UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
CURSO DE ENGENHARIA INDUSTRIAL ELÉTRICA/ELETROTÉCNICA

ALBERTO AKIRA INOUE
PIATAN SFAIR PALAR

INTEROPERABILIDADE DE REDES INDUSTRIAIS NO BARRAMENTO PROFIBUS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2016

ALBERTO AKIRA INOUE
PIATAN SFAIR PALAR

INTEROPERABILIDADE DE REDES INDUSTRIAIS NO BARRAMENTO PROFIBUS

Trabalho de Conclusão de Curso de Graduação do curso de Engenharia Industrial Elétrica: Ênfase em Eletrotécnica do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do titulo de engenheiro.

Orientador: Prof. Msc. Daniel Balieiro Silva

CURITIBA
2016

Alberto Akira Inoue
Piatan Sfair Palar

Interoperabilidade de Redes Industriais no Barramento PROFIBUS

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro Eletricista, do curso de Engenharia Elétrica do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 29 de novembro de 2016.

_____
Prof. Emerson Rigoni, Dr.
Coordenador de Curso
De Engenharia Elétrica

_____
Profa. Annemarlen Gehrke Castagna, Ma.
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia Elétrica do DAELT

ORIENTAÇÃO

_____
Daniel Balieiro Silva, Me.
Universidade Tecnológica Federal do Paraná
Orientador

BANCA EXAMINADORA

_____
Jorge Assade Leludak, Dr.
Universidade Tecnológica Federal do Paraná

_____
José da Silva Maia, Me.
Universidade Tecnológica Federal do Paraná

_____
Vilmair Ermenio Wirmond, Me.
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Elétrica.

RESUMO

INOUE, Alberto A.; PALAR, Piatan S. **Interoperabilidade de redes industriais no barramento PROFIBUS.** 2016. 113 f. Trabalho de Conclusão de Curso (Graduação) – Universidade Tecnológica Federal do Paraná, Curso Superior de Engenharia Elétrica, Curitiba, 2016.

Este trabalho se inicia com um breve histórico dos Controladores Lógicos Programáveis e Redes Industriais, e a teoria do funcionamento dos mesmos. Em seguida apresenta um manual didático para a programação e configuração de uma rede PROFIBUS contendo um CLP Siemens S7-300 e uma Interface Homem-Máquina Eaton XV, utilizando os *softwares* STEP7 v5.5 e Galileo v7.2.8. Em uma primeira parte são descritos os procedimentos necessários para a criação de um novo projeto de Controlador Lógico Programável, com informações sobre os diferentes blocos de organização, bem como a configuração de *hardware* do CLP e da rede PROFIBUS no programa STEP7. É também informado um passo-a-passo sobre como utilizar o simulador que vem incorporado ao STEP7 e os procedimentos para *download* de um projeto através da interface *Simatic USB PC Adapter*. A segunda parte do manual contém informações detalhadas para, através do *software* Galileo, criar-se um novo projeto nas Interfaces Homem-Máquina da fabricante Eaton, configurar uma rede PROFIBUS, associar novas *tags* com as memórias do CLP através de *Data Blocks (DB)*, criar telas que permitam a visualização e manipulação destas *tags* e fazer a transferência destas telas para a IHM através de uma rede Ethernet e do aplicativo *FTP Server* na IHM. Este trabalho contém também dois exemplos de programas comentados, na parte de apêndices.

Palavras-Chave: Redes Industriais. PROFIBUS. Controlador Lógico Programável. Interface Homem-Máquina. Manual Didático.

ABSTRACT

INOUE, Alberto A.; PALAR, Piatan S. **Interoperability of Industrial Networks using PROFIBUS.** 2016. 113f. Trabalho de Conclusão de Curso (Graduação) – Universidade Tecnológica Federal do Paraná, Curso Superior de Engenharia Elétrica, Curitiba, 2016.

This paper starts with a brief background on programmable logic controllers, industrial networking and how they operate together. It also presents an user manual for programming and configuring a PROFIBUS network consisting of a Siemens S7-300 programmable logic controller and a Eaton XV human-machine interface, with programming softwares Step7 v5.5 and Galileo v7.2.8. The first section of the user manual describes the procedures needed for creating a new PLC project, with information regarding different organization blocks, PLC hardware and PROFIBUS configuration in Step7. An step-by-step guide for using the Step7 Simulator is also provided, as well as guidelines for downloading a new project using a Simatic USB PC Adapter. The second section of the user manual contain detailed information for creating a new HMI project, setting up a PROFIBUS network, linking user created tags with Data Blocks, creating masks and downloading projects on Eaton software Galileo. This paper also provides as appendix two example projects.

Key Words: Industrial Networking. PROFIBUS. Programmable Logic Controller. Human-Machine Interface. User Manual.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

CAN	*Controller Area Network*
CLP	Controlador Lógico Programável
CPU	Unidade de Processamento Central
DB	*Data Block*
FB	*Function Block*
FC	*Function*
FTP	*File Transfer Protocol*
HART	*Highway Addressable Remote Transducer*
HMI	*Human-Machine Interface*
IHM	Interface Homem-Máquina
IP	*Internet Protocol*
ISO	*International Standards Organization*
LAN	*Local Area Network*
MAN	*Metropolitan Area Network*
OB	*Organization Block*
OSI	*Open Systems Interconnection*
MPI	Multi-Point Interface
UDT	*User Data Type*
VAT	*Variable Table*
WAN	*Wide Area Network*

SUMÁRIO

1. INTRODUÇÃO

1.1. TEMA

Os sinais pneumáticos foram os primeiros a serem utilizados como forma de transmissão de dados em plantas industriais, ainda na década de 1930. Apenas no início da década de 1960 foi surgindo a comunicação através dos sinais elétricos, o que reduziu o custo de instalação da rede, e também o tempo de transmissão dos sinais (GUTIERREZ: PAN, 2008).

Os sistemas de controle iniciaram na década de 1960 com controladores de uma única malha de realimentação (*Single-Loop Controllers - SLC).* Passaram para sistemas baseados em computadores digitais (*Direct Digital Controller - DDC)*, pelos Sistemas de Controle Distribuído (DCS) e, na década de 1970, surgiram os Sistemas Digitais de Controle Distribuído (SDCD), que são implementados com computadores que realizam o controle, supervisão e configuração dos dispositivos de entrada e saída. Sua arquitetura básica é formada pelas estações locais de interface com o processo, interfaces homem-máquina e redes de comunicação. (ALBUQUERQUE, 2009).

1.2. DELIMITAÇÃO DO TEMA

Através das pesquisas feitas será escrito um manual didático para operar a bancada, que possui CLP Siemens S7-300 comunicando com a IHM Eaton XV-102-B2-35TQR-10, através do protocolo PROFIBUS. Também conterá dois exemplos de programas feitos no *software* STEP7 e no *software* da IHM Galileo.

1.3. PROBLEMA E PREMISSA

A grande variedade de fabricantes ligados ao controle e supervisão de processos existentes no mercado mundial possibilita o usuário escolher dispositivos que mais se adequem aos requisitos da planta. Por este motivo, em muitas plantas industriais, os equipamentos não são da mesma fabricante. Para fazer a

comunicação entre esses dispositivos, faz-se o uso de protocolos de redes industriais abertos. Se por um lado, isso beneficia o cliente, por outro acarreta um problema, que seria a complexidade em fazer a comunicação entre equipamentos de diferentes fabricantes.

Como fazer a comunicação de tais dispositivos, de forma a aproximar o conteúdo acadêmico com a realidade em plantas industriais?

1.4. OBJETIVOS

1.4.1. Objetivo geral

Desenvolver um material didático para a bancada que contendo CLP Siemens S7-300 comunicando com a IHM Eaton XV-102-B2-35TQR-10.

1.4.2. Objetivos específicos

- Levantar as funcionalidades e itens da bancada
- Compreender a plataforma de programação Simatic;
- Compreender a plataforma de programação Galileo*;*
- Desenvolver dois programas exemplos no CLP e suas respectivas telas do supervisório;
- Testar comunicação entre o CLP e IHM através da rede PROFIBUS;
- Escrever um manual didático;

1.5. JUSTIFICATIVA

Com a crescente demanda de engenheiros que estejam capacitados a desenvolver e atuar sistemas de controle e automação faz-se necessário que os discentes tenham contatos com softwares e hardwares atuais. A universidade disponibilizou um laboratório completo e pronto para uso, contendo bancadas com

CLPs e IHMs, que os alunos podem usufruir. Entretanto devido a falta de manuais explicando a comunicação entre os componentes não é possível a utilização plena dos mesmos.

A fim de solucionar o problema descrito, este projeto visa elaborar um material didático especifico para a bancada do laboratório.

1.6. PROCEDIMENTOS METODOLÓGICOS

Esse trabalho possui um caráter teórico, explorando através de livros, manuais de instruções, trabalhos de conclusão de curso, artigos e internet, o funcionamento do controlador lógico programável Siemens S7-300 e sua comunicação com a IHM Eaton XV-102-B2-35TQR-10.

1.7. ESTRUTURA DO TRABALHO

O trabalho será apresentado em quatro capítulos:

- Capitulo 1 – Introdução ao tema, apresentando o tema, os problemas e premissas, objetivos, justificativas e estrutura do trabalho.
- Capitulo 2 – Referencial Teórico, apresentando o embasamento teórico deste trabalho;
- Capitulo 3 – Manual Didático da Bancada, apresentando o manual de programação e configuração do Controlador Lógico Programável e da Interface Homem-Máquina.
- Capitulo 4 – Considerações Finais, apresentando as conclusões acerca do trabalho apresentado.
- Apêndices – Apresentando dois programas exemplos, do *software* STEP7 e o manual.

2. REFERENCIAL TEÓRICO

2.1. CONTROLADORES LÓGICOS PROGRAMÁVEIS

2.1.1. Definições

O controlador lógico programável pode ser definido como um equipamento eletrônico capaz de desempenhar diversas funções, como lógica, sequenciamento, registros, para controlar vários processos (ROQUE, 2014).

Os primeiros sistemas de controle eram mecânicos e foram desenvolvidos no final do século XIX, com a função de automatizar os processos de fabricação da época. Durante o início do século XX, os dispositivos mecânicos foram substituídos por relés, pois permitiam funções de controle mais complexas e sofisticadas; (ROQUE, 2014)

Com o desenvolvimento de circuitos integrados baseados em tecnologias TTL (*Transistor-Transistor Logic*) ou CMOS (*Complementary Metal Oxide Semiconductor),* possibilitou um novo tipo de sistema de controle baseados em CLPs (ROQUE, 2014).

Os primeiros controladores possuíam baixa capacidade de processamento e seu uso era restrito a máquinas e processos com operações repetitivas. Com o surgimento dos microprocessadores, os controladores aumentaram sua velocidade de processamento e apresentaram grande flexibilidade de programação (SOUZA, 2001).

2.1.2. Arquitetura do CLP

Segundo Prudente (2013), os CLPs são formados por três componentes básicos:

- Unidade central de processamento;
- Unidade de entrada e saída (Unidade I/O);
- Unidade de programação.

A figura 1 mostra os componentes básicos do CLPs:

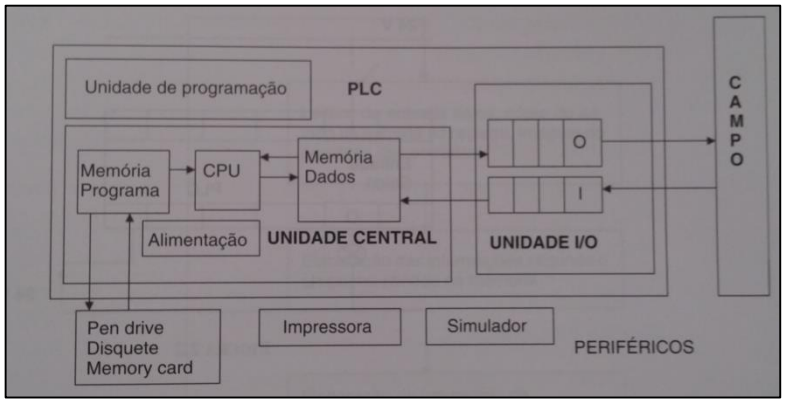


Figura 1 - Componentes básicos do Controlador Lógico Programável.
Fonte: Prudente (2013).

2.1.2.1. Unidade Central de Processamento

A unidade central organiza as funções de controle do CLP através do processador e da memória. (GEORGINI, 2003).

O processador é o dispositivo principal do CLP. O processador executa uma lista de operações lógicas, de acordo com uma sequência pré-estabelecida, chamado de programa. Através de lógicas binárias, o processador pode comandar máquinas enviando sinais elétricos, como se faz em eletromecânica com circuitos elétricos. (PRUDENTE, 2013)

A memória da unidade central é divida em Memória do Sistema de Operação e Memória de Aplicação (GEORGINI, 2003). A figura 2 mostra a estrutura do sistema de memória:

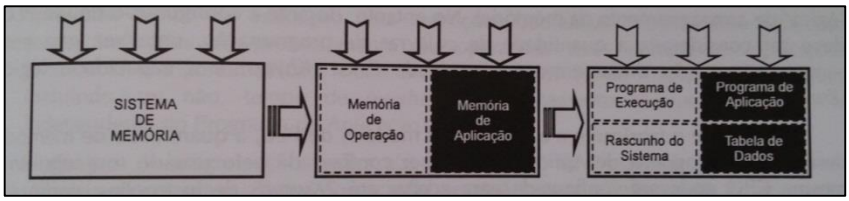


Figura 2 - Estrutura Básica do Sistema de Memória.
Fonte: Georgini (2003).

Na Memória do Sistema de Operação se encontra o programa de execução, desenvolvido pelo fabricante, que fixa como o sistema deve operar e o rascunho do sistema, memória reservada para o armazenamento temporário de dados. (GEORGINI, 2003)

Na Memória de Aplicação se encontra o programa de aplicação, onde o programa desenvolvido pelo usuário é armazenado, e a tabela de dados, área onde os dados utilizados pelo programa de aplicação são mantidos. (GEORGINI, 2003)

2.1.2.2. Unidade de entrada e saída

A unidade de entrada e saída é responsável pela comunicação da unidade de processamento com o meio externo (GEORGINI, 2003). De forma geral, pode ser dividida entre módulo de entrada, que recebe sinais de entrada oriundos de sensores e chaves, e módulo de saída, que emitem sinais da CPU para dispositivos de saída. A figura 3 mostra a interação das interfaces de entrada e saída de dados:

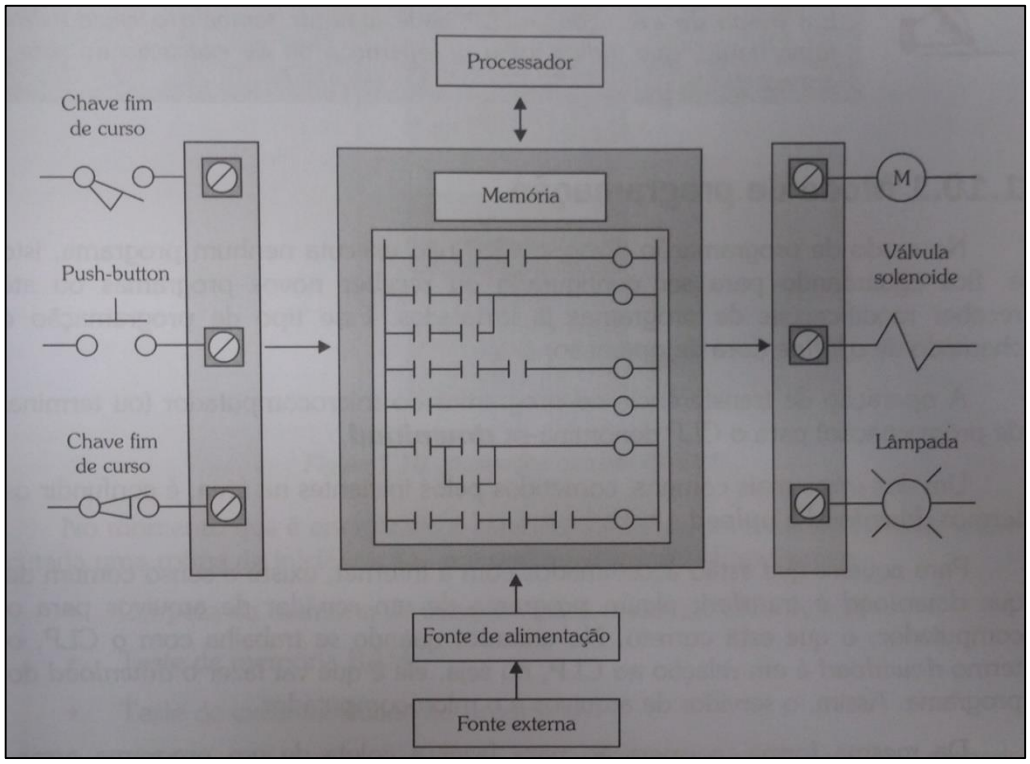


Figura 3 - Interação das interfaces de entrada/saída de dados.
Fonte: Franchi; Camargo (2008).

Os sinais da unidade podem ser digitais ou analógicos. Os sinais digitais podem assumir dois estados: *on* (quando há presença de tensão) e *off* (quando não há presença de tensão) (PRUDENTE, 2013). Então no CLP não existe contato físico aberto ou fechado, mas apenas o nível de tensão no contato. (PRUDENTE, 2013).

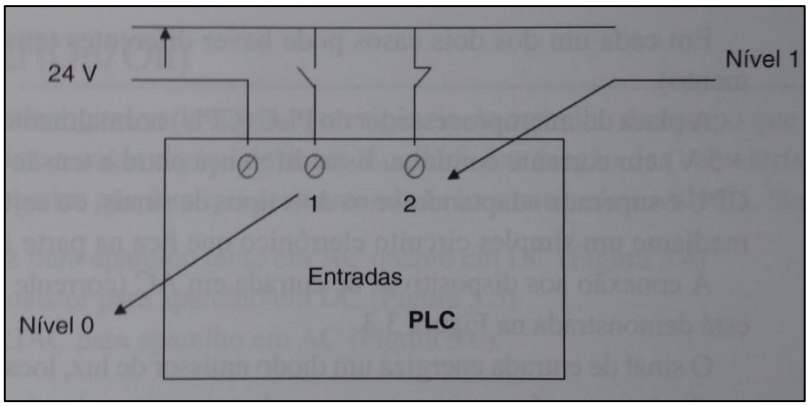


Figura 4 - Contatos do CLP.
Fonte: Prudente (2013).

Os sinais analógicos podem assumir qualquer valor dentro de um limite inferior e superior (*range).* Devido à variedade de valor, o tratamento desses sinais são mais difíceis sendo necessário o uso de memórias de 16 e 32 *bits*. (PRUDENTE, 2013)

2.1.2.3. Unidade de programação

A unidade de programação é o dispositivo que se encontra a interface Homem/Máquina do CLP. (PRUDENTE, 2013). Pode ser um computador ou teclado, onde é escrito o programa na memória do CLP. (PRUDENTE, 2013)

2.1.3. Controlador Lógico Programável Siemens S7-300

2.1.3.1. Visão geral do Sistema de automação SIMATIC S7-300

O SIMATIC S7-300 é um CLP Modular para sistemas de baixo ou médio desempenho. Dependendo dos requisitos do projeto, o Programador pode ser

expandido com módulos de entrada/saída, para sinais analógicos e digitais, em até quatro *racks* com oito módulos cada. (BERGER, 2014)

As expansões são possíveis através da comunicação PROFIBUS ou PROFINET. O projeto especial dos módulos permite que sejam instalados diretamente no local das máquinas ou na planta. (BERGER, 2014)

O programa utilizado para configurar e programar os controladores é o STEP7. A troca de dados entre os controladores é feita através do SIMATIC NET. (BERGER, 2014)

2.1.3.2. Componentes do SIMATIC S7-300

Os componentes mais importantes do CLP S7-300 estão mostradas na figura 5:

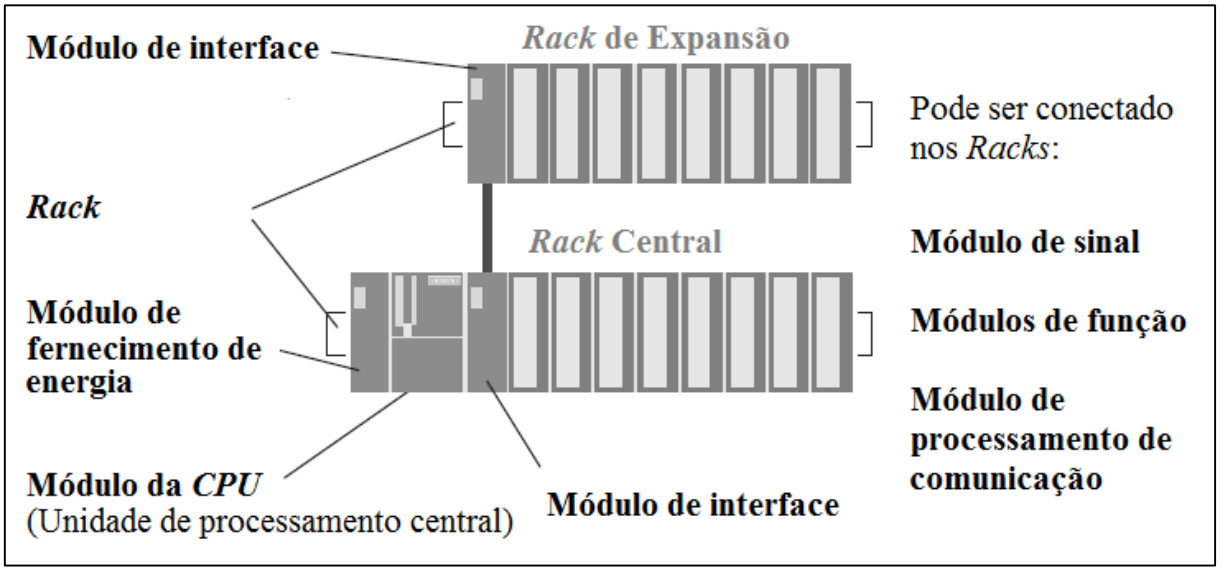


Figura 5 - Componentes do controlador S7-300.
Fonte: Adaptado de BERGER (2014).

O sistema operacional e programas do usuário se localizam na CPU. Os programas do usuário também são guardados em um cartão de memória micro, que é inserido na CPU. O barramento de interface presente na CPU estabelece a comunicação com outros CLPs. (BERGER,2014)

Os módulos de sinal são responsáveis pela conexão com a planta controlada. Os sinais de entrada/saídas podem ser analógicos ou digitais. (BERGER, 2014)

Os módulos de função são responsáveis pelo processamento de funções que a CPU não consegue executar rápido suficiente, como contar pulsos, posicionamento ou controlar *drivers*.

Os módulos de processamento de comunicação permitem um tráfego maior de dados, em relação às interfaces padrões.

O módulo de interface possibilita expandir o *rack* central em até três *racks* de expansão adicionais.

O módulo de fornecimento de energia fornece a tensão necessária para o funcionamento do CLP. (BERGER, 2014)

2.2. REDES INDUSTRIAIS

Segundo Albuquerque (2009), os computadores eram utilizados de uma maneira isolada, sem o compartilhamento de seus recursos entre usuários remotos até o início da década de 1960. Com os avanços tecnológicos na área principalmente de telecomunicação, tornou-se possível o advento das redes industriais, com o processamento de dados remotamente.

2.2.1. Arquitetura de Redes Industriais

As redes industriais podem se caracterizar por diferentes configurações e topologias. As topologias mais utilizadas são: em anel, em barramento, em satélite e em estrela, cada qual com suas vantagens e desvantagens (STEMMER, 2001). Podem também diferir quanto a seu meio físico de transmissão, que podem se separar segundo Albuquerque (2009) nos seguintes tipos: par trançado, cabo coaxial, fibra óptica, transmissão sem fio, *spread spectrum,* modem, sistema de telefonia móvel e *bluetooth*.

Também pode ser aplicado um critério de classificação das redes em relação à distância entre os usuários interligados. Conforme Albuquerque (2009) esta classificação é dada da seguinte maneira:

- Redes Locais - LAN (*Local Area Network):* geralmente um escritório ou uma empresa com os equipamentos não longes entre si;
- Redes Metropolitanas - MAN (*Metropolitan Area Network):* compreendidas em uma região metropolitana, de uma cidade;
- Redes Geograficamente Distribuídas - WAN (*Wide Area Network):* interligando cidades, estados ou países.

Como em muitos sistemas o gerenciamento de processos é dado de forma integrada, ocorre a utilização de vários níveis de rede e em cada um destes níveis, um protocolo de rede diferente. Como exemplo, os sensores e atuadores de um sistema podem se comunicar com os CLP através de um barramento CAN, os CLP podem se comunicar entre si através de um barramento PROFIBUS e com um *gateway* através do protocolo *Ethernet* (ALBUQUERQUE, 2009).

Segundo Stemmer (2001), os sistemas distribuídos e interconectados estão presentes em diversas aplicações, como controle de processos, automação, aplicações de gerenciamento bancário, correio eletrônico, entre outros.

As atuais redes industriais trouxeram muitas vantagens para a automação, tais como (GUTIERREZ; PAN, 2008):

- Redução dos custos com instalação da rede;
- Configuração, diagnóstico e detecção de falhas através do acesso remoto;
- Distribuição de tarefas de controle a dispositivos de campo;
- Troca de informações entre os dispositivos conectados ao mesmo barramento em tempo real.

