UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA – DAELN
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA – DAELT
CURSO SUPEREIOR DE TECNOLOGIA EM MECATRÔNICA INDUSTRIAL
CURSO SUPEREIOR DE TECNOLOGIA EM AUTOMAÇÃO INDUSTRIAL

ANDRÉ LUIS BRUGNEROTTO
WAGNER LUIZ KUME

REDE DE CONTROLE E SUPERVISÃO PARA INVERSORES DE FREQUÊNCIA UTILIZANDO LABVIEW

PROJETO DE PESQUISA

CURITIBA
2014

ANDRÉ LUIS BRUGNEROTTO
WAGNER LUIZ KUME

REDE DE CONTROLE E SUPERVISÃO PARA INVERSORES DE FREQUÊNCIA UTILIZANDO LABVIEW

Trabalho de Conclusão de Curso de graduação apresentado à disciplina de Trabalho de Diplomação, realizado em conjunto envolvendo as coordenações dos Cursos Superiores de Tecnologia em Mecatrônica Industrial ligado aos Departamentos de Eletrônica e Mecânica e de Tecnologia em Automação Industrial, do Departamento Acadêmico de Eletrotécnica – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.
Orientador: Prof. Walter Sanchez, Dr.

CURITIBA
2014

ANDRÉ LUIS BRUGNEROTTO
WAGNER LUIZ KUME

REDE DE CONTROLE E SUPERVISÃO PARA INVERSORES DE FREQUÊNCIA UTILIZANDO LABVIEW

Este Trabalho de Diplomação foi julgado e aprovado como requisito parcial para a obtenção do Título de **Tecnólogo em Mecatrônica Industrial e Tecnólogo em Automação Industrial**, Modalidade Automação industrial, do **Curso Superior de Tecnologia em Mecatrônica e Curso Superior de Tecnologia em Automação Industrial** da **Universidade Tecnológica Federal do Paraná**.

Curitiba, 13 de dezembro de 2013

Prof. Milton Luiz Polli, Dr.
Coordenador de Curso
Departamento Acadêmico de Mecânica

Prof. José da Silva Maia, M. Eng.
Coordenador de Curso
Departamento Acadêmico de Eletrotécnica

Prof. Sérgio Moribe, Esp.
Responsável pela atividade de Trabalho de conclusão de Curso
Departamento Acadêmico de Eletrônica

Prof. Rafael Fontes Souto, M. Eng.
Responsável pelo Trabalho de Diplomação da Tecnologia
Departamento Acadêmico de Eletrotécnica

BANCA EXAMINADORA

Prof. Walter Sanchez, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

Prof. José Silva Maia, M.Eng.
Universidade Tecnológica Federal do Paraná

Prof. Jorge Assade Leludak, M.Sc.
Universidade Tecnológica Federal do Paraná

Prof. Luiz Fernando Copetti, M.Sc.
Universidade Tecnológica Federal do Paraná

Prof. Carlos Henrique da Silva, Dr.
Universidade Tecnológica Federal do Paraná

RESUMO

BRUGNEROTTO, André L.; KUME, Wagner L. **Rede de Controle e Supervisão para Inversores de Frequência Utilizando LabVIEW**. 2013. 96f. Trabalho de conclusão de Curso (Curso superior de Tecnologia em Mecatrônica Industrial e Tecnologia em Automação Industrial) UTFPR – Universidade Tecnológica Federal do Paraná. 2014.

Tempo de produção sempre foi um aspecto importante tratando-se de indústrias por refletir diretamente nos lucros gerados. Visto a crescente necessidade de maior velocidade em processos como o de manutenção de máquinas e linhas de produção, ou seja, otimização e melhor aproveitamento do tempo desprendido em tarefas auxiliares à produção escolhemos um equipamento vastamente usado em qualquer indústria e o tornamos o objeto de análise deste trabalho. O Inversor de frequência é um equipamento que necessita ser programado para o fim desejado e parametrizado por uma pessoa qualificada para conseguir usá-lo para suprir uma necessidade. Usando o *software* LabVIEW para criar uma interface para ser executado em computadores pessoais ou industriais, capaz de parametrizar um inversor remotamente, o qual possa estar ligado em uma rede industrial. O programa é o principal foco do trabalho e é onde toda a inteligência de funcionamento do projeto está contida. Os resultados pretendidos são reduzir o tempo de parametrização do inversor de frequência, salvar um arquivo de *backup* dos parâmetros para uso futuro caso seja necessário realizar a substituição rápida do equipamento em uma manutenção e assegurar a correta parametrização do equipamento por diminuir a intervenção humana.

Palavras-chave: Sistema Supervisório. LabVIEW. Inversor de frequência. Protocolo ModBus.

ABSTRACT

BRUGNEROTTO, André L.; KUME, Wagner L. **Network Control and Supervision for Frequency Inverters Using LabVIEW**. 2013. 96f. Course Final Assignment (Technology in Industrial Mechatronics and Technology in Industrial Automation), UTFPR - Technological Federal University of Paraná. 2014.

Production time has always been an important aspect in the case of industries, as it reflects directly on the profits generated. Since the growing need for higher speed processes such as maintenance of machines and production lines, in other words, optimization and best use of time in detached ancillary tasks of production we chose a widely used equipment in any industry and become the object of analysis in this work. The frequency inverter is a device that must be programmed to the desired end, parameterized by a qualified person to be able to use it to meet a need. Using the *software* LabVIEW to create an interface to run on personal or industrial computers, able to parameterize a drive that can be remotely connected in an industrial network. The program is the focus of the project and is where all the intelligence of operation of the project is contained. The intended results is to reduce the time parameterization of the frequency inverter, save a backup file of the parameters for future use if necessary to make the quick replacement of equipment in a maintenance and ensure the correct parameterization of the equipment by reducing human intervention.

Key words: Supervisory system. LabVIEW. Frequency inverter. Protocol ModBus.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CA	Corrente Alternada
CC	Corrente Contínua
CLP	Controlador Lógico Programável
IHM	Interface Homem-Máquina
LabVIEW	*Laboratory Virtual Instrument Engineering Workbench*)
VI	Virtual Instrument

SUMÁRIO

1. INTRODUÇÃO

1.1. TEMA

A necessidade de criação de linhas de produção onde se tenha alta flexibilidade e a possibilidade de alterar a produção da linha em pouco tempo é um grande desafio para as instalações fabris.

> Uma sequência de produtos, todos diferentes, mas dentro do "pacote" de capacitações do sistema, poderia ser processada em qualquer ordem, e sem demora para troca entre os produtos. (SLACK et al., 2002).

O presente estudo surge a partir da descentralização do controle de inversores de frequência de uma linha de produção para a redução de tempo e facilidade na alteração nos parâmetros dos dispositivos de controle para a mudança do processo.

> *Flexible Manufacturing presents flexibility as a means for maintaining a desirable level of productivity. A production process having a higher degree of flexibility offers the advantages of a lower level of inventory, less balanced loading among stations or departments, and less need for short-term scheduling*. (LENZ, 1988).

A centralização de informações sobre tais dispositivos proporcionará uma economia de tempo na mudança de processos de produção e possibilitará a visualização em tempo real do funcionamento de cada um.

Deste modo, o objetivo do trabalho é propor a programação de um sistema com uma comunicação via rede que realize o controle das funções básicas dos inversores de frequência, aquisição de suas principais informações, e estes dados serem visualizados de forma gráfica e uma forma de realizar mudanças e guardar os parâmetros rapidamente de um inversor, permitindo uma melhora na produção.

1.2. DELIMITAÇÃO DO TEMA

Necessidade de uma visualização ampla, com o controle e aquisição de dados de inversores de frequência utilizando um sistema de controle centralizado, para a melhoria e a rápida modificação dos parâmetros da produção.

> *A SCADA (Supervisory Control and Data Acquisition) system allows an operator in a location central to a widely distributed process, such as an oil or gas field, pipeline system, irrigation system or hydroelectric generating complex, to make set point changes on distant process controllers, to open or close valves or switches, to monitor alarms, and to gather measurement information.* (BOYER, 2004)

No escopo da pesquisa proposta serão utilizadas informações sobre inversores de frequência do modelo da WEG CFW-09, o *software* LabVIEW para a programação do sistema supervisório e o protocolo de comunicação ModBus para permitir a comunicação entre o sistema e o inversor de frequência, que poderá ser aplicado a uma planta industrial.

1.3. PROBLEMA E PREMISSAS

Os principais problemas encontrados, e que objetivaram o trabalho, foram:

1. Controle descentralizado de dispositivos de uma linha de produção;
2. Demora e dificuldade em fazer mudanças simples de parâmetros nos dispositivos;

Em decorrência dos problemas citados, acabam gerando resultados negativos à produção como: perda de tempo entre a alteração dos parâmetros de um produto com especificações diferentes, redução de produtividade e um plano de manutenção ineficiente.

Com a instalação de um sistema supervisório integrando inversores de frequência utilizados em uma linha de produção conectados por uma rede, acredita-se que estes problemas serão minimizados ou anulados.

1.4. OBJETIVOS

1.4.1. Objetivo geral

Desenvolver um sistema supervisório utilizando o *software* LabVIEW para a programação que faça o controle e aquisição de dados em inversores de frequência do modelo da WEG CFW-09. No sistema supervisório utilizando o protocolo de comunicação ModBus para permitir a comunicação entre o sistema e o inversor de frequência e com a construção de uma rede acessando mais dispositivos para um controle centralizado.

1.4.2. Objetivos específicos

- ✓ Adquirir os valores de monitoramento do inversor e mostrar em um gráfico;
- ✓ Salvar arquivos com os parâmetros do inversor para futura reutilização;
- ✓ Realizar busca automática de dispositivos na rede;
- ✓ Deixar a interface gráfica amigável ao usuário;

1.5. JUSTIFICATIVA

O uso de inversores nas indústrias tem papel importante na automação de processos o que torna-o interessante estudar, pesquisar e investir em melhorias relacionadas com este dispositivo.

Cada fabricante desenvolve suas interfaces de parametrização e monitoramento dos dispositivos separados em *softwares* distintos. A unificação dessas interfaces com o objetivo de melhorar todo o processo de programação dos dispositivos facilitaria o seu uso.

Isso é uma tendência, grandes fabricantes como a Siemens, por exemplo, atualmente estão investindo em integrações, tendo desenvolvido um *software* integrado capaz de programar IHMs (Interface Homem-

Máquina) e CLPs (Controlador Lógico Programável) de um determinado padrão em uma única interface que antes eram separadas.

A integração simplifica o processo, melhora o aproveitamento do tempo do usuário evitando que ele gaste seu tempo desvendando informações peculiares ao equipamento e não necessárias para desenvolver o projeto para o qual o equipamento foi adquirido.

Uma interface amigável e unificada de parametrização contribui para uma maior fluência no desenvolvimento, destinando mais tempo para o processo de programação que já é naturalmente longo devido a sua minuciosidade.

O trabalho proposto, além de unificar a programação de inversores, trará a capacidade de programação em rede, reduzindo o tempo de parametrização desses equipamentos e erros devidos à interferência humana.

Dentre outros benefícios pode-se citar:

- Armazenamento da parametrização realizada em arquivo para obtenção de *backup* prático e de fácil entendimento;

- Sistema para restauração do *backup* sem necessidade de intervenção física direta no equipamento.

De modo geral a unificação resultará na eficiência do uso do tempo em manutenção e programação dos equipamentos englobados no *software*.

1.6. PROCEDIMENTOS METODOLÓGICOS

Classifica-se o estudo proposto como de natureza científica aplicada, devido ao fato de existir um problema claro (perda de tempo em parametrizações dos equipamentos) e uma proposta de solução (CERVO; 2007).

Em relação ao objetivo macro a pesquisa enquadra-se como sendo descritiva e bibliográfica. Descritiva porque, busca conhecer e analisar as contribuições científicas do passado existentes sobre um determinado assunto, tema ou problema. (CERVO; 2007). A pesquisa bibliográfica

decorre da necessidade do aprofundamento na teoria do tema procurando possíveis soluções.

Serão realizadas pesquisas em campo com o processo de coleta dos dados, “Em pesquisas descritivas faz-se a descrição detalhada de todos os passos da coleta e registro de dados.” (CERVO; 2007) e será estabelecida uma comparação entre a parametrização manual e a automática de modo tentar deixar clara a vantagem de um sistema de parametrização mais automática.

1.7. REFERENCIAL TEÓRICO

“O LabVIEW, *Laboratory Virtual Instruments Engineering Workbench*, é uma linguagem utilizada para desenvolver aplicativos. O LabVIEW é uma linguagem de programação gráfica (G), e o que a diferencia das outras linguagens de programação convencionais, como Delphi e Basic, é a forma de programação. Embora os dois tipos sejam direcionados a objetos, há uma diferença fundamental entre eles. A linguagem G é uma ferramenta de programação gráfica, altamente produtiva para a construção de sistemas de aquisição de dados, instrumentação e controle, entre outras aplicações.” (REGAZZI; 2005).

Considerando a necessidade de ter desenvolvimento próprio de *software* para atingir o objetivo desse projeto, o LabVIEW é uma ferramenta com potencial devido a sua grande flexibilidade na programação da comunicação com equipamentos externos ao computador e ao ótimo ambiente de programação propiciado.

A Figura 1 mostra um exemplo do ambiente de programação do LabVIEW.

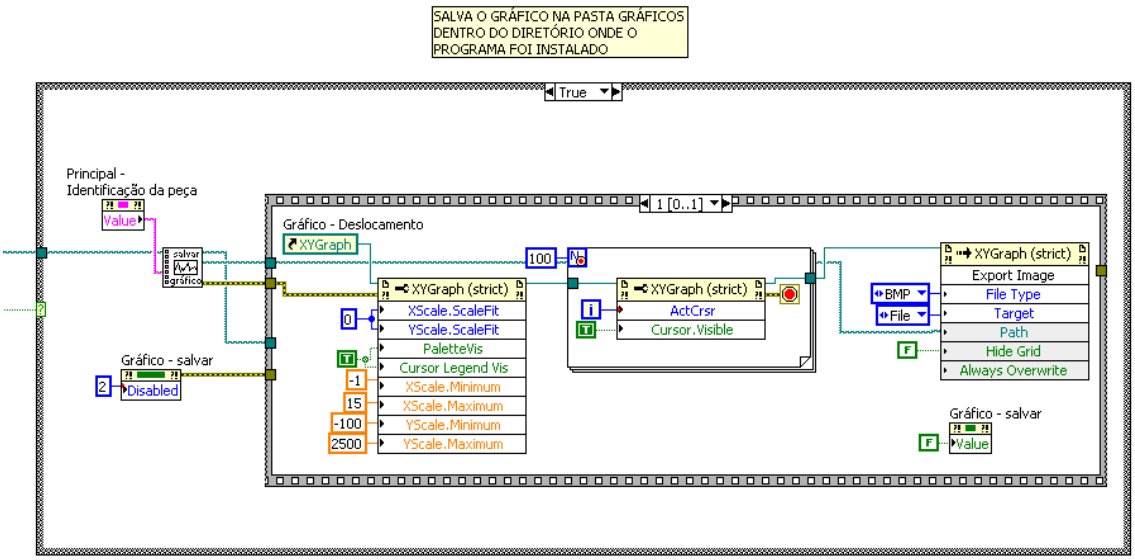


Figura 1 – Ambiente de programação.

Fonte: Autoria própria

O LabVIEW incorpora em seus recursos a possibilidade de desenvolver a interface da forma desejada e códigos altamente flexíveis que permitirão gravar as parametrizações em arquivos que posteriormente poderão ser lidos novamente pelo programa. Possibilita criação de banco de dados interno para facilitar a busca do parâmetro desejado a se parametrizar não necessitando consultar o manual do inversor, desta forma o parâmetro deixa de ser um número e passa a ser uma descrição da função. A Figura 2 mostra um exemplo da interface com o usuário.

Figura 2 – Exemplo de interface com o usuário do LabVIEW

Fonte: Autoria própria

O inversor de frequência é um equipamento elétrico capaz de produzir uma variação dos valores de frequência elétrica que alimenta o motor, produzindo uma variação da sua rotação ou velocidade (TSUTIYA, 2004).

O inversor de frequência da marca WEG, modelo CFW-09 é o escolhido para ser estudado e usado como um modelo dentro do programa a ser desenvolvido, este inversor é capaz de ingressar em uma rede industrial e quando instalado o módulo de interface apropriado consegue realizar esta comunicação.

A comunicação entre o *software* parametrizador e a rede de equipamentos será feita no padrão RS-485, partindo do PC no meio físico do tipo RS-232, sendo então convertido para o tipo RS-485 através de um conversor para este fim. O protocolo de comunicação será o MODBUS-RTU que é vastamente utilizado na indústria para comunicação de equipamentos que não exijam extrema velocidade e volume no tráfego de dados entre dispositivos. “Em aplicações de automação e controle industriais típicas, baseadas em CLPs, o módulo de Comunicação de Campo utiliza comunicação ponto-a-ponto, como por exemplo, o protocolo ModBus” (SEGURA, 2005, pg. 117).

1.8. ESTRUTURA DO TRABALHO

- Capítulo 1 - Introdução com apresentação do tema, problemas e premissas, objetivos principais, justificativa, procedimentos metodológicos, referencial teórico e estrutura do trabalho.
- Capítulo 2 - Inversores de Frequência: conceito, características, aplicação.
- Capítulo 3 - LabVIEW: conceito, linguagem G, aplicação no projeto.
- Capítulo 4 - Desenvolvimento no LabVIEW: introdução, hipóteses e postulados para desenvolvimento do programa, estrutura do programa, comunicação entre o dispositivo e sistema supervisório e comparação entre parametrização manual e sistema supervisório.

- Capítulo 5 - Descritivo de telas do programa desenvolvido: tela principal, tela de lista de receitas, tela de parametrização do inversor de frequência, tela de acompanhamento do dispositivo, pop-up de *login* de usuário e pop-up de usuário logado.
- Capítulo 6 - Conclusão.
- Referências.
- Apêndice(s).
- Anexo(s).

2 INVERSORES DE FREQUÊNCIA

2.1 CONCEITO

Os inversores de frequência ou também chamados de conversores de frequência, são equipamentos projetados especificamente para trabalhar dentro de painéis elétricos e com a finalidade de controlar a velocidade ou torque de motores CA (corrente alternada) com eficiência a partir da variação da tensão e frequência. “Dispositivos modernos de controle de velocidade podem ser empregados para manter a velocidade de uma máquina com uma variação de no máximo 0.1%, independente da carga aplicada.” (FRANCHI, Claiton Moro. Pg17)

“Um conversor de frequência CA é projetado para controlar a frequência e a tensão aplicadas ao motor. (FRANCHI, Claiton Moro. Pg102)

> “De uma maneira geral, os dispositivos para controle de velocidade são usados para as seguintes operações:
> - Ajuste da velocidade de um motor elétrico visando à rapidez do processo;
> - Ajuste de torque de um conjunto de acordo com as necessidades do processo;
> - Redução do consumo de energia e aumento da eficiência.” (FRANCHI, Claiton Moro. Pg17)

Os motores de corrente alternada são os mais utilizados no meio industrial devido a terem diversas vantagens sobre os de corrente contínua que tem a aplicação mais específica. Dentre as vantagens incluem-se principalmente custo e facilidade de manutenção, devido a isto está sendo abordado apenas o motor de indução na descrição do conceito do inversor. “Uma grande infinidade de equipamentos foi desenvolvida para as mais diversas variedades de aplicações e setores industriais. Um dos equipamentos mais utilizados nesses processos, junto com os controladores lógicos programáveis (CLP), é o inversor de frequência.” (FRANCHI, Claiton Moro. Pg19)

> “O princípio de funcionamento de um motor de indução é baseado na Lei de Faraday da indução eletromagnética, que afirma que, se um condutor é movimentado através de um campo magnético (B), uma tensão é induzida. Se o condutor é um circuito fechado, uma corrente (I) vai circular. Quando o condutor é movimentado uma força (F), que é

> perpendicular ao campo magnético, vai agir sobre o condutor. Este é o princípio do gerador elétrico.” (FRANCHI, Claiton Moro. Pg27)

A Figura 3 mostra o bloco diagrama do inversor modelo CFW-09, utilizado no projeto.

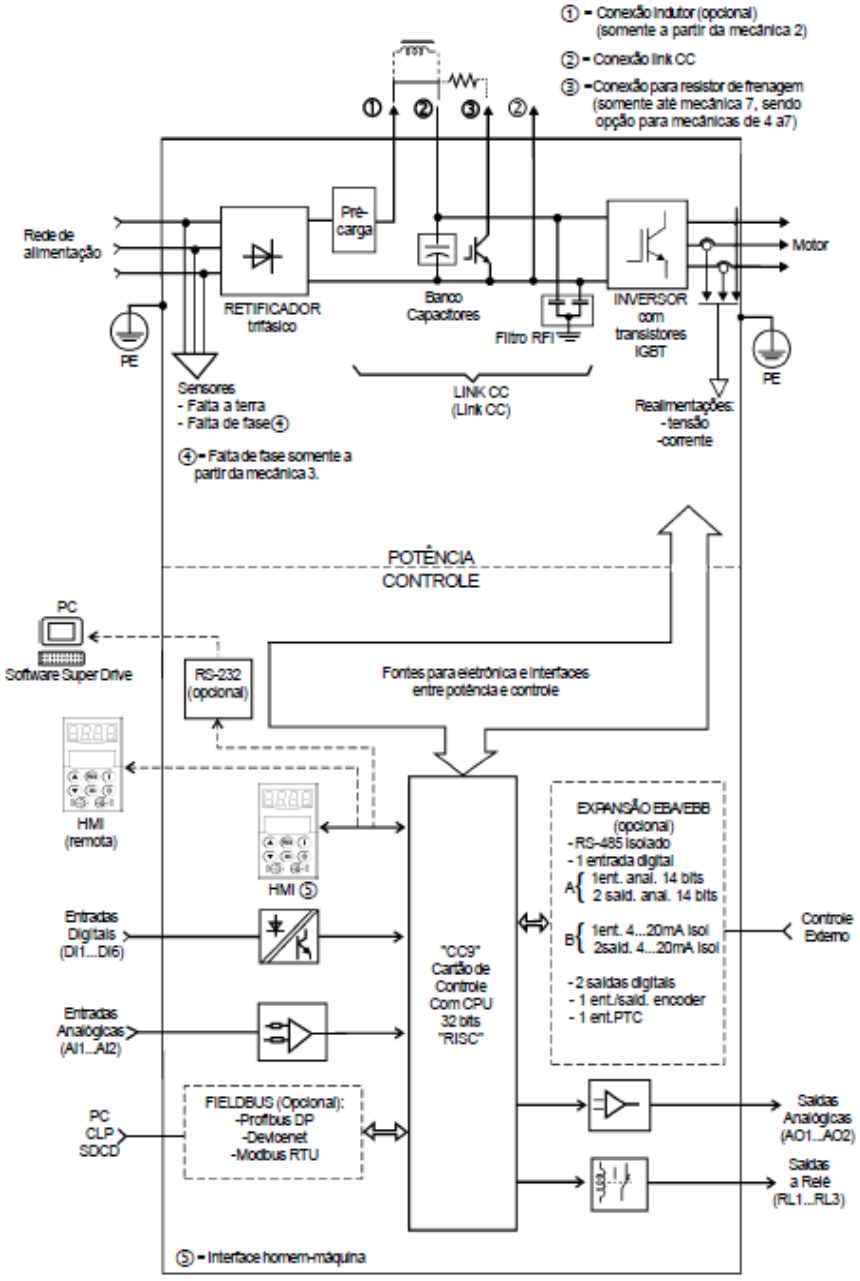


Figura 3 – Bloco diagrama de conjunto inversor CFW-09

Fonte: Manual completo WEG CFW-09

2.2. CARACTERÍSTICAS

As principais características do inversor é a de que ele é parametrizável tornando possível ser usado em diversos tipos de processos. Suas configurações podem ser inseridas via interface de comunicação (que é o foco do projeto) ou via IHM fornecida pelo fabricante. “As funções de um inversor de frequência são executadas de acordo com parâmetros predefinidos alocados na CPU. Os parâmetros são agrupados de acordo com as suas características e particularidades” (FRANCHI, Claiton Moro. Pg93)

As principais categorias de parâmetros dos inversores são:

- “Parâmetros de Leitura: variáveis que podem ser visualizadas no display, mas não podem ser alteradas pelo usuário.
- Parâmetros de regulação: são os valores ajustáveis a serem utilizados pelas funções do inversor de frequência, como, por exemplo, tensão inicial, tempo de rampa de aceleração, tempo de rampa de desaceleração etc.
- Parâmetros de configuração: como as entradas e saídas como, definem as características do inversor de frequência, as funções a serem executadas, bem como as entradas e saídas, por exemplo, parâmetros dos reles de saída e das entradas do inversor de frequência.
- Parâmetros do motor: indicam as características nominais do motor, como, por exemplo, ajuste de corrente do motor, fator de serviço.” (FRANCHI, Claiton Moro. Pg93)

2.3. APLICAÇÃO

Os inversores através de suas funções permitem que processos sejam controlados utilizando diferentes recursos com a finalidade de suprir uma necessidade de maneira mais eficiente, recursos como controle de malha aberta e controle de malha fechada são alguns meios de controle de processo pertinentes aos inversores em geral.

> “O controle digital aplicado no conversor de frequência automatiza este processo. Por exemplo, quando o operador seleciona a velocidade do motor através de um potenciômetro, o sistema de controle implementa essa seleção por meio do ajuste da frequência de saída e da tensão para assegurar que o motor gire em uma determinada velocidade.” (FRANCHI, Claiton Moro. Pg102)

O Controle em malha aberta permite que o inversor controle a velocidade ou torque de um motor sem precisar de um transdutor conectado

a alguma de suas entradas analógicas, sendo esta uma forma de controle mais simples e barata.

"...em muitas aplicações o inversor de frequência é simplesmente usado para controlar a velocidade de uma carga baseado em um valor de ajuste (*setpoint*) fornecido por um operador ou controlador de um processo..." (FRANCHI, Claiton Moro. Pg101)

> "O único controle de corrente que existe é o limite de corrente cujo valor atinge um nível máximo, por exemplo, de 160% de uma corrente total de carga. Não existe nenhuma informação sobre a velocidade atual para verificar se o motor está rodando na velocidade desejada." "... Este método é chamado de controle em malha aberta e é adequado para controlar cargas que variem pouco em aplicações mais simples, como, por exemplo, bombas centrífugas, ventiladores, esteiras transportadoras em que as mudanças de velocidade que possam ocorrer não tem consequências para o controle do processo." (FRANCHI, Claiton Moro. Pg102)

O controle de malha fechada usa um transdutor para retroalimentar o inversor e informar sobre o estado da variável controlada ao controlador.

> "Na indústria existem aplicações críticas cuja velocidade/torque deve ser precisamente e continuamente controlada. Essa precisão requerida de controle é muito importante e deve ser levada em consideração no momento da escolha de um conversor de frequência." "...No controle em malha fechada a velocidade/torque é lida continuamente, controlada e mantida no valor desejado automaticamente." (FRANCHI, Claiton Moro. Pg103)

"Um exemplo desse controle é um inversor que aciona uma motobomba que faz circular um fluido numa dada tubulação. O próprio inversor pode fazer o controle da vazão nessa tubulação." "...Outros exemplos de aplicação: controle de nível, temperatura, dosagem etc...") (FRANCHI, Claiton Moro. Pg105).

