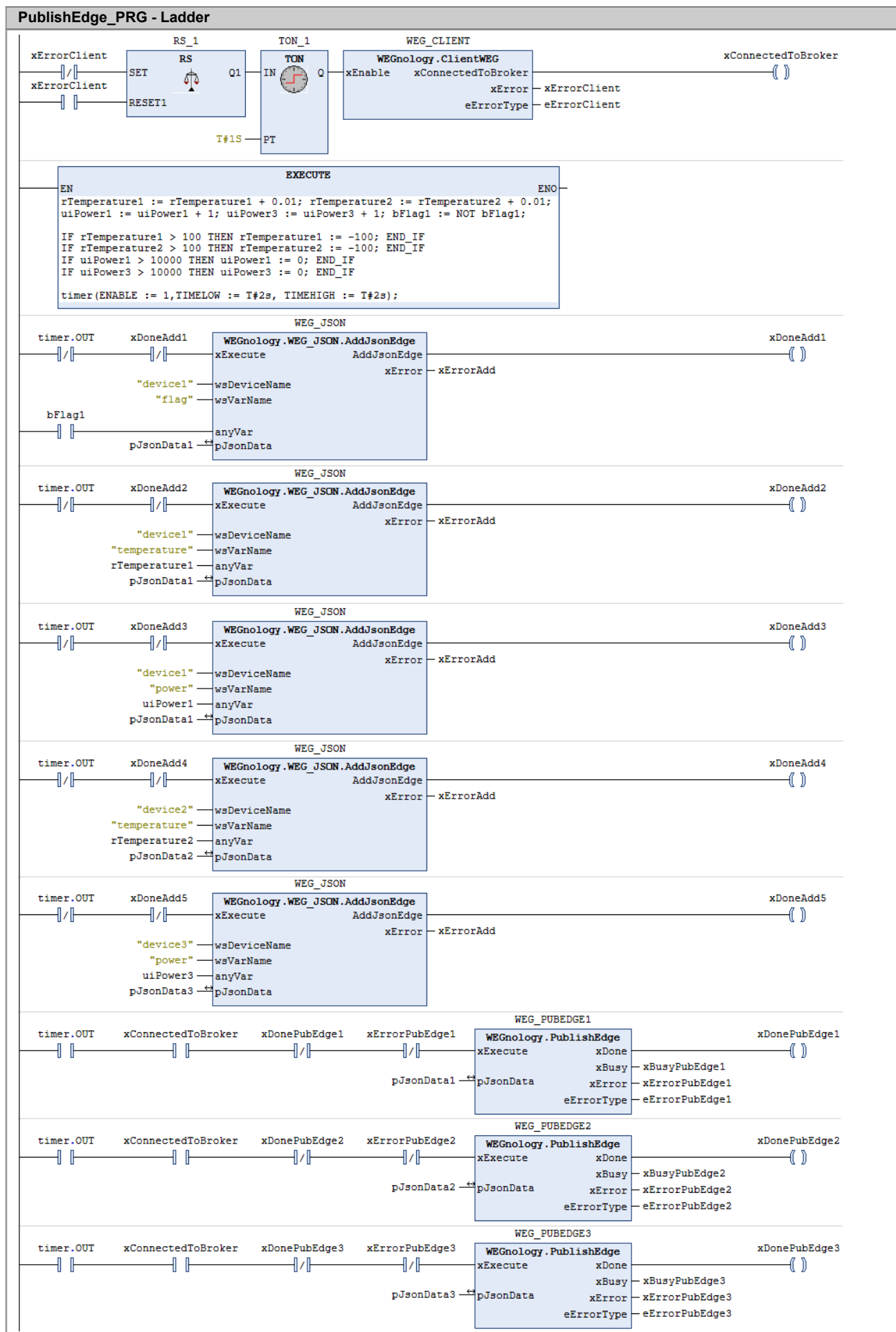


Figura 4.15: Programa PublishEdge em ladder.