Algumas características de redes industriais é que necessitam de maior qualificação dos usuários e das equipes de manutenção, além das redes precisarem atender requisitos rigorosos de resistência mecânica, operarem em ambientes agressivos e corrosivos, em altas temperaturas e com altas interferências eletromagnéticas (GUTIERREZ: PAN, 2008).

2.2.1.1. Padrão OSI/ISSO

Dada a diversidade dos equipamentos e das soluções referentes a comunicação, uma padronização fez-se necessária. Neste intuito, a ISO (*International Standards Organization)* trabalhou para criar um sistema de referência, sobre o qual seriam baseadas as arquiteturas das redes de comunicação, permitindo a interoperabilidade entre equipamentos heterogêneos. Através de uma série de documentos, a ISO publicou um Modelo de Referência para a Interconexão de Sistemas Abertos (*Reference Model for Open Systems Interconnection - RM-OSI)* (STEMMER, 2001).

Segundo Albuquerque (2009), o modelo OSI é composto por sete camadas bem definidas, com os protocolos de cada camada se comunicando independentemente das outras camadas.

A figura 6 mostra a arquitetura do modelo OSI:

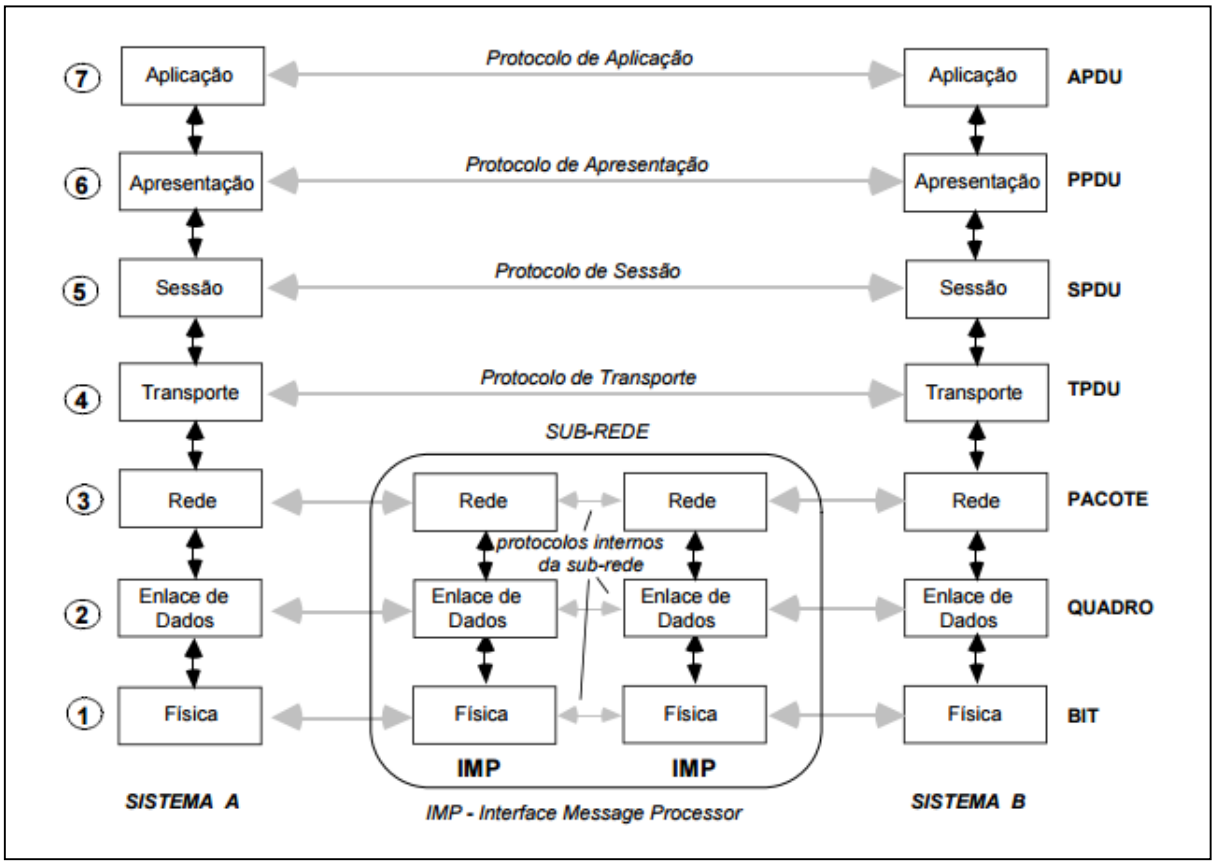


Figura 6 - Arquitetura a sete camadas do modelo OSI.

Fonte: Stemmer (2001).

2.2.2. Protocolos Industriais de Comunicação

As redes industriais que surgiram paralelamente às em ambientes administrativos são denominadas barramentos de campo (*fieldbuses)* e compartilham algumas semelhanças com as redes administrativas, porém se diferenciam nos aspectos de confiabilidade e necessidade de operação em tempo real. O conjunto de convenções e procedimentos que indicam a transmissão de dados entre vários equipamentos é chamado de protocolo (ALBUQUERQUE, 2009).

Atualmente operam no mercado protocolos digitais industriais proprietários e abertos. Aqueles foram os primeiros a surgir, pelas próprias empresas fabricantes dos dispositivos utilizados. Já os abertos foram desenvolvidos por instituições internacionais compostas por associações de fabricantes, e beneficiaram os usuários que podem utilizar dispositivos diferentes concomitantemente em uma mesma planta ou processo, bem como fabricantes de pequeno porte, que podem vender seus pequenos pacotes para a automação em conjunto com grandes fornecedores (GUTIERREZ; PAN, 2008).

Em uma pesquisa realizada nos Estados Unidos e divulgada na revista *InTech*, da ISA (*The Instrumentation Systems and Automation Society*), podemos ver a base instalada na época dos protocolos industriais:

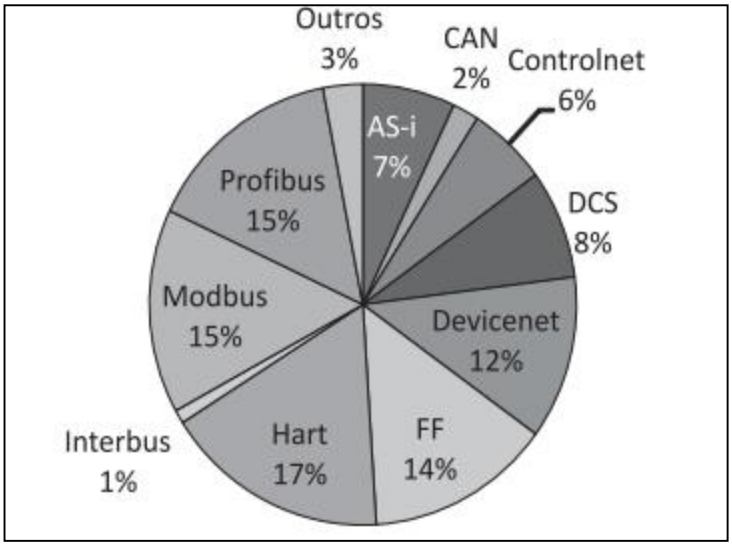


Gráfico 1 - Base instalada de protocolos em 2007 nas indústrias nos EUA.

Fonte: Gutierrez; Pan (2008).

2.2.2.1. MODBUS

O protocolo MODBUS foi criado em 1978 pela empresa *MODICON*, que hoje é conhecida como *Schneider Automation*. Inicialmente era um protocolo proprietário, mas com o tempo tornou-se aberto e é utilizado por diversos fabricantes como uma maneira simples de transferir dados entre dispositivos através, inicialmente da porta serial RS232, mas hoje também das portas RS422 e RS485, sobre vários meios de transmissão. Utiliza o modelo mestre-escravo (STEMMER, 2001).

2.2.2.2. FIELDBUS FOUNDATION *(FF)*

O *FOUNDATION* utiliza características de outros padrões, mas também traz alguns diferenciais como o *Link Active Scheduler* (LAS), uma entidade que realiza a gerência de troca de mensagens entre diversos equipamentos, como sensores e atuadores, CLP e também indicadores dedicados e leitores de código de barras, por exemplo. Algumas vantagens do Fieldbus FOUNDATION a ser citadas são: aumento da robustez do sistema devido aos dados digitais, ligação direta dos equipamentos na rede, sem necessidade das estações de controle de campo (ECL), indicação de falhas em tempo real, e diagnóstico preventivo com base nas informações fornecidas pelos equipamentos e em análises estatísticas destes (ALBUQUERQUE, 2009).

2.2.2.3. HART

De acordo com Albuquerque (2009), o protocolo HART (*Highway Addressable Remote Transducer)* surgiu na década de 1980 através da empresa *Rosemount* e transmite dados em longas distâncias com alta fidelidade. Está presente em milhões de instalações e segue crescendo em popularidade, suportando mais de dois terços dos instrumentos inteligentes de comunicação de dados atualmente. Possui dois modos de utilização (digital e analógico 4-20 mA), e dentre seus diferenciais podemos destacar ser uma solução compatível com os

dispositivos já instalados na planta e aumento no controle de integridade do sistema através de detecção automática de mudanças.

2.2.2.4. CAN

A rede CAN (*Controller Area Network*) foi desenvolvida pela empresa alemã *Bosch* em 1984 com auxílio da Intel para inicialmente atuar na eletrônica embarcada, e posteriormente para transmissão de dados de sensores. É muito utilizada na indústria automotiva, onde um único automóvel pode possuir cerca de 200 microprocessadores (frenagem ABS, controle de temperatura do óleo e do radiador, pressão do óleo de freio, entre outros). Utiliza a topologia em barramento ou estrela e comporta um máximo de 64 nós (LEME, 2001).

2.2.2.5. INTERBUS-S

O protocolo INTERBUS-S foi desenvolvido pela empresa *Phoenix Contact* em 1987 na Alemanha para integrar sensores e atuadores com CLP ou CNC. Utiliza a topologia em anel e o formato mestre-escravo, e hoje em dia mais de 600 fabricantes no mundo inteiro atuam com este protocolo. Lida com um processo denominado *shift register,* onde o mestre monta um quadro contendo dados do processo e parâmetros de configuração do escravo, em campos reservados para cada um dos escravos, que recebem, preenchem este quadro e passam ao próximo escravo no anel. Neste processo o ciclo de varredura é determinístico e a velocidade otimizada. O INTERBUS-S atua principalmente na indústria automotiva, mas esta solução também está inserida em indústrias de papel, alimentícia, transformação de madeira e transporte de materiais via esteira (LEME, 2001).

2.2.2.6. DeviceNET

Derivada da rede CAN, a DeviceNET segundo Leme (2001) teve seu início em 1994 nos Estados Unidos através da empresa *Rockwell Automation* e é uma

rede de baixo nível que comunica diversos equipamentos simples e complexos, como sensores e CLP utilizando o mesmo meio físico. A faixa de dispositivos que podem ser conectados através do protocolo DeviceNET é muito ampla e relativamente de baixo custo, e esta rede inteligente de chão-de-fábrica fornece dados importantes para diagnósticos e análises, úteis para embasar programas de manutenção preditiva e análises de performance. A topologia permite até 64 nós em um tronco principal com ramificações, a comunicação pode ser cíclica ou acíclica, com um ou vários mestres, e o tráfego na rede é reduzido pelas mensagens de rede serem de pequeno tamanho (até 8 *bytes*).

2.2.2.7. PROFIBUS

Desenvolvido na Alemanha em 1987, pelas empresas Siemens, Bosch e Klockner-Moeller, o PROFIBUS (*Process Field Bus*) é uma proposta para a padronização internacional de protocolos de campo. Conta com o apoio de dezenas de empresas europeias e internacionais, e já é um padrão adotado na comunidade através das *European Standard* EN 50170 e EN 50254 e pelo padrão da International Eletrotechnical Comission IEC 61158 (STEMMER, 2001).

2.2.2.7.1. Características Básicas

O PROFIBUS é um protocolo do tipo mestre e escravo, e os dispositivos mestres determinam a transmissão de dados e emitem mensagens sem pedido externo, sendo chamados de estações ativas. Os dispositivos escravos (estações passivas) incluem sensores, transmissores de medida, válvulas e dispositivos de entrada e saída e não possuem o direito ao acesso direto do barramento, somente respondendo às mensagens do mestre (ALBUQUERQUE, 2009).

2.2.2.7.2. Tecnologias de transmissão

Para escolha da tecnologia de transmissão, são necessários alguns conhecimentos básicos da planta segundo Albuquerque (2009), como distância a ser coberta, velocidade da transmissão requisitada e segurança da transmissão, além de fatores eletromecânicos. A tecnologia utilizada mais frequentemente é a RS-485 por ser fácil de instalar e permitir a instalação de novas estações sem alterar as outras estações, e essa tecnologia utiliza um cabo de cobre blindado e um par de condutores. Nesta tecnologia, a velocidade de transmissão pode atingir até 12 M*bits* por segundo, com topologia em barramento, comunicação bilateral e permite até 127 estações com o uso de repetidores.

Para indústrias que necessitam de segurança intrínseca, como na indústria química e petroquímica, há uma tecnologia determinada pelo padrão IEC 61158-2, que também permite o acesso ao barramento de transmissão pelos dispositivos de campo. Nesta tecnologia, há a transmissão contínua, com sincronismo *bit* a *bit*, cabo de par trançado blindado ou não, com segurança de dados *preamble*e *error-proof*, velocidade de transmissão de 31,25 *kbit*/seg e até 126 estações com o uso de repetidores (ALBUQUERQUE, 2009).

Em ambientes com elevada interferência eletromagnética, para um aumento da distância máxima ou para o aumento da velocidade de transmissão, segundo Albuquerque (2009), podem ser utilizados condutores de fibra óptica, que podem cobrir áreas de até 80 km. Os fornecedores já oferecem conversores para a conexão RS-485 e fibra óptica e vice-versa.

2.2.2.7.3. Arquitetura do Protocolo PROFIBUS

Há três tipos de arquiteturas para o protocolo PROFIBUS, PROFIBUS DP, PROFIBUS-FMS e PROFIBUS-PA, cada qual com suas particularidades. As camadas utilizadas no PROFIBUS podem ser vistas na figura 7:

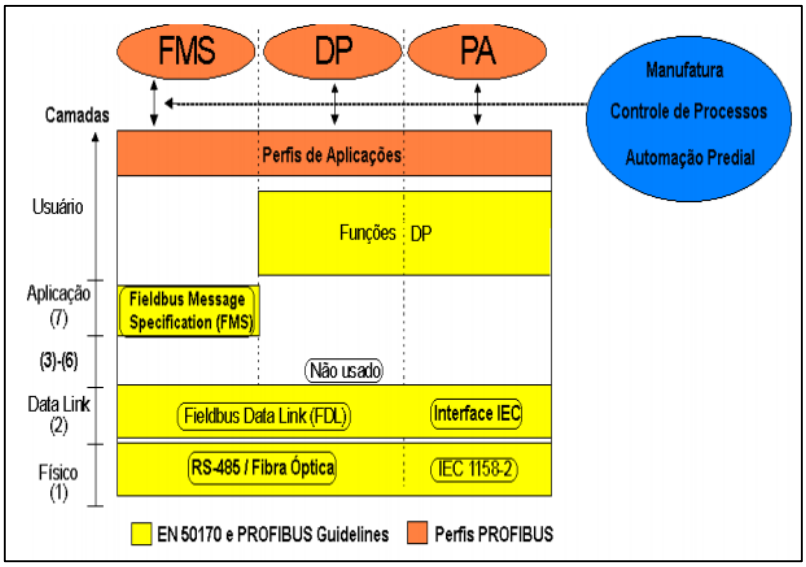


Figura 7- Arquitetura de comunicação do protocolo PROFIBUS.
Fonte: Cassiolato; Torre (2007).

2.2.2.7.3.1. PROFIBUS DP

Utilizado em 90% das aplicações envolvendo escravos PROFIBUS, esta solução de alta velocidade foi criada para a comunicação entre sistemas de automação e equipamentos descentralizados. O PROFIBUS DP (*Decentralized Peripherals)* substitui os convencionais 4 a 20 mA, HART e está ligada a sistemas de controle com dispositivos de entrada e saída distribuídos, pois requer apenas 2 ms para transmitir 1k*byte* de entrada e saída. É muito utilizada em controles onde é necessário um tempo crítico, e está atualmente disponível em três versões (CASSIOLATO; TORRE, 2007):

- DP-V0: de 1993, que permite a troca de dados cíclicos entre mestres e escravos e diagnósticos;
- DP-V1: de 1997, que acrescentou *Alarm Handling*, *Fail Safe* e blocos funcionais, e
- DP-V2: de 2002, a versão mais sofisticada tecnologicamente, possui modo tempo de ciclo determinístico, sincronização de *clock* e *Time stamp* e redundância.

2.2.2.7.3.2. PROFIBUS-FMS

Esta solução aceita a comunicação entre sistemas de automação (mestre-mestre) e troca de dados entre equipamentos inteligentes, proporcionando uma grande seleção de funções em nível de controle. O PROFIBUS-FMS (*Fieldbus Message Specification)* concorre com o protocolo *Ethernet* (CASSIOLATO; TORRE, 2007).

2.2.2.7.3.3. PROFIBUS-PA

Esta solução substitui diretamente a tecnologia analógica 4 a 20 mA, conectando sistemas de controle de processo com os dispositivos de controle (controladores de pressão, temperatura, nível). O PROFIBUS-PA (*Process Automation)* utiliza funções do PROFIBUS-DP para medidas dos estados e transmissão de valores, e pode gerar economias com cabeamentos, manutenção e comunicação de até 40% (ALBUQUERQUE, 2009).

2.3. SISTEMA SUPERVISÓRIO

O sistema supervisório também chamado de *SCADA* (*Supervisory Control and Data Aquisition*) é um *software* que permite aquisição e monitoramento de dados do processo produtivo. (MELO; MOSCARDI, 2005)

A função principal do sistema supervisório é converter os dados coletados no processo em informações ao usuário do sistema, através de telas de monitoramento e controle. (MOLODOWSKI; OLIVEIRA, 2010)

Os componentes principais de um sistema *SCADA* são:

- Sensores e atuadores – são dispositivos que fazem a leitura das informações do processo e realizam comandos enviados pela estação central.
- Rede de comunicação – é a plataforma onde as informações são transmitidas. As transmissões de dados podem ser feitas através de cabos Ethernet, fibras ópticas e linhas *dial-up*.

- Estações remotas – são os controladores lógicos programáveis ou RTU (*Remote Terminal Units*) que fazem a leitura de entradas, os cálculos e o controle do processo.
- Estação de monitorização central – é o componente principal do sistema supervisório, local onde são processadas as informações geradas pelas estações remotas. (SILVA; SALVADOR, 2011)

A figura 8 mostra os componentes do sistema supervisório:

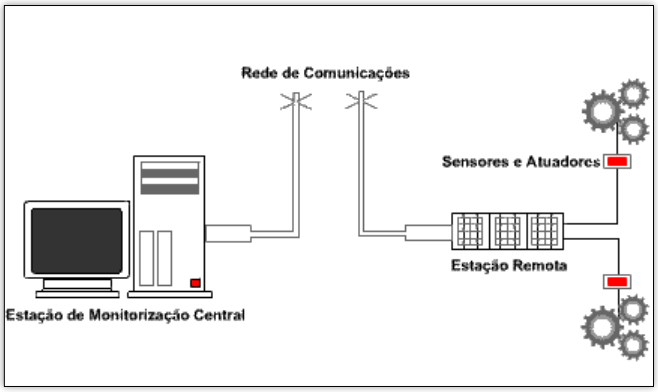


Figura 8 - Sistema de supervisão e controle.
Fonte: Silva; Salvador (2011).

3. MANUAL DIDÁTICO DA BANCADA

3.1. DESCRIÇÃO DA BANCADA UTILIZADA

A bancada a ser utilizada no trabalho, como ilustra a figura 9, consiste em:

(1) CLP Siemens S7-300, possuindo uma fonte de alimentação PS307 (1a), com entrada de 220 V e saída de 24 V; uma *CPU 313C-2* DP, com 16 entradas digitais e saídas digitais, e uma porta de comunicação, proprietária da Siemens, MPI (Multi-Point Interface), e uma porta PROFIBUS; e com dois módulos adicionais, um módulo (1c) possuindo quatro entradas analógicas e duas saídas digitais e a outra (1d) dezesseis saídas digitais. A bancada possui indicadores (1e) das saídas e entradas do CLP.

(2) IHM Eaton *XV-102-B2-35TQR-10*, com tela de toque resistivo com 3.5 polegadas e resolução de 320x240. Possui conexão via *Ethernet e* PROFIBUS.

(3) Inversor de frequência AWB 8230-1603 ligado ao um motor de indução trifásico Siemens com potência de 0,25 CV.

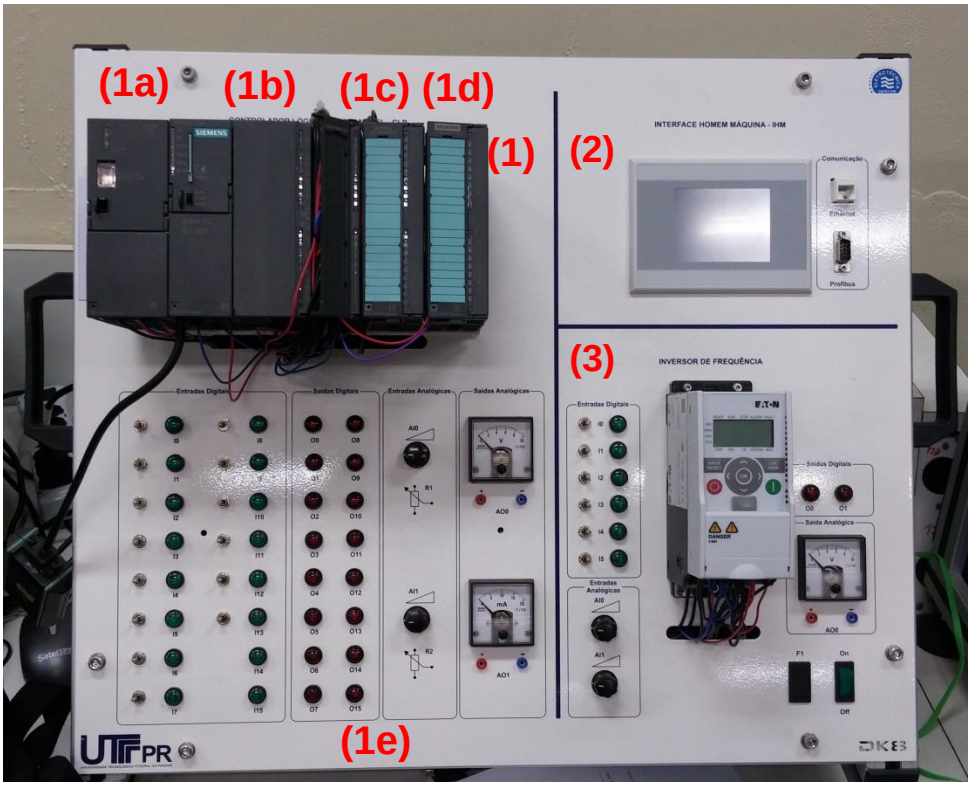


Figura 9 - Bancada utilizada.

Fonte: Autoria própria.

3.2. PROCEDIMENTOS INICIAIS

Antes de ligar a bancada na tomada, verificar se a tensão do mesmo é 220 V. Não alterar a tensão e não desligar a fonte do CLP. Somente ligar e desligar a bancada pelo botão verde, localizado no canto inferior direito da mesma. Não retirar o cartão de memória do CLP.

3.3. MANUAL DE PROGRAMAÇÂO DO CLP

3.3.1. Criando um arquivo novo no STEP7 v5.5

- Para a programação do CLP, será utilizado o software STEP7, da fabricante Siemens, em sua versão 5.5. Para acessá-lo, deve-se abrir o atalho “SIMATIC Manager”. O ícone do programa está ilustrado na figura 10:

Figura 10 - Ícone do programa STEP7**.**
Fonte: Autoria própria.

- Ao abrir o STEP7, é possível que algum arquivo já venha aberto automaticamente, que deve ser fechado selecionando a aba “*File/Close*”. A figura 11 mostra o local onde se encontra o item:

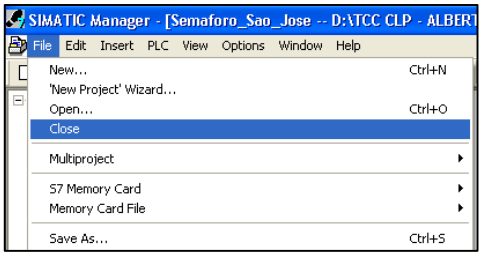


Figura 11 – Localização do item “*Close*”.
Fonte: Autoria própria.

- E então criar um novo arquivo, na aba “*File/New*”, ilustrado na figura 12:

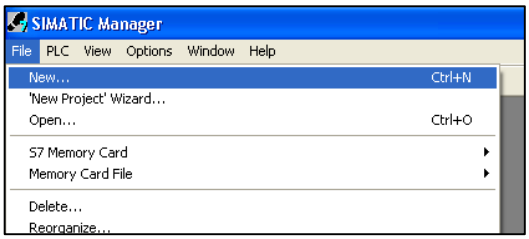


Figura 12 – Local para criar um novo programa.
Fonte: Autoria própria.

