Apostila do Curso
PTC3418
Laboratório de Automação

Escola Politécnica da USP
PTC - Depto. de Engenharia de Telecomunicações e Controle
LAC - Laboratório de Automação e Controle

Edição 2022

Sumário

1 Controle nebuloso **4**
- 1.1 Objetivo . . . 4
- 1.2 Introdução teórica . . . 4
  - 1.2.1 Conjuntos Fuzzy . . . 4
  - 1.2.2 Funções de pertinência . . . 5
  - 1.2.3 Operações com conjuntos fuzzy . . . 7
  - 1.2.4 Variáveis linguísticas . . . 10
- 1.3 Modelo de Mamdani . . . 11
  - 1.3.1 Nebulizador ou fuzzificador . . . 12
  - 1.3.2 Base de regras . . . 14
  - 1.3.3 Máquina de inferência nebulosa . . . 16
  - 1.3.4 Desnebulizador ou desfuzzificador . . . 20
- 1.4 Controle Fuzzy . . . 21
  - 1.4.1 Tipos de Controladores Fuzzy . . . 22
  - 1.4.2 Algumas considerações relevantes . . . 24
- 1.5 Preparação da atividade de simulação . . . 25
  - 1.5.1 Projeto de controlador nebuloso . . . 25
- 1.6 Atividades práticas . . . 26
  - 1.6.1 Sistema de Aquisição de Dados e Servomecanismo . . . 26
  - 1.6.2 Implementação do controlador nebuloso . . . 27
- 1.7 Dicas . . . 28
- 1.8 Relatório . . . 29

2 Introdução aos Controladores Lógicos Programáveis **30**
- 2.1 Objetivo . . . 30
- 2.2 Introdução teórica . . . 30
- 2.3 Programação Ladder . . . 32
  - 2.3.1 Contatos de Entrada e Saída . . . 32
  - 2.3.2 Algumas Instruções de Temporização . . . 33
  - 2.3.3 Algumas Instruções de Contagem . . . 34
- 2.4 Programação SFC . . . 35
- 2.5 O CLP do laboratório e o Ambiente de Programação . . . 38
  - 2.5.1 Como iniciar uma seção de programação . . . 38

2.5.2 Meu primeiro programa em Ladder no RSLogix 5000 42
2.5.3 Modos de funcionamento do CLP 43
2.5.4 Funções básicas em Ladder do RSLogix 5000 44
2.5.5 SFC do RSLogix 5000 50
2.5.6 Problemas com Solução 54
2.6 Dispositivos Pneumáticos 72
2.6.1 Cuidados com a Segurança 77
2.7 Atividades 78
2.7.1 Exercícios simples 78
2.7.2 Exercício usando contadores 78
2.7.3 Exercícios usando temporizadores 79
2.7.4 Exercício usando contadores e temporizadores 80
2.7.5 Dispositivos Pneumáticos - Cancela em linha ferroviária 81
2.7.6 Dispositivos Pneumáticos - Porta de um vagão do metrô 81
2.7.7 Dispositivos Pneumáticos - Sistema de eclusas do Canal do Panamá 82
2.8 Relatório 83
Apêndice I — Simulador CODESYS 84
Ladder CODESYS 86
SFC CODESYS 92
Atividades Adaptadas para o Simulador CODESYS 96

1 Controle nebuloso

1.1 Objetivo

Esta experiência tem por objetivo a familiarização com a técnica de Controle Nebuloso (*Fuzzy Control*, em inglês). Para isso será projetado um controlador de posição para o servomecanismo do laboratório.

1.2 Introdução teórica

Os fundamentos da teoria de conjuntos nebulosos foram apresentados pela primeira vez em 1965 pelo Prof. L.A. Zadeh no artigo [Zad65]. Essa teoria permite que informação imprecisa, qualitativa, possa ser expressa e manipulada através de um formalismo matemático. Como seu nome sugere, ela apresenta uma generalização do conceito tradicional de conjunto [Ton77].

O nosso interesse aqui é apresentar algumas noções básicas do assunto com a perspectiva de utilização em sistemas de controle. Contudo, são inúmeras as áreas atualmente em que tal teoria tem sido aplicada, dentre as quais podem ser citadas: reconhecimento de imagem, reconhecimentos de voz, sistemas especialistas (incluindo-se aqui sistemas de apoio à decisão, diagnóstico médico, etc...), aplicações na área de negócios, etc. [TAS94].

1.2.1 Conjuntos Fuzzy

Um sistema cujas proposições são verdadeira ou falsa, mas não ambas, usa apenas dois valores lógicos, que representam apenas uma aproximação da razão humana. Os seguintes exemplos não constituem conjuntos no sentido matemático usual:

- o conjunto de homens altos;
- o conjunto de mulheres bonitas;
- o conjunto de carros caros;
- o conjunto de pessoas obesas;
- o conjunto de idades infantis;
- o conjunto de temperaturas quentes.

Para definir, por exemplo, o conjunto de homens altos poderia se definir que uma altura x é alta se $x \geq 176$ cm. Esta é uma aproximação abrupta para o significado "alto".

Usando uma função com um grau de pertinência pode-se fazer a transição de "não alto" para "alto" de uma forma gradual, conforme é mostrado na Figura 1.1 .

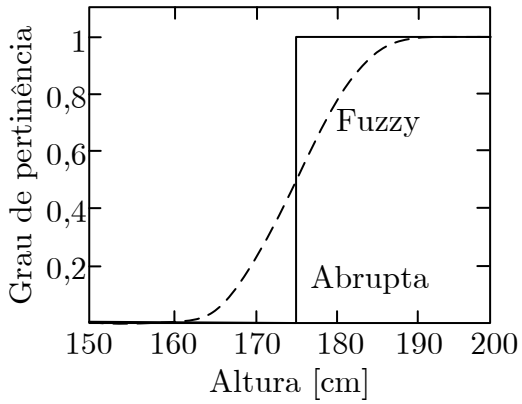


Figura 1.1: Duas definições para o conjunto de homens "altos": conjunto abrupto e conjunto fuzzy.

Definição: dada uma coleção de objetos U, um conjunto fuzzy A é o conjunto do par ordenado

$$A = \{[\ x, \mu(x)\ ] \mid x \in U\} \tag{1.1}$$

sendo que $\mu(x)$ é chamada de função de pertinência do conjunto de todos os objetos x de U.

Para cada x há uma grau de pertinência $\mu(x)$ no intervalo fechado $[0, 1]$.

O conjunto fuzzy sugere uma região limitada, ao invés da fronteira abrupta dos conjuntos clássicos.

Alguns exemplos:

- O conjunto de temperaturas altas, o conjunto de ventos fortes ou o conjunto de dias ensolarados são conjuntos fuzzy em relatórios de previsão de tempo.
- O conjunto de pessoas jovens. Um bebê de 1 ano seria claramente membro do conjunto, enquanto que um idoso de 100 anos não seria membro do conjunto. Uma pessoa de 30 anos poderia ser membro do conjunto, mas com grau de pertinência de $0,5$.
- O conjunto de pessoas adultas. Nas eleições do Brasil é permitido votar as pessoas com idade igual ou superior a 16 anos. Segundo esta definição o conjunto de pessoas adultas é um conjunto abrupto.

1.2.2 Funções de pertinência

A função de pertinência $\mu(x)$, de um conjunto fuzzy contínuo A, expressa o quanto um elemento x pertence ao conjunto A, conforme ilustra a Figura 1.2.

A Figura 1.3 ilustra quatro possíveis funções de pertinência.

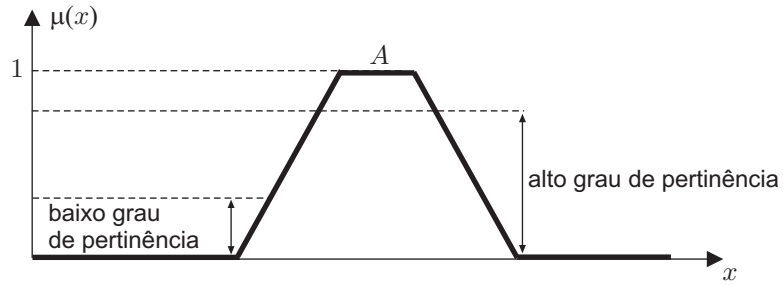


Figura 1.2: Graus de pertinência de um conjunto fuzzy A.

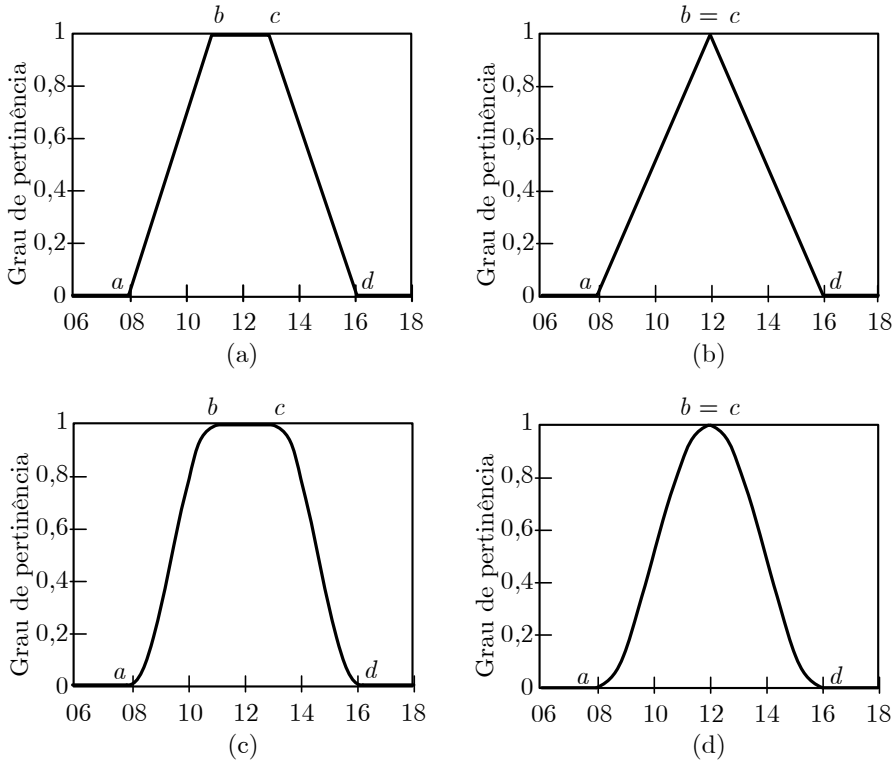


Figura 1.3: Funções de pertinência representando as horas por volta do meio-dia. (a) trapezoidal (b) tringular (c) trapezoidal suave (d) triangular suave.

A função de pertinência trapezoidal é uma função contínua, *linear por trechos*, com quatro parâmetros a, b, c, d, dada por

$$\mu(x) = \begin{cases} 0 & , \ x < a \\ \frac{x-a}{b-a} & , \ a \leq x < b \\ 1 & , \ b \leq x < c \\ \frac{d-x}{d-c} & , \ c \leq x < d \\ 0 & , \ d \leq x \end{cases} \tag{1.2}$$

A função de pertinência triangular é resultante da função trapezoidal com $b = c$, ou seja

$$\mu(x) = \begin{cases} 0 & , \ x < a \\ \frac{x-a}{b-a} & , \ a \leq x < b \\ \frac{d-x}{d-b} & , \ b \leq x < d \\ 0 & , \ d \leq x \end{cases} \tag{1.3}$$

Funções de pertinência suaves podem ser obtidas *trocando-se a parte linear pela função cosseno.* A função de pertinência trapezoidal suave é dada por

$$\mu(x) = \begin{cases} 0 & , \ x < a \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-b}{b-a}\pi\right) & , \ a \leq x < b \\ 1 & , \ b \leq x < c \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-c}{d-c}\pi\right) & , \ c \leq x < d \\ 0 & , \ d \leq x \end{cases} \tag{1.4}$$

Fazendo $b = c$ na função (1.4), a função de pertinência triangular suave resulta como

$$\mu(x) = \begin{cases} 0 & , \ x < a \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-b}{b-a}\pi\right) & , \ a \leq x < b \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-b}{d-b}\pi\right) & , \ b \leq x < d \\ 0 & , \ d \leq x \end{cases} \tag{1.5}$$

Outras funções de pertinência podem ser utilizadas como a gaussiana, sigmoide, etc.

1.2.3 Operações com conjuntos fuzzy

Dois conjuntos fuzzy A e B são iguais se eles têm a mesma função de pertinência para todo x

$$A = B \ \equiv \ \mu_A(x) = \mu_B(x) \ . \tag{1.6}$$

Um conjunto fuzzy A está contido num conjunto fuzzy B, se a função de pertinência de A é menor ou igual a de B, ou seja

$$A \subseteq B \ \equiv \ \mu_A(x) \leq \mu_B(x) \ . \tag{1.7}$$

A união clássica dos conjuntos X e Y, simbolizado por $X \cup Y$, é o conjunto de todos os objetos que são membros de X ou Y, ou ambos, isto é,

$$X \cup Y \ \equiv \ \{x \mid x \in X \text{ ou } x \in Y\} \ . \tag{1.8}$$

Por exemplo,

$$\{1, 2, 3\} \cup \{1, 3, 4\} = \{1, 2, 3, 4\} \ .$$

A intersecção clássica dos conjuntos X e Y, simbolizado por $X \cap Y$, é o conjunto de todos os objetos que são membros de ambos X e Y, isto é,

$$X \cap Y \ \equiv \ \{x \mid x \in X \text{ e } x \in Y\} \ . \tag{1.9}$$

Por exemplo,

$$\{1, 2, 3\} \cap \{1, 3, 4\} = \{1, 3\} \ .$$

O complemento clássico de um conjunto X, simbolizado por $\overline{X}$, é o conjunto dos membros x que não pertencem a X, isto é,

$$\overline{X} \ \equiv \ \{x \mid x \notin X\} \ . \tag{1.10}$$

Os diagramas com operações clássicas de conjuntos são apresentados na Figura 1.4.

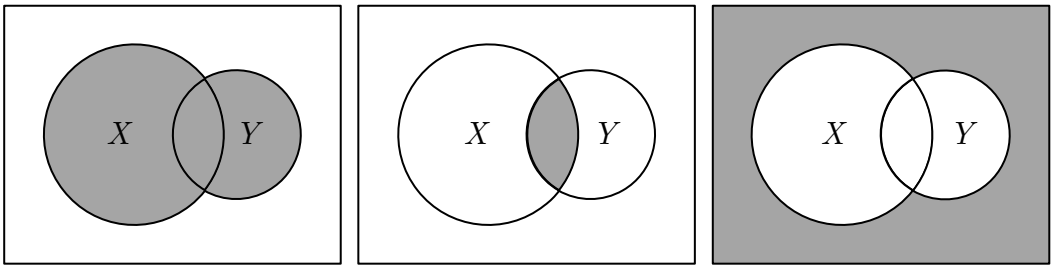


Figura 1.4: Operações clássicas de conjuntos: (a) união $X \cup Y$ (b) intersecção $X \cap Y$ (c) complemento $\overline{X \cup Y}$.

No caso de conjuntos fuzzy, deve-se considerar as funções de pertinência.

A união de dois conjuntos fuzzy A e B é

$$A \cup B \equiv \{[\ x, \mu_{A \cup B}(x)\ ] \mid \mu_{A \cup B}(x) = \max[\ \mu_A(x), \mu_B(x)\ ]\ \} \ . \tag{1.11}$$

A intersecção de dois conjuntos fuzzy A e B é

$$A \cap B \equiv \{[\, x, \mu_{A\cap B}(x)\,] \mid \mu_{A\cap B}(x) = \min[\, \mu_A(x), \mu_B(x)\,]\, \}\ . \tag{1.12}$$

O complemento de um conjunto fuzzy A é

$$\overline{A} \equiv \{[\, x, \mu_{\overline{A}}(x)\,] \mid \mu_{\overline{A}}(x) = 1 - \mu_A(x)\}\ . \tag{1.13}$$

Os diagramas com operações de conjuntos fuzzy são apresentados na Figura 1.5.

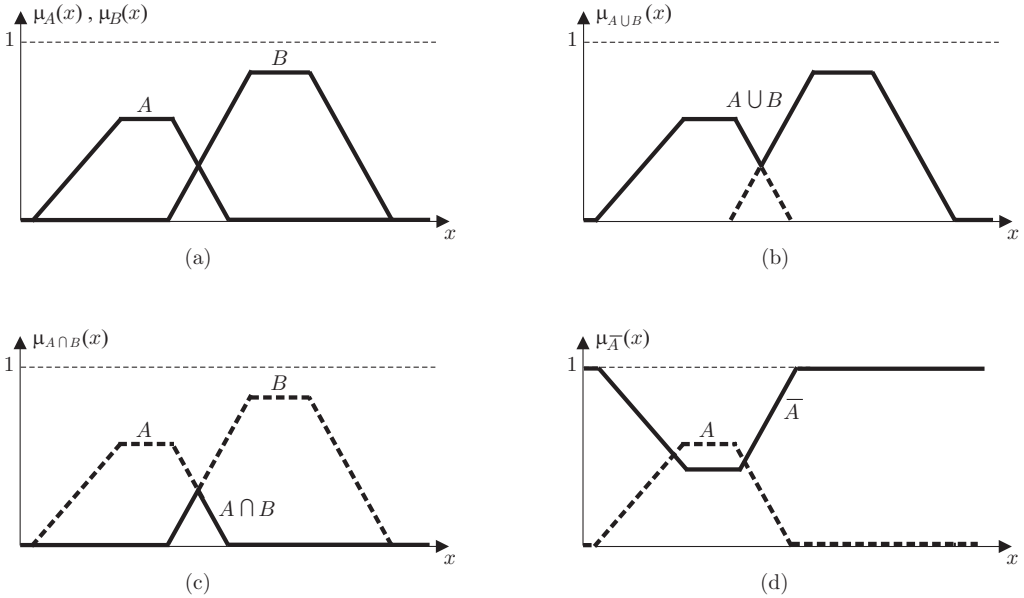


Figura 1.5: Operações com conjuntos fuzzy: (a) $\mu_A(x)$, $\mu_B(x)$ (b) $\mu_{A\cup B}(x)$ (c) $\mu_{A\cap B}(x)$ (d) $\mu_{\overline{A}}(x)$.

Considere o seguinte exemplo:

Uma família de quatro pessoas deseja comprar uma casa. O conjunto de casas disponíveis no mercado possui o seguinte número de quartos

$$U = \{1\ ;\ 2\ ;\ 3\ ;\ 4\ ;\ 5\ ;\ 6\ ;\ 7\ ;\ 8\ ;\ 9\ ;\ 10\}\ .$$

O conjunto fuzzy C representa o Conforto de cada casa, com grau de pertinência

$$\mu_C = \{0,2\ ;\ 0,5\ ;\ 0,8\ ;\ 1\ ;\ 0,7\ ;\ 0,3\ ;\ 0\ ;\ 0\ ;\ 0\ ;\ 0\}\ .$$

O conjunto fuzzy G representa o quão Grande é cada casa, com grau de pertinência

$$\mu_G = \{0\ ;\ 0\ ;\ 0,2\ ;\ 0,4\ ;\ 0,6\ ;\ 0,8\ ;\ 1\ ;\ 1\ ;\ 1\ ;\ 1\}\ .$$

A intersecção dos conjuntos Conforto e Grande é

$$C \cap G = \min(\mu_C, \mu_G) = \{0\ ;\ 0\ ;\ 0,2\ ;\ 0,4\ ;\ 0,6\ ;\ 0,3\ ;\ 0\ ;\ 0\ ;\ 0\ ;\ 0\}\ .$$

Logo, a melhor solução é uma casa com cinco quartos que tem um grau de pertinência igual a $0,6$.

A união dos conjuntos Conforto e Grande é

$$C \cup G = \max(\mu_C, \mu_G) = \{0,2\ ;\ 0,5\ ;\ 0,8\ ;\ 1\ ;\ 0,7\ ;\ 0,8\ ;\ 1\ ;\ 1\ ;\ 1\ ;\ 1\}\ .$$

No caso da união, uma casa com quatro ou de sete a dez quartos é completamente adequada, pois o grau de pertinência vale 1.

Se os filhos estiverem para se casar e forem mudar da casa em alguns meses, então é necessário comprar uma casa com Conforto e não Grande, com conjunto

$$C \cap \overline{G} = \min(\mu_C, 1 - \mu_G) = \{0,2\ ;\ 0,5\ ;\ 0,8\ ;\ 0,6\ ;\ 0,4\ ;\ 0,2\ ;\ 0\ ;\ 0\ ;\ 0\ ;\ 0\}\ ,$$

sendo que o melhor resultado é uma casa com três quartos, com grau de pertinência igual a $0,8$.

1.2.4 Variáveis linguísticas

Uma variável linguística possui palavras ou sentenças como valores.

Suponha, por exemplo, que **idade** seja uma variável linguística de um conjunto fuzzy U, definido como

$$U(\text{idade}) = \{\text{jovem}, \text{muito jovem}, \text{não muito jovem}, \text{mais ou menos velho}, \text{velho}\}\ . \tag{1.14}$$

Cada termo é definido num intervalo de, por exemplo, inteiros de 0 a 100 anos.

Um termo pode ter o seu significado modificado. Por exemplo, na sentença "muito próximo de zero", a palavra "muito" modifica o termo "próximo de zero". Outros exemplos são: "pequeno", "mais ou menos", "possivelmente", etc.

O efeito de "muito" é o de intensificar o valor da função de pertinência, ou seja,

$$\text{muito } A \equiv \{x, \mu_{\text{muito } A}(x) \mid \mu_{\text{muito } A}(x) = \mu_A^2(x), x \in X\}\ . \tag{1.15}$$

O efeito de "mais ou menos" é o oposto

$$\text{mais ou menos } A \equiv \{x, \mu_{\text{mais ou menos } A}(x) \mid \mu_{\text{mais ou menos } A}(x) = \sqrt{\mu_A(x)}, x \in X\}\ . \tag{1.16}$$

As funções de pertinência de "muito jovem" e "não muito jovem" são oriundas de "jovem" e a função de pertinência de "mais ou menos velho" é oriunda de "velho", conforme é ilustrado na Figura 1.6.

Uma família de funções de pertinência pode ser gerada por μ_A^k ou $\mu_A^{1/k}$ de acordo com os termos modificadores de significado.

Considere como exemplo o conjuntos de idades

$$U = \left[\ 0 \quad 20 \quad 40 \quad 60 \quad 80\ \right] \tag{1.17}$$

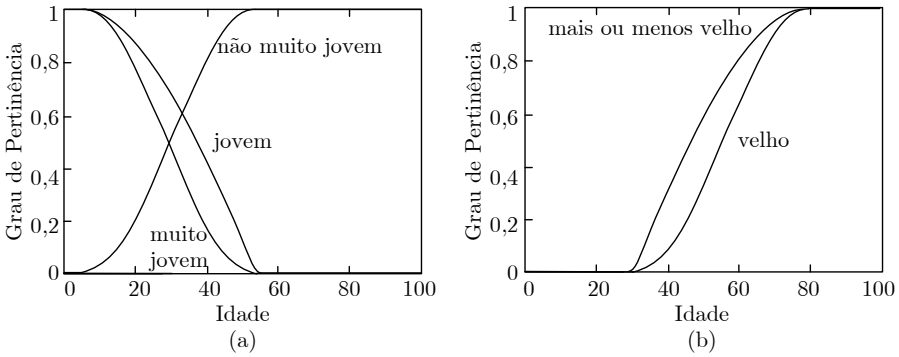


Figura 1.6: Funções de pertinência modificadas.

com funções de pertinência

$$\mu_{\text{jovem}} = \begin{bmatrix} 1 & 0,6 & 0,1 & 0 & 0 \end{bmatrix} \quad \text{e} \quad \mu_{\text{velho}} = \begin{bmatrix} 0 & 0 & 0,1 & 0,6 & 1 \end{bmatrix} . \tag{1.18}$$

A função de pertinência do conjunto muito jovem é

$$\mu_{\text{muito jovem}} = \mu^2_{\text{jovem}} = \begin{bmatrix} 1 & 0,36 & 0,01 & 0 & 0 \end{bmatrix} . \tag{1.19}$$

A função de pertinência do conjunto mais ou menos velho é

$$\mu_{\text{mais ou menos velho}} = \sqrt{\mu_{\text{velho}}} = \begin{bmatrix} 0 & 0 & 0,32 & 0,77 & 1 \end{bmatrix} . \tag{1.20}$$

1.3 Modelo de Mamdani

Frequentemente usada em sistemas fuzzy. O esquema básico é apresentado na Figura 1.7.

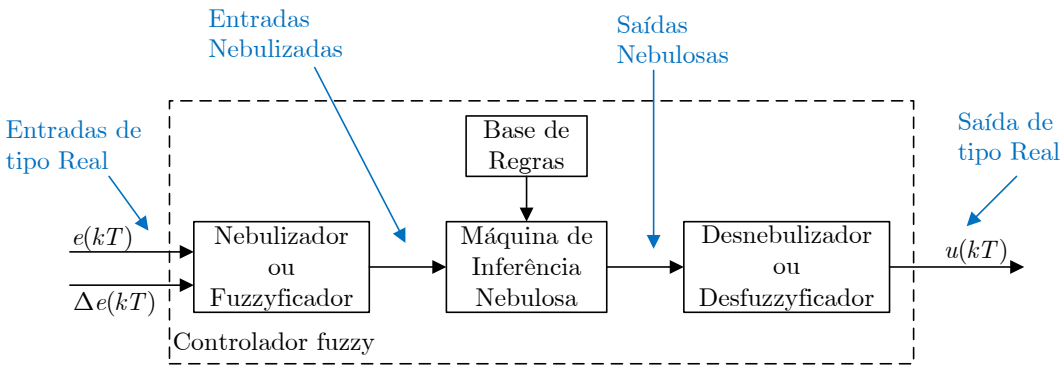


Figura 1.7: Estrutura de um controlador fuzzy.

O **Nebulizador** ou **fuzzificador** tem por funções:

- converter esses valores numéricos em valores linguísticos associados aos conjuntos nebulosos.

- mudar as escalas das variáveis de entrada do controlador, normalizando-as de maneira a pertencerem aos universos de discurso dos conjuntos nebulosos correspondentes;
- discretizar os valores numéricos das variáveis de entrada;

A **Base de regras** é uma base de conhecimento do processo e dos objetivos de controle, traduzida num conjunto de regras linguísticas de controle.

A **Máquina de inferência nebulosa** realiza a lógica de tomada de decisões, baseando-se na **Base de regras** e nos valores linguísticos das variáveis medidas e controladas.

O **Desnebulizador** ou **desfuzzificador** é responsável por converter os valores nebulizados para variáveis numéricas.

A inferência Mamdani é do tipo Máx-Min. Utiliza as operações de **união** (lógica **ou**) e de **interseção** de conjuntos (lógica **e**) através dos operadores de **máximo** e de **mínimo**, respectivamente. Sejam, por exemplo, A e B dois conjuntos fuzzy definidos em x e y, respectivamente. Assim, a condição A **e** B resulta na seguinte composição:

$$\mu(x, y) = \min[\mu_A(x), \mu_B(y)] \ . \tag{1.21}$$

A seguir serão detalhados tais blocos.

1.3.1 Nebulizador ou fuzzificador

Considere por exemplo um sistema que tem como entrada a temperatura, que assume diversos valores fuzzy e que tem como saída o fluxo de calor, que também assume valores fuzzy. Este sistema pode ser representado pelo modelo linguístico:

se temperatura é `alta` **então** o fluxo de calor é `positivo`.

se temperatura é `ideal` **então** o fluxo de calor é `nulo`.

se temperatura é `baixa` **então** o fluxo de calor é `negativo`.

As funções de pertinência para a temperatura podem ser escolhidas como, por exemplo, na Figura 1.8. Foram escolhidas funções trapezoidais com centros em `alta, ideal` e `baixa.` As larguras e os centros dos trapézios são parâmetros que podem ser ajustados com base na experiência do operador. Outros tipos de funções poderiam ser escolhidas, como, por exemplo, gaussianas ou triangulares.

Note que a temperatura de 25^o é `ideal` com grau de pertinência $\mu = 1$ e ao mesmo tempo é `alta` com grau de pertinência $\mu = 0,6$ e é `baixa` com grau de pertinência $\mu = 0,6$.

Retomando o exemplo da Tabela 1.2, para cada um dos conjuntos nebulosos `GP, PP, ZE, PN, GN` deve-se associar funções de pertinência μ. A forma da função é arbitrária. Por simplicidade, as mais utilizadas são as funções trapezoidais e triangulares.

Na Figura 1.9 são ilustradas funções triangulares para cada um dos conjuntos.

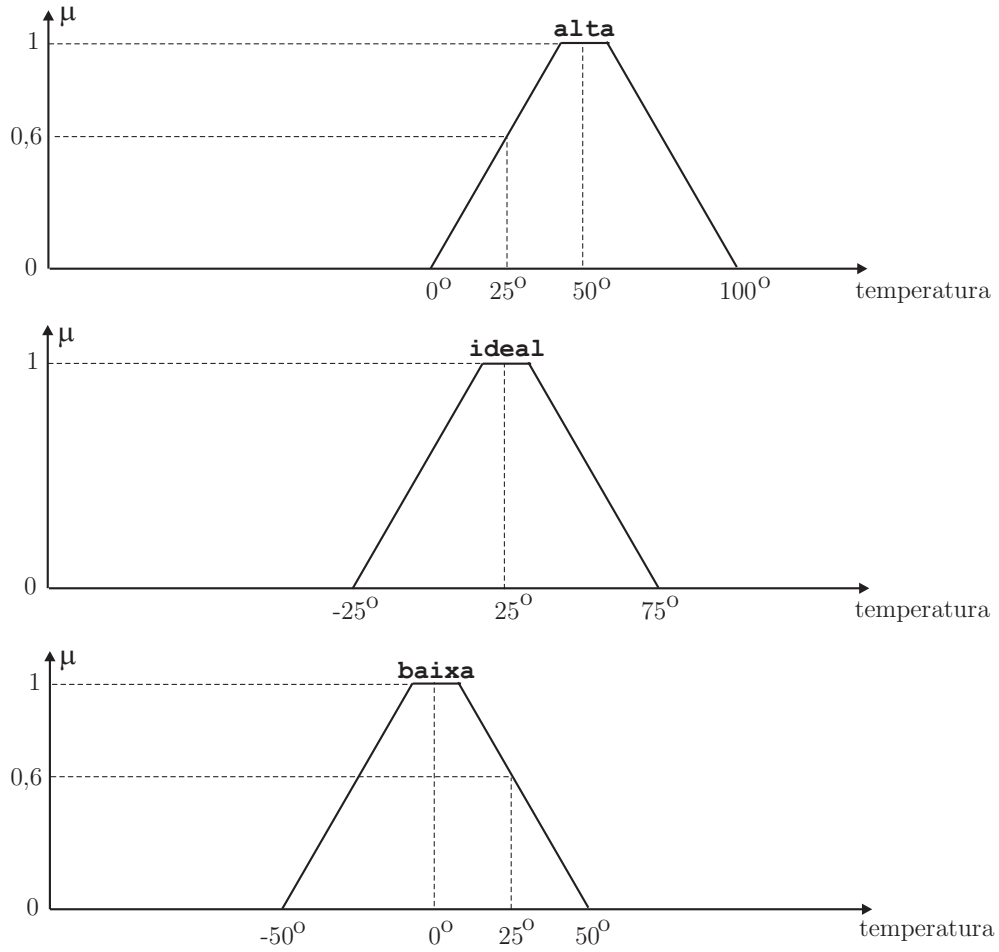


Figura 1.8: Funções de pertinência.

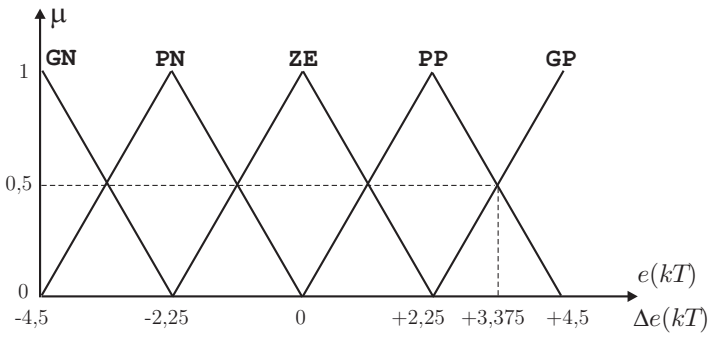


Figura 1.9: Funções de pertinência ou de associação.

Note que o erro $e(kT) = 3,375$ pertence ao conjunto `PP` e `GP` com o mesmo grau de pertinência $\mu = 0,5$. Por extensão, ele também é `GN`, `PN` e `ZE` com grau de pertinência 0.

Assim, o **Nebulizador** ou **fuzzificador**, como eram suas atribuições, muda as escalas das variáveis de entrada do controlador, normalizando-as de maneira a pertencerem aos universos de discurso dos conjuntos nebulosos correspondentes, e converte esses valores numéricos em valores linguísticos associados aos conjuntos nebulosos.

1.3.2 Base de regras

Como anteriormente dito, um controlador fuzzy se apoia em regras. As regras do controlador são baseadas na linguagem natural e podem ser entendidas por operadores ou por equipes de manutenção. Alguns critérios para a elaboração das regras são apresentados a seguir.

