UNIVERSIDADE DO SAGRADO CORAÇÃO

ÉDERSON DE SOUZA LEAL

AUTOMAÇÃO MODULAR

BAURU

2016

ÉDERSON DE SOUZA LEAL

AUTOMAÇÃO MODULAR

Trabalho de conclusão de curso, apresentado ao Centro de Ciências Exatas e Sociais Aplicadas, da Universidade do Sagrado Coração, como parte da obtenção do título de Bacharel em Engenharia de Computação, sob orientação do Profº Me. Alexander da Silva Maranho.

BAURU

2016

ÉDERSON DE SOUZA LEAL

AUTOMAÇÃO MODULAR

Trabalho de conclusão de curso, apresentado ao Centro de Ciências Exatas e Sociais Aplicadas, da Universidade do Sagrado Coração, como parte da obtenção do título de Bacharel em Engenharia de Computação, sob orientação do Profº Me. Alexander da Silva Maranho

BANCA EXAMINADORA

_____

Profº Me. Alexander da Silva Maranho
Universidade do Sagrado Coração

_____

Profº Me. Márcio Henrique Castilho Cardin
Universidade do Sagrado Coração

_____

Profº Dr. Sérgio Koodi Kinoshita
Universidade do Sagrado Coração

BAURU, 06 de dezembro de 2016

‘Leal, Ederson de Souza de

L435a

Automação modular / Ederson de Souza Leal. -- 2017.

81f. : il.

Orientador: Prof. M.e Alexander da Silva Maranho.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Automação Industrial. 2. Ethernet. 3. CLP. 4. Codesys. 5. Sistema Modular. I. Maranho, Alexander da Silva . II. Título.

Dedico esta monografia aos meus pais, Casimiro Borges Leal e Joana de Souza Ribeiro Leal, que mesmo longe foram a base para tudo na minha vida.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus que sempre me orientou nos momentos mais difíceis da minha vida, me deu forças e me amparou.

Aos meus pais pela construção dos meus valores, a força necessária em momentos de cansaço, a calma em dias de desespero e o carinho quando a saudade batia em minha porta.

Aos irmãos desta jornada, uma segunda família que esteve presente nos piores e melhores momentos.

Para as pessoas que compartilharam conhecimento comigo, Professores e Mestres para uma vida.

A Gabriela Valotti e Fernanda Pontes que me ajudam a ser menos chato.

Todos que direta ou indiretamente fizeram parte de minha formação e torceram pelo meu sucesso, o meu sincero muito obrigado.

“É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota.” (Theodore Roosevelt)

RESUMO

Custos em plantas Industriais podem ser uma preocupação, fatores como processos ineficientes, projetos caros e manutenção constante podem encarecer o produto final, por isso as Indústrias vivem em constante mudança para oferecer maior flexibilidade na produção, reduzindo tempo e aumentando qualidade. Neste ponto vê-se a automação de processos acompanhando de forma paralela ao mercado, oferecendo hardware para aplicações, mas não fornecendo soluções aos processos. As plantas Industriais têm similaridades, como partidas de motores, sistemas de controles de níveis, controles de temperaturas, controles de pressão e afins, que podem ser parametrizadas pelo próprio usuário final dispensando assim a mão de obra cara e horas de engenharia de um programador com experiência em programação de CLPs. O sistema proposto é um modelo modular, onde a programação fica em baixo nível e o usuário não precisa de conhecimento de programação para alterar a sua Indústria, exigindo apenas conhecimentos de processos e usando uma interface simples para compor um sistema de automação.

Palavras chave: Automação Industrial. Ethernet. CLP. Codesys. Sistema Modular. Supervisório. IHM. Arduino. CLP. Vijeo. Sistemas Embarcados.

ABSTRACT

Costs in industrial plants can be a concern, factors such as inefficient processes, expensive projects and maintenance can endear the final product, so the Industries live constantly changing to provide greater flexibility in production, reducing time and increasing quality. At this point one can see the process automation tracking parallel to the market, offering hardware to applications, but not providing solutions to processes. The industrial plants have similarities, such as motor starters, level control systems, temperature controls, pressure controls, and the like, which can be parameterized by way own end user dispensing the hand of man workmanship and engineering hours a programmer experience in PLC programming. The proposed system is a modular model, where programming is at low level and the user does not need programming knowledge to change your industry, requiring only knowledge of processes and using a simple interface to compose an automation system.

Keywords: Industrial Automation. Ethernet. CLP. CoDeSys. Modular system. Supervisory. IHM. Arduino. Vijeo. Embedded systems.

LISTA DE ILUSTRAÇÕES

LISTA ABREVIATURAS E SIGLAS

APT: *Automatically Programmed Tools* (Ferramentas de Programação Automática)

CFC: *Continuos Function Chart* (Gráfico de Função Continua)

CLP: Controlador logico programável

CPU: *Central Processing Unit* (Unidade Central de Processamento)

ERP: *Enterprise Resource Planning* (Planejamento dos Recursos Empresariais)

FB: *Function Block* (Bloco de Função)

FBD: *Function Block Diagram* (Diagrama Bloco de Função)

GB: Gigabytes

GND: *Ground* (Terra)

I/O: *Input/Output* (Entrada/Saída)

IDE: *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)

IEC: *International Electrotechnical Commission* (Comissão Eletrotécnica Internacional)

IHM: Interface Homem Máquina

IP: Internet Protocol (Protocolo de Internet)

ISA: *International Society of Automation* (Sociedade Internacional de Automação)

OLE*: Object Linking and Embedding (*Incorporação de Objetos Vinculados*)*

OPC: *OLE for Process Control (*OLE para Controle de Processos)

PID: Proporcional Integral Derivativa

PWM: *Pulse Width Modulation* (Modulação de Largura de Pulso)

SCADA: *Supervisory Control and Data Acquisition* (Sistema de Controle e Aquisição de dados)

SDCD: Sistema digital de Controle Distribuído

ST: *Structured Text* (Texto Estruturado)

TCP: *Tansmission Control Protocol* (Protocolo de Controle de Transmissão)

LISTA SIMBOLOS

A: Ampère – Unidade de corrente elétrica

b: *bit*

B: *Byte*

Vcc: Tensão em Corrente Contínua

V: Volt – Unidade de tensão elétrica

W: Watt – Unidade de potência

SUMÁRIO

1 INTRODUÇÃO

Os Processos Industriais estão em constantes alterações que podem ocorrer para suprir a necessidades ou diminuir custos, neste contexto podemos avaliar que as manutenções industriais têm um impacto forte no preço final e na competitividade entre empresas.

Alterações no sistema que podem ser caracterizadas como: inclusão de equipamentos, ajustes nos controles, alteração do sistema, inclusão de telas e relatório e outros. Para tal necessita-se de profissionais qualificados e tempo de parada, um tempo em que a planta precisa ser parada para que as alterações possam entrar em vigor.

A manutenção deve ser encarada como uma função estratégica na obtenção dos resultados da organização e deve estar direcionada ao suporte do gerenciamento e à solução de problemas apresentados na produção, lançando a empresa em patamares competitivos de qualidade e produtividade (Kardec & Nascif, 2001).

O conceito de "Engenharia" se aplica em projetos completos, empresas oferecem soluções *TurnKey*[1], para uma planta industrial é muito mais vantajoso que uma única empresa forneça o hardware e software a ela, evitando diversas empresas alterando sistemas e a planta física (elétrica, automação, mecânica).

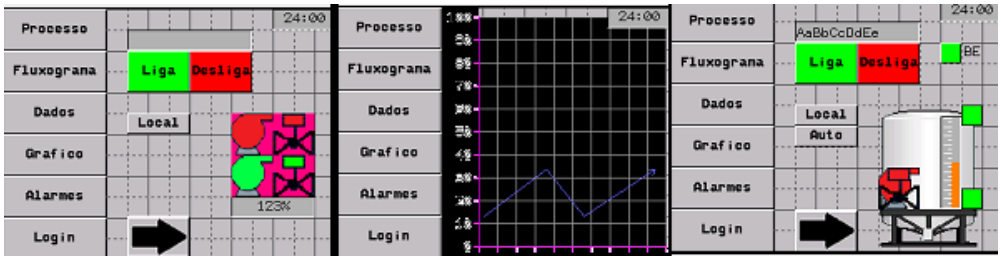
De vista aos problemas mais encontrados, existem casos de softwares que não integram a outros, a requisição constante do programador, a falta de parametrização e ajustes de controles, e um falta de conhecimento do processo, assim os projetos de engenharia demandam uma carga horaria altíssima. O TI (Tecnologia de Informação) já está sendo obrigado a ser embutido dentro da Automação, com arquiteturas que usam dos protocolos TCP/IP (*Tansmission Control Protocol* – Protocolo de Controle de Transmissão) e Switchs que trafegam tantas informações de ERPs (*Enterprise Resource Planning* - Planejamento dos Recursos Empresariais) quanto de controles de processos. Sistemas modulares podem preencher a lacuna que existe entre sistemas que já são programados por meio de bibliotecas, assim integrando em perfeita harmonia telas de supervisão e telas de IHM (Interface Homem Máquina).

A modularização de sistemas pode conter um hardware dedicado, mas o conceito da programação pode ser aplicado tanto a plantas já existentes quanto a plantas que ainda estão em fase de desenvolvimento, já que a tela é definida junto com o processo. Empresas de médio e

[1] TurnKey: Projetos feitos pela mesma empresa em diversas áreas de engenharias

pequeno porte poderão utilizar estes sistemas para automatizar sistemas pequenos sem ter o custo alto de implementação de engenharia, possuindo um sistema simples que lhe proporcione gráficos de produção, acionamentos remotos, controles automáticos e acessos tanto para Web/Mobile e Desktop.

Figura 1 - Exemplo tela simplificada



Fonte: Elaborada pelo Autor

Os processos automatizados tendem a ser divididos em micro, médio e grande porte, sendo a básica diferença entre eles a capacidade de equipamentos, integração com outros sistemas e velocidade de execução/processamento (Sistemas *Motion*). Porem mesmo nos grandes sistemas existe o problema pós implementação, quando surge a necessidade de trocar ou inserir motores, válvulas, sistemas simples de automação. As empresas cobram pela implementação e hardware adicionais, o custo cobrado em campo (visita técnica) pode variar conforme a disponibilidade técnica das empresas, épocas de paradas programadas. Com o sistema proposto o cliente ou a área de manutenção poderia trocar, inserir e remover os módulos que compõem a automação de uma maneira rápida e fácil, sem a necessidade de implementar, programar e compilar a aplicação.

Sistemas de automação podem alcançar um patamar de programação elevado, geralmente não se utiliza todo o potencial da CPU (*Central Processing Unit*) por motivos de velocidade de processamento. Porem se um programa com multiopções for executado na maneira correta e de forma que delibere rotinas não executadas, o processamento se manteria e apenas o tamanho do programa (em bytes) seria acrescido. Os custos da automação em plataformas industriais tendem a não manter um padrão lógico, isso por motivos de falta de conhecimento de normas vigentes para Tags[2] conforme a ISA (*International Society of Automation),* normas para programação como a IEC 61131-3 (*International Electrotechnical Commission*) citada mais aprofundada neste trablho, normas para execução dos softwares como gerenciamento de processamento de rotinas. Essa *plataforma* inicia com processos simples,

[2] Tags: Indentificador de equipamentos ou produtos

controles de tanques, aquecedores (Controle), partidas de motores e outros, com o tempo e o uso constante do software, bibliotecas de processos podem ser adicionadas por profissionais de automação. Indiferente a língua ou modo de programar, a biblioteca utiliza um hardware padrão, por isso um manual complementar da biblioteca deve ser feito, para instruir como usar, parametrizar, e instalar eletricamente o modulo no sistema. Esse conceito Open source poderá, com o tempo, ganhar diversas áreas industriais, comerciais e residenciais, mantendo sempre a mesma ideia, o desenvolvedor adiciona a biblioteca no sistema que é integrado de acordo com a necessidade especifica do cliente. Não existe tal plataforma totalmente livre de hardware e com software implementável para a automação, apenas plataformas privadas como os sistema PL7(Siemens), TIA (Siemens), RSLogix/FactoryTalk(Rockwell Automation), SoMachine/Vijeo(Schneider), CentumVP(Yokogawa. O alinhamento com uma biblioteca unificada em um padão de programação e voltada totalmente a controle de processos similares na indústria como exemplo controles de temperatura, controles de pressão, controles de níveis, partidas de motores, e uma infinidade de processos usados hoje que não mudam sua essência logica, mudam potência, quantidade, equipamentos modelos, porem o processo continua o mesmo, a atuação neste nicho visa disponibilizar e facilitar a vida do programador, tirando a carga de criar bibliotecas repetidas ou programas que outro progamador já executou, padronizar uma programação, escrever uma documentação, ajudando principalmente na disseminação de conhecimento uma vez que disponibilizada e testada essa rotina logica pode ser implementada por outro programador usando o padrão CodeSys, ou reescrevendo a logica em *Ladder*.

1.1 OBJETIVO GERAL

Elaborar um sistema modular pré-programado, com biblioteca de processos OpenSource e hardware dedicados.

1.2 OBJETIVOS ESPECÍFICOS

- Elaborar hardware livre em Arduino;
- Usar do padrão Codesys para programar o Controlador Logico Programavel;
- Usar a plataforma de prototipagem Arduino como módulos de IO (Input/Output – Entrada/Saída);
- Desenvolver um sistema de Supervisão usando o *Vijeo Designer;*

- Desenvolver base de dados em OPC (*OLE (Object Linking and Embedding) for Process Control*) e publicar em WebOS (*Operation System Web);*
- Desenvolver bibliotecas de processos, acionamentos, registros e gráficos;
- Desenvolver telas para que o Usuário possa parametrizar e definir sua automação;
- Desenvolver rede interligando os equipamentos;
- Validar o protótipo.

2 REVISÃO BIBLIOGRÁFICA

2.1 AUTOMAÇÃO

Automação deriva do termo em latim Automatus, que significa “mover-se por si”, este termo foi usado para definir sistemas que possuem movimentação ou mesmo respostas programadas, estes sistemas existem – não necessariamente ao mesmo tempo – em três áreas: Pneumática, Elétrica e Computacional, atuando para compor sistemas “inteligentes”, por exemplo, a elétrica utiliza de componentes como contatores, reles, botões, motores, chaves, sensores, para criar intertravamentos[3]. Pneumática utiliza pistões, válvulas seletores de passagem de ar, reguladores para criar movimentação na máquina. E a área computacional utiliza o Software para tomar decisões, coletar dados, gerenciar relatórios, definir produções e atuar sistemas, logo a junção destas se torna necessário para criar um sistema autômato completo. (Dorf, R. & Robert, 2001).

A Automação Industrial surgiu para preencher lacunas como intertravamentos complexos, controles analógicos, registros e outros onde a elétrica não conseguia contemplar e também para resolver problemas que surgiram devido à Revolução Industrial, a qual deu início a cadeia evolutiva da área de Automação, na década de 40 os processos industriais precisavam se tornar mais eficientes, pois as Indústrias necessitavam produzir quantidades inalcançáveis de produtos e naquele momento eram utilizados métodos artesanais, o que exigia longas horas de trabalho, implicando em problemas sindicais, assim a situação levou a criar métodos para mantê-las funcionando 24 horas, com alta produção e com menos funcionários.

Em 1948, John T. Parsons utilizou métodos de cartões perfurados que serviam para controlar movimentos de uma máquina. Este método foi apresentado para a Força Aérea, que investiu em outros projetos do Laboratório de Servomecanismos do Instituto Tecnológico de Massachusetts (MIT). Este conceito foi utilizado em uma fresadora de três eixos com controle de posicionamento, desencadeando uma onda em que todas as fabricantes de fresas (Maquinas de Usinagem), na época, começaram a investir nesta área. O MIT criou o APT (Ferramentas de Programação Automática), uma linguagem padronizada para as máquinas da época. A General Motors instalou robôs em sua linha de produção para a soldagem de carrocerias, a eficiência e

[3] Intertravamentos: Endente-se como sequenciamento lógico de ações para acionamentos.

ângulos proporcionados por estes robôs era algo inalcançável por humanos, os processos de automação industrial continuaram a evoluir desde então. (Comat Releco World Relays**,** 2013)

2.2 CONTROLADORES

2.2.1 MICROPROCESSADORES

O primeiro microprocessador foi desenvolvido pela Intel em 1971 para atender uma empresa Japonesa que necessitava de um CI (Circuito Integrado) específico para suas atividades. Após este, a própria Intel viu seu grande potencial e continuou sua fabricação iniciando assim a rampa ascendente desta tecnologia.

Conhecido como processador ou CPU, construído num único Circuito Integrado a unidade lógica/aritmética e registradores (Acumulador e outros) assim tornando-se o cérebro lógico das máquinas, executando funções programadas, aceitando dados digitais como entrada, compilando e fornecendo dados como saída. Resumidamente a CPU tem como finalidade unificar o sistema, controlar as funções realizadas por cada unidade funcional, e a execução de todos os programas do sistema, que deverão estar armazenados na memória principal.

2.2.2 CLP

O CLP (Controlador Lógico programável) teve sua origem em 1968 dentro da General Motors como um protótipo dedicado com mais vantagens em relação aos sistemas usados na época. O protótipo surgiu quando alterações em painéis elétricos se tornaram custosas e pouco viáveis. O protótipo era simples porem usual, tinha um processador onde os programadores desenvolviam a lógica a ser executada e o espelhamento das entradas e saídas.

O uso de CLPs na Indústria a partir de então foi aumentando e diversas empresas criaram seus controladores e disponibilizaram no mercado, tais como Siemens, Allen Bradley, GE (General Eletric), Schneider e outras muitas. A Indústria ganhou um aliado na busca de confiabilidade e interfaces amigáveis, com um funcionamento que segue a mesma base teórica de um computador, uma placa prototipada com processador e entradas e saídas de sinais, módulos de comunicação e parte programável separada do SO (*Operation System*) este mesmo que é dedicado e inalterável (exeto pelo fabricante) gerando uma confiabilidade, pois evita acessos de terceiros que possam infectar o sistema com virus, otimizando seu processamento pois o SO é feito especificamente para o hardware.

Seu processamento é cíclico, assim executando todo o software e espelhando assim as suas saídas, conforme a Figura 1, onde ocorre uma leitura das entradas, um processamento e espelho da saída.

Figura 2 - Processamento CLP

Fonte: IEE UERJ (2016)

A programação de um CLP, desde seu surgimento, segue padrões pré-estabelecidos por meio de operadores lógicos binários, instruções matemáticas e funções. Estas instruções e funções serão inseridas na memória do CLP através de um console (PC/Display) e poderão ser alteradas.

O programa é introduzido via software nos endereços de memória do CLP, contendo cada instrução os parâmetros de definição, função ou operador e todos os parâmetros requeridos por essa instrução. Toda CPU de um CLP tem um limite de memórias para serem utilizadas, e cada fabricante utiliza de métodos diferentes para proteger o software escrito sendo está uma das grandes vantagens do uso do CLP.

O processador inicia o programa após ter lido o estado de todas as entradas e ter guardado toda a informação em memória, a execução é feita da primeira até a última linha após a executada o espelhamento das memorias é feito nas saídas digitais, este termo é chamado de *SCAN*.

2.2.3 LINGUAGENS DE PROGRAMAÇÃO

Entre as linguagens mais usadas por programadores estão a *Ladder*, Lista de instruções e *GRAFCET*, porém existem outras como FB (*Function Block*), CFC (*Continuos Function Chart*), FBD (*Function Block Diagram*) e ST (*Structured Text*). O uso destas seguem as normas da ISA (*International Society of Automation)* e IEC 61131-3 (*International Electrotechnical Commission*), logo a migração entre fabricantes segue a mesma base teórica, porém o esquema

das memórias e sua codificação mudam (programa compilado), assim cada fabricante tem seu leque de possibilidades, mas deve seguir a norma para que seus produtos possam ser considerados como um CLP.

A linguagem *Ladder* é composta por elementos visuais usados para desenvolver o programa, usualmente ele é comparado a esquemas elétricos conforme a Figura 2.

Figura 3 - Exemplo Ladder

Fonte: Elaborada pelo Autor

A função do *Ladder* é controlar saídas logicas baseadas em lógicas, utilizando *rungs* ou linhas/Networks onde são inseridos contatos, temporizadores, ou funções matemáticas e booleanas, executando sempre da esquerda para a direita e de cima para baixo.

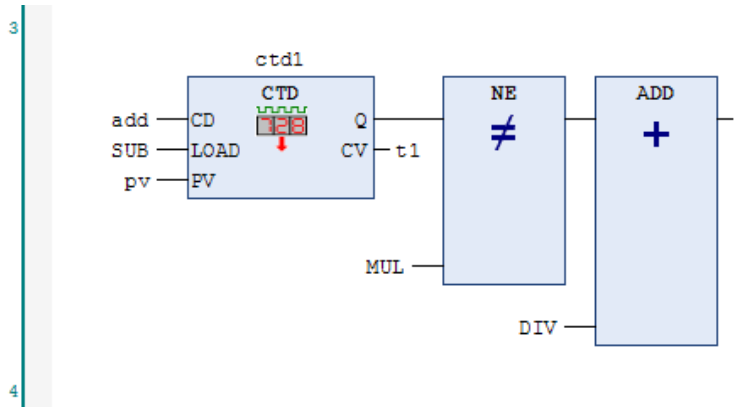
Em resumo, dentre os operadores lógicos mais usados podemos listar:

- Contato NA (Normalmente Aberto)
- Contato NF (Normalmente Fechado)
- Bobina Comum
- Bobina de Set
- Bobina de Reset
- Funções AND, OR, NOT
- Temporizadores
- Contatos ou funções de Pulso

A lógica FBD ou FB segue a teoria de espaço de programação aberto, onde blocos funcionais são linkados para interagirem entre sim, conforme a Figura 3 ilustra. Esta

esquematização de programação tem muitas aplicações em funções analógicas ou em programas com orientação a funções.