4.5.4 Subscrição em tópicos do Edge Agent

Com a utilização do bloco SubscribeEdge é possível subscrever a diferentes tópicos MQTT, desde que possuam a sintaxe `plc500ed/fromAgent/command/<device_name>`, onde `<device_name>` é o nome do dispositivo de qual as variáveis são provenientes. São disponibilizados métodos para manipular variáveis de diferentes tipos.

Inicialmente, é criada uma aplicação na plataforma WEGnology para publicar payloads em três tópicos específicos, representando três diferentes dispositivos, através dos nodes de saída **MQTT**. É utilizado o trigger MQTT para fins de debug, através do qual é possível visualizar o payload enviado. A Figura 4.17 mostra o workflow implementado na plataforma.



Figura 4.16: Workflow para envio dos payloads SubscribeEdge.

Na Figura 4.17 são apresentados os payloads do tipo inteiro e do tipo objeto JSON configurados para publicação pelo workflow. Os demais nodes de saída MQTT seguem a mesma estrutura. A Figura 4.17(a) mostra o conteúdo dos nodes MQTT e a Figura 4.17(b) o debug realizado pela plataforma, confirmando a sintaxe correta.

Broker: WEGnology Broker

Topic Template: plc500ed/fromAgent/command/device1

Message Template:

```
{
  "name": "JSON",
  "payload": {
    "INT": 789,
    "REAL": 7.89
  }
}
```

Payload Timing

```
data {} 2 keys
  "name": "INT"
  "payload": 100
```

PLC500ED-21:00:1F / 2023-01-25T13:19:07 / Debug (8ms)

Debug Node Output

qua 25 de jan de 2023 10:54:32.150 GMT-03:00

Payload Timing

```
data {} 2 keys
  "name": "JSON"
  "payload": {} 2 keys
    "INT": 789
    "REAL": 7.89
```

(a)
(b)

Figura 4.17: Payloads publicados pelo workflow SubscribeEdge.

Na aplicação do Codesys, o bloco ClientWEG é utilizado para conectar o PLC500ED ao broker interno. Os blocos SubscribeEdge se inscrevem nos tópicos padrão referentes a cada dispositivo específico. Os métodos aguardam pela chegada dos payloads e verificam se o nome do atributo é válido. Se for, o valor da variável é atualizado. No exemplos em questão, são mostrados métodos para receberem variáveis do tipo BOOL, LINT, LREAL e objeto JSON. A Figura 4.18 apresenta a declaração das variáveis da aplicação SubscribeEdge.

O programa completo da aplicação SubscribeEdge do Codesys, em texto estruturado, é mostrado na Figura 4.20, e o respectivo programa em Ladder, é apresentado na Figura 4.21. Na Figura 4.19 é mostrada a aplicação SubscribeEdge em funcionamento, recebendo um JSON no tipo WSTRING.



NOTA!

Para a utilização das funcionalidades de processamento de borda, a imagem do **Edge Agent** deve ser habilitada na página web do produto, em **Página de Configuração > Docker > On.**

```
SubscribeEdge_PRG
PROGRAM SubscribeEdge_PRG
VAR
  // CLIENT
  WEG_CLIENT : WEGnology.ClientWEG;
  xErrorClient : BOOL;
  eErrorClient : MQTT.MQTT_ERROR;
  xConnectedToBroker : BOOL;
  RS_1 : RS;
  TON_1 : TON;

  // SUBSCRIBE EDGE
  WEG_SUBEDGE1, WEG_SUBEDGE2, WEG_SUBEDGE3 : WEGnology.SubscribeEdge;
  xErrorSubEdge1, xBusySubEdge1, xSubActiveEdge1, xReceivedSubEdge1 : BOOL;
  xErrorSubEdge2, xBusySubEdge2, xSubActiveEdge2, xReceivedSubEdge2 : BOOL;
  xErrorSubEdge3, xBusySubEdge3, xSubActiveEdge3, xReceivedSubEdge3 : BOOL;
  eErrorSubEdge1, eErrorSubEdge2, eErrorSubEdge3 : MQTT.MQTT_ERROR;
  xDonebSubEdge1, xDoneliSubEdge1, xDonelrSubEdge1, xDoneJsonSubEdge1 : BOOL;
  xDoneliSubEdge2, xDonelrSubEdge3 : BOOL;
  wsPayloadSubEdge1 : WSTRING(2048);

  // PGR
  bSubVar1 : BOOL;
  liSubVar1, liSubVar2 : LINT;
  lrSubVar1, lrSubVar3 : LREAL;
END_VAR
```

Figura 4.18: Declaração de variáveis SubscribeEdge.

