Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

VICTOR JOSÉ DIAS REGINATO

ELABORAÇÃO DE EXPERIMENTOS NA ÁREA DE AUTOMAÇÃO INDUSTRIAL E CONTROLE UTILIZANDO PLATAFORMAS EXPERIMENTAIS

Campina Grande, Paraíba
Março de 2015

VICTOR JOSÉ DIAS REGINATO

ELABORAÇÃO DE EXPERIMENTOS NA ÁREA DE AUTOMAÇÃO INDUSTRIAL E CONTROLE UTILIZANDO PLATAFORMAS EXPERIMENTAIS

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Automação Industrial

Orientador:
Professor George Acioli Júnior, D. Sc.

Campina Grande, Paraíba
Março de 2015

VICTOR JOSÉ DIAS REGINATO

ELABORAÇÃO DE EXPERIMENTOS NA ÁREA DE AUTOMAÇÃO INDUSTRIAL E CONTROLE UTILIZANDO PLATAFORMAS EXPERIMENTAIS

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Automação Industrial

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor George Acioli Júnior, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho a toda minha família e aos amigos que estiveram comigo durante o curso, e sempre me apoiaram e me motivaram a nunca desistir.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por sempre me dar forças a continuar nesse caminho, colocando um sorriso no meu rosto em cada momento de desânimo.

Agradeço a minha família, que mesmo distante me apoiaram e me incentivaram, além de toda educação, carinho e condição que me deram para eu me formar e me tornar o que sou hoje.

Agradeço a minha namorada, Camila Peres, por todo apoio e atenção nos momentos mais complicados do curso. Onde sua paciência prevaleceu acima de tudo.

Agradeço a essa Instituição pela minha acolhida e pelas condições oferecidas, que me permitiram concluir este trabalho.

Agradeço ao meu orientador, George Acioli Júnior, pela paciência, tempo dedicado, apoio e ensinamentos, e também ao Mestrando Raphael Baltar, pela disponibilidade em ajudar nesse trabalho.

Enfim, agradeço a todos que, de alguma forma, passaram pela minha vida e contribuíram para a minha formação, em especial aos amigos: Beethoven Nobrega, Pablo Fabrício, Felipe Nascimento, Felipe Fernandes, Henrique Vanderlei, Rafael de Melo, Eduardo Souza, Cristovam André Trócoli, Rodrigo Almeida, entre outros.

"O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo.
Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo,
fará coisas admiráveis."

José de Alencar.

RESUMO

O trabalho consiste na formulação de experimentos, utilizando plataformas experimentais, para as disciplinas de Sistema de Automação Industrial e Controle Digital. Para a primeira disciplina, foram feitos guias com uma explanação sobre Controlador Lógico Programável e sua linguagem de programação, *Ladder*. Nesses guias, foi explorada a maioria das funcionalidades presente no CLP, a implementação delas e atividades básicas utilizando o mesmo. Os guias se dividem em etapas como Introdução, Fundamentação teórica, Atividades de Preparação e Atividades Práticas. A utilização dos módulos experimentais, no caso, módulos de temperatura, foi destinada a experimentos finais da disciplina de automação e na complementação de experimentos já existente na disciplina de Controle Digital, utilizando comunicação via OPC entre o CLP e o PC.

Palavras-chave: CLP, Automação, Módulos de Temperatura, Controle, Comunicação OPC, Linguagem *Ladder*.

ABSTRACT

The work consists in formulating experiments using experimental platforms for the disciplines of Industrial Automation Systems and Digital Control. For the first discipline, guides with an explanation of Programmable Logic Controller and its programming language, Ladder, were made. In these guides, was explored most of the functionality present in CLP, their implementation and basic activities. The guides are divided into steps as Introduction, Theoretical basis, Preparation Activities and Practical Activities. The use of experimental modules, in this case, temperature modules, was designed to end experiments of the discipline of automation and complementing existing experiments in the discipline of Digital Control using OPC communication between the PLC and the PC.

Keywords: PLC, Automation, Temperature Modules, Control, OPC, Ladder Language.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A automação industrial consiste em um conjunto de métodos que tem por objetivo a inserção de equipamentos físicos (hardware) e programas destinados ao controle desses equipamentos (software). A aplicação dessas tecnologias dentro do ambiente industrial possui vantagens competitivas de grande importância para o mercado globalizado.

Tendo em vista a necessidade do conhecimento dessas novas tecnologias, o trabalho a seguir trata de explicar e exemplificar o uso de um dos elementos que hoje é essencial em um processo produtivo industrial: o Controlador Lógico Programável.

1.1 CONTROLADOR LÓGICO PROGRAMÁVEL

Controlador Lógico Programável, também conhecido por sua sigla CLP ou de expressão inglesa PLC (*Programmable Logic Controller*), é um computador especializado, que usa circuitos integrados junto a um microprocessador que desempenha funções de controle através de softwares desenvolvidos pelo fabricante do CLP (BRYAN, 1997). Segundo a ABNT, é um equipamento eletrônico digital com hardware e software compatíveis com aplicações industriais.

Esse equipamento é capaz de armazenar instruções, em sequência, para controlar máquinas e processos diversos (BRYAN, 1997). Com esse objetivo, os CLPs são projetados e fabricados para trabalhar em ambientes diversos, capazes de suportar variações de temperatura e pressão, ambientes sujos, ruídos elétricos, e diferentes vibrações e impactos.

1.2 OBJETIVO E MOTIVAÇÃO

O objetivo principal é justificar a teoria vista nas disciplinas de Sistemas de Automação Industrial e Controle Digital, através de atividades práticas e comuns em um ambiente industrial.

De forma detalhada, será apresentado ao aluno o CLP e a linguagem de programação *Ladder* através de guias experimentais, onde serão explicadas as diferentes funcionalidades presentes no CLP disponível para uso nas disciplinas a qual esse trabalho é destinado e, além disso, propor diferentes atividades utilizando o equipamento. Esses guias serão apresentados nos capítulos a seguir, divididos segundo a complexidade das funções presentes no CLP.

2 APRESENTAÇÃO DO CLP E DA LINGUAGEM *LADDER*

No decorrer deste tópico, será apresentado a estrutura do primeiro experimento referente a disciplina de Sistemas de Automação Industrial. Ele tratará de conhecimentos básicos para o uso do CLP.

2.1 DEFINIÇÃO E PRINCÍPIO DE FUNCIONAMENTO

Controlador Lógico Programável é um dispositivo eletrônico utilizado para a automação de processos industriais, como controle e acionamento de motores, válvulas, sensores, atuadores, etc.

CLPs trabalham de forma cíclica, executando determinados passos de forma sequencial. Os três passos básicos do ciclo de execução de uma tarefa, após a inicialização do CLP, pode ser dividido em: Leitura das entradas, Execução do Programa e Escrita nas Saídas.

O primeiro passo consiste na varredura das entradas do CLP, onde elas são lidas e seus valores são copiados para a memória do CLP para serem usados durante a execução do programa.

Com os valores das entradas armazenados na memória, o programa principal é executado sequencialmente. Caso necessário, algum valor pode ser modificado na memória. É importante alertar que o programa não interage diretamente com as entradas e saídas do CLP, mas com a posição de memória referente a elas.

No último passo, o CLP escreve nas saídas os valores definidos pelo código do programa.

2.2 HARDWARE DE UM CLP

Um CLP é composto basicamente por quatro partes: fonte de alimentação, unidade de processamento, memória e bateria.

I. Fonte de alimentação: Converte a tensão da rede de 110/220V alternada em +5V, +12V ou +24V contínua, para alimentar os circuitos eletrônicos, entradas e saídas, e bateria interna.

II. Unidade de Processamento: Normalmente conhecida por CPU, é composto por microprocessadores ou microcontroladores que executam as instruções do programa. Com mais funções sendo incorporadas ao CLP e a necessidade de controles mais rápidos, o *clock* das unidades de processamento chegam, hoje em dia, a unidade de *Gigahertz*.

III. Memória: No CLP, existem três tipos de memória, cada uma com sua especificidade para funcionamento correto do CLP.

   a) A memória do programa supervisor é onde o programa supervisor do CLP é guardado. Esse programa monitora as entradas e saídas, inicializa o CLP, e realiza testes de diagnósticos. Pode ser dito que este programa é para o CLP o que um sistema operacional é para um computador. Essa memória normalmente é do tipo PROM, EPROM ou EEPROM.

   b) O programa escrito pelo usuário é salvo em outro tipo de memória chamada memória do usuário. Esta pode ser interna do tipo RAM, EEPROM ou FLASH-EPROM, ou pode ser externa, com cartuchos de memória.

   c) Por último, temos a memória de dados, responsável pelo armazenamento de todas as variáveis utilizadas pelo programa. Nesta também é guardado a memória-imagem das entradas e saídas, que são atualizadas pelo CLP nos primeiro e último passo do ciclo de execução.

IV. Bateria: A bateria interna serve para a alimentação do circuito de relógio em tempo real, retenção de dados em caso de queda de energia e alimentação da memória do usuário, caso esta seja do tipo RAM.

2.3 TIPOS DE ENTRADAS E SAÍDAS

Os CLPs podem trabalhar tanto com varáveis digitais como analógicas. A seguir serão detalhados esses dois tipos de entradas e também alguns exemplos de dispositivos compatíveis com as mesmas.

I. Entradas e Saídas Digitais

São aquelas que fornecem apenas dois valores, 0 (falso, inativo, desligado) ou 1 (verdadeiro, ativo, ligado). O uso desse tipo de dado é interessante para dispositivos que necessitam apenas o controle de ligado ou desligado Alguns exemplos de dispositivos digitais que podem ser interligados as entradas digitais do CLP são: botoeiras, botões, chaves seletoras.

II. Entradas e Saídas Analógicas

São aquelas que podem assumir qualquer valor. Para o CLP, estas entradas analógicas assumem qualquer valor pré-determinado em uma faixa de tensão, que será convertida para unidade digital por um conversor A/D. Para as saídas analógicas ocorre o inverso, onde o dado passa por um conversor D/A, e assumem certos valores dependendo da resolução do conversor do CLP. Alguns exemplos de transdutores analógicos que podem ser interligados a essas entradas: sensores de luminosidade, temperatura, pressão e umidade. Podemos utilizar as saídas para controle de velocidade de motores, controle de abertura de válvula, intensidade luminosa da lâmpada, entre outras aplicações.

2.4 KIT DE TREINAMENTO EZTK900

É importante ressaltar que cada fabricante de CLP desenvolve seu próprio *software* de supervisão e programação. Nessa disciplina, utilizaremos o kit de treinamento eZTK900, acompanhado do programa SPDSW (HI TECNOLOGIA, 2015), ambos produzidos pela HI tecnologia, empresa brasileira sediada na cidade de Campinas, São Paulo, que atua nas áreas de fabricação de equipamentos e prestação de serviços para automação industrial.

2.4.1 APRESENTAÇÃO DO KIT

O kit eZTK900 é um kit de treinamento baseado no controlador eZAP900. Esse kit proporciona uma fácil interface de uso a partir de chaves, reostatos e LEDs para o eZAP900. A Figura 1 ilustra o kit que será utilizado nos experimentos.

Figura 1. Foto ilustrativa do kit eZTK900

Esse kit é alimentado por uma tensão de 85 a 265Vac e consome uma corrente máxima de 200mA, com potência de 1,3 Watts. No total, são 12 entradas digitais, sendo 8 ativadas por chaves do tipo alavanca com LEDs de supervisão e 4 entradas via bornes tipo banana para entradas externas. Para as saídas digitais, conta-se com 8 LEDs de supervisão mais 4 bornes tipo banana para conexão externa. A saída O0 pode ser utilizada como geradora de frequência e PWM. Seguem as Figuras 2 e 3 que ilustram as entradas e saídas.

Figura 2. Foto ilustrativa das entradas digitais do kit.

Figura 3. Foto ilustrativa das saídas digitais do kit.

O kit ainda possui um painel para duas entradas analógicas que podem ser controladas via reostatos ou sinais externos, de 0 a 5V, de acordo com a posição da chave seletora (local ou externa). Há também uma saída analógica que pode ser conectada externamente a partir de bornes tipo banana. As Figuras 4 e 5 ilustram as entradas e saídas analógica.

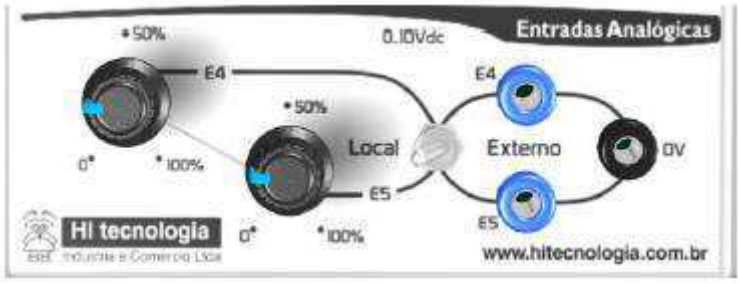


Figura 4. Foto ilustrativa das entradas analógicas do kit.

Figura 5. Foto ilustrativa da saída analógica do kit.

O CLP eZAP900 conta ainda com uma interface homem-máquina que permite a supervisão de variáveis como também a programação de parâmetros, sinalização de eventos e alarmes.

Figura 6. Foto ilustrativa do painel de interface homem-máquina.

Além disso, o CLP ainda fornece, em sua parte traseira, uma porta Ethernet e uma porta seria para comunicação com outros dispositivos a partir de protocolos como MODBUS-RTU, MODBUS-TCP e ASCII (SOUZA, 2010). A parte traseira também fornece entradas e saídas extras, caso necessário.

Figura 7. Foto ilustrativa da parte traseira do CLP.

2.4.2 APRESENTAÇÃO E CONFIGURAÇÃO DO *SOFTWARE* SPDSW

O SPDSW é um software proprietário da HI tecnologia que permite a programação lógica de controle do CLP através da linguagem Ladder. A interface principal é mostrada na Figura 8.

Figura 8. Interface principal do SPDSW.

Para começar a programar é necessária a criação de um novo projeto. Para isso devemos realizar os seguintes passos:

1º Passo: Clique na janela *Projeto* > *Novo*. Com isso, temos uma nova janela que disponibiliza a entrada de alguns dados como nome do programa, descrição, responsável, conforme a Figura 9.

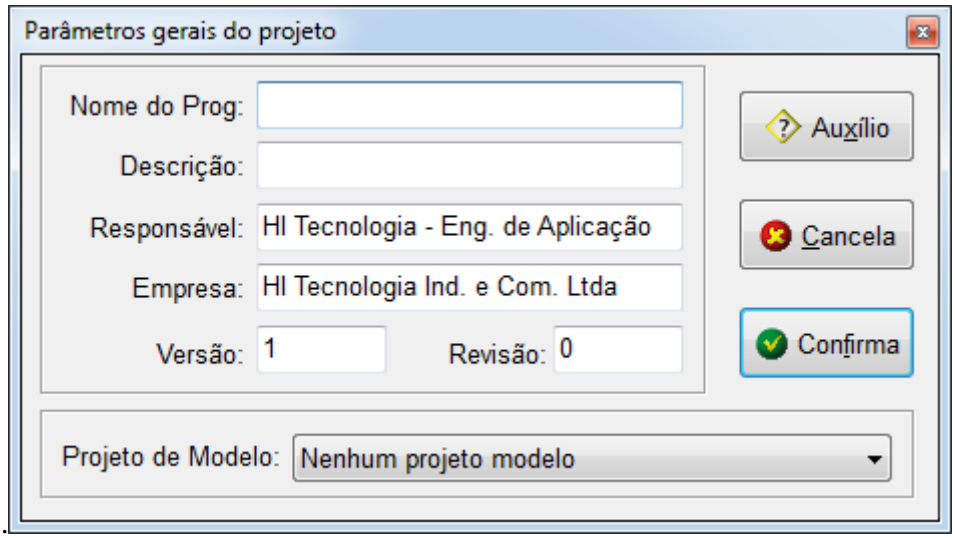


Figura 9. Interface de criação de projetos no SPDSW.

2º Passo: Após definido todos os parâmetros do novo projeto, deve-se selecionar o CLP que será utilizado. Para isto, clica-se em *Programa* > *Controlador programável* (Figura 8). Uma janela se abrirá com uma lista de CLPs da HI tecnologia. No caso dos experimentos realizados nessa disciplina, escolhe-se o CLP eZAP900 e clica-se em *Confirma*. Obs.: Caso o CLP na sua bancada seja diferente do citado no guia, escolha-o corretamente baseado nas opções disponíveis no programa.

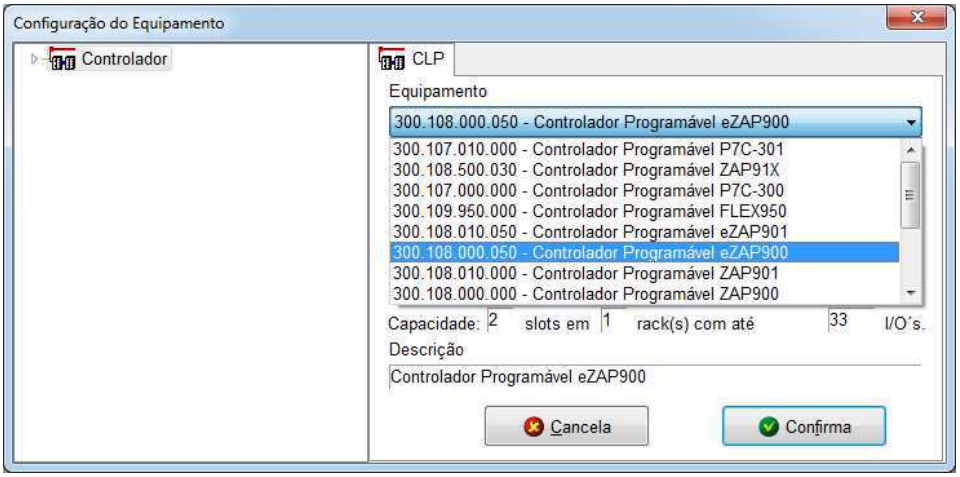


Figura 10. Interface de seleção do CLP

3º Passo: Após criação de projeto e seleção do CLP, devemos realizar a conexão entre o CLP e o programa através do botão *Conectar*, presente na interface principal do

programa (Figura 8). Essa conexão pode ser feita via Ethernet ou Serial (verifique se o CLP está conectado fisicamente).

4º Passo: Depois de feita a conexão, pode-se então programar o CLP clicando em *Editor Ladder* (ver Figura 8). O usuário pode programar com o CLP não conectado, mas será impossível testar o programa desenvolvido. O editor é representado pela Figura 11

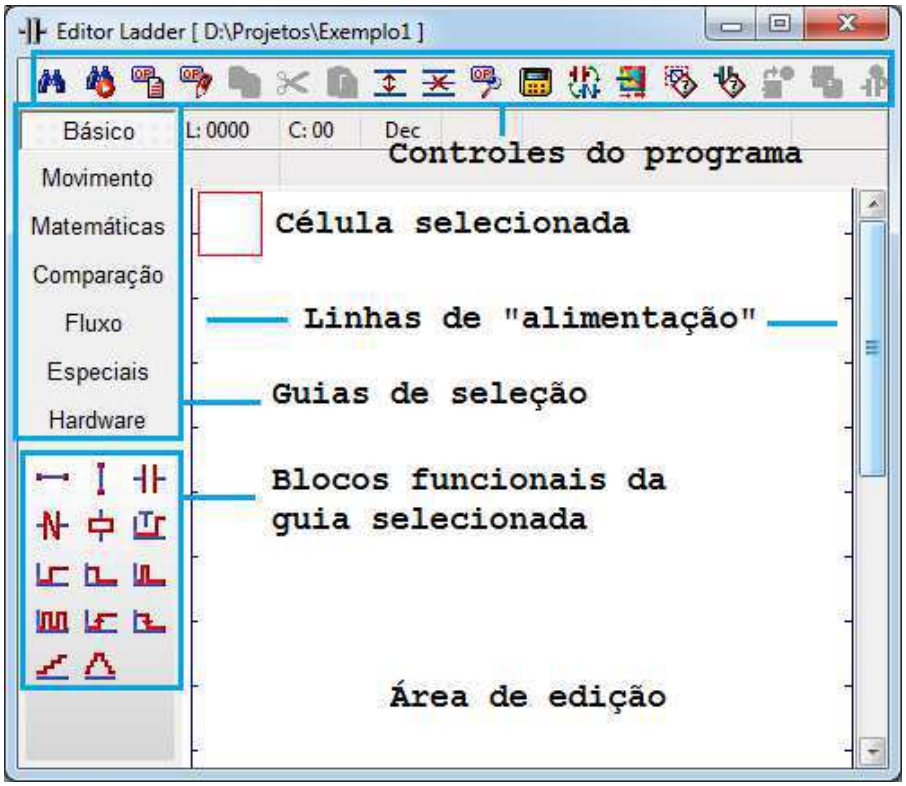


Figura 11. Interface do editor Ladder.

