UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTOS ACADÊMICOS DE ELETRÔNICA E MECÂNICA
CURSO SUPERIOR DE TECNOLOGIA EM MECATRÔNICA INDUSTRIAL

LUIS FERNANDO DA SILVA ALVES
THOMAS PHILIP DÜCK

SISTEMA WEB DE SUPERVISÃO E CONTROLE

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2013

LUIS FERNANDO DA SILVA ALVES
THOMAS PHILIP DÜCK

SISTEMA WEB DE SUPERVISÃO E CONTROLE

Trabalho de Conclusão de Curso de Graduação, apresentado ao Curso Superior de Tecnologia em Mecatrônica Industrial, dos Departamentos Acadêmicos de Eletrônica e Mecânica, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Marcelo Victor Wüst Zibetti

CURITIBA
2013

TERMO DE APROVAÇÃO

LUIS FERNANDO DA SILVA ALVES
THOMAS PHILIP DÜCK

SISTEMA WEB DE SUPERVISÃO E CONTROLE

Este trabalho de conclusão de curso foi apresentado no dia 7 de outubro de 2013, como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial, outorgado pela Universidade Tecnológica Federal do Paraná. Os alunos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

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

_____
Prof. Esp. Sérgio Moribe
Responsável pela Atividade de Trabalho de Conclusão de Curso
Departamento Acadêmico de Eletrônica

BANCA EXAMINADORA

_____
Prof. M.Sc. Guilherme Alceu Schneider
UTFPR

_____
Prof. Dr. Sérgio Leandro Stebel
UTFPR

_____
Prof. Dr. Marcelo Victor Wüst Zibetti
Orientador - UTFPR

“A Folha de Aprovação assinada encontra-se na Coordenação do Curso”

RESUMO

ALVES, Luis Fernando da Silva; DÜCK, Thomas Philip. **Sistema WEB de Supervisão e Controle**. 2013. 83 f. Trabalho de Conclusão de Curso (Curso de Tecnologia em Mecatrônica Industrial), Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

O presente trabalho tem por objetivo elaborar um sistema *web* de supervisão e controle de processos. A interface de supervisão é desenvolvida em linguagem PHP e possibilita o controle do sistema utilizando navegadores de internet ou dispositivos móveis. A comunicação será realizada de maneira indireta entre o sistema de supervisão e o controlador do processo utilizando o mapeamento da base de dados OPC no Elipse E3, responsável pela conversão do protocolo OPC em *Modbus* TCP. A página *web* utiliza o protocolo *Modbus* TCP para se comunicar com o Elipse E3.

Palavras-chave: Sistema *Web*. Sistema de Supervisão. *Modbus*.

ABSTRACT

ALVES, Luis Fernando da Silva; DÜCK, Thomas Philip. **WEB-based Supervision and Control System**. 2013. 83 f. Trabalho de Conclusão de Curso (Curso de Tecnologia em Mecatrônica Industrial), Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

The purpose of the current work is to elaborate a web-based supervisory and control system. The supervisory interface is developed in PHP, allowing it to be viewed and controlled through web browsers and mobile devices. The communication between the supervisory system and the process controller will be done indirectly by mapping the OPC database on the Elipse E3 application, which will be responsible for the conversion between the OPC and Modbus TCP protocols. The web page uses the Modbus TCP protocol to communicate with Elipse E3.

Keywords: Web system. Supervisory system. Modbus.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ACRÔNIMOS

ADU	Application Data Unit
AJAX	Asynchronous JavaScript and XML
CLP	Controlador Lógico Programável
CSS	Cascading Style Sheets
FMS	Sistema Flexível de Manufatura
DOM	Document Object Model
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IIS	Internet Information Services
IP	Internet Protocol
ISO	International Organization for Standardization
ITS	Institute for Telecommunication Sciences
MAC	Media Access Control
MBAP	Modbus Application Protocol Header
OLE	Object Linking and Embedding
OPC	Open Platform Communication
OS	Operational System
OSI	Open System Interconnection
PDU	Protocol Data Unit
PHP	Hypertext Preprocessor
POP3	Post Office Protocol
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
TCP	Transmission Control Protocol
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Segundo Groover (2008), sistema supervisório é um sistema de monitoramento, operação e controle que gerencia as atividades de um número de unidades operacionais para atingir determinados objetivos econômicos dentro do processo. Busca aperfeiçoar alguns objetivos definidos, normalmente baseados em critérios como a produtividade, o custo, a qualidade ou outros aspectos relacionados ao desempenho do processo.

Os primeiros sistemas supervisórios surgiram na década de 1960 e eram basicamente telemétricos, informando o estado corrente de determinado processo industrial através de painéis de lâmpadas e indicadores. Atualmente, a maior parte dos sistemas supervisórios utiliza computadores que executam programas específicos para supervisão e controle.

No modelo proposto por Groover (2001), o sistema supervisório atua diretamente sobre o controlador do processo. O bloco Supervisão do Controle na figura 1 normalmente está conectado ao controlador do processo utilizando uma rede interna e sem possibilidade de acesso remoto. Com o advento da era da informação, torna-se necessário o acesso remoto e dinâmico de informações sobre determinados processos. Para isso, pode-se utilizar uma página *web* para visualizar e controlar as variáveis do processo.

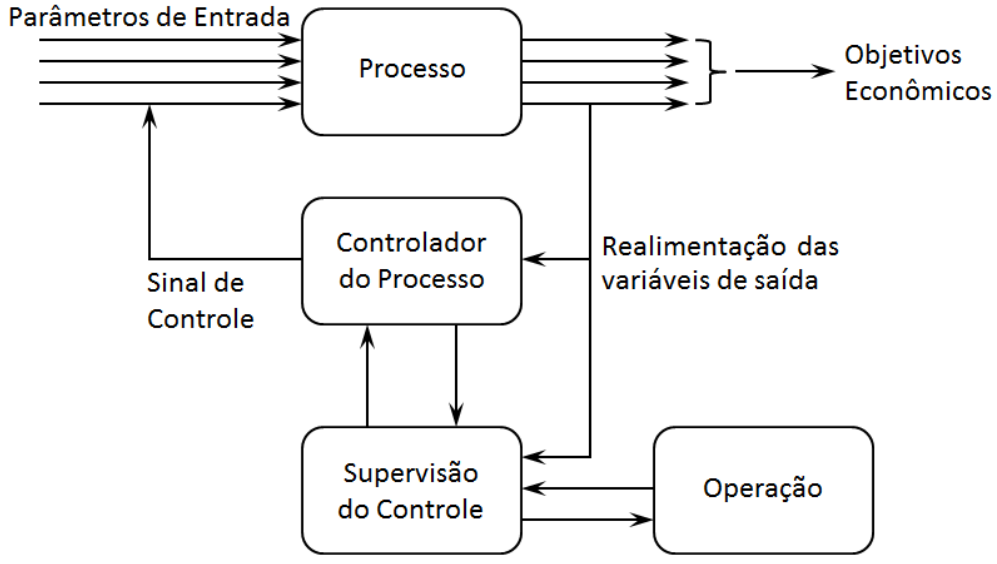


Figura 1 - Sistema supervisório
Fonte: Adaptado de Groover, 2001.

1.1 DELIMITAÇÃO DO TEMA

O escopo do presente trabalho é o desenvolvimento de um sistema que permita visualizar e controlar variáveis de um controlador em um navegador de internet.

A utilização de sistemas de supervisão baseados em acesso remoto através da rede mundial de computadores será utilizada em processos que não são críticos, pois a disponibilidade do sistema estará limitada à confiabilidade e estabilidade do acesso à internet.

Neste projeto o protocolo de comunicação abordado será o *Modbus* TCP, que será utilizado para que a página da *web* possa visualizar e controlar as variáveis do processo no controlador.

A comunicação entre a página da *web* e o controlador será feita por intermédio do Elipse E3, que tem mapeada a base de dados *Modbus* TCP. Essa comunicação torna-se necessária pois o controlador utilizado não possui suporte nativo ao protocolo *Modbus*. A comunicação entre o Elipse E3 e o controlador será feita através do protocolo aberto OPC (da sigla em inglês para *OLE for Process Control*).

Em teoria, o presente trabalho permite a comunicação direta com qualquer controlador compatível com *Modbus* TCP ou a comunicação indireta utilizando um *gateway* com controladores com *drivers* compatíveis.

1.2 PROBLEMA

Os sistemas de supervisão tradicionais possibilitam a aquisição e o tratamento de dados localmente, ou seja, as informações da planta são coletadas, tratadas e visualizadas dentro do próprio ambiente industrial. Nesses casos, para que as informações sejam disponibilizadas na rede mundial de computadores, é necessário que sejam instalados diversos aplicativos no computador do cliente e até mesmo um alto investimento para implementar esta solução.

A implantação de um *driver* de comunicação diretamente em *Hypertext Preprocessor* (PHP) é um diferencial em relação aos sistemas de supervisão tradicionais uma vez que desta forma a aplicação será desenvolvida com foco exclusivo para acesso remoto.

A comunicação entre o sistema supervisório baseado em internet, será realizada através do protocolo de comunicação *Modbus* TCP. Com base nos estudos da especificação do protocolo e ferramentas disponíveis na linguagem PHP, será possível desenvolver um sistema genérico para quaisquer aplicações comerciais que disponham de equipamentos compatíveis com o protocolo de comunicação que será a base do sistema de supervisão.

1.3 JUSTIFICATIVA

Usabilidade, mobilidade, acessibilidade e baixo custo são características que definem o grande potencial de crescimento de aplicações móveis baseadas em ambientes desenvolvidos para *web*. A venda de *tablets* nos EUA será maior que a de *notebooks* já a partir do ano de 2013, fato que evidencia a importância de explorar esse nicho de mercado (Forrester Research, 2012).

O presente projeto possibilita que ambos os campos sejam integrados através da utilização de conhecimentos em protocolos de comunicação, redes e controladores programáveis para implementar uma ferramenta capaz de integrar o ambiente industrial à rede mundial de computadores. Grandes empresas como *Danfoss Drives* – consolidada fabricante de inversores de frequência e *soft starters* – bem como a pioneira *Siemens AG*, são membros integrantes da *Modbus Organization*.

A utilização de um protocolo amplamente difundido e utilizado no ambiente industrial torna a gama de possibilidades de uso do presente projeto muito ampla. A leitura de dados direta de equipamentos como inversores de frequência ou *soft starters Danfoss*, multimedidores e controladores *Siemens*, são algumas das possibilidades a serem exploradas.

A implantação de um sistema capaz de solicitar dados *online* através de um *gateway*, de um controlador programável ou até mesmo multimedidores de grandezas viabiliza e reduz os custos de um desenvolvimento tradicional, onde é necessária a configuração de um *driver* e desenvolvimento de uma aplicação para *desktop* para que depois seja gerada uma versão *web*. O desenvolvimento trata de uma concepção pouco difundida, além de possuir aplicação direta no meio industrial e despertar interesse comercial.

1.4 OBJETIVOS

Nesta seção são apresentados os objetivos geral e específicos do trabalho, relativos ao problema apresentado anteriormente.

1.4.1 Objetivo Geral

Implementar um sistema de supervisão para internet, cuja função é monitorar e controlar as variáveis discretas do processos do FMS da UTFPR através de um navegador, independente do sistema operacional.

1.4.2 Objetivos Específicos

- Interpretar a especificação do protocolo *Modbus* TCP na camada de aplicação do modelo OSI.
- Implementar o protocolo *Modbus* TCP em linguagem PHP.
- Desenvolver um sinótico do processo do FMS e interfaces de comando para navegador de internet.
- Desenvolver o mapeamento da base de dados do CLP no protocolo Modbus TCP, através do Elipse E3.

- Desenvolver um programa de CLP para realizar o controle das variáveis discretas do processo do FMS.
- Elaborar a documentação referente ao projeto para possibilitar aprimoramentos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para implementar o sistema web de supervisão e controle foi necessário compreender o funcionamento da arquitetura cliente servidor, o modelo de referência OSI, o protocolo Modbus TCP, a linguagem de programação PHP, a implementação de um servidor de internet, a comunicação através de *sockets*, a metodologia Ajax, a especificação do protocolo OPC e o conceito de *gateway*. Estes itens serão abordados detalhadamente no decorrer deste capítulo.

2.1 ARQUITETURA CLIENTE SERVIDOR

Segundo a Microsoft (2013), os sistemas com arquitetura cliente-servidor são projetados de maneira que os serviços e dados fiquem concentrados num computador central, nomeado servidor e possam ser acessados por vários usuários, conforme figura 2. Os acessos podem ser realizados através de clientes remotos ou a partir de uma aplicação no servidor.

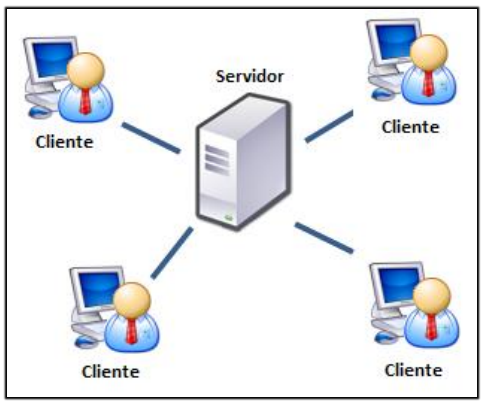


Figura 2 - Arquitetura Cliente-Servidor
Fonte: Autoria própria

O usuário roda a aplicação em seu computador que é o cliente da arquitetura. O cliente conecta-se ao servidor e solicita os dados, que são respondidos na sequência pelo servidor. A adoção da arquitetura tem como

benefícios a redução em custo de hardware do cliente, uma vez que todas as solicitações são processadas pelo servidor, bem como facilita as regras de segurança dos dados, pois os mesmos ficam concentrados em um único servidor (Microsoft, 2013).

Dentro do escopo do trabalho, a concepção de cliente-servidor é a arquitetura base do protocolo Modbus TCP e define o modo de acesso da página web ao servidor Modbus TCP. O servidor adquire os dados do controlador programável presente no FMS e distribui os dados para a web. Entretanto, os dados são enviados apenas quando há uma solicitação de leitura pelo navegador de internet.

2.2 MODELO DE REFERÊNCIA OSI

De acordo com a ITU - International Telecommunication Union (1994), o modelo de referência de sistemas de interconexão abertos (OSI - *Open Systems Interconnection*) tem como objetivo definir padrões que permitam a interconexão e a cooperação entre sistemas abertos.

Segundo a Acromag Incorporated (2005), a arquitetura do modelo OSI divide os sistemas de comunicação em sete camadas de abstração, conforme pode ser visualizado no quadro 1: Física, Enlace, Rede, Transporte, Sessão, Apresentação e Aplicação.

Grupo	Camada	Nome
Aplicação	7	Aplicação
	6	Apresentação
	5	Sessão
Transporte	4	Transporte
Rede	3	Rede
	2	Enlace
	1	Física

Quadro 1 - Camadas do modelo OSI
Fonte: Adaptado de Acromag Incorporated (2005).

Segundo a ITU (1994) as camadas do modelo OSI podem ser definidas da seguinte maneira:

- Camada física: refere-se aos meios de conexão mecânicos ou elétricos através dos quais ocorre a transferência de dados.
- Camada de enlace de dados: fornece os meios para transferir dados entre entidades de rede e detectar e opcionalmente corrigir erros que podem ocorrer na camada física.
- Camada de rede: é responsável pelo endereçamento dos pacotes de rede. Ela associa endereços lógicos (IP) a endereços físicos (MAC) fazendo com que os pacotes de rede consigam chegar corretamente ao destino.
- Camada de transporte: fornece a transferência transparente de dados entre os usuários finais, fornecendo serviços confiáveis de transferência de dados para as camadas superiores.
- Camada de sessão: controla as conexões entre computadores, permitindo que seja estabelecida uma sessão de comunicação.
- Camada de apresentação: converte o formato dos dados recebidos da camada de aplicação em um formato a ser utilizado na transmissão desses dados.
- Camada de aplicação: corresponde aos aplicativos que serão utilizados para promover uma interação entre a máquina e o usuário final da aplicação.

2.2.1 Rede Ethernet

Segundo Davis et al. (2004), a rede ethernet foi desenvolvida a 30 anos pela Xerox PARC e foi originalmente usada com apenas um cabo único conectando os nós em uma rede. Com a explosão da *Internet,* a arquitetura cliente servidor se tornou a referência e cada vez mais computadores se conectavam através da rede através de cabos de par trançado. O modelo de ligação por cabos de par trançado foram concebidos, pois permitem que nós de rede sejam ligados ou desligados, sem causar problemas de transmissão de dados.

A rede ethernet tem a capacidade de prover um canal de alta de velocidade para troca de dados entre computadores e outros dispositivos localizados em uma rede, (Intel, 1982). Segundo Davis et al. (2004), na rede ethernet, cada computador tem seu endereço. O endereço é único para evitar conflito entre os nós. Como os recursos da rede ethernet são compartilhados, cada nó recebe todos os dados de uma determinada rede, sendo o computador responsável por determinar se a mensagem deve ser descartada ou respondida.

Segundo Davis et al. (2004), o endereço da rede ethernet não é o mesmo do protocolo TCP/IP, localizado na camada de transporte do modelo OSI. O endereço de um nó da rede ethernet é o endereço físico atrelado à interface de hardware conectado ao cabo de rede.

No presente projeto, a rede ethernet está presente no sistema flexível de manufatura e é utilizada como interface física para comunicação entre o controlador lógico programável e os computadores presentes no laboratório.

2.3 PROTOCOLO MODBUS TCP

Segundo a *Acromag* (2005), o *Modbus* foi desenvolvido em 1979 pela *Modicon*, inicialmente incorporado para sistemas de automação industrial e para os controladores programáveis *Modicon*.

Desde seu nascimento, o *Modbus* se tornou um protocolo industrial padrão para a transferência de dados discretos, analógicos e de registros de dados entre os sistemas de supervisão e controle. O *modbus* sendo um protocolo aberto, de domínio público, é atualmente totalmente difundido e utilizado no ambiente industrial.

Dispositivos utilizados no *modbus* se comunicam utilizando a arquitetura cliente-servidor, onde apenas o cliente pode iniciar a comunicação. Os outros dispositivos respondem os dados solicitados pelo cliente ou executam as ações solicitadas na requisição. Um servidor é qualquer dispositivo periférico – controlador, transdutor ou multimedidor - que processa

a informação e envia a resposta para o cliente, através do *Modbus* (Acromag, 2005).

Segundo a *Modbus Organization* (2006), o *Modbus* é um protocolo de comunicação que está presente na camada de aplicação, posicionado no nível 7 do modelo de referência OSI e é utilizado na arquitetura cliente / servidor entre os dispositivos conectados em rede. O *Modbus* TCP é simplesmente o *Modbus* RTU (ou apenas *Modbus*), com uma interface TCP utilizando o meio físico *ethernet*.

A estrutura das mensagens do *Modbus* está presente no nível 7 do modelo OSI e definem as regras para organizar e interpretar os dados independente do meio de transmissão.

Um cabeçalho dedicado é utilizado no TCP/IP para identificar o *Modbus Application Data Unit*. O cabeçalho recebe o nome de MBAP (*Modbus Application Protocol Header*), conforme pode ser visualizado na figura 3. Esse cabeçalho possui diversas diferenças em relação ao *Modbus* RTU.

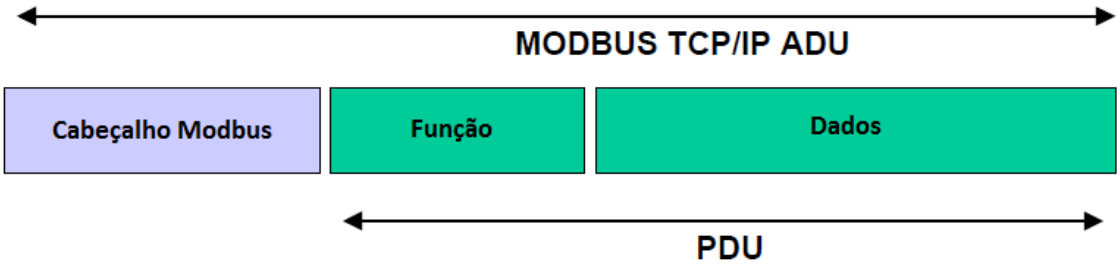


Figura 3 - *Modbus* TCP ADU / PDU
Fonte: *Modbus Organization*, 2006.

O endereço do servidor, utilizado nas comunicações seriais *Modbus*, é substituído por um byte nomeado '*Unit Identifier*', constituindo o MBAP. O '*Unit Identifier*' é utilizado para comunicações de dispositivos que funcionam como *gateways* que possuem apenas um endereço IP, mas suportam diversos endereços *modbus* (Modbus, 2006).

No *Modbus* TCP, todas as requisições e respostas foram projetadas de maneira que o servidor ou cliente possa verificar que a mensagem foi finalizada. Para as funções onde há leitura ou escrita de apenas uma variável, o tamanho do PDU é fixo. Para as funções onde a resposta tem um tamanho variável existe um *byte* para indicar o tamanho do PDU (Modbus, 2006).

Segundo a *Modicon* (1996), a comunicação via protocolo *Modbus* TCP obedece a um *frame* constituído basicamente de um cabeçalho inicial, tamanho

da mensagem a ser enviada, endereço do servidor, comando (leitura / escrita), registro inicial a ser lido e o número de posições a serem lidas a partir de determinado registro. O *frame* de comunicação do *Modbus* TCP pode ser visualizado no quadro 2.

Cabeçalho MBAP							Função	Dados			
Trans. Identif.		Protocol Ident.		Length		Unit Ident.	Comando	Registro		N° de Registros	
0	0	0	0	0	6	5	3	0	1	0	120

Quadro 2 - *Frame* de comunicação *Modbus* TCP
Fonte: Adaptado de *Modbus Organization*, 2006.

Segundo a *Modbus Organization* (2006), o cabeçalho MBAP é constituído por 7 bytes:

- *Transaction Identifier* é utilizado para sincronização entre as mensagens do servidor / cliente.
- *Protocol Identifier* é utilizado para multiplexação intra-sistemas. O protocolo *modbus* é identificado quando os dois bytes são 0.
- *Length* é responsável por identificar a quantidade de bytes em sequência no frame de comunicação *modbus*. O tamanho da mensagem inclui o *Unit Identifier*, função e pacote de dados da mensagem.
- *Unit identifier* indica qual o endereço do servidor a se conectar. O *Unit Identifier* é mais utilizado quando o servidor é um *gateway* de vários escravos *modbus*, dessa forma, cada um dos escravos possui um endereço definido no servidor.

O código da função *modbus* indica ao servidor qual ação deverá ser executada e o valor do código da função varia de 1 a 255, sendo os registros 128 a 255 reservados (*Modbus Organization*, 2006).

No quadro 3 estão as principais funções do tipo *holding register* implementadas no presente projeto.

Comando	Função
3	Leitura de registros retentivos
6	Escrita em um único registro
16	Escrita em uma quantidade variável de registros

Quadro 3 - Funções *Holding Register* (Implementadas)
Fonte: Adaptado de *Modbus Organization*, 2006

As funções de leitura e escrita de registros são utilizadas para leitura de variáveis tipo *word*, compostas por 2 *bytes*. As funções de leitura e escrita de *coils* apresentadas no quadro 4 e não implementadas no presente projeto, são utilizadas na leitura ou escrita direta de um *bit*. Segundo a *Modbus Organization* (2006), essas funções atuam diretamente na leitura das entradas digitais e comandos nas saídas digitais dos controladores.

Comando	Função
1	Leitura de Multiple Coils (Saída Digitais)
2	Leitura de Entradas Digitais
5	Escrita em Single Coil (Atua Saída Digital)
15	Escrita em Multiple Coil (Atua Saída(s) Digital(is))

Quadro 4 - Funções Leitura/Escrita *Coil*
Fonte: Adaptado de *Modbus Organization*, 2006

A parte de dados a serem enviados ou recebidos da mensagem é composta basicamente pelo valor dos registros solicitados, ordenados sequencialmente a partir do primeiro registro solicitado até o último da requisição (definido pelo número de bytes solicitados pelo cliente). O valor do registro é composto por 2 *bytes*, sendo seu valor formado de acordo com a equação abaixo:

$$ValorRegistro = MsB * 256 + LsB$$

Lei de formação do número de registros, limitado a 120 conforme especificação do protocolo, de acordo com a equação abaixo:

$$NRegistro = MsB * 256 + LsB$$

2.4 LINGUAGEM DE PROGRAMAÇÃO PHP

PHP, um acrônimo de *Hypertext Preprocessor*, é uma linguagem de programação para internet, de código aberto, desenvolvida por e para programadores de *web*. PHP é uma linguagem de *script* do tipo *server-side* que

pode ser embutida dentro do HTML, do inglês *HyperText Markup Language*, linguagem base para desenvolvimento de páginas de internet (Morgan, 2004).