Figura 4 - Exemplo FBD



Fonte: Elaborada pelo Autor

A lógica CFC se assemelha a fluxogramas, usada em logicas continuas e de processos onde um fluxograma de ações é definido, assim como a Figura 4 ilustra a semelhança entre um fluxograma e a programação.

2.2.4 CODESYS

CoDeSys é uma plataforma independente de programação de CLP, seguindo a norma IEC 61131-3, contém todas as linguagens de programação padrão. A estrutura baseada em componentes torna possível uma configuração cliente específica e a extensão da interface do usuário. (Oliveira, Marcos, 2011)

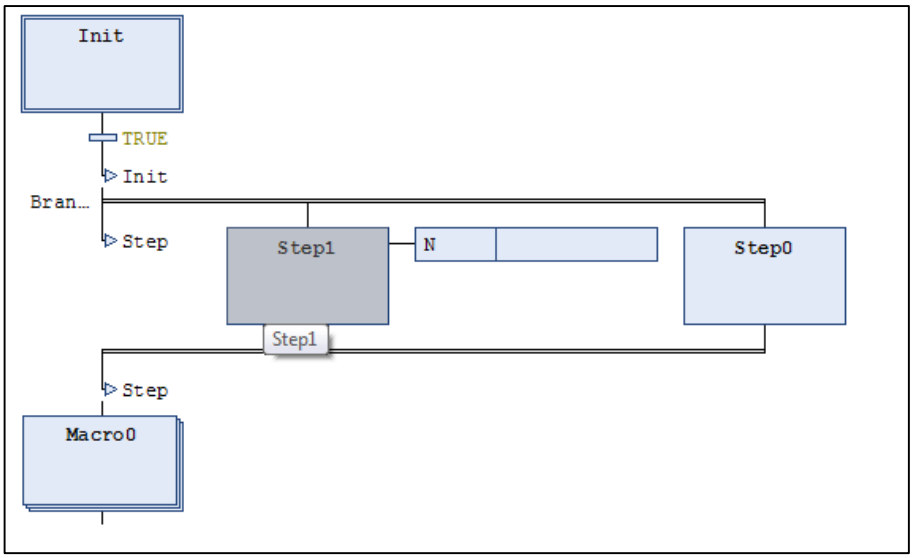
A CodeSys é uma empresa que tornou possível a unificação da programação de diversos fabricantes no mesmo ambiente de trabalho e graças a esta grande compatibilidade, seu nome ganhou grande notoriedade no mercado de trabalho. A ferramenta está disponível para engenharia (desenvolvimento), Visualização, *Field* (Campo), e diagnostico.

Segundo a própria CodeSys, marcas famosas já utilizam dos seus sistemas: ABB Automation Products GmbH, Advantech Europe B.V., Altus Sistemas de Informática S.A., Bosch Rexroth AG, Eaton Automation AG, EXOR Deutschland GmbH, Festo AG & Co. KG, IFM electronic gmbh, Mitsubishi Electric Pvt. Ltd., Schneider Automation S.A.S.

A migração para a sua plataforma vem em um Mercado onde os CLPS abração uma classe de controles menores, onde a necessidade de um hardware robusto e de alta velocidade

se torna muito menor. O desenvolvimento de um SO se torna custoso e a aplicação sobre o Codesys uma vantagem financeira e segura para o setor.

Figura 5 - Exemplo de CFC



Fonte: Elaborada pelo Autor

2.3 INTERFACES

Interfaces são todos os tipos de meios possíveis para comunicação entre o "operador" e a máquina ou sistema, conhecidas como *HMI (Human Machine Interface)*, ou supervisório.

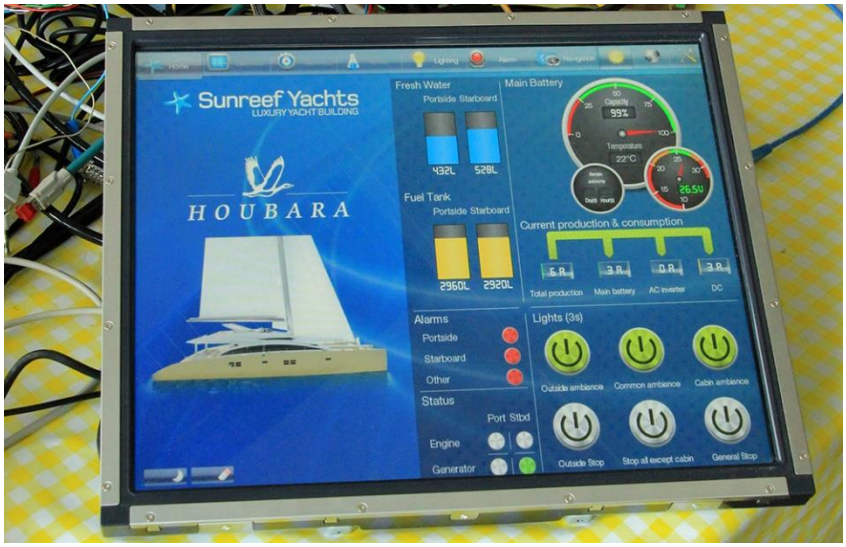
Na área Industrial os sistemas possuem o nome de SCADA (*Supervisory Control and Data Acquisition* – Sistema de Supervisão e Controle*)*, os quais possuem comunicações diretas com o CLP e podem mostrar em tela as variáveis e controlar o processo, alarmes e informações para controle da máquina conforme Figura 4. Para tais comunicações o PC ou IHM podem estar equipados com placas dedicadas ou drives para transcorrer o protocolo industrial e decodificar a informação. A arquitetura deste supervisório pode ser mista, como Cliente-Servidor, Stand-Alone[4], ou ter Historiador[5], Supervisório e gerenciador de ativos na mesma Máquina.

Gerenciadores de ativo são sistemas capazes de passar pelo CLP e chegar até o "Campo", parametrizando equipamentos, essas parametrizações de campo servem para definir escalas de processo, limites para alarme e configurações de endereçamento. IHMs podem ter serviços de servidor para gerenciar produção, ter receitas para facilitar a vida do operador.

[4] Stand-Alone: Servidor isolado, sem conexões externas
[5] Historiador: Servidor/banco de dados dedicado onde são guardados as informações do processo.

Algumas empresas de software de automação estão começando a adotar a acessibilidade *Web/Mobile* em seus Sistemas.

Figura 6 - Exemplo de Interface



Fonte: Annecyelectronique, 2016

2.4 REDES IP

A Automação usa o meio físico Ethernet para encapsular protocolos industriais dentro do TCP, como exemplo ProfiNet (Profibus), TCP/IP Modbus (Modbus), DeviceNet (ControlNet), e até mesmo a comunicação entre CLPs e Supervisórios feita por meio do OPC, um meio simples e rápido de indexação de variáveis.

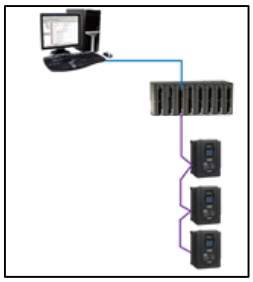
O Endereço IP (*Internet Protocol*), é uma identificação de um equipamento em uma rede, métodos convencionais de redes podem ser usados atualmente na automação, como servidores DNS *(Domain Name System)*, servidores DHCP *(Dynamic Host Configuration Protocol)* e Arquiteturas Servidores-Clientes.

De Acordo com Douglas Rocha Mendes, a Ethernet é uma arquitetura usada em redes locais, onde seu conceito é o envio de pacotes por cabeamentos elétricos. Os padrões para estes pacotes são definidos por convenção e normas, e estes mesmos padrões para envio de pacotes são aplicados á Wireless.

Para compor uma arquitetura é usual usar equipamentos como Switches, Hubs, e roteadores, cada com sua função especifica para distribuir o sinal, e compartilhar informações pela rede. A criação de Hubs foi a solução para que uma rede física existisse em topologia estrela conforme Figura 7, assim múltiplos controladores de interface de rede podem enviar

dados ao hub transportando os dados a todos equipamentos na mesma rede. Entretanto Hubs não realizam o gerenciamento de rede, o equipamento adequado para isto é o Switch.

Figura 7 -Sistema Automação (PC – CLP – Equipamentos)

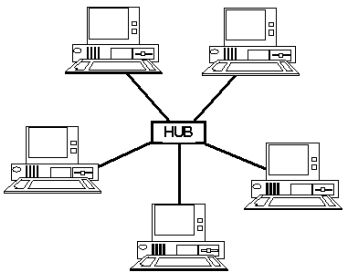


Fonte: Elaborada pelo Autor

O Switch surgiu para evitar *Broadcasts* (Relativo a excesso de pacotes transitando em todas as portas) na rede, diminuindo o direcionamento excessivo de informações a todas as portas e criando um gerenciamento de rede. Switchs de mercado como DLink, TPLink, ou alguns empresariais como CISCO, HP possuem tanto gerenciamento como acesso a VLAN (Desvios de rotas em redes), gerenciamentos de usuários, bloqueio de IPs, acessos remotos e uma infinidade de parâmetros que melhoram a qualidade da rede e sua segurança.

A indústria usa o protocolo ethernet com encapsulamentos de protocolos industriais, assim garantindo a confiabilidade das informações e maior abrangência na migração de plataformas antigas com as atuais. Um destes protocolos é o Modbus/TCP, que segue o padrão da Serial e registros, ao migrar para a ethernet, a Modbus ganhou a característica de endereçamento IP mas manteve seu sistema de escritas em registros.

Figura 8 - Topologia Hub-Estrela



Fonte: Novatec,2011

A Modbus/TCP por característica trabalha de forma mais organizada, pois separa registros digitais, analógicos e leitura e escrita, assim cada espaçamento de informações é

ocupado, mesmo que seu valor for 0. Uma das grandes vantagens do uso da rede IP é a grande abrangência de equipamentos que podem estar na mesma rede conforme a figura 8.

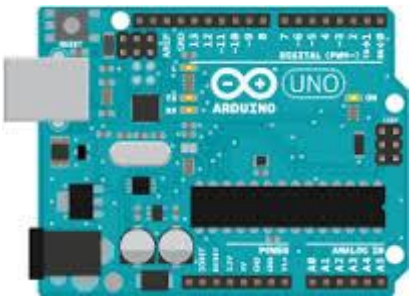
2.5 ARDUINO

O Arduino é composto pela sua placa prototipada, e seu software (IDE). O Arduino surgiu para facilitar o uso em escolas e universidades, já que não era necessário cansáveis soldas e trilhas.

A placa do Arduino é composta de um chip que é o elemento, que possui todos os componentes necessários para o correto funcionamento do microcontrolador. Pinos digitais de entrada/saída, pinos de entrada analógica, entrada USB utilizada tanto para alimentação (5V) quanto para o envio do programa para o microcontrolador, pino para alimentação externa de até 12 volts. (Banzi, 2011). O Arduino possui diversos modelos, mas seu conceito principal é o mesmo, suas variações aumentadas de entradas digitais e processamento.

A IDE do Arduino traduz o código para uma linguagem de baixo nível que o microcontrolador é capaz de entender, além de compilar o software (Verifica erros na escrita do programa), após a compilação, o programa é carregado e executado no Arduino. (Banzi,2011).

Figura 9 - Arduino



Fonte: Arduino.cc

Atualmente a plataforma Arduino possui diversos “Shields”, ou placas que conforme imagem implementam funções ao Arduino, podendo ser para aumentar suas saídas, ou adicionar uma rede especifica, como a placa ethernet, que permite que sua placa Arduino torne-se online e monitore estado dos sensores pelo navegador, além de permitir o acesso às informações na rede local, que se estiver conectado à Internet, pode ser monitorado e acessado de qualquer lugar do mundo. Este Shield, ilustrado na Figura 9, é baseado no Ethernet Chip Wiznet W5100 e fornece um endereço IP compatível com os protocolos TCP e UDP, juntando a IDE Arduino

com o Shiled Ethernet, a possibilidade de criar sistemas baseados em C com a integração de bibliotecas dedicadas para os sistemas ethernet, assim protocolos e funções da ethernet.

O Shield abre o leque de possibilidades para executar sob a plataforma, servidores WEB, banco de dados, sistemas de sites, conversores de sistemas, e outras plataforma. (THOMSEN, 2014; ARDUINOECIA,2013).

Figura 10 - Shield Ethernet

Fonte: Arduino.cc

2.7 PANORAMA ATUAL E TEMPO DE PROJETO

Um projeto de um processo ou uma planta Industrial passa por diversas áreas da Engenharia, como projetos elétricos, projetos mecânicos, projetos civis, projetos de meio ambiente, e em todos os casos, um bom planejamento evita desperdiço de recursos e auxilia na integração de diversas áreas. Conforme a DEVMedia, desenvolvedora de softwares de gerenciamento de projetos, as empresas gastam 70% do tempo corrigindo erros. Com este alto teor de confusão, a documentação se atrasa, ou fica ineficiente. Problemas são passados à construção do processo, gerando custos para refazer ou arrumar problemas de construção.

Testes e a escrita de software são muitas vezes iniciadas com atraso, e feito pelos desenvolvedores, inibindo um teste real de campo. Devido à falta de conhecimento os desenvolvedores não padronizam a linguagem e se preocupam muitas vezes em entregar o software pronto, ineficiente a entregar um projeto consolidado. Este é um dos grandes problemas entre o link entre o campo e a sala de desenvolvimento, os técnicos ou engenheiros não se importam tanto com o processo em campo durante o desenvolvimento do software, e erros comuns tornam ainda mais difícil a interligação por meio de bibliotecas dedicadas totalmente ao processo, essas que poderiam facilitar o uso para programadores, e abrir um leque de desenvolvimento que ainda é pouco explorado por programadores experientes, e um campo muito difícil para um programador jovem explorar. O uso padrão na programação deve ser

seguido por diversos campos, tanto em aplicações web, aplicações de SO e futuramente entre processo e automação.

2.8 TRABALHOS CORRELATOS

Sistema de Controle, utilizando CLP E Supervisório, para correção de fator de potência e balanceamento de fases no secundário de um transformador de uma subestação.

Rodrigo Luiz Guedesz, em seu trabalho de conclusão de curso publicado em 2009 pela universidade federal de ouro preto escola de minas colegiado do curso de engenharia de controle e automação – cecau, propõe o uso de um sistema de automação industrial utilizando o conceito de clps e supervisórios para controlar e corrigir problemas típicos em subestações.

Projeto e execução de um freio mecânico automatizado

Dos autores Lourenço Junior, Gerson Marques e Paulo Henrique Coutinho, o trabalho de conclusão de curso publicado em 2013 pela Universidade Tecnológica Federal do Paraná, Medianeira, foi executado um freio automatizado com o uso de CLPs, a maior característica que relaciona a este TCC é o uso do Easy Soft CoDeSys, uma versão do Codesys que possibilita também o uso de IHMs, ligações e conexões se aproximam pois ambos TCCs possuem a característica de utilizar o padrão Industrial em suas ligações, assim mantendo a característica deste TCC, mesmo usando controladores não usuais no ambiente industrial. O Sistema além disto também verificava uma grande quantidade de sinais analógicos, e contatores e sistemas de partidas elétricas, além de ter uma grande contribuição com o conhecimento de conexões em motores .

Projeto de implantação de um sistema de automação para silos de armazenamento de matéria prima

Do autor Rodrigo Amaral, publicado em 2012 pela universidade federal de ouro preto escola de minas colegiado do curso de engenharia de controle e automação – cecau, foi feito uma implementação de um processo de armazenamento de Soja em Silos. Devido ao grande uso de medidores de vazão, uso de CLPs se assemelha ao conteúdo deste TCC. O sistema proposto pelo Rodrigo dispõe de uma vasta característica técnica sobre o processo de Soja que foi implementado neste TCC.

3. MATERIAIS

O sistema aqui proposto tem o objetivo de unificar a área de processos Industriais em uma única plataforma, disponibilizando em telas a configuração do sistema. A lógica dentro do controlador interpreta a parametrização do processo e, por meio de cadeia de decisões, define a melhor rota para ativar os blocos pré-definidos.

Os módulos serão compostos na plataforma Arduino. A utilização deste neste trabalho se deve ao fato da redução de custo da construção do protótipo, pois sua característica frágil a campos eletromagnéticos impede o uso em Industrias, porém, para demonstrar o código/implementação do ambiente controlado aqui proposto, seu uso é adequado.

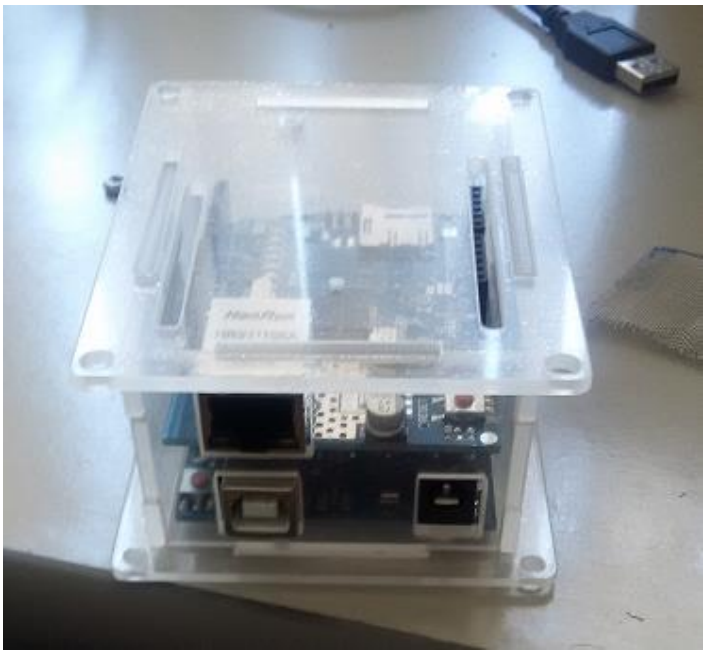
Para compor o sistema Modular, inicialmente serão utilizados:

- 2 Arduino Uno R3
- 2 Shield Ethernet HanRun HR911105A
- 1 Switch Intelbras WRS 240E
- Cabos Ethernet
- Placa Protoboard
- Materiais Gerais elétricos como Reles/Contatores/Botões e Sinaleiros
- Notebook para teste
- CLP – Schneider Modicon TM241CE24R
- IHM – Schneider Magelis HMIS5T
- Fontes WLC24-1-25 24Volts
- Fonte CGC S-3512 5Volts
- 2 Cases em acrílico para Arduino
- Shield com 4 Reles JQX-3F SH-05VDC
- 4 Reles Tianbo HJR-3FF-S-Z
- Trilho para painel modelo DIN
- Bornes conectores

4. METODOLOGIA

O método de instalação é fixado em trilho DIN, assim mais próximo ao meio industrial, para resistência mecânica o Arduino foi montado dentro de uma carcaça de acrílico, assim evitando impactos mecânicos. Conforme a figura 11, o Arduino fica dentro do invólucro fixado por 4 parafusos.

Figura 11 - Arduino e Case de acrílico



Fonte: Elaborada pelo Autor

A programação de cada Arduino, conforme apêndices, é o mais simplificada possível, pois a característica modular do sistema está em definir o IP do modulo, e definir as entradas e saídas, para concepção da plataforma, foi definido 4 shields, 2 de controle de tanque por nível e 2 de controle PID. A troca de informações passam pela IHM e pelo CLP para que o usuário da plataforma modular habilite ou desabilite o modulo. Conforme figura 10, usando a IDE Arduino, as entradas e saídas foram espelhadas em registros e definidas como entrada ou saída. Para usar o protocolo TCP/IP modbus, duas bibliotecas são necessárias a Ethernet.h e a Mudbus.h, ambas são de livre acesso e obtidas gratuitamente pelo Github[6]. Além da SPI.h para uso das demais funções do arduino. Para uso na plataforma, o arduino terá as entradas analógicas limitadas, com o intuído de prevenir erros e interferências, a plataforma é dedicada em 5 entradas digitais, e 1 saída digital, 1 saída PWM e 1 entrada analógica. Todas as funções

[6] GitHub – Site onde programadores compartilham códigos

são replicadas em registros como o Mb.R[0] define o registro 0 da modbus e usado para leituras analógicas, e o Mb.C[2] define a leitura do registro booleano da modbus, todo esse encapsulamento de informações está espelhado pela biblioteca ethernet.h que quando iniciada juntamente a Mudbus.h coleta as informações e transporta-as para a TCP.