O editor *Ladder* é bastante simples e intuitivo. As linhas verticais nas laterais, à esquerda e à direita, são linhas de "alimentação" dos blocos funcionais, sendo a linha esquerda equivalente ao $+V_{cc}$, comparando ao um diagrama elétricos, e a linha direita ao terra.

A área de edição é onde os blocos funcionais são dispostos para construir o diagrama lógico. A célula selecionada fica destacada por um retângulo vermelho. Com a célula desejada selecionada, pode-se inserir um bloco funcional no diagrama a partir das opções disponíveis no programa.

Existem botões de controle do programa para tarefas como inserção e exclusão de linhas, compilação do programa, e programação da interface homem-máquina, caso necessário, destacado na Figura 11. O *software* também conta com um botão dedicado a

ajuda, que permite ver uma breve descrição do bloco funcional selecionado na área de edição.

Os blocos funcionais são organizados em guias, que ficam na parte esquerda da janela, de acordo com suas funcionalidades. As tabelas a seguir descrevem os blocos funcionais presentes em cada guia.

Tabela 1. Descrição do Guia Básico.

Guia: Básico			
Ícone	**Descrição (Atalho)**	**Ícone**	**Descrição (Atalho)**
	Linha Horizontal (H)		Linha Vertical (V)
	Chave NA (A)		Chave NF (F)
	Relé (B)		Temporizador (T)
	*SET* (S)		*RESET* (R)
	Pulso (P)		Oscilador (O)
	*SET* de Borda		*RESET* de Borda
	Contador Simples		Contador Bidirecional

Tabela 2. Descrição do guia movimento

Guia: Movimento			
Ícone	**Descrição (Atalho)**	**Ícone**	**Descrição (Atalho)**
	Movimentação (M)		Movimentação Indexada (X)
	Mov. Indexada Estendida		Inicialização de Dados (I)
	Conversão de Dados		

Tabela 3. Descrição do guia Matemática

Guia: Matemática			
Ícone	**Descrição (Atalho)**	**Ícone**	**Descrição (Atalho)**
	Soma (+)		Subtração (-)
	Multiplicação (*)		Divisão (/)
	Raiz Quadrada		Logaritmo na Base 10
	Exponencial		Potenciação
	Lógica *AND*		Lógica *OR*
	Lógica *XOR*		Deslocamento para Esquerda
	Deslocamento para Direita		

Tabela 4. Descrição do Guia Comparação

Guia: Comparação			
Ícone	**Descrição (Atalho)**	**Ícone**	**Descrição (Atalho)**
	Igual		Diferente
	Maior Que		Maior ou Igual a
	Menor Que		Menor ou Igual a
	Teste *AND*		

Tabela 5. Descrição do Guia Fluxo

Guia: Fluxo			
Ícone	**Descrição (Atalho)**	**Ícone**	**Descrição (Atalho)**
	Início de Relé Mestre		Fim de Relé Mestre
	Início de Bl. de Lógica		Fim de Bl. de Lógica
	Bloco de Lógica		Fim de Programa

Tabela 6. Descrição dos Guias Especiais e Hardware

Guias: Especiais e Hardware			
Ícone	**Descrição (Atalho)**	**Ícone**	**Descrição (Atalho)**
	PID		*Fast Counter*
	Gerador de Frequência		

3 EXPERIMENTO 01 – INTRODUÇÃO A LINGUAGEM *LADDER*: CONTATOS E FUNÇÕES LÓGICAS BÁSICAS.

3.1 INTRODUÇÃO.

A linguagem *Ladder* foi a primeira linguagem a surgir para a programação de CLPs. Foi desenvolvida juntamente com os primeiros equipamentos na década de 70 para substituir a lógica de relés. É uma linguagem gráfica, desenvolvida para assemelhar-se aos diagramas elétricos que eram utilizados naquela época, como mostra a Figura 12.

Como foi dito anteriormente na explicação do *software* utilizado, *Ladder* possui duas barras verticais que representa a alimentação e o neutro/terra. Os elementos são colocados em linhas horizontais, interligando as duas barras e construindo a lógica do programa, como mostra a Figura 13. É importante ressaltar que os diagramas são executados sequencialmente, da esquerda para a direita, de cima para baixo.

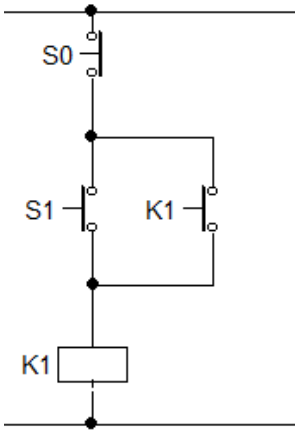


Figura 12. Diagrama Elétrico.

Figura 13. Diagrama em *Ladder*.

3.2 CHAVES, RELÉS E CONTATOS AUXILIARES.

Chaves e relés são os elementos principais e mais básicos desse tipo de linguagem de programação. E com eles é possível implementar diferentes lógicas para as mais diversas aplicações como partida de motores, controle de nível de tanques, etc.

Em *Ladder*, as chaves podem ser Normalmente Abertas (NA) ou Normalmente Fechada (NF), ambas representadas por duas barras paralelas. Os relés são representados por um quadrado cortado. A Figura 14 representa os elementos abordados, presentes na Tabela 1.

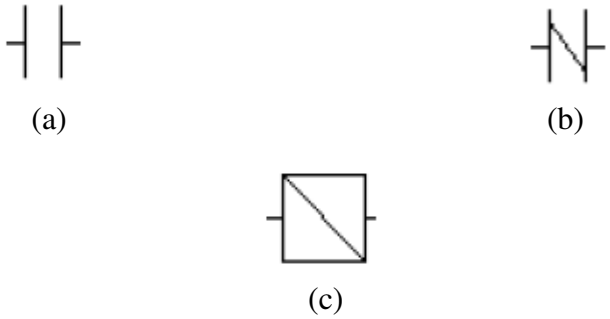


Figura 14. Representação das chaves (a) normalmente aberta, (b) normalmente fechada e (c) relé.

3.3 FUNÇÕES LÓGICAS EM *LADDER*.

A partir das chaves NA e NF, podemos programar diversas lógicas booleanas, como AND, OR, NAND, NOR, etc. Neste tópico, é mostrado a implementação de algumas destas funções, além de outras mais complexas.

a) Função AND

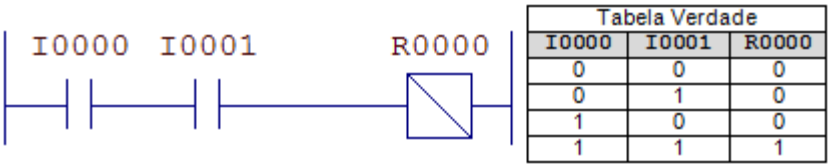


Tabela Verdade		
I0000	I0001	R0000
0	0	0
0	1	0
1	0	0
1	1	1

Figura 15. Implementação da função lógica AND.

b) Função OR

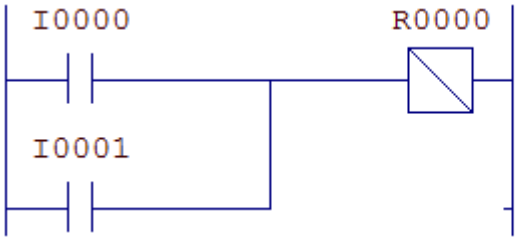


Tabela Verdade		
I0000	I0001	R0000
0	0	0
0	1	1
1	0	1
1	1	1

Figura 16. Implementação da função lógica OR.

c) Função NAND e NOR

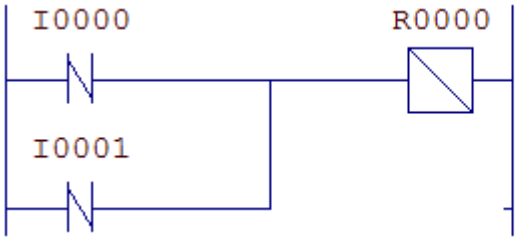


Tabela Verdade		
I0000	I0001	R0000
0	0	1
0	1	1
1	0	1
1	1	0

Figura 17. Implementação da função lógica NAND.

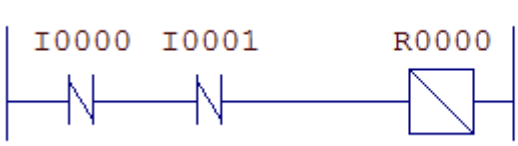


Tabela Verdade		
I0000	I0001	R0000
0	0	1
0	1	0
1	0	0
1	1	0

Figura 18. Implementação da função lógica NOR.

d) Função XOR

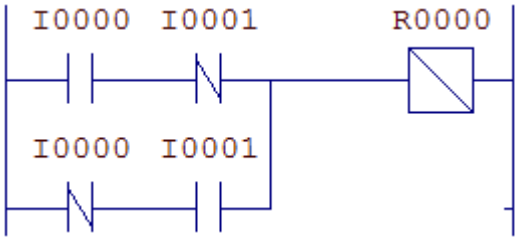


Tabela Verdade		
I0000	I0001	R0000
0	0	0
0	1	1
1	0	1
1	1	0

Figura 19. Implementação da função lógica XOR.

3.4 GUIA EXPERIMENTAL

3.4.1 EXERCÍCIOS DE PREPARAÇÃO

1) Projete diagramas *Ladder* para executar as seguintes equações logicas e descreva a tabela verdade de cada uma.
    - $Y = A + B + \overline{C}$
    - $Y = A * B + \overline{A} * C$
    - $Y = \overline{A} * (B + Y)$
    - $Y = B + \overline{A} * Y$

2) A função lógica XNOR é a negação da função XOR. Também chamada de função coincidência, sua saída é verdadeira somente se as duas entradas forem iguais e é representada da forma $Z = A \odot B = A * B + \overline{A} * \overline{B}$. Implemente esta função em *Ladder* e descreva sua tabela verdade.

3) Utilizando funções lógicas podemos implementar equações lógicas mais complexas para gerar diferentes resultados. Implemente em *Ladder* um circuito meio-somador. Sabemos que o resultado da soma é $Z = A \oplus B$ e que o carry-out é $C = A * B$. Construa a tabela verdade para auxiliar no entendimento.

4) Um tanque deve ter seu volume mantido numa certa faixa. O controle de volume do tanque é feito a partir de duas válvulas e dois sensores de nível, como mostrado na figura a seguir. As válvulas são controladas pelas saídas digitais O0000 e O0001, que determinam se a válvula será aberta ou fechada. Os sensores de nível 1 e 2 são ligados nas entradas digitais I0000 e I0001, respectivamente, e detectam se o nível do tanque está acima deles. Quando o nível do tanque estiver acima de um sensor de nível, este sensor é ativado. Com base nestas informações, projete o controle do tanque de modo que:
    - A válvula 1 é aberta quando o nível do tanque estiver menor que o nível do sensor 1, e fechada quando o nível estiver maior que o nível do sensor 2;

- A válvula 2 é aberta quando o nível do tanque estiver maior que o nível 1; fechada caso contrário.

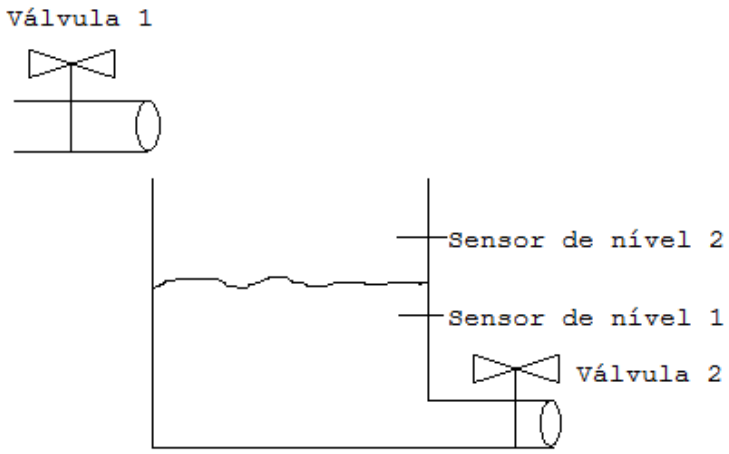


Figura 20. Tanque controlado por Válvulas.

3.4.2 PRÁTICA EXPERIMENTAL

Dicas rápidas:

- Quando selecionado o elemento na linha, aperte o botão *Edita operando selecionado* ou apenas pressione F6 para opções como nome, etc.
- Na hora de colocar o relé, não é necessário selecionar a última célula da linha para que o mesmo se encontre a linha de neutro. O SPDSW automaticamente fará isso, portanto pode-se usar uma célula qualquer.
- Para determinar o fim do programa utiliza-se o bloco *END* que se encontra na aba de *Fluxo*.
- Para testar o programa clica-se em *Compila/Carrega/Depura* ou simplesmente pressionando Ctrl+Z.
- Para inserir uma nova linha, clica-se na que está no local onde você quer inserir uma nova linha e clica-se em *Insere Linha*, ou simplesmente Ctrl+Ins.

1) Abra o Programa SPDSW, crie um novo projeto (Projeto > Novo), selecione o tipo de CLP (Programa > Controlador Programável), verifique a conexão entre o CLP e o *Software* e abra o Editor *Ladder* para começar a programar.

2) Implemente o diagrama *Ladder* de uma porta AND para testar se o CLP está conectado ao *software*.

3) Implemente duas funções lógicas do exercício de preparação 1 em *Ladder*. Compare o resultado com a tabela verdade correspondente.

4) Implemente a função lógica XNOR (exercício de preparação 2). Verifique sua tabela verdade.

5) Implemente o resultado obtido do exercício de preparação 3.

6) Implemente o resultado do exercício de preparação 4 e descreva a lógica das válvulas em álgebra de Boole. Compare o resultado com o que foi pedido na questão. O que acontece quando o nível do líquido está maior que o nível 1 e menor que nível 2? (Responda mostrando os estados das entradas e das saídas digitais).

4 EXPERIMENTO 02 – CIRCUITO DE SELO, CONTATOS AUXILIARES E FLIP-FLOP EM *LADDER*.

4.1 INTRODUÇÃO DO EXPERIMENTO

Tendo conhecimentos de blocos básicos, vamos utilizar deles para fazer algumas atividades como partidas de motor, controle válvulas e implementação de flip-flops. Mas para isso, será necessário tomar conhecimento de algumas práticas muito usadas em qualquer programa em *Ladder*: Circuitos de selo, contatos auxiliares e blocos de SET, RESET e Osciladores.

4.2 FUNDAMENTAÇÃO TEÓRICA

4.2.1 CONTATO DE SELO

O contato ou circuito de selo é sempre ligado em paralelo com o contato de fechamento da botoeira. Sua finalidade é de manter a corrente circulando pelo contator, mesmo após o operador ter retirado o dedo da botoeira. A Figura 21 representa um diagrama de partida direta de um motor em *Ladder*. Analisando esse diagrama, temos que I0000 e I0001 representam entradas e O0000 a saída. Quando I0001 é habilitada, o relé é energizado e, assim, a chave O0000 (contato de selo) sempre estará ligada, fazendo o relé permanecer ativo independente do estado da chave. O relé só desligará se pressionada a chave I0000. Este tipo de circuito é chamado de circuito de selo com **prioridade no desligamento**, pois se as duas chaves forem habilitadas ao mesmo tempo, o circuito sempre estará desligado.

Figura 21. Circuito de uma partida direta de um motor em *Ladder*.

Existe também o circuito de selo com **prioridade no ligamento**. Esse circuito é menos utilizado por razões de segurança, afinal, quando habilitamos as duas chaves ao mesmo tempo o circuito ligará. A Figura 22 representa esse circuito.

Figura 22. Partida direta com prioridade no ligamento.

As chaves utilizadas em ambos os diagramas são do tipo I, que são variáveis de entrada digital, enquanto a saída é do tipo O, que é uma variável de saída digital. Você pode observar essas entradas e saídas na parte frontal do CLP. Elas são o meio de comunicação deste com o meio externo.

4.2.2 CONTATOS AUXILIARES

Em algumas lógicas de programação, se faz necessário o uso de relés temporários para implementação da mesma. Quando for o caso, utilizamos relés internos do tipo R, que também são chamados de contatos auxiliares. A seguir temos a implementação de uma porta XOR (explicada no capítulo anterior) de quatro entradas que se faz uso de contatos auxiliares.

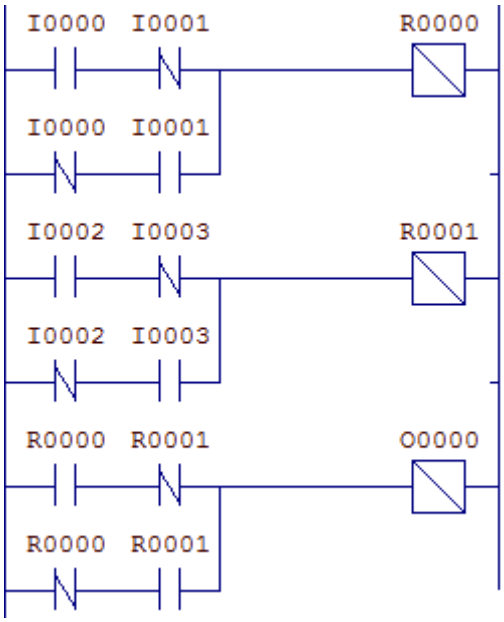


Figura 23. Diagrama Ladder de uma porta XOR de 4 entradas.

Podemos ver que o uso de contatos auxiliares facilita a leitura, assim também como a diminuição do diagrama.

Outro exemplo interessante é uma partida sequencial, ou seja, uma ação só pode ser executada se outra determinada ação já tiver sido executada. Podemos ver no diagrama da Figura 24 que existe um primeiro selo para o contato auxiliar R0000 a partir de I0001. Esse contato auxiliar, quando ligado, ativará a saída O0000 na linha imediatamente depois. No segundo selo, só é possível ligar O0001 se R0000 estiver ligado. Lembrando que I0000 desliga todo o circuito.