- *Experiência de funcionários e conhecimento de engenheiros*: elaborar um questionário detalhado a ser preenchido por funcionários ou engenheiros com o objetivo de extrair regras de controle fuzzy.
- *Baseado nas ações de controle do operador*: observando as ações de controle do operador pode-se encontrar relações do tipo entrada-saída que podem ser utilizadas na elaboração das regras fuzzy.
- *Examinando manuais* também podem ser obtidas informações úteis para a elaboração das regras.
- *Baseado no modelo da planta*: uma regra linguística pode ser vista como o inverso da planta. Assim, é possível obter regras fuzzy invertendo o modelo da planta. Este método é restrito a sistemas de baixa ordem.

Num sistema de controle, o sinal do erro é usualmente numérico. Num servomecanismo posicionador, como será o caso desta prática, o ângulo de saída pode ser uma tensão variando, por exemplo, entre $-4,5$V a $+4,5$V. Vamos imaginar que a experiência de processo tenha levado à quantificação do erro e da variação do erro da seguinte forma:

```
GP = Grande Positivo;

PP = Pequeno Positivo;

ZE = Zero;

PN = Pequeno Negativo;

GN = Grande Negativo.
```

E do esforço de controle:

```
GP = Grande Positivo;
```

```
MP = Médio Positivo;

ZE = Zero;

MN = Médio Negativo;

GN = Grande Negativo.
```

Essa mesma experiência de processo levou também à definição das seguintes regras básicas apresentadas na Tabela 1.1.

Tabela 1.1: Regras básicas de controle.

Erro	Variação do Erro	Sinal de Controle
GN	GN	GN
GN	PN	GN
GN	ZE	GN
GN	PP	MN
GN	GP	MN
PN	GN	MN
PN	PN	MN
PN	ZE	ZE
PN	PP	ZE
PN	GP	MP
ZE	GN	MN
ZE	PN	MN
ZE	ZE	ZE
ZE	PP	MP
ZE	GP	MP
PP	GN	MN
PP	PN	ZE
PP	ZE	ZE
PP	PP	MP
PP	GP	MP
GP	GN	MP
GP	PN	MP
GP	ZE	GP
GP	PP	GP
GP	GP	GP

Uma forma mais compacta é apresentar as regras na forma matricial da Tabela 1.2.

Tabela 1.2: Forma compacta de visualização das regras básicas de controle.

		Erro				
Variação do Erro		GN	PN	ZE	PP	GP
	GN	GN	MN	MN	MN	MP
	PN	GN	MN	MN	ZE	MP
	ZE	GN	ZE	ZE	ZE	GP
	PP	MN	ZE	MP	MP	GP
	GP	MN	MP	MP	MP	GP

Nota importante: para manter a simplicidade desta seção, não foram definidas as regras para Erro e Variação do Erro em MN e MP. Obviamente, para a implementação da tabela de regras é necessário que todos os casos tenham sido previstos.

A definição da Tabela 1.2 é o ponto de partida para a definição do conteúdo de todos os outros blocos do controlador fuzzy da Figura 1.7.

1.3.3 Máquina de inferência nebulosa

Como citado anteriormente, a **Máquina de inferência nebulosa** realiza a lógica de tomada de decisões, baseando-se na **Base de regras** e nos valores linguísticos das variáveis medidas e controladas.

Um controlador fuzzy pode assumir a forma de um sistema aditivo como o da Figura 1.10.

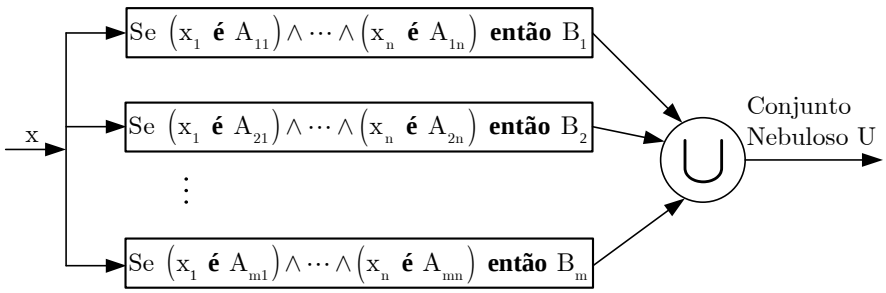


Figura 1.10: Sistema nebuloso aditivo.

Suponha que quatro regras tenham sido formuladas por um operador experiente para o controle de um servomecanismo posicionador:

- **regra 1: se** $e(kT)$ é `ZE` **e** $\Delta e(kT)$ é `PP` **então** $u(kT)$ deve ser `PP`;

  ou

- **regra 2: se** $e(kT)$ é `ZE` **e** $\Delta e(kT)$ é `ZE` **então** $u(kT)$ deve ser `ZE`;

ou

- **regra 3: se** $e(kT)$ é `PN` **e** $\Delta e(kT)$ é `PN` **então** $u(kT)$ deve ser `PN`;

  ou

- **regra 4: se** $e(kT)$ é `GP` **e** $\Delta e(kT)$ é `ZE` **então** $u(kT)$ deve ser `GP`.

Note que cada regra apresenta um conectivo **e,** caracterizando assim uma operação de intersecção entre conjuntos nebulosos. Além disso, entre as regras há um conectivo **ou,** indicando operações de união.

Suponha que num certo instante o servomecanismo posicionador apresente um erro $e(kT) = -0,75$V e uma variação de erro $\Delta e(kT) = +1,2$V. Nessa situação, os graus de pertinência são ilustrados na Figura 1.11.

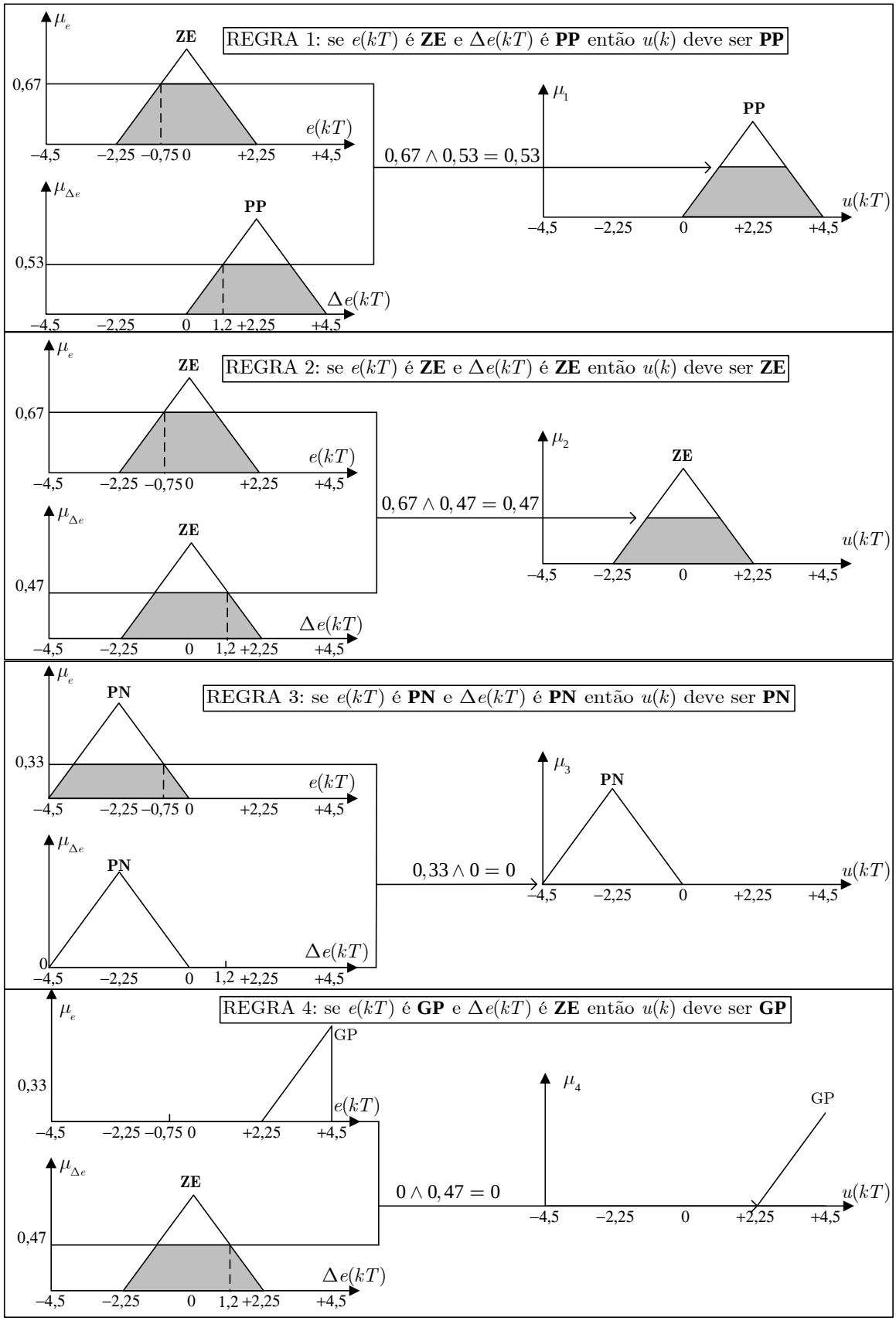


Figura 1.11: Graus de pertinência para $e(kT) = -0,75$V e $\Delta e(kT) = +1,2$V.

Dado o valor do erro $e(kT)$ e da variação do erro $\Delta e(kT)$, em cada instante kT, o valor da função de pertinência de cada regra pode ser calculado por meio da equação (1.3) para funções triangulares:

regra 1

$$\mu_e = \frac{x-a}{b-a} = \frac{x-(-2,25)}{0-(-2,25)} = \frac{e(kT)+2,25}{2,25} = \frac{-0,75+2,25}{2,25} \cong 0,67\ . \tag{1.22}$$

$$\mu_{\Delta e} = \frac{x-a}{b-a} = \frac{x-0}{2,25-0} = \frac{\Delta e(kT)}{2,25} = \frac{1,2}{2,25} \cong 0,53\ . \tag{1.23}$$

regra 2

$$\mu_e = \frac{x-a}{b-a} = \frac{x-(-2,25)}{0-(-2,25)} = \frac{e(kT)+2,25}{2,25} = \frac{-0,75+2,25}{2,25} \cong 0,67\ . \tag{1.24}$$

$$\mu_{\Delta e} = \frac{d-x}{d-b} = \frac{2,25-x}{2,25-0} = \frac{2,25-\Delta e(kT)}{2,25} = \frac{2,25-1,2}{2,25} \cong 0,47\ . \tag{1.25}$$

regra 3

$$\mu_e = \frac{d-x}{d-b} = \frac{0-x}{0-(-2,25)} = \frac{-e(kT)}{2,25} = \frac{-(-0,75)}{2,25} \cong 0,33\ . \tag{1.26}$$

$$\mu_{\Delta e} = 0\ . \tag{1.27}$$

regra 4

$$\mu_e = 0 \tag{1.28}$$

$$\mu_{\Delta e} = \frac{d-x}{d-b} = \frac{2,25-x}{2,25-0} = \frac{2,25-\Delta e(kT)}{2,25} = \frac{2,25-1,2}{2,25} \cong 0,47\ . \tag{1.29}$$

O grau de pertinência μ_i $(i = 1, 2, 3, 4)$ de cada conjunto U_i $(i = 1, 2, 3, 4)$ é obtido por meio da regra de Mamdani:

- **regra 1:** $\mu_1 = \min[\mu_e, \mu_{\Delta e}] = \min[0,67; 0,53] = 0,53.$
- **regra 2:** $\mu_2 = \min[\mu_e, \mu_{\Delta e}] = \min[0,67; 0,47] = 0,47.$
- **regra 3:** $\mu_3 = \min[\mu_e, \mu_{\Delta e}] = \min[0,33; 0] = 0.$
- **regra 4:** $\mu_4 = \min[\mu_e, \mu_{\Delta e}] = \min[0; 0,47] = 0.$

O conjunto fuzzy resultante é mostrado na Figura 1.12, que é obtido por meio da união dos conjuntos U_i $(i = 1, 2, 3, 4)$ de cada regra.

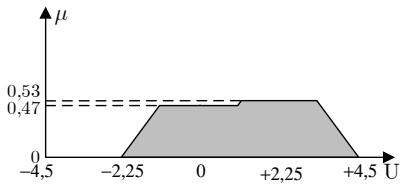


Figura 1.12: Conjunto fuzzy resultante.

1.3.4 Desnebulizador ou desfuzzificador

O **Desnebulizador** ou **desfuzzificador** é responsável por converter os valores nebulizados para variáveis numéricas. As regras nebulosas produzem como saída o conjunto nebuloso U da Figura 1.12. Para obter o sinal de saída do controlador $u(kT)$, que vai ser aplicado na entrada da planta no instante kT, é necessário converter os graus de pertinência numa variável numérica. Uma das formas (não a única) de se efetuar a desfuzzificação consiste no método das médias ponderadas, que consiste na seguinte expressão

$$u(kT) = \frac{\sum b_i \,.\mu_i}{\sum \mu_i} \,, \tag{1.30}$$

sendo b_i a abscissa central com o maior grau de pertinência em cada regra i. Este método é computacionalmente simples e resulta em desempenho semelhante ao método do centroide, que é computacionalmente mais complexo.

Com as regras definidas na Figura 1.11, tem-se:

$$u(kT) = \frac{0,53.(+2,25) + 0,47.(0) + 0.(-2,25) + 0.(+4,5)}{0,53 + 0,47 + 0 + 0} \approx 1,19 \,. \tag{1.31}$$

A Figura 1.13 ilustra o método de desfuzzificação por médias ponderadas para o exemplo dado.

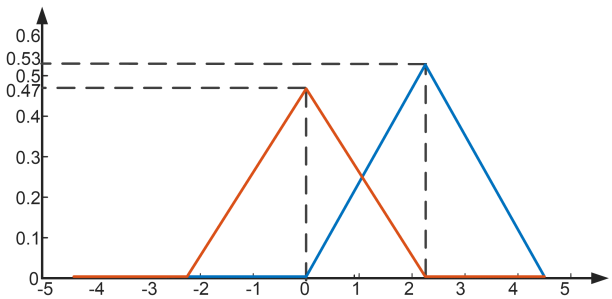


Figura 1.13: Exemplo gráfico de desfuzzificação por médias ponderadas.

No método do centroide, considerando funções de pertinência discreta, tem-se a seguinte expressão para a desfuzzificação

$$u(kT) = \frac{\sum x_i \mu(x_i)}{\sum \mu(x_i)} \,, \tag{1.32}$$

em que x_i representa a i-ésima amostra do universo de discurso da saída. No caso de funções de pertinência contínua, a desfuzzificação pelo método do centroide é dada por:

$$u(kT) = \frac{\int x\mu(x)dx}{\int \mu(x)dx} \,. \tag{1.33}$$

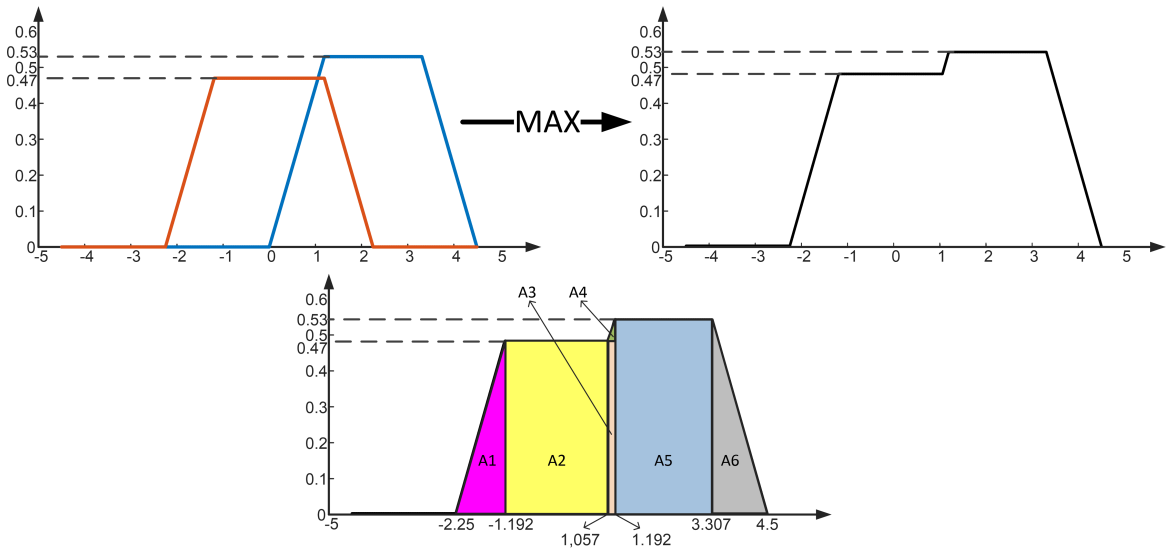


Figura 1.14: Exemplo gráfico de desfuzzificação pelo método do centroide.

A Figura 1.14 ilustra o método de desfuzzificação pelo centroide para o exemplo dado.

Para calcular manualmente, note que a área resultante na saída pode ser fragmentada em subáreas A_1 a A_6 como visto na Figura 1.14. Assim, de forma equivalente, o valor desfuzzificado pode ser calculado como:

$$u(kT) = \frac{\sum A_i \bar{x}_i}{\sum A_i} \ , \tag{1.34}$$

onde $\bar{x}_i$ é a coordenada x do centroide da i-ésima área. Os valores de A_i e $\bar{x}_i$, $i = 1, 2, \ldots, 6$, são apresentados no quadro seguinte:

i	A_i	$\bar{x}_i$
1	0,2486	-1,5447
2	1,0570	-0,0675
3	0,0639	1,1250
4	0,0041	1,1477
5	1,1204	2,2500
6	0,3161	3,7047

Com isso, o valor desfuzzificado é dado por:

$$u(kT) = \frac{\sum_{i=0}^{6} A_i \bar{x}_i}{\sum A_i} \approx 1,18 \ . \tag{1.35}$$

1.4 Controle Fuzzy

Controle fuzzy significa controlar com regras. O diagrama de blocos de um sistema de controle típico com realimentação é apresentado na Figura 1.15.

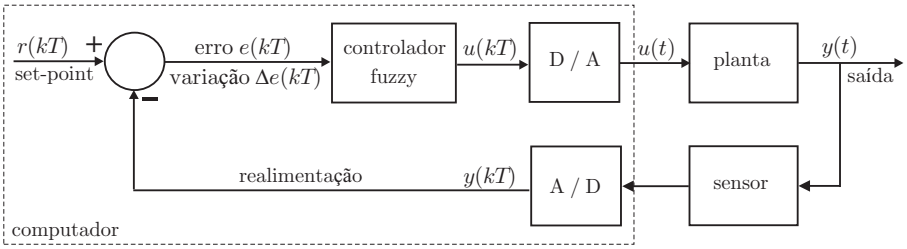


Figura 1.15: Diagrama de blocos de um sistema de controle em malha fechada.

Um controle fuzzy pode possuir regras empíricas, baseadas principalmente em plantas controladas por operadores. As regras são do tipo `se-então` com a premissa do lado do `se` e a conclusão do lado do `então`. Exemplo:

Se o erro é `N` **e** a variação do erro é `N` **então** o controle é `GN`.

Se o erro é `N` **e** a variação do erro é `Z` **então** o controle é `MN`.

Os termos linguísticos `N (Negativo)` ou `Z (Zero)` são premissas e `GN (Grande Negativo)` ou `MN (Médio Negativo)` são conclusões.

Um computador ou equipamento digital pode executar as regras e calcular o sinal de controle $u[kT]$ a partir de medidas do erro e da variação do erro, ou seja,

$$\text{erro atual} = \text{set-point} - \text{saída atual da planta: } e[kT] = r[kT] - y[kT] \ , \tag{1.36}$$

$$\text{variação do erro} = \text{erro atual} - \text{erro no passo anterior: } \Delta e[kT] = e[kT] - e[(k-1)T] \ . \tag{1.37}$$

1.4.1 Tipos de Controladores Fuzzy

Um controlador fuzzy pode se assemelhar a um controlador **PID**. A seguir são apresentados alguns tipos de controladores com base em [Jan07].

- **Fuzzy P**

Um controlador proporcional amplifica o sinal de erro e possui uma função do tipo

$$u(t) = K_p \, e(t) \ . \tag{1.38}$$

Um controlador fuzzy proporcional tem regras do tipo

`Se erro` $e(kT)$ `= A então sinal de controle` $u(kT)$ `= B.`

sendo A e B conjuntos fuzzy com funções de pertinência linguísticas atribuídas às variáveis $e(kT)$ e $u(kT)$.

O diagrama de blocos é apresentado na Figura 1.16.

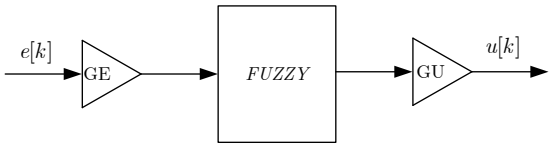


Figura 1.16: Digrama de blocos do controlador fuzzy P.

- **Fuzzy PD**

Um controlador proporcional+derivativo tem a função de melhorar a estabilidade relativa do sistema e possui uma função do tipo

$$u(t) = K_p\, e(t) + K_d \frac{de(t)}{dt}\ . \tag{1.39}$$

Um controlador fuzzy proporcional+derivativo tem regras do tipo

`Se erro` $e(kT)$ `= A e variação do erro` $\Delta e(kT)$ `= B então sinal de controle` $u(kT)$ `= C.`

sendo A, B e C conjuntos fuzzy com funções de pertinência linguísticas atribuídas às variáveis $e(kT)$, $\Delta e(kT)$ e $u(kT)$.

O diagrama de blocos é apresentado na Figura 1.17.

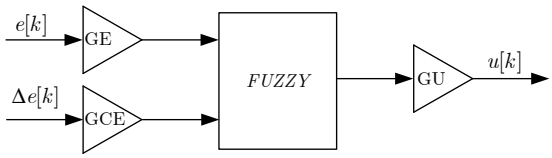


Figura 1.17: Digrama de blocos do controlador fuzzy PD.

Note que tal esquema não é exatamente um PD, pois as entradas são o erro e a variação do erro, ao invés do erro e da derivada do erro.

- **Fuzzy Incremental (equivalente PI)**

Um controlador proporcional+integral tem a função de eliminar o erro estacionário entre a referência e a saída do processo e possui uma função do tipo

$$u(t) = K_p\, e(t) + K_i \int e(t)dt \quad \text{ou} \tag{1.40}$$

$$\frac{du(t)}{dt} = K_p\, \frac{de(t)}{dt} + K_i\, e(t)\ . \tag{1.41}$$

Um controlador fuzzy proporcional+integral tem regras do tipo

`Se erro` $e(kT)$ `= A e variação do erro` $\Delta e(kT)$ `= B então variação do controle` $\Delta u(kT)$ `= C.`

Note que o resultado da conclusão da regra é a variação do sinal de controle $\Delta u(kT)$ e não $u(kT)$ como nos controladores fuzzy P ou PD. Trata-se de um controlador incremental, onde a saída do controlador é dada por um incremento, Δu, positivo ou negativo na ação de controle. A Figura 1.18 apresenta o diagrama de blocos deste controlador.

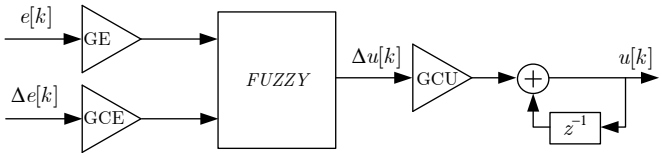


Figura 1.18: Digrama de blocos do controlador fuzzy incremental (PI).

- **Fuzzy PD+I**

Trata-se de um controlador fuzzy PD, onde o erro e a variação do erro são as entradas do bloco fuzzy. Uma ação acumulativa (integral) do erro é adicionada à ação de controle, conforme ilustrado na Figura 1.19.

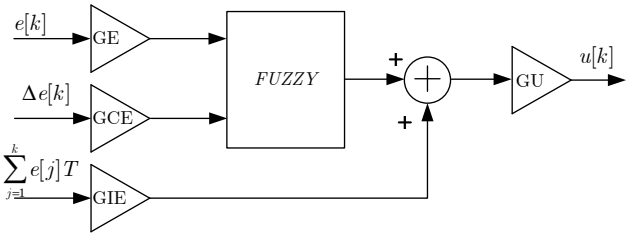


Figura 1.19: Digrama de blocos do controlador fuzzy PD+I.

1.4.2 Algumas considerações relevantes

Um controlador fuzzy baseado em Mamdani possui, internamente, os mesmos blocos da Figura 1.7.

Ao projetar um controlador fuzzy, o engenheiro se depara sempre com dois problemas subjetivos que dependem da preferência de escolha do projetista:

1. Qual formato deve ser utilizado para o conjunto?
2. Quantos conjuntos devem ser utilizados?

Algumas regras que podem orientar nas escolhas são:

- os conjuntos devem possuir uma quantidade razoável de sobreposições. Caso contrário, o controlador pode retornar uma saída indefinida;
- se há uma lacuna na vizinhança de dois conjuntos, então o sinal de controle fica indefinido nesta região;

- a quantidade de conjuntos depende da largura dos conjuntos e da faixa de funcionamento do sistema;
- um conjunto deve ser suficientemente largo para que seja diferenciado o ruído na medida do sinal.

1.5 Preparação da atividade de simulação

A seguir são descritas as atividades de preparação para a experiência 1.

- Veja o material da Disciplina PTC3312 (Lab. de Controle), disponível no Moodle da disciplina.
- Assista ao vídeo `Controle Fuzzy (breve introdução)`, que consiste na aula desse capítulo, versão 2021.

1.5.1 Projeto de controlador nebuloso

Escreva uma rotina em MATLAB de controle nebuloso PD para o servomecanismo do laboratório. Considere como saída a tensão do potenciômetro que mede a posição do eixo do servomecanismo. **Não é permitido utilizar a toolbox de controle nebuloso** para a execução da atividade.
SUGESTÕES:

- Considere os conjuntos *Fuzzy* da Figura 1.20.

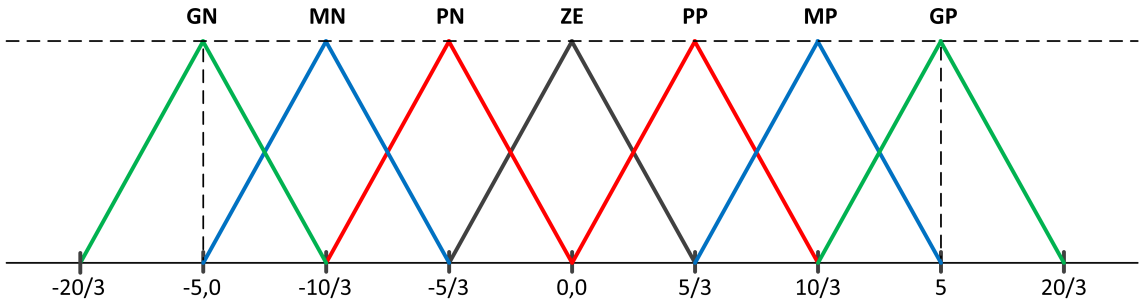


Figura 1.20: Conjuntos *fuzzy*.

- Utilize o conjunto de regras de controle da Tabela 1.3 para a implementação da Máquina de Inferência Nebulosa.

Tabela 1.3: Regras de controle para o servomecanismo

<table>
<tr><td></td><td colspan="8">Erro</td></tr>
<tr><td rowspan="8">Variação do Erro</td><td></td><td>GN</td><td>MN</td><td>PN</td><td>ZE</td><td>PP</td><td>MP</td><td>GP</td></tr>
<tr><td>GN</td><td colspan="2" rowspan="4">GN</td><td colspan="2" rowspan="2">MN</td><td>PN</td><td colspan="2">PP</td></tr>
<tr><td>MN</td><td>PN</td><td>ZE</td><td colspan="2">MP</td></tr>
<tr><td>PN</td><td rowspan="2">MN</td><td>ZE</td><td rowspan="2">MP</td><td colspan="2" rowspan="3">GP</td></tr>
<tr><td>ZE</td><td>ZE</td><td>PP</td></tr>
<tr><td>PP</td><td colspan="2">MN</td><td>ZE</td><td>PP</td></tr>
<tr><td>MP</td><td colspan="2" rowspan="2">PN</td><td rowspan="2">PP</td><td colspan="3" rowspan="2">MP</td></tr>
<tr><td>GP</td></tr>
</table>

1.6 Atividades práticas

A seguir serão descritas as atividades práticas da experiência 1 a serem desenvolvidas no laboratório.

1.6.1 Sistema de Aquisição de Dados e Servomecanismo

Explore o conteúdo copiado de `c:/labaut` e leia o manual "Acesso ao Módulo Lynx AC1160-VA" para familiarização com as funções Matlab de configuração e uso do módulo de aquisição de dados Lynx.

Ligue o servomecanismo conforme a Figura 1.21.

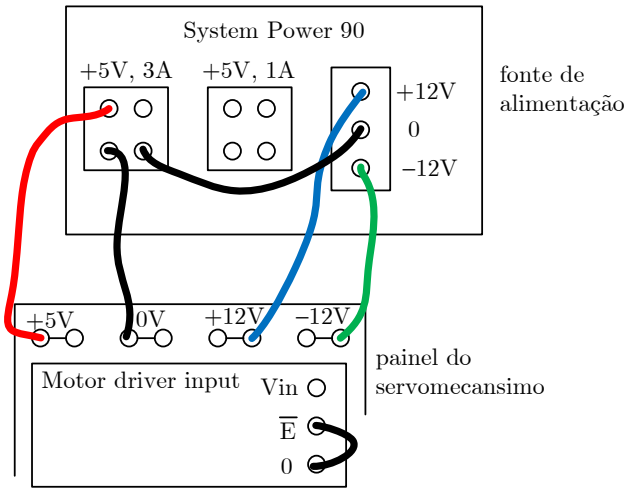


Figura 1.21: Ligação da fonte de alimentação ao servomecanismo.

O esquema de conexão da Figura 1.22 deve ser utilizado para a realização desta prática. A saída de tensão do processo é o potenciômetro que mede a posição do eixo do servomecanismo.

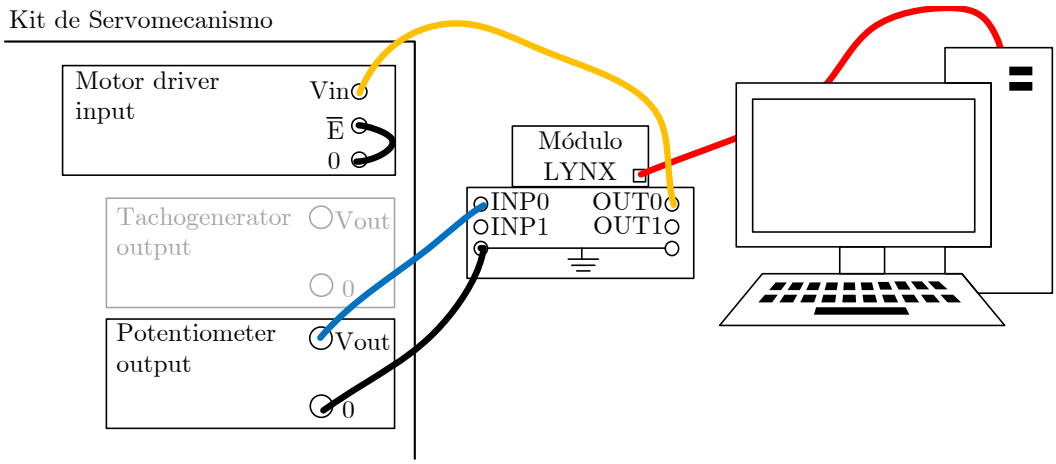


Figura 1.22: Conexão do sistema de aquisição de sinais para a experiência 1.

Algumas observações importantes:

- As entradas e saídas do módulo de aquisição são do pondo de vista do controlador digital (PC), ou seja, o que entra no PC deve ser conectado na(s) entrada(s) INP0 ou/e INP1, e a(s) ação(ões) de controle "sai/saem" do PC pela(s) conexão(ões) de saída OUT0 e OUT1.
- **IMPORTANTE: nunca inverta entradas com saídas.**
- **IMPORTANTE: jamais ligue os terminais OUT0 ou OUT1 diretamente a um terminal de terra ou aos terminais GND.**
- **IMPORTANTE: jamais ligue os terminais OUT0 ou OUT1 a uma fonte de tensão (como um gerador de funções ou a rede elétrica).**
- **IMPORTANTE: jamais ligue os terminais INP0 ou INP1 a fontes de tensão acima da faixa de ± 10V (como, por exemplo, à rede elétrica).**

1.6.2 Implementação do controlador nebuloso

a) Implemente a rotina projetada no item 1.5.1.

b) Fixe o período de amostragem em $T_s = 1/20$ s.

c) Faça o servomecanismo rastrear uma onda quadrada com amplitude $\pi/2$ e período de 20 segundos. Note que a saída é a tensão do potenciômetro. Logo, na referência, a amplitude do sinal deve ser multiplicada por $K_p = 1,6330$.

d) Tente melhorar o desempenho alterando a base de regras da Tabela 1.3.