Figura 12 - Programação IDE Arduino

UnoTCC_V1

```
#include <SPI.h>
#include <Ethernet.h>
#include <Mudbus.h>

Mudbus Mb;
//TCC USC 2016 Engenharia de Computação - Automação Modular
//Por: Ederson de Souza Leal

//Modulo 1 - PTDQ com saida para inversor/valvula ou partida direta, modulo de entrada analogica ou digital com 2 sensores
void setup()
{
  uint8_t mac[]     = { 0x90, 0xA2, 0xDA, 0x00, 0x51, 0x06 };
  uint8_t ip[]      = { 10, 10, 0, 10 };
    uint8_t subnet[]  = { 255, 255, 0, 0 };
  Ethernet.begin(mac, ip, subnet);

  //Avoid pins 4,10,11,12,13 when using ethernet shield

}

void loop()
{
  Mb.Run();

  //Analog inputs 0-1023
  Mb.R[0] = analogRead(A0); //pin A0 to Mb.R[0]
  Mb.R[1] = analogRead(A1);

  //Analog outputs 0-255
  analogWrite(6, Mb.R[6]); //pin ~6 from Mb.R[6]

  //Digital inputs
  Mb.C[2] = digitalRead(2);  //Retro VV/PD/INV
  Mb.C[3] = digitalRead(3);  //BE
  Mb.C[5] = digitalRead(5);  //Sensor LSL
  Mb.C[7] = digitalRead(7);  //Sensor LSH
  Mb.C[8] = digitalRead(9);  //Liga/Desliga Campo

  //Digital outputs


  digitalWrite(9, Mb.C[9]); //Liga Motor/VV
}
```

Fonte: Elaborada pelo Autor

Cada modulo possui uma faixa de IP dedicada, assim a plataforma se orienta e define o modulo conectado. A faixa de IP define o que cada equipamento é na rede, foi definido as faixas de IP. As saídas do Arduino são conectadas a protoboard de testes, e cada um dos Arduinos são separados por bornes de passagem, o arduino foi apenas separado e levemente fixado ao trilho, para fixação completa o Arduino pode ser fixado ao borne e ambos conectados ao trilho.

Os Arduinos são alimentados por duas situações, uma pela USB do computador ou outra por uma fonte de 5V. Para testes cada um é alimentado por uma fonte. A fonte da CGC S-3512 é uma fonte de mercado usada para circuito de câmeras, é usada neste projeto pois seu custo é reduzido, uma fonte bivolt que aceita entrada de 110V a 220V chaveada, sua tensão de saída é de 5V a 3A. Composto neste sistema também é inserido na área de controle um CLP da marca Schneider TM241CE24R, junto com uma IHM Magelis HMIS5T, este conjunto tem a função de controlar e executar logicas, além da interface gráfica conforme Apêndices A e B.

O IHM é da linha STU da Schneider Eletric, uma interface sensível ao toque de 3.4'', com resolução de 200x80 pixels. A IDE de programação da IHM é o Vijeo Designer, e como figura 13 demonstra, foi desenvolvido um sistema de navegação fixo do lado direito da tela, onde a plataforma segue a navegação livre entre telas. Conforme módulos são adicionados e executados, a plataforma libera no sistema mais telas. (Schneider Eletric,2016).

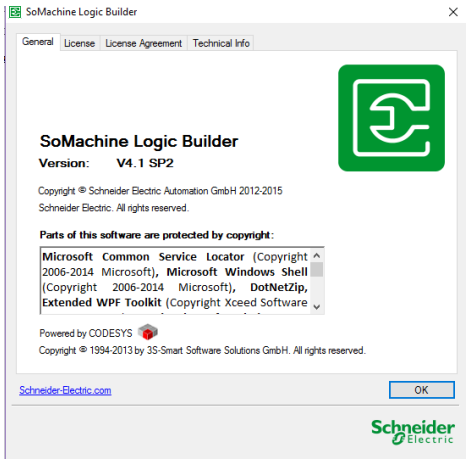
O CLP da linha TM241CE24R, da fabricante Schneider, é um controlador logico programável com algumas características de uso, ele tem alimentação direto em 110v, e uma fonte de saída de 24V, a fonte de saída do CLP interliga a IHM que tem uma alimentação de 24V, possui 10 saídas digitais e 14 entradas discretas. A IDE de programação dele é o Codesys, com uma interface gerada pela Schneider. (Schneider Eletric,2016).

Outros materiais foram adicionados como chaves e reles de saída, para ter mais funcionalidades descritas nos testes. Os relês são os dispositivos pelos quais será realizado o desligamento acionamento das saídas digitais do Arduino, com a função de interligar e isolar o Arduino do nível de campo. Sensores serão acoplados ao sistema, como a possibilidade de Inversor para controle de velocidade, contatores para partida. Sinais analógicos são gerados para teste das entradas no Arduino, estes sinais são simulados por um gerador de sinal de 0-5V, assim podendo simular um transmissor analógico, a saída analógica é conectada em no multimedidor para verificar o sinal analógico da saída PWM, alguns equipamentos são conectados nas saídas e entradas simulando os equipamentos digitais, como um led representando o comando de um motor.

O método usado para programação do CLP é o LADDER e FBD, para isto é encapsulado o CodeSys dentro do CLP, usarei uma biblioteca dedicada e fornecida pela própria Schneider em parceria com a CodeSys chamado de SoMachine, conforme figura 13, este método permite que o CLP funcione de forma cíclica, adiciona o protocolo TCP/IP Modbus em sua comunicação Ethernet, além de permitir a programação em Ladder.

A programação segue o conceito de cadeia de decisões, ao ser adicionado um modulo, seu IP é identificado, a lógica interpreta o modulo e possibilita que o supervisório atue no modulo, na tela da IHM caso o modulo for inserido aparecerá a opção para configura-lo, definir tags e o sistema.

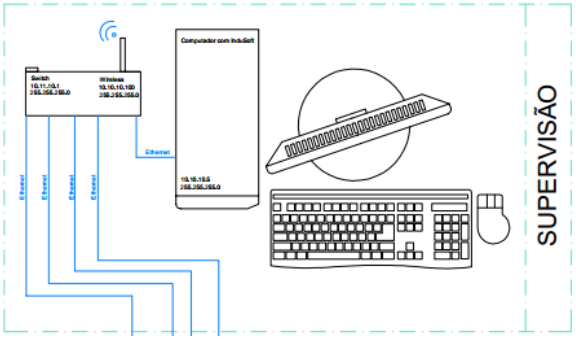
Figura 13 - Somachine Schneider/Codesys



Fonte: Elaborado pelo Autor

Conforme arquitetura apresentada na Figura 14, o sistema é dividido em níveis, supervisão, controle, e processo. O primeiro nível é o das interfaces, este será feito com base no Vijeo Designer (Schneider Eletric) conforme figura 15, possuindo ferramentas necessárias para configurar as telas e o cliente TCP/IP Modbus que faz a comunicação entre o CLP e a IHM. A navegação terá menu fixo na barra lateral fixa, conforme imagens da própria plataforma desenvolvida. Inicialmente as chamadas das telas são habilitadas conforme o usuário ativa o modulo e confirme o mesmo em tela. Esta confirmação serve para realmente confirmar a ação do usuário e limitar inicialmente o número de módulos inseridos.

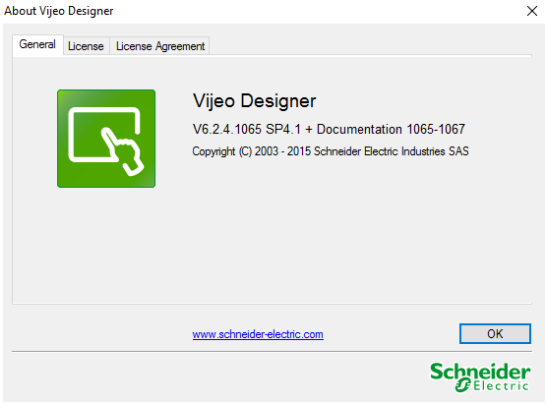
Figura 14 - Arquitetura Nível em nível (1)



Fonte: Elaborado pelo Autor

Esta é uma representação da supervisão, onde o computador é usado para acessar a plataforma remotamente, a ampliação do sistema possui uma Interface gráfica (IHM) para operação local.

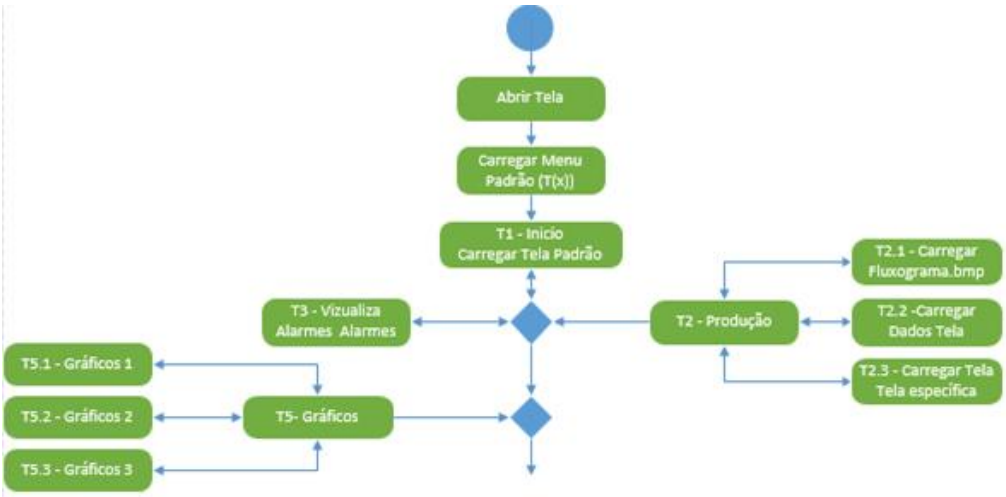
Figura 15 - Vijeo Designer



Fonte: Elaborado pelo Autor

A navegação no supervisório terá níveis de acesso e sequencial, separando níveis de operação de níveis de parametrização. Conforme a Figura 16 e 17, o sistema carrega suas informações e já disponibiliza telas de alarmes ou tela de produção. Para apresentar tais o sistema deverá ter uma parametrização previa.

Figura 16 - Diagrama UML Supervisório (Operação)



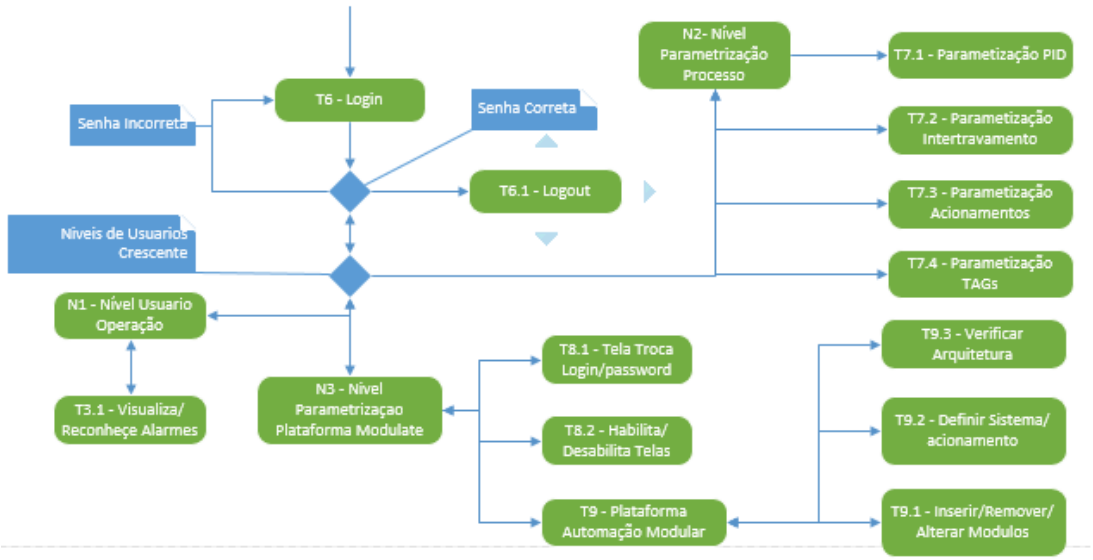
Fonte: Elaborado pelo Autor

No nível de parametrização será necessário ter uma senha para configurar o sistema, esta pode ser alterada uma vez dentro da aba parametrização. Conforme apresentado na Figura

16. O Nível de parametrização é o mesmo nível de gestor, por isto neste mesmo menu o responsável pelo sistema instalado pode reconhecer os alarmes e mudar parâmetros de produção e Setpoint[7]. A rede neste nível tem a característica de possuir IP Fixo, uma característica do sistema proposto, caso necessário trocar a faixa de IP, o sistema deve ser configurado anteriormente.

O nível de controle é composto pelo CLP da Schneider que terá a biblioteca dedicada do Codesys encapsulada, usando a IDE. O uso do Codesys tem como principal característica pois com essa base consolidada no mercado, e padrão de linguagem a migração para uma plataforma usando CLP seria mais fácil e sem grandes problemas. O programa é dividido em *calls* (chamadas), neste ponto da programação também pode ser dividida em: Chamada de verificação, esta é responsável por verificar módulos acoplados ao sistema; chamada de decisão, responsável junto com o supervisório a definir o sistema, quais blocos devem ser ativos e quais parâmetros passados a eles; e pôr fim a chamada de FBD, onde fica todos os blocos sequencialmente organizados e cada função terá seu linha de programação com um "salto" de lógica, assim a ordem das chamadas deve se manter e o ganho no processamento, já que não é toda logica executada, apenas os blocos configurados.

Figura 17 - Diagrama UML Supervisório (Parametrização)



Fonte: Elaborado pelo Autor

[7] Setpoint: Nome usado para referenciar variável que define valores para máquina.

O processamento da CPU pode ser contabilizado e ter ciclo aceitável de no máximo 500ms, pois assim os controles analógicos podem ter uma taxa de atualização rápida. Como o sistema é configurado em um hardware paralelo a taxa de PID[8] só poderá ser constatada com a arquitetura pronta. Neste nível de controle também terá os Arduinos UNO como remotas de IO, a função do Arduino, os módulos terão uma lógica simples, usarão um Shield ethernet e todo seu IO é espelhado na rede para o controlador, assim a função será isolar o controlador do campo, ter rede remota possibilitando o uso do sistema modular direto no campo.

O Arduino não é a aplicação perfeita para a indústria, pois o processador fica muito exposto a campos eletromagnéticos, para amenizar tal problema foi feito uma carcaça em acrílico e uma gaiola de Faraday para isolar campos eletromagnéticos conforme Figura 12, como proteção extra também isolarei pela gaiola com um terra e fusível na fonte. Com tais proteções o uso será mais característico e se aproximara do uso industrial.

As saídas e entradas também são isoladas galvanicamente por módulos de rele. As analógicas, no entanto, não terão isolação pois o custo do projeto se elevaria muito, para compensar será usado um cabo com Shield e malha aterrados, conforme Figura 13. O Shield do cabo poderá ser isolado junto com o do Arduino, assim o diferencial de potencial entre sinais é reduzido, porem essa ligação não deve ser seguida caso a aplicação tenha um campo eletromagnético muito forte, pois os campos poderiam influenciar a gaiola de Faraday do Arduino e gerar oscilações no processador. Os módulos a rele serão os da plataforma Arduino conforme Figura 18, pois se aplicam melhor a este projeto, o Arduino fornece a corrente que, que ao passar uma bobina, cria um campo magnético que atrai uma alavanca, esta alavanca proporciona a abertura ou fechamento do segundo circuito (Creder,2012). A ligação do mesmo é isolada da parte do Arduino e da saída, assim podendo alternar tensões (5V-220V). O uso deste facilita a isolação por segurança e disponibiliza uma tensão maior para acionar componentes. No caso esses módulos acionarão um motor e o inverso informara o status de ligado do motor.

Figura 18 - Modulo rele

Fonte: Arduinobr.com

[8] PID: Controle analógico baseado no algoritmo de proporcional, derivada e integral

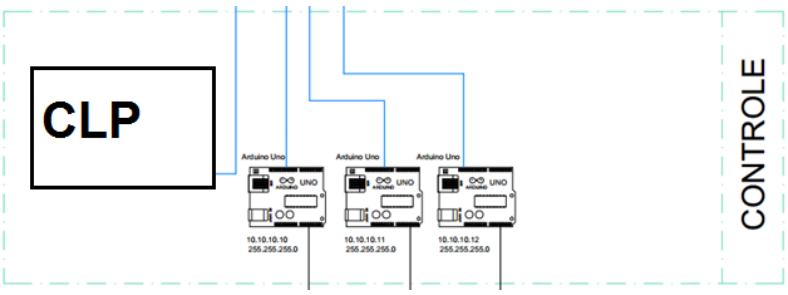
Logo, a arquitetura de controle termina com o Shield Ethernet comunicando com o CLP através de um swtich o shield Ethernet figura 19, é um equipamento que tem a função de criar uma rede IP conforme a Figura 20, o modulo se encaixa sobre o modulo Arduino criando uma camada entre o Shield[9]. Não existe nenhuma ligação física entre o Arduino e o CLP, essa característica previne descargas eletromagnéticas que poderiam trafegar por ligações elétricas.

Figura 19 - Shield Ethernet

Fonte: Arduino.com

Assim a arquitetura de controle ficará caracterizada pela conexão ethernet, módulos sem conexões físicas entre eles e trocando dados pela rede. A rede trabalha na rede IPv4 com IPs fixos sem DNS e sem DHCP. Cada Arduino antes de ser instalado deve ter seu IP configurado para não ter erros de IPs fora da faixa de produto e não sobrepor outro IP. Inicialmente como existira apenas 2 tipos de módulos, ficou reservado as faixas 10.10.XX.XX para os módulos. O Arduino informa seu modelo via programação para o CLP, e a comunicação entre eles segue o padrão TCP de 100Mbps, o protocolo usado é o Ethernet/IP entre eles, porem entre o supervisório e o CLP, usarei o TCP/IP Modbus, dedicado por registros e uso facilitado.

Figura 20 - Arquitetura de Controle

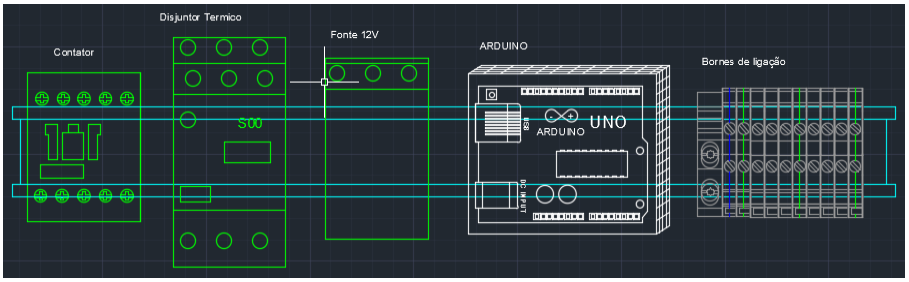


Fonte: Elaborado pelo Autor

[9] Shield: Nome usual dos módulos do arduino

Seguindo o projeto, o próximo nível é o de equipamentos de campo, neste nível é locado os equipamentos de campo[10], podendo variar muito. O Sistema Modular interpreta sinais, logo cabe ao usuário do sistema identificar ao sistema os equipamentos. Podendo ser válvulas, motores, entradas analógicas, sensores digitais, botões, etc. A única ressalva que deve ser observada é a tensão e esquema de ligação dos equipamentos, pois o sistema é responsável apenas pelo sinal que recebe e envia, isolando assim defeito de terceiros. O *bus*[11]de IOs é a interligação entre a camada do nível de controle e nível de equipamentos, este é caracterizado por cabos interligando as saídas de bornes[12] com o modulo do Arduino. Em suma os módulos devem possuir seus bornes isolados dos reles, facilitando a instalação, para isso o modulo Arduino deve ser instalado em rack com trilho DIN, este trilho é padronizado em instalações elétricas, por isso o grande uso e a facilidade de se operar. Os bornes seguirão locados de forma horizontal ao modulo, por isso o projeto para o uso do sistema modular deve conceber instalação de campo em painel. Conforme a figura a 21, o projeto elétrico consiste em instalar o modulo em um painel, alimentar o mesmo e conectar as saídas ou entradas conforme necessidade. O conceito modular se aplica idealmente em quesitos onde a Industria pode ter deficiência de um programador em seu quadro de funcionário.

Figura 21 - Referencia trilho Din



Fonte: Elaborado pelo Autor

O projeto elétrico dos Arduinos concebe a instalação de um modulo no trilho, e conexões à borne, nos testes é interligado direto na protoboard conforme Apêndices A e B, a alimentação é feita pela fonte de 24V para campo e 5V dos Arduinos. O esquema apresentado se aplica em todos os 2 módulos, pois as diferenças não se aplicam ao uso do Arduino em processos analógicos, diferenciando apenas as ligações. O modulo deve ser bem identificado

[10] Campo: Lugar onde o processo é instalado, ou maquina
[11] *Bus:* Denominação de rede, caminho ou trafego de informações.
[12] Borne: Peça metálica de um circuito elétrico à qual se liga a um fio deu ma ligação interna para uma externo

para não ocorrer erros, durante a instalação o operador pode consultar a plataforma de supervisório para checar se o modulo ficou online, consultar manual de instalação com informações cabíveis ao modulo, assim como um passo-a-passo de como parametrizar a aplicação.

Neste nível existe uma grande variedade de aplicação a necessidade, processos podem variar infinitamente, como exemplo tamanho de bombas, tamanho de válvulas, rotas de produção, controles, intertravamentos. Porem com o conceito claro do processo o sistema poderá orientar o usuário a avançar ao próximo nível

No nível de equipamentos o operador/eletricista deve instalar os equipamentos junto ao modulo, fazer as conexões já conhecidas pelo fabricante. Com base do processo fornecido, o operador constrói seu sistema e adapta ele a plataforma modular. Um exemplo seria ele automatizar um nível de um tanque com operação remota, um sistema relativamente simples porem muitas vezes ineficiente devido ao custo ou dificuldade de montagem do sistema. Este sistema é independente de entradas, por isso o cliente deve estar atento ao uso de contatos e botões em tensões diferente da entrada, um fusível pode proteger, mas não garantir a queima do modulo. A plataforma não identifica equipamentos, e sim sinais elétricos.