- Abrirá então uma tela onde se deve nomear o projeto (campo “*Name*”), selecionar a pasta onde será salvo o projeto (no botão “*Browse...*”) e então clicar no botão “*OK*”. A figura 13 mostra a janela “*New Project”*:

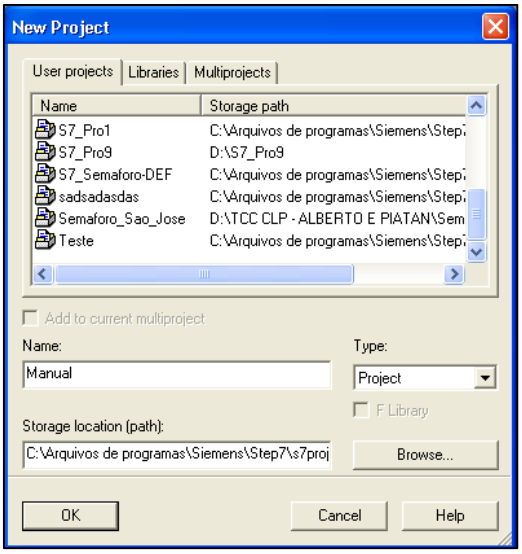


Figura 13 - Janela do novo projeto.
Fonte: Autoria própria.

- O projeto será criado, porém ainda sem as informações de *hardware* necessárias para iniciar a programação.

3.3.2. Configuração do *hardware* no programa STEP7

- Para configurar o *hardware*, selecionar a aba “*Insert/Station/*SIMATIC *300 Station*”. A figura 14 indica a localização do item:

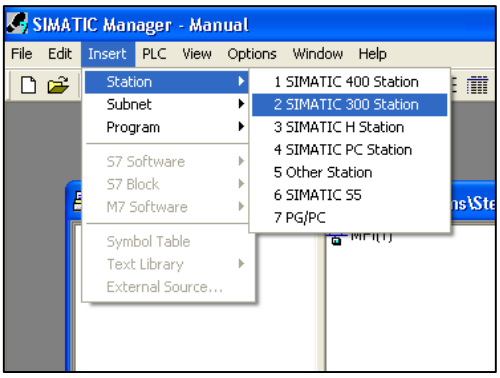


Figura 14 – Localização do "SIMATIC ***300 Station*****".**

Fonte: Autoria própria.

- E então, maximizar o projeto (clicando no botão “+” ao lado do nome do projeto) para exibir a nova estação criada e clicar duas vezes no “*Hardware”*. O local do item está ilustrado na figura 15:

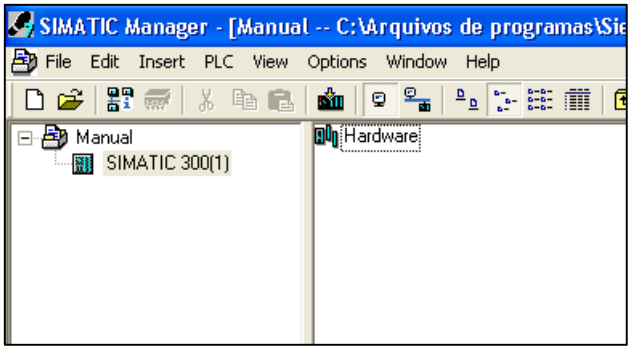


Figura 15 - Localização de "*Hardware*".

Fonte: Autoria própria.

- Abrirá uma nova janela, denominada “*HW Config*”, onde serão adicionadas as configurações do *hardware*. Primeiramente, deve-se adicionar o *rack* onde serão inseridos o CLP e seus módulos. Para isto, maximizar a guia “SIMATIC 300”, então a guia “RACK-300”, e clicar duas vezes no item “*Rail*”. A figura 16 mostra o local do item “*Rail*”:

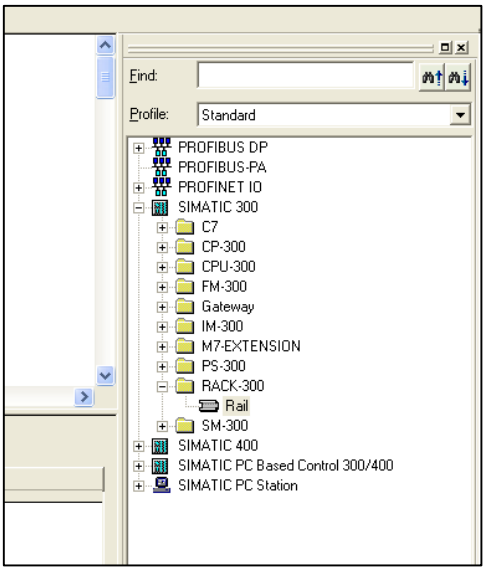


Figura 16 - Localização do item "*Rail*".

Fonte: Autoria própria.

- Para saber qual o modelo do CLP, deve-se olhar diretamente na bancada na parte superior do CLP. A figura 17 indica o local do modelo do CLP

Figura 17 - Fotografia do CLP utilizado.

Fonte: Autoria própria.

- Neste caso a CPU é “CPU313C-2 DP” com o código “313-6CE00-0AB0”. Para adicioná-la ao *rack*, encontra-se a aba *“*CPU-300*”*, então “CPU 313C-2 DP”,

então um duplo clique no código correspondente e "*OK*" na janela que se abrirá. Alternativamente, pode-se utilizar a função "*Find*" ("Pesquisa") e digitar o código diretamente para localizá-lo. O local está indicado na figura 18:

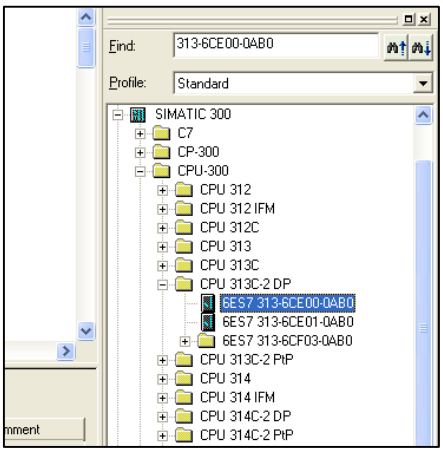


Figura 18 - Localização da CPU no STEP7**.**

Fonte: Autoria própria.

- Módulos adicionais podem ser adicionados de maneira similar, buscando seu código na bancada. A figura 19 mostra o módulo adicional;

Figura 19 - Fotografia do módulo adicional.

Fonte: Autoria própria.

- Neste caso, a bancada possui um módulo com 4 entradas e 2 saídas analógicas, com o código “334-0CE01-0AA0”. Buscando na área “*Find*”, pode-se adicioná-lo ao *rack*. A figura 20 ilustra o local do módulo no “SIMATIC *Manager*”:

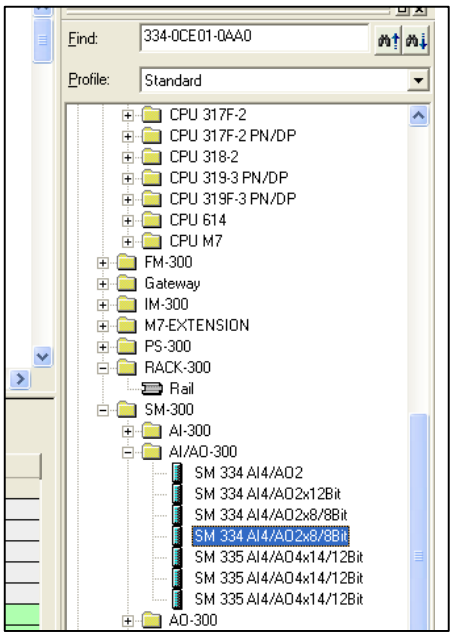


Figura 20 – Localização do módulo adicional.

Fonte: Autoria própria.

OBS: alguns módulos têm restrições quanto ao *slot* que pode ser inserido. Caso apareça alguma mensagem de erro, selecionar outro *slot* (na figura 21, foi selecionado o *slot* 4) e tentar novamente;

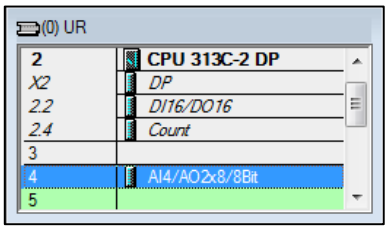


Figura 21 - *Rail0.*

Fonte: Autoria própria.

- Nesta mesma janela “*HW Config*”, pode-se verificar e alterar o endereço para acesso a estes módulos dentro do programa. A figura 22 mostra as propriedades do CLP;

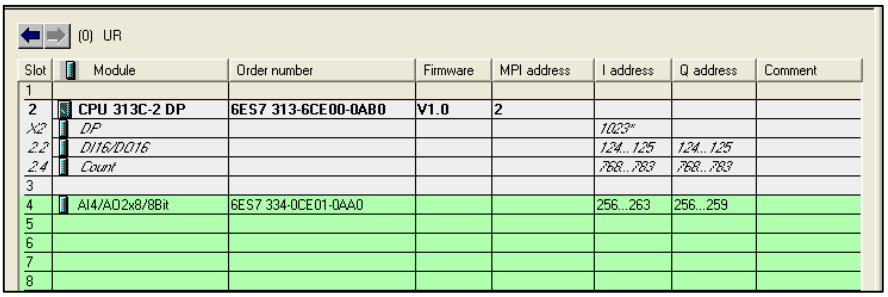


Figura 22 - Local para alterar o endereço de acesso.

Fonte: Autoria própria.

- As entradas digitais estão endereçadas nos *bytes* I124 e I125, indo, portanto de I124.0 até I125.7, as saídas nos *bytes* Q124 e Q125, as entradas analógicas ocupam os 8 *bytes* de I256 até I263 e as saídas analógicas os 4 *bytes* de Q256 até Q259. Para alterar estes valores, basta clicar duas vezes no item correspondente, selecionar a aba "*Addresses"*, desmarcar a opção "*System default"* e alterar para o valor desejado. A figura 23 ilustra a janela de propriedades do módulo de entrada:

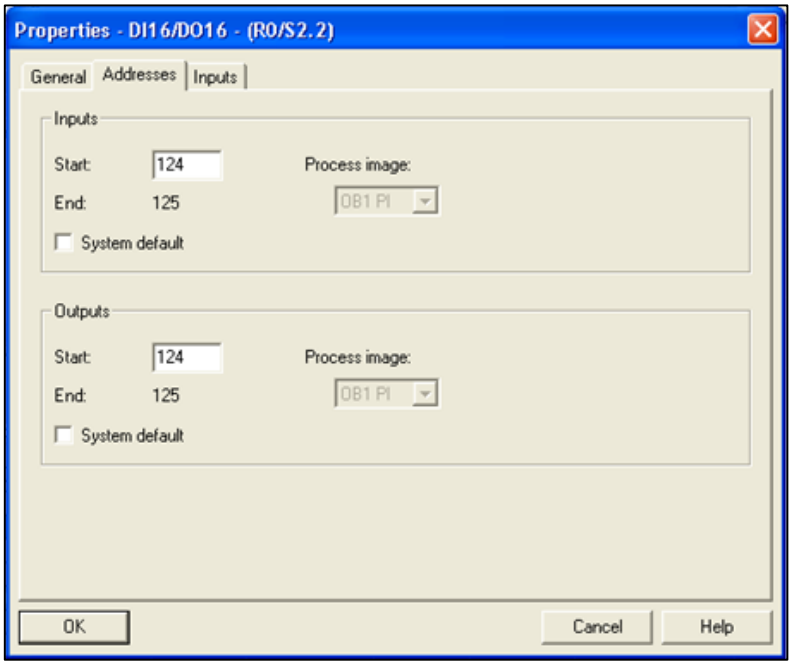


Figura 23 - Propriedades do módulo de entrada.

Fonte: Autoria própria.

- Uma vez que foram adicionados todos os módulos e configurados seus endereços, salvar as configurações de *hardware* e fechar a janela "*HW Config"*.

3.3.3. Programação do CLP

- Para iniciar a programação, abrir o objeto “OB1”, localizado dentro da pasta “*Blocks”*. A figura 24 indica a localização do bloco “OB1”.

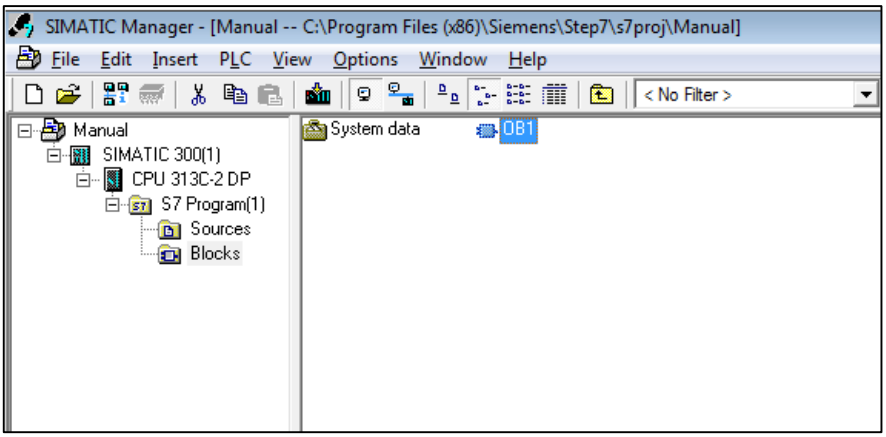


Figura 24 - Localização do OB1.

Fonte: Autoria própria.

- O OB1, ou *Organization Block* 1 é o bloco inicial do programa e o único obrigatório e necessário para que o programa funcione corretamente. Ao ligar o CLP, o programa iniciará com o OB1 e reproduzirá este bloco ciclicamente por um número indefinido de vezes. Pode-se realizar toda a programação no OB1, mas em programas mais complexos pode surgir a necessidade de uma maior organização no programa, e para isso outros tipos de blocos lógicos existem, quais sejam:

- Outros OB (*Organization Block)*: estes outros blocos OB são utilizados para tratamento de interrupções, que podem ser definidas em uma escala de tempo (de acordo com a hora ou em um intervalo estipulado), uma interrupção de *hardware*, ou mesmo OBs para tratamento de erros. Exemplo: ao final do dia em um determinado horário, há uma interrupção no programa principal para cálculo da vazão total diária verificada em um processo.

- FB (*Function Block)*: são funções criadas pelo usuário para executar uma determinada sub-rotina, que criam novos dados a serem armazenados cada vez que esta função é chamada. É utilizada geralmente para realizar cálculos ou procedimentos que se repetem várias vezes no programa. Exemplo: uma FB que

controla vários motores, cada um com seus parâmetros (velocidade, tempo de operação) próprios.

- FC (*Function)*: semelhantemente aos FBs, são funções criadas pelo usuário, porém sem salvamento de dados a cada vez que é chamada. É mais utilizada então para simplificar ações que são realizadas repetidas vezes. Exemplo: uma FC que realiza um cálculo matemático e retorna o seu resultado.
- DB (*Data Block)*: são tabelas de dados, sem nenhum tipo de ação ou lógica. São utilizadas nos outros blocos. Exemplo: um DB contendo diferentes parâmetros de diversos motores.
- UDT (*User Data Type)*: tipos de dados criados pelo usuário (um *Struct)* para melhor organização do programa. Exemplo: é criado um UDT Tanque, que contém dados de nível, vazão e temperatura. No programa, podem-se criar outras variáveis de tipo Tanque e utilizá-las ao longo de todo o programa.
- VAT (*Variable Table)*: são tabelas utilizadas para monitorar, modificar e forçar valores. São utilizadas em testes do programa. Exemplo: cria-se uma VAT para forçar valores de sensores ainda não instalados e verificar a funcionalidade do programa.

- Ao abrir-se o OB1, é possível selecionar a linguagem *Ladder (LAD)* na opção “*Created in Language:*” e então clicar “*OK*”. A figura 25 mostra a tela de propriedades do bloco de organização;

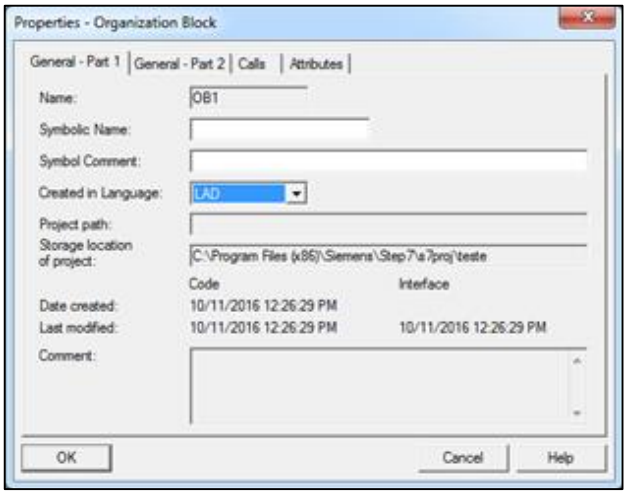


Figura 25 - Propriedades do bloco OB1.

Fonte: Autoria própria.

- E então a tela do OB1, ilustrada na figura 26, será aberta. No lado esquerdo da tela se apresentam os outros blocos lógicos do projeto que podem ser utilizados, na parte superior aparece a declaração de variáveis. Na parte inferior ficam os detalhes do projeto e no centro da tela estão as instruções do programa.

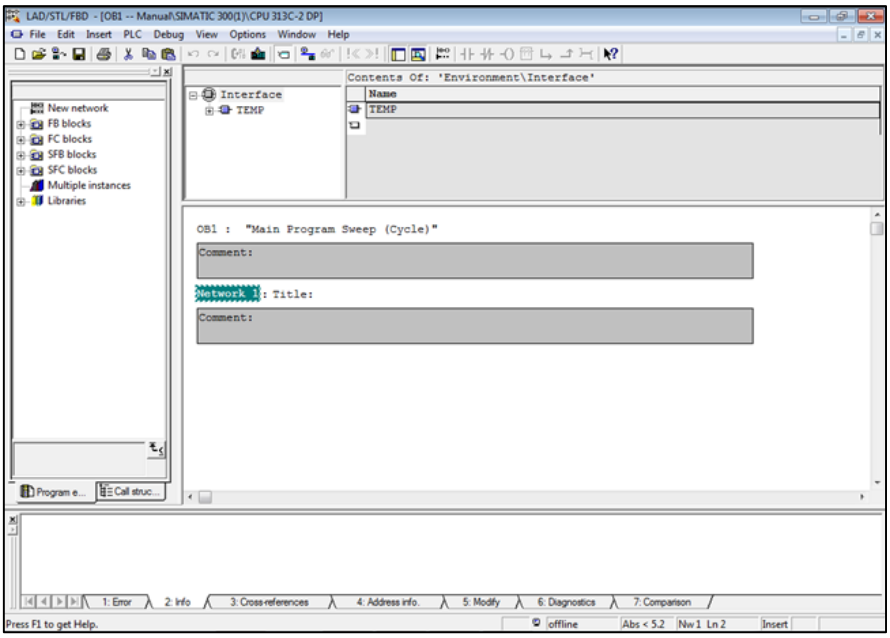


Figura 26 - Tela do OB1.

Fonte: Autoria própria.

- As partes escuras onde aparece o escrito "*Comment:"* são para comentários do programador e não serão executadas pelo CLP. As *Networks* são as linhas do código, e para incluir novas *Networks* clica-se no menu "*Insert/Network"*, ou através do atalho *Ctrl* + R, ou mesmo do botão "*New Network"*, ilustrado na figura 27:

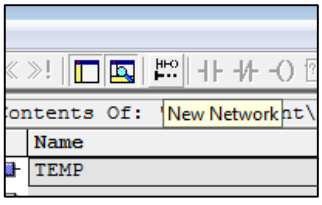


Figura 27 - Botão para criar uma nova *network.*

Fonte: Autoria própria.

- Para alterar a linguagem de programação de STL (*Statement List)* para *Ladder,* clica-se em "*View/LAD"*. A figura 28 mostra o local:

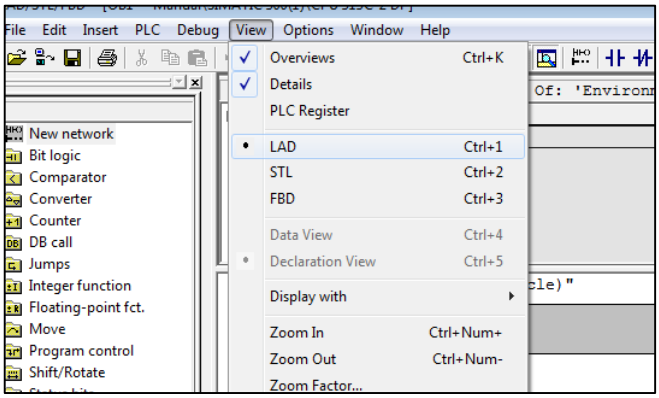


Figura 28 - Local para alterar a linguagem de programação.
Fonte: Autoria própria.

- Para a lógica em *Ladder*, deve-se utilizar os botões localizados na parte superior direita da barra de botões, ou no menu "*Insert/LAD Language Elements* ilustrado na figura 29:

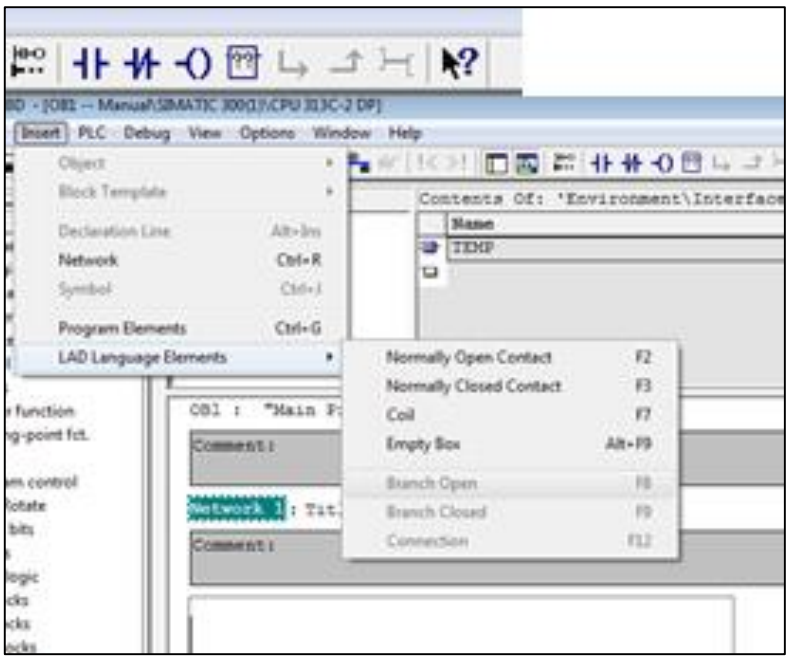


Figura 29 - Métodos para inserir botões.
Fonte: Autoria própria.

- Para o endereço dos contatos, pode-se utilizar o endereço físico (I para entradas, Q para saídas e M para endereços de memória) ou criar símbolos para

facilitar o reconhecimento dos contatos. Para criar símbolos, pode-se clicar com o botão direito no contato desejado e depois em "*Edit Symbols...*". A figura 30 mostra o local para criar contatos:

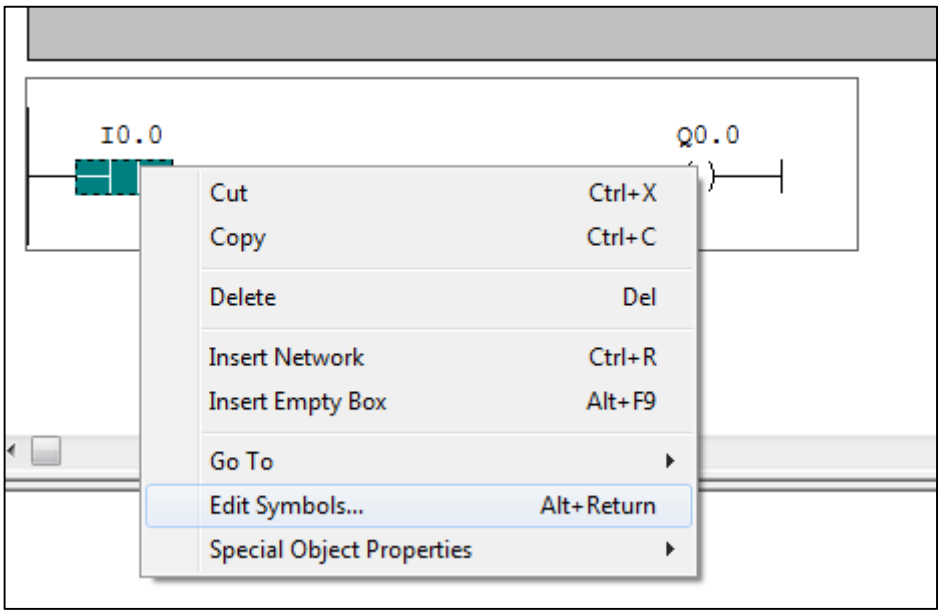


Figura 30 - Local para criar símbolos.

Fonte: Autoria própria.

- Preencher então a aba "*Symbol"* para o nome que se deseja e clicar em "*OK"*. A figura 31 mostra a janela para editar os símbolos.