1.7 Dicas

Crie uma função para retornar o grau de pertinência considerando como entrada os pontos $(a(i), b(i), d(i))$ da i-ésima função triangular e o valor do erro (ou da variação do erro) e como saída o grau de pertinência do valor nesse conjunto:

$$\texttt{functionmu} = \texttt{triang\_membership\_function}(\texttt{a(i)}, \texttt{b(i)}, \texttt{d(i)}, \texttt{x})$$

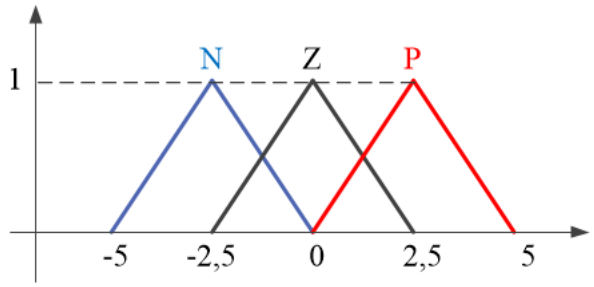
Para a "fuzzyficação", utilize um laço for de 1 a 7 e gere um vetor `mu_e` com o grau de pertinência do valor do erro para cada conjunto fuzzy, utilizando a função `triang_membership_function criada`. Faça o mesmo com o valor da variação do erro.

Para a base de regras:

- Opção 1: utilize 49 condições if para varrer todas as regras;
- Opção 2: solução matricial. Crie uma matriz 7x7, chamada `Mu_s`, que aplica a regra de Mamdani aos valores de `mu_e` e `mu_de` calculados. Por exemplo, com apenas 3 regras, seria algo como:

$$\texttt{Mu\_s} = \begin{bmatrix} \texttt{min(mu\_e(1), mu\_de(1))} & \texttt{min(mu\_e(2), mu\_de(1))} & \texttt{min(mu\_e(3), mu\_de(1))} \\ \texttt{min(mu\_e(1), mu\_de(2))} & \texttt{min(mu\_e(2), mu\_de(2))} & \texttt{min(mu\_e(3), mu\_de(2))} \\ \texttt{min(mu\_e(1), mu\_de(3))} & \texttt{min(mu\_e(2), mu\_de(3))} & \texttt{min(mu\_e(3), mu\_de(3))} \end{bmatrix}$$

Crie uma matriz `B_s` com o valor da abscissa central de saída, de acordo com a tabela de regras. Por exemplo, suponha que os conjuntos fuzzy de saída e a tabela de regras sejam da seguinte forma:



	Erro		
Variação do Erro	P	Z	N
	Z	Z	Z
	P	Z	N

Assim, tal matriz seria do tipo:

$$\texttt{B\_s} = \begin{bmatrix} 2,5 & 0 & -2,5 \\ 0 & 0 & 0 \\ 2,5 & 0 & 2,5 \end{bmatrix}$$

Depois, utilize o método de médias ponderadas, tal que a saída defuzzyficada, em notação Matlab, fica como:

$$\texttt{u} = \texttt{sum(sum(B\_s.} * \texttt{Mu\_s))/sum(sum(Mu\_s))}$$

A Opção 2 funciona bem no Matlab, mas em C/C++ talvez seja melhor fazer com os if's mesmo.

1.8 Relatório

Um relatório desta experiência deverá ser entregue.

2 Introdução aos Controladores Lógicos Programáveis

2.1 Objetivo

Esta experiência tem por objetivo a familiarização com Controladores Lógicos Programáveis (CLPs), com foco nas linguagens de programação Ladder e SFC.

2.2 Introdução teórica

Os CLPs são dispositivos digitais capazes de armazenar instruções para implementação de **funções de controle**, tais como sequências lógicas, temporizações e contagens, bem como realizar **operações lógicas e aritméticas, manipulações de dados e comunicações em rede**.

São largamente utilizados na indústria e no controle de sistemas automatizados [Geo06].

Seus principais componentes são *a unidade central de processamento (CPU)*, *os módulos de I/O* (ou módulos de entrada/saída), *a fonte de alimentação* e *a base*.

- A *CPU* do CLP compreende `o microprocessador`, `o sistema de memória` (ROM e RAM) e `os circuitos auxiliares de controle`.
- Os *módulos de I/O* são dispositivos através dos quais podemos conectar sensores, atuadores ou outros equipamentos à CPU do CLP. Assim, a CPU pode ler sinais de entrada, ou enviar sinais para a saída do CLP através dos módulos de I/O. Esses módulos podem ser discretos ou analógicos.
- A *fonte de alimentação* é responsável pela tensão de alimentação fornecida à CPU e aos módulos de I/O.
- A *base* do CLP proporciona conexão mecânica e elétrica entre a CPU, os módulos de I/O e a fonte. Ela contém o barramento de comunicação entre eles, em que estão presentes os sinais de dados, endereço, controle e tensão de alimentação [Geo06].

A Figura 2.1 ilustra a estrutura de um CLP.

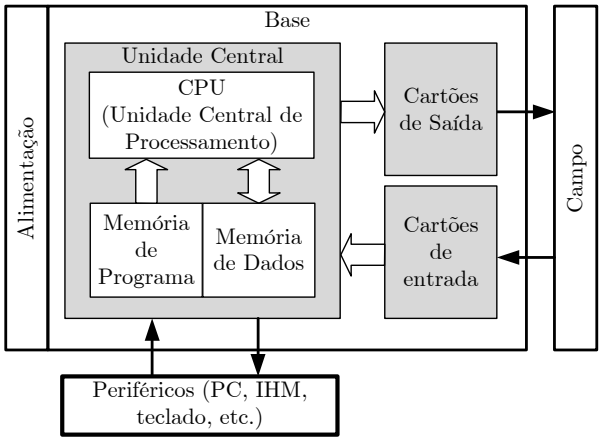


Figura 2.1: Diagrama de um CLP.

Há cinco linguagens de programação de CLPs padronizadas na norma IEC 61131-3. São elas: Lista de Instruções (LT), Diagrama de Blocos Funcionais (FBD), Texto estruturado (ST), Sequenciamento Gráfico de Funções (SFC) e Ladder (LD), sendo SFC e LD as mais populares. A Figura 2.2 apresenta exemplos simples dessas linguagens:

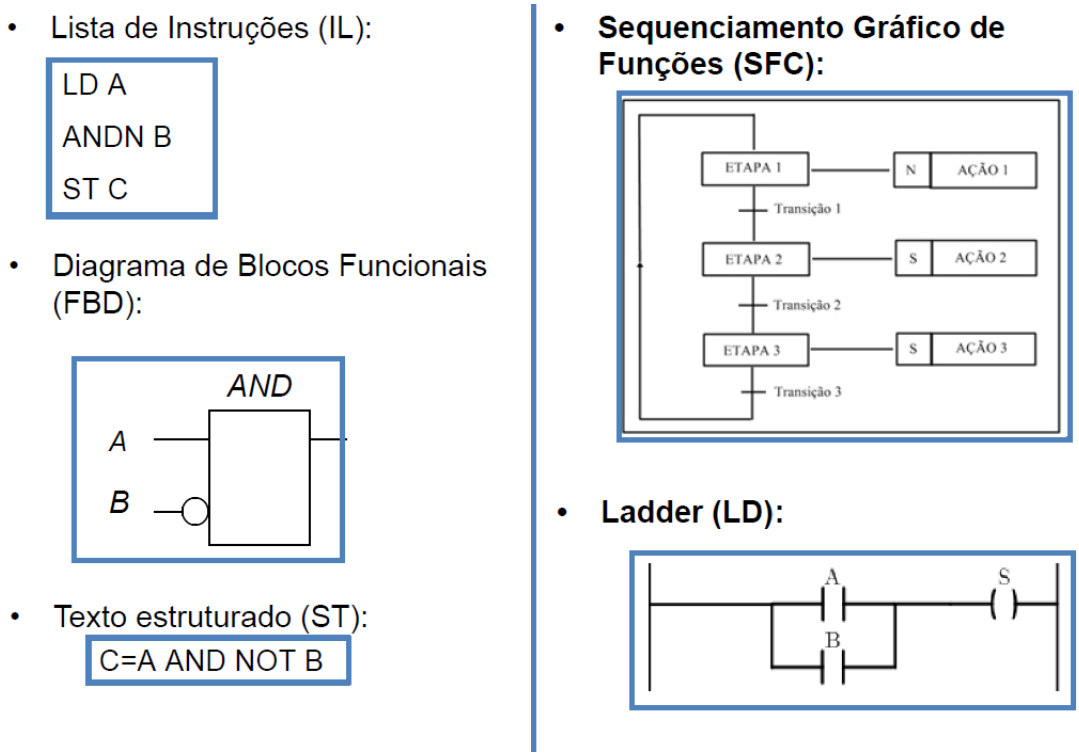


Figura 2.2: Linguagens de programação de CLPs da norma IEC 61131-3).

Para uma introdução mais completa sobre CLPs, o leitor está convidado a consultar [Pru13] e [Geo06].

2.3 Programação Ladder

A linguagem Ladder é, sem dúvida, a mais popular. Ela recebeu este nome devido à sua semelhança com uma escada (ladder), na qual duas barras verticais paralelas são interligadas pela lógica de controle, formando os degraus (*rungs*) da escada [Geo06]. Na Figura 2.3 temos uma representação de lógica de controle através da linguagem ladder:

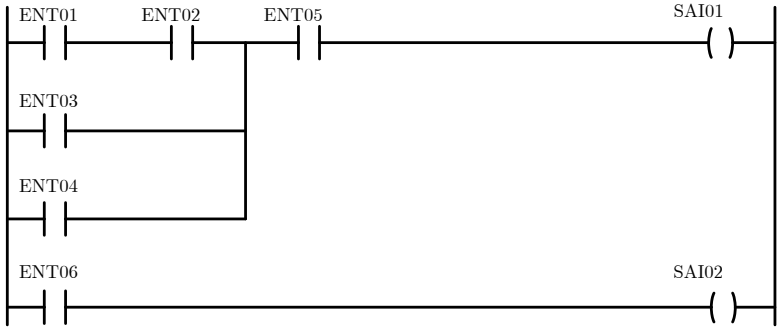


Figura 2.3: Exemplo de diagrama Ladder.

Nota-se uma lógica de controle com dois *rungs*: o primeiro é formado por 3 linhas (primeira linha – ENT01, ENT02, ENT05 e SAI01; segunda linha – ENT03; terceira linha – ENT04) e o segundo é formado por uma linha (ENT06 e SAI02). Este diagrama é formado por contatos e bobinas. As entradas são os contatos, e as saídas são as bobinas.

2.3.1 Contatos de Entrada e Saída

A Tabela 2.1 apresenta as instruções básicas, que são contatos de entrada e de saída (bobinas), da linguagem Ladder.

Tabela 2.1: Instruções de bit.

Contatos	--| |--	normalmente aberto
	--| / |--	normalmente fechado
Bobinas	--( )--	bobina simples
	--(S)--	bobina tipo *set* ou *latched*
	--(R)--	bobina tipo *reset* ou *unlatched*

As bobinas S (liga um bit) e R (desliga um bit) são retentivas, ou seja, manterão seu estado mesmo que as condições de entrada da linha se tornem falsas.

Uma lógica muito comum em programação Ladder é o contado de selo. Ela permite que um sistema seja capaz de se manter energizado, mesmo que um contato não retentivo, que o acionou, seja desativado. A Figura 2.4 ilustra esse conceito.

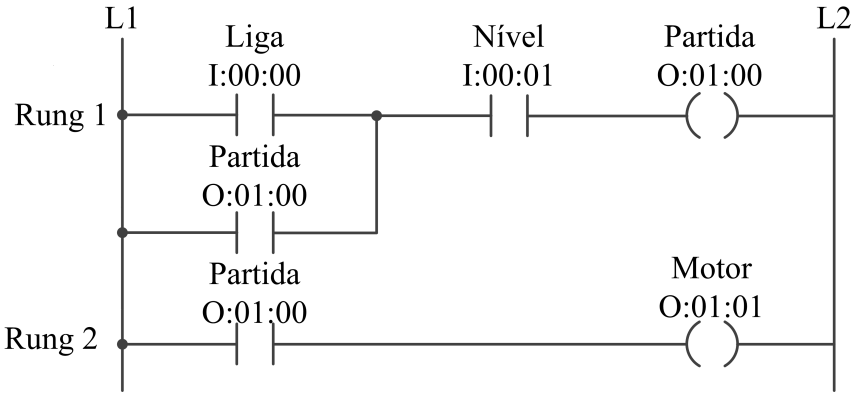


Figura 2.4:

O contato (não retentivo) “Liga” é acionado e a bobina partida é acionada. Mesmo que o contato “Liga” seja desativado, o contato auxiliar partida fecha o selo e mantém a condição da primeira linha (*Rung* 1) ativa (ha uma lógica OU). No (*Rung* 2), o contato auxiliar “Partida” aciona a saída “Motor”, até que o sensor “Nível”, que é fisicamente normalmente fechado, seja desativado.

2.3.2 Algumas Instruções de Temporização

A seguir serão descritas duas instruções bem comuns de temporização presentes no CLP: TON e TOF.

Instrução TON - Temporizador na Energização

O Temporizador na Energização (TON) conta o intervalo de tempo enquanto a condição da linha for verdadeira.

A seguir são descritos alguns bits de estado:

- Bit de habilitação (EN): É verdadeiro se a condição da linha do temporizador for verdadeira. No TON, tal bit é “setado” e permanece assim enquanto a condição da linha for verdadeira.
- Bit de cronometragem (TT): É verdadeiro se o valor do acumulador do temporizador estiver mudando, ou seja, se a linha for verdadeira e o temporizador estiver contando o tempo. Permanece “setado” até que o valor do acumulador (ACC) atingir o valor pré-ajustado (PRE).
- Bit de finalização (DN): Muda de estado se o valor do acumulador (ACC) alcança o valor pré-ajustado (PRE). No TON, tal bit é “setado” quando ACC=PRE. Permanece assim até que a condição da linha seja falsa.

Tais bits também podem ser utilizados como *tags* de contatos no programa Ladder.

Instrução TOF - Temporizador na Desenergização

Por outro lado, o Temporizador na Desenergização (TOF) conta o intervalo de tempo quando a condição de linha passa de verdadeira para falsa. Alguns bits de estado dessas instruções são descritos a seguir:

- Bit de habilitação (EN): É falso enquanto a condição da linha estiver falsa.
- Bit de cronometragem (TT): É verdadeiro se o valor do acumulador do temporizador estiver mudando, ou seja, se a linha for falsa e ACC $<$ PRE.
- Bit de finalização (DN): Tal bit é "resetado" quando ACC $=$ PRE. Permanece assim até que a condição da linha seja verdadeira.

Detalhes específicos dessas instruções, bem como outras instruções de temporização, serão vistos quando o RSLogix 5000 for apresentado.

2.3.3 Algumas Instruções de Contagem

A seguir serão apresentadas duas instruções de contagem muito utilizadas na programação de CLPs.

Instrução CTU - Contador Crescente

Efetua uma contagem ascendente até o valor máximo definido no PRE. O incremento unitário ocorre quando existe transição de falso para verdadeiro na linha. Para zerar o contador, utiliza-se uma instrução instrução de reset (RES) com o mesmo endereço do contador.

Alguns bits de estado são descritos a seguir. Tais bits também podem ser utilizados como *tags* de contatos no programa Ladder.

- Bit de Overflow (OV): É "setado" quando o valor do ACC chegar no limite máximo de contagem.
- Bit de finalização (DN): É verdadeiro se ACC $\geq$ PRE. Permanece assim até que ACC $<$ PRE.
- bit de habilitação (CU): É "setado" se as condições da linha são verdadeiras. Permanece "setado" até que as condições da linha sejam falsas ou RES seja executado.

Instrução CTD - Contador Decrescente

Efetua uma contagem descendente até o valor máximo definido no PRE. O incremento unitário ocorre quando existe transição de falso para verdadeiro na linha. Para zerar o contador, utiliza-se uma instrução instrução de reset (RES) com o mesmo endereço do contador.

Os principais bits de estado do CTD são semelhantes aos CTU. Detalhes serão vistos nas instruções específicas do software RSLogix 5000.

Outras instruções também serão vistas quando o RSLogix 5000 for apresentado.

2.4 Programação SFC

A linguagem SFC (*Sequential Function Chart*) tem mais a ver com uma técnica de programação do que com uma linguagem propriamente dita. É usada para modelar lógicas de controle baseadas na sequência temporal de eventos de processo [Pru13]. Surgiu em 1977 na França, onde foi chamada de GRAFCET. Tem estreita relação com a rede Petri.

Um ciclo industrial é formado por passos e transições. Cada passo possui um determinado número de operações. A passagem de um passo para outro denomina-se transição. Uma transição ocorre quando certas condições são satisfeitas. A Figura 2.5 apresenta um exemplo de passos com transição.

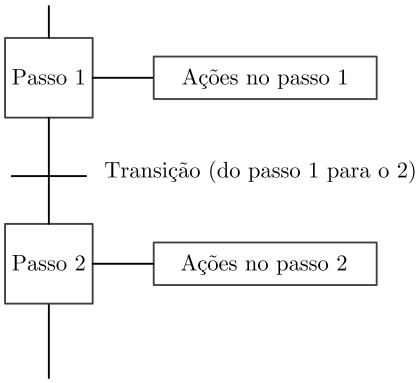


Figura 2.5: Esquema de um fragmento de código SFC.

Na Figura 2.6 são descritos alguns símbolos e gráficos, segundo a norma IEC 60848 (*GRAFCET specification language for sequential function charts*).

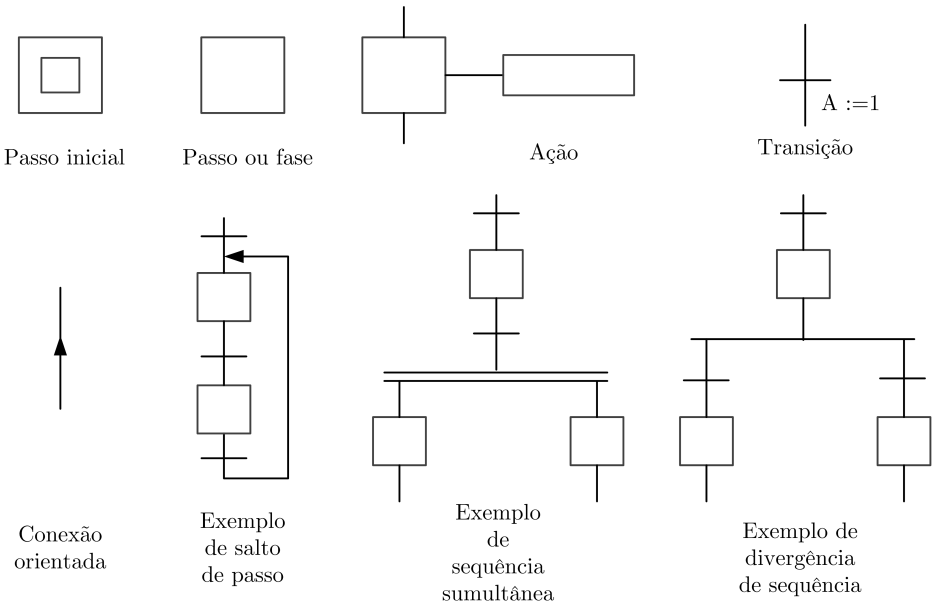


Figura 2.6: Alguns símbolos SFC.

Para elaboração de um programa em SFC, algumas regras devem ser respeitadas [Pru13]:

- deve-se ter pelo menos um passo inicial;

- deve-se ter alternância entre passo e transição;
- a superação de uma transição representa a finalização das ações do passo anterior e a ativação das ações do(s) passo(s) seguinte(s).

Pode-se realizar sequências alternativas através de caminhos divergentes e convergentes, conforme mostrado na Figura 2.7. Caso mais de uma transição seja TRUE, a sequência situada mais à esquerda terá prioridade de execução.

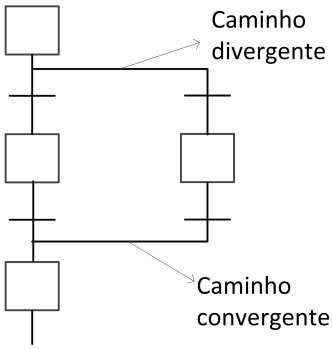


Figura 2.7: Exemplo de ramos divergentes.

Pode-se realizar sequências simultâneas como mostrado na Figura 2.8. A Execução é de forma paralela e a convergência acontece somente quando todos os últimos passos de cada ramo forem ativados.

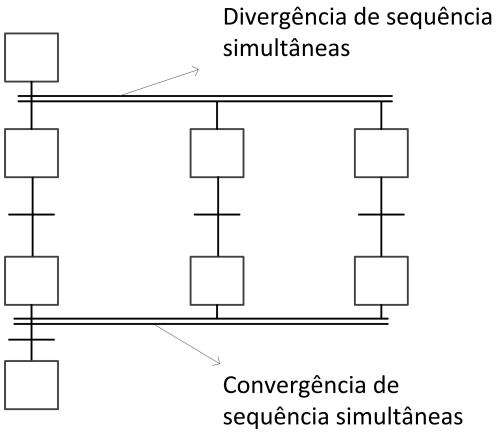


Figura 2.8: Exemplo de ramos simultâneos.

Em relação a ações, tem-se **ações IEC** e **ações de passos** que representam um extensão das ações IEC.

As ações IEC são representadas por uma caixa bipartida, conectada à direita de um passo via uma linha de conexão. No lado esquerdo, é exibido o qualificador da ação e, no direito, o nome da ação, conforme ilustrado na Figura 2.9.

A Tabela 2.2 apresenta os tipos de qualificadores de ações IEC.

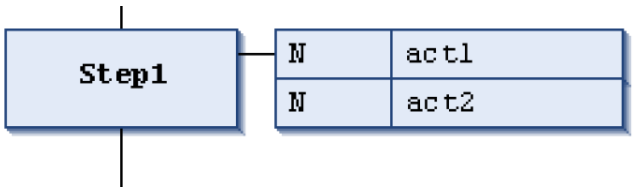


Figura 2.9: Diagrama de representação de ações IEC.

Tabela 2.2: Tipos de qualificadores de ações IEC.

QUALIFIC.	**NOME**	**DESCRIÇÃO**
N	Não-armazena	A ação está ativa enquanto o passo estiver ativo.
S	Set	Memoriza uma ação ativa. A ação continuará a ser executada até um qualificador R ser encontrado.
R	Reset	Desliga uma ação memorizada.
L	Limita no tempo	Termina após um período estipulado.
D	Atrasa no tempo	Começa após um período de tempo
P	Pulso	Executada uma única vez quando a etapa é ativada.
SD	Armazena e atrasa no tempo	A ação é memorizada após um tempo estipulado, mesmo que a respectiva etapa seja desativada antes do tempo de atraso
DS	Atrasa e armazena	A ação é atrasada e armazenada. Se a etapa é desativada antes do período de atraso, a ação não é armazenada
SL	Armazena e limita. no tempo	A ação é iniciada e executada por um período de tempo.

Uma observação importante é que, após a desativação de uma ação IEC, a mesma será executada ainda mais uma vez no próximo ciclo de varredura. Portanto, caso se queira incrementar um contador a cada vez que um passo é executado, não se deve utilizar ações tipo IEC, pois haverá dois incrementos, um na execução do passo e outro ao sair do passo.

2.5 O CLP do laboratório e o Ambiente de Programação

Os CLPs do laboratório são do fabricante Allen-Bradley, modelo CompactLogix 1769-L30ER. Eles podem ser programados, configurados e monitorados, via interface de comunicação USB, diretamente dos PCs, utilizando-se o programa RSLogix 5000.

A Figura 2.10 ilustra o CLP do laboratório.

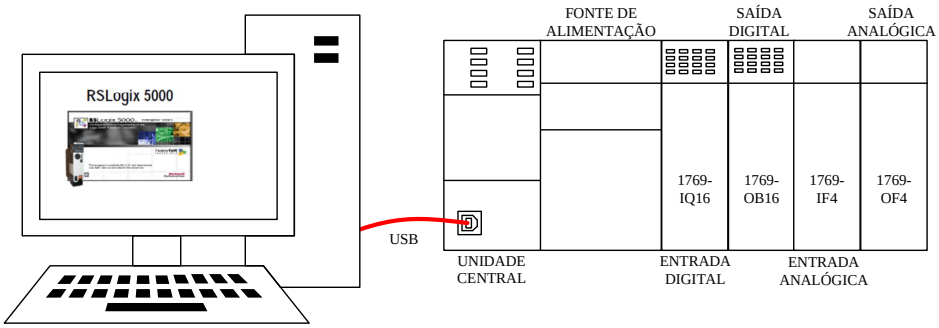


Figura 2.10: CLP do laboratório.

2.5.1 Como iniciar uma seção de programação

A seguir são apresentados os passos para criação de um novo projeto e elaboração de um programa básico em linguagem Ladder.

1. Conecte o CLP ao PC via cabo USB.

2. Abra o programa RSLogix 5000. Isso pode demorar alguns segundos;

3. Criação de um novo projeto.

   Clique em `New Project` e escolha o modelo `1769-L30ER`, como mostrado na Figura 2.11. Atribua um nome ao projeto.

   Observações:

   - Ao nomear projetos, *tags*, rotinas, módulos de E/S, etc, deve-se usar apenas letras, números e *underline* (“_”), onde o primeiro caractere não pode ser um número;
   - Não há distinção entre letras maiúsculas e minúsculas.

4. Selecione a porta de comunicação com o CLP a ser empregada, que no nosso caso é USB. Clique no botão Who Active, conforme apresentado na Figura 2.12. Selecione `Set Project Path`. Isso é necessário para posteriormente fazer o download do programa no CLP. Feche a janela.

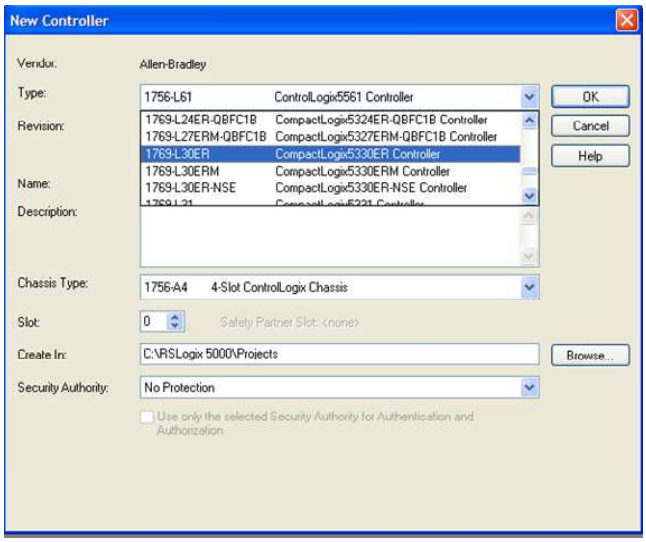


Figura 2.11: Criação do projeto e escolha do CLP.

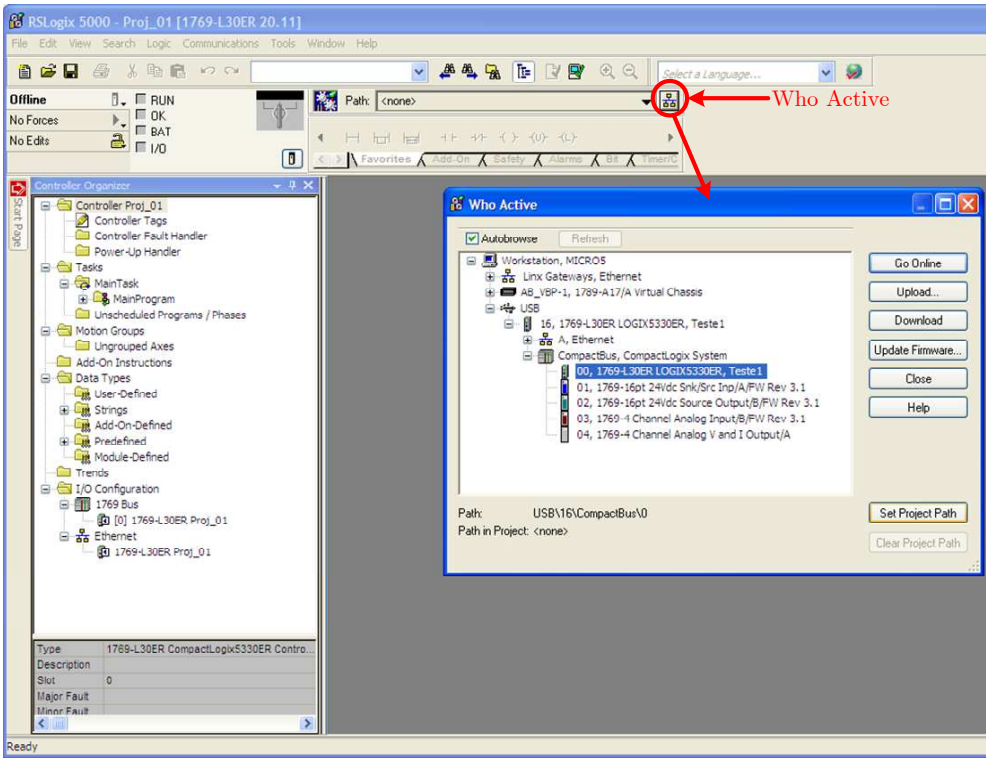


Figura 2.12: Definição da porta de comunicação USB.

5. Adicionar cartões ao CLP.

   Clique com o botão da direita em `I/O Configuration -> 1769 Bus`, e selecione `New Module...`, conforme a Figura 2.13.

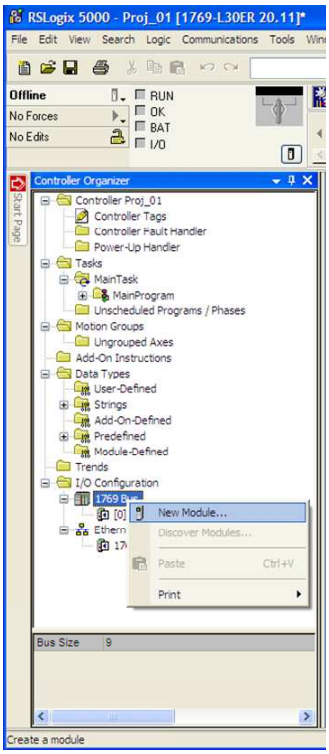


Figura 2.13: Configuração dos cartões do CLP.

No chassi do CLP há quatro *slots* para cartões de I/O. A seguinte montagem está disponível no laboratório:

Código	Descrição Simplificada	Slot
1769-IQ16	16 Entradas Digitais, 24Vdc	1
1769-OB16	16 Saídas Digitais, 24Vdc	2
1769-IF4	4 Entradas Analógicas	3
1769-OF4	4 Saídas Analógicas	4

Observações:

- Os módulos devem ser adicionados **respeitando a ordem** que se encontram no chassi do CLP, da esquerda para a direita;
- **Não inverter a posição** de um cartão no slot do CLP.

Na tela `Select Module Type`, deve-se adicionar cada um dos quatro cartões separadamente em seus respectivos *slots*. Depois de encontrado o módulo, selecione `Create`. Como exemplo, para selecionar o módulo de entrada digital na posição 1, faça como apresentado na Figura 2.14.

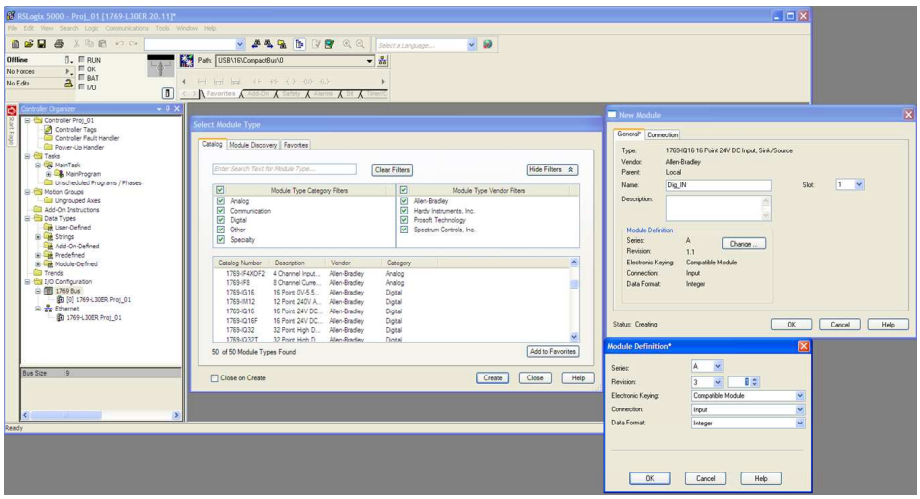


Figura 2.14: Configuração dos cartão de entrada digital.

- Na tela `New Module`, coloque um `Alias` para o cartão. Ex: `Dig_IN`
- O cartão de Entrada Digital está no *Slot* 1
- Clique em `Module Definition -> Change Module` e escolha a revisão mais recente, mas deixe-a na opção `Compatible Module`.

Como o endereçamento foi feito criando-se um *alias*, quando for necessário utilizar elementos já existentes, basta fazer um duplo clique no símbolo "?" do elemento e digitar o *alias*.

Repita as operações para os três cartões restantes.

Pronto, o CLP está configurado e o primeiro programa pode ser efetuado!

6. Para criar uma nova rotina de programa em Ladder, selecione a opção `Tasks -> MainTask -> MainRoutine`. Note que abrirá uma tela para iniciar o programa. No menu superior aparacerá uma série de contatos e componentes para compor o programa.