O Protótipo simula os dados em sinais analógicos e digitais de 5V enquanto os sinais analógicos são simulados por um multimedidor/gerador de sinais, como a plataforma é grande o suficiente para ser expandida, a simulação destes sinais pode ser usado facilmente para demonstração do algoritmo sendo executado e telas em funcionamento. Diversos processos podem ser simulados com diversos equipamentos, mas o contexto de aplicação se mantem uma vez que o equipamento não tem sua funcionalidade primaria alterada.

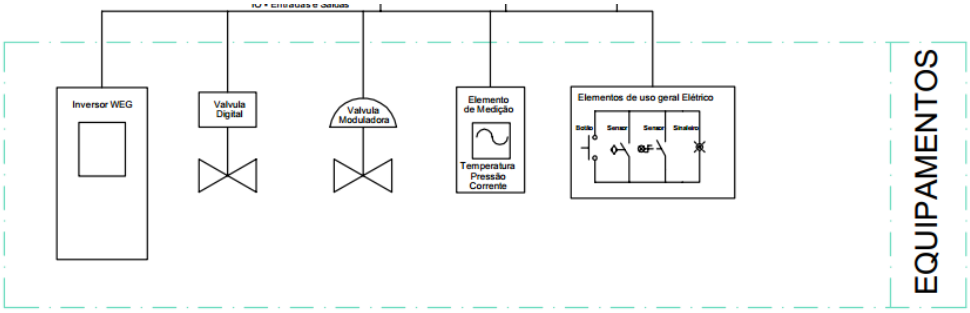
A montagem do protótipo é executada em 2 trilhos DINs, 1 para o CLP e outro para equipamentos de campo, conforme exemplo da figura 21. Esta montagem é familiarizada em ambientes industriais devido a facilidade e padrão já adotado no meio industrial.

Em um último nível, chamado de processo, baseia-se no conhecimento de como organizar os equipamentos para gerar o processo desejado. O protótipo usara o mesmo processo apresentado na Figura 22, um controle simples, porém eficiente e que demonstra a utilidade. Este depende unicamente do usuário que for instalar o sistema modular, para compor basta verificar a necessidade, e qual modulo melhor se aplica a sua lógica de processo. Algumas variações podem ser arquitetadas dentro do mesmo processo, como exemplo, trocar o nível por um controle digital de sensores, trocar a partida do motor por um inversor, trocar o controle

analógico por um controle PWM, inverter as saídas caso contatos normalmente abertos e contatos normalmente fechados e etc.

A versatilidade do processo se dá ao não apego de características especificas no algoritmo, assim contemplando apenas a “lógica crua”, e deixando que o usuário preencha as lacunas com informações necessárias. Algumas delas serão de característica obrigatória, como TAG, Tipo, Sistema de Segurança, outras menos importante, como descrição, Setpoint, alarmes. Durante este processo a lógica não permite a execução da partida/controle enquanto o sistema não estiver corretamente configurado, porem como todos sistemas de automação, o usuário deve ser responsável com o sistema e evitar burlar ou executar ações que não preza o uso de normas de segurança (NR10) e normas de segurança em maquinas (NR12).

Figura 22 - Nível de equipamentos



Fonte: Elaborado pelo Autor

Algumas dessas exceções podem ser previstas e informada pelo sistema, como falta de disjuntor, falta de botão de emergência, falta de sensores para compor o sistema de controle (exemplo criar um sistema de controle de nível, porém não inserir o range[13] da medição).

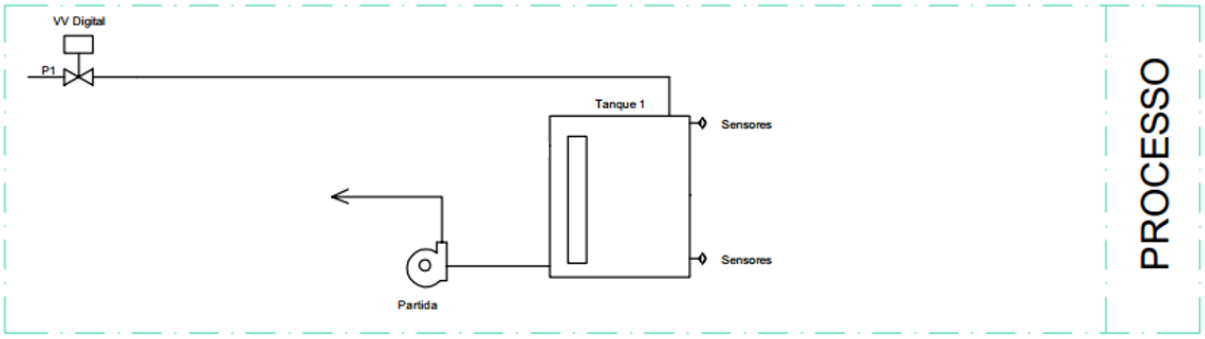
Com isso o sistema modular fecha o Ciclo de 4 etapas (Supervisão, controle, equipamentos e processo), cada um se comunicando e compondo a plataforma, o sistema não funciona sem uma delas, é necessário compor de todas. Conforme as mudanças o sistema cresce conforme mais bibliotecas e hardware se tornarem mais poderosos, pois conforme dito, o uso de equipamentos como CLP e Arduino se deve ao fato de redução de custos do projeto, mas em uma aplicação real, poderia facilmente usar a plataforma dentro de um CLP, garantindo a flexibilidade do sistema somado a robustez de um controlador.

O algoritmo criado para este projeto é basicamente o uso de blocos de funções, somado a cadeia de decisões, onde a biblioteca pode ser fechada com senha, mas o código implementado

[13] Range: Maximo e mínimo, escala de Entradas/Saídas analógicos

não tem um fim valido, apenas questões como processamento e tamanho usável da CPU que podem limitar o sistema. Logo com uma quantidade de blocos o sistema pode ganhar uma dinâmica muito maior na área de processos, aumentando o conhecimento em áreas especificas, como processos industriais, maquinas automatizadas. A plataforma é uma maneira de prestar um serviço tanto de instalação quanto de suporte de uma maneira mais eficiente, pois seguindo padrões lógicos, programadores ficariam mais aliviados em não ter que conhecer todos tipos de processos existentes, existindo uma plataforma onde abrigaria diversos conhecimentos de maquinas e processos similares. O intuito é iniciar com pequenos blocos e com funções simples para verificar a viabilidade deste sistema, porem contribuindo com a comunidade de programadores de processos, programadores de CLP e o próprio usuário final, que dependera muito menos de um programador, tendo um supervisório padronizado, e um suporte mais simples. Inicialmente foi executado um bloco chamado PDTQ (partida direta e tanque) conforme figura 23, para controle de um tanque conforme seu nível, este controle pode ser mudado para controle com válvula e seu nível controlado conforme enchimento ou esvaziamento do tanque. Para simulação serão usados chaves de simulando contatos de nível, e a geração analógica para representar o nível do tanque em um sinal escalonar de 5V.

Figura 23 - Nível Processo



Fonte: Elaborado pelo Autor

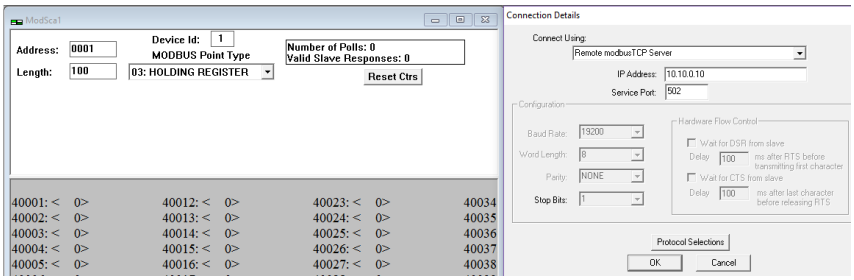
A versatilidade industrial é gigantesca, por isso o algoritmo contemplará alguns conceitos de processos industriais conhecidos, como exemplo o controle de um tanque, o uso do tanque indefere, seu produto, seu tamanho o tamanho da bomba que controlada, e as válvulas de entrada e saída, o controle será o mesmo com um PID e uma saída analógica com PWM, este conceito é a grande chave para o uso de bibliotecas padronizada, a característica principal deste

trabalho de conclusão de curso. Outros módulos como entrada e saída digital foram implementados para mostrar a capacidade da plataforma.

5. TESTES E RESULTADOS

Após sistema montado os testes foram feitos na ordem de software para campo, para validar o uso da Modbus/TCP foi usado o software Modscan que monitora a modbus e verifica seus registros. Conforme imagem 21.

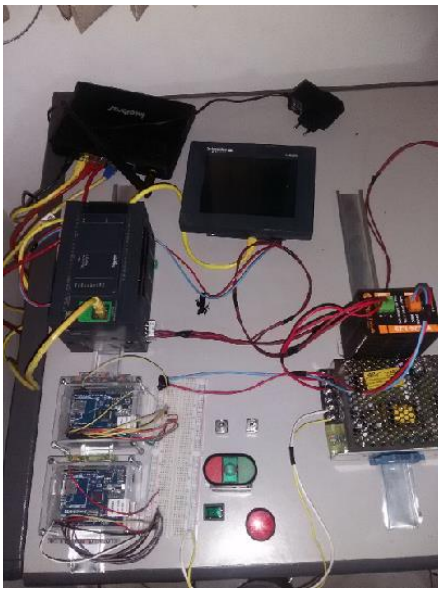
Figura 24 - Software Modscan



Fonte: Elaborado pelo Autor

No Modscan é possível verificar os registros programados no Arduino e no CLP, com essa ferramenta é possível pular a tela de operação para testar a viabilidade dos Arduinos. Com o software testado a plataforma foi montada conforme figura 25, sob uma mesa os equipamentos foram testados individualmente.

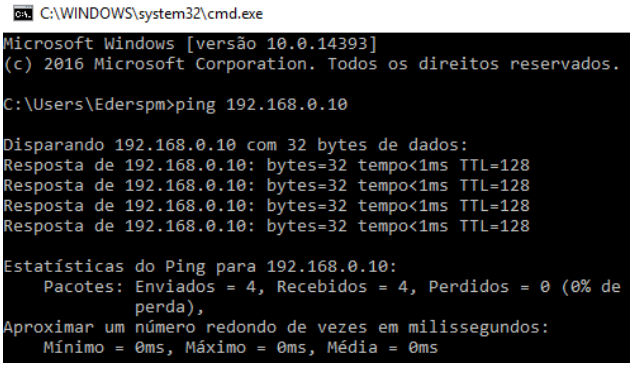
Figura 25 - Bancada de testes



Fonte: Elaborado pelo Autor

A fase de teste de software ainda era composta por alguns testes com o comando ping do Windows, assim uma validação de equipamentos online e velocidade de transmissão de dados conforme figura 26.

Figura 26 - Comando Ping



Fonte: Elaborado pelo Autor

A plataforma ainda permite uma monitoração em tempo real do CLP, assim é possível verificar as variáveis que comunicam até a IHM, outra vantagem da plataforma é simular a tela de operação no computador, e disponibilizar via web a mesma tela. Seguindo os testes, o hardware foi montado separadamente dos testes de software, logo todos os pinos do Arduinos foram conectados, mesmo que o modulo de PID use apenas 1 entrada analógica e 1 saída, assim todos os pinos usados no Arduino estão na protoboard. Por recomendação, não foi usado os pinos 4,10,11,12,13,14 pois o Shield ethernet ocupa estes espaços.

Seguindo para os testes de hardware, na protoboard foi separado conforme esquema, as entradas das saídas com reles, e assim que interligados para teste individualizado, cada Arduino recebeu um programa conforme apêndices A e B (modulo PID e modulo PDTQ), assim energizados pela fonte 5V. Antes de entrar na plataforma cada entrada é verificada, e a analógica é interligada na sua saída para ter um retorno loop, testando assim o PWM em uma entrada analógica.

Ao iniciar o sistema, uma tela de proteção foi desenvolvida conforme figura 27, assim antes da operação ou configuração, é identificado a planta ou sistema.

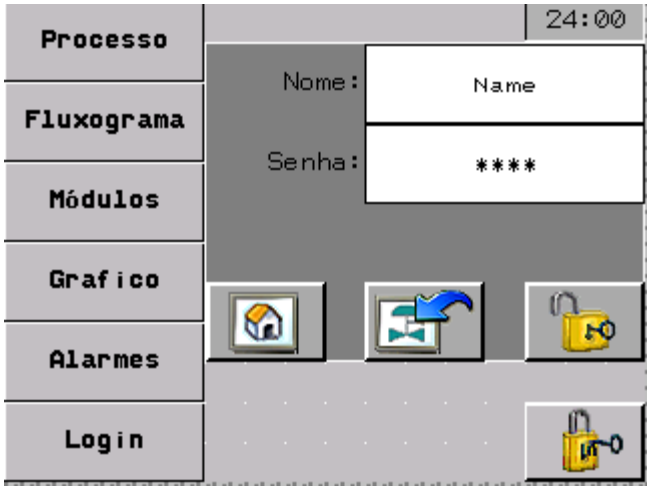
Antes da configuração é necessário um login na plataforma, conforme figura 28, após isto é necessário que o usuário habilite o modulo, dos 4 em execução.

Figura 27 - Tela Inicial

Fonte: Elaborado pelo Autor

Os usuários são criados na plataforma Vijeo Designer, e apenas em programação podem ser alterados, assim o sistema ganha em nível de proteção.

Figura 28 - Tela Login

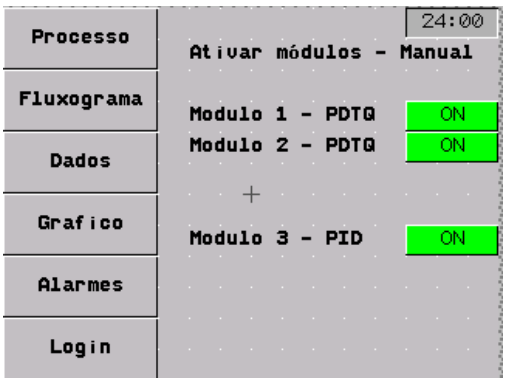


Fonte: Elaborado pelo Autor

Continuando o teste do sistema a seguir habilitado um de cada modulo e habilitado, quando esta função é ativa, o sistema começa a executar as linhas de programação do

bloco programado em questão. Assim gerando as funções necessárias para a plataforma seguir com o modulo. Conforme figura 29, todos os módulos disponíveis para teste. Cada modulo executou como previsto, e a leitura esta cíclica em 40ms em cada modulo, o sistema opera com uma capacidade limitada do Switch, pois este só possui 4 portas disponíveis.

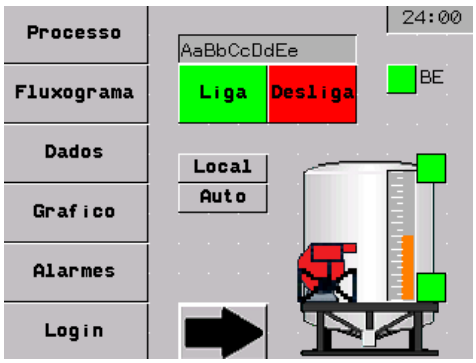
Figura 29 - Tela ativar módulos



Fonte: Elaborado pelo Autor

As telas de operação seguem o conceito de visibilidade, quando recebe a confirmação de modulo ativo, ou de usuário habilitando o modulo, o CLP envia informações e a IHM mostra as opções para o usuário preencher, na tela do primeiro modulo, existe a opção de acionar uma válvula ou um motor, o nível ser analógico ou digital, o motor encher ou esvaziar o tanque, opção de inserir um TAG, e opção de seguir para próxima tela, ligar ou desligar, informa o botão de emergência, opção para acionamento local ou remoto, e função automática controlando o nível em 50% ou pelos sensores, conforme figura 30.

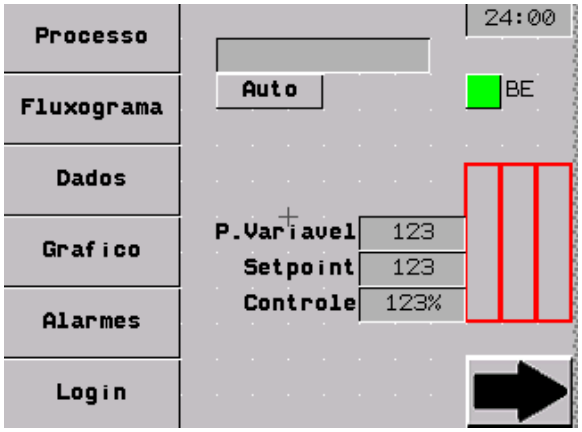
Figura 30 - Tela controle de tanque



Fonte: Elaborado pelo Autor

O modulo PID tem as opções de verificar o botão de emergência, TAG, variáveis de entrada, de setpoint e de controle. As variáveis de proporcional, integral e derivada são fixas no CLP, geradas para melhor otimização conforme ciclo SCAN do CLP.

Figura 31 - Controle PID



Fonte: Elaborado pelo Autor

O sistema foi verificado com máxima capacidade ativa, e verificou-se que o *Scan* estava entorno de 100ms e com a plataforma estava variando entorno de 80-90ms, esta medição foi obtida a partir de um monitoramento direto da CPU pelo SoMachine.

5.1 DIFICULDADES OBTIDAS

Devido ao tamanho da plataforma e sua expansão da plataforma, os Arduinos não comportaram adequadamente em velocidades o protocolo Modbus, causando quedas no sistema, para isto o CLP executou algumas funções em sua saída. Com o uso de um CLP de pequeno para médio porte, seu ciclo *Scan* é muito otimizado, e a para comparativo a plataforma teve que ser exposta a situações maiores que as programadas, assim gerando um programa teste com diversos módulos copiados e executando ao mesmo tempo para checagem de tempo de *Scan*. O Uso da Modbus se tornou muito manual, e a falta de um protocolo OPC tornou muito difícil a expansão de diversos módulos. A alta

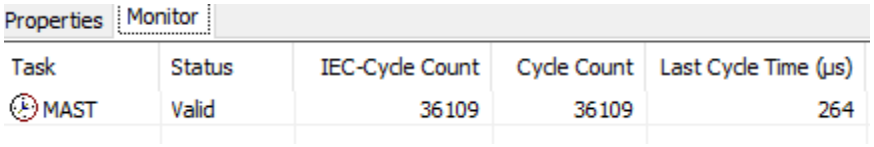
complexibilidade de programação torna a plataforma muito fechada para implementar OpenSource.

5.2 RESULTADOS OBTIDOS

Alguns fatores foram testados para obtenção de dados, o primeiro foi tempo usado para programar, comparado a uma programação padrão. Neste quesito a plataforma modular perde, pois sua primeira programação ou implementação de novos módulos é muito complexa, e exclui programadores iniciantes, logo o tempo de desenvolvimento e ampliação é muito demorado.

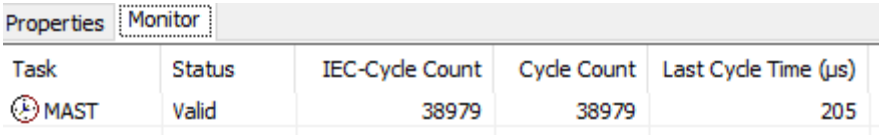
Outro fator analisado é a programação ativando os módulos e desativando os módulos, assim é possível perceber a oscilação do tempo de leitura do CLP, abaixo uma variação aproximadamente de 50microsegundos de tempo, isto com poucos módulos inseridos.

Figura 32 - *Scan* Modulos ativados



Fonte: Elaborado pelo Autor

Figura 33 - Scan modulos filtrados



Fonte: Elaborado pelo Autor

Outro fator analisado na plataforma é a segurança das telas e layout facilitado, um ganho em operação e sistemas padronizados. O PID se mostrou útil a aplicações de menores porte e sua execução ocorreu sem muitos problemas de execução. Sua funcionalidade aplicada a saída PWM se manteve dentro dos parâmetros aceitáveis de controle. A resposta do PID é mais influenciada aos seus parâmetros comparados ao Scan da rede e CLP.

5.3 PROPOSTA PARA TRABALHOS FUTUROS

Como inicialmente a plataforma teve conceito de ser OpenSource, a esquematiza de programação foi feita para se adaptar e comportar novos módulos programados, assim nada impede deste trabalho possa ser implementado em campo.

Melhorias nos módulos Arduinos podem ser elaboradas para correção ou uso de outros protocolos de rede, aumentando a diversidade de equipamentos e reduzindo a complexibilidade do código.

6. CONCLUSÃO

Conclui-se que a utilização da plataforma se torna útil quando a planta industrial pode se ampliar. Outra função dela foi a mobilidade, este sistema pode ser empregado muito mais rápido e fácil que o convencional, assim a mesma programação e sistema pode ser aplicado diversas vezes na mesma planta. A redução secundaria na programação foi de considerável, pois o nível da programação foi muito complexo, porem mesmo assim houve ganho quando o sistema foi multiplicado. O tempo *Scan* do CLP foi melhorado comparado ao padrão de programação convencional, onde linhas desnecessárias não são executadas.

A biblioteca pode ser usada independentemente da plataforma, outro ganho inesperado da plataforma, assim o conhecimento pode ser dissolvido muito além do esperado. A demonstração do código executando por completo e executando apenas os módulos ativos confirma que o sistema pode ganhar em versatilidade.