2.4. CARTÃO EBA

Utilizado para realizar a comunicação RS485 com o inversor, realizando a troca de dados entre o inversor de frequência e outro dispositivo que possua a mesma interface de comunicação. O inversor de frequência

WEG CFW-09 dispõe de quatro modelos de cartões cada uma com uma configuração especifica aumentando os recursos do inversor. Os 4 cartões não podem ser utilizados simultaneamente. A diferença entre os cartões opcionais EBA (Figura 4) e EBB está nas entradas/saídas analógicas. O cartão EBC1 é para conexão de encoder. O cartão EBE é para RS485 e PT. Na Tabela 1 é informado quais as funções que o cartão EBA adiciona ao inversor de frequência.

Figura 4 – Cartão de expansão de funções EBA

Fonte: Manual CFW-09 WEG

COMUNICAÇÃO	INTERFACE SERIAL	☑ Serial RS-485 isolada (a utilização da serial RS-485 impede a utilização da serial RS-232 - não podem ser utilizadas simultaneamente)
ENTRADAS	ANALÓGICAS	☑ 01 Entrada analógica (AI4), linearidade 14 bits (0.006% do range [±10V]), bipolar, -10V a +10V, (0 a 20) mA, (4 a 20) mA, programável
	ENCODER INCREMENTAL	☑ Alimentação/realimentação para encoder incremental, fonte interna isolada 12V/ 200mA máx, entrada diferencial, uso como realimentação de velocidade para regulador de velocidade, medição digital de velocidade, resolução 14 bits, sinais (100kHz máx.) A, $\overline{A}$, B, $\overline{B}$, Z e $\overline{Z}$
	DIGITAIS	☑ 01 Entrada digital (DI7): isolada, programável, 24Vcc ☑ 01 Entrada digital (DI8) para termistor-PTC do motor, programável, atuação 3.9kΩ, release 1.6kΩ
SAÍDAS	ANALÓGICAS	☑ 02 Saídas analógicas (AO3/AO4): linearidade 14 bits (0.006% do range [±10V]), bipolares, -10V a +10V, programáveis
	ENCODER	☑ Saída de encoder bufferizada: repetidora dos sinais de entrada, isolada, saída diferencial, alimentação externa 5V a 15V
	DIGITAIS	☑ 02 Saídas a transistor isoladas (DO1/DO2): open collector, 24Vcc, 50mA, programáveis

Tabela 1 – Descrição das funções do cartão EBA

Fonte: Manual CFW-09 WEG

3. LABVIEW

3.1. CONCEITO

O LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) é uma linguagem de programação de padrão gráfica desenvolvida pela *National Instruments.* A primeira versão da linguagem surgiu em 1986 para o Macintosh. Atualmente existem também versões para plataformas como o Windows, Linux e Solaris.

Os programas do LabVIEW são chamados de Instrumentos Virtuais, ou VIs (*Virtual Instrument)* mais comumente utilizado, porque sua aparência e operação imitam instrumentos físicos, tais como osciloscópios e multímetros. Cada VI tem funções que manipulam a entrada pela interface do usuário ou de outras fontes e indicam essa informação ou a movem para outros arquivos ou outros computadores.

3.2. LINGUAGEM G

Os programas em G, fazem uso de uma interface interativa com usuário e um diagrama de fluxo de dados onde se está o código fonte. A programação gráfica LabVIEW se estrutura em: painel frontal, diagrama de bloco e ícone de conexão. (REGAZZI; PEREIRA; SILVA JR, 2005).

O painel frontal permite o fornecimento de valores de entrada pelo usuário, e que o mesmo observe os valores de saída processados no diagrama em blocos. O painel frontal é análogo a um instrumento de medição. As entradas e saídas são chamadas de controle e indicadores, respectivamente. Pode ser utilizada uma variedade de controladores e indicadores como botões, gráficos, indicadores analógicos, etc. Tanto os indicadores como os controladores são referenciados por um texto (*label*), que ajuda na localização durante o processo de programação. (REGAZZI; PEREIRA; SILVA JR, 2005).

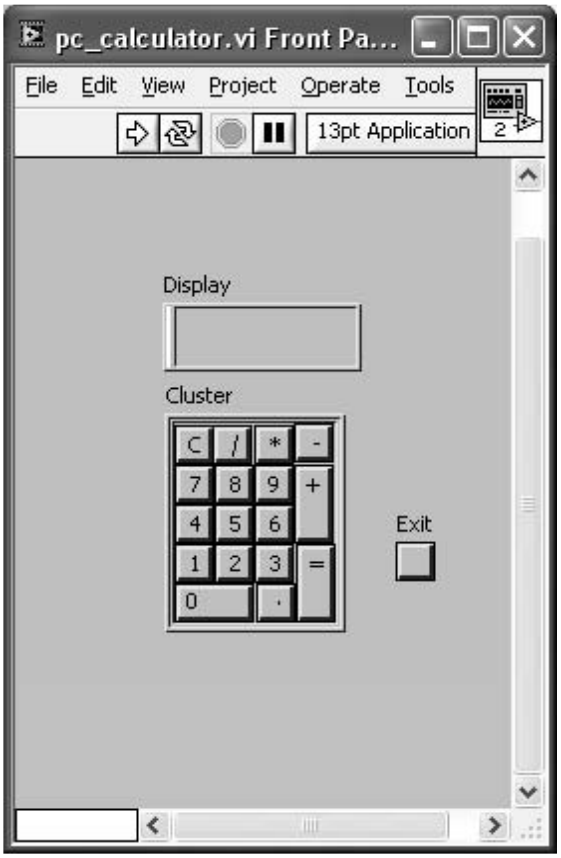


Figura 5 – Exemplo de um painel frontal

Fonte: *LabVIEW Advanced Programming Techniques - BITTER Rick - CRC Press 2006*

O diagrama de blocos é o código fonte do programa. Cada painel frontal é acompanhado por um diagrama de blocos. Os componentes do diagrama de blocos representam os nós por onde as informações passam, obedecendo sempre a lógica do programador. (REGAZZI; PEREIRA; SILVA JR, 2005).

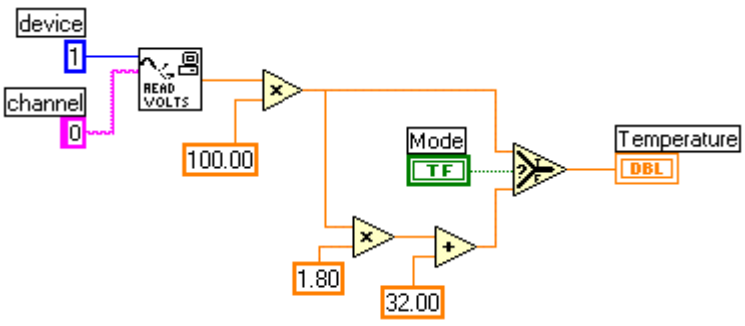


Figura 6 – Exemplo de diagrama de blocos

Fonte: Curso introdutório de programação com LabVIEW- Ricardo A. Langer – UP 2006

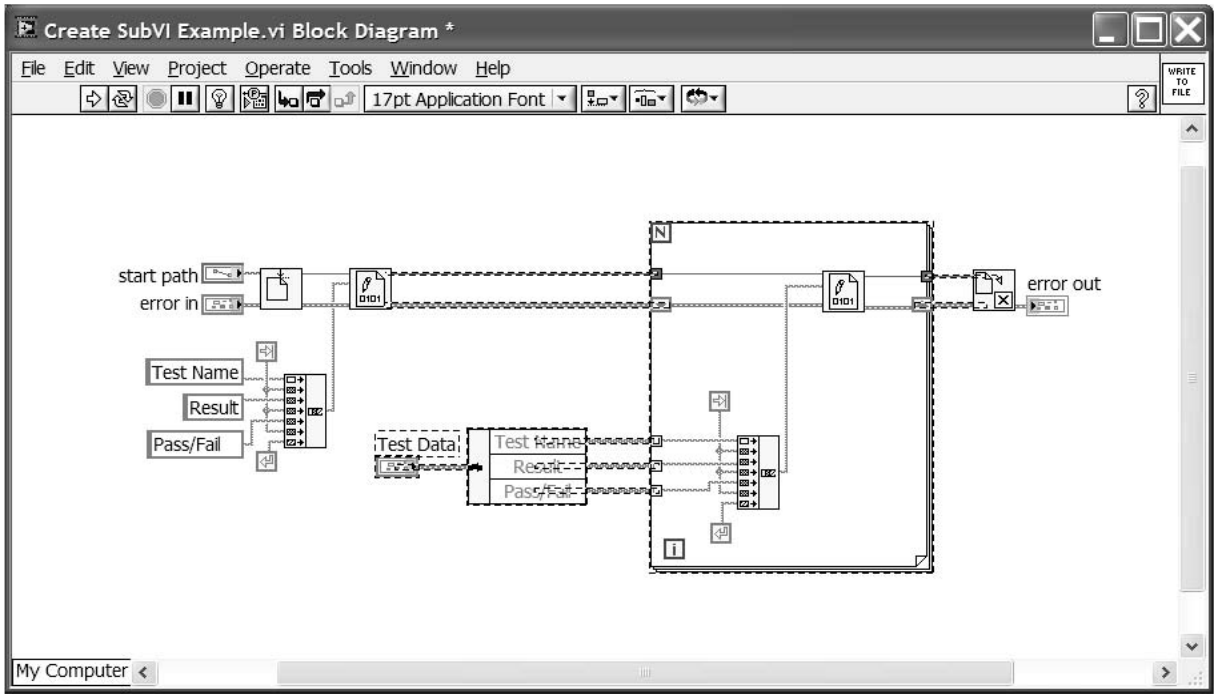


Figura 7 – Exemplo de diagrama de blocos

Fonte: *LabVIEW Advanced Programming Techniques - BITTER Rick - CRC Press 2006*

O ícone de conexão é utilizado para transformar um programa em uma sub-rotina dentro do diagrama de blocos de um segundo programa. O conector de terminais determina onde são colocadas as entradas (passagem de parâmetros e valores para a sub-rotina) e a saída que representa a resposta em função lógica e dos algoritmos da sub-rotina. Sendo que os controles representam as entradas e os indicadores representam a saída. (REGAZZI; PEREIRA; SILVA JR, 2005).

3.3. APLICAÇÃO NO PROJETO

A conteúdo principal e o diferencial deste projeto está na programação feita usando o LabVIEW. Toda a inteligência de funcionamento do projeto está contida no código fonte do programa, desenvolvido usando-se técnicas de programação já consagradas em tutoriais. Além disso, lógicas foram desenvolvidas exclusivamente para realizar as operações necessárias provenientes da necessidade do projeto.

Recursos como *loops while, for,* estruturas de eventos, rotinas, interface de comunicação serial, estruturas de sequenciamento, estruturas *case*, variáveis locais e globais, sub-rotinas indexadas desenvolvidas especificamente para este projeto, rotinas de leitura e gravação a arquivos de texto e diversos outros recursos foram usados para conseguir atender à necessidade proposta.

A linguagem G proporcionou uma grande versatilidade e simplificação da programação, em virtude também a diversos recursos inerentes a ferramenta de programação o desenvolvimento, em algumas etapas, foi enxuto e rápido.

A escolha da ferramenta foi unânime graças ao seu grande leque de recursos de programação e interface com o usuário de fácil criação tornou o LabVIEW a melhor opção para atingir o objetivo de conseguir manipular muitos dados e formata-los de maneira eficaz garantindo maior velocidade na execução das operações de comunicação e de navegação no *software*.

4. DESENVOLVIMENTO NO LABVIEW

4.1. INTRODUÇÃO

O desenvolvimento do programa foi feito em *LabVIEW* versão 2010, para tal utilizamos o ambiente de programação em diagrama de blocos que é o padrão e a área de interface de usuário, também criamos sub-rotinas para compactar o código e poder utiliza-las em outras partes do programa sem ter que criar a lógica novamente para agilizar o processo de desenvolvimento. Utilizamos bastante recursos de desenvolvimento do *LabVIEW* para atingir nosso objetivo. A Interface gráfica do *LabVIEW* é bastante amigável, o que permitiu criar um ambiente agradável e diferenciado para o usuário operar, trazendo cores e objetos chamativos e bem destacados facilitando a observação de todas as funções do programa na tela.

4.2. HIPÓTESES E POSTULADOS PARA DESENVOLVIMENTO DO PROGRAMA

O uso do *LabVIEW* para desenvolver o projeto foi um diferencial, pois não depende de licença para o usuário final podendo ser utilizado por qualquer pessoa, e também como o *LabVIEW* está sendo cada vez mais utilizado, este projeto trouxe exemplos de desenvolvimento que podem auxiliar outras pessoas que almejem aprender a linguagem de programação G. Sobretudo criamos um *software* que facilita a parametrização de inversores e concentra suas informações em um aplicativo, aumentando a confiabilidade da parametrização e garantindo repetitividade quando se usa os arquivos de *backup*. A utilização do *software* elimina a necessidade de ter uma IHM em cada inversor para parametrizá-lo e ler os parâmetros atuais do dispositivo. Este é certamente um motivo para criarmos o programa.

Alguns dados do inversor, como corrente, frequência no motor etc. são mais bem interpretados usando um gráfico no domínio do tempo. O programa

consegue fazer a leitura das diversas variáveis disponibilizadas para monitoramento e plotá-las em gráfico que pode ser ajustado conforme a necessidade do usuário, tornando o uso do programa adequado para fazer uma análise de processo mais aprofundada.

Para atingir este objetivo irá se utilizar de recursos de programação do *LabVIEW* capazes de comunicar com o inversor e ler seus dados em tempo real, com objetos gráficos disponibilizados, funções de leitura de dados, funções para criar o gráfico no domínio do tempo, funções de ajuste de escala para permitir melhor visualização das penas gráficas entre outros recursos, garantindo que o objetivo seja atingido.

4.2.1 Hipóteses

Para escolher o *LabVIEW* foi levado em consideração se os seus recursos supririam a nossa necessidade, para isto foi considerado os seguintes aspectos:

1) Há como comunicar com um equipamento externo usando *LabVIEW* e o protocolo ModBus?
2) Tratar os dados na programação para realizar a parametrização inteira de uma só vez?
3) É possível armazenar as receitas em arquivo?
4) Plotar dados em gráfico usando o *LabVIEW*?
5) Conhecimento ou suporte do *LabVIEW* para nos ajudar a chegar no objetivo proposto?
6) Inversor disponível para uso?
7) Como é gerado o sinal (ex. velocidade, frequência, etc.) no inversor e como o *software* os interpreta?
8) Como decodifica o *software* e reenvia ao inversor?
9) Transferência de dados ou sinais?
10) Como interpreta o inversor e como o *software* interpreta as mensagens?

4.2.2 Postulados

Em resposta as questões levantadas anteriormente, estão listados nos tópicos abaixo as diretivas de cada item questionado.

1) A utilização do *LabVIEW* comunicando com algum dispositivo via protocolo ModBus, seja via meio físico serial ou ethernet já teve sua implementação realizada em outros projetos desenvolvidos previamente, assegurando a possibilidade de atingir o objetivo, diversos exemplos também são encontrados nas literaturas sobre o LabVIEW e na internet.
2) A ferramenta também já nos mostrou o seu poder para trabalhar dados e informações no seu código, sua linguagem possibilita uma boa visualização do programa, com experiência prévia é garantido de trabalhar os dados no código conforme definido.
3) Para gravar dados em arquivo existem diversas ferramentas disponibilizadas, como o trabalho não exige que grande quantidade de dados seja salva em arquivos e nem grande complexidade organizacional dos dados, será utilizado um padrão binário para salvar os arquivos e estes serão separados, cada arquivo representando uma receita.
4) O LabVIEW tem seu principal foco na geração de gráficos e aquisição de dados do meio externo, sendo uma poderosa ferramenta para detalhamento de informações de modo visual e simplificado.
5) Com experiência prévia em LabVIEW é assegurado que o objetivo proposto será atingido, entretanto a universidade possui professores com conhecimento na ferramenta caso for necessário auxílio, além de haverem diversos livros e apostilas sobre o assunto.
6) Há a necessidade de ter um inversor de frequência CFW-09 disponível, será utilizado o inversor que já está eletricamente ligado no laboratório das dependências da UTFPR, conforme mostrado na – Inversor CFW-09 WEG do laboratório da UTFPR.
7) Os sinais são gerados internamente pelo inversor e transformados em números que representam grandezas, estes valores são transmitidos pela interface de comunicação a pedido do mestre de rede conforme explicado

no item 4.4 que refere-se a comunicação entre o inversor e o supervisório.

8) Os telegramas e mensagens no padrão utilizado estão descritos no trabalho na sessão de comunicação, item 4.4.4.
9) A transferência de dados é o que prevalece, pois a comunicação com o protocolo ModBus-RTU para acessar os valores das grandezas monitoradas e não criar qualquer outro tipo de interface entre o supervisório e o dispositivo para acessar sinais.
10) O programa interpreta o inversor como um dispositivo qualquer de rede no padrão do protocolo ModBus, sendo apenas um endereço onde estes são capazes de trocar dados em específicos endereços de comunicação, se fosse colocado outro inversor de outra marca que tivesse o funcionamento idêntico ao CFW-09 com os mesmos endereçamentos e funções o *software* também conseguiria parametrizá-lo.

Assim sendo a lista das principais hipóteses levantadas tiveram suas respostas positivas, viabilizando o desenvolvimento do projeto.

Figura 8 – Inversor CFW-09 WEG do laboratório da UTFPR

Fonte: Autoria própria

4.3 ESTRUTURA DO PROGRAMA

A estrutura é definida em dois módulos, um de inicialização e outro de execução, como pode ser observado na Figura 9 indicado pelo número um e número dois respectivamente.

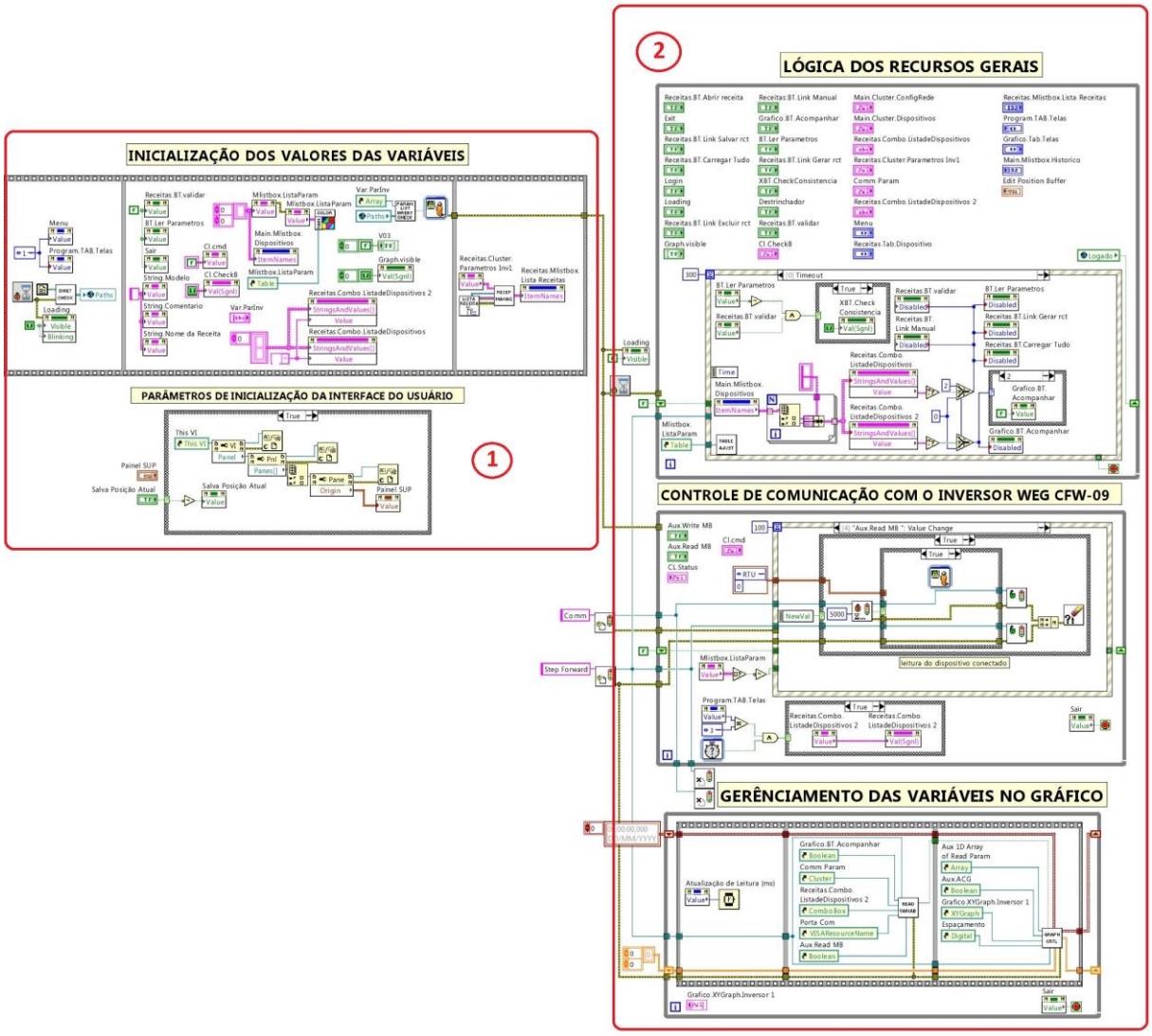


Figura 9 – Estrutura principal da programação

Fonte: Autoria própria

O Módulo de inicialização predispõe todas as variáveis para os estados iniciais, ajusta as propriedades dos controles e também carrega os arquivos de dados e tabelas para a memória tornando estes dados disponíveis para as rotinas de execução acessarem os dados mais rapidamente. A inicialização também ajusta algumas características da interface do usuário como, por exemplo, a troca de telas, existe uma função que ao inicializar automaticamente chama a tela principal na interface e a posiciona no centro do monitor. O detalhamento da lógica de inicialização de dados pode ser visto na Figura 10.

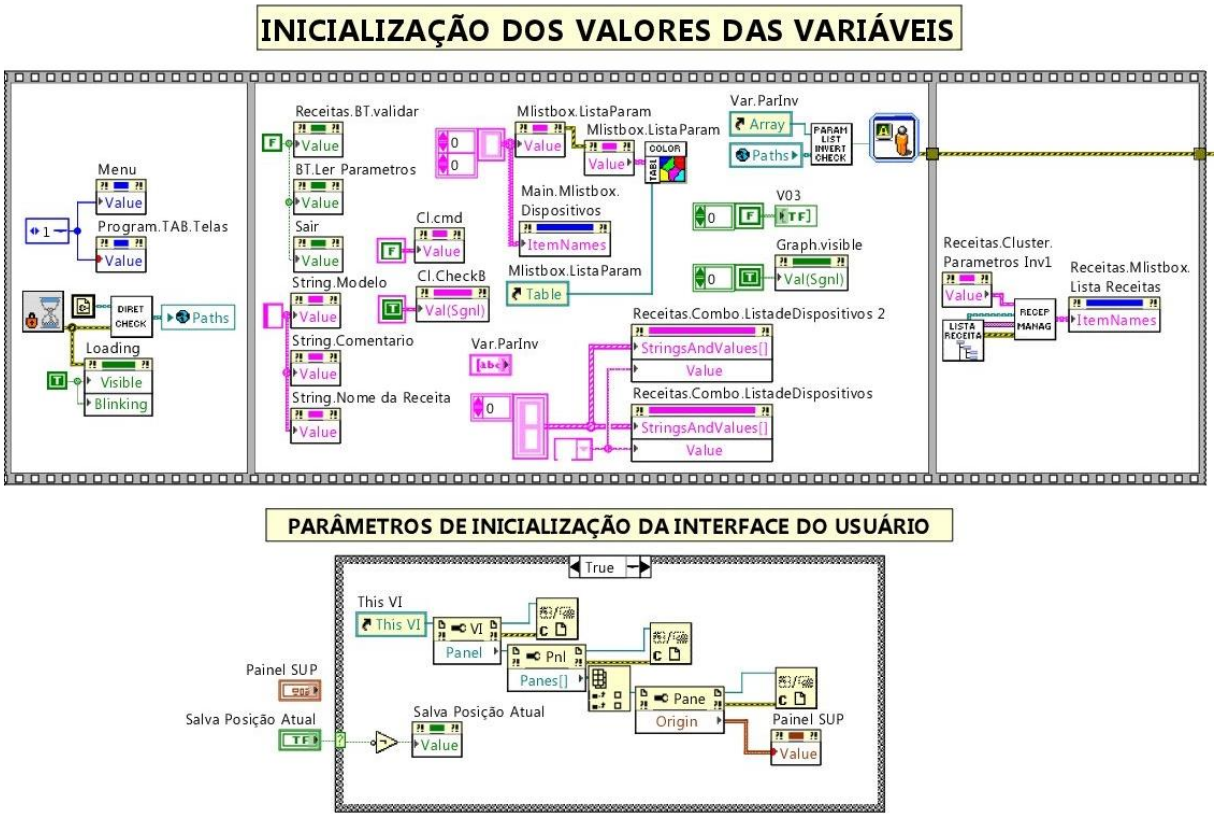


Figura 10 – Lógica de inicialização de valores

Fonte: Autoria própria

O Módulo de execução é dividido em parte de operação, comunicação e visualização de dados no gráfico, estas estruturas estão ilustradas na Figura 11 identificados com a numeração um, dois e três respectivamente.

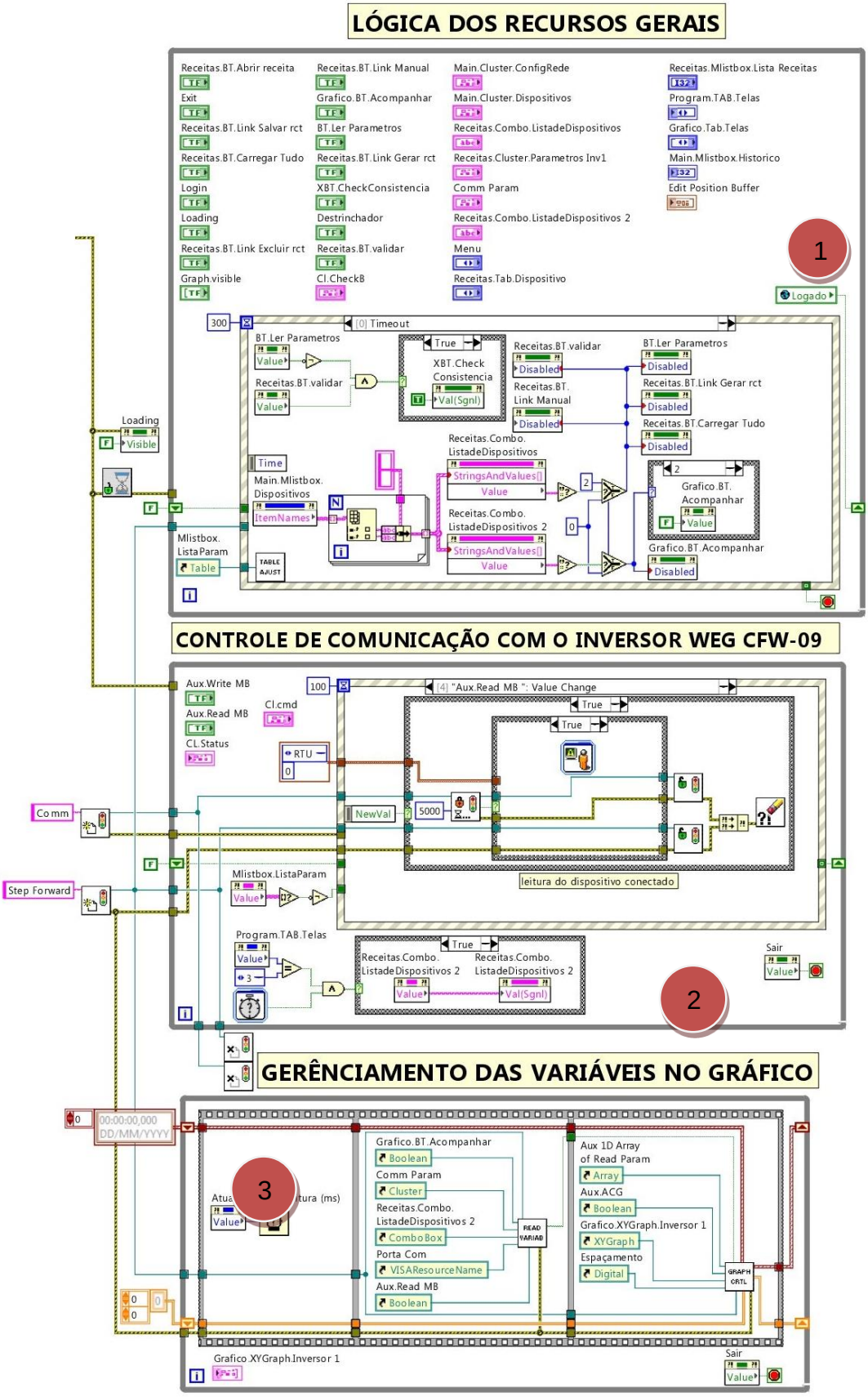


Figura 11 – Módulo de execução

Fonte: Autoria própria

A parte de operação representa todas as funções que são chamadas ao executar uma ação pelo usuário na interface ou a pedido de alguma outra função interna do programa. São estas funções que dão vida ao programa, por exemplo estas funções trocam de telas, criam as tabelas de dados, verificam os valores que o usuário insere no programa e analisa se são válidos ou não, gera o gráfico, ajustam o gráfico na escala conforme a configuração, enviam comandos e recebem dados das funções de comunicação, bloqueiam e desbloqueiam a interface conforme programado, etc. Estas funções estão predominantemente dentro de estruturas de evento, que são estruturas chamadas ao executar uma operação no programa pelo usuário, como um clique em um botão por exemplo. Os eventos que o programa executa esta indicado na Figura 12 destacado em vermelho, a seguir será feita uma breve descrição dos comandos executados em cada evento.