   No CLP CompactLogix um endereço segue o formato `Local:Slot:Tipo.Membro.Bit`, conforme o quadro abaixo:

`Local`	Indica que a localização do módulo de E/S está no mesmo rack do CLP
`Slot`	Número do Slot de E/S no rack
`Tipo`	`I`: entrada (input) `O`: saída (output) `C`: configuração
`Membro`	Para um módulo de E/S discretas, um membro do tipo Data armazena os valores dos bits de E/S
`Bit`	Ponto específico de um módulo de E/S discretas

2.5.2 Meu primeiro programa em Ladder no RSLogix 5000

A seguir são apresentados os passos para a construção de um programa bem simples, que consiste em acender uma lâmpada (ligar um contato de saída) quando um botão (contato de entrada normalmente aberto — NA) é acionado.

a) **Inserção de um contato NA.** Adicione um contato de entrada clicando em seu símbolo na barra de ferramentas. O contato será endereçado criando-se um "*alias*". Clique com o botão direito do mouse sobre o símbolo "?" do contato e selecione `New Tag`. A tela da Figura 2.15 será aberta.

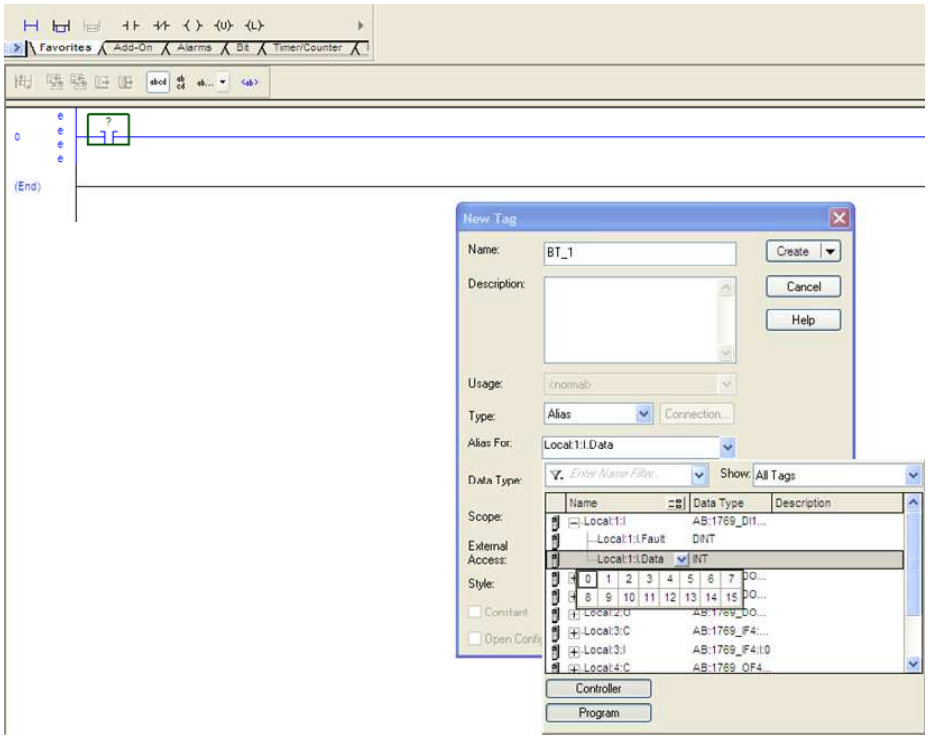


Figura 2.15: Configuração e endereçamento do contato de entrada para o botão.

Escolha as seguintes configurações:

- `Name`: Escolha um nome para a tag do botão (Ex: BT_1);
- `Type`: Alias;
- `Alias For`: selecione `Local:1:I.Data`. Clique na flecha para baixo e escolha a entrada digital referente ao botão (0 a 7). Por exemplo, se a entrada 0 for selecionada, deve aparecer o endereço `Local:1:I.Data.0`;
- `Data Type`: BOOL.

b) **Inserção de um contato de saída.** Adicione um contato de saída clicando em seu símbolo na barra de ferramentas. Clique com o botão direito do mouse sobre o símbolo "?" do contato e selecione `New Tag`. A tela da Figura 2.16 será aberta.

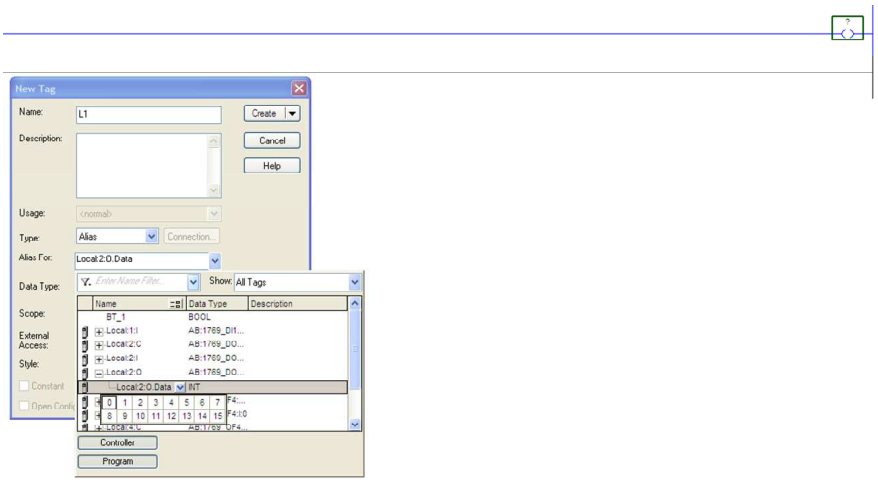


Figura 2.16: Configuração e endereçamento do contato de saída para a lâmpada.

Escolha as seguintes configurações:

- `Name`: Escolha um nome para a tag do botão (Ex: L_1);
- `Type`: Alias;
- `Alias For`: selecione `Local:2:O.Data`. Clique na flecha para baixo e escolha a saída digital referente à Lâmpada (0 a 7). Por exemplo, se a saída 0 for selecionada, deve aparecer o endereço `Local:2:O.Data.0`;
- `Data Type`: BOOL.

Se tudo estiver certo, as indicações de erro no lado esquerdo das linhas do programa desaparecerão.

c) Salve o programa. Em seguida, clique na flecha ↓ ao lado de `Offline` no menu superior e selecione a opção `Download`. Verifique as mensagens.

d) Faça as ligações físicas no CLP, conforme apresentado na Figura 2.17. Selecione a opção `Run Mode` para executar o programa. Para modificar o código, vá novamente para o modo `Offline`.

2.5.3 Modos de funcionamento do CLP

A seguir é apresentada uma descrição sucinta dos modos de operação do CLP.

- Modo RUN: modo em que o programa carregado é atualizado e as saídas são atualizadas. Não permite edição do programa;
- Modo PROG: modo de criação e edição de códigos *offline*, ou seja, as saídas estão desabilitadas;
- Modo REM: modo remoto, com as seguintes opções:

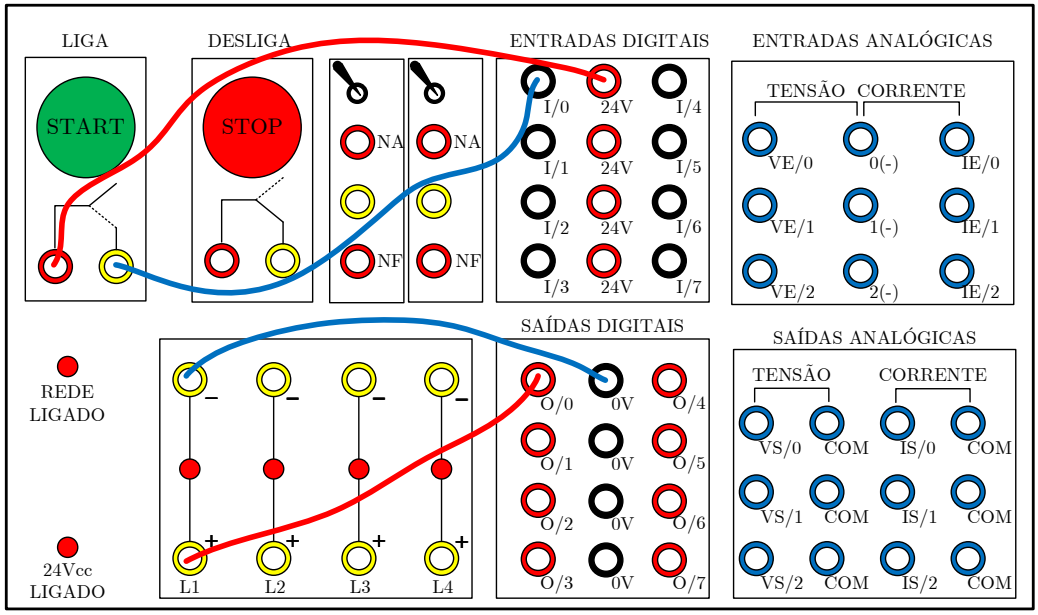


Figura 2.17: Configuração e endereçamento do contato de saída para a lâmpada.

- REMOTE RUN: modo onde o programa carregado é executado e as saídas são atualizadas, mas permite a edição *online*;
- REMOTE PROG: permite a edição online do programa, mas as saídas estão desabilitadas.
- REMOTE TEST: semelhante ao REMOTE RUN, mas com saídas desabilitadas.

O CLP a ser utilizado no laboratório está no modo REM. Como sugestão, para criar/editar o programa, deixe-o em modo Offline.

2.5.4 Funções básicas em Ladder do RSLogix 5000

A Tabela 2.3 apresenta as instruções de bit (contatos e bobinas) no editor Ladder do RSLogix 5000:

Tabela 2.3: Instruções de bit.

Contatos	--| |--	normalmente aberto
	--| / |--	normalmente fechado
Bobinas	--( )--	bobina simples
	--(L)--	bobina tipo *latched*
	--(U)--	bobina tipo *unlatched*

As bobinas L (liga um bit) e U (desliga um bit) são retentivas, ou seja, manterão seu estado mesmo que as condições de entrada da linha se tornem falsas. As bobinas L e U são, respectivamente, equivalentes a bobinas do tipo S (set) e R (reset).

O comando $|-|$ insere uma nova linha de comando no diagrama Ladder.

O comando $|\overline{|\cdot|}|$ insere um ramo paralelo (*branch*) no diagrama Ladder.

No Ladder do RSLogix há uma instrução denominada ONS (*one shot*), com símbolo $\_[\mathrm{ONS}]\_$. De forma simplificada, essa expressão gera um pulso de duração de um período de *scan* quando a linha é energizada. Para gerar outro pulso, a linha precisa ser desenergizada e energizada novamente.

2.5.4.1 Algumas Instruções de Temporização do RSLogix

A seguir serão descritas três instruções de temporização presentes no CLP: TON e TOF e RTO. As duas primeiras instruções são não retentivas, ou seja, o acumulador de contagem de tempo é "resetado" quando o contador não está habilitado. Já a terceira, trata-se de um temporizador retentivo, que mantém o valor do contador quando desabilitado.

Instrução TON - Temporizador na Energização

A Figura 2.18 ilustra o diagrama Ladder da instrução

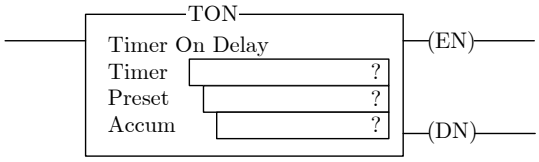


Figura 2.18: Instrução TON.

É utilizada para ligar ou desligar uma saída após um tempo especificado no valor Preset (PRE). A contagem de tempo se inicia quando a condição de linha é verdadeira. O Acumulador (ACC) é incrementado a cada ciclo de scan enquanto a linha permanece verdadeira, até que o valor de PRE seja alcançado. O ACC é "resetado" quando a condição da linha se torna falsa, independentemente do valor de PRE ter sido atingido. A base de tempo é em milissegundos (ms).

A tabela 2.4 apresenta a descrição dos bits de estado.

Tabela 2.4: Bits de estado de TON

Bit	Descrição
Timer Done	É "setado" quando o valor do ACC é igual ao PRE
(DN)	Permanece "setado" até que as condições da linha sejam falsas
Timer Timing	É "setado" se as condições da linha são verdadeiras e ACC $<$ PRE
(TT)	Permanece "setado" até que as condições da linha sejam falsas ou DN seja "setado"
Timer Enable	É "setado" se as condições da linha são verdadeiras
(EN)	Permanece "setado" até que as condições da linha sejam falsas

Tais bits podem ser utilizados como *tags* de contatos no programa Ladder. Por exemplo, se um timer for denominado `T1`, pode-se atribuir a *tag* `T1.DN` a um contato.

Ao inserir um bloco TON, deve-se atribuir uma tag ao temporizador, como ilustrado na Figura 2.19

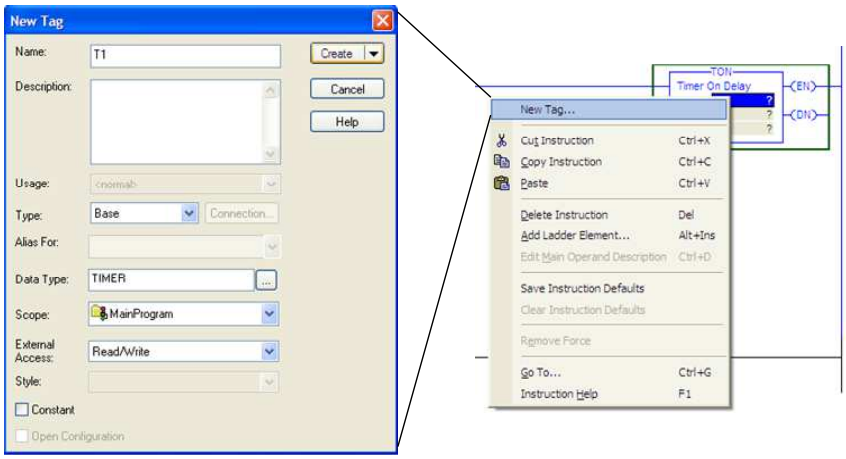


Figura 2.19: Atribuição de tag ao módulo TON.

Em seguida, atribua um valor para o Preset e, se for necessário, ao Acumulador, lembrando que a base de tempo é em milissegundos. Procedimentos similares devem ser executados para os demais temporizadores e contadores.

Instrução TOF - Temporizador na Desenergização

A Figura 2.20 ilustra o diagrama Ladder da instrução.

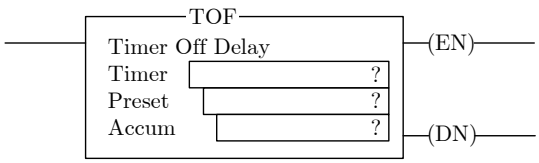


Figura 2.20: Instrução TOF.

Também utilizada para ligar ou desligar uma saída após um tempo especificado no valor Preset (PRE). A contagem de tempo se inicia quando a condição de linha passa de verdadeira para falsa. O Acumulador (ACC) é incrementado a cada ciclo de scan enquanto a linha permanece na condição falsa, até que o valor de PRE seja alcançado. O ACC é "resetado" quando a condição da linha se torna verdadeira, independentemente do valor de PRE ter sido atingido.

A tabela 2.5 apresenta a descrição dos bits de estado.

Tais bits também podem ser utilizados como *tags* de contatos no programa Ladder.

- **Instrução RTO - Temporizador Retentivo**

A Figura 2.21 ilustra o diagrama Ladder da instrução

Também utilizada para ligar ou desligar uma saída após um tempo especificado no valor Preset (PRE). A diferença é que o valor do acumulador é mantido quando a condição da linha se torna falsa,

Tabela 2.5: Bits de estado de TOF

Bit	Descrição
Timer Done (DN)	É “resetado” quando o valor do ACC é igual ao PRE Permanece “resetado” até que as condições da linha sejam verdadeiras
Timer Timing (TT)	É “setado” se as condições da linha são falsas e ACC $<$ PRE Permanece “setado” até que as condições da linha sejam verdadeiras ou DN seja “resetado”
Timer Enable (EN)	É “resetado” se as condições da linha são falsas Permanece “resetado” até que as condições da linha sejam verdadeiras

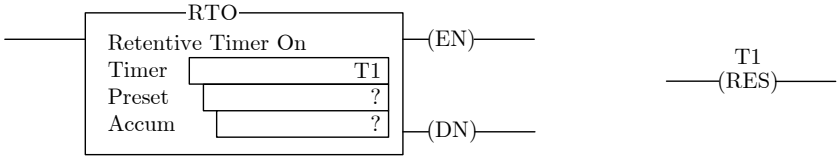


Figura 2.21: Instrução RTO.

quando o modo de programação passa de RUN para PROGRAM, quando a alimentação é perdida ou uma falha ocorrer. A memória acumulada é, portanto, não volátil. Ao voltar para o modo RUN, a contagem é restabelecida a partir do valor armazenado.

Para “resetar” os bits de estado e o valor do ACC, é necessário programar uma instrução de reset (RES) com o endereço do temporizador em uma outra linha do código Ladder.

A descrição dos bits de estado é igual ao do TON apresentada na Tabela 2.4. Tais bits também podem ser utilizados como *tags* de contatos no programa Ladder.

2.5.4.2 Algumas Instruções de Contagem do RSLogix

A seguir serão apresentadas duas instruções de contagem utilizadas na programação do CLP: CTU e CTD. A primeira se trata de um contador crescente, enquanto a segunda representa um contador decrescente.

Instrução CTU - Contador Crescente

A Figura 2.22 ilustra o diagrama Ladder da instrução.

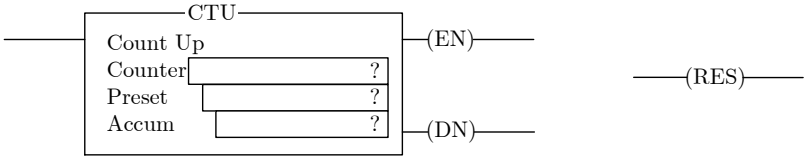


Figura 2.22: Instrução CTU.

Em tal instrução, o valor do acumulador é incrementado quando ocorrem transições falso $\rightarrow$ verdadeiro da linha, que podem ser devidas a eventos internos de programação ou dispositivos externos, como

botões, sensores de presença, etc. O valor do acumulador, assim como dos bits de estado, são retidos quando a linha torna-se falsa. Portanto, para "resetar" o contador, deve-se programar uma instrução de reset (RES) com o mesmo endereço do contador em outra linha.

A tabela 2.6 apresenta a descrição dos bits de estado. Tais bits podem ser utilizados como *tags* de contatos no programa Ladder.

Tabela 2.6: Bits de estado de CTU

Bit	**Descrição**
Overflow	É "setado" quando o valor do ACC $> +2.147.483.647$
(OV)	Permanece "setado" até que RES seja executado ou seja decrementado utlizando CTD
Done	É "setado" se ACC $\geq$ PRE
(DN)	Permanece "setado" até que ACC $<$ PRE
Count Up Enable	É "setado" se as condições da linha são verdadeiras
(CU)	Permanece "setado" até que as condições da linha sejam falsas ou RES seja executado

Instrução CTD - Contador Decrescente

A Figura 2.23 ilustra o diagrama Ladder da instrução.

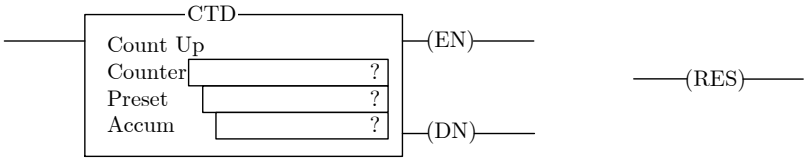


Figura 2.23: Instrução CTD.

Em tal instrução, o valor do acumulador é decrementado quando ocorrem transições falso $\rightarrow$ verdadeiro da linha, que também podem ser devidas a eventos internos de programação ou dispositivos externos, como botões, sensores de presença, etc. O valor do acumulador, assim como dos bits de estado, são retidos quando a linha torna-se falsa. Portanto, para "resetar" o contador, deve-se programar uma instrução de reset (RES) com o mesmo endereço do contador em outra linha.

A tabela 2.7 apresenta a descrição dos bits de estado. Tais bits também podem ser utilizados como *tags* de contatos no programa Ladder.

Tabela 2.7: Bits de estado de CTD

Bit	Descrição
Underflow (UN)	É "setado" quando o valor do ACC $< -2.147.483.648$ Permanece "setado" até que RES seja executado ou seja incrementado utilizando CTU
Done (DN)	É "setado" se ACC $\geq$ PRE Permanece "setado" até que ACC $<$ PRE
Count Up Enable (CU)	É "setado" se as condições da linha são verdadeiras Permanece "setado" até que as condições da linha sejam falsas ou RES seja executado

2.5.4.3 Algumas Instruções de Comparação do RSLogix

Instrução EQU - *Equal to*

A Figura 2.24 ilustra o diagrama Ladder da instrução. A linha é energizada se a condição de *source A* for igual à *source B*.

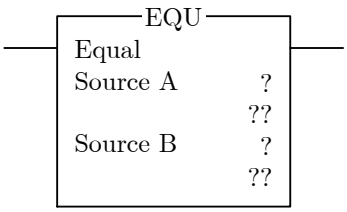


Figura 2.24: Instrução EQU.

Há também as instruções GEQ (*Greater Than or Equal To*), GRT (*Greater Than*), LEQ (*Less Than or Equal To*), LES (*Less Than*), com a mesma sintaxe da instrução EQU. Há ainda a instrução CMP (*Compare*) que verifica se a condição de uma dada expressão é verdadeira para energizar a linha. Todas as fuções de comparação descritas anteriormente podem ser implementadas com a CMP.

Para verificar a gama completa de instruções do RSLogix5000, favor consultar o manual de referência *Logix5000 Controllers General Instructions Reference Manual*.

2.5.5 SFC do RSLogix 5000

O exemplo a seguir ilustra um programa SFC no RSLogix 5000 [AB16].

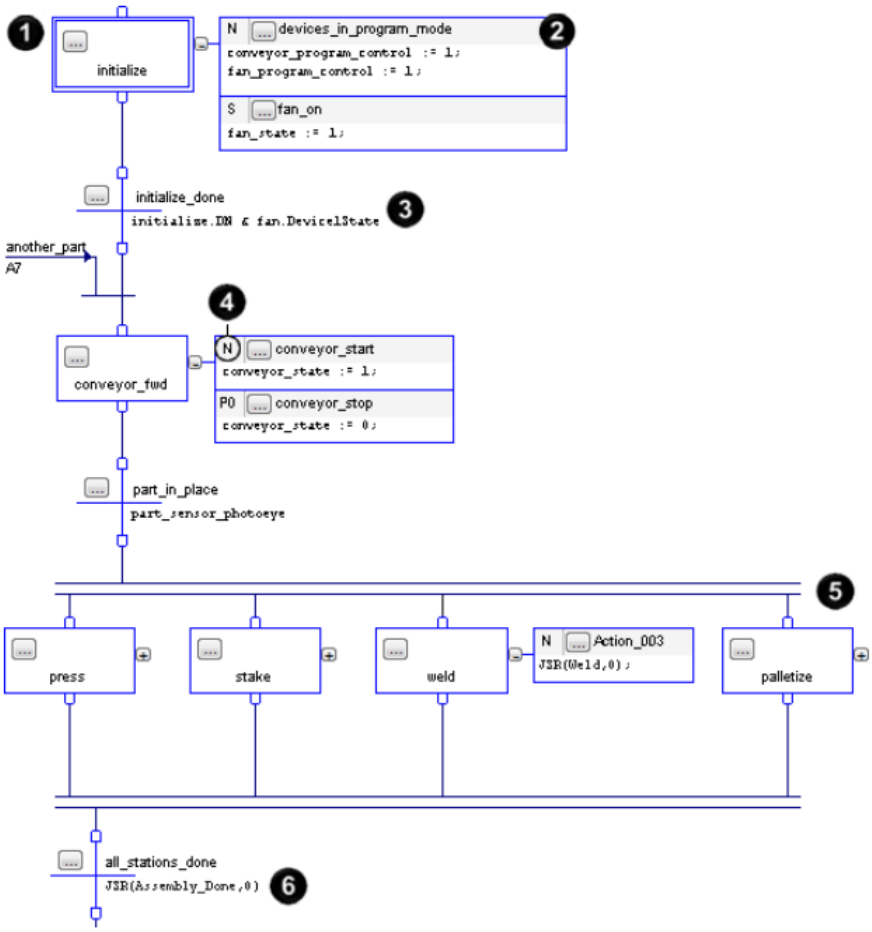


Figura 2.25: Exemplo de programa SFC. Fonte: [AB16].

Os itens da Figura 2.25 são apresentados a seguir:

1. O **passo** representa um estado e contém as ações a serem executadas em um dado instante;
2. Uma **ação** é uma das funções executadas por um passo;
3. Uma **transição** é uma condição VERDADEIRA ou FALSA que determina a passagem de um passo a outro;
4. Um **qualificador** determina quando a ação começa e termina;
5. Um **ramo simultâneo** executa mais de um passo de uma vez;
6. A **instruçao JSR** chama uma sub-rotina;

Ao iniciar uma nova rotina no RSLogix 5000, automaticamente o programa escolhe a linguagem Ladder. Para criar uma rotina em SFC, a *main routine* em Ladder deve ser apagada e, em seguida, uma

rotina SFC deve ser criada, conforme as Figuras 2.26 e 2.27.

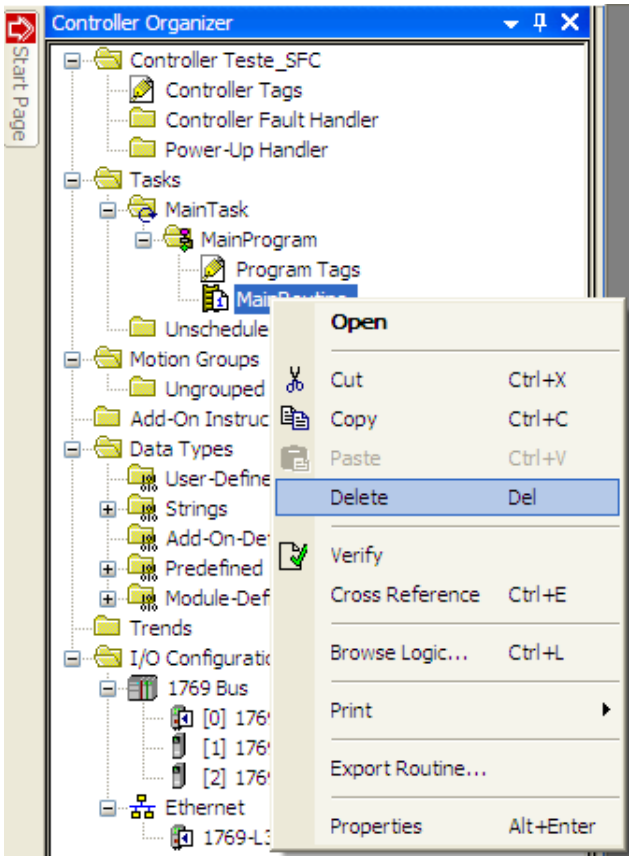


Figura 2.26: Apagar rotina Ladder.

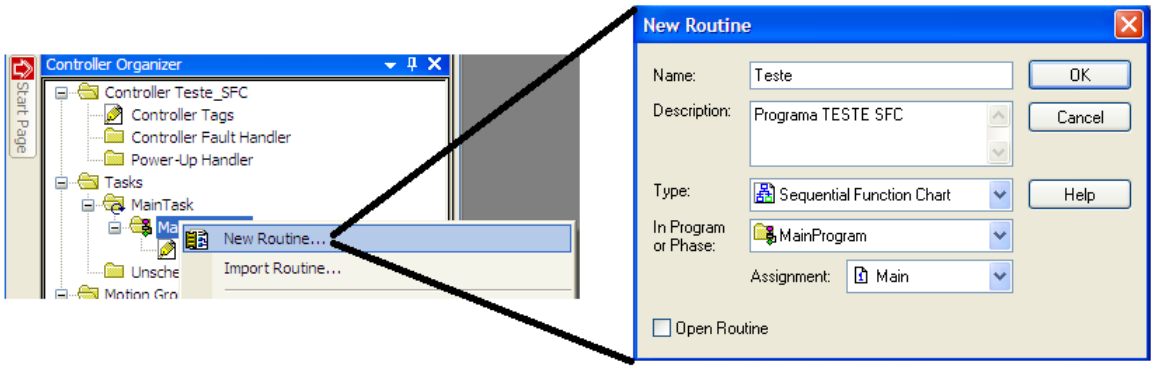


Figura 2.27: Criar rotina SFC.

O *template* inicial do programa SFC é apresentado na Figura 2.28.

Para acrescentar passos e transições, acesse a barra de menu superior do RSLogix 5000. Para acrescentar ações a um passo, clique com o botão direito no passo e selecione *Add Action*, ou clique no campo correspondente no menu superior.

Para programar uma ação, o RSLogix 5000 só aceita a linguagem de Texto Estruturado (ST). Por exemplo, `L1:=0;  L2:=1;`. Para escrever o código, clique no símbolo "?", conforme a Figura 2.29.

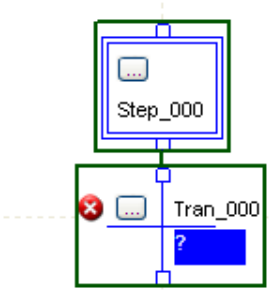


Figura 2.28: Template inicial da rotina em SFC.

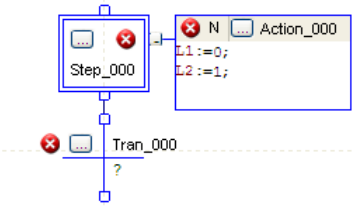


Figura 2.29: Inserção de código ST na ação.

Para definir uma *tag*, basta clicar com o botão direito do mouse sobre a variável e em *New Tag*, como ilustrado na Figura 2.30. As *tags* também podem ser definidas antes da edição do programa pela opção *Edit Tags* na janela principal do RSLogix 5000.

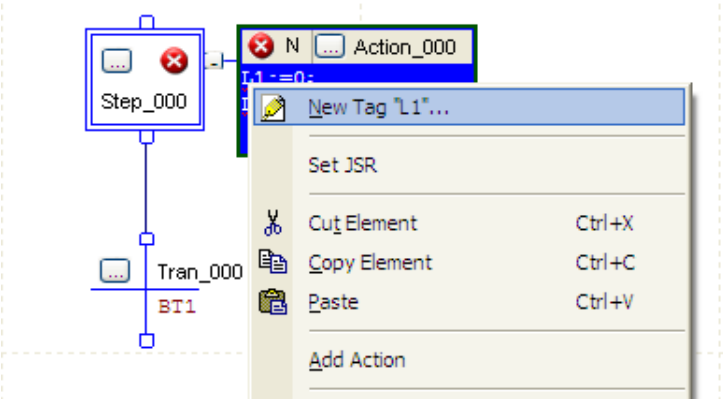


Figura 2.30: Criando *tags*.

Para definir o qualificador a ser utilizado na ação (que não é ação IEC), clique no símbolo "...". Os três principais qualificadores são:

- N: executa continuamente enquanto o passo estiver ativo;
- P1: executa uma única vez quando o passo é ativado;
- P0: executa uma única vez quando o passo é desativado.

Para definir a condição de una transição, clique no símbolo "?" e adicione o código ST.

A conexão entre passos e transições é feita posicionando-se o mouse sobre o ponto de entrada ou saída do elemento até o aparecimento de um círculo verde. Clique sobre o mesmo e então arraste-o até o ponto de conexão desejado.

Um passo possui algumas variáveis internas que podem ser utilizadas no código, por exemplo, como temporizadores ou contadores. As principais são:

- `Step_001.X` : indica se o passo 001 está ativo.
- `Step_001.Count`: indica quantas vezes o passo 001 foi ativado.
- `Step_001.T`: determina por quanto tempo o passo está ativo, em milisegundos. Tem valor indefinido quando o passo não estiver ativado.

2.5.6 Problemas com Solução

Ladder

Problema 1: Verificar através de botões e de uma lâmpada a tabela da verdade da função ou-exclusivo. Elabore um programa em linguagem Ladder e teste no CLP.

Solução: Inicialmente, as seguintes ligações apresentadas na Figura 2.31 são efetuadas.

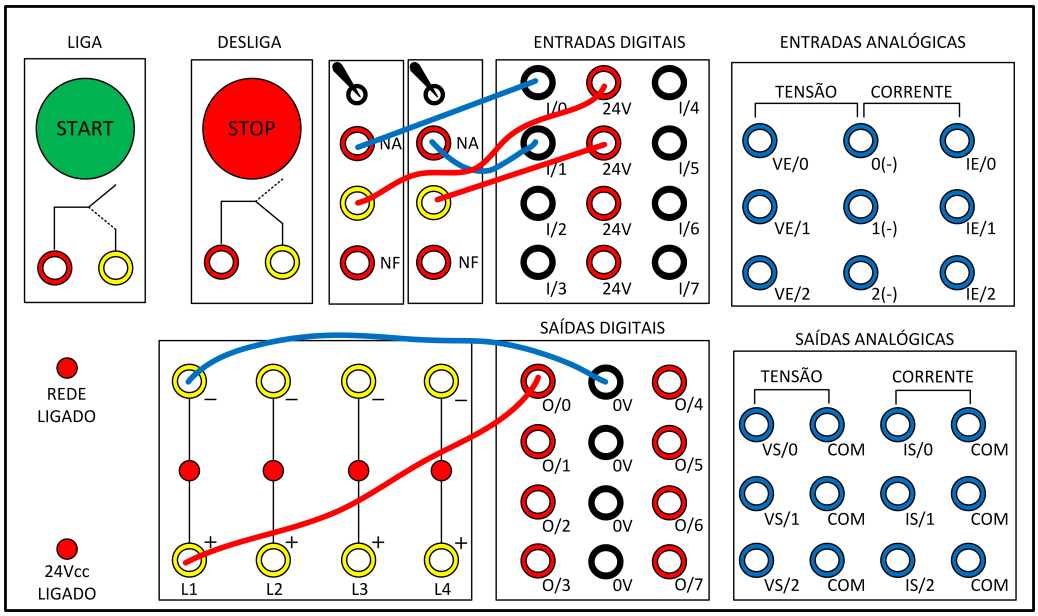


Figura 2.31: Ligações do exemplo.