O servidor é responsável por interpretar o código juntamente com o interpretador do PHP e gerar um código HTML que é interpretado pelo navegador e é finalmente exibido ao usuário (PHP.NET, 2012).

Segundo o grupo de fundadores do PHP, a linguagem pode ser utilizada na maioria dos sistemas operacionais, incluindo Linux, variantes do Unix, Microsoft Windows, Mac OS e RISC OS. O *Hypertext Preprocessor* é também suportado pela maioria dos servidores existentes, dentre os principais o Apache, da *Apache Software Foundation*, utilizado em larga escala em servidores Linux e o IIS, da Microsoft.

Uma das mais fortes e significativas características do PHP é a variedade de banco de dados que a linguagem suporta. Além dessa característica, a linguagem tem suporte para métodos de comunicação utilizando vários protocolos como o POP3 e HTTP. Com essas características, é possível abrir *sockets* e interagir diretamente com qualquer protocolo (PHP.NET, 2012).

Utilizando as principais características e marcos da linguagem, o presente projeto pretende utilizar métodos de comunicação do tipo *socket* para estabelecer comunicação com servidores remotos, neste caso em particular um *gateway* ou mesmo um controlador.

2.5 APACHE

O Servidor HTTP Apache, ou simplesmente Apache, é um projeto colaborativo da *Apache Software Foundation* cujos esforços visam desenvolver e manter um servidor HTTP robusto, comercialmente viável, gratuito e de código aberto para sistemas operacionais modernos, como UNIX e Windows.

Segundo a *Apache Software Foundation*, o desenvolvimento do servidor Apache foi iniciado em 1995 por um grupo de programadores a partir do servidor HTTP Daemon e desde 1996 ele é o servidor HTTP mais utilizado no mundo.

Dentro do escopo do presente trabalho, o servidor Apache será utilizado para hospedar a página e disponibilizar os dados interpretados pelo *Hypertext Preprocessor.*

2.6 SOCKET

Segundo Mitchell et al. (2001), um *socket* é um terminal cuja função é estabelecer uma comunicação bidirecional entre um servidor e um ou mais clientes. O *socket* associa ao servidor uma porta especifica na máquina onde ele está rodando para que qualquer cliente na rede que tenha um *socket* associado com a mesma porta possa se comunicar com o servidor.

Normalmente, o servidor fornece recursos a diversos clientes na rede. Os clientes enviam requisições para o servidor, que responde a elas.

Uma maneira de fazer com que o servidor possa tratar requisições de mais de um cliente é fazer com que ele seja *multi-threaded*. Um *thread* é uma sequencia de instruções ou lista de execução que roda independentemente do programa e de qualquer outra *thread*. Um servidor *multi-thread* cria uma linha de execução para cada comunicação aceita do cliente. Dessa forma, o programa servidor pode aceitar a comunicação com um cliente, iniciar uma lista de execução para aquela comunicação e continuar monitorando requisições de outros clientes.

2.7 ASYNCHRONOUS JAVASCRIPT AND XML (AJAX)

Ajax, do acrônimo inglês para *Aynchronous JavaScript and XML,* é o uso metodológico de técnicas de desenvolvimento *web*, como JavaScript, XML, CSS e DOM, para tornar páginas da *web* mais dinâmicas e interativas. Para isso, utiliza-se de solicitações assíncronas de envio e recebimento de dados do servidor sem interferir na exibição e no comportamento da página em si. O conceito da metodologia Ajax foi inicialmente desenvolvido por Jessé James

Garret, porém as ferramentas e técnicas de desenvolvimento utilizadas, como o JavaScript, já existiam anteriormente.

A premissa básica da metodologia Ajax é o uso do objeto JavaScript `XMLHttpRequest` para obter informações de um servidor *web* de modo dinâmico e assíncrono (BABIN, 2007). As figuras 4 e 5 demonstram a diferença no modelo de solicitação/resposta tradicional e utilizando Ajax.

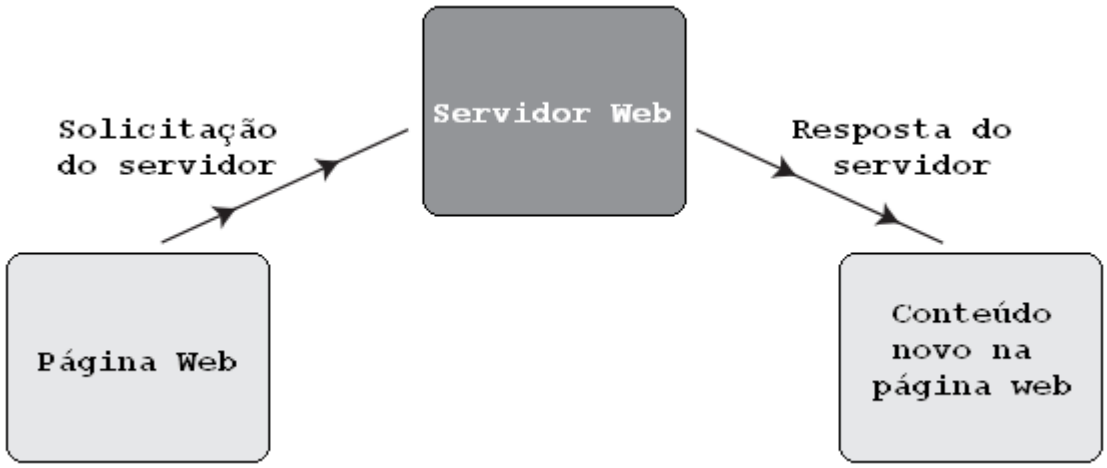


Figura 4 - Modelo tradicional de solicitação/resposta do servidor
Fonte: Adaptado de BABIN, 2007.

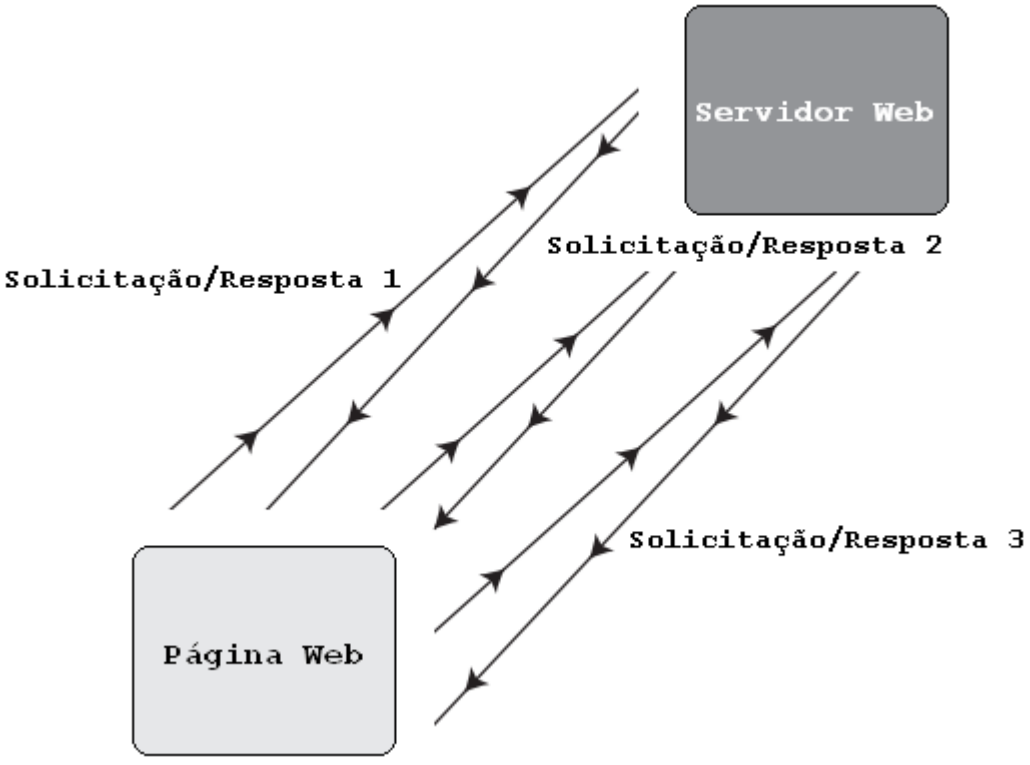


Figura 5 - Modelo de solicitação/resposta do servidor utilizando Ajax
Fonte: Adaptado de BABIN, 2007.

A metodologia Ajax será utilizada no presente trabalho de conclusão de curso para permitir a interatividade com a página *web* e evitar que ela seja atualizada completamente para cada nova solicitação ou envio de dados.

2.8 PROTOCOLO OPC

OPC é a sigla correspondente ao protocolo de comunicação OLE (*Object Linking and Embedding*) para controle de processos, desenvolvido a partir de 1996 para solucionar problemas de interoperabilidade em sistemas de automação industrial. Como atualmente a especificação é utilizada tanto para controle de processos quanto para manufatura discreta e automação predial e não utiliza mais somente OLE, mas também XML e .NET *Framework*, a OPC *Foundation* mudou o significado do acrônimo para *Open Platform Communication*.

Segundo a OPC *Foundation* (1998), antes do surgimento do OPC uma aplicação-cliente como um sistema supervisório, por exemplo, requeria acesso a fonte de dados do sistema através de um *driver*. O desenvolvimento desse *driver* gerava problemas como duplicação de esforços, inconsistências entre *drivers*, falta de suporte quando houvesse mudanças de funcionalidade de *hardware* e conflitos de acessos. Com o surgimento do padrão OPC, a comunicação entre os componentes do sistema pode ser feita sem a necessidade de *drivers* proprietários e fechados.

Conforme pode ser visto na figura 6, o modelo utilizado no OPC é o cliente-servidor, sendo que o servidor gerencia os objetos OPC e oferece suas respectivas interfaces (OPC Foundation, 2003).

Figura 6 - Arquitetura clássica do OPC
Fonte: Adaptado de *OPC Foundation*, 2003.

No presente trabalho, o protocolo OPC será utilizado na comunicação entre o *gateway* (Elipse E3) e o CLP. Sua escolha se deve ao fato de ser um protocolo aberto, de larga utilização na área industrial e por permitir a comunicação entre sistemas de fabricantes diferentes sem a necessidade de *drivers* proprietários. O servidor OPC utilizado será o KEPServerEx e o cliente OPC será o Elipse E3.

2.9 ELIPSE E3 / GATEWAY

Segundo a ITS (do inglês *Institute for Telecommunication Sciences*), um *gateway* é um nó de rede equipado para fazer a interface com outra rede que usa um protocolo de comunicação diferente. Um *gateway* também pode ser um computador ou aplicativo configurado para realizar essa interface entre redes com protocolos diferentes.

Na Figura 7 pode-se observar um exemplo de utilização de um *gateway* para realizar a comunicação entre o sistema de supervisão e o controlador, convertendo o protocolo *Modbus* TCP/IP para o protocolo de comunicação do controlador ou OPC, por exemplo.

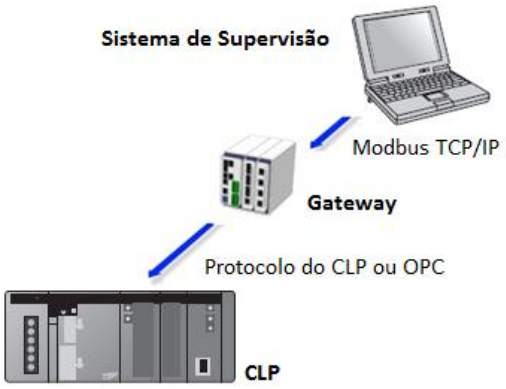


Figura 7 - Exemplo de utilização de um *gateway*
Fonte: Autoria própria.

Dentro do escopo do presente trabalho de conclusão de curso, existe a necessidade de converter o protocolo de comunicação do CLP Omron para o protocolo *Modbus* TCP, visto que o controlador não possui suporte nativo ao *Modbus*. Optou-se por não utilizar um *gateway* físico para fazer a conversão dos protocolos mas sim por utilizar um programa para desempenhar essa função. Para converter os protocolos é feito o mapeamento das *tags* OPC e um espelhamento dessas *tags* no protocolo *Modbus*.

Os programas Elipse E3 e LabVIEW foram considerados para realizar essa conversão de protocolos e a decisão tomada pela equipe foi a de utilizar o Elipse E3, visto que este já foi utilizado pelos membros da equipe em implementações anteriores. Visto que a função do Elipse E3 neste projeto é basicamente a de um gateway implementado em *software*, ele será referenciado como *gateway* ao longo to trabalho.

2.10 SISTEMAS SCADA

Segundo Bailey (2003), a expressão “Controle de Supervisão e Aquisição de Dados”, do inglês *“Supervisory Control And Data Acquisition”* (SCADA), está presente no ambiente industrial desde o nascimento dos sistemas de controle. Os primeiros sistemas SCADA utilizados para controle e aquisição dos dados surgiram na forma de sinalizadores, medidores e registradores de papel.

Segundo Boyer (2004), SCADA é a tecnologia que permite ao usuário coletar dados de uma ou de mais instalações distantes e enviar instruções de controle para essas instalações.

O conceito de SCADA, presente no trabalho, é utilizado para entendimento do sistema de supervisão do FMS.

3 DESENVOLVIMENTO

Neste capítulo será abordada a programação do CLP, a configuração do servidor OPC, a programação do Elipse E3, a configuração do servidor HTTP, o desenvolvimento do sistema de supervisão em PHP e sua interface de controle.

O desenvolvimento do trabalho compreendeu as seguintes etapas:

- Interpretação do protocolo Modbus
- Implementação das funções Modbus na linguagem PHP
- Programação do CLP Omron CJ1M
- Configuração do servidor OPC
- Mapeamento das *tags* OPC no Elipse E3 para comunicação com o protocolo Modbus
- Instalação e configuração do servidor Apache
- Desenvolvimento da página *web* e da interface de controle

As etapas do projeto podem ser visualizadas no diagrama da figura 8.

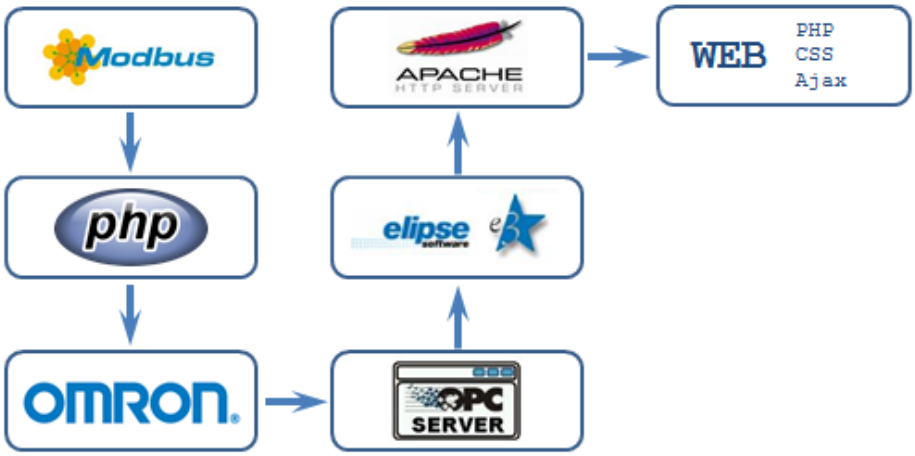


Figura 8 - Diagrama das etapas do projeto
Fonte: Autoria própria.

Os softwares utilizados no desenvolvimento foram o Elipse E3, o Apache HTTP Server, o servidor OPC KepServer, PHP, o software programador do CLP CX One e um editor de textos ASCII.

3.1 PROGRAMAÇÃO DO CLP

Para realizar a programação do CLP foi necessário compreender o funcionamento do FMS presente na UTFPR, apresentado na figura 9, definir um processo a ser usado para exemplificar o funcionamento do projeto, mapear as entradas e saídas do CLP, elaborar a lógica de controle e implementar o programa utilizando o aplicativo CX One.

Figura 9 – Sistema flexível de manufatura da UTFPR
Fonte: Autoria própria.

O FMS presente na UTFPR é constituído dos seguintes dispositivos:

- Controlador Omron CJ1M
- Armazém automatizado
- Esteira transportadora
- Braços robóticos Mitsubishi
- Torno e fresa CNC Denford
- Estação de trabalho para controle de qualidade utilizando visão de máquina.

Visto que o processo do FMS e a programação do CLP não eram o foco principal do trabalho, optou-se pela definição de um processo simples sem o uso dos braços robóticos, das máquinas CNC e do equipamento de visão de máquina e a lógica foi desenvolvida com controle sequencial simples.

O controlador utilizado no projeto foi o Omron CJ1M presente no FMS e o aplicativo CX One foi usado para fazer a programação do mesmo. A lógica foi desenvolvida para que o sistema opere de modo automático e manual, sendo o modo automático responsável por executar a lógica sequencial demonstrada no grafcet da figura 10. A seleção do modo de operação é feita pelo sistema de supervisão e qualquer comando pode ser executado quando o sistema opera manualmente.

Para execução dos comandos pelo sistema de supervisão foram mapeados todos os comandos numa área de memória determinada e esta área de memória foi mapeada no servidor OPC.

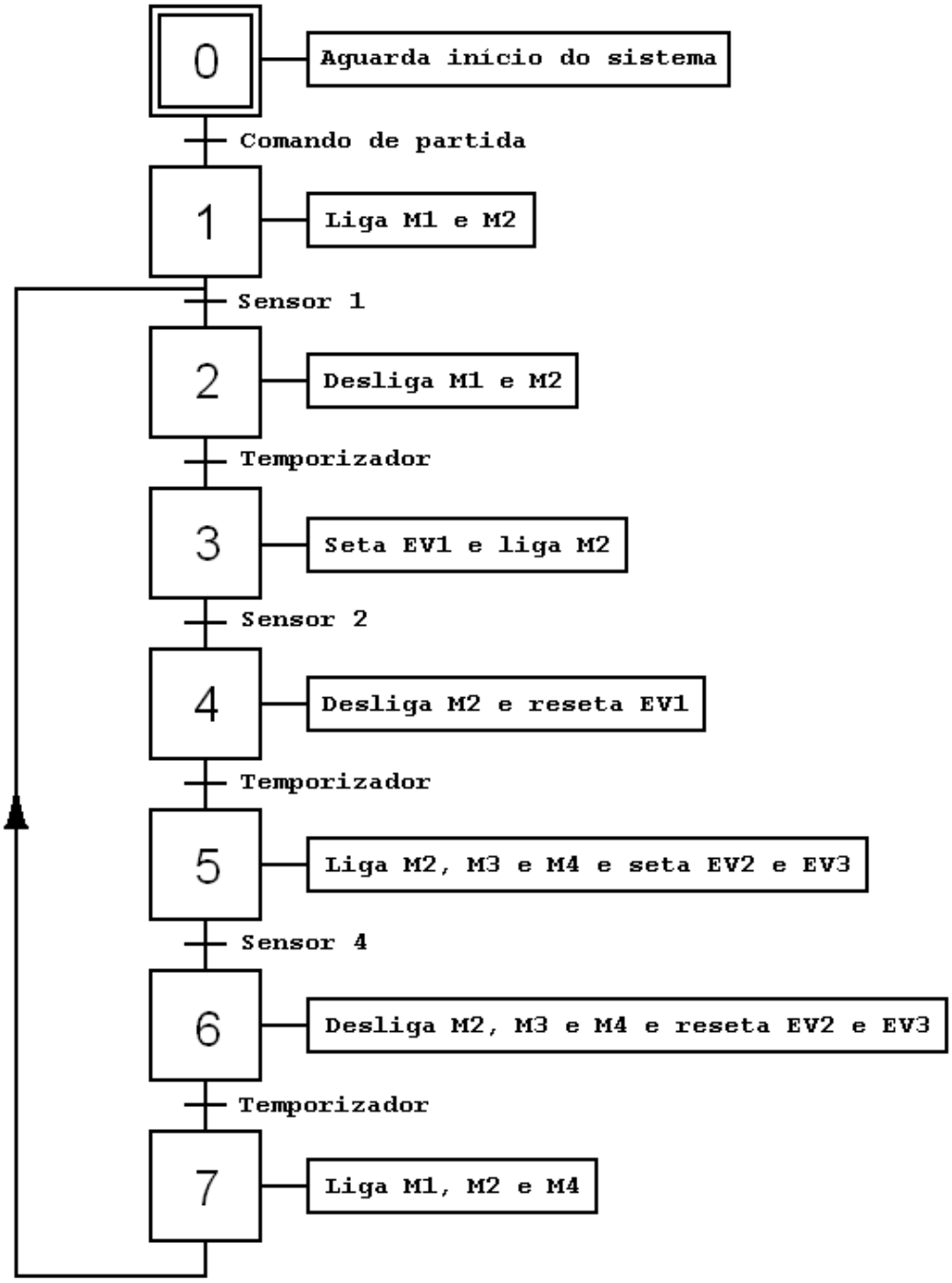


Figura 10 - GRAFCET do programa do CLP
Fonte: Autoria própria.

3.2 SERVIDOR HTTP

No presente projeto foi adotado o Apache HTTP Server v.2.2.22 como servidor hospedeiro do sistema de supervisão. Para funcionamento adequado do servidor, deve ser instalado em conjunto o módulo do PHP. Após

instalado o PHP com o componente *socket* o servidor HTTP está pronto para rodar o sistema de supervisão.

3.3 SERVIDOR OPC

O aplicativo KEPServerEx foi utilizado como servidor OPC neste projeto. Através do mapeamento de dados feito pelo KEPServerEx o cliente OPC (*gateway*) irá converter essa mesma base para o protocolo *Modbus* TCP. Para configuração do servidor OPC, deve-se seguir uma metodologia básica.

A metodologia básica trata-se da seleção do meio físico a ser utilizado, o *driver* de comunicação do controlador utilizado, bem como o modelo do próprio controlador. No presente projeto, foi adotado o meio físico *ethernet*, sendo utilizado o OMROM FINS Ethernet como *driver* de comunicação entre o controlador CJ1M e o servidor OPC.

Após ser realizada a configuração básica, a configuração essencial para funcionamento é a configuração do protocolo FINS. Para isso é necessário saber o endereço de origem (CLP) e o endereço do servidor de origem (servidor OPC).

Conforme pode ser observado na figura 11, o endereço de destino do nó é o último *byte* do endereço IP do controlador. Quando o endereço do controlador é 192.168.0.180 por exemplo, o endereço de destino do nó será 180. O mesmo raciocínio pode ser adotado para o endereço de origem, sendo este comumente variável.

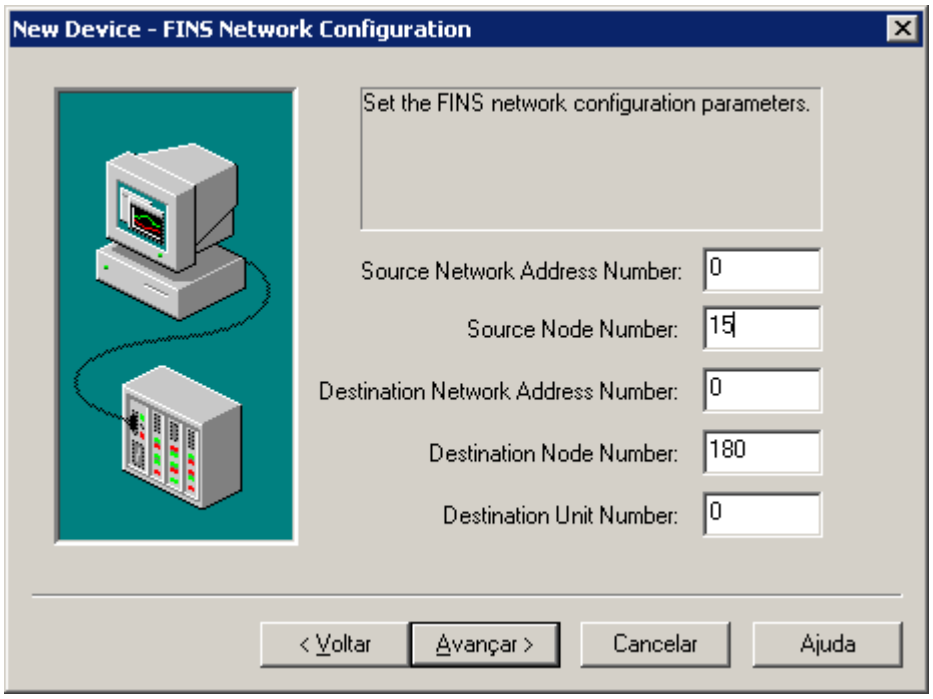


Figura 11 - Configuração do protocolo FINS
Fonte: Autoria própria.