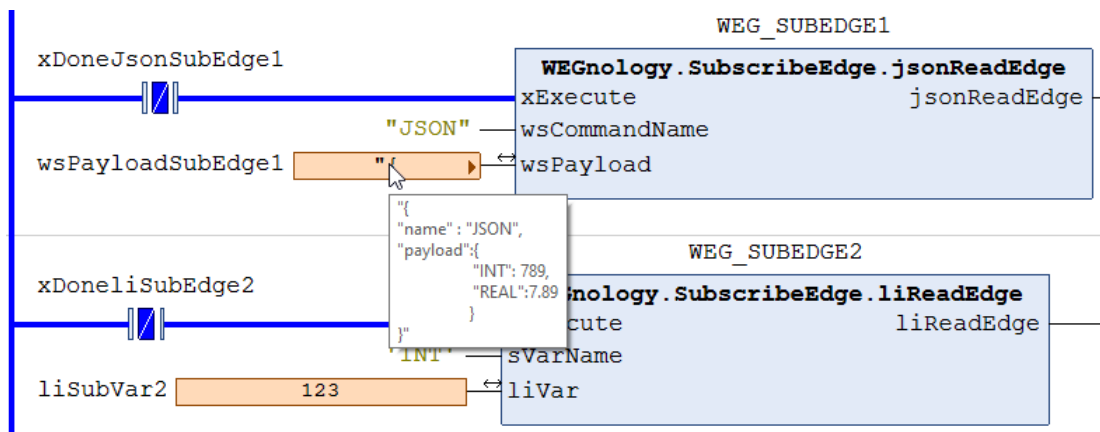


Figura 4.19: Aplicação SubscribeEdge recebendo um JSON no tipo WSTRING.

SubscribeEdge_PRG - Structured text (ST)

```

// Set Reset
RS_1(SET := NOT xErrorClient, RESET1 := xErrorClient);

// Timer
TON_1(IN := RS_1.Q1, PT := T#1S);

// FB ClientWEG
WEG_CLIENT( xEnable := TON_1.Q,
  xConnectedToBroker => xConnectedToBroker, xError => xErrorClient, eErrorType => eErrorClient);

// Subscribe to the "device1"
WEG_SUBEDGE1( xEnable := (NOT xErrorSubEdge1) AND (NOT xReceivedSubEdge1),
  xBusy => xBusySubEdge1, XError => xErrorSubEdge1, eErrorType => eErrorSubEdge1,
  xReceived => xReceivedSubEdge1, xSubscribeActive => xSubActiveEdge1,
  wsDeviceName := "device1");

// Subscribe to the "device2"
WEG_SUBEDGE2( xEnable := (NOT xErrorSubEdge2) AND (NOT xReceivedSubEdge2),
  xBusy => xBusySubEdge2, XError => xErrorSubEdge2, eErrorType => eErrorSubEdge2,
  xReceived => xReceivedSubEdge2, xSubscribeActive => xSubActiveEdge2,
  wsDeviceName := "device2");

// Subscribe to the "device3"
WEG_SUBEDGE3( xEnable := (NOT xErrorSubEdge3) AND (NOT xReceivedSubEdge3),
  xBusy => xBusySubEdge3, XError => xErrorSubEdge3, eErrorType => eErrorSubEdge3,
  xReceived => xReceivedSubEdge3, xSubscribeActive => xSubActiveEdge3,
  wsDeviceName := "device3");

// Methods to receive attributes with different types of payloads (BOOL, LINT, LREAL and JSON) from the device1
WEG_SUBEDGE1.bReadEdge(
  xExecute := NOT xDonebSubEdge1, sVarName := 'DO1', bVar := bSubVar1, bReadEdge => xDonebSubEdge1);

WEG_SUBEDGE1.liReadEdge(
  xExecute := NOT xDoneliSubEdge1, sVarName := 'INT', liVar := liSubVar1, liReadEdge => xDoneliSubEdge1);

WEG_SUBEDGE1.lrReadEdge(
  xExecute := NOT xDonelrSubEdge1, sVarName := 'REAL', lrVar := lrSubVar1, lrReadEdge => xDonelrSubEdge1);

WEG_SUBEDGE1.jsonReadEdge(
  xExecute := NOT xDoneJsonSubEdge1, wsCommandName := "JSON", wsPayload := wsPayloadSubEdge1, jsonReadEdge =>
xDoneJsonSubEdge1);

// Method to receive an attribute of the LINT type from the device2
WEG_SUBEDGE2.liReadEdge(
  xExecute := NOT xDoneliSubEdge2, sVarName := 'INT', liVar := liSubVar2, liReadEdge => xDoneliSubEdge2);

// Method to receive an attribute of the LREAL type from the device3
WEG_SUBEDGE3.lrReadEdge(
  xExecute := NOT xDonelrSubEdge3, sVarName := 'REAL', lrVar := lrSubVar3, lrReadEdge => xDonelrSubEdge3);

