UM LABORATÓRIO PARA UM CURSO DE AUTOMAÇÃO INDUSTRIAL UTILIZANDO A TEORIA DE SISTEMAS A EVENTOS DISCRETOS

José Ricardo da Silva Dias

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISISTOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

_____

Prof. João Carlos dos Santos Basilio, Ph. D.

_____

Prof. Fernando Cesar Lizarralde, D. Sc.

_____

Prof. Paulo César Marques Vieira, D. Sc.

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2005

DIAS, JOSÉ RICARDO DA SILVA

Um Laboratório para um Curso de Automação Industrial utilizando a Teoria de Sistemas a Eventos Discretos [Rio de Janeiro] 2005

VIII, 127 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Sistemas a Eventos Discretos
2. Autômato
3. Redes de Petri
4. Controladores Lógicos Programáveis
5. Linguagem de Programação Ladder
3. Experiências de Laboratório

I. COPPE/UFRJ II. Título (série)

À minha mãe Conceição, à minha esposa Andréa.
Aos meus filhos André Ricardo e Samuel Elias, e
aos meus irmãos Eduardo, Fátima, Berna, Glória e Bete.

AGRADECIMENTOS

A Deus Pai pela vida, pela saúde, cuidado, proteção e disposição concedidos sem medida e a Jesus, por seu sacrifício e sua interseção por nós.

À minha mãe, por ter dedicado sua vida para que nós (seus filhos) tivéssemos condições de estudar e todo o seu esforço para fazer-nos pessoas de bem, e aos meus irmãos que sempre me ajudaram nas dificuldades.

À minha esposa Andréa e meus filhos André e Samuel, pela compreensão, paciência e apoio durante mais essa jornada.

Aos professores Afonso Celso, Amit Bhaya, Fernando Lizarralde, Liu Hsu e Ramon Romankevicius da UFRJ/COPPE, à professora Marly Guimarães e ao professor Cícero Costa da UFAM e à SUFRAMA, pela disposição e colaboração no ensino.

Ao professor Dionísio pelo empréstimo de vários livros e a todos que, de maneira direta e indireta , colaboraram para a conclusão deste trabalho.

E, em especial, ao meu professor e orientador João Carlos dos Santos Basílio, que mesmo enfrentando problemas de saúde, nunca desanimou e nem me fez desanimar, continuando sempre firme para conclusão deste trabalho, e pelos laços de amizade que se formaram pela convivência, mesmo que à distância.

Resumo da Teses apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

UM LABORATÓRIO PARA UM CURSO DE AUTOMAÇÃO INDUSTRIAL UTILIZANDO A TEORIA DE SISTEMAS A EVENTOS DISCRETOS

José Ricardo da Silva Dias

Abril/2005

Orientador: João Carlos dos Santos Basílio

Programa : Engenharia Elétrica

O ensino de automação industrial, em cursos de engenharia, além dos fundamentos teóricos, requer, a nível de projeto, a utilização de redes de Petri e na implementação, a prática com o hardware e o software dos controladores lógicos programáveis.

Este trabalho tem por finalidade elaborar experiências para um laboratório de automação industrial utilizando a teoria de sistemas a eventos discretos, tratando de algumas técnicas usadas para modelagem desses sistemas. As experiências serão implementadas utilizando-se um controlador lógico programável (CLP) que será programado em linguagem ladder para executar o controle de seqüências de eventos pré-determinadas. O modelo a ser implementado será formalizado em rede de Petri e posteriormente gerado um programa em linguagem ladder de forma heurística.

As teorias de autômatos e redes de Petri, que são alguns dos formalismos utilizados para representar um sistema a eventos discretos, serão também estudadas neste trabalho.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master in Sciences (M. Sc.)

A LABORATORY FOR A COURSE IN INDUSTRY AUTOMATION USING THE THEORY OF DISCREET EVENTS SYSTEMS

José Ricardo da Silva Dias

April/2005

Advisor: João Carlos dos Santos Basilio

Department: Electrical Engineering

The teaching of industry automation, at undergraduating level, and its besides theoretical background, also requires the use of Petri Nets implementation using the hardware and software of programmable logic controllers.

The objective of this work is to propose experiments for a laboratory of a course in industry automation using the theory of discrete events systems, dealing with some techniques used for modeling these systems. The experiments will be implemented using a programmable logical controller (PLC) that will be programmed in ladder language to execute the control of pre-determined sequences of events. The model to be implemented will be formalized in Petri net and in the sequel, a program in ladder language will be developed in a heuristic way.

Automata and Petri nets theories, some of the formalisms used to represent a discrete events system will also be studied in this work.

Sumário

Capítulo 1. Introdução

As queixas mais freqüentes de alunos dos cursos técnicos da área de indústria são a falta de práticas nos laboratórios do curso e a necessidade de mais embasamento prático para que os alunos possam se desenvolver melhor durante seus estágios curriculares.

Segundo MARTIN e BROWN (1998), durante os últimos dez anos, foram administradas entrevistas com os alunos dos cursos superiores do Departamento de Engenharia Elétrica da Universidade de Engenharia Elétrica de Arkansas em Fayetteville, Arkansas 72701, aproximadamente 2 semanas antes de completarem o semestre final. Dos muitos anos de entrevistas, ficou claro que muitos estudantes sentiam que as disciplinas do currículo carecem de treinamento prático adequado, i.e., não bastava dispensar algum tempo no laboratório, se que os exercícios de laboratório não estiverem bem relacionados ao material dirigido em sala de aula. Além disso, nas discussões abertas com empregadores dos mesmos estudantes, verificou-se também que os alunos precisam de uma exposição a uma variedade mais ampla de experiências e equipamentos.

Um laboratório de sistemas de engenharia é uma amálgama de criar métodos, habilidades, princípios e disciplinas. Os métodos de engenharia incluem descoberta, avaliação e investigação. As habilidades de engenharia aplicáveis incluem experimentação, análise de dados e modelagem, (MIDDLETON *et al.*, 1996).

Enquanto foram aplicadas idéias inovadoras a muitos aspectos do currículo de engenharia em recentes anos (GRAYSON, 1994, ASEE, 1994, BORDOGNA *et al*., 1993, CIBUZAR *et al*., 2001), o componente de laboratório geralmente arrastou-se no processo de reforma, com a exceção de algum progresso notável ao nível de iniciantes (QUINN, 1993). Ainda hoje, os empregadores das indústrias pedem que a educação dos estudantes seja recheada de inovações para adquirir experiências práticas afinadas com habilidades de integração necessária aos engenheiros modernos. De acordo com Norman Augustine (Lockheed Martin Corporation), “Uma educação de engenharia tem que incluir aplicações claras que certamente têm que incluir as mãos no trabalho....” (AUGUSTINE, 1994), De acordo com o Conselho de Engenharia, o currículo tem que encarnar uma “perspectiva de sistemas”, uma “perspectiva multi-disciplinar”, e uma “integração de conhecimento”. (BAUM *et al.*, 1994).

A teoria de sistemas a eventos discretos é um campo de conhecimentos em expansão. Seu surgimento justifica-se, entre outras coisas, em face da necessidade de um tratamento formal requerido por diversos sistemas construídos pelo homem, como redes de comunicação, sistemas de manufatura, sistemas de tráfego automatizado e sistemas computacionais, guiados a eventos cujo tratamento, baseado classicamente em equações diferenciais, se torna extremamente complexo. A teoria tem caráter interdisciplinar e inclui princípios e conceitos extraídos da ciência da computação, teoria de controle e pesquisa operacional (KUMAR e GARG, 1995).

Devido ao grande avanço na pesquisa em torno de formalismos e linguagens que podem executar, implementar e controlar um sistema a eventos discretos, é de significativa importância a criação de um laboratório que vise tratar do assunto em termos de implementação, pois muito do que é visto tem caráter apenas conceitual. A experiência de ensino em engenharia de automação industrial, em nível de graduação, tem mostrado a importância de três elementos: a) a prática com o hardware e o software dos Controladores Lógicos Programáveis; b) as redes de Petri aplicadas ao projeto de automação; c) um conjunto consistente de experiências de laboratório (MORAES e CASTRUCCI, 2002).

É evidente a importância para a economia nacional de um maior número de graduados em Automação Industrial, e especificamente com experiência nos Controladores Lógicos Programáveis (CLPs). As inúmeras aplicações possíveis exigem do engenheiro: i) presença nas longas fases da especificação, em diálogo com o cliente; ii) projeto do sistema; iii) geração do software do PLC; iv) “start-up”, na própria planta industrial, isto é, hoje em dia, em qualquer parte do território nacional. (MORAES e CASTRUCCI, 2002). Também, segundo MORAES e CASTRUCCI (2002), parece difícil a migração de tais instrumentos para a prática do engenheiro. Nesta, portanto, o projeto depende de tentativas, de intuições, de simulações, enfim de “engenho e arte”.

O projeto de sistemas de controle automáticos requer muito conhecimento teórico. A conseqüência deste fato é que um número grande de conceitos novos tem que ser introduzido em um primeiro curso em sistemas de controle. Porém, estes conceitos são introduzidos em geral de alguma maneira independentemente e isto coloca sérios problemas quando os estudantes são exigidos a lidar com o projeto inteiro de um sistema de controle. Neste sentido um laboratório de controle deve ser proposto com a visão a reunir todos os conceitos introduzidos em um curso teórico ministrado previamente (BASILIO, 2002, BASILIO e MOREIRA, 2004). No que se refere a

sistemas a eventos discretos, para o conhecimento das ferramentas de modelagem é necessário um profundo conhecimento dos formalismos de modelagem utilizados em sistemas a eventos discretos, um conhecimento amplo dos sistemas de controladores lógicos programáveis e suas linguagens de programação e uma metodologia na elaboração de projetos, com base em linguagem ladder e redes de Petri. Segundo VENKATESH e ZHOU (1998), um sistema industrial "discreto" consiste de várias unidades simultâneas como máquinas, robôs, veículos com guias automatizados, controladores lógicos programáveis e computadores que funcionam de forma assíncrona para se encontrar as necessidades dinâmicas das variáveis do mercado. Softwares integrados que desenvolvem métodos para modelar, analisar, controlar, e simular tais sistemas são também importantes.

Muitos usuários industriais de CLPs preferem programar em linguagem ladder que usa métodos heurísticos. Para sistemas simples, é fácil escrever para programas de PLC que usam o método heurístico. Porém, à medida que o sistema se torna mais complexo, fica muito difícil de controlar problemas efetivamente (CHIRN e McFARLANE, 1999, VENKATESH *et al.*, 1994*b*). Estes problemas foram reconhecidos desde que a linguagem ladder foi extensamente usada. Alguns nivelamentos foram propostos às ferramentas de projeto para ajudar a solucionar estes problemas (IEC, 1992; DAVID, 1995). Redes de Petri (PETERSON, 1981, ZURAWSKI e ZHOU, 1994) são ferramentas comumente usadas neste aspecto, por causa do sucesso em sistemas de controle de evento discretos (SED).

O objetivo desse trabalho não é comparar redes de Petri e linguagem ladder, mas mostrar que existem estudos a este respeito e na parte de elaboração das experiências de laboratório mostrar as duas formas de representação. Contudo os controles serão implementados usando a linguagem ladder.

Por causa do planejamento e vantagens de organização de redes de Petri, vários investigadores tentaram desenvolver métodos para transformar redes de Petri (desenvolvidos na fase de projeto) em linguagem ladder (fase de implementação) (SATO e NOSE, 1995, JAFARI e BOUCHER, 1994, BURNS e BINDANDA, 1994, TAHOLAKIAN e HALES, 1997, VENKATESH *et al.*, 1994*a*, LEE e HSU, 2001). Além disso, UZAM *et al*. (1996) propôs um método lógico para converter redes de Petri em linguagem ladder. Porém, estas aproximações estão focalizadas tipicamente somente na fase de projeto em lugar de as outras fases do desenvolvimento de sistemas de controle. As metodologias apontam para traduzir a rede de Petri na sintaxe de

linguagem ladder. Porém, problemas na fase de teste e fase de manutenção posterior ainda são grandes. Não somente o tempo de projeto mas também o tempo de manutenção pode ser reduzido se uma aproximação apropriada estiver disponível (CHIRN e McFARLANE, 2000). Mais recentemente, LUCAS e TILBURY (2003) conduziram um estudo com o objetivo de determinar os métodos atuais de projeto de sistemas de automação usados na industria automotiva. Neste sentido, verificou-se que embora a linguagem ladder seja ainda a mais empregada, é necessário, dada a necessidade de constantes modificações das programações nas linhas de produção, que outras formas de implementações devam ser usadas. Neste contexto, surge a chamada linguagem SFC (sequential flow chart) que permite uma implementação mais rápida dos programas no sistema industrial. Apesar disto, neste trabalho, a implementação será feita usando-se linguagem ladder.

Este trabalho está estruturado da seguinte forma. O capítulo 2 apresenta os fundamentos da teoria de sistemas a eventos discretos: autômatos e redes de Petri. O autômato como formalismo de modelagem é discutido em detalhe. É apresentado, também, o formalismo de modelagem por redes de Petri e discute-se a análise e o controle de modelos de rede de Petri independente do tempo. No final do capítulo, as duas classes de modelos independentes do tempo, autômato e redes de Petri, são refinadas para incluir o "tempo" por meio de uma estrutura de temporização, resultando no *autômato temporizado* e nas *redes de Petri temporizadas.*

O capítulo 3 descreve de forma geral um CLP (software e hardware), com suas definições e características principais. Os principais blocos que compõem um CLP são descritos. É feita uma classificação dos tipos de CLPs. A linguagem que será utilizada para implementação das experiências deste trabalho é a linguagem ladder. São mostradas as implementações das funções lógicas utilizadas em projetos de circuitos digitais (funções NOT, AND, OR, NAND, NOR, OR Exclusivo e NOR Exclusivo). Finalizando o capítulo, é feita uma comparação entre linguagem ladder e redes de Petri, onde são descritos alguns métodos relacionados à tentativa de conversão direta entre redes de Petri e linguagem ladder.

No capítulo 4 são apresentados todos os equipamentos que serão utilizados no laboratório proposto, quais sejam: o CLP TSX 3722 (hardware e software), a esteira transportadora, o dispositivo com um conjunto de lâmpadas e o conjunto de chaves sem retenção. Cada um desses equipamentos é detalhado de forma a mostrar suas características técnicas. Também é descrito o programa PL7 Micro que é o software

destinado ao modelo de CLP utilizado, dando uma visão geral do ambiente de programação e dos recursos e ferramentas que são usadas para criação dos programas em linguagem ladder.

O capítulo 5 é dedicado aos experimentos que serão propostos. São sete experimentos ao todo, sendo dois experimentos utilizando-se os recursos do CLP para acionamento de uma carga, no caso uma lâmpada, três experimentos com o conjunto de esteiras, procurando simular um ambiente de produção e dois experimentos utilizando o conjunto de lâmpadas, simulando o funcionamento de semáforos.

Finalmente, no capítulo 6 são apresentados as conclusões e os trabalhos futuros.

Capítulo 2. Fundamentos da teoria de sistemas a eventos discretos

Um sistema a eventos discretos (SED) é um sistema a estado discreto, dirigido por eventos, ou seja, sua evolução de estado depende inteiramente da ocorrência de eventos discretos assíncronos no tempo. Neste sentido, ATTIÉ (1998) escreve que "quando o espaço de estados de um sistema é naturalmente descrito por um conjunto discreto, e as transições de estado são observadas somente em pontos discretos do tempo, associam-se estas transições a eventos. O conceito de evento é um desses conceitos primitivos, cuja compreensão deve ser deixada à intuição, mais do que a uma exata definição. Não se pode, porém, deixar de enfatizar que um evento deve ser pensado como de ocorrência instantânea e como causador de uma transição no valor (discreto) do estado do sistema".

Neste capítulo será feita uma revisão dos principais conceitos e dos fundamentos da teoria de sistemas a eventos discretos, estando estruturado da seguinte forma: na seção 2.1 são apresentados os fundamentos da teoria de sistemas a eventos discretos, com a introdução dos conceitos de evento, sistemas controlados pelo tempo e sistemas baseados em eventos, tratados em comparação aos sistemas dinâmicos de variáveis contínuas; na seção 2.2 são estudado as linguagens e autômatos, e; na seção 2.3 é estudado o segundo formalismo utilizado neste trabalho que são as rede de Petri, como uma alternativa para substituir os modelos de autômatos de SED, sendo apresentados os fundamentos de redes de Petri, as equações de estado e as dinâmicas da rede de Petri para uma classe especial de redes, comparando, ao final, redes de Petri e autômato; na seção 2.4 é introduzida a estrutura de temporização nas redes de Petri, gerando as redes de Petri temporizadas.

2.1. Sistemas a eventos discretos

Nesta seção serão apresentados os principais conceitos para o estudo de sistemas a eventos discretos.

2.1.1. Evento

"Evento" é um conceito primitivo e necessita de uma boa base intuitiva para compreender o seu significado. Deve ser enfatizado que um evento deve ser pensado como alguma coisa acontecendo instantaneamente e que causa transições de um valor

de estado para outro. Um evento pode ser identificado como uma ação específica; por exemplo alguém aperta um botão, um computador deixa de funcionar, a chave de ignição de um automóvel é ligada etc, ou pode ser o resultado de várias condições que, de repente, acontecem.

Neste trabalho será usado o símbolo "*e*" para denotar um evento. Ao considerar um sistema afetado por tipos diferentes de eventos, define-se um conjunto "*E*" cujos elementos são todos estes eventos. Claramente, *E* é um conjunto discreto.

O conceito de evento pode ser melhor entendido com a ajuda do seguinte exemplo: um sistema de armazenamento de cargas. Neste caso pode-se perfeitamente verificar que há, no mínimo, dois eventos: um evento é a "chegada de produto" e o outro é a "chegada de caminhão". Neste caso, pode-se definir um conjunto de eventos $E = \{P,T\}$ onde *P* denota o evento "chegada de produto", e *T* denota o evento "chegada de caminhão", que corresponde à "saída do produto".

2.1.2. Sistemas controlados pelo tempo e sistemas baseados em eventos

Em sistemas de estados contínuos o estado muda geralmente com mudanças de tempo. Isto é particularmente evidente em modelos a tempo-discreto: um sinal de "clock" determina a seqüência de amostras a serem obtidas, pois é esperado que a cada marcação desse sinal, ocorra uma mudança no estado do sistema. Neste caso, a variável de tempo (*t,* em tempo contínuo, ou *k*, em tempo-discreto) é uma variável independente que aparece como sendo o argumento de toda a contribuição de entrada dos estados, e em funções de saída. Por essa razão, esses sistemas são denominados dirigidos pelo tempo.

Em sistemas de estados discretos, as mudanças de estado só ocorrem em certos pontos por transições instantâneas e, a cada uma dessas transições, pode-se associar um evento. Suponha que exista um relógio pelo qual é tomado o tempo, e considere as duas possibilidades:

1. A toda marcação do sinal de clock, um evento *e* será selecionado de um conjunto fixo *E*. Se nenhum evento acontecer, pode-se pensar em um "evento nulo" como pertencendo a *E* cuja propriedade é não causar nenhuma mudança de estado.
2. Em vários momentos de tempo (não necessariamente conhecidos com antecedência, e não coincidindo com as marcações de tempo), algum evento determinado *e* irá ocorrer.

Há uma diferença fundamental entre 1 e 2 acima. Em 1, as transições de estado são sincronizadas pelo relógio, isto é, a toda marcação de tempo, um evento (ou nenhum

evento) é selecionado. O tempo é responsável por toda e qualquer possível transição de estado. Em 2, todo evento *e* de *E* define um processo distinto pelo qual os momentos de tempo, quando *e* acontece, são determinados. As transições de estado são o resultado das combinações destes processos de eventos assíncronos e simultâneos. Além disso, estes processos não precisam ser independentes um do outro.

A distinção entre 1 e 2 dá origem às definições de sistemas dirigidos pelo tempo (1) e sistemas dirigidos por eventos (2). É importante ressaltar que a idéia de transições de estado baseadas em eventos corresponde a uma noção familiar, que é de uma "interrupção" em sistemas de computador. Enquanto muitas das funções em um computador são sincronizadas por um relógio, e são controladas pelo tempo, outras são resultados de chamadas assíncronas que podem acontecer a qualquer hora como, por exemplo, o pedido de um usuário externo ou uma mensagem de intervalo pode acontecer como resultado de eventos específicos, mas completamente independentes do relógio do computador.

2.1.3. Propriedades características de sistemas a evento discretos

A maioria dos sistemas de controle em engenharia são baseados em modelos de equações diferenciais ou em equações a diferenças lineares. Para usar estes modelos matemáticos, esses sistemas devem satisfazer as seguintes propriedades: devem ser de estado contínuo; com o mecanismo de transição de estado dirigido pelo tempo. A primeira propriedade permite definir o estado por meio de variáveis contínuas que podem assumir qualquer valor real (ou complexo). Quantidades físicas comuns como posição, velocidade, aceleração, temperatura, pressão, fluxo etc., estão nesta categoria desde que se possa definir naturalmente as derivadas para estas variáveis contínuas. A segunda propriedade vem o fato de que o estado geralmente evolui em função do tempo.

Os sistemas considerados neste trabalho são os Sistemas Dinâmicos a Eventos Discretos (SDED) ou, mais amplamente, Sistemas a Eventos Discretos (SED). As suas principais características são: (i) o espaço de estado é um conjunto discreto; (ii) o mecanismo de transição de estados é baseado em eventos.

Essas propriedades levam a seguinte definição de SED.

Definição 2.1. Um Sistema a Eventos Discretos (SED) é um sistema de estado discreto baseado em eventos, isto é, a evolução dos estados depende somente da ocorrência de eventos discretos assíncronos. □

Muitos sistemas, particularmente tecnológicos, são na realidade sistemas de estados discretos. Até mesmo se este não for o caso, para muitas aplicações de interesse, uma visão de estado discreto de um sistema complexo pode ser necessária. Alguns exemplos simples de sistemas de estados discretos são: (i) O estado de uma máquina pode ser selecionado de um conjunto como {LIGADA, DESLIGADA} ou {OCUPADO, OCIOSO, LIVRE}; (ii) um computador que executa um programa pode ser visto como estando em um de três estados: {ESPERANDO POR INSTRUÇÕES, EXECUTANDO, PARADO}; (iii) qualquer tipo de inventário que consiste de valores discretos (por exemplo produtos, unidades monetárias, pessoas) tem um espaço de estado natural nas grandezas não negativas {0,1,2,...}, (iv) a maioria dos jogos pode ser modelado como tendo um espaço de estado discreto (em xadrez, por exemplo, toda possível configuração do tabuleiro define um estado); o espaço resultante é enorme, mas é discreto.

A propriedade baseada em eventos de SED decorre do fato de que o estado só pode mudar no tempo em pontos discretos, que correspondem fisicamente a ocorrências assíncronas de eventos discretos. De um ponto de desenvolvimento de um modelo, isto tem a seguinte implicação: se for possível identificar um conjunto qualquer de "eventos" que podem causar uma transição de estado, então o tempo já não serve ao propósito de dirigir tal sistema e não pode ser uma variável independente apropriada.

As duas características fundamentais que distinguem Sistemas Dinâmicos de Variáveis Contínuas (SDVC) de SED são claramente mostradas ao se comparar trajetórias típicas de cada uma destas classes de sistema, como na figura 2.1. Para o SDVC mostrado, o espaço de estado X é o conjunto de números reais R, e x(t) pode assumir algum valor fixo. A função x(t) é a solução da equação diferencial $\dot{x}(t)=f[x(t), u(t), t]$, onde $u(t)$ é a entrada. Para o SED, o espaço é algum X fixo e discreto é igual a $\{s_1, s_2, s_3, s_4, s_5, s_6\}$. De acordo com a trajetória mostrada na figura 2.1.b, o estado só muda de um valor para outro se um evento ocorrer. Vê-se, inclusive, que um evento pode acontecer, mas não causar uma transição de estado, como no caso de e_3.

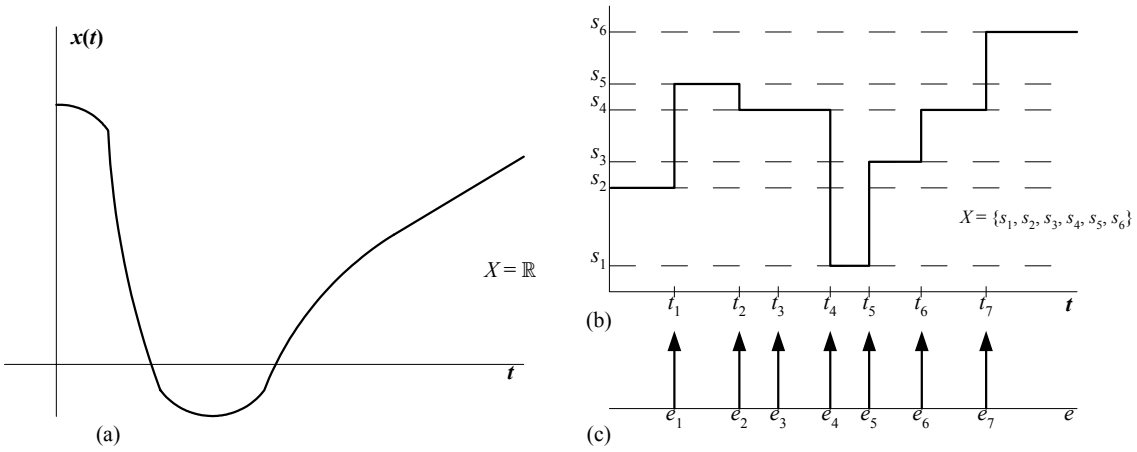


Figura 2.1: Comparação de caminhos de amostra para Sistemas Dinâmicos de Variáveis Contínuas (SDVC) e Sistemas de Evento Discretos (SED).

2.1.4. Exemplos de sistemas a evento discretos

Nesta seção serão apresentados três exemplos de SED utilizados no mundo real e experiências comuns em engenharia. O primeiro desses exemplos representa uma estrutura simples que servirá para representar muitos SED de interesse.

1. Sistemas de filas

O termo fila decorre de um fato intrínseco que em muitos dos sistemas mais comuns, para se usar certos recursos, deve-se esperar. Por exemplo, para usar os recursos de um caixa de banco, as pessoas formam uma fila e esperam; para usar o recurso de um caminhão, produtos acabados esperam em um armazém. Semelhantemente, para usar os recursos da CPU, várias tarefas esperam em algum lugar no computador até que seja dado acesso às mesmas por mecanismos potencialmente complexos.

Há três elementos básicos em um sistema de filas:

1 - As entidades que fazem a espera para utilização dos recursos. Estas entidades são usualmente denominadas **clientes**.

2 - Os recursos para os quais a espera é realizada. Desde que os recursos provejam alguma forma de serviço aos clientes, devemos genericamente os chamá-los de **servidores**.

3 - O espaço onde a espera é realizada. A esse elemento dá-se o nome **fila**.

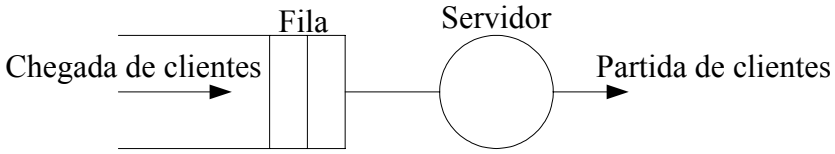


Figura 2.2: Um simples sistema de fila.

A cada chegada, o cliente, ou se dirige ao SERVIDOR e é servido, ou tem que esperar primeiro na FILA até que o servidor esteja disponível. Após ser atendido, cada cliente parte. Exemplos de clientes são: pessoas (por exemplo, esperando em um banco ou em um ponto de ônibus), mensagens transmitidas de algum meio de comunicação, tarefas, trabalhos ou transações executadas em um sistema de computador, produção em um processo de fabricação e carros que usam uma rede de estradas. Exemplos de servidores são: pessoas (por exemplo, caixas de banco ou caixas de saída de supermercado), canais de comunicação responsáveis pela transmissão de mensagens, processadores de computador ou dispositivos periféricos, vários tipos de máquinas usadas na fabricação e semáforos que regulam o fluxo de carros. Exemplos de filas são encontrados em vários locais, como por exemplo banco, pontos de ônibus ou supermercados. Porém, filas também estão presentes em redes de comunicação ou sistemas de computador onde também são alocadas formas menos tangíveis de clientes, como telefonemas ou tarefas a serem executadas em áreas de espera.

Graficamente, um simples sistema de fila será representado como mostrado na figura 2.2. O círculo representa um servidor, e uma caixa aberta representa uma fila que precede a este servidor. Aberturas de fila indicam os clientes em modo de espera. Os clientes são vistos como chegando à fila e partindo do servidor. Supõe-se ainda que o processo de servir os clientes, normalmente leva uma quantidade estritamente positiva de tempo (caso contrário não haveria espera). Assim, um servidor pode ser visto como um "bloco de atraso" que retém um cliente por algum tempo até a realização do serviço.

Visto como um SED, o sistema de fila da figura 2.2 tem um conjunto de eventos $E = \{a, d\}$, onde $\{a\}$ denota um evento de chegada e $\{d\}$ denota um evento de saída. Uma variável de estado é o número de clientes na fila ou o comprimento da fila. Assim, o espaço de estado é o conjunto de valores não negativos $X = \{0,1,2,...\}$.

2. Sistemas de manufatura

Em um processo industrial, os clientes são as peças ou partes de peças da produção. Essas peças estão dispostas para o acesso aos vários servidores da fábrica que são as máquinas que executam operações específicas e dispositivos de manipulação de material, como robôs e correias transportadoras. Quando as peças não estão sendo trabalhadas, elas são armazenadas em uma fila até que o servidor libere o acesso para a próxima operação que está disponível. Por causa de reais limitações físicas, filas em um sistema industrial têm normalmente capacidades finitas.

Uma vez mais, modelos de filas provêem uma conveniente descrição para sistemas industriais. Um exemplo simples é mostrado na figura 2.3, onde as peças passam por duas máquinas, sendo a capacidade da primeira fila infinita, enquanto a capacidade da fila da segunda máquina limitada a dois. Como resultado, é possível que uma parte de serviço da máquina 1 seja completado porém a máquina 2 esteja ocupada e além disso a fila esteja completa. Neste caso, a peça tem que permanecer na máquina 1 embora não requeira mais nenhum serviço; além disso, são forçadas outras peças a esperar o acesso na máquina 1 permanecendo em fila. O conjunto de eventos fixado para este exemplo é $E = \{a, c_1, d_2\}$, onde a é uma chegada para a primeira máquina, c_1 é uma conclusão de serviço da primeira máquina e d_2 é uma partida para a fila da segunda máquina.

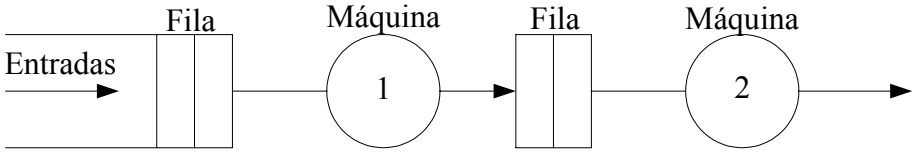


Figure 2.3: Sistema industrial de filas.

Observe que o evento c_1 não implica em movimento de uma peça da máquina 1 para a fila da máquina 2, desde que esta possibilidade esteja bloqueada. O estado do sistema pode ser definido como um vetor $x = [x_1, x_2]^T$ correspondendo aos comprimentos de fila das duas máquinas. Neste caso, x_2 é restrito aos valores $\{0,1,2,3\}$. Porém, note que quando $x_2 = 3$, a máquina 1 é bloqueada, pois acabou de executar o serviço na peça e a fila da segunda máquina está completa. Para modelar o fenômeno de bloqueio necessitamos introduz uma variável adicional B que x_2 pode gerar. O espaço de estado se torna o conjunto discreto $X = \{(x_1, x_2) : x_1 \geq 0, x_2 \in \{0, 1, 2, 3, B\}\}$. Para ilustrarmos a flexibilidade do processo modelado (dependendo do nível de detalhe que

se deseja capturar) pode-se gerar um espaço de estado alternativo que pode ser: $X = \{x_1, x_2) : x_1 \in \{I, W, B\}$ e $x_2 \in \{I, W\}\}$ onde x_1 é o estado da primeira máquina, que pode assumir os seguintes valores: inativo (I), trabalhando (W) ou bloqueado (B), e x_2 é o estado da segunda máquina, que pode assumir os seguintes valores: inativo (I) ou trabalhando (W). Neste modelo, não são focalizados os comprimentos das filas, mas sim os estados lógicos de cada máquina.

3. Sistemas de tráfego

Considere, agora, como exemplo uma simples interseção em T (figura 2.4). Há quatro tipos movimentos de veículos: (a) veículos vindo de ponto 1 e virando para o ponto 2; (b) veículos vindo de 1 e virando para o ponto 3; (c) veículos que vão diretamente do ponto 2 ao 3, e (d) veículos que vão do ponto 3 ao 2. O semáforo funciona da seguinte forma: fica vermelho para os veículos vindo da posição 1 e verde para os veículos vindo das posições 2 e 3, permitindo assim os movimentos "c"e "d", ou ao contrário, vermelho para os veículos vindo das posições 2 e 3 e verde para os veículos vindo da posição 1, permitindo os movimentos "a"e "b".

Neste caso, o conjunto de eventos é determinado por:

$$E = \{a_{12}, a_{13}, a_{23}, a_{32}, d_{12}, d_{13}, d_{23}, d_{32}, g, r\},$$

onde a_{12}, a_{13}, a_{23}, a_{32} são as chegadas de veículo em cada uma das quatro possibilidades e d_{12}, d_{13}, d_{23} e d_{32} são as partidas de veículo quando o semáforo permite o tráfego, g e r indicam o estado do semáforo.

Um possível espaço de estado é definido pelos comprimentos de fila formados pelos quatro tipos de veículo e o estado do próprio semáforo, isto é :

$$X = \{(x_{12}, x_{13}, x_{23}, x_{32}, y) : x_{12}, x_{13}, x_{23}, x_{32} \geq 0, y \in \{g_1, g_2, g_3, r_1, r_2, r_3\},$$

onde x_{12}, x_{13}, x_{23}, x_{32} são os quatro comprimentos de fila, e y é o estado da luz (g_i e r_i denotam, respectivamente, verde e vermelho para os veículos que vem dos pontos indicados).

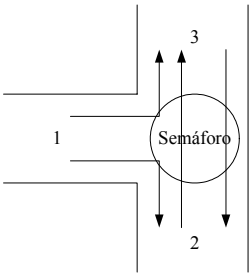


Figura 2.4: Uma simples interseção T controlada por semáforos.

2.2. Linguagens e autômato

Nesta seção será apresentado o primeiro formalismo de modelagem utilizado para representar um SED, que é o autômato, uma vez que os sistemas a eventos discretos (SED) não são adequadamente modelados através de equações diferenciais, como são modelados os Sistemas Dinâmicos de Variáveis Contínuas (SDVC).

O autômato forma a classe básica de modelos de SED. São intuitivos, fáceis de usar, com facilidade para operações de composição e para análise (no caso de estado finito), mas carecem de uma estrutura e, por esta razão, podem levar a grandes espaços de estados quando do modelamento de sistemas complexos. O segundo formalismo de modelagem a ser considerado neste trabalho é redes de Petri. Como será visto, as redes de Petri têm mais estruturas que os modelos de autômatos, embora não possuam, no geral, o mesmo poder analítico que os autômatos. Deve ser ressaltado que as redes de Petri é que serão utilizadas para gerar a modelagem dos experimentos do laboratório propostos neste trabalho.

2.2.1. Linguagens

Um SED possui um conjunto de eventos *E* associado a ele que pode ser visto como sendo o “alfabeto” de uma linguagem. As seqüências de eventos associados a este conjunto são definidas como "palavras" desta linguagem. Para entender o seu sentido considere o seguinte exemplo: suponha que após um carro ser ligado, as seguintes tarefas básicas devam ser realizadas: (a) quando o carro é ligado (ON), primeiramente deve ser enviado um sinal informando que foi ligado e está no estado ON, e então; (b) realizar um simples relatório informando as seguintes condições: 1-“tudo OK”, 2-“checar óleo”, ou 3-“necessito de gasolina”, e, (c) concluir com outro sinal informando que o “relatório de condições foi feito". Cada um destes sinais define um evento e todos os possíveis sinais que o carro puder emitir definem um alfabeto. Assim, esse sistema tem as mesmas características de um SED dirigido por esses eventos, sendo responsável por reconhecer eventos e dar a própria interpretação para qualquer seqüência particular recebida. Por exemplo, a seqüência de eventos: “estou ON”, “tudo está OK”, “relatório de condições feito”, completam com sucesso a tarefa. Por outro lado, a seqüência de eventos: “estou ON” e “relatório de condições feito”, sem que seja relatada a condição real entre os eventos, deve ser interpretado como uma condição anormal requerendo atenção especial. Pode-se, portanto, pensar nas combinações de sinais emitidos pelo

carro, como palavras pertencendo à linguagem particular falada por este carro. Neste exemplo, a linguagem de interesse tem somente três palavras ou eventos.