Após os parâmetros básicos de comunicação serem finalizados, deve-se iniciar a criação da base de dados a ser disponibilizada para o sistema de supervisão.

Conforme a figura 12, para criar um *tag* OPC deve-se escolher um nome para o mesmo e apontá-lo a um endereço real no controlador. No presente projeto a base de dados OPC é idêntica à base de dados utilizada no sistema de supervisão. A base de dados foi dividida em variáveis de leitura e comandos e posteriormente foi feito o mapeamento das variáveis de acumuladores.

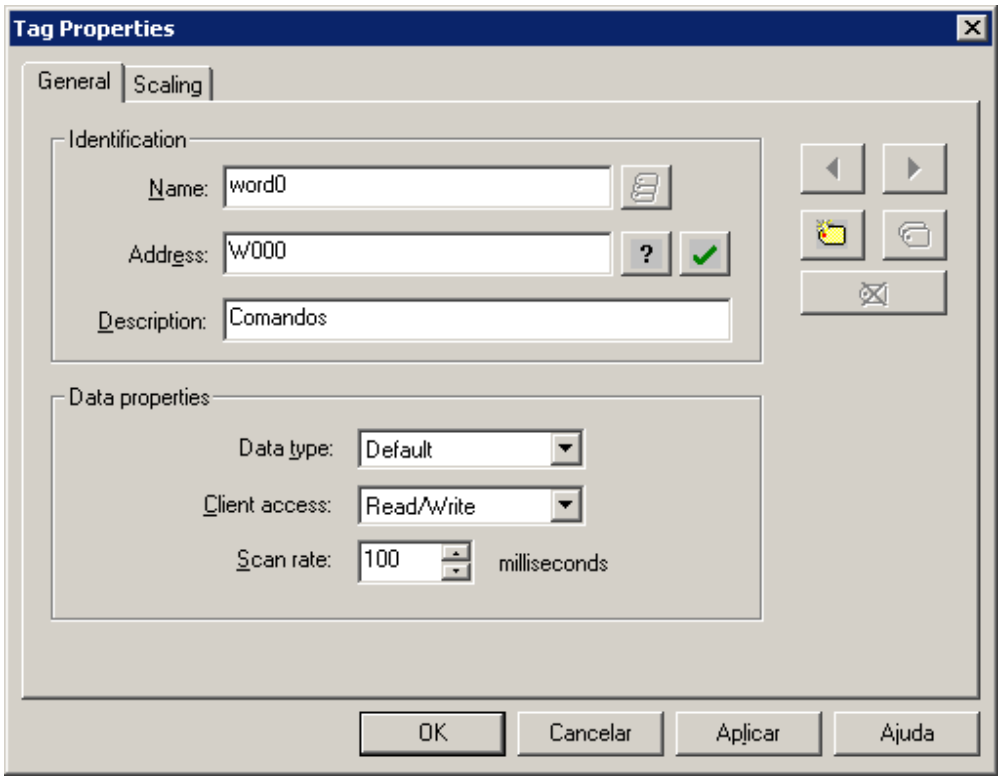


Figura 12 - Mapeando um *tag* OPC
Fonte: Autoria própria.

3.4 GATEWAY

O conceito de *gateway* será adotado nesta seção, para auxiliar no entendimento da conversão de protocolos. No presente projeto, para realizar a conversão dos protocolos OPC-Modbus TCP, os dados disponibilizados no protocolo OPC foram mapeados no protocolo Modbus TCP, esse mapeamento de dados será nomeado *gateway*. Para conversão dos protocolos, a solução foi desenvolvida no Elipse E3 que fará a conversão de protocolos e possibilitará a integração entre os sistemas.

A configuração do *gateway* é um passo fundamental para o funcionamento da solução, pois o *gateway* torna transparente a comunicação entre o servidor *web* e o controlador. Para tanto, o *gateway* é configurado como um cliente OPC (realiza leituras/escritas no servidor OPC) e ao mesmo tempo é um servidor *Modbus* TCP para o sistema de supervisão, espelhando a base de dados do protocolo OPC.

A etapa inicial de parametrização parte do mapeamento da base de dados OPC, anteriormente configurada e definida no servidor OPC. Para tanto é necessário primeiro selecionar o servidor OPC KepServer, conforme figura 13.

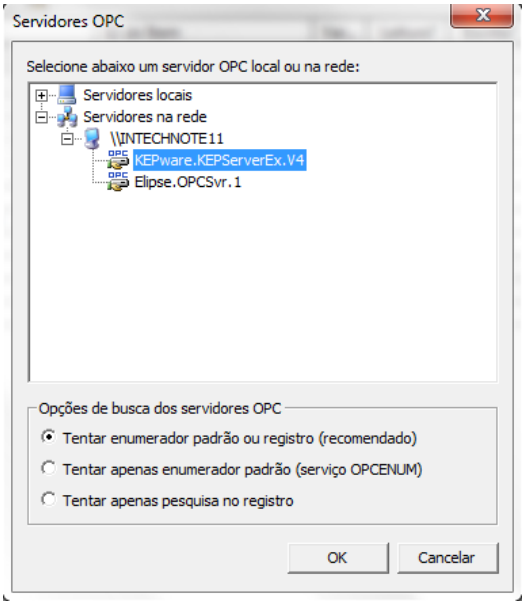


Figura 13 - Seleção de um servidor OPC
Fonte: Autoria própria.

Com a seleção do servidor de dados, o *gateway* já fica apto a receber a lista de pontos do controlador. No Elipse E3, o mapeamento dos pontos pode ser feito manualmente, criando as variáveis diretamente na área de trabalho da base de dados ou importando diretamente pela ferramenta de importação de tags, conforme figura 14.

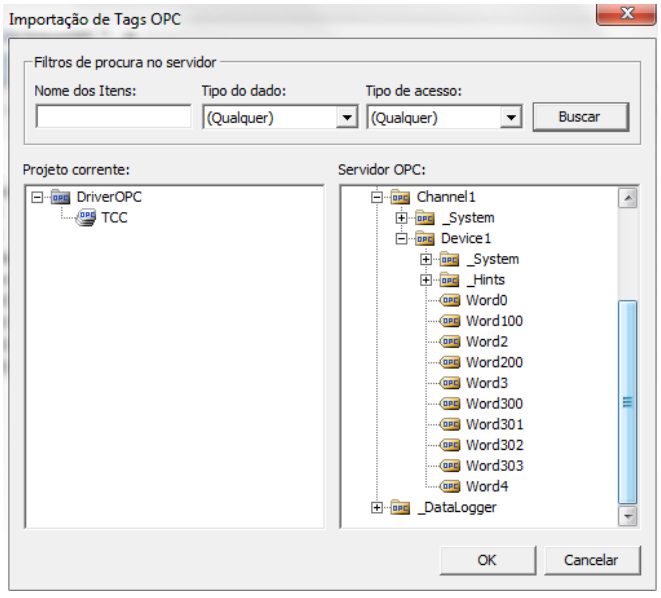
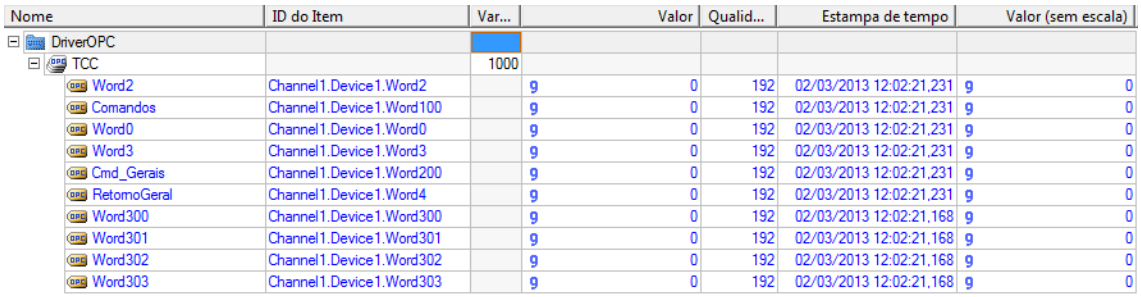


Figura 14 - Importação de tags OPC
Fonte: Autoria pópria.

No presente projeto, foi utilizada a ferramenta de importação para mapeamento da base de dados, uma vez que não existe vantagem no mapeamento manual, pois a base de dados criada no servidor OPC é idêntica à base de dados utilizada pelo *gateway*. A base de dados OPC pode ser vista na figura 15.



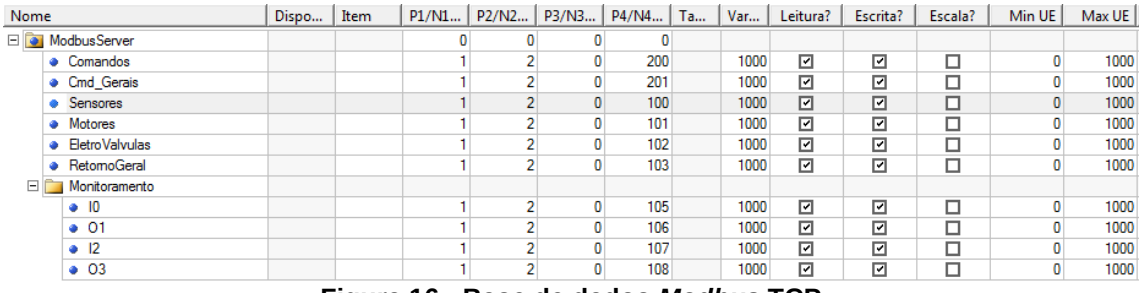
Nome	ID do Item	Var...	Valor	Qualid...	Estampa de tempo	Valor (sem escala)
DriverOPC						
TCC		1000				
Word2	Channel1.Device1.Word2		0	192	02/03/2013 12:02:21,231	0
Comandos	Channel1.Device1.Word100		0	192	02/03/2013 12:02:21,231	0
Word0	Channel1.Device1.Word0		0	192	02/03/2013 12:02:21,231	0
Word3	Channel1.Device1.Word3		0	192	02/03/2013 12:02:21,231	0
Cmd_Gerais	Channel1.Device1.Word200		0	192	02/03/2013 12:02:21,231	0
RetornoGeral	Channel1.Device1.Word4		0	192	02/03/2013 12:02:21,231	0
Word300	Channel1.Device1.Word300		0	192	02/03/2013 12:02:21,168	0
Word301	Channel1.Device1.Word301		0	192	02/03/2013 12:02:21,168	0
Word302	Channel1.Device1.Word302		0	192	02/03/2013 12:02:21,168	0
Word303	Channel1.Device1.Word303		0	192	02/03/2013 12:02:21,168	0

Figura 15 - Base de dados OPC
Fonte: Autoria própria.

O segundo passo da configuração do *gateway* é definir a base de dados do protocolo *Modbus* TCP que será utilizada pelo sistema de supervisão. Nesta etapa, pode-se espelhar a base de dados previamente mapeada no cliente OPC, ou definir apenas as variáveis que serão lidas e escritas via *web*.

Para criar uma base de dados *Modbus* TCP, deve-se selecionar o tipo de dados que será distribuído, endereço do escravo e endereço do registro a ser lido.

É nessa etapa que o *gateway* realizará sua função de conversor de protocolo. Na configuração do protocolo *Modbus* TCP, cada um dos *tags* que serão lidos pelo sistema de supervisão deverão receber os valores das variáveis do cliente OPC, conforme ilustrado na figura 16.



Nome	Dispo...	Item	P1/N1...	P2/N2...	P3/N3...	P4/N4...	Ta...	Var...	Leitura?	Escrita?	Escala?	Min UE	Max UE
ModbusServer			0	0	0	0							
Comandos			1	2	0	200		1000	☑	☑	☐	0	1000
Cmd_Gerais			1	2	0	201		1000	☑	☑	☐	0	1000
Sensores			1	2	0	100		1000	☑	☑	☐	0	1000
Motores			1	2	0	101		1000	☑	☑	☐	0	1000
EletroValvulas			1	2	0	102		1000	☑	☑	☐	0	1000
RetornoGeral			1	2	0	103		1000	☑	☑	☐	0	1000
Monitoramento													
I0			1	2	0	105		1000	☑	☑	☐	0	1000
O1			1	2	0	106		1000	☑	☑	☐	0	1000
I2			1	2	0	107		1000	☑	☑	☐	0	1000
O3			1	2	0	108		1000	☑	☑	☐	0	1000

Figura 16 - Base de dados *Modbus* TCP
Fonte: Autoria própria.

Na figura 17 é possível visualizar uma conexão simples de dados entre a variável lida no protocolo OPC (*word* 0) e o *tag* “Sensores” no protocolo *Modbus* TCP. Segundo a Elipse Software, as conexões simples alteram o valor do *tag* associado cada vez que o valor de origem for alterado. Com a conexão

realizada, o *tag* mapeado no servidor *modbus* já está preparado para receber os valores reais do CLP.

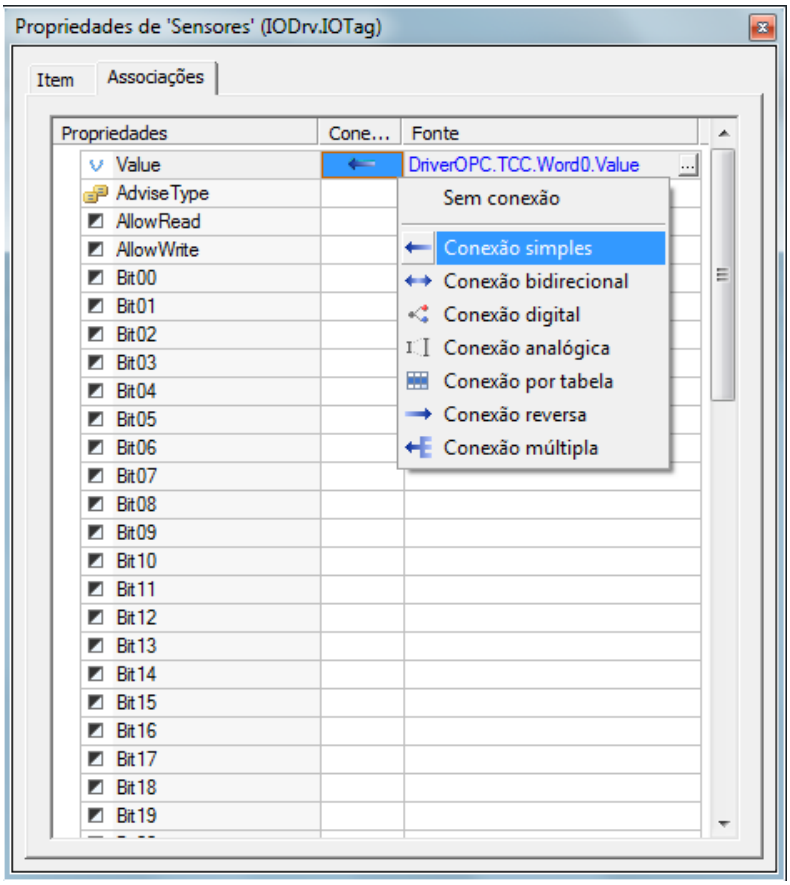


Figura 17 - Associação de Leitura OPC-*Modbus* TCP
Fonte: Autoria própria.

O mesmo processo foi realizado em cada uma das variáveis de leitura utilizadas pelo sistema de supervisão. Um processo semelhante foi adotado para os comandos efetuados pelo sistema de supervisão. Neste caso, a origem da variável é o protocolo *Modbus* TCP, sendo realizada uma conexão reversa, conforme pode ser visto na figura 18. Segundo a *Elipse Software*, uma conexão reversa é uma associação do objeto para a fonte, sendo neste caso aplicada para o protocolo *Modbus* TCP. Para manter o padrão, quando há uma variação no valor do registro de comando, o *gateway* escreve diretamente no driver de comunicação OPC.

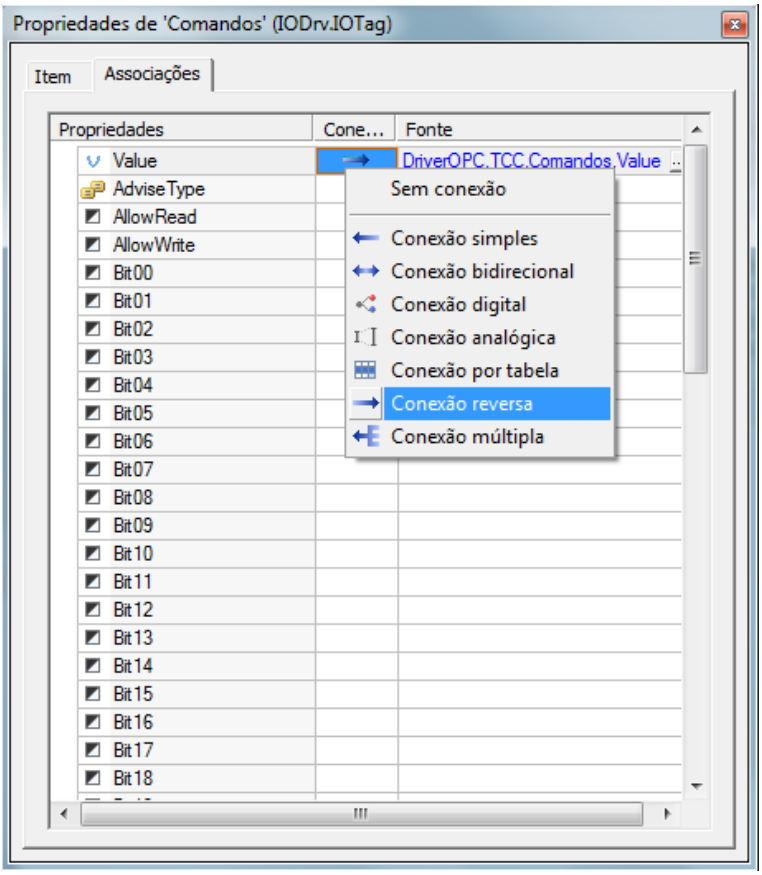


Figura 18 - Associação de Escrita OPC-*Modbus* TCP
Fonte: Autoria própria.

3.5 PHP

O sistema de supervisão, desenvolvido na linguagem de programação PHP, conecta-se ao sistema de controle através de uma conexão *socket* com o servidor *Modbus* TCP. Para tanto é necessária a instalação do PHP no servidor *web* e efetuar a instalação do módulo *Socket*, sendo este o módulo essencial para estabelecer instâncias de conexão para quaisquer protocolos de comunicação e/ou portas abertas em servidores. Utilizando o *socket*, operações como estabelecer conexão entre máquinas, enviar e receber dados e encerrar as conexões podem ser efetuadas. O mesmo tem função crucial e básica para início da comunicação entre o sistema *web* e o *gateway* ou qualquer dispositivo que possa conectar.

Conforme pode ser observado na figura 19, antes de iniciar a leitura das variáveis, a página abre a conexão com o servidor por meio de *socket*.

Com a conexão estabelecida a troca de mensagens entre o *gateway* e a página pode iniciar a qualquer momento contudo, em caso de falha, o usuário recebe como retorno uma mensagem de erro. Como a arquitetura do projeto possui um *gateway* e a página se conecta diretamente ao mesmo, a mensagem de erro informa que o problema é causado por possível queda no *gateway*.

```php
<?php

    // Inclusão da biblioteca e parâmetros
    include("./config/config_param.php");
    include("funcoes.php");

    // Instância Socket - Conexão com o Servidor Modbus TCP
    $con = @fsockopen($serverip,$porta, $errono, $errostr, 1) or
    die($errostr."<br>Erro número: ".$errono."<br>Este erro provavelmente ocorreu porque o Gateway está fora do ar.
    <br>Para alterar as configurações clique <a href=\"configuracoes.php\">aqui</a>");

    // Leitura de 4 Registros, apartir do registro 100
    $entrada    = LeMultRegistro(100,4,$con);

?>
```

Figura 19 - Estabelecendo Conexão com Servidor
Fonte: Autoria própria.

O sistema *web* foi subdividido em módulos. Dessa forma, as funções de leitura e escrita no protocolo *Modbus* funcionam de maneira independente das funções de setar e zerar *bit*, além da função de verificar qual o valor do *bit*. Em caso de problemas no retorno das funções básicas de leitura e escrita, a execução da página será interrompida e será gerada uma mensagem de erro para o usuário.

As funções essenciais para o funcionamento são prioritárias frente às demais, evitando assim que o sistema fique tentando executar o restante do programa e gerando custo computacional.

Conforme pode ser observado no fluxograma da figura 20, o sistema *web* conecta-se ao servidor *Modbus* TCP – neste caso o *gateway* – e inicia a requisição de dados através do canal de comunicação estabelecido. Nesse processo, a página aguarda a resposta do servidor e considera um tempo para *timeout* de recebimento da mensagem. Caso o servidor responda a mensagem, é feita uma verificação para validar a integridade do *frame* de comunicação. A validação é feita verificando o tamanho do *frame* de comunicação recebido e a verificação do cabeçalho (MBAP) *modbus* TCP. Uma mensagem muito curta, por exemplo, identifica que possivelmente o servidor não enviou todos os dados dentro do tempo esperado.

Com a validação do recebimento da mensagem, o *frame* de comunicação é desmontado de forma que apenas os dados relevantes (valores

dos registros) sejam o resultado do processo. O retorno da função é variável de acordo com a quantidade de registros solicitada para leitura, obtendo sempre um *array* como resposta.

É importante ressaltar que a utilização de registros inteiros, agrupando 16 *bits* é favorável pois reduz o tamanho do *frame* de comunicação e diminui o tamanho da base de dados do *gateway*.