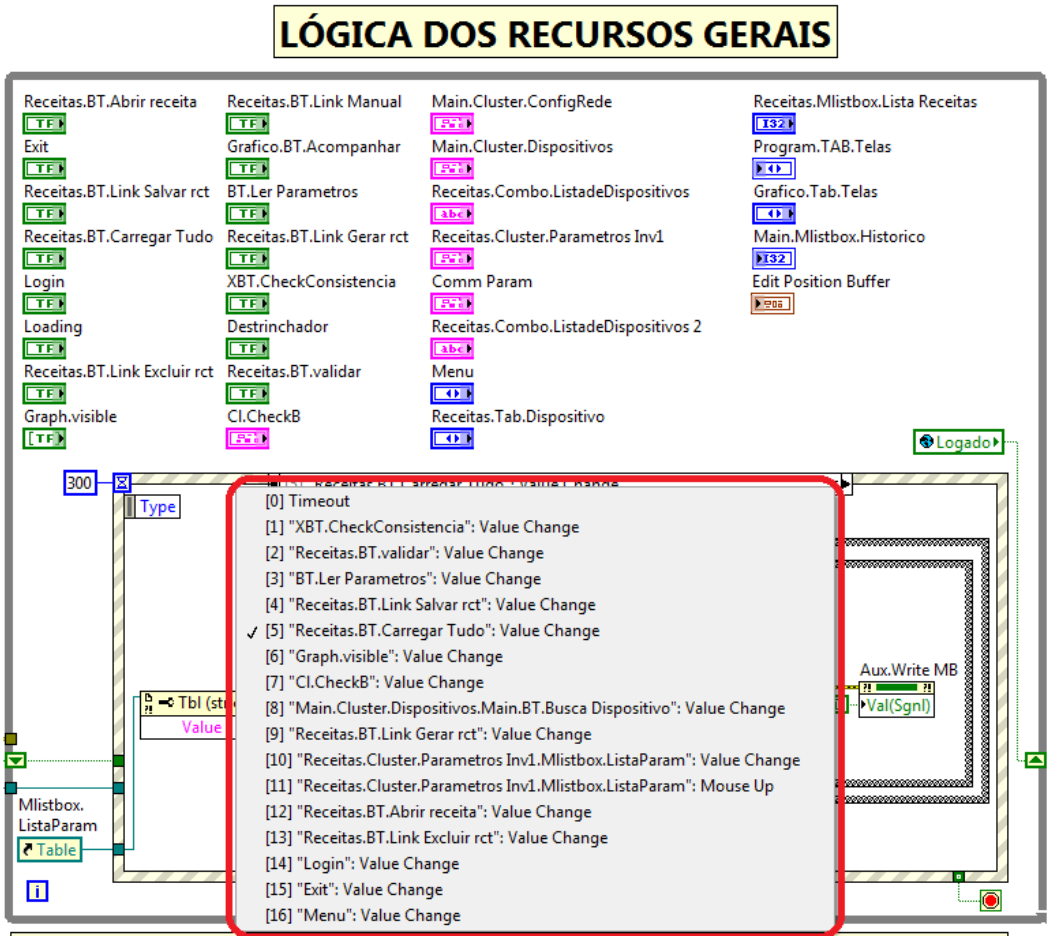


Figura 12 – Eventos executados pelo programa

Fonte: Autoria própria

No evento numerado com o valor zero (*Timeout)* os comandos contidos nele são executados a todo momento enquanto não for gerado nenhum outro evento, nesta estrutura estão comandos para gerar a validação de uma receita, alterar o estados dos controles na tela de receita e de gráfico. Na Figura 13 demonstra a programação no evento de *Timeout*.

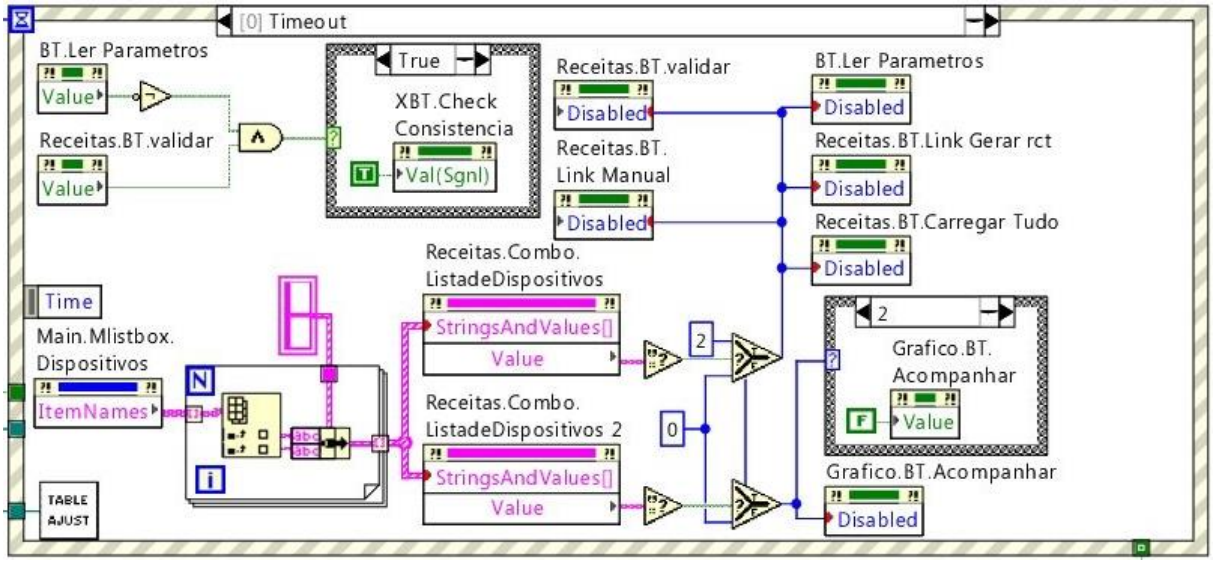


Figura 13 – Programação evento *Timeout*

Fonte: Autoria própria

No próximo evento ele é gerado após ser pressionado o botão de validação da receita onde compara os parâmetros da tabela com os parâmetros que estão no inversor de frequência, na Figura 14 é demonstrada a programação. Este evento está relacionado aos eventos de número dois e três, que são executados antes deste evento para executar a tarefa de validação da receita.

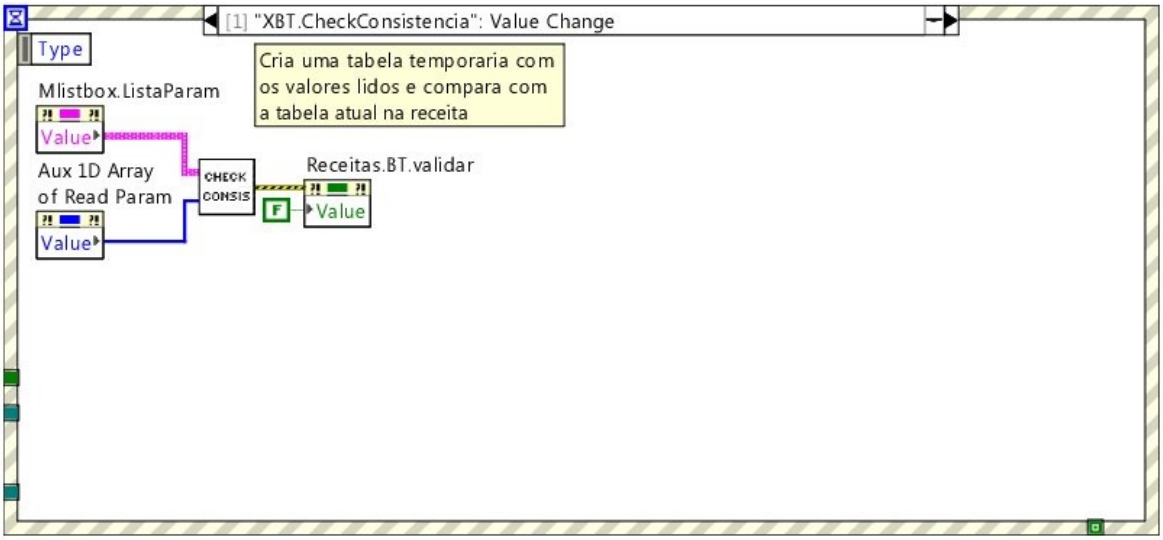


Figura 14 – Evento de validação da receita

Fonte: Autoria própria

No evento de número dois ele faz parte do comando de validar a receita como foi descrito acima, ele é a primeira lógica a ser executada quando é dado o comando de validar a receita. A lógica executada nesta estrutura é de verificar se existem parâmetros na tabela para serem comparados com as do inversor. Se existem parâmetros na tabela é acionado o evento de número três. Na Figura 15 a lógica executada neste evento.

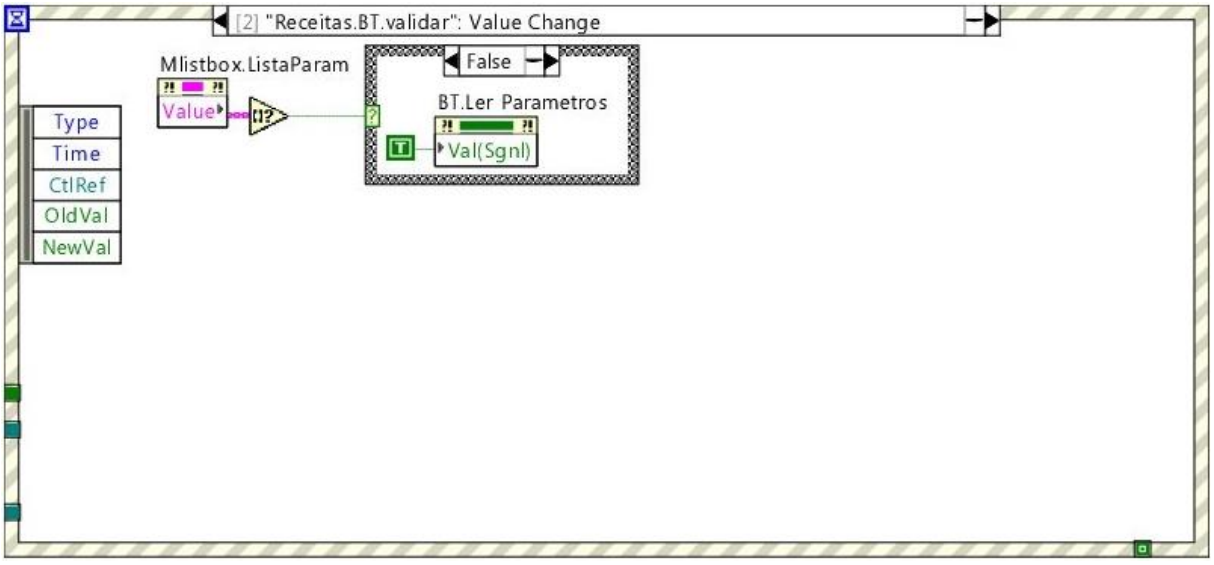


Figura 15 – Evento de número dois

Fonte: Autoria própria

O evento de número três é executado após o evento dois, nele é dado o comando para ser feita a leitura dos parâmetros que estão no inversor. Na Figura 16 é apresentada a programação descrita anteriormente.

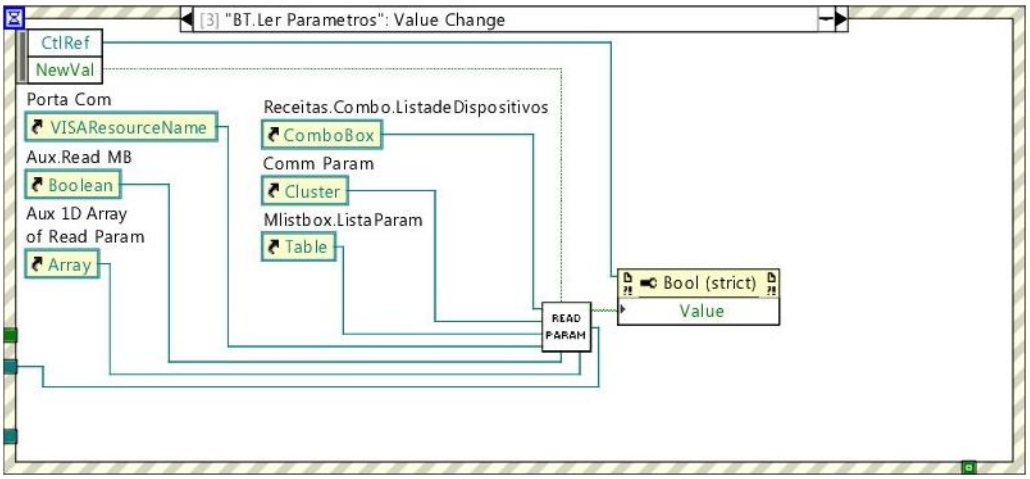


Figura 16 – Evento de número três

Fonte: Autoria própria

No evento relacionado ao controle “Receitas.BT.Link Salvar rct” com o valor quatro é executado o comando para salvar uma receita.

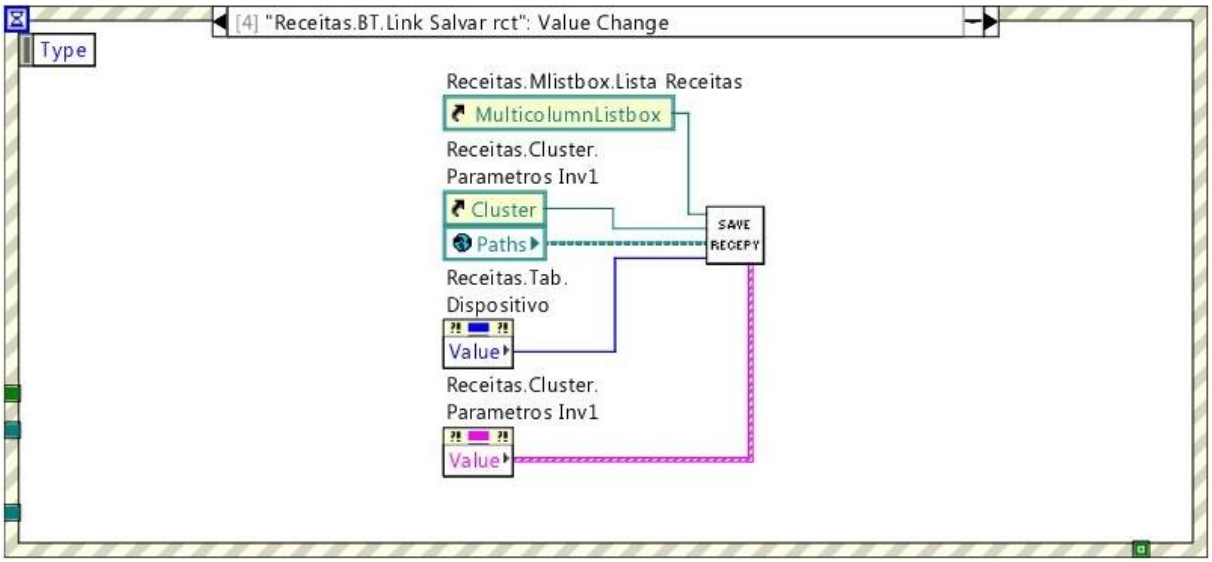


Figura 17 – Evento salvar receita

Fonte: Autoria própria

No evento de número cinco executa o comando de carregar os parâmetros das receitas criadas pelo sistema supervisório para o inversor de frequência selecionado.

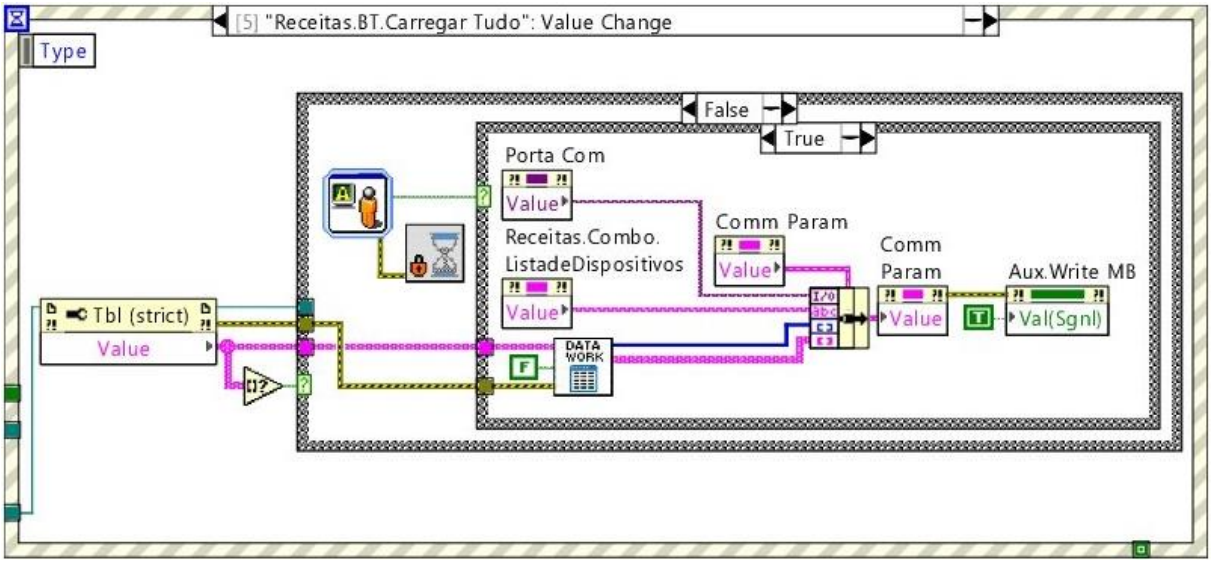


Figura 18 – Escrita dos parâmetros da receita

Fonte: Autoria própria

No evento da Figura 19 é feito o controle de visualização das variáveis do dispositivo selecionado.

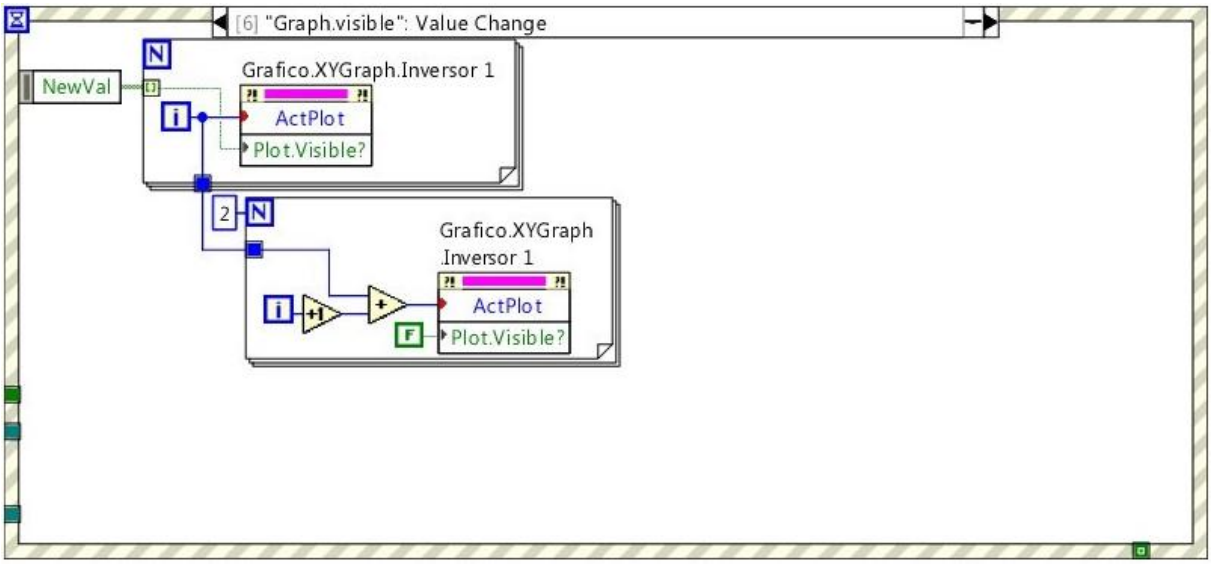


Figura 19 – Visualização das variáveis do dispositivo

Fonte: Autoria própria

Programação para realizar o controle de escala do gráfico, executando o ajuste da área do gráfico para os limites horizontais e verticais do gráfico com os dados visualizados. Nesta parte do programa também é feita a habilitação do gráfico para realizar o acompanhamento da atualização dos

dados em uma faixa ajustável de número de aquisições definida pelo valor do campo espaçamento na interface do usuário.

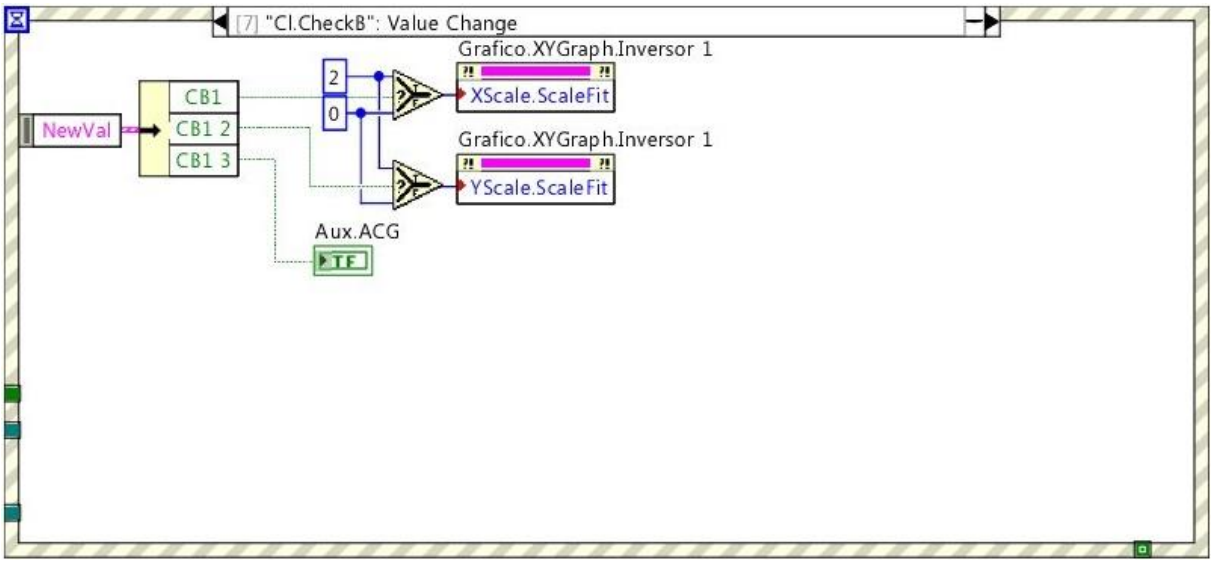


Figura 20 – Controles de visualização do gráfico

Fonte: Autoria própria

Nesta parte do programa é realizada a busca de novos dispositivos na rede e todos os dispositivos encontrados são disponibilizados para serem visualizados em uma tabela.

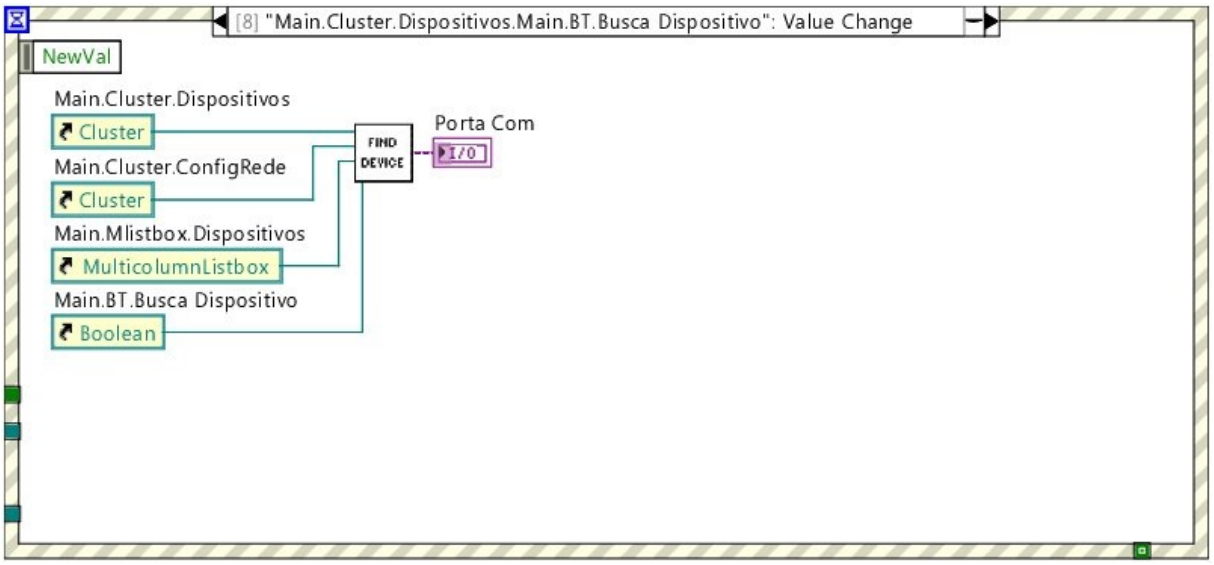


Figura 21 – Execução da busca dos dispositivos na rede

Fonte: Autoria própria

Para realizar a criação de uma nova receita a partir do comando feito na interface do usuário é iniciado a execução da lógica apresentada na Figura 22.