Uma linguagem é um caminho formal para descrever o comportamento de um SED. A linguagem especifica todas as seqüências admissíveis de eventos que o SED é capaz de "processar" ou "gerar", não necessitando de qualquer estrutura adicional.

Definição 2.2. (Notação de Linguagem) O conjunto de eventos *E* de um SED é visto como um alfabeto, e será suposto finito. Uma seqüência de eventos tirados desse alfabeto forma uma "palavra" ou "seqüência". Uma seqüência que consiste de nenhum evento é chamado de *seqüência vazia* e é denotada por ε (o símbolo ε não deve ser confundido com o símbolo genérico *e* para um elemento de *E)*. O comprimento de uma sequência é o número de eventos contidos nesta seqüência, contando ocorrências múltiplas do mesmo evento. Se *s* é uma seqüência, seu comprimento é denotado por $|s|$. Por convenção, o comprimento da seqüência vazia ε é zero, isto é, $|\varepsilon|=0$. □

Definição 2.3. (Linguagem) Uma linguagem definida em um conjunto de eventos *E* é um conjunto de seqüências de comprimentos finitos formados de eventos de *E.* □

Como exemplo, seja $E = \{a, b, g\}$ um conjunto de eventos. Podem-se definir as seguintes linguagens: $L_1 = \{\varepsilon, a, abb\}$, consistindo somente de três seqüências, ou uma linguagem; L_2 = {todos as possíveis seqüências de comprimento 3 que começam com o evento *a*}, ou seja, $L_2 = \{aaa, aab, aag, aba, abb, abg, aga, agb, agg\}$, que contém nove seqüências; ou linguagem L_3 = {todos as seqüências de possíveis comprimentos finitos que começam com evento *a*}, que contém um número infinito de seqüências etc.

2.2.2. Operações em linguagens

Seja *E** o conjunto de todas as seqüências finitas de elementos de *E*, incluindo ε. O conjunto *E** é denominado fechamento de Kleene de *E*. Por exemplo, se $E=\{a, b, c\}$ então $E^*=\{\varepsilon, a, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...\}$. Note que *E** é infinito porém contável e tem seqüências de comprimento arbitrariamente longos.

As operações fixas habituais são: união, interseção, diferença e complemento com respeito a *E*,* e são aplicáveis para linguagens, desde que sejam conjuntos. Além disso, são usadas as seguintes operações:

1 – *Concatenação*. S*eja* $L_a, L_b \subseteq E^*$. Então:

$$L_aL_b := \{s \in E^* : (s = s_as_b) \text{ e } (s_a \in L_a) \text{ e } (s_b \in L_b)\} .$$

Uma seqüência está em L_aL_b se ela puder ser escrita como a concatenação de uma

seqüência em L_a com uma seqüência em L_b.

2 – *Fechamento de Prefixos*. O fechamento de prefixos de L *é* a linguagem denotada por $\overline{L}$ e consiste de todo os prefixos de todas as seqüências em L. No geral, $L \subseteq \overline{L}$. Seja $L \subseteq E^*$, então:

$\overline{L} := \{s \in E^* : \exists t \in E^* (st \in L)\}$.

Uma linguagem L é dita ser de prefixo fechado se $L = \overline{L}$. Assim a linguagem L é de prefixo fechado se qualquer prefixo de qualquer seqüência em L for também um elemento de L.

3 – *Fechamento de Kleene:* seja $L \subseteq E^*$, então

$L^* := \{\varepsilon\} \cup L \cup LL \cup LLL \cup ...$

Para melhor entender os conceitos acima, considere o seguinte exemplo: Seja $E = \{a,b,g\}$, e considere as linguagens $L_1 = \{\varepsilon, a, abb\}$ e $L_4 = \{g\}$. Note que L_1 e L_4 não são de prefixo fechado uma vez que $ab \notin L_1$ e $\varepsilon \notin L_4$. Então:

L_1L_4	=	{g, ag, abbg}
$\overline{L_1}$	=	{ε, a, ab, abb}
$\overline{L_4}$	=	{ε, g}
$L_1\overline{L_4}$	=	{ε, a abb, g, ag, abbg}
L_4^*	=	$\{\varepsilon , g, gg, ggg, ...\}$
L_1^*	=	$\{\varepsilon , a, abb, aa, aabb, abba, abbabb, ...\}$

Observação 2.1: É importante observar que:

(i) $\varepsilon \notin \emptyset$;

(ii) $\{\varepsilon\}$ é uma linguagem não-vazia, contendo unicamente uma seqüência vazia;

(iii) Se $L = \emptyset$ então $\overline{L} = \emptyset$, e se $L \neq \emptyset$ então, necessariamente, $\varepsilon \in \overline{L}$;

(iv) $\emptyset^* = \{\varepsilon\}$ e $\{\varepsilon\}^* = \{\varepsilon\}$.

2.2.3. Autômato

A dificuldade de se trabalhar com linguagens simplesmente é que representações "simples" de linguagem não são, em geral, fáceis de especificar ou de trabalhar. É necessário, pois, um conjunto de "estruturas" compactas que definam linguagens e que possam ser manipuladas através de operações claras de modo que possam construir e, subseqüentemente, manipular e analisar linguagens arbitrariamente complexas.

Um autômato é um dispositivo que é capaz de representar uma linguagem de acordo com regras claras. O modo mais simples para apresentar a noção de autômato é considerar sua representação de gráfico direcionada, ou diagrama de transição de estado. Considere, portanto, o seguinte exemplo: considere o grafo da figura 2.5, onde os nós representam *estados* e os arcos rotulados representam *transições* entre esses estados. Este gráfico fornece uma descrição completa de um autômato. O conjunto de nós é o conjunto de estados do autômato, $X = \{x, y, z\}$. O conjunto de rótulos para as transições é o conjunto de evento (alfabeto) do autômato, $E = \{a, b, g\}$. Os arcos no gráfico fornecem uma representação gráfica da *transição funcional* do autômato, sendo denotado como $f : X \times E \rightarrow X$: $f(x, a) = x$, $f(x, g) = z$, $f(y, a) = x$, $f(y, b) = y$, $f(z, b) = z$ e $f(z, a) = f(z, g) = y$. A notação $f(y, a) = x$ representa os meios de representar que o autômato está no estado y, e com a "ocorrência" de um evento a, o autômato fará uma transição rápida para o estado x. A causa da ocorrência do evento a é irrelevante; o evento pode ser uma entrada externa para o sistema modelado pelo autômato, ou pode ser um evento espontaneamente "gerado" pelo sistema modelado.

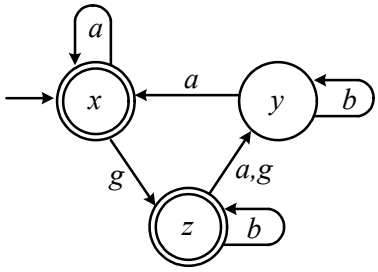


Figura 2.5: Diagrama de transição de estado.

Três observações são válidas com respeito ao exemplo acima: primeira, um evento pode ocorrer sem mudar o estado, como em $f(x, a) = x$; segunda, dois eventos distintos podem ocorrer em um dado estado causando a mesma exata transição, como em $f(z, a) = f(z, g) = y$ (o que é interessante sobre o último fato é que é possível não distinguir os eventos a e g simplesmente observando-se uma transição do estado z para o estado y); terceira, a função f é uma função parcial com domínio em $X \times E$, isto é, não precisa ser uma transição definida para cada evento em E e cada estado de X (por exemplo, $f(x, b)$ e $f(y, g)$ não são definidas).

Para se definir completamente um autômato, é necessário ainda um estado inicial, denotado por x_0, e um subconjunto X_m de X que representa os estados de X que

são *marcados.* Os estados são marcados quando é necessário dar um significado especial para eles e são referidos como estados "finais". O estado inicial será identificado por uma flecha apontando para dentro e estados pertencentes a X_m serão identificados por círculos duplos.

Pode-se, agora, dar uma definição formal de um autômato.

Definição 2.4. (Autômato determinístico) Um *autômato Determinístico* denotado por *G,* é uma sextúpla $G = (X, E, f, \Gamma, x_0, X_m)$, onde X é o conjunto de estados, E é o conjunto finito de *eventos* associados com as transições em G, $f : X \times E \rightarrow X$ é a *função de transição* $f(x, e) = y$, que significa que existe uma transição rotulada pelo evento e do estado x para o estado y (no geral, f é uma função *parcial* em seu domínio), Γ: $X \rightarrow 2^E$ é *função de evento ativa* (ou possível função de evento), $\Gamma(x)$ é o conjunto de todos os eventos de E tais que $f(x, e)$ é definida (isto é chamado de conjunto de evento ativo ou possível conjunto de evento de G em x), x_0 é o *estado inicial* e $X_m \subseteq X$ *é* o conjunto de estados marcados. □

O autômato G opera como segue: começa no estado inicial x_0 e na ocorrência de um evento $e \in \Gamma(x_0) \subseteq E$ fará uma transição de estado $f(x_0, e) \in X$. Este processo então continua baseado nas transições para as quais f é definida.

Por conveniência, f é sempre estendida do $X \times E$ *para o* domínio $X \times E^*$ da seguinte maneira recursiva: $f(x, e) := x$ e $f(x, se) := f(f(x, s), e)$ para $s \in E^*$ e $e \in E$.

2.2.4. Representação de linguagens por autômatos

A conexão entre linguagem e autômato é feita facilmente por inspeção do diagrama de transição de estado do autômato, considerando todos os caminhos diretos que podem ser seguidos no diagrama de transição de estado, começando no estado inicial, e, dentre esses, aqueles caminhos que terminam em um estado marcado. Isto conduz às noções de linguagens geradas e marcadas por um autômato.

Definição 2.5. (Linguagens geradas e marcadas) A linguagem gerada por $G = (X, E, f, \Gamma, x_0, X_m)$ é $\mathcal{L}(G) := \{s \in E^* : f(x_0, s)$ é definida$\}$. A linguagem marcada por G é $\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\}$. □

A linguagem $\mathcal{L}(G)$ representa todos os caminhos diretos que podem ser seguidos ao longo do diagrama de transição de estado, começando no estado inicial. Portanto, uma seqüência s está em $\mathcal{L}(G)$ se e somente se essa corresponde a um caminho admissível no diagrama de transição de estados ou equivalentemente, se e somente se f é

definida em (x_0, s). A segunda linguagem representada por G, $\mathcal{L}_m(G)$, é o subconjunto de $\mathcal{L}(G)$ consistindo unicamente da seqüência s tais que $f(x_0, s) \in X_m$, quer dizer, essas seqüências correspondem aos caminhos que terminam em um estado marcado no diagrama de transição de estado.

As definições de linguagens gerada e marcada podem ser melhor compreendidas com a ajuda do seguinte exemplo: seja $E = \{a, b\}$ um conjunto de eventos e considere o autômato de estado finito $G = (X, E, f, \Gamma, x_0, X_m)$ onde $X = \{0, 1\}$, $x_0 = 0$, $X_m = \{1\}$, e f é definida como : $f(0, a) = 1$, $f(0, b) = 0$, $f(1, a) = 1$, $f(1, b) = 0$, representado na figura 2.6. Como 0 é o estado inicial, a linguagem gerada por esse autômato é o próprio E^* isto é: $\mathcal{L}(G) = \{a, b, aa, ba, bb, aaa, ...\}$. Como o único estado marcado é o 1, então esse estado só pode ser atingido pelas seqüências de $\mathcal{L}(G)$ em que o último evento é o a. Portanto $\mathcal{L}_m(G) = \{a, aa, ba, aaa, aba, baa, bba, ...\}$. Pode-se, então, concluir que um autômato G é a representação de duas linguagens $\mathcal{L}(G)$ e $\mathcal{L}_m(G)$.

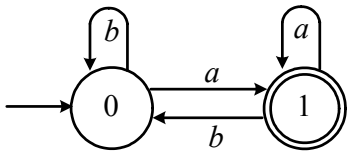


Figura 2.6: Autômato.

2.2.5. Bloqueio

A partir das definições de G, $\mathcal{L}(G)$, e $\mathcal{L}_m(G)$ tem-se que $\mathcal{L}_m(G) \subseteq \overline{\mathcal{L}_m(G)} \subseteq \mathcal{L}(G)$. A primeira inclusão de conjunto é devida ao fato de X_m ser um subconjunto do próprio X, enquanto a segunda inclusão de conjunto é uma conseqüência da definição de $\mathcal{L}_m(G)$ e do fato de que $\mathcal{L}(G)$ ser, por definição, de prefixo fechado.

Um autômato G pode alcançar um estado x, tal que $\Gamma(x) = \emptyset$ mas $x \notin X_m$ isto é, f não é definida para x. Isto é chamado de trancamento definitivo (deadlock) tendo em vista que nenhum evento adicional pode ser executado. Se um trancamento definitivo acontecer, então necessariamente $\overline{\mathcal{L}_m(G)}$ será um subconjunto próprio de $\mathcal{L}(G)$, uma vez que qualquer seqüência de $\mathcal{L}(G)$ que termina no estado x não pode ser um prefixo de uma seqüência em $\mathcal{L}_m(G)$.

Outro tipo de trancamento é quando o sistema entra em um determinado conjunto de estados não marcados e não consegue sair deles. A esse tipo de

trancamento, dar-se o nome de trancamento cíclico (livelock). Também nesse caso, é fácil observar que $\overline{\mathcal{L}_m(G)}$ é um subconjunto próprio de $\mathcal{L}(G)$. Pode-se, então, apresentar a seguinte definição:

Definição 2.6. Um autômato G é dito estar bloqueando se $\overline{\mathcal{L}_m(\mathrm{G})} \subset \mathcal{L}(G)$ onde a inclusão de conjunto é própria, e não bloqueia quando $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$. □

Para entender esses conceitos, considere o autômato G descrito na figura 2.7. Claramente, o estado 5 é um estado de trancamento definitivo. Além disso, os estados 3 e 4, com suas transições associadas *a, b,* e *g*, formam um componente absorvente fortemente conectado; uma vez que 3 e 4 não são marcados, qualquer seqüência que alcança o estado 3 conduzirá a um trancamento cíclico. Note que a seqüência $ag \in \mathcal{L}(G)$ mas $ag \notin \overline{\mathcal{L}_m(G)}$; o mesmo é verdade para qualquer seqüência em $\mathcal{L}(G)$ que começa com *aa*. Assim G está bloqueando uma vez que $\overline{\mathcal{L}_m(G)}$ é um subconjunto próprio de $\mathcal{L}(G)$.

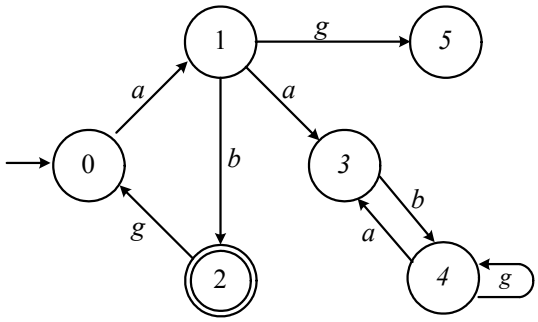


Figura 2.7: Autômato bloqueando.

2.3. Redes de Petri

Uma alternativa para substituir os modelos de autômatos de SED é fornecida pelas redes de Petri. Uma rede de Petri é um dispositivo que manipula eventos de acordo com regras estabelecidas. Uma de suas características principais é a existência de condições explícitas sob as quais um evento pode ou não ser habilitado, permitindo representações de SED muito gerais, cuja operação depende de esquemas de controle potencialmente complexos.

2.3.1. Fundamentos de redes de Petri

A definição de uma rede de Petri é feita em dois passos: primeiro, define-se o *grafo da rede de Petri,* também chamado *estrutura da rede de Petri,* que é análogo ao

diagrama de transição de estado de um autômato; em seguida, junta-se a esse grafo um estado inicial, um conjunto de estados marcados, e uma função de transição rotulada, resultando no modelo completo da rede de Petri.

Em uma rede de Petri, os eventos estão associados às "transições". Para que uma transição aconteça, várias condições devem ser satisfeitas. Estas condições são localizadas nos próprios estados ou "lugares" com suas informações relacionadas a cada transição. Os lugares podem ser definidos como "entradas" ou "saídas" para uma transição. Transições, lugares, e as relações entre esses definem os componentes básicos de um *grafo da rede de Petri.* O grafo de uma rede de Petri tem dois tipos de nós (lugares e transições) e setas conectando estes nós. Por esta razão a rede de Petri é um *grafo bipartido,* no sentido que as setas não podem conectar diretamente nós do mesmo tipo, isto é, as setas conectam nós de lugares para nós de transição e nós de transição para nós de lugares. Pode-se, então apresentar a seguinte definição.

Definição 2.7. (Grafo da rede de Petri) Um *grafo* ou *estrutura da rede de Petri* é um grafo ponderado bipartido (P, T, A, w), onde P é o conjunto finito de *lugares*, T é o conjunto finito de *transições*, $A \subseteq (P \times T) \cup (T \times P)$ *é* o conjunto de setas de lugares para transições e de transições para lugares no gráfico e $w : A \rightarrow \{1,2,3,...\}$ é a função dos pesos das setas (um número inteiro positivo). □

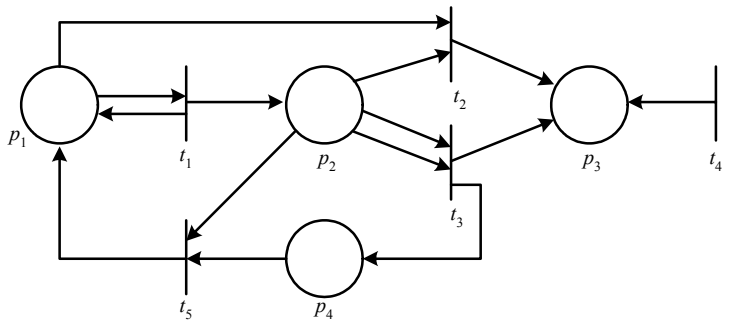


Figura 2.8: Gráfico de rede do Petri.

Para tornar mais clara a definição 2.7, considere o grafo de uma rede de Petri mostrado na figura 2.8. Tem-se que o conjunto de lugares é $P = \{p_1, p_2, p_3, p_4\}$, o conjunto de transições é $T = \{t_1, t_2, t_3, t_4, t_5\}$, $A=\{(p_1, t_1), (p_1, t_2), (p_2, t_2), (p_2, t_3), (p_2, t_5), (p_4, t_5), (t_1, p_1), (t_1, p_2), (t_2, p_3), (t_3, p_3), (t_3, p_4), (t_4, p_3), (t_5, p_1)\}$, $w(p_1, t_1)=1$, $w(p_1, t_2)=1$, $w(p_2, t_2)=1$, $w(p_2, t_3)=2$, $w(p_2, t_5)=1$, $w(p_4, t_5)=1$, $w(t_1, p_1)=1$, $w(t_1, p_2)=1$, $w(t_2, p_3)=1$, $w(t_3, p_3)=1$, $w(t_3, p_4)=1$, $w(t_4, p_3)=1$ e $w(t_5, p_1)=1$.

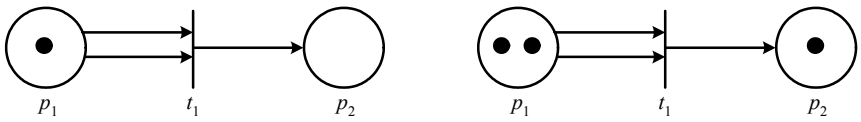
Note que como a transição t_4 não tem nenhum lugar de entrada, então o evento

correspondente a t_4 acontece de forma incondicional. Em contraste, o evento correspondente à transição t_2, depende de certas condições relacionadas aos lugares p_1 e p_2 para que possa ocorrer.

Voltando à idéia de que as transições em um grafo de redes de Petri representam os eventos que fazem a evolução de um SED e que os lugares descrevem as condições sob as quais esses eventos podem ocorrer, tornam-se, então, necessários mecanismos que identifiquem se essas condições foram, de fato, satisfeitas ou não. Isto é feito atribuindo-se "discos" aos lugares para indicar que a condição descrita por aquele lugar é satisfeita. Esses discos definem uma marca. Formalmente, uma *marcação*, *x,* de um grafo da rede de Petri *(P, T, A, w)* é uma função $x : P \rightarrow \mathbb{N} = \{0, 1, 2,...\}$. Assim, a marcação de *x* define um vetor linha $\underline{x} = [x(p_1), x(p_2),..., x(p_n)]$ onde *n* é o número de lugares na rede de Petri e a *i-ésima* entrada deste vetor indica o número de discos no lugar p_i, $x(p_i) \in \mathbb{N}$. Em um grafo da rede de Petri, um disco é indicado por um ponto escuro posicionado no interior do círculo que define o lugar.

Definição 2.8. (Rede de Petri marcada) Uma *rede de Petri marcada é* uma Quíntupla (*P, T, A, w, x*), onde (*P, T, A, w*) *é* um grafo da rede de Petri e *x* é a marcação de um conjunto de lugares *P,* isto é, $\underline{x} = [x(p_1), x(p_2),...,x(p_n)] \in \mathbb{N}^n$ é o vetor linha associado a *x*. □

Para ilustrar o conceito acima, considere a rede de Petri representada pelo grafo da figura 2.9. Nessa figura estão mostradas duas possíveis marcações, correspondentes aos vetores linha $\underline{x}_1 = [1, 0]$ e $\underline{x}_2 = [2, 1]$.



$\underline{x}_1=[1, 0]$ $\underline{x}_2=[2, 1]$

Figura 2.9: Duas marcações, $\underline{x}_1$ e $\underline{x}_2$, ao gráfico de rede de Petri.

Observação 2.2. (a) Por simplicidade, uma rede de Petri marcada será, de agora em diante, referida simplesmente como Rede de Petri; (b) O número de discos atribuídos a um lugar é um número inteiro arbitrário e não negativo. Segue-se, então, que o número de estados que se pode ter é, em geral, infinito. Assim, o espaço de estado *X,* de uma rede de Petri, com *n* lugares é definido por todos os vetores n-dimensionais, isto é, $X=\mathbb{N}^n$.

Enquanto o termo “marcação” é mais comum que “estado” na literatura sobre rede de Petri, o termo estado é consistente com o papel de estado em sistemas dinâmicos. Além disso, o termo estado evita uma potencial confusão entre marcação em grafos da rede de Petri e marcação no sentido de *estados marcados* em autômato.

2.3.2. Evolução dinâmica das redes de Petri

A definição 2.8 não descreve explicitamente os mecanismos de transição das redes de Petri, embora este seja um ponto crucial na utilização das redes de Petri para modelar SED dinâmico. Para Tanto a seguinte definição é necessária.

Definição 2.9. (Transição habilitada) Uma transição $t_j \in T$ em uma rede de Petri é dita estar *habilitada* se $x(p_i) \geq w(p_i, t_j)$ para todo $p_i \in I(t_j)$, onde $I(t_j)$ denota o conjunto de lugares conectados à transição t_j por meio de setas. □

De acordo com a definição 2.9 a transição t_j na rede de Petri estará habilitada quando o número de discos no lugar p_i é maior ou igual ao peso da seta que conecta p_i a t_j, para todo lugar p_i que são entradas para a transição t_j. Note na figura 2.9 que para o estado $\underline{x}_1$, $x(p_1) = 1 < w(p_1, t_1) = 2$, e portanto t_1 não está habilitada. Por outro lado para o estado $\underline{x}_2$, tem-se $x(p_1)=2= w(p_1, t_1)$ e então, t_1 está habilitada.

Nos autômatos, o mecanismo de transição de estado é diretamente capturado pelos arcos conectando os nós (estados) no diagrama de transição de estado, equivalentemente pela função de transição *f*. O mecanismo de transição de estado em redes de Petri é dado pelo movimento dos discos através da rede, conseqüentemente, pela mudança do estado da rede de Petri. Quando uma transição está habilitada, diz-se que ela pode *disparar*. A função de transição de estado de uma rede de Petri é definida através da mudança no estado da rede de Petri devido ao disparo de uma transição habilitada. Algumas vezes podem haver diversas transições habilitadas e poder-se-ia pensar em disparos simultâneos. No presente trabalho, será suposto que os disparos acontecem um de cada vez.

Definição 2.10. (Dinâmicas da rede de Petri) A *função de transição de estado, f :* $\mathbb{N}^n \times T \rightarrow \mathbb{N}^n$, da rede de Petri (P, T, A, w, x) é definida pela transição $t_j \in T$ se e somente se

$$x(p_i) \geq w(p_i, t_j) \text{ para todo } p_i \in I(t_j) \qquad (2.1)$$

Se $f(\underline{x}, t_j)$ é definida, então o próximo vetor de estados $\underline{x}' = f(\underline{x}, t_j)$ é definido como

$$x'(p_i) = x(p_i) - w(p_i, t_j) + w(t_j, p_i), \quad i = 1,...,n. \qquad (2.2)$$ □

A condição (2.1) assegura que a função de transição de estado seja definida unicamente por transições que são habilitadas. Assim, o próximo estado, definido pela condição (2.2) depende explicitamente dos lugares de entrada e de saída de uma transição e dos pesos das setas que conectam esses lugares à transição. É importante observar que o número de discos não precisa necessariamente ser conservado após o disparado de uma transição em uma rede de Petri. Em geral, é inteiramente possível que depois de vários disparos de transição, o estado resultante seja x = [0,...,0], ou que o número de símbolos em um ou mais lugares cresça arbitrariamente depois de um número arbitrariamente grande de disparos de transição.

Para ilustrar o processo de disparo de transições e mudanças de estado de uma rede de Petri, considere a rede de Petri da figura 2.10(a), onde o estado "inicial" é $\underline{x}_0$ = [2, 0, 0, 1]. É fácil verificar que a única transição habilitada é t_1, uma vez que ela requer um único símbolo do lugar p_1 e tem-se $x_0(p_1) = 2$. Em outras palavras, $x_0(p_1) \geq w(p_1, t_1)$, e a condição (2.1) é satisfeita para a transição t_1. Quando t_1 dispara, um símbolo é removido de p_1, e um símbolo é colocado em cada lugar de p_2 e p_3, como pode ser visto no gráfico da rede de Petri. Esse mesmo resultado poderia ser obtido também aplicando-se diretamente a equação (2.2) para obter o novo estado $\underline{x}_1$= [1, 1, 1, 1], como mostrado na figura 2.10(b). Neste estado, todas as três transições t_1, t_2 e t_3 estão habilitadas.

Considere, agora, o disparo da transição t_2. Um disco é removido de cada um dos lugares de entrada, p_2 e p_3. Como os lugares de saída são p_2 e p_4, então, um disco retorna ao lugar p_2, uma vez que $p_2 \in I(t_2) \cap O(t_2)$; além disso, um disco é adicionado a p_4. O novo estado é $\underline{x}_2$ = [1, 1, 0, 2], como mostrado na figura 2.10(c). Neste estado, t_2 e t_3 já não estão habilitados, mas t_1 ainda está.

Voltando ao estado $\underline{x}_1$ da figura 2.10(b) e ao invés de disparar t_2, suponha se dispare t_3. É fácil verificar que de cada um dos lugares de entrada, p_1, p_3, e p_4, mover-se-a um disco. Como não há lugares de saída, o novo estado denotado por $\underline{x}'_2$ será dado por $\underline{x}'_2$ = [0, 1, 0, 0], como mostrado na figura 2.10(d). Vê-se que nenhuma transição está habilitada e, assim, nenhuma mudança de estado adicional é possível, isto é, o estado [0, 1, 0, 0] é um estado de “trancamento definitivo” desta rede de Petri.

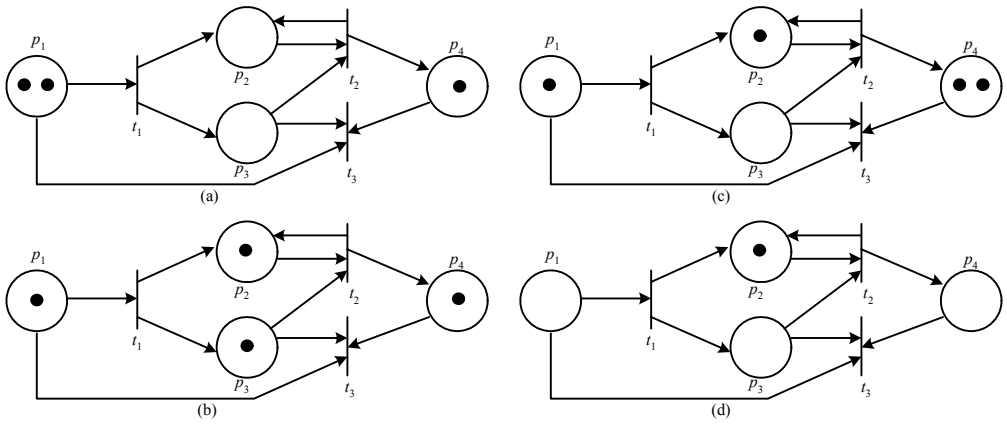


Figura 2.10: Seqüências de disparos de transições em uma rede de Petri.

Uma observação importante sobre o comportamento dinâmico de redes de Petri é que nem todos os estados em $\mathbb{N}^n$ podem necessariamente ser alcançados em um gráfico da rede de Petri com um dado estado inicial. Por Exemplo, ao examinar o grafo da figura 2.9, tem-se que para o estado inicial $\underline{x}_2 = [2, 1]$, o único estado que pode ser alcançado de $\underline{x}_2$ é [0, 2]. Isto leva à definição de conjunto de estados alcançáveis, $R[(P, T, A, w, x)]$, da rede de Petri (P, T, A, w, x). Neste sentido, primeiramente, é necessário estender a função de transição de estado f do domínio $\mathbb{N}^n \times T$ ao domínio $\mathbf{N}^n \times T^*$, isto é:

$$\begin{cases} f(\underline{x}, \varepsilon) := \underline{x} \\ f(\underline{x}, st) := f(f(\underline{x}, s), t) \text{ para } s \in T^* \text{ e } t \in T \end{cases}$$

onde o símbolo ε é interpretado como a ausência de disparo de transição.

Definição 2.11. (Estados Alcançáveis) O conjunto de estados *alcançáveis* da rede de Petri (P, T, A, w, x) é

$$R[(P, T, A, w, x)] := \{\boldsymbol{y} \in \mathbb{N}^n : \exists s \in T^*(f(\mathbf{x}, s) = \boldsymbol{y})\}$$ □

2.3.3. Equações de estado

Considere novamente a equação (2.2), que descreve como o valor de estado de um lugar individual muda quando com o disparo de uma transição. Não é difícil ver que é possível gerar um sistema de equações a partir da equação (2.2) para obter o próximo estado da rede de Petri $\underline{x}' = [x'(p_1), x'(p_2),..., x'(p_n)]$ a partir do estado atual $\underline{x} = [x(p_1), x(p_2),..., x(p_n)]$ dado que uma transição particular, t_j, tenha disparado. Para tanto, deve-se, primeiro, definir o *vetor disparo* $\underline{\mathbf{u}}$, isto é um vetor linha de dimensão m da forma $\underline{\mathbf{u}}=[0,..., 0, 1, 0,..., 0]$, onde um único 1 aparece na j-ésima posição, $j \in \{1, ..., m\}$, para indicar o fato que a j-ésima transição está, neste momento, disparando. Além disso,

defina a *matriz de incidência* **A** de uma rede de Petri, uma matriz *m* x *n* cujo elemento *(j, i)* é da forma

$$a_{ji} = w(t_j, p_i) - w(p_i, t_j) \tag{2.3}$$

Usando a matriz de incidência **A**, pode-se agora obter a seguinte equação de estados

$$\underline{x}' = \underline{x} + \underline{u}A \tag{2.4}$$

que descreve o processo de transição de estado como resultado de uma "entrada" **u,** isto é, uma transição particular disparando. O *i*-ésimo elemento da equação (2.4) é precisamente a equação (2.2). Portanto, $f(\underline{x}, t_j) = \underline{x} + \underline{u}A$**,** onde $f(\underline{x}, t_j)$ é a função de transição definida anteriormente. O argumento t_j nesta função indica que a j-ésima entrada em $\underline{u}$ é, não zero. A equação de estado fornece uma ferramenta algébrica conveniente e uma alternativa à análise gráfica para descrever o processo de transições após disparos e mudanças de estado de uma rede de Petri.

Para ilustrar a evolução de um SED a partir das equações de estado, considere a rede de Petri da figura 2.10(a), com o estado inicial $\underline{x}_0 = [2, 0, 0, 1]$. Pode-se, primeiramente, escrever a matriz de incidência por inspeção do gráfico da rede de Petri, que neste caso é:

$$A = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

A entrada (1, 2), por exemplo, é dada por $w(t_1, p_2) - w(p_2, t_1) = 1 - 0$. Usando a equação (2.4), a equação de estado quando a transição t_1 dispara no estado $\underline{x}_0$ é

$$\underline{x}_1 = [2\ 0\ 0\ 1] + [1\ 0\ 0] \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

$$\underline{x}_1 = [2\ 0\ 0\ 1] + [-1\ 1\ 1\ 0] = [1\ 1\ 1\ 1]$$

que é precisamente o foi obtido no exemplo de figura 2.10(b). Similarmente, os outros estados também podem ser obtidos.

2.3.4 - Linguagens de rede de Petri

Seja *E* o conjunto de eventos de um SED cuja linguagem é modelada por uma rede de Petri. O modelo por redes de Petri desse sistema deve ser tal que a cada transição em *T* corresponda um evento distinto no conjunto de eventos *E,* e vice-versa.

Contudo, isto poderia ser desnecessariamente restritivo, uma vez que em modelos de autômatos é permitido haver duas setas diferentes (originando de dois estados diferentes) rotulados com o mesmo evento. Isto conduz à definição de uma *rede de Petri rotulada.*

Definição 2.12. (Rede de Petri rotulada) Uma *rede de Petri rotulada N* é uma Óctupla $N = (P, T, A, w, E, \ell, \underline{x}_0, \mathbf{X}_m)$, onde (*P, T, A, w*) é um gráfico de rede de Petri, *E* é o conjunto de eventos por transição rotulada, $\ell: T \rightarrow E$ é a função de transição rotulada, $\underline{x}_0 \in \mathbb{N}^n$ é o estado inicial da rede (i.e., o número inicial de símbolos em cada lugar) e $\mathbf{X}_m \subseteq \mathbb{N}^n$ é o conjunto de *estados marcados da* rede. □

A seguir é introduzido o conceito de estados marcados para definir uma linguagem *marcada* de uma rede de Petri rotulada.

Definição 2.13. (Linguagens geradas e marcadas) A linguagem gerada por uma rede de Petri rotulada $N = (P, T, A, w, E, \ell, \underline{x}_0, \mathbf{X}_m)$ é

$$\mathcal{L}(N) := \{\ell(s) \in E^* : s \in T \text{ e } f(\underline{x}_0, s) \text{ é definida } \}.$$

A linguagem marcada por *N* é

$$\mathcal{L}_m(N) := \{\ell(s) \in \mathcal{L}(N) \ : s \in T^* \text{ e } f(\underline{x}_0, s) \in \mathbf{X}_m \}.$$ □

Pode-se ver que essas definições estão completamente consistentes com as definições correspondentes aos autômatos. A linguagem $\mathcal{L}(N)$ representa todas as seqüências de transições rotuladas que são obtidas por todos as possíveis (finitas) seqüências de disparos de transição em *N,* começando no estado inicial $\underline{x}_0$ de *N.* A linguagem marcada $\mathcal{L}_m(N)$ *é* o subconjunto destas seqüências que deixam a rede de Petri em um estado que é um membro do conjunto de estados marcados da definição de *N.*

A classe de linguagens que podem ser representados por redes de Petri rotuladas é

$$\mathcal{PNL} := \{K \subseteq E^*: \exists N = (P, T, A, w, E, \ell, \underline{x}_0, \mathbf{X}_m)[\ \mathcal{L}m(N)=K]\}$$

Esta é uma definição geral e as propriedades de $\mathcal{PNL}$ dependem fortemente das suposições específicas que são feitas sobre ℓ (por exemplo, se é injetiva ou não) e $\mathbf{X}_m$ (por exemplo, se é finito ou infinito). Nesse trabalho serão adotadas as seguintes hipóteses: ℓ não é necessariamente injetiva e $\mathbf{X}_m$ não precisa ser finito.