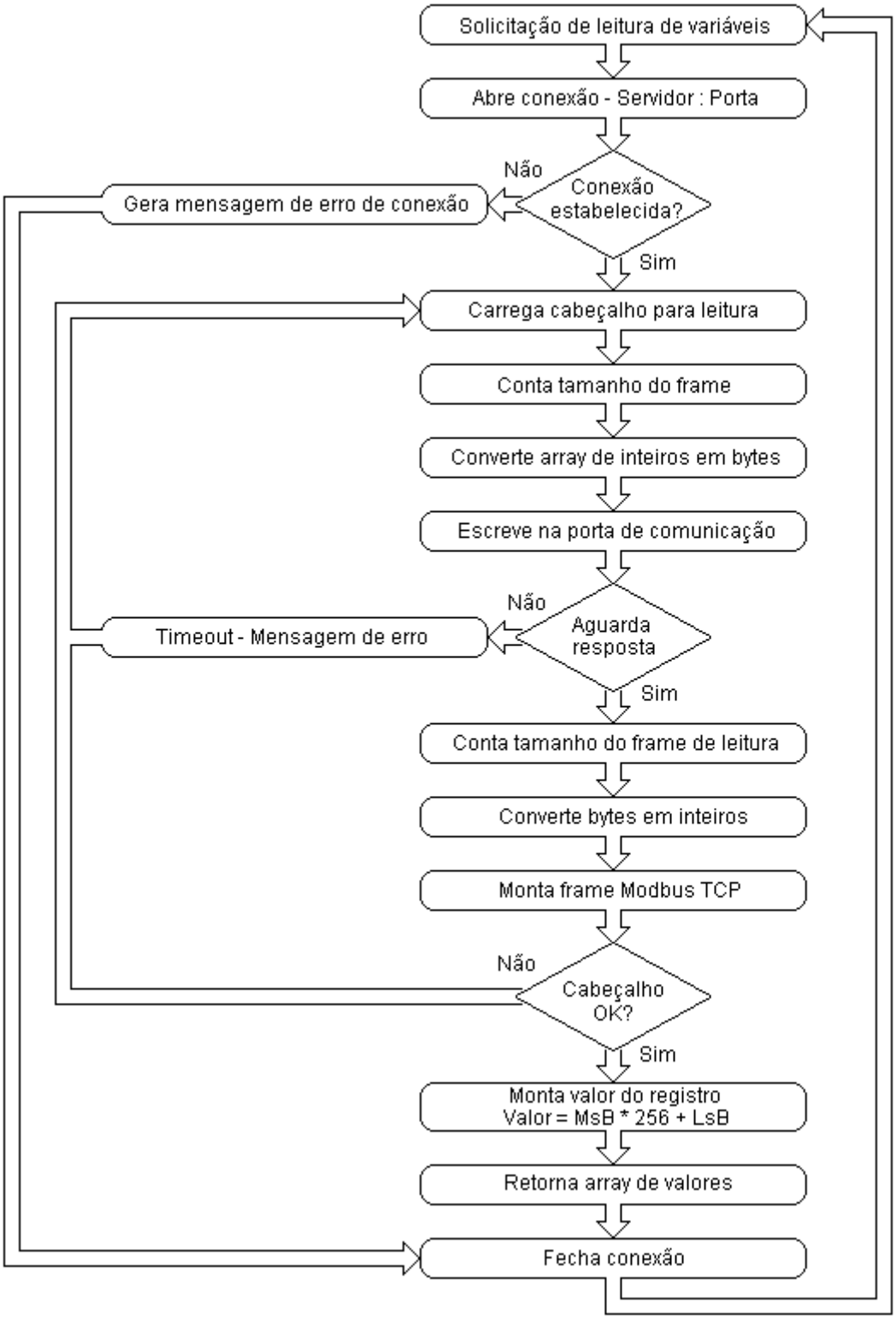


Figura 20 - Fluxograma da função de leitura
Fonte: Autoria própria.

Com o valor do registro obtido, as funções secundárias do sistema iniciam sua execução. O valor do registro que é obtido em valores inteiros é separado em *bits* para que a página saiba efetivamente qual o valor de determinada variável. As animações de tela são feitas de acordo com o valor

de cada *bit* das variáveis solicitadas. Para cada animação existem duas figuras, ou seja, o sistema de supervisão imprime para o usuário uma figura de acordo com o nível lógico do bit. Neste caso, o usuário final não percebe a diferença entre as figuras, uma vez que ambas as figuras possuem o mesmo alinhamento.

Como pode ser observado na figura 21, o sistema compara o valor do registro 101, *bit* 1 para gerar a animação para o usuário. Caso o motor esteja ligado, o usuário recebe a figura do motor na cor verde e em caso contrário o motor ficará vermelho, indicando estado desligado.

```
/*******************************************************/
#          TRATATIVA DE MOTORES                        #
/*******************************************************/
if(ValorBit($entrada[1],1)==1) { # Motor 2 (Próximo do Torno) Registro 101, Bit 1
    echo "<img class=\"ch_i00\" src=\"./Figuras/anima/motores/m1_ligado.png\">";
}
else {
    echo "<img class=\"ch_i00\" src=\"./Figuras/anima/motores/m1_desligado.png\">";
}
```

Figura 21 - Comparação do Valor de *Bit*
Fonte: Autoria própria.

O mesmo pode ser observado para os sensores e eletroválvulas, conforme figura 22. A variável das leituras é sempre o registro e *bit* de leitura, que será variável de acordo com o sinal desejado.

```
/*******************************************************/
#          TRATATIVA DE ELETRO VALVULAS                #
/*******************************************************/
if(ValorBit($entrada[2],0)==1) { // Eletrovalvula 1, Registro 102 - Bit 0
    echo "<img class=\"ev1\" src=\"./Figuras/anima/EV/EV1_atuada.PNG\">";
}
else {
    echo "<img class=\"ev1\" src=\"./Figuras/anima/EV/EV1_desatuada.png\">";
}
```

Figura 22 - Comparação do Valor de *Bit* Eletroválvulas
Fonte: Autoria própria.

Para executar comandos, o sistema de supervisão deve instanciar o *bit* e o registro *modbus* referente ao equipamento a ser comandado, conforme ilustrado na figura 23. Para isso, na rotina de executar comandos, são declarados o valor do registro e o *bit* referente a cada um deles, sendo possível então efetuar comandos individuais sem afetar outros comandos que já foram executados.

```php
include("./config/config_param.php");
include("funcoes.php");

$condicao = $_GET['valor'];         // Comando a ser efetuado (Ligar/Desligar/Auto/Partir)
$equipamento = $_GET['equipamento'];// Equipamento a ser comandado

// Definição de Bits e Registros
// De acordo com o Popup
    if($equipamento == "M1") {          // Comandos no Motor M1
        $registro = 200;
            if($condicao== "Ligar") {
                $bit = 0;               // Registro 200 bit 0, Ligar M1
            }
            else {
                $bit = 4;               // Registro 200 bit 4, Desligar M1
            }
    }
```

Figura 23 - Definição de Registro e *Bit* a ser comutado
Fonte: Autoria própria

Depois de declarados todos os comandos possíveis pelo sistema *web*, o sistema efetivamente escreve o valor do *bit*, no registro desejado, diretamente na conexão estabelecida. A parte do programa que representa a execução do comando está representada na figura 24.

```php
$con = @fsockopen ($serverip,$porta, $errono, $errostr, 1);

if($equipamento != "auto") {
    SetaBit($registro,$bit,$con);
    sleep(1);
    ZeraBit($registro,$bit,$con);
} else {
    if($condicao=="Automatico") {
        SetaBit($registro,$bit,$con);
    } else {
        ZeraBit($registro,$bit,$con);
    }
}
```

Figura 24 - Execução do comando
Fonte: Autoria própria

O posicionamento de objetos em tela é feito de maneira dinâmica, estabelecendo posições relativas dos objetos em relação à referência do navegador do usuário. O posicionamento pode ser realizado desta maneira graças à tecnologia *Cascading Style Sheets*, ou CSS, linguagem de estilo utilizada para definir a maneira gráfica como é apresentada a página ao usuário final. No presente projeto, foram definidas duas folhas de estilo: posicionamento e estilo.

A folha de posicionamento, como o próprio nome evidencia, define a posição de cada imagem animada do sistema.

A folha de estilo nomeada estilo, define as características visuais da página, como menu, área de visualização do sinótico e menu lateral.

3.6 INTERFACE DO USUÁRIO

A interface de operação do sistema – sinótico – foi desenvolvida utilizando a linguagem de programação PHP, a metodologia Ajax e o desenvolvimento de folhas de estilo para estilização do ambiente gráfico.

No sinótico do sistema (figura 25) é possível visualizar todo o sistema flexível de manufatura controlado numa vista superior. Com acesso ao sistema, é possível monitorar o estado dos motores, eletroválvulas e sensores, além de efetuar comandos de partida e parada.

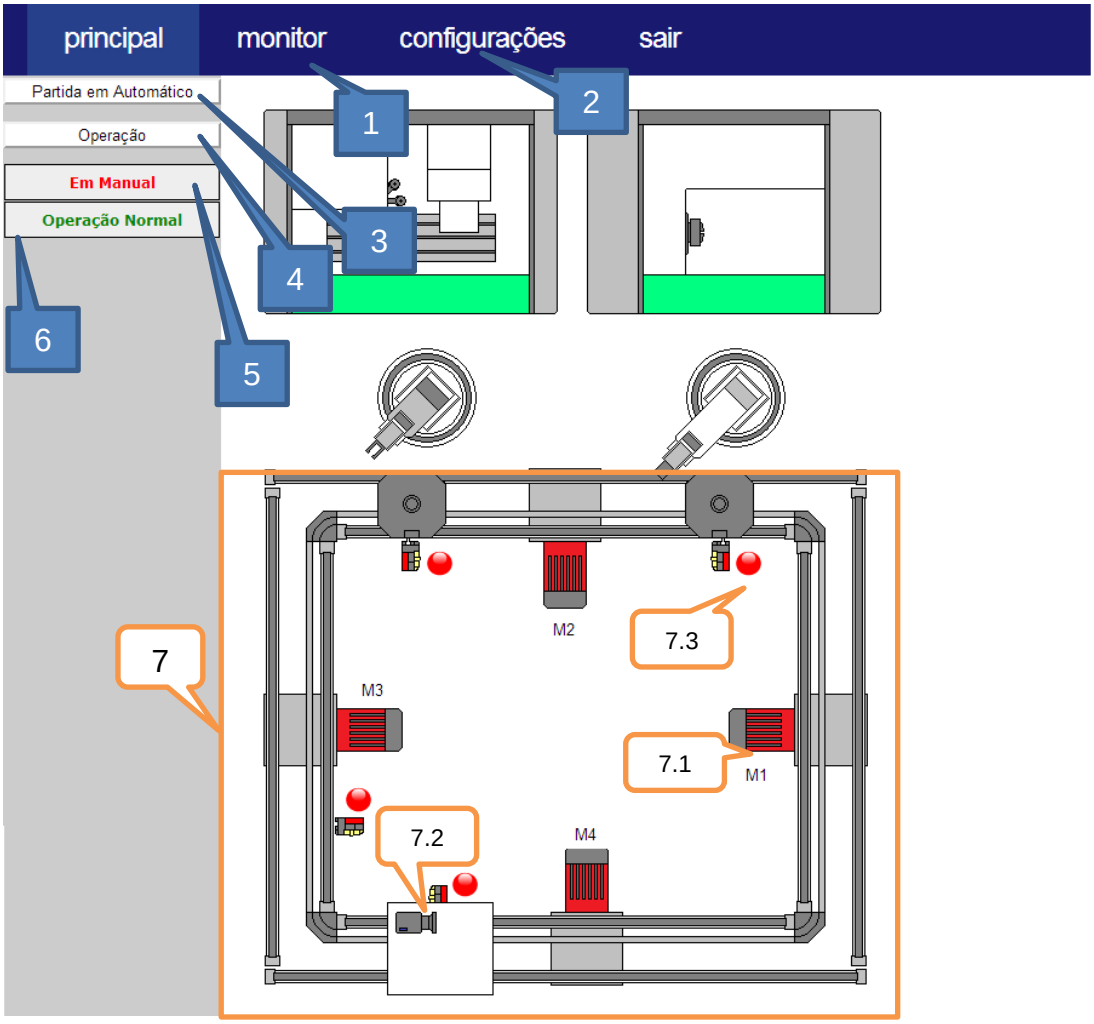


Figura 25 – Página principal (sinótico)
Fonte: Autoria própria.

A área 7, ilustrada na figura 25, foi delineada para representar a região onde é feito o controle do sistema. A área 7 é composta pelos sub-itens:

- 7.1 – Motores: Os motores podem ser comandados e são animados conforme o estado de funcionamento. Quando encontram-se ligados, sua figura no sinótico aparece na cor verde e vermelho quando desligados.
- 7.2 – Eletroválvulas: As eletroválvulas são animadas conforme estado de atuação. Quando as válvulas estão recuadas, sua indicação fica vermelha em tela, quando estão em posição de avanço sua indicação fica verde.
- 7.3 - Sensores: Os sensores são animados conforme estado de atuação. Quando o sensor está atuado, sua indicação fica verde e vermelho, quando desatuado.

O item 1, ilustrado na figura 25, permite acesso ao monitoramento de sinais de entrada e saída do CLP CJ1M, conforme figura 26.

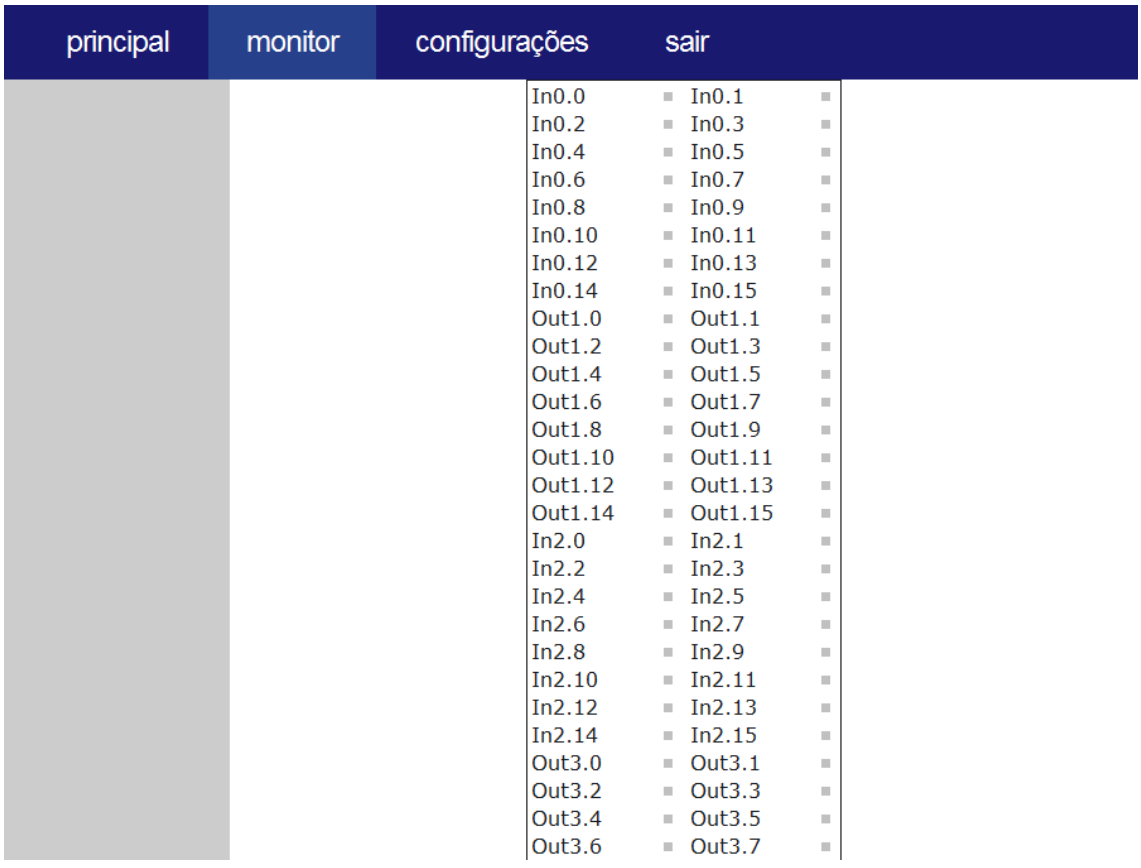


Figura 26 – Tela de monitoramento de sinais
Fonte: Autoria própria.

A tela de monitoramento exibe individualmente o estado de cada sinal digital do controlador. Sua função facilita a identificação de erros no sistema, pois o programador pode visualizar de forma clara e exata o estado do sistema sem a necessidade de utilizar o programa CX One, programador do CLP.

O item 2, permite acesso às configurações de conexão da página. A página de configuração não será demonstrada em figura, por não ser objeto de interesse no sistema.

O item 3, permite efetuar comandos de partida e parada da lógica de controle do processo, conforme ilustrado na figura 27.

Figura 27 – Tela de comandos de partida e parada
Fonte: Autoria própria.

O item 4, permite efetuar comandos de mudança do modo de operação do sistema, definindo se o controle será feito de maneira automática ou manual, conforme ilustrado na figura 28.

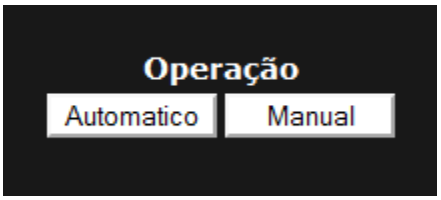


Figura 28 – Tela do modo de operação
Fonte: Autoria própria

O item 5 mostra a condição atual do controle do sistema, identificando se o sistema está em manual ou automático. O item 6 ilustra se existe alguma emergência atuada. Para comandar os motores, é necessário apenas um clique sob cada um que será exibido sua janela de comando, conforme figura 29.

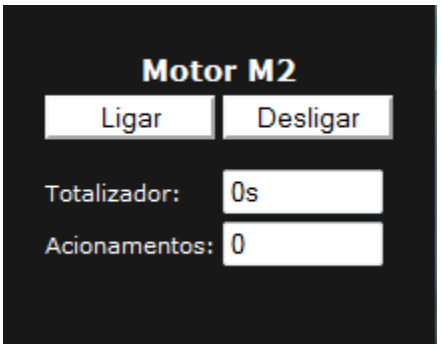


Figura 29 – Tela de comando dos motores
Fonte: Autoria própria

Na tela de comando de cada um dos motores é possível ligar ou desligar os mesmos, desde que seu modo de operação esteja em manual, além de visualizar informações como o tempo de operação e o número de acionamentos ocorridos.

4 CONSIDERAÇÕES FINAIS

Conclui-se que o Sistema WEB de Supervisão e Controle é um sistema viável para utilização no meio industrial e para fins didáticos e de pesquisa, entretanto sua aplicação deve ser limitada a processos que não são críticos, pois sua disponibilidade está limitada à qualidade da conexão com a internet.

O conceito de acesso ao ambiente industrial, como apresentado no trabalho, pode ser implantado para níveis de acesso gerenciais, em que a disponibilidade do sistema não é um fator determinante. Dessa forma, gerentes e diretores podem ter acesso em tempo real à dados produtivos e condições do processo , sem a possibilidade de efetuar comandos diretos.

Pelo controlador programável presente no FMS não possuir o protocolo *Modbus TCP*, foi necessária a utilização de um gateway para a comunicação. Para implantação de um projeto industrial, tal limitação poderia ser contornada especificando um controlador com tal suporte.

O projeto apresenta como principais vantagens em relação aos sistemas proprietários o baixo custo, a utilização de um protocolo de comunicação aberto e a exclusividade de um sistema web. Com essas características aliadas, o presente projeto foi considerado satisfatório e viável para soluções comerciais.

4.1 SUGESTÕES PARA TRABALHOS FUTUROS

Os seguintes itens foram identificados como sugestões para trabalhos futuros ou melhorias no sistema apresentado neste trabalho:

- Desenvolvimento de uma interface de *login* e definição de níveis de acesso a usuários, aumentando a segurança do sistema.
- Permitir acesso de operação do sistema somente para o primeiro usuário logado como administrador. Depois de implementado, caso um segundo usuário entre no sistema como administrador, ele não

terá a opção de enviar comandos ao sistema, somente visualizar sua operação.

- Armazenamento de dados do processo em um banco de dados para efetuar consultas online.
- Substituição do conceito do *gateway* para conexão direta entre controlador do processo e página *web*.
- Estudo da viabilidade de implantação de mais funções do protocolo Modbus TCP, possibilitando suporte a controladores de mais fabricantes.

REFERÊNCIAS

ACROMAG INCORPORATED, **Technical Reference – Modbus TCP/IP an Introduction to Modbus TCP/IP**, 2005. Disponível em: http://www.acromag.com. Acesso em: 26 março 2013.

APACHE SOFTWARE FOUNDATION, Disponível em: http://httpd.apache.org. Acesso em: 02 abril 2013.

BABIN, Lee. **Beginning Ajax with PHP: From Novice to Professional**. 1ª ed, Apress, 2007.

BOYER, Stuart A. **SCADA: Supervisory Control and Data Acquisition.** 3ª ed, Research Triangle Park: ISA, 2004.

DAVIS, Keir; TURNER, John W.; YOCOM, Nathan. **The definitive guide to Linux Network Programming.** 1ª ed, Apress, 2004.

Elipse Software, **Tutorial do E3**. 3ª ed., 2008.

Forrester Research, **Tablet Market Dynamics In Asia Pacific And Japan, 2012 To 2016.** Disponível em: http://www.forrester.com/Tablet+Market+Dynamics+In+Asia+Pacific+And+Japan+2012+To+2016/fulltext/-/E-RES75161. Acesso em: 25 janeiro 2013.

GARRET, Jesse J. **Ajax: A New Approach to Web Application**. Disponível em: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications. Acesso em: 02 abril 2013.

GROOVER, Mikell P. **Automation, Production Systems, and Computer-Integrated Manufacturing.** 2 ed.Upper Saddle River: Pearson-Prentice Hall, 2001.

Institute for Telecommunication Sciences, **Telecommunications: Glossary of Telecommunication Terms**, 1996.
Disponível em: http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm. Acesso em: 05 abril 2013.

Intel Corporation, **The Ethernet: A local Area Network**, 2ª ed, 1982.

International Telecommunication Unit, **Basic Reference Model: The Basic Model, 1994**.
Disponível em: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.200-199407-I!!PDF-E&type=items. Acesso em: 27 março 2013.

MICROSOFT, **Client/Server Architecture.** Disponível em: http://technet.microsoft.com/en-us/library/cc917543.aspx. Acesso em: 08 outubro 2013.

MITCHELL, Mark; OLDHAM, Jeffrey; SAMUEL Alex. **Advanced Linux Programming.** 1ª ed, New Riders Publishing, 2001.

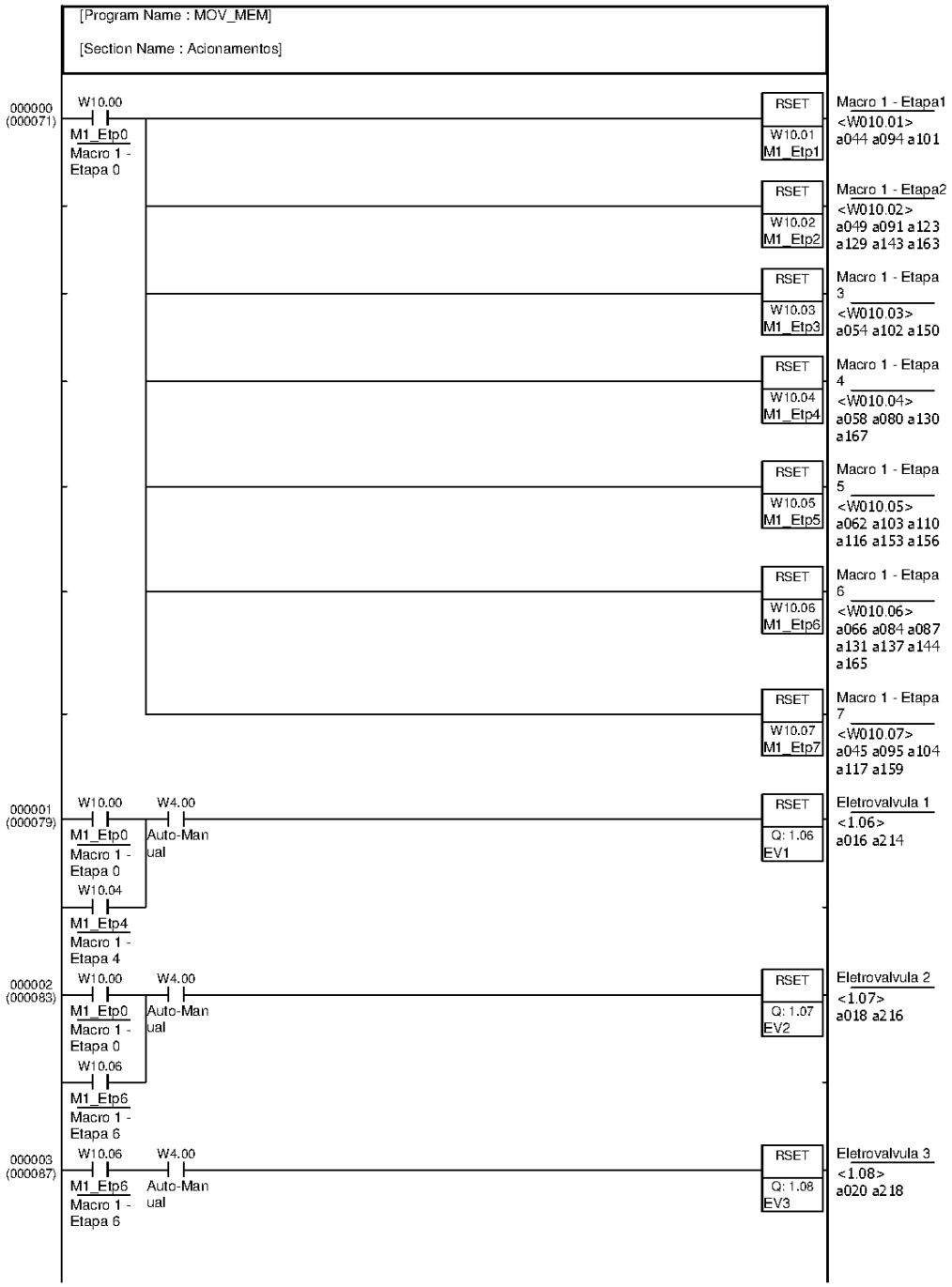
MODBUS ORGANIZATION, **Modbus Application Protocol Specification**, 2006. Disponível em: http://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. Acesso em 27 de março de 2013.