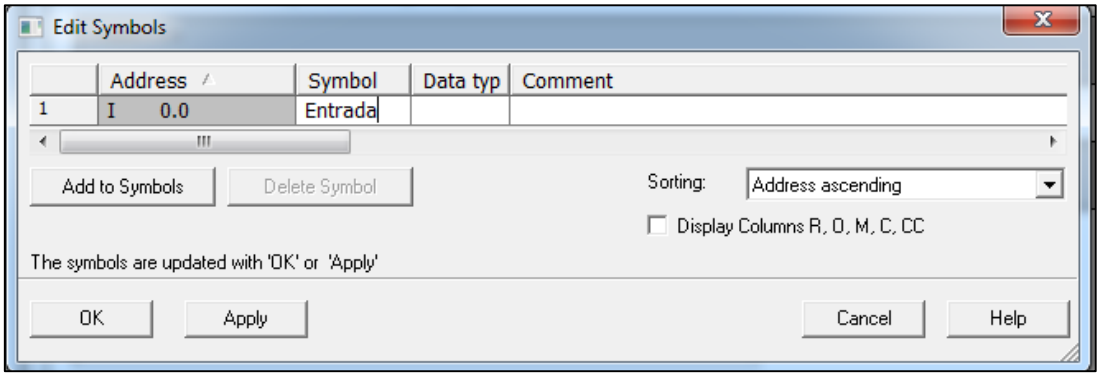


Figura 31 - Janela para criar símbolos.

Fonte: Autoria própria

- Na tela inicial do SIMATIC STEP7, pode-se também editar e criar novos símbolos, através do editor de símbolos (*Symbols)*, que pode ser localizado na pasta "*S7 Program(1)*". A figura 32 mostra o local do editor de símbolos:

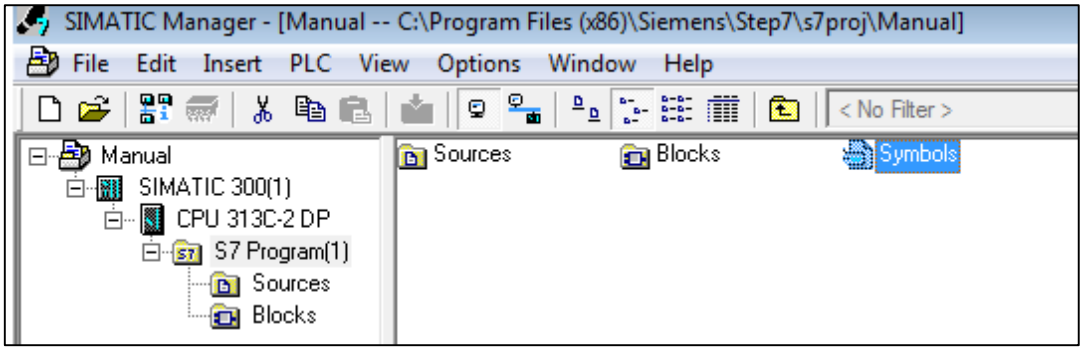


Figura 32 - Local do editor de símbolos.

Fonte: Autoria própria.

3.3.4. Utilizando o simulador

- O software STEP7 possui uma função de simulador, que pode ser utilizada para testar a funcionalidade do programa sem precisar realizar o *download* para o CLP. Para acessar o simulador, deve-se primeiramente alterar a interface clicando-se em "*Options/Set PG/PC Interface…*", ilustrado na figura 33:

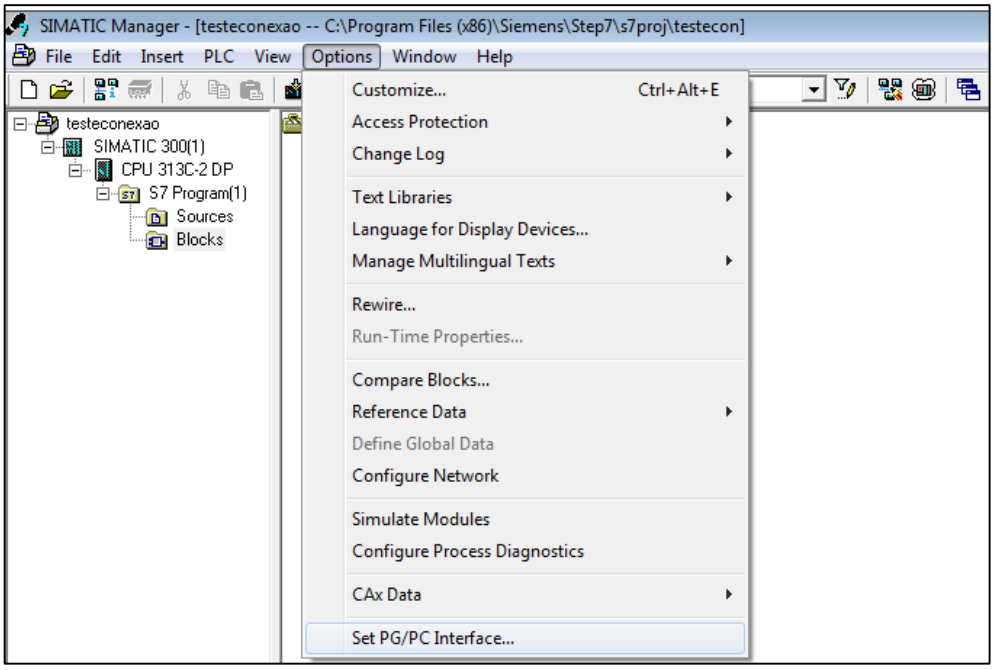


Figura 33 - Local da escolha de interface de comunicação.

Fonte: Autoria própria.

- E então selecionar “*PLCSIM.MPI.1*” na tela “*Set PG/PC Interface*”, indicado na figura 34:

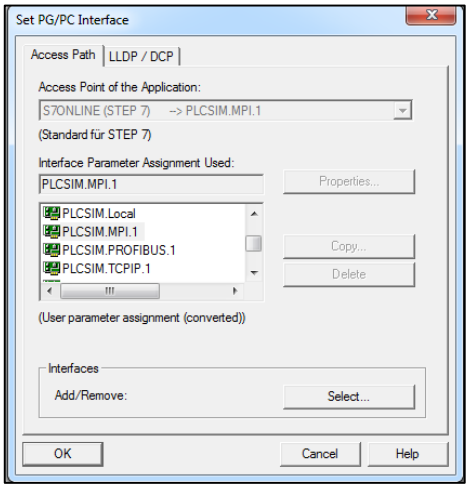


Figura 34 - Tela de escolha da interface

Fonte: Autoria própria

- E então clicar em “*Options/Simulate Modules”*, ilustrado na figura 35. Ou no ícone “*Simulation On/Off”*, indicado na figura 36:

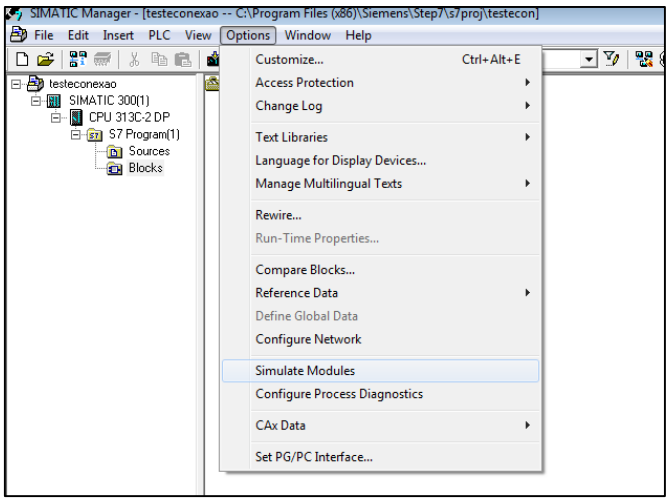


Figura 35 - Local do item "*Simulate Modules*"

Fonte: Autoria própria.

Figura 36 - Ícone do item "*Simulation On/Off*"

Fonte: Autoria própria.

- Com a tela de simulação aberta, clicar no ícone “*Download”* na tela principal do SIMATIC *Manager*, ou em “*PLC/Download”*. A figura 37 mostra a localização do item:

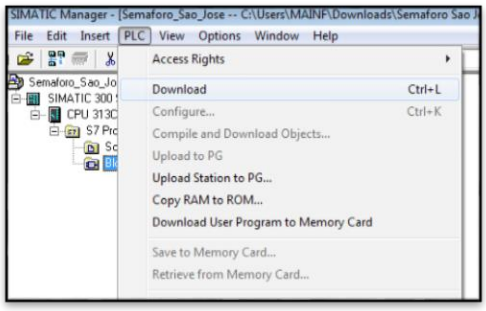


Figura 37 - Local do item "*Download*".

Fonte: Autoria própria.

- Selecionar a janela do simulador (S7-PLCSIM1) e clicar no campo “*RUN*” para iniciar a simulação. A figura 38 mostra a janela do simulador:

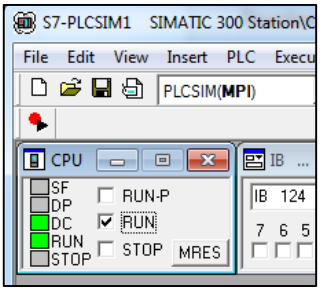


Figura 38 - Janela do simulador

Fonte: Autoria própria

- Através da aba “*Insert”*, indicado na figura 39, ou de seus ícones (figura 40) é possível adicionar memórias de entrada, saída, ou bits de memória para modificá-los e verificar a funcionalidade do programa;

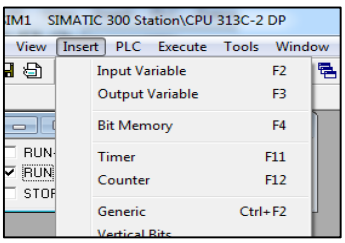


Figura 39 - Itens que podem ser adicionados na simulação

Fonte: Autoria própria

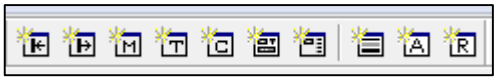


Figura 40 - Atalhos dos itens

Fonte: Autoria própria

- No exemplo abaixo, têm-se uma janela de memórias de entrada (IB) com o *byte* 124 (que pode ser alterado selecionando e mudando seu valor) mostrado em bits. A forma como as memórias são mostradas pode ser alterada clicando em "*Bits"* e alterando para o modo que se deseja. Para forçar os valores dos *bits* basta clicar na caixa correspondente. Na figura 41, o bit I124.0 está ligado;

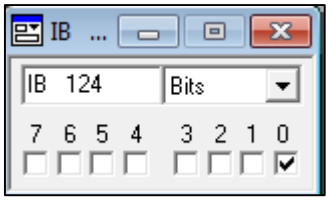


Figura 41 - Exemplo do bit ligado.

Fonte: Autoria própria.

3.3.5. Configurando a interface com o SIMATIC *USB PC Adapter*

- Para a comunicação entre o CLP e o PC, é utilizado o adaptador SIMATIC *USB PC Adapter,* que possui uma entrada USB para a conexão com o computador e outra conexão MPI para o CLP. A figura 42 mostra o adaptador presente no laboratório:

Figura 42 - *PC Adapter USB*.

Fonte: Autoria própria.

- Após certificar-se que o adaptador está conectado ao PC e ao CLP, o Windows automaticamente instalará o *driver* necessário caso uma conexão com a internet esteja disponível. Caso a instalação se dê corretamente, o LED verde na parte “USB” do conector ficará ligado.

- Para configurar a interface, na tela principal do SIMATIC, selecionar “*Options/Set PG/PC Interface…”*, e na janela que abrir, escolher a opção “*PC Adapter.MPI.1”*. A figura 43 mostra a janela com o “*PC.Adapter.MPI.1”* escolhido:

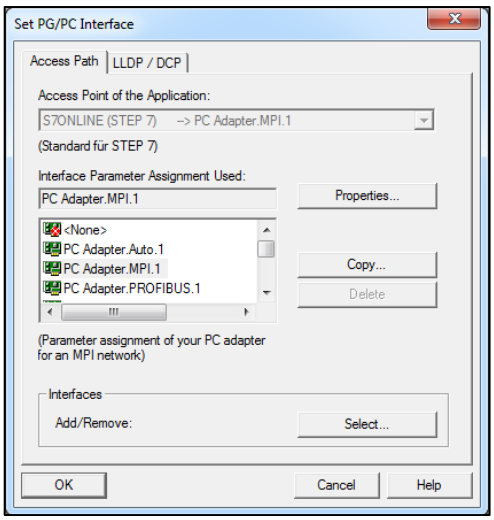


Figura 43 - Janela para a escolha da interface de comunicação.
Fonte: Autoria própria.

- Nesta tela ainda é possível alterar a taxa de transmissão e outros parâmetros clicando-se em “*Properties”*. A figura 44 mostra a janela citada:

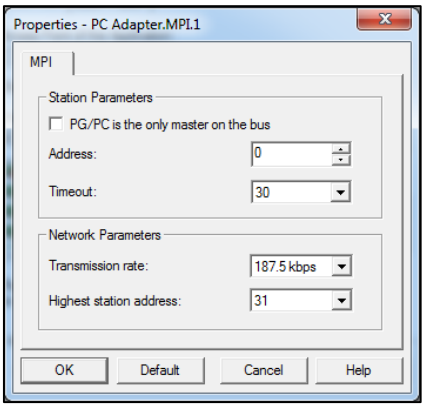


Figura 44 - Janela de parâmetro da interface de comunicação.
Fonte: Autoria própria.

3.3.6. Compilando e testando o programa

- Quando o programa estiver pronto para ser testado no CLP, selecionar todos os blocos e clicar no botão “*Download”,* indicado na figura 45, ou através do menu “*CLP/Download”* (figura 46):

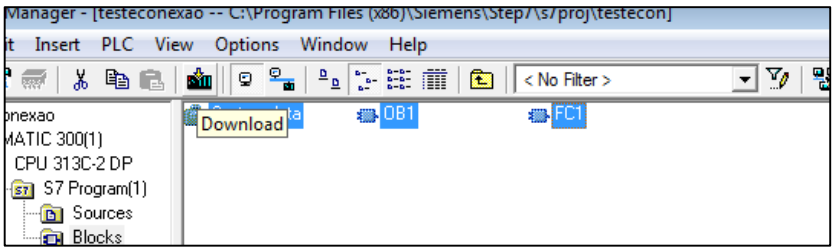


Figura 45 - Método para fazer *download.*

Fonte: Autoria própria.

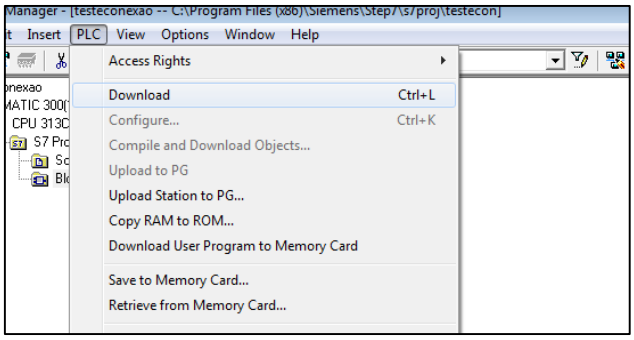


Figura 46 - Método para fazer *Download.*

Fonte: Autoria própria.

- Diversas mensagens podem aparecer, relatando que já existem programas no CLP. Clicar no botão “*All”,* na janela indicado na figura 47, para transferir o programa ao CLP.

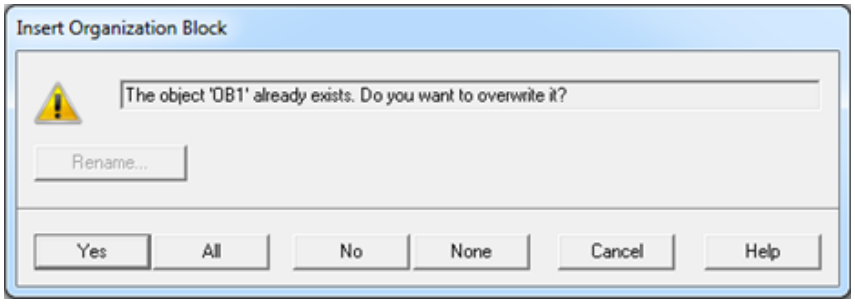


Figura 47 - Mensagem relatando a existência de programa no CLP.

Fonte: Autoria própria.

- Clicar “*Yes”* na tela seguinte, ilustrado na figura 48, para carregar as configurações de *Hardware;*

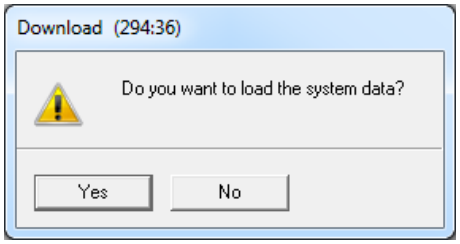


Figura 48 - Janela de confirmação do *download.*

Fonte: Autoria própria.

- E “*OK*” para o CLP ser colocado no modo “STOP” (necessário ao se transferir um programa por completo). A figura 49 mostra a janela indicando que o módulo irá parar:

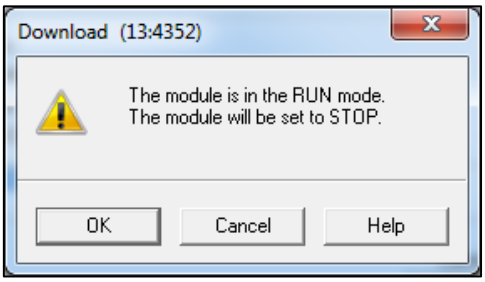


Figura 49 - Janela informando que o modulo irá parar.

Fonte: Autoria própria.

- Finalmente em “*Yes”,* na janela indicado na figura 50, para colocar o CLP no modo “*RUN”* novamente;

OBS: verificar se no CLP a chave de posição está no modo “*RUN*”.

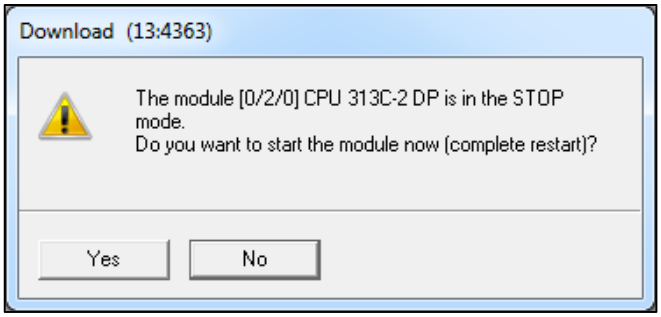


Figura 50 - Janela perguntando se deseja alterar o estado do CLP.

Fonte: Autoria própria.

OBS: caso a mensagem da figura 51 seja apresentada, é provável que o *Memory Card* inserido no CLP possua dados inconsistentes com o módulo. Isto pode ser confirmado verificando se a luz de "*STOP"* está piscando e o indicador *SF* da CPU está aceso.

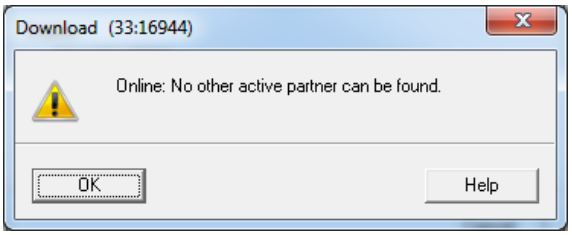


Figura 51 - Janela de erro no download

Fonte: Autoria própria

- Com o programa rodando, pode-se monitorá-lo de maneira direta abrindo um bloco ou função e clicando-se no menu "*Debug/Monitor"*, indicado na figura 52:

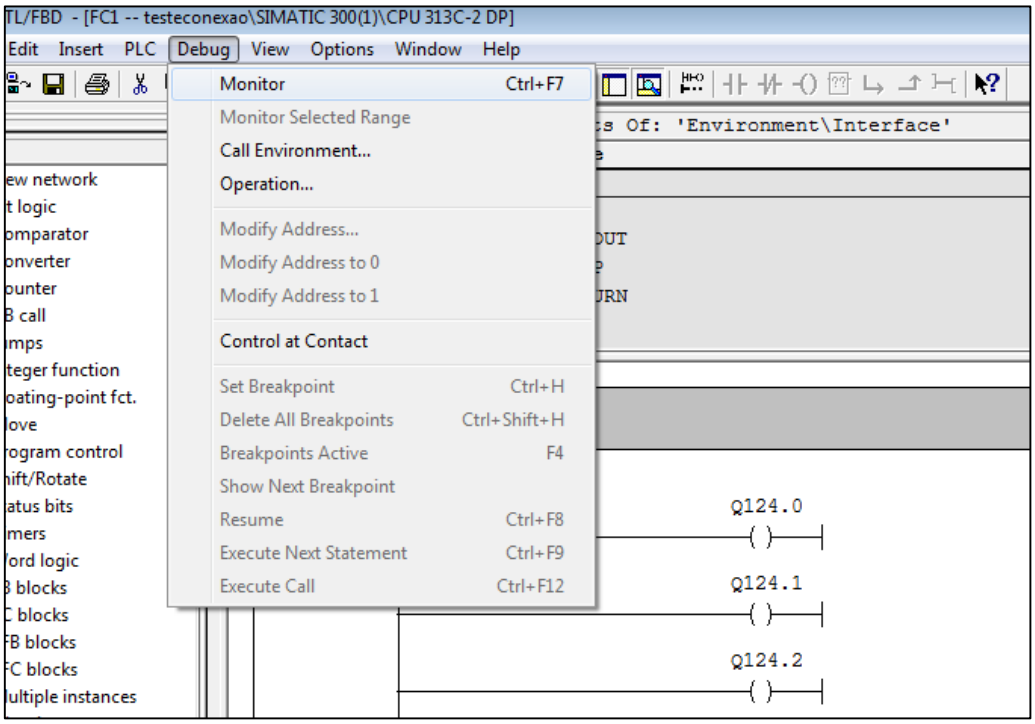


Figura 52 - Localização do item "*Monitor*".

Fonte: Autoria própria.

- Neste modo monitor, linhas verdes representam o fluxo de energia e pode-se visualizar o funcionamento do programa. Para se fazer alterações porém, é necessário o desligamento do modo "*Monitor"* e um novo *download* do bloco em que

foram feitas as alterações. A figura 53 mostra um exemplo de programa com o monitoramento ligado:

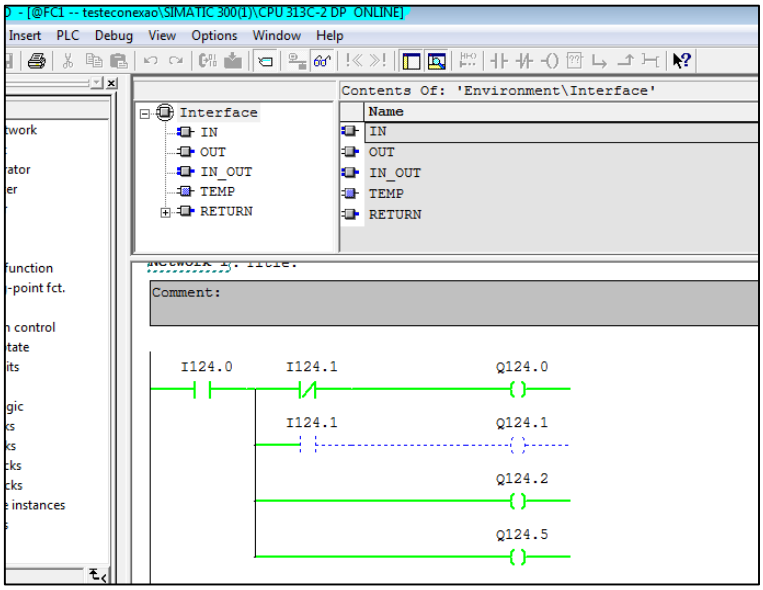


Figura 53 - Janela com o monitoramento ligado.

Fonte: Autoria própria.

3.3.7. Adaptação do projeto do CLP para comunicação com a IHM

- Após testar o programa, deve se adaptar o mesmo para a comunicação com a IHM. É necessário criar um DB, clicando com o botão direito na tela “*Blocks”* e ir ao item "*Insert New Object/ Data Block*". A figura 54 mostra o item descrito:

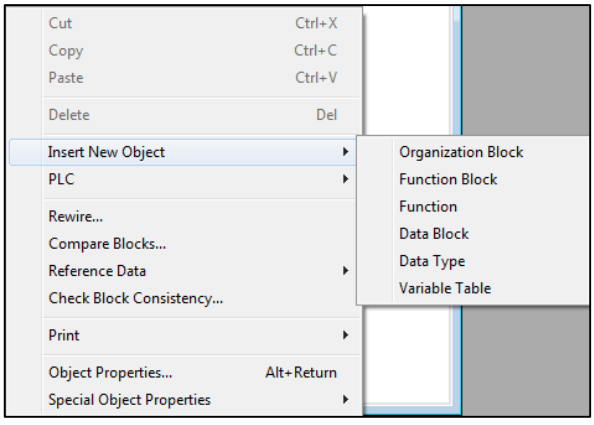


Figura 54 - Localização do item para criar a DB.

Fonte: Autoria própria.