2.3.5 - Modelos de redes de Petri para sistemas com filas

Considere o grafo da figura 2.11 em que três eventos ou transições dirigem um

sistema com filas, quais sejam: chegada de cliente (*a*), começo do serviço (*s*) e serviço completo e partida do cliente (*c*). A partir desses eventos é possível formar o conjunto de transição $T = \{a, s, c\}$. Note que, nesse exemplo não é necessário considerar redes de Petri rotulada; equivalentemente, pode-se supor que $E = T$ e que ℓ é um mapa um-para-um entre esses dois conjuntos. A transição *a* é espontânea e não requer condições (lugares de entrada). Por outro lado, a transição *s* conta com duas condições: a presença de clientes na fila, e que o servidor esteja inativo. Essas duas condições serão representadas por dois lugares de entradas para esta transição, lugar *Q* (fila) e lugar *I* (servidor inativo). Finalmente, a transição *c* requer que o servidor esteja ocupado, assim introduziremos um lugar de entrada *B* (servidor ocupado) para isto. Assim, o conjunto de lugares desse sistema é $P = \{Q, I, B\}$.

O gráfico da rede de Petri completo, junto com o simples modelo de sistema de fila, é mostrado nas figuras 2.11(a) e (b). Nenhum símbolo é colocado em *Q,* indicando que a fila está vazia, e um símbolo é colocado em *I*, indicando que o servidor está inativo. Isto define o estado inicial $\underline{x}_0 = [0, 1, 0]$. Uma vez que a transição de estado *a* está sempre habilitada, é possível gerar vários caminhos de amostra possíveis. Como um exemplo, a figura 2.11(c) mostra o estado [2, 0, 1] resultante do disparo da seqüência de transições $\{a, s, a, a, c, s, a\}$. Este estado corresponde a dois clientes esperando na fila, enquanto um terceiro está em serviço (a primeira chegada na seqüência já tem partido depois da transição *c*).

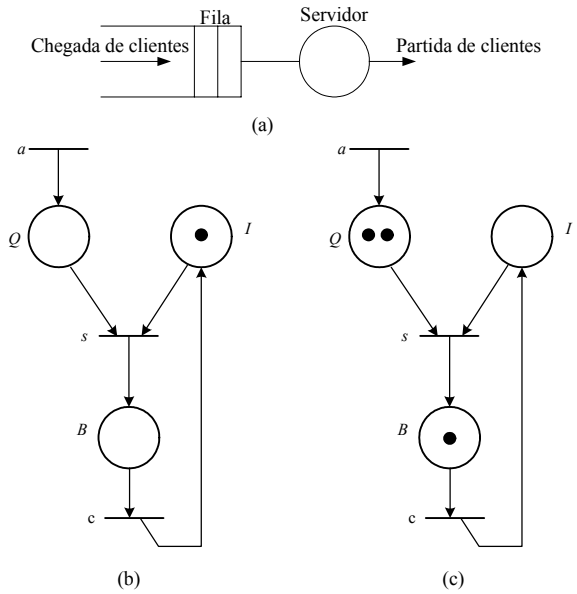


Figura 2.11: (a) Simples sistema de fila, (b) Modelo de rede de Petri para um sistema de fila simples com estado inicial [0, 1, 0]. (c) Modelo de rede de Petri de um sistema de

fila simples com estado inicial [0, 1, 0] depois de disparada a seqüência {*a, s, a, a, c, s, a*}.

2.3.6. Comparação entre redes de Petri e autômatos

Autômatos e redes de Petri podem ser utilizadas para representar o comportamento de um SED. Nos autômatos, isto é feito enumerando-se explicitamente todos estados possíveis e "conectando-se" esses estados com as possíveis transições entre eles, resultando na função de transição do autômato. Em redes de Petri os estados não são enumerados, pois a informação de estado é "distribuída" dentre um conjunto de lugares que capturam as condições chaves que governam a operação do sistema e, então, conectam esses lugares corretamente às transições.

Não é possível afirmar qual é o melhor formalismo de modelagem, pois, modelar sempre envolve questões pessoais e muitos freqüentemente dependem de aplicação considerada. Entretanto é possível comparar segundo critérios específicos, quais sejam:

(a) Expressividade de Linguagem

Como primeiro critério para comparar autômatos e redes de Petri, tem-se a classe de linguagens que pode ser representada por cada formalismo, quando restrito a modelos que requerem *memória finita* (uma óbvia consideração prática). A classe $\mathcal{PNL}$ é estritamente maior que a classe $\mathcal{R}$, significando que redes de Petri com conjuntos finitos de lugares e transições podem representar (isto é, marcar) mais linguagens em E^* que os autômatos de estado finito. Um exemplo de criação de uma rede de Petri em que não há um correspondente autômato, pode ser visto em CASSANDRAS e LAFORTUNE (2000).

(b) Modelo de Construção Modular

A despeito da complexidade potencial de gráficos da rede de Petri exigidos para modelar até mesmo um SED relativamente simples, a estrutura da rede de Petri possui algumas vantagens inerentes. Uma dessas vantagens é sua capacidade para decompor ou modular um sistema potencialmente complexo. Suponha que haja dois sistemas que possuem os espaços de estado X_1 e X_2 modelados como autômatos. Ao combinar esses dois sistemas dentro de um, seu espaço de estado, X, pode ser tão grande quanto os estados de $X_1 \times X_2$; em particular, este limite superior será alcançado se os dois sistemas

não têm eventos comuns. Isto significa combinar sistemas múltiplos, aumentando-se ligeiramente a complexidade de modelos de autômatos. Por outro lado, se os sistemas são modelados através da rede de Petri, o sistema combinado é freqüentemente mais fácil de se obter por deixar as redes originais como eles são e simplesmente somando-se uns poucos lugares e/ou transições (ou fundindo alguns lugares) representando a união efetuada entre os dois. Além disso, ao olhar tal gráfico da rede de Petri, uma pessoa pode convenientemente ver os componentes individuais, discernir o nível de sua interação, e finalmente decompor o sistema dentro de módulos distintos lógicos.

(c) Capacidade de Tomar Decisões

Outra maneira de se comparar redes de Petri com autômatos é quanto a capacidade de tomar decisão. Suponha que seja feita a seguinte pergunta: pode uma certa seqüência de eventos ser reconhecida por um determinado autômato de estado finito? A este problema dá-se o nome de "capacidade de tomar decisão", isto é, se há um algoritmo ou procedimento específico que permite dizer "sim" ou "não" como resposta. Um recurso atrativo dos autômatos de estado finito é que tais perguntas podem ser respondidas precisamente porque o espaço de estado é finito. Infelizmente, isto nem sempre é verdadeiro em procedimentos com redes de Petri, refletindo um compromisso natural entre a capacidade de tomar decisão e a riqueza do modelo.

Como conclusão é mais conveniente pensar em redes de Petri e autômatos como abordagens de modelagens complementares e não como técnicas que competem entre si. Será a aplicação considerada que definirá qual a técnica de modelagem (autômato ou rede de Petri) a ser adotada.

2.4. Modelos temporizados

Nesta seção será feita uma breve revisão da teoria de modelos temporizados de SED. O estudo será limitado a uma descrição de entrada que é completamente especificada, a fim de se ter uma compreensão das dinâmicas de um SED com eventos básicos contendo tempo, independente da caracterização probabilística de sua entrada. Não serão considerados nesse estudo, autômatos temporizados, uma vez que os modelos a serem utilizados nos experimentos propostos são as redes de Petri.

2.4.1. Redes de Petri temporizadas

Será, agora, introduzido uma temporização à estrutura de redes de Petri. Para tanto, uma seqüência de tempo v_j é agora associada com uma transição t_j. Um número real positivo, $v_{j,k}$, atribuído a t_j terá o seguinte significado: quando a transição t_j for habilitada no k-ésimo instante, ela não dispara imediatamente, mas incorre em um atraso no disparo dado por $v_{j,k}$; durante este atraso, os discos são guardados na entrada do lugar t_j.

É importante ressaltar que nem todas as transições terão que disparar com atraso. Algumas transições podem disparar tão logo sejam habilitadas. Assim, T (o conjunto das transições ou eventos) pode ser dividido em dois subconjuntos T_0 e T_D, tais que $T = T_0 \cup T_D$, onde T_0 é um conjunto de transições que sempre ocorrem sem atrasos no disparo, e T_D é o conjunto de transições que geralmente incorrem em algum atraso no disparo. O último é chamado de transições dependentes do tempo

As seguintes definições podem ser apresentadas.

Definição 2.14. A *estrutura de tempo* associada a um conjunto de transições temporizadas $T_D \subseteq T$ de uma rede de Petri marcada (P, T, A, w, x) é um conjunto V = $\{v_j : t_j \in T_D\}$ de seqüências de temporização (tempo de vida), $v_j = \{v_{j,1}, v_{j,2}, ...\}$, $t_j \in T_D$, $V_{j,k} \in R_+$, $k = 1, 2, ...$ □

Graficamente, as transições sem atraso de disparo são representadas por barras, ao passo que transições temporizadas são representadas por retângulos, conforme mostra a figura 2.12. A seqüência de tempo associada com uma transição temporizada é normalmente escrita próximo ao retângulo.

Definição 2.15. Uma *Rede de Petri Dependente do Tempo* (*com temporização*) é uma sextupla (*P, T, A, w, x,* V*)* onde (*P*, *T, A, w, x)* é uma rede de Petri marcada, e V = $\{v_j : t_j \in T_D\}$ é uma estrutura de tempo. □

Para ilustrar as definições de estrutura de tempo para redes de Petri, considere uma seqüência de duas tarefas T_1 e T_2 que são desempenhadas simultaneamente com uma terceira tarefa T_3, e então uma quarta tarefa T_4 é executada para combinar as saídas de T_2 e T_3. Uma possível modelagem deste processo é mostrada na figura 2.12.

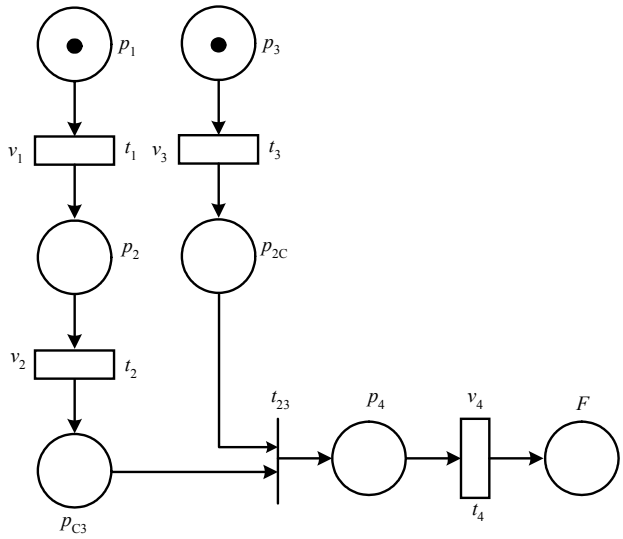


Figura 2.12: Rede de Petri dependente do tempo.

As transições dependentes do tempo t_1, t_2, t_3, t_4 correspondem aos eventos de conclusão das tarefas são indicadas pelos retângulos acompanhados pelos correspondentes tempos de atrasos de disparo v_1, v_2, v_3, v_4. Quando a tarefa 2 é concluída, um disco é somado para o lugar p_{C3} (indicando que T_2 está completa, mas T_3 ainda pode estar em execução). Similarmente, para o lugar p_{2C}. Assim, a transição de tempo t_{23} será habilitada quando ambas as tarefas 2 e 3 são completadas. Note que os lugares p_{C3} e p_{2C} podiam diretamente habilitar a transição t_4, omitindo t_{23} e p_4. O processo é terminado quando um disco é adicionado ao lugar F. Na figura 2.12 a rede de Petri tem um estado inicial tal que as tarefas 1 e 3 estão sendo realizadas.

Pode-se explorar a estrutura da rede de Petri decompondo o modelo dentro de dinâmicas de transições individuais. As equações de estado em um tal modelo geram seqüências de disparo de transições da forma $\{\tau_{j,1}, \tau_{j,2}, ...\}$, $j = 1, ..., m$ onde $\tau_{j,k}$ é o k-ésimo tempo de disparo da transição t_j, $k = 1, 2,...,$ que é ilustrado na figura 2.13. A obtenção desses modelos é, em geral, uma tarefa bastante complicada, porém para uma classe especial de sistemas (sistemas de filas, por exemplo), ele pode ser obtido com relativa facilidade.

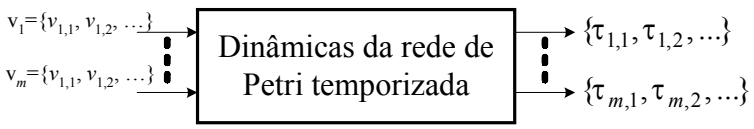


Figura 2.13: Modelo de uma rede de Petri dependente do tempo de um SED.

Será considerado, inicialmente, o caso em que um lugar p_i tem somente uma transição de entrada t_r. Para tanto, seja $\pi_{i,k}$ o instante de tempo quando o lugar p_i recebe seu k-ésimo disco, k = 1, 2,.... Suponha inicialmente que $x(p_i) = 0$. Então, o k-ésimo instante em que um disco é depositado em p_i é precisamente o k-ésimo tempo de disparo t_r. que é denotado por $\tau_{r,k}$. Se, por outro lado, p_i inicialmente contém x_{i0} discos, então, o k-ésimo tempo de disparo de t_r é o tempo quando p_i recebe seu $(x_{i0} + k)$-ésimo disco. Portanto tem-se a seguinte relação:

$$\pi_{i,k+x_{i0}} = \tau_{r,k}, p_i \in O(t_r), \qquad k = 1, 2, ... \tag{2.5}$$

onde x_{i0} é a marcação inicial do lugar p_i e $\pi_{i,k} = 0$ para todo $k = 1,... , x_{i0}$. Equivalentemente,

$$\pi_{i,k} = \tau_{r,k-x_{i0}}, \; p_i \in O(t_r), \qquad k = x_{i0} + 1, x_{i0} + 2, ... \tag{2.6}$$

Será, agora, considerado o caso em que o lugar p_i tem somente uma única transição de saída t_j. Para tanto, suponha que p_i é o único lugar de entrada de t_j. Se t_j é não dependente do tempo, então o k-ésimo tempo de disparo de t_j é precisamente o tempo em que p_i recebe seu k-ésimo disco, e habilita t_j. Então, tem-se que $\tau_{j,k} = \pi_{i,k}$, k = 1, 2, ... Por outro lado, t_j *é* dependente do tempo com uma seqüência de tempo $\mathbf{v}_j$, então este relacionamento torna-se

$$\tau_{j,k} = \pi_{i,k} + v_{j,k} \qquad k = 1, 2, ... \tag{2.7}$$

Finalmente se p_i não é a única entrada do lugar t_j, então t_j é habilitado para o k-ésimo tempo sempre que o último lugar de entrada do conjunto $I(t_j)$ recebe seu k-ésimo disco, isto é, em algum instante $\pi_{s,k}$, $p_s \in I(t_j)$, tal que $\pi_{s,k} \geq \pi_{i,k}$ para todo $p_i \in I(t_j)$. Pode-se, então, expressar este simples fato com a expressão

$$\tau_{j,k} = \max_{p_i \in I(t_j)} \{ \pi_{i,k} \} + v_{j,k}, \qquad k = 1, 2, \ldots \tag{2.8}$$

Em resumo, a combinação de (2.5) até (2.8) fornece um conjunto de equações recursivas que permitem determinar o tempo de disparo das transições para esta classe de redes de Petri.

Para ilustrar as dinâmicas da rede de petri, considere um sistema de filas com redes de Petri temporizadas, sendo $P = \{Q, I, B\}$ o conjunto que representa os estados do servidor e $\{a, d\}$ o conjunto que representa as chegadas de clientes à fila e às partidas de clientes do servidor. Um modelo de rede de Petri temporizado é mostrado na figura 2.14 com estado x = [0, 1, 0], isto é, quando a fila está vazia e o servidor está ocioso. Neste caso, o conjunto de transição dependente do tempo é $t_D = \{a, d\}$,

correspondente para chegadas de cliente e para partidas do servidor. A transição *s*, por outro lado, não precisa de nenhum atraso de disparo. O serviço começa tão logo o servidor fique ocioso e um cliente esteja na fila. A estrutura de tempo para este modelo consiste de $\mathbf{v}_a = \{v_{a,1}, v_{a,2}, ...\}$ e $\mathbf{v}_d = \{v_{d,1}, v_{d,2}, ...\}$ e é a mesma estrutura de tempo de um modelo de autômato dependente do tempo. Além disso, pode-se fazer $x(t) = x(Q) + x(B)$ denotar o número total de discos nos lugares *Q e B* com o tempo *t*.

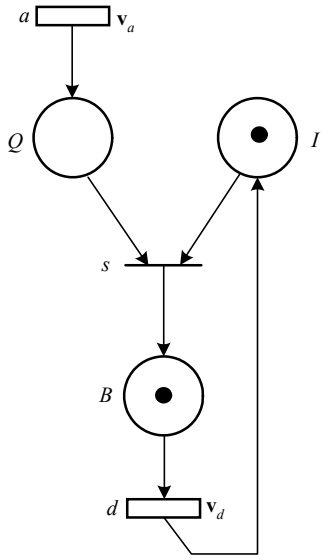


Figura 2.14: Rede de Petri dependente do tempo de um simples sistema de filas.

Observando a figura 2.14, pode-se ver que o modelo satisfaz os requisitos de um grafo marcado. Portanto, pode-se imediatamente derivar as equações da forma (2.5) a (2.8) para descrever a dinâmica de disparo das transições de um simples sistema de filas. Fazendo:

a_k : *k*-ésimo instante de chegada.

d_k : *k*-ésimo instante de partida.

s_k : *k*-ésimo instante de começo do serviço.

$\pi_{Q,k}$: instante em que *Q* recebe seu *k*-ésimo disco.

$\pi_{I,k}$: instante em que *I* recebe seu *k*-ésimo disco, com $\pi_{I,1} = 0$.

$\pi_{B,k}$: instante em que *B* recebe seu *k*-ésimo disco.

De (2.6), ou diretamente por inspeção, pode-se escrever:

$a_k = a_{k-1} + v_{a,k}$, $\quad k = 1, 2, ...\ a_0 = 0$

$s_k = \max\{\pi_{Q,k}, \pi_{I,k}\}$, $\quad k = 1, 2, ...$

$d_k = \pi_{B,k} + v_{d,k}$, $\quad k = 1, 2, ...$

$\pi_{Q,k} = a_k$, $\quad k = 1, 2, ...$

$\pi_{I,k} = d_{k-1}$, $\quad k = 1, 2, ..., \pi_{I,1} = 0$

$\pi_{B,k} = s_k$, $\quad k = 1, 2, ...$

Combinando as equações acima para eliminar $\pi_{Q,k}$, $\pi_{I,k}$ e $\pi_{B,k}$, resulta:

$s_k = \max\{a_k, d_{k-1}\}$, $\quad k = 1, 2, \ldots, d_0 = 0$

$d_k = s_k + v_{d,k}$, $\quad k = 1, 2, \ldots$

Pode-se ainda eliminar s_k para obter a seguinte relação fundamental que é importante para este simples sistema de fila, obtendo

$d_k = \max\{a_k, d_{k-1}\} + v_{d,k}$, $\quad k = 1, 2, \ldots, d_0 = 0$,

que representa uma simples relação recursiva caracterizando os instantes de partida dos clientes, que representa o fato de que a k-ésima partida ocorre no tempo $v_{d,k}$ unidades depois da (k - l)-ésima partida, exceto quando $a_k > d_{k-1}$. Este último caso ocorre quando a partida em d_{k-1} esvazia a fila; o servidor deve então esperar a próxima chegada no instante a_k,e gerar a próxima partida no instante $a_k + v_{d,k}$.

Reescrevendo a_k e d_k para $k = 2, 3, ...$ obtém-se:

$a_k = a_{k-1} + v_{a,k}$, $\quad a_0 = 0$

$d_k = \max\{a_{k-1} + v_{a,k}, d_{k-1}\} + v_{d,k}$, $\quad d_0 = 0$

que representa um modelo em espaço de estado na estrutura da figura 2.13. O modelo de sistema de filas é dirigido pelas seqüências de tempo $\mathbf{v}_a$ e $\mathbf{v}_d$, e sua saída consiste de seqüências de tempo de chegada e partida $\{a_1, a_2, ...\}$ e $\{d_1, d_2, ...\}$ geradas através das equações de estado de a_k e d_k para $k = 1, 2, ..., a_0 = 0$ e $d_0 = 0$.

Capítulo 3. Programação e utilização de controladores lógicos programáveis

Neste capítulo será feita uma análise geral sobre controladores lógicos programáveis (CLP) com suas características. Serão apresentados, também, alguns CLP disponíveis no mercado e abordadas cada uma de suas partes com seus detalhes técnicos.

Este capítulo está estruturado da seguinte forma: na seção 3.1 é descrito o CLP com suas definições e características principais, sendo os principais blocos que compõem um CLP descritos, e também, uma classificação dos tipos de CLPs; na seção 3.2 é realizado um estudo sobre linguagem ladder (linguagem esta que será utilizada para implementação dos experimentos deste trabalho), apresentando também as implementações das funções utilizadas em projetos de circuitos digitais (funções NOT, AND, OR, NAND, NOR, OR Exclusivo e NOR Exclusivo); na seção 3.3 são vistas formas de conversão entre linguagem ladder e redes de Petri, onde são descritos alguns métodos relacionados à tentativa de conversão direta, sendo mostrados ainda alguns exemplos de conversão utilizando um desses métodos, tendo como objetivo principal a redução do esforço de projeto devido à complexidade de problemas de controle e prover critérios de projeto para ajustar as aproximações.

3.1. Controlador lógico programável

3.1.1. Introdução

De acordo com a Associação Brasileira de Normas Técnicas (ABNT), o controlador lógico programável (CLP) é um equipamento eletrônico digital com hardware e software compatíveis com aplicações industriais. A National Electrical Manufacturs Association (NEMA), de acordo com a International Electrotechnical Commission (IEC), segundo a norma IEC 1131-1, define CLP como sendo um aparelho eletrônico digital que utiliza uma memória programável para o armazenamento interno de instruções para implementações específicas, tais como lógica, seqüenciamento, temporização, contagem e aritmética para controlar, através de módulos de entradas e saídas, vários tipos de máquinas ou processos.

O controlador lógico programável pode, também, ser definido como um

dispositivo de estado sólido, com memória programável para armazenamento de instruções para controle lógico programável e pode executar funções equivalentes às de um painel de relés ou de um sistema de controle lógico, também realizando operações lógicas e aritméticas, manipulação de dados e comunicação em rede, sendo utilizado no controle de sistemas automatizados.

O CLP e seus periféricos, ambos associados, são projetados de forma a poder ser integrados dentro de um sistema de controle industrial e finalmente usados a todas as funções às quais são destinados. A figura 3.1 apresenta uma aplicação do CLP.

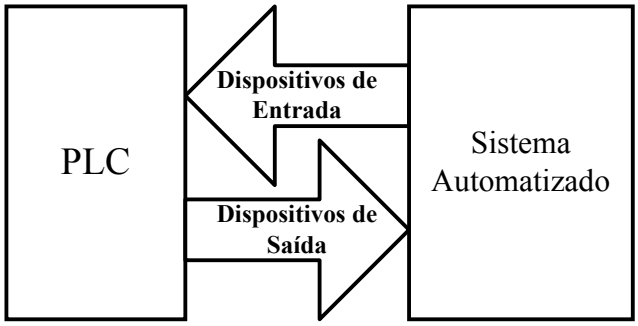


Figura 3.1: Aplicação geral do controlador lógico programável.

Os principais blocos que compõem um CLP são:

1) CPU (Unidade Central de Processamento). Compreende o processador, o sistema de memória e os circuitos auxiliares de controle. O sistema de memória compreende:

1.1) Memória de programa: Armazena as instruções do software aplicativo e do usuário (programas que controlam a máquina ou a operação do processo), que são continuamente executados pela CPU. Pode ser memória RAM, EPROM, EEPROM, NVRAM ou FLASH-EPROM.

1.2) Memória de dados: Armazena temporariamente os estados de I/O (entrada/saída), marcadores ou presets de temporizadores, contadores e valores digitais para que a CPU possa processá-los. A cada ciclo de varredura, a memória de dados é atualizada. Geralmente é uma memória RAM, sendo também conhecida como memória de rascunho.

2) Módulos de I/O. Podem ser discretos com sinais digitais, contatos normalmente abertos, contatos normalmente fechados ou analógicos.

3) Fonte de Alimentação. Dispositivo responsável pela tensão de alimentação

fornecida à CPU e aos Módulos (circuitos) de I/O. Em alguns casos proporciona saída auxiliar de baixa corrente.

4) Base ou Rack. Serve de conexão mecânica e elétrica entre a CPU, os módulos de I/O e a fonte de alimentação. Contém o barramento de comunicação entre os dispositivos, no qual os sinais de dados, endereço, controle e tensão de alimentação estão presentes.

A figura 3.2 mostra a estrutura básica de um CLP por meio dos blocos descritos. Um CLP comercial é apresentado na figura 3.3.

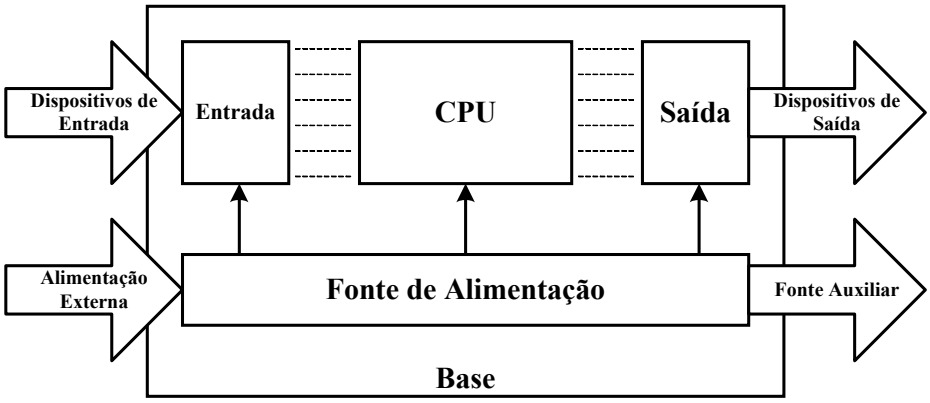


Figura 3.2: Estrutura básica do CLP.

CLP Allen Bradley: sistema de I/O Compact Logix 1769

Fonte de Tensão Processador Módulos de I/O

Figura 3.3: Exemplo de CLP disponível no mercado.

3.1.2. Operação Básica

A CPU controla todas as ações do CLP executando a leitura das condições e estados dos dispositivos por meio dos módulos de I/O. Essas condições são armazenadas em memória para serem processadas pelo programa de aplicação desenvolvido pelo usuário e armazenado na memória no CLP. O processador atualiza os status dos dispositivos de saída por meio dos módulos de I/O, realizando a lógica de controle, para garantir o ciclo de varredura.

A programação pode ser feita através de um programador manual ou com software de programação no computador para posterior transferência. A linguagem ladder (Relay Ladder Logic), é a mais utilizada. Esta linguagem é a representação lógica da seqüência elétrica de operação, como ilustrado nas figuras 3.4 e 3.5.

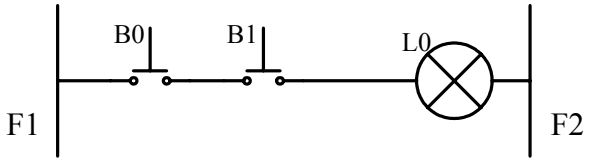


Figura 3.4: Lógica convencional - contatos elétricos.

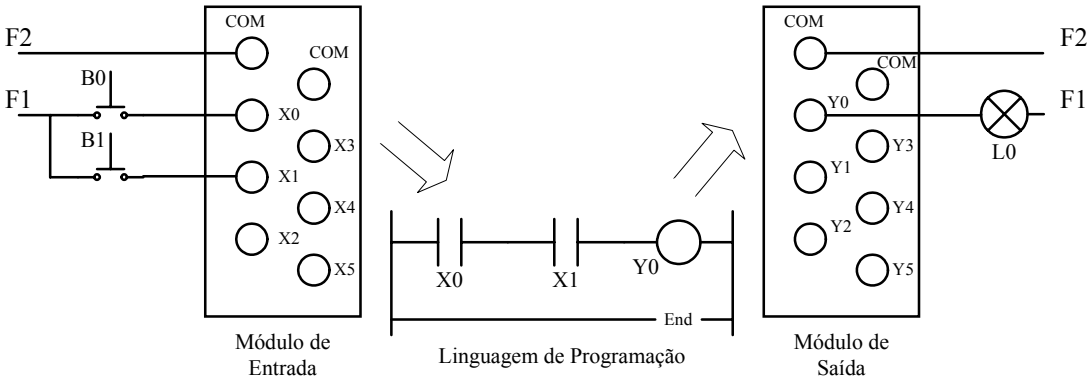


Figura 3.5: Implementação da lógica convencional por meio de CLP.

De acordo com as figuras 3.4 e 3.5 (SILVEIRA, 1999), vê-se que a lógica implementada pelo CLP é muito similar à convencional, sendo que os dispositivos de entrada (chaves B0 e B1) são conectados aos módulos de entrada e os dispositivos de saída (lâmpada L0) aos módulos de saída. O programa de aplicação determina o acionamento da saída em função das entradas. Qualquer alteração desejada nesta lógica é realizada por meio de alterações no programa, permanecendo as mesmas conexões nos módulos de I/O.

3.1.3. Arquitetura básica do CLP

A figura 3.6 ilustra a estrutura básica de um CLP. Esta compreende o processador, o sistema de memória, os barramentos de dados, de endereços e de controle, além dos circuitos auxiliares de controle. A figura 3.7 apresenta três modelos de CPUs disponíveis em uma mesma família de CLP, relacionando algumas de suas características.

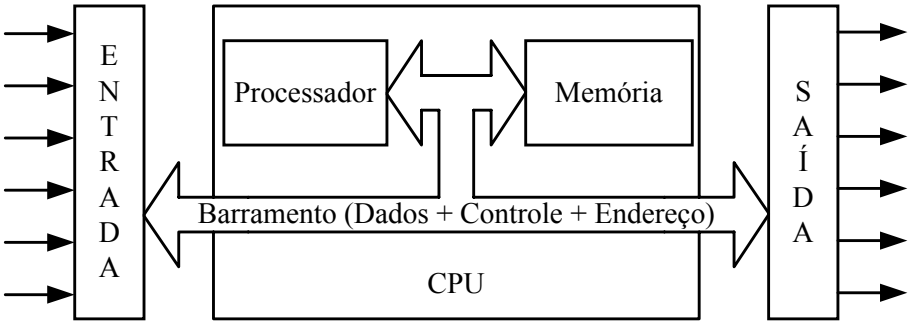


Figura 3.6: Estrutura básica da CPU.

Controllogix Allen Bradley Compact Logix 1769 CPU PremiumTelemecanique

Figura 3.7: Exemplos de CPUs disponíveis no mercado (modelos Allen Bradley).

A seguir , as principais características desses elementos serão apresentadas.

a) Processador

Os processadores podem ser do tipo microprocessador/controlador convencional 80286, 80386, 8051, até um processador dedicado, um DSP (Processador Digital de Sinais). Os processadores atualmente utilizados em CLPs são dotados de alta capacidade computacional. Existem CPUs que possuem processamento paralelo, no qual dois ou mais processadores executam o programa de aplicação, comparando os resultados obtidos após cada ciclo de varredura.

O processador é responsável pelo gerenciamento total do sistema, controlando os barramentos de endereços, de dados e de controle, interpreta e executa as instruções do

programa de aplicação, controla a comunicação com dispositivos externos e verifica a integridade de todo o sistema realizando relatórios ou diagnósticos do sistema operacional. Pode operar com registros e palavras de instrução, ou de dados, de diferentes tamanhos (8, 16 ou 32 bits), de acordo com a capacidade do acumulador e pela lista de instruções disponíveis para cada CPU.

b) Sistema de memória

O sistema de memória da CPU é composto por: (i) memória do sistema de operação, onde é armazenado o programa de execução desenvolvido pelo fabricante, e que determina como o sistema deve operacionalizar, incluindo a execução dos programas do usuário, controle de serviços periféricos, atualização dos módulos de I/O etc. (ii) memória de aplicação ou memória de usuário, onde o programa desenvolvido pelo usuário para implementação (chamado de programa de aplicação) é armazenado. Juntamente com o programa de aplicação, são armazenados os dados do sistema em uma tabela para realização dos controles dos módulos de I/O utilizados. Cada ponto de I/O conectado aos módulos, tem um endereço específico na tabela de dados, o qual é acessado pelo programa de aplicação. Essa memória é do tipo RAM. A tabela 3.1 apresenta uma comparação entre as memórias do sistema de operação e do usuário.

Tabela 3.1: Sistema de memória da CPU.

Sistema de Memória da CPU	
Memória do Sistema de Operação	**Memória de Aplicação**
Programa de Execução: ROM / EPROM	Programa de Aplicação: RAM (Bateria) / EPROM / EEPROM
Rascunho do Sistema: RAM (Bateria opcional)	Tabela de Dados: RAM (Bateria opcional)

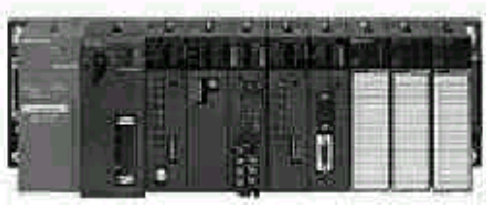
c) Módulos de entrada/saída (I/O)

Os módulos de I/O realizam a comunicação entre a CPU e os dispositivos externos por meio das entradas e saídas dos módulos, garantindo isolação e proteção à CPU. Os módulos de entrada recebem os sinais dos dispositivos de entrada tais como sensores, chaves e transdutores e convertem esses sinais em níveis adequados para

serem processados pela CPU. Os módulos de saída enviam os sinais de controle aos dispositivos externos tais como motores, atuadores e sinalizadores. Esses sinais são resultantes da lógica de controle, pela execução do programa de aplicação, ou podem ser forçados pelo usuário, independente da lógica de controle.

Para CLP compactos com CPU e I/O alojados em um único invólucro, usa-se o termo "circuitos de I/O" e para CLP modulares com CPU e I/O disponíveis de forma independente, usa-se o termo "módulos de I/O". A figura 3.8 mostra exemplos de CLP compacto e CLP modular.

CLP Compacto Mitsubishi FX2N



CLP Modular Mitsubishi serie AnSH

Figura 3.8: Exemplos de CLPs compacto e modular disponíveis no mercado.

Os módulos de I/O são classificados como Discretos (Digitais) ou Analógicos, existindo também os especiais em algumas famílias de CLPs. Os módulos discretos de I/O tratam sinais digitais e são utilizados em sistemas seqüenciais e na maioria das aplicações com CLPs, mesmo como parte de sistemas contínuos, conforme mostrado na figura 3.9. Os módulos analógicos tratam os sinais analógicos que são utilizados pelos sistemas contínuos. Os módulos analógicos de entrada convertem sinais analógicos provenientes dos dispositivos de entrada (transdutor, conversor, termopar) em sinais digitais por meio de conversores analógico/digital, disponibilizando-os adequadamente ao barramento da CPU. Os módulos analógicos de saída convertem sinais digitais, disponíveis no barramento da CPU, em sinais analógicos por meio de conversor digital/analógico, enviando-os aos dispositivos de saída (driver, amplificador). A figura 3.10 ilustra o funcionamento dos módulos analógicos de entrada e de saída. Na figura 3.11 são mostrados alguns elementos de entrada e saída analógicos.