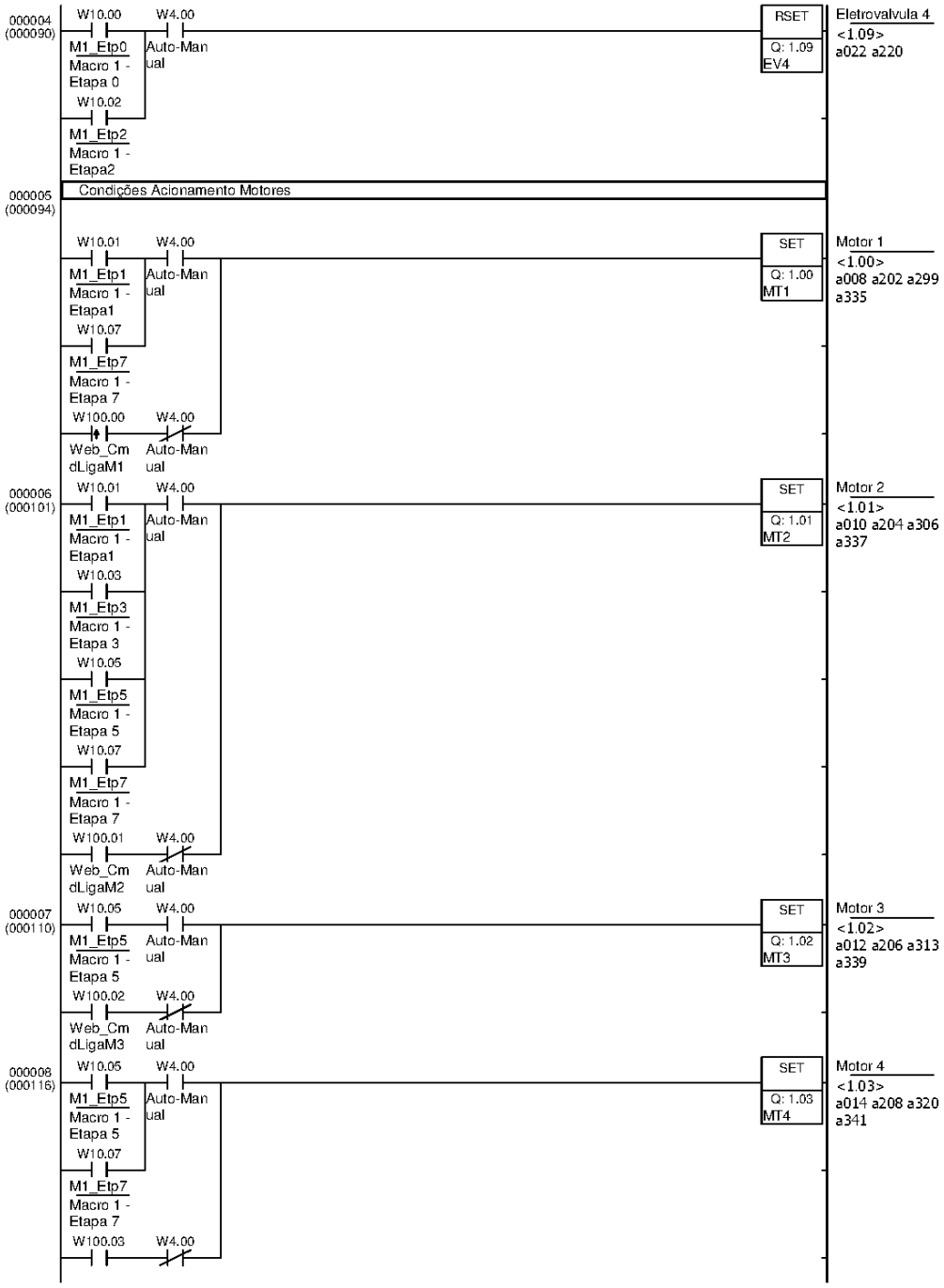
_____. **Modbus Messaging on TCP/IP Implementation Guide**, 2006. Disponível em: http://cars9.uchicago.edu/software/epics/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. Acesso em 25 de março de 2013.

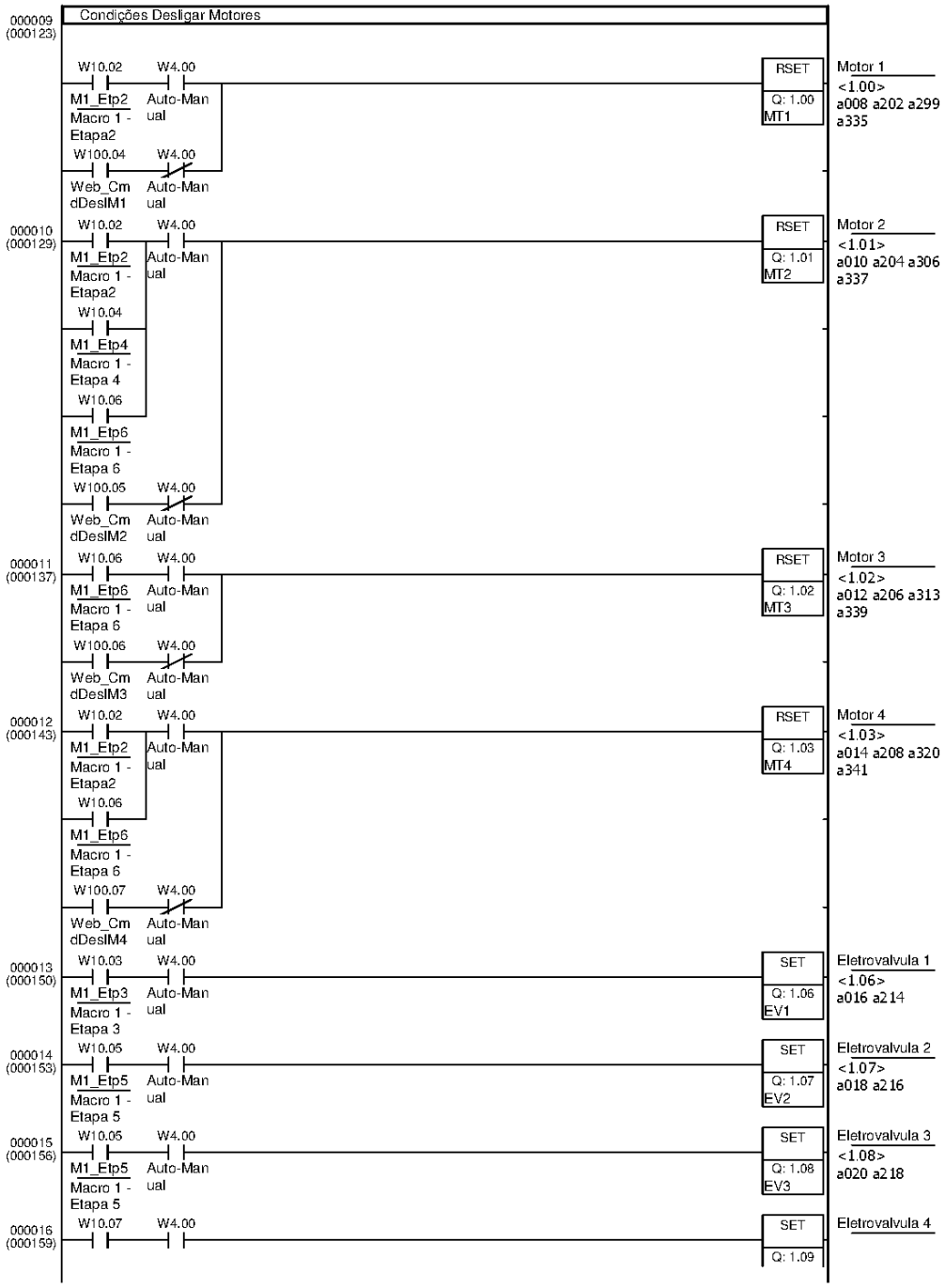
MODICON, Inc, **Modbus Protocol Reference Guide**, 1996. Disponível em: http://modbus.org/docs/PI_MBUS_300.pdf. Acesso em 27 de março de 2013.

APÊNDICE A – PROGRAMA DO CLP

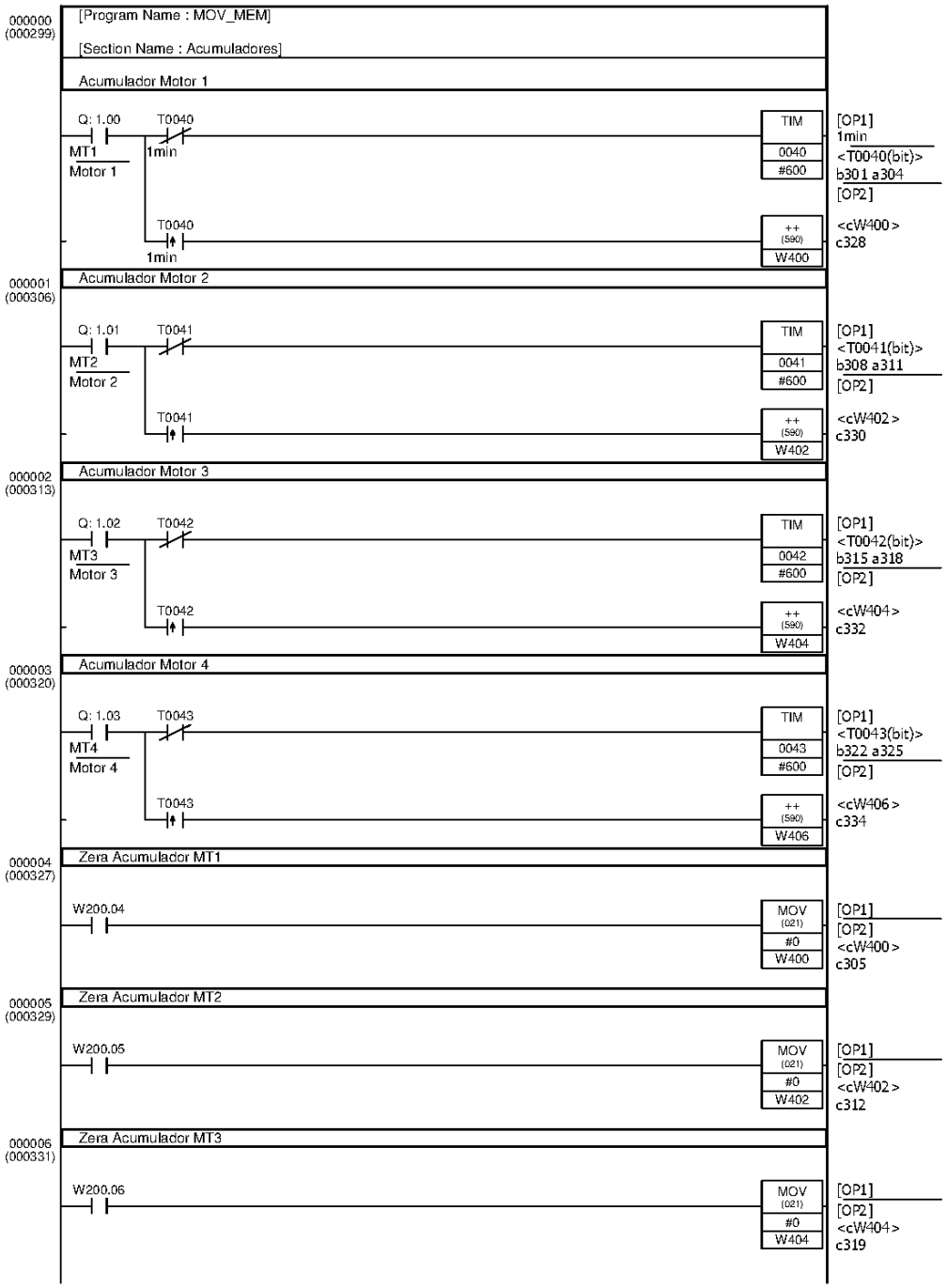
A seção de acionamentos do programa do CLP contém o controle para atuar diretamente nos atuadores do processo.

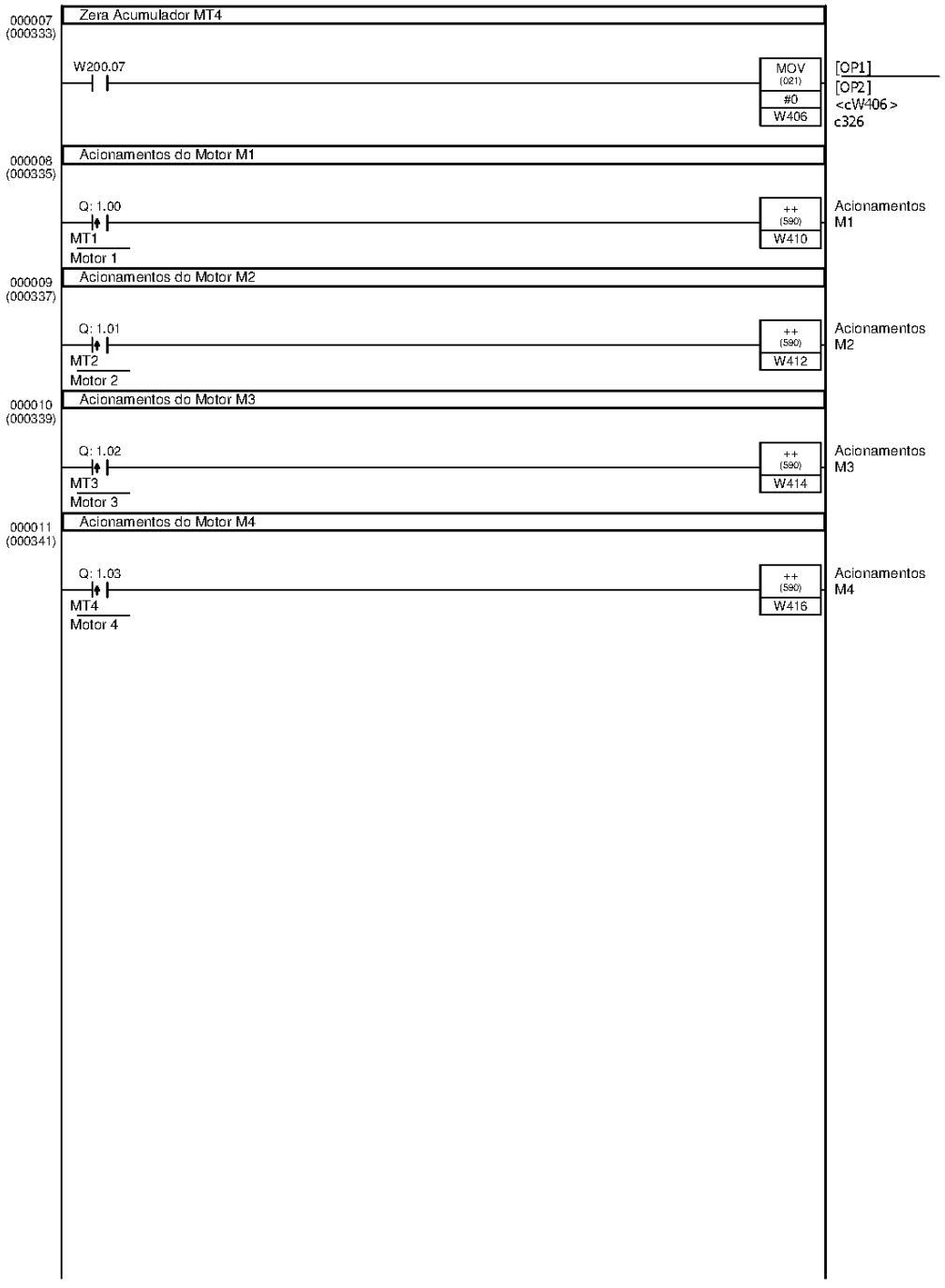



000004
(000090)
W10.00
M1_Etp0
Macro 1 - Etapa 0
W10.02
M1_Etp2
Macro 1 - Etapa2
W4.00
Auto-Manual
RSET
Q: 1.09
EV4
Eletrovalvula 4
<1.09>
a022 a220
000005
(000094)
Condições Acionamento Motores
000006
(000101)
W10.01
M1_Etp1
Macro 1 - Etapa1
W10.07
M1_Etp7
Macro 1 - Etapa 7
W100.00
Web_CmdLigaM1
W10.03
M1_Etp3
Macro 1 - Etapa 3
W10.05
M1_Etp5
Macro 1 - Etapa 5
W100.01
Web_CmdLigaM2
SET
Q: 1.00
MT1
Motor 1
<1.00>
a008 a202 a299 a335
Q: 1.01
MT2
Motor 2
<1.01>
a010 a204 a306 a337
000007
(000110)
W100.02
Web_CmdLigaM3
Q: 1.02
MT3
Motor 3
<1.02>
a012 a206 a313 a339
000008
(000116)
W100.03
Q: 1.03
MT4
Motor 4
<1.03>
a014 a208 a320 a341

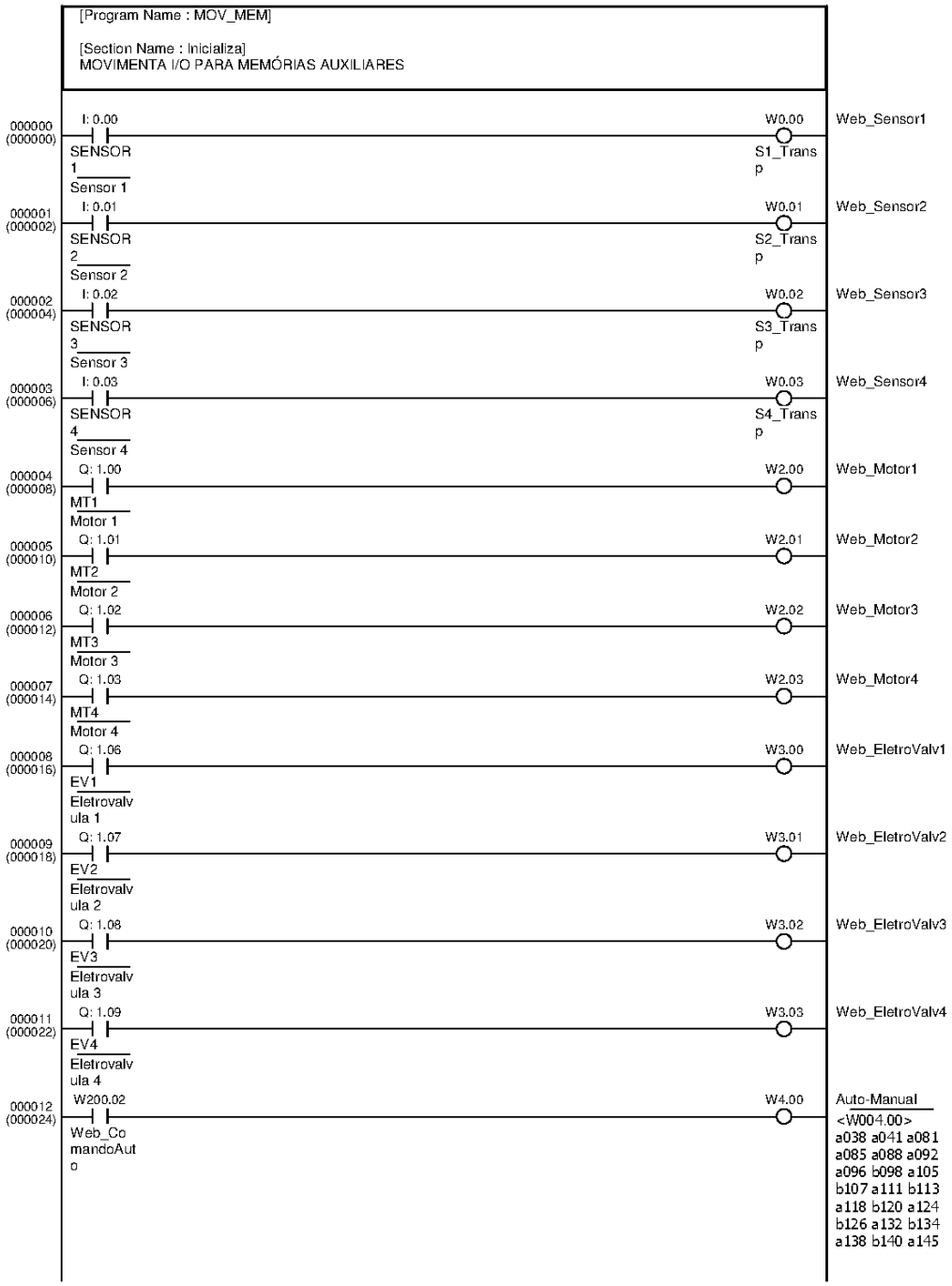

000009
(000123)
Condições Desligar Motores
W10.02
M1_Etp2
Macro 1 - Etapa2
W4.00
Auto-Man ual
W100.04
Web_Cm dDesIM1
W4.00
Auto-Man ual
RSET
Q: 1.00
MT1
Motor 1
<1.00>
a008 a202 a299 a335
000010
(000129)
W10.02
M1_Etp2
Macro 1 - Etapa2
W10.04
M1_Etp4
Macro 1 - Etapa 4
W10.06
M1_Etp6
Macro 1 - Etapa 6
W4.00
Auto-Man ual
W100.05
Web_Cm dDesIM2
W4.00
Auto-Man ual
RSET
Q: 1.01
MT2
Motor 2
<1.01>
a010 a204 a306 a337
000011
(000137)
W10.06
M1_Etp6
Macro 1 - Etapa 6
W4.00
Auto-Man ual
W100.06
Web_Cm dDesIM3
W4.00
Auto-Man ual
RSET
Q: 1.02
MT3
Motor 3
<1.02>
a012 a206 a313 a339
000012
(000143)
W10.02
M1_Etp2
Macro 1 - Etapa2
W10.06
M1_Etp6
Macro 1 - Etapa 6
W4.00
Auto-Man ual
W100.07
Web_Cm dDesIM4
W4.00
Auto-Man ual
RSET
Q: 1.03
MT4
Motor 4
<1.03>
a014 a208 a320 a341
000013
(000150)
W10.03
M1_Etp3
Macro 1 - Etapa 3
W4.00
Auto-Man ual
SET
Q: 1.06
EV1
Eletrovalvula 1
<1.06>
a016 a214
000014
(000153)
W10.05
M1_Etp5
Macro 1 - Etapa 5
W4.00
Auto-Man ual
SET
Q: 1.07
EV2
Eletrovalvula 2
<1.07>
a018 a216
000015
(000156)
W10.05
M1_Etp5
Macro 1 - Etapa 5
W4.00
Auto-Man ual
SET
Q: 1.08
EV3
Eletrovalvula 3
<1.08>
a020 a218
000016
(000159)
W10.07
W4.00
SET
Q: 1.09
Eletrovalvula 4

A seção de acumuladores do programa do CLP contém as lógicas de incremento do número de acionamentos dos motores, bem como a totalização do número de horas de operação.