Com o uso correto das entradas e saídas, monitoramento e uso de equipamentos industriais, a plataforma pode ser aplicada em industrias e sistemas que dependem de controle remoto, alinhando a confiabilidade de um sistema industrial a um custo de implementação reduzido.

REFERÊNCIAS

Automação Industrial–Definição e História, **Comat Releco World Relays**, [2013], Disponível em: <https://comatreleco.com.br/automacao-industrial-historia/>. Acesso em: 13 abril. 2016.

Microprocessador e Microcontroladores, **Universidade Federal de Goiás UFG ,** Disponível em: <http://www.emc.ufg.br/~jwilson/aulasmicro/Teoria%20de%20Micro%20-%20Parte%201.pdf/>. Acesso em: 11 abril. 2016.

Adruino CC, **Site Oficial da família dos encapsulamento Arduino**, Disponivel no seguinte link <https://www.arduino.cc/en/Guide/HomePage> Acesso em 12 Abril. 2016.

Valdinei Rodrigues dos Reis, **Arduinobr**, Disponivel em: <https://www.arduino.com> Acesso em 12 Abril. 2016.

BANZI, M. **Primeiros passos com Arduino**. 1ª ed. [São Paulo]: Novatec, 2011.

Curso de Controladores Lógico Programáveis, **UERJ Universidade do Estado do Rio de Janiero,** Disponível em: <http://www.lee.eng.uerj.br/downloads/cursos/clp/clp.pdf>.Acesso em 13 Abril 2016.

AGUIRRE, L**. Enciclopédia De Automática (Controle e Automação).** 1ª ed. [São Paulo]: Blucher, 2007.

DORF, R. BISHOP; ROBERT, H. **Sistemas de controle modernos**. 8 ª ed. [Rio de Janeiro]: Livros Técnicos e Científicos, 2001.

GUEDES, L. **TCC Sistema de controle, utilizando CLP e supervisório, para correção de fator de potência e balanceamento de fases no secundário de um transformador de uma subestação**, Disponível em: <http://http://www.em.ufop.br/cecau/monografias/2009/Rodrigo%20L%20Guedes.pdf> Acesso em 13 Abril 2016.

AMARAL, R., **Projeto de Implantação de um Sistema de Automação para Silos de Armazenamento de Matéria Prima**, Disponível em: <http://www.em.ufop.br/cecau/monografias/2012/Rodrigo%20Amaral.pdf> Acesso em 20 Maio 2016

JUNIOR, L., MARQUES, G. COUTINHO,P. **Projeto e execução de um freio mecânico automatizado**, Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/832> Acesso em 20 Maio 2016

Annecyelectronique, ***Embedded Systems & Expertise, diagnostic and training solutions*** Dísponivel em: <https:annecyelectronique.fr> Acesso em 20 Maio 2016

Schneider Eletric, **Magelis STO_STU Informativos**, Dísponivel em: <http://www.schneider-electric.com/en/product-range/5774-magelis-sto---stu/> Acesso em 7 Novembro 2016

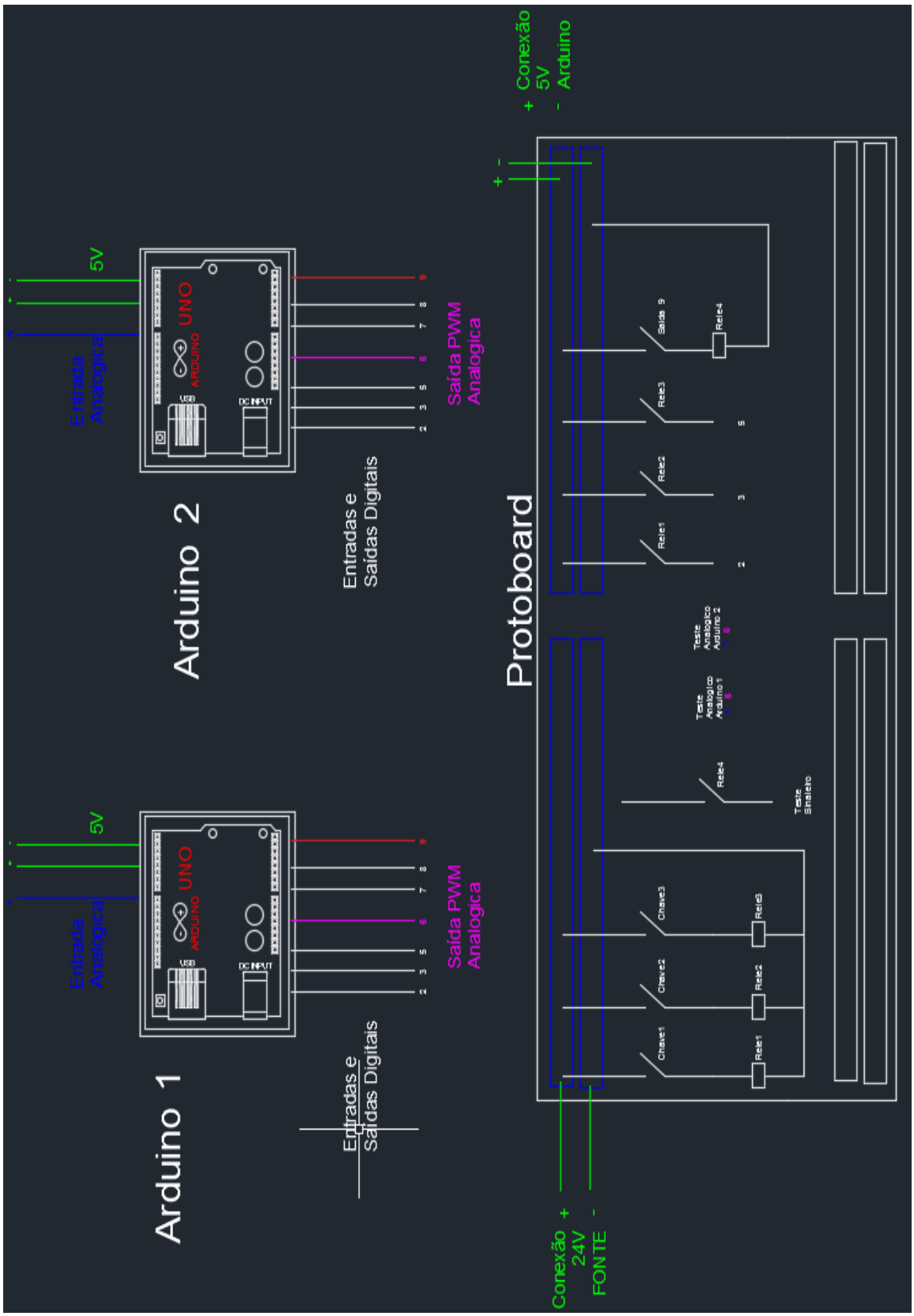
MENDES, D. ROCHA, **Redes de Computadores Teoria e Prática**, 1ª ed. [São Paulo]: Novatec, 2011

RAMOS, D. **DEVMedia**, Disponível em: <http://www.devmedia.com.br/estimativa-de-tempo-em-gerencia-de-projetos-de-software-revista-engenharia-de-software-magazine-51/25757> Acesso em 31 de Maio 2016

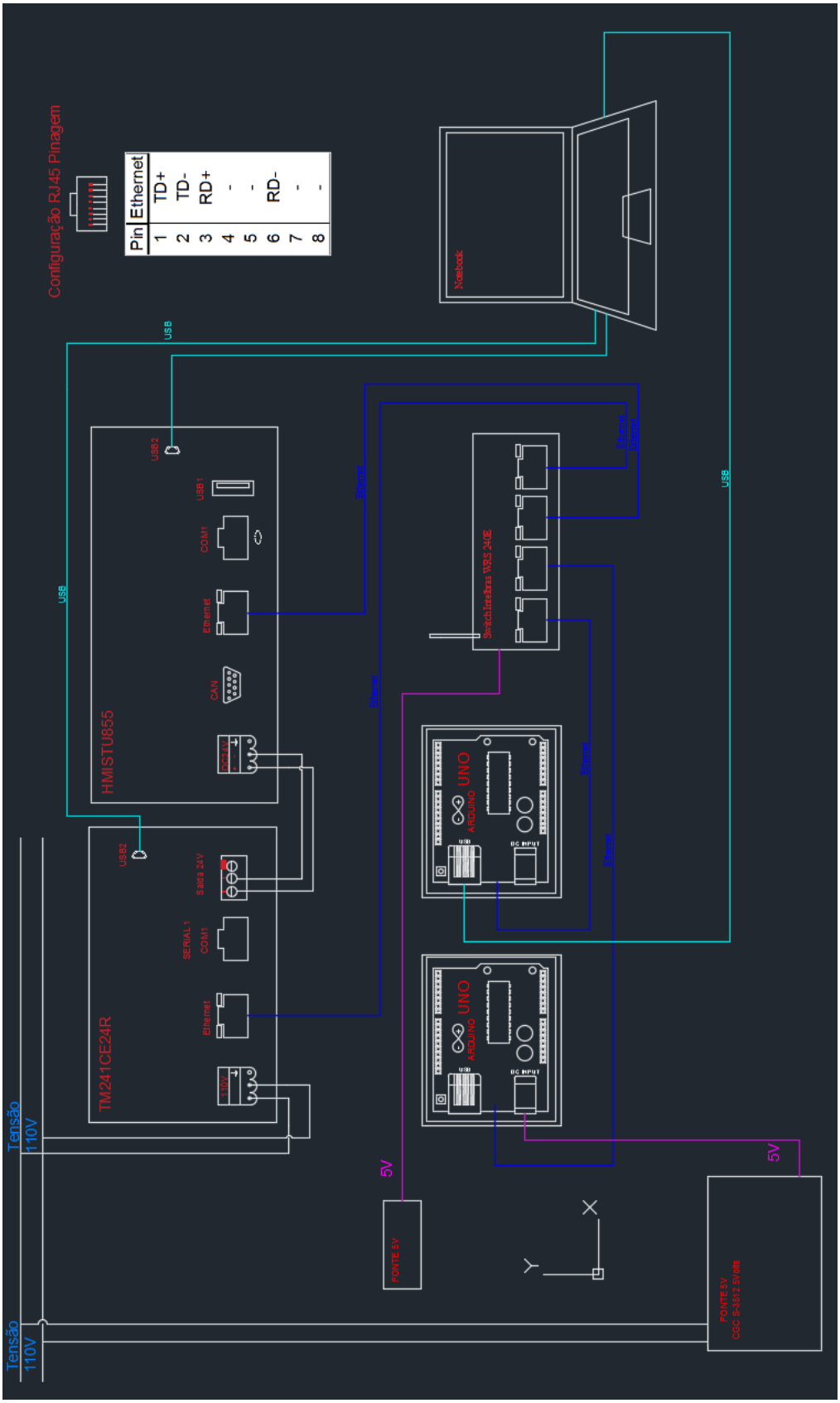
RAMOS, D. **DEVMedia**, Disponível em: <http://www.devmedia.com.br/qualidade-de-software-uma-questao-de-eficiencia/17803> Acesso em 20 Maio 2016.

Fonseca, O. Marcos, **Aplicando a norma IEC61131 na automação industrial,** 1ª ed. [São Paulo]: Jõao Aristides Bottura Filho - ISA

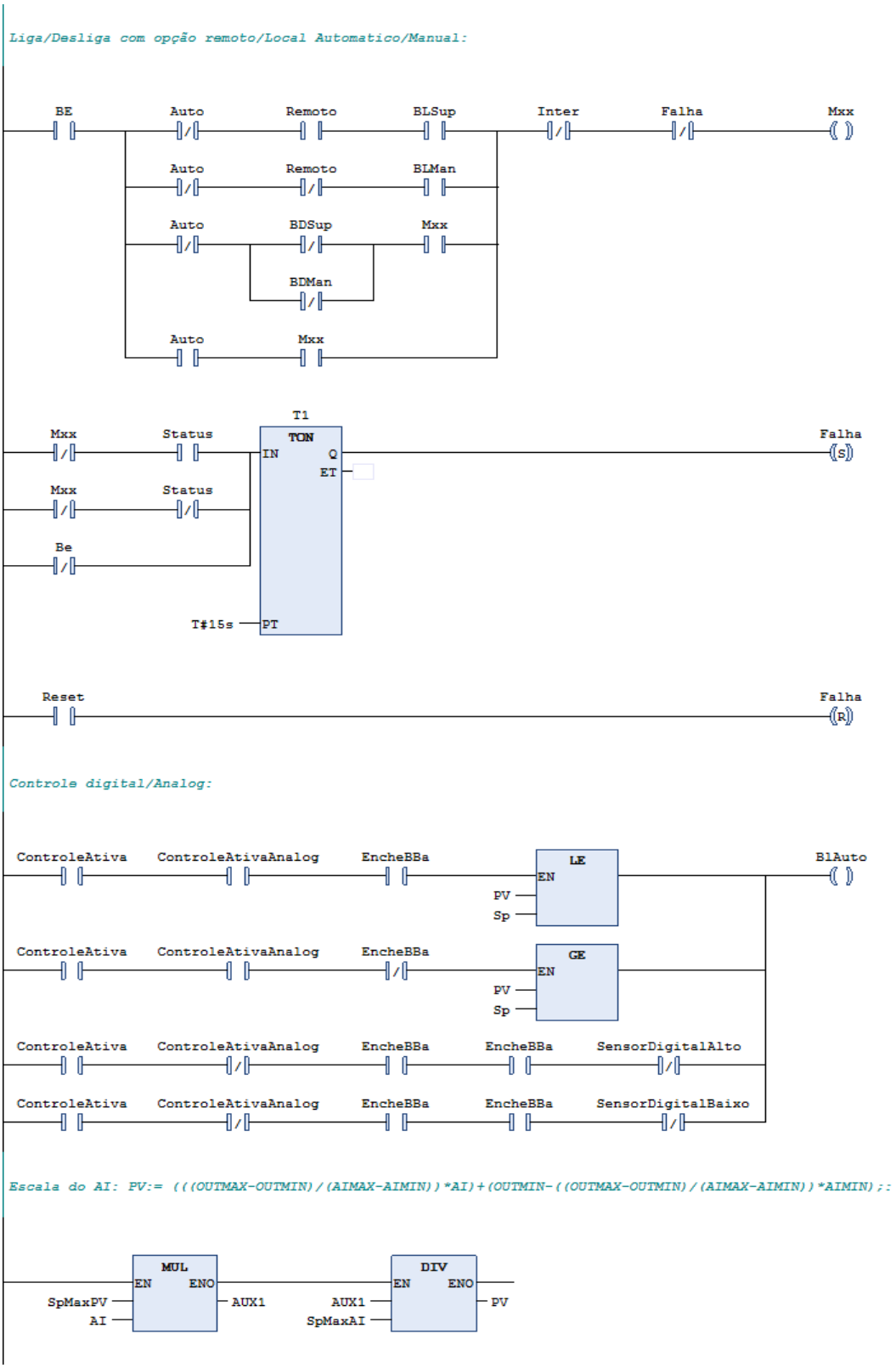
APÊNDICE A – ESQUEMA ELÉTRICO DO DISPOSITIVO



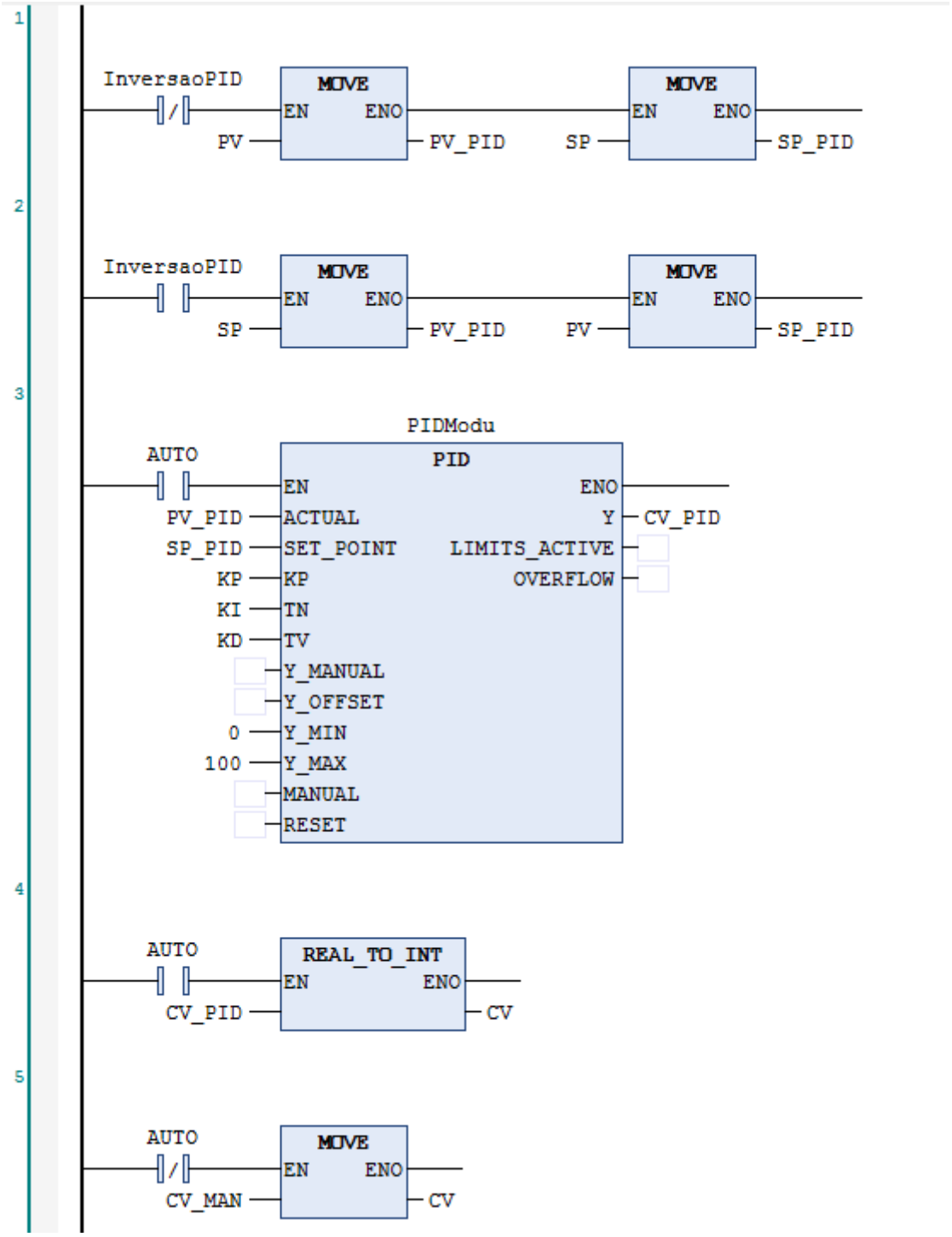
APÊNDICE B – ESQUEMA ELÉTRICO DE REDE



APÊNDICE C – PROGRAMA BLOCO FUNCIONAL PARTIDA TANQUE



APÊNDICE D – PROGRAMA BLOCO FUNCIONAL PID



APÊNDICE E – PROGRAMA COMUNICAÇÃO IHM

```
//Md1 -Leitura/escrita HoldingRegisters
String Md1Valid_scr;
Md1Valid_scr = Md1ReadValid_IHM.getStringValue();
Md1ReadMd1.write(Md1Valid_scr);
String Md1AI_scr;
Md1AI_scr = Md1ReadA1_IHM.getStringValue();
Md1ReadA0.write(Md1AI_scr);
 String Md1R2_scr;
Md1R2_scr = Md1Read2_IHM.getStringValue();
Md1Read2.write(Md1R2_scr);
String Md1R3_scr;
Md1R3_scr = Md1Read3_IHM.getStringValue();
Md1Read3.write(Md1R3_scr);
  String Md1R5_scr;
Md1R5_scr = Md1Read5_IHM.getStringValue();
Md1Read5.write(Md1R5_scr);
String Md1R7_scr;
Md1R7_scr = Md1Read7_IHM.getStringValue();
Md1Read7.write(Md1R7_scr);
String Md1R8_scr;
Md1R8_scr = Md1Read8_IHM.getStringValue();
Md1Read8.write(Md1R8_scr);
String Md1AO_scr;
Md1AO_scr = Md1WritePWM.getStringValue();
Md1WriteAO_IHM.write(Md1AO_scr);
   String Md1_9_scr;
Md1_9_scr = Md1Write9.getStringValue();
Md1Write9_IHM.write(Md1_9_scr);
```