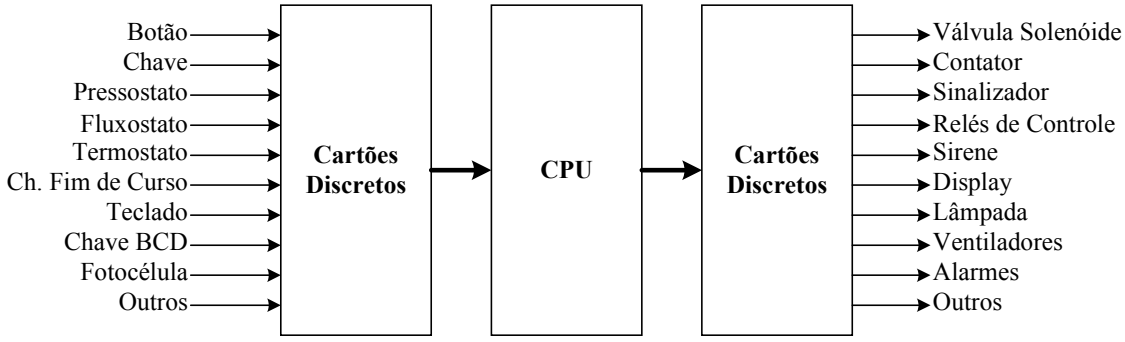


Figura 3.9: Elementos de entrada e saída discretos.

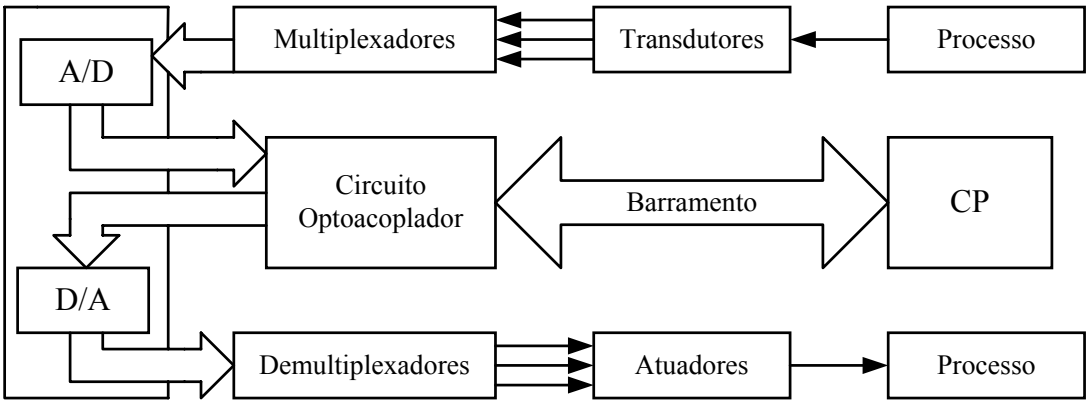


Figura 3.10: Módulos de entrada e saída analógica.

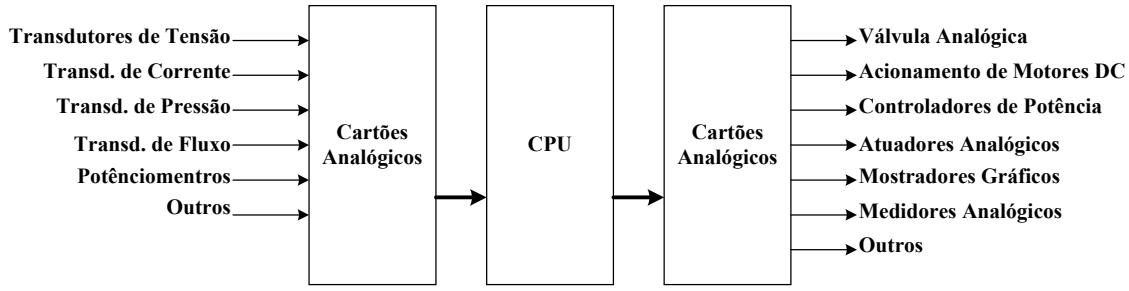


Figura 3.11: Elementos de entrada e saída analógicos.

d) Fonte de alimentação

É o dispositivo responsável pela conversão da tensão de alimentação fornecida pela rede de energia elétrica aos níveis aceitáveis de funcionamento para cada tipo de aplicação. A fonte de alimentação fornece todos os níveis de tensão para a alimentação da CPU e dos módulos de I/O e funciona como um dispositivo de proteção. A fonte do

CLP é responsável, em alguns casos, pela alimentação do circuito lógico dos módulos de I/O, sendo que a fonte externa alimenta os circuitos de potência, ou circuitos externos. A figura 3.12 mostra duas possibilidades de apresentação da fonte de alimentação.

Atualmente, as fontes de alimentação para os CLPs são do tipo chaveadas. Em alguns casos, a tensão de entrada não é fixa e nem selecionável pelo usuário, mas a fonte possui ajuste automático, proporcionando maior versatilidade e qualidade ao sistema. As proteções externas são basicamente um transformador de isolação ou supressor de ruídos para rede e aterramento.

Fonte Fonte de alimentação acoplada a base

Figura 3.12: Fonte de alimentação (Allen Bradley Controllogix 1756).

e) Base ou Rack

A Base é responsável pela sustentação mecânica dos elementos. Possui o barramento que faz a conexão elétrica entre os elementos do CLP, no qual estão presentes os sinais de dados, endereço e controle necessários para comunicação entre a CPU e os módulos de I/O, além dos níveis de tensão fornecidos para que possam operar. A figura 3.13 apresenta um exemplo de base com indicação do barramento interno.

Figura 3.13: Exemplo de base do modelo Allen Bradley Controllogix 1756.

Cada posição da base é denominada de slot (ranhura, abertura) e cada slot da base tem uma identificação própria, conforme o fabricante (figura 3.14). Uma mesma família de CLP pode possuir bases com diferentes quantidades de slots, com o objetivo de

atender às necessidades específicas de cada aplicação.

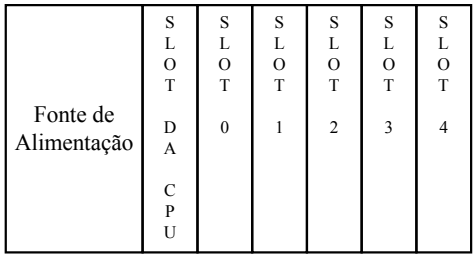


Figura 3.14: Exemplo de identificação dos Slots da base.

3.1.4.Classificação dos CLPs

Segundo GEORGINI (2000), os CLPs podem ser divididos em grupos específicos de acordo com a estrutura que apresentem, à quantidade de pontos de I/O que a CPU pode controlar e à quantidade de memória de programação disponível. São esses:

1 - Micro CLP, com até 64 pontos de I/O e até 2Kwords de memória.

2 - Pequeno CLP, com 64 a 512 pontos de I/O e até 4Kwords de memória.

3 - CLP Médio, com 256 a 2048 pontos de I/O e dezenas de Kwords de memória.

4 - CLP Grande, que possui acima de 2048 pontos de I/O e centenas de Kwords de memória.

Em 1997, os controladores lógicos programáveis com até 14 pontos de I/O e tamanho muito reduzido foram lançados no mercado, tendo sido denominados pelos fabricantes de Nano CLP.

Entre os micros e pequenos CLP é, ainda, possível encontrar outra divisão:

1 – CLP Compacto: que tem quantidade fixa de pontos de I/O.

2- CLP Modular: que permite a configuração, por parte do usuário, da quantidade e combinação dos pontos de I/O.

3.2. Linguagem de programação (ladder)

A linguagem de programação que será utilizada neste trabalho para implementação nos CLP é a linguagem ladder. Essa é uma linguagem gráfica baseada em símbolos, semelhantes aos contatos de bobinas nos esquemas elétricos. Por sua semelhança com sistemas de controle a relés é facilmente compreendida.

Existem outros tipos de linguagem que são utilizadas nos CLPs, como a linguagem de “lista de instruções” que é amplamente difundida na Europa. A lista de

instruções é uma linguagem textual, semelhante ao assembly, e faz parte das linguagens básicas normalmente disponíveis em um CLP.

As outras linguagens de programação encontrados no mercado para programação dos CLP são as linguagens C e BASIC. A Norma IEC 61131 (inicialmente 1131), de agosto de 1992, apresenta atualmente oito partes (IEC 61131-1 a IEC 61131-8). A terceira parte (IEC 61131-3), aborda as linguagens de programação, e define, também, a estrutura de um projeto, os tipos de dados e a organização interna do programa. As cinco linguagens de programação definidas com sintaxe e semântica de duas linguagens textuais e duas linguagens gráficas, e estruturação por diagramas funcionais, podendo, inclusive, ser interligadas, são: ladder (LD), lista de instruções (IL), texto estruturado (ST), diagrama de blocos de função (FBD) e diagrama funcional seqüencial (SFC) (NATALE, 1995, OLIVEIRA, 1993).

3.2.1. Programadores

As principais ferramentas para programação disponíveis atualmente para as famílias de CLP encontrados no mercado são o programador manual e o software de programação para PC. Ambas as ferramentas possuem recursos para monitoração de condições internas à CPU (diagnósticos e erros), verificação da execução do programa de aplicação e controle sobre os modos de operação, entre outros. Cada fabricante, e em alguns casos cada família de CLP, tem suas próprias ferramentas de programação que não podem ser usadas para CLP (ou CPU) distintos.

O programador manual é uma ferramenta de menor custo e é utilizada para pequenas alterações. Conforme mostrado na figura 3.15, o programador manual possui um display de cristal líquido com duas linhas para apresentação das informações (endereço e dados do programa, condição dos pontos de I/O e diagnósticos internos) e um teclado de membrana para entrada dos dados. O programador manual, contudo, não é indicado para o desenvolvimento de todos os programas de aplicação, pois permite edição e alteração apenas por meio de mnemônicos (linguagem de lista de instruções). Porém, é bastante útil como ferramenta de manutenção para trabalho de campo, proporcionando visualização, monitoração e alteração de parâmetros e do programa de aplicação com muita rapidez e com a vantagem de ser portátil e resistente ao ambiente industrial.

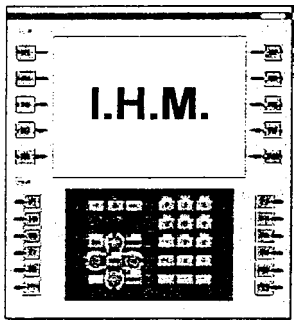


Figura 3.15: Ferramentas de programação - programador manual.

O software de programação para PC é desenvolvido de acordo com as normas da IEC, variando muito de acordo com o fabricante em seu ambiente de trabalho e em suas configurações internas. Mais adiante será descrito o ambiente de programação utilizado neste trabalho, que é o PL7 micro.

3.2.2. Fundamentos de programação em linguagem ladder

A figura 3.16 mostra alguns componentes da linguagem ladder e na figura 3.17 um exemplo de um programa em linguagem ladder.

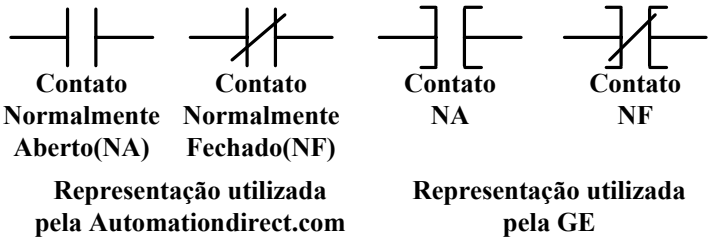


Figura 3.16: Exemplo de instruções em linguagem ladder.

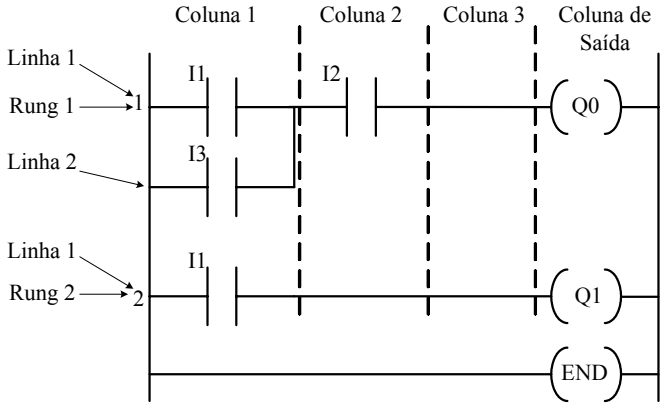


Figura 3.17: Componentes da programação em linguagem ladder.

A quantidade de colunas e linhas, ou elementos e associações, que cada "rung" pode conter é determinada pelo fabricante do PLC, podendo variar conforme a CPU utilizada. Em geral, este limite não representa uma preocupação ao usuário durante o desenvolvimento do programa de aplicação, pois os softwares de programação indicam se tal quantidade foi ultrapassada, por meio de erro durante a compilação do programa de aplicação.

Cada elemento da lógica de controle representa uma instrução da linguagem ladder, sendo alocada em um endereço específico e consumindo uma quantidade determinada de memória (word) disponível para armazenamento do programa de aplicação, conforme a CPU utilizada. Um mesmo símbolo gráfico da linguagem ladder (contato normalmente aberto, por exemplo) pode representar instruções diferentes, dependendo da localização na lógica de controle.

3.2.3. Implementação da lógica de controle

A figura 3.18 apresenta um diagrama de blocos que mostra como resolver em etapas um problema de lógica.

Figura 3.18: Organização do raciocínio na solução de problemas de lógica.

A análise para a criação de programas no CLP está baseada na lógica binária. Esse sistema é conhecido como álgebra de Boole, o qual está baseado em um conjunto de operações entre variáveis binárias. O sistema binário é um sistema de numeração que consta de dois valores, 0 e 1. Cada dígito de um número representado no sistema binário recebe o nome de bit, de maneira tal que cada bit pode tomar o valor 0 ou 1. A notação aqui utilizada é $\overline{A}$ para negar A, "**.**" e "**+**" para indicar "**e**" e "**ou**", respectivamente (BIGNELL e DONOVAN, 1995). Os teoremas booleanos são:

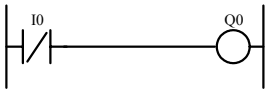
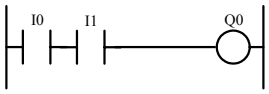
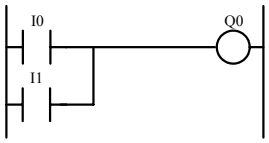
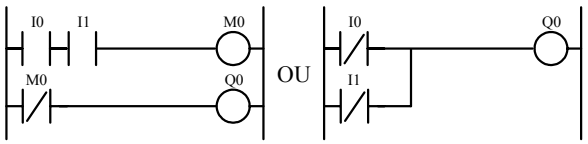
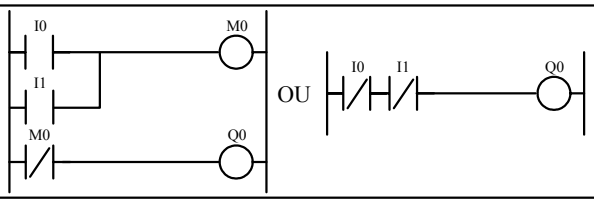
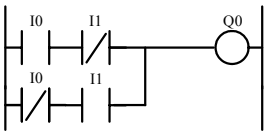
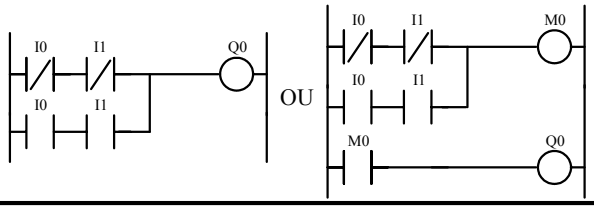
1. $\overline{\overline{A}} = A$
2. $A.0 = 0$
3. $A + 0 = A$
4. $A.1 = A$
5. $A + 1 = A$
6. $A + A = A$
7. $A.A = A$
8. $A + \overline{A} = A$
9. $A.\overline{A} = 0$
10. $A.B + A.C = A.(B+C)$
11. $A + \overline{A}.B = A + B$

Esses teoremas, juntamente com os teoremas de DE MORGAN permitem minimizar as relações lógicas entre as variáveis. Os teoremas de De Morgan são:

1. $\overline{A \,.\, B} = \overline{A} + \overline{B}$
2. $\overline{A + B} = \overline{A} \,.\, \overline{B}$

A tabela 3.2 a seguir considera as funções lógicas e suas implementações utilizando a linguagem ladder.

Tabela 3.2: Implementação da lógica booleana/De Morgan em linguagem ladder.

Função	Tabela-verdade	Lógica booleana / De Morgan	Implementação em Linguagem ladder
NOT	$\overline{A}$ | A 0 | 1 1 | 0	$A = \overline{A}$	
AND	A | B | Y 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1	Y = A.B	
OR	A | B | Y 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1	Y = A+B	
NAND	A | B | Y 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0	$Y = \overline{A.B} = \overline{A}+\overline{B}$	
NOR	A | B | Y 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 0	$Y = \overline{A+B} = \overline{A}.\overline{B}$	
OR Exclusivo	A | B | Y 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0	$Y=A.\overline{B}+\overline{A}.B= A \oplus B$	
NOR Exclusivo	A | B | Y 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1	$Y = \overline{A.\overline{B} + \overline{A}.B} =$ $Y = (\overline{A}+B).(A+\overline{B})$	

3.2.4. Implementação da lógica de controle por funções do CLP

a) Função SET e RESET

Em muitas situações é necessário fixar o valor lógico de uma variável no nível lógico 1, após o acionamento de um sensor através do uso da sentença SET, que permite ativar uma saída ante uma condição de entrada e permanecer ativada mesmo mudando o estado da entrada e podendo desativar este estado na execução de um programa mediante o uso de uma instrução de RESET, a qual desativa a saída ante uma condição da entrada e permanece desativada mesmo mudando essa condição de entrada.

Os CLP diferem na forma de implementar essas funções variando de acordo com o tipo de fabricante dos controladores. Um exemplo geral é mostrado na figura 3.19. Neste diagrama, após o acionamento da entrada I0, se estabelecerá (SET) na saída Q0 ou estado lógico 1, o qual permanecerá nesse estado indefinidamente, mesmo que o estado da entrada seja alterado. Somente mediante o acionamento da entrada I1, se desativará a saída Q0, voltando ao nível lógico 0, permanecendo nesse estado desde que I0 não esteja acionada. Em muitos CLPs a sentença de RESET prevalece sobre as sentenças de SET, não causando travamento do CLP.

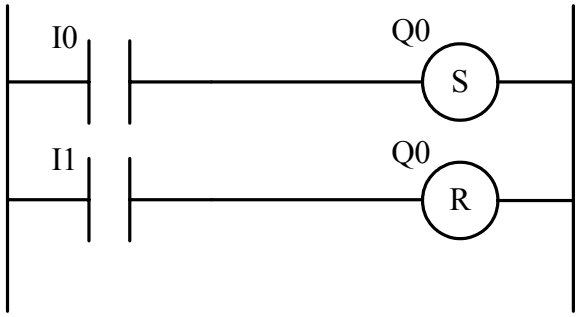


Figura 3.19: Circuito de SET e RESET.

b) Temporizadores

Os CLPs atuais possuem um recurso de temporização, acionando os dispositivos após um retardo de tempo. A maneira como ele é inserido no diagrama de contatos, assim como algumas de suas particularidades são próprias de cada modelo de CLP. Um diagrama de contatos para o uso de um temporizador está mostrado na figura 3.20. Note que o tempo ajustado é de 10 segundos. Assim ao se ativar a entrada I0, se ativará a base de tempo T0, a qual esperará um tempo 10 s. Passado esse tempo, se ativará a

chave T0 e, portanto, a saída Q0, a qual permanecerá ativada até a entrada I0 se desativar. Se, no entanto, a entrada I0 permanecer ativada um tempo menor que 10s, a saída Q0 não se ativará. O diagrama de tempos mostrado na figura 3.21 ilustra essa situação.

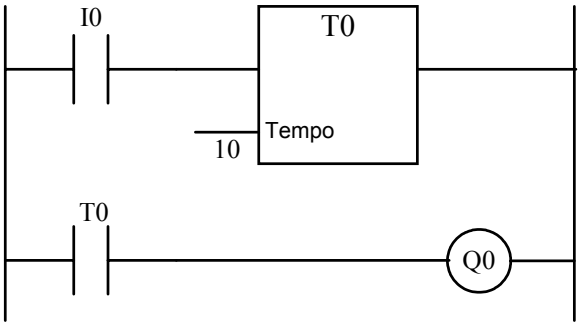


Figura 3.20: Diagrama “ladder” de um temporizador.

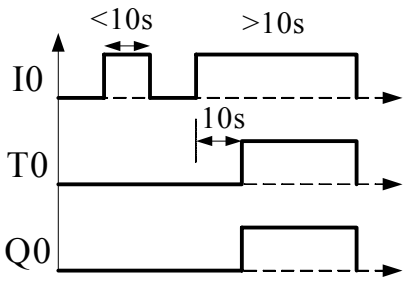


Figura 3.21: Diagrama de tempos de um circuito temporizador.

c) Contadores

Os CLPs possuem funções de contagem, que podem ter como base o tempo, pois variam de acordo com os sinais de entrada que ele recebe, permitindo controlar quantidades, número de ocorrências e com esses dados efetuar operações matemáticas internas do CLP. O modo de implementar um contador em um diagrama de contatos ou linguagem ladder também varia de um modelo para outro em um CLP. Existem contadores que realizam essa função em ordem crescente ou decrescente de acordo com o especificado no dispositivo. Um exemplo geral de um contador está mostrado na figura 3.22.

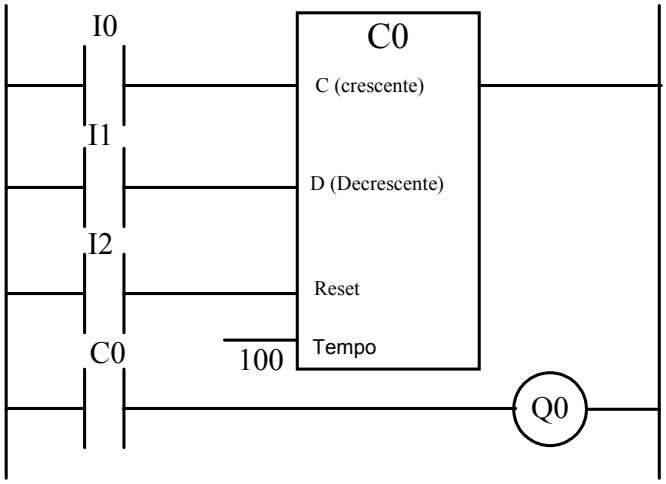


Figura 3.22: Diagrama de contatos de um circuito contador.

Note na figura 3.22 que o contador C0 possui 4 entradas: a entrada de contagem crecente (incrementando) C, controlada pela entrada I0, a entrada de contagem descendente D, controlada pela entrada I1, a entrada R para o iniciar a contagem, é controlada pela entrada I2, e finalmente a entrada PV, para especificar depois de quantas ocorrências se ativará o contador. Assim, a chave C0 será acionada, alimentando a saída Q0, após um número determinado de ocorrências, que neste caso será de 100.

d) Relés Especiais

São funções existentes em muitos CLPs e predefinidas pelo fabricante, por exemplo, instruções internas de memória, como indicação de status do acumulador da CPU, monitoramento do sistema, indicação de erros e base de tempo-real. São identificados por SP ou por outro símbolo de acordo com o tipo de CLP e são associadas às instruções booleanas de entrada, ou seja, contatos normalmente abertos ou contatos normalmente fechados. A tabela 3.3 apresenta alguns exemplos de relés especiais.

Tabela 3.3: Relés especiais (nível lógico alto 1 e nível lógico baixo 0).

Relé	Função	Descrição
SP0	Primeira varredura	Ativado apenas na primeira varredura da CPU; desativado nas demais
SP1	Sempre ligado	Ativado em todas as varreduras
SP2	Sempre deslidado	Desativado em todas as varreduras
SP3	Clock de 1 minuto	30seg em nivel alto e 30seg em nível baixo
SP4	Clock de 1 segundo	0,5s em nível alto e 0,5s em nível baixo
SP5	Clock de 100 mseg	50 ms em nível alto e 50 ms em nível baixo
SP6	Clock de 50 mseg	25 ms em nível alto e 25 ms em nível baixo

3.3. Conversão entre redes de Petri e linguagem ladder

Nesta seção serão apresentadas algumas formas de conversão entre redes de Petri e linguagem ladder. Devido à grande facilidade de análise, as redes de Petri têm sido empregadas em grande escala para análise de sistemas a eventos discretos e por esta razão, muitos têm estudado formas de conversão direta entre redes de Petri e linguagem ladder. O objetivo desta seção é mostrar que existe esta possibilidade, mas, sem entrar na questão de qual das programações é a melhor. Ao longo desta seção serão colocadas as formas de conversão, que muitas das vezes necessitam de um conhecimento prévio para análise e programação.

3.3.1. Métodos de conversão entre redes de Petri e linguagem ladder

Durante as últimas décadas, foram desenvolvidos muitos métodos para implementação em CLP baseados em redes de Petri e linguagem ladder. Esses métodos visam a conversão direta entre redes de petri e linguagem ladder. BOUCHER *et al*. (1989), desenvolveram um controlador para um robô e controle numérico de um torno mecânico com redes de Petri e carregadores e compiladores de linguagem ladder. Foi comparado o desempenho de controle, tendo as redes de Petri descrito um fluxo de processo mais eficaz que a linguagem ladder correspondente. JAFARI e BOUCHER (1994) desenvolveram uma interface entre uma especificação de alto-nível de um sistema e seu controlador lógico, cuja interface foi baseada em várias regras para transformar as especificações em um controlador baseado em redes de Petri e transferir

as especificações para linguagem ladder. CAZZOLA *et al*. (1995), investigaram como o desempenho de um algoritmo de controlador lógico é afetado pelo modo particular de paridade no qual um esquema de redes de Petri é convertido em um algoritmo corrente em um CLP. Os métodos sistemáticos para formular redes de Petri para CLP e comparar redes de Petri e linguagem ladder para sistemas industriais podem ser visto em VENKATESH e ZHOU (1998).

A tabela 3.4 mostra os elementos básicos que constituem uma linguagem ladder e uma rede de Petri e na tabela 3.5 são mostradas a lógica modelada construída por redes de Petri e linguagem ladder, que podem ser usadas para calcular o número de elementos básicos necessários para uma rede de Petri e linguagem ladder (PENG e ZHOU, 2001, LEE e HSU, 2001, PENG e ZHOU, 2004). Os critérios de comparação incluem complexidade de projeto (medida pelo número de nós básicos e ligações), facilidade de compreensão (habilidade para avaliar a lógica programada, verificar a transformação e manter o sistema de controle), flexibilidade (facilidade de modificar o sistema de controle quando ocorrem mudanças nas especificações) e tempo de resposta (tempo de varredura da linguagem programada).

Tabela 3.4: Elementos básicos da linguagem ladder e da rede de Petri.

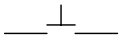
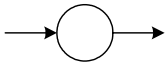
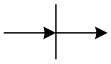
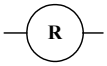
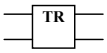
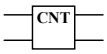
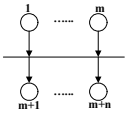
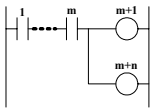
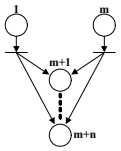
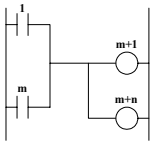
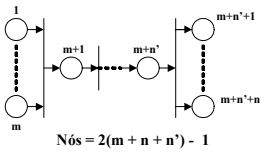
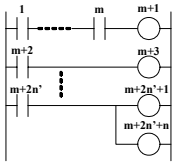
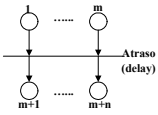
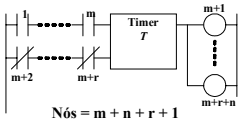
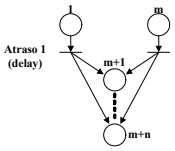
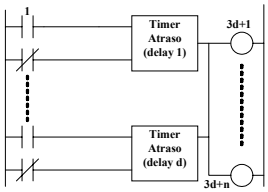
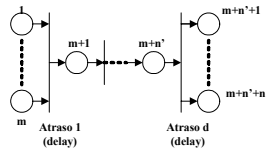
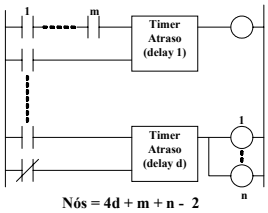
Elementos básicos	Linguagem ladder		Rede de Petri	
Nós	Botão de Pressão		Lugar	
	Chave NA		Transição	
	Chave NF			
	Relé			
	Temporizador			
	Contador			
	Solonóide			
Links	Linha		Arco normal	
			Arco inibidor	

Tabela 3.5: Lógica para conversão entre redes de Petri e linguagem ladder.

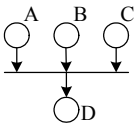
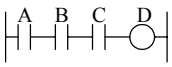
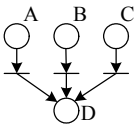
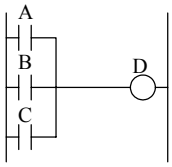
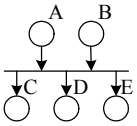
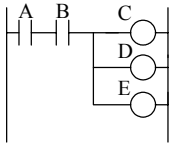
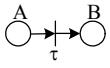
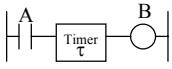
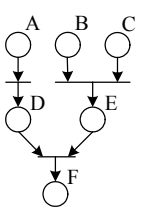
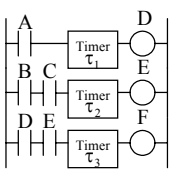
Lógica	Rede de Petri	Linguagem ladder
Lógica AND	 Nós = m + n + 1 Links = m + n	 Nós = m + n Links = m + 2n
Lógica OR	 Nós = 2m + n Links = m (1 + n)	 Nós = m + n Links = 2(m + n)
Modelo Sequencial	 Nós = 2(m + n + n') - 1 Links = m + n + 2n'	 Nós = m + n + 2n' Links = 3(n' + 1) + (m -1) + 2(n - 1)
Lógica AND Temporizada	 Nós = m + n + 1 Links = m + n	 Nós = m + n + r + 1 Links = m + 2n + r + 2
Lógica OR Temporizada	 Nós = 2d + n Links = d (1 + n)	 Nós = 3d + n Links = 5d + 2n
Modelo Sequencial Temporizado	 Nós = 2d + m + n - 1 Links = 2d + m + n - 2	 Nós = 4d + m + n - 2 Links = 6d + m + 2n - 3

3.3.2. Um método para converter redes de Petri em linguagem ladder

Para facilitar a análise deste assunto, deve-se salientar que é possível criar blocos lógicos para redes de Petri, tal como Lógica OR, Lógica AND, Concorrência e Sincronização. O método apresentado é, em particular, baseado em uma classe especial de redes de Petri, costurada para modelar o fluxo de controle em sistemas discretos. O objetivo principal da metodologia proposta é reduzir o esforço de projeto devido à complexidade de problemas de controle e prover critérios de projeto para ajustar as aproximações (LEE e HSU, 2001, JIMÉNEZ *et al*., 2001, JONES *et al*., 1996).

A lógica e outros blocos de construção básicos usados para controle em seqüência que são modelados por redes de Petri e linguagem ladder são mostrados na tabela 3.6. Na tabela, as primeiras quatro linhas mostram em fila como os elementos básicos de redes de Petri são usados para modelar condições, estados, atividades, informações e fluxo de material e recursos. Note que a linguagem ladder não possui as representações explícitas correspondentes. A lógica AND e a lógica OR podem facilmente modelar redes de Petri e linguagem ladder com complexidades semelhantes, os outros conceitos mencionados como concorrência e sincronização são também ilustrados na tabela 3.6, sendo acrescentado, também, a questão da temporização.

Tabela 3.6: Modelagem de redes de Petri e linguagem ladder.

Lógica de Construção	Rede de Petri	Linguagem ladder
Condição ou status do elemento do sistema	 Lugar	Representação não explicita
Atividade	Transição	Representação não explicita
Fluxo de informação ou material	Arco direcionado	Representação não explicita
Objetos ativos: máquinas, robôs, pallets, etc...	 Símbolo presente no lugar	Representação não explicita
Lógica AND IF A=1 AND B=1 AND C=1 THEN D=1		
Lógica OR IF A=1 OR B=1 OR C=1 THEN D=1		
Concorrência IF A=1 AND B=1 THEN C=1 AND D=1 AND E=1		
Temporizado IF A=1 THEN após atraso de tempo τ B=1		
Sincronização IF A=1 THEN Após atraso de tempo τ_1 D=1 IF B=1 AND C=1 THEN Após atraso de tempo τ_2 E=1 IF D=1 AND E=1 THEN Após atraso de tempo τ_3 F=1		

Para ilustrar a aplicação da tabela 3.6, suponha a seguinte função lógica: Y = ((X0 + X1 + X2).(X4.X5)) + X3, essa função gera respectivamente as seguintes linguagens ladder, conforme figura 3.23, e rede de Petri, figura 3.24.

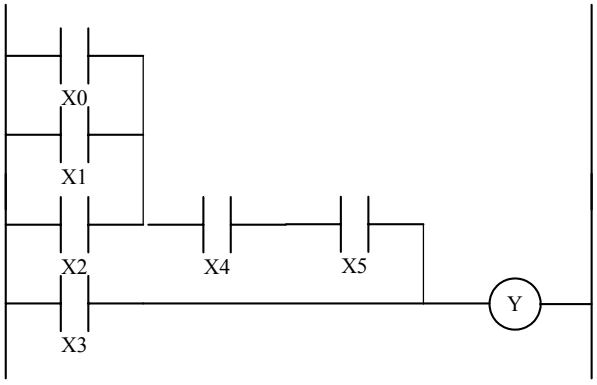


Figura 3.23: Linguagem ladder da função Y = ((X0 + X1 + X2).(X4.X5)) + X3.

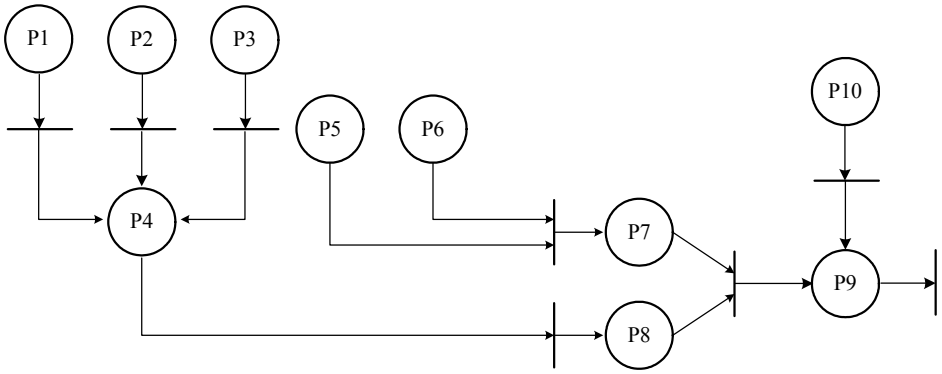


Figura 3.24: Rede de Petri da função Y = ((X0 + X1 + X2).(X4.X5)) + X3.

Considere, agora, um simples sistema industrial, conforme mostrado na figura 3.25. Este sistema consiste de uma estação de armazenamento de estrado (pallet), uma estação de montagem, uma estação de inspeção, um transportador, e um robô. Duas peças são colocadas, através do robô, na estação de montagem. Depois que a montagem das duas peças é feita, o robô descarrega o produto acabado em uma mesa de trabalho. O transportador de pallet leva um pallet vazio da estação de armazenamento de pallet para a estação de montagem. O robô também descarrega o produto acabado da mesa de trabalho e coloca no pallet que está no transportador. Finalmente, o transportador transporta o produto acabado carregado em pallet da estação de montagem para estação de inspeção.