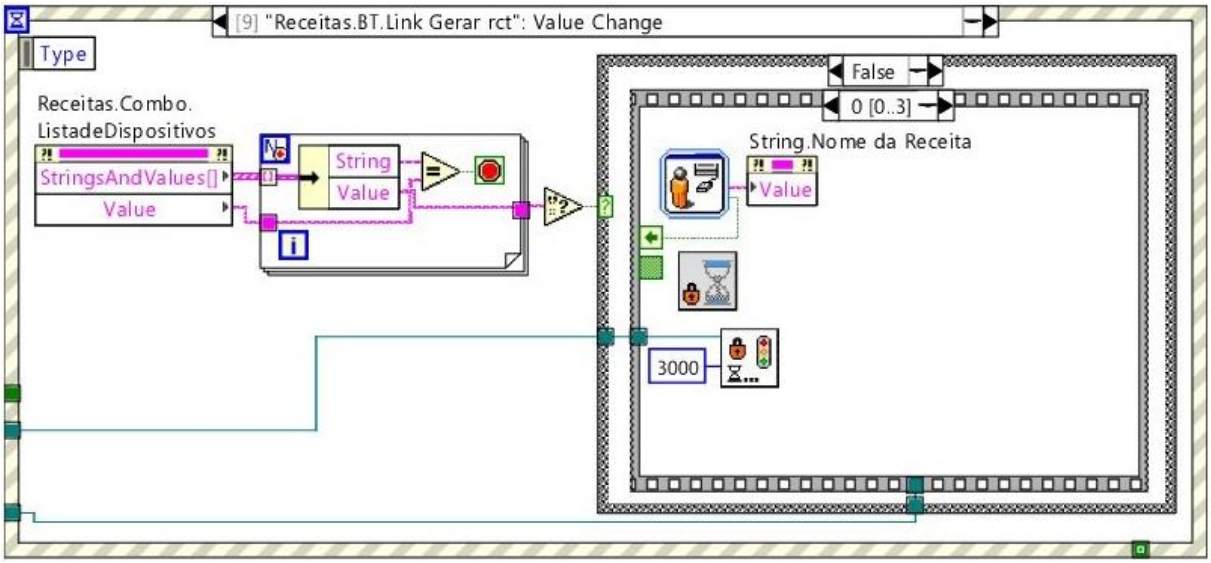


Figura 22 – Criar uma nova receita

Fonte: Autoria própria

Ao inserir valores nos campos da tabela de valores de parâmetro do dispositivo é necessário fazer uma verificação de qual parâmetro está sendo alterado para certificar de que o novo valor será valido. Na Figura 23 esta verificação é realizada verificando o parâmetro que está sendo alterado e certificando que ele está entre os limites de valores.

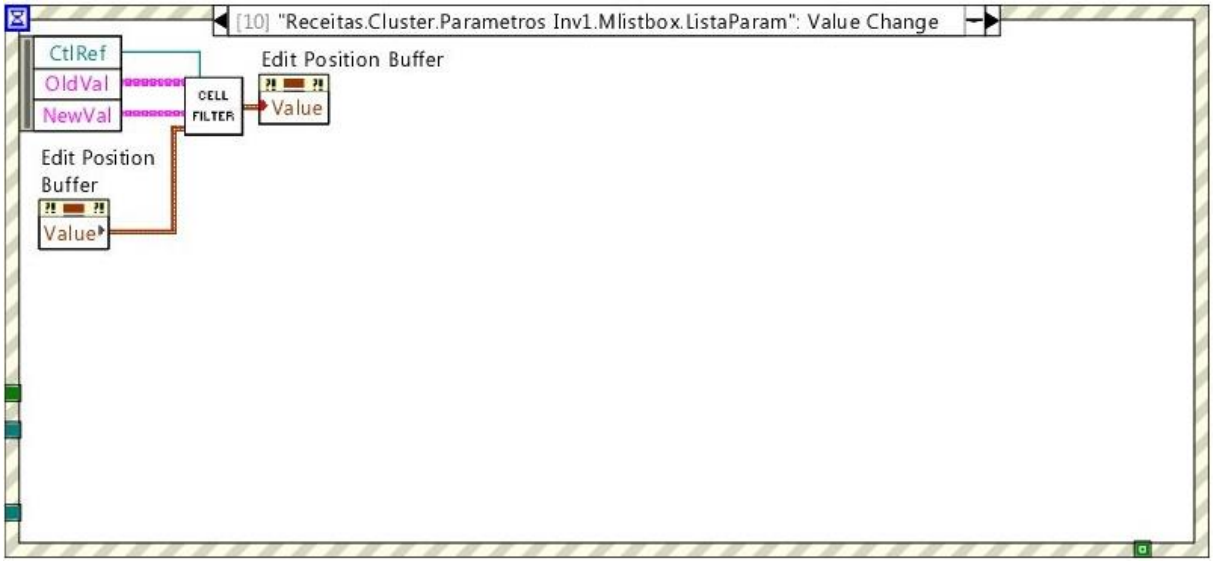


Figura 23 – Análise da tabela de parametrização

Fonte: Autoria própria

Para o controle dos valores válidos de cada parâmetro na tabela é necessário fazer o controle de qual célula está sendo selecionada para alteração de valor, na Figura 24 é feito este controle.

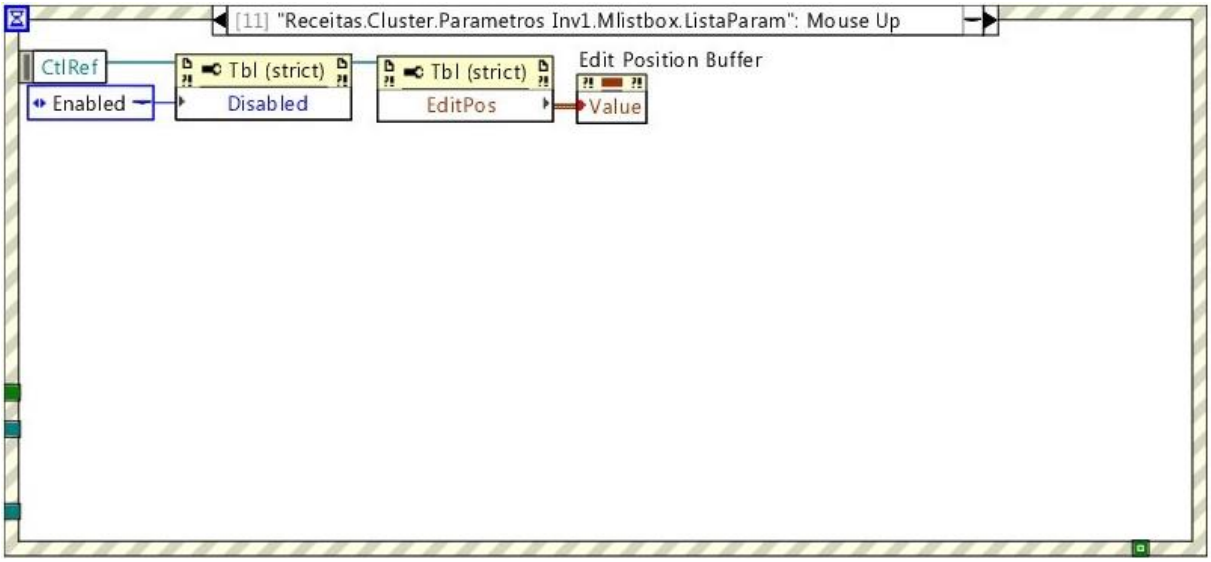


Figura 24 – Análise na posição da célula selecionada

Fonte: Autoria própria

Na lista de receitas salvas no sistema localizada na tela de receitas na aba arquivos de receita após selecionar uma receita e clicar no botão de “CARREGAR ARQUIVO DE RECEITA” é transferido todos os valores de parâmetros do arquivo salvo para a tabela de parâmetros executando a lógica na Figura 25.

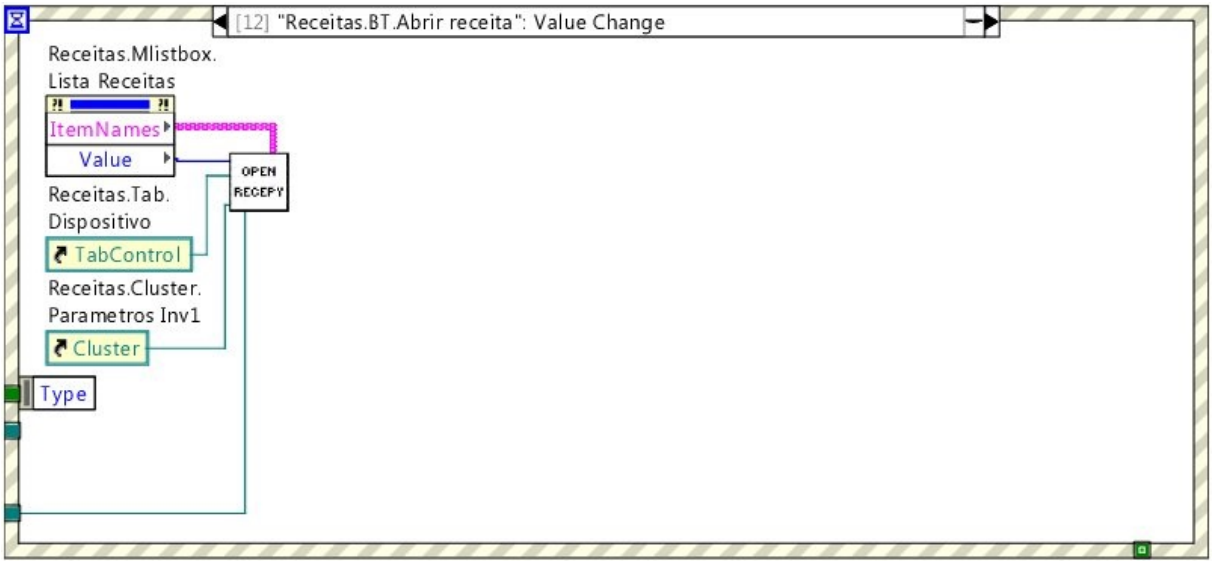


Figura 25 – Carrega receita salva

Fonte: Autoria própria

Para deletar uma receita que foi salva é executa a lógica na Figura 26 que realiza a verificação de qual receita está selecionada na tabela e em seguida pede uma confirmação se o comando deve ser realmente efetuado. Se a exclusão da receita foi realizada com sucesso uma mensagem é disponibilizada informando.

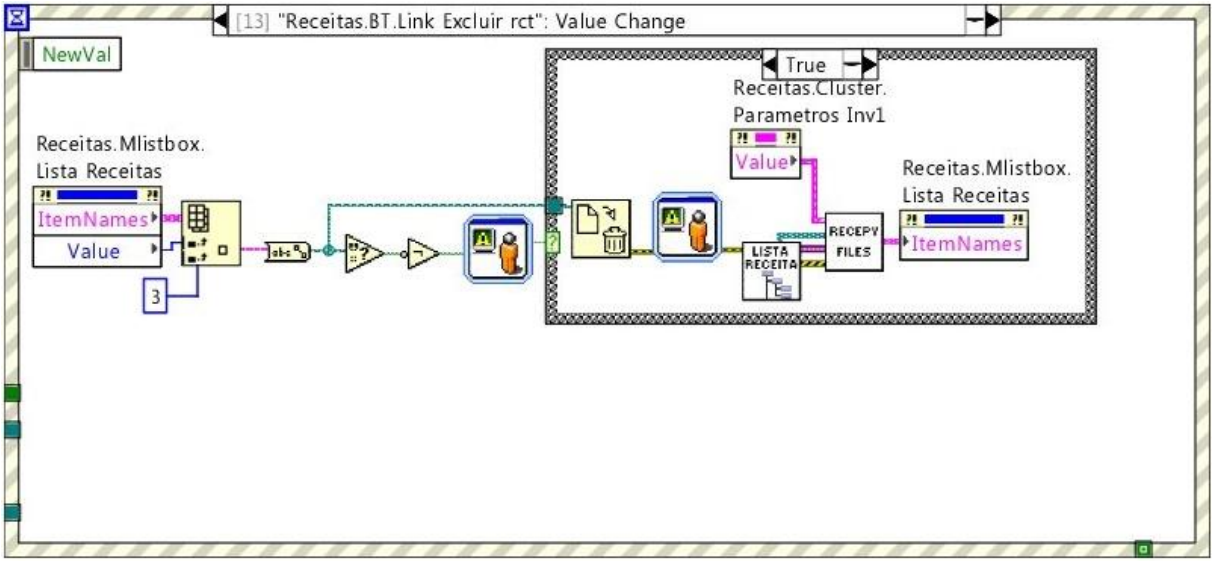


Figura 26 – Exclui uma receita salva no computador

Fonte: Autoria própria

Para ter acesso as telas de receitas e gráfico é necessário que esteja com um login válido, no evento da Figura 27 é demonstrada a lógica que faz este controle que é chamada ao se clicar no ícone de um cadeado no cabeçalho da interface do usuário.

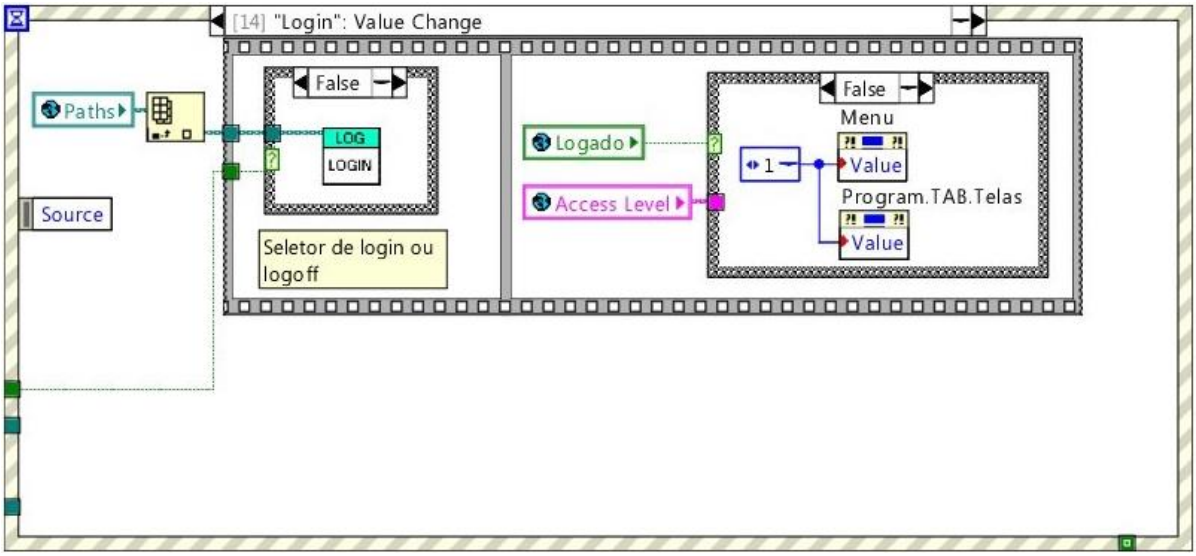


Figura 27 – Controle de *login*

Fonte: Autoria própria

Para sair do programa é necessário clicar no ícone com símbolo de desligar e ao clicar neste botão é executada a lógica da Figura 28, onde é exibida uma mensagem de confirmação para sair do programa e se for confirmada o programa é fechado.

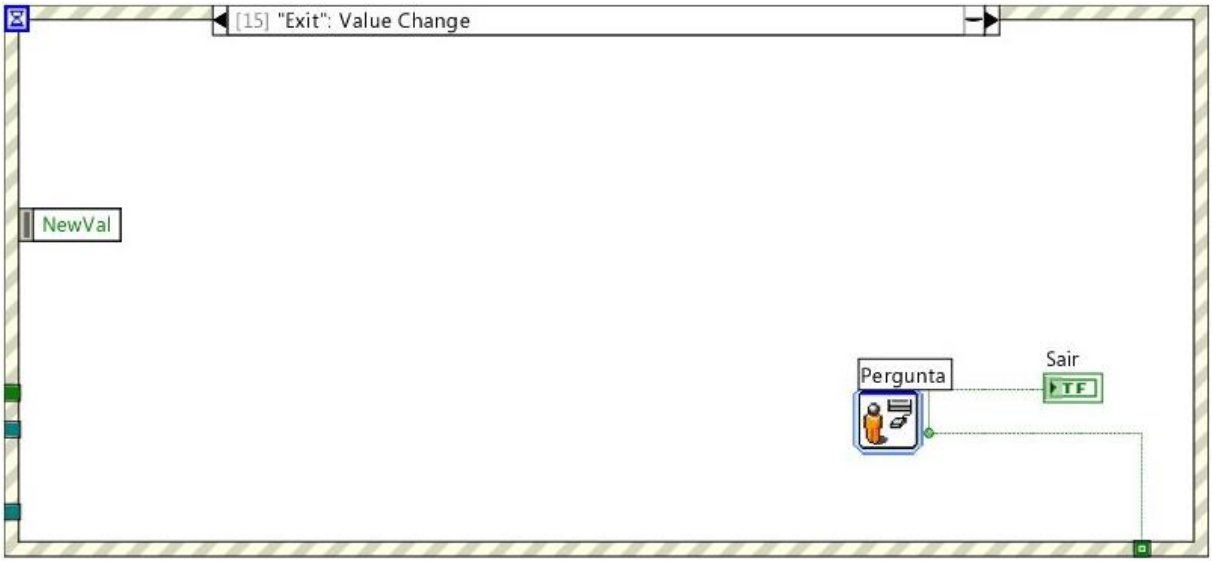


Figura 28 – Sair do programa

Fonte: Autoria própria

Na lógica do evento de número dezesseis é feito o controle da mudança de telas conforme a seleção feita no menu do cabeçalho do

software e fazendo o controle se existe um usuário que possui acesso para fazer estes comandos.

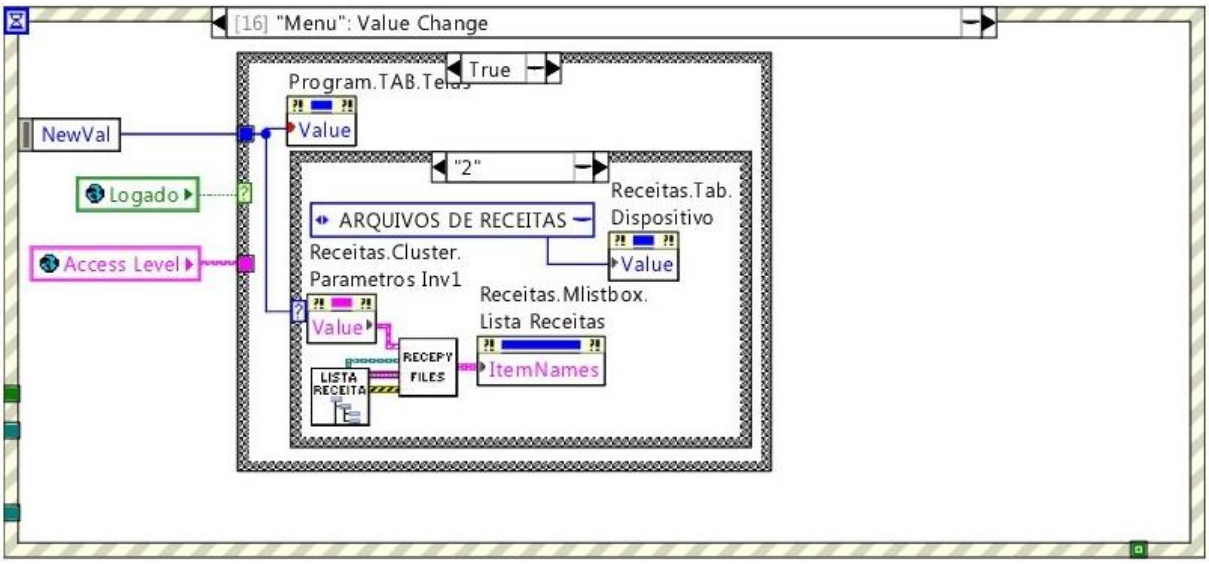


Figura 29 – Controle de troca de telas

Fonte: Autoria própria

A estrutura de comunicação é responsável por enviar e receber dados ao dispositivo desejado, e também por gerenciar o sequenciamento do envio e recebimento dos dados e garantir o bom funcionamento da troca de informações entre o supervisório e o dispositivo. Esta estrutura está sempre aguardando um pedido de trabalho que é enviado por alguma função através da estrutura de execução do programa. Ao receber o pedido as funções de leitura ou escrita são executadas e analisadas, se tudo ocorrer bem os dados são enviados a um buffer interno no programa ou despachados para o dispositivo. Na Figura 30 mostra como o *software* gerencia a comunicação e a lógica executada pelo programa.

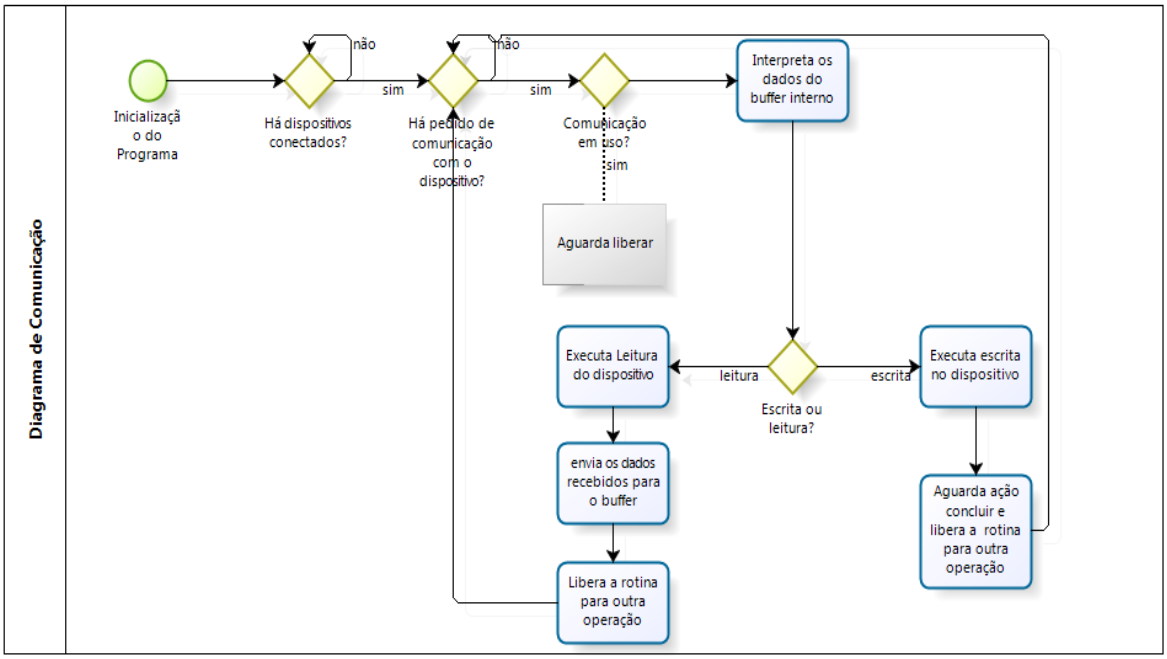


Figura 30 – Diagrama esquemático do módulo de comunicação

Fonte: Autoria própria

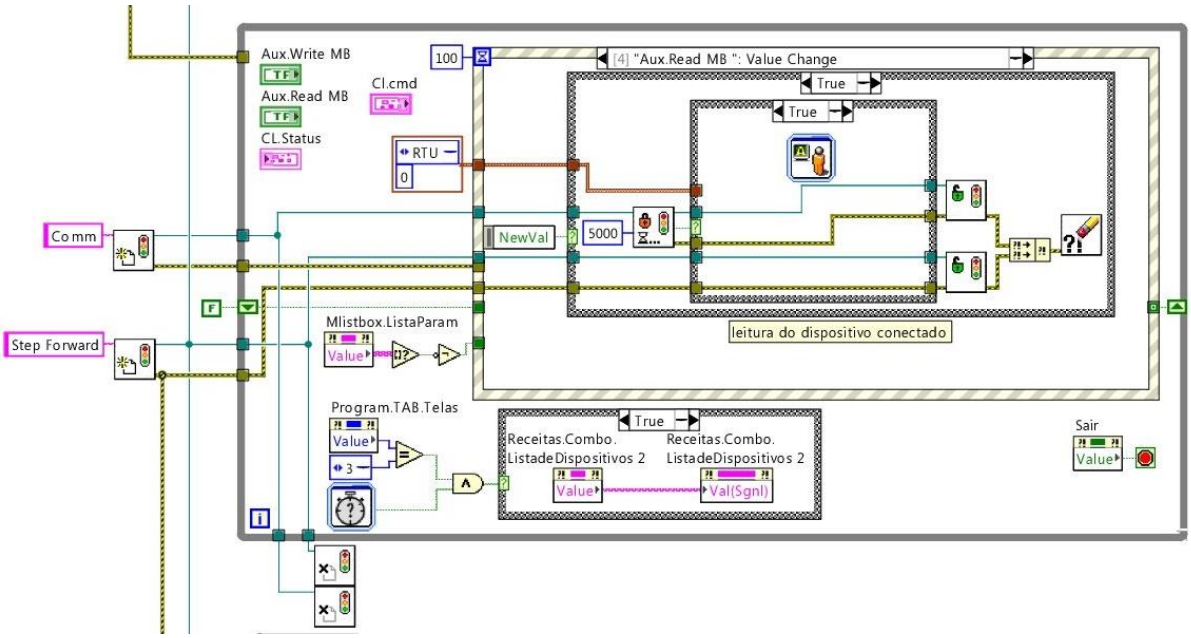


Figura 31 – Lógica de comunicação

Fonte: Autoria própria

A última estrutura do programa que está indicado na Figura 32 pelo número três, é a estrutura que faz a visualização das variáveis do inversor no gráfico em tempo real conforme a seleção de quais variáveis estão selecionadas no controle ao lado do gráfico. Nesta estrutura também é verificado se os comandos de acompanhamento do gráfico e ajuste do gráfico conforme os valores mínimos e máximos das variáveis visualizadas estão selecionados.

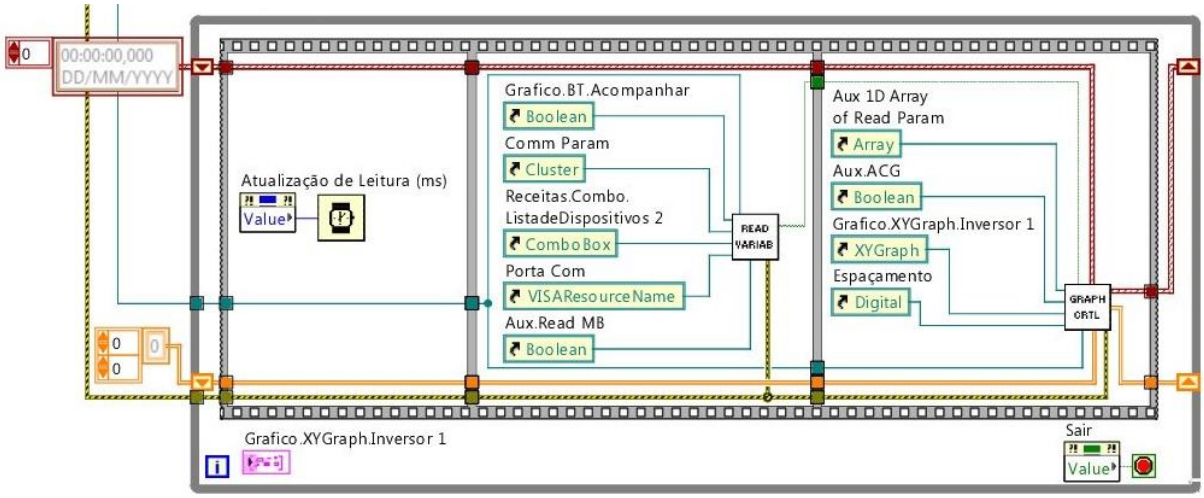


Figura 32 – Estrutura de controle do gráfico

Fonte: Autoria própria

4.3.1. Exemplo de rotina no ambiente de programação LabVIEW

A seguir são mostradas algumas figuras do diagrama que faz o pedido de leitura dos dados para a comunicação e a plotagem dos dados no gráfico, será mostrada apenas uma função para explicar uma das ideias de funcionamento do programa. A rotina exemplificada nos parágrafos seguintes fazem parte do bloco de execução, sendo apenas uma função deste bloco.