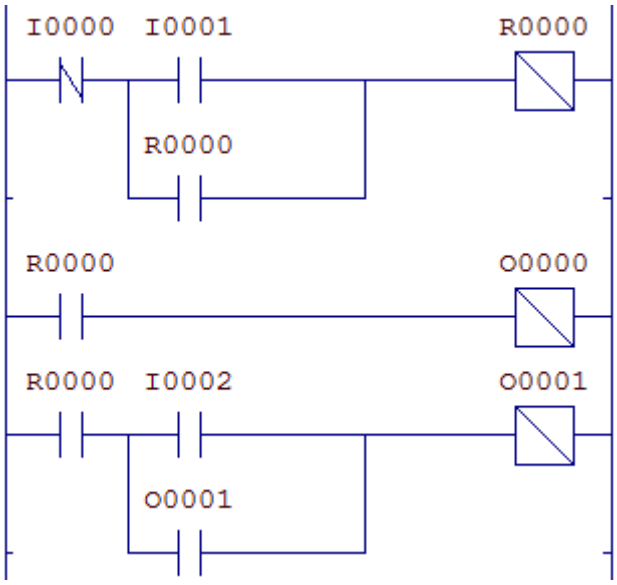


Figura 24. Diagrama Ladder de uma partida sequencial.

4.2.3 BLOCOS FLIP-FLOP

Em eletrônica e circuitos digitais, o flip-flop é um circuito pulsado, capaz de servir como uma memória de um bit. Existem diversos tipos de flip-flop, cada um com sua particularidade. A Figura 25 representa um flip-flop do tipo J-K.

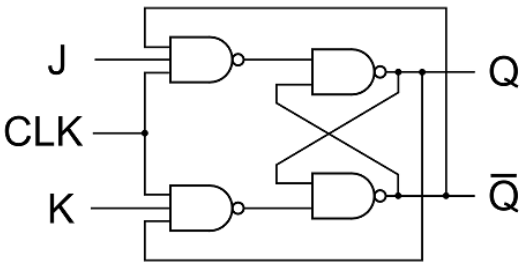


Tabela Verdade		
J	K	Q_t
0	0	Q_{t-1}
0	1	0
1	0	1
1	1	$\overline{Q}_{t-1}$

Figura 25. Flip-Flop J-K e sua tabela verdade.

Através do contato de selo podemos manter um estado ativo mesmo quando uma entrada é desativada. Porém, pode-se chegar ao mesmo resultado utilizando outros blocos do tipo SET e RESET (Blocos referentes a Tabela 1). Estes blocos em conjunto desempenham a função de outro tipo de flip-flop: tipo S-R.

Figura 26. Representação dos blocos SET e RESET.

O bloco SET verifica a entrada, e se esta estiver ativa, a saída é então ativada, mantendo este estado até que seja modificado por outros blocos ou chaves. É importante ressaltar que o bloco SET não desativa a saída, mesmo quando a entrada é desativada.

O bloco RESET faz exatamente o contrário: quando a entrada está ativa, a saída é desativada.

A seguir temos o circuito de selo com prioridade no desligamento e no ligamento. Podemos ver que a mudança está na ordem dos blocos SET e RESET. Como a execução ocorre de cima para baixo, temos que o último bloco no diagrama é o último bloco a ser ativado.

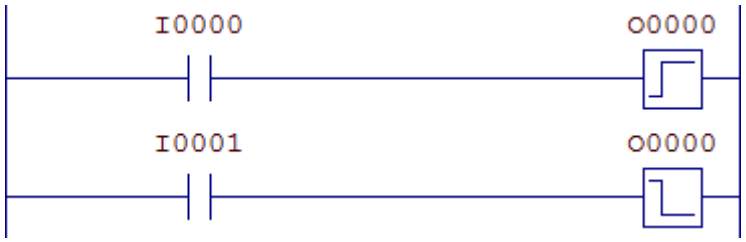


Figura 27. Selo com prioridade no desligamento.

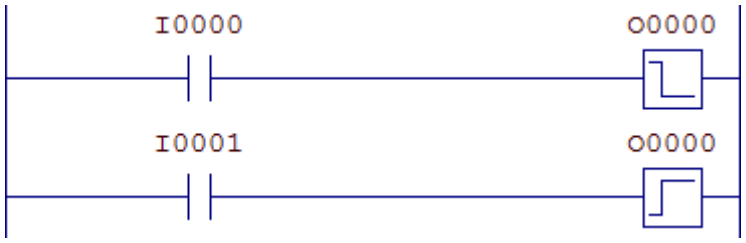


Figura 28. Selo com prioridade no ligamento.

Há também os blocos SET e RESET de borda (Tabela 1), que funcionam do mesmo modo, mudando apenas que são ativados na borda de subida (quando a entrada sai de inativa para ativa) do sinal de entrada, e não durante todo o tempo em que a entrada estiver ativa. Selos feitos com esses blocos não tem prioridade de ligamento ou desligamento, portanto não é aconselhável o uso destes blocos para esse tipo de circuito.

Figura 29. Representação *Ladder* dos blocos SET e RESET de borda.

Outro bloco importante relacionado a flip-flop é o bloco oscilador (Tabela 1). Este bloco funciona da mesma maneira que um flip-flop do tipo T: quando o sinal de entrada varia de inativo para ativo, a saída é invertida. Podemos simular um contador binário com esse tipo de flip-flop. A Figura 31 abaixo representa o esquema do diagrama feito com osciladores e contatos auxiliares, baseado no circuito da Figura 30.

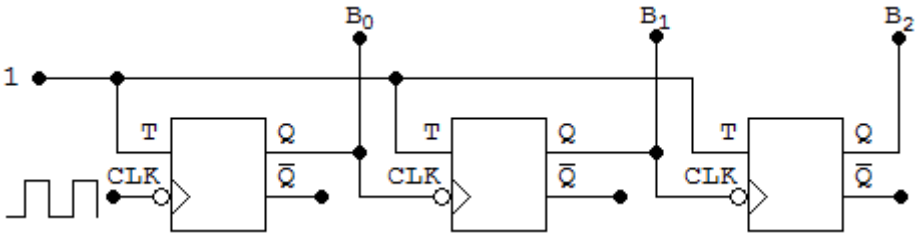


Figura 30. Contador de 3 bits composto de flip-flops tipo T.

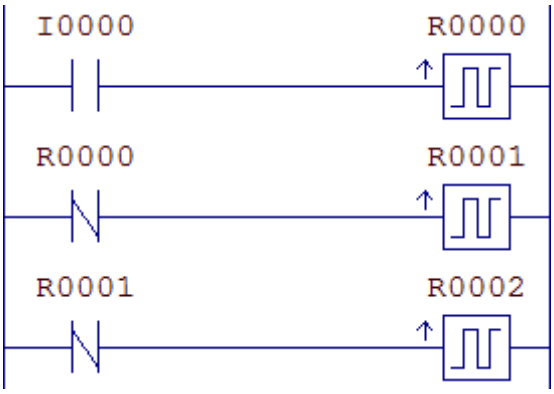


Figura 31. Representação Ladder do contador de 3 bits.

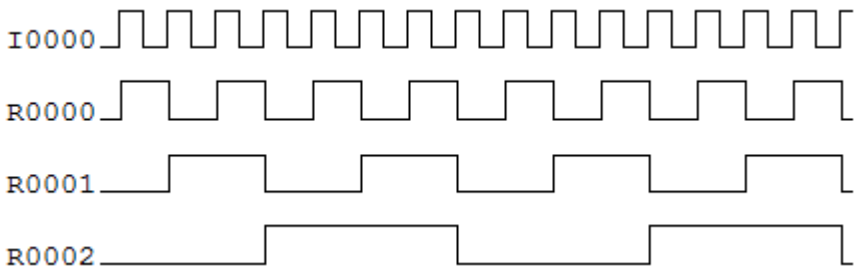


Figura 32. Gráfico de onda da entradas e das saídas do contador.

4.3 GUIA EXPERIMENTAL

4.3.1 EXERCÍCIOS DE PREPARAÇÃO

1) Um determinado motor deve ser ligado quando duas chaves são ativadas ao mesmo tempo. Programe o diagrama *Ladder* deste circuito. Depois, utilizando mais uma chave, implemente um selo com prioridade no desligamento para o mesmo motor.

2) Programe em *Ladder* o diagrama elétrico apresentado na Figura 33 abaixo utilizando circuitos de selo. Lembrando que Sx são entradas e Kx saídas.

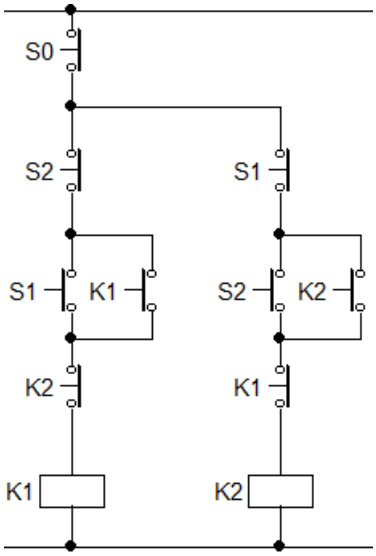


Figura 33. Diagrama elétrico de uma partida com reversão.

3) Determine uma solução para a questão anterior implementando o selo com prioridade no desligamento utilizando blocos SET e RESET.

4) Para o contador composto de blocos osciladores na figura 11 deste guia, elabore um modo de reiniciar a contagem utilizando blocos RESET.

5) Utilizando blocos RESET, modifique o programa da questão anterior para que o contador conte apenas até 5 e então reinicie a contagem. Lembrando que o usuário pode reiniciar a contagem quando quiser, segundo a lógica da questão anterior.

6) No experimento anterior, montamos um circuito de um meio-somador. Utilizando o conhecimento de contatos auxiliares, desenvolva um programa em *Ladder* para um circuito de somador completo.

4.3.2 PRÁTICA EXPERIMENTAL

Dica Rápida:

- Quando for fazer o selo, lembre-se de utilizar chaves NF ou NA com o nome do relé que você deseja ativar. Isso também serve para os contatos auxiliares.

1) Implemente o resultado obtido do exercício de preparação 1. Depois, modifique o diagrama para um selo com prioridade no ligamento. Compare os resultados obtidos dos dois diagramas.

2) Implemente o resultado do exercício de preparação 2. Compare o diagrama obtido com o diagrama elétrico. Quais as semelhanças e diferenças?

3) Implemente os resultados obtidos do exercício de preparação 3. Depois, determine e escreva o código de um selo com prioridade no ligamento.

4) Implemente o resultado do exercício de preparação 5 e verifique o sinal de saída.

5) Implemente o circuito somador completo da questão de preparação 6.

5 EXPERIMENTO 03 – CONTADORES, TEMPORIZADORES E OPERAÇÕES.

5.1 INTRODUÇÃO DO EXPERIMENTO

Para uma maior variedade de aplicações, vamos trabalhar nesse capítulo com blocos mais complexos. Além disso, será estudado como ler e escrever em sensores e atuadores analógicos, como também realizar operações matemáticas e de comparação entre eles.

5.2 FUNDAMENTAÇÃO TEÓRICA

5.2.1 TIPOS DE DADOS

Pelo que foi estudado nos capítulos anteriores, já usamos variáveis do tipo I, O e R. A seguir temos uma tabela com todos os tipos de dados disponíveis no ambiente *Ladder*.

Tabela 7. Tipos de dados disponíveis em Ladder

Tipo	Valores	Descrição
I	0 ou 1	Entrada lógica, aceita apenas valores binários.
O		Saída lógica, aceita apenas valores binários.
R		Variável interna lógica, aceita apenas valores binários.
E	-32e3 a +32e3	Entrada analógica inteira de 16 *bits*.
S		Saída analógica inteira de 16 *bits*.
M		Variável interna inteira de 16 *bits*.
K		Constante interna inteira de 16 *bits*.
D	- 2e31 a + 2e31	Variável interna real de 32 *bits*.
Q		Constante interna real de 32 *bits*.
X	Texto ASCII	Variável interna de texto de até 48 *bytes*.
W		Constante interna de texto de até 48 *bytes*.
T	-	Identificador de blocos e sub-rotinas do programa.

Analisando a tabela, podemos ver que existem cinco tipos de dados no geral: lógico, inteiro, real, texto e identificador. Os demais tipos são variações desses cinco primeiros.

Como já sabemos, os dados em *Ladder* são exibidos na forma YXXXX onde Y caracteriza o tipo de dado e XXXX é o seu número de identificação na memória.

5.2.2 CONTADORES

Foi apresentado no experimento anterior um contador de 3 *bits* feito a partir de flip-flops do tipo T. Apesar de serem simples de implementar, o mesmo não é muito utilizado. Isso graças à existência de blocos contadores disponíveis em *Ladder*, que são mais versáteis e simples de utilizar.

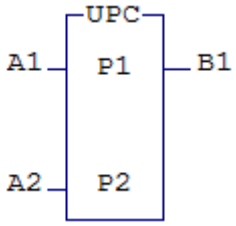


Figura 34. Representação Ladder de um bloco contador.

O bloco contador (Tabela 1) caracteriza-se por possuir duas entradas e leva dois parâmetros. A entrada A1 determina a contagem do bloco: quando esta passa do estado inativo para ativo, o contador é incrementado. A entrada A2 habilita ou zera o contador: quando A2 está inativo, o contador é zerado, independente de A1, e quando ativo, permite a contagem. O parâmetro P1 guarda o estado da contagem e deve ser do tipo M. Esta variável é incrementada ao passo que A1 é ativada, ou zerada quando A2 fica inativo. P2 é o máximo valor que o contador pode chegar, podendo ser do tipo M ou K. Quando a contagem atingir o valor de P2, a saída B1 é ativada. Para exemplificar, suponhamos que K0000 = 7.

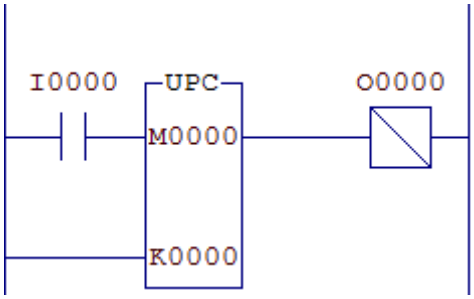


Figura 35. Diagrama Ladder do exemplo.

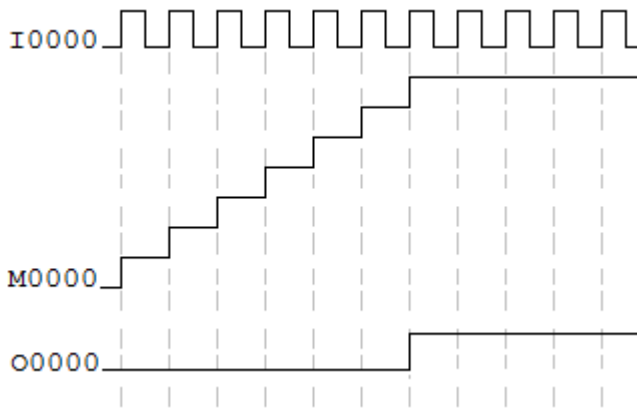


Figura 36. Gráfico de onda da entrada e saída do contador.

Podemos ver que o contador apenas conta uma vez e para, até que I0000 novamente seja ativado. Ao atingir o valor definido em K0000 (M0000 = K0000), a saída O0000 é então ativada, permanecendo assim indefinidamente pelo fato de que a entrada A2 está sempre ativa.

Existe outro tipo de contador capaz de contar de forma crescente ou decrescente. Esse contador é chamado bidirecional (Tabela 1). A Figura 37 representa esse bloco.

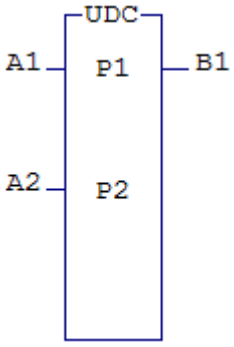


Figura 37. Representação Ladder de um contador bidirecional.

Este bloco funciona de maneira similar, apenas com uma entrada a mais. A1 incrementa ou decrementa a contagem, dependendo do estado de A2. Caso A2 esteja ativo, o sentido de contagem é crescente (incrementa o contador) e, se inativa, o sentido de contagem é decrescente (decrementa o contador). A entrada A3 tem a função de habilitar ou reiniciar a contagem. Quando a contagem é reiniciada, deve-se analisar o estado de A2: se ativa, o contador é zerado, e se estiver inativa, o contador recebe o valor máximo definido pelo parâmetro P2. O parâmetro P3 é inexistente. Considere o exemplo a seguir, onde K0000 = 3.

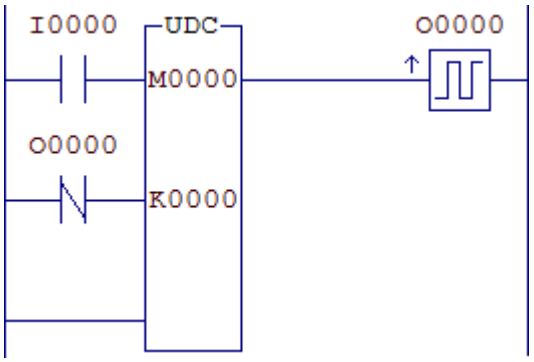


Figura 38. Diagrama Ladder do exemplo com contador bidirecional.

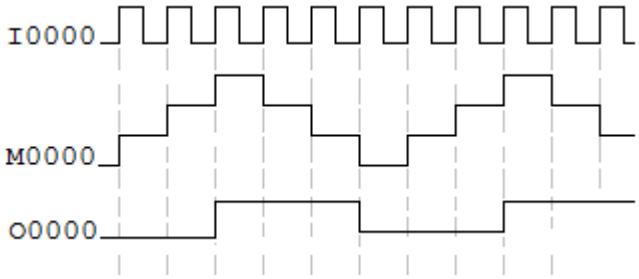


Figura 39. Gráfico de conda da entrada e saída do bloco contador bidirecional.

5.2.3 TEMPORIZADORES

Temporizadores (Tabela 1) são utilizados em diversas aplicações como partida de motores, atuação de válvulas, etc. Em *Ladder* dispomos também desse artificio.

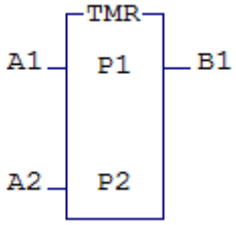


Figura 40. Representação Ladder do temporizador.

O funcionamento do bloco temporizador é semelhante ao contador. Suas variáveis de entrada e parâmetros funcionam da seguinte forma:

- P1 é o tempo atual da contagem, significando quanto tempo falta para o término. Deve ser do tipo inteiro (M);
- P2 é o tempo inicial da contagem e deve ser um inteiro variável ou constante (K);
- A1 habilita ou pausa a contagem;
- A2 habilita ou reinicia a contagem.

O funcionamento ocorre de acordo com as combinações dos parâmetros de entrada.

Tabela 8. Opções de controle do bloco temporizador.

A1	A2	Ação
0	0	Reinicia o temporizador e pausa a contagem
0	1	Contador pausado
1	0	Reinicia o temporizador
1	1	Habilita a temporização

Uma observação importante tem que ser feita: o bloco temporizador conta em passos de 10*ms*. Ou seja, deve-se seguir a equação:

$$P2 = 100 * t_{desejado} \quad (1)$$

Por exemplo, caso o usuário queira um tempo de quatro segundos, P2 deve ser igual a 400. Quando o temporizador atinge zero, a saída B1 é ativada e assim permanece até que o temporizador seja reiniciado.

5.2.4 MANIPULAÇÃO DE DADOS

Para manipular diferentes valores lidos por sensores, sejam eles digitais ou analógicos, precisamos tomar conhecimento do bloco funcional MOV, que permite a leitura e escrita de valores na memória do CLP.

i) Leitura e escrita de variáveis analógicas

O uso de sensores analógicos é de extrema importância e comum para controle de processos. São diversos os tipos: temperatura, pressão, luminosidade, pH, etc. Para trabalhar com os valores provenientes desses sensores, no CLP, utilizamos o bloco MOV (Tabela 2).