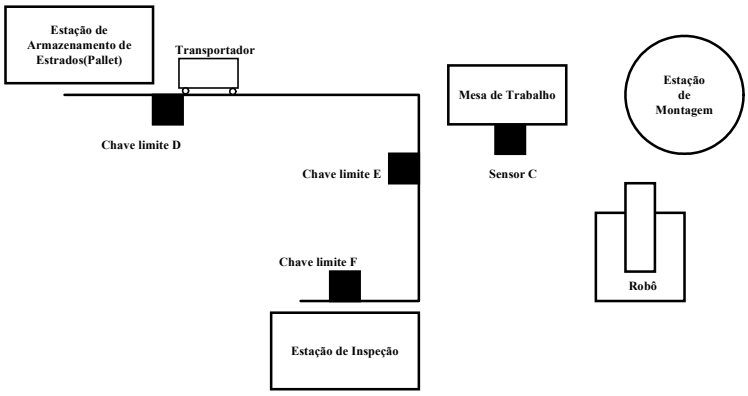
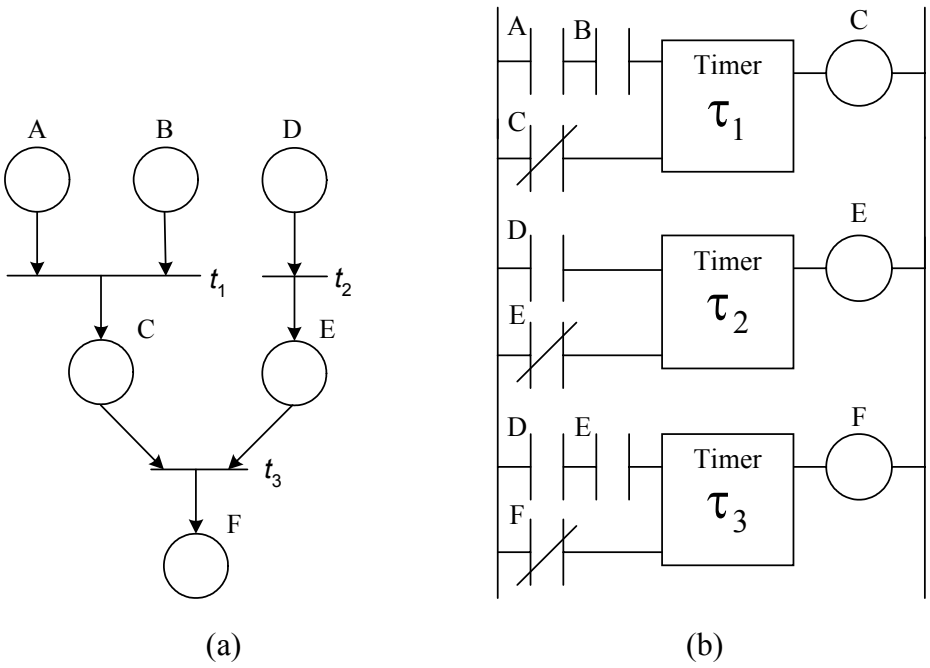


Figura 3.25: Esquema de manufatura.

As disponibilidades das peças são verificadas através de dois sensores A e B, não mostrados na figura. As chaves limite D, E, e F verificam a presença do transportador nos pontos determinados. O sensor C reconhece o carregamento do produto acabado na mesa de trabalho através do Robô. A duração de tempo de ajuntamento pelo Robô e a transferência do produto acabado da mesa de trabalho é τ_1 unidades, τ_2 e τ_3 são as durações de tempo para o transportador se deslocar da estação de armazenamento de pallet à estação de montagem e da estação de montagem para a estação de inspeção, respectivamente.

O modelo de rede de Petri do sistema é mostrado na figura 3.26(a) e na figura 3.27(b) é mostrada a respectiva linguagem ladder para o modelo.



(a) (b)

Figura 3.26: (a) Rede de Petri e (b) Linguagem ladder da figura 3.25.

Considere mais um exemplo para análise entre redes de Petri e linguagem ladder. Suponha que em uma estação de teste exista uma linha de transporte usando detectores para encontrar produtos defeituosos. A condição do produto é conferida quando o produto entrar na estação de teste. Se um defeito é achado no produto, ele é expelido através do acionamento de uma alavanca que faz com o produto defeituoso siga por outro caminho; caso contrário, ele atravessa a linha de transporte e vai para a próxima estação. O mecanismo para este exemplo é ilustrado na figura 3.27.

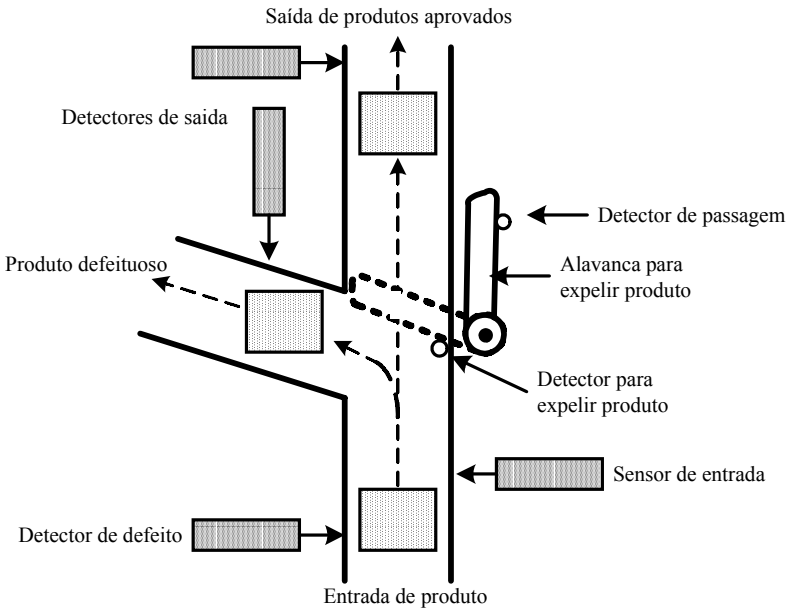


Figura 3.27: Layout do exemplo.

A seqüência de controle para a estação de teste é descrita pelo diagrama da rede de Petri na figura 3.28. Nesta figura, os cinco lugares (P_1 até P_5) representam os estados progressivos operados pela estação de teste, enquanto as seis transições (T_1 até T_6) decidem a sucessão destes estados. Quatro ações (A_1 até A_3) são ativadas respectivamente do estado P_2 até P_5, para executar as ações externas.

A simbologia usada pela linguagem ladder é de um PLC Omron C-200H (OMRON, 1994) e é empregado para implementar a modelagem feita por uma rede de petri. As descrições das transição (condição externa) e ação para a rede de petri. São mostradas nas figura 3.29(a) e 3.29(b) as linguagens ladder que foram convertidas a partir do modelo de rede de Petri da figura 3.28.

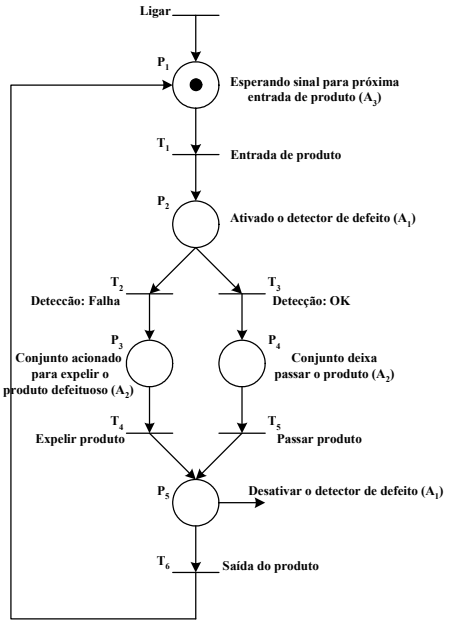
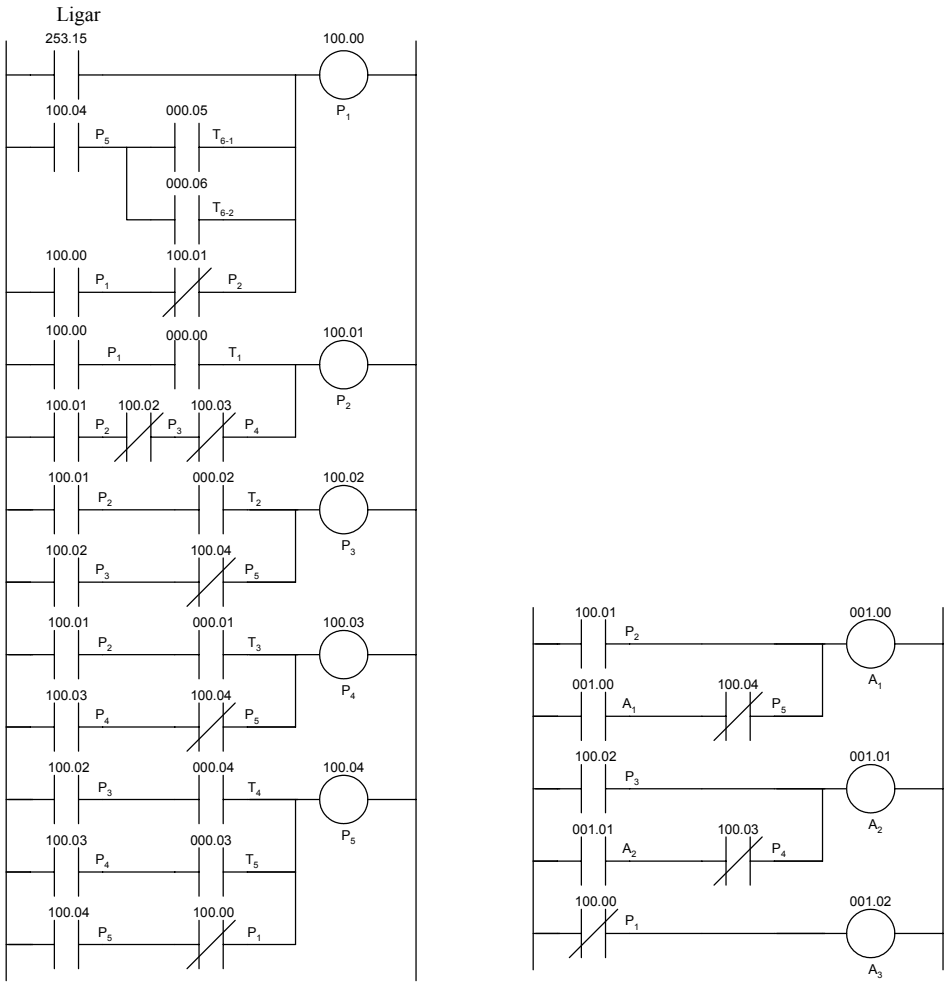


Figura 3.28: Rede de Petri da estação de teste.



(a) (b)

Figura 3.29: (a) Linguagem ladder da seqüência de controle e (b) Linguagem ladder para as ações de saída.

Capítulo 4. Equipamentos do laboratório proposto

Neste capítulo serão descritos os equipamentos que fazem parte do laboratório proposto neste trabalho. São eles: o CLP TSX 3722 (hardware e software), a esteira transportadora, o dispositivo com um conjunto de lâmpadas e o conjunto de chaves de impulso sem retenção. O laboratório, também, possui um computador para a programação do CLP.

Este capítulo está estruturado da seguinte forma: na seção 4.1 é descrito o CLP TSX 37-22 Telemecanique modular, com suas características de entrada e saída bem como algumas de suas vantagens de utilização e operação; na seção 4.2 é visto o software de programação (programa PL7 Micro) projetado especificamente para o CLP TSX 37-22, que operam no ambiente windows, onde é estudada a composição do programa e seu ambiente de programação juntamente com suas ferramentas de manipulação; na seção 4.3 é descrita a esteira transportadora em termos de seus componentes (sensores, painel de controle e o motor interno da esteira); na seção 4.4 é descrito o conjunto de lâmpadas com suas características técnicas e o conjunto de chaves de impulso sem retenção, e também é visto um pequeno esquema de ligação das lâmpadas com o CLP; na seção 4.5 são vistos os esquemas de ligações utilizados nos experimentos do laboratório, isto é, entre o CLP e a esteira transportadora e entre o CLP e o conjunto de lâmpadas.

4.1. CLP TSX 37-22

O CLP TSX 3722 é um CLP Telemecanique modular, com as seguintes características: (i) pode receber até 140 entradas e saídas com bornes para conexão; (ii) não possui módulos de entrada e saída embutidos como padrão; (iii) a memória de programa do usuário pode ser aumentada; (iv) aceita um módulo de comunicação; (v) a alimentação é feita em AC e DC; (vi) possui funções de contagem rápida e analógicas; (vii) a flexibilidade e a praticidade; (viii) permitem a expansão do número de slots para a inserção de módulos através de um rack de extensão, conforme mostrado na figura 4.1.

A alimentação do CLP é feita de modo AC, uma alimentação de 24 V da base do CLP fornece alimentação para seus sensores e para os I/Os de extensão, se

requeridos, desde que o consumo de corrente seja menor ou igual a 400 mA. Se este não é o caso, é necessário o uso de uma alimentação auxiliar de 24 VDC.

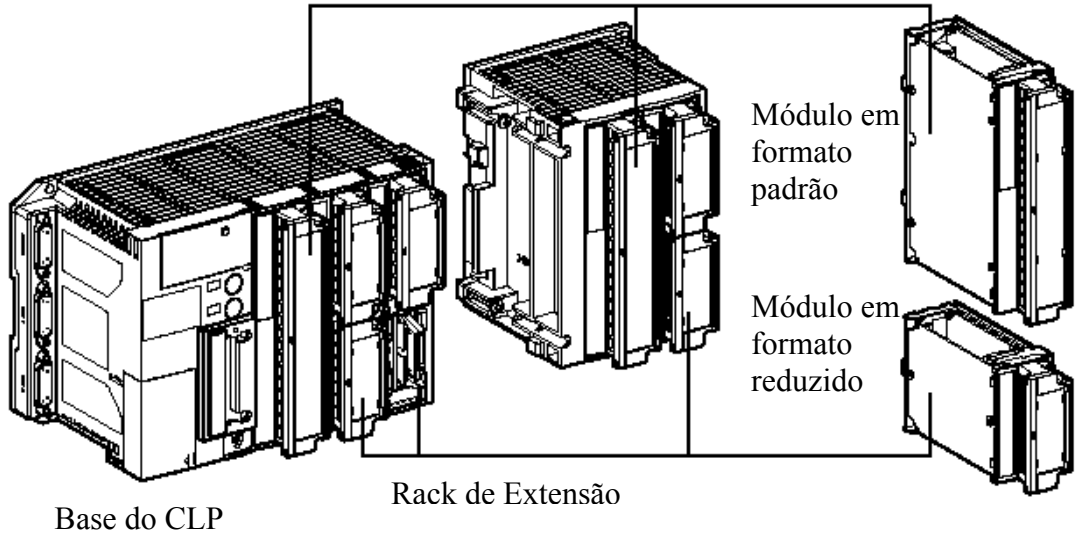


Figura 4.1: CLP com expansão do número de slots.

Os CLPs da linha TSX 37 não possuem módulos de I/O integrados. Qualquer necessidade relacionada a estas é sanada por meio do uso de módulos. Os módulos de I/O discretos diferem-se não somente em seu formato, mas também pela sua modularidade (de 4 saídas a 64 entradas e saídas), pelo tipo de entradas (DC ou AC), pelo tipo de saídas (transistor ou relé) e pelas conexões (bloco de bornes ou conectores HE10). Estas entradas e saídas discretas são utilizadas para o acionamento de atuadores, sensoriamento etc.

O CLP oferece 3 métodos de contagem: (i) usando as entradas discretas do primeiro módulo; (ii) usando canais contadores integrados no CLP TSX 3722 e (iii) usando módulos contadores que podem ser inseridos em qualquer posição disponível.

O CLP TSX 37-22 é o único da linha TSX 37 que possui recursos integrados relativos ao processamento de sinais analógicos. Esse módulo integrado (figura 4.2) é constituído de 8 entradas e 1 saída, com conversores de 8 bits, 0-10V.

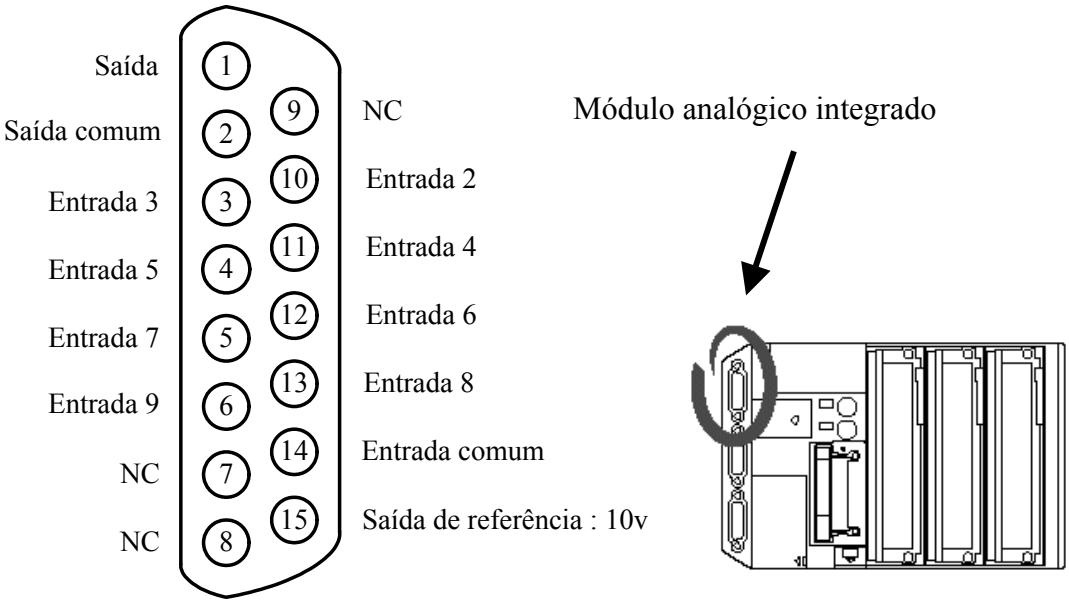


Figura 4.2: Módulo analógico integrado.

A estrutura de memória do CLP TSX 3722 é composta por uma memória RAM onde são executados os programas e uma memória Flash EPROM onde fica armazenada uma cópia de "backup" do programa que está sendo executado. Caso seja necessário um espaço maior de memória, pode-se conectar um cartão de memória PCMCIA, que possibilita um aumento de memória de 32 ou 64 Kwords. Para maior compreensão dessa estrutura de memória observa-se a figura 4.3 (sem cartão de memória PCMCIA). Na parte "Dados" ficam armazenadas as palavras (words) de sistema, funções do FB (contadores, temporizadores, monoestáveis, registradores e "drums"), words internas ou words comuns. Na parte "Programa" é feita a descrição e execução das tarefas pré-definidas. As "Constantes" podem ser valores iniciais ou configuração de I/Os; finalmente o "Backup" é o local onde fica armazenado uma cópia do programa que está sendo executado (memória Flash EPROM). Como se pode observar na figura 4.3, é feito um "backup" através de uma memória Flash EPROM do programa que está sendo executado; na falta de alimentação o programa fica armazenado.

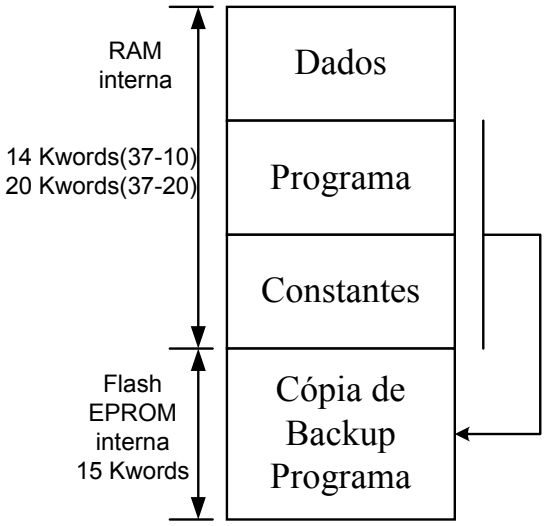


Figura 4.3: Estrutura de memória.

O CLP possui um bloco de visualização dos estados e falhas dos módulos, este bloco centraliza um grupo de serviços que são requeridos para “setup”, operação, diagnóstico e manutenção. Pode ser melhor descrito na figura 4.4. A visualização do estado do CLP é melhor vista na figura 4.5, com os seguintes terminais: 5 LEDs (RUN , TER ,I/O , ERR , BAT).

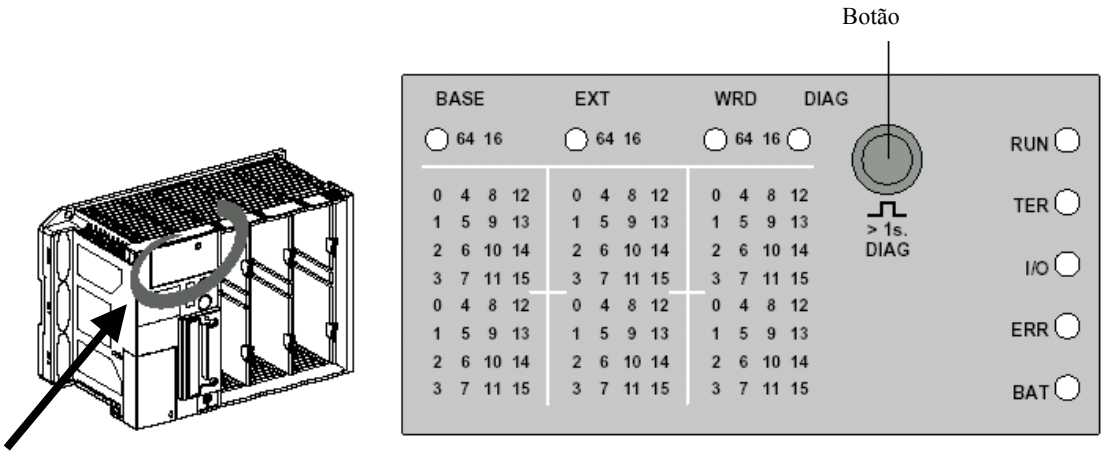


Figura 4.4: Blocos de visualização.

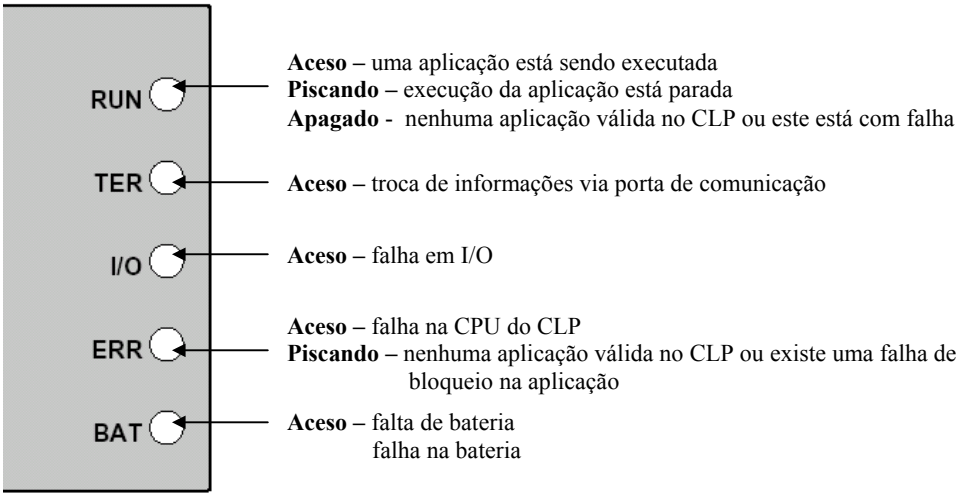


Figura 4.5: Visualização do estado do CLP.

As falhas são apresentadas no modo diagnóstico, acessível por um pressionamento longo (maior que 1 segundo) do botão no bloco de visualização, se uma entrada ou saída está com falha, seu LED correspondente pisca rapidamente. Se um módulo está com falha (módulo faltando, não está de acordo com a configuração, módulo desligado etc), todos os LEDs correspondentes a seu slot piscam lentamente. Este modo permite a visualização de falhas em todos os módulos (módulos de I/O discretos, módulos contadores etc).

O CLP TSX 3722 possui duas portas de comunicação distintas, assinaladas como TER e AUX, que são funcionalmente idênticas (figura 4.6). Elas permitem conexão simultânea de um terminal de programação e de uma interface homem-máquina.

A porta de comunicação assinalada como TER (comum a todos os tipos de CLP da linha TSX 37), pode ser usada para conectar qualquer dispositivo suportando o protocolo UNI-TELWAY, e em particular dispositivos que não tenham sua própria fonte de alimentação (terminal de programação FTX 117, cabo conversor RS 485 / RS 232, caixa de isolação TSX P ACC 01 etc). A porta de comunicação assinalada como AUX (encontrada no TSX 37-22), pode ser usada somente para conectar dispositivos que tenham sua própria fonte de alimentação (painel do operador, CLP etc). A figura 4.7 ilustra o uso das portas de comunicação.

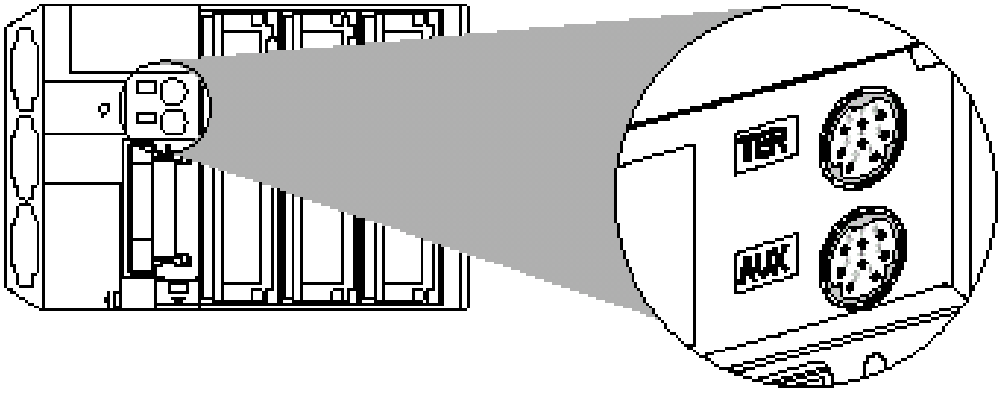


Figura 4.6: Portas de comunicação.

A porta de comunicação do CLP pode manipular dois dispositivos simultaneamente, que podem ser um terminal de programação e um painel do operador.

O CLP TSX 3722 , também, é conectado em redes e barramentos de comunicação por meio de cartões de comunicação PCMCIA. Os padrões físicos suportados pelos cartões são RS-232-D e RS-485 (compatível com o RS 422).

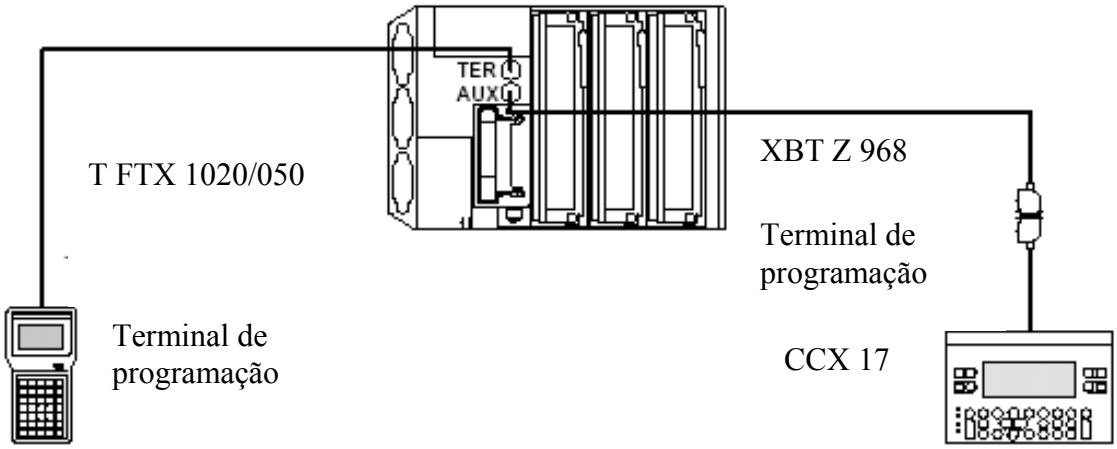


Figura 4.7: Portas de comunicação e os dispositivos que podem ser conectados.

4.2. Linguagem de programação do TSX 3722

O programa PL7 Micro é um software de programação projetado para o CLP TSX 37, operando em ambiente Windows. O programa é composto de: a) Uma linguagem gráfica que é linguagem de contatos com transcrição de esquemas de relés,

que está adaptada ao tratamento combinatório com as seguintes características: (i) oferece símbolos gráficos de base (contatos e bobinas); (ii) a escrita de cálculos numéricos pode ser efetuada nos blocos de operação; e, (iii) uma linguagem booleana que é a linguagem de lista de instruções, sendo uma linguagem de "máquina" com escrita de tratamentos lógicos e numéricos; b) Uma linguagem literal estruturada que é uma linguagem do tipo "informática" com uma escrita estruturada de tratamentos lógicos e numéricos; c) Uma linguagem Grafcet que permite representar graficamente e de forma estruturada o funcionamento de um automatismo seqüencial.

Estas linguagens formulam blocos de funções predefinidas (temporizadores, contadores etc), funções específicas (analógica, comunicação, contagem etc) e funções particulares (controle de tempo, cadeia de caracteres etc). Os objetos da linguagem podem ser simbolizados mediante o editor de variáveis na área de trabalho do programa.

4.2.1. Estrutura de execução das tarefas

Sua estrutura de execução de tarefas possui uma estrutura monotarefa e multitarefa. A Estrutura monotarefa é uma estrutura pré-determinada do programa, contendo uma só tarefa que é a tarefa mestre. A tarefa mestre pode ser executada de forma cíclica (funcionamento pré-determinado) ou periódica. Em funcionamento cíclico, as execuções da tarefa se encadeiam uma com as outras tarefas, sem tempo de espera. No funcionamento periódico, as execuções das tarefas se encadeiam em um período determinado pelo usuário.

As tarefas de um programa PL7 são compostas de várias partes denominadas seções e subprogramas, sendo cada uma das seções com uma linguagem apropriada ao tratamento que se deseja realizar. Esta divisão em seções permite criar um programa estruturado e gerar ou incorporar facilmente os módulos de programa. Pode ser feita uma chamada aos subprogramas de qualquer seção da tarefa à qual pertence ou de outros subprogramas da mesma tarefa.

4.2.2. Ambiente de trabalho do PL7 Micro

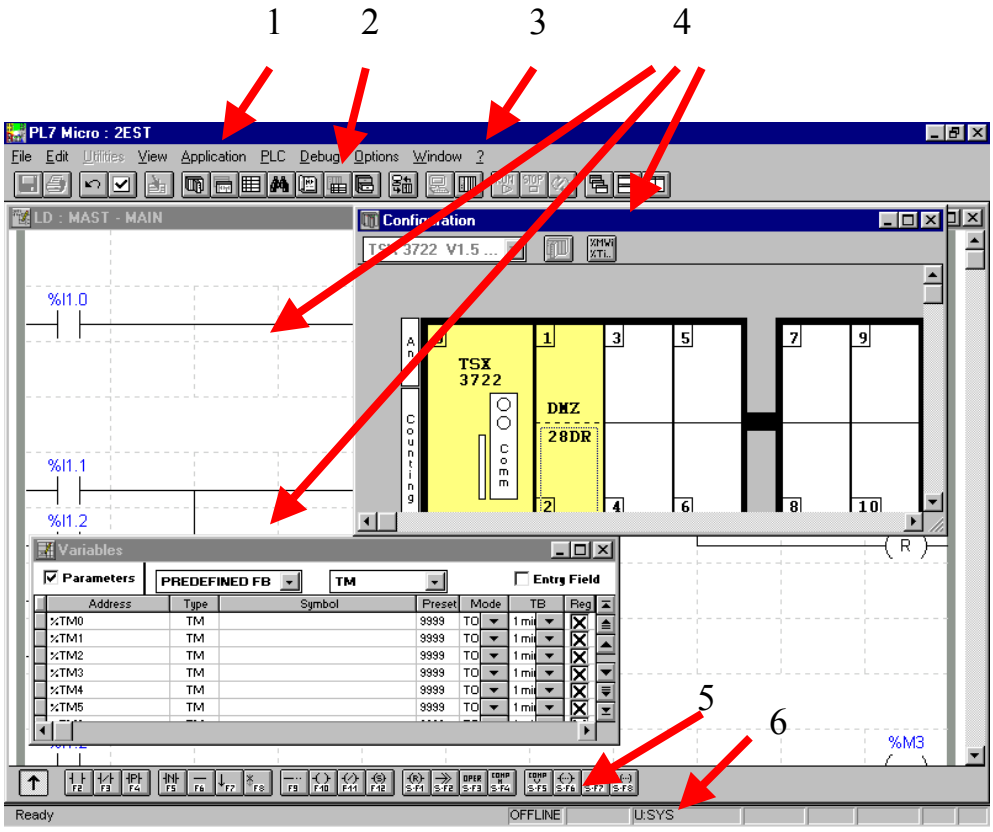


Figura 4.8: Ambiente de programação do PL7 Micro.

O ambiente de programação do PL7 Micro é explicado de maneira simplificada na figura 4.8, onde a numeração corresponde a: barra de Menu, que permite acesso a todas as funções do programa (1), barra de Ferramentas, que oferece acesso rápido mediante o mouse a todas as funções básicas e a todos os editores (2), ajuda, que proporciona informações sobre o programa (3), editores, que permite a criação, depuração e a utilização das aplicações (4), paleta de elementos gráficos, que permite o acesso direto a todas as ferramentas para criação dos programas em linguagem ladder (5) e barra de estado, que fornece um conjunto de informações vinculadas ao estado do programa (6).

4.2.3. Programação

A linguagem de contatos pode mostrar simultaneamente variáveis e símbolos. Os objetos da linguagem podem ser simbolizados usando-se o editor de variáveis na área de trabalho do programa. O usuário pode introduzir e visualizar os objetos

mediante suas variáveis (%M1, %M2), ou mediante uma cadeia de caracteres (máximo 32 caracteres) denominada símbolo (Ligar, Sensor_1, Sensor_2, por exemplo), conforme mostra a figura 4.9.

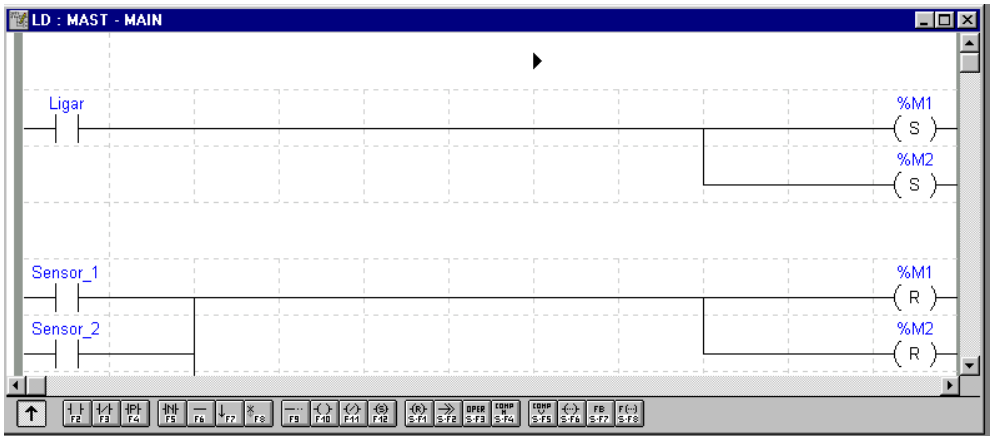


Figura 4.9: Variáveis usadas no programa.

Antes de iniciar-se a programação no software **PL7 Micro**, deve-se considerar de grande importância os parâmetros de comunicação entre o micro computador e o CLP. A opção **UNI-TELWAY**, mostrada na figura 4.10, deve ser configurada de acordo com as características do computador utilizado.

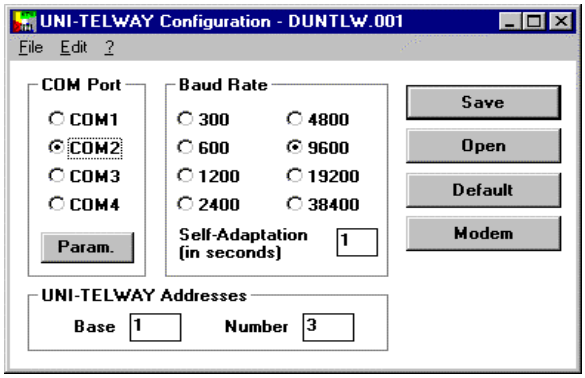


Figura 4.10: Configuração da comunicação.

Quando se deseja criar um novo programa deve-se obedecer a uma seqüência de comandos e selecionar adequadamente as opções em que se desejar executar o novo programa. Quando é pedido pelo software a criação de um novo programa, irá aparecer na tela um quadro (figura 4.11) para que seja selecionado o tipo de CLP e a versão utilizada do programa gerador da linguagem ladder. Neste trabalho será utilizado o CLP

TSX 3722 versão 1.5 do software. Note que, no mesmo quadro aparece a opção para ser utilizada a linguagem Grafcet, deve-se clicar na opção **No**, pois a linguagem a ser utlizada será a linguagem ladder, a ser selecionada mais adiante. As opções de 32K ou 64K, mostradas na figura 4.11, só podem ser selecionadas quando existir a placa PCMCIA para a expansão de memória. No caso do laboratório proposto neste trabalho, o modelo utilizado não possui esta placa; portanto, deve ser selecionada a opção **None**.