```

Figura 4.20: Programa SubscribeEdge em texto estruturado.

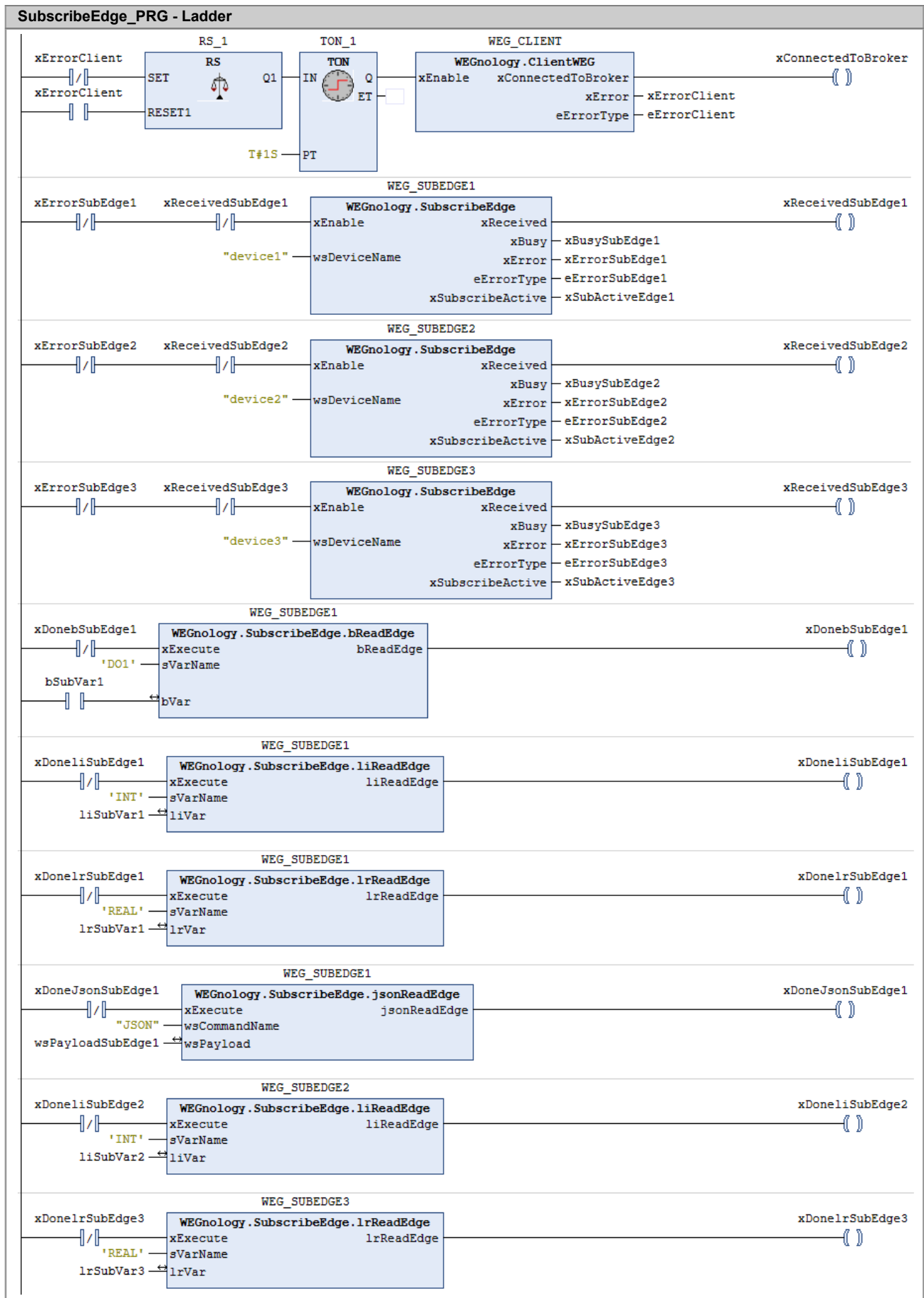


Figura 4.21: Programa SubscribeEdge em ladder.

5 GUIA DE INÍCIO RÁPIDO

Nesta seção é mostrado de forma resumida como podem ser feitas as configurações básicas do produto para estabelecer uma conexão com a plataforma WEGnology e assim rodar uma primeira aplicação IoT. Para mais detalhes, consulte o restante desta Nota de Aplicação e o Manual do Usuário do PLC500, disponível em www.weg.net.

O início rápido pode ser realizado basicamente em cinco etapas:

5.1 ESTABELECEER A CONEXÃO COM O PRODUTO

Para ter acesso ao dispositivo, seja pela página web ou pelo Codesys, o usuário precisa colocar o computador ou dispositivo móvel na mesma rede do produto. O primeiro acesso pode ser feito via USB ou ETH. Seguem os IPs iniciais das interfaces:

Tabela 5.1: Endereços de IP padrão.

Conexão	Endereço de IP padrão
ETH1	192.168.1.10
ETH2	192.168.2.10
USB2	192.168.234.234

5.2 CONECTAR O PRODUTO NA INTERNET VIA CODESYS OU WEBPAGE

Com o dispositivo conectado na mesma rede do produto, o usuário pode configurar o acesso à internet através da página web ou do Codesys.