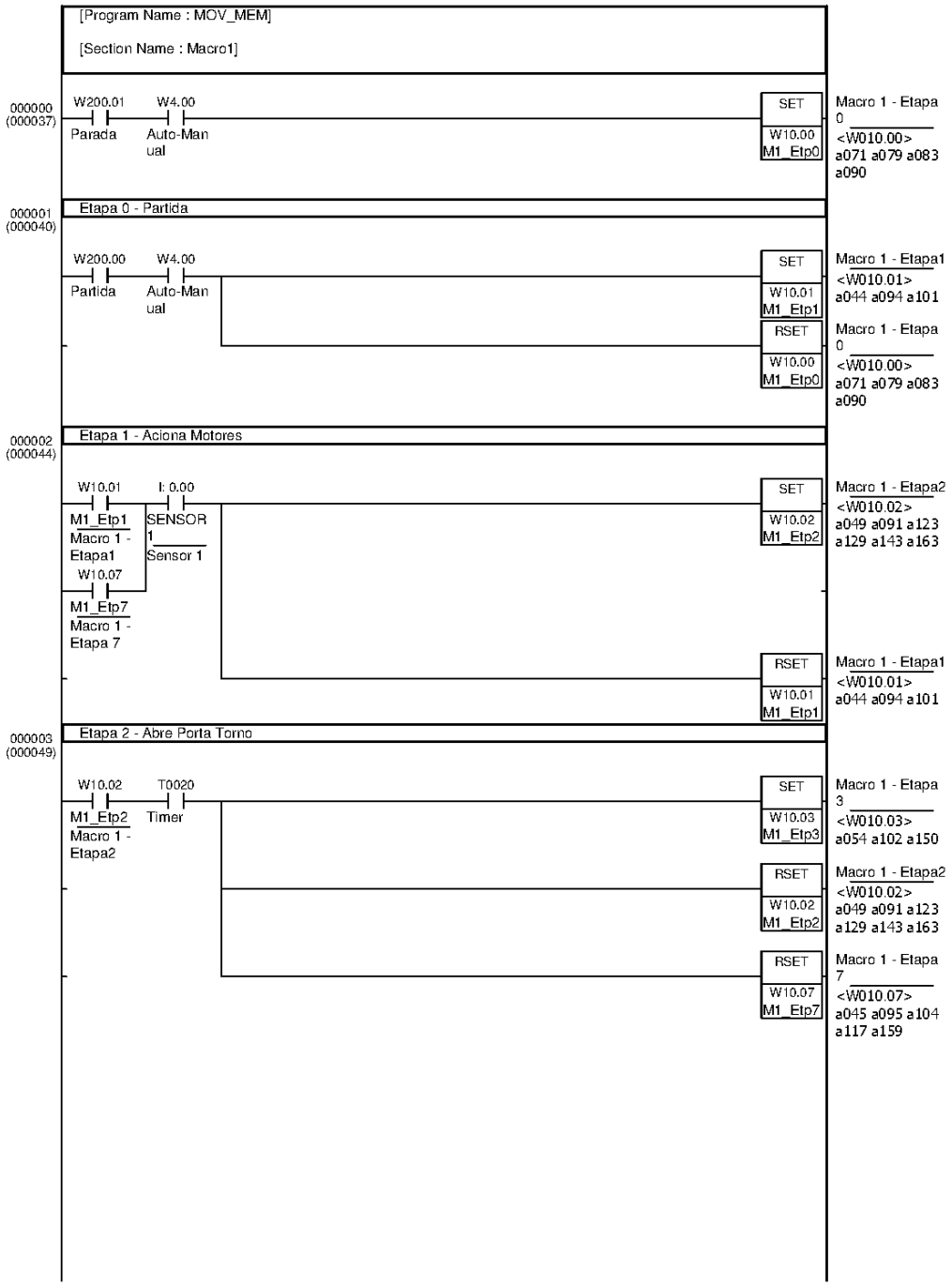


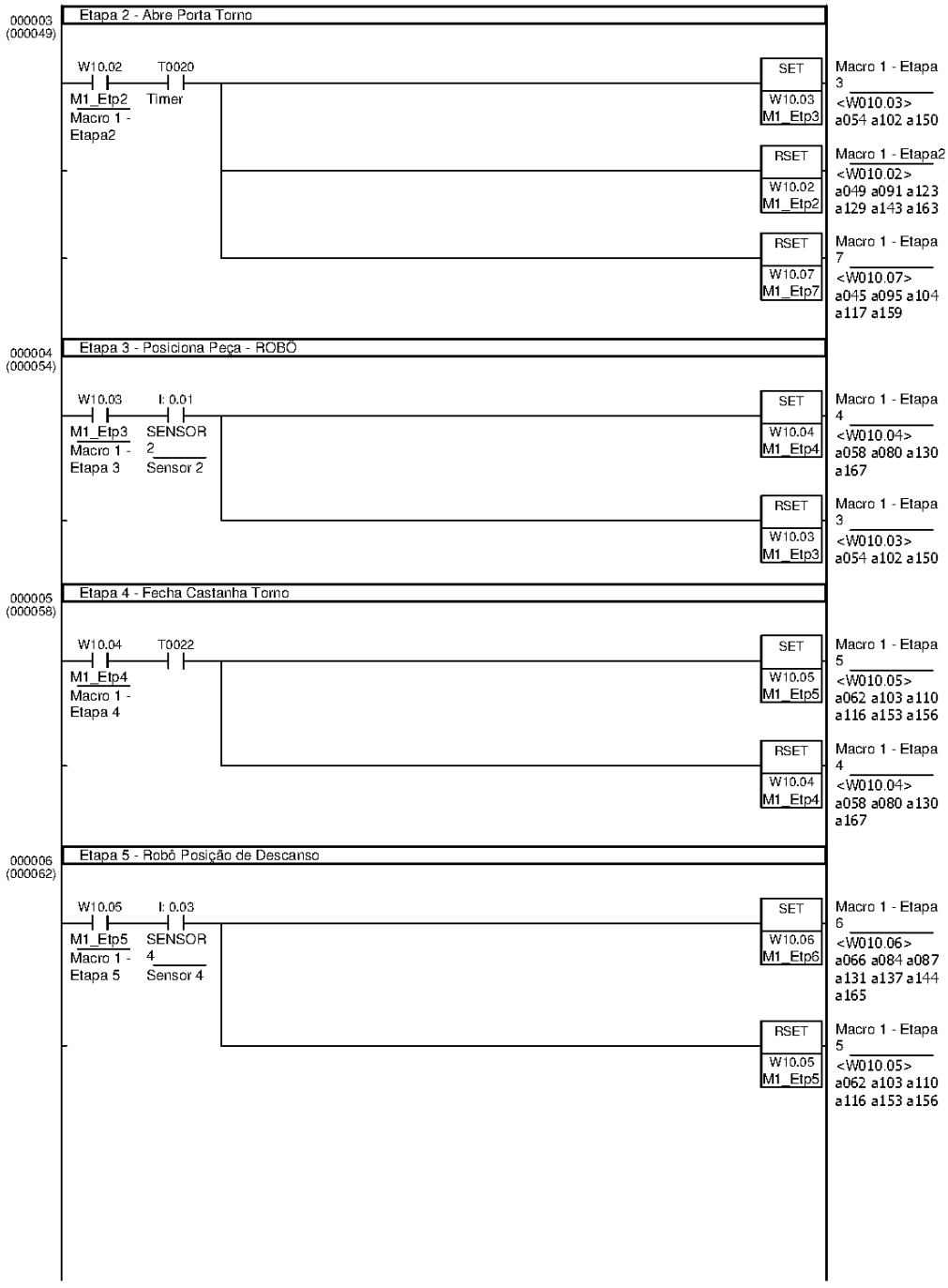

000007
(000333)
Zera Acumulador MT4
W200.07
MOV
(021)
#0
W406
[OP1]
[OP2]
<cW406>
c326
000008
(000335)
Acionamentos do Motor M1
Q: 1.00
MT1
Motor 1
++
(590)
W410
Acionamentos
M1
000009
(000337)
Acionamentos do Motor M2
Q: 1.01
MT2
Motor 2
++
(590)
W412
Acionamentos
M2
000010
(000339)
Acionamentos do Motor M3
Q: 1.02
MT3
Motor 3
++
(590)
W414
Acionamentos
M3
000011
(000341)
Acionamentos do Motor M4
Q: 1.03
MT4
Motor 4
++
(590)
W416
Acionamentos
M4

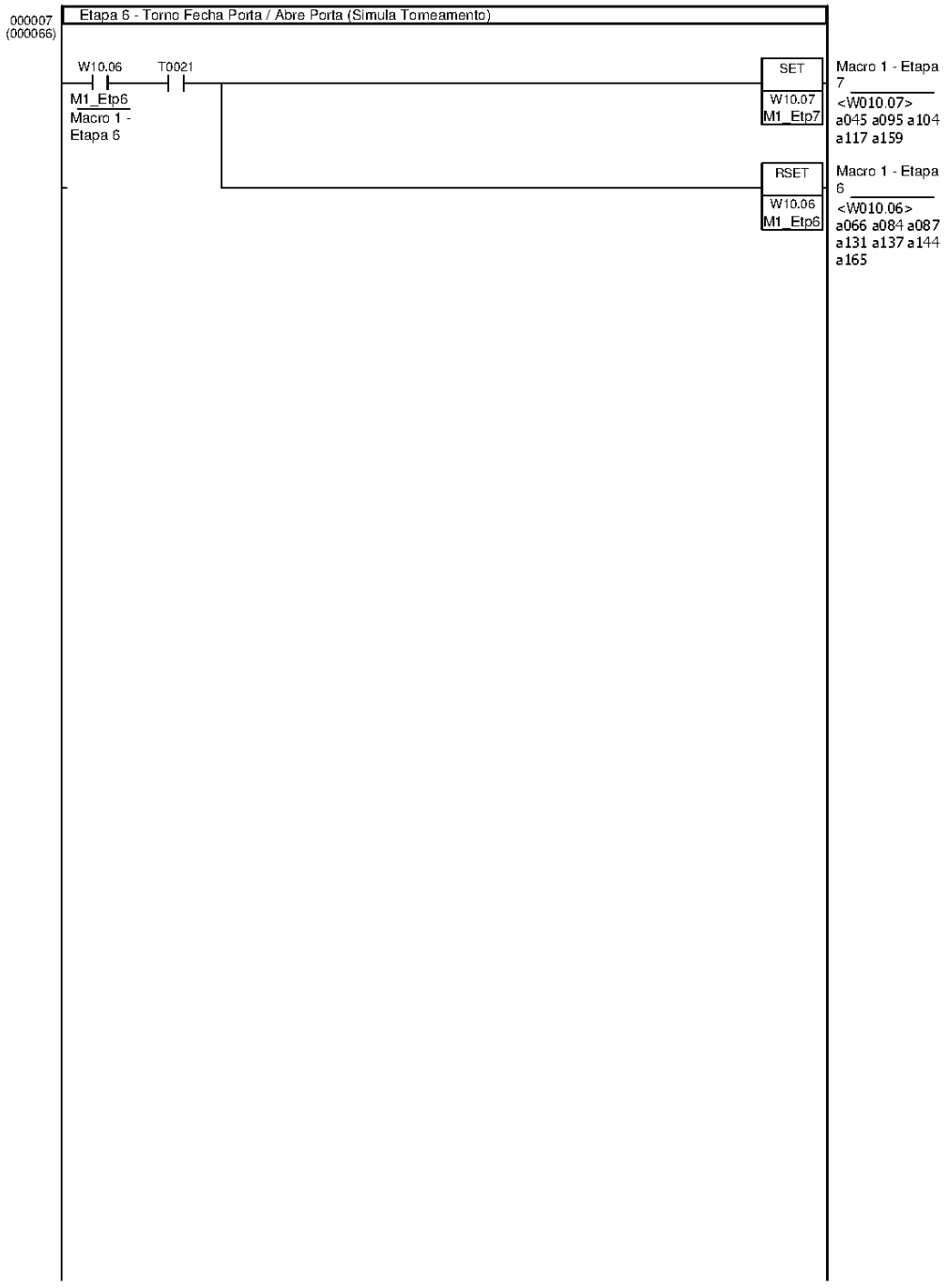
A seção Inicializa do programa do CLP, contém a movimentação dos sinais digitais do controlador, para memórias auxiliares mapeadas no protocolo OPC. A seção de inicialização é essencial para comunicação do sistema.



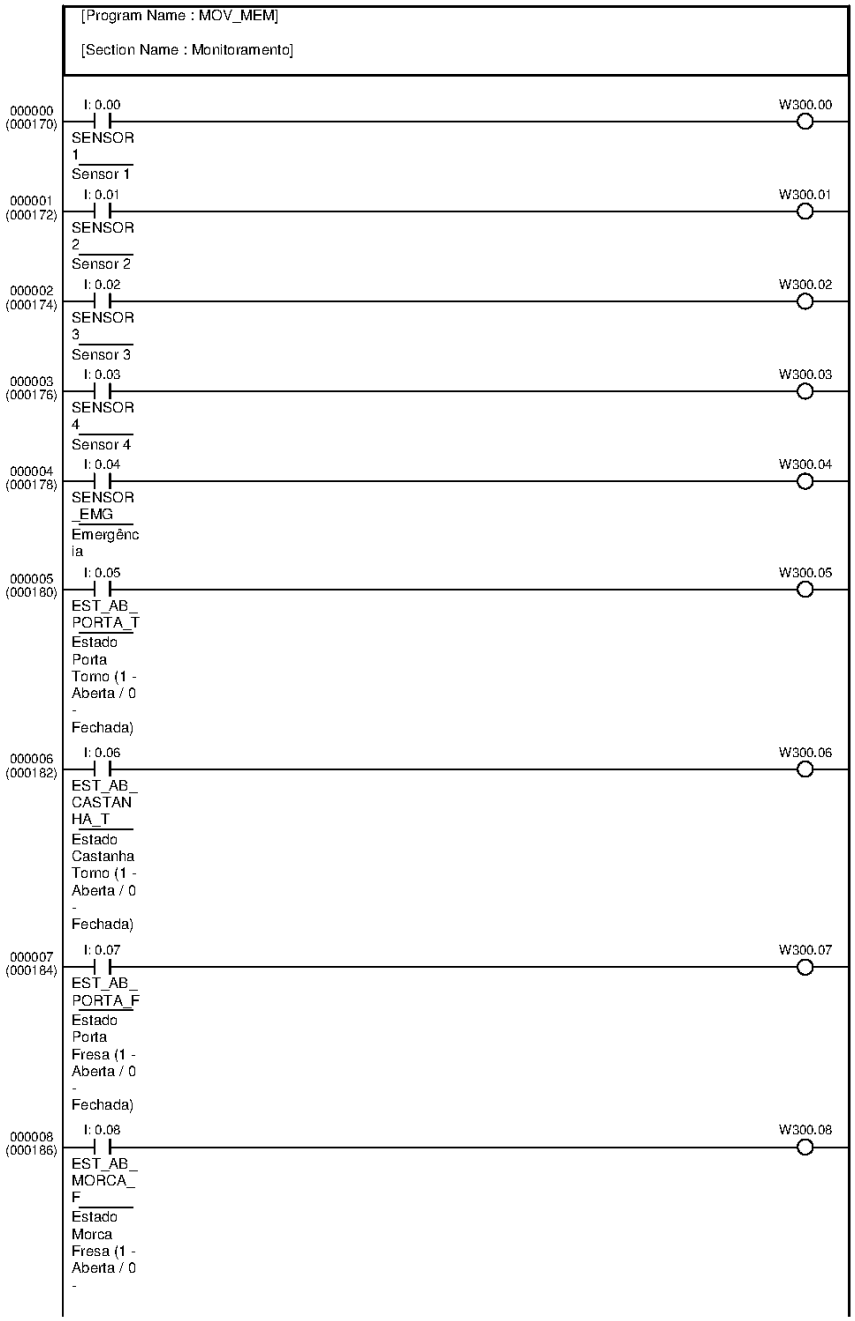
A seção Macro do programa do CLP, contém a lógica do GRAFCET abordada no item 3.1 do presente trabalho. É nessa seção do programa que a lógica de controle é processada.

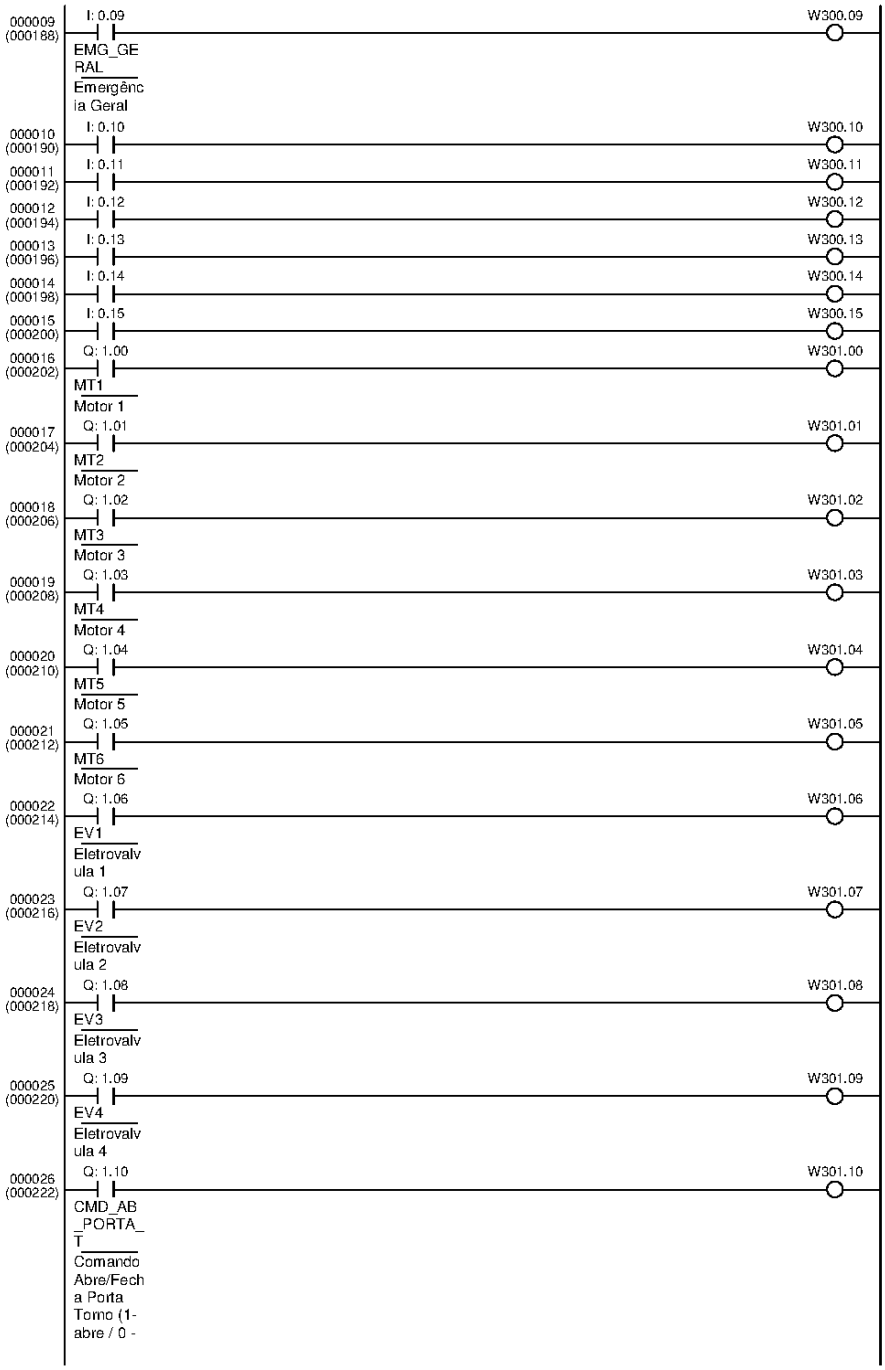


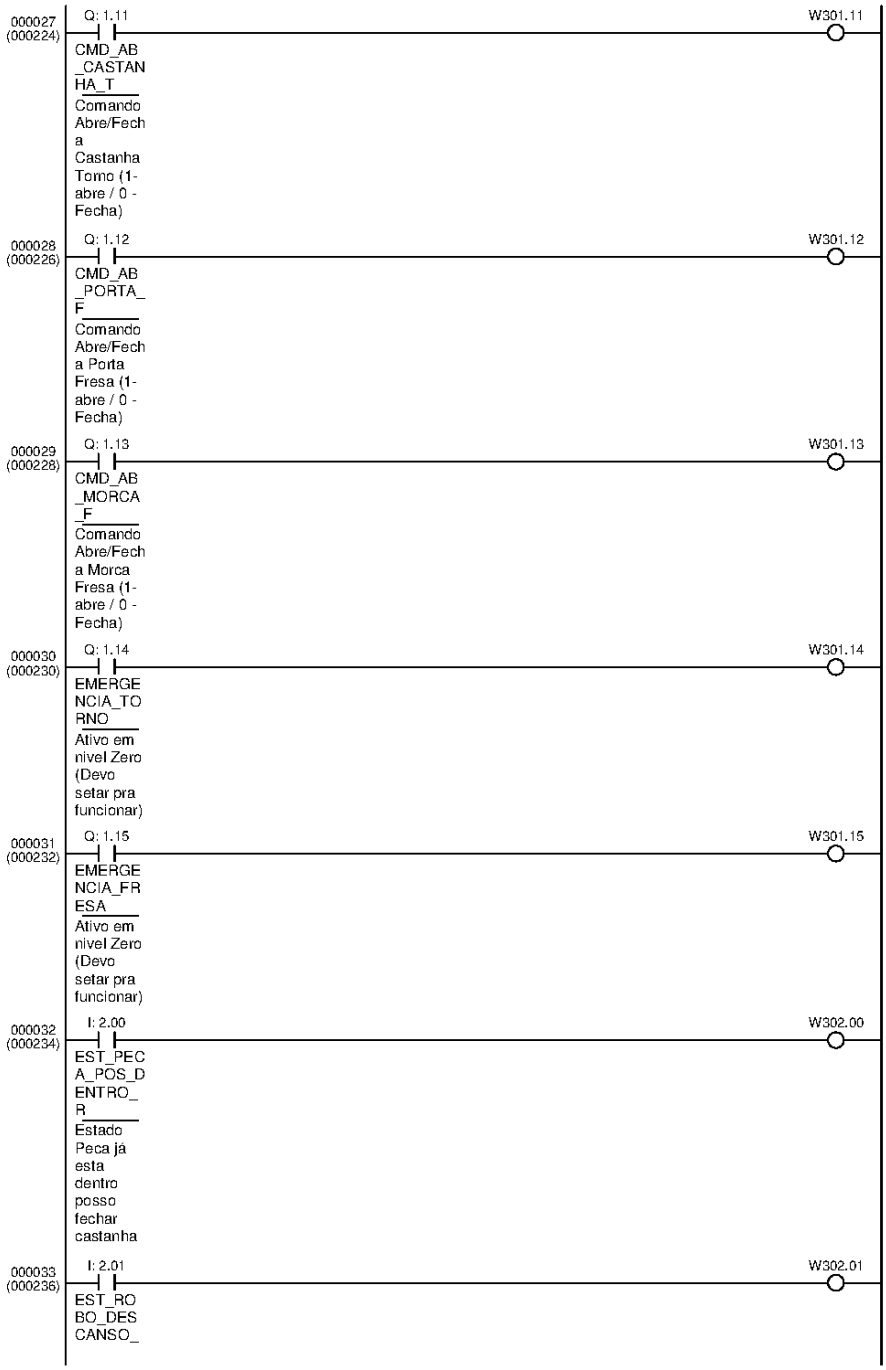

000003
(000049)
Etapa 2 - Abre Porta Torno
W10.02
M1_Etp2
Macro 1 - Etapa2
T0020
Timer
SET
W10.03
M1_Etp3
Macro 1 - Etapa 3
<W010.03>
a054 a102 a150
RSET
W10.02
M1_Etp2
Macro 1 - Etapa2
<W010.02>
a049 a091 a123 a129 a143 a163
RSET
W10.07
M1_Etp7
Macro 1 - Etapa 7
<W010.07>
a045 a095 a104 a117 a159
000004
(000054)
Etapa 3 - Posiciona Peça - ROBÔ
W10.03
M1_Etp3
Macro 1 - Etapa 3
I: 0.01
SENSOR 2
Sensor 2
SET
W10.04
M1_Etp4
Macro 1 - Etapa 4
<W010.04>
a058 a080 a130 a167
RSET
W10.03
M1_Etp3
Macro 1 - Etapa 3
<W010.03>
a054 a102 a150
000005
(000058)
Etapa 4 - Fecha Castanha Torno
W10.04
M1_Etp4
Macro 1 - Etapa 4
T0022
SET
W10.05
M1_Etp5
Macro 1 - Etapa 5
<W010.05>
a062 a103 a110 a116 a153 a156
RSET
W10.04
M1_Etp4
Macro 1 - Etapa 4
<W010.04>
a058 a080 a130 a167
000006
(000062)
Etapa 5 - Robô Posição de Descanso
W10.05
M1_Etp5
Macro 1 - Etapa 5
I: 0.03
SENSOR 4
Sensor 4
SET
W10.06
M1_Etp6
Macro 1 - Etapa 6
<W010.06>
a066 a084 a087 a131 a137 a144 a165
RSET
W10.05
M1_Etp5
Macro 1 - Etapa 5
<W010.05>
a062 a103 a110 a116 a153 a156