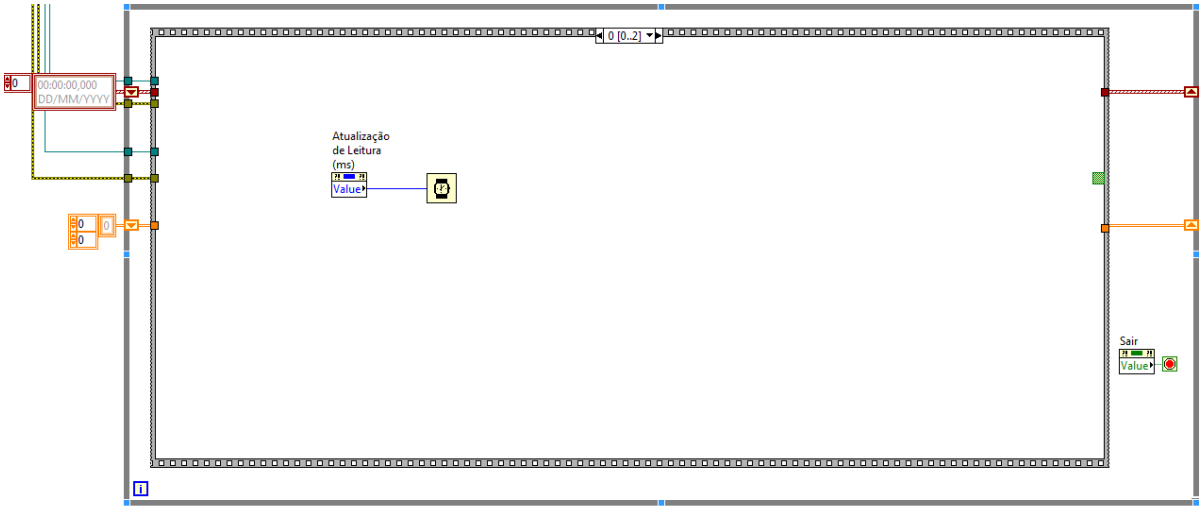


Figura 33 – 1ª etapa da rotina de monitoramento gráfico

Fonte: Autoria própria

Na Figura 33 mostra a primeira etapa da sequência de execução da rotina do gráfico, a estrutura mais externa (borda cinza preenchida) representa uma estrutura do tipo *while* e a estrutura cinza mais interna que é vazada é uma estrutura de sequenciamento, sendo que está possuí a etapa mostrada (na linha superior no centro) e mais outras duas que estão descritas abaixo. Nesta página da estrutura de sequência (página 0) um temporizador (bloco com desenho de relógio) garante que o *loop while* tenha uma pausa na execução para não sobrecarregar a comunicação que será requisitada na próxima página desta estrutura, como mostrado na Figura 34.

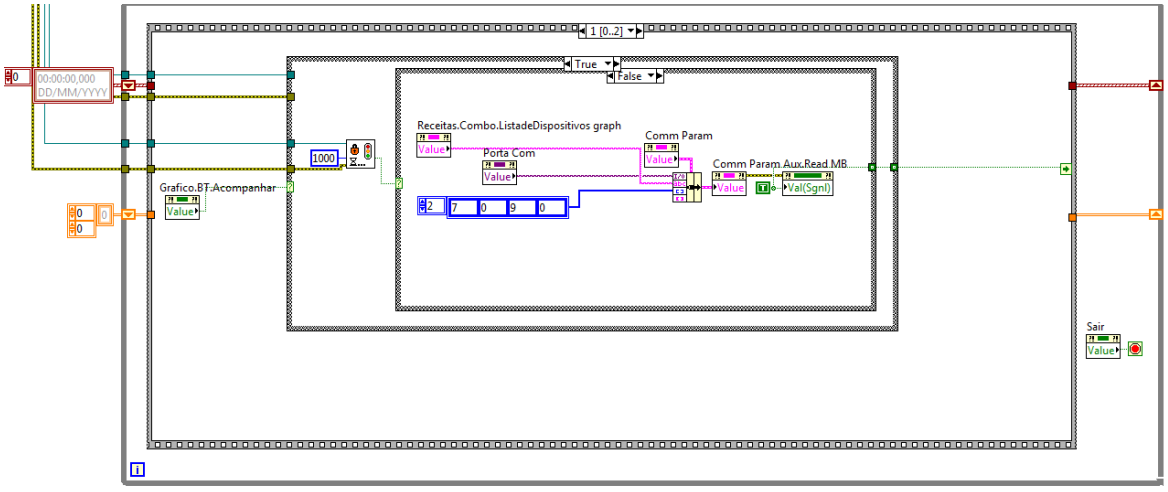


Figura 34 – 2ª etapa da rotina de monitoramento gráfico

Fonte: Autoria própria

Nesta etapa, mostrada na Figura 34, é enviado um “pedido de dados” através da parametrização de um objeto do tipo *cluster* que significa basicamente um aglomerado de objetos de variáveis de diferentes tipos, chamado na foto de “Comm Param Aux Read MB”. Os fios que são conectados a este objeto são os dados de pedido enviado pela estrutura. Este objeto é monitorado constantemente pela estrutura de comunicação, e assim que a estrutura de comunicação recebe as informações ela executa a leitura do dispositivo através da função de leitura do protocolo ModBus e retorna os dados conforme requisitados pela função para um outro objeto, que será lido na próxima etapa da sequência de execução.

Para tudo isto acontecer é necessário que o botão de acompanhamento esteja ativo e que a comunicação não esteja ocupada, estas questões são verificadas pelos dois componentes que estão ligados a entrada das duas estruturas condicionais (cinza com hachura) que estão dentro das outras duas estruturas explicadas anteriormente.

A próxima e última etapa de execução da função de monitoramento é o trabalho de usar os dados recebidos da comunicação, escaloná-los na grandeza correta e plotá-los no gráfico, que é feito pela etapa na Figura 35.

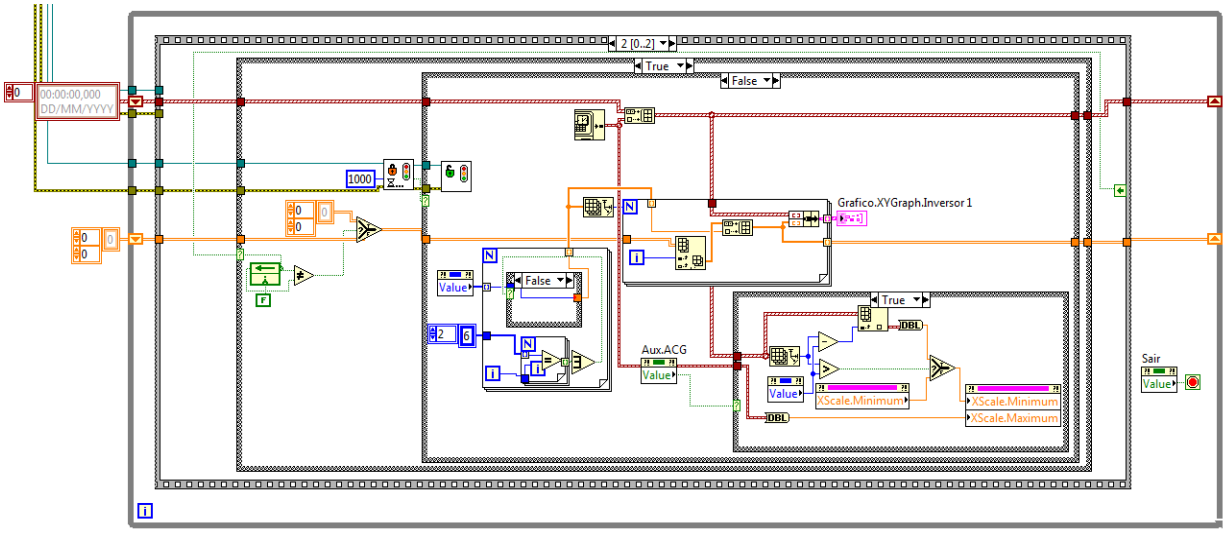


Figura 35 – 3ª etapa da rotina de monitoramento gráfico

Fonte: Autoria própria

Esta última etapa exemplificada acima, escalona os dados, assegura algumas condições básicas de como, por exemplo, haver dados para plotar,

aguarda a estrutura de comunicação responder com os dados pedidos, formata os dados lidos no padrão necessário para que o objeto gráfico seja mostrado na tela do usuário dentro da escala correta das grandezas e redimensiona a escala do gráfico caso a opção de “acompanhar gráfico” na tela de usuário esteja selecionada.

Nesta etapa faz-se necessário gerar um histórico com os dados já lidos e formatados por esta função anteriormente, tudo isto é armazenado em um *buffer* chamado de *Shift Register,* este *buffer* é representado pela seta sobre o *loop While*. A cada interação de *loop* o bloco *build array* junta os valores já armazenados anteriormente com os novos valores lidos da comunicação, formando um vetor crescente a cada interação com os dados a serem plotados, e também a cada interação de *loop* estes dados são plotados diretamente no objeto gráfico, criando assim a perspectiva de plotagem em tempo real.

Analisando da esquerda para a direita a função mostrada na Figura 35 temos a Figura 36, esta parte da rotina realiza algumas comparações básicas, como a de verificar se o botão "iniciar acompanhamento" pertencente a tela de monitoramento foi recém apertado, de modo a limpar o gráfico quando isto acontece, e também aguarda a resposta da comunicação com os dados do pedido que foi enviado previamente pela etapa anterior da rotina.

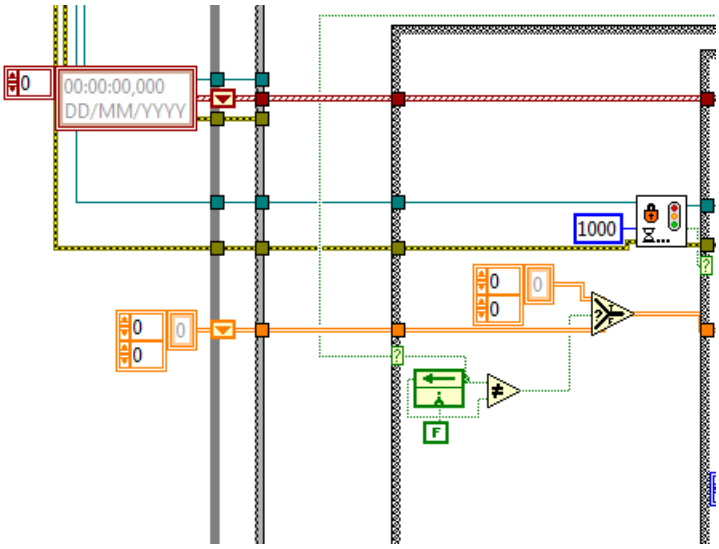


Figura 36 – 1º recorte da 3ª etapa da rotina de gráfico

Fonte: Autoria própria

Assim que a rotina receber os dados de comunicação, a etapa mostrada na Figura 37 executa algumas ações para efetivamente concretizar com o que foi determinado a esta função na programação, que é plotar um gráfico coerente. A seguir está detalhado o que os dois *loops* "*For*" e a estrutura condicional *Case* representam na rotina.

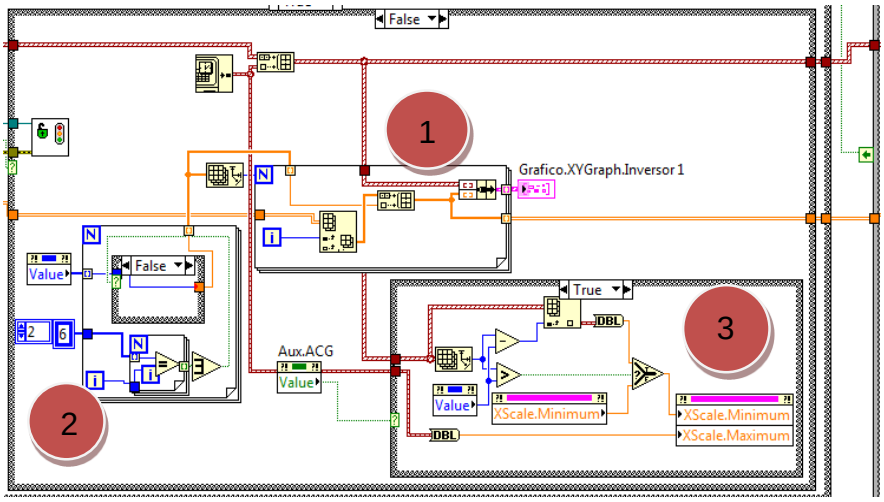


Figura 37 – 2° recorte da 3ª etapa da rotina de gráfico

Fonte: Autoria própria

1) Este pequeno *loop* "*For*" executa uma operação de formatação dos dados recebidos, isto se deve ao inversor enviar os dados lidos em escalas diferentes, por exemplo: O valor de tensão ao ser lido é enviado como "380", representando 380V, estando no caso dentro da escala correta, já o valor de frequência é recebido pelo supervisório como "450" que precisa de um ajuste para adequar a uma escala mais usual de frequência que é o Hertz, este valor recebido na saída da função é transformado em "45,0" e reanexado a um vetor com os outros dados que serão plotados no gráfico pelo *loop* "*For*" nº2. A explicação de se usar um *loop* é que realiza-se uma busca nos valores e este *loop* tem uma constante que indica quais valores são recebidos em uma escala não usual, e a cada interação o próximo valor do vetor recebido é analisado e adequado caso necessário.

2) Este *loop* "*For*" anexa os dados do relógio atual do computador com todos os dados recebidos do *loop* nº1 já formatados na grandeza mais usual, cria um *Cluster* com estes dados por exigência da função de gráfico do LabVIEW e os plota no objeto gráfico na sua saída a esquerda, concluindo a operação principal desta rotina que é gerar um gráfico.
3) Esta estrutura condicional serve para criar dinamismo na amostragem do gráfico para o usuário, ela interpreta o valor de espaçamento inserido pela interface e trabalha o vetor com os dados armazenados previamente pelas interações anteriores desta função garantindo que somente a quantidade informada no campo de espaçamento será visualizada no gráfico. Se por exemplo houvessem mais de 100 pontos no gráfico, e a opção de espaçamento estivesse com o valor 20, somente os últimos 20 dados do vetor de 100 pontos serão mostrados para o usuário. Esta estrutura só funciona se a opção "Acompanhar gráfico" estiver ativa na interface do usuário.

   Obs.: A interface referida é mostrada na Figura 56 deste documento.

Existem inúmeras outras funções no programa para realizar as outras operações existentes, como troca de tela, análise de parâmetros da tabela para checar coerência, busca de dispositivos etc, as etapas descritas nos parágrafos anteriores servem como um exemplo de como o projeto foi pensado e executado a programação somente no quesito de monitoramento e plotagem em gráfico. A explanação detalhada do programa é extensa, deste modo só uma função, a de monitoramento, foi explorada a fundo e exemplicada através do ambiente de programação e cria a noção de fluxo de execução e de como funciona a nivel de programação o LabVIEW.

4.3.2. Resultados

A separação por estruturas com funções diferentes no programa permitiu uma melhor organização e melhora nos tempos de ciclo (*loops*),

deixando a resposta ao usuário de uma ação cometida mais rápida, reduzindo os *slow downs* que uma estrutura única de programação causa por ela ser sequencial sem paralelismo.

A utilização dos recursos de estruturas paralelas, além de usar melhor os recursos do hardware são mais eficientes, pois são assíncronas umas às outras não bloqueando a execução de uma estrutura que realiza operações mais lentas que a outra, e, em casos de necessidade de sincronismo, outras ferramentas do LabVIEW foram usadas na programação, podendo então ter a estrutura sincronizada e dessincronizadas uma da outra de acordo com a necessidade no momento.

O *software* atingiu o principal objetivo proposto, em testes de campo conseguimos fazer a parametrização, o acompanhamento e enviar os comandos para o inversor. As receitas estão funcionando de maneira interessante, pois dispõe todos os parâmetros em uma tabela modificável e possível de salvar em arquivo para futuros acessos, tornando o programa um objeto capaz de melhorar a usabilidade e parametrização de inversores CFW-09.

4.4 COMUNICAÇÃO ENTRE O DISPOSITIVO E SISTEMA SUPERVISÓRIO

4.4.1 Introdução

Diferentes dispositivos conseguem trocar dados através de um meio de comunicação, o inversor de frequência CFW-09 possui um padrão de comunicação aberto que depende de ter uma placa com este recurso, neste trabalho utilizaremos a placa de comunicação do inversor (EBA) para atingir nossos objetivos. Aliado ao supervisório usaremos as portas de comunicação do computador para interfacear com o dispositivo.

A comunicação é de extrema importância no projeto, pois sem ela não seria possível atingir nosso objetivo, detalharemos então nos tópicos a seguir como ela funciona e os resultados no trabalho.

4.4.2 Tipo de comunicação

A comunicação aberta a dispositivos de terceiros que é permitida pelo inversor de frequência da WEG CFW-09 é a via meio físico Serial RS-485 que depende da placa de comunicação. Utilizaremos este meio físico para acessar o inversor, e como o computador que usamos no projeto não possuí uma porta serial RS-485 utilizamos conversores de meio físico para adaptar o padrão do computador com o padrão do inversor.

Os conversores podem ser eliminados ao ter uma porta serial RS-485 direto o computador, através de uma placa PCI ou de um cartão PCM-CIA. Este padrão físico permite que sejam ligados até trinta e um inversores em rede, porém a quantidade não é o foco do projeto. Na Figura 38 um exemplo de placa serial que poderia ser utilizado também.

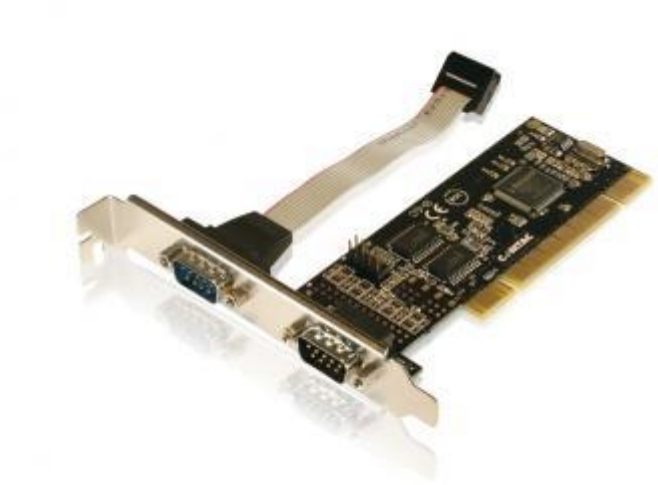


Figura 38 – Placa de comunicação serial[1]

4.4.3 Definições de protocolo

O protocolo de comunicação escolhido é o ModBus-RTU, pois é o protocolo que o inversor de frequência suporta assim como o LabVIEW, é um protocolo vastamente conhecido e usado nas indústrias apesar de estar sendo cada vez menos utilizado devido a evolução de novos protocolos mais rápidos e confiáveis.

A grande vantagem é que este protocolo é aberto e gratuito para uso e facilmente se tem suporte sobre ele. A ModBus e o meio físico serial RS-485 atendem os requisitos do projeto, pois não precisamos de grandes velocidades de comunicação e não há tráfego de muitos pacotes de rede entre o sistema supervisório e o inversor de frequência.

A ModBus possuí limitação na simultaneidade das operações, podendo apenas executar uma ordem de escrita ou leitura por vez, caso contrário é retornado um erro ao programa informando o uso da rede por outro pacote.

[1] Disponível em: <http://www.comtac.com.br/?url=produto&id=26> Acesso em ago. 2013.

4.4.4 Mensagens e telegramas

A comunicação padrão ModBus tem sempre pelo menos dois dispositivos, um obrigatoriamente único o mestre e outro(s) o(s) escravo(s). O sistema supervisório é quem gerencia a comunicação e pede e envia dados para a rede, sendo ele o mestre.

Para enviar e receber dados é necessário ter os seguintes parâmetros definidos:

- Porta de comunicação;
- Tipo de comunicação (RTU ou ASCII);
- Tipo de operação (escrita de uma ou mais variáveis ou leitura);
- Endereço da memória inicial padrão ModBus;
- Endereço do dispositivo na rede (00 até 31);
- Quantidade de dados (em caso de leitura);
- Tempo de *timeout* de comunicação.

O bloco de comunicação ModBus do LabVIEW possuí estas entradas e saídas de dados para parametrização e leitura, simplificadamente bastando parametriza-las e analisar a resposta de saída do bloco.

Ao escrever uma informação é necessário que ela esteja no formato WORD de 16 *bits* única ou em vetor, pois é o padrão de escrita do tipo *Multiple Registers.* Na Figura 39 é mostrado o bloco de comunicação do protocolo ModBus utilizado no projeto.

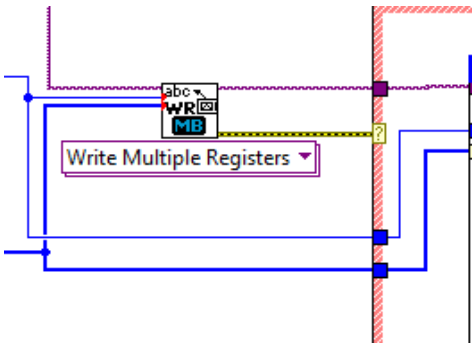


Figura 39 – Bloco de comunicação do protocolo ModBUS

Fonte: Autoria própria

Na Figura 39 é mostrado o bloco do protocolo ModBus parametrizado como escrita, as ligações provenientes do lado esquerdo do bloco são as

entradas dos parâmetros e dados a serem transmitidos para o dispositivo, os fios de cor azul mais fino e grosso representam respectivamente o endereço de escrita e os dados a serem transmitidos, ao lado direito são as saídas, a de cor roxa representa uma ID de comunicação interna do programa e o fio de cor amarela representa a resposta de erro do bloco, contendo nele informações como o código de erro e *status.* Este bloco é usado sempre que é preciso ser feito alguma comunicação com o dispositivo, no caso de utilizá-lo como leitura terá que mudar o *list box* embaixo do bloco para outra opção contendo na lista.

No caso de usar o bloco de comunicação no modo de leitura, os dados lidos seriam retornados em uma *WORD* ou também em vetor de dados numéricos de 16 *Bits*, conforme o tipo de memória lida estes dados precisam ser convertidos para poder serem usados adequadamente pelo programa e seria representado também por um fio azul pelo ambiente de programação.

Em alguns casos como o de monitoramento de estados do inversor, o valor recebido é um numérico de 16 *bits*, mas deve ser tratado *bit* a *bit* individualmente para poder interpretar a informação de estado. Abaixo são mostrados os possíveis estados que o inversor informa para a comunicação, sendo cada estado listado um *bit* da palavra de estado.

- Pronto;
- Sub-tensão;
- Funcionando;
- Local/Remoto;
- Erro;
- JOG;
- Sentido de rotação;
- Modo de ajuste após reset para o padrão de fábrica;
- Modo de ajuste após alteração do modo de controle de Escalar para Vetorial;
- Modo auto ajuste.

Para envio de comandos, utiliza-se também uma palavra *WORD* de 16 *bits*, sendo cada *bit* tratado como um comando lógico (C.L), estes *bits* são enviados pelo mestre para o dispositivo de acordo com a estrutura abaixo.

***Bits* superiores** - selecionam a função que se quer acionar, quando o *bit* é colocado em 1.

CL.15 - Reset de erros do inversor;

CL.14 - Sem função;

CL.13 - Salvar alterações do parâmetro P169/P170 na EEPROM;

CL.12 - Comando Local/Remoto;

CL.11 - Comando JOG;

CL.10 - Sentido de Giro;

CL.09 - Habilita Geral;

CL.08 - Girar/Parar.

***Bits* inferiores** - determinam o estado desejado para a função selecionada nos *bits* superiores.

CL.07 - Reset de Erros do inversor: sempre que variar de 0 para 1;

CL.06 - Sem função;

CL.05 - Salvar P169/P170 na EEPROM: 0 = Salvar, 1 = Não Salvar;

CL.04 - Comando Local/Remoto 0 = Local, 1 = Remoto;

CL.03 - Comando JOG: 0 = Inativo, 1 = Ativo;

CL.02 - Sentido de giro: 0 = Anti-horário, 1 = Horário;

CL.01 - Habilita Geral: 0 = Desabilitado, 1 = Habilitado;

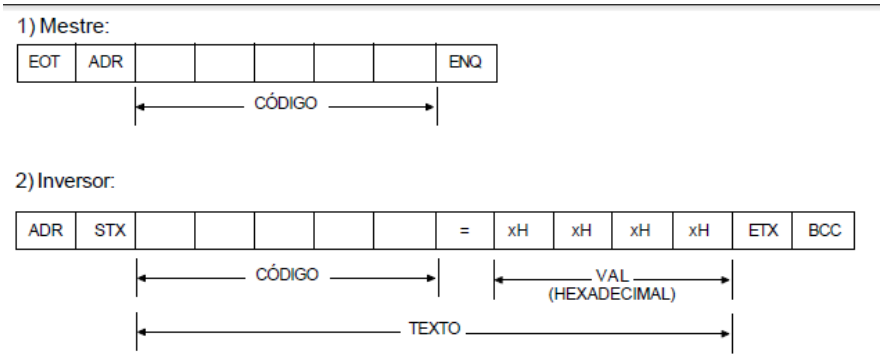
CL.00 - Girar/Parar: 0 = Parar, 1 = Girar.

OBS.: Para a execução da função acontecer deve-se ter o *bit* superior (C.L 08 à C.L 15) respectivo ao inferior selecionado (C.L 00 à C.L 07), ativo também.

Para dois equipamentos distintos trocarem dados entre sí é preciso um padrão para que ambos consigam interpretar os comandos e dados enviados um para o outro e este padrão é definido pelo protocolo ModBUS,

que tem como parte da sua estrutura exemplificada na Figura 40. A estruturação da palavra de envio dos dados pelo LabVIEW é feito internamente pelo bloco de comunicação ModBUS que é gratuito e fornecido pela *National Instruments*, não havendo necessidade de se trabalhar esta palavra mostrada na Figura 40, fica apenas a título de curiosidade de como o protocolo ModBUS padroniza a troca de mensagens entre dois ou mais dispositivos.

Na Figura 40 é mostrado duas estruturas, uma descrita como “mestre” e a outra descrita como “inversor”, estas palavras representam o envio (pedido) e a resposta do dispositivo de acordo com o pedido do mestre. Por exemplo, se o mestre pedir o valor de um parâmetro do inversor, ele envia para o dispositivo a palavra nº1 e obtem como resposta a palavra nº2 contendo nela o conteúdo do endereço da memória (parâmetro ou qualquer outro dado) do dispositivo que foi pedido pela palavra de leitura enviada pelo mestre.