Passo 1: Identifique as entradas e saídas do sistema.

As seguintes entradas são identificadas:

- $A \rightarrow$ estado da chave CH1;
- $B \rightarrow$ estado da chave CH2;

Saída:

- L estado da lâmpada;

Passo 2: Construção da Tabela Verdade. A tabela 2.8 é elaborada.

Tabela 2.8: Tabela Verdade do problema 1 com solução.

Linha	A	B	L
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Passo 3: Obtenção de soma canônica (ou produto canônico). A soma canônica é obtida pela soma dos mintermos da tabela verdade. Um mintermo pode ser definido como o termo produto que resulta em exatamente "1" em uma dada linha da tabela verdade. A soma canônica é a expressão lógica que representa a tabela verdade Assim:

$$L = \sum_{A,B} (1,2) = (\bar{A}B) + (A\bar{B})$$

Passo 4: Simplificação da expressão. Neste caso, a soma canônica é a soma mínima, ou seja, não é possível simplificar mais a expressão resultante.

Passo 5: Elaboração da lógica em linguagem Ladder. As seguintes atribuições são feitas, de acordo com o painel de ligações do laboratório:

- $A \rightarrow$ chave CH1 (NA);
- $B \rightarrow$ chave CH2 (NA);
- $L \rightarrow$ lâmpada L;

O diagrama Ladder resultante é apresentado na Figura 2.32

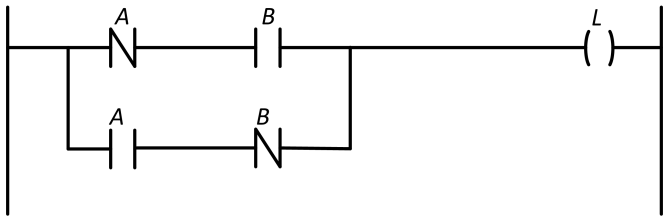


Figura 2.32: Diagrama Ladder da solução do problema 1.

A Figura 2.33 apresenta a tela da rotina programada no RSLogix 5000.

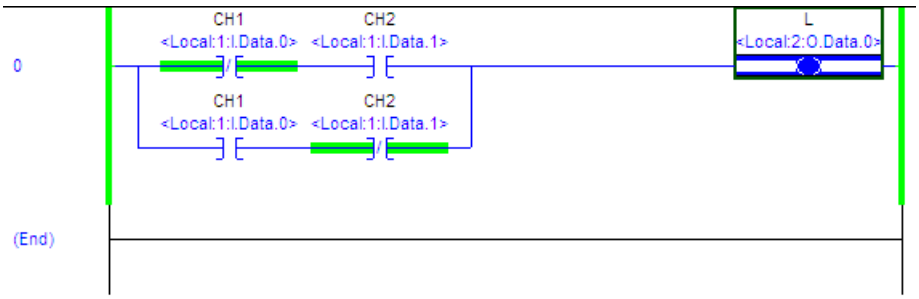


Figura 2.33: Diagrama Ladder da solução do problema 1 no RSLogix 5000.

Problema 2: A Figura 2.34 mostra o cruzamento de uma rodovia com uma via de acesso.

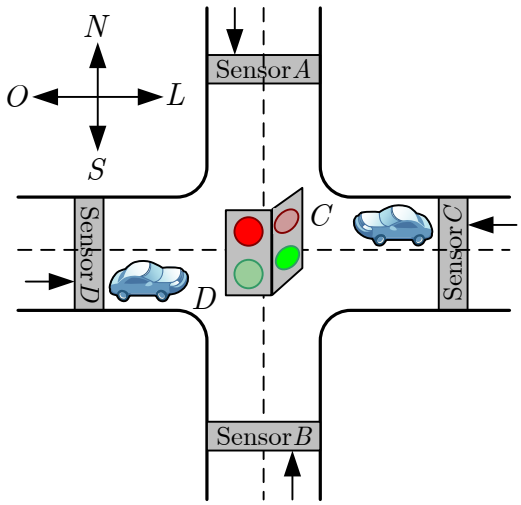


Figura 2.34: Ilustração do problema 2 com solução.

Sensores detectores de veículos são colocados ao longo das pistas C e D da rodovia e das pistas A e B da via de acesso. A saída desse tipo de sensor está em BAIXO quando não existe nenhum carro presente e está em ALTO quando um veículo está presente. Um sinal de trânsito colocado no cruzamento deve funcionar de acordo com a seguinte lógica:

- O sinal da direção leste-oeste LO deve estar verde quando as pistas C **E** D estiverem ocupadas.
- O sinal da direção LO deve estar verde quando C **OU** D estiverem ocupadas, e **NÃO** houver carros nas pistas A **E** B.
- O sinal da direção norte-sul NS deve estar verde quando ambas as pistas A **E** B estiverem ocupadas, e **NÃO** houver carros nas pistas C **E** D.
- O sinal da direção NS deve estar verde quando A **OU** B estiverem ocupadas e enquanto ambas as pistas C **E** D estiverem vazias.
- O sinal da direção LO deve estar verde quando **NÃO** houver nenhum veículo presente.

Utilizando as saídas dos sensores A, B, C e D como entradas, desenvolva uma lógica em Ladder que controle esse sinal de trânsito e implemente no CLP do laboratório. Devem existir duas saídas, NS e LO, que devem ir para ALTO quando o sinal correspondente estiver verde.

Solução: Inicialmente, as seguintes ligações apresentadas na Figura 2.35 são efetuadas.

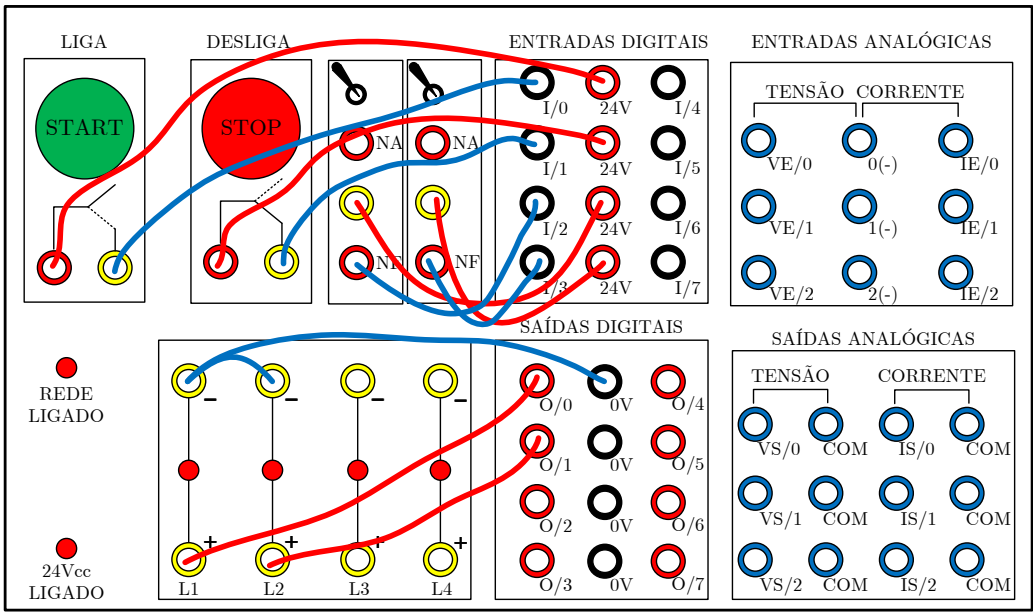


Figura 2.35: Ligações do exemplo.

Note que o problema é de natureza combinatória. Assim, o seguinte procedimento pode ser utilizado:

Passo 1: Identifique as entradas e saídas do sistema.

As seguintes entradas são identificadas:

- $A \rightarrow$ sensor da pista A;
- $B \rightarrow$ sensor da pista B;
- $C \rightarrow$ sensor da pista C;
- $D \rightarrow$ sensor da pista D;

Saídas:

- $NS \rightarrow$ Sentido Norte$\leftrightarrow$Sul habilitado;
- $LO \rightarrow$ Sentido Leste$\leftrightarrow$Oeste habilitado;

Passo 2: Construção da tabela verdade. De acordo com o enunciado, a Tabela 2.9 é construída.

Note que as saídas LO e NS são complementares, ou seja, basta resolver o problema para uma delas. Nesse caso, será escolhida a saída NS.

Tabela 2.9: Tabela Verdade do problema 2 com solução.

Linha	A	B	C	D	LO	NS
0	0	0	0	0	1	0
1	0	0	0	1	1	0
2	0	0	1	0	1	0
3	0	0	1	1	1	0
4	0	1	0	0	0	1
5	0	1	0	1	1	0
6	0	1	1	0	1	0
7	0	1	1	1	1	0
8	1	0	0	0	0	1
9	1	0	0	1	1	0
10	1	0	1	0	1	0
11	1	0	1	1	1	0
12	1	1	0	0	0	1
13	1	1	0	1	0	1
14	1	1	1	0	0	1
15	1	1	1	1	1	0

Passo 3: Obtenção de soma canônica (ou produto canônico). A soma canônica é obtida pela soma dos mintermos da tabela verdade. Um mintermo pode ser definido como o termo produto que resulta em exatamente "1" em uma dada linha da tabela verdade. A soma canônica é a expressão lógica que representa a tabela verdade. Assim:

$$NS = \sum_{A,B,C,D} (4, 8, 12, 13, 14) = (\bar{A}B\bar{C}\bar{D}) + (A\bar{B}\bar{C}\bar{D}) + (AB\bar{C}\bar{D}) + (AB\bar{C}D) + (ABC\bar{D})$$

Passo 4: Simplificação da expressão. Pode-se utilizar os axiomas da álgebra de Boole ou um método gráfico, como o mapa da Karnaugh. A segunda opção é escolhida[1]. O mapa de Karnaught da Figura 2.36 é gerado.

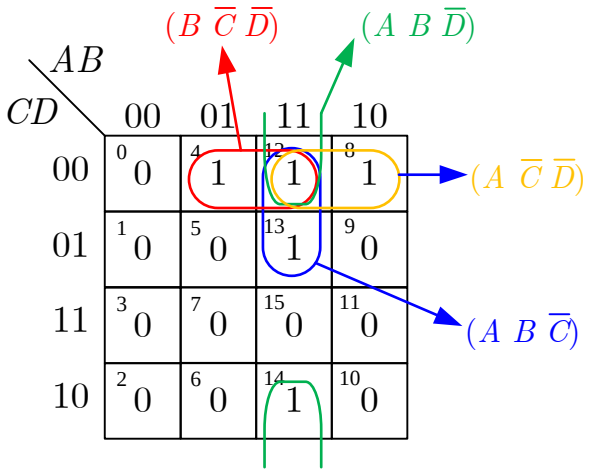


Figura 2.36: Mapa de Karnaught da solução do problema 2.

Logo, a expressão simplificada é dada por:

$$NS = (B\bar{C}\bar{D}) + (AB\bar{D}) + (A\bar{C}\bar{D}) + (AB\bar{C})$$

Passo 5: Elaboração da lógica em linguagem Ladder. As seguintes atribuições são feitas, de acordo com o painel de ligações do laboratório:

- $A \rightarrow$ botão Start (NA);
- $B \rightarrow$ botão Stop (NF - trocar B por $\bar{B}$ e vice-versa);
- $C \rightarrow$ chave CH1;
- $D \rightarrow$ chave CH2;
- $NS \rightarrow$ lâmpada L1;

[1] Esta opção pode ficar complicada para elevado número de variáveis.

- $LO \rightarrow$ lâmpada L2;

O diagrama Ladder resultante é apresentado na Figura 2.37.

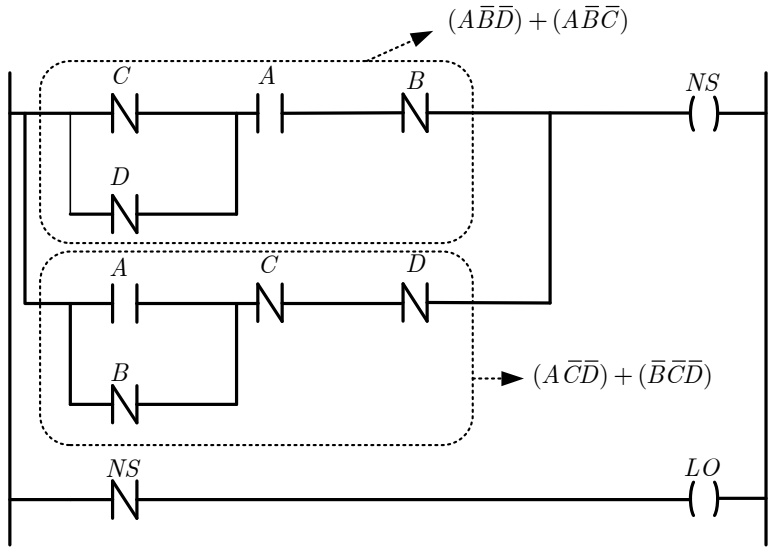


Figura 2.37: Diagrama Ladder da solução do problema 2.

A Figura 2.38 apresenta a tela da rotina programada no RSLogix 5000.

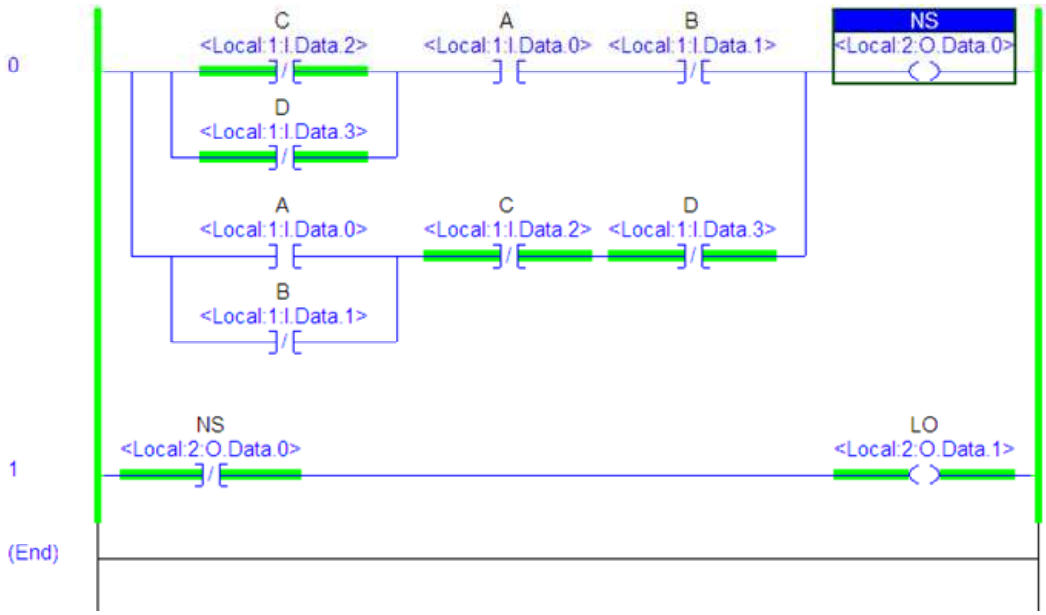


Figura 2.38: Diagrama Ladder da solução do problema 2 no RSLogix 5000.

Problema 3: Desenvolva uma aplicação em linguagem Ladder com as seguintes especificações:

- Quando BT1 (NA) é pressionado, as lâmpadas L1 e L2 ligam de forma alternada e sequenciada a cada 2 segundos.
- Quando BT2 (NF) é pressionado, as lâmpadas apagam e o ciclo é iniciado somente se BT1 for novamente pressionado.

`Solução`: Inicialmente, as seguintes ligações apresentadas na Figura 2.39 são efetuadas.

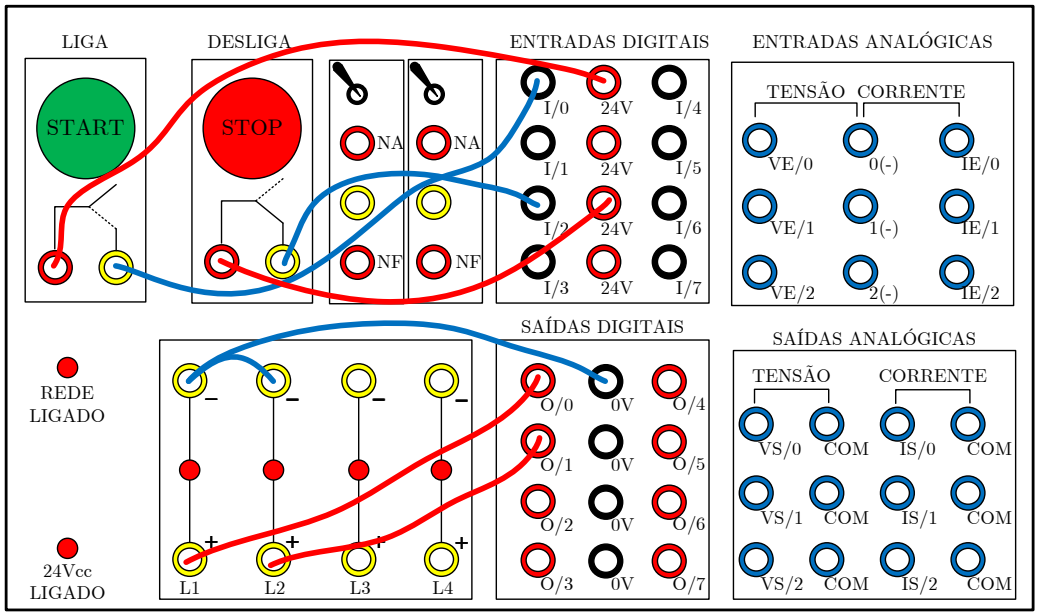


Figura 2.39: Ligações do exemplo.

O problema pode ser resolvido utilizando-se apenas um temporizador. No entanto, será considerado aqui o uso de dois temporizadores do tipo TON (Timer1 e Timer2), um para cada lâmpada, com período de contagem de 2 segundos em cada.

Nota-se que o problema é de natureza sequencial. Assim, o seguinte procedimento pode ser utilizado:

Passo 1: Identifique as entradas e saídas do sistema, os estados do sistema, e as transições de estado

As seguintes entradas e saídas são identificadas:

- Entradas:
  - $BT1 \rightarrow$ botão de habilitação (NA);
  - $BT2 \rightarrow$ botão de desabilitação (NF);
- Saídas:
  - $L1 \rightarrow$ lâmpada 1;
  - $L2 \rightarrow$ lâmpada 2;

Os seguintes estados são identificados.

- S0: L1 e L2 desligadas;
- S1: L1 ligada e L2 desligada;
- S2: L2 ligada e L1 desligada;

Transições:

- T1: S0 $\rightarrow$ S1 ($\mathrm{BT1} = 1$);
- T2: S1 $\rightarrow$ S2 ($\mathrm{Timer1.DN} = 1$);
- T3: S2 $\rightarrow$ S1 ($\mathrm{Timer2.DN} = 1$);

Passo 2: Crie um mapa de transição de estados, onde as transições ocorrem devido à mudança de nível lógico de variáveis booleanas, e também provocam mudanças de estado em variáveis booleanas. O mapa criado é apresentado na Figura 2.40.

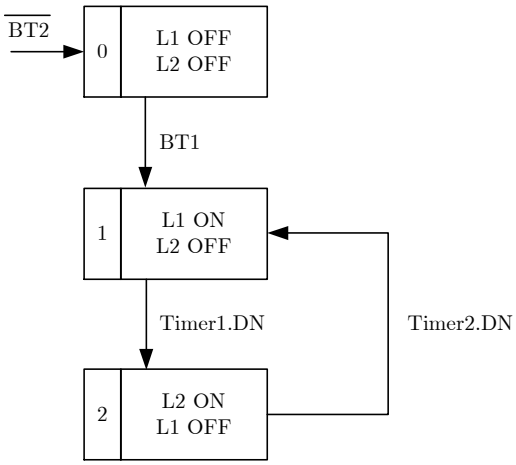


Figura 2.40: Mapa de transição de estados da solução do problema 3.

Note que se o botão B2 (NF) for pressionado, o sistema retorna ao estado inicial sem depender de qualquer outra condição.

Passo 3: Elaboração da lógica em linguagem Ladder a partir do mapa de transições. As seguintes atribuições são feitas, de acordo com o painel de ligações do laboratório:

- $BT1 \rightarrow$ botão Start (NA);
- $BT2 \rightarrow$ botão Stop (NF);
- $L1 \rightarrow$ lâmpada L1;
- $L2 \rightarrow$ lâmpada L2;

A passagem direta de mapa de transições de estado para Ladder ocorre quase que de forma direta, utilizando-se bobinas do tipo *latched* e *unlatched* para os estados, e efetuando o processo em duas etapas: **transições de estado** e **ações nos estados**.

A lógica desenvolvida está apresentada na Figura 2.41.

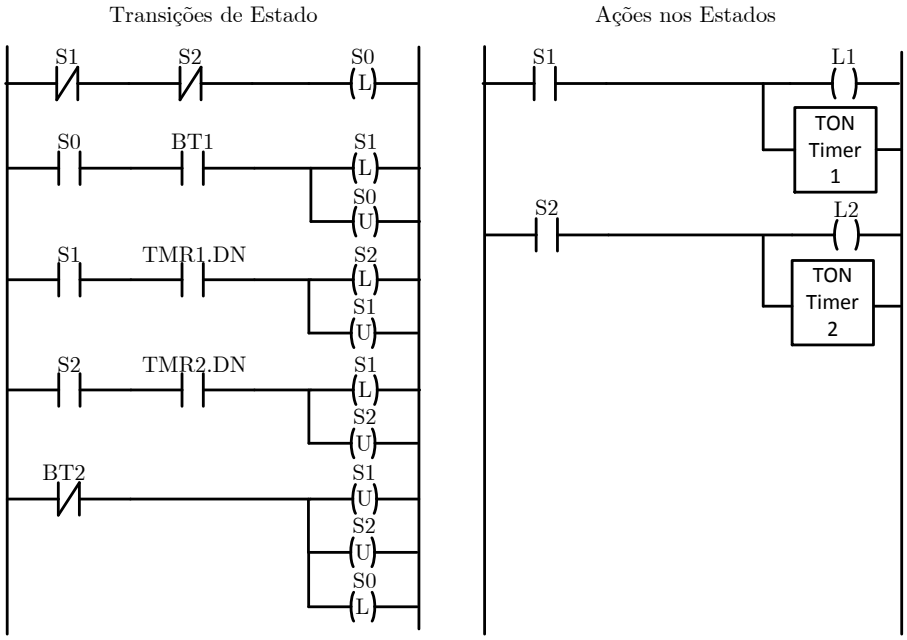


Figura 2.41: Lógica em Ladder da solução do problema 3.

A Figura 2.42 apresenta a tela da rotina programada no RSLogix 5000.

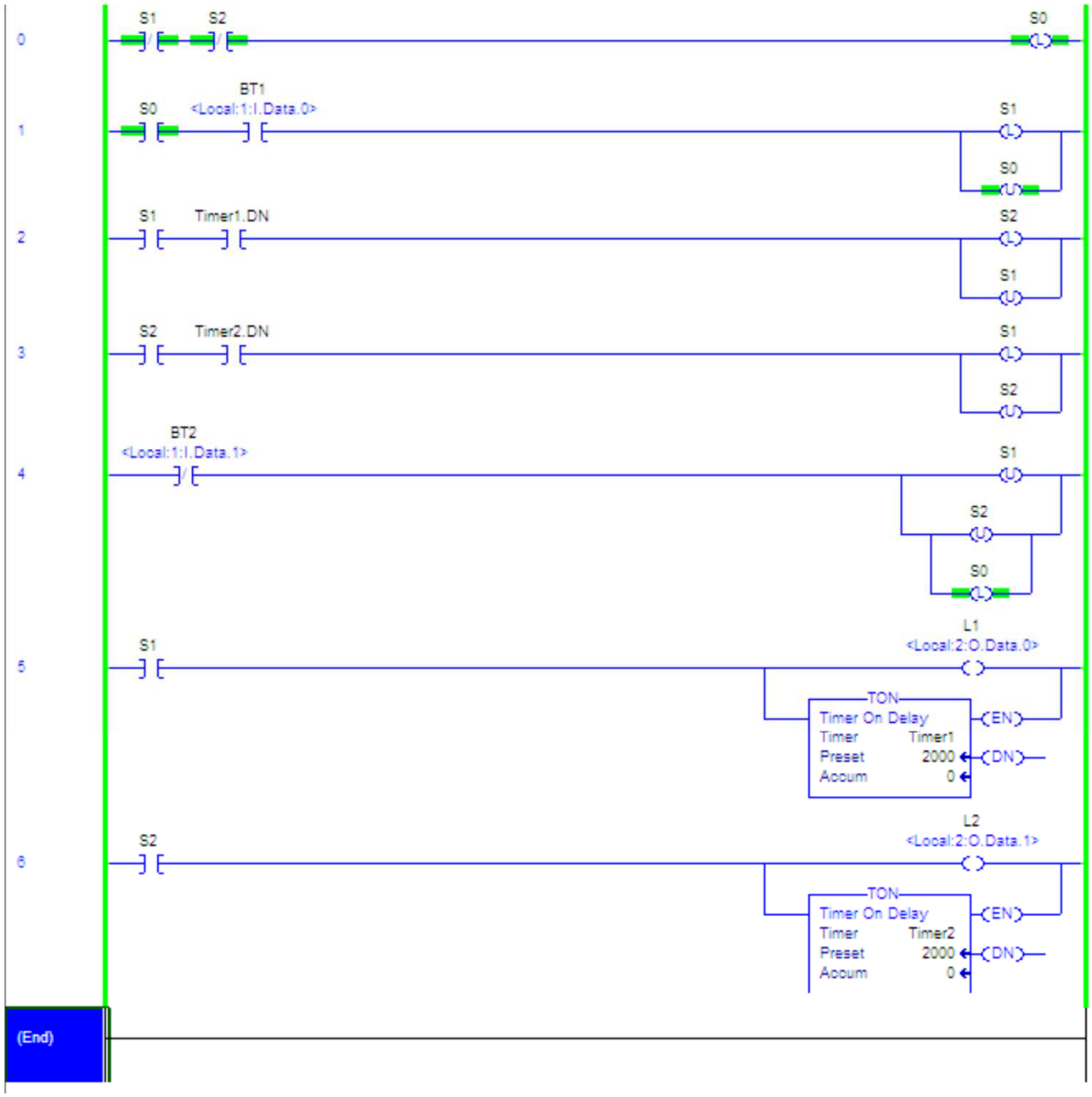


Figura 2.42: Lógica em Ladder da solução do problema 3.

Outra forma de solucionar o problema utilizando contatos de saídas convencionais é apresentada na Figura 2.43, utilizando o conceito de selo. Tal solução não é imediata, principalmente para problemas com maiores dimensões.

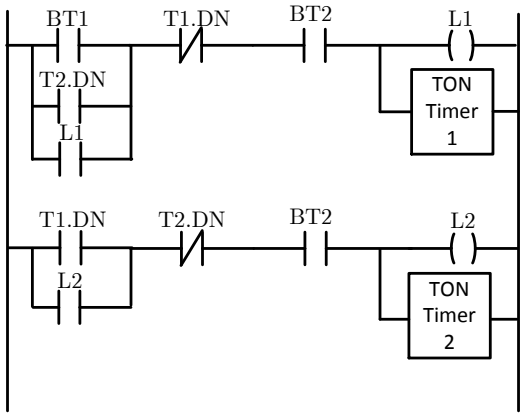


Figura 2.43: Lógica em Ladder da solução do problema 3 com contatos de saída convencionais.

A Figura 2.44 apresenta a tela da rotina programada no RSLogix 5000.

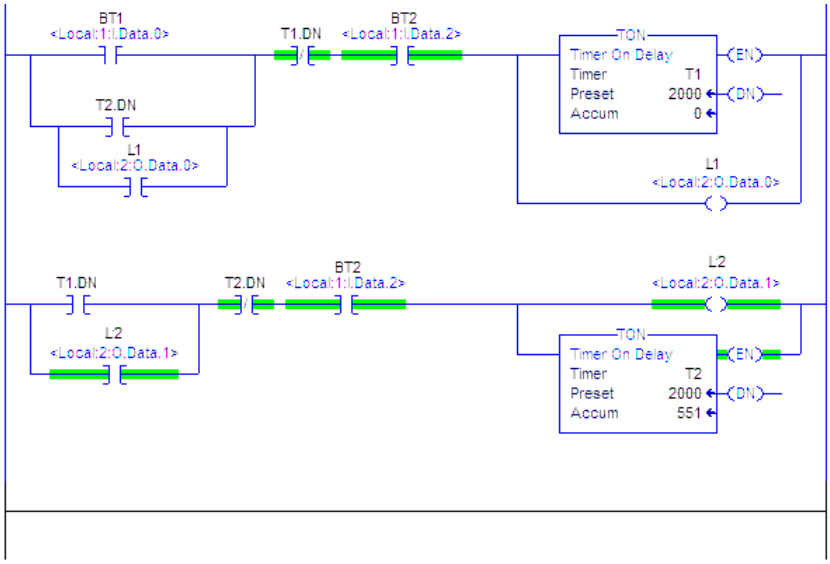


Figura 2.44: Lógica em Ladder da solução do problema 3 com contatos de saídas convencionais no RSLogix 5000.

Tente resolver o problema utilizando apenas um temporizador!

SFC

Problema 4: Refaça o **Problema 3**, mas agora em SFC:

- Quando BT1 (NA) é pressionado, as lâmpadas L1 e L2 ligam de forma alternada e sequenciada a cada 2 segundos.
- Quando BT2 (NF) é pressionado, as lâmpadas apagam e o ciclo é iniciado somente se BT1 for novamente pressionado.

Solução: O código final é apresentado na Figura 2.45.

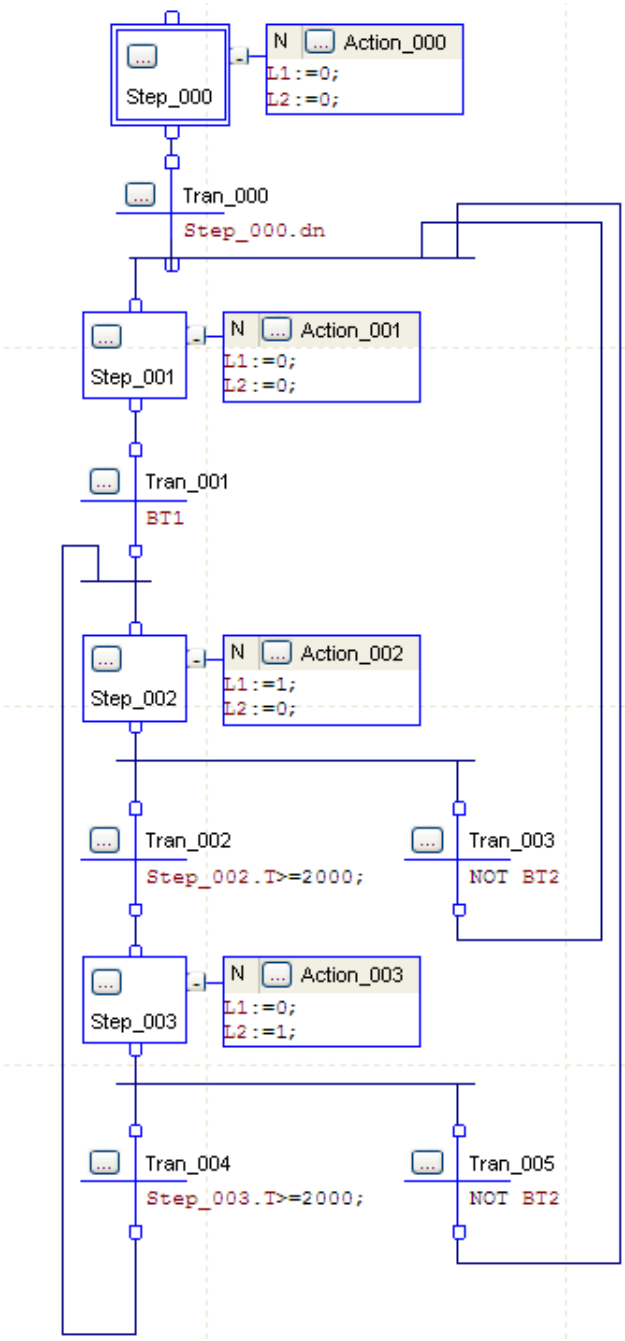


Figura 2.45: Solução do Problema 4.

Outra forma de solução é apresentada na Figura 2.46.