- Abrirá uma janela com as propriedades do DB. Nesta tela é possível escolher o índice do DB, tipo, nome simbólico e escrever alguns comentários. A figura 55 mostra um exemplo de DB criado:

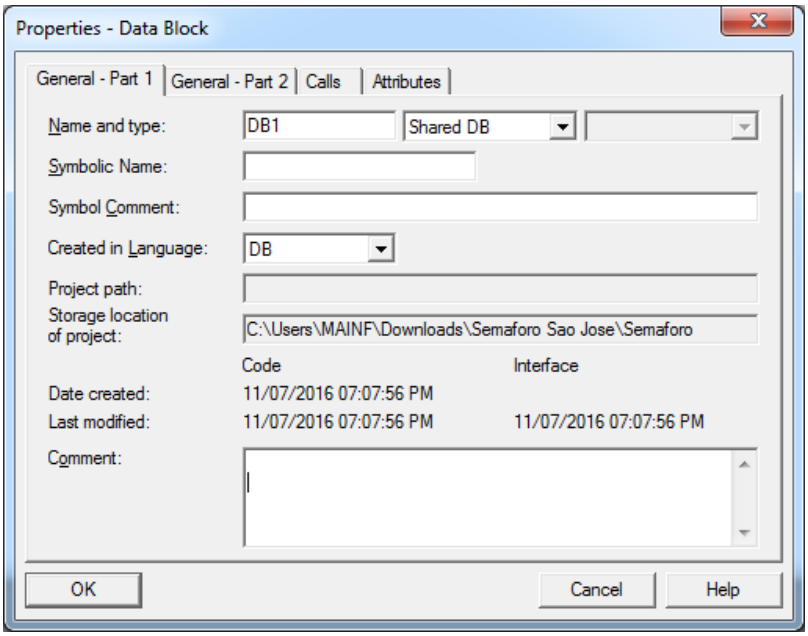


Figura 55 - Janela de propriedades do DB.

Fonte: Autoria própria.

- Após criar a DB, deve-se abri-lo. A figura 56 mostra a localização do bloco:

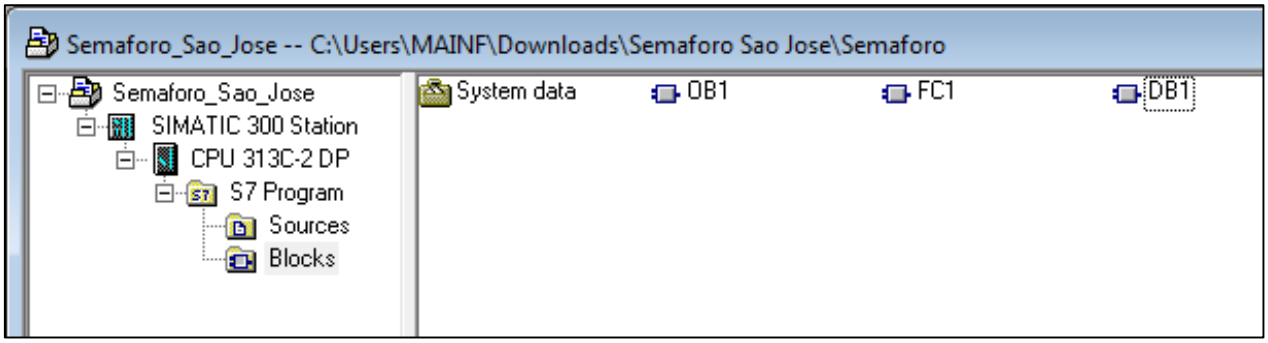


Figura 56 - Local do bloco DB.

Fonte: Autoria própria.

- Aberto o *Data Block,* é necessário declarar os dados que serão utilizados no programa. A figura 57 mostra o DB aberto, onde é possível escolher o nome, o tipo, o valor inicial e escrever comentários. Os tipos que podem ser utilizados são

boleana, do tipo *word,* do tipo *doubleword.* Ao lado do nome, estão os endereços que serão utilizados no programa do CLP.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Boleana	BOOL	FALSE	BOOL
+2.0	Words	WORD	W#16#0	WORD
+4.0	DoubleWords	DWORD	DW#16#0	DWORD
=8.0		END_STRUCT		

Figura 57 - Bloco DB

Fonte: Autoria própria

- Dentro do programa OB1, devem-se colocar os dados declarados no DB. A tabela 1 mostra como é chamado os dados da DB, de acordo com seu tipo, onde deve-se substituir o valor “a” pelo índice do DB, o valor “b” pelo byte e o valor “c” pelo bit.

Tabela 1 - Tipo de dado e seu respectivo símbolo.

Tipo	Simbolo
Bit	DB**a**.DBX**b**
Byte	DB**a**.DBB**c**
Word	DB**a**.DBW**c**

Fonte: Autoria própria.

- A figura 58 mostra um exemplo de como o dado declarado no DB é utilizado dentro de um programa.

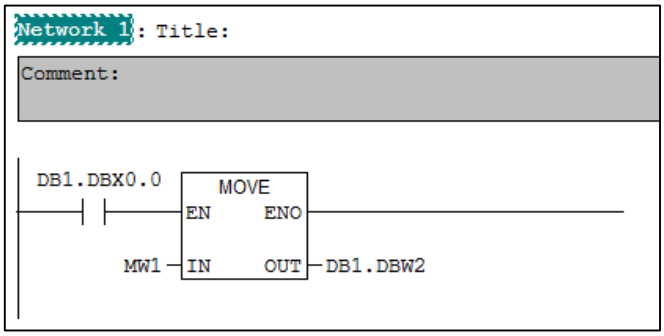


Figura 58 - Exemplo de programa com dados da DB

Fonte: Autoria própria

3.4. MANUAL DE PROGRAMAÇÃO DA IHM

3.4.1. Criando um novo projeto no Galileo v.7.2.8 (11223)

- Para a programação da IHM, será utilizado o *software* Galileo, da fabricante Eaton, em sua versão 7.2.8(11223). Para acessá-lo, deve-se abrir o atalho “Galileo”, ilustrado na figura 59:

Figura 59 - Ícone do *software* Galileo.
Fonte: Autoria própria.

- Para criar um novo projeto, deve-se ir à aba “*Project/New”*, no canto superior esquerdo da tela. A figura 60 mostra a localização do item:

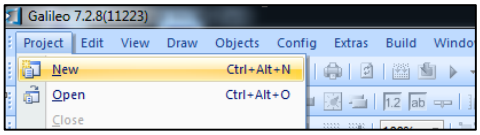


Figura 60 - Localização do item para criar um novo projeto.
Fonte: Autoria própria.

- Ao clicar em “*New”*, abrirá uma tela onde se deve nomear o projeto (campo “*Nome*”), selecionar a pasta onde será salvo o projeto. Escolhido o local, surgirá uma janela onde será escolhido o tipo de painel. Para selecionar, deve-se clicar em “*Panel Selection*”, indicado na figura 61:

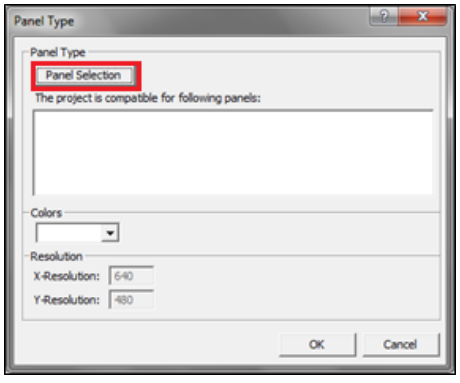


Figura 61 - Janela para a escolha do painel.
Fonte: Autoria própria

- Abrirá uma nova janela, denominada “*Panel Selection*”, ilustrado na figura 62, onde será escolhido o modelo da IHM;

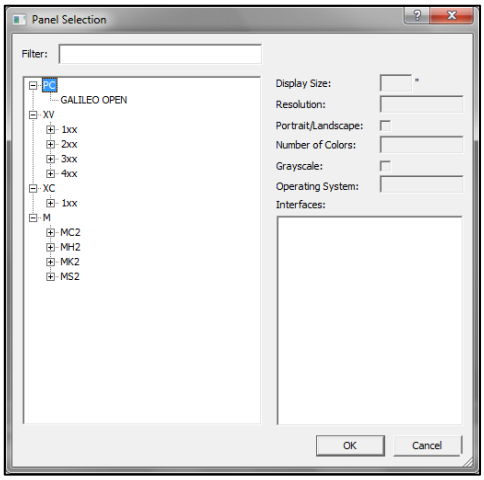


Figura 62 - Janela para a escolha do modelo da IHM.
Fonte: Autoria própria.

- Para saber qual o modelo da IHM, deve-se ligar a mesma e no menu iniciar, ir em “*Program/Control Panel*”. A figura 63 mostra o local do painel de controle:

Figura 63 - Localização do item *"Panel Control"*.
Fonte: Autoria própria.

- Na tela do painel de controle, dar duplo-clique em “*System”,* indicado na figura 64*:*

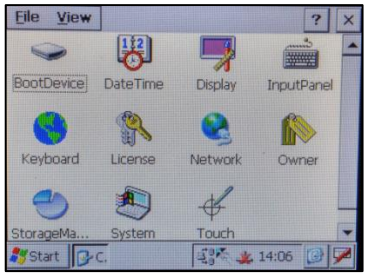


Figura 64 - Local do item "*System*".
Fonte: Autoria própria.

- Abrirá a tela com nome “*System Properties”*, com as especificações da IHM, como o processador, o sistema operacional, entre outros. O modelo da IHM está ao lado do item “*Panel Type*”. Pela figura 65, o modelo utilizado é o XV-102-B2-35TQR-10.

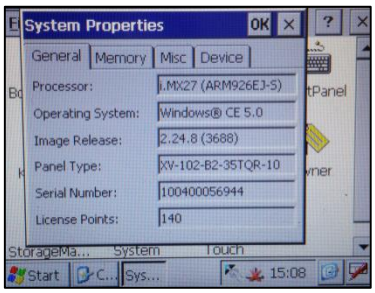


Figura 65 - Especificações da IHM.

Fonte: Autoria própria.

- De volta ao *software* Galileo, na tela “*Panel Selection”*, o modelo pode ser encontrado maximizando os itens “XV”, “1xx”, “color”, “10x”. Alternativamente, pode-se utilizar a função “*Filter*” (“Filtro”) e digitar o código diretamente para localizá-lo;

- Após a escolha do modelo do painel, na janela “*Panel Type”* irá aparecer quais painéis o projeto será compatível, quantas cores o painel pode mostrar e qual o sistema operacional presente no painel. A figura 66 mostra a janela “*Panel Type”*:

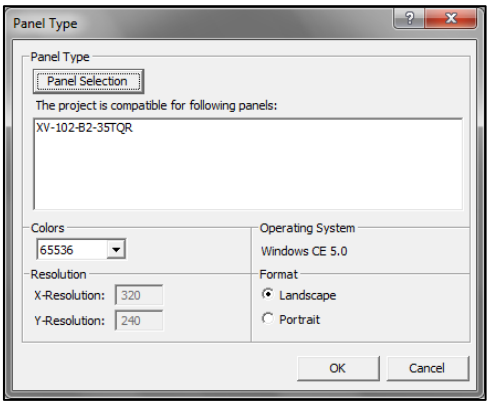


Figura 66 - Janela com o painel escolhido.

Fonte: Autoria própria.

- Escolhido o painel, abrirá uma janela nomeada “*Select PLC*”, ilustrada na figura 67, onde se deve clicar em “*add”* para a escolha da fabricante e o tipo de comunicação utilizado.

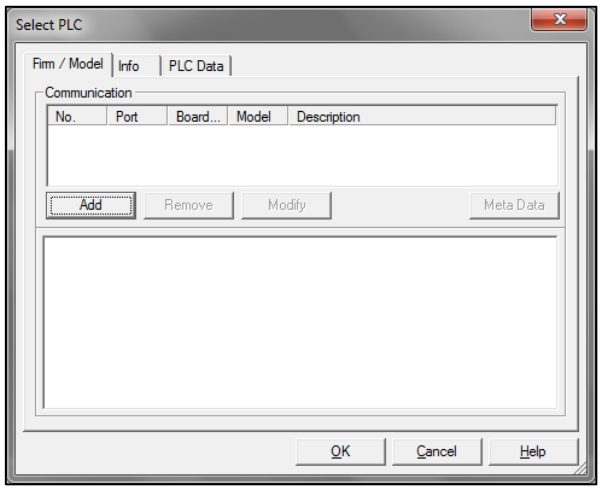


Figura 67 - Janela de escolha do modelo do CLP.
Fonte: Autoria própria.

- Na janela “*Select Communication”*, em *“Model”* abrirá uma lista para a escolha do modelo e a rede utilizada. O modelo utilizado na bancada é da fabricante Siemens, sendo escolhida a rede PROFIBUS *Standard Profile.* A figura 68 demonstra a janela com a marca escolhida e sua comunicação:

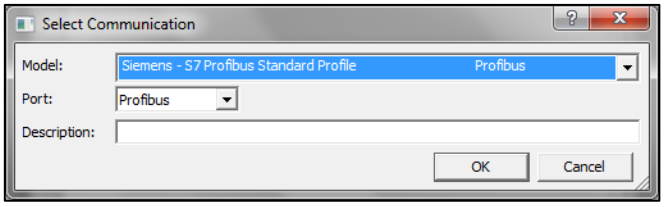


Figura 68 - Janela da escolha do fabricante e comunicação do CLP.
Fonte: Autoria própria.

- De volta na janela “*Select Model”*, surgirá às especificações da comunicação escolhida na parte inferior da tela. Deve-se verificar se os parâmetros estão de acordo com a especificação do CLP, A figura 69 mostra a tela com as propriedades do controlador escolhido:

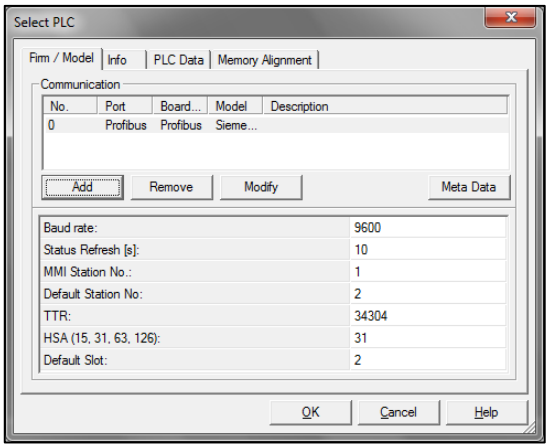


Figura 69 - Janela com o modelo do CLP escolhido.

Fonte: Autoria própria.

- Para verificar os parâmetros do CLP, deve-se abrir o *software* STEP7, clicar na aba “SIMATIC *300(1)*” e abrir o bloco “*Hardware*”, indicado na figura 70:

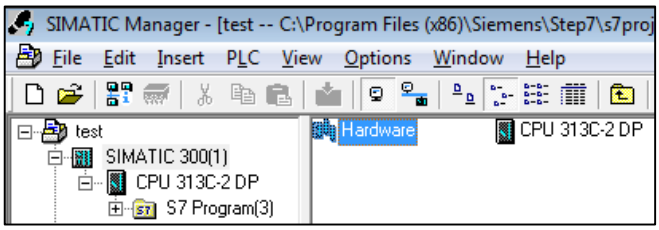


Figura 70 - Localização do item *"Hardware"* no STEP7*.*

Fonte: Autoria própria.

- Na tela “*HW Config”,* clicar no modulo DP, dentro da janela “(0) UR”. A figura 71 mostra a janela do *Rail(0)*, com os módulos selecionados:

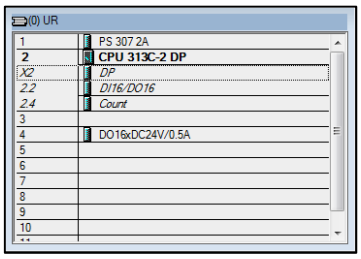


Figura 71 - Janela do *Rail* 0.

Fonte: Autoria própria.

- Na janela “*Properties”*, clicar no botão “*Properties...*” dentro do retângulo onde está escrito “*Interface”.* A figura 72 indica o local do botão:

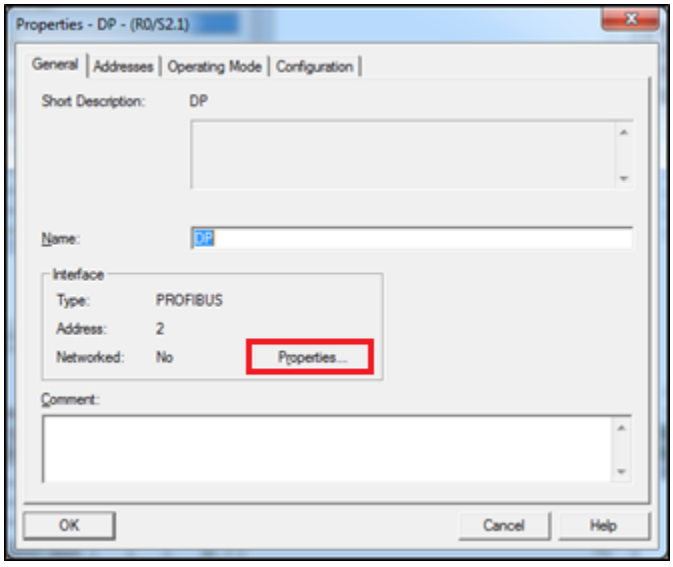


Figura 72 - Janela de propriedades do CLP.

Fonte: Autoria própria.

- Abrirá a janela “*Properties* – PROFIBUS *interface* DP *(R0/S2.1)”*. Mostrado na figura 73. Deve-se clicar em “*New...”*, para criar um novo *subnet* PROFIBUS.

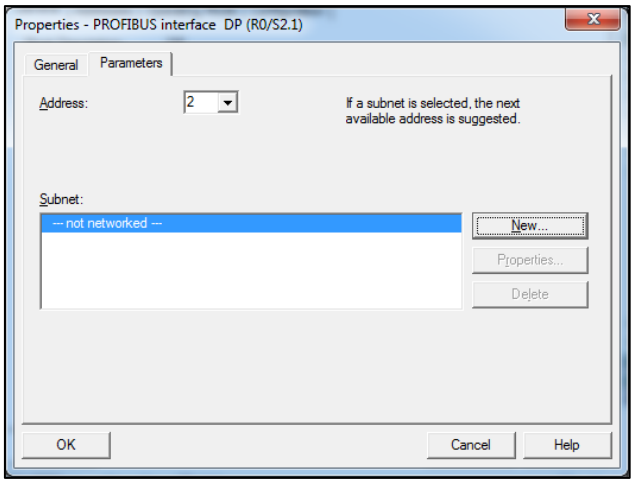


Figura 73 - Janela de propriedades da comunicação do CLP.

Fonte: Autoria própria.

- Na aba “*Network Setting”* dentro da janela “*Properties – New subnet* PROFIBUS*”,* pode-se escolher os parâmetros da comunicação, como mostra a figura 74:

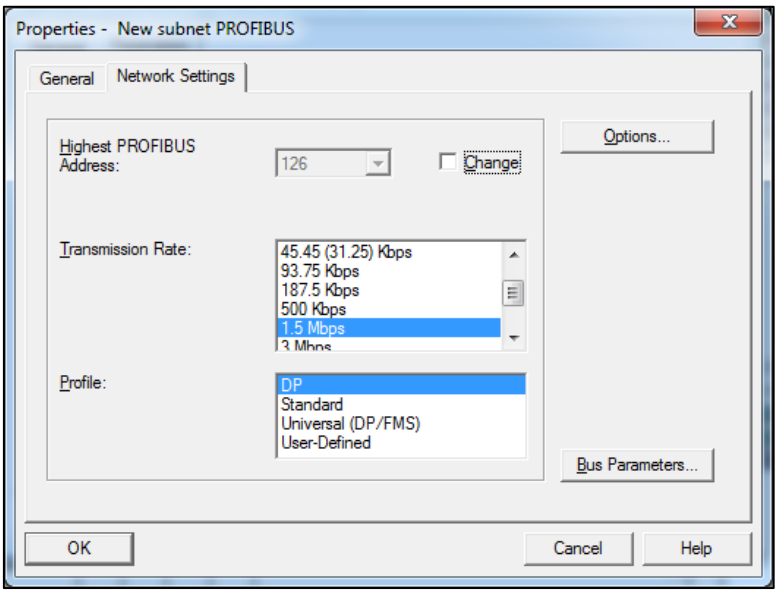


Figura 74 - Janela com os parâmetros da comunicação PROFIBUS*.*
Fonte: Autoria própria.

No item “*Highest* PROFIBUS *Adress:” ou HSA,* pode-se escolher qual o maior endereço do sistema. Só se altera o valor desse parâmetro, quando há outro dispositivo ativo.

Em “*Transmission Rate”,* pode-se alterar a taxa de transmissão de dados. A taxa de transmissão deve ser escolhida de acordo com comprimento do cabo.

O quadro 1 mostra o comprimento máximo do cabeamento de acordo com o “*Baud Rate”:*

Baud Rate (kb/s)	9,6	19,2	93,75	187,5	500	1500	2000
Comprimento (m)	1200	1200	1200	1000	400	200	100

Quadro 1 - O comprimento do cabo de acordo com o taxa de transmissão.
Fonte: http://www.smar.com/brasil/profibus

Como na bancada, o CLP e a IHM estão próximas, deixa-se deixar o valor padrão.

A figura 75 mostra um comparativo dos parâmetros do CLP (tela na esquerda da figura) com a janela da IHM (tela na direita da figura)

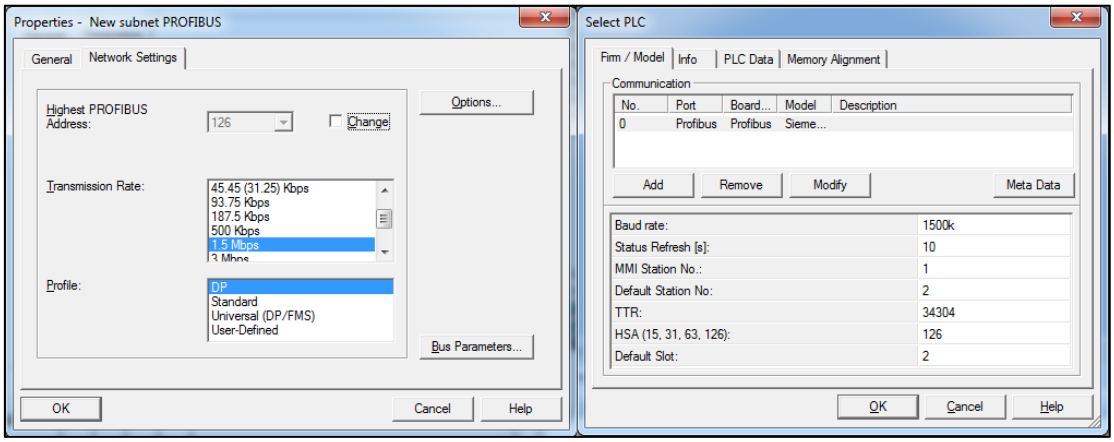


Figura 75 - Comparativo entre os parâmetros do CLP e da IHM.
Fonte: Autoria própria.

- Após configurar os parâmetros, clicar em "*OK*" no *software* Galileo, para iniciar a programação da IHM.

3.4.2. Programação da IHM

- Depois de adaptar o projeto, deve-se criar *tags*. Cada objeto na máscara que processam dados utiliza uma *tag.* Ela pode ser endereçada e representa o valor ou o bit do CLP. Através dela, é possível realizar um comando ou sinalização pela IHM. Se uma *tag* não for endereçada, ela não é processada pelo CLP, apenas internamente na IHM.

- Para criar *tags* é necessário clicar no ícone indicado na figura 76

Figura 76 - Localização do ícone para criação de *tags*.
Fonte: Autoria própria.

- Ao clicar no ícone, aparecerá uma lista com os tipos de *tags*. A figura 77 lista os tipos de tags apresentado pelo *software:*

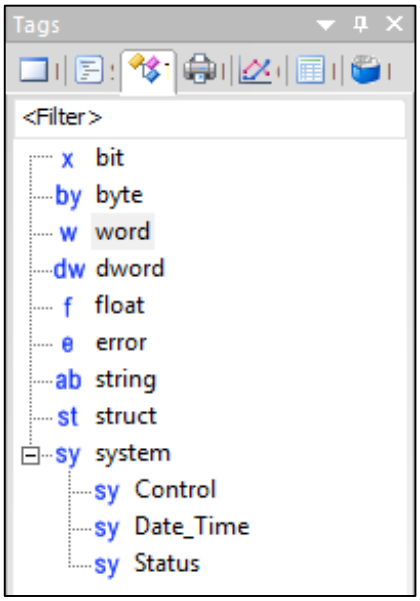


Figura 77 - Lista de tipo de *tags*

Fonte: Autoria própria

- A figura 78 mostra descrição dos tipos de *tags:*

Dados	Binário	Decimal	Decimal +/-	BCD
Bit	0 ou 1	X	X	X
Byte	0 a 255	0 a 255	-127 a 127	0 a 153
Word	0 a 65535	0 a 65535	-32768 a 32767	0 a 39321
DWord	0 a 2^{32}	0 a 2^{32}	-2^{31} a $2^{31}-1$	X
Float		X	+/- $10^{+/-308}$	
Error	Dado utilizado para configuração de alarmes e é do tipo Bit			
CharArray	Dado tipo texto enviado do CLP			
System	Dados do sistema de comunicação, estado do CLP e dados de data e hora do CLP			

Figura 78 - Descrição das *tags.*

Fonte: Viana; Bim, 2012.

- Após escolher um dos tipos, clicar com o botão direito do *mouse.* Aparecerá uma lista com várias opções. Pode-se criar uma *tag* individual, ou uma

array de tags, clicando na opção *"New Tag"* ou *"New Array".* A figura 79 mostra as opções:

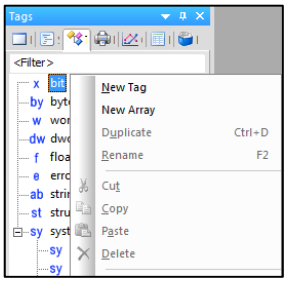


Figura 79 - Lista de opções para criar *tag*.