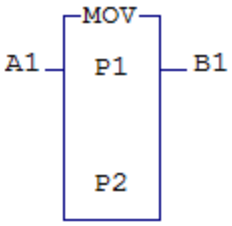


Figura 41. Representação Ladder do bloco MOV.

O bloco tem apenas uma entrada, que, quando ativa, permite a operação de movimentação do dado para a memória. Caso a movimentação tenha sido bem sucedida, a saída B1 será ativada. O parâmetro P1 é o valor que será atribuído à P2, ou seja, este bloco faz com que P2 = P1.

Considere o sistema de controle de temperatura da Figura 42.

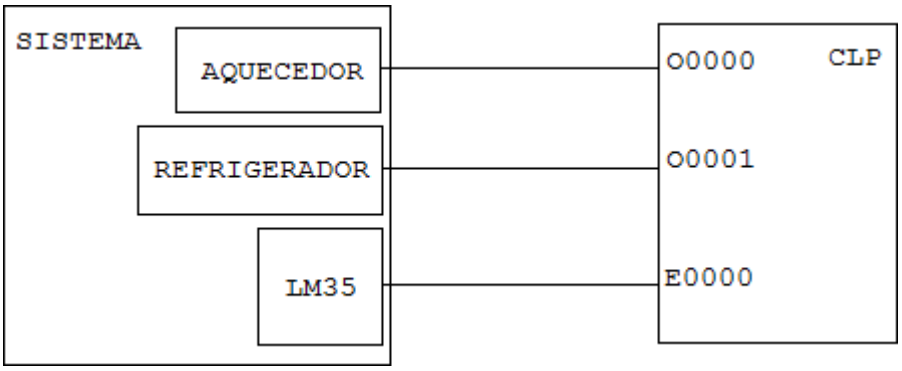


Figura 42. Diagrama de ligação entre o sistema de controle e um CLP.

O sensor LM35 é um sensor de temperatura. O valor lido pelo sensor é armazenado na variável de entrada E0000. Com esse dado guardado no E0000, podemos “move-lo” para onde o programador quiser. No caso uma variável do tipo M.

A escrita nas saídas analógicas do CLP é feita de forma análoga. Neste caso, o parâmetro P2 deve ser do tipo S, que é o tipo de dado referente à saída analógica.

ii) Leitura e escrita de variáveis digitais

Assim como quando tratamos de dados analógicos, o bloco MOV é também utilizado para tratamento de dados digitais. Para a escrita, o parâmetro P2 deve ser do tipo O e seu índice deve ser múltiplo de 16 (0, 16, 32...). Neste modo, o CLP escreve o parâmetro P1 (que pode ser um inteiro variável ou constante) *bit* a *bit* nas saídas digitais. Ou seja, um valor de M0000 gravado em O0000, significa a escrita de M0000 nas saídas digitais de O0000 a O0015.

Também é possível fazer a leitura em bloco de entradas digitais com o mesmo bloco MOV. Para isso, P1 deve ser do tipo I e seu índice deve também ser múltiplo de

16. Já o parâmetro P2 deve ser do tipo M. Exemplo: Se I0015,...,I0000 = 1110110010110011, o valor lido seria M0000 = -4941, de acordo com o a representação numérica de complemento de 2.

5.2.5 Operações Matemáticas

Tendo conhecimento de como ler e escrever dados em entradas e saídas analógicas, vamos agora tratar esses dados que muitas vezes são brutos. Para isso, *Ladder* oferece blocos de funções matemáticas para o tratamento destes dados. Vamos utilizar o exemplo do LM35 para demonstrar essas operações.

Segundo o *datasheet* do sensor LM35, a saída V_{out}é linearmente proporcional à temperatura, com proporção de $10mV/°C$, seguindo a equação:

$$V_{out} = 0{,}01 * T \tag{2}$$

Temos que levar em conta também a resolução de 12 *bits* da entrada analógica do CLP além de trabalhar numa faixa de 0 a +5V. Neste caso, o valor máximo lido (o máximo valor que E0000 pode ter) é de 4095, ou $(2^{12} - 1)$. Assim, dividindo-se a faixa de valores pela resolução, encontra-se a sensibilidade S da entrada analógica.

$$S = \Delta V/R$$

$$S = 5/4096$$

$$S \approx 1{,}22mV/unidade \tag{3}$$

Com as equações (2) e (3), podemos determinar a ligação direta entre o valor lido na entrada e o valor da temperatura em $°C$, sabendo que $V_{out} = S * E$.

$$T_{real} = \frac{V_{out}}{0{,}01}$$

$$= 100 * V_{out}$$

$$= 100 * S * E$$

$$\approx 0{,}122 * E \tag{4}$$

Para resolver esse problema de manipulação de dados, contamos com blocos que realizam operações matemáticas (Tabela 3).

- Soma (bloco ADD): $P3 = P1 + P2$
- Subtração (bloco SUB): $P3 = P1 - P2$
- Multiplicação (bloco MUL): $P3 = P2 * P1$
- Divisão (bloco DIV): $P3 = P1/P2$
- Potenciação (bloco POW): $P3 = P1^{P2}$
- Raiz Quadrada (bloco SQR): $P2 = \sqrt{P1}$
- Exponencial (bloco EXP): $P2 = \exp P1$
- Logaritmo base 10 (bloco LOG): $P2 = \log P1$

5.2.6 Operações de Comparação

Podemos desenvolver um simples projeto de controle de temperatura para um sistema, utilizando blocos de comparação. A Figura 43 mostra os blocos de comparação disponíveis em *Ladder* (Tabela 4).

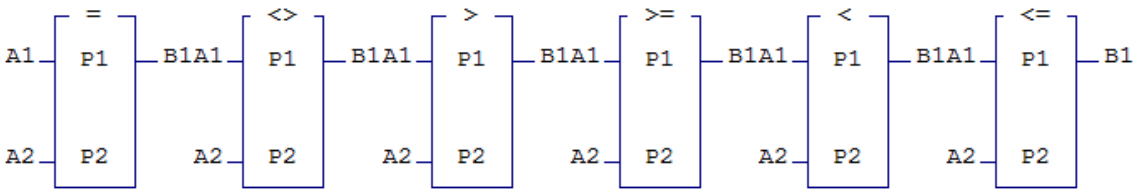


Figura 43. Blocos funcionais de comparação.

Esses blocos funcionam comparando os dois valores estabelecidos por P1 e P2. O teste é feito sempre de P1 em relação a P2. Caso o teste lógico seja verdadeiro, a saída B1 será ativada. Só existe uma única entrada para que o bloco seja ativado, A1.

5.3 Guia Experimental

5.3.1 Exercícios de Preparação

1) Dado o diagrama elétrico da Figura 44, elabore um programa em *Ladder* que obtenha semelhante funcionamento. Defina o valor da constante do temporizador.

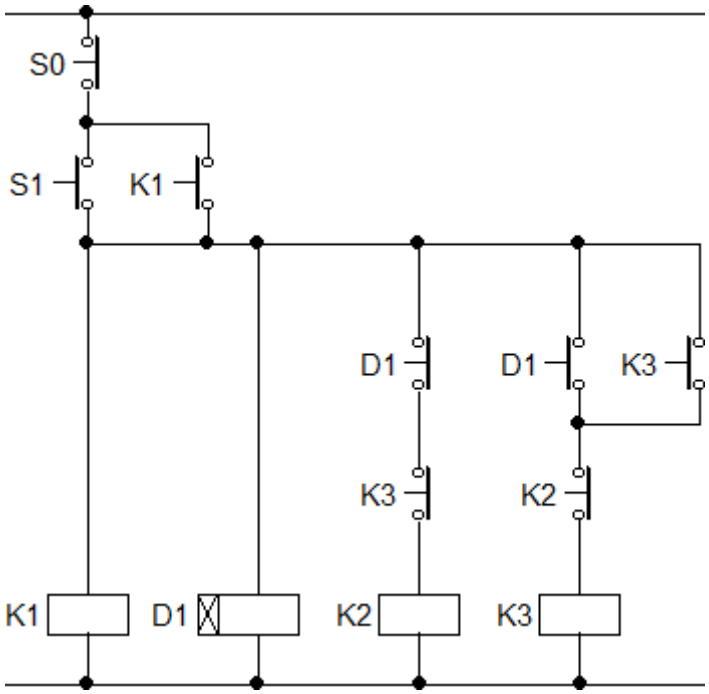


Figura 44. Diagrama elétrico de uma partida Δ-Y com temporizador.

2) Em uma linha de produção, uma esteira possui um sensor que é ativado quando um objeto cruza certo ponto. Após 10 objetos cruzarem esta esteira, o sistema de controle deve parar a esteira e enviar um sinal para que seja iniciado o processo de empacotamento. A empacotadeira, por sua vez, envia um sinal de volta ao sistema de controle, indicando que o processo de empacotamento foi finalizado e que a esteira pode voltar a funcionar. Considere que I0000 é o sinal recebido do sensor, O0000 controla o movimento da esteira, e I0001 e O0001 são os sinais recebidos e enviados à empacotadeira, respectivamente. Com base nesta descrição, elabora um diagrama *Ladder* para o sistema de controle.

3) Considerando a mesma esteira do exercício anterior, os objetos que passam sobre ela são de dois tipos: A e B. Um sensor identifica qual o tipo do objeto que está passando pela esteira e envia dois sinais para um segundo sistema de controle: o primeiro sinal indica que há um objeto passando; o segundo sinal indica o tipo do objeto. É sabido que a ordem de passagem dos objetos deve ser sempre alternada. Deste modo, o sistema de controle verifica a ordem dos objetos e caso esteja incorreta, envia um sinal de erro que permanece ligado. Considere que os sinais do sensor estão ligados aos pinos I0000 e I0001 do CLP, e que o sinal de erro é ligado

ao pino O0000. Projete este sistema de controle em *Ladder* utilizando um contador bidirecional.

4) Implemente em *Ladder* a equação: $Y = 2 * X^2 + 3 * Z$, onde $Y = D0000$, $X = D0001$ e $Z = D0002$.

5) Implemente um programa baseado no controlador de temperatura da figura 7 do guia, de forma que o sistema tente manter a temperatura entre 27°C e 30°C. Lembre de fazer a manipulação do dado lido na entrada analógica E0000 baseado na equação (3) do guia. O sistema de controle deve funcionar da seguinte forma: se o sensor estiver medindo uma temperatura menor ou igual a 27°C, o aquecedor (O0000) deve ser ligado; e se a temperatura for maior ou igual a 30°C, o ventilador (O0001) deve ser ativado.

6) Elabore um diagrama *Ladder* para gerer um sinal quadrado com período de 1 segundo.

7) Elabore um diagrama que permita comutar entre a leitura de duas entradas analógicas, E0004 e E0005, e escrever a entrada selecionada na saída analógica S0000.

8) O jogo de “mais ou menos” tem regras bastante simples: é escolhido um número aleatório e o jogador deve adivinhar este número chutando valores; se o valor escolhido pelo jogador for maior que o número correto, ele será avisado que deve chutar um número menor, e o contrário para o caso do número ser menor que o correto. O jogador vence quando o número chutado é o certo. Utilize a entrada analógica E0004 para escolher o número aleatório e as entradas I0000...I0007 para chutar o número em binário. Elabore um diagrama *Ladder* para este jogo. Use a leitura digital em bloco. (Dica: Divida o número lido de E0004 por 16 para ficar na faixa de 0 a 255, que é o máximo valor que as entradas digitais podem atingir).

5.3.2 PRÁTICA EXPERIMENTAL

1) Implemente o resultado obtido do exercício de preparação 1. Compare com o diagrama elétrico

2) Implemente o resultado do exercício de preparação 2. Verifique se o diagrama atende às necessidades de controle.

3) Implemente os resultados obtidos do exercício de preparação 3. Há algum cenário possível em que o sistema falhe?

4) Implemente o resultado do exercício de preparação 6 e verifique o sinal de saída.

Para implementar os programas a seguir, utilize os reostatos do kit que são ligados às entradas E0004 e E0005 do CLP.

5) Implemente e teste o resultado obtido do exercício 5.

6) Implemente o resultado do exercício de preparação 7.

7) Implemente o resultado do exercício de preparação 8.

6 EXPERIMENTO 04 – CONTROLE DE FLUXO DE EXECUÇÃO, PWM, FILTROS E CONTROLADORES PID

6.1 INTRODUÇÃO DO EXPERIMENTO

Neste capítulo, estudaremos formas adicionais de controle que necessitam um maior conhecimento de técnicas como PWM e controle PID. Além disso, será tratado como implementar um filtro de primeira ordem em *Ladder.*

6.2 FUNDAMENTAÇÃO TEÓRICA

6.2.1 CONTROLE DE FLUXO DE EXECUÇÃO

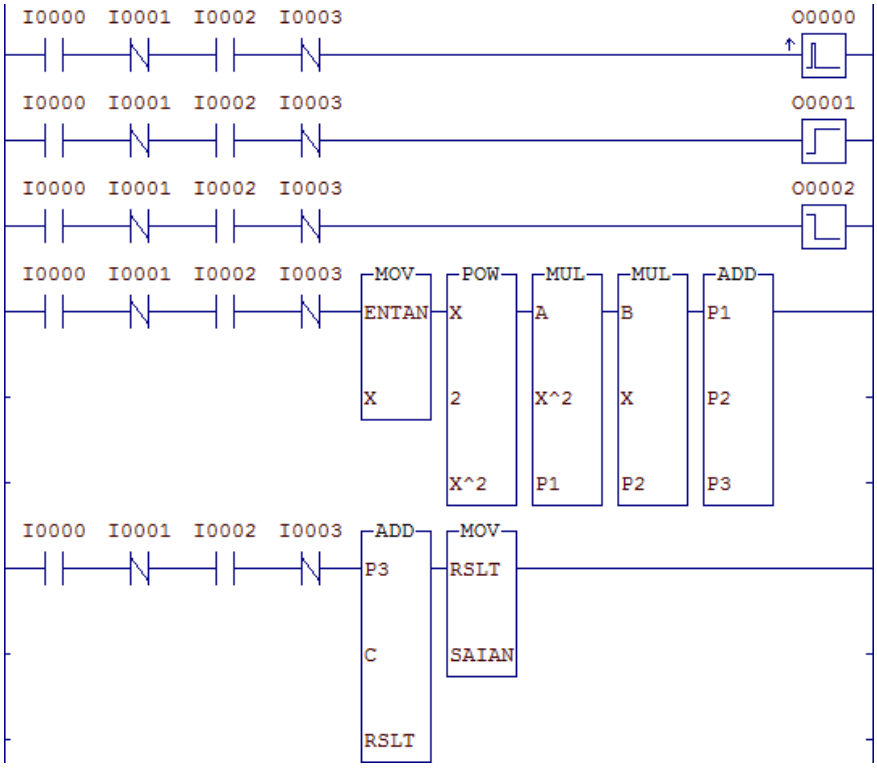
Existem em *Ladder* blocos funcionais que ajudam a acelerar a programação e melhorar a reutilização de código. A seguir, falaremos alguns desses blocos e seu funcionamento e vantagens.

i) Relés Mestres

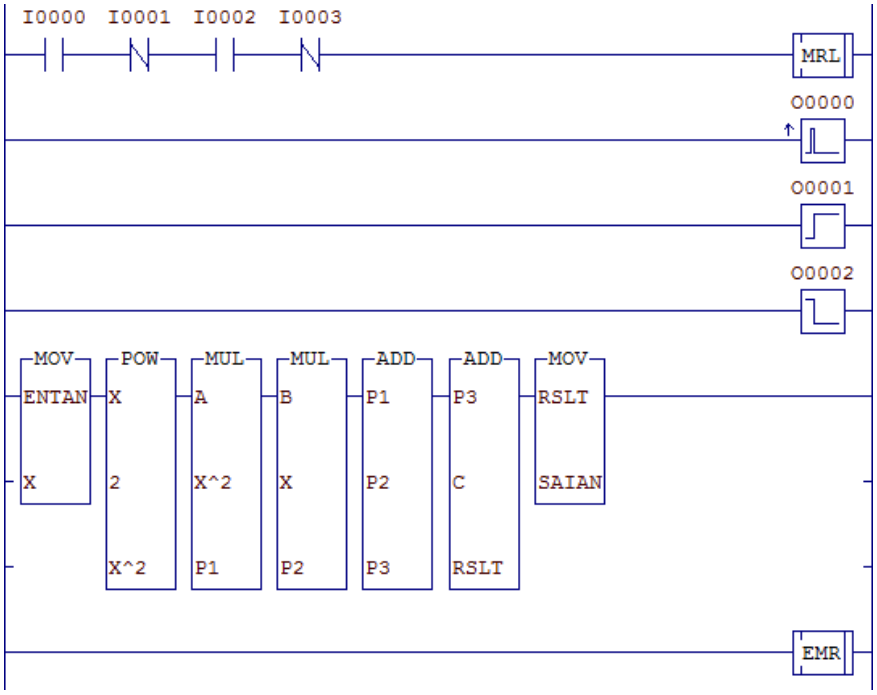
Relés mestres (Tabela 5) são blocos funcionais importantes para uma boa prática de programação *Ladder*. Eles permitem delimitar uma região de linhas, com blocos Início de Relé Mestre e Fim de Relé Mestre, que serão executadas somente se o bloco Início for ativado.

Figura 45. Representação Ladder dos blocos início de relé mestre e fim de relé mestre.

Estes blocos funcionais são extremamente úteis, pois permitem que sejam programadas várias linhas para uma mesma condição lógica. Segue um exemplo comparando dois diagramas, com e sem o uso de relé mestre.



(a)



(b)

Figura 46. Comparação de dois diagramas com a mesma lógica, sem relé mestre (a) e com relé mestre (b).

O programa verifica a lógica entre $I_0 * \overline{I}_1 * I_2 * \overline{I}_3$. Caso verdadeira, um pulso é enviado a saída O0000, O0001 é ativada, O0002 é desativada e a saída analógica SAIAN recebe o resultado da equação de segundo grau definida pelos blocos matemáticos, onde x é a leitura da entrada analógica ENTAN.

Com o uso do Relé Mestre, não foi necessária a definição da lógica entre as chaves em todas as linhas do código, evitando a poluição da interface e facilitando uma posterior interpretação do código, além de evitar o carregamento do CLP com operações repetidas.

No diagrama da Figura 46(b), quando o bloco MRL é ativado, as linhas entre ele e o bloco EMR são executadas de uma vez, sem a necessidade de um novo teste da lógica das entradas digitais a cada linha.

Assim, concluímos que o uso dos blocos de Relé Mestre melhora o desempenho do programa, além de manter uma estética visual melhor.

ii) Blocos de Lógica

Um dos conceitos mais importantes em programação é o de funções. Podem-se definir funções como chamadas a rotinas que recebem certos parâmetros de entrada e retornam valores de acordo com sua lógica. Temos alguns exemplos bem comuns como senos e cossenos.

Podemos fazer algo muito parecido em *Ladder*. Para isso, utilizamos os blocos de lógica (Tabela 5).

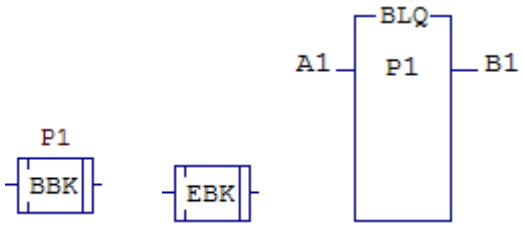


Figura 47. Representação dos blocos Início BBK e Fim EBK de Bloco Lógico, e o Bloco de Chamada de blocos lógicos BLQ.

Os blocos funcionais de início e fim de bloco lógico funcionam de maneira similar aos blocos de relé mestre. Eles delimitam uma região de linhas que serão executadas quando o Bloco de Chamada for ativado.