Formato do telegrama de leitura:

EOT: caracter de controle End Of Transmission;
ADR: endereço do inversor (ASCII@, A, B, C, ...) (ADdRess);
CÓDIGO: endereço da variável de 5 dígitos codificados em ASCII;
ENQ: caracter de controle ENQuiry (solicitação);

Formato do telegrama de resposta do inversor:

ADR: 1 caracter - endereço do inversor;
STX: caracter de controle - Start of TeXt;
TEXTO: consiste em:
☑ **CÓDIGO:** endereço da variável;
☑ **“ = “:** caracter da separação;
☑ **VAL:** valor em 4 dígitos HEXADECIMAIS;
ETX: caracter de controle - End of TeXt;
BCC: Byte de CheCksum - EXCLUSIVE OR de todos os bytes entre STX (excluído) e ETX (incluído).

Figura 40 – Palavras de Leitura do protocolo ModBUS.

Fonte: Manual Completo Inversor CFW-09 da WEG

As informações descritas na Figura 40 representam como é tratado as mensagens e telegramas da comunicação, elas são usadas constantemente pelo programa supervisório para fazer a interface com o meio exterior através do padrão de comunicação determinado pelo protocolo ModBus. Esta palavra é internamente trabalhada pelo LabVIEW, apenas os parâmetros "ADR", "Código" e "Val" são parametrizados para comunicar, pois os outros a própria função trata de definir automaticamente e analisar em caso de erro na estrutura.

Ocorrendo sucesso na comunicação a palavra recebida é passada ao código do programa e utilizada conforme necessário finalizando a etapa de leitura de uma variável do inversor, sendo esta estrutura utilizada diversas vezes pela comunicação com parâmetros diferentes para conseguir extrair informação por informação do dispositivo uma por vez, este gerenciamento de repetição é realizado pela função programada de acordo com a necessidade em cada momento.

Como exemplos existem inúmeros casos facilmente encontrados em literaturas ou na internet, citando alguns pode-se afirmar que é possível usar o protocolo ModBus comunicando com supervisórios, PLCs, inversores, estações remotas de entradas e saídas que aceitem o protocolo, medidores de energia em geral entre outros diversos dispositivos.

4.4.5 Testes no campo

Usando dois fios comuns realizamos a ligação entre o computador e o inversor, sendo o mestre o computador e o inversor de frequência o escravo de rede. Um exemplo de topologia segue na Figura 41.

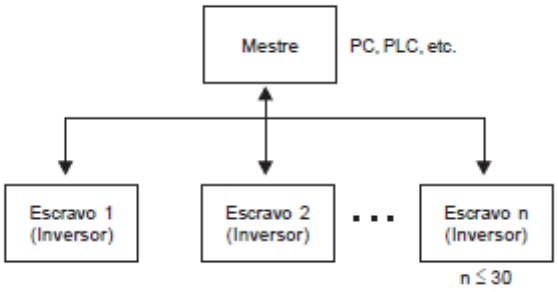


Figura 41 – Topologia de rede RS-485

Fonte: Manual do Inversor de Frequência CFW-09

A Figura 41 exemplifica uma topologia de rede chamada de mestre-escravo onde um dispositivo capaz de tomar decisões por meio de programação controla outros dispositivos escravos pendurados nos nós de rede, sendo que a quantidade de dispositivos (n) pendurados na rede ModBus não deve ultrapassar a quantidade de trinta.

A ligação entre o computador e o dispositivo necessitou de dois conversores de meio físico, partindo da porta USB para serial RS-232 e outro conversor de serial RS-232 para serial RS-485. A partir desta configuração os testes de comunicação foram feitos. Na Figura 42 é mostrado o esquema de ligação física dos dispositivos de um modo figurado, sendo que as setas bidirecionais representam os fios que interligam os membros da rede.

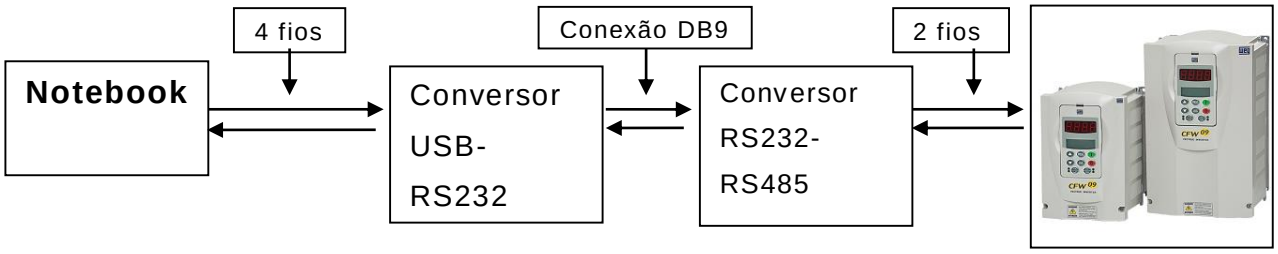


Figura 42 – Esquema de ligação física

Fonte: Autoria própria

Foram feitos testes com um conversor USB -> RS232 adquirido especialmente para o projeto porém por um motivo desconhecido o

conversor não sustentava a comunicação enquanto o motor estava ligado, gerando erros no supervisório e em alguns momentos inutilizando a porta de comunicação obrigando a reinicialização de todo o sistema, e, como havia um outro conversor no laboratório usamos ele nos testes e demonstrou suportar os requisitos de manter a comunicação ativa mesmo com o motor ligado, este conversor mostrado na Figura 43 passou a ser o utilizado em todos os testes daí por diante. Ao lado direito na Figura 43 encontra-se o notebook onde o cabo cinza USB do conversor está conectado, na outra ponta do cabo do conversor vai conectado o terminal DB9 referente ao conversor RS-485.

Figura 43 – Conversor USB -> RS-232

Fonte: Autoria própria

O outro conversor usado na estrutura física do projeto, RS232 -> RS485, se demonstrou suficiente para comunicar em todos os casos de funcionamento do dispositivo, com ou sem motor ligado e não foi problema para o trabalho. Na Figura 44 a continuação do cabo mostrado na Figura 43 é conectado ao conversor RS-485 universal mostrado no centro da imagem e ao lado esquerdo dois fios marrom e azul saem do conversor e são ligados ao inversor.

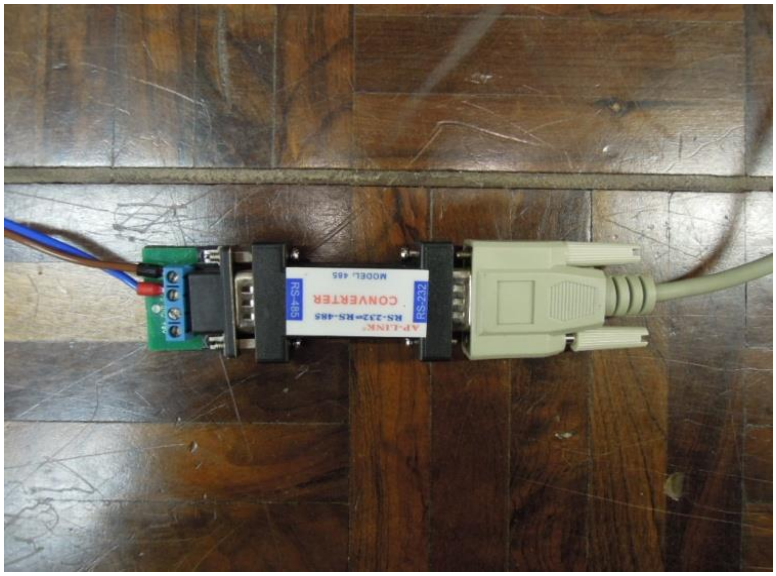


Figura 44 – Conversor RS-232 -> RS-485

Fonte: Autoria própria

Mesmo existindo entrada para alimentação externa neste conversor, não foi necessário utilizá-la pois usamos o computador ligado ao dispositivo em uma distância curta não causando perda de potência do sinal de transmissão.

Para concluir o circuito de rede, um par de fios convencionais foram usados para ligar a parte RS-485 do conversor mostrado na Figura 44 ao inversor de frequência mostrado na Figura 45. Os bornes 11 e 12 da placa EBA instalada no dispositivo inversor foram usados, estes dois terminais são próprios para receber tal comunicação, onde no borne 11 é a conexão A-LINE (-) e no borne 12 B-LINE (+) do padrão RS-485. Na Figura 45 o inversor sem a capa frontal é mostrada e os dois fios marrom e azul estão conectados a placa. A frente a IHM deste modelo está conectada mostrando o inversor ligado pronto para operação.

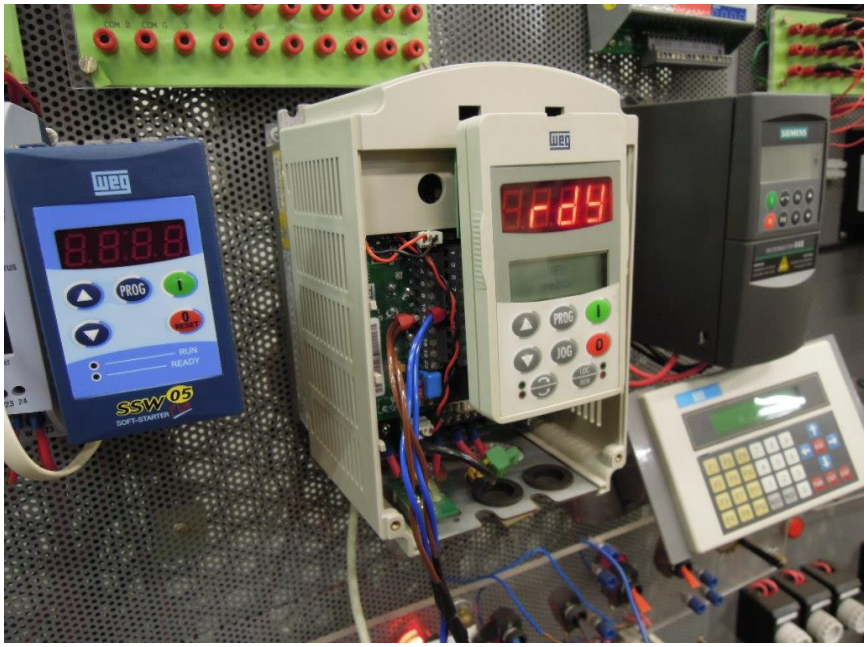


Figura 45 – Conexão dos fios de comunicação no inversor

Fonte: Autoria própria

O inversor de frequência da Figura 45 controla um motor disposto no painel do laboratório que usamos para realizar o trabalho, este motor Siemens está conectado fisicamente a um gerador, que serviu como uma carga leve para o motor mesmo que não tenhamos ligado os polos do gerador e usado-o para algum fim, não é parte do trabalho demonstrar um processo. O motor (cor verde) é mostrado na Figura 46, e o gerador (cor bege) está à frente conectado no eixo deste motor.

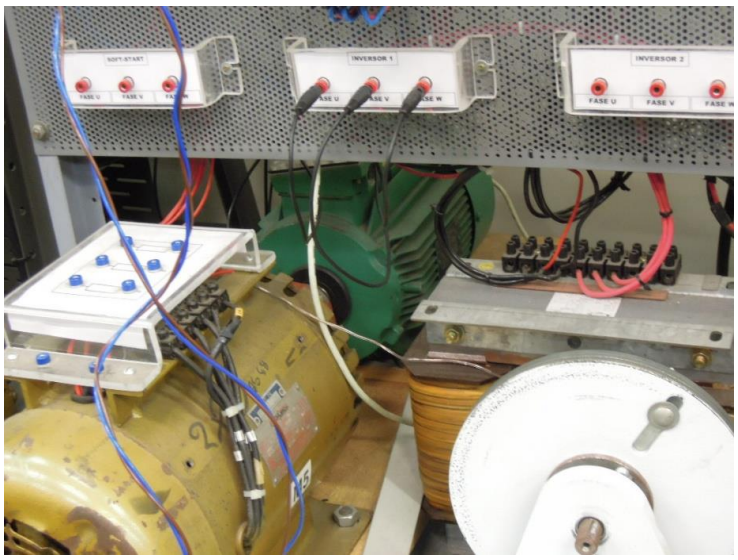


Figura 46 – Motor controlado pelo inversor

Fonte: Autoria própria

Obtemos êxito na comunicação após várias tentativas e ajustes no meio físico, foram feitas tentativas de usar a blindagem para amenizar ruídos, inversão dos canais no conversor nos casos de não conseguir localizar o dispositivo na rede, tentamos duas marcas de conversores USB para RS-232 devido a falha de monitoração dos dados com origem desconhecida, que conseguimos amenizar usando outro conversor conforme descrito. Todos estes problemas foram identificados e contornados graças aos testes de campo.

Com a comunicação com o inversor funcionando foi iniciado os testes do programa, onde vários aspectos foram sendo programados e modificados conforme a análise dos testes para conseguir atingir o fim desejado do programa. As primeiras tentativas de configuração foram frustradas, pois os padrões de dados escritos não estavam na grandeza imposta na tabela de parametrização, por exemplo, quando o usuário escolhia um parâmetro como "5,0" que significa 5 segundos, o valor no inversor era escrito como 0,5 segundos. Isto se deve a escrita ser feita somente em números inteiros e o inversor interpreta os valores na escala conforme ele está programado. Para contornar isto foi criada uma estrutura que analisa e informa a escala de cada parâmetro e ao ler e escrever no inversor uma função adapta os dados antes de mandar para a o dispositivo de modo que estes cheguem ao destino na escala correta e vice-versa.

Nos testes feitos na interface do usuário foi possível avaliar a usabilidade e correto funcionamento da navegação e funções, onde várias mudanças e implementações foram feitas para resolver os problemas de interface.

O ensaio foi feito especialmente para gerar as imagens e utilizar o programa para extrair as informações do inversor e mostrar na tela de acompanhamento do programa, outros diversos ensaios foram feitos durante a confecção e bateria de testes do programa, contudo não foram documentados por não haver necessidade.

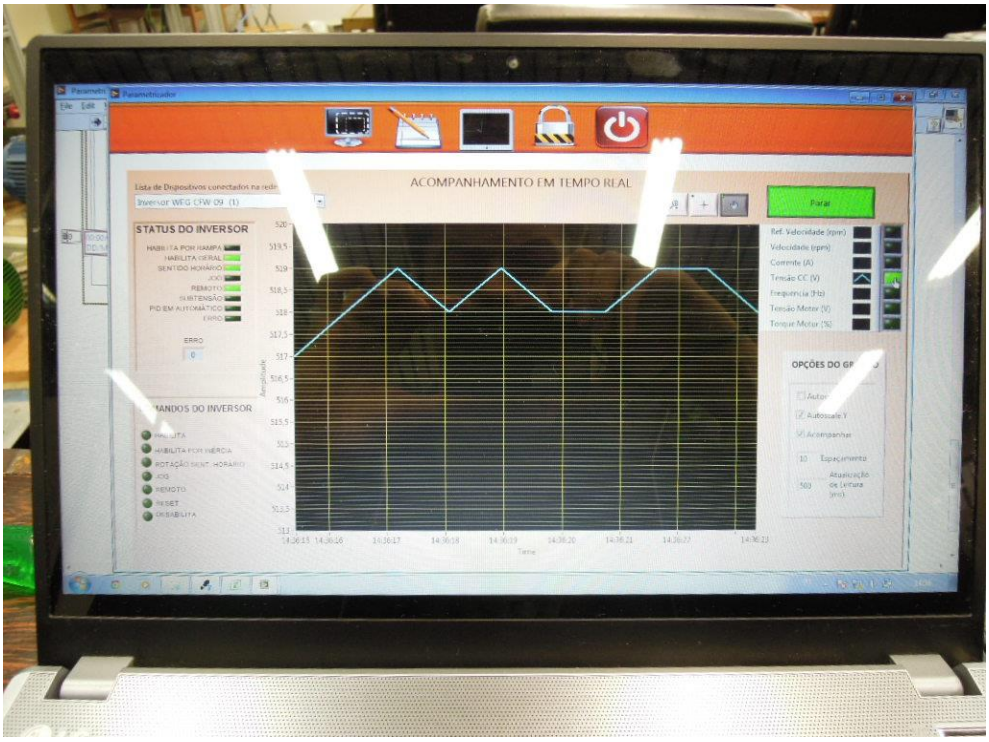


Figura 47 – Geração de gráfico do processo pelo supervisório

Fonte: Autoria própria

As funções e significado de cada objeto na tela são explanados nas sessões seguintes do trabalho, contudo a Figura 47 mostra o valor de tensão CC que é uma das sete variáveis de monitoramento do inversor que podem ser exibidas no gráfico quando as penas estão habilitadas. Esta imagem conclui o circuito de comunicação, provando que o supervisório é capaz de criar gráfico de variáveis validando assim todo o trabalho de comunicação feito incluindo *software* e *hardware*.

Na Figura 48 é mostrada uma foto do ambiente de trabalho que foi utilizado para testes e aprimoramento do projeto, o desenvolvimento foi em grande parte feito fora do laboratório, mas muitas modificações para melhorias foram feitos em campo.

Figura 48 – Local de trabalho no laboratório

Fonte: Autoria própria

Figura 49 – Inversor CFW-09 do laboratório

Fonte: Autoria própria

Na Figura 49 o inversor é mostrado com a capa de proteção frontal parafusada e com a IHM conectada.

4.4.6 Resultados

A comunicação funcionou conforme o esperado, foi atingido tempos de monitoramento inferiores a 100ms permitindo boa velocidade na aquisição dos dados para o gráfico em tempo real. A escrita de mensagens na rede é um pouco mais lenta, mas não gera nenhum problema para o bom funcionamento do programa parametrizador.

Na Figura 50 é demonstrado um fluxograma que representa como o programa trabalha internamente para acessar um dispositivo da rede, apenas a estrutura de leitura é mostrado, porém há também a estrutura de escrita não exibida que trabalha paralelamente ao diagrama.

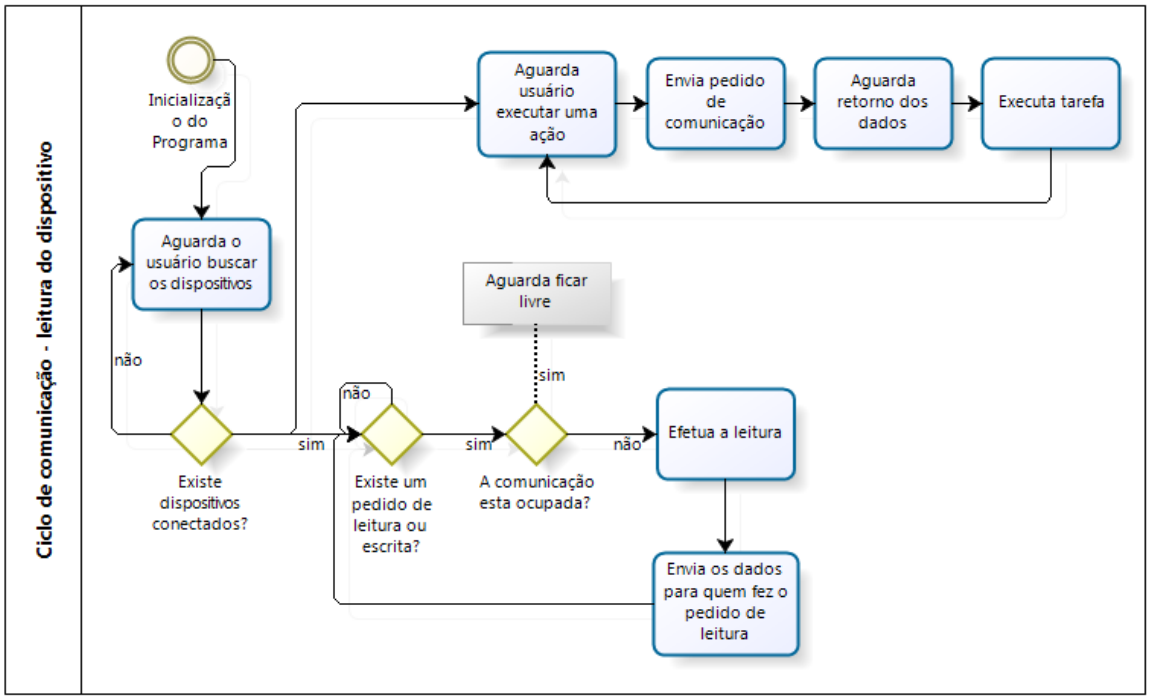


Figura 50 – Fluxograma de leitura do dispositivo

Fonte: Autoria própria

O fluxograma da Figura 50 mostra o resultado de como está funcionando a rotina de leitura no programa, diversos testes já foram feitos e o sistema funciona bem concomitantemente com as rotinas de escritas, que não podem efetuar uma operação de escrita junto com a leitura.

A parametrização da receita via comando do usuário ocorre corretamente e o inversor aceita a escrita dos parâmetros que estão de acordo com o princípio de funcionamento, e em caso de não aceitar ele não retorna nenhum erro impossibilitando a análise parâmetro a parâmetro, deste modo um botão de validação teve que ser criado para realizar a comparação dos dados após ter sido feita a parametrização. Isto não impede o correto funcionamento e é apenas uma restrição do dispositivo que funciona desta maneira, sendo um resultado positivo para a ação.

A tela de parametrização mostrada na Figura 51, exibe os campos editáveis e os campos das descrições integrados em uma tabela editável (os detalhes sobre esta tela são explicados no Capítulo 5) sendo que os campos em verde podem ser alterados pelo usuário. As funções dos botões exibidos no canto inferior direito da Figura 51 foram testados e criados para atender a necessidade do projeto tornando o sistema maleável, pois permite ao usuário escrever os dados ou ler somente no momento em que clicar nos botões, a ideia foi elaborada desta forma para evitar que um sistema automático, caso fosse, acabasse executando uma ação indesejada sem reconhecimento do usuário.

.

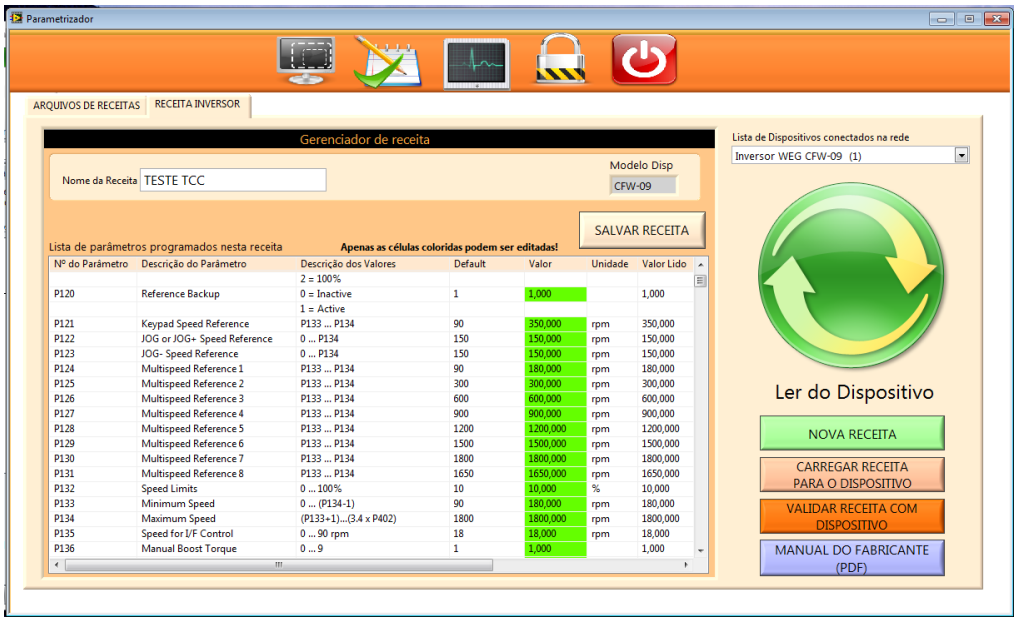


Figura 51 – Tela de parametrização

Fonte: Autoria própria

A aquisição de dados também funcionou, como também as funções do gráfico exibindo dados em tempo real do inversor e estes são exibidos na interface do usuário onde também a visualização deles pode ser manipulada conforme o usuário desejar. Um exemplo de saída de dados está mostrado na Figura 52, onde foi recortado a aquisição no momento em que um comando de desligar é enviado ao inversor, o gráfico é maleável e pode-se navegar para valores anteriores usando as ferramentas da tela (os detalhes sobre esta tela são explicados no Capítulo 5).

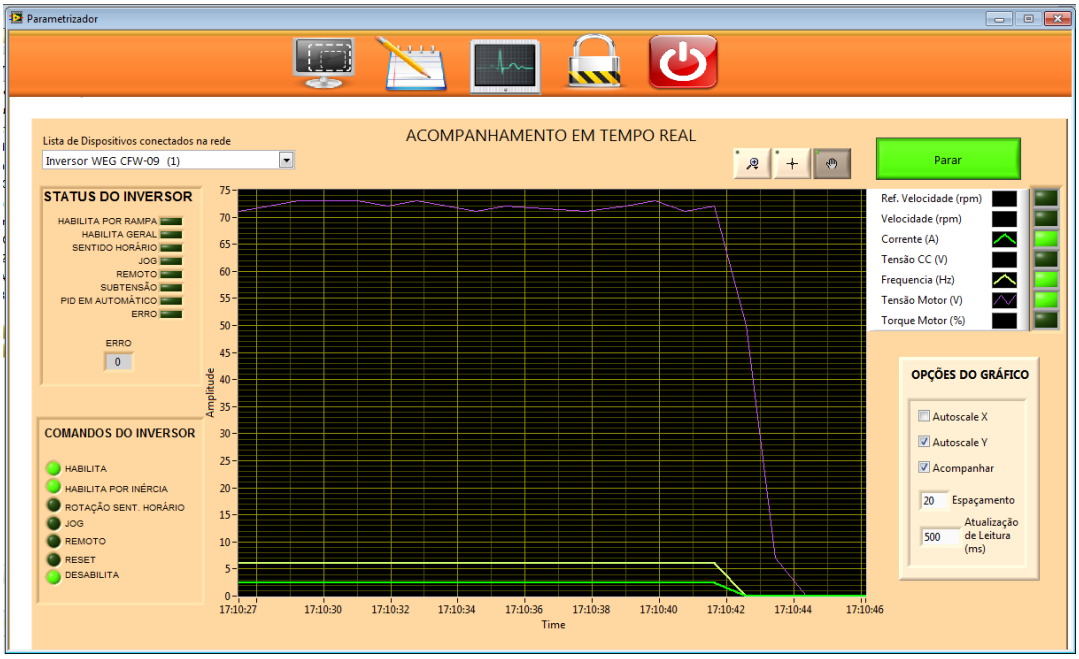


Figura 52 – Gráfico de desligamento do inversor

Fonte: Autoria própria

4.5 COMPARAÇÃO ENTRE PARAMETRIZAÇÃO MANUAL E SISTEMA SUPERVISÓRIO

4.5.1 Introdução

Para realizar um comparativo de ganho de tempo entre a parametrização manual e a via *software* efetuamos a coleta de dados, onde estipulamos um caso hipotético, simular as ações e cronometrar o tempo de modo a mostrar os ganhos de tempo. Foram excluídos os outros fatores dos

testes realizados por não ser possível simular, como por exemplo, a confiabilidade na parametrização, tempo de deslocamento da pessoa até o dispositivo no caso que este estivesse em campo, estes pontos foram menosprezados nos testes e apenas analisado o tempo.

4.5.2 Coleta de dados em teste de laboratório

A coleta de dados serve para efeito comparativo da vantagem de usar um *software* que faça a parametrização ao invés de usar a IHM do inversor, analisa basicamente o tempo de parametrização, desconsiderando outros fatores que também são importantes mas não são o foco do projeto.

Basicamente a coleta foi feita definindo-se uma condição inicial, que é o inversor nos parâmetros de fábrica, e um objetivo que é ter o inversor parametrizado para um processo hipotético.