- **Codesys:** através da aba **Setup**, configure a respectiva ETH como **DHCP** e faça o download da aplicação.
- **Webpage:** na **Página de Configuração** da página web, configure a respectiva ETH. Marque **Yes** na opção **DHCP** e em **Default Route**.

Após todos os campos terem sido preenchidos, salve a configuração clicando no botão **Save Configuration**. Uma janela se abrirá indicando que a aplicação será reiniciada. Aguarde até que o processo seja finalizado e a tela de login apareça novamente.

A conexão com a internet foi estabelecida quando a flag **Internet Status** aparecer como **Connected**, na **Página de Estado > Informações do Sistema**, conforme mostra a Figura 5.1.

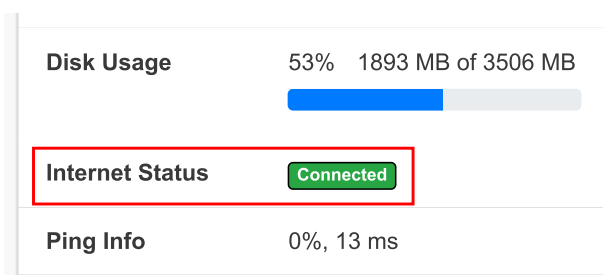


Figura 5.1: Conexão com a internet realizada com sucesso.

5.3 CONFIGURAR A APLICAÇÃO IOT PELA WEBPAGE

Para realizar a configuração da aplicação IoT, é necessário que a conexão com a internet esteja ativa, conforme mostrado no item anterior. Além disso, a aplicação IoT deve ser previamente criada na plataforma [WEGnology](#).