Os blocos BBK e BLQ recebem apenas um parâmetro, P1, que deve ser do tipo T, que é o identificador de bloco. O identificador do bloco é utilizado para referenciar qual bloco de lógica é utilizado.

Podemos perceber uma pequena diferença entre os blocos de relé e de lógica. O bloco lógico é ativado por um terceiro bloco, BLQ, enquanto o Relé Mestre é executado se o bloco início for ativado. Outro ponto importante é que os blocos lógicos devem ser definidos após o bloco de fim de programa (END). Segue um exemplo onde se implementa uma equação de segundo grau através de blocos lógicos.

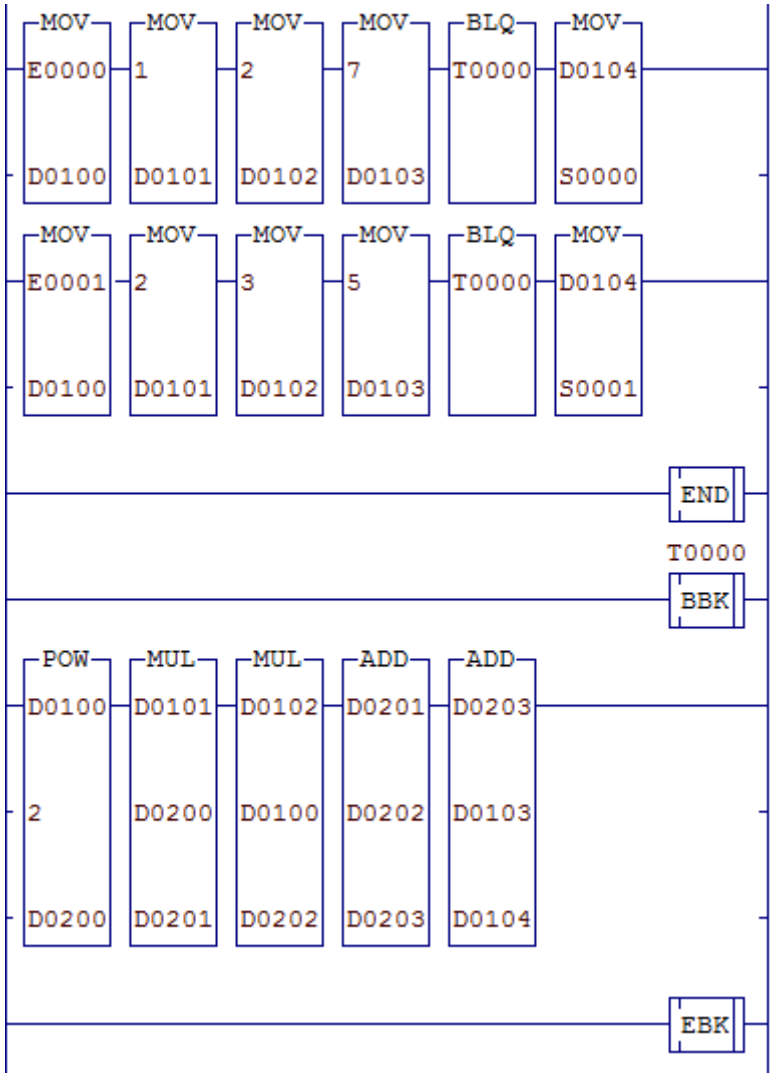


Figura 48. Diagrama da implementação de uma equação utilizando o bloco lógico.

6.2.2 PWM

Quando era preciso controlar a potência fornecida de uma fonte a uma carga, eram-se utilizados potenciômetros em série, que funcionava como um divisor de tensão e consumia parte da potência fornecida pela fonte. A Figura 49 mostra um exemplo de como funcionava o sistema.

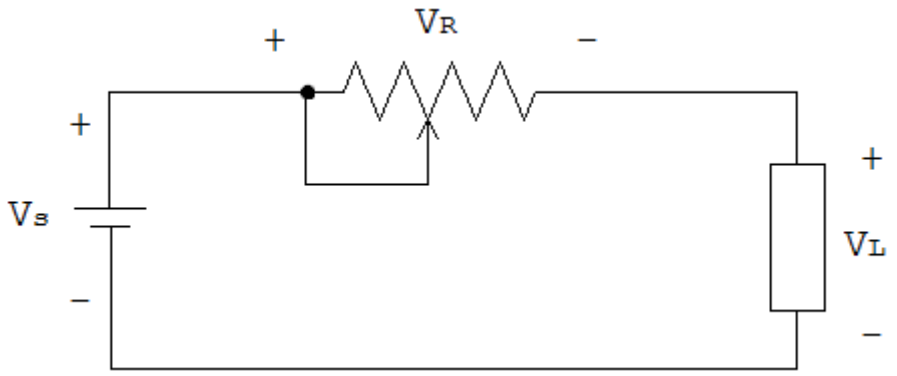


Figura 49. Circuito de uma carga alimentada por uma fonte em série com um potenciômetro.

A utilização deste método implica em uma eficiência energética baixa, pois parte da potência fornecida pela fonte é sempre perdida no potenciômetro em forma de calor.

Com o avanço da eletrônica e da eletrônica de potência, novos métodos foram desenvolvidos para minimizar estas perdas, e obter mais controle sobre a potência fornecida. Um destes métodos é o chaveamento da alimentação da carga utilizando um sinal PWM.

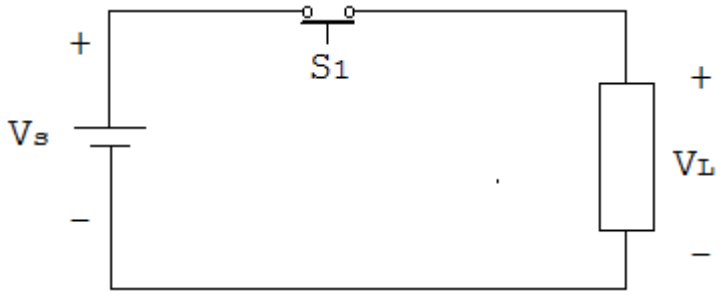


Figura 50. Circuito de uma carga alimentada por uma fonte em série com uma chave.

O método consiste em controlar a chave por um sinal que pode abri-la ou fechá-la, determinando por quanto tempo ela fica aberta ou fechada. Fazendo isso, podemos controlar a potência média fornecida à carga resistiva. Por exemplo, se quisermos que a fonte forneça 70% de sua potência, a chave S1 deve permanecer ligada 70% do tempo e desligada 30%. A Figura 51 mostra o gráfico de potência média e instantânea fornecida para a carga de acordo com o chaveamento de S1.

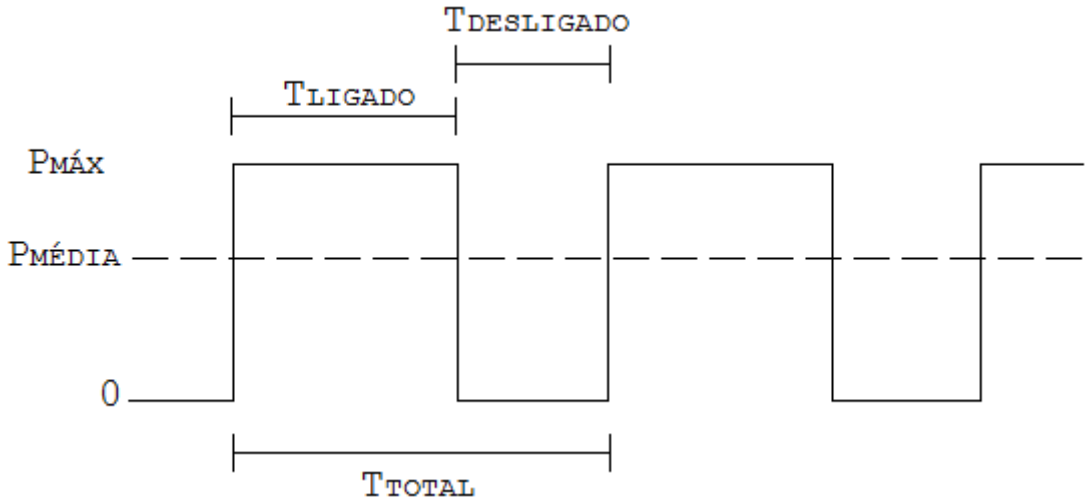


Figura 51. Gráfico da potência média e instantânea fornecida para a carga.

O sinal descrito anteriormente é um sinal PWM. Em outras palavras, PWM é um sinal retangular que tem a largura do pulso variável. A relação entre o tempo que a chave permanece ligada e o período do sinal é chamado de *duty cicle*, que significa razão cíclica de trabalho, e é definida pela equação:

$$Duty = \frac{T_{ON}}{T_{TOTAL}} * 100\% \quad (4)$$

Em *Ladder*, podemos gerar um sinal PWM a partir do bloco funcional gerador de frequência (Tabela 6). Na Figura 52 é mostrada a representação deste bloco.

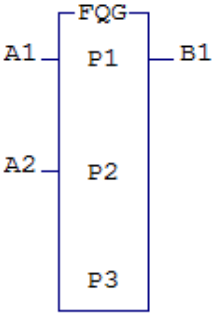


Figura 52. Representação Ladder de um bloco gerador de frequência.

Suas variáveis de entrada e parâmetros funcionam da seguinte forma:

- P1 é do tipo T e indica em qual canal será gerado o sinal;
- P2 é a frequência (em Hertz) do sinal;
- P3 é a razão cíclica. A razão cíclica é definida em décimos de porcentagem (0,1%), ou seja, se o sinal PWM tem razão cíclica de 80%, P3 deve ser igual a 800;

- As entradas A1 e A2 ativam ou bloqueiam o gerador de frequência. Para que o bloco seja ativado, é necessário que as duas entradas estejam ativados. Na prática, podemos curto circuitar uma das entradas e utilizar uma chave na outra para habilitar ou desabilitar o bloco. Ou simplesmente curto circuitar ambas as entradas e controlar apenas o seu *duty cycle*.

A seguir temos um exemplo de diagrama *Ladder* que permite a permutação da razão cíclica entre 25% e 75% do bloco gerador de frequência. Considere K0000 = 250, K0001 = 750 e M0000 = 2000. Ou seja, o gerador de frequência opera a 2kHz (período de 0,5 ms) com razão cíclica de 25% ou 75%, dependendo das entradas I0000 e I0001.

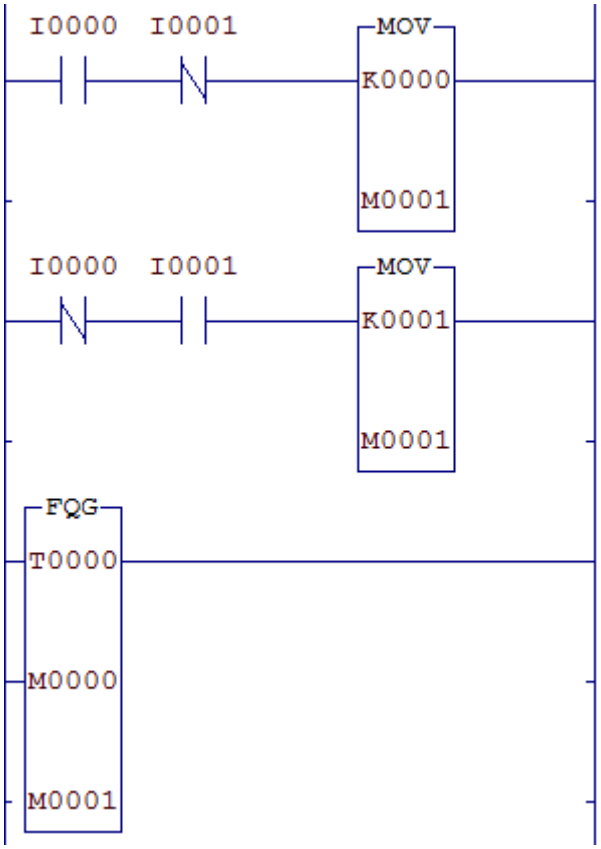


Figura 53. Diagrama Ladder de um gerador de frequência com razão cíclica mutável.

6.2.3 FILTROS

Quando trabalhamos com sinais analógicos, muitas vezes esses sinais estão propícios à distúrbios e ruídos, fazendo com que o sinal lido sofra interferências e divirja do valor real. Para isso, utilizamos filtros para permitir a passagem das frequências desejadas, e atenuar ou anular as demais.

Circuitos RC, como na Figura 54, podem operar como filtro passa-baixa ou passa-alta. Para o caso de passa-baixa, o sinal de saída deve ser lido sobre o capacitor, então, pode-se deduzir a equação do filtro a partir da relação entre o sinal de saída, Vc, e o sinal de entrada, Vs.

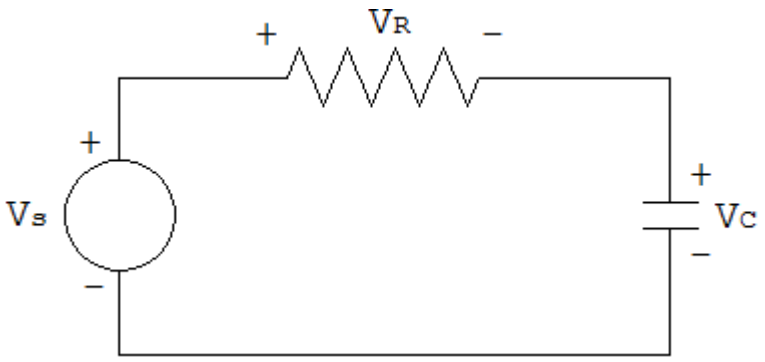


Figura 54. Circuito RC.

Para o caso do filtro passa baixa temos a seguinte equação:

$$V_{out}(k) = V_{in}(k) * (1 - \alpha) + V_{out}(k-1) * \alpha \text{ onde } \alpha = \frac{\tau}{\Delta T + \tau} \quad (5)$$

Para o caso de passa-alta, o sinal de saída deve ser lido pelo resistor. Então, a equação do filtro passa-alta é dada por:

$$V_{out}(k) = [V_{in}(k) - V_{in}(k-1)] * \alpha + V_{out}(k-1) * \alpha \text{ onde } \alpha = \frac{\tau}{\Delta T + \tau} \quad (6)$$

Para ambos os casos, τ é a constante de tempo e através dela podemos calcular a frequência de corte do filtro pela equação:

$$f = \frac{1}{2\pi * \tau} \quad (7)$$

Assim, a maior dificuldade para implementação em *Ladder* destes filtros se resume apenas em definir a variação de tempo, ΔT. Há duas maneiras de se definir esse parâmetro: utilizando-se blocos temporizadores ou estimando o tempo de varredura do sistema.

Como já foi visto, os blocos temporizadores tem passo mínimo de 10ms, o que equivale a uma frequência máxima de 100Hz. Então, segundo o teorema da amostragem, a frequência máxima para o sinal de entrada deve ser de 50Hz, que pode ser baixa, dependendo da aplicação.

O outro método é através da estimação do tempo de varredura do sistema, ou seja, quanto tempo é gasto para o CLP executar o programa novamente. Pode-se estimar o tempo de varredura de várias maneiras, inclusive, alguns editores *Ladder* dão a estimativa de quanto tempo é necessário para o programa ser executado. Com este método, podemos trabalhar com maiores frequências, mas peca na precisão do cálculo da variação de tempo. A seguir temos a implementação deste segundo método para o caso de um filtro passa-baixa, e o algoritmo utilizado.

```
CONT = CONT+1
IN = ENTAN

SE TEM<=0 ENTÃO
  DT = 0,02/CONT
  CONT = 0;
  SOMA = TAU+DT
  ALFA = TAU/SOMA
  1-ALF = 1-ALFA
  RESET TIMER
FIM

P1 = 1-ALF*IN
P2 = ALFA*OUT
OUT = P1+P2
SAIAN = OUT
```

Figura 55. Algoritmo para o filtro passa-baixa.

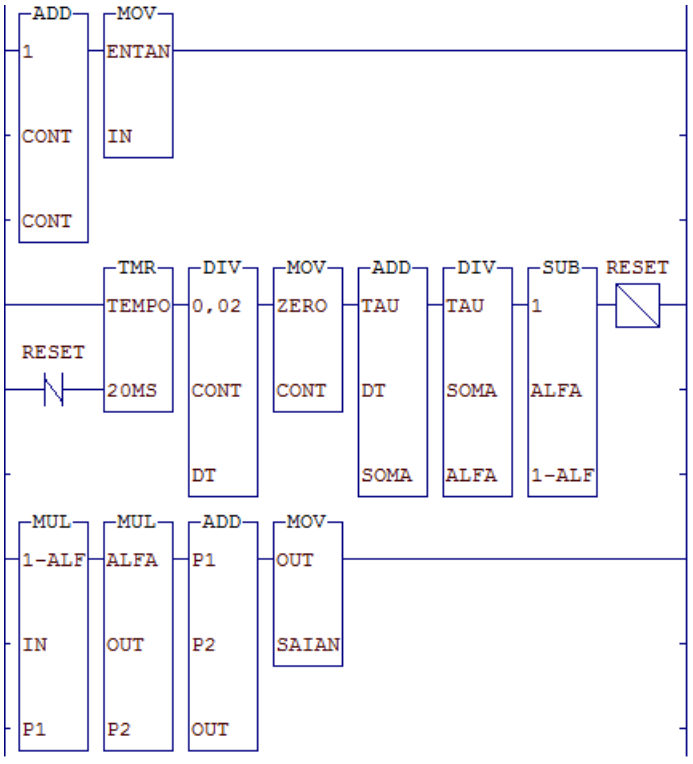


Figura 56. Diagrama Ladder do filtro passa-baixa com estimação de tempo de varredura.

Algumas observações importantes podem ser feitas em relação a esse programa:

- O valor de 20ms para o temporizador foi escolhido arbitrariamente. Este valor pode ser modificado de acordo com a necessidade, lembrando-se de modificar também o cálculo de DT.
- O valor de DT é estimado com uma média simples de valores passados e não garante que as próximas execuções terão essa média. O cálculo pode ser modificado para melhorar a estimativa, utilizando-se médias de valores passados de ΔT ou estimando de valores futuros, por exemplo.
- Alguns CLPs possuem filtros nativos em suas entradas e saídas analógicas que funcional de variadas formas.

6.2.4 CONTROLADORES PID

Controlador proporcional integral derivativo é uma técnica de controle de processos que une as ações derivativa, integral e proporcional, fazendo assim com que o sinal de erro seja minimizado pela ação proporcional, zerado pela ação integral e obtido com uma velocidade antecipada pela ação derivativa.

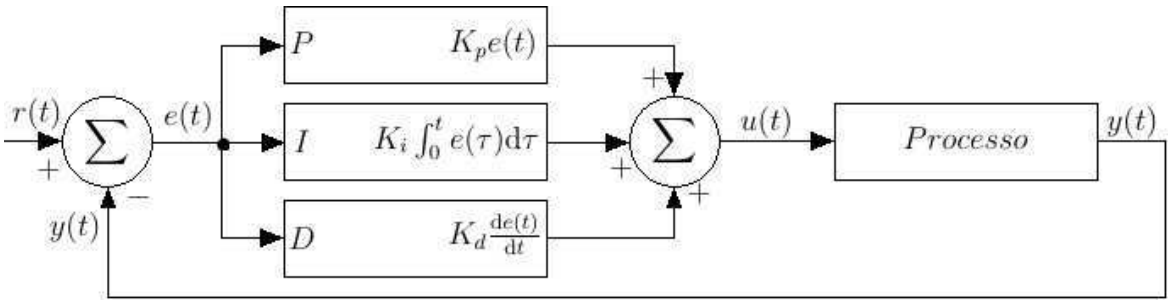


Figura 57. Diagrama de blocos de um controlador PID com realimentação.