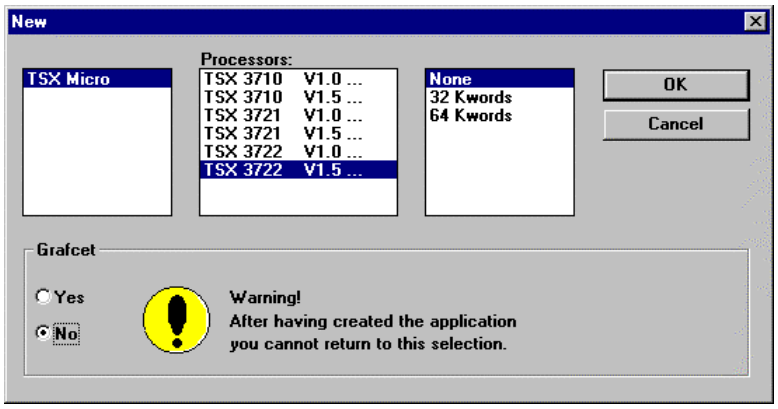


Figura 4.11: PLC e versão utilizada do software.

Finalizando a seqüência anterior, o programa permite a seleção do tipo de programa a ser utilizado. Para tanto, as opções **MAST** e **MAIN** serão utilizadas, conforme ilustrado na figura 4.12. Após esta seleção, uma última tela aparecerá para que possa ser selecionado o tipo de linguagem a ser utilizada, conforme mostra a figura 4.13. Deve-se, então, selecionar a opção Ladder (LD). Após todos esses procedimentos o ambiente de programação estará disponível para ser utilizado, conforme mostrado na figura 4.14.

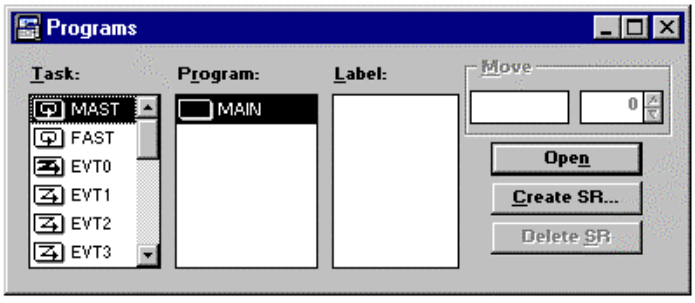


Figura 4.12: Tipos de programas que podem ser gerados.

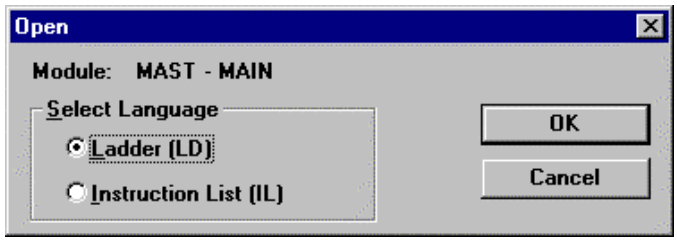


Figura 4.13: Seleção da linguagem a ser utilizada.

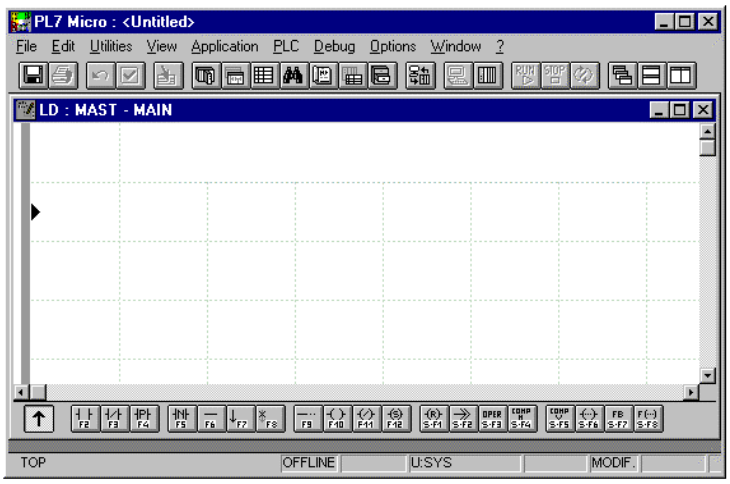


Figura 4.14: Ambiente de programação.

Deste ponto em diante, pode-se começar a construir um programa neste ambiente de programação que será transferido e utilizado no CLP. Deste ambiente de programação, é importante descrever o palete para construção das aplicações em linguagem ladder, que é completamente detalhado na figura 4.15. Contudo, um passo importante ainda é necessário, que é a configuração do tipo de CLP a ser utilizado. Deve ser lembrado que o CLP possui disponível 16 entradas e 12 saídas no módulo de I/O. As instruções utilizadas pelo CLP possuem o seguinte formato: %I é a entrada do CLP; %Q é a saída do CLP; %M é um bit Interno; %S é um bit do Sistema e %BLK é um bit dos blocos de funções. Podem-se utilizar entradas de %I1.0 até %I1.15 para ocupar as 16 entradas e de %Q2.0 até %Q2.11 para as 12 saídas disponíveis no CLP. Outros elementos importantes são: as funções booleanas, que podem ser declaradas através de um endereço de memória qualquer (bit interno) ou através de módulos de I/O digitais (bit de I/O). As instruções booleanas têm um símbolo correspondente e um operando pode ser associado a este símbolo, tal como é ilustrado na figura 4.16.

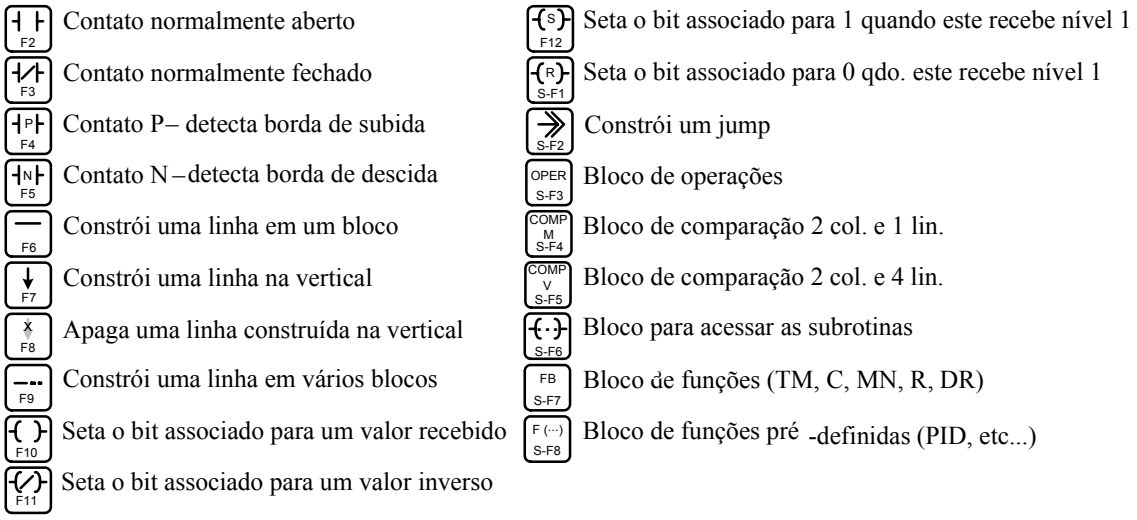


Figura 4.15: Palete para construção da aplicação em ladder.

Símbolo	Operando
┤ ├	%I, %Q, %M, %S, %BLK
┤/├	%I, %Q, %M, %S, %BLK
┤P├	%I, %Q, %M
┤N├	%I, %Q, %M
-( )-	%Q, %M, %FBs
-(/)-	%Q, %M, %FBs
-(S)-	%Q, %M, %FBs
-(R)-	%Q, %M, %FBs

Figura 4.16: Símbolos e operandos correspondentes.

O CLP utilizado nesse laboratório é o TSX 3722, que possui (para desenvolvimento deste trabalho) somente a opção do módulo TSX DMZ 28 DR que é um módulo digital com 16 entradas e 12 saídas que está inserido nas posições 1 e 2 do rack do CLP.

4.2.4. Criando um programa em linguagem ladder no PL7 Micro

Para implementar o programa em linguagem ladder mostrado na figura 4.17, deve-se proceder da seguinte maneira:

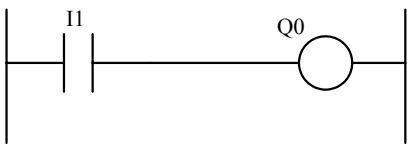


Figura 4.17: Linguagem ladder.

Estando o programa no ambiente de programação, conforme mostrado na figura 4.18, clique na função F2 (no teclado do computador) ou arraste o mouse até o ponto indicado pela seta "A" mostrada na figura 4.18 e clique no botão esquerdo do mouse; a partir deste momento, quando o mouse for arrastado, a figura da função selecionada aparecerá ao lado da seta do mouse no monitor. Arraste o mouse até o ponto de início do programa indicado por "B" que é a célula inicial. Neste momento irá aparecer o contato NA selecionado e um retângulo aparecerá sobre o contato pedindo que seja especificada a descrição do contato, conforme figura 4.19, neste caso deve-se digitar %I1.0. Após digitar I1.0 aperte no botão **ENTER** do teclado. O resultado é mostrado na figura 4.20, estando, portanto, criada a chave NA referente ao circuito da figura 4.17.

Para criar a saída, clique com o botão direito do mouse sobre o ponto indicado pela seta C (figura 4.18); a partir deste momento acontece o mesmo que na criação da chave NA, a figura referente a saída aparecerá ao lado da seta do mouse. Arraste o mouse para posicionar ao lado da figura da chave NA %I1.0 e clique com o botão do lado direito do mouse. O resultado é mostrado na figura 4.21. Digite, então, o tipo de saída desejada, neste caso deve-se digitar Q2.0 e apertar, agora, duas vezes na tecla **ENTER** do teclado do computador. O resultado é mostrado na figura 4.22, o programa está criado e pronto para ser transferido para o PLC.

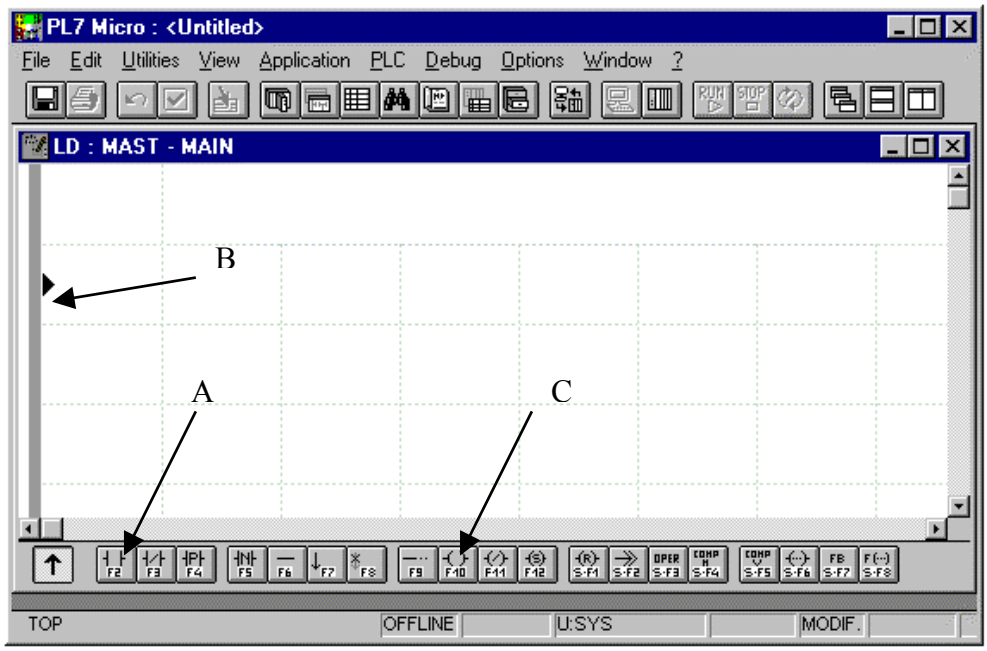


Figura 4.18: Ambiente de programação.

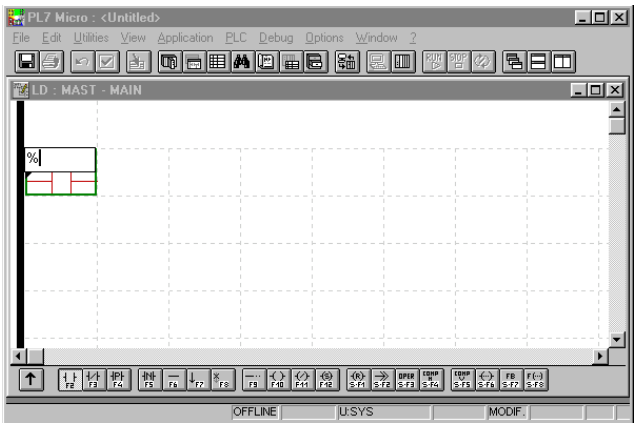


Figura 4.19: Início do programa: especificar o contato NA.

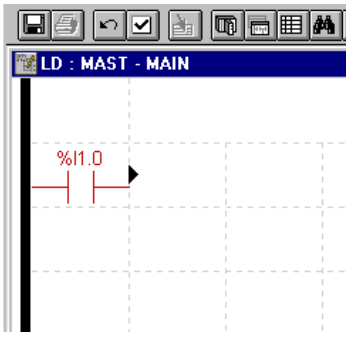


Figura 4.20: Contato NA especificado.

Figura 4.21: Criando e especificando uma saída.

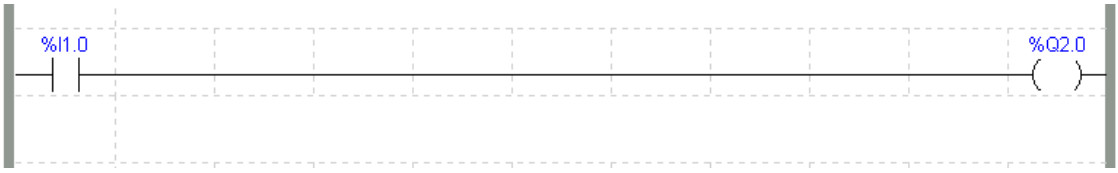


Figura 4.22: Programa completo.

Após ser criado o programa de aplicação, o passo seguinte é a transferência para o CLP. No menu principal deve-se clicar com o mouse em **PLC**. Com isso serão mostradas várias opções, devendo, então, clicar em **Tranfer...**. Pode acontecer de o arquivo necessitar ser salvo antes de ser tranferido. Se isso acontecer, uma tela irá aparecer solicitando tal procedimento. Em seguida, clica-se novamente em **PLC** e posteriormente em **Connect**. Novamente clica-se em **PLC** e em seguida em **Run...**. Agora, sim, o programa é transferido e está sendo executado pelo CLP.

4.3. Esteira transportadora

Figura 4.23: Conjunto esteira transportadora.

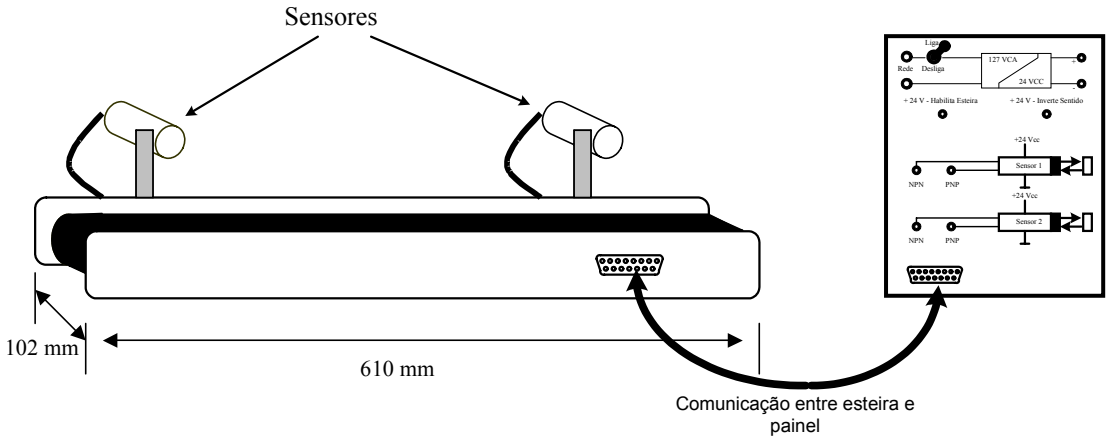


Figura 4.24: Esteira transportadora e detalhes de esteira e painel.

O conjunto esteira transportadora[1] (figura 4.23) é composto de uma esteira deslizante feita de lona, dois sensores PNP, um motor interno para o arrasto da esteira e um painel para ligação dos componentes da esteira com o CLP.

O painel de controle, ou ligação, faz a conexão entre a esteira e o CLP. Quando os pontos de +24 volts, que habilitam a esteira ou inverte o sentido do movimento, são alimentados via CLP, a esteira tem seu funcionamento iniciado. Os sinais de atuação dos sensores são retirados dos pontos NPN e PNP do painel e levados aos terminais de entrada de sinal do CLP para realização da lógica de controle. A figura 4.25 ilustra

1- Um dos fabricantes desse módulo é a INTERDIDACTIC Sistemas Educacionais LTDA. Rua Oriçanga, 217 CEP 04 052 030 – São Paulo – SP Telefax 011 5078 9910 www.interdidatic.com.br Email: comercial@interdidactic.com.br

detalhadamente o painel. Internamente o painel é constituído por um transformador de 127 volts para 24 volts e uma placa eletrônica para realizar a inversão de alimentação do motor, isto é, para que o motor possa inverter o sentido de rotação.

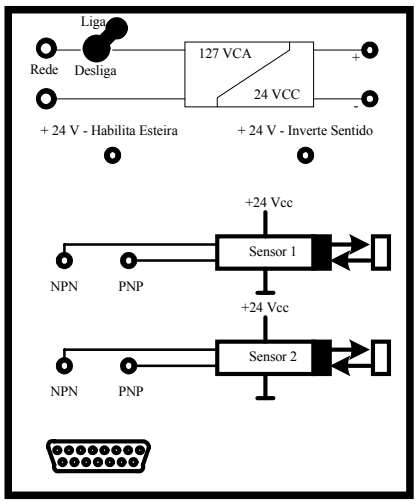


Figura 4.25: Painel de conexão entre esteira e CLP.

O Motor é de fabricação da Buehler Products, INC. É um motor reversível com tensão de alimentação de 24 volts, torque nominal aproximado de 150 mN.m, rotação de 121 rpm (carga) e 160 rpm (sem carga), corrente nominal de 250 mA. Fica localizado dentro da esteira, e faz a correia transportadora girar pelo acionamento de um cilindro interligado a seu eixo por engrenagens, como ilustrado pela figura 4.26.

Os sensores de esteira são do tipo PNP com tensão de alimentação entre 10 e 30Vcc, corrente aproximada de 150mA, alcance de detecção entre 10cm e 30cm e temperatura de operação entre –25ºC e +55ºC. Sua forma de conexão é ilustrada pela figura 4.27.

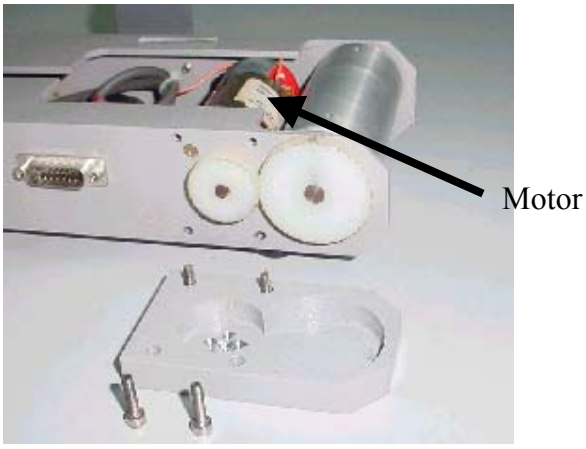


Figura 4.26: Motor da esteira, transmissão feita por engrenagens.

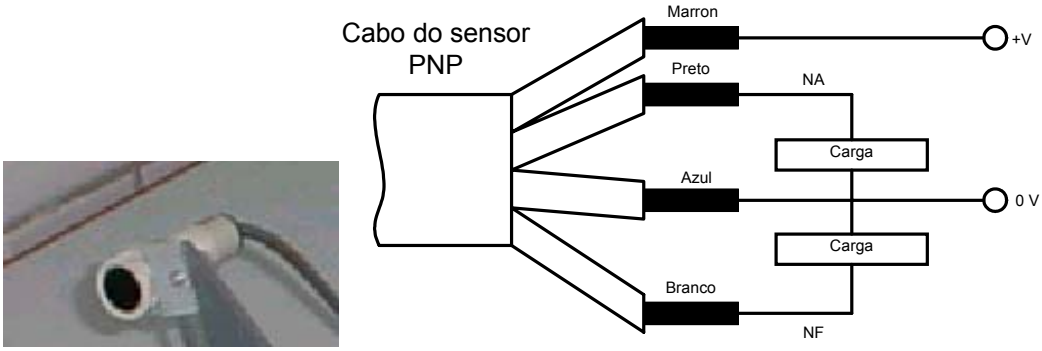


Figura 4.27: Sensor e conexão do cabo do sensor.

4.4. Conjunto de lâmpadas e chaves de impulso sem retenção

O conjunto de lâmpadas é composto por 9 lâmpadas coloridas (3 vermelhas, 3 verdes e 3 amarelas) de características 130V/15W. A figura 4.28 mostra o conjunto completo, a figura 4.29 ilustra os pontos de alimentação de 127 V e 24 VDC. Cada lâmpada é acionada individualmente por um relé, ilustrado na figura 4.30, que recebe a tensão de 24 VDC enviada pelo CLP, de acordo com a lógica de controle. Finalmente a figura 4.31 indica, de forma simplificada, como é realizada a conexão entre o conjunto de lâmpadas e o CLP.

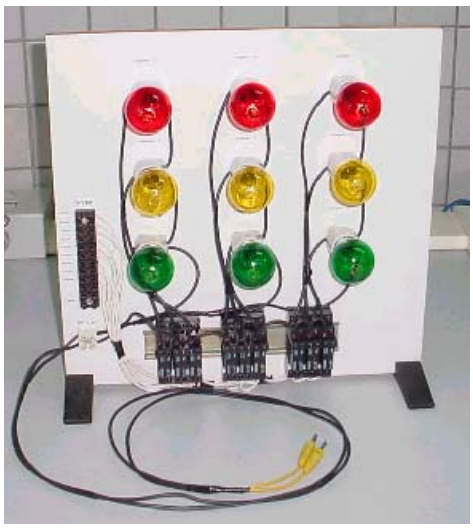


Figura 4.28: Conjunto de lâmpadas.

Figura 4.29: Pontos de alimentação 127 V da rede elétrica e 24 VDC via CLP.

Figura 4.30: Relés de chaveamento para acionamento das lâmpadas.

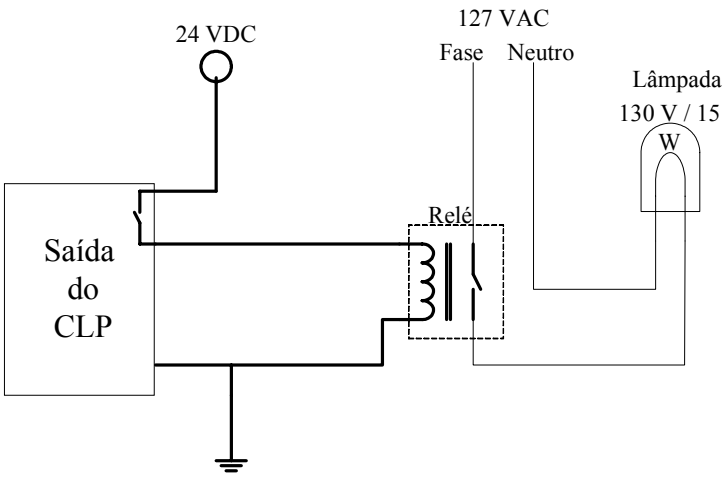


Figura 4.31: Esquema simplificado para acionamento de uma lâmpada via CLP.

O conjunto de chaves é formado por 8 chaves de impulso sem retenção. A figura 4.32 mostra como é formado o esquema do conjunto de chaves.

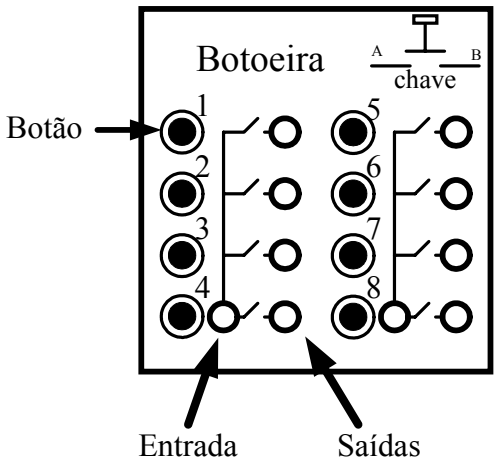


Figura 4.32: Conjunto de chaves de impulso sem retenção.

4.5. Esquemas de ligações utilizados nos experimentos

A figura 4.33 mostra o esquema de ligação do módulo TSXDMZ 28DR e as figuras 4.34 a 4.39 mostram os esquemas de ligação entre o CLP e a esteira e entre o CLP e o conjunto de lâmpadas.

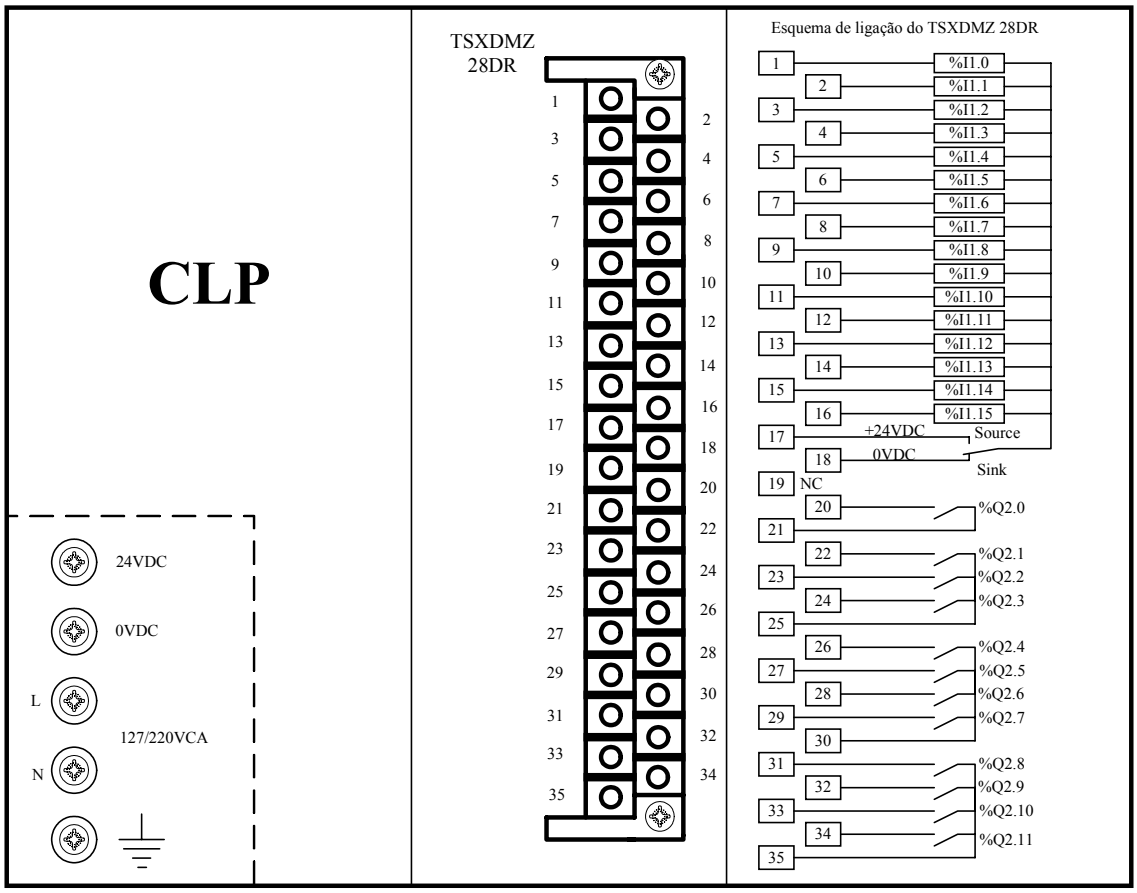


Figura 4.33: Esquema de ligação do módulo TSXDMZ 28DR do CLP TSX 37-22.

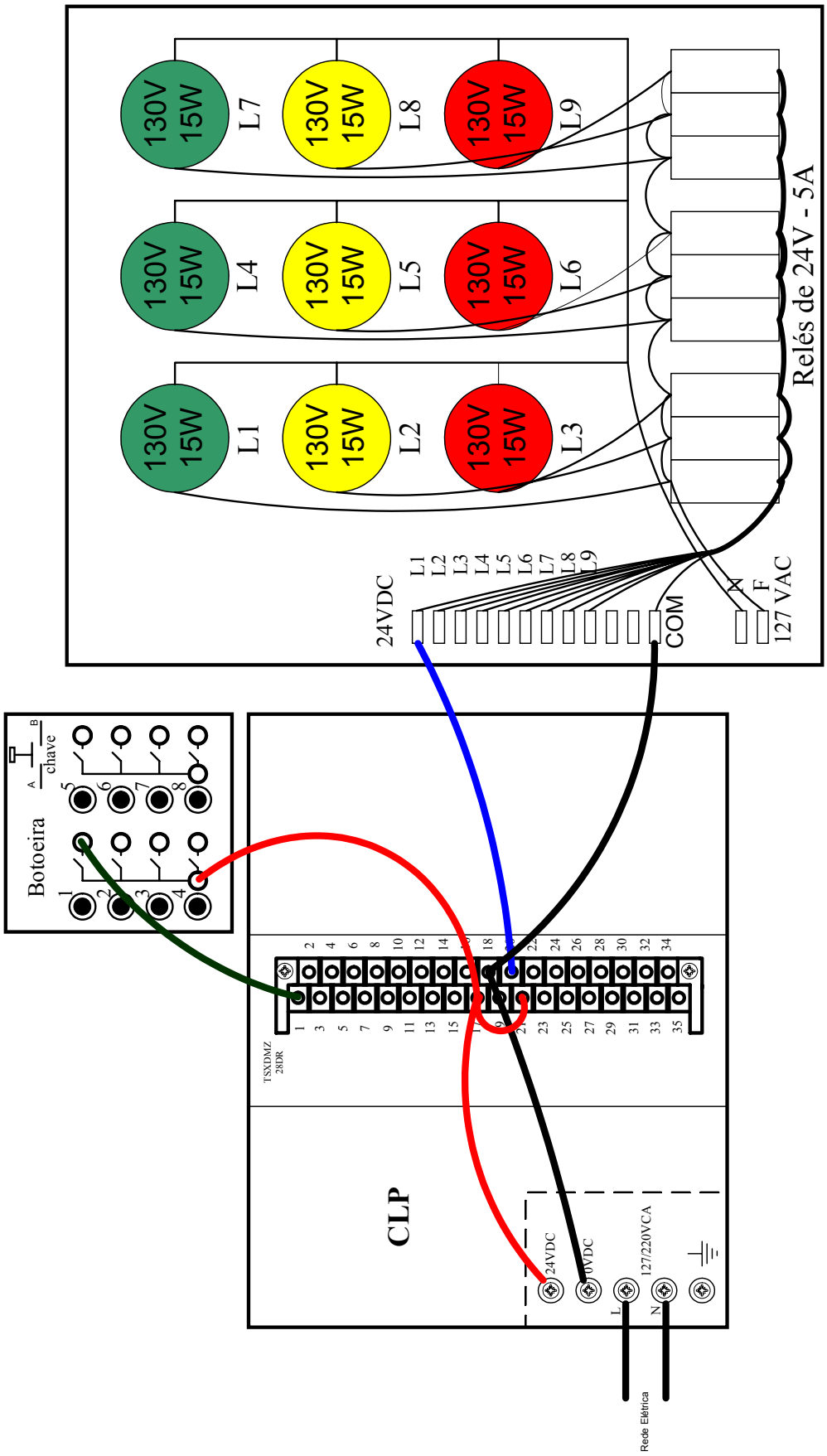


Figura 4.34: Esquema de ligação dos experimentos 1 e 2.

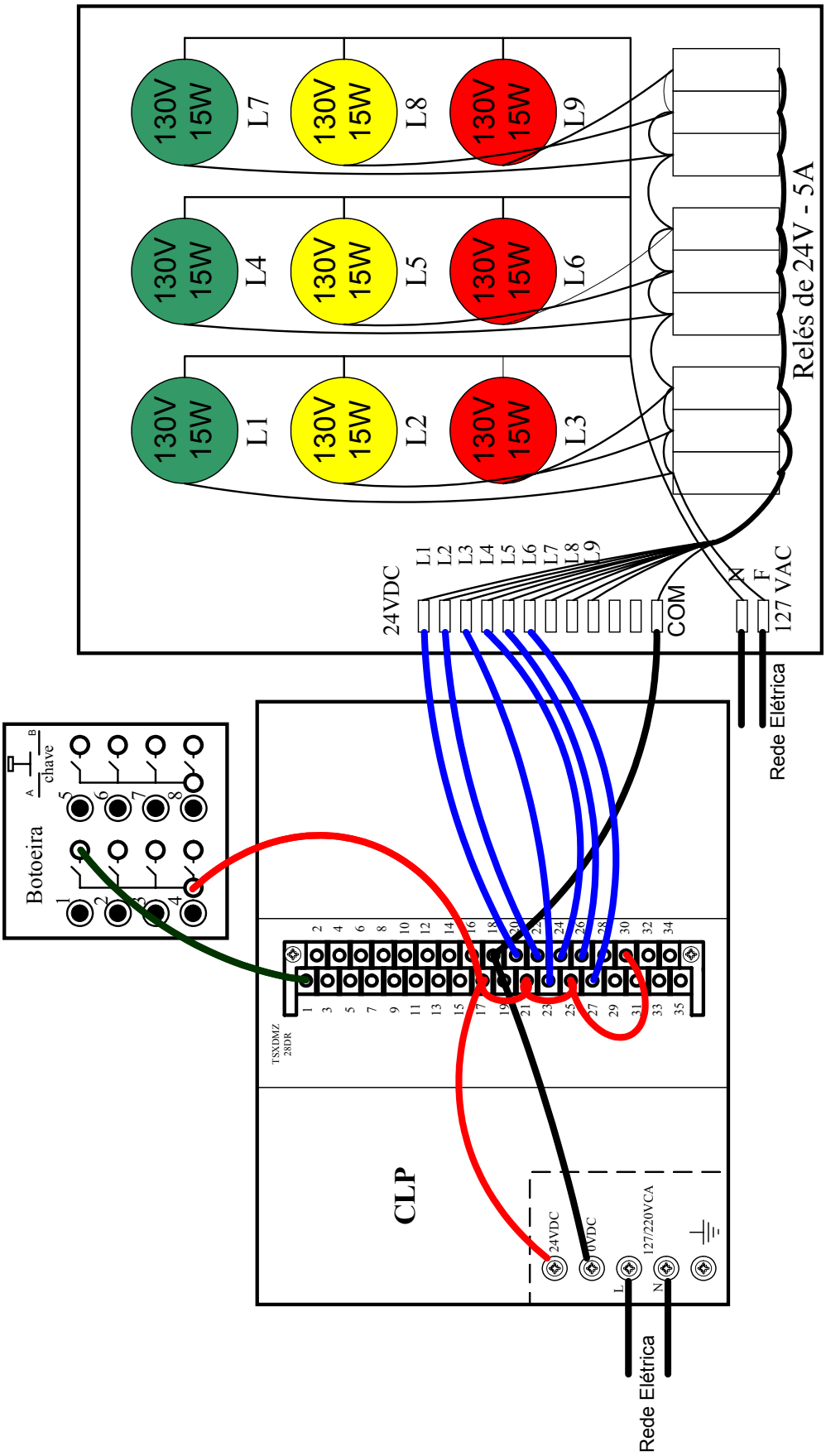


Figura 4.35: Esquema de ligação do experimento com os semáforos (experimento 3).