Fonte: Autoria própria.

- Ao criar a *tag*, deve-se clicar duas vezes no mesmo, para abrir a janela de propriedades, ilustrada na figura 80.

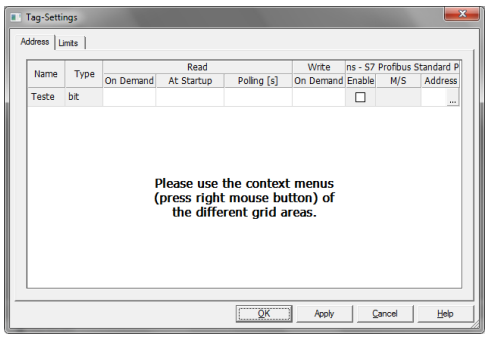


Figura 80 - Janela de propriedades da *tag.*

Fonte: Autoria própria.

- Na janela de propriedades, clicar no botão "...", abrindo uma nova janela onde será configurado o endereço da *tag.* A figura 81 ilustra a janela de configuração:

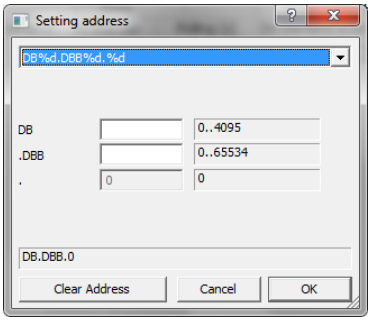


Figura 81 - Janela "*Setting address".*

Fonte: Autoria própria.

- Deve-se escolher o DB (*Data Block*) criado no programa do CLP e qual *byte* será endereçado a *tag.* Não é possível escolher o *bit* especifico, para isso é necessário criar uma *array.*

- Endereçado todas as *tags,* é necessário criar uma tela onde serão colocados os botões de comando e os displays de informações é necessário clicar na aba "*Mask",* onde serão mostrados os tipos de telas disponíveis pela IHM, ilustrado na figura 82:

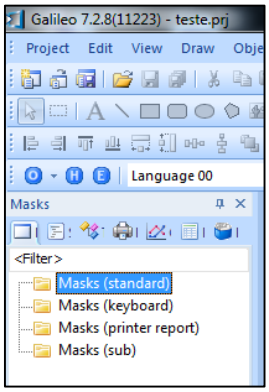


Figura 82 - Localização do item "*Mask*".
Fonte: Autoria própria.

- Deve-se clicar com o botão direito do *mouse* na opção "*Mask (standard)/New"* e digitar um nome para a tela. A figura 83 exemplifica um nome para a *Mask:*

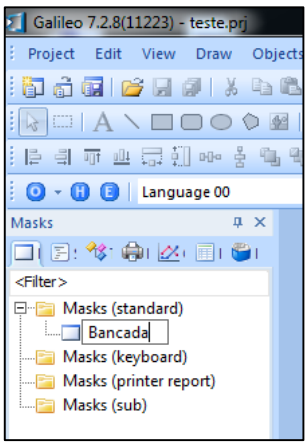


Figura 83 - Escolhendo um nome para a *mask.*
Fonte: Autoria própria.

- O Galileo automaticamente mostra a mascara criada. É habilitado também a barra de ferramentas na parte superior da tela, como mostra a figura 84:

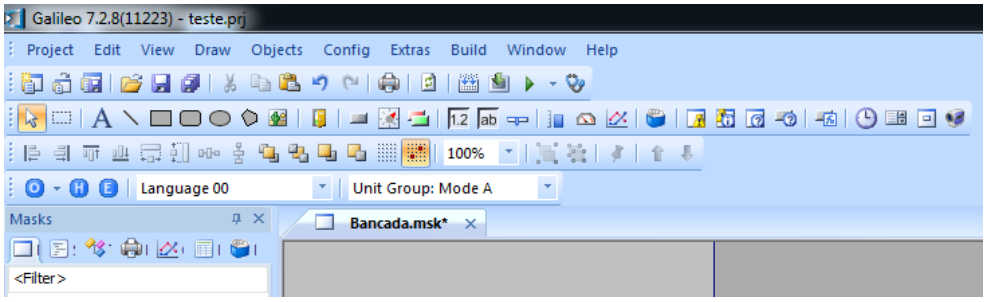


Figura 84 - Tela exibindo a máscara criada.

Fonte: Autoria própria.

- A figura 85 ilustra os botões e a função dos mesmos:

N	Funções dos botões da barra de ferramenta
1	Seleção de objetos
2	Seleção de uma área da tela
3	Criar texto
4	Criar linhas
5	Criar retângulos
6	Criar retângulos com cantos arredondados
7	Criar círculos
8	Criar polígonos
9	Importar uma imagem no formato bitmap
10	Criar botão de troca de *mask*
11	Inserir botão de comando
12	Inserir interruptor
13	Inserir *display* da sinalização
14	Inserir *display* de valor
15	Inserir *display* de texto
16	Inserir *slider*
17	Criar gráfico de barras
18	Criar medidor
19	Criar histogramas
20	Criar receitas
21	Inserir janela de erro
22	Inserir janela de erro estendida
23	Inserir janela de ajuda
24	Inserir botão de ajuda
25	Criar um botão de funções especiais
26	Criar um *display* de data e tempo
27	Criar uma lista de parâmetros
28	Inserir uma *sub-mask*
29	Inserir imagens importados de uma câmera

Figura 85 - Itens da barra de ferramentas.

Fonte: Autoria própria.

Para adicionar um item, deve-se:

- Escolher a tela, ou *mask*, desejada;
- Escolher o item da barra de ferramenta;
- Arrastar com o cursor do *mouse*, definindo o tamanho do objeto.

- Após criar o objeto, é necessário clicar duas vezes no mesmo, abrindo uma janela de configuração do mesmo, ilustrado na figura 86:

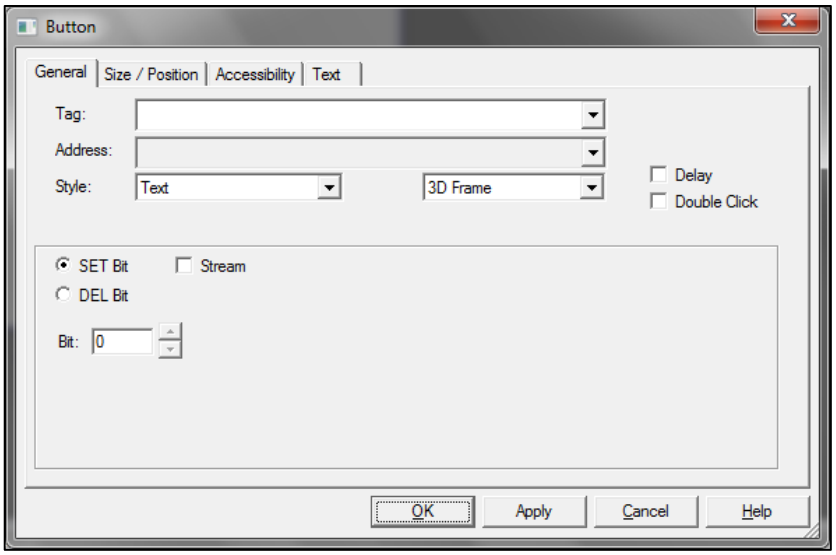


Figura 86 - Tela de propriedades do item escolhido.

Fonte: Autoria própria.

- Na aba "*General"* no item *"Tag"*, é necessário associa-lo a uma *tag.* Para alterar a posição do item, deve-se acessar a aba "*Size/Position"*. Na aba "*Accessability"* é o local onde se configura quando o botão é visível. Para adicionar um texto ou alterar a cor do elemento, deve-se acessar a aba "*Text"*.

OBS: É obrigatório criar o botão função especial "*Shut Down",* pois não há outra forma de sair da tela do programa na IHM, sem abrir a bancada. Se for esquecido, deve-se abrir a bancada e apertar o botão *reset* da IHM,

3.4.3. Compilando e fazendo download do projeto na IHM

- Quando o programa estiver pronto para ser testado na IHM, deve-se compilar para verificar se não há erros. Para compilar deve-se apertar o botão localizado na parte superior, indicado na figura 87:

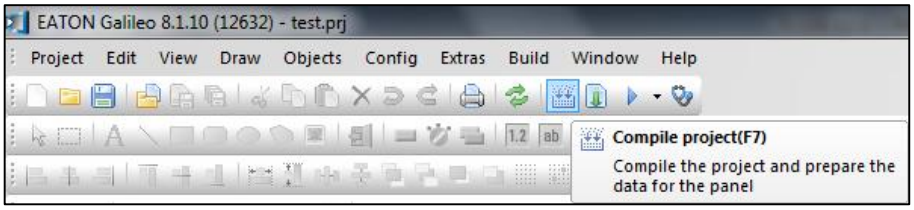


Figura 87 - Localização do botão para compilar ma barra de ferramenta.
Fonte: Autoria própria.

- Ou na aba "B*uild/Compile",* ilustrado na figura 88:

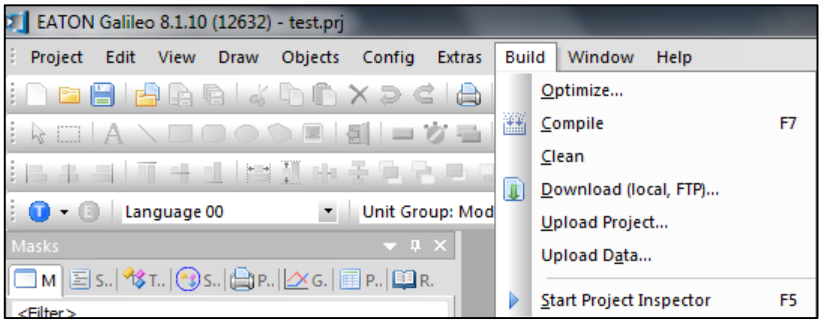


Figura 88 - Localização do item "*Compile*".
Fonte: Autoria própria.

- Após compilar, na parte inferior do *software* mostra o diagnóstico. Se houver algum erro, será mostrada uma mensagem indicando qual o erro, como na figura 89:

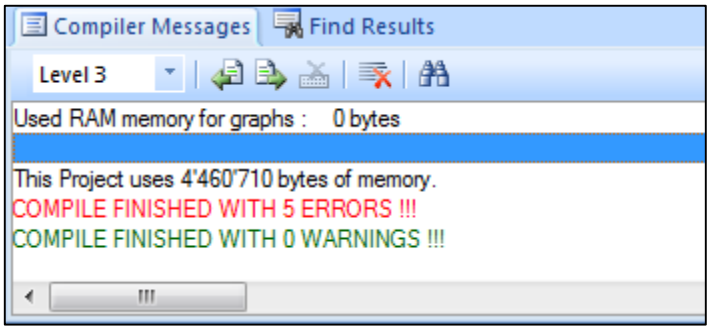


Figura 89 - Tela de erro.
Fonte: Autoria própria.

- Se não houver nenhum erro, aparecerá uma mensagem indicando o sucesso, mostrado na figura 90:

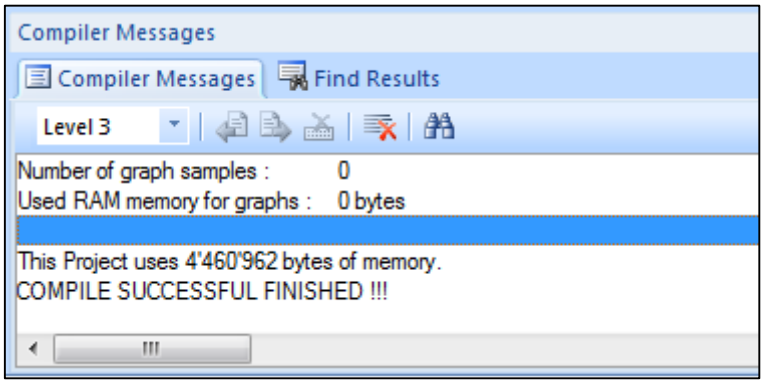


Figura 90 - Tela informando o sucesso do processo de compilar.
Fonte: Autoria própria.

- Depois de compilar com sucesso, é necessário fazer *download* do projeto. Para isso é preciso configurar o IP da IHM. Antes de tudo, o painel deve estar conectado com o computador através de um cabo *Ethernet.* Para verificar o IP, deve-se abrir o *prompt* de comando, localizado no "Menu Iniciar/Acessórios" mostrado na figura 91:

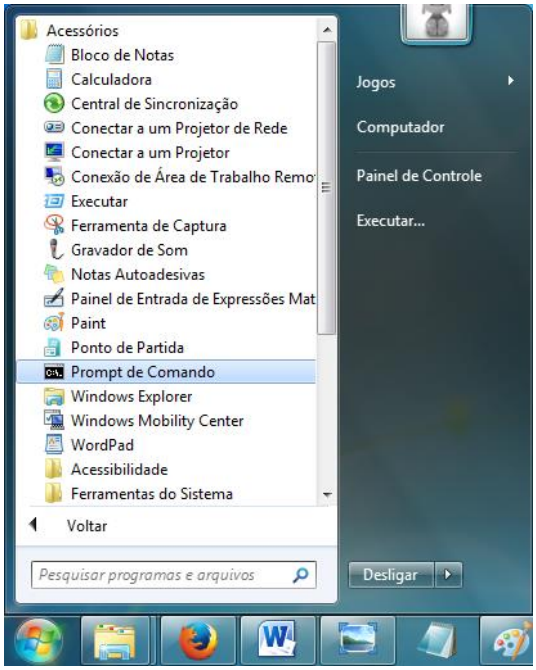


Figura 91 - Localização do *Prompt de Comando.*
Fonte: Autoria própria.

- No *prompt de comando,* digitar o comando ipconfig, para mostrar a configuração do IP. A figura 92 mostra o *prompt* de comando:

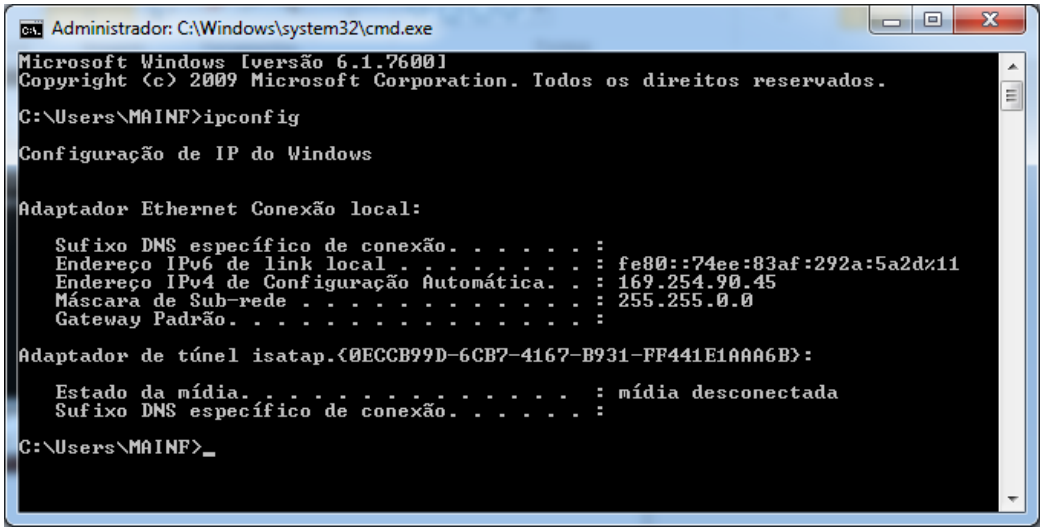


Figura 92 - *Prompt* de comando com as configurações de IP do computador.
Fonte: Autoria própria.

- O endereço de IP e da máscara de Sub-rede, mostrado no *prompt* de comando, deve ser anotado, pois será utilizado mais tarde. Para configurar o valor do IP da IHM, é necessário entrar no painel de controle, localizado em "*Menu Iniciar/Programs".* A figura 93 mostra a localização do painel de controle na IHM:

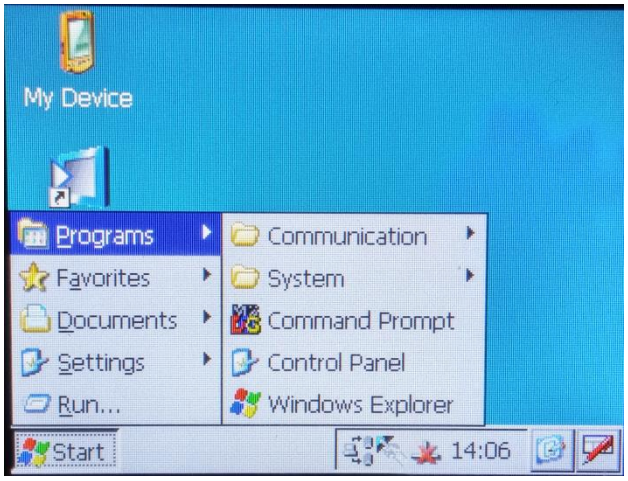


Figura 93 - Localização do painel de controle.
Fonte: Autoria própria.

- Na janela “*Panel Control”,* mostrada na figura 94, apertar o ícone “*Network”* que abrirá outra janela.

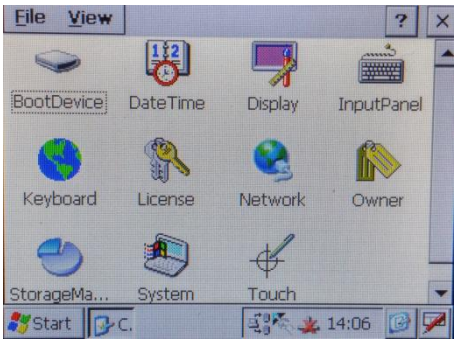


Figura 94 - Localização do item "*Network".*

Fonte: Autoria própria.

- Na tela que se abriu, clicar em “*ONBOARD1”.* A figura 95 indica o item citado:

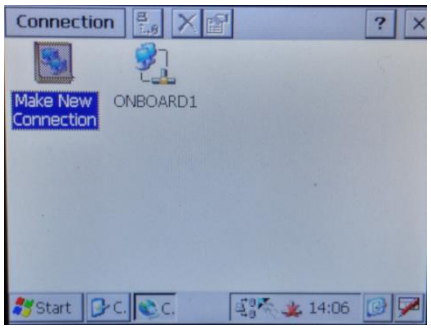


Figura 95 - Localização do item "*ONBOARD1".*

Fonte: Autoria própria.

-Na janela “*FEC Ethernet Driver*” que abriu, ilustrado na figura 96, deve-se colocar o endereço de IP e máscara de *Subnet* anotado, porém com o ultimo digito diferente para não haver interferência na hora de comunicar.

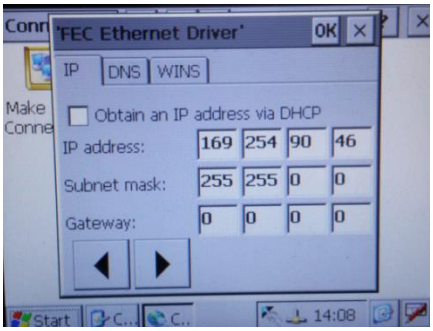


Figura 96 - Tela de endereçamento do IP.

Fonte: Autoria própria.

-Após configurar a comunicação da IHM e do computador, é possível fazer o *download* do programa. Para fazer isso é necessário clicar no botão de *download* na parte superior do *software* Galileo. A figura 97 mostra o local do botão

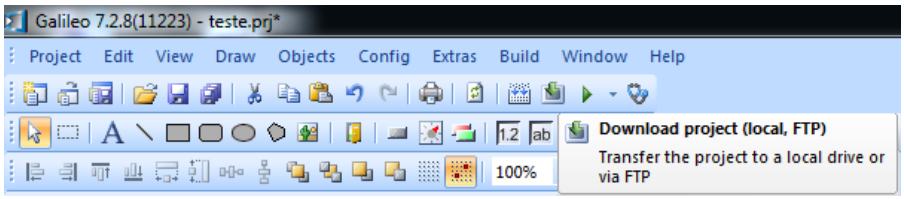


Figura 97 - Localização do botão de *download*

Fonte: Autoria própria

- Depois de apertar o botão, irá abrir uma janela. Para fazer o download é necessário criar um caminho para o *FTP.* Para isso, deve-se clicar no botão "*FTP Path".* A figura 98 indica o local do botão citado:

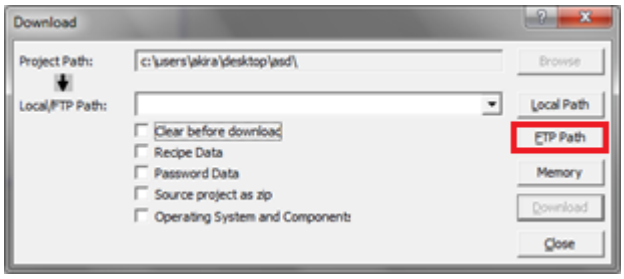


Figura 98 - Janela de *download* do Galileo*.*

Fonte: Autoria própria.

- Abrirá a janela "*FTP-connection",* onde deve-se criar uma nova conexão, clicando no botão "*New Connection".* A figura 99 mostra janela citada:

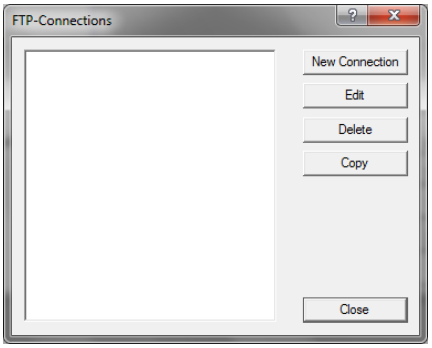


Figura 99 - Janela "*FTP-Connections*".

Fonte: Autoria própria.

- Na nova tela que abriu, no item “*Title*” coloca-se o nome da conexão e no item “*Server / IP-Adress”* deve-se colocar o numero do endereço do IP igual ao que foi colocado na IHM. A figura 100 mostra a janela com as propriedades do *FTP.*

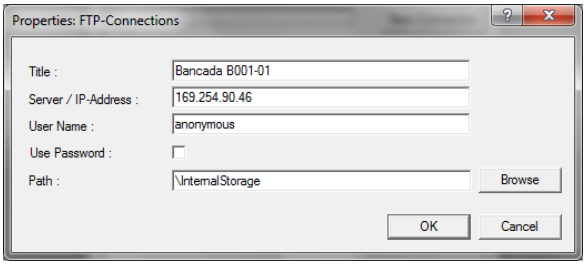


Figura 100 - Propriedades da conexão *FTP.*
Fonte: Autoria própria.

-Antes de fazer o *download*, é preciso que o programa *FTP Server* esteja aberto no painel da IHM. O programa se localiza no “Menu Iniciar/*Programs/Communication*”. A figura 101 mostra a localização do *FTP Server:*

Figura 101 - Localização do programa *FTP Server.*
Fonte: Autoria própria.

- Com o *FTP Server* aberto, é possível oculta-lo na barra de tarefas, apertando o botão *“Hide*” indicado na figura 102:

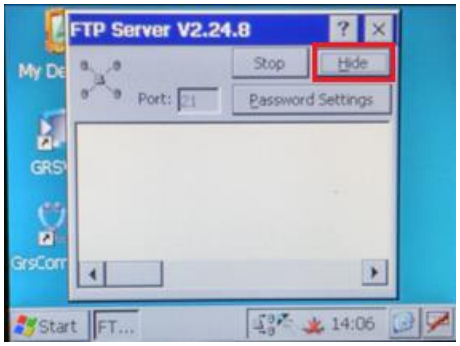


Figura 102 - Painel da IHM com o *FTP Server* aberto.
Fonte: Autoria própria.

- Após abrir o *FTP Server,* é possível fazer o *download.* Para isso basta apertar o botão de *download* na janela do Galileo. Se a comunicação for feita corretamente irá aparecer uma tela indicando que é necessário que o painel esteja com o sistema operacional *Windows CE 5.0: Core.* Para dispensar a tela de aviso, ilustrado na figura 103, basta apertar "*OK".*

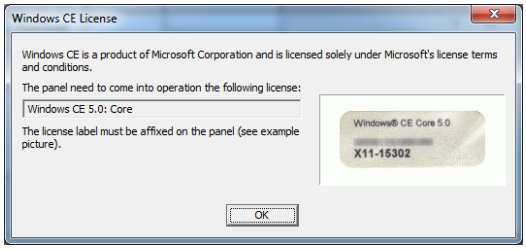


Figura 103 - Tela de aviso.

Fonte: Autoria própria.

- Depois de dispensar o aviso, aparecerá uma tela para escolher as opções de reiniciar a IHM ("*HMI reboot")* ou apenas iniciar a aplicação ("Start GRS"). A figura 104 mostra a tela:

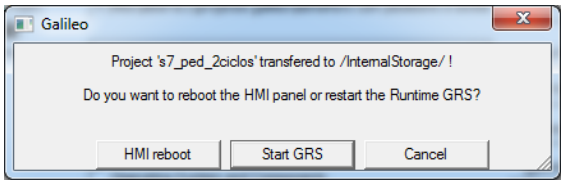


Figura 104 - Tela de escolha.

Fonte: Autoria própria.

Após isso, o download será feito e o programa será automaticamente carregado na IHM.