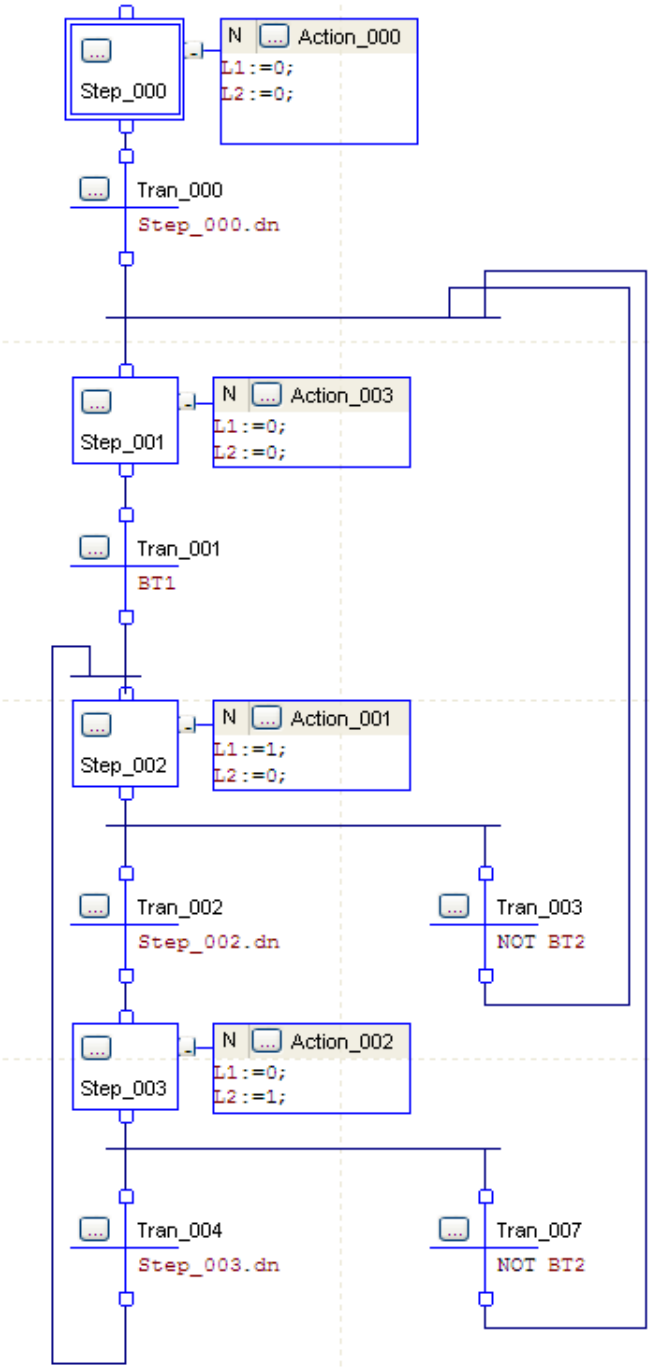


Figura 2.46: Solução alternativa do Problema 4.

Nesta solução alternativa, para definir o tempo de 2 segundos em cada estado, utilizou-se o valor de *preset* do temporizador do passo. Ele é configurado clicando-se no símbolo "..." do passo, conforme a Figura 2.47. Note que a transição é efetuada quando a condição `Step_002.dn` é satisfeita.

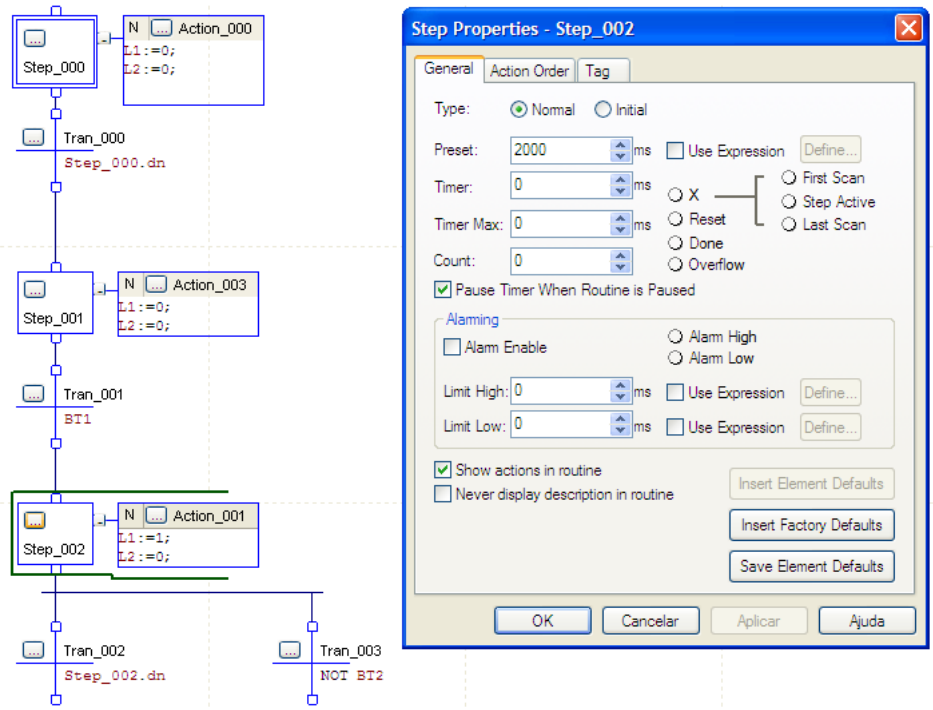


Figura 2.47: Preset do timer para a solução alternativa do Problema 4.

Problema 5: Ao pressionar um botão NA (BT1), uma lâmpada (L1) deve piscar 5 vezes com um período de 1 segundo. Após isso, o ciclo pode ser iniciado ao clicar o botão novamente.

Solução: O código final é apresentado na Figura 2.48.

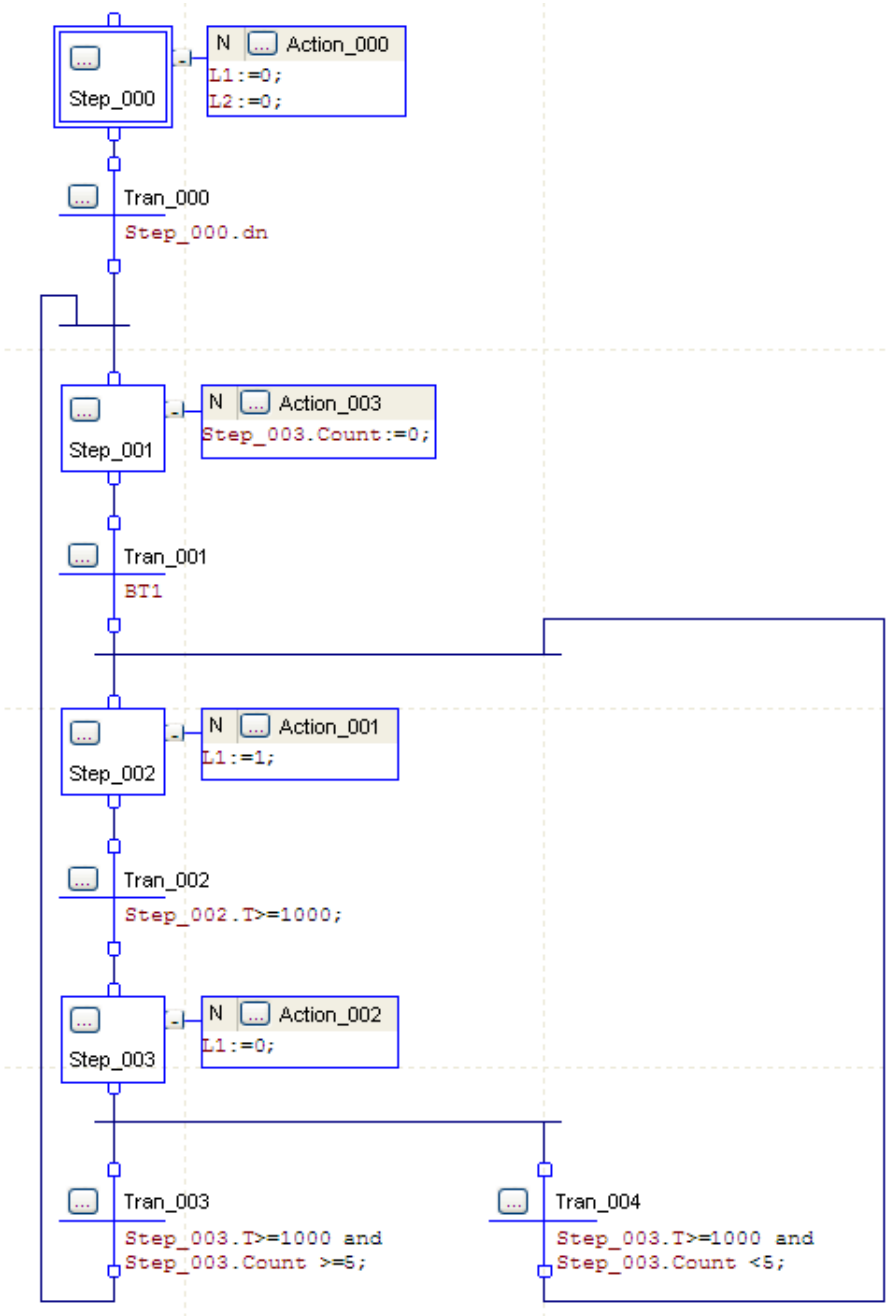


Figura 2.48: Solução do Problema 5.

Uma solução alternativa do Problema 5 é apresentada na Figura 2.49. Note que ao `Step_002` foram atribuídas 2 ações com qualificadores diferentes. Na segunda ação, utilizou-se o qualificador P1 para incrementar um contador. Nesta solução, utilizou-se o *preset* dos *timers* para temporização, como na solução alternativa do Problema 4.

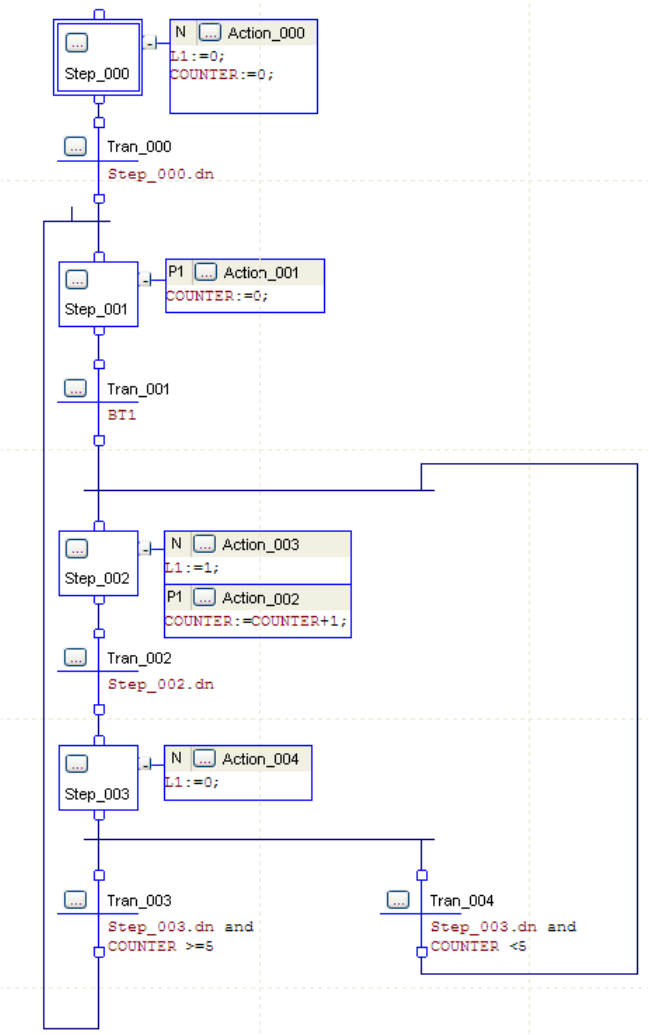


Figura 2.49: Solução alternativa do Problema 5.

2.6 Dispositivos Pneumáticos

O laboratório dispõe de diversos dispositivos pneumáticos e outros que podem ser utilizados em conjunto com os CLPs. O objetivo desta aula é implementar algumas aplicações utilizando atuadores pneumáticos, sensores de fim de curso, válvulas solenóides e o CLP.

A seguir serão descritos alguns sensores componentes básicos de pneumáticas presentes no laboratório.

- **Unidade de Conservação**: tem a finalidade de purificar o ar comprimido, ajustar uma pressão constante do ar e, em alguns casos, acrescentar uma fina neblina de óleo ao ar comprimido, para fins de lubrificação. Aumenta consideravelmente a segurança de funcionamento dos equipamentos pneumáticos. A Figura 2.50 ilustra uma unidade de conservação sem lubrificador, juntamente com seu diagrama.

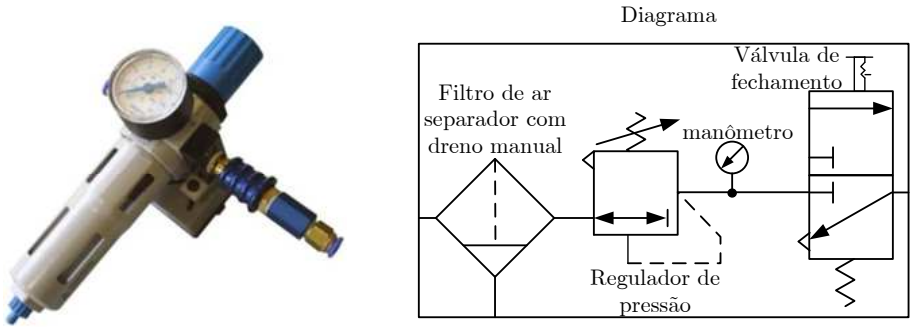


Figura 2.50: Unidade de conservação.

A unidade de conservação disponível no laboratório possui as seguintes características:

- conjunto de filtro, regulador de pressão, manômetro e válvula de fechamento;
- dreno manual;
- pressão de operação: de 0 a 12 bar;
- vazão nominal: 750 lpm;
- escala métrica: de 0 a 16 bar; escala inglesa: de 0 a 220 psi;
- válvula deslizante de acionamento manual biestável.

- **Bloco Distribuidor**: funciona como um demultiplexador de ar comprimido. A entrada de ar comprimido é redistribuída em várias saídas. A Figura 2.57 ilustra um bloco distribuidor e seu diagrama.

O bloco distribuidor disponível no laboratório possui uma entrada e oito saídas de ar comprimido.

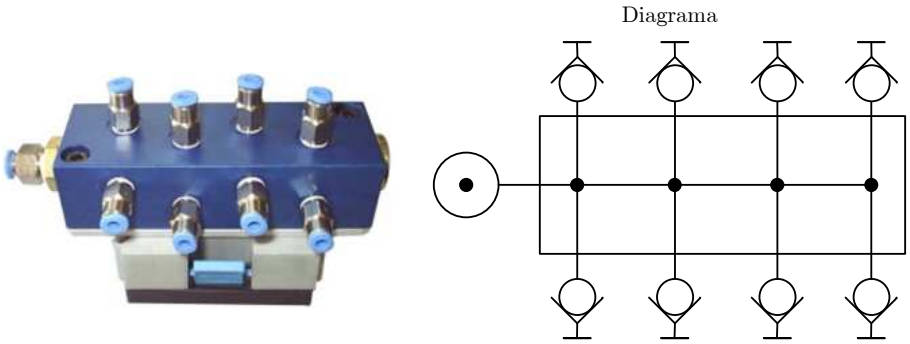


Figura 2.51: Bloco distribuidor.

- **Cilindro de Simples Ação**: A Figura 2.52 apresenta um exemplo de cilindro de simples ação com avanço pneumático e retorno por mola.

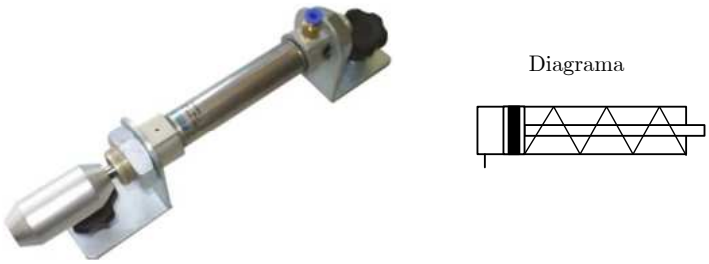


Figura 2.52: Cilindro de simples ação.

A pressão máxima de trabalho do modelo presente no laboratório é de 10 bar.

- **Eletroválvula direcional de 3/2 vias NF**: Válvula acionada por servocomando elétrico e piloto. É importante frisar que o acionamento por servocomando é indireto, ou seja, não é o solenóide quem aciona diretamente o carretel da válvula; ele epenas abre uma passagem para o ar comprimido (piloto) que aciona o carretel e muda a posição da válvula. Possui 3 vias de trabalho e 2 posições. O contato é normalmente fechado. A Figura 2.53 apresenta um exemplo desta válvula e seu diagrama.

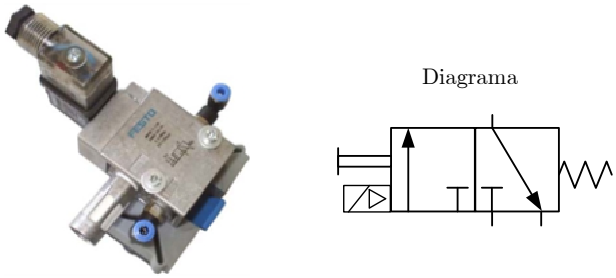


Figura 2.53: Eletroválvula direcional de 3/2 vias NF.

Algumas características adicionais:

- acionamento por servocomando, simples piloto e solenóide de 24 Vcc.

- retorno por mola;
- possibilidade de acionamento manual de emergência;
- pressão de operação: de 1,5 a 8 bar;
- vazão nominal: 500 lpm.

- **Cilindro de Dupla Ação**: A Figura 2.54 apresenta um exemplo de cilindro de dupla ação com avanço e retorno pneumáticos.

Figura 2.54: Cilindro de dupla ação.

A pressão máxima de trabalho do modelo presente no laboratório é de 10 bar. Além disso, o modelo possui êmbolo magnético para detecção por sensores sem contato físico,

- **Eletroválvula Direcional de 5/2 vias com mola de reposição**: Válvula acionada por servocomando elétrico e piloto. Possui 5 vias de trabalho e 2 posições. A Figura 2.55 apresenta um exemplo desta válvula e seu diagrama.

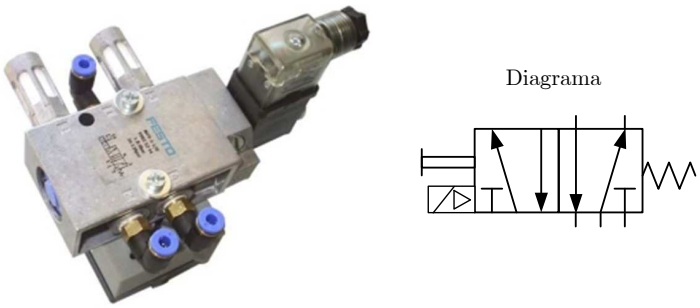


Figura 2.55: Eletroválvula direcional de 5/2 vias com retorno por mola.

As características adicionais são as mesmas da eletroválvula anterior.

- **Eletroválvula Direcional de 5/2 do tipo memória**: Válvula acionada por servocomando elétrico e piloto. Possui 5 vias de trabalho e 2 posições. Também chamada de válvula de impulso. Para sua comutação, basta emitir um pulso de pilotagem, não sendo necessário mantê-lo após a mudança de posição. A Figura 2.56 apresenta um exemplo desta válvula e seu diagrama.

Figura 2.56: Eletroválvula direcional de 5/2 vias com duplo acionamento e memória.

- **Chave de Fim de Curso**: chave de contato mecânico utilizada como sensor de fim de curso por contato físico. A Figura 2.57 ilustra uma chave de fim de curso e seu diagrama.

Figura 2.57: Chave fim de curso.

Algumas características da chave de fim de curso disponível no laboratório:

- reposicionado por mola;
- corrente: 5 A.

- **Sensor Capacitivo**: Os sensores de proximidade capacitivos registram a presença de qualquer tipo de material. A distância de detecção varia de 0 a 20 mm, dependendo da massa do material a ser detectado e das características determinadas pelo fabricante. A Figura 2.58 apresenta um sensor capacitivo e seu diagrama.

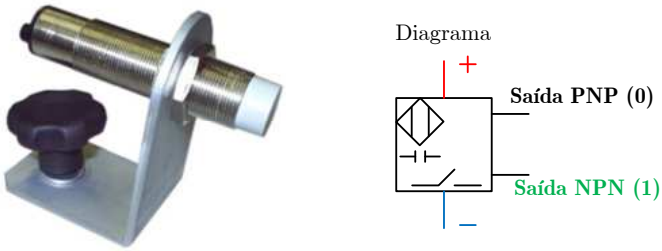


Figura 2.58: Sensor de proximidade capacitivo.

O sensor capacitivo presente no laboratório possui as seguintes características:

- distância de sensoriamento: 20 mm;

- tensão de alimentação: 10 a 30 Vcc;
- frequência máxima: 100 Hz;
- sinal de saída: 24 Vcc PNP;
- positivo: **vermelho**, negativo: **azul**, saída PNP: **preto**.

• **Sensor Magnético**: Os sensores de proximidade magnéticos detectam apenas a presença de materiais metálicos e magnéticos. São utilizados com maior frequência em máquinas e equipamentos pneumáticos e são montados diretamente sobre as camisas dos cilindros dotados de êmbolos magnéticos. A Figura 2.59 apresenta um sensor magnético e seu diagrama.

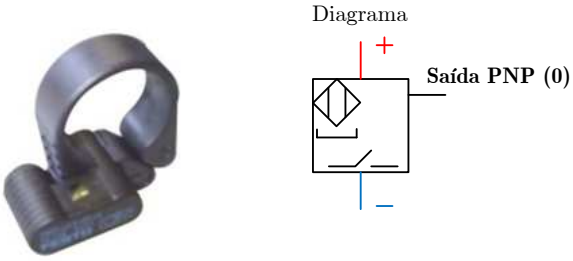


Figura 2.59: Sensor de proximidade magnético.

Algumas características do sensor disponível no laboratório:

- sensoriamento de êmbolos magnéticos de cilindros, sem contato físico;
- tensão de comutação: de 12 a 27 Vcc;
- freqüência máxima: 800 Hz;
- sinal de saída: 12 a 27 Vcc PNP;
- positivo: **vermelho**, negativo: **azul**, saída PNP: **preto**.

• **Sensor Indutivo**: Os sensores de proximidade indutivos são capazes de detectar apenas materiais metálicos. A Figura 2.60 apresenta um sensor indutivo e seu diagrama.

O sensor indutivo presente no laboratório possui as seguintes características:

- distância de sensoriamento: 5 mm;
- tensão de alimentação: 10 a 30 Vcc;
- frequência máxima: 800 Hz;
- sinal de saída: 24 Vcc PNP;
- positivo: **vermelho**, negativo: **azul**, saída PNP: **preto**.

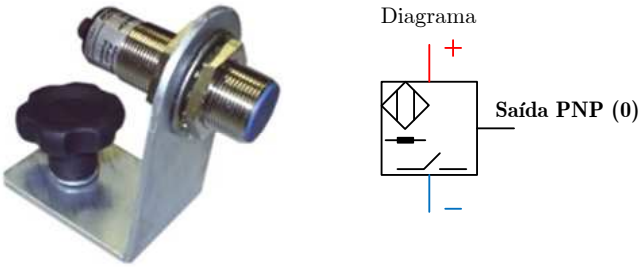


Figura 2.60: Sensor de proximidade indutivo.

2.6.1 Cuidados com a Segurança

- **Antes de pressurizar o sistema verifique se as mangueiras estão bem presas aos conectores. Mangueiras pressurizadas soltas podem chicotear e causar acidentes graves.**
- **Não mexa nas conexões das mangueiras com o sistema pressurizado.**
- **Mantenha suas mãos e objetos a uma distância segura do curso de acionamento dos pistões.**

2.7 Atividades

2.7.1 Exercícios simples

a) Um dispositivo de uma indústria metalúrgica tem como função a fixação de peças em um molde. Esta fixação é feita por um **atuador linear de dupla ação** (ou seja, que opera com dois sinais de atuação distintos), que avança mediante o acionamento de dois botões ($S1$ e $S2$) e retorna caso qualquer um dos botões seja desativado. Elabore um programa em linguagem Ladder que resolva este problema e teste no CLP.

b) Elaborar um programa Ladder para controlar duas saídas ($M1$ e $M2$) de tal maneira que $M1$ pode ser acionada de forma independente e $M2$ só pode ser acionada de $M1$ estiver acionada, mas pode continuar ligada após o desligamento de $M1$. As saídas são ligadas pelas botoeiras $L1$ e $L2$, e são desligadas pelas botoeiras $D1$ e $D2$.

c) Elaborar um programa Ladder de controle para o reservatório da Figura 2.61 composto de uma válvula de entrada $V1$, duas bombas (acionadas por $M1$ e $M2$), um alarme AL e quatro sensores de nível (A, B, C, D). A vazão do líquido de entrada é variável e dependente de processos anteriores. As condições de funcionamento são as seguintes: se o nível chegar em A, então fecha-se a válvula $V1$. Se o nível for inferior a B então abre-se a válvula $V1$. Acima de B, $M1$ e $M2$ bombeiam. Abaixo de B, somente $M1$ bombeia. Abaixo de C soa o alarme AL e a lampada $L1$. Abaixo de D, nenhuma das bombas deverá funcionar.

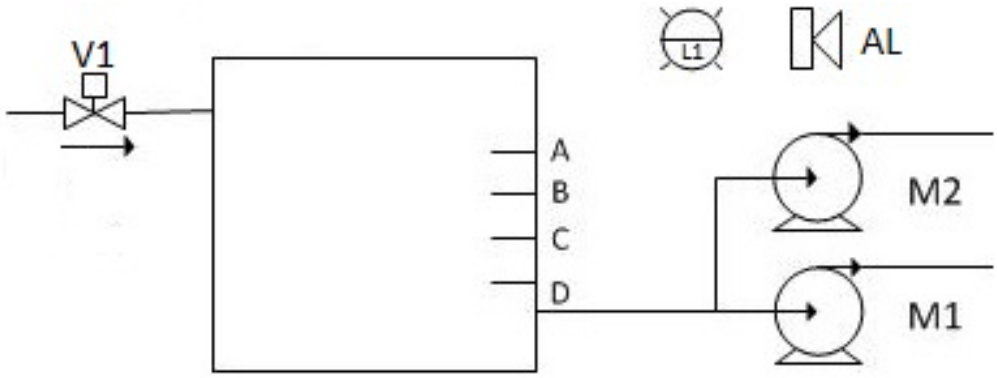


Figura 2.61: Diagrama do controle de nível no tanque.

2.7.2 Exercício usando contadores

a) Deseja-se contar o número de caixas que passa pelo sensor $S1$ em uma esteira. O processo é iniciado com o acionamento do botão $BT1$ e a esteira $M1$ é ligada. Após 4 caixas passarem pelo sensor $S1$, $M1$ é desligada e a lâmpada $L1$ deve ser acionada para que o fardo de caixas seja fechado. Neste ponto, o processo é interrompido para que o fardo seja transportado. O processo

é reiniciado quanto o botão $BT1$ for pressionado. Quando o total de caixas que passa pelo sensor for igual a 20, o processo para e a lâmpada $L2$ é acionada para que o lote seja fechado. Tudo reinicia quando $BT1$ for pressionado novamente. Se BT1 for pressionado acidentalmente no meio do processo (antes de fechar um fardo), nada acontece. O botão $BT2$ é de emergência: quando pressionado, tudo para e o contador é zerado. Para reiniciar o processo, deve-se pressionar $BT1$ novamente. Elabore um programa em linguagem Ladder que resolva este problema.

2.7.3 Exercícios usando temporizadores

a) Um sistema de dois semáforos controla o tráfego de um cruzamento de duas ruas (rua A e rua B), conforme a Figura 2.62, sendo que cada semáforo está posicionado numa das ruas. A sequência de acionamento de cada fase (amarelo, vermelho e verde) dos semáforos é mostrada na Tabela 2.10.

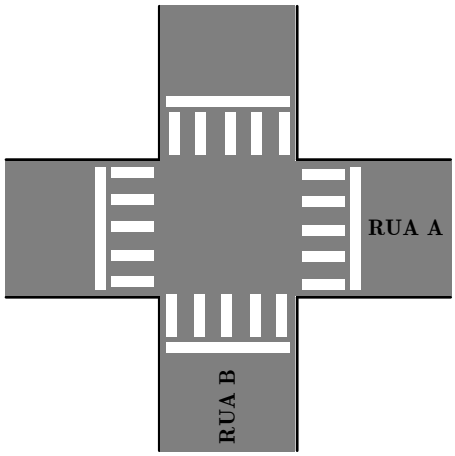


Figura 2.62: Cruzamento de ruas.

Tabela 2.10: Tabela de funcionamento do sistema de semáforos.

Fase	**Tempo** (s)	**Semáforo A**	**Semáforo B**
1	10	Verde	Vermelho
2	3	Amarelo	Vermelho
3	2	Vermelho	Vermelho
4	10	Vermelho	Verde
5	3	Vermelho	Amarelo
6	2	Vermelho	Vermelho

Implemente o semáforo em um programa em linguagem Ladder.

b) Resolva o item a) utilizando SFC.

2.7.4 Exercício usando contadores e temporizadores

a) Deseja-se envasar vacina em fracos de modo automático utilizando-se um CLP. Os frascos movimentam-se através de uma esteira rolante acionada por um motor elétrico M, o qual é ligado e desligado pelo CLP, conforme a Figura 2.63. Há um botão $BT1$ que inicia o processo. Quando cinco frascos passarem por um sensor de presença (Sensor A), o motor deve ser desligado e um conjunto de cinco bicos injetores de vacina, acionados pelo atuador B, deve permanecer ligado por 1 segundo (para encher os frascos); após esse tempo de 1 segundo, o motor da esteira deve voltar a movimentá-la, até que outros cinco frascos vazios passem pelo Sensor A; quando isso ocorrer, o processo se repetirá. Há um botão $BT2$ de emergência que para o processo, zera o contador e o temporizador.

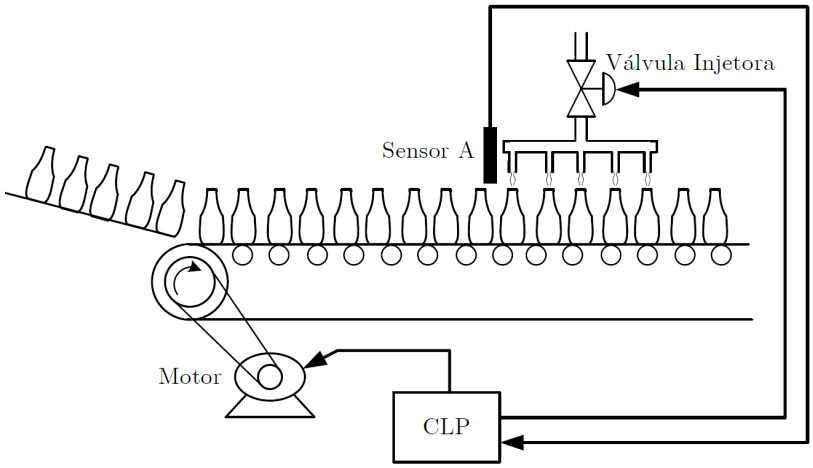


Figura 2.63: Sistema de envase de vacinas.

b) Resolva o item a) utilizando SFC.

2.7.5 Dispositivos Pneumáticos - Cancela em linha ferroviária

A cancela possui dois sensores de presença espaçados de 100m, conforme a Figura 2.64. O acionamento da cancela é feito através de um pistão pneumático. Considere que os trens podem vir de ambos os sentidos e que podem (ou não) ter mais de 100m de comprimento. Considere também as seguintes características funcionais do sistema.

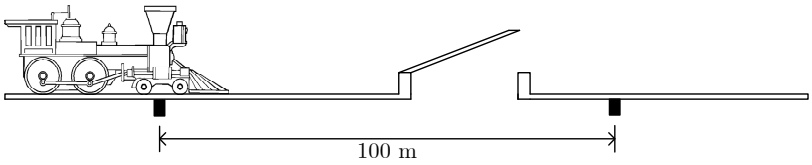


Figura 2.64: Cancela em linha ferroviária.

i) Caso haja problemas com o pistão e a cancela não seja acionada (um sensor de fim de curso verifica essa situação), um alarme luminoso deve ser acionado.

ii) Caso demore mais de 30s para um trem passar pelo segundo sensor após ter passado pelo primeiro, o mesmo alarme deve ser acionado.

Implemente um programa em Ladder para o sistema e teste no CLP em duas etapas.

a) Inicialmente assuma que os trens podem vir apenas de um sentido.

b) Altere o programa para considerar os dois sentidos de deslocamento dos trens.

2.7.6 Dispositivos Pneumáticos - Porta de um vagão do metrô

A porta de um vagão do metrô é mostrada na Figura 2.65. Dois pistões pneumáticos (Pistões A e B) são empregados para abrir e fechar a porta automaticamente. A presença de pessoas no curso das portas é verificada por sensores instalados em cada lado da porta (Sensores A e B).

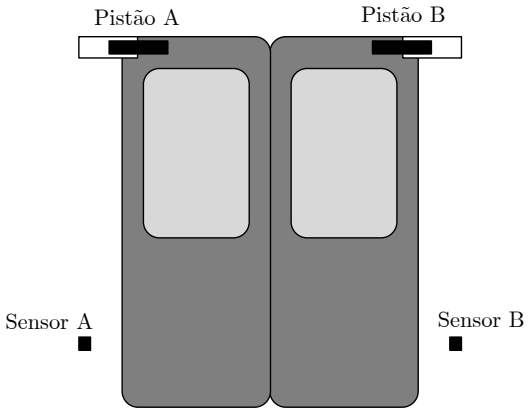


Figura 2.65: Porta de vagão do metrô.