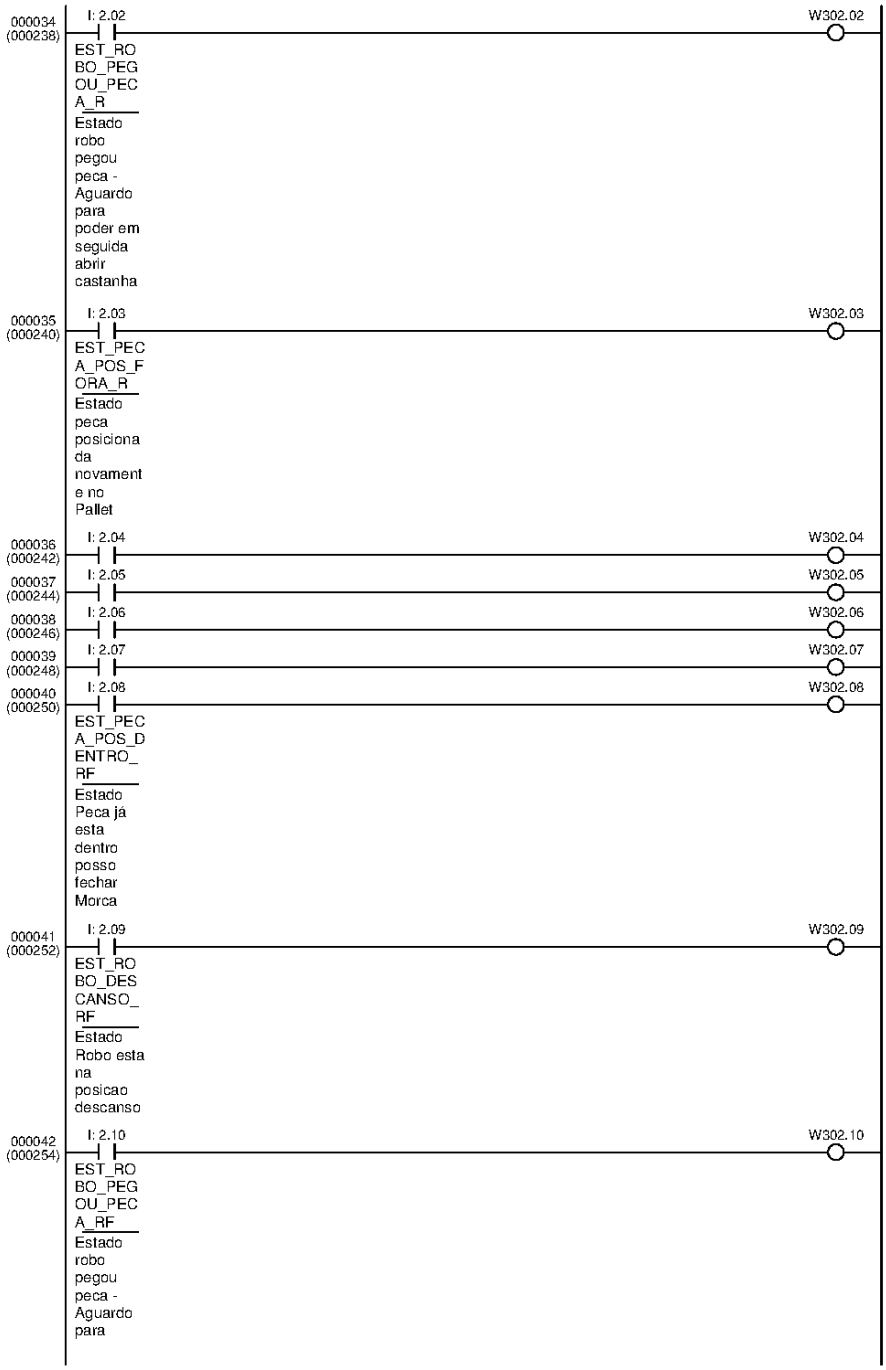

000007
(000066)
Etapa 6 - Torno Fecha Porta / Abre Porta (Simula Tomeamento)
W10.06
M1_Etp6
Macro 1 -
Etapa 6
T0021
SET
W10.07
M1_Etp7
Macro 1 - Etapa 7
<W010.07>
a045 a095 a104 a117 a159
RSET
W10.06
M1_Etp6
Macro 1 - Etapa 6
<W010.06>
a066 a084 a087 a131 a137 a144 a165

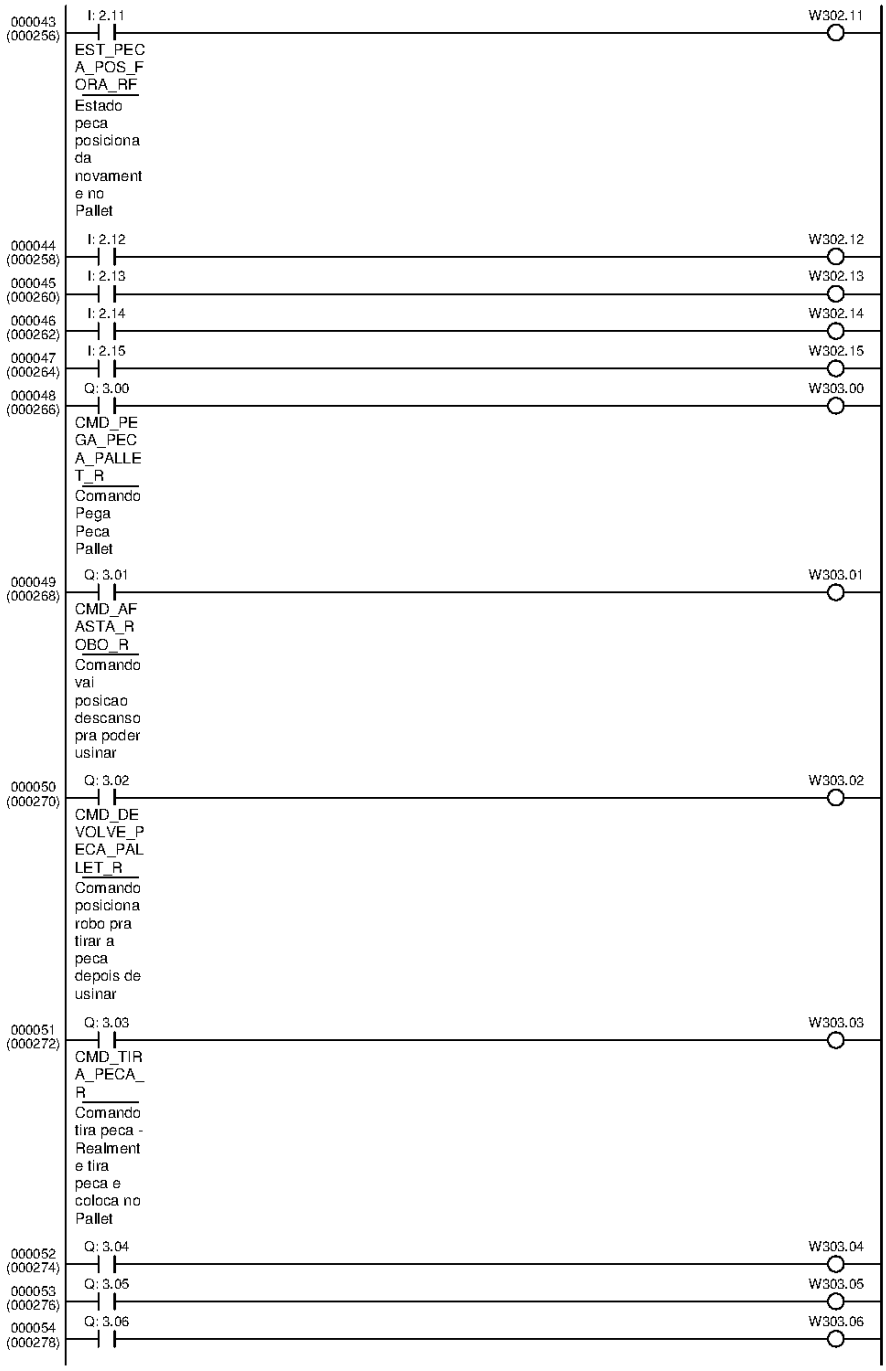
A seção Monitoramento do programa do CLP contém a movimentação dos sinais digitais do controlador, para memórias auxiliares mapeadas no protocolo OPC. A diferença em relação a seção de inicialização, esta seção é utilizada pela a página de monitoramento do sistema, que acessa os dados presentes na área de memória presente nessa seção.

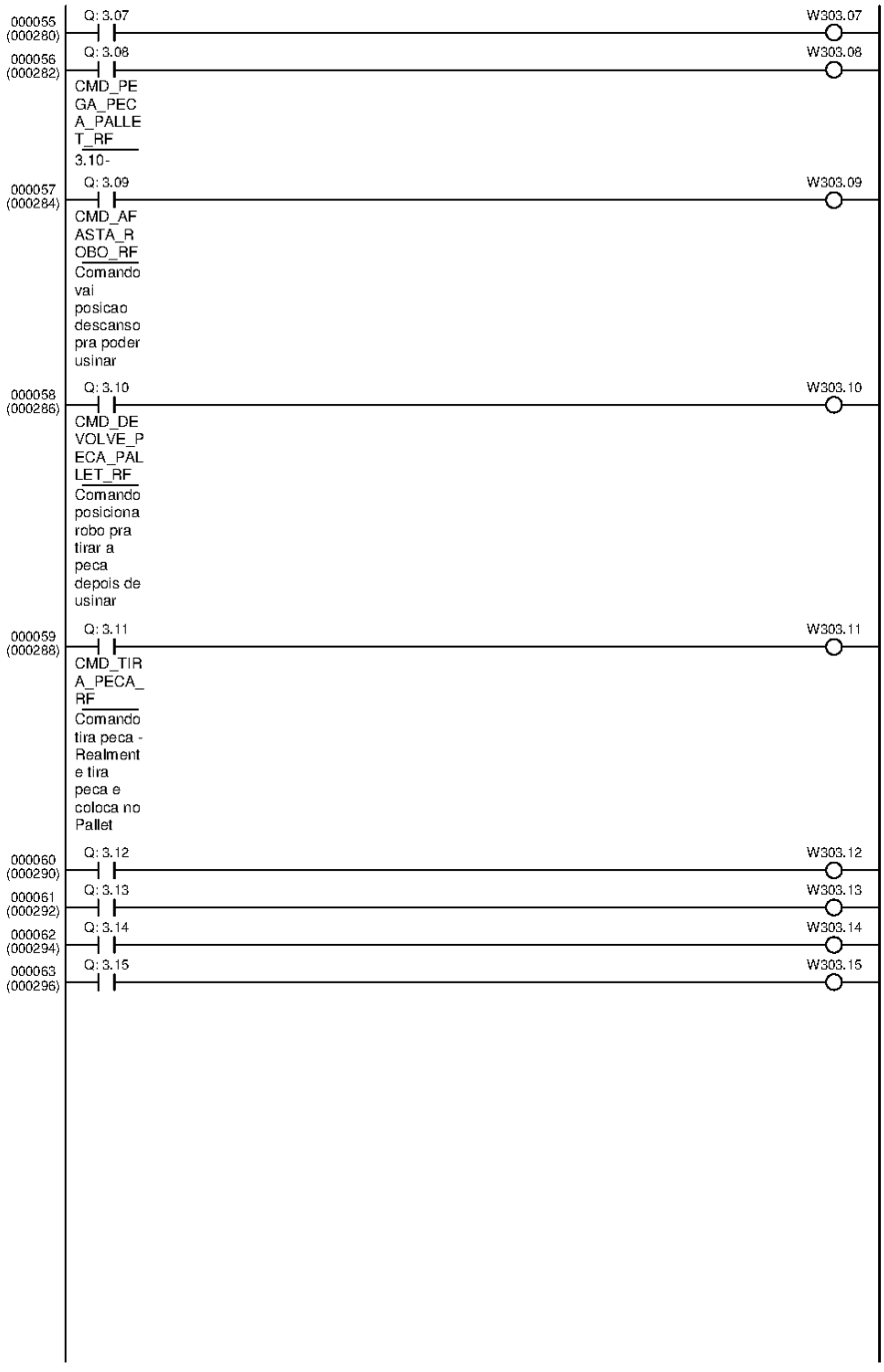



000009
(000188)
I: 0.09
EMG_GERAL
Emergência Geral
W300.09
000010
(000190)
I: 0.10
W300.10
000011
(000192)
I: 0.11
W300.11
000012
(000194)
I: 0.12
W300.12
000013
(000196)
I: 0.13
W300.13
000014
(000198)
I: 0.14
W300.14
000015
(000200)
I: 0.15
W300.15
000016
(000202)
Q: 1.00
MT1
Motor 1
W301.00
000017
(000204)
Q: 1.01
MT2
Motor 2
W301.01
000018
(000206)
Q: 1.02
MT3
Motor 3
W301.02
000019
(000208)
Q: 1.03
MT4
Motor 4
W301.03
000020
(000210)
Q: 1.04
MT5
Motor 5
W301.04
000021
(000212)
Q: 1.05
MT6
Motor 6
W301.05
000022
(000214)
Q: 1.06
EV1
Eletrovalvula 1
W301.06
000023
(000216)
Q: 1.07
EV2
Eletrovalvula 2
W301.07
000024
(000218)
Q: 1.08
EV3
Eletrovalvula 3
W301.08
000025
(000220)
Q: 1.09
EV4
Eletrovalvula 4
W301.09
000026
(000222)
Q: 1.10
CMD_AB_PORTA_T
Comando Abre/Fecha Porta Torno (1- abre / 0 -
W301.10

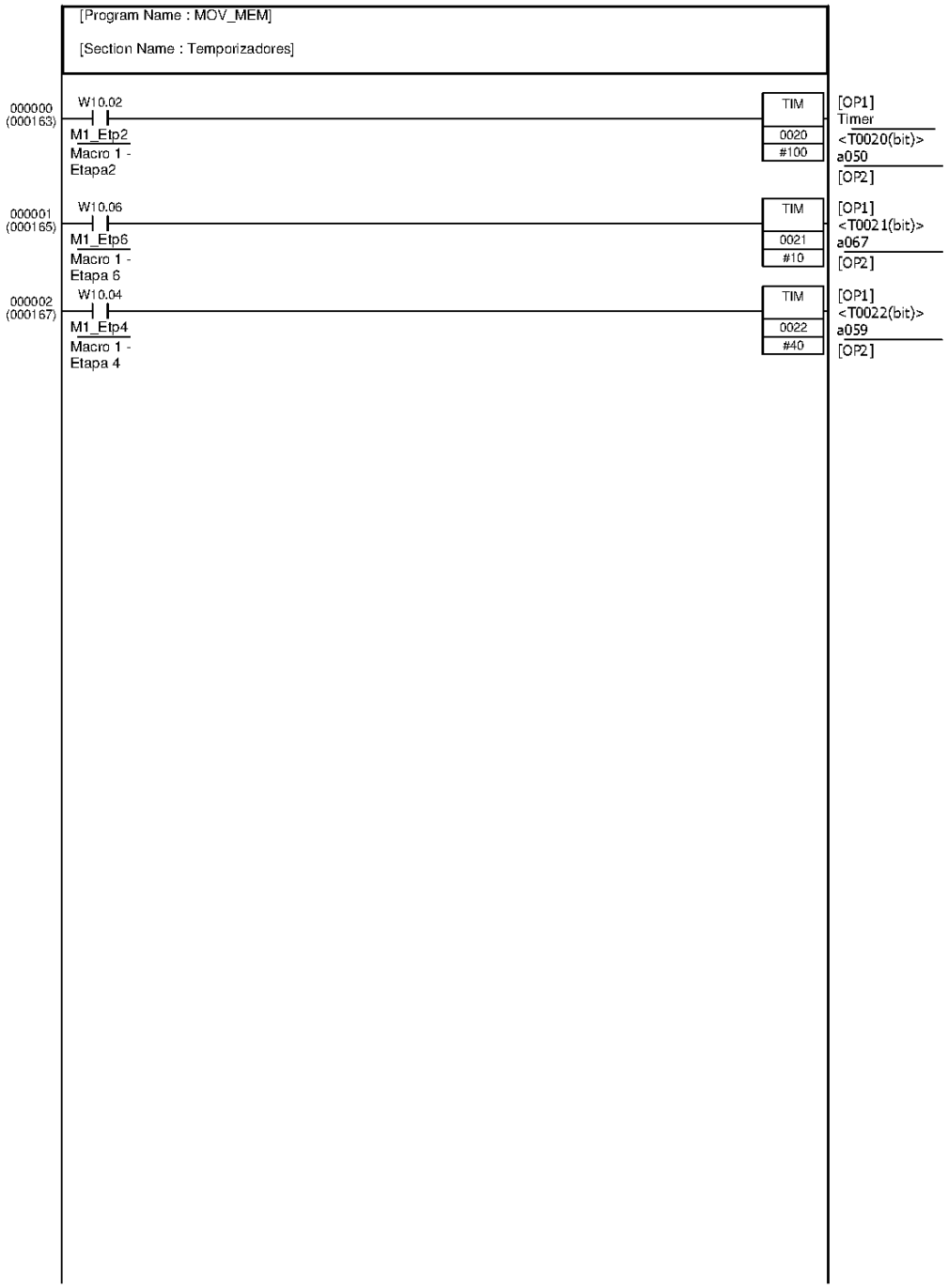

000027
(000224)
Q: 1.11
CMD_AB_CASTANHA_T
Comando Abre/Fecha Castanha Tomo (1- abre / 0 - Fecha)
W301.11
000028
(000226)
Q: 1.12
CMD_AB_PORTA_F
Comando Abre/Fecha Porta Fresa (1- abre / 0 - Fecha)
W301.12
000029
(000228)
Q: 1.13
CMD_AB_MORCA_F
Comando Abre/Fecha Morca Fresa (1- abre / 0 - Fecha)
W301.13
000030
(000230)
Q: 1.14
EMERGENCIA_TORNO
Ativo em nivel Zero (Devo setar pra funcionar)
W301.14
000031
(000232)
Q: 1.15
EMERGENCIA_FRESA
Ativo em nivel Zero (Devo setar pra funcionar)
W301.15
000032
(000234)
I: 2.00
EST_PECA_POS_DENTRO_R
Estado Peca já esta dentro posso fechar castanha
W302.00
000033
(000236)
I: 2.01
EST_ROBO_DESCANSO_
W302.01


000034
(000238)
I: 2.02
EST_ROBO_PEGOU_PECA_R
Estado robo pegou peca - Aguardo para poder em seguida abrir castanha
W302.02
000035
(000240)
I: 2.03
EST_PECA_POS_FORA_R
Estado peca posiciona da novament e no Pallet
W302.03
000036
(000242)
I: 2.04
W302.04
000037
(000244)
I: 2.05
W302.05
000038
(000246)
I: 2.06
W302.06
000039
(000248)
I: 2.07
W302.07
000040
(000250)
I: 2.08
EST_PECA_POS_DENTRO_RF
Estado Peca já esta dentro posso fechar Morca
W302.08
000041
(000252)
I: 2.09
EST_ROBO_DESCANSO_RF
Estado Robo esta na posicao descanso
W302.09
000042
(000254)
I: 2.10
EST_ROBO_PEGOU_PECA_RF
Estado robo pegou peca - Aguardo para
W302.10


000043
(000256)
I:2.11
EST_PECA_POS_FORA_RF
Estado peca posicionada novamente no Pallet
W302.11
000044
(000258)
I:2.12
W302.12
000045
(000260)
I:2.13
W302.13
000046
(000262)
I:2.14
W302.14
000047
(000264)
I:2.15
W302.15
000048
(000266)
Q:3.00
CMD_PEGA_PECA_PALLET_R
Comando Pega Peca Pallet
W303.00
000049
(000268)
Q:3.01
CMD_AFASTA_ROBO_R
Comando vai posicao descanso pra poder usinar
W303.01
000050
(000270)
Q:3.02
CMD_DEVOLVE_PECA_PALLET_R
Comando posiciona robo pra tirar a peca depois de usinar
W303.02
000051
(000272)
Q:3.03
CMD_TIRA_PECA_R
Comando tira peca - Realmente tira peca e coloca no Pallet
W303.03
000052
(000274)
Q:3.04
W303.04
000053
(000276)
Q:3.05
W303.05
000054
(000278)
Q:3.06
W303.06


000055
(000280)
Q: 3.07
W303.07
000056
(000282)
Q: 3.08
W303.08
CMD_PEGA_PECA_PALLET_RF
3.10 -
000057
(000284)
Q: 3.09
W303.09
CMD_AFASTA_ROBO_RF
Comando vai posicao descanso pra poder usinar
000058
(000286)
Q: 3.10
W303.10
CMD_DEVOLVE_PECA_PALLET_RF
Comando posiciona robo pra tirar a peca depois de usinar
000059
(000288)
Q: 3.11
W303.11
CMD_TIRA_PECA_RF
Comando tira peca - Realmente tira peca e coloca no Pallet
000060
(000290)
Q: 3.12
W303.12
000061
(000292)
Q: 3.13
W303.13
000062
(000294)
Q: 3.14
W303.14
000063
(000296)
Q: 3.15
W303.15

A seção de temporizadores contém os totalizados do processo e facilita na visualização das condições de temporização do processo.



APÊNDICE B – PROGRAMAÇÃO PHP

O arquivo index.php apresenta a estrutura da página. O arquivo contém o cabeçalho do sistema, constituído pelo menu de navegação, menu lateral e divisão de conteúdo, além de solicitar a página index_ajax.php para apresentar o conteúdo.

A página principal possui a tecnologia Ajax, responsável pela solicitação dinâmica de dados ao servidor de dados.

Arquivo index.php

```
<html><title>TCC UTFPR 2013 - Luis e Thomas</title>
    <META content="text/html; charset=iso-8859-1" http-equiv=content-
type>
    <LINK rel=stylesheet type=text/css href="./css/estilo.css">
    <LINK rel=stylesheet type=text/css
href="./css/posicionamento.css">
    <script type="text/javascript">


    var http = false;

    if(window.XMLHttpRequest){
        http = new XMLHttpRequest();
        }else {
        http = new ActiveXObject("Microsoft.XMLHTTP")
        }

    function Modbus() {
        http.abort();
        http.onreadystatechange=function() {
        if(http.readyState == 4 && http.status == 200) {
            document.getElementById('conteudo').innerHTML =
http.responseText;

        }
    }
    http.open("GET", "index_ajax.php", true)
    http.send();
    }
    </script>

    <!-- Função de abrir popup-->
        <script type="text/javascript">
            function abrir (URL){

window.open(URL,"janela1","width=220,height=170,location=no,resizable=
no,status=no,scrollbars=NO,title=no")
            }
        </script>
    <body>

    <!-- Definição Layout -->
    <div id="fundo">
    <div id="menu">
```

```
            <ul>
            <li class="current_page_item"><a
href="#">Principal</a></li>
            <li><a href="monitor.php">Monitor</a></li>
            <li><a
href="configuracoes.php?funcao=1">Configurações</a></li>
            <li><a href="http://www.utfpr.edu.br">Sair</a></li>

        </ul>
    </div>
    <div id=conteudo>

<p align=center>Carregando...</p>

</div>
</div>
    <script type="text/javascript">
        setInterval ("Modbus()", 3500);
    </script>
</html>
```

O arquivo index_ajax.php apresenta o conteúdo que aparece na página. O conteúdo é composto pela figura que compõe o sinótico do FMS, figuras que animam os motores, eletroválvulas e sensores, além de todos os botões de comando de equipamentos.