4. CONSIDERAÇÕES FINAIS

Este trabalho conseguiu resolver satisfatoriamente todos os seus objetivos específicos e problemas propostos, baseados na premissa de que a rede PROFIBUS é um protocolo aberto e pode fazer a interoperabilidade de equipamentos de diferentes fabricantes.

A bancada didática foi extensamente testada e analisada, pesquisando-se todos os itens que a compõe e a funcionalidade dos mesmos separadamente e em suas posições na bancada. Por vezes teve-se que abrir a bancada para analisar a ligação elétrica dos componentes e acessar as partes cobertas dos dispositivos.

O programa Simatic STEP7 foi pesquisado com diversas referências em manuais e trabalhos acadêmicos, a maioria em inglês, além de muita investigação sobre todas as funcionalidades do programa. Foi possível analisar suas características de programação, configuração de projetos, configuração de *hardware*, redes PROFIBUS, blocos de funções e de organização, tabelas de variáveis, uso do simulador e *download* para o CLP com uso de interface USB. Muitos erros foram encontrados em diversas etapas até que se encontrasse a solução correta, e os mais comuns foram descritos no manual didático junto com sua solução.

A plataforma de programação Galileo também foi extensamente investigada para, primeiramente descobrir todas as suas funções e aplicações, e em outro momento para descobrir como operar na rede PROFIBUS e compartilhar os dados que estavam sendo analisados no CLP. A parte de comunicação da IHM com o computador para fazer o *download* das telas também foi um caso com muitas dificuldades, pois é necessária uma configuração de rede Ethernet para esta tarefa. Contou-se com a ajuda de outros trabalhos acadêmicos para resolver este entrave.

Dois exemplos de programas com comentários foram colocados à disposição como apêndices neste trabalho. O primeiro é um semáforo com diversos pontos de luz vermelha e verde, que se apagam em ordem e indicam o tempo aproximado para o semáforo alterar entre os estados de aberto e fechado, como ocorre na cidade de São José dos Pinhais no estado do Paraná. Para a implementação, foram utilizados blocos do tipo *Shift Register* e temporizadores. Uma tela de supervisório com um espelho das saídas digitais foi criada no laboratório. O segundo programa consiste em um cruzamento de duas ruas, com semáforos para

carros e também para pedestres, que podem ser solicitados através de um botão para pedestres. Neste exemplo, foram criadas duas telas na IHM: uma contendo a representação das ruas e dos semáforos, e na outra as memórias do CLP em uso e possibilidade de *reset* de temporizadores.

A parte de comunicação entre CLP e IHM através da rede PROFIBUS exigiu bastante pesquisa, e foi alcançada com base em muitos trabalhos acadêmicos e publicações na internet, a maioria em inglês também.

O objetivo geral de desenvolver um manual didático para a bancada utilizada no laboratório de CLP da UTFPR foi alcançado com êxito. Foram descritos os procedimentos iniciais sobre como energizar a bancada e cuidados ao operá-la, passando pela programação e configuração do CLP e IHM, até a parte de *download* e comunicação entre os dispositivos em rede industrial PROFIBUS.

Com este projeto, também se verificou que a rede PROFIBUS pode ser utilizada para conectar diversos Controladores Lógicos Programáveis e diversas bancadas entre si, o que pode ser um trabalho de conclusão de curso futuro.

REFERÊNCIAS

ALBUQUERQUE, Pedro U. B. de. **Redes industriais: Aplicações em sistemas digitais de controle distribuído protocolos industriais, aplicações SCADA,** 2ed. São Paulo: Ensino Profissional, 2009.

BERGER, Hans. **Automating with SIMATIC S7-300 inside TIA Portal - Configuring, Programming and Testing with STEP7 Professional**, 2.ed. Berlin, Munique: Publicis Publishing, 2014.

CASSIOLATO, César; TORRE, Ana C. D. Uma visão de PROFIBUS, desde a instalação até a configuração básica. **Revista Controle e Instrumentação,** ed.129. São Paulo, 2007. Disponível em: <http://www.profibus.org.br/artigos_tecnicos/uma-visao-de-profibus-desde-a-instalacao-ate-a-configuracao-basica-parte-1>. Acesso em 01 mai. 2016

FRANCHI, Claiton M.; CAMARGO, Valter L. A. de. **Controladores lógicos programáveis: sistemas discretos**, 2. ed. São Paulo: Érica, 2009.

GEORGINI, Marcelo. **Automação Aplicada - Descrição e Implementação de Sistemas Seqüeciais com PLC**, 5. ed. Tatuapé: Érica, 2003.

GUTIERREZ, Regina M. V.; PAN, S. S. K. **Complexo Eletrônico: Automação do Controle Industrial**, 2008. Disponível em: <http://www.bndes.gov.br/SiteBNDES/export/sites/default/bndes_pt/Galerias/Arquivos/conhecimento/bnset/set2807.pdf>. Acesso em: 01 mai. 2016.

LEME, Murilo. Notas de aula da disciplina de Redes Industriais, Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2011. Disponível em: <http://www.ebah.com.br/content/ABAAAfOgQAB/redes-industriais>. Acesso em: 02 mai. 2016.

MELO, Aguinaldo Goes de; MOSCARDI, Rodrigo. **Desenvolvimento de uma bancada didática de controle de temperatura**. 2005. 169 f. Trabalho de Conclusão

de Curso (Graduação) – Centro Federal de Educação Tecnológica do Paraná, Curso Superior de Engenharia Industrial Elétrica, Ênfase em Eletrotécnica, Curitiba, 2005.

MOLODOWSKI, Daniel Lucchesi; OLIVEIRA, Ronaldo Adriano. **Projeto de um sistema supervisório para auxiliar o controle de manutenção e produção de uma empresa do setor cerâmico.** 2010. P111. Trabalho de Conclusão de Curso (Graduação) – Universidade Tecnológica Federal do Paraná, Curso Superior de Tecnologia em Eletrotécnica, Ênfase em Automação Industrial, Curitiba, 2010.

PRUDENTE, Francesco. **Automação industrial: PLC: Teoria e aplicação: curso básico**, 2.ed. Rio de Janeiro: LTC, 2013.

ROQUE, Luiz A. O. L. **Automação de processos com linguagem ladder e sistemas supervisórios,** 1.ed. Rio de Janeiro: LTC, 2014.

SILVA, Ana P. G.; SALVADOR Marcelo. **O que são sistemas supervisórios?**, 2011. Disponível em: <http://kb.elipse.com.br/pt-br/questions/62/O+que+s%C3%A3o+sistemas+supervis%C3%B3rios%3F>. Acesso em: 28 set. 2016.

SOUZA, Luiz E. de. Controladores Lógicos Programáveis. Notas de aula. Curso promovido pela Fundação de Pesquisa e Assessoramento a Indústria. Disponível em: < http://www.ebah.com.br/content/ABAAAer7YAG/apostila-clp>. Acesso em: 15 abr. 2016.

STEMMER, Marcelo R. **Das 5331 - sistemas distribuídos e REDES de computadores para controle e AUTOMAÇÂO industrial**, 2001. DIsponível em: <http://alvarestech.com/temp/simprebal/Relatorios_Tecnicos-Publicacoes-Dissertacoes/docs/cursos/Aula6-Apostila-Sistemas_Distribuidos_E_Redes_De_Computadores_Para_Controle.pdf> Acesso em: 02 mai. 2016.

APENDICE A – PROGRAMA DO STEP7: SEMÁFORO

OB1 - <offline>

""

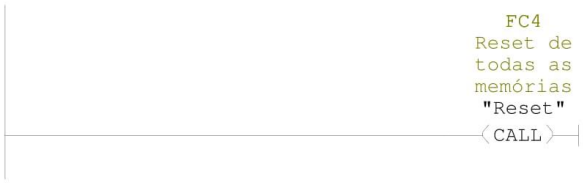
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/27/2016 11:03:19 PM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00136 00020 00020

Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

Block: OB1 "Main Program Sweep (Cycle)"

Network: 1

Network: 2



Network: 3

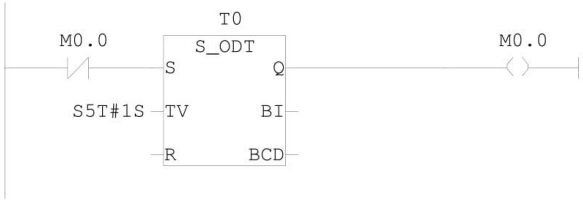
FC1 - <offline>

"Prog_Semaf" Programa do Semáforo
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/27/2016 11:16:43 PM
Interface: 06/15/2016 11:19:30 AM
Lengths (block/logic/data): 00718 00582 00006

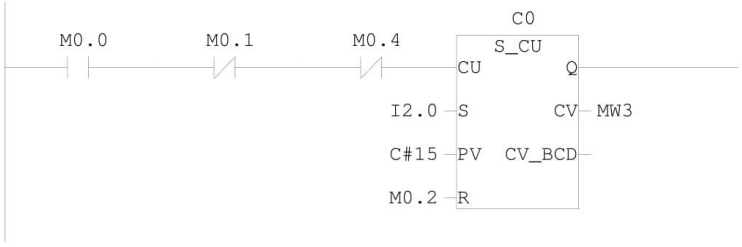
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC1

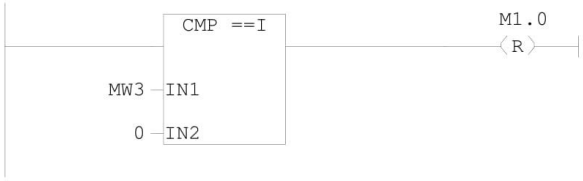
Network: 1



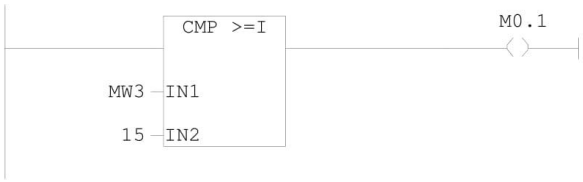
Network: 2



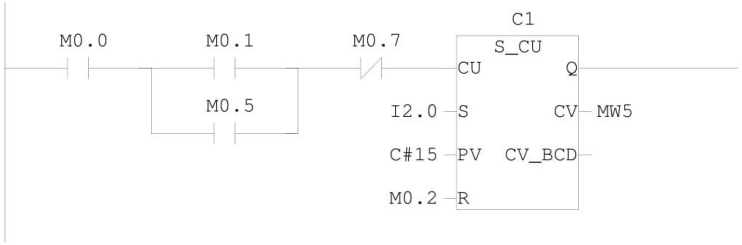
Network: 3



Network: 4



Network: 5

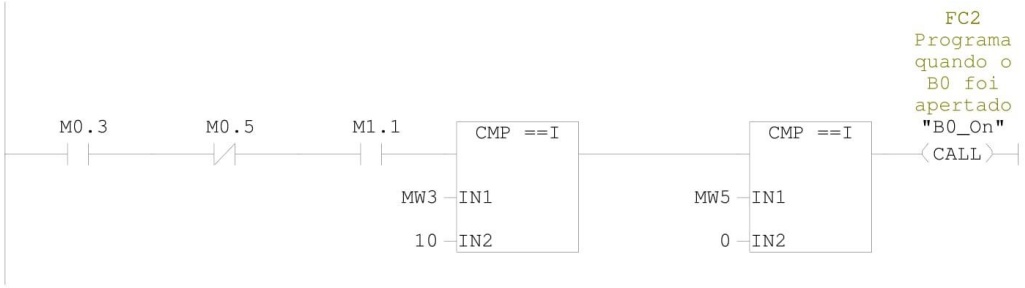


Network: 6

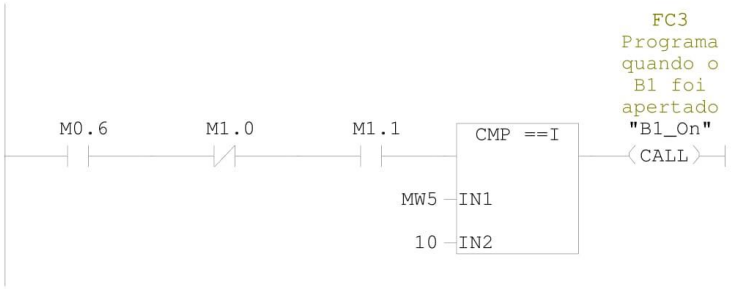
Network: 7

Network: 8

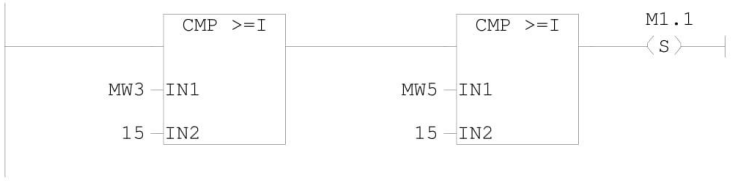
Network: 9



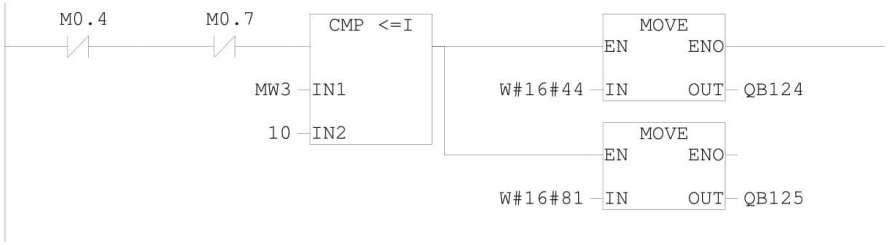
Network: 10



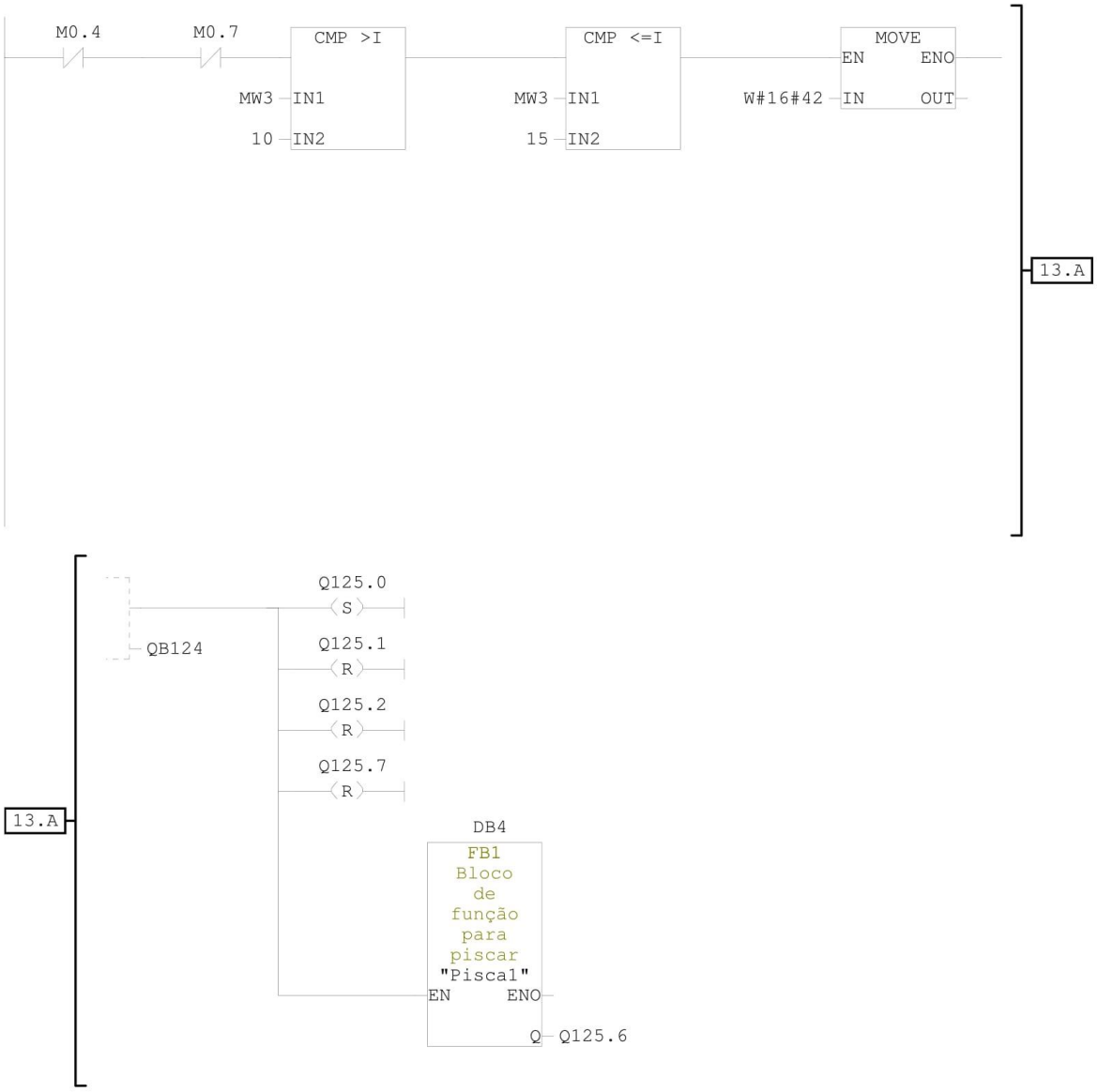
Network: 11



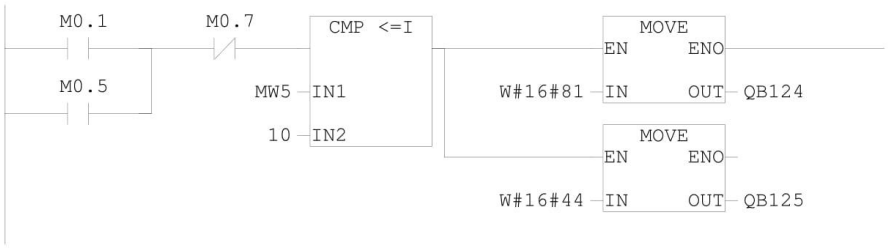
Network: 12



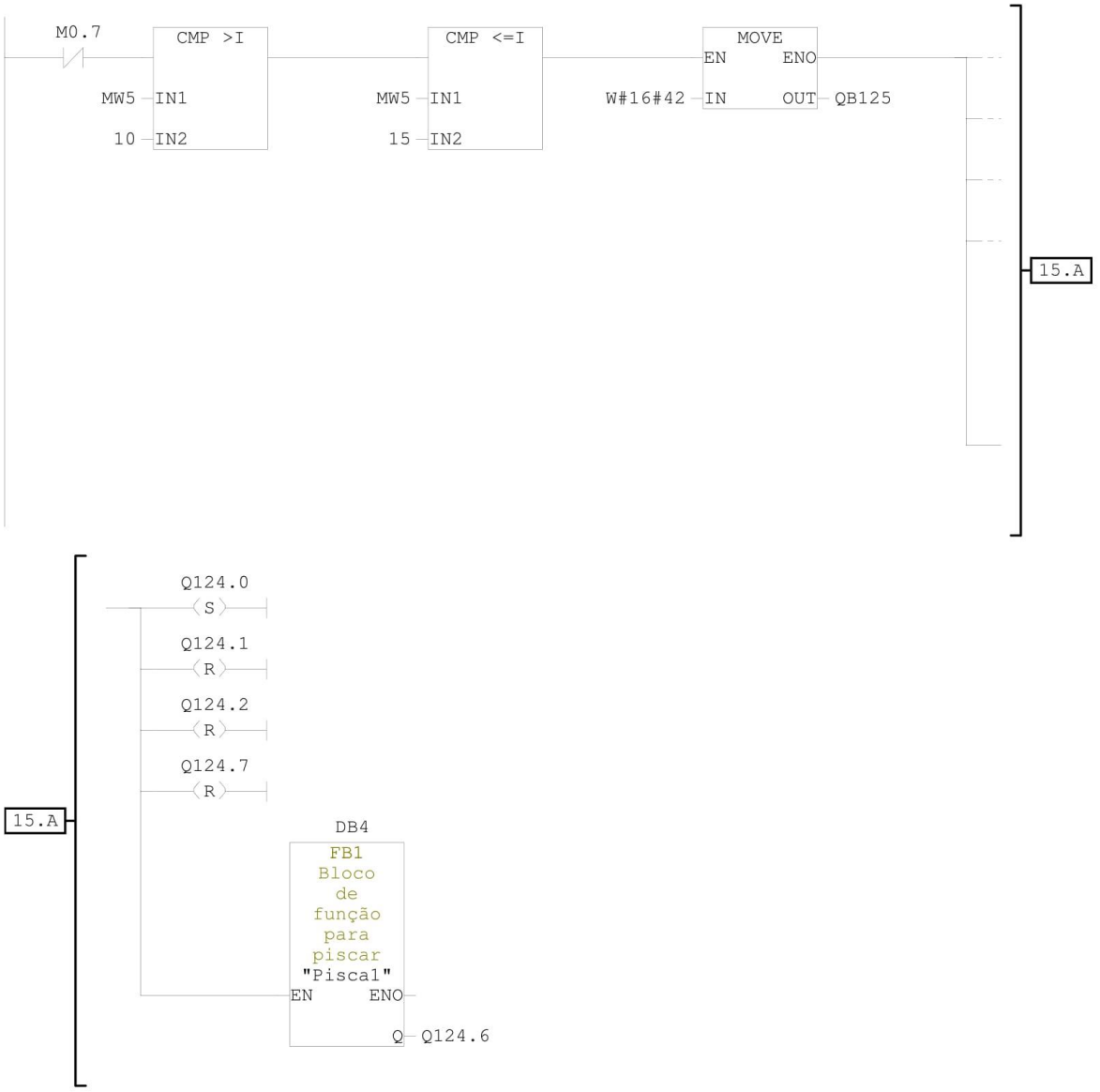
Network: 13


M0.4
M0.7
CMP >I
MW3
IN1
10
IN2
CMP <=I
MW3
IN1
15
IN2
MOVE
EN
ENO
W#16#42
IN
OUT
13.A
13.A
QB124
Q125.0
S
Q125.1
R
Q125.2
R
Q125.7
R
DB4
FB1
Bloco
de
função
para
piscar
"Pisca1"
EN
ENO
Q
Q125.6

Network: 14


M0.1
M0.7
CMP <=I
M0.5
MW5
IN1
10
IN2
MOVE
EN
ENO
W#16#81
IN
OUT
QB124
MOVE
EN
ENO
W#16#44
IN
OUT
QB125

Network: 15


M0.7
CMP >I
MW5
IN1
10
IN2
CMP <=I
MW5
IN1
15
IN2
MOVE
EN
ENO
W#16#42
IN
OUT
QB125
15.A
15.A
Q124.0
S
Q124.1
R
Q124.2
R
Q124.7
R
DB4
FB1
Bloco
de
função
para
piscar
"Pisca1"
EN
ENO
Q
Q124.6

FC2 - <offline>

"B0_On" Programa quando o B0 foi apertado

Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/28/2016 10:22:39 AM
Interface: 06/27/2016 10:08:28 AM
Lengths (block/logic/data): 00378 00264 00006

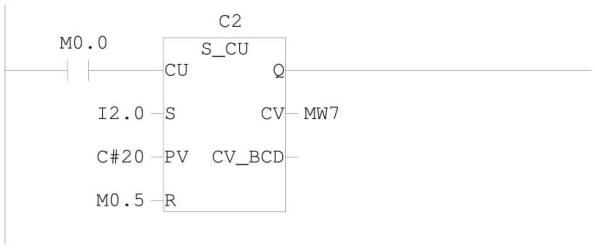
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC2

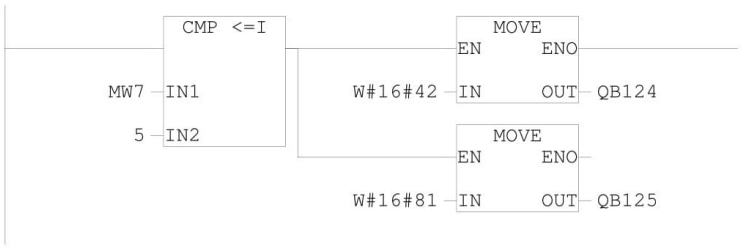
Network: 1

Network: 2

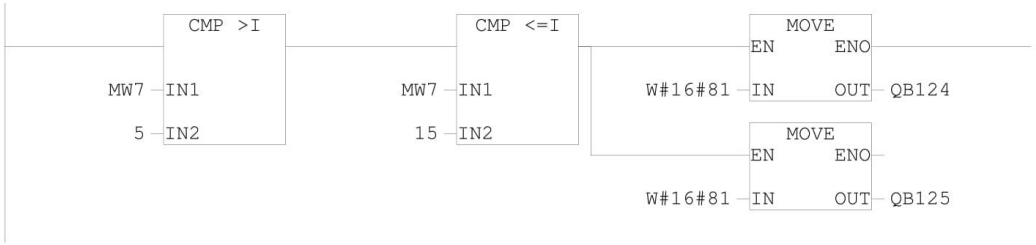
Network: 3



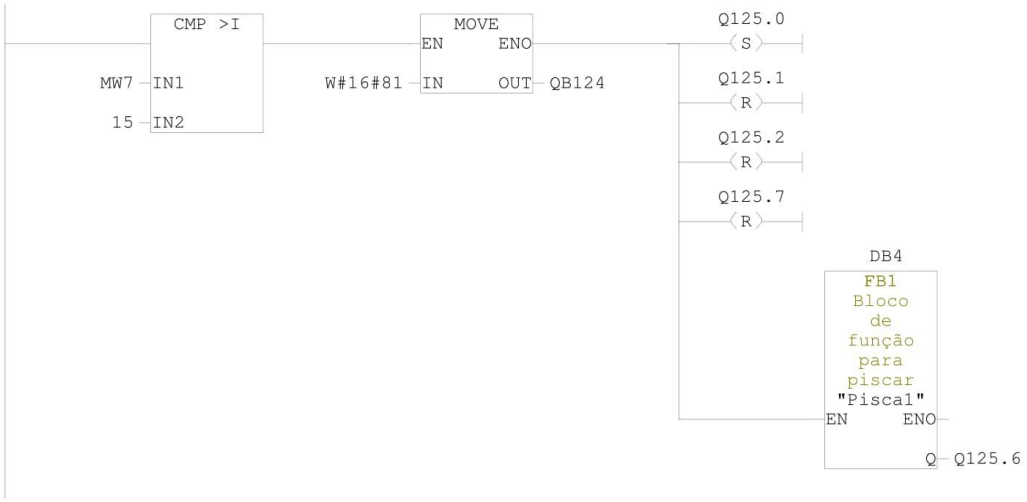
Network: 4



Network: 5



Network: 6


CMP >I
MW7
IN1
15
IN2
MOVE
EN
ENO
W#16#81
IN
OUT
QB124
Q125.0
S
Q125.1
R
Q125.2
R
Q125.7
R
DB4
FB1
Bloco
de
função
para
piscar
"Pisca1"
EN
ENO
Q
Q125.6