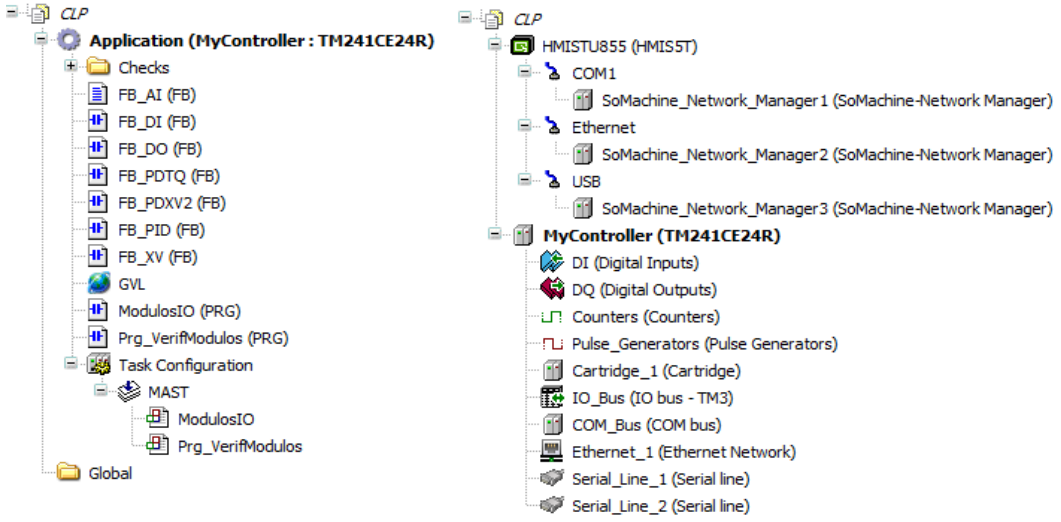
APÊNDICE F – PROGRAMA ARDUINO MODULO 1

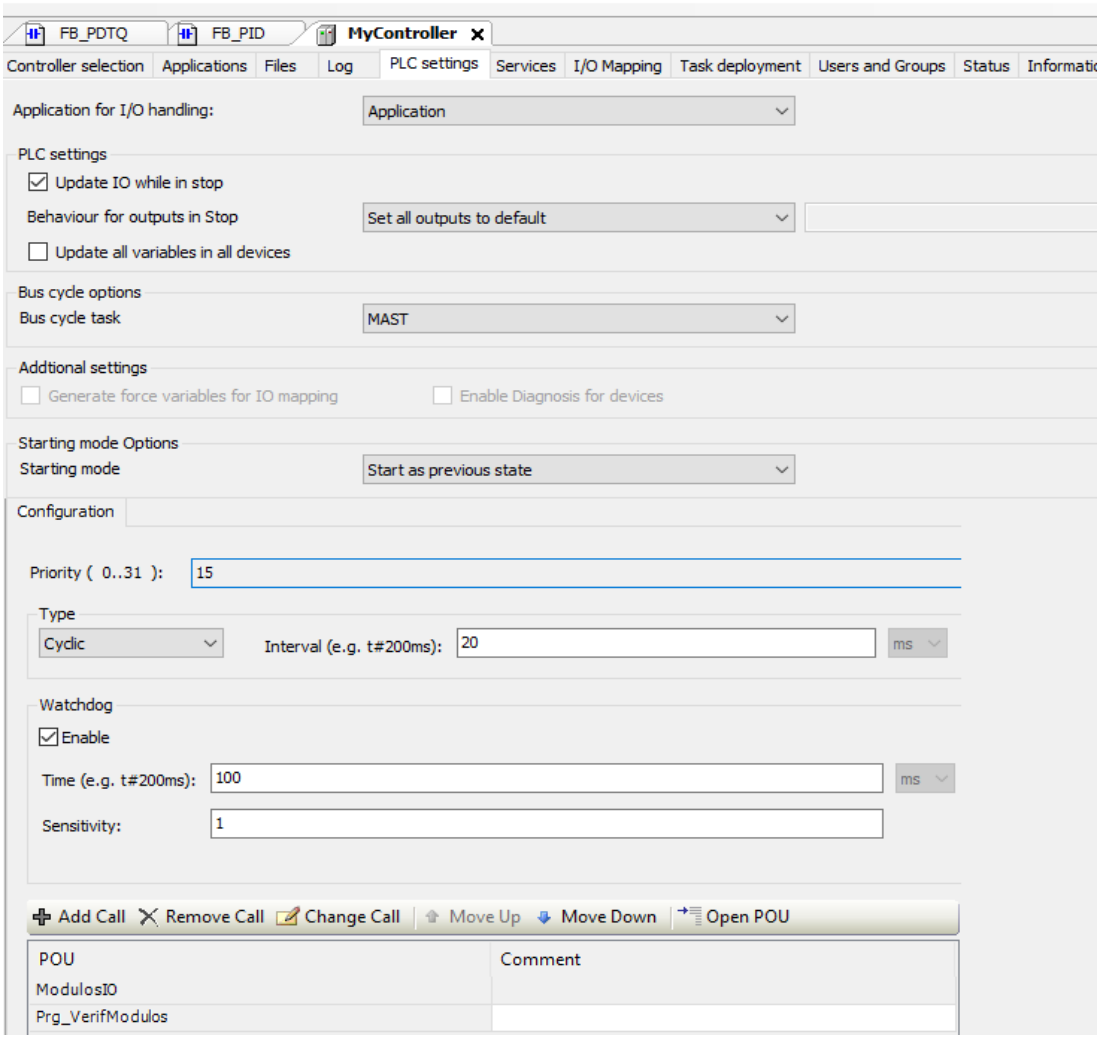
```
#include <SPI.h>
#include <Ethernet.h>
#include <Mudbus.h>
Mudbus Mb;
//TCC USC 2016 Engenharia de Computação - Automação Modular
//Por: Ederson de Souza Leal
//Modulo 1 - PTDQ com saida para inversor/valvula ou partida direta, modulo de entrada
analogica ou digital com 2 sensores
void setup(){
  uint8_t mac[]     = { 0x90, 0xA2, 0xDA, 0x00, 0x51, 0x06 };
  uint8_t ip[]      = { 10, 10, 0, 10 };
    uint8_t subnet[]  = { 255, 255, 0, 0 };
  Ethernet.begin(mac, ip, subnet);
    //Avoid pins 4,10,11,12,13 when using ethernet shield }
void loop(){
  Mb.Run();
    //Analog inputs 0-1023
  Mb.R[0] = analogRead(A0); //pin A0 to Mb.R[0]
  Mb.R[1] = analogRead(A1);
  //Analog outputs 0-255
  analogWrite(6, Mb.R[6]); //pin ~6 from Mb.R[6]
  //Digital inputs
  Mb.C[2] = digitalRead(2);  //Retro VV/PD/INV
  Mb.C[3] = digitalRead(3);  //BE
  Mb.C[5] = digitalRead(5);  //Sensor LSL
  Mb.C[7] = digitalRead(7);  //Sensor LSH
  Mb.C[8] = digitalRead(8);  //Liga/Desliga Campo
  //Digital outputs
  digitalWrite(9, Mb.C[9]); //Liga Motor/VV
```

APÊNDICE G – PROGRAMA ARDUINO MODULO 2

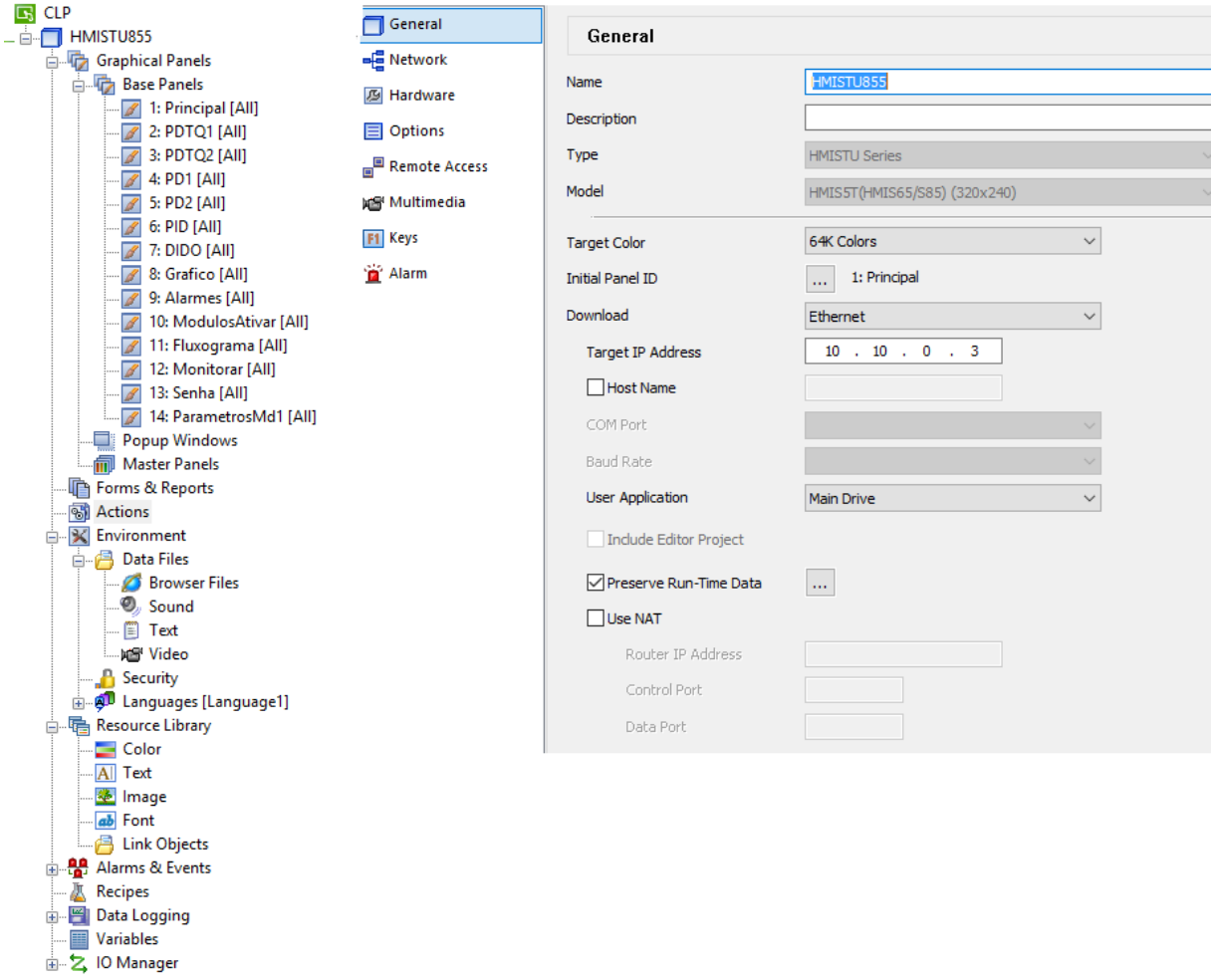
```
#include <SPI.h>
#include <Ethernet.h>
#include <Mudbus.h>
Mudbus Mb;
//TCC USC 2016 Engenharia de Computação - Automação Modular
//Por: Ederson de Souza Leal
//Modulo 1 PID
void setup(){
  uint8_t mac[]     = { 0x90, 0xA2, 0xDA, 0x00, 0x51, 0x06 };
  uint8_t ip[]      = { 10, 10, 0, 20 };
    uint8_t subnet[]  = { 255, 255, 0, 0 };
  Ethernet.begin(mac, ip, subnet);
    //Avoid pins 4,10,11,12,13 when using ethernet shield }
void loop(){
  Mb.Run();
    //Analog inputs 0-1023
  Mb.R[0] = analogRead(A0); //pin A0 to Mb.R[0]
  Mb.R[1] = analogRead(A1);
  //Analog outputs 0-255
  analogWrite(6, Mb.R[6]); //pin ~6 from Mb.R[6]
  //Digital inputs
  Mb.C[2] = digitalRead(2);  //Retro VV/PD/INV
  Mb.C[3] = digitalRead(3);  //BE
  Mb.C[5] = digitalRead(5);  //Sensor LSL
  Mb.C[7] = digitalRead(7);  //Sensor LSH
  Mb.C[8] = digitalRead(8);  //Liga/Desliga Campo
  //Digital outputs
  digitalWrite(9, Mb.C[9]); //Liga Motor/VV
```