A configuração das credenciais é realizado na **Página de Configuração > Painel de Integração com a Nuvem** da página web. Existem duas formas de configurar as credenciais de acesso:

- **Preenchimento Manual:** preencher os cinco campos descritos abaixo:
 - **Integrator:** WEGnology v1 - serviço de integração de nuvem.
 - **Application ID:** ID de uma aplicação existente no WEGnology.
 - **API Token:** Token de permissão de gerenciamento da aplicação
 - **Access Key:** Chave de autenticação para estabelecer a conexão do dispositivo na aplicação do WEGnology.
 - **Secret:** Senha de autenticação para estabelecer a conexão do dispositivo na aplicação do WEGnology.
- **Upload de arquivos:** as informações de **Token** e de **Access Keys** podem ser carregados por meio de arquivos .txt contendo tais credenciais. Para isso, basta clicar em **Choose File** e selecionar os arquivos correspondentes (que devem ser baixados no momento da criação da aplicação IoT).

Com as configurações realizadas, clique no botão **Save Configuration** para salvar e reiniciar a aplicação. Após o término deste processo, a tela de login será carregada novamente. O estado da configuração da conexão com a plataforma pode ser conferida em **MQTT Configuration Status**, conforme Figura 5.2.

The screenshot displays the 'CLOUD INTEGRATION' configuration interface. It includes a dropdown menu for 'Integrator' set to 'WEGnology v1'. Below it are input fields for 'Application ID' (62d9388ff49a970b1e7758da), 'API Token' (masked with dots), 'Access Key' (618ef46b-9a7a-4329-9c30-bc58e1be7d65), and 'Secret' (masked). To the right, there are two optional file selection fields: 'Token file: (optional)' and 'Access Keys file: (optional)', both with 'Escolher arquivo' buttons and the text 'Nenhum arq... escolhido'. A red rectangular box highlights the 'MQTT Configuration Status' indicator, which shows 'Locally Configured' in a blue pill-shaped button.

Figura 5.2: Conexão com a plataforma WEGnology realizada com sucesso.

5.4 HABILITAR UMA IMAGEM DOCKER

Na aba **Página de Configuração > Docker** é possível habilitar/desabilitar imagens de acordo com a necessidade do usuário. Para isso, basta alterar o **Status** entre **On** e **Off**, e então clicar no botão **Save Configuration** para salvar e reiniciar a aplicação.

Na versão de firmware **1.2.0** do PLC500ED, e **2.3.0** do Coreapp, é disponibilizado por padrão o container **Edge Agent** versão **1.34.0**.

A Figura 5.3 mostra a Página de Estado do produto, onde percebe-se que o dispositivo está conectado à internet, com uma aplicação IoT configurada e com a imagem do container Edge Agent ativa.