A equação que representa o controlador é dada por:

$$u(t) = K[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}] \quad (8)$$

Discretizando a equação anterior, temos:

$$u(k) = Ke(k) + \left[I(k-1) + \frac{K\Delta T}{T_i}e(k)\right] + \frac{KT_d}{\Delta T}[e(k) - e(k-1)] \quad (8)$$

Para implementar esta equação em *Ladder* podemos fazer de duas maneiras: implementar manualmente ou utilizando o bloco funcional PID.

Vamos esquecer-nos do método manual devido a sua complexidade e a falta de um supervisório para acompanha a evolução do sistema. Segue então o bloco funcional que representa o PID (Tabela 6) na Figura 58.

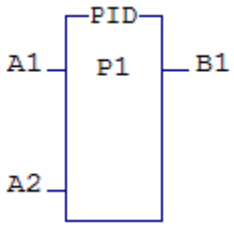


Figura 58. Representação Ladder do bloco PID.

Este bloco tem apenas um parâmetro do tipo T que é o identificador do bloco. As entradas habilitam ou desabilitam o funcionamento do bloco. A saída B1 é ativada quando o bloco funcional está operacional.

Para a programação do bloco PID, cada editor *Ladder* implementa uma forma de edição diferente. Apesar disto, o bloco PID tem sempre os mesmos parâmetros. São estes:

- PV (Process Variable): Posição na memória onde é guardado a variável do processo, y(t);
- SP (Set Point): Posição na memória onde é guardada a entrada de referência, r(t);
- OV (Output Variable): Posição na memória onde é guardado a saída do controlador, u(t);
- GP: Posição na memória onde é definido o ganho proporcional, K;
- TI: Posição na memória onde é definido o tempo integral, Ti;
- TD: Posição na memória onde é definido o tempo derivativo, Td;
- OV Máx, OV Min e OV Ini: Valor máximo, mínimo e inicial que pode ser assumido por OV, em %.
- Para editar essas varáveis que o bloco PID controla, seleciona-se o bloco e então pressiona-se Ctrl+T**.**

Podemos exemplificar o uso do bloco PID na variação de velocidade de um motor para controlar a vazão de ar em uma tubulação. Dado um determinado valor de SP, observa-se PV (no caso, a vazão atual na tubulação lida, por exemplo, com um medidor de vazão) e a partir dessa diferença (PV – SP) temos a saída OV, que "mandará" o motor aumentar ou diminuir a velocidade, levando em consideração os parâmetros GP, TI e TD.

Determinado $GP = 2$, $TI = 10$, $TD = 0$ e $SP = 3000$, e o sensor de vazão medindo $2000\,m^3/h$, podemos detalhar a ação de cada parâmetro:

1) GP provoca um ganho na saída de acordo com o valor dado. Um degrau de 10% provoca, para $GP = 2$, um degrau de 20% em OV.
2) A ação de TI no bloco PID é determinar o intervalo de tempo em que OV responderá a uma variação de PV em função do erro. Analisando somente a ação de correção integral, quanto maior o valor de TI maior será o tempo em que OV gastará para atingir um mesmo valor de correção. A saída OV aumentará como uma rampa em um intervalo de 10 segundos.
3) TD busca acelerar o processo, ou seja, busca adiantar uma correção na saída após uma mudança de PV. Quando o processo apresenta um atraso grande, isto é, uma variação da saída não provoca uma mudança imediata em PV. Para nosso exemplo, consideramos esse valor igual à zero.
4) Lido o valor de PV e feita a diferença ($2000 - 3000 = -1000$), temos que o valor de saída vai ser negativo, e isso fará com que o motor gire mais rápido para atingir a vazão definida como referência. Levemos em consideração que o processo é inverso, se o valor for negativo, quer dizer que a rotação do motor é menor que a necessária e, portanto, a corrente aumenta para atingir o SP definido. Caso o valor for positivo, isso significa que a potência do motor está maior que a necessária, e então ocorre uma diminuição da corrente.

Uma observação importante é que as entradas e saídas deste bloco não estão em unidades de engenharia. São programados de acordo com a resolução de *n bits* das portas analógicas, geralmente *n = 12 bits*, ou seja, estas variáveis assumem apenas os valores inteiros de 0 a $2^n - 1$. A conversão destes valores para unidades de engenharia deve ser feita externamente ao bloco.

Alguns editores também incluem sistemas supervisórios para os blocos PID, que é o caso do SPDSW, utilizado nesse laboratório. Assim é possível monitorar graficamente o sistema ao controlador, facilitando a sintonização do mesmo. Para isso, basta ir na janela principal do programa, clicar em *Supervisão → Supervisão PID*.

6.3 GUIA EXPERIMENTAL

6.3.1 EXERCÍCIOS DE PREPARAÇÃO

1) Elabore um diagrama *Ladder* para gerar um sinal PWM com o *duty cycle* que varia linearmente com a entrada E0004. Quando E0004 = 4095, o *duty cycle* deve ser de 99,9%.

2) No controle de velocidade de um motor CC, usa-se um sinal PWM para controlar a potência fornecida para este. Um sensor mede a velocidade deste motor e envia um sinal analógico à entrada E0004 do CLP. Este sensor é calibrado de forma que o valor lido já é dado em RPM, ou seja, se o valor lido for E0004 = 1200 significa que o motor está a 1200rpm. Elabore um diagrama que controle o *duty cycle* do PWM gerado de modo que, se o motor estiver acima da velocidade desejada, o valor do ciclo de trabalho deve ser diminuído linearmente até que o motor esteja na velocidade desejada. O mesmo se o motor estiver abaixo da velocidade desejada.

3) Elabore o diagrama *Ladder* de um filtro passa-alta com frequência de corte de 60Hz, utilizando o método de variação de tempo constante.

4) Elabore o diagrama *Ladder* de um filtro passa-alta com frequência de corte de 60Hz, utilizando o método de variação de tempo mutável.

6.3.2 PRÁTICA EXPERIMENTAL

1) Implemente os circuitos da figura 2 e 4 deste guia, afim de obter prática no uso de Relés Mestres e Blocos Lógicos. Os valores das constantes da equação de segundo grau podem ser escolhidos arbitrariamente. Lembrando que em alguns blocos estão escrito os valores, mas os mesmos devem ser armazenados em constantes no programa (não pode colocar diretamente o valor desejado no bloco matemático).

2) Implemente e teste o resultado obtido do exercício de preparação 1.

3) Implemente e teste o resultado do exercício de preparação 2. Considere que a velocidade desejada seja de 1800rpm.

4) Implemente os resultados dos exercícios 3 e 4, com E0004 sendo variável de entrada e o resultado sendo escrito em S0000? Existe diferença entre os dois filtros?

7 MÓDULOS EXPERIMENTAIS E APLICAÇÕES

Tendo conhecimento de diversas funcionalidades presentes no CLP, vamos agora explorá-las utilizando módulos experimentais.

7.1 MÓDULO DE CONTROLE DE TEMPERATURA CI - 53003

7.1.1 APRESENTAÇÃO DO MÓDULO

O módulo de temperatura CI – 53003 foi desenvolvido para fornecer os circuitos e elementos finais necessárias para sistemas de controle de temperatura. Ele é equipado com um transdutor de temperatura de precisão e circuito de condicionamento, uma carga DC, uma carga AC e um cooler de resfriamento. As cargas DC ou AC podem ser usadas para o aquecimento em uma aplicação prática do módulo. Com este módulo, podemos facilmente construir um sistema de controle de temperatura seja de malha aberta ou fechada.

O módulo apresenta as seguintes partes:

- Unidade Principal

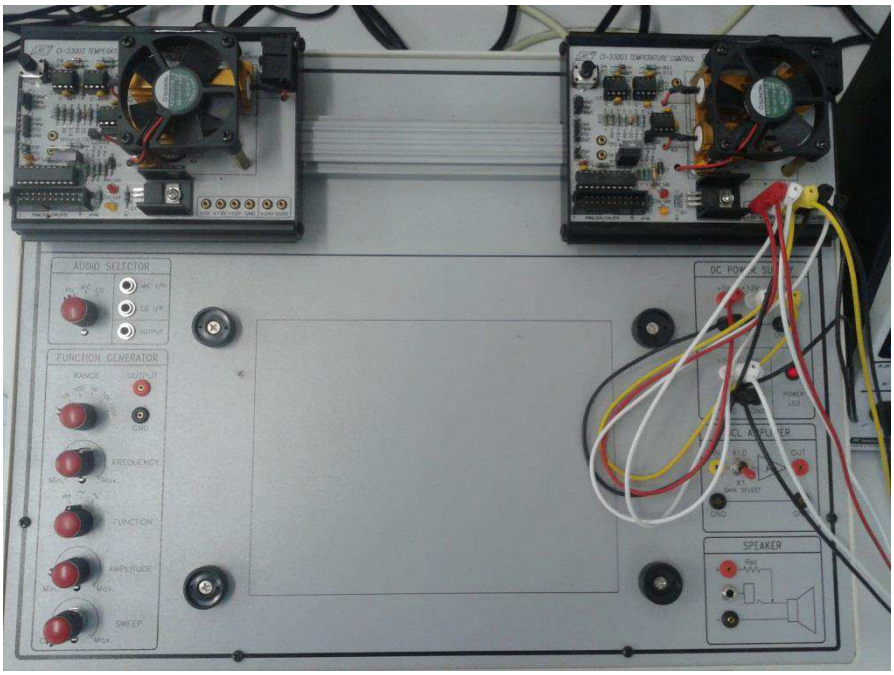


Figura 59. Unidade Principal onde é acoplado o módulo de temperatura.

Essa parte do módulo apresenta: um gerador de função, que varia de 1 Hz até 100 kHz, possui uma saída que pode ser de onda quadrada, triangular, senoidal e de varredura e apresenta uma variação de amplitude; um amplificador de áudio; uma saída para alto-falante; e uma saída de tensão DC.

- Controle de Temperatura

Figura 60. Módulo de temperatura CI – 53003.

Essa parte não é fixa, e caso necessário, pode ser trocado por outros módulos adaptáveis ao kit do fabricante. O módulo oferece as seguintes funcionalidades para o experimento que desejamos:

- PWM para controle de condução de energia (24V/1A) e carga (resistência) de aquecimento.
- Sensor AD590 para detecção e medição de temperatura (100mV/1°C).
- Ventilador DC 12V/1.7W para controle de temperatura.

7.1.2 PROPOSTA DE UM SISTEMA DE CONTROLE EM *LADDER*

Tomando conhecimento do módulo apresentado e da linguagem de programção *Ladder*, vamos agora desenvolver um sistema de controle de temperatura, utilizando blocos PID e PWM do software SPDSW, fornecido pelo fabricando do CLP eZAP910.

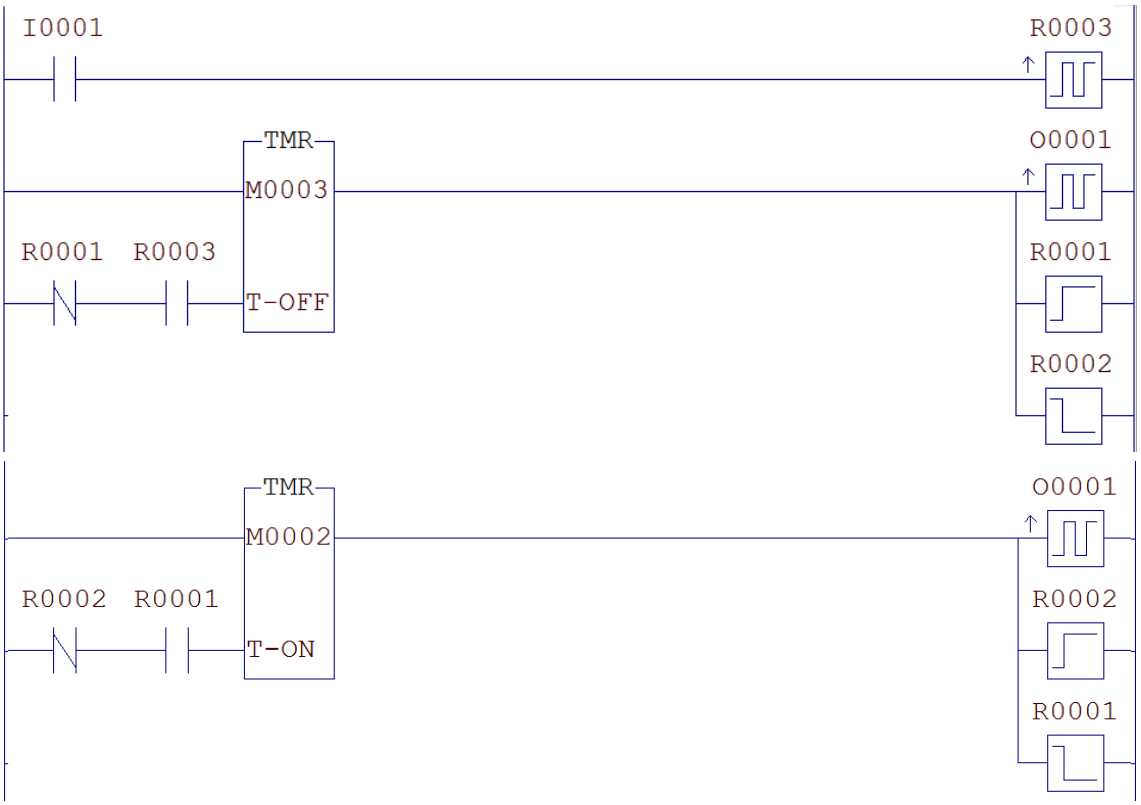


Figura 61. Código do Controle de Temperatura (Parte 1).

Primeiramente utilizamos uma chave que aciona o ventilador responsável pelo resfriamento da carga aquecida. A lógica funciona da seguinte forma: os temporizadores ativam e desativam a saída O0001 utilizando temporizadores, que estão diretamente ligado ao valor da velocidade do ventilador que o usuário informa através da interface homem-máquina disponível no CLP. Posteriormente será explicado como se dá o cálculo dos valores de T-OFF e T-ON.

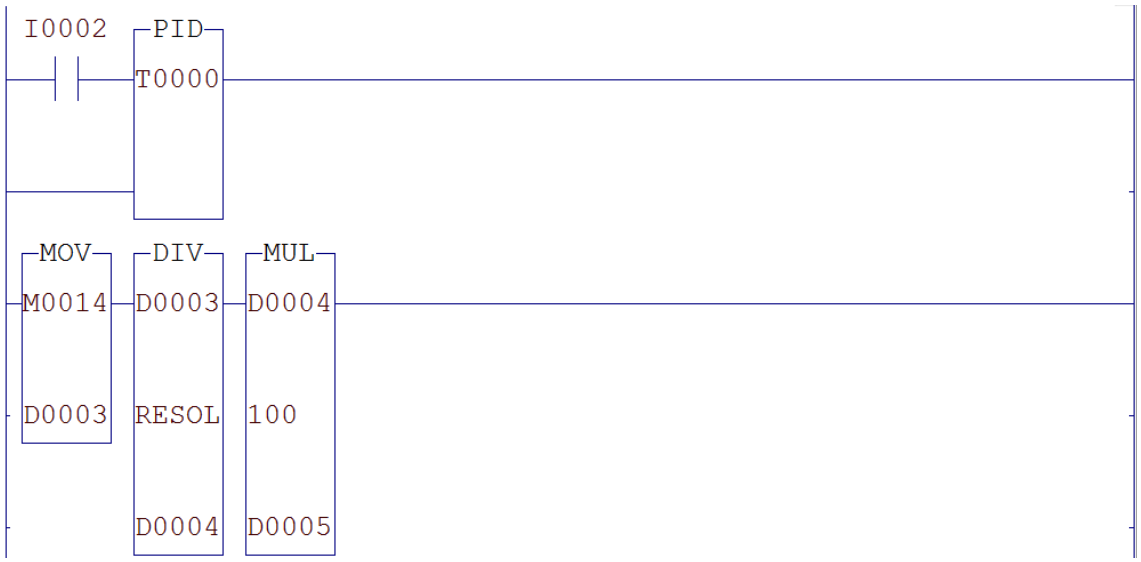


Figura 62. Código do Controle de Temperatura (Parte 2).

Nessa segunda parte do código, inserimos um bloco PID, que é ativado pela chave I0002. Os valores de PV, SP e OV do bloco PID estão respectivamente nas posições de memória M0005, M0015, M0014. A varável de saída passa por um tratamento matemático onde dividimos o valor de saída do bloco pela resolução, e multiplicamos por 100 para obter esse valor em porcentagem.

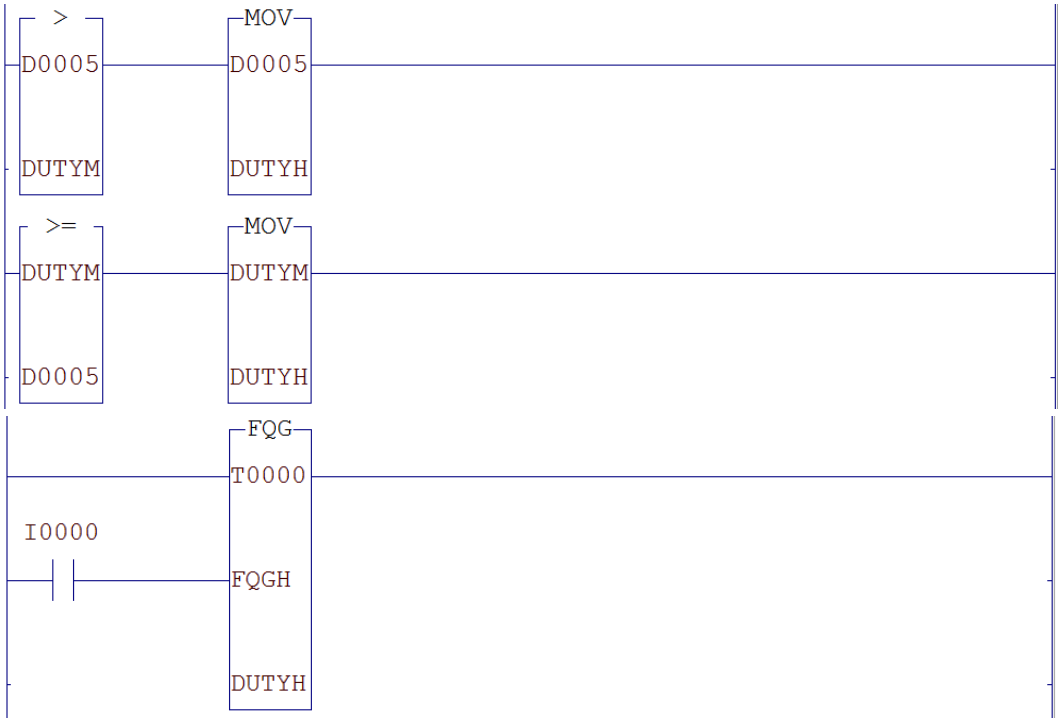


Figura 63. Código do Controle de Temperatura (Parte 3).

Dado o valor em porcentagem da saída do bloco PID, esse será comparado com uma constante denominada DUTYMIN. Essa constante foi definida pela análise, via osciloscópio, do sinal de saída do PWM do CLP, onde para uma frequência de 500 Hz, o valor mínimo do *duty cycle* é de 4%, devido ao comportamento do bloco. Com o aumento da frequência, havia o aumento desse valor mínimo.

Caso o valor de saída seja maior que o valor mínimo, o valor de saída é atribuído ao bloco gerador de frequência. Caso contrário, o valor mínimo é atribuído.