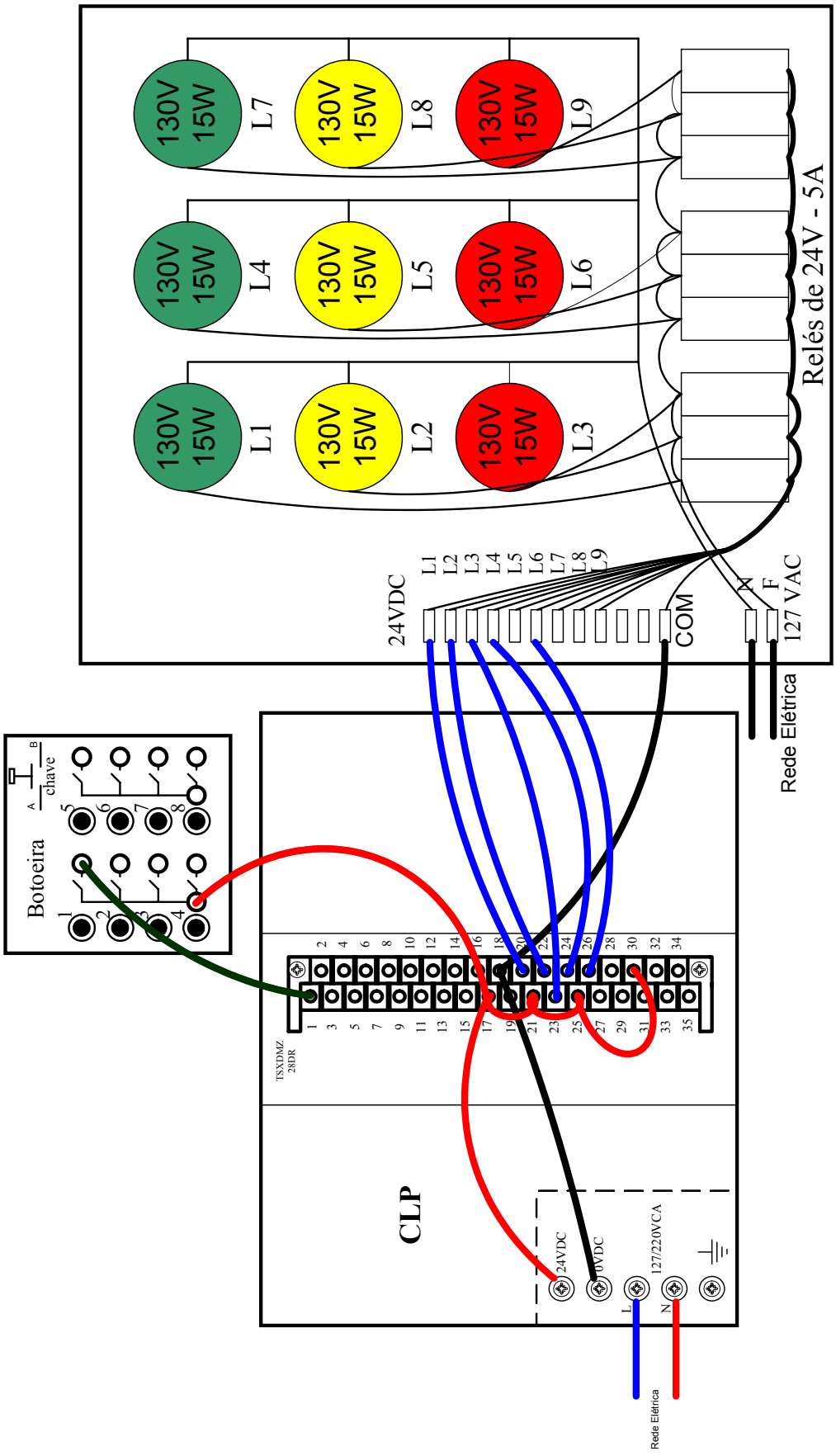


Figura 4.36: Esquema do experimento do semáforo da passagem de pedestre (experimento 4).

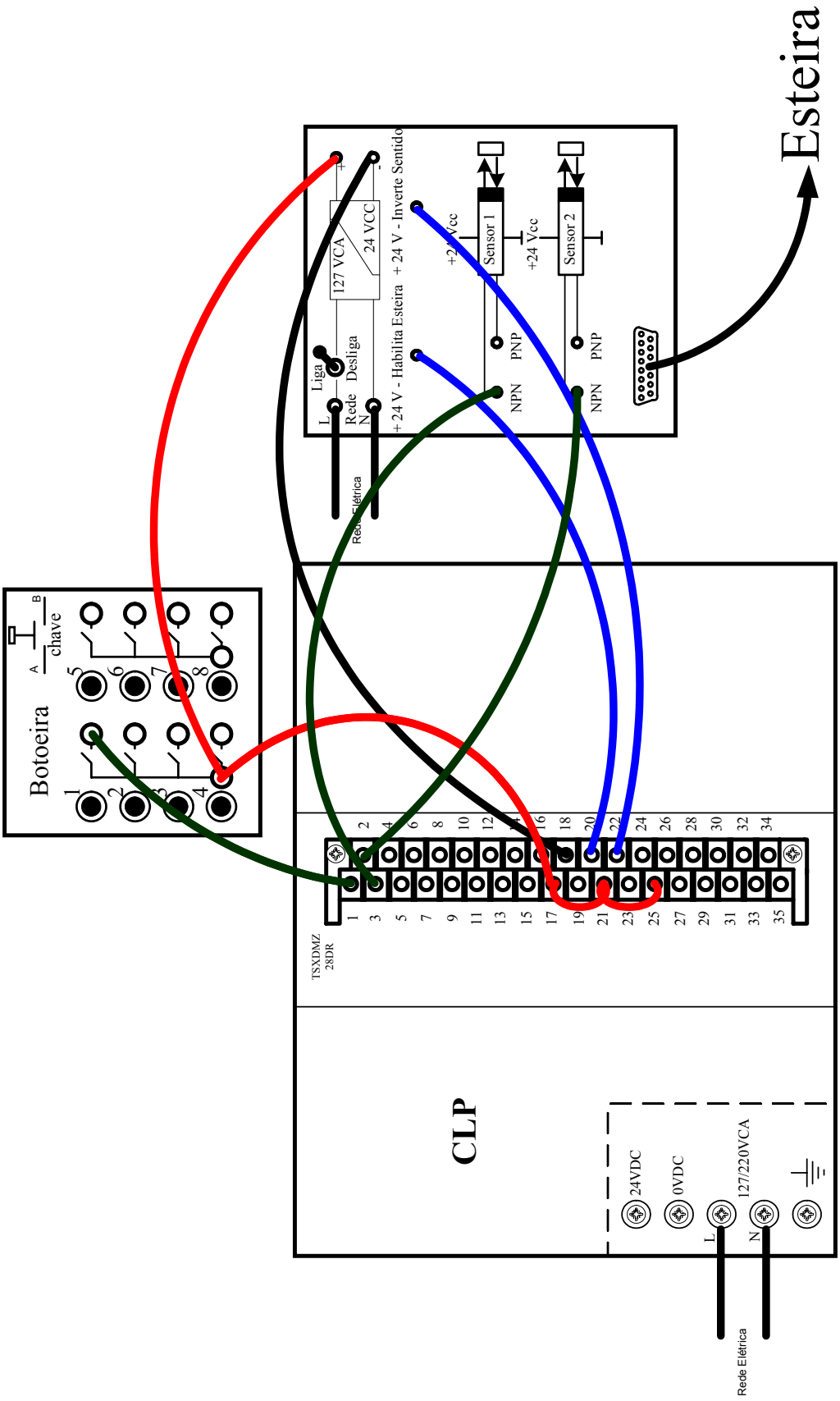


Figura 4.37: Esquema de ligação do experimento com 1 esteira e 2 sensores (experimento 5).

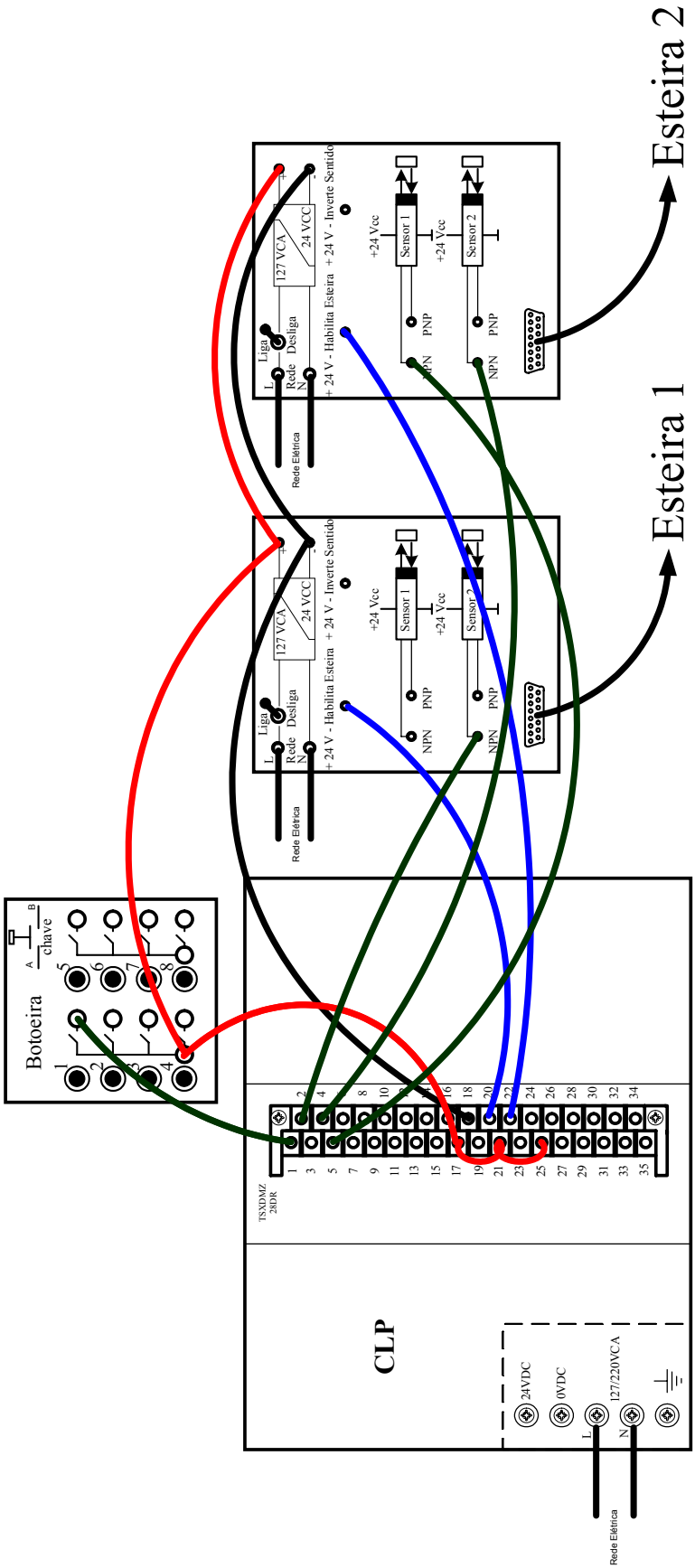


Figura 4.38: Esquema de ligação do experimento com 2 esteiras e 3 sensores (experimento 6).

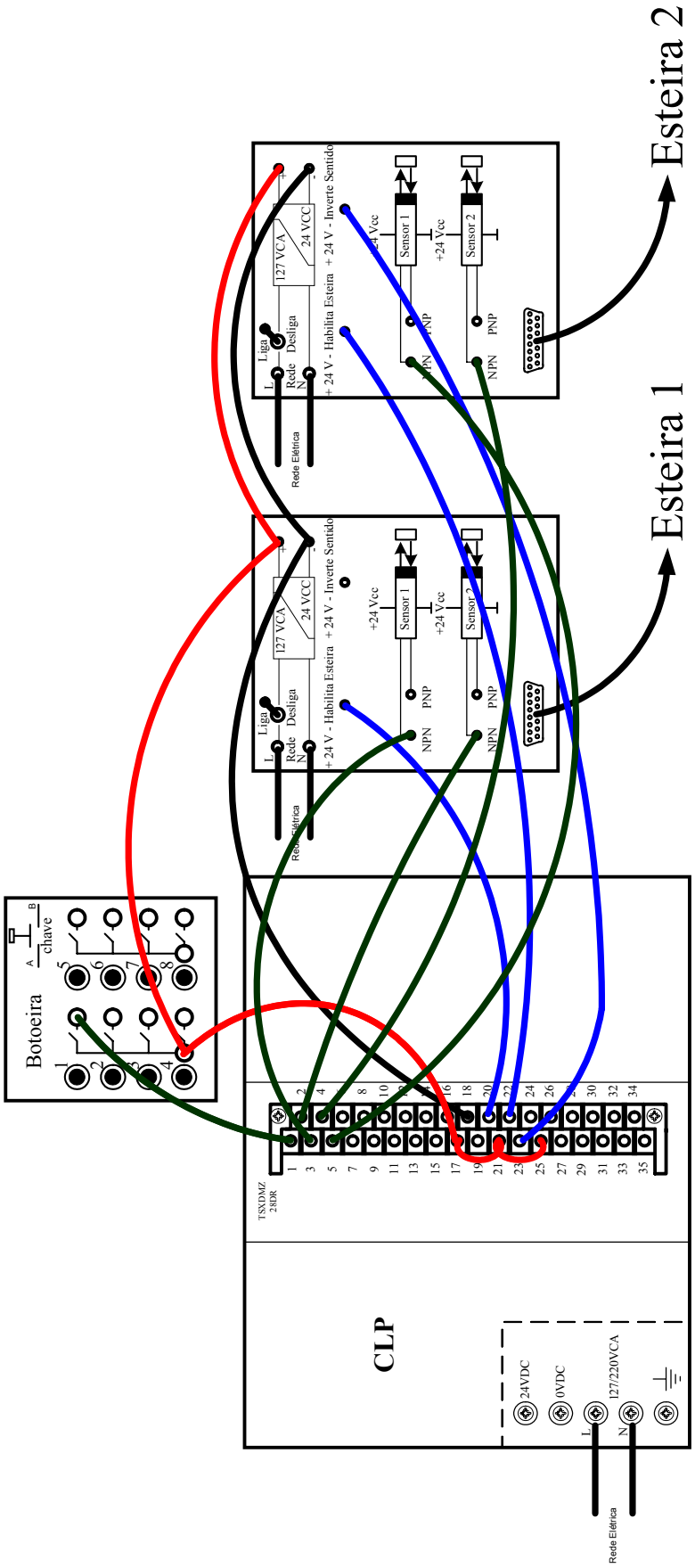


Figura 4.39: Esquema de ligação do experimento com 2 esteiras e 4 sensores (experimento 7).

Capítulo 5. Experimentos do laboratório

Neste capítulo serão propostos sete experimentos para o laboratório de automação industrial, os dois primeiros experimentos implantados utilizarão os recursos de temporização e contagem do CLP. Os três experimentos seguintes utilizam esteiras e os dois últimos utilizam um conjunto de lâmpadas para ilustrar o funcionamento de semáforos (CASSANDRAS e LAFORTUNE, 2000, pág. 47).

De acordo com o capítulo 2, as redes de Petri serão redes marcadas (*P*, *T*, *A*, *w, x)* com um conjunto V = $\{v_j : t_j \in T_D\}$ que é uma estrutura de tempo e que $v_j = \{v_{j,1}, v_{j,2}, ...\}$, $t_j \in T_D$, $V_{j,k} \in R^+$, $k = 1, 2, ...$ Graficamente, as transições sem atraso de disparo serão representadas por barras, ao passo que transições temporizadas serão representadas por retângulos; os sensores terão sua representação em forma de barras, mas sua condição de disparo é dependente da detecção do objeto. Serão mostradas as dinâmicas da rede decompondo o modelo dentro das dinâmicas de transições individuais (capítulo 2, seção 2.4.1), usando a notação $\pi_{i,k}$ para o instante de tempo quando o lugar p_i recebe seu *k*-ésimo disco e $\tau_{j,k}$ é o *k*-ésimo tempo de disparo da transição t_j.

Serão ainda mostrados em alguns experimentos os estados dos componentes, simbolizados com ON para as situações de estado ligado, detectando, acesa (lâmpada) etc e OFF para as situações de desligado, não detectando, apagada (lâmpada) etc; para a esteira será considerado que ela está avançando quando o seu movimento for no sentido de A para B, e retrocedendo, quando o seu movimento for no sentido de B para A, conforme ilustrado pela figura 5.1. Também, serão utilizadas as redes de Petri como formalismo de modelagem para tais experimentos e a implementação no CLP será efetuada em linguagem ladder.

Este capítulo está estruturado da seguinte forma: na seção 5.1 é visto a utilização dos recursos de temporização e contagem do CLP; na seção 5.2 é visto controle de tráfego de veículos e o controle de uma passagem de pedestre; na seção na seção 5.3 é implementado uma seqüência de comandos de controle para: (i) uma esteira com dois sensores; (ii) duas esteiras e três sensores; e (iii) duas esteiras e quatro sensores.

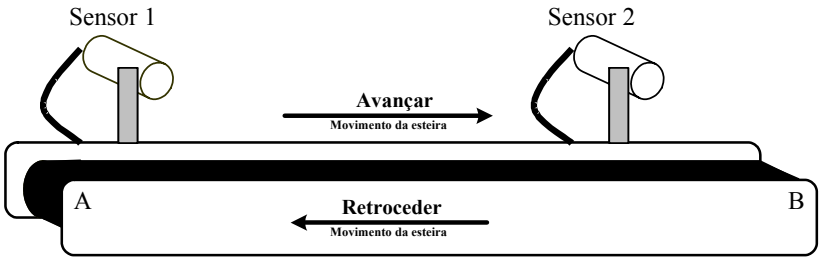


Figura 5.1: Movimentos da esteira.

5.1. Experimentos básicos para familiarização com o CLP

Estes experimentos consistem em ligar e desligar uma lâmpada utilizando os recursos do CLP.

(*i*) Primeiro experimento

O primeiro experimento consiste em ligar e desligar uma lâmpada utilizando os recursos de temporização do CLP. Quando o botão é acionado, uma lâmpada ou um conjunto de lâmpadas acende por um determinado período de tempo e depois é desligada automaticamente. Uma forma de modelagem deste experimento é através de redes de Petri e, para tanto devem ser definidos os elementos da sêxtupla (P, T, A, w, x, V). Note que os eventos são: (i) acionar um botão; (ii) enquanto a lâmpada estiver acesa o acionamento do botão é desabilitado; (iii) acionar as lâmpadas e (iv) apagar as lâmpadas, definindo assim o conjunto de transições $T = \{t_1, t_2, t_3, t_4\}$. Os estados da lâmpada são ON e OFF e dentro da idéia de que os lugares denotam as condições para que um evento possa ocorrer, então $P = \{p_1, p_2, p_3, p_4\}$, onde: p_1 representa a habilitação para ligar a lâmpada, p_2 a lâmpada acesa, p_3 a lâmpada apagada e p_4 desabilita o acionamento do botão enquanto a lâmpada estiver ligada, conforme é ilustrada na figura 5.2. É fácil ainda verificar que $A=\{(p_1, t_2), (p_1, t_4), (p_2, t_3), (p_3, t_2), (p_4, t_3), (p_4, t_4), (t_1, p_1), (t_2, p_2), (t_2, p_4), (t_3, p_3), (t_4, p_4)\}$, $w(p_1, t_2)=1$, $w(p_1, t_4)=1$, $w(p_2, t_3)=1$, $w(p_3, t_2)=1$, $w(p_4, t_3)=1$, $w(p_4, t_4)=1$, $w(t_1, p_1)=1$, $w(t_2, p_2)=1$, $w(t_2, p_4)=1$, $w(t_3, p_3)=1$, $w(t_4, p_4)=1$. Finalmente; para este experimento, será adotado V=$\{\underline{v}_3\}$ e $\underline{v}_3=\{v_{3,1}, v_{3,2}, v_{3,3}, ...\}=\{10, 10, 10, ...\}$.

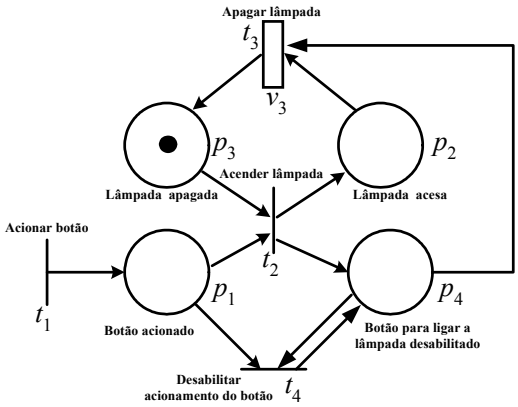


Figura 5.2: Rede de Petri para ligar a lâmpada primeiro experimento.

Suponha ainda que o estado inicial desta rede de Petri é $\underline{x}_0$= [0, 0, 1, 0], isto é, a lâmpada se encontra apagada e o botão ainda não foi apertado.

Para obter a dinâmica desta rede de Petri, deve-se inicialmente encontrar a matriz de incidência **A**, que será uma matriz 4x4 cujos elementos *(j, i)* são da forma $a_{ji}=w(t_j, p_i) - w(p_i, t_j)$, sendo, portanto, dada por:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

Usando a matriz de incidência **A**, pode-se, a partir da equação de estados $\underline{\boldsymbol{x}}' = \underline{\boldsymbol{x}} + \mathbf{uA}$, simular os resultados de todos os próximos estados que podem ser gerados a partir do disparo das transições.

Com o disparo de t_1 tem-se u_1=[1, 0, 0, 0], então:

$\underline{x}_1= \underline{x}_0 + \underline{u}_1.A$, $\underline{x}_1$ = [0, 0, 1, 0] + [1, 0, 0, 0].**A** = [1, 0, 1, 0]

A partir do disparo de t_1, a transição t_2 fica habilitada, disparando imediatamente. Assim, a lâmpada acenderá e portanto:

$\underline{x}_2= \underline{x}_1 + \underline{u}_2.A$, $\underline{x}_2$= [1, 0, 1, 0] + [0, 1, 0, 0].**A** = [0, 1, 0, 1]

$\underline{x}_3= \underline{x}_2 + \underline{u}_3.A$, $\underline{x}_3$= [0, 1, 0, 1] + [0, 0, 1, 0].**A** = [0, 0, 1, 0] = $\underline{x}_0$

Se t_1 for disparada antes do final do ciclo, t_4 dispara desabilitando o acionamento do botão, tem-se:

$\underline{x}'_2$= [0, 1, 0, 1] + [1, 0, 0, 0].**A** = $\underline{x}'_2$= [1, 1, 0, 1]

$\underline{x}'_3$= [1, 1, 0, 1] + [0, 0, 0, 1].**A** = $\underline{x}'_3$= [0, 1, 0, 1] = $\underline{x}_3$

Lembrando que $\pi_{i,k}$ denota o instante em que o lugar p_i recebe seu k-ésimo disco e que $\tau_{j,k}$ é o k-ésimo tempo de disparo da transição t_j, então para este experimento pode-se decompor o modelo dentro das dinâmicas de transições individuais, que são:

Equações dos instantes de tempo que os lugares recebem os discos (1).

$$\pi_{1,k} = \tau_{1,k}, \qquad k = 1, 2, ...$$

$$\pi_{2,k} = \tau_{2,k}, \qquad k = 1, 2, ...$$

$$\pi_{3,k} = \tau_{3,k\text{-}1}, \qquad k = 2, 3, ..., \quad \pi_{3,1} = 0$$

$$\pi_{4,k} = \tau_{2,k}, \qquad k = 1, 2, ...,$$

Equações dos instantes de tempo de disparo das transições (2).

$$\tau_{1,k} = \pi_{1,k}, \qquad k = 1, 2, ...$$

$$\tau_{2,k} = \max\{\pi_{1,k}, \pi_{3,k}\}, \qquad k = 1, 2, ...$$

$$\tau_{3,k} = \max\{\pi_{2,k}, \pi_{4,k}\} + v_{3,k}, \qquad k = 1, 2, ...,$$

$$\tau_{4,k} = \max\{\pi_{1,k}, \pi_{4,k}\}, \qquad k = 1, 2, ...,$$

As equações acima podem ser arranjadas, levando à seguinte forma recursiva e desprezando $\tau_{4,k}$, pois é apenas uma condição para desabilitar o botão e não influencia no tempo do ciclo, tem-se:

$$\tau_{1,k} = \pi_{1,k},$$

$$\tau_{2,k} = \max\{\pi_{1,k}, \pi_{3,k}\}, \text{ com condição inicial } \pi_{3,1} = 0,$$

$$\tau_{3,1} = \pi_{3,k} + v_{1,k}.$$

Note que a evolução desse sistema depende de $\pi_{1,k}$. Para ilustrar esse experimento, considere a seguinte sequência: $\pi_{1,k}=\{0, 12, 25\}$. Tem-se então:

$$\tau_{1,1} = \pi_{1,1} = 0,$$

$$\tau_{2,1} = \max\{\pi_{1,1}, \pi_{3,1}\} = \max\{0, 0\} = 0,$$

$$\tau_{3,2} = \pi_{3,1} + v_{1,1} = 0 + 10 = 10,$$

A lâmpada fica, agora, 2 segundos apagada antes de ser acesa novamente.

$$\tau_{1,2} = 12,$$

$$\tau_{2,2} = \max\{\pi_{1,1}, \pi_{3,2}\} = \max\{12, 10\} = 12,$$

$$\tau_{3,3} = 12 + 10 = 22,$$

A lâmpada fica, agora, 3 segundos apagada antes de ser acesa novamente.

$$\tau_{1,3} = 25,$$

$$\tau_{2,2} = \max\{25, 22\} = 25,$$

$$\tau_{3,4} = 25 + 10 = 35,$$

O programa em linguagem ladder para implementação no CLP é mostrado na figura 5.3, onde %I1.0 é a entrada do CLP acionada por uma chave para ligar a lâmpada e %Q2.0 é a saída do CLP para acionar o módulo de lâmpadas e %TM0 é o temporizador, ajustado para um atraso de 10s.

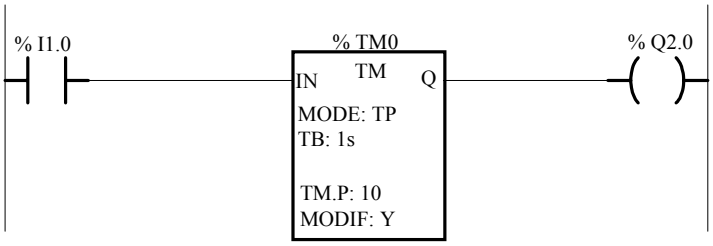


Figura 5.3: Linguagem ladder do primeiro experimento.

(*ii*) Segundo experimento

O segundo experimento é semelhante ao primeiro, sendo agora acrescentado o recurso de contagem, ao lugar, da rede de Petri, que anteriormente representava a condição do acionamento do botão. No experimento atual, os arcos passam a representar um contador, pois a transição t_2 deverá ocorrer após o botão ser acionado quatro vezes, fazendo com que a lâmpada passe do estado desligada para o estado ligada. Assim como no experimento anterior, após a lâmpada acender ou o conjunto de lâmpadas acender, permanece acesa por um determinado período de tempo e depois é desligada automaticamente. Uma forma de representação por redes de Petri é mostrada na figura 5.4.

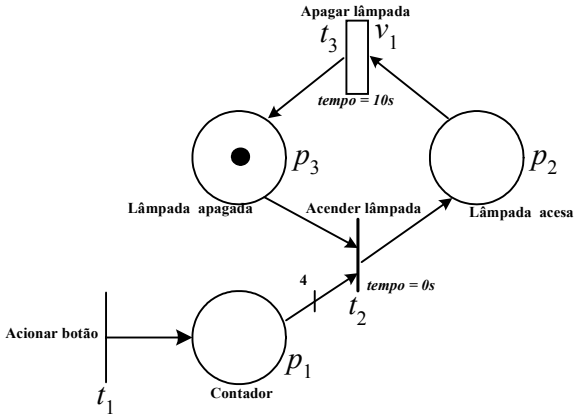


Figura 5.4: Redes de Petri para ligar uma lâmpada utilizando um contador.

A rede de Petri (*P, T, A, w, x,* **V***)* conta com os mesmos elementos $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2, t_3\}$, $A=\{(p_1, t_2), (p_2, t_3), (p_3, t_2), (t_1, p_1), (t_2, p_2), (t_3, p_3)\}$, definidos no

experimento anterior, e $w(p_1, t_2) = 4$, $w(p_2, t_3) = 1$, $w(p_3, t_2) = 1$, $w(t_1, p_1) = 1$, $w(t_2, p_2) = 1$, $w(t_3, p_3) = 1$. (V=$\{\underline{v}_3\}$ e $\underline{v}_3=\{v_{3,1}, v_{3,2}, v_{3,3}, ...\} = \{10, 10, 10, ...\}$).

O estado inicial desta rede de Petri é $\underline{x}_0= [0, 0, 1]$, e por inspeção pode-se encontrar a matriz de incidência **A** da rede de Petri da figura 5.4, uma matriz 3x3 cujo elemento *(j, i)* é da forma $a_{ji}= w(t_j, p_i) – w(p_i, t_j)$, dada por:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

Usando a matriz de incidência **A**, pode-se, a partir da equação de estados $\underline{\boldsymbol{x}}' = \underline{\boldsymbol{x}} + \mathbf{uA}$, simular os resultados de todos os próximos estados que podem ser gerados a partir do disparo das transições.

Com o disparo de t_1 tem-se $u_1=[1, 0, 0]$, então:

$\underline{x}_1= \underline{x}_0 + uA$

$$\underline{x}_1 = [0, 0, 1] + [1, 0, 0].\begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

$\underline{x}_1= [1, 0, 1]$

A partir do primeiro disparo de t_1, nenhuma transição é habilitada, pois para que a transição t_2 possa ser habilitada é necessário que ocorram quatro disparos da transição t_1, que habilitará t_2 fazendo com que o próximo disparo estado seja [0, 1, 0], conforme descrito na seqüência a seguir:

$\underline{x}_2= [1, 0, 1] + [1, 0, 0].\mathbf{A}$

$\underline{x}_2= [2, 0, 1]$

$\underline{x}_3= [2, 0, 1] + [1, 0, 0].\mathbf{A}$

$\underline{x}_3= [3, 0, 1]$

$\underline{x}_4= [3, 0, 1] + [1, 0, 0].\mathbf{A}$

$\underline{x}_4= [4, 0, 1]$

$\underline{x}_5= [4, 0, 1] + [0, 1, 0].\mathbf{A}$

$\underline{x}_5= [0, 1, 0]$

Assim como no experimento anterior, os próximos estados dependerão do número de vezes que o botão seja acionado antes que sejam decorridos $\underline{v}_3$ segundos. A evolução desse sistema é melhor visualizada utilizando-se as equações dos instantes de disparo das transições . Para tanto note que:

Equações dos instantes de tempo que os lugares recebem os discos (1).

$\pi_{1,k} = \tau_{1,k}$, $k = 1, 2, ...$

$\pi_{2,k} = \tau_{2,k}$, $k = 1, 2, ...$

$\pi_{3,k} = \tau_{3,k-1}$, $k = 2, 3, ...,$ $\pi_{3,1} = 0$

Equações dos instantes de tempo de disparo das transições (2).

$\tau_{1,k} = \pi_{1,k}$, $k = 1, 2, ...$

$\tau_{2,k} = \max\{\pi_{1,4k}, \pi_{3,k}\}$, $k = 1, 2, ...$

$\tau_{3,k} = \pi_{3,k} + v_{3,k}$, $k = 1, 2, ...$

Essas equações podem ser colocadas na seguinte forma recursiva:

$\tau_{1,k} = \pi_{1,k}$,

$\tau_{2,k} = \max\{\tau_{3,k-1}, \tau_{1,4k}\}$, $\tau_{3,1} = 0$

$\tau_{3,k} = \tau_{2,k} + v_{3,k}$,

que depende inicamente das sequências $\pi_{1,k}$ e $v_{3,k}$. Considere, então a sequência: $\pi_{1,k}$ = {1, 2, 5, 10, 12, 14, 16, 19, 25, 36, 39, 41}. Portanto:

$\tau_{1,1} = 1$, $\tau_{1,2} = 2$, $\tau_{1,3} = 5$, $\tau_{1,4} = 10$

$\tau_{2,1} = \max\{\tau_{3,1}, \tau_{1,4}\} = \max\{0, 10\} = 10$

$\tau_{3,1} = 10 + 10 = 20$

$\tau_{1,5} = 12$, $\tau_{1,6} = 14$, $\tau_{1,7} = 16$, $\tau_{1,8} = 19$

$\tau_{2,2} = \max\{\tau_{3,2}, \tau_{1,8}\} = \max\{20, 19\} = 20$

$\tau_{3,1} = 20 + 10 = 30$

$\tau_{1,9} = 25$, $\tau_{1,10} = 36$, $\tau_{1,11} = 39$, $\tau_{1,12} = 41$

$\tau_{2,3} = \max\{\tau_{3,3}, \tau_{1,12}\} = \max\{30, 41\} = 41$

$\tau_{3,1} = 41 + 10 = 51$

definindo, portanto, a seguinte seqüência de disparos: $t^S = \{t_1, t_1, t_1, t_1, t_2, t_1, t_1, t_1, t_1, t_3, t_2, t_1, t_3, t_1, t_1, t_1, t_1, t_2, t_3\}$, levando à seguinte evolução de estados: [0, 0, 1], [1, 0, 1], [2, 0, 1], [3, 0, 1], [4, 0, 1], [0, 1, 0], [1, 1, 0], [2, 1, 0], [3, 1, 0], [4, 1, 0], [4, 0, 1], [0, 1, 0], [0, 0, 1], [1, 0, 1], [2, 0, 1], [3, 0, 1], [4, 0, 1], [0, 1, 0] e [0, 0, 1].

O programa em linguagem ladder que será implementado no CLP para o experimento 2 é mostrado na figura 5.5, onde %I1.0 é a entrada do CLP acionada por uma chave para ligar a lâmpada, %Q2.0 é a saída do CLP para acionar o módulo de

lâmpadas, %TM0 é o temporizador, ajustado para um atraso de 10s e %C0 é contador, ajustado para o número de contagem igual a 4.

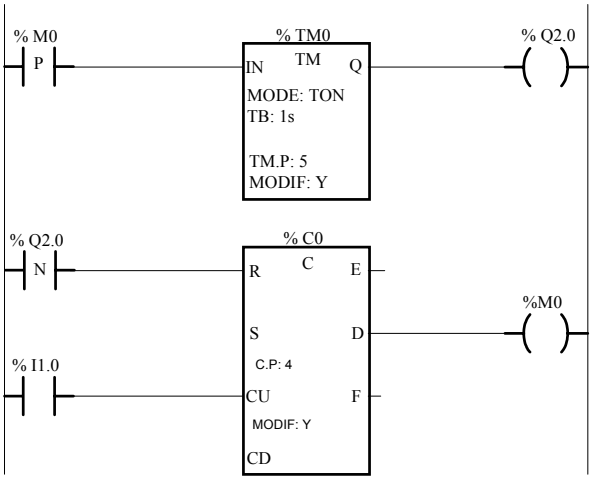


Figura 5.5: Linguagem ladder para implementação do segundo experimento.

5.2. Experimentos para controle de tráfego

Nestes experimentos serão consideradas duas situações: o controle de semáforos em uma interseção T e o controle de um sinal de trânsito para pedestre.

(*i*) Terceiro experimento

Este experimento tem por finalidade implementar o controle de tráfego de uma simples interseção em T, mostrado na figura 5.6. O fluxo de veículos na avenida A ocorre num único sentido, caminhando no sentido da avenida B. Na avenida B o fluxo ocorre nos dois sentidos e não existe acesso para avenida A. Deseja-se controlar o tráfego de maneira que o fluxo nos dois sentidos da avenida B seja interrompido ao mesmo tempo e permita o acesso dos veículos da avenida A para avenida B nos dois sentidos. Após isso, o fluxo da avenida A é interrompido permitindo que o fluxo nos dois sentidos da avenida B seja restabelecido, e assim sucessivamente. A evolução dos estados é mostrada na tabela 5.1, a modelagem por rede de Petri para este experimento é vista na figura 5.7. Embora sejam dois os semáforos que controlam o fluxo na avenida B, será considerado como sendo somente um, uma vez que operam em paralelo. Assim, tem-se apenas, dois semáforos controlando o fluxo de veículos nas duas avenidas: o semáforo 1, que controla o fluxo na avenida A e o semáforo 2, que controla o fluxo na avenida B. Cada semáforo possui três lâmpadas sinalizadoras (verde, amarela e vermelha). Os semáforos 2 e 3 funcionam da mesma maneira e ao mesmo tempo

(sincronizados). Em todos os semáforos, o tempo que o sinal verde permanecerá ligado será de 15s e o tempo do sinal amarelo é de 6s. Deve-se ressaltar que a mudança de sinal entre os dois semáforos só ocorrerá 3s após o sinal vermelho acender em todos os semáforos. Por exemplo, no semáforo 1 o sinal vermelho só passará para o sinal verde após 3s dos semáforos 2 e 3 estiverem passado para o sinal vermelho.