APÊNDICE H – CONFIGURAÇÃO CLP

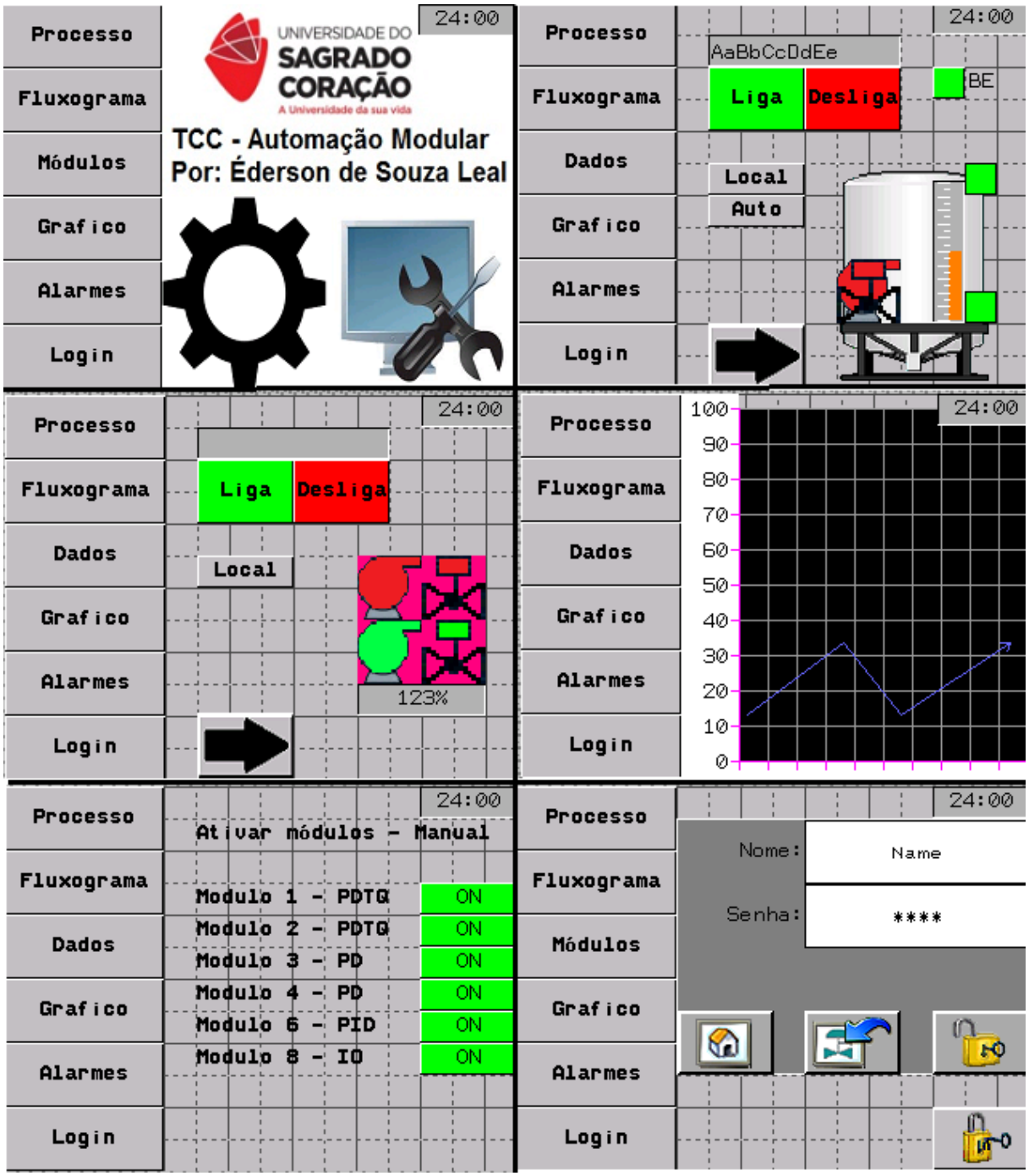




APÊNDICE I – CONFIGURAÇÃO IHM

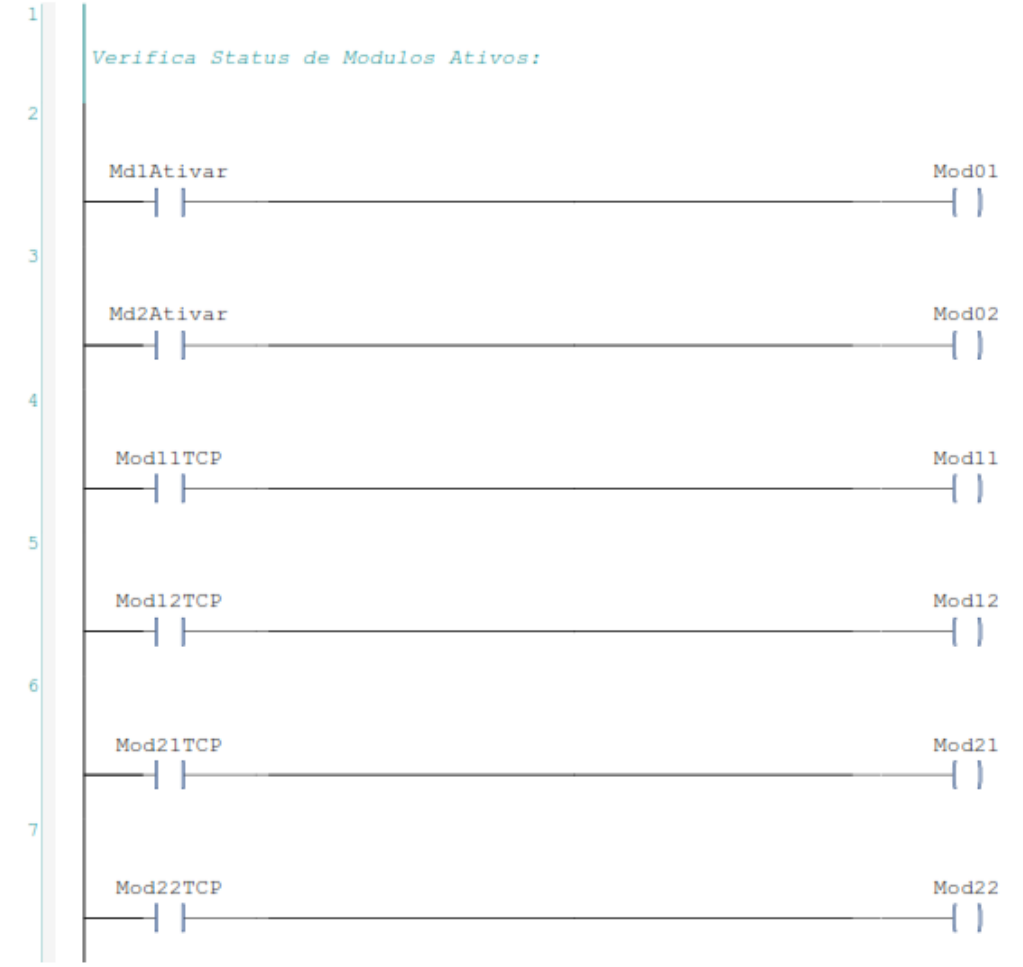


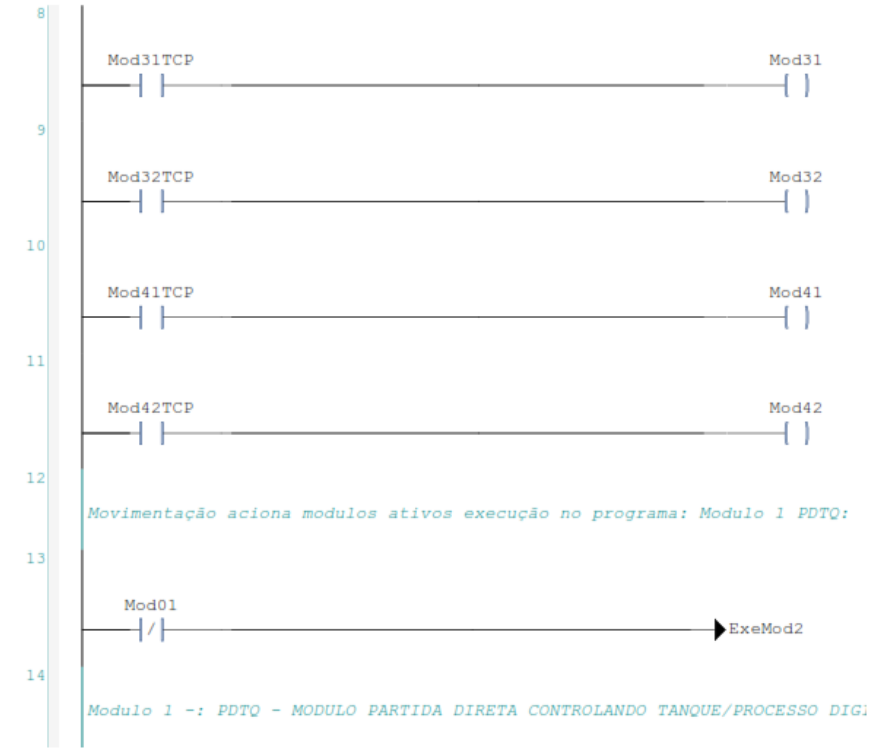
APÊNDICE J – TELAS DO SISTEMA

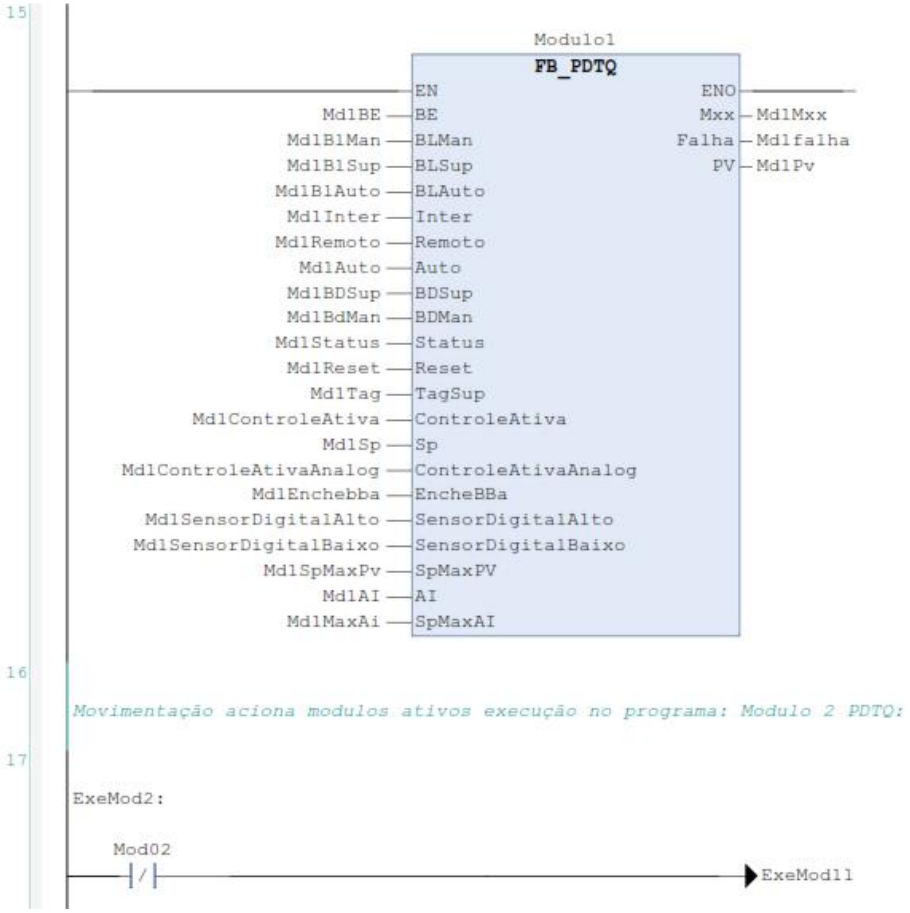


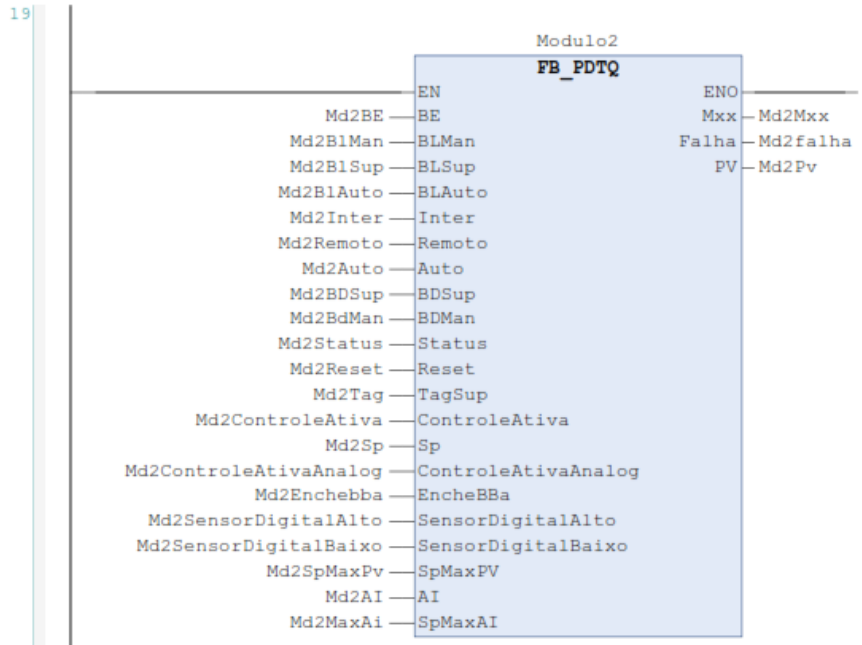
APÊNDICE K – PROGAMAÇÃO DE CALLS CLP

```
1   PROGRAM Prg_VerifModulos
2   VAR
3       Modulo1 : FB_PDTQ ;   //FB
4       Modulo2 : FB_PDTQ ;  //FB
5       Modulo11 : FB_PDXV2 ;
6       Modulo12 : FB_PDXV2 ;
7       Modulo21 : FB_PDXV2 ;
8       Modulo22 : FB_PDXV2 ;
9       Modulo31 : FB_PID ;
10      Modulo32 : FB_PID ;
11      Modulo41 : FB_DI ;
12      Modulo42 : FB_AI ;
13
14  END_VAR
15
```

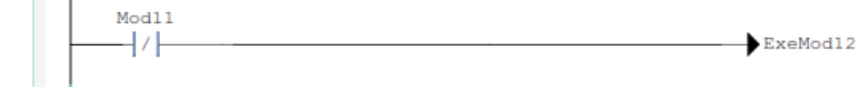


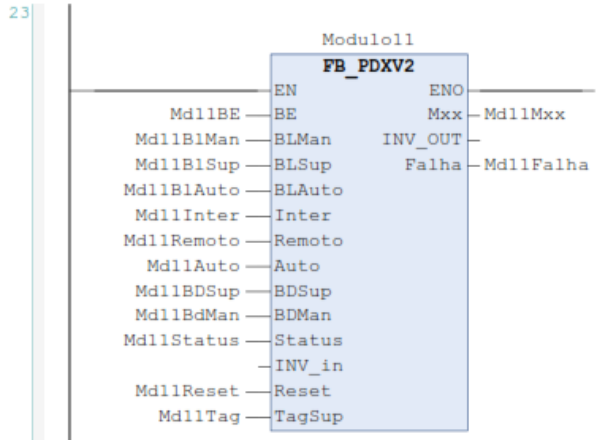

8
Mod31TCP
Mod31
9
Mod32TCP
Mod32
10
Mod41TCP
Mod41
11
Mod42TCP
Mod42
12
Movimentação aciona modulos ativos execução no programa: Modulo 1 PDTQ:
13
Mod01
ExeMod2
14
Modulo 1 -: PDTQ - MODULO PARTIDA DIRETA CONTROLANDO TANQUE/PROCESSO DIG


15
Modulo1
FB_PDTQ
EN
ENO
Mdl1BE
BE
Mdl1Mxx
Mdl1Bl1Man
BLMan
Mdl1falha
Mdl1Bl1Sup
BLSup
Mdl1Pv
Mdl1Bl1Auto
BLAuto
Mdl1Inter
Inter
Mdl1Remoto
Remoto
Mdl1Auto
Auto
Mdl1BDSup
BDSup
Mdl1BdMan
BDMan
Mdl1Status
Status
Mdl1Reset
Reset
Mdl1Tag
TagSup
Mdl1ControleAtiva
ControleAtiva
Mdl1Sp
Sp
Mdl1ControleAtivaAnalog
ControleAtivaAnalog
Mdl1Enchebba
EncheBBa
Mdl1SensorDigitalAlto
SensorDigitalAlto
Mdl1SensorDigitalBaixo
SensorDigitalBaixo
Mdl1SpMaxPv
SpMaxPV
Mdl1AI
AI
Mdl1MaxAi
SpMaxAI
16
Movimentação aciona modulos ativos execução no programa: Modulo 2 PDTQ:
17
ExeMod2:
Mod02
ExeMod11



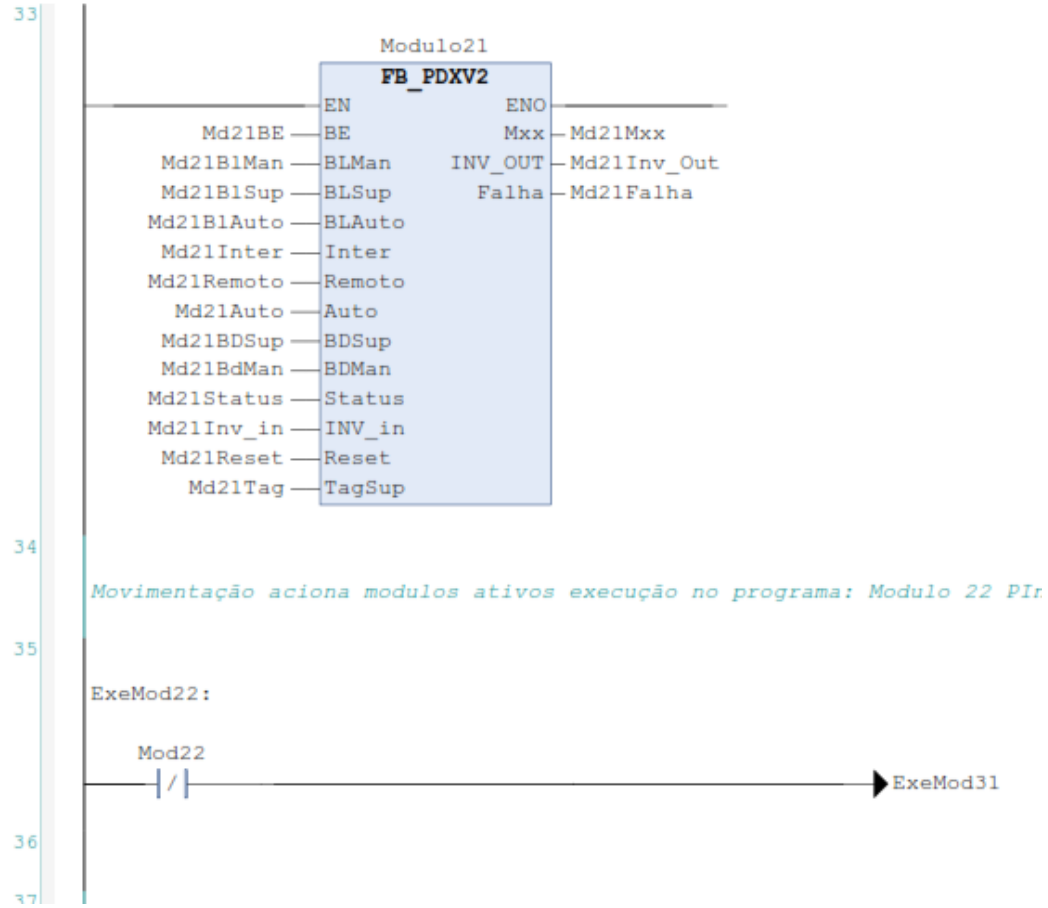

```
20
Movimentação aciona modulos ativos execução no programa: Modulo 11 PDXV2:
21
ExeMod11:
```

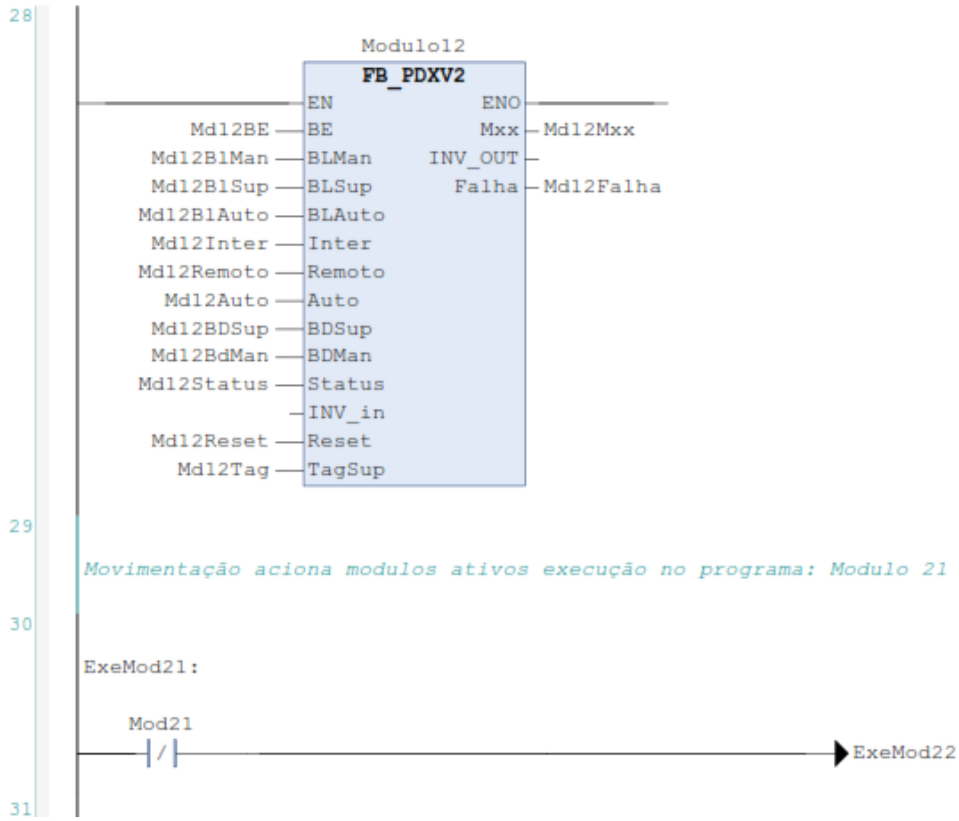




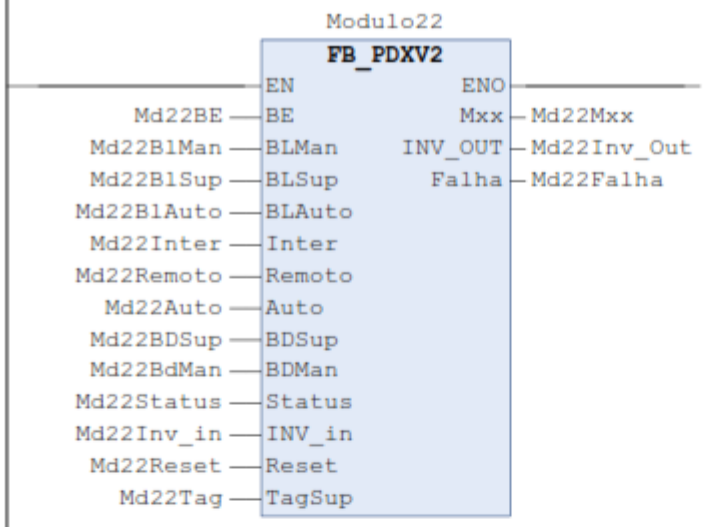

```
24
Movimentação aciona modulos ativos execução no programa: Modulo 12 PDXV2:
25
ExeMod12:
```


```
26
```


33
Modulo21
FB_PDXV2
EN
ENO
Md21BE
BE
Mxx
Md21Mxx
Md21BlMan
BLMan
INV_OUT
Md21Inv_Out
Md21BlSup
BLSup
Falha
Md21Falha
Md21BlAuto
BLAuto
Md21Inter
Inter
Md21Remoto
Remoto
Md21Auto
Auto
Md21BDSup
BDSup
Md21BdMan
BDMan
Md21Status
Status
Md21Inv_in
INV_in
Md21Reset
Reset
Md21Tag
TagSup
34
Movimentação aciona modulos ativos execução no programa: Modulo 22 PIr
35
ExeMod22:
Mod22
ExeMod31
36


28
Modulo12
FB_PDXV2
EN
ENO
Md12BE
BE
Mxx
Md12Mxx
Md12BlMan
BLMan
INV_OUT
Md12BlSup
BLSup
Falha
Md12Falha
Md12BlAuto
BLAuto
Md12Inter
Inter
Md12Remoto
Remoto
Md12Auto
Auto
Md12BDSup
BDSup
Md12BdMan
BDMan
Md12Status
Status
INV_in
Md12Reset
Reset
Md12Tag
TagSup
29
Movimentação aciona modulos ativos execução no programa: Modulo 21 PInv:
30
ExeMod21:
Mod21
ExeMod22
31

38



39

Movimentação aciona modulos ativos execução no programa: Modulo 31 PID:

40

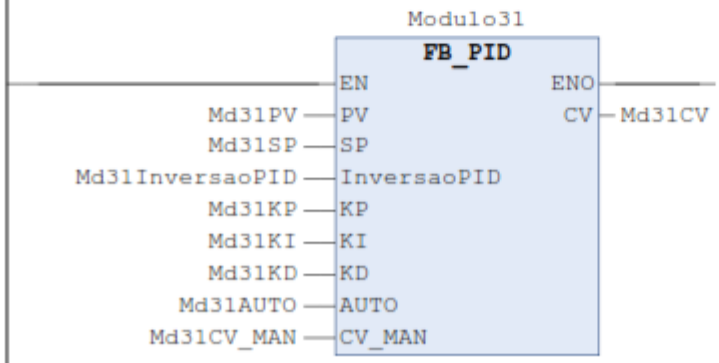
ExeMod31:

41

42

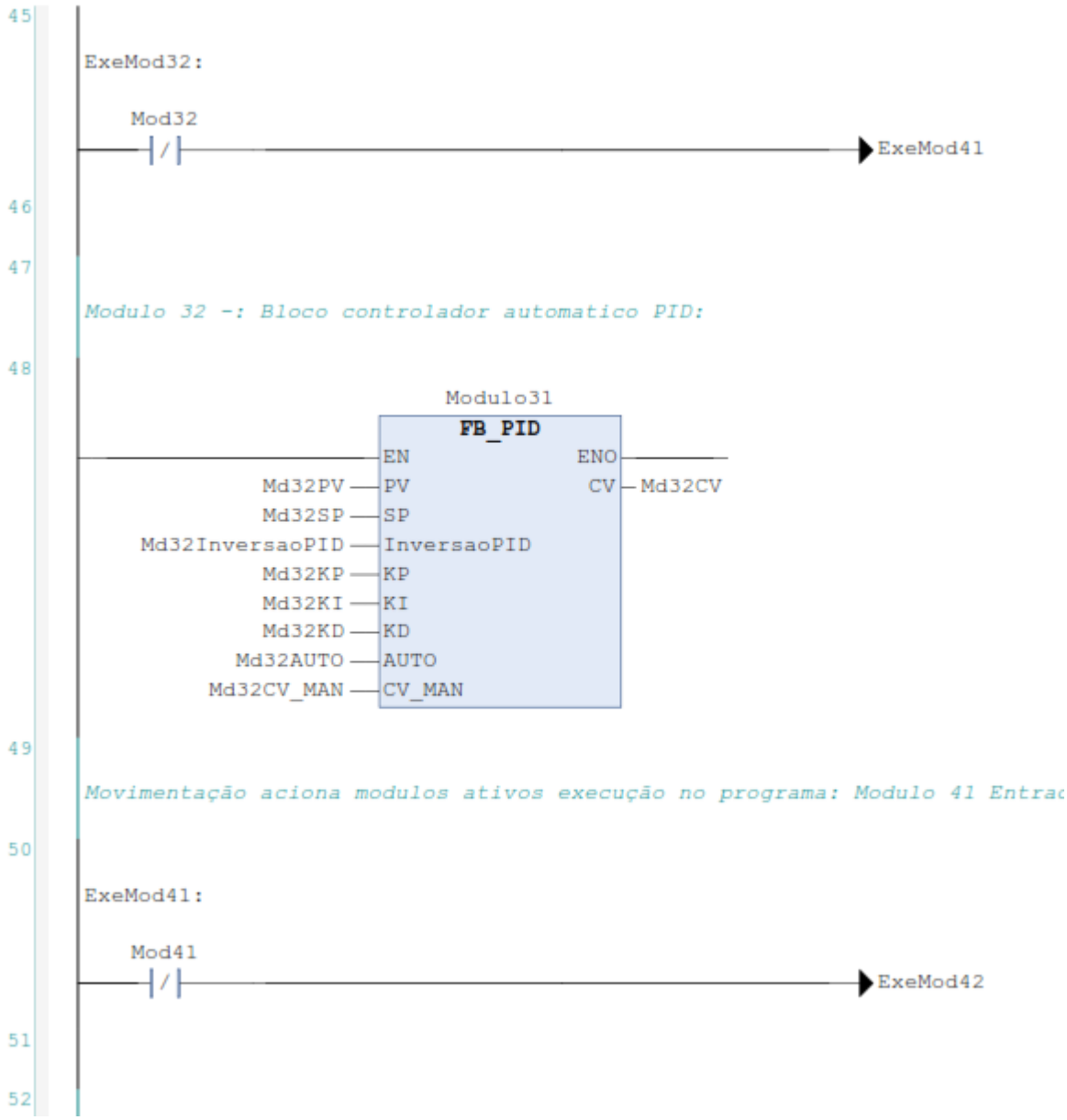
Modulo 31 -: Bloco controlador automatico PID:

43

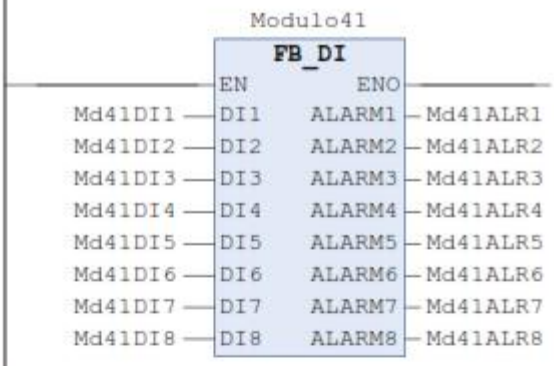


44

Movimentação aciona modulos ativos execução no programa: Modulo 32 PID:


45
ExeMod32:
Mod32
/
ExeMod41
46
47
Modulo 32 -: Bloco controlador automatico PID:
48
Modulo31
FB_PID
EN
ENO
Md32PV
PV
CV
Md32CV
Md32SP
SP
Md32InversaoPID
InversaoPID
Md32KP
KP
Md32KI
KI
Md32KD
KD
Md32AUTO
AUTO
Md32CV_MAN
CV_MAN
49
Movimentação aciona modulos ativos execução no programa: Modulo 41 Entrac
50
ExeMod41:
Mod41
/
ExeMod42
51
52

53



54

```
Movimentação aciona modulos ativos execução no programa: Modulo 41 Entrac
```