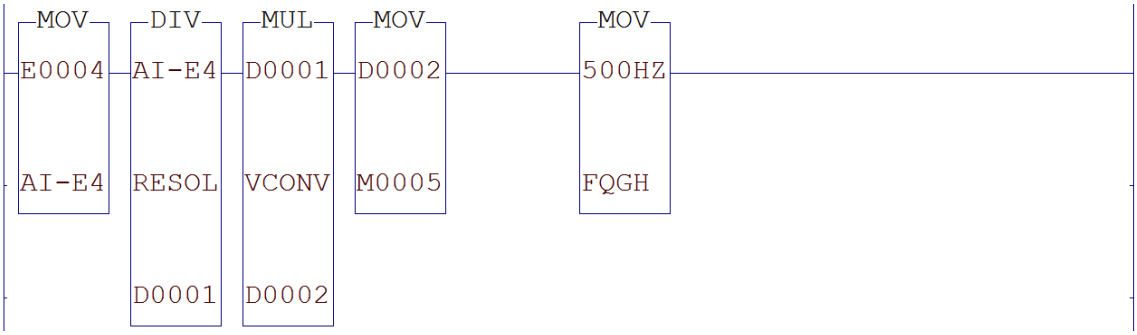


Figura 64. Código do Controle de Temperatura (Parte 4).

Nesse pedaço do código, fazemos a leitura da entrada analógica E0004, que informa o valor lido pelo sensor AD590 do módulo. É feito novamente a divisão pela resolução e multiplicamos por uma constante, que foi definida pela aproximação a um sistema linear através dos resultados obtidos. Esse valor de constante é igual a 520.

Podemos ver também, que atribuímos o valor de 500 Hz para a frequência do bloco FQG, mostrado na Figura 65.

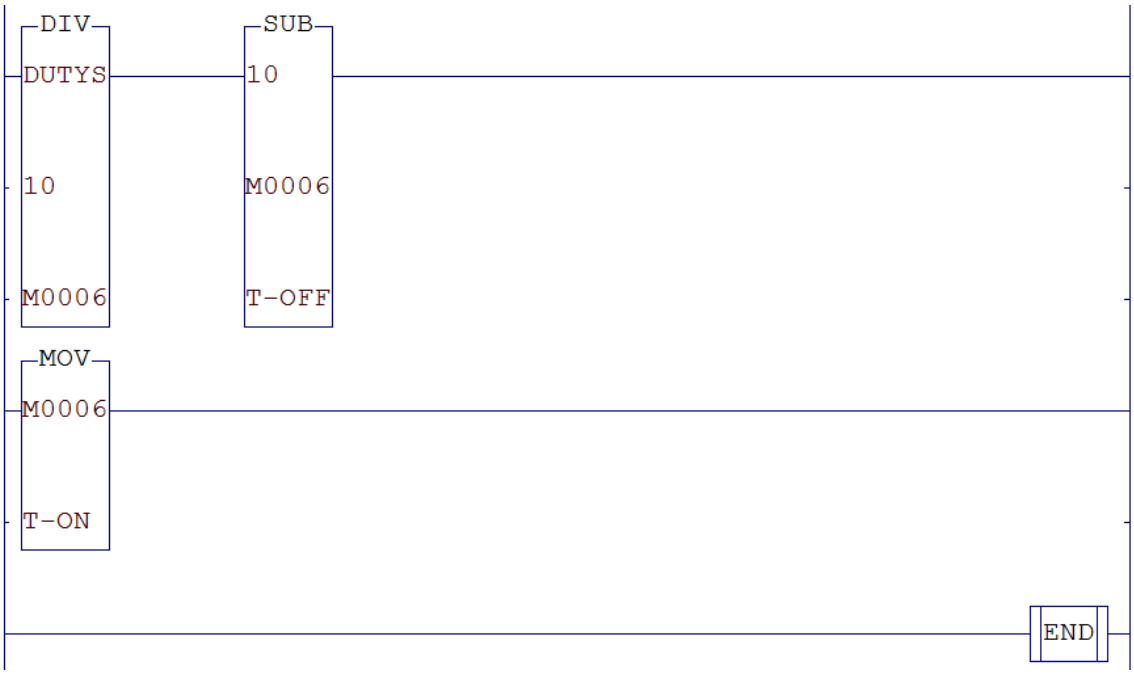


Figura 65. Código do Controle de Temperatura (Parte 5).

E finalizando o programa, definimos agora os valores de T-OFF e T-ON, apresentados na Figura 61. Dado o valor da velocidade do ventilador (DUTYS), a qual é informada pelo usuário e é definida entre 0 e 100, dividimos esse número por 10 e atribuímos a posição de memória M0006. Logicamente, esse valor varia de 0 a 10, dependendo do que foi informado pelo usuário. Ao subtrairmos 10 por esse valor, temos o tempo que a saída O0001 ficará desligada. O valor de M0006 será o valor em que O0001 ficará ligada. De forma resumida, quanto maior a velocidade do ventilador, mais rápido o LED da saída O0001 piscará e vice versa.

Para atuação do código escrito, definimos valores arbitrários de Kp, Ti e Td, que estão alocados nas posições de memória D0010, D0011 e D0012 respectivamente, e definimos um valor de referência para o controlador. Com isso, ativamos as chaves I0000 e I0001 para ativar o bloco FQG e o bloco PID do programa. Temos então que o valor de saída do controlador varia, e, portanto, ativa o PWM de forma a aquecer o sistema.

7.1.3 COMUNICAÇÃO OPC COM O MATLAB®

OPC ou OLE (Object Linking and Embedding) for Process Control, consiste na especificação de um conjunto de padrões de interface, facilitando a interoperacionalidade em aplicações de Controle de Processo e Automação da Manufatura. Em outras palavras, é feita a comunicação de dados da planta em tempo real com os dispositivos de controle de diferentes fabricantes.

Para isso, utilizamos outro programa da HI-Tecnologia, chamado HT1 Power Tool, que tem como função criar um servidor OPC. O passo a passo para criação do servidor se encontra logo abaixo.

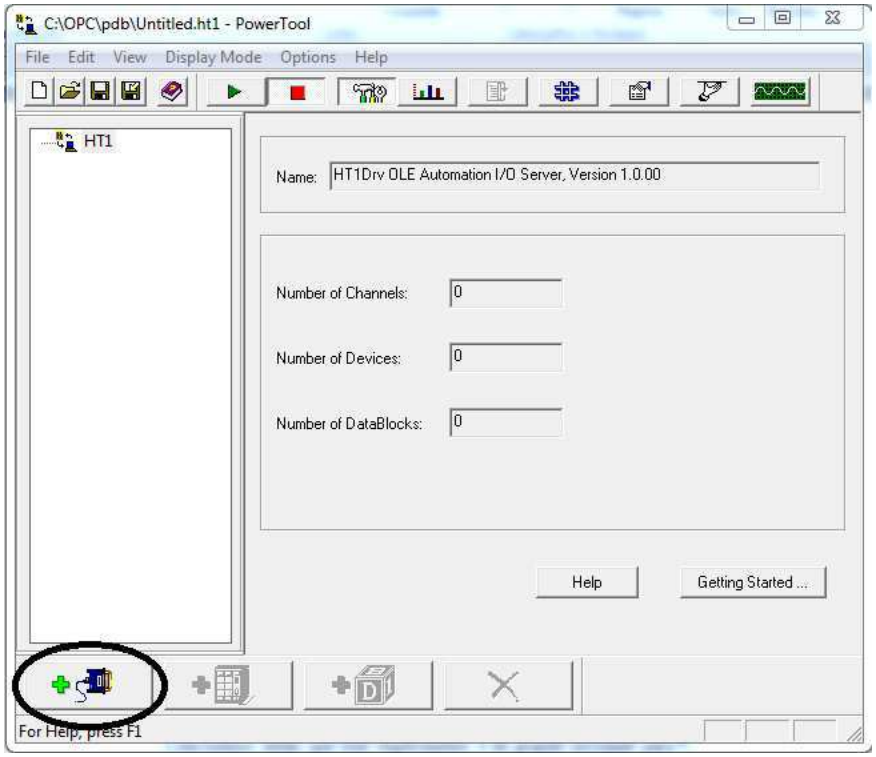


Figura 66. Configuração do servidor OPC (Parte 1).

Depois de inicializado e conectado a um servidor local, temos a tela da Figura 66. O primeiro passo é criar um canal. Clique no botão destacado na figura e selecione o nome do canal e a descrição do mesmo. Depois, ative o canal marcando a opção *Enable*. Por último, aperte no botão para adicionar um dispositivo.

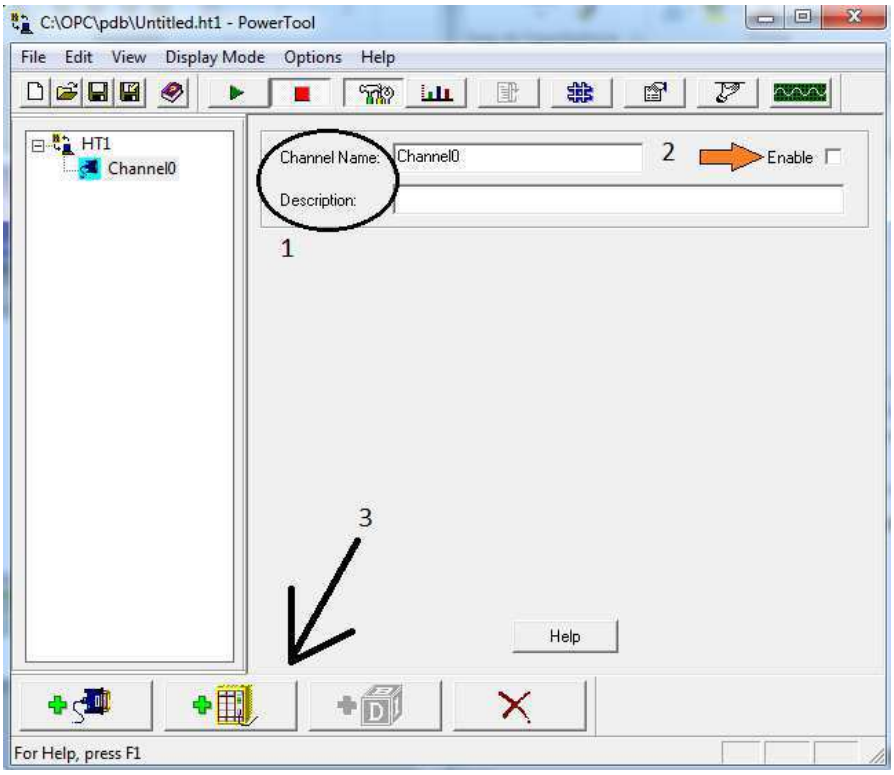


Figura 67. Configuração do servidor OPC (Parte 2).

Na janela para configuração do dispositivo adicionado, você deve novamente selecionar um nome e descrição, e ativar o bloco. Determine também o endereço de IP do CLP. Ao finalizar, clique no botão de adicionar bloco de dados.

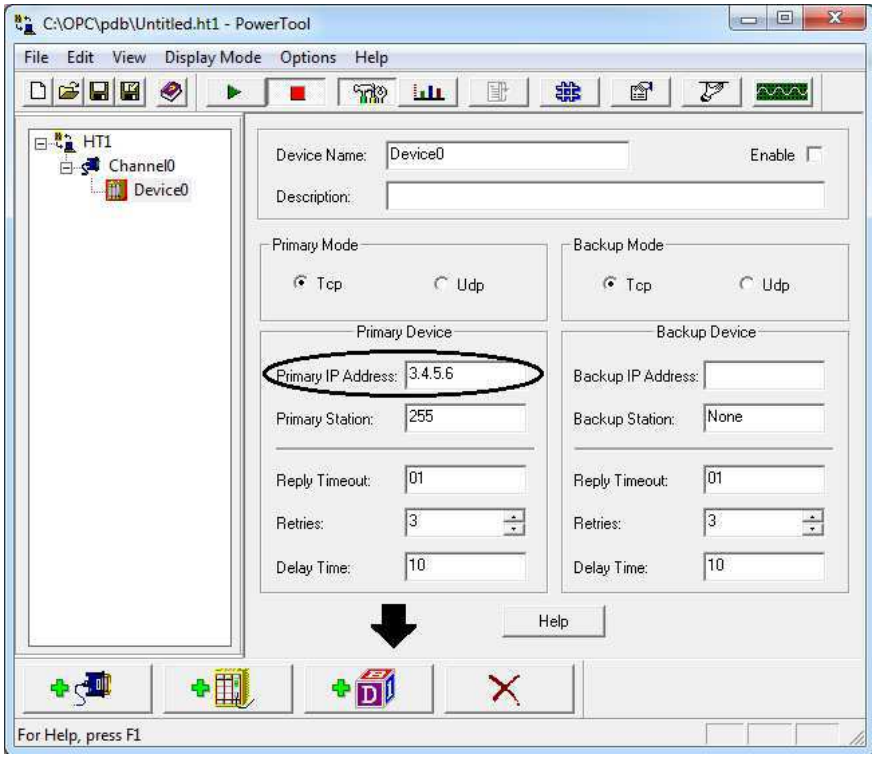


Figura 68. Configuração do servidor OPC (Parte 3).

Nessa última etapa, definimos o endereço de memória que queremos visualizar e salvar no servidor. Você pode também selecionar um vetor de endereços, informando endereço inicial e final. Configure outro parâmetro importante que é o tempo de acesso conforme a sua necessidade. Depois de definido todos os blocos de dado da sua escolha, aperte o botão *Play* para ativar a leitura de dados do CLP. Caso queira excluir algum bloco de dados, dispositivo ou canal, selecione-o na janela da esquerda e clique no botão *Delete*.

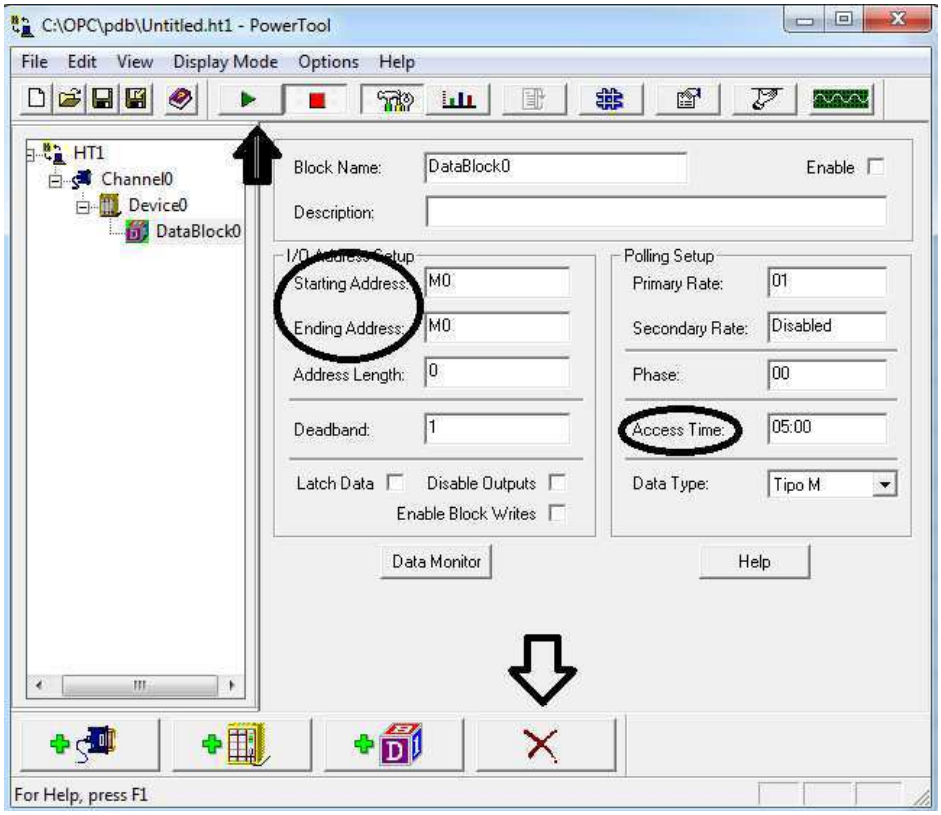


Figura 69. Configuração do servidor OPC (Parte 4).

Para o experimento utilizando o módulo de temperatura, foi feito o seguinte servidor OPC:

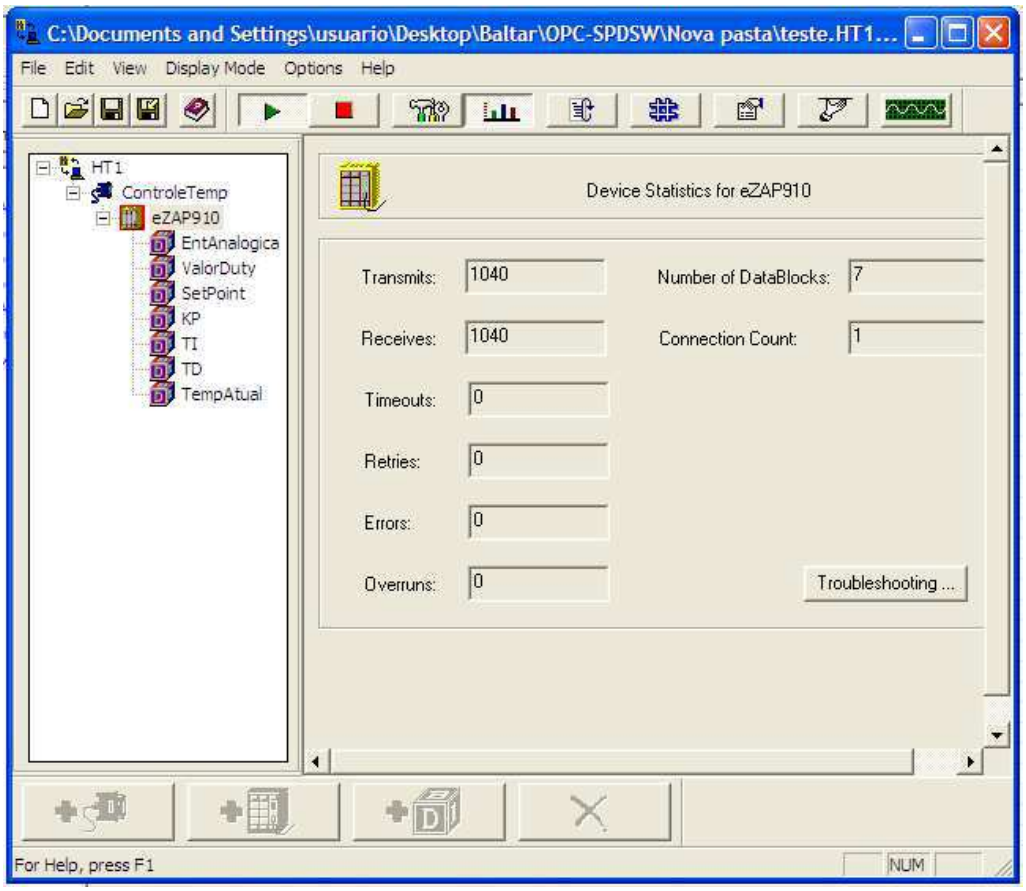


Figura 70. Servidor OPC para o módulo de controle de temperatura.

Utilizamos o MatLab® para ler, escrever e supervisionar esse sistema, através da ferramenta *OPCTool*. Selecionamos o servidor criado pelo HT1 Power Tool, e depois selecionamos as varáveis disponibilizadas pelo servidor que queremos trabalhar. No caso da Figura 71, escolhemos o valor de referência do PI, o valor da temperatura lida pelo módulo e o valor de *Duty Cycle*. Podemos ver que o valor de *Dutycycle* é mínimo pois a temperatura da carga é maior que o valor de referência (M15). Na figura 72, modificamos o valor do *SetPoint* para 40, utilizando a opção de escrita do *OPCTool*. Com isso, o valor do *Duty Cycle* aumenta, a fim de fornecer uma tensão média maior para a carga.