Arquivo index_ajax.php

```
    <LINK rel=stylesheet type=text/css href="./css/estilo.css">
    <LINK rel=stylesheet type=text/css
href="./css/posicionamento.css">

<?php


    // Inclusão da biblioteca e parametros
    include("./config/config_param.php");
    include("funcoes.php");

    // Inst□ia Socket - Conexão com o Servidor Modbus TCP
    $con = @fsockopen($serverip,$porta, $errono, $errostr, 1);

    // Leitura de 4 Registros, apartir do registro 100
    $entrada    = LeMultRegistro(100,4,$con);

$navegacao = $_GET['navegacao'];

    ?>

            <div id="lateral">
    <?php
```

```
        echo "<form
action=\"javascript:abrir('comandos.php?equipamento=partida');\">";
        echo "<input style=\"background:white;width:180px;\"
        type=\"submit\"
OnMouseOver=\"javascript:this.style.background='#EEEEEE'\"
onMouseOut=\"javascript:this.style.background='#FFFFFF'\"
value=\"Partida em Autom&aacute;tico\"></form>";


        echo "<form
action=\"javascript:abrir('comandos.php?equipamento=auto');\">";
        echo "<input style=\"background:white;width:180px;\"
        type=\"submit\"
OnMouseOver=\"javascript:this.style.background='#EEEEEE'\"
onMouseOut=\"javascript:this.style.background='#FFFFFF'\"
value=\"Opera&ccedil;&atilde;o\"></form>";


    ?>
    <table style="font-size:120%;" cellPadding=0 cellspacing=1>
       <tbody>
       <tr style="background:#EEEEEE; height:30px;" align=center>
       <td width="205px" style="border:1px solid black;"><?php
if($auto_man=$entrada[3] == 0){ echo "<font color=red><b>Em
Manual</b>"; }else{ echo "<font color=#008800><b>Em
Autom&aacute;tico</b>";}?></td></tr>
        <tr style="background:#EEEEEE; height:30px;" align=center><td
width="205px" style="border:1px solid black;"><?php
if(ValorBit($entrada[0],4)==1) { echo "<font
color=red><b>Emerg&ecircncia Atuada</b>"; }else{ echo "<font
color=#008800><b>Opera&ccedil;&atilde;o Normal</b>";}?></td>

       </tr>
</tbody></table>

    <?php
        if(!$con) {
        echo "<br><br><br>Gateway fora do ar.
        <br>Para alterar as configura&ccedil;&otilde;es clique <a
href=\"configuracoes.php\">aqui</a>";
        }

        ?>
    </div>
    <?php
if(!isset($navegacao)) {
    echo "<img src=\"./figuras/FMS_v4.png\">";
        /*****************************************************/
        #           TRATATIVA DE MOTORES                      #
        /*****************************************************/
        if(ValorBit($entrada[1],1)==1) { # Motor 2 (Próximo do Torno)
Registro 101, Bit 1
            echo "<img class=\"ch_i00\"
src=\"./Figuras/anima/motores/m1_ligado.png\">";
        }
        else {
            echo "<img class=\"ch_i00\"
src=\"./Figuras/anima/motores/m1_desligado.png\">";
        }
        if(ValorBit($entrada[1],2)==1) { # Motor 3 (Próximo ao Quadro)
Registro 101, Bit 2
```

```
            echo "<img class=\"ch_i01\"
src=\"./Figuras/anima/motores/m2_ligado.png\">";
        }
        else {
            echo "<img class=\"ch_i01\"
src=\"./Figuras/anima/motores/m2_desligado.png\">";
        }
        if(ValorBit($entrada[1],3)==1) { # Motor 4 (Sistema de Visão
Registro 101, Bit 3
            echo "<img class=\"ch_i02\"
src=\"./Figuras/anima/motores/m3_ligado.png\">";
        }
        else {
            echo "<img class=\"ch_i02\"
src=\"./Figuras/anima/motores/m3_desligado.png\">";
        }
        if(ValorBit($entrada[1],0)==1) { # Motor 1 (Próximo do
armazem) Registro 101, Bit 0
            echo "<img class=\"ch_i03\"
src=\"./Figuras/anima/motores/m4_ligado.png\">";
        }
        else {
            echo "<img class=\"ch_i03\"
src=\"./Figuras/anima/motores/m4_desligado.png\">";
        }
        /*****************************************************/
        #           TRATATIVA DE SENSORES                     #
        /*****************************************************/

        if(ValorBit($entrada[0],0)==1) {
            echo "<img class=\"sensor1\"
src=\"./Figuras/anima/verde.PNG\">";
        }
        else {
            echo "<img class=\"sensor1\"
src=\"./Figuras/anima/vermelho.png\">";
        }

        if(ValorBit($entrada[0],1)==1) {
            echo "<img class=\"sensor2\"
src=\"./Figuras/anima/verde.PNG\">";
        }
        else {
            echo "<img class=\"sensor2\"
src=\"./Figuras/anima/vermelho.PNG\">";
        }

        if(ValorBit($entrada[0],2)==1) {
            echo "<img class=\"sensor3\"
src=\"./Figuras/anima/verde.PNG\">";
        }
        else {
            echo "<img class=\"sensor3\"
src=\"./Figuras/anima/vermelho.PNG\">";
        }
        if(ValorBit($entrada[0],3)==1) {
            echo "<img class=\"sensor4\"
src=\"./Figuras/anima/verde.PNG\">";
        }
        else {
```

```
            echo "<img class=\"sensor4\"
src=\"./Figuras/anima/vermelho.PNG\">";
        }

        /*****************************************************/
        #           TRATATIVA DE POPUPS                       #
        /*****************************************************/
        echo "<form
action=\"javascript:abrir('comandos.php?equipamento=M1');\">";
        echo "<input class=\"comando\" style=\"position:relative;top:-
280px;left:597px;\" type=\"submit\"
OnMouseOver=\"javascript:this.style.background='#EEEEEE'\"
onMouseOut=\"javascript:this.style.background='#FFFFFF'\"
value=\"M1\"></form>";

        echo "<form
action=\"javascript:abrir('comandos.php?equipamento=M2');\">";
        echo "<input class=\"comando\"
        style=\"position:relative;top:-435px;left:438px;\"
        type=\"submit\"
OnMouseOver=\"javascript:this.style.background='#EEEEEE'\"
onMouseOut=\"javascript:this.style.background='#FFFFFF'\"
value=\"M2\"></form>";

        echo "<form
action=\"javascript:abrir('comandos.php?equipamento=M3');\">";
        echo "<input class=\"comando\"
        style=\"position:relative;top:-420px;left:280px;\"
        type=\"submit\"
OnMouseOver=\"javascript:this.style.background='#EEEEEE'\"
onMouseOut=\"javascript:this.style.background='#FFFFFF'\"
value=\"M3\"></form>";

        echo "<form
action=\"javascript:abrir('comandos.php?equipamento=M4');\">";
        echo "<input class=\"comando\"
        style=\"position:relative;top:-336px;left:456px;\"
        type=\"submit\"
OnMouseOver=\"javascript:this.style.background='#EEEEEE'\"
onMouseOut=\"javascript:this.style.background='#FFFFFF'\"
value=\"M4\"></form>";

        /*****************************************************/
        #           TRATATIVA DE ELETRO VALVULAS              #
        /*****************************************************/
        if(ValorBit($entrada[2],0)==1) { // Eletrovalvula 1, Registro
102 - Bit 0
            echo "<img class=\"ev1\"
src=\"./Figuras/anima/EV/EV1_atuada.PNG\">";
        }
        else {
            echo "<img class=\"ev1\"
src=\"./Figuras/anima/EV/EV1_desatuada.png\">";
        }
        if(ValorBit($entrada[2],1)==1) { // Eletrovalvula 1, Registro
102 - Bit 1
            echo "<img class=\"ev2\"
src=\"./Figuras/anima/EV/EV2_atuada.PNG\">";
        }
        else {
```

```
            echo "<img class=\"ev2\"
src=\"./Figuras/anima/EV/EV2_desatuada.png\">";
        }
        if(ValorBit($entrada[2],2)==1) { // Eletrovalvula 1, Registro
102 - Bit 2
            echo "<img class=\"ev3\"
src=\"./Figuras/anima/EV/EV3_atuada.PNG\">";
        }
        else {
            echo "<img class=\"ev3\"
src=\"./Figuras/anima/EV/EV3_desatuada.png\">";
        }
        if(ValorBit($entrada[2],3)==1) { // Eletrovalvula 1, Registro
102 - Bit 3
            echo "<img class=\"ev4\"
src=\"./Figuras/anima/EV/EV4_atuada.PNG\">";
        }
        else {
            echo "<img class=\"ev4\"
src=\"./Figuras/anima/EV/EV4_desatuada.png\">";
        }

    }

?>
```

O arquivo envia_cmd.php envia os comandos ao gateway do sistema, conforme os dados recebidos da popup de comando, nomeada de comandos.php.

Arquivo envia_cmd.php

```
<?php

    include("./config/config_param.php");
    include("funcoes.php");

    $condicao = $_GET['valor'];          // Comando a ser efetuado
(Ligar/Desligar/Auto/Partir)
    $equipamento = $_GET['equipamento'];// Equipamento a ser comandado

    // Definição de Bits e Registros
    // De acordo com o Popup
        if($equipamento == "M1") {           // Comandos no Motor M1
            $registro = 200;
                if($condicao== "Ligar") {
                    $bit = 0;                // Registro 200 bit 0,
Ligar M1
                }
                else {
                    $bit = 4;                // Registro 200 bit 4,
Desligar M1
                }
        }
        else if($equipamento == "M2") {      // Comandos no Motor M2
            $registro = 200;
                if($condicao== "Ligar") {
```

```
                    $bit = 1;                   // Registro 200 bit 1,
Ligar M2
                }
                else {
                    $bit = 5;                   // Registro 200 bit 5,
Desligar M2
                }
        }
        else if($equipamento == "M3") {         // Comandos no Motor M3
            $registro = 200;
                if($condicao== "Ligar") {
                $bit = 2;                       // Registro 200 bit 1,
Ligar M3
                }
                else {
                    $bit=6;                     // Registro 200 bit 5,
Desligar M3
                }
        }
        else if($equipamento == "M4") {         // Comandos no Motor M4
            $registro = 200;
                if($condicao== "Ligar") {
                    $bit = 3;                   // Registro 200 bit 3,
Ligar M4
                }
                else {
                    $bit=7;                     // Registro 200 bit 7,
Desligar M4
                }
        }
        else if($equipamento == "partida") { // Partida do sistema em
Automпco
            $registro = 201;
            if($condicao== "Partir") {
                $bit = 0;
            }
            else {
                $bit=1;
            }
        }

        else if($equipamento == "auto") {  // Chave de comuta磯
Automпco - Manual
            $registro = 201;
            $bit = 2;
        }

    $con = @fsockopen ($serverip,$porta, $errono, $errostr, 1);

    if($equipamento != "auto") {
        SetaBit($registro,$bit,$con);
        sleep(1);
        ZeraBit($registro,$bit,$con);
    } else {
        if($condicao=="Automatico") {
            SetaBit($registro,$bit,$con);
        } else {
            ZeraBit($registro,$bit,$con);
        }
    }
?>
```

O arquivo comandos.php exibe os dados de todas as *popups* de comando do processo. O arquivo é responsável por enviar os parâmetros que definem qual o equipamento vai ser comandado e qual a função do comando a ser efetuado.

Arquivo comandos.php

```php
<?php
        // Inclusão da biblioteca e par□tros
        include("./config/config_param.php");
        include("funcoes.php");

        // Word 100 do CLP = Registro 200 Modbus
        $equipamento = $_GET['equipamento'];

        if($equipamento == "M1") {
            // Registros para leitura de horas e acionamentos do motor
            $registro_horas = 400;
            $registro_acc = 410;

            // Texto para impressão no popup
            $titulo = "Motor M1";
            $txt_liga = "Ligar";
            $txt_desliga = "Desligar";
        }
        else if($equipamento == "M2") {
            // Registros para leitura de horas e acionamentos do motor
            $registro_horas = 401;
            $registro_acc = 411;

            // Texto para impressão no popup
            $titulo = "Motor M2";
            $txt_liga = "Ligar";
            $txt_desliga = "Desligar";
        }
        else if($equipamento == "M3") {
            // Registros para leitura de horas e acionamentos do motor
            $registro_horas = 402;
            $registro_acc = 412;

            // Texto para impressão no popup
            $titulo = "Motor M3";
            $txt_liga = "Ligar";
            $txt_desliga = "Desligar";
        }
        else if($equipamento == "M4") {
            // Registros para leitura de horas e acionamentos do motor
            $registro_horas = 403;
            $registro_acc = 413;

            // Texto para impressão no popup
            $titulo = "Motor M4";
            $txt_liga = "Ligar";
            $txt_desliga = "Desligar";
        }
        else if($equipamento == "partida") {
            $titulo = "Partida";
```

```
            $txt_liga = "Partir";
            $txt_desliga = "Parar";
        }

        else if($equipamento == "auto") {
            $titulo = "Opera&ccedil;&atilde;o";
            $txt_liga = "Automatico";
            $txt_desliga = "Manual";
        }


?>

<LINK rel=stylesheet type=text/css href="/css/estilo.css">
<TABLE style="color:white; font-family:verdana; font-size: 80%;">
<th style="font-size: 130%;"colspan=2><?php echo $titulo; ?></th>
   <tr>
        <td>
            <form action=envia_cmd.php>
                <input type=hidden name=equipamento value="<?php echo
$equipamento; ?>">
                <input style="width:85px; background-color:white;"
type="submit" name=valor value="<?php echo $txt_liga; ?>">
            </form>
        </td>
        <td>
            <form action=envia_cmd.php>
                <input type=hidden name=equipamento value="<?php echo
$equipamento; ?>">
                <input style="width:85px; background-color:white;"
type="submit" name=valor value="<?php echo $txt_desliga; ?>">
            </form>
        </td>
    </tr>
    <?php if ($equipamento=="M1" or $equipamento=="M2" or
$equipamento=="M3" or $equipamento=="M4") {
        $con = @fsockopen ($serverip,$porta, $errono, $errostr, 1);
        $contador_h = LeMultRegistro($registro_horas,1,$con);
        if ($contador_h[0] <60) {
            $unidade = "s";
        } else {
            $contador_h[0]=$contador_h[0]/60;
            $unidade = "min";
        }
        $contador_acc    = LeMultRegistro($registro_acc,1,$con);

    ?>
    <tr><td>Totalizador:</td><td><input style="width:80px;" type=name
name=totalhoras value="<?php echo $contador_h[0].$unidade;
?>"></td></tr>
    <tr><td>Acionamentos:</td><td><input style="width:80px;" type=name
name=acionamentos value="<?php echo $contador_acc[0]; ?>"></td></tr>
    <?php } ?>

    </table>
```

O arquivo funcoes.php possui o núcleo de funções do sistema. O arquivo é responsável pelas funções de leitura e escrita de registros no Modbus TCP, verificação do valor do bit em uma word, funções de formatação de valores com casa decimal, funções de setar e zerar bits.

Arquivo funcoes.php

```
<?php

    // Função Leitura Registro Modbus TCP / Múltiplos Registros
    // Desenvolvida por Luis F. S. Alves e Thomas P. Duck
    function LeMultRegistro($registro,$nreg,$con) {

//  $con = @fsockopen
($GLOBALS['serverip'],$GLOBALS['porta'],$errono,$errostr, 2) or
die("Erro ao estabelecer conexão com o servidor Modbus TCP.");
        if($con)
        {
            $prefix=$registro/256;


$cabecalho=array(0,0,0,0,0,6,$GLOBALS['escravo'],3,$prefix,$registro,0
,$nreg);

            $tam_ar=count($cabecalho);
            $i=0;

            for($i=0;$i<$tam_ar;$i++)
            {
                $guarda=$guarda.(chr($cabecalho[$i]));
                $gd2=$gd2.ord($guarda[$i]);
            }

            $escreve=fwrite($con,$guarda);
            $frec=strlen($guarda);

            $ler = fread($con,350);
            $tam=strlen($ler);
            $i=0;

            for($i=0;$i<$tam;$i++)
                {
                $rcb=$rcb.ord($ler[$i]);
                }

            $tam_rcb=strlen($rcb);

            $c_col=0;
            $conta=9;

            for($conta=9;$conta<$tam-1;$conta=$conta+2)
            {
                $conta_r=$c_col;
                $c_col=++$c_col;
                $cont_cel=$registro+$conta_r;
                $b1=ord($ler[$conta]);
                $word=$b1;
                $b0=ord($ler[$conta+1]);
```

```
                $word=$word*256;
                $word=$word + $b0;
                if($conta_r==0) {
                    $leitura_words = array($word);
                }
                else {
                    $adiciona =
array_splice($leitura_words,$conta_r,0,$word);
                }
            }
            return $leitura_words;
        #    fclose($con);
        }
    }

        // Função Retorna Valor do Bit de Determinado Número Inteiro -
08/02/2011
    function ValorBit($inteiro,$bit) {
        $binario = decbin($inteiro);
        for($string=strlen($binario);$string<16;$string++){
            $binario='0'.$binario;
        }
        $localizador=($bit+1)*(-1);
        $valor_bit = substr($binario,$localizador,1);
        return $valor_bit;
    }

        function WriteRegistro($registro,$valor,$con) {

            // $con = @fsockopen
($GLOBALS['serverip'],$GLOBALS['porta'], $errono, $errostr, 5) or
die("Erro ao estabelecer conexão com o servidor Modbus TCP.");

            if($con)
            {
                $prefix=$registro/256;
                $prefix_val=$valor/256;


$cabecalho=array(0,0,0,0,0,9,1,16,$prefix,$registro,0,1,2,$prefix_val,
$valor);

                $tam_ar=count($cabecalho);
                $i=0;

                for($i=0;$i<$tam_ar;$i++)
                    {
                    $guarda=$guarda.(chr($cabecalho[$i]));
                    $gd2=$gd2.ord($guarda[$i]);
                    }

                $escreve=fwrite($con,$guarda);
                $frec=strlen($guarda);

                $ler = fread($con,355);
                $tam=strlen($ler);
                $i=0;

                for($i=0;$i<$tam;$i++)
                {
                    $rcb=$rcb.ord($ler[$i]);
```

```
                }
            }
        }

    function SetaBit($registro,$bit,$con){

        $valor_atual = LeMultRegistro($registro,1,$con);
        $multiplicador=1;
        $valor_atual = $valor_atual[0];

        for($conta_bit=0;$conta_bit<=$bit;$conta_bit++) {

        if($conta_bit!=0) $multiplicador=$multiplicador*2;

            if($bit==$conta_bit) {
                $valor_atual = $valor_atual | $multiplicador;
            }
        }
        WriteRegistro($registro,$valor_atual,$con);
    }

    function ZeraBit($registro,$bit,$con) {

        $valor_atual = LeMultRegistro($registro,1,$con);
        $valor_atual = $valor_atual[0];

        $multiplicador = 1;
        $const_16bits = 65535;

        for($conta_bit=0;$conta_bit<=$bit;$conta_bit++) {

            if($conta_bit!=0)   $multiplicador = $multiplicador * 2;
            $const_zera = $const_16bits - $multiplicador;

            if($bit==$conta_bit) {
                $valor_atual = $valor_atual & $const_zera;
            }
        }
        WriteRegistro($registro,$valor_atual,$con);
    }

    function formata($numero)
    {
        $separa=explode(".",$numero);
        if(strlen($separa[1])>2){
            $separa[1]=substr($separa[1],0,2);
        }
        else if(strlen($separa[1]==1)){
            $separa[1]=$separa[1]."0";
        }

        if(strlen($separa[1]!=0)){
            $resultado = $separa[0].".".$separa[1];
        }else {
            $resultado = $separa[0];
        }
        return($resultado);
    }

?>
```

O arquivo configuracoes.php possibilita o usuário alterar as configurações de acesso da página. A página lê e escreve no arquivo que define as configurações do sistema.

Arquivo configuracoes.php

```
<html><title>TCC UTFPR 2013 - Luis e Thomas</title>
<!-- Chama folhas de estilo -->
    <LINK rel=stylesheet type=text/css href="./css/estilo.css">
    <LINK rel=stylesheet type=text/css
href="./css/posicionamento.css">


<?php

    include("./config/config_param.php");
    include("funcoes.php");

    $con = @fsockopen($serverip,$porta, $errono, $errostr, 1);
    $entrada    = LeMultRegistro(100,3,$con);
    $saida      = LeMultRegistro(200,15,$con);

?>


    <!-- Função de abrir popup-->
        <script type="text/javascript">
            function abrir (URL){

window.open(URL,"janela1","width=220,height=170,location=no,resizable=
no,status=no,scrollbars=NO,title=no")
            }
        </script>
    <body>
    <!-- Definição Layout -->
    <div id="fundo">
    <div id="menu">
            <ul>
            <li><a href="index.php">Principal</a></li>
            <li><a href="monitor.php">Monitor</a></li>
            <li class="current_page_item"><a
href="configuracoes.php">Configurações</a></li>
            <li><a href="http://www.utfpr.edu.br">Sair</a></li>

        </ul>
    </div>
    <div id=conteudo>
        <div id="lateral">
        </div>
        <?php
            include("./config/config_param.php");
        if ($funcao==1 or !isset($funcao)) {
        ?>
            <form action="">
                <table><tr>
                <td>Servidor</td>
                <td>
```

```
                <input type="text" name="servidor" value="<?php echo
$serverip; ?>">
                </td>
                </tr><tr>
                <td>Porta
                </td>
                <td>
                    <input type="text" name="porta" value="<?php echo
$porta; ?>">
                </td>
                </tr>
                <tr>
                <td>Escravo</td><td>
                    <input type="text" name="escravo" value="<?php
echo $escravo; ?>"></td>
                </tr><tr>
                <td></td>
                <td>
                    <input type=submit value=Atualizar>
                </td>
                </table>
                <input type=hidden name=funcao value=2>
            </form>
    <?php
    }
    else if($funcao==2){

    $porta  =   $_GET['porta'];
    $serverip   = $_GET['servidor'];
    $escravo    = $_GET['escravo'];
    $t_atualiza = $_GET['tempo_atualiza'];

        $arquivo=fopen("./config/config_param.php","w");
        fwrite($arquivo,'<?php
//Porta
$porta = "'.$porta.'";
//Servidor
$serverip = "'.$serverip.'";
// Endereço do Escravo Modbus
$escravo = "'.$escravo.'";
// Tempo de atualização leitura registros
$tempo_reg = "'.$t_atualiza.'";
?>');

    echo "<b>Dados atualizados com sucesso:</b>";
    echo "<br>Servidor: ".$servidor;
    echo "<br>Porta: ".$porta;
    echo "<br>Escravo: ".$escravo;
    //echo "<br>Tempo entre requisições Ajax: ".$t_atualiza;
    echo "<br><a href=configuracoes.php>Voltar</a>";
    }
    ?>

</div>
</div>

</html>
```

O arquivo posicionamento.css é a folha de estilo que define o posicionamento relativo das figuras em tela.

Arquivo posicionamento.css

```
img.ch_i00 {
 position:relative;
 left:-320px;
 top: -361px;
 }

 img.ch_i01 {
 position:relative;
 left:-527px;
 top: -242px;
 }

 img.ch_i02 {
 position:relative;
 left:-397px;
 top: -111px;
 }

 img.ch_i03 {
 position:relative;
 left:414px;
 top: -282px;
 }

 img.ch_i04 {
 position:relative;
 left:-211px;
 top: 146px;
 }

 img.ch_i05 {
 position:relative;
 left:-217px;
 top: 146px;
 }

 img.ch_i06 {
 position:relative;
 left:-223px;
 top: 146px;
 }

 img.ch_i07 {
 position:relative;
 left:-229px;
 top: 146px;
 }

 img.sensor1 {
 position:relative;
 left:365px;
 top: -430px;
 }

 img.sensor2 {
 position:relative;
```

```
left:85px;
top: -430px;
}

img.sensor3 {
position:relative;
left:-10px;
top: -235px;
}

img.sensor4 {
position:relative;
left:60px;
top: -160px;
}

img.q00 {
position:relative;
left:-389px;
top: -100px;
}

img.ev1 {
position:relative;
left:579px;
top: -598px;
}

img.ev2 {
position:relative;
left:304px;
top: -598px;
}

img.ev3 {
position:relative;
left:232px;
top: -380px;
}

img.ev4 {
position:relative;
left:278px;
top: -326px;
}
```