Os requisitos funcionais do sistema de controle das portas são esquematizados a seguir.

i) A porta é aberta automaticamente quando o trem chega à estação, ativando um sensor de contato sobre a linha férrea.

ii) A porta permanece aberta por 30s, independente de haver ou não pessoas entrando e saindo pela porta. Considera-se que haja a presença de pessoas na porta se qualquer um dos Sensores A ou B for ativado.

iii) Após os 30s iniciais, a porta fecha caso não haja pessoas, ou permanece aberta por mais 10s. Se nesse período ninguém passar pela porta, ela fecha, e caso contrário permanece aberta por mais 10s e assim por diante.

iv) Caso a porta tenha permanecido aberta por três ciclos de 10s, totalizando 60s aberta, um alarme luminoso deve ser acionado para que o pessoal de segurança do metrô verifique a situação, por exemplo desobstruindo a porta ou limitando a entrada de passageiros. Note que mesmo com o acionamento do alarme, o algoritmo do item anterior deve continuar sua execução.

Utilize o CLP do laboratório para implementar o sistema em duas etapas.

a) Conforme acima.

b) Incluindo sensores nos pistões de cada porta para verificar se quando a porta é acionada ela efetivamente fecha. Em caso de falha o comando de abrir deve ser enviado à porta e um alarme luminoso deve ser ativado. Inclua um botão de ”Acknowledge”de falha das portas.

2.7.7 Dispositivos Pneumáticos - Sistema de eclusas do Canal do Panamá

O canal do Panamá (Figura 2.66) possui um sistema de eclusas para possibilitar a navegação sobre o istmo do Panamá, entre os Oceanos Atlântico e Pacífico.

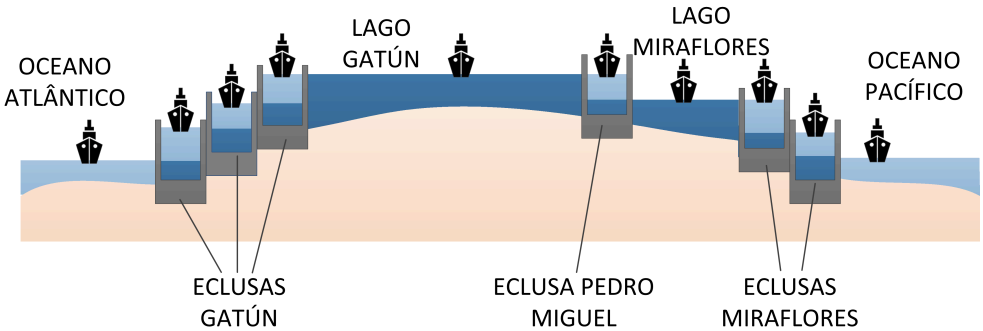


Figura 2.66: Sistema de eclusas do Canal do Panamá.

A operação simplificada de um sistema de automação da operação de uma eclusa é apresentada a seguir.

i) Quando um barco chega à entrada de uma eclusa, esta deve ser aberta caso não esteja ocupada ou em processo de enchimento ou esvaziamento. Considere que haja sensores que detectam a chegada de um barco assim como a presença de um barco na eclusa.

ii) Quando o barco entra a eclusa deve ser fechada.

iii) A eclusa deve encher até atingir o nível de saída. Considere que um sensor de nível avise quando este limite é atingido.

iv) A saída da eclusa deve ser aberta para o barco sair.

v) Após a saída do barco a saída deve ser fechada e a eclusa esvaziada até o nível de entrada, quando o ciclo recomeça.

Utilize esta descrição para obter os requisitos funcionais de um programa em Ladder que resolva o problema.

Considere que as entradas e saídas das eclusas são acionadas através de pistões pneumáticos e que há sensores que detectam a presença de embarcações na entrada, na saída e dentro da eclusa.

Elabore um programa em Ladder para implementação deste sistema em duas etapas.

a) Conforme acima.

b) Considerando que barcos podem chegar de ambos os sentidos. Neste caso, um sistema de intertravamento de chegada de barcos deve ser adicionalmente implementado.

Para entender melhor o funcionamento da eclusa, vide GIF animado em `http://www.dh.sp.gov.br/eclusagem/`.

2.8 Relatório

Um relatório desta experiência deverá ser entregue.

Apêndice I — Simulador CODESYS

CODESYS é uma plataforma de desenvolvimento de aplicações de CLP, utilizada por várias empresas, que possui uma opção para simulação. A versão escolhida para uso na disciplina é a da empresa EATON. Tal versão possui uma licença DEMO que pode ser baixada e utilizada gratuitamente. Para baixar o software, acesse o site `http://applications.eaton.eu/sdlc/?lx=11`, e escolha a opção como indicado na Figura 2.67. Ao baixar, escolha a opção mais recente.

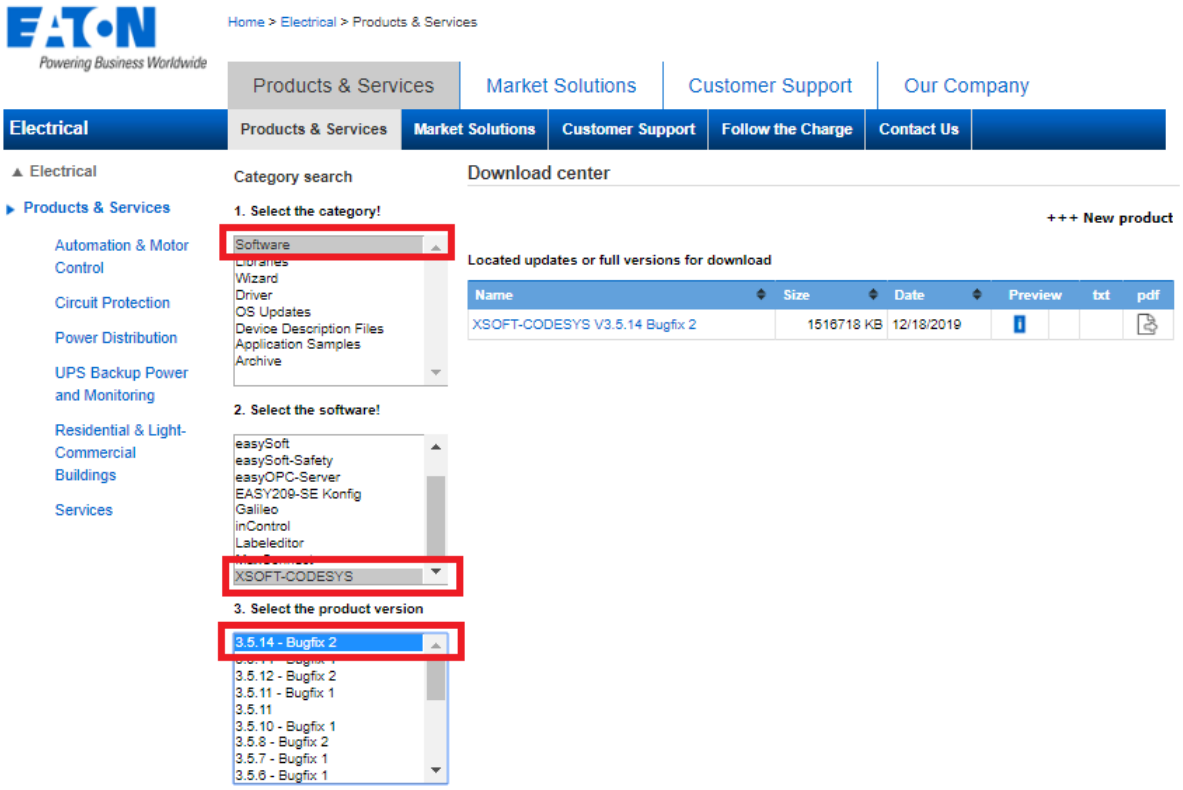


Figura 2.67: Site CODESYS EATON.

Após a instalação COMPLETA, execute o programa XSOFT-CODESYS V3.5.14. Ao executar, a tela inicial possui a forma apresentada na Figura 2.68. Para criar um novo projeto, escolha a opção `New Project...`. Então, escolha a opção `Standard Project`. Em seguida, a tela da Figura 2.69 deve ser exibida.

Note que todas a linguagens de programação padronizadas pela norma IEC 61131-3 estão disponíveis. Vamos começar pela linguagem Ladder.

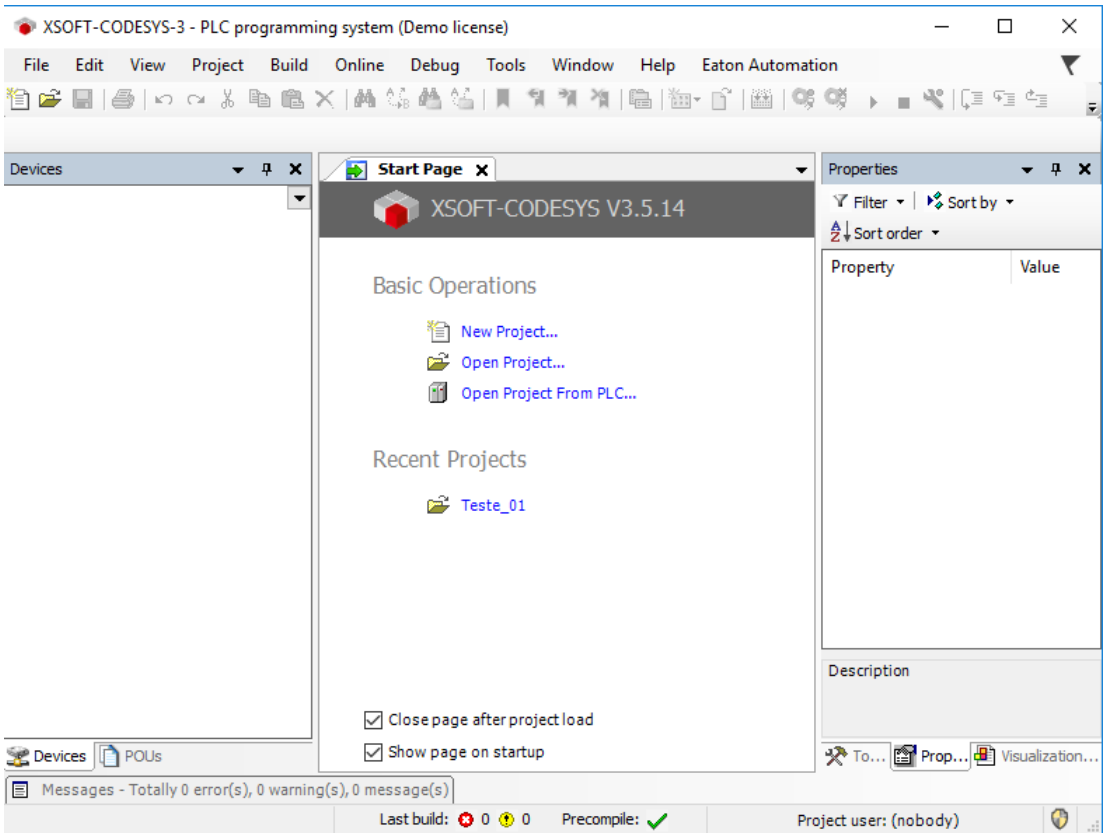


Figura 2.68: Tela inicial CODESYS.

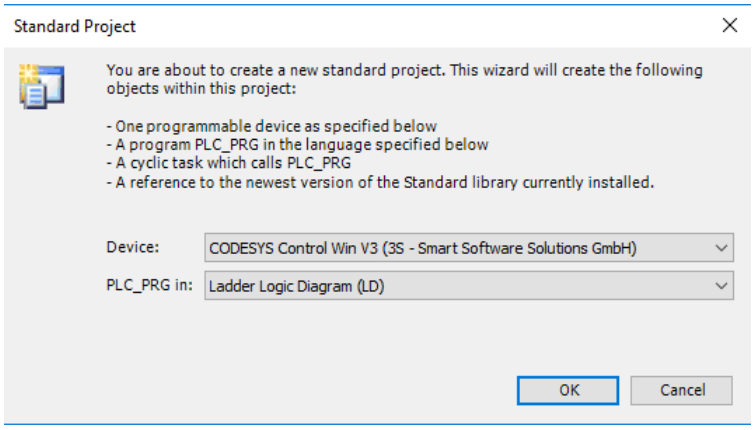


Figura 2.69: Linguagens CODESYS.

Ladder CODESYS

Ao selecionar `Main Task -> PLC_PRG`, a primeira linha de programação será exibida e o painel com os contatos na barra de tarefas será habilitado, conforme ilustrado na Figura 2.70[2].

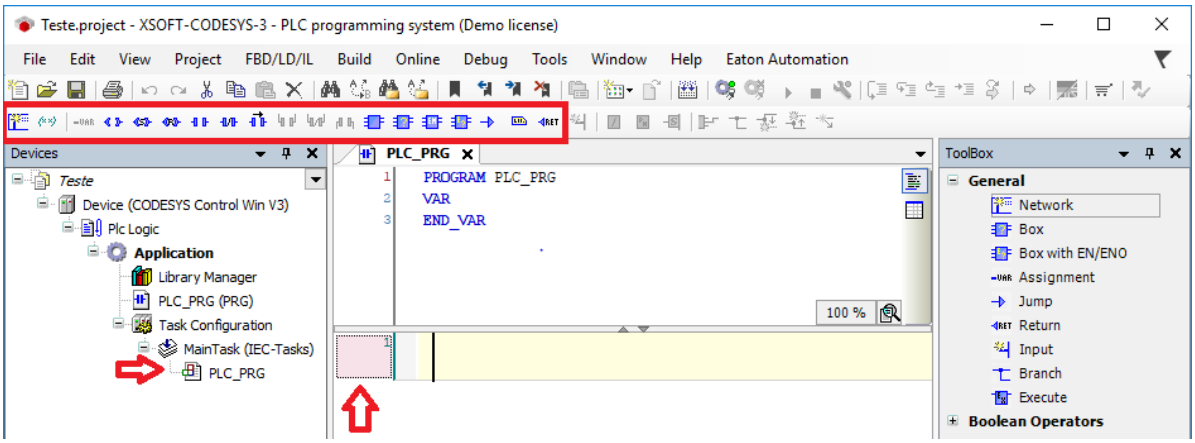


Figura 2.70: Ladder CODESYS - primeiro passo.

Para inserir uma nova linha acima ou abaixo da linha existente, clique com o botão direito sobre a linha e escolha a opção `Insert Network`, ou `Insert Network (below)`, respectivamente.

Vamos criar uma aplicação simples: ao clicar em BT1 (NA), `L1` liga e permanece ligado enquanto o botão estiver pressionado. Inicie inserindo o contato de entrada NA, clicando sobre seu símbolo na barra de ferramentas, e atribuindo a ele uma *tag* (Figura 2.71). Para definir o tipo de variável, digite `BT1` e `Enter`. Uma nova janela abrirá.

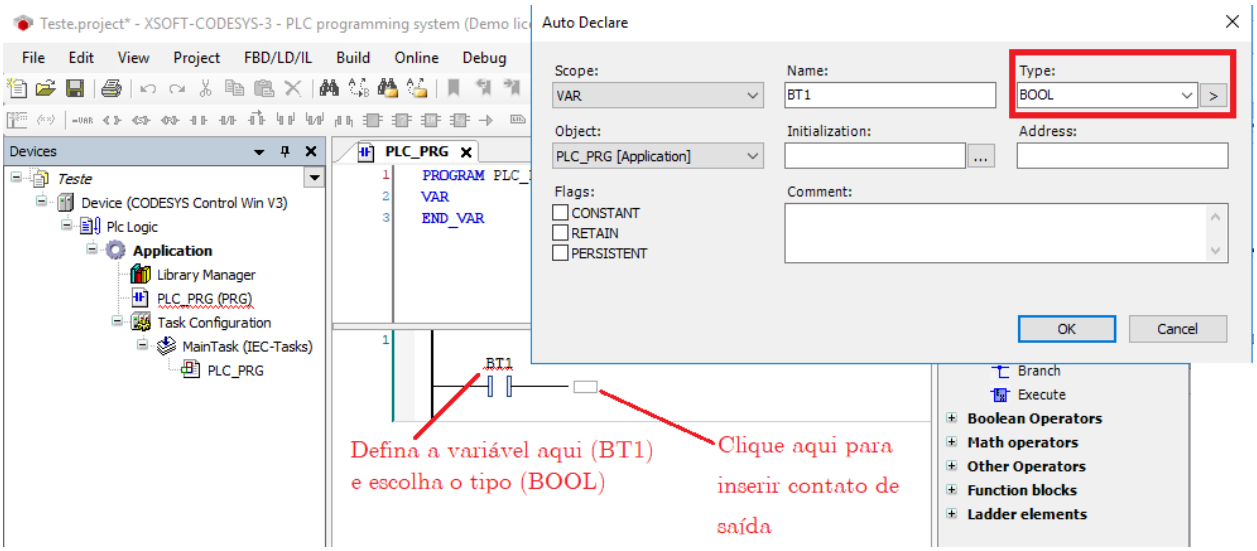


Figura 2.71: Ladder CODESYS - contato de entrada.

[2]OBS: caso o painel `Devices` seja fechado acidentalmente, ele pode ser novamente ativado pelo menu `View -> Devices` (ou Alt+0).

Após inserir os contatos de entrada e saída e fazer as devidas definições, chega-se ao esquema apresentado na Figura 2.72.

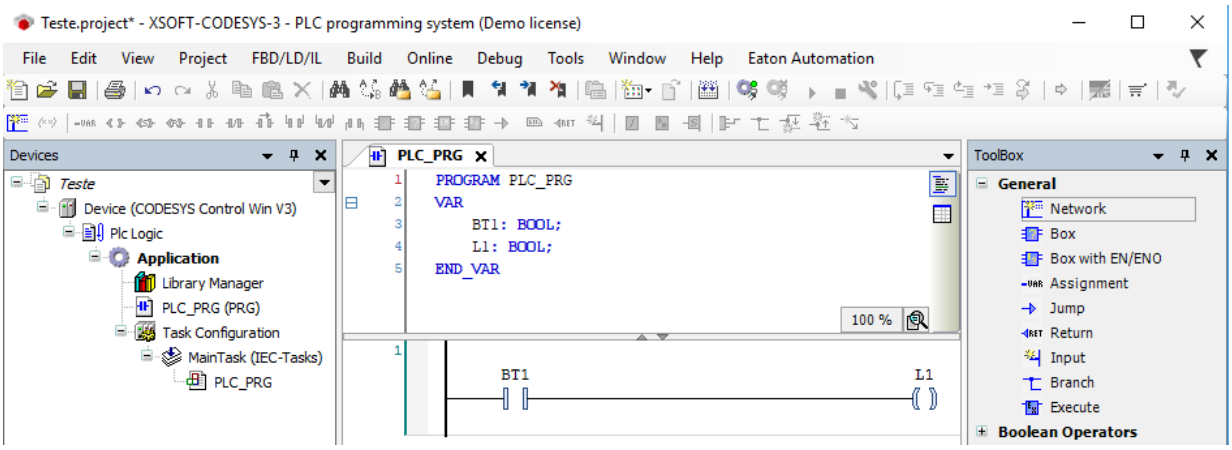


Figura 2.72: Ladder CODESYS - contatos de entrada e saída.

Agora precisamos criar um supervisório simples para emular um painel de interface com botões e lâmpadas. Para isso, clique com botão da direira sobre `Application -> Add Object -> Visualization`, selecione `Active` e `Add` (Figura 2.73).

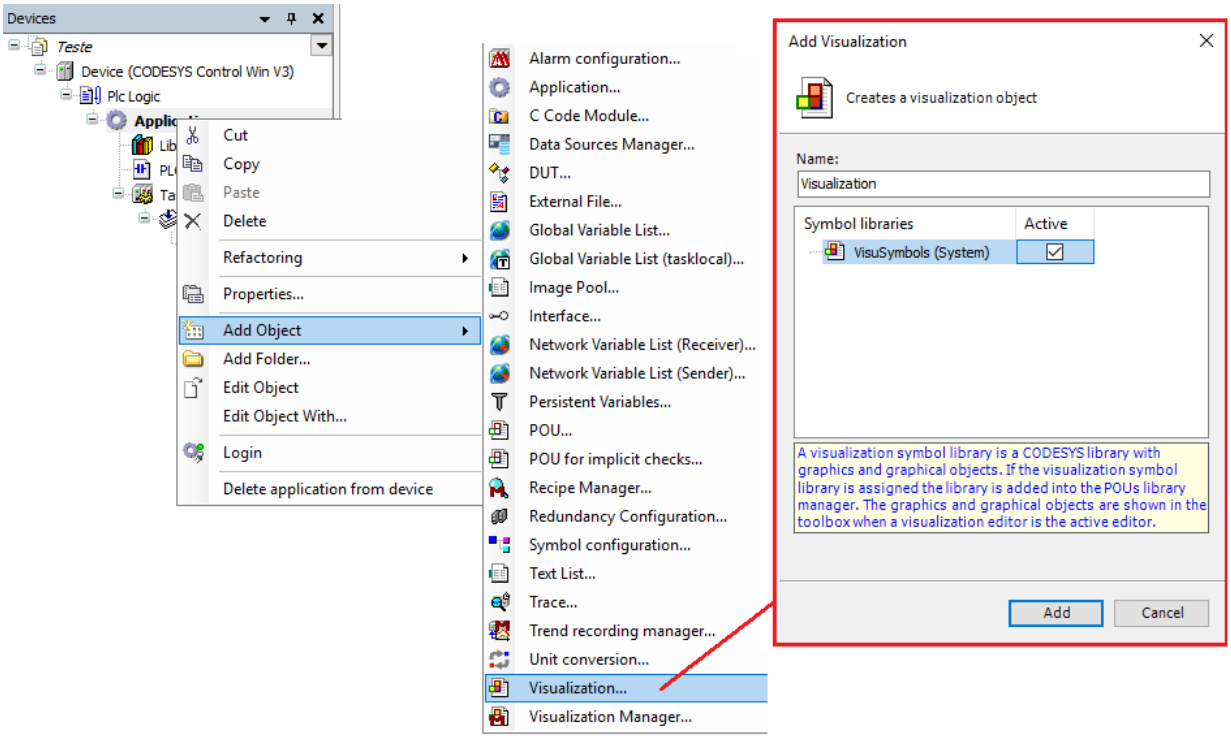


Figura 2.73: Criar painel Supervisório Simples.

Uma aba `Visualization` será criada com `Visualization Toolbox` na lateral inferior direita. Os objetos que utilizaremos estão em `Lamps/Switches/Bitmaps`, como visto na Figura 2.74.

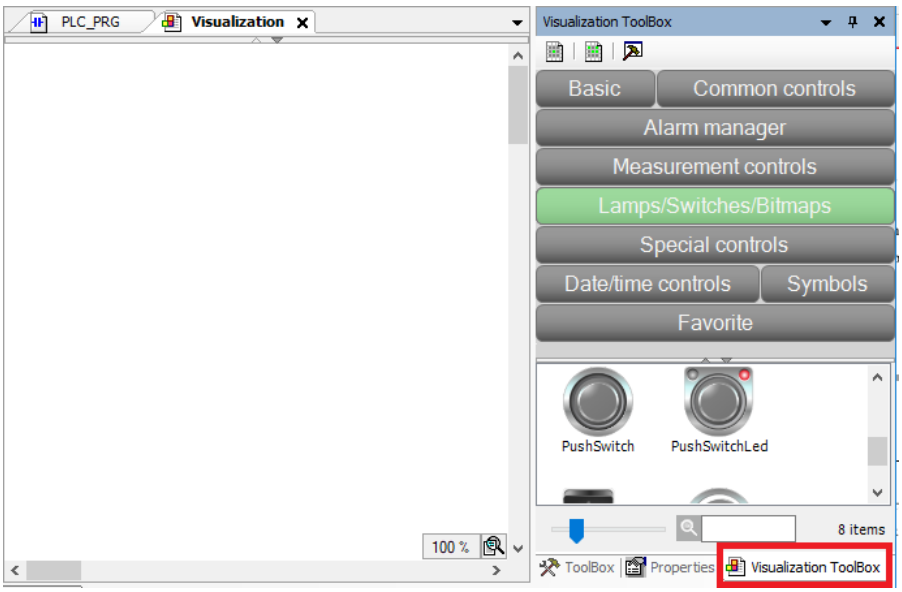


Figura 2.74: Aba `Visualization` e `Visualization Toolbox`.

Ao escolher o objeto `PushSwitch`, por exemplo, ele deve ser configurado. Clique sobre o desenho do botão e, inicialmente, configure o `Element behavior`. Na opção `Image toggler`, o botão será retentivo, ao passo que na opção `Image tapper`, ele será não retentivo. A segunda opção será escolhida nesta aplicação.

Em seguida, o botão deve ser mapeado na *tag* do contato NA de entrada BT1. Para isso, vá na opção `Variable` (clique no quadro à direira). Aparecerá um menu conforme a Figura 2.75. Escolha a variável `BT1` em `Applications -> PLC_PRG -> BT1`.

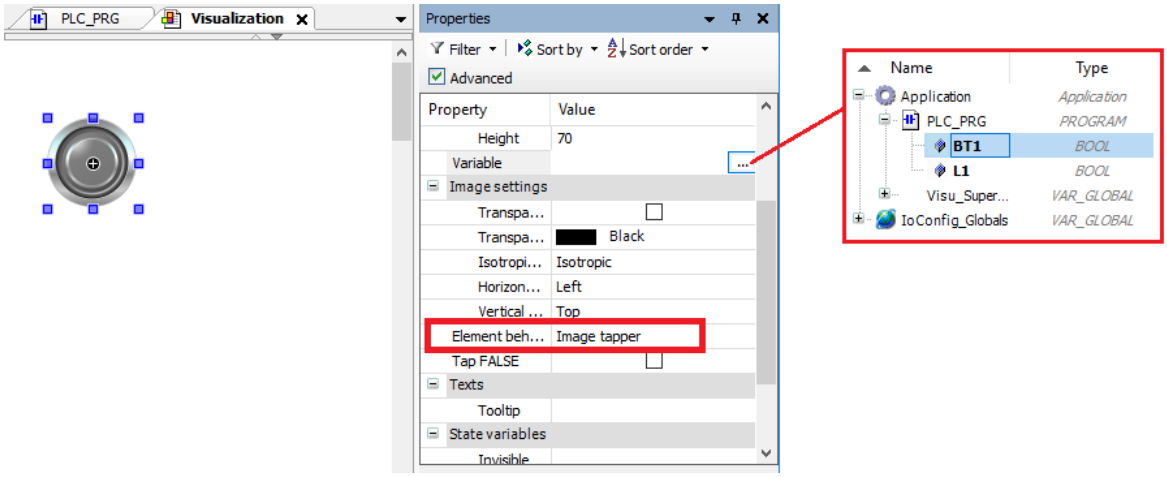


Figura 2.75: Configurar botão não retentivo em um contato de entrada.

Em seguida, insira uma lâmpada e mapeie na variável `L1`, resultando no esquema da Figura 2.76.

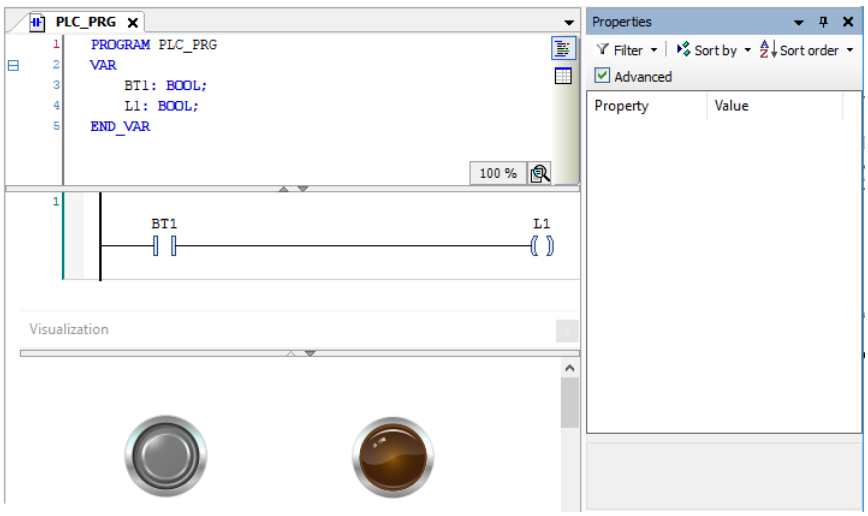


Figura 2.76: Aplicação simples com painel supervisório.

Pronto...o código já pode ser compilado e testado via simulação. Para isso, siga os seguintes passos:

1. Compile o código: `Menu Build -> Build` (ou F11);
2. Defina modo simulação: `Menu Online -> Simulation`;
3. Entre em modo Online: `Menu Online -> Login` (ou Alt+F8);
4. Inicie a simulação: `Menu Debug -> Start`;

A simulação é então executada. O código final é apresentado na Figura 2.77. Note que se pode observar o status das variáveis `BT1` e `L1` e os contatos acionados no diagrama Ladder. Para cancelar a simulação, vá em `Debug -> Stop` (ou Shift+F8) e depois em `Online -> Logout` (ou Ctrl+F8).

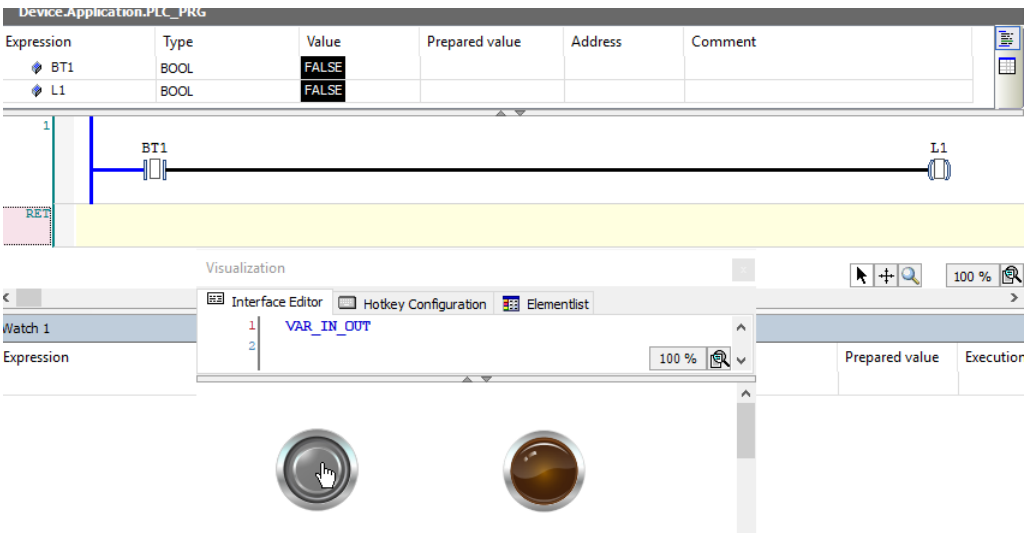


Figura 2.77: Primeira simulação CODESYS.

Vamos agora resolver o **Problema 3** no CODESYS. Utilizaremos a solução apresentada na Figura 2.43. Para inserir um ramo paralelo de entrada, pode-se utilizar a função , e para inserir um ramo paralelo do lado da saída, pode-se usar . Para inserir timer TON, considere a Figura 2.78.

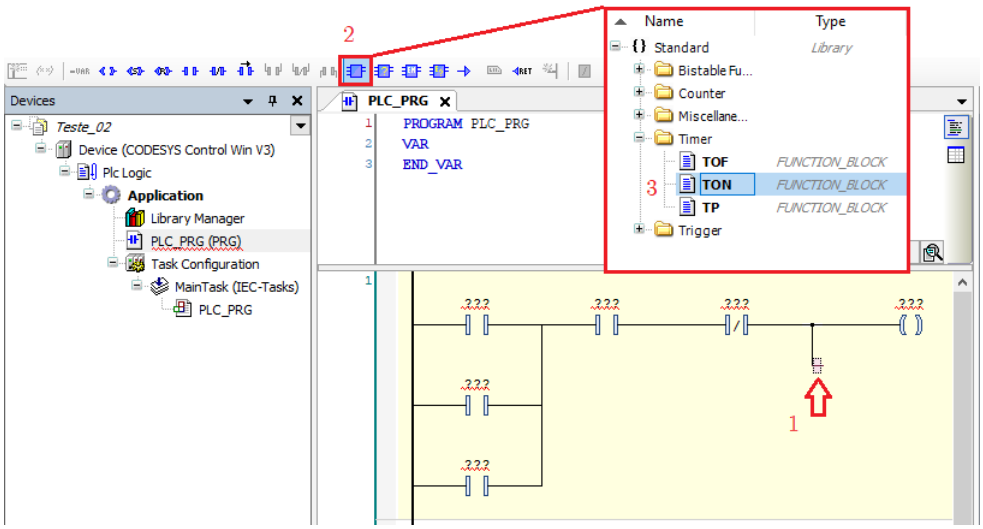


Figura 2.78: Inserir Timer TON.

Deve-se definir a variável `TON_0` como tipo `TON`. O TON é configurado da seguinte forma: em `PT`, deve-se inserir o valor do *preset* do *timer*, com a sintaxe `T#Xs`, em que `X` é o tempo em segundos. A variável para o `ET` (*estimate time*) , definida como tipo `TIME`, mostra o tempo acumulado do temporizador. Veja mais detalhes na Figura 2.79.