Os tempos de parametrização foram cronometrados, foi definido 28 parâmetros aleatórios e os tempos obtidos foram de cinco minutos para a parametrização manual e dois minutos e meio para a parametrização via *software*.

4.5.3 Análise de dados

Comparando os tempos coletados e algumas vantagens que o programa traz constatou-se que o *software* se sobressai em diversos aspectos, como por exemplo:

- Tempo de parametrização, o usuário consegue ganhar tempo em parametrizações que sejam mais longas pois os valores só serão escritos no final do trabalho de parametrização feito na janela de programação.
- Pode-se espelhar a parametrização criada para outros inversores dentro de uma rede. Como as parametrizações podem ser gravadas em arquivos, para fazer o espelhamento basta que o usuário conecte outro inversor a rede e o selecione

no menu, tornando ele alvo das alterações e executar a função de gravar os dados que todos os parâmetros do inversor serão atualizados de uma só vez.

- Não é necessário estar próximo ao equipamento, poupando tempo de deslocamento ou dificuldade de acesso. Tanto faz a distância desde que seja respeitada as limitações físicas da comunicação, o comando e os dados são enviados ao inversor onde quer que ele se encontre, bastando apenas estar conectado e endereçado na rede.
- Os parâmetros podem ser salvos, ganhando em tempo na próxima programação. Arquivos são gerados quando a função de salvar é executada, podendo ser reutilizado pelo programa para parametrizar outro inversor.

As informações do inversor que são lidas em tempo real pelo programa são exibidas em gráfico como já foi mostrado, representando um ganho em relação ao uso somente do equipamento com a IHM, especialmente quando há a necessidade de diagnosticar e analisar o comportamento do motor em um processo onde muitas vezes precisa-se detalhar os dados e criar histórico para conseguir chegar a uma solução. As variações de carga no motor decorrentes da partida, desligamento e pleno funcionamento são geralmente dinâmicos e varia em conjunto com outros diversos fatores, como o tempo, desgaste dos rolamentos, carga no eixo etc. tornando o uso exclusivo da IHM para diagnóstico inviável, neste quesito então o programa é benéfico por converter os dados numéricos em informações gráficas mais facilmente compreensíveis.

5 DESCRITIVO DE TELAS DO PROGRAMA DESENVOLVIDO

As telas do programa desenvolvido estão exibidas abaixo junto com a descrição de cada uma.

5.1 TELA PRINCIPAL

A tela principal exibe as configurações de rede e informações sobre os dispositivos conectados.

Figura 53 – Tela Principal do Programa

Fonte: Autoria própria

A Figura 53 – Tela Principal do Programa é a tela inicial de abertura, nesta tela o usuário tem acesso as configurações da rede ModBus, parâmetros de rede como a porta de comunicação, velocidade, paridade e controle de fluxo podem ser personalizados conforme a necessidade. Também nesta tela são listados os inversores, que estão conectados no

momento após feito a busca na rede que estejam de acordo dos parâmetros especificados.

O sistema de busca de inversores simplesmente envia um pacote para cara endereço dentro da faixa especificada e analisa a resposta, se houver resposta o dispositivo é listado na tabela e pode ser usado pela parte de parametrização do programa.

5.2 TELA DE LISTA DE RECEITAS

Nesta tela são mostradas as receitas que já foram feitas e gravados no computador, e podem ser carregadas para parametrizar um novo inversor ou ter os parâmetros editados.

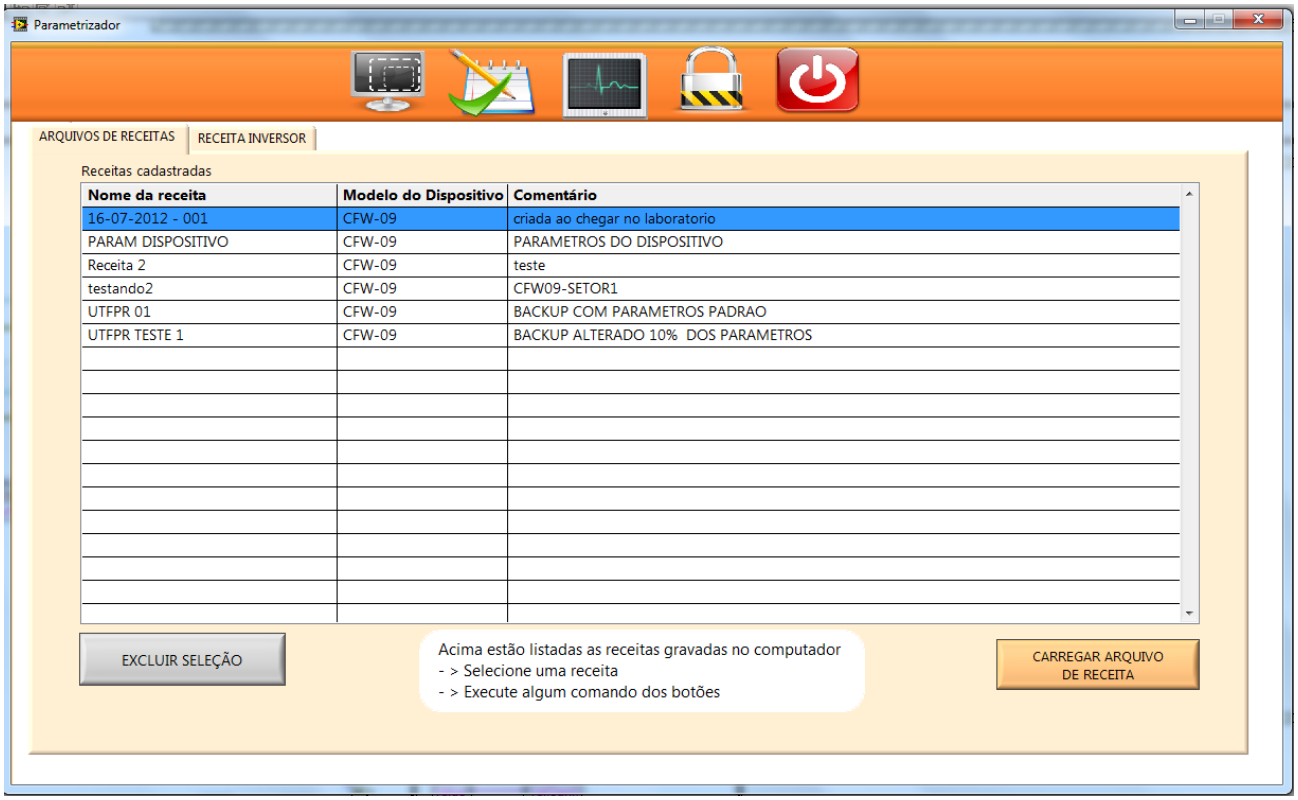


Figura 54 – Tela Lista de Receitas

Fonte: Autoria própria

A Figura 54 – Tela Lista de Receitas contém a lista de todas as receitas que o programa localizou em seu banco de dados, cada receita possui um campo que mostra o modelo do dispositivo que a receita pertence

e um comentário criado por quem a salvou e serve para auxiliar o usuário a saber qual o propósito da receita. Nesta tela a receita pode ser tanto carregada como excluída. Bastando para tal selecionar uma receita da lista já existente e clicar nos dois únicos botões na tela para executar uma ação, carregar ou excluir. Ao clicar em carregar uma *pop-up* é exibida pedindo confirmação da ação, e ao clicar em carregar, automaticamente a receita é carregada para a tela de parametrização mostrada no próximo tópico, e exibida no formato de tabela.

5.3 TELA DE PARAMETRIZAÇÃO DO INVERSOR DE FREQUENCIA

A tela de parametrização mostra todos os parâmetros da receita, permite personalização pelo usuário e faz a leitura dos parâmetros do inversor selecionado.

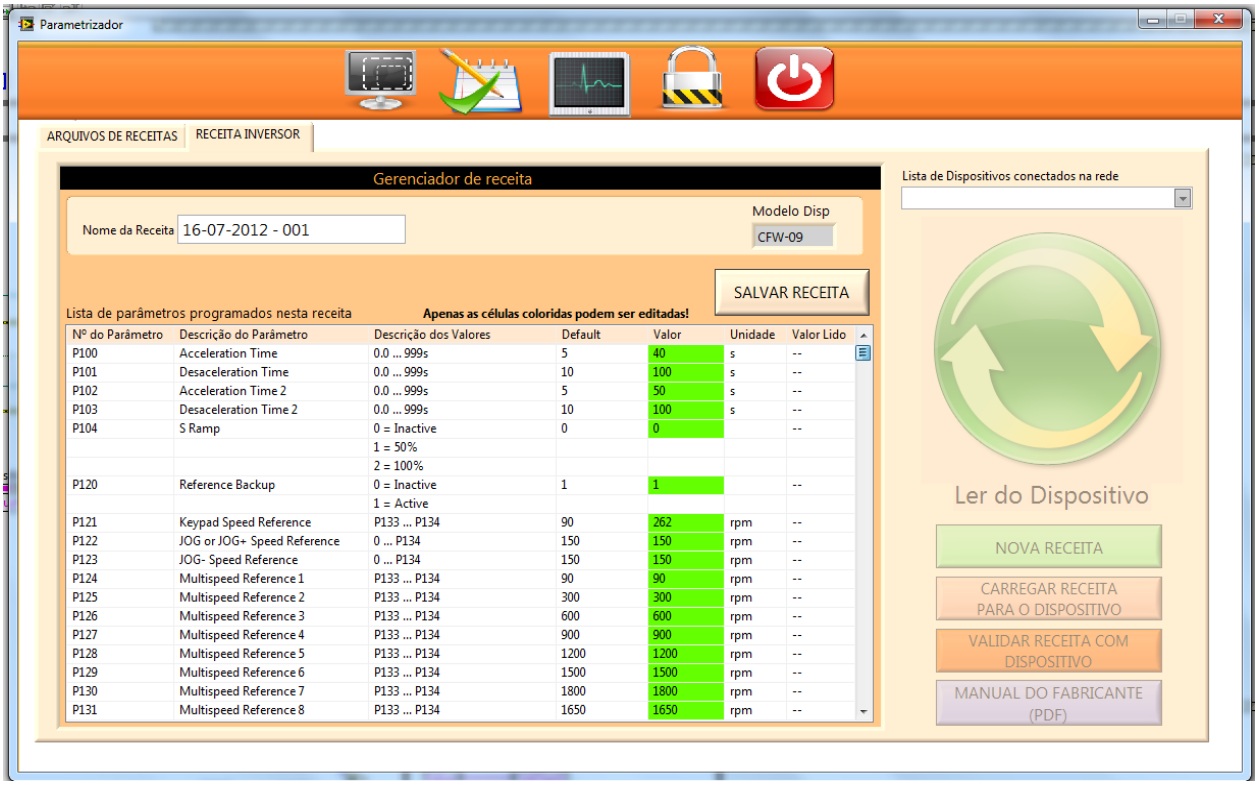


Figura 55 – Tela de Parametrização

Fonte: Autoria própria

Grande parte da funcionalidade do programa está contida na tela mostrada na Figura 55. Os dispositivos localizados na tela principal são mostrados na lista de dispositivos desta tela, permitindo ao usuário escolher o dispositivo em que deseja trabalhar e executar as ações existentes no programa. O usuário pode criar uma receita com parâmetros padrões da WEG, ou criar uma receita a partir dos parâmetros do dispositivo selecionado e salvá-los em arquivo para futuro uso.

A personalização dos parâmetros é feita integralmente direto nos campos em verde da tabela, e cada receita pode ser nomeada conforme desejado e salva no banco de dados.

O programa permite que a receita inteira seja descarregada no inversor de frequência, sendo que o *download* dos parâmetros são analisados pelo programa e se caso algum erro ocorrer o usuário é informado. Se por exemplo por falha de comunicação algum parâmetro falhar na escrita, uma mensagem é mostrada ao usuário para alertá-lo, tendo então que o usuário averiguar o problema e executar novamente o comando para carregar os parâmetros para o inversor.

Existe o recurso de validação de receita com o dispositivo, com a função de garantir que os parâmetros desejados foram completamente escritos e aceitos pelo dispositivo, havendo alguma discordância o usuário é informado qual(ais) parâmetro(s) esta(ão) divergente(s). Se por exemplo o parâmetro P100 no inversor estiver como "4,0" e este parâmetro estiver escrito no campo verde referente ao P100 da Figura 55 outro valor se não o "4,0", ao executar a validação o *software* acusará discrepância nos dados e mostrará que o parâmetro da receita P100, no caso, está em discordância com a receita. É emitido apenas um aviso informativo e esta análise auxilia o usuário para garantir que a parametrização desejada realmente foi aceita e está operante no inversor.

5.4 TELA DE ACOMPANHAMENTO DO DISPOSITIVO

A tela de acompanhamento gera gráfico em tempo real e permite o usuário mandar comandos ao dispositivo selecionado que está conectado na rede.

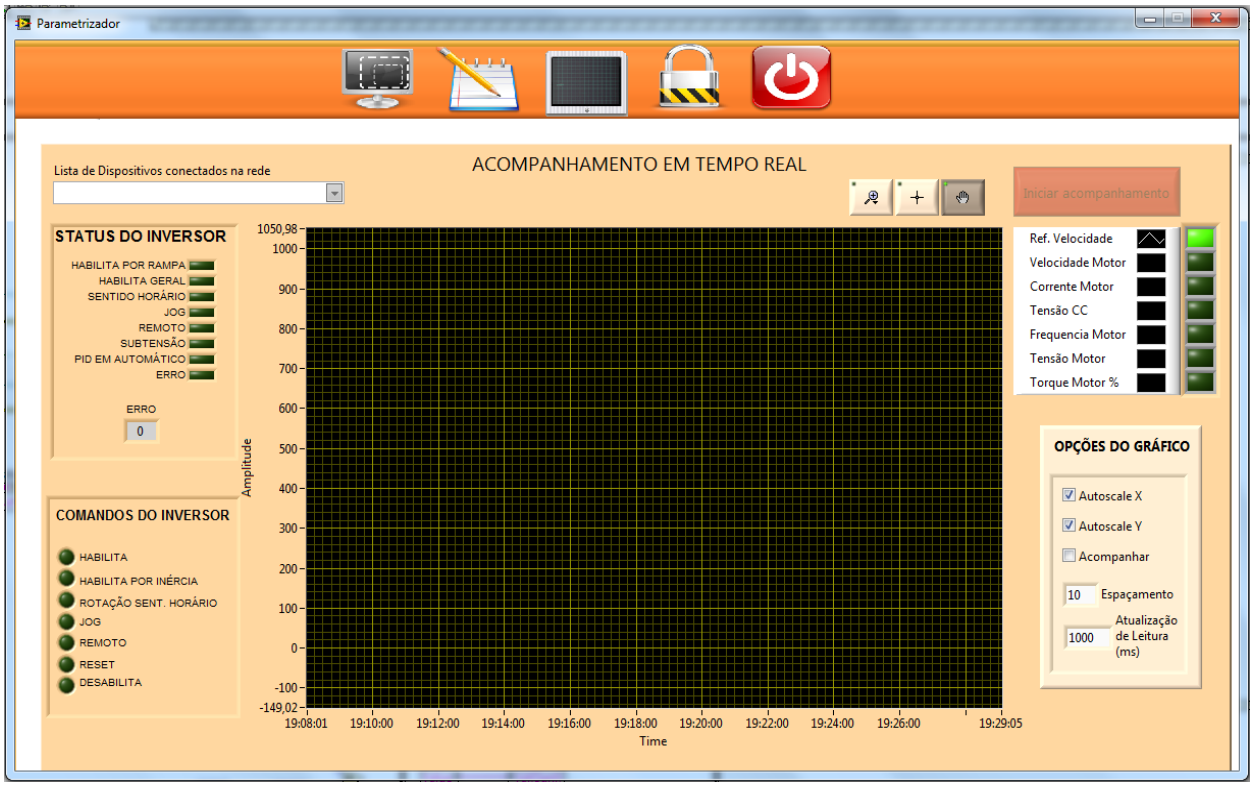


Figura 56 – Tela de Acompanhamento

Fonte: Autoria própria

O gráfico é gerado a partir de todas as grandezas que o inversor de frequência consegue medir internamente, a visualização das penas é personalizável através dos recursos criados e dos recursos do próprio objeto gráfico que são mostrados ao usar o botão direito do mouse sobre as penas. Existem também ferramentas que estão na parte superior do gráfico onde o usuário pode usar os recursos de *zoom,* e permite deslizar o gráfico conforme os cliques do *mouse*. Ao selecionar o botão  , o usuário consegue clicar no gráfico e modificar o *zoom* dele conforme desejado. A ferramenta  de "mão" serve para deslocar o gráfico no eixo X e Y, bastando apenas clicar e arrastar sobre o gráfico que ele se deslocará no

sentido do movimento feito com o *mouse.* O outro ícone de cruz não é utilizado no programa.

A maioria dos comandos que o inversor comporta está listado na parte esquerda da Figura 56. O usuário pode partir, parar, escolher o sentido de giro, usar o recurso de "JOG" para partida instantânea ao comando, habilitar por onde os comandos são ativados, reiniciar os comandos e desabilitar o dispositivo. Na parte de "*status do inversor*" mostrada à esquerda superior na Figura 56 as informações de *status* são exibidas em tempo real.

Para exemplificar, a Figura 57 mostra a tela contendo um gráfico com dados extraídos do inversor referente ao acompanhamento de um motor em tempo real pelo supervisório. Apenas algumas penas estão ativas no instante, sendo mostrado apenas as que estão ao lado dos botões em verde aceso ao lado direito superior na Figura 57.

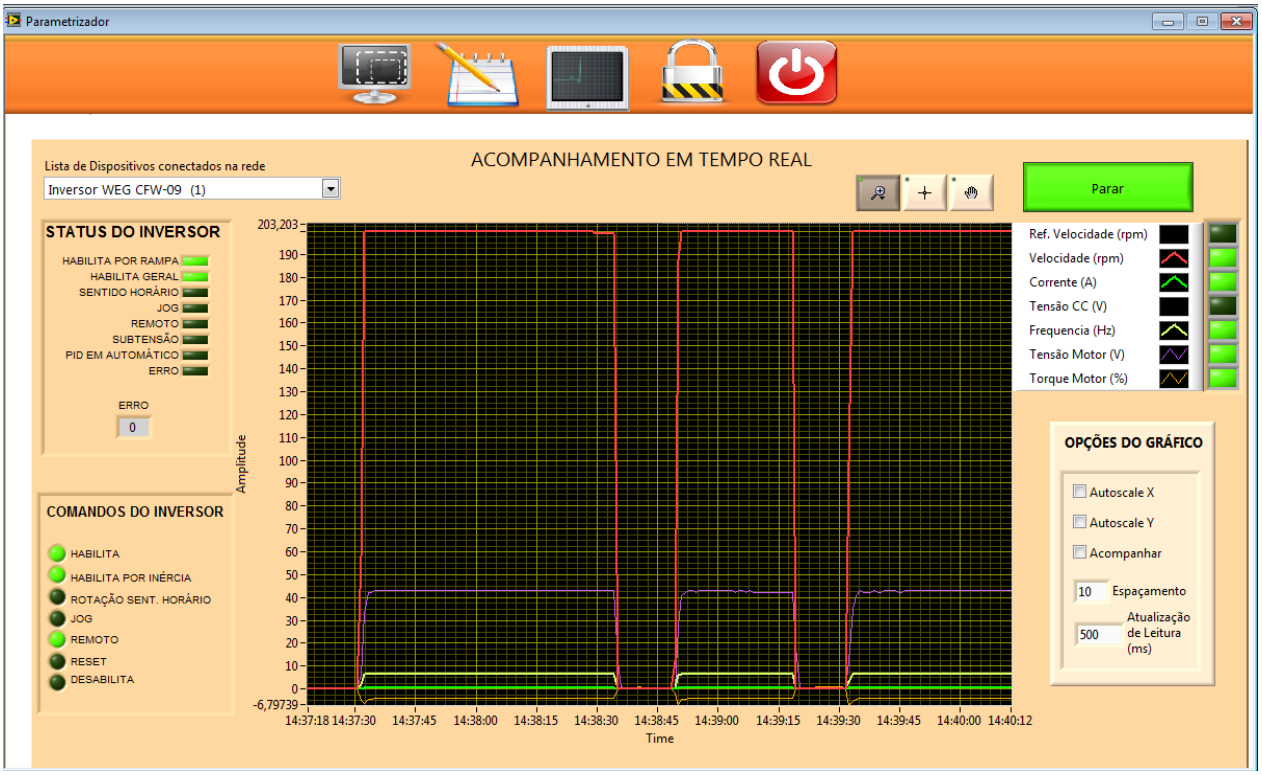


Figura 57 – Aquisição de dados

Fonte: Autoria própria

Os *bits* de estado e comando referente ao dispositivo são mostrados respectivamente a esquerda superior e inferior, estes sempre que estão acesos significam que o programa está se comunicando corretamente com

o inversor conectado. Este exemplo mostra toda a funcionalidade da parte de aquisição do programa em funcionamento, sendo parte importante também do projeto além da parametrização.

5.5 POP-UP DE LOGIN DE USUÁRIO

A tela de *login* contém os campos para que o usuário identifique-se ao programa e assim possa utilizá-lo sem restrições de acesso.

Figura 58 – Tela *login* de usuário

Fonte: Autoria própria

Cabe ao usuário inserir o *login* e a senha cadastradas no programa para obter acesso e pressionar o botão "OK" ou cancelar a operação através do botão "Cancelar".

5.6 POP-UP DE USUÁRIO LOGADO

A tela *pop-up* de usuário logado mostra algumas opções referente ao bloqueio e desbloqueio do programa, assim como permite gerenciar os usuários que tem permissão de uso do programa.

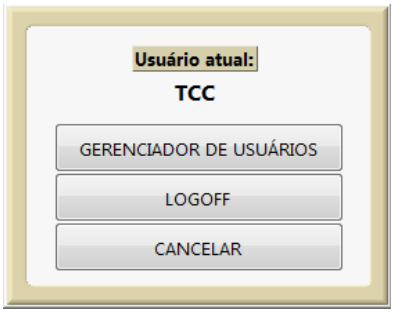


Figura 59 – Tela de usuário logado

Fonte: Autoria própria

O botão "gerenciador de usuários" permite que seja criado um novo usuário ou excluído um usuário existente, somente usuários cadastrados podem usar o programa.

O botão "*Logoff*" desconecta o usuário logado do programa, abrindo a opção para efetuar o "*Login*" novamente.

A opção "cancelar" fecha a janela e retorna a tela anterior.

6 CONCLUSÃO

O *software* desenvolvido trouxe vantagens sobre a parametrização via IHM, pois melhora o gerenciamento das informações e salva os parâmetros em arquivos organizados. Entretanto fica dependente de um computador conectado à rede ou ponto a ponto ao equipamento para utilizar o sistema. No geral o saldo de vantagens é positivo conforme demonstrado na Tabela 2.

VANTAGENS	DESVANTAGENS
✓Backup dos parâmetros	×Depende do cartão EBA
✓Rapidez na programação	×Depende de ter um PC
✓Interface amigável	×Conversores e cabeamento requeridos
✓Organização	
✓Recurso gráfico para monitoramento	
✓Não precisa ter IHM	

Tabela 2 – Vantagens e desvantagens

Há possibilidade de ser aprimorado, ampliando o número de modelos de inversores possíveis a programar e também de outros fabricantes, que não é o foco deste projeto.

A programação feita usando o LabVIEW promoveu uma maior velocidade na questão de tempo de programação e trouxe uma boa qualidade gráfica à interface de usuário graças aos seus recursos.

O objetivo geral e específicos do projeto foram alcançados e trouxe diversos conhecimentos novos quanto à programação, inúmeros problemas ocorreram durante o processo, porém foram contornados ou solucionados. Este projeto pode-se também ser usado como referência a iniciantes em programação de LabVIEW que desejem obter informações sobre como comunicar com equipamentos usando a rede ModBus, como salvar arquivos, gerenciar dados em tabelas, estruturar um *software*, no geral auxiliar na programação de outros projetos.

Algumas das maiores barreiras são as de que nem sempre o inversor aceita o parâmetro enviado, tendo que haver outro parâmetro que tem

relação com o que deseja escrever ativo, desativo ou dentro de uma faixa de valor para dar a condição de alteração de outros parâmetros. Para fazer essa verificação seria necessário estruturar a escrita e alteração de parâmetros na tabela de receita de uma maneira diferente e mais flexível, pois a estruturação do envio de informações utilizado no projeto é otimizada para ser rápida e sempre alterar todos os parâmetros do inversor, não havendo internamente no *software* lógicas que informem que o parâmetro alterado pelo usuário na tabela será ou não aceito pelo dispositivo antes mesmo de enviar as mensagens. Usando o recurso de “validação” para identificar as divergências, mas não informa o motivo delas terem acontecido cabendo ao usuário conhecer o equipamento e identificar o erro.

REFERÊNCIAS

CERVO, A. L.; BERVIAN, P. A. **Metodologia Científica**. 5ª Edição.

REGAZZI, R. D.; PEREIRA, Paulo S.; SILVA JR, M. F. **Soluções Práticas de Instrumentação e Automação – Utilizando a programação gráfica LabVIEW.** Rio de Janeiro: KWG, 2005.

TSUTIYA, M. T.; **Utilização de inversores de frequência para diminuição do consumo de energia elétrica em sistemas de bombeamento**. VI SEREA – Seminário Ibero-americano sobre sistemas de Abastecimento Urbano de água. João Pessoa. Paraíba, junho de 2006.

WERNER, L.; **Control of electrical drives**. 1926, 3rd ed.

SEGURA, R. B.; MIELLI, F. M. **O Ethernet TCP/IP Modbus**. In: Controle & Instrumentação, Outubro 2005, p. 117-120.

BOYER, S. A.; **SCADA Supervisory control and data acquisition**. 3rd ed. ISA, 2004.

LENZ, J. E.; **Flexible Manufacturing:** Benefits for the low – inventory factory. Marcel Deker, 1988.

Slack, N.; Chambers, Stuart; Johnston, Robert; **Administração da produção**. 2° Edição. Editora Atlas, 2002.

LANGER, R. A. **Curso introdutório de programação com LabVIEW.** Centro Universitário Positivo, 2006.

BITTER R.; MOHIUDDIN T.; NAWROCKI M.; ***Labview Advanced Programming Techniques.*** *2º Edition.* Editora CRC Press Taylor & Francis Group. 2006.

FRANCHI, C. M. **Inversores de Frequência: Teoria e Aplicações**. 2ª edição. Editora Érica, 2009.

WEG. **Manual do Inversor de Frequência CFW-09**, 2006.

APÊNDICE A – RELATÓRIO DE TRABALHO EM LABORATÓRIO

Novembro/2011 à Dezembro/2011 - Primeira etapa do projeto

Iniciamos as atividades referentes ao nosso projeto. Iniciamos energizando o painel e verificando se havia algum problema que impossibilitava o correto funcionamento elétrico do inversor, como falta de fusíveis por exemplo. Conseguimos energizar o inversor CFW-09 e partimos para a parametrização via HMI onde realizamos alterações em alguns parâmetros que seriam de reinicialização do inversor para as configurações de fábrica e para efetuar comunicação ModBus. Os parâmetros para carregar os padrões de fábrica foi efetuar a liberação da alteração dos parâmetros inserindo o valor 5 em P000, em seguida foi feito o *reset* da sua configuração de fábrica através do parâmetro P204 inserindo o valor 6 e feita a sua reinicialização. A configuração para a comunicação ModBus RTU com um *baudrate* de 9600 kbps e sem paridade foi feita com os seguintes parâmetros, feita a liberação de alteração dos parâmetros, e alterado os seguintes valores P308 igual a 3 para definir o endereço da rede do dispositivo, para outros dispositivos conectados na rede como ocorrerá de ter mais dispositivos cada um deverá ter um endereço diferente. O protocolo ModBus permite ter em sua rede até 255 dispositivos, cada dispositivo podendo assumir o valor de 1 até 255 porém os equipamentos WEG se limitam até o endereço 30 podendo apenas assumir endereços entre 1 e 30. E P312 igual a 1 definindo a configuração da rede como ModBus-RTU com uma velocidade de 9,6 kbps e sem paridade, futuramente quando houver mais equipamentos se comunicando na rede será melhor aumentar a velocidade da rede provavelmente para a maior velocidade dos dispositivos que seria 38,4 kbps substituindo P312 pelo valor 7.