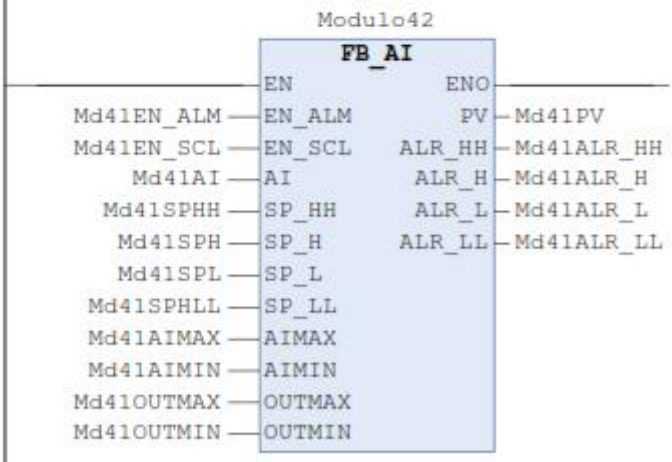
55

```
ExeMod42:
```

56

```
Modulo 42 -: Entrada Analogicas:
```

57



58

```
Fim da verificações e blocos executados:
```

59

```
Fim:
```

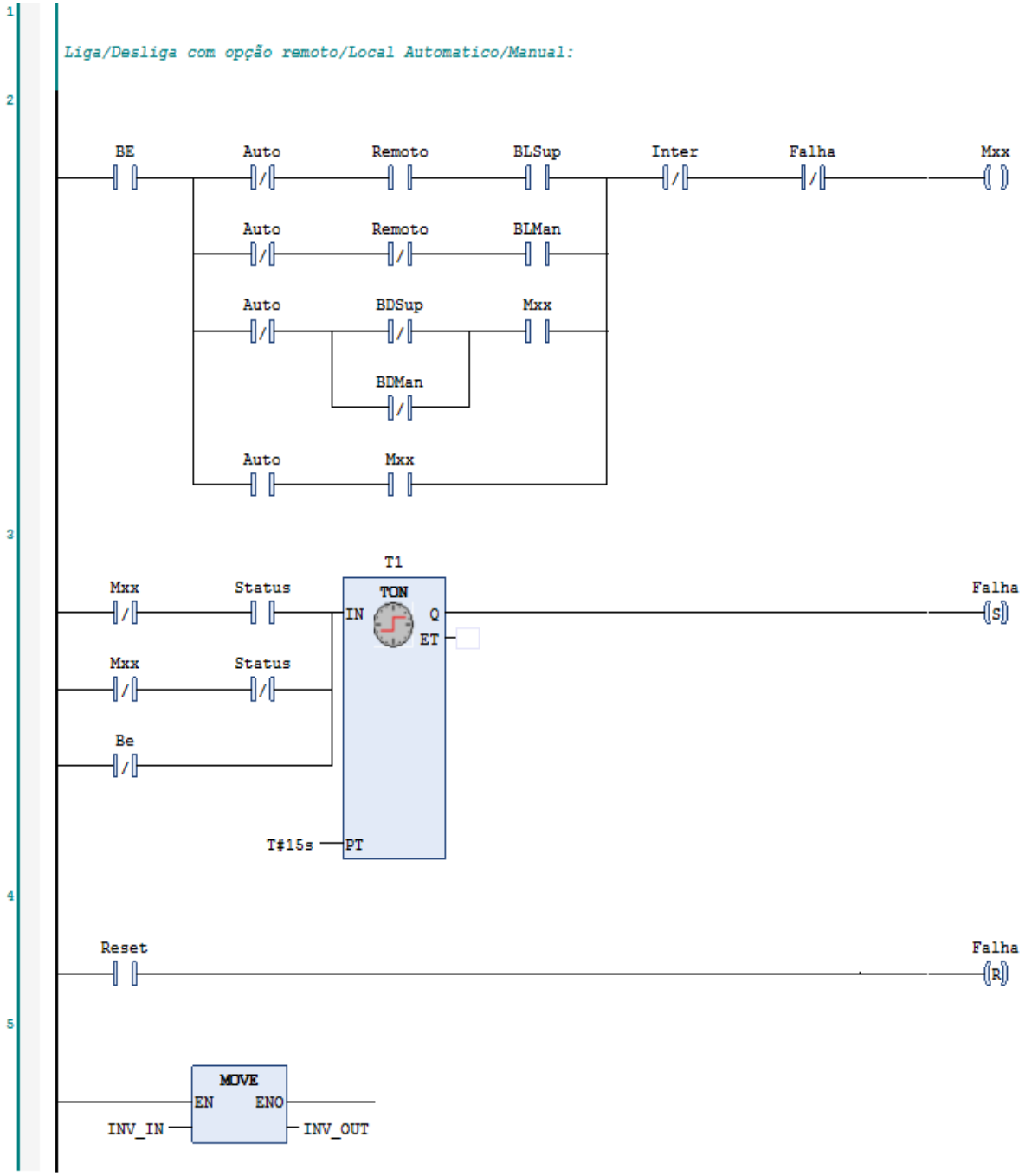
APÊNDICE L – PROGRAMA VERIFICAÇÃO ERROS

```
// Implicitly generated code : Only an Implementation suggestion
IF  index < lower THEN
        CheckBounds := lower;
ELSIF  index > upper THEN
        CheckBounds := upper;
ELSE
        CheckBounds := index;
END_IF
// Implicitly generated code : Only an Implementation suggestion
IF divisor = 0 THEN
        CheckDivDInt:=1;
ELSE
        CheckDivDInt:=divisor;
END_IF;
// Implicitly generated code : Only an Implementation suggestion
IF divisor = 0 THEN
        CheckDivLInt:=1;
ELSE
        CheckDivLInt:=divisor;
END_IF;
// Implicitly generated code : Only an Implementation suggestion
IF divisor = 0 THEN
        CheckDivReal:=1;
ELSE
        CheckDivReal:=divisor;
END_IF;
// Implicitly generated code : Only an Implementation suggestion
IF divisor = 0 THEN
        CheckDivLReal:=1;
ELSE
        CheckDivLReal:=divisor;
```

```
END_IF;
// No standard way of implementation. Fill your own code here
CheckPointer := ptToTest;
// Implicitly generated code : Only an Implementation suggestion
IF (value < lower) THEN
        CheckRangeSigned := lower;
ELSIF(value > upper) THEN
        CheckRangeSigned := upper;
ELSE
        CheckRangeSigned := value;
END_IF
// Implicitly generated code : Only an Implementation suggestion
IF (value < lower) THEN
        CheckRangeUnsigned := lower;
ELSIF(value > upper) THEN
        CheckRangeUnsigned := upper;
ELSE
        CheckRangeUnsigned := value;
END_IF
```

APÊNDICE M – PROGRAMA FB MODULOS COMPLEMENTARES

```
IF (PV >= SP_HH AND EN_ALM) THEN
        ALR_HH := TRUE;
        ELSE
        ALR_HH :=FALSE;
END_IF
IF (PV >= SP_H AND EN_ALM) THEN
        ALR_H := TRUE;
        ELSE
        ALR_H :=FALSE;
END_IF
IF (PV <= SP_L AND EN_ALM) THEN
        ALR_L := TRUE;
        ELSE
        ALR_L :=FALSE;
END_IF
IF (PV <= SP_LL AND EN_ALM) THEN
        ALR_LL := TRUE;
        ELSE
        ALR_LL :=FALSE;
END_IF
//Bloco linearizador da PV
IF (EN_SCL) THEN
        PV:= (((OUTMAX-OUTMIN)/(AIMAX-AIMIN))*AI)+(OUTMIN-((OUTMAX-
OUTMIN)/(AIMAX-AIMIN))*AIMIN);
        ELSE
        PV :=(AI/10) ;
END_IF
```


1
Liga/Desliga com opção remoto/Local Automatico/Manual:
2
BE
Auto
Remoto
BLSup
Inter
Falha
Mxx
Auto
Remoto
BLMan
Auto
BDSup
Mxx
BDMan
Auto
Mxx
3
T1
Mxx
Status
TON
IN
Q
ET
Falha
Mxx
Status
Be
T#15s
PT
4
Reset
Falha
5
MOVE
EN
ENO
INV_IN
INV_OUT

APÊNDICE N – VARIÁVEIS DO SISTEMA

```
VAR_GLOBAL
    Mod01: BOOL; //modulo 1 ativo - PDD + TQ
    Mod02: BOOL; //modulo 2 ativo - PDD + TQ
    Mod01TCP: BOOL; //modulo 1 confirmação tcp ativo - PD + TQ
    Mod02TCP: BOOL; //modulo 2 confirmação tcp ativo - PD + TQ
    Mod11: BOOL; //modulo 11 ativo - PDD x3
    Mod12: BOOL; //modulo 12 ativo - PDD x3
    Mod11TCP: BOOL; //modulo 11 confirmação tcp ativo - PDD x3
    Mod12TCP: BOOL; //modulo 12 confirmação tcp ativo - PDD x3
    Mod21: BOOL; //modulo 21 ativo - PINV
    Mod22: BOOL; //modulo 22 ativo - PINV
    Mod21TCP: BOOL; //modulo 21 confirmação tcp ativo - PINV
    Mod22TCP: BOOL; //modulo 22 confirmação tcp ativo - PINV
    Mod31: BOOL; //modulo 21 ativo - PID
    Mod32: BOOL; //modulo 22 ativo - PID
    Mod31TCP: BOOL; //modulo 21 confirmação tcp ativo - PID
    Mod32TCP: BOOL; //modulo 22 confirmação tcp ativo - PID
    Mod41: BOOL; //modulo 21 ativo - PID
    Mod42: BOOL; //modulo 22 ativo - PID
    Mod41TCP: BOOL; //modulo 21 confirmação tcp ativo - PID
    Mod42TCP: BOOL; //modulo 22 confirmação tcp ativo - PID
    //MD1 Modulo 1 PDTQ
    Md1BlMan: BOOL; // Liga Manual
    Md1BE: BOOL;  // Botao de emergencia
    Md1BlSup: BOOL; // Botao Liga Tela
    Md1BlAuto: BOOL; // Liga Automatico
    Md1Inter: BOOL;  //Intertravamento (*Implementando*)
    Md1Remoto: BOOL; //Local Remoto
    Md1BDSup: BOOL; // Botao desliga Supervisorio
    Md1BdMan: BOOL; // Botao desliga local
    Md1Status: BOOL; //Status de ligado
```

```
        Md1Mxx: BOOL; // Bobina rele
        Md1Falha: BOOL; //Falha
        Md1Tag: STRING; //TAG do motor
        Md1ControleAtiva: BOOL; //Ativa controle
        Md1Sp: INT; //Setpoint do nivel do tanque
        Md1PV: INT; // Variavel de processo do tanque
        Md1ControleAtivaAnalog: BOOL; //Ativa Controle analogico
        Md1EncheBBa: BOOL; //Tanque enche o tanque
        Md1SensorDigitalAlto: BOOL; //Sensor Nivel alto
        Md1SensorDigitalBaixo: BOOL; //Sensor nivel baixo
        Md1Reset: BOOL; // Reset
        Md1SpMaxPv: INT; // Setpoint maximo variavel de processo
        Md1AI: INT; // Entrada analogica
        Md1MaxAi: INT; //Maximo entrada analogica
        Md1Auto: BOOL; // Automatico ativa
    //MD2 Modulo 2 PDTQ
        Md2BlMan: BOOL; // Liga Manual
        Md2BE: BOOL;  // Botao de emergencia
        Md2BlSup: BOOL; // Botao Liga Tela
        Md2BlAuto: BOOL; // Liga Automatico
        Md2Inter: BOOL;  //Intertravamento (*Implementando*)
        Md2Remoto: BOOL; //Local Remoto
        Md2BDSup: BOOL; // Botao desliga Supervisorio
        Md2BdMan: BOOL; // Botao desliga local
        Md2Status: BOOL; //Status de ligado
        Md2Mxx: BOOL; // Bobina rele
        Md2Falha: BOOL; //Falha
        Md2Tag: STRING; //TAG do motor
        Md2ControleAtiva: BOOL; //Ativa controle
        Md2Sp: INT; //Setpoint do nivel do tanque
        Md2PV: INT; // Variavel de processo do tanque
        Md2ControleAtivaAnalog: BOOL; //Ativa Controle analogico
        Md2EncheBBa: BOOL; //Tanque enche o tanque
```

```
Md2SensorDigitalAlto: BOOL; //Sensor Nivel alto
Md2SensorDigitalBaixo: BOOL; //Sensor nivel baixo
Md2Reset: BOOL; // Reset
Md2SpMaxPv: INT; // Setpoint maximo variavel de processo
Md2AI: INT; // Entrada analogica
Md2MaxAi: INT; //Maximo entrada analogica
Md2Auto: BOOL; // Automatico ativa
Md11BlMan: BOOL; // Liga Manual
Md11BE: BOOL;  // Botao de emergencia
Md11BlSup: BOOL; // Botao Liga Tela
Md11BlAuto: BOOL; // Liga Automatico
Md11Inter: BOOL;  //Intertravamento (*Implementando*)
Md11Remoto: BOOL; //Local Remoto
Md11BDSup: BOOL; // Botao desliga Supervisorio
Md11BdMan: BOOL; // Botao desliga local
Md11Status: BOOL; //Status de ligado
Md11Mxx: BOOL; // Bobina rele
Md11Falha: BOOL; //Falha
Md11Tag: STRING; //TAG do motor
Md11ControleAtiva: BOOL; //Ativa controle
Md11Sp: INT; //Setpoint do nivel do tanque
Md11PV: INT; // Variavel de processo do tanque
Md11ControleAtivaAnalog: BOOL; //Ativa Controle analogico
Md11EncheBBa: BOOL; //Tanque enche o tanque
Md11SensorDigitalAlto: BOOL; //Sensor Nivel alto
Md11SensorDigitalBaixo: BOOL; //Sensor nivel baixo
Md11Reset: BOOL; // Reset
Md11SpMaxPv: INT; // Setpoint maximo variavel de processo
Md11AI: INT; // Entrada analogica
Md11MaxAi: INT; //Maximo entrada analogica
Md11Auto: BOOL; // Automatico ativa
Md12BlMan: BOOL; // Liga Manual
Md12BE: BOOL;  // Botao de emergencia
```

```
Md12BlSup: BOOL; // Botao Liga Tela
Md12BlAuto: BOOL; // Liga Automatico
Md12Inter: BOOL;  //Intertravamento (*Implementando*)
Md12Remoto: BOOL; //Local Remoto
Md12BDSup: BOOL; // Botao desliga Supervisorio
Md12BdMan: BOOL; // Botao desliga local
Md12Status: BOOL; //Status de ligado
Md12Mxx: BOOL; // Bobina rele
Md12Falha: BOOL; //Falha
Md12Tag: STRING; //TAG do motor
Md12ControleAtiva: BOOL; //Ativa controle
Md12Sp: INT; //Setpoint do nivel do tanque
Md12PV: INT; // Variavel de processo do tanque
Md12ControleAtivaAnalog: BOOL; //Ativa Controle analogico
Md12EncheBBa: BOOL; //Tanque enche o tanque
Md12SensorDigitalAlto: BOOL; //Sensor Nivel alto
Md12SensorDigitalBaixo: BOOL; //Sensor nivel baixo
Md12Reset: BOOL; // Reset
Md12SpMaxPv: INT; // Setpoint maximo variavel de processo
Md12AI: INT; // Entrada analogica
Md12MaxAi: INT; //Maximo entrada analogica
Md12Auto: BOOL; // Automatico ativa
Md21BlMan: BOOL; // Liga Manual
Md21BE: BOOL;  // Botao de emergencia
Md21BlSup: BOOL; // Botao Liga Tela
Md21BlAuto: BOOL; // Liga Automatico
Md21Inter: BOOL;  //Intertravamento (*Implementando*)
Md21Remoto: BOOL; //Local Remoto
Md21BDSup: BOOL; // Botao desliga Supervisorio
Md21BdMan: BOOL; // Botao desliga local
Md21Status: BOOL; //Status de ligado
Md21Mxx: BOOL; // Bobina rele
Md21Falha: BOOL; //Falha
```

Md21Tag: STRING; //TAG do motor
Md21ControleAtiva: BOOL; //Ativa controle
Md21Sp: INT; //Setpoint do nivel do tanque
Md21PV: INT; // Variavel de processo do tanque
Md21ControleAtivaAnalog: BOOL; //Ativa Controle analogico
Md21EncheBBa: BOOL; //Tanque enche o tanque
Md21SensorDigitalAlto: BOOL; //Sensor Nivel alto
Md21SensorDigitalBaixo: BOOL; //Sensor nivel baixo
Md21Reset: BOOL; // Reset
Md21SpMaxPv: INT; // Setpoint maximo variavel de processo
Md21AI: INT; // Entrada analogica
Md21MaxAi: INT; //Maximo entrada analogica
Md21Auto: BOOL; // Automatico ativa
Md21Inv_in: INT; // Variavel entrada Inversor
Md21Inv_Out: INT; // Variavel Saida Inversor
Md22BlMan: BOOL; // Liga Manual
Md22BE: BOOL; // Botao de emergencia
Md22BlSup: BOOL; // Botao Liga Tela
Md22BlAuto: BOOL; // Liga Automatico
Md22Inter: BOOL; //Intertravamento (*Implementando*)
Md22Remoto: BOOL; //Local Remoto
Md22BDSup: BOOL; // Botao desliga Supervisorio
Md22BdMan: BOOL; // Botao desliga local
Md22Status: BOOL; //Status de ligado
Md22Mxx: BOOL; // Bobina rele
Md22Falha: BOOL; //Falha
Md22Tag: STRING; //TAG do motor
Md22ControleAtiva: BOOL; //Ativa controle
Md22Sp: INT; //Setpoint do nivel do tanque
Md22PV: INT; // Variavel de processo do tanque
Md22ControleAtivaAnalog: BOOL; //Ativa Controle analogico
Md22EncheBBa: BOOL; //Tanque enche o tanque
Md22SensorDigitalAlto: BOOL; //Sensor Nivel alto

```
Md22SensorDigitalBaixo: BOOL; //Sensor nivel baixo
Md22Reset: BOOL; // Reset
Md22SpMaxPv: INT; // Setpoint maximo variavel de processo
Md22AI: INT; // Entrada analogica
Md22MaxAi: INT; //Maximo entrada analogica
Md22Auto: BOOL; // Automatico ativa
Md22Inv_in: INT; // Variavel entrada Inversor
Md22Inv_Out: INT; // Variavel Saida Inversor
Md31PV        :INT;  //VARIAVEL processo PID
Md31SP        :INT;  // VARIAVEL SETPOINT PID
Md31InversaoPID     :BOOL; // INVERSAO SAIDA DO PID
Md31KP        :REAL; // PROPORCIONAL
Md31KI        :REAL; // INTEGRAL
Md31KD: REAL; // DERIVATIVA
Md31AUTO: BOOL; // AUTOMATICO
Md31CV_MAN: INT; // CV EM MANUAL
Md31CV: REAL; // CV ESCRITA
Md32PV        :INT;  //VARIAVEL processo PID
Md32SP        :INT;  // VARIAVEL SETPOINT PID
Md32InversaoPID     :BOOL; // INVERSAO SAIDA DO PID
Md32KP        :REAL; // PROPORCIONAL
Md32KI        :REAL; // INTEGRAL
Md32KD: REAL; // DERIVATIVA
Md32AUTO: BOOL; // AUTOMATICO
Md32CV_MAN: INT; // CV EM MANUAL
Md32CV: REAL; // CV ESCRITA
Md41DI1: BOOL;
Md41DI2: BOOL;
Md41DI3: BOOL;
Md41DI4: BOOL;
Md41DI5: BOOL;
Md41DI6: BOOL;
Md41DI7: BOOL;
```

```
Md41DI8: BOOL;
Md41ALR1: BOOL;
Md41ALR2: BOOL;
Md41ALR3: BOOL;
Md41ALR4: BOOL;
Md41ALR5: BOOL;
Md41ALR6: BOOL;
Md41ALR7: BOOL;
Md41ALR8: BOOL;
Md41EN_Alm: BOOL;
Md41EN_SCL: BOOL;
Md41AI: REAL;
Md41SPHH: REAL;
Md41SPH: REAL;
Md41SPL: REAL;
Md41SPHLL: REAL;
Md41AIMAX: REAL;
Md41AIMIN: REAL;
Md41outMAX: REAL;
Md41OUTMIN: REAL;
Md41PV: REAL;
Md41ALR_HH: BOOL;
Md41ALR_H: BOOL;
Md41ALR_L: BOOL;
Md41ALR_LL: BOOL;
Md1ReadMd1 AT %MW100 :INT;
Md1ReadA0 AT %MW101 :INT;
Md1WritePWM  AT %MW102       :INT;
Md1Read2     :BOOL;
Md1Read3     :BOOL;
Md1Read5     :BOOL;
Md1Read7     :BOOL;
Md1Read8     :BOOL;
```

```
Md1Write9     :BOOL;
Md1XVPD: BOOL;
Md1Graph: BOOL;
Md1Ativar: BOOL;
Md1ReadMd1 AT %MW103 :INT;
Md1ReadA0 AT %MW104 :INT;
Md1WritePWM  AT %MW105       :INT;
Md1Read2      :BOOL;
Md1Read3      :BOOL;
Md1Read5      :BOOL;
Md1Read7      :BOOL;
Md1Read8      :BOOL;
Md1Write9     :BOOL;
Md2XVPD: BOOL;
Md2Graph: BOOL;
Md2Ativar: BOOL;
```