Item ID	Active	Value	Quality	Timestamp	Write Value
eZAP910:D2	☑	39.619053	Good: Non-specific	11h33min4s GM...	
eZAP910:M15	☑	38	Good: Non-specific	11h32min16s G...	
eZAP910:M9	☑	4	Good: Non-specific	11h32min11s G...	

Figura 71. Variáveis selecionadas do servidor OPC.

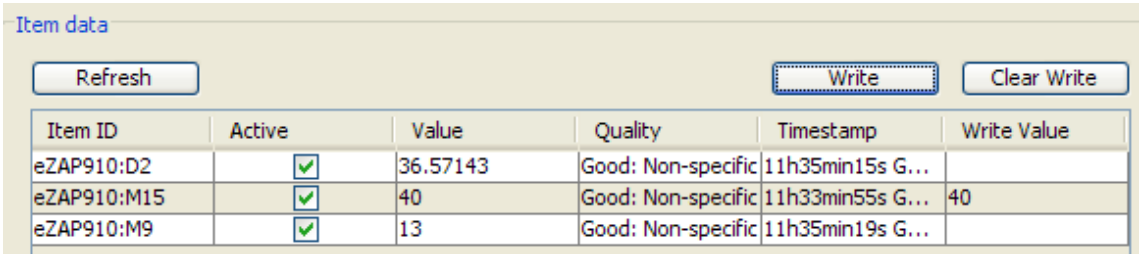


Figura 72. Mudança do valor de referência utilizando o OPCTool.

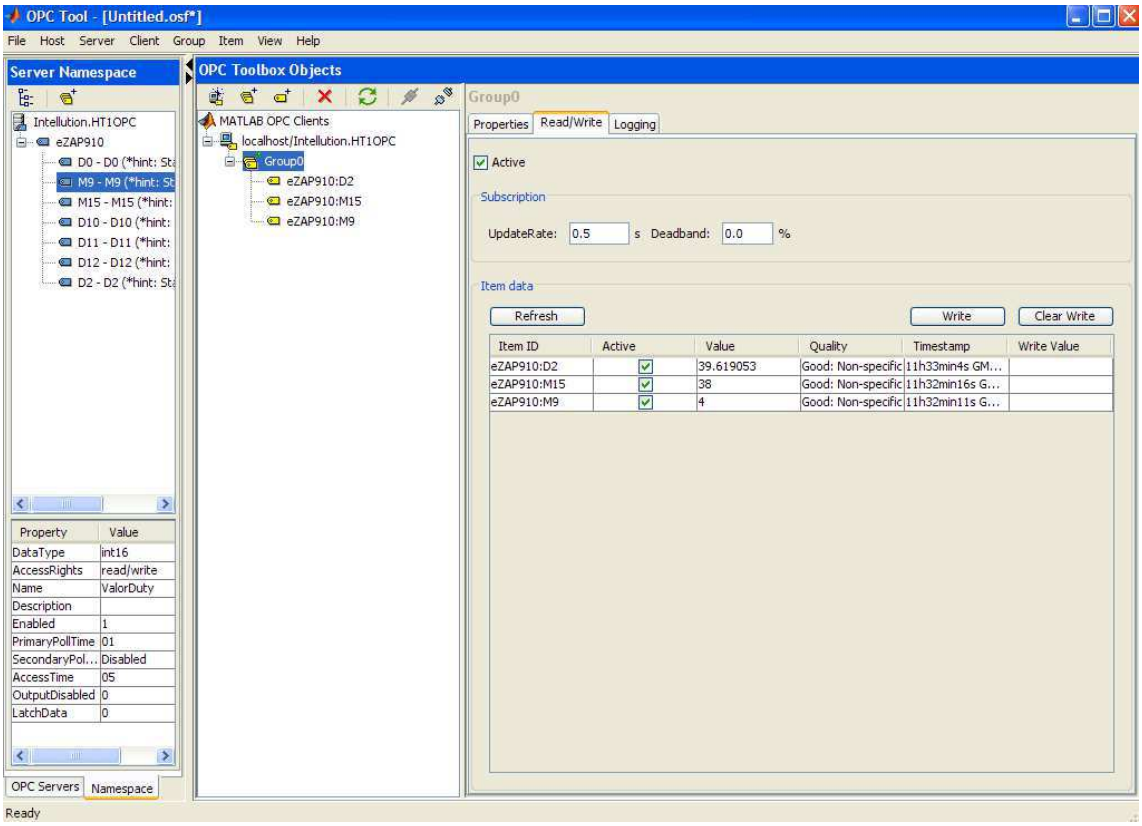


Figura 73. Janela de apresentação do OPCTool.

7.1.4 Conclusão do Experimento

O sistema funciona perfeitamente, mas como o resfriamento é dado apenas pelo cooler, não temos como controlar perfeitamente o mesmo. Além disso, a inércia térmica do aquecimento da carga é alta, e atrapalha ainda mais o controle.

Concluímos então que esse experimento é de grande utilidade para o entendimento de do bloco PID, assim também como o uso das ferramentas de comunicação via OPC. Caso o controle do processo fosse perfeito, poderíamos utilizar outras ferramentas do MatLab® (Simulink, por exemplo) para desenvolver atividades como controle preditivo, resposta ao degrau, entre outros.

8 ATIVIDADES EXPERIMENTAIS UTILIZANDO O MÓDULO DE PELTIER E O SECADOR DE GRÃOS

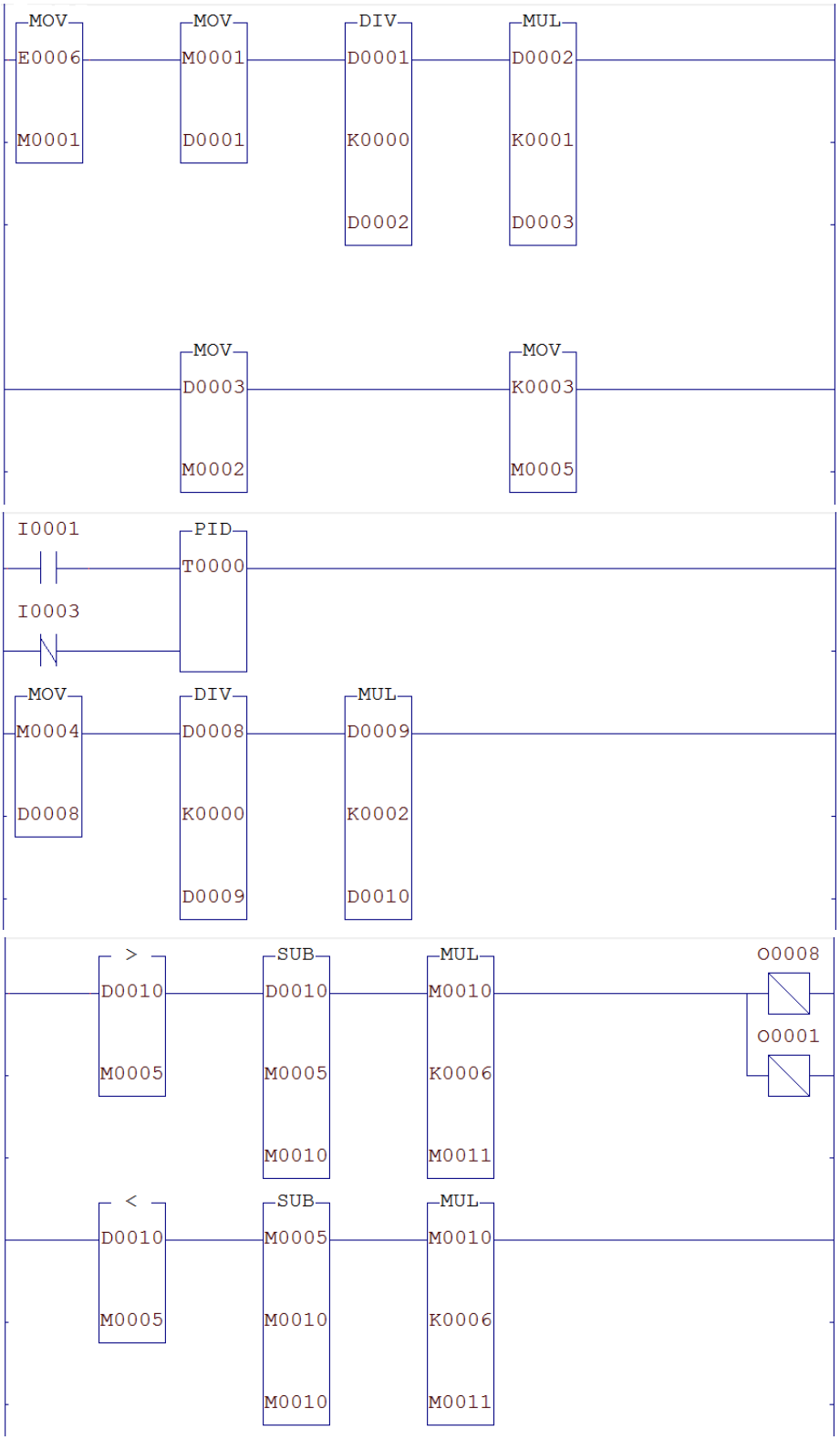
Outra atividade proposta é a utilização de dois outros módulos de temperatura: o módulo Peltier e o secador de grãos. Com eles, pode-se trabalhar um pouco mais com o ambiente de programação em *Ladder*, além de também utilizar a comunicação OPC. Nesta parte do trabalho, buscou-se revisar os estudos feitos por outros alunos, propondo atividades e/ou realizando aperfeiçoamentos na parte física ou de código dos módulos experimentais. Com isso, fica a disposição da disciplina, trabalhar experimentalmente com esses módulos.

8.1 MÓDULO PELTIER

Efeito Peltier é a produção de um gradiente de temperatura em duas junções de dois condutores (ou semicondutores) de materiais diferentes quando submetidos a um tensão elétrica em um circuito fechado. O módulo, ou célula de Peltier, é geralmente utilizado em aplicações de resfriamento de microprocessadores, computadores e dispositivos eletrônicos de médio porte.

Quando aplicada uma diferença de potencial nos terminais do dispositivo, é estabelecida também uma diferença de temperatura que faz o calor transferir-se de uma interface a outra, enquanto a corrente gerada atravessa os semicondutores. A eficiência da transferência é determinada pela corrente no interior do dispositivo e o número de semicondutores no interior da célula. No módulo estudado, utiliza-se um dissipador acoplado ao módulo de forma a evitar aquecimentos em outras partes do módulo.

Tendo em vista os trabalhos desenvolvidos por outros alunos, foi feita uma atualização no código, que tem como objetivo o controle da temperatura, utilizando bloco PID do SPDSW. Segue o código atualizado.


MOV
E0006
M0001
MOV
M0001
D0001
DIV
D0001
K0000
D0002
MUL
D0002
K0001
D0003
MOV
D0003
M0002
MOV
K0003
M0005
I0001
I0003
PID
T0000
MOV
M0004
D0008
DIV
D0008
K0000
D0009
MUL
D0009
K0002
D0010
>
D0010
M0005
SUB
D0010
M0005
M0010
MUL
M0010
K0006
M0011
O0008
O0001
<
D0010
M0005
SUB
M0005
M0010
M0010
MUL
M0010
K0006
M0011

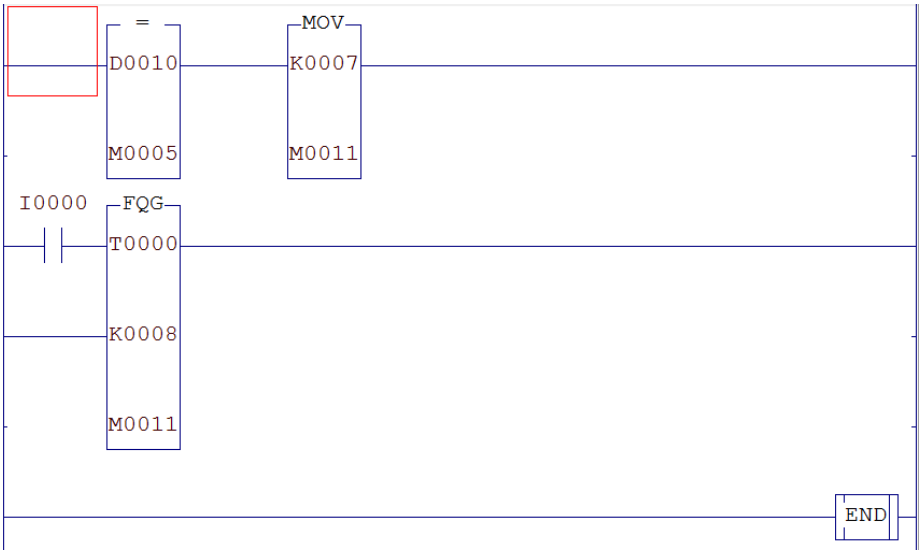


Figura 74. Código para controle de temperatura do módulo Peltier.

Parâmetros:

- E0006: valor lido no LM35;
- K0000: Resolução (4095);
- K0001: Fator de Conversão do LM35;
- D0003: valor lido do LM35, transformado em °C;
- K0003: valor de referência (50);
- M0002, M0003 e M0004: PV, SP e OV do bloco PID;
- K0002: valor para transformar em porcentagem (100);
- K0006: índice para dobrar o valor a diferença de temperatura (2);
- K0007: valor mínimo para *Duty Cycle* (4);
- K0008: Frequência do gerador (500);

O programa funciona da seguinte forma: Depois de lido, o valor da entrada analógica é transformado em °C. O valor inicial do PID é 50, assim como a referência. Isso é dado para determinar se ocorre o aquecimento ou resfriamento do módulo (maior que 50, aquece, menor que 50, esfria). Ativando a chave I0001, o PID é ativado e, portanto, realiza o controle. Ocorre a comparação da saída, em porcentagem, do bloco PID com o valor de referência. Caso seja maior, ativa-se a saída O0008, a qual ativa o PWM e impõe o valor do *Duty cycle* do gerador de frequência. I0000 ativa o gerador.

É importante ressaltar que é possível fazer a comunicação via OPC como foi feito no module de controle de temperatura, apresentado no capítulo anterior. Os passos para isso são semelhantes aos que foram explanados.

8.2 SECADOR DE GRÃOS

Para esse trabalho, onde sua implementação via Hardware e Software foi realizado pelo aluno Felipe Lira Furtado, no seu relatório de estagio, foi feito um divisor de tensão com um intuito de deixar fixo no módulo do secador. Esse divisor de tensão se faz necessário para limitar a tensão de controle, pois a faixa de tensão que sai do CLP é de 0 a 24 Volts. O secador trabalha numa faixa de 0 a 6 Volts, pois o mesmo alcança temperaturas próximas a 60 graus, o seu limite de operação.

O mesmo trabalho de comunicação via OPC pode ser feito nesse processo, seguindo os passos explicados no capítulo 6 desse trabalho.

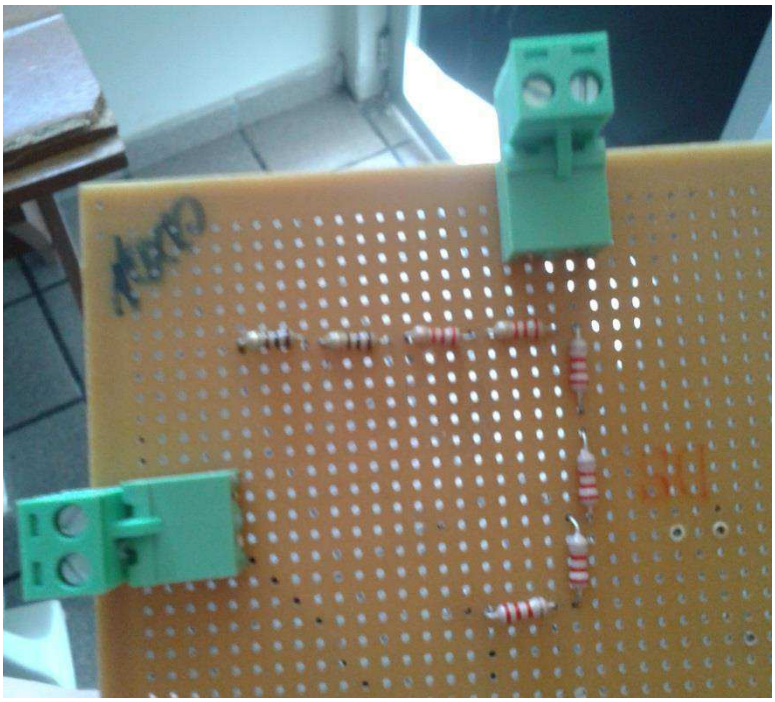


Figura 75. Divisor de Tensão.

9 CONCLUSÃO

Neste trabalho, ficou evidenciado que a principal contribuição do mesmo foi o desenvolvimento de guias, que auxiliarão o ensinamento e entendimento sobre controladores lógicos programáveis e sua linguagem de programação *Ladder*. Além disso, o objetivo de propor atividades experimentais, a fim de complementar o conteúdo teórico da disciplina, foi alcançado.

Destaca-se que o trabalho feito vai além do ambiente teórico e de simulações, permitindo o envolvimento com situações reais de uso do conhecimento obtido em toda a graduação.

Conclui-se que é de extrema importância o envolvimento de alunos de graduação na elaboração de atividades para determinada disciplinas (principalmente aquelas que oferecem um ambiente prático).

BIBLIOGRAFIA

BRYAN, L. A.; BRYAN, E., A. Programmable Controllers - Theory and Implementation, 2ª Ed. Atlanta: Industrial Text, 1997. p. 4.

FURTADO, F. L. Experimentos de Controle de Temperatura: Módulo Peltier e Secador de Grãos. Campina Grande, 2014.

HI TECNOLOGIA. Kit de Treinamento com o controlador eZAP900: Minas Gerais. Disponível em: http://www.hitecnologia.com.br/produtos/hardware/kit-de-treinamento/kit-familia-zap900/eztk900. Acesso em Março de 2015.

HI TECNOLOGIA. Driver OPC para Comunicação Através do Protocolo SCP-H1 HS1-Serial: Minas Gerais. Disponível em: http://www.hitecnologia.com.br/download/ENA0003300.pdf.

IDOETA, I. V.; CAPUANO, F. G. Elementos de Eletrônica Digital, 40ª Ed. São Paulo: Editora Érica, 2008.

MARTINS, G. M. Princípios de Automação Industrial. Santa Maria, 2012.

NETO, J. T. C.; CAVALCANTI, A. L. O. Controladores Lógicos Programáveis. Natal, 2011.

NOBREGA, R. D. Linguagem *Ladder*. Campina Grande, 2003.

PARR, E. A. Programmable Controllers: Na Engineer's Guide. 3 Ed. Burlington: Newnes, 2003.

SILVA, R. C. Sistema de comunicação OPC para uma coluna de destilação piloto. Campos dos Goytacazes, 2008.

SOUZA, V. A. O Protocolo Modbus. Rio de Janeiro: Cerne-Tecnologia e Treinamento Ltda., 2010.

TECHNO-TEST. Educational Products - CIC-500 DSP Development and Experimenta System, Quebec. Disponível em: http://catalogue.techno-test.com/products/6-Educational_Products/348-K_H_CIC_500-CIC_500_DSP_Development_and_Experiment_System.pdf. Acesso em Março de 2015.