Para a utilização do inversor em uma rede recorremos a um conversor que o laboratório disponibilizava o MIW-02, pois a interface que o inversor possuí além da interface HMI ela possibilita a comunicação RS-232 pelo conector RJ-11 onde já foi utilizada para uma aplicação anterior que aquisitava e controlava o conversor ponto a ponto com protocolo ModBus e

utilizando o LabVIEW em que este aplicativo será utilizado como base para esta nova aplicação de controle para vários dispositivos em uma rede.

Para verificar a utilização do inversor em uma rede levamos um PLC Siemens S7-200 como segundo elemento para construirmos uma rede e um aplicativo simples no LabVIEW em que realiza a leitura de variáveis em dois equipamentos em uma rede ModBus onde se pode configurar os parâmetros do protocolo como velocidade, paridade, stop *bits* e outros e também desabilitar ou habilitar a leitura do segundo equipamento na rede a interface do aplicativo é demonstrada na Figura 60. No PLC foi realizada uma programação de uma lógica que realiza o incremento de uma variável disponível para leitura na rede.

Figura 60 – Interface do aplicativo de teste LabVIEW

Fonte: Autoria própria

Notamos que o cabo de comunicação que levamos não possibilitava a conexão física entre os equipamentos, que seria o MIW-02 e o inversor CFW-09 que utilizam conectores RJ-11, visto isso tivemos que cancelar os testes previstos para o dia.

Levamos um cabo de nossa própria autoria para iniciar os testes, verificamos a pinagem do cabo, o estado dos conectores, ligamos o conversor da WEG MIW-02 para fazermos a rede RS-485 entre 2 equipamentos diferentes e o mestre, no nosso caso o supervisório em LabVIEW, e notamos a falta do driver do conversor. Conseguimos instalar o driver usando um driver genérico e o conversor funcionou perfeitamente, o testamos comunicando com o CLP S7-200 que está sendo usado como um elemento de rede, pois não temos ainda o outro dispositivo.

Com todos os equipamentos ligados notamos que a comunicação MODBUS com o CLP não funcionava mais, e notamos que quando desligava o inversor CFW-09 da rede a comunicação com o CLP era restabelecida.

A partir deste resultado levantamos a hipótese do conversor MIW-02 estar com defeito. Notamos também um erro de ligação entre os fios no nosso cabo que fabricamos e imediatamente corrigimos, mesmo assim os testes foram mal sucedidos e não obtivemos sucesso na comunicação em rede. No manual do conversor MIW-02 encontramos esquemas de ligação do cabo entre eles como mostra a Figura 61.

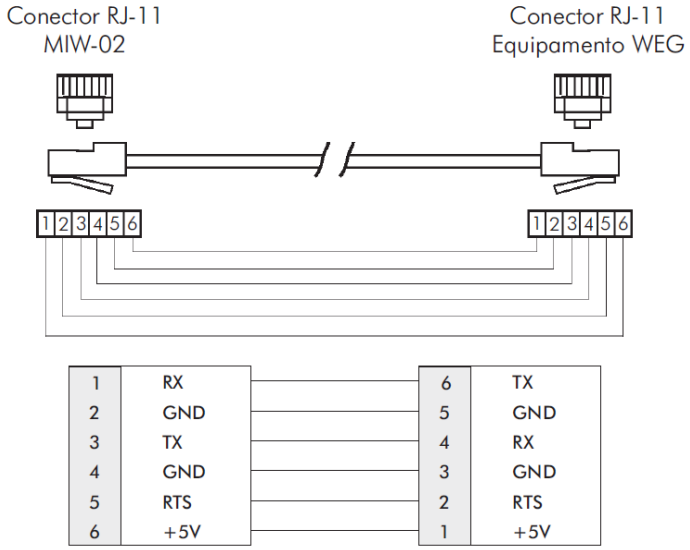


Figura 61 – Esquema de conexão

Fonte: Manual MIW-02 pág. 23

Na sala de testes montamos nossos equipamentos novamente no painel da bancada móvel da universidade que estávamos usando para

realizar os testes e ao invés de usarmos o MIW-02 que supostamente estaria estragado, usamos outro de outra bancada.

Mudamos o computador para outro notebook e ao tentar iniciar a comunicação notamos que o notebook estava com falta dos drivers internos do supervisório para interface com as portas de comunicação do computador, tentamos buscar na internet, mas não obtivemos sucesso. Não pudemos prosseguir com os testes.

Montamos novamente a estrutura física para realizarmos a rede com o MIW-02 de outra bancada, partimos para os testes de comunicação mais infelizmente novamente não conseguimos integrar o inversor CFW-09 na rede, o problema ocorria da mesma maneira que com o outro conversor MIW-02, então descartamos a possibilidade de falha no equipamento WEG.

Fizemos teste de comunicação usando interface RS-232 para assegurar que o inversor estava com os parâmetros de comunicação configurados adequadamente para a nossa rede. A comunicação serial RS-232 funcionou corretamente com o supervisório.

Fizemos diversas tentativas, analisamos minuciosamente o cabo, nos certificamos que nosso cabo estava de acordo com o manual WEG e mesmo assim não conseguimos obter sucesso no teste em rede RS-485.

Decidimos então buscar um conversor RS232-RS485 para substituir o MIW-02 por acreditarmos que este conversor teria alguma restrição para funcionar em rede.

No laboratório de testes, já com o conversor em RS485-RS232 em mãos montamos os equipamentos e primeiramente iniciamos o teste para comunicar somente o inversor CFW-09 diretamente com o supervisório usando as conversões saindo RS-232 do inversor, convertendo para RS-485 e convertendo novamente o sinal RS-485 para USB, pois o nosso conversor RS-485 somente se conecta via USB com o computador. Novamente não obtivemos sucesso na comunicação com o inversor usando o meio físico RS-485.

A solução apontada e discutida junto ao professor que está nos dando suporte então seria a de adquirir um módulo conversor da própria WEG que

disponibilize direto do inversor um sinal RS-485 para só então continuarmos nossos testes.

Realizada a comunicação com sucesso do dispositivo CFW-09, após realizar uma pesquisa dos equipamentos periféricos do inversor de frequência foi observado que existe diversas placas que dispõem da interface de comunicação serial RS-485. Foi enviado ao professor Walter Sanchez um e-mail perguntando se ele conseguiria a aquisição de uma destas placas, porém foi dito por ele que o laboratório já possui uma destas que é o modelo EAB-01 possuindo vários recursos além da comunicação serial RS-485.

Havia no laboratório um inversor com a placa EAB-01 já instalada onde foi realizada a tentativa da utilização de uma aplicação com a leitura de encoder através do CFW-09.

Realizamos a comunicação com o aplicativo de teste desenvolvido no LabVIEW confirmando a utilização do equipamento em uma rede ModBus RS-485, juntamente com outro dispositivo um PLC Siemens S7-200. Com isto iniciaremos agora o desenvolvimento do aplicativo definitivo no sistema supervisório com as funções descritas na proposta para o inversor de frequência WEG CFW-09.

Fevereiro/2012

Assim que obtivemos êxito nas tentativas de comunicação com um inversor, iniciamos o desenvolvimento do *software* que teve início em fevereiro de 2012 e começou em casa. Começamos a trabalhar e estruturar o programa e desenvolver a interface pensando principalmente no objetivo principal que é parametrizar um inversor.

A tela principal foi criada, os objetos plotados e dimensionados, os escritos do programa começaram a ser digitados e a estruturação da tela principal e do código foram basicamente definidos.

Como não temos o inversor em casa, adiantamos o que podemos fora do laboratório para minimizar o tempo de teste fora de casa, reservamos todo o tempo trabalhando no laboratório para iniciar os testes de comunicação e parametrização do o inversor.

Março/2012 - Setembro/2012

Nestes meses construímos e testamos todo o programa, compramos os conversores necessários, montamos os cabos definitivos para a comunicação e terminamos todo o desenvolvimento do programa.

Em meio a este tempo os maiores problemas enfrentados foram na questão de monitoramento do inversor, onde o conversor USB - RS232 comprado falhava na comunicação quando o inversor era ligado, a real causa é desconhecida, tivemos então que tentar com outro conversor que aparentemente minimizou as falhas e permitiu que os testes de monitoramento e envio de comandos pudessem continuar.

Outro grande problema é adequar as unidades numéricas lidas do dispositivo com a tabela de parametrização, onde por exemplo recebemos o valor "400" pela comunicação, mas na verdade ele pode representar dependendo o parâmetro "40,0". Identificar as casas decimais foi um problema complicado de contornar, tivemos que criar uma tabela de máscara como base onde contém um valor numérico de adaptação das casas decimais, o *software* usa esta tabela para saber quais parâmetros ele deve adaptar.

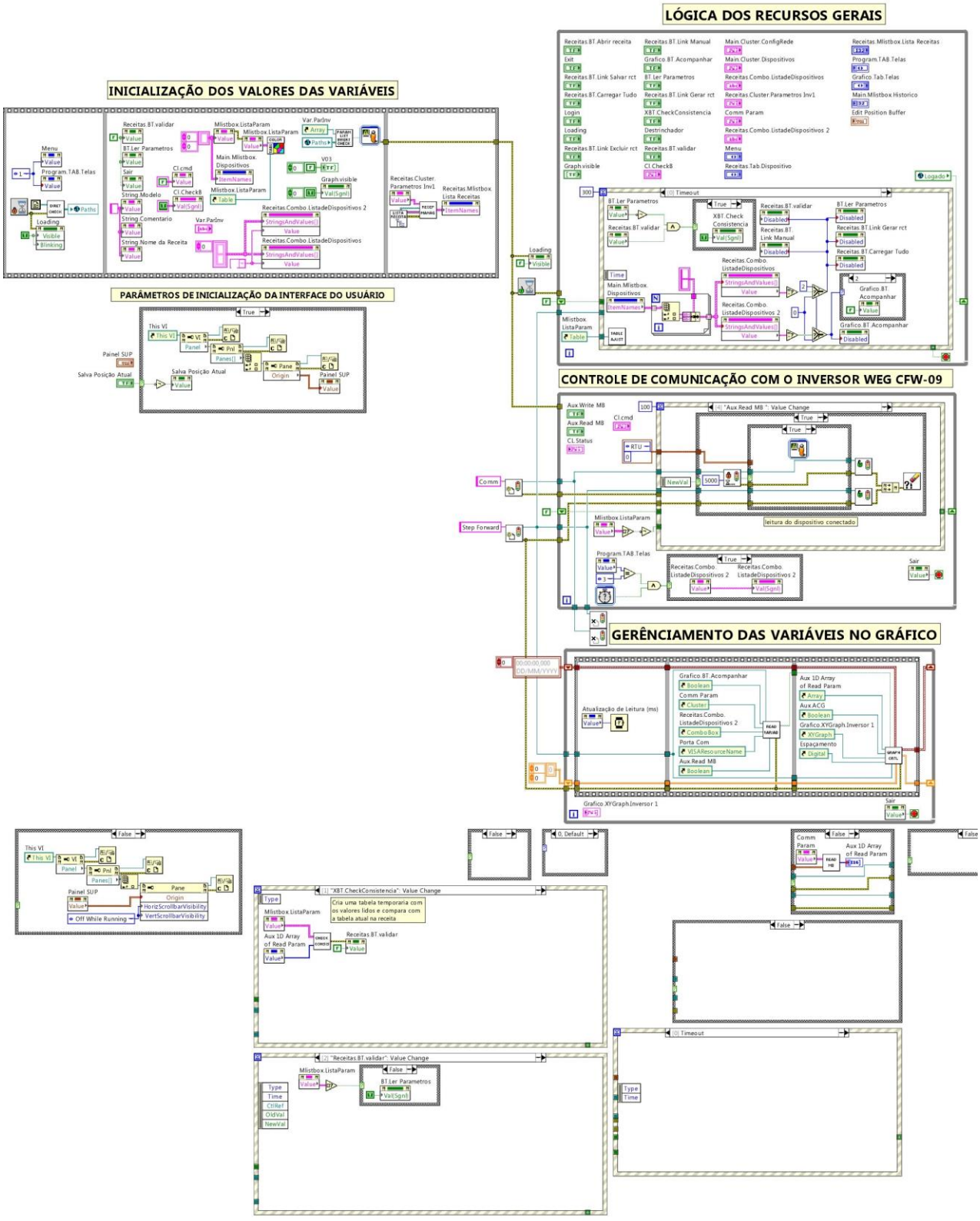
Diversos testes de parametrização foram feitos com o inversor, e graças a eles fomos moldando o programa para funcionar o melhor possível que permitisse atingirmos nosso objetivo.

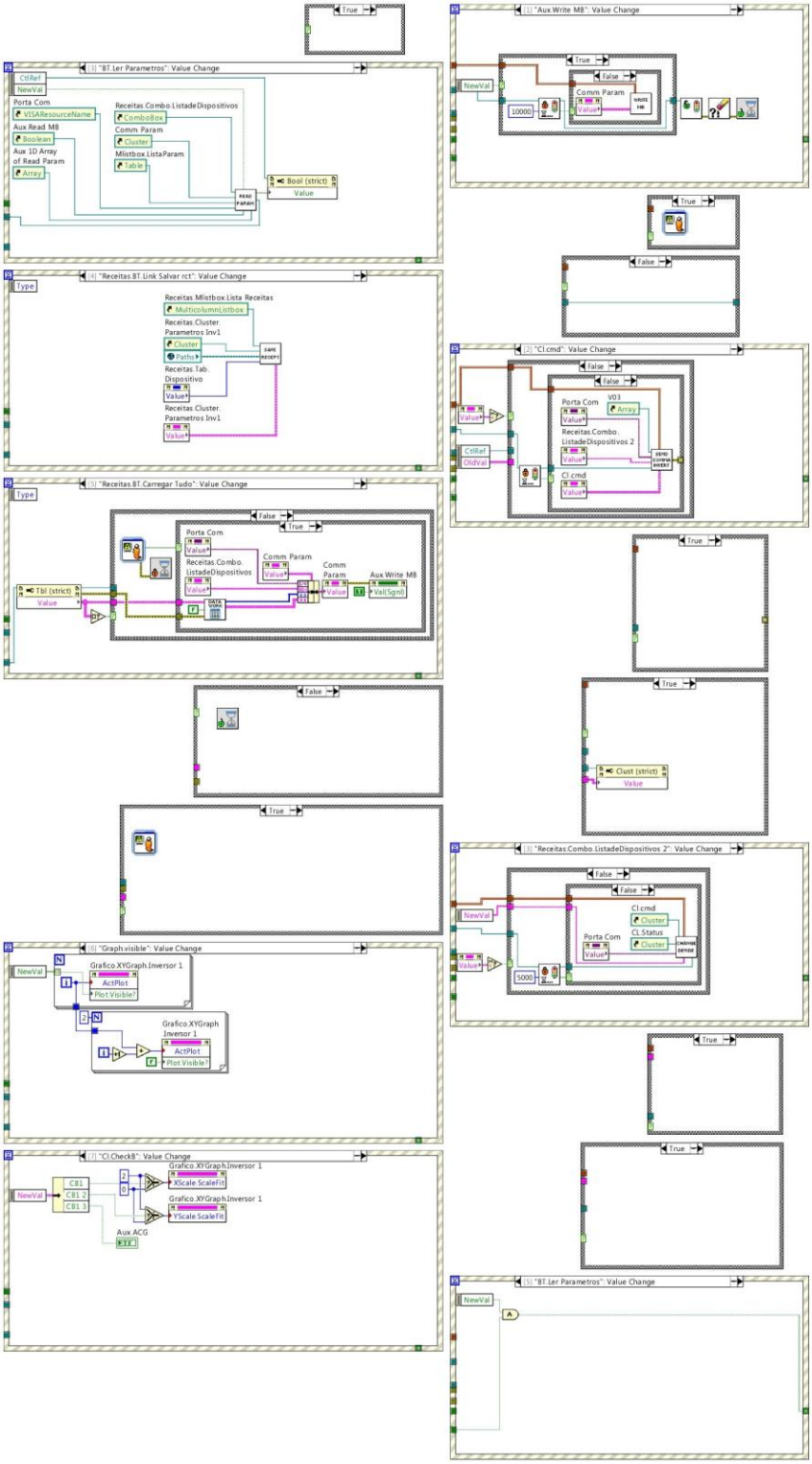
Analisamos os retornos de erro de comunicação do inversor e constatamos que quando o inversor recusa um parâmetro o mesmo não emite erro, sendo um problema, pois não poderíamos analisar em tempo real se o parâmetro enviado na rede foi aceito ou recusado pelo inversor. Contornamos o problema criando então um recurso chamado de "validação dos parâmetros" de modo que este recurso lê os parâmetros do inversor e compara com os parâmetros escritos na tabela (os parâmetros desejados), e assim informa se estão iguais ou diferentes.

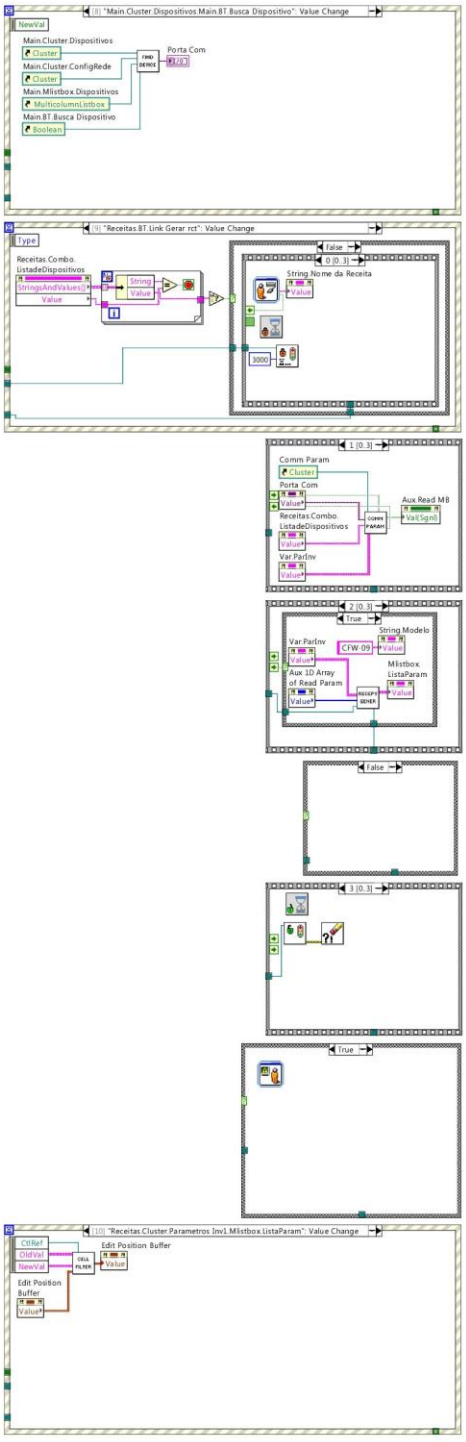
A pedidos criamos dois espaços na área de monitoramento em tempo real do inversor para enviar comandos e analisar o estado do inversor em

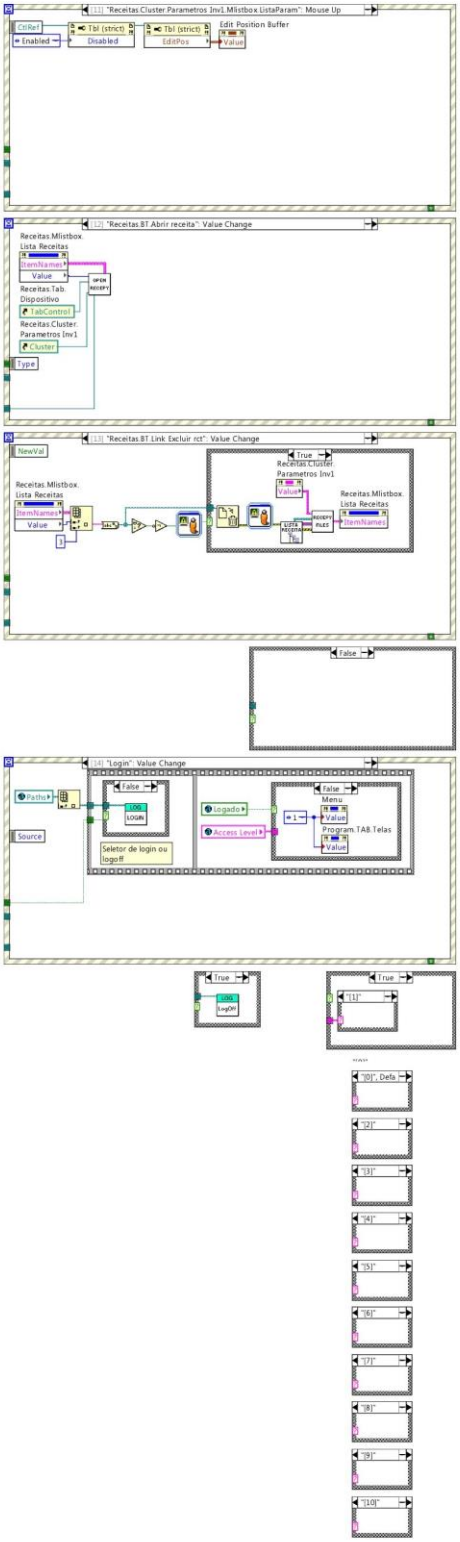
tempo real, controlar o inversor não é o foco do projeto mas mesmo assim criamos este recurso para deixar o programa mais completo.

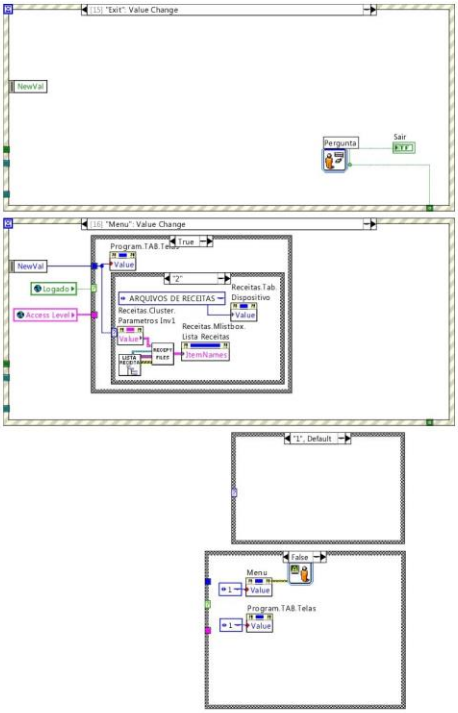
APÊNDICE B – PROGRAMAÇÃO LABVIEW




[3] "BT.Ler Parametros": Value Change
[1] "Aux.Write MB": Value Change
[4] "Receitas.BT.Link Salvar rct": Value Change
[2] "CI.cmd": Value Change
[5] "Receitas.BT.Carregar Tudo": Value Change
[0] "Receitas.Combo.ListadeDispositivos 2": Value Change
[6] "Graph.visible": Value Change
[7] "CI.Check8": Value Change
[5] "BT.Ler Parametros": Value Change


[0] "Main.Cluster.Dispositivos.Main.BT.Busca Dispositivo": Value Change
NewVal
Main.Cluster.Dispositivos
Cluster
Main.Cluster.ConfigRede
Cluster
Main.MIistbox.Dispositivos
MulticolumnListbox
Main.BT.Busca Dispositivo
Boolean
Porta Com
[1] "Receitas.BT.Link Gerar rct": Value Change
Type
Receitas.Combo.
ListadeDispositivos
StringsAndValues
Value
String
Value
False
0 [0..3]
String.Nome da Receita
Value
3000
1 [0..3]
Comm Param
Cluster
Porta Com
Value
Receitas.Combo.
ListadeDispositivos
Value
Var.ParInv
Value
Aux Read MB
Val(Sgnl)
2 [0..3]
True
Var.ParInv
Value
CFW-09
String.Modelo
Value
Aux 1D Array
of Read Param
Value
Mlistbox
ListaParam
Value
False
3 [0..3]
True
[10] "Receitas.Cluster.Parametros Inv1.MIistbox.ListaParam": Value Change
CtlRef
OldVal
NewVal
Edit Position Buffer
Value
Edit Position
Buffer
Value


[11] "Receitas.Cluster.Parametros Inv1.Mlistbox.ListaParam": Mouse Up
CtlRef
Enabled
Tbl (strict)
Disabled
Tbl (strict)
EditPos
Edit Position Buffer
Value
[12] "Receitas.BT.Abrir receita": Value Change
Receitas.Mlistbox. Lista Receitas
ItemNames
Value
Receitas.Tab. Dispositivo
TabControl
Receitas.Cluster. Parametros Inv1
Cluster
Type
OPEN RECIPE
[13] "Receitas.BT.Link Excluir rct": Value Change
NewVal
Receitas.Mlistbox. Lista Receitas
ItemNames
Value
True
Receitas.Cluster. Parametros Inv1
Value
Receitas.Mlistbox. Lista Receitas
ItemNames
RECIPE FILES
LISTA RECEITAS
False
[14] "Login": Value Change
Paths
Source
False
LOGIN
Seletor de login ou logoff
Logado
Access Level
False
Menu
Value
1
Program.TAB.Telas
Value
True
LOGIN
True
[11]"
"[0]", Defa
"[2]"
"[3]"
"[4]"
"[5]"
"[6]"
"[7]"
"[8]"
"[9]"
"[10]"


[15] "Exit": Value Change
NewVal
Pergunta
Sair
TRUE
[10] "Menu": Value Change
Program.TAB.Telas
True
Value
NewVal
Logado
Access Level
"2"
ARQUIVOS DE RECEITAS
Receitas.Cluster.
Parametros Inv1
Value
Receitas.Tab
Dispositivo
Value
Receitas.Mlistbox.
Lista Receitas
ItemNames
"1", Default
False
Menu
Value
1
Program.TAB.Telas
Value
1

ANEXO A - INVERSOR DE FREQUÊNCIA CFW-09

"O inversor de frequência CFW-09 é um produto de alta performance o qual permite o controle de velocidade e torque de motores de indução trifásicos. A característica central deste produto é a tecnologia “Vectrue”, a qual apresenta as seguintes vantagens:

- Controle escalar (V/F), VVW ou controle vetorial programáveis no mesmo produto;
- O controle vetorial pode ser programado como “sensorless” (o que significa motores padrões, sem necessidade de encoder) ou como controle vetorial com encoder no motor;
- O controle vetorial sensorless permite alto torque e rapidez na resposta, mesmo em velocidades muito baixas ou na partida;
- Função “Frenagem ótima” para o controle vetorial, permitindo a frenagem controlada do motor sem usar resistor com chopper de frenagem;
- Função “Auto-Ajuste” para o controle vetorial, permitindo o ajuste automático dos reguladores e parâmetros de controle a partir da identificação (também automática) dos parâmetros do motor e da carga utilizados.".

Figura 62 – Inversor de frequência CFW-09

Fonte: Manual CFW-09 WEG