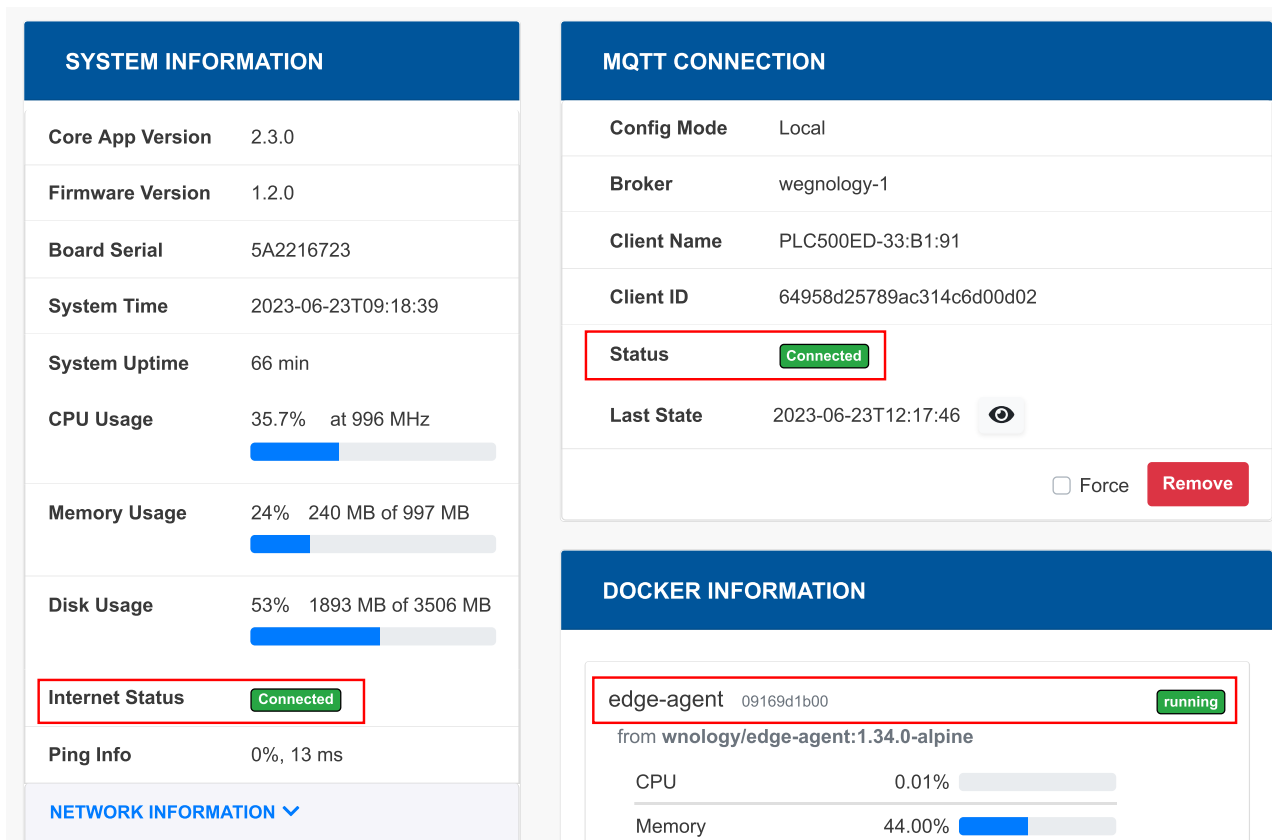


Figura 5.3: Dispositivo conectado à internet, com uma aplicação IoT configurada e uma imagem Docker ativa.

5.5 CRIAÇÃO DA APLICAÇÃO

Com o produto conectado à internet e com uma aplicação IoT configurada, o próximo passo é a criação da aplicação. O PLC500ED permite diversos tipos de aplicações, tais como:

- **Application Workflows:** Esses workflows são executados na nuvem da WEGnology e são para fins gerais de processamento e tratamento de dados. Eles oferecem uma ampla variedade de triggers para facilitar a aquisição e o manuseio de dados de várias fontes ou serviços de dados.
- **Edge Workflows:** Os Edge Workflows rodam no próprio hardware do PLC500ED e são executados pelo container Edge Agent. Eles fornecem nodes e triggers adicionais projetados especificamente para interagir com periféricos e fontes de dados locais.
- **Codesys:** Utilizando a biblioteca **WEGnology** para o Codesys, é possível utilizar o protocolo MQTT para publicar e subscrever em tópicos da plataforma **WEGnology**, bem como do container **Edge Agent**, conforme a necessidade. Os blocos de funções que comunicam diretamente com a plataforma possuem o sufixo **WEG**, e os utilizados com o Edge Agent possuem o sufixo **Edge**.

Exemplos de aplicações podem ser encontrados no Capítulo 4, seção 4.5, disponibilizados nas linguagens **Ladder** e **Texto Estruturado (ST)**.



NOTA!

Os três tipos de aplicações mencionados podem ser utilizados simultaneamente, levando em consideração as necessidades individuais do usuário, possibilitando aplicações versáteis e personalizadas.



WEG Drives & Controls - Automação LTDA.
Jaraguá do Sul - SC - Brasil
Fone 55 (47) 3276-4000 - Fax 55 (47) 3276-4020
São Paulo - SP - Brasil
Fone 55 (11) 5053-2300 - Fax 55 (11) 5052-4212
automacao@weg.net
www.weg.net