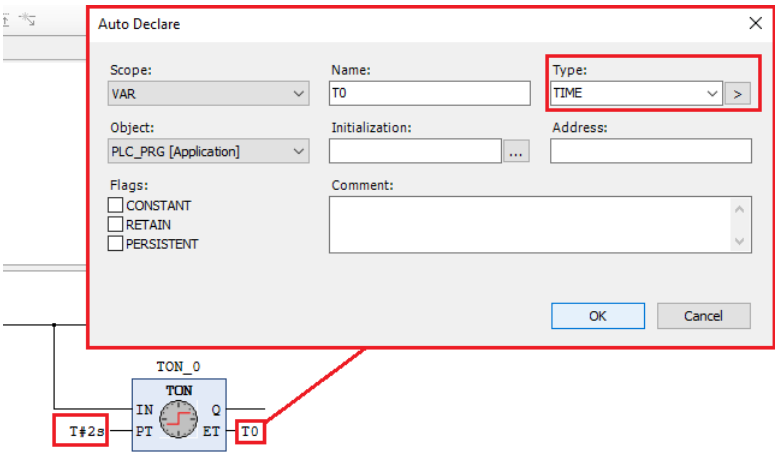


Figura 2.79: Configurar Timer TON.

Neste exemplo, ambos botões ON e OFF são do tipo NA. A Figura 2.80 apresenta a aplicação criada.

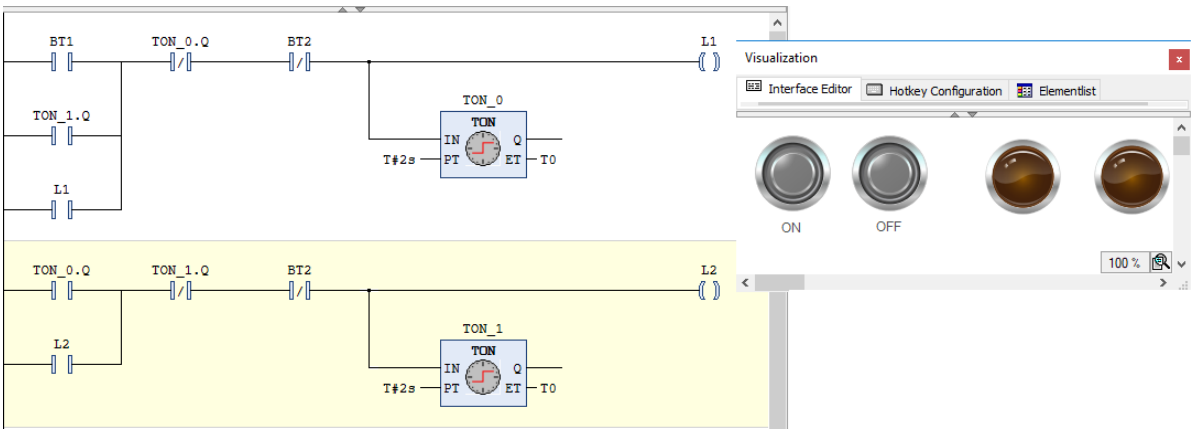


Figura 2.80: Solução do **Problema 3** no CODESYS. Os labels `ON` e `OFF` foram adicionados na opção `Label`, disponível no menu `Common controls` da `Vizualisation Tolbox`.

Para exemplificar o uso de contadores, vamos agora resolver o seguinte problema: ao pressionar um botão NA (`BT1`), as lâmpadas `L1` e `L2` devem piscar alternadamente, com período de um segundo, cinco vezes cada. Após isso, o ciclo pode ser reiniciado ao clicar em `BT1` novamente.

Assim como o temporizador, o contador UP (CTU) encontra-se na opção `Insert Box` (vide Figura 2.78), só que ele está no menu `Counter`. Deve-se definir a variável `CTU_0` como tipo `CTU`. A configuração é simples: em `PV` coloca-se o valor de *preset* do contador. Quando o acumulador chegar em `PV`, o bit `CTU_X.Q` vai para nível lógico 1. Em `CV` tem-se o valor atual do acumulador, que pode ser definido como uma variável tipo `WORD` ou tipo `INT`. A Figura 2.81 apresenta o código Ladder desenvolvido.

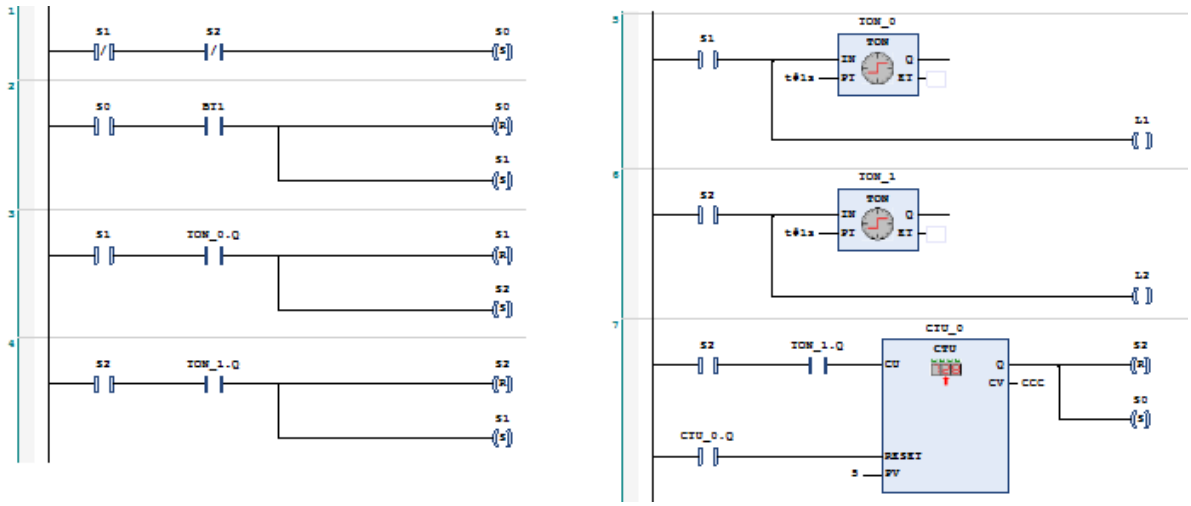


Figura 2.81: Solução do problema proposto em Ladder no CODESYS, com temporizador e contador.

Mais detalhes sobre o uso de SFC no CODESYS podem ser encontrados em: `https://help.codesys.com/api-content/2/codesys/3.5.12.0/en/_cds_programming_in_ld/`.

SFC CODESYS

Para ilustrar o uso de SFC no CODESYS, vamos novamente resolver o **Problema 3**. Ao criar um novo projeto, escolha agora a linguagem *Sequential Function Chart* (SFC). Ao selecionar `Main Task -> PLC_PRG`, o template da Figura 2.82 irá aparecer

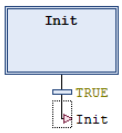


Figura 2.82: Template inicial de SFC no CODESYS.

Em seguida, o desvio para `Init` é apagado e um novo par Estado/Transição é adicionado, resultando na Figura 2.83.

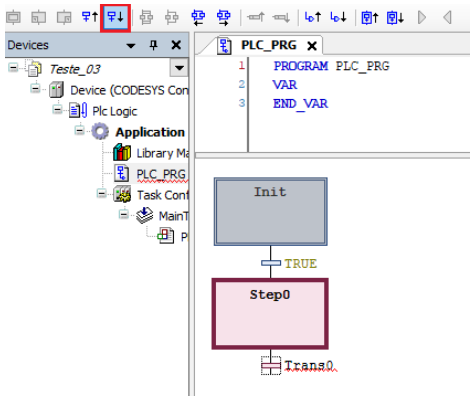


Figura 2.83: Inserção de par Estado/Transição.

A transição de `Init` para `Step0` ocorrerá ao clicar em `BT1` (NA). Quando `BT1` for associada à transição, a variável Booleana deve ser definida, como mostrado na Figura 2.84.

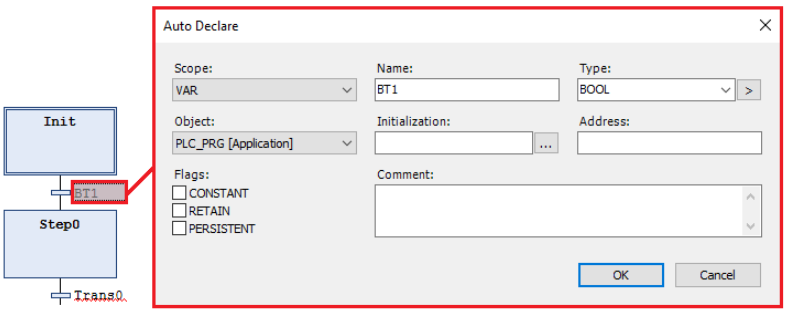


Figura 2.84: Criando *tag* do botão BT1.

Em seguida, atribui-se a(s) ação(ões) do estado `Step0` (Figura 2.85).

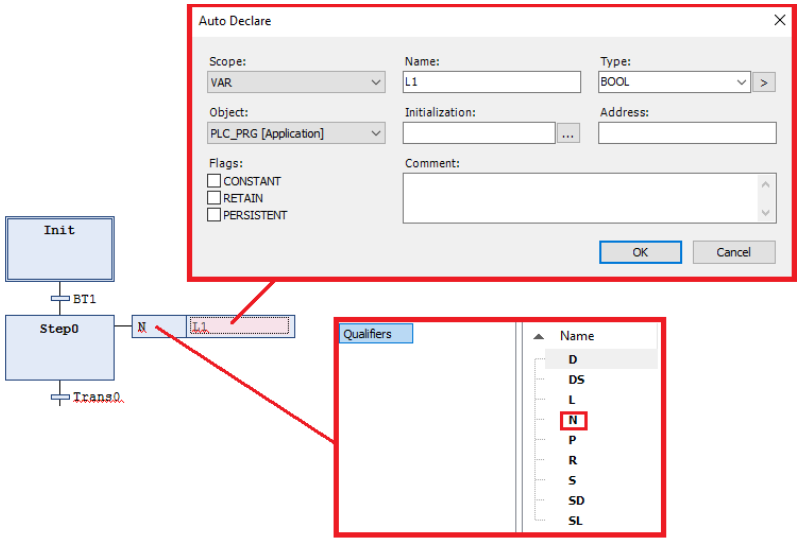


Figura 2.85: Inserir ação, definir tipo de ação e criar *tag* para a saída L1.

Um qualificador tipo `N` foi escolhido, pois deseja-se que a ação seja executada continuamente enquanto o estado estiver acionado.

Há uma ramificação de transição do estado `Step0`. Caso o tempo de execução de `Step0` for maior do que 2 segundos (`Step0.t>t#2s`), então deve-se ir para o estado `Step1` onde `L2` deve ser acionada (`L1` é apagada). Mas, se `BT2` for pressionado, então o sistema deve ser desviado para o estado `Init` e aguardar `BT1` ser novamente pressionado para reiniciar o ciclo. Para criar a ramificação, clique sobre a transição do `Step0` e escolha no menu o item `Insert Branch`. O diagrama ficará como mostrado na Figura 2.86 (a). Em seguida, configure as transições, como mostrado na Figura 2.86 (b).

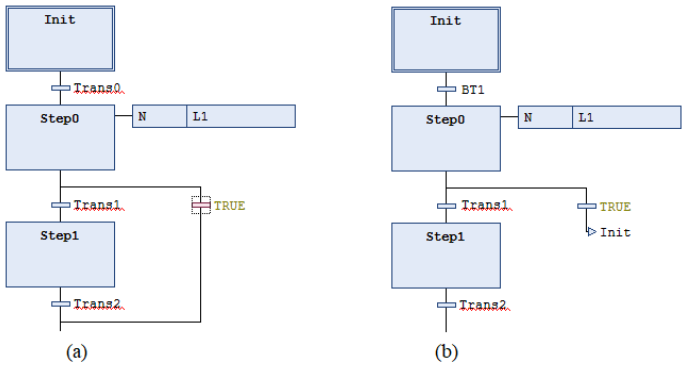


Figura 2.86: (a) Inserção de ramificação; (b) ajuste das condições de transição.

Seguindo os mesmos passos, o diagrama final é apresentado na Figura 2.87.

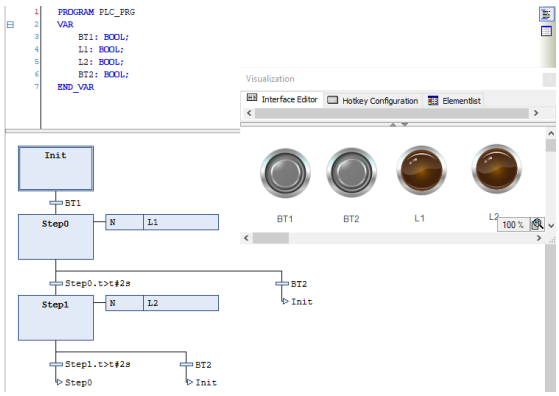


Figura 2.87: Diagrama final de uma solução com SFC no CODESYS para o **Problema 3**. O painel supervisório simples foi feito como no exemplo da linguagem Ladder.

Estado e transição devem ser inseridos em conjunto. Caso contrário, haverá erro na compilação. Em alguns casos, a variável `StepX.t` não aparece naturalmente ao definir uma transição de estados. Se isso ocorrer, clique com o botão direto sobre o estado de interesse e depois escolha `Browse -> Browse Call Tree`. A janela que vai abrir na parte inferior pode ser fechada.

Para exemplificar o uso de contadores no SFC do CODESYS, vamos resolver o seguinte problema: ao pressionar um botão NA (`BT1`), as lâmpadas `L1` e `L2` devem piscar alternadamente, com período de um segundo, cinco vezes cada. Após isso, o ciclo pode ser iniciado ao clicar em `BT1` novamente. Há também um botão de emergência (`BT2`) que para o processo. O código SFC está apresentado na Figura 2.88.

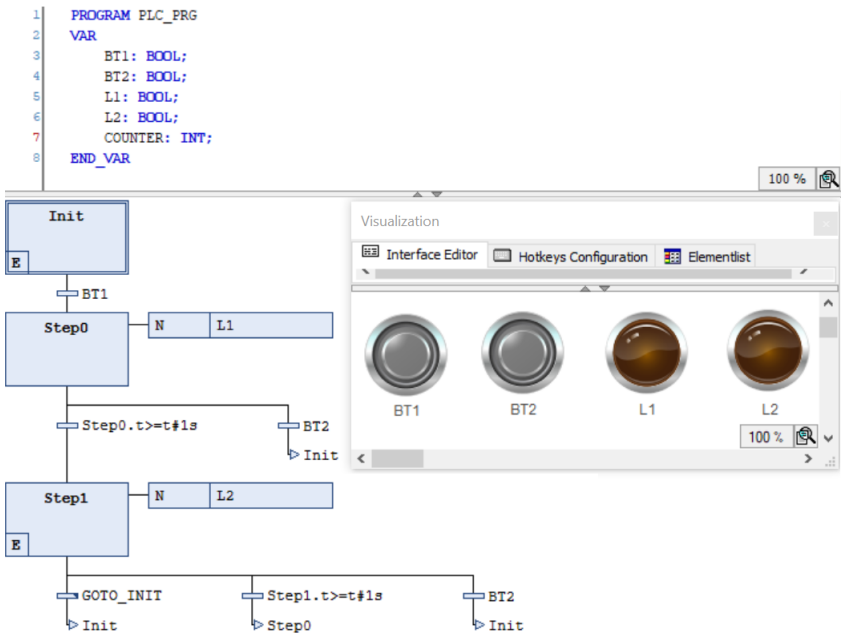


Figura 2.88: Solução do problema proposto em SFC no CODESYS, com temporizador e contador.

Para inserir um contador, inicialmente crie a variável `COUNTER` (pode escolher outro nome) como

tipo `INT` na definição de variáveis (entre `VAR` e `END_VAR`), como mostrado na Figura 2.89.

```
1  PROGRAM PLC_PRG
2  VAR
3      BT1: BOOL;
4      L1: BOOL;
5      L2: BOOL;
6      COUNTER:INT;
7  END_VAR
```

Figura 2.89: Exemplo de criação de uma variável tipo `INT`.

Em seguida, no estado `Step0`, cria-se uma ação de saída, clicando com o botão da diteira sobre o estado e escolhendo `Add exit action`, como visto na Figura 2.90.

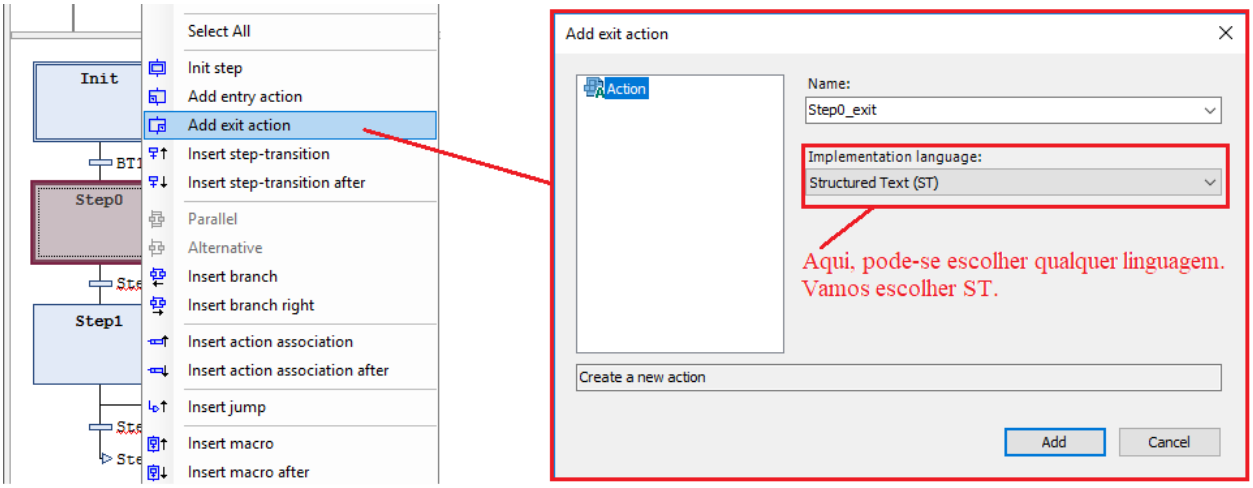


Figura 2.90: Criação de ação de saída de um estado.

Neste ponto vale ressaltar que qualquer linguagem de programação CLP pode ser utilizada para programar a ação. É uma maneira elegante de resolver problemas complexos. Aqui será escolhida a linguagem texto estruturado (`ST`). A programação de incremento da variável contador é feita como apresentado na Figura 2.91.

```
PLC_PRG   PLC_PRG.Step0_exit
1  COUNTER:=COUNTER+1;
```

Figura 2.91: Incremento de variável na linguagem `ST`.

Por outro lado, no estado `Init` foi criada uma ação de entrada `Add entry action`, também com programação `ST`, com o código `COUNTER:=0;`, ou seja, zerando o contador.

No ramo mais à esquerda do `Init` há uma transição denominada `GOTO_INIT`. Tal transição foi programada para desviar para o passo `INIT` quando o tempo de permanência em `Step1` for maior ou igual a 1 s e `COUNTER = 5`. Para inserir uma transição personalizada, clique com o botão da direita em `PLC_PRG (PRG) -> Add Object -> Action...`. Defina um nome para a transição e escolha

uma linguagem para programá-la. No caso deste exemplo, ST foi escolhida novamente e o código da Figura 2.92 foi programado.

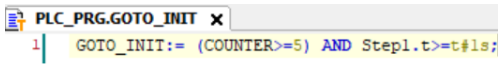


Figura 2.92: Transição programada em linguagem `ST`.

Assim, conforme visto na Figura 2.88, se o ciclo se repetir por cinco vezes, o programa salta para o estado inicial (`Init`), onde o contador é zerado e o ciclo pode ser reiniciado ao clicar em `BT1`.

Mais detalhes sobre o uso de SFC no CODESYS podem ser encontrados em: `https://help.codesys.com/api-content/2/codesys/3.5.12.0/en/_cds_programming_in_sfc/`

Há vários vídeos disponíveis na Internet sobre uso do CODESYS. Uma boa recomendação é o canal Youtube do Prof. Tohid Alizadeh, disponível em `https://www.youtube.com/channel/UC_qsTO2Z8tAt3Vx4P2SdH6w`

Atividades Adaptadas para o Simulador CODESYS

Exercícios simples

a) Para simular no CODESYS, utilize dois botões retentivos para simular $S1$ e $S2$ e uma lâmpada L para representar o acionamento do sistema.

b) Para simular no CODESYS, utilize botões não retentivos para simular $L1$, $L2$, $D1$ e $D2$, e duas lâmpadas para representar as saídas $M1$ e $M2$.

c) Para simular no CODESYS, utilize botões retentivos para simular os sensores A, B, C e D, e lâmpadas para representar as saídas $V1$, $M1$, $M2$ e $L1$.

Exercício usando contadores

a) No CODESYS, utilize botões não retentivos para $BT1$, $BT2$, $S1$ e $S2$, e lâmpadas para as saídas $M1$, $L1$ e $L2$.

Exercícios usando temporizadores

a) Adaptação para o simulador CODESYS é imediata.

b) Adaptação para SFC do simulador CODESYS é imediata.

Exercício usando contadores e temporizadores

a) No CODESYS, utilize botões não retentivos para as entradas $BT1$, $BT2$ e A, e lâmpadas para representar as saídas M e B.

b) Adaptação para SFC do simulador CODESYS é imediata.

CODESYS - Cancela em linha ferroviária

Para este exercício, crie (pelo menos) as variáveis mostradas na Figura 2.93.

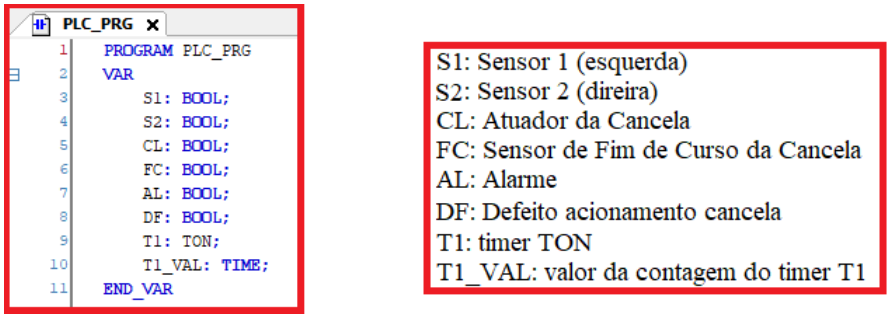
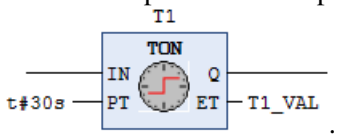


Figura 2.93: Conjunto mínimo de variáveis para o problema da cancela em linha ferroviária.

Os sensores `S1` e `S2` devem estar associados a botões não retentivos (configurar como `Image tapper`). A variável `DF` deve estar associada a uma chave HH alavanca, que simulará o defeito no acionamento da cancela, ou seja, caso a chave seja ligada, o sensor de fim de curso não acusa o abrimento da cancela (vide item (i) na descrição do problema).

As variáveis `CL`, `FC`, `AL` representam, respectivamente, o atuador da cancela, o sensor de fim de curso e o alarme. Essas três variáveis (de saída) devem ser mapeadas em lâmpadas indicativas.

No *timer* `T1`, tem-se a seguinte configuração: 

. Para simular o exercício, deve-se criar um painel como apresentado na Figura 2.94.

Figura 2.94: Painel para simular o problema da cancela em linha ferroviária.

O *display* para visualizar o tempo de `T1` é adicionado e configurado como mostrado na Figura 2.95.

CODESYS - Porta de um vagão de metrô

Para este exercício, crie (pelo menos) as seguintes variáveis apresentadas na Figura 2.96.

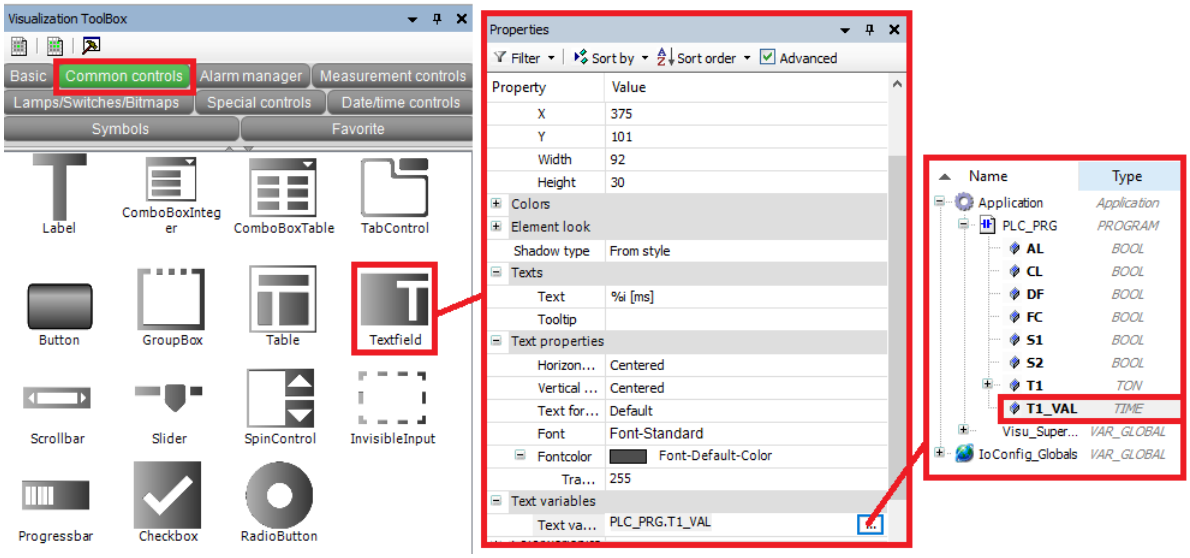


Figura 2.95: Inclusão de `Textfield` para mostrar o tempo acumulado no *timer*.

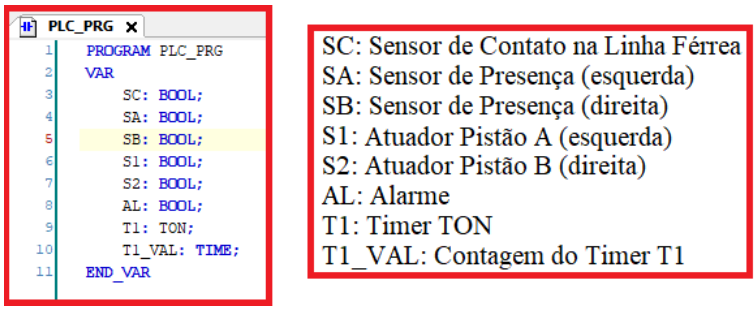


Figura 2.96: Conjunto mínimo de variáveis para o problema da porta de um vagão de metrô.

O sensor `SC` deve ser mapeado em um botão retentivo (configurar o botão como `Image toggler`). Os sensores de presença `SA` e `SB` devem ser mapeados em chaves HH alavanca. Os atuadores `S1` e `S2` dos pistões A e B, respectivamente, bem como o alarme `AL` devem ser mapeados em lâmpadas indicativas. Uma forma de fazer a contagem de tempo consiste em inserir apenas um *timer* (`T1`), ajustar seu *preset* para 60 segundos e comparar a variável `T1_VAL` com tempos especificados, conforme apresentado na Figura 2.97. Neste esquema, depois de 10 segundos, a saída `AL` será acionada. Fique à vontade para fazer de outra forma.

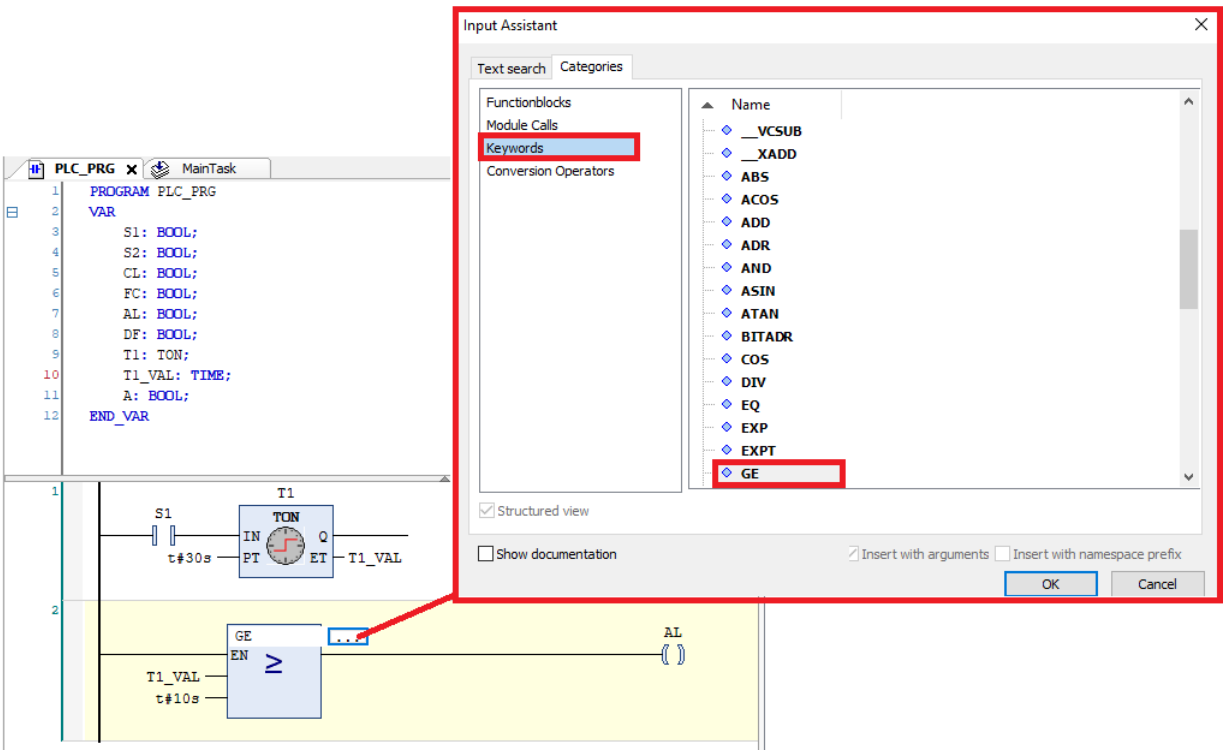


Figura 2.97: Uso de bloco de comparação GE (*Greater than or Equal to*) no Ladder do CODESYS.

Deve-se criar um painel como o apresentado na Figura 2.98 para simular o problema no CODESYS.

Figura 2.98: Painel para simular o problema da porta de um vagão de metrô.

Para fazer o item (b), insira uma variável `DF` (chave HH alavanca) que simula o defeito de fechamento das portas, mais dois sensores (`S1A` e `S1B`) (lâmpadas) indicando que as portas foram devidamente fechadas e mais uma lâmpada (`AKN`) para o sinal de "Acknowledge". Caso `DF` esteja acionado, quando os atuadores `S1` e `S2` dos pistões A e B forem acionados, as portas não fecham.

CODESYS - Sistema de Eclusas do Canal do Panamá

Considere as variáveis presentes na Figura 2.99. Esse é o conjunto mínimo que deve ser considerado no problema.

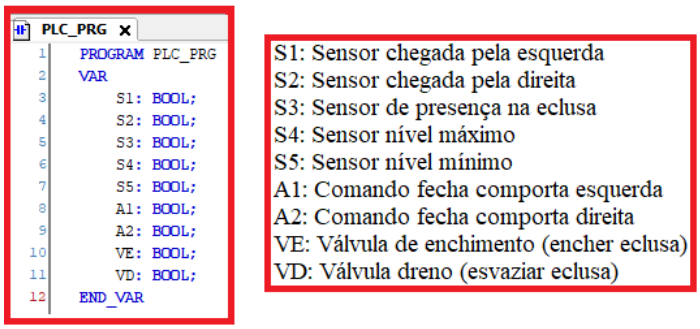


Figura 2.99: Conjunto mínimo de variáveis para o problema do Sistema de Eclusas do Canal do Panamá.

Os sensores `S1`, `S2` e `S3` dever ser mapeados em botões retentivos. Para fechar a comporta da esquerda, o comando (lâmpada) `A1` deve ser acionado, já para fechar a da direita, aciona-se o comando (lâmpada) `A2`. Para encher a eclusa, a válvula de enchimento (lâmpada) `VE` deve ser acionada e, para esvaziá-la, aciona-se a válvula (lâmpada) `VD`.

Considere um tempo arbitrário para encher ou esvaziar a eclusa. Não precisa usar *timer*. Quando o nível for máximo (eclusa cheia), o sensor de nível `S4` é acionado e, quando o nível for mínimo, o sensor `S5` é acionado. Utilize chaves HH alavanca para simular os sensores `S4` e `S5`.

Deve-se criar um painel como o apresentado na Figura 2.100 para simular o problema no CODESYS.

Figura 2.100: Painel para simular o problema do Sistema de Eclusas do Canal do Panamá.

Referências Bibliográficas

[AB16] Allen-Bradley. Programming manual - logix5000 controllers sequential function charts, 2016.

[Geo06] M. Georgini. *Automação aplicada: descrição e implementação de sistemas seqüenciais com PLCs*. Ed. Érica, São Paulo, Brasil, 2a edition, 2006.

[Jan07] J. Jantzen. *Foundations of fuzzy control*. John Wiley & Sons, West Sussex, England, 2007.

[Pru13] F. Prudente. *Automação Industrial PLC: Teoria e Aplicações*. LTC, Rio de Janeiro, Brasil, 2a edition, 2013.

[TAS94] Toshiro Terano, Kiyoji Asai, and Michio Sugeno, editors. *Applied Fuzzy Systems*. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[Ton77] R.M. Tong. A control engineering review of fuzzy systems. *Automatica*, 13(6):559 – 569, 1977.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.