FC3 - <offline>

"B1_On" Programa quando o B1 foi apertado
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/28/2016 10:23:29 AM
Interface: 06/27/2016 10:40:25 AM
Lengths (block/logic/data): 00386 00272 00006

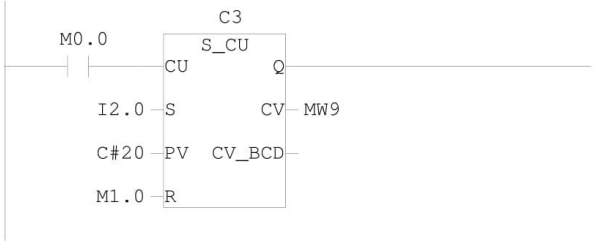
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC3

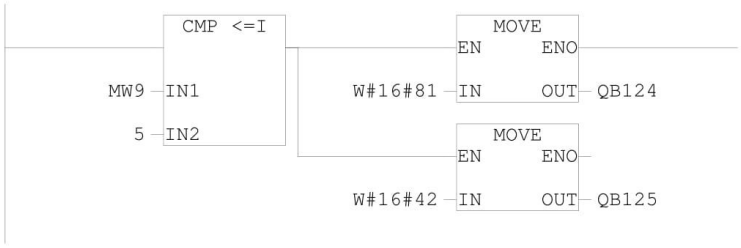
Network: 1

Network: 2

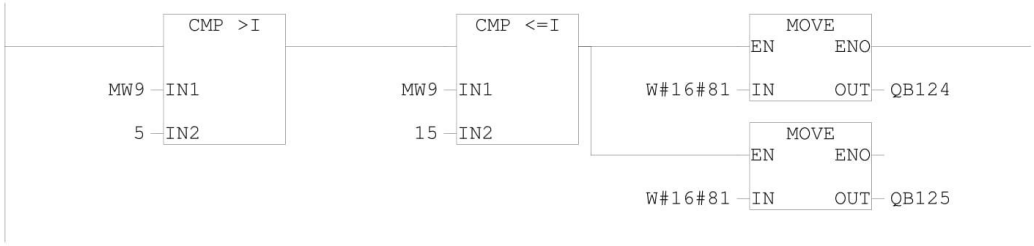
Network: 3



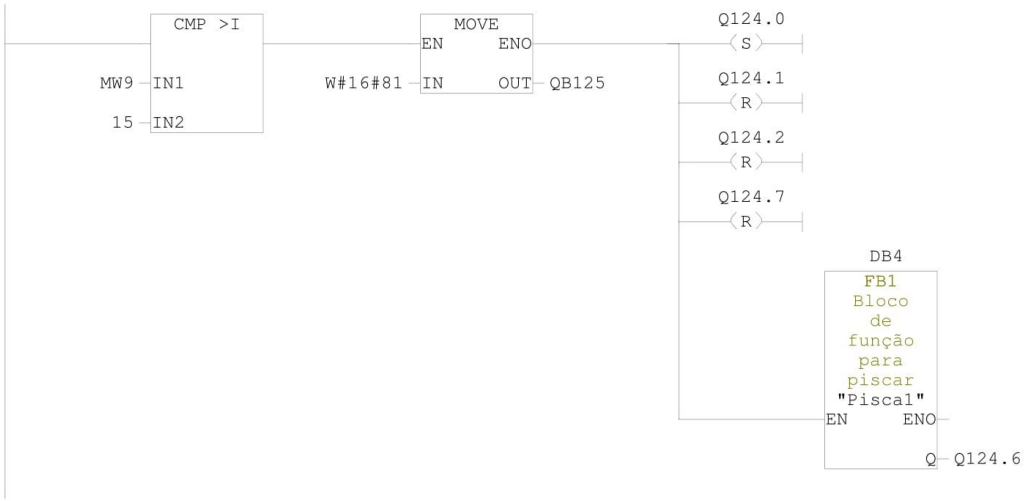
Network: 4



Network: 5



Network: 6



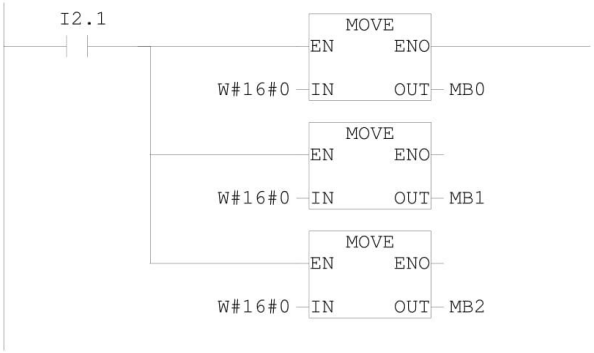
FC4 - <offline>

"Reset" Reset de todas as memórias
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/27/2016 11:00:06 PM
Interface: 06/27/2016 10:58:49 PM
Lengths (block/logic/data): 00154 00056 00002

Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC4

Network: 1



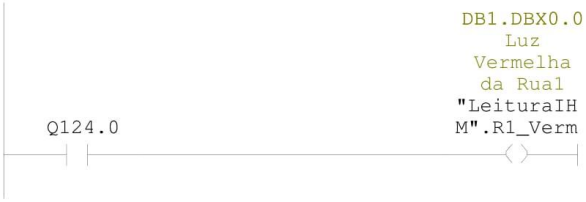
FC10 - <offline>

"Espel" Espelhamento das entradas e saídas
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/28/2016 10:18:16 AM
Interface: 06/27/2016 09:50:53 AM
Lengths (block/logic/data): 00400 00250 00000

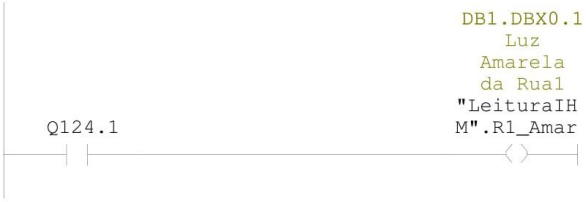
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC10

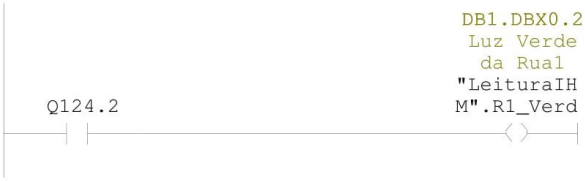
Network: 1



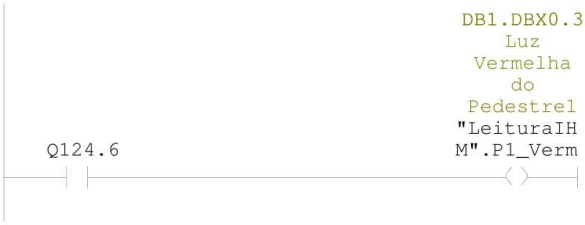
Network: 2 Luz Amarela da Rua1



Network: 3 Luz Verde da Rua1



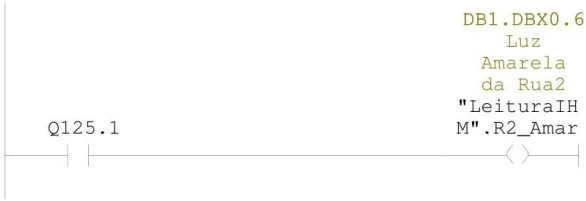
Network: 4 Luz Vermelha do Pedestre1



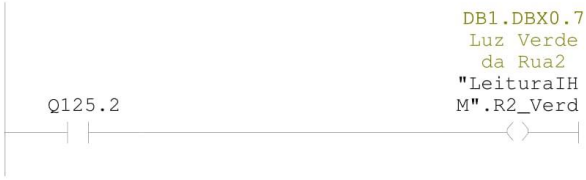
Network: 5 Luz Verde do Pedestre1

Network: 6 Luz Vermelha da Rua2

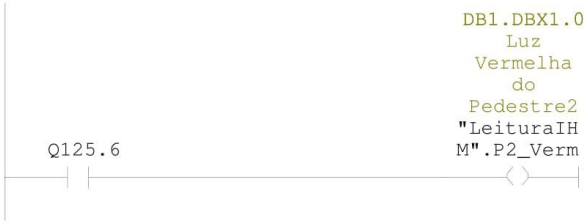
Network: 7 Luz Amarela da Rua2



Network: 8 Luz Verde da Rua2

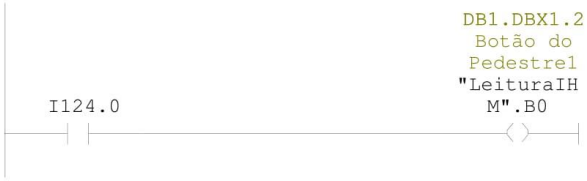


Network: 9 Luz Vermelha do Pedestre2



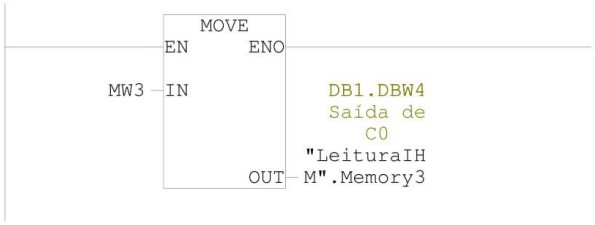
Network: 10 Luz Verde do Pedestre2

Network: 11 Botão do Pedestre1

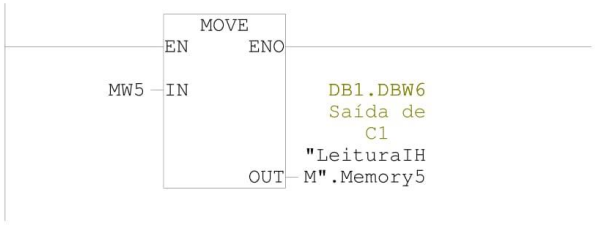


Network: 12 Botão do Pedestre2

Network: 13

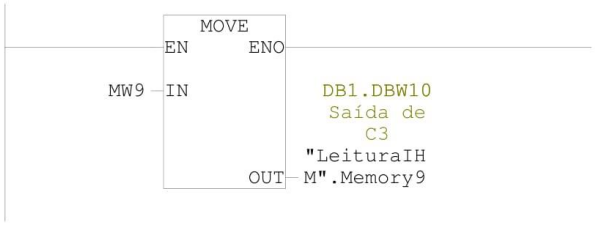


Network: 14



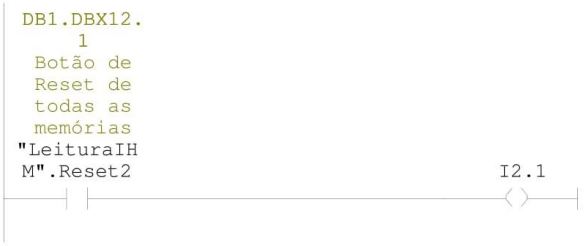
Network: 15

Network: 16



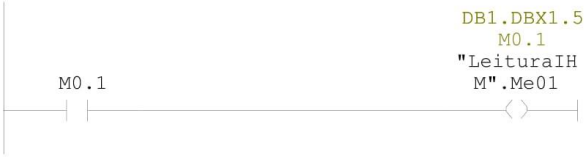
Network: 17 Botão de Reset de todas as memórias

Network: 18 Botão de Reset de todas as memórias



Network: 19 M0.0

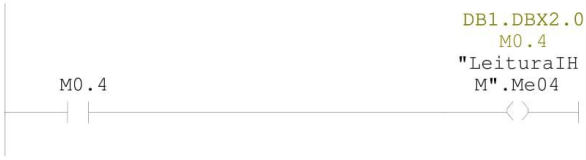
Network: 20 M0.1



Network: 21 M0.2

Network: 22 M0.3

Network: 23 Luz Amarela da Rua1

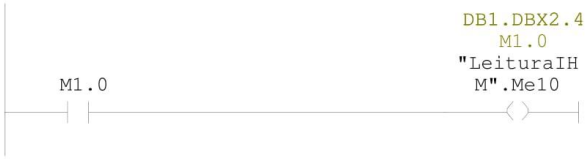


Network: 24 Luz Verde da Rua1

Network: 25 Luz Vermelha do Pedestre1

Network: 26 Luz Verde do Pedestre1

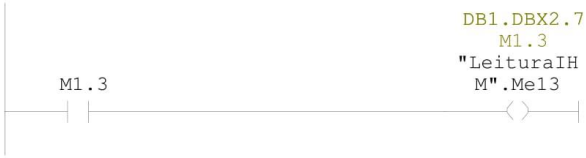
Network: 27 Luz Vermelha da Rua2



Network: 28 Luz Amarela da Rua2

Network: 29 Luz Verde da Rua2

Network: 30 Luz Vermelha do Pedestre2



DB1 - <offline> - Declaration view

"LeituraIHM" Database para leitura da IHM
Global data block DB 1
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/28/2016 12:10:59 AM
Interface: 06/27/2016 11:53:38 PM
Lengths (block/logic/data): 00162 00014 00000

Block: DB1

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	R1_Verm	BOOL	FALSE	Luz Vermelha da Rua1
+0.1	R1_Amar	BOOL	FALSE	Luz Amarela da Rua1
+0.2	R1_Verd	BOOL	FALSE	Luz Verde da Rua1
+0.3	P1_Verm	BOOL	FALSE	Luz Vermelha do Pedestre1
+0.4	P1_Verd	BOOL	FALSE	Luz Verde do Pedestre1
+0.5	R2_Verm	BOOL	FALSE	Luz Vermelha da Rua2
+0.6	R2_Amar	BOOL	FALSE	Luz Amarela da Rua2
+0.7	R2_Verd	BOOL	FALSE	Luz Verde da Rua2
+1.0	P2_Verm	BOOL	FALSE	Luz Vermelha do Pedestre2
+1.1	P2_Verd	BOOL	FALSE	Luz Verde do Pedestre2
+1.2	B0	BOOL	FALSE	Botão do Pedestre1
+1.3	B1	BOOL	FALSE	Botão do Pedestre2
+1.4	Me00	BOOL	FALSE	M0.0
+1.5	Me01	BOOL	FALSE	M0.1
+1.6	Me02	BOOL	FALSE	M0.2
+1.7	Me03	BOOL	FALSE	M0.3
+2.0	Me04	BOOL	FALSE	M0.4
+2.1	Me05	BOOL	FALSE	M0.5
+2.2	Me06	BOOL	FALSE	M0.6
+2.3	Me07	BOOL	FALSE	M0.7
+2.4	Me10	BOOL	FALSE	M1.0
+2.5	Me11	BOOL	FALSE	M1.1
+2.6	Me12	BOOL	FALSE	M1.2
+2.7	Me13	BOOL	FALSE	M1.3
+4.0	Memory3	WORD	W#16#0	Saída de C0
+6.0	Memory5	WORD	W#16#0	Saída de C1
+8.0	Memory7	WORD	W#16#0	Saída de C2
+10.0	Memory9	WORD	W#16#0	Saída de C3
+12.0	Reset1	BOOL	FALSE	Botão de Reset de todos os contadores
+12.1	Reset2	BOOL	FALSE	Botão de Reset de todas as memórias
=14.0		END_STRUCT		

FB1 - <offline>

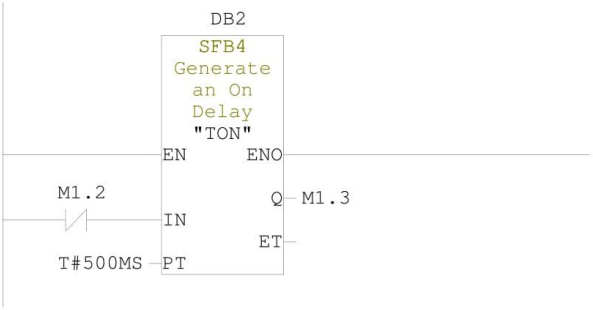
"Pisca1" Bloco de função para piscar

Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/27/2016 09:09:50 PM
Interface: 06/27/2016 09:02:54 PM
Lengths (block/logic/data): 00242 00144 00006

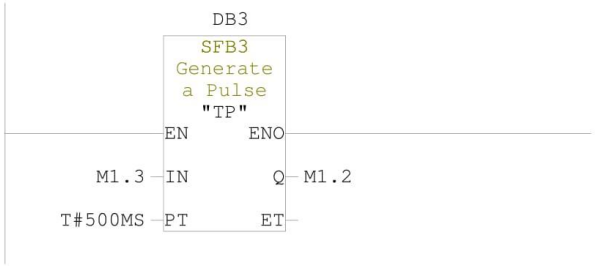
Name	Data Type	Address	Initial Value	Comment
IN		0.0		
OUT		0.0		
Q	Bool	0.0	FALSE	
IN_OUT		0.0		
STAT		0.0		
TEMP		0.0		

Block: FB1

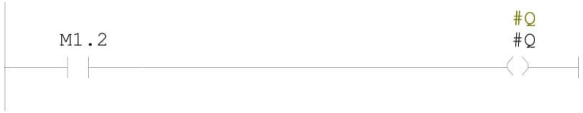
Network: 1



Network: 2



Network: 3



APENDICE B – PROGRAMA DO STEP7: SEMÁFORO SÃO JOSÉ DOS PINHAIS

OB1 - <offline>

""

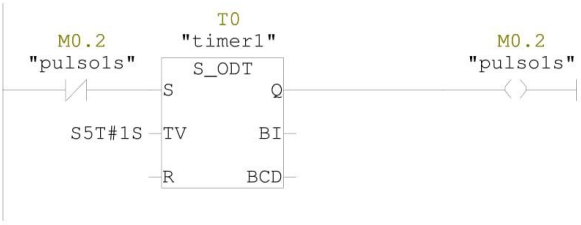
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/22/2016 10:12:47 AM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00156 00034 00020

Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

Block: OB1 "Main Program Sweep (Cycle)"

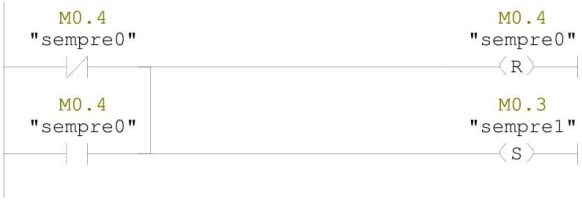
Network: 1

Pulso de 1 segundo.



Network: 2

Memórias "true" e "false".



Network: 3

Chamando a função do Semáforo.

FC1 - <offline>

""

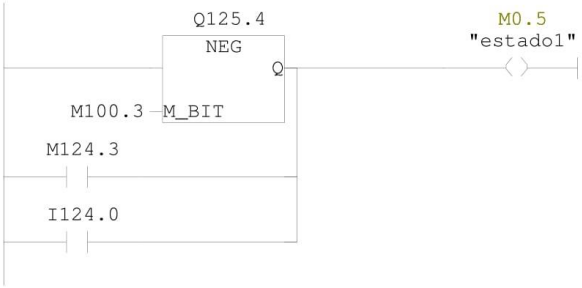
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 06/22/2016 10:13:11 AM
Interface: 06/21/2016 10:49:12 AM
Lengths (block/logic/data): 00318 00200 00002

Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC1

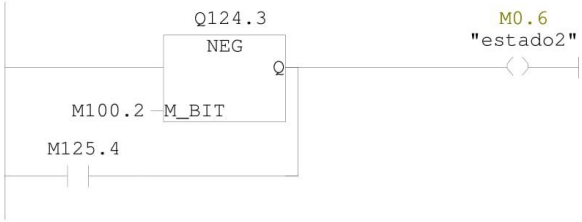
Network: 1

Entra no "estado1" (sinais vermelhos), quando ocorre uma borda de descida do sinal amarelo, e permanece nesse estado enquanto o último sinal vermelho estiver aceso.



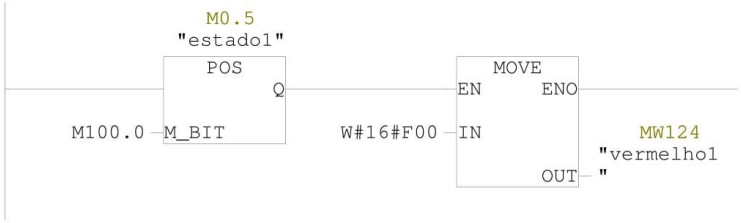
Network: 2

Entra no "estado2" (sinais verdes e amarelo) quando ocorrer uma borda de descida do último sinal vermelho, e permanece nesse estado enquanto a memória referente ao sinal amarelo estiver em 1.



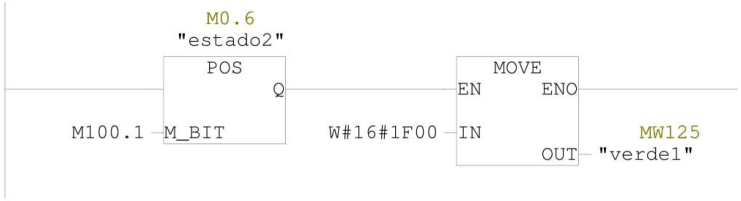
Network: 3

Nas bordas de subida do "estado1" (sinais vermelhos), move o valor "0000.1111.0000.0000" para a memória referente aos sinais vermelhos.

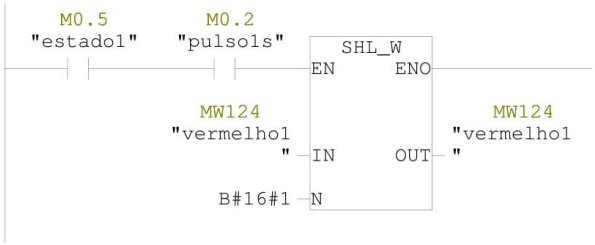


Network: 4

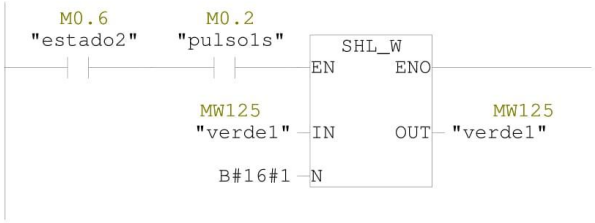
Nas bordas de subida do "estado2" (sinais verdes e amarelo), move o valor "0001.1111.0000.0000" para a memória referente aos sinais verdes e amarelo.



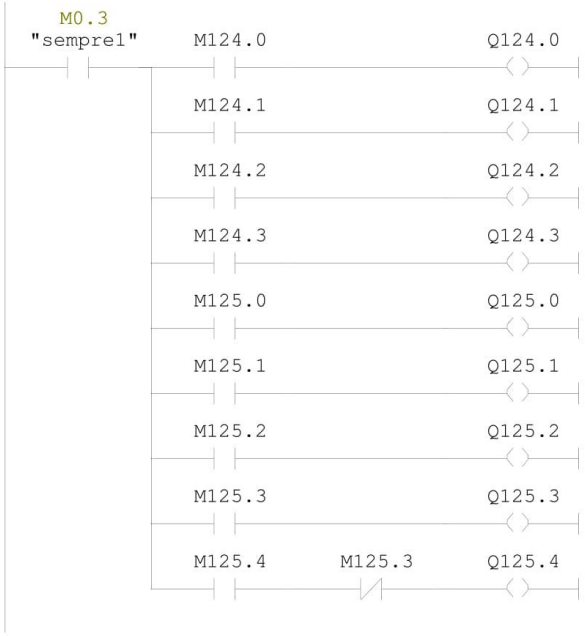
Network: 5
Estando no "estado1", a cada pulso de segundo, movimenta para à esquerda em 1 unidade todos os bits da memória referente aos sinais vermelhos.



Network: 6
Estando no "estado2", a cada pulso de segundo, movimenta para a esquerda em 1 unidade todos os bits da memória referente aos sinais verdes e amarelo.



Network: 7
Movimenta os bits da memória dos sinais vermelhos e verde para as saídas digitais na bancada. A última linha movimenta o sinal de amarelo, mas apenas quando os sinais verdes se apagaram.



APENDICE C – MANUAL

Disponivel em <https://www.dropbox.com/s/2s01na3qnj7t6ey/Manual.pdf?dl=0>.