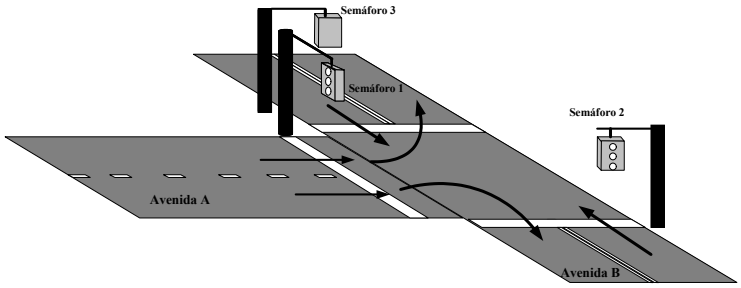


Figura 5.6: Interseção T do terceiro experimento.

Tabela 5.1: Evolução dos estados das lâmpadas do semáforo.

Estado das lâmpadas						
	Semáforo 1			Semáforos 2 e 3		
	Vermelha	Amarela	Verde	Vermelha	Amarela	Verde
1	OFF	OFF	ON	ON	OFF	OFF
2	OFF	ON	OFF	ON	OFF	OFF
3	ON	OFF	OFF	ON	OFF	OFF
4	ON	OFF	OFF	OFF	OFF	ON
5	ON	OFF	OFF	OFF	ON	OFF
6	ON	OFF	OFF	ON	OFF	OFF
7	OFF	OFF	ON	ON	OFF	OFF

Para obter o modelo (*P, T, A, w, x,* V*)* da rede de Petri, note que os eventos desse sistema são: (i) ligar semáforos (lâmpada verde do semáforo 1 juntamente com as lâmpadas vermelhas dos semáforos 2 e 3 acendem) ; (ii) desligar lâmpada verde semáforo 1 e ligar lâmpada amarela do semáforo 1; (iii) desligar lâmpada amarela do semáforo 1 e ligar lâmpada vermelha do semáforo 1; (iv) desligar lâmpadas vermelhas dos semáforos 2 e 3 e ligar lâmpadas verdes dos semáforos 2 e 3; (v) desligar lâmpadas verdes dos semáforos 2 e 3 e ligar lâmpadas amarelas dos semáforos 2 e 3; (vi) desligar lâmpadas amarelas dos semáforos 2 e 3 e ligar lâmpadas vermelhas dos semáforos 2 e 3; (vii) desligar lâmpada vermelha do semáforo 1 e ligar a lâmpada verde do semáforo 1, voltando ao estado inicial. Esses eventos dão origem ao conjunto $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. Os lugares, que representam as condições para que os eventos ocorram são: lâmpada verde semáforo 1 acesa (p_1), lâmpadas vermelhas dos semáforos 2 e 3 acesas

(p_2), lâmpada amarela do semáforo 1 acesa (p_3), lâmpada vermelha do semáforo 1 acesa (p_4), lâmpadas verdes dos semáforos 2 e 3 acesas (p_5), lâmpadas amarelas dos semáforos 2 e 3 acesas (p_6), indica mudança de sinal do amarelo para o vermelho nos semáforos 2 e 3 (p_7), indica mudança do sinal amarelo para o vermelho no semáforo 1 (p_8), definindo, portanto, o conjunto $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$. Os elementos da rede de Petri são: $A=\{(p_1, t_2), (p_2, t_4), (p_3, t_3), (p_4, t_7), (p_5, t_5), (p_6, t_6), (p_7, t_7), (p_8, t_4), (t_1, p_1), (t_1, p_2), (t_2, p_3), (t_3, p_4), (t_3, p_8), (t_4, p_5), (t_5, p_6), (t_6, p_2), (t_6, p_7)\}$, $w(p_1, t_2)=1$, $w(p_2, t_4)=1$, $w(p_3, t_3)=1$, $w(p_4, t_7)=1$, $w(p_5, t_5)=1$, $w(p_6, t_6)=1$, $w(p_7, t_7)=1$, $w(p_8, t_4)=1$, $w(t_1, p_1)=1$, $w(t_1, p_2)=1$, $w(t_2, p_3)=1$, $w(t_3, p_4)=1$, $w(t_3, p_8)=1$, $w(t_4, p_5)=1$, $w(t_5, p_6)=1$, $w(t_6, p_2)=1$, $w(t_6, p_7)=1$, $\underline{x}[\underline{x}(p_1), \underline{x}(p_2), \underline{x}(p_3), \underline{x}(p_4), \underline{x}(p_5), \underline{x}(p_6), \underline{x}(p_7), \underline{x}(p_8)]$, onde $\underline{x}_0=[0, 0, 0, 0, 0, 0, 0, 0]$ e $v_2=\{v_{2,1}, v_{2,2}, v_{2,3}, ...\}=\{15, 15, 15, ...\}$, $v_3=\{v_{3,1}, v_{3,2}, v_{3,3}, ...\}=\{6, 6, 6, ...\}$, $v_4=\{v_{4,1}, v_{4,2}, v_{4,3}, ...\}=\{3, 3, 3, ...\}$, $v_5=\{v_{5,1}, v_{5,2}, v_{5,3}, ...\}=\{15, 15, 15, ...\}$, $v_6=\{v_{6,1}, v_{6,2}, v_{6,3}, ...\}=\{6, 6, 6, ...\}$, $v_7=\{v_{7,1}, v_{7,2}, v_{7,3}, ...\}= \{3, 3, 3, ...\}$.

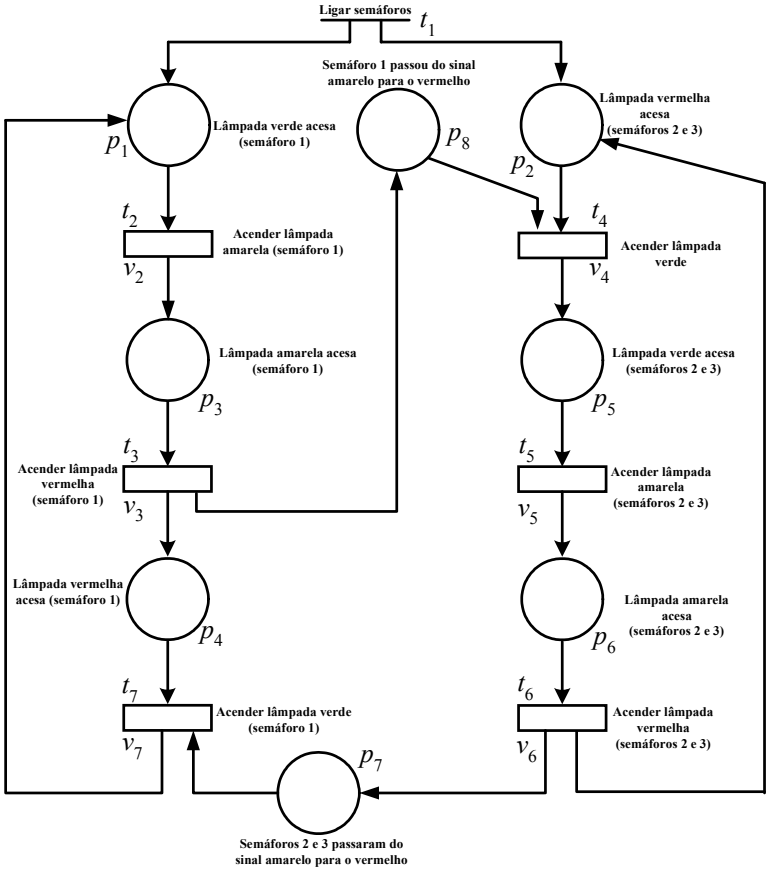


Figura 5.7: Rede de Petri da interseção T.

Uma vez caracterizados todos os elementos da rede de Petri, o próximo passo é obter a evolução dos estados. A partir da figura 5.7 pode-se obter a matriz de incidência **A**. Portanto:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \end{bmatrix}$$

Usando a matriz de incidência **A**, pode-se a partir da equação de estados, $\underline{x}' = \underline{x} + \underline{u}.A$, simular os resultados de todos os próximos estados que podem ser gerados a partir do disparo das transições.

t_1: $\underline{u}_1 = [1, 0, 0, 0, 0, 0, 0]$ $\quad \underline{x}_1 = \underline{x}_0 + \underline{u}_1.A = [1, 1, 0, 0, 0, 0, 0, 0]$

t_2: $\underline{u}_2 = [0, 1, 0, 0, 0, 0, 0]$ $\quad \underline{x}_2 = \underline{x}_1 + \underline{u}_2.A = [0, 1, 1, 0, 0, 0, 0, 0]$

t_3: $\underline{u}_3 = [0, 0, 1, 0, 0, 0, 0]$ $\quad \underline{x}_3 = \underline{x}_2 + \underline{u}_3.A = [0, 1, 0, 1, 0, 0, 0, 1]$

t_4: $\underline{u}_4 = [0, 0, 0, 1, 0, 0, 0]$ $\quad \underline{x}_4 = \underline{x}_3 + \underline{u}_4.A = [0, 0, 0, 1, 1, 0, 0, 0]$

t_5: $\underline{u}_5 = [0, 0, 0, 0, 1, 0, 0]$ $\quad \underline{x}_5 = \underline{x}_4 + \underline{u}_5.A = [0, 0, 0, 1, 0, 1, 0, 0]$

t_6: $\underline{u}_6 = [0, 0, 0, 0, 0, 1, 0]$ $\quad \underline{x}_6 = \underline{x}_5 + \underline{u}_6.A = [0, 1, 0, 1, 0, 0, 1, 0]$

t_7: $\underline{u}_7 = [0, 0, 0, 0, 0, 0, 1]$ $\quad \underline{x}_7 = \underline{x}_6 + \underline{u}_7.A = [1, 1, 0, 0, 0, 0, 0, 0]$

As equações de estado para os tempos de disparos das transições, são:

Equações dos instantes de tempo para os lugares receberem o disco (1).

$$\pi_{1,k} = \begin{cases} \tau_{1,k}, & k = 1 \\ \tau_{1,k}, & k \neq 1 \end{cases}$$

$$\pi_{2,k} = \begin{cases} \tau_{1,k}, & k = 1 \\ \tau_{6,k}, & k \neq 1 \end{cases}$$

$\pi_{3,k} = \tau_{2,k}$

$\pi_{4,k} = \tau_{3,k}$

$\pi_{5,k} = \tau_{4,k}$

$\pi_{6,k} = \tau_{5,k}$

$\pi_{7,k} = \tau_{6,k}$

$\pi_{8,k} = \tau_{3,k}$

Equações do instante de tempo de disparo das transições (2).

$\tau_{1,k}$ (definido apenas para k=1): $\tau_{1,1} = 0$

$\tau_{2,k} = \pi_{1,k} + v_{2,k}$

$\tau_{3,k} = \pi_{3,k} + v_{3,k}$

$\tau_{4,k} = \max\{\pi_{8,k};\ \pi_{2,k}\} + v_{4,k}$

$\tau_{5,k} = \pi_{5,k} + v_{5,k}$

$\tau_{6,k} = \pi_{6,k} + v_{6,k}$

$\tau_{7,k} = \max\{\pi_{4,k};\ \pi_{7,k}\} + v_{7,k}$

Substituindo as equações de 1 em 2, temos:

$\tau_{1,k}$ (definido apenas para k=1): $\tau_{1,1} = 0$

$$\tau_{2,k} = \begin{cases} \tau_{1,k} + v_{2,k}, & k = 1 \\ \tau_{7,k\text{-}1} + v_{2,k}, & k \neq 1 \end{cases}$$

$\tau_{3,k} = \tau_{2,k} + v_{3,k}$

$$\tau_{4,k} = \begin{cases} \max\{\tau_{3,k}, \tau_{1,k}\} + v_{4,k}, & k = 1 \\ \max\{\tau_{3,k}, \tau_{6,k\text{-}1}\} + v_{4,k}, & k \neq 1 \end{cases}$$

$$\tau_{4,k} = \begin{cases} \max\{\tau_{2,k} + v_{3,k}, \tau_{1,k}\} + v_{4,k}, & k = 1 \\ \max\{\tau_{2,k} + v_{3,k}, \tau_{6,k\text{-}1}\} + v_{4,k}, & k \neq 1 \end{cases}$$

$$\tau_{4,k} = \begin{cases} \max\{\tau_{1,k} + v_{2,k} + v_{3,k}, \tau_{1,k}\} + v_{4,k}, & k = 1 \\ \max\{\tau_{7,k\text{-}1} + v_{2,k} + v_{3,k}, \tau_{6,k\text{-}1}\} + v_{4,k}, & k \neq 1 \end{cases}$$

como $\tau_{2,1} = 0$, então:

$$\tau_{4,k} = \begin{cases} v_{2,k} + v_{3,k} + v_{4,k}, & k = 1 \\ \max\{\tau_{7,k\text{-}1} + v_{2,k} + v_{3,k}, \tau_{6,k\text{-}1}\} + v_{4,k}, & k \neq 1 \end{cases}$$

$\tau_{5,k} = \pi_{5,k} + v_{5,k}$

$$\tau_{5,k} = \begin{cases} v_{2,k} + v_{3,k} + v_{4,k} + v_{5,k}, & k = 1 \\ \max\{\tau_{7,k\text{-}1} + v_{2,k} + v_{3,k}, \tau_{6,k\text{-}1}\} + v_{4,k} + v_{5,k}, & k \neq 1 \end{cases}$$

$\tau_{6,k} = \pi_{6,k} + v_{6,k}$

$$\tau_{6,k} = \begin{cases} v_{2,k} + v_{3,k} + v_{4,k} + v_{5,k} + v_{6,k}, & k = 1 \\ \max\{\tau_{7,k\text{-}1} + v_{2,k} + v_{3,k}, \tau_{6,k\text{-}1}\} + v_{4,k} + v_{5,k} + v_{6,k}, & k \neq 1 \end{cases}$$

$\tau_{7,k} = \max\{\pi_{4,k};\ \pi_{7,k}\} + v_{7,k}$

$$\tau_{6,k} = \begin{cases} \max\{\max\{\tau_{3,k}, \tau_{1,k}\}, \tau_{6,k}\} + v_{7,k}, & k = 1 \\ \max\{\max\{\tau_{3,k}, \tau_{6,k\text{-}1}\}, \tau_{6,k}\} + v_{7,k}, & k \neq 1 \end{cases}$$

Portanto:

$\tau_{1,k}$ (definido apenas para k=1): $\tau_{1,1} = 0$

$$\tau_{2,k} = \begin{cases} \tau_{1,k} + v_{2,k}, & k = 1 \\ \tau_{7,k\text{-}1} + v_{2,k}, & k \neq 1 \end{cases}$$

$$\tau_{3,k} = \tau_{2,k} + v_{3,k}$$

$$\tau_{4,k} = \begin{cases} \tau_{1,k} + v_{2,k} + v_{3,k} + v_{4,k}, & k = 1 \\ \tau_{7,k\text{-}1} + v_{2,k} + v_{3,k} + v_{4,k}, & k \neq 1 \end{cases}$$

$$\tau_{5,k} = \begin{cases} \tau_{1,k} + v_{2,k} + v_{3,k} + v_{4,k} + v_{5,k}, & k = 1 \\ \tau_{7,k\text{-}1} + v_{2,k} + v_{3,k} + v_{4,k} + v_{5,k}, & k \neq 1 \end{cases}$$

$$\tau_{6,k} = \begin{cases} \tau_{1,k} + v_{2,k} + v_{3,k} + v_{4,k} + v_{5,k} + v_{6,k}, & k = 1 \\ \tau_{7,k\text{-}1} + v_{2,k} + v_{3,k} + v_{4,k} + v_{5,k} + v_{6,k}, & k \neq 1 \end{cases}$$

$$\tau_{7,k} = \begin{cases} \tau_{1,k} + v_{2,k} + v_{3,k} + v_{4,k} + v_{5,k} + v_{6,k} + v_{7,k}, & k = 1 \\ \tau_{7,k\text{-}1} + v_{2,k} + v_{3,k} + v_{4,k} + v_{5,k} + v_{6,k} + v_{7,k}, & k \neq 1 \end{cases}$$

O programa em linguagem ladder que realiza a seqüência de transições para implementação do experimento dos semáforos da interseção T no CLP é mostrado na figura 5.8. %I1.0 é a chave para ligar o sistema, %Q2.0 acende a lâmpada verde do semáforo 1, %Q2.1 acende a lâmpada amarela do semáforo 1, %Q2.2 acende a lâmpada vermelha do semáforo 1, %Q2.3 acende a lâmpada verde do semáforo 2, %Q2.4 acende a lâmpada amarela do semáforo 2 e %Q2.5 acende a lâmpada vermelha do semáforo 2.

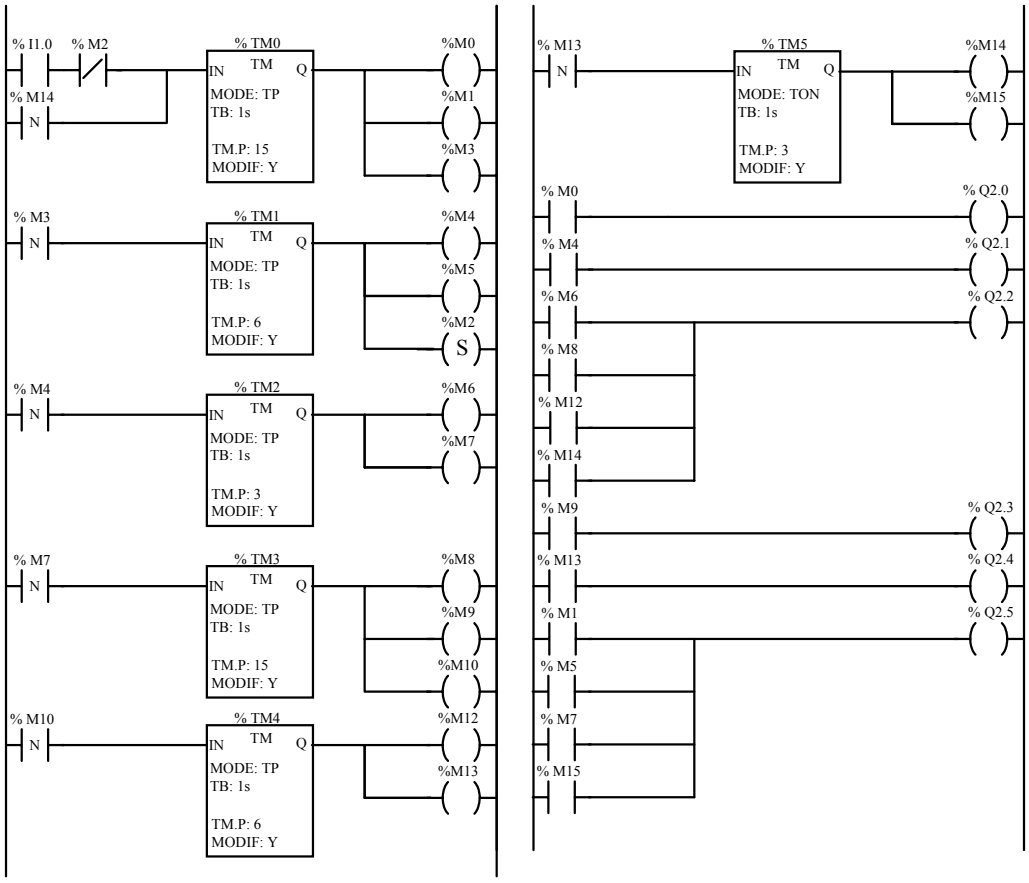


Figura 5.8: Linguagem ladder para a interseção T.

(*ii*) Quarto experimento

Neste experimento será implementado um programa para executar o controle de sistema de tráfego de veículos em um ponto de travessia de pedestre. O sistema possui dois semáforos, um para os veículos com três lâmpadas sinalizadoras (verde, amarela e vermelha) e outro para o pedestre com duas lâmpadas sinalizadoras (verde e vermelha).

Quando o pedestre chega a uma via pública (figura 5.9) e deseja atravessar esta via, usando o recurso do semáforo de controle de trânsito, ele deverá acionar um botão no poste próximo a passagem em um painel de comando para poder ativar o sistema. Após acionado o pedido de passagem, o programa deverá executar um tempo de espera de 20s para poder iniciar o ciclo de mudança nas lâmpadas do semáforo. Quando a mudança começar, a luz verde, para passagem de veículos, deverá piscar quatro vezes até que mude para o sinalizador amarelo, e em seguida para o vermelho. Durante a passagem do sinal amarelo para o sinal vermelho, a luz amarela deve permanecer por 6s acesa. Após 3s da passagem do sinal amarelo para o sinal vermelho no semáforo para passagem de veículos, o sinalizador para passagem de pedestre passará do sinal vermelho para o verde, permanecendo assim por 15s. Ao final deste período a lâmpada verde, do sinal de pedestre deverá, também, piscar quatro vezes, indicando o final do período e, em seguida, retornar para o vermelho. Após 3s da passagem do sinal verde para o vermelho da passagem de pedestre, é que o sinalizador para passagem de veículos voltará para o sinal verde. A tabela 5.2 mostra a evolução dos estados do experimento e a figura 5.10 mostra a modelagem em rede de Petri.

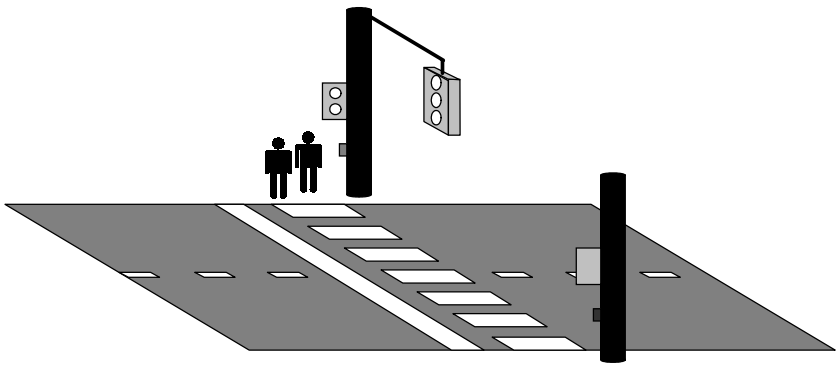


Figura 5.9: Ilustração de uma passagem de pedestre.

Tabela 5.2: Evolução de estados da passagem de pedestre.

	Estado das lâmpadas					
	Semáforo - Carros			Semáforo - Pedestre		
	Vermelha	Amarela	Verde	Vermelha	Verde	Botão
1	OFF	OFF	ON	ON	OFF	OFF
2	OFF	OFF	ON	ON	OFF	ON
3	OFF	OFF	ON / OFF	ON	OFF	XX
4	OFF	ON	OFF	ON	OFF	XX
5	ON	OFF	OFF	ON	OFF	XX
6	ON	OFF	OFF	OFF	ON	XX
7	ON	OFF	OFF	OFF	ON / OFF	XX
8	ON	OFF	OFF	ON	OFF	XX
9	OFF	OFF	ON	ON	OFF	ON ou OFF

Para obter o modelo (*P, T, A, w, x,* V*)* da rede de Petri, note que os eventos desse sistema são: (i) apertar botão; (ii) fazer lâmpada verde semáforo de veículos piscar; (iii) desligar lâmpada verde do semáforo de veículos e ligar lâmpada amarela do semáforo de veículos; (iv) desligar lâmpada amarela do semáforo de veículos e ligar lâmpada vermelha do semáforo de veículos; (v) desligar lâmpada vermelha do semáforo de pedestre e ligar lâmpada verde do semáforo de pedestre; (vi) piscar lâmpada verde do semáforo de pedestre; (vii) desligar lâmpada verde do semáforo de pedestre e ligar lâmpada vermelha do semáforo de pedestre; (viii) desligar lâmpada vermelha do semáforo de veículos e ligar lâmpada verde do semáforo de veículos, voltando ao estado inicial. Esses eventos dão origem ao conjunto $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}\}$. Os lugares, que representam as condições para que os eventos ocorram são: lâmpada verde do semáforo de veículos acesa (p_1), botão para passagem de pedestre acionado (p_2), torna inválido todos os acionamentos do botão durante a realização do ciclo de passagem de pedestre (p_3), condição para lâmpada verde do semáforo de veículos começar a piscar (p_4), lâmpada verde do semáforo de veículos apagada (p_5), número de vezes que a lâmpada verde do semáforo de veículos pisca (p_6), lâmpada amarela do semáforo de veículos acesa (p_7), lâmpada vermelha do semáforo de veículos acesa (p_8), indica mudança do sinal amarelo para o vermelho no semáforo de veículos (p_9), lâmpada vermelha do semáforo de pedestre acesa (p_{10}), lâmpada verde do semáforo de pedestre acesa (p_{11}), condição para lâmpada verde do semáforo de pedestre começar a piscar (p_{12}), lâmpada verde do semáforo de pedestre apagada (p_{13}), número de vezes que a lâmpada verde do semáforo de pedestre deve piscar (p_{14}), indica mudança do sinal verde para o vermelho no semáforo de passagem de pedestre (p_{15}), definindo, portanto,

o conjunto $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}\}$. Os elementos da rede de Petri são: $A=\{(p_1, t_1), (p_1, t_2), (p_2, t_1), (p_3, t_4), (p_3, t_8), (p_4, t_2), (p_4, t_6), (p_5, t_5), (p_5, t_6), (p_6, t_6), (p_7, t_7), (p_8, t_8), (p_9, t_9), (p_{10}, t_9), (p_{11}, t_{10}), (p_{11}, t_{11}), (p_{12}, t_{11}), (p_{12}, t_{13}), (p_{13}, t_{12}), (p_{13}, t_{13}), (p_{14}, t_{13}), (p_{15}, t_8), (t_1, p_3), (t_1, p_4), (t_1, p_5), (t_2, p_4), (t_2, p_5), (t_2, p_6), (t_3, p_2), (t_4, p_3), (t_5, p_1)$ $(t_6, p_7), (t_7, p_8), (t_7, p_9), (t_8, p_1), (t_9, p_{11}), (t_{10}, p_{12}), (t_{11}, p_{12}), (t_{11}, p_{13}), (t_{11}, p_{14})$ $(t_{12}, p_{11}), (t_{13}, p_{10}), (t_{13}, p_{15})\}$, $w(p_1, t_1)=1$, $w(p_1, t_2)=1$, $w(p_2, t_1)=1$, $w(p_3, t_4)=1$, $w(p_3, t_8)=1$, $w(p_4, t_2)=1$, $w(p_4, t_6)=1$, $w(p_5, t_5)=1$, $w(p_5, t_6)=1$, $w(p_6, t_6)=4$, $w(p_7, t_7)=1$, $w(p_8, t_8)=1$, $w(p_9, t_9)=1$, $w(p_{10}, t_9)=1$, $w(p_{11}, t_{10})=1$, $w(p_{11}, t_{11})=1$, $w(p_{12}, t_{11})=1$, $w(p_{12}, t_{13})=1$, $w(p_{13}, t_{12})=1$, $w(p_{13}, t_{13})=1$, $w(p_{14}, t_{13})=4$, $w(p_{15}, t_8)=1$, $w(t_1, p_3)=1$, $w(t_1, p_4)=1$, $w(t_1, p_5)=1$, $w(t_2, p_4)=1$, $w(t_2, p_5)=1$, $w(t_2, p_6)=1$, $w(t_3, p_2)=1$, $w(t_4, p_3)=1$, $w(t_5, p_1)=1$, $w(t_6, p_7)=1$, $w(t_7, p_8)=1$, $w(t_7, p_9)=1$, $w(t_8, p_1)=1$, $w(t_9, p_{11})=1$, $w(t_{10}, p_{12})=1$, $w(t_{11}, p_{12})=1$, $w(t_{11}, p_{13})=1$, $w(t_{11}, p_{14})=1$, $w(t_{12}, p_{11})=1$, $w(t_{13}, p_{10})=1$, $w(t_{13}, p_{15})=1$, $\underline{x}[\underline{x}(p_1), \underline{x}(p_2), \underline{x}(p_3), \underline{x}(p_4), \underline{x}(p_5), \underline{x}(p_6), \underline{x}(p_7), \underline{x}(p_8), \underline{x}(p_9), \underline{x}(p_{10}), \underline{x}(p_{11}), \underline{x}(p_{12}), \underline{x}(p_{13}), \underline{x}(p_{14}), \underline{x}(p_{15})]$, onde $\underline{x}_0=[1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$ e $\underline{v}_1=\{v_{1,1}, v_{1,2}, v_{1,3}, ...\}= \{16, 16, 16, ...\}$, $\underline{v}_2=\{v_{2,1}, v_{2,2}, v_{2,3}, ...\}= \{0.5, 0.5, 0.5, ...\}$, $v_3=\{v_{3,1}, v_{3,2}, v_{3,3}, ...\}=\{0, a, b, ...\}$, $v_5=\{v_{5,1}, v_{5,2}, v_{5,3}, ...\}=\{0.5, 0.5, 0.5, ...\}$, $v_7=\{v_{7,1}, v_{7,2}, v_{7,3}, ...\}=\{6, 6, 6, ...\}$, $v_8=\{v_{8,1}, v_{8,2}, v_{8,3}, ...\}=\{3, 3, 3, ...\}$, $v_9=\{v_{9,1}, v_{9,2}, v_{9,3}, ...\}=\{3, 3, 3, ...\}$, $v_{10}=\{v_{10,1}, v_{10,2}, v_{10,3}, ...\}= \{11, 11, 11, ...\}$, $v_{11}=\{v_{11,1}, v_{11,2}, v_{11,3}, ...\}=\{0.5, 0.5, 0.5, ...\}$, $v_{12}=\{v_{12,1}, v_{12,2}, v_{12,3}, ...\}=\{0.5, 0.5, 0.5, ...\}$.

Uma vez caracterizados todos os elementos da rede de Petri, o próximo passo é obter a evolução dos estados. A partir da figura 5.10 pode-se obter a matriz de incidência **A**. Portanto:

$$\mathbf{A} = \begin{bmatrix} -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & -4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -4 & 1 \end{bmatrix}$$

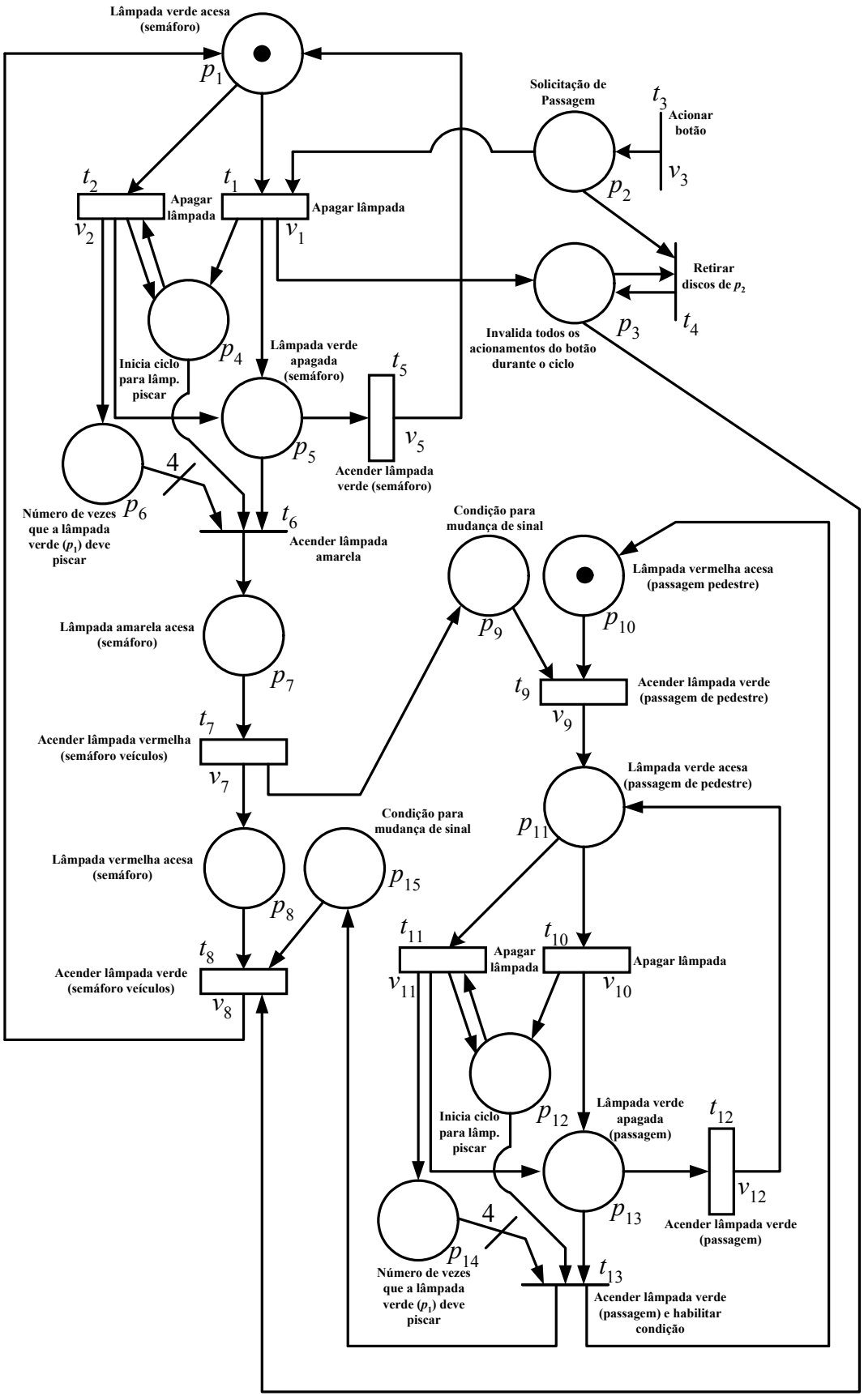


Figura 5.10: Rede de Petri para passagem de pedestre.

Usando a matriz de incidência **A**, pode-se, a partir da equação de estados, $\underline{x}' = \underline{x} + \underline{u}.A$, simular os resultados de todos os próximos estados que podem ser gerados a partir do disparo das transições. $\underline{x}_0$=[1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

A partir do disparo da transição t_3, tem-se:

t_3: $\underline{u}_1$ =[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_1 = \underline{x}_0 + \underline{u}_1.A$ = [1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_1: $\underline{u}_2$ =[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_2 = \underline{x}_1 + \underline{u}_2.A$ = [0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_5: $\underline{u}_3$ =[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_3 = \underline{x}_2 + \underline{u}_3.A$ = [1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_2: $\underline{u}_4$ =[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_4 = \underline{x}_3 + \underline{u}_4.A$ = [0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_5: $\underline{u}_5$ =[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_5 = \underline{x}_4 + \underline{u}_5.A$ = [1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_2: $\underline{u}_6$ =[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_6 = \underline{x}_5 + \underline{u}_6.A$ = [0, 0, 1, 1, 1, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_5: $\underline{u}_7$ =[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_7 = \underline{x}_6 + \underline{u}_7.A$ = [1, 0, 1, 1, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_2: $\underline{u}_8$ =[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_8 = \underline{x}_7 + \underline{u}_8.A$ = [0, 0, 1, 1, 1, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_5: $\underline{u}_9$ =[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_9 = \underline{x}_8 + \underline{u}_9.A$ = [1, 0, 1, 1, 0, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_2: $\underline{u}_{10}$ =[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_{10} = \underline{x}_9 + \underline{u}_{10}.A$ = [0, 0, 1, 1, 1, 4, 0, 0, 0, 1, 0, 0, 0, 0, 0]

t_6: $\underline{u}_{11}$ =[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

$\underline{x}_{11} = \underline{x}_{10} + \underline{u}_{11}.A$ = [0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0]

t_7: $\underline{u}_{12}$ =[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

$\underline{x}_{12} = \underline{x}_{11} + \underline{u}_{12}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0]

t_9: $\underline{u}_{13}$ =[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

$\underline{x}_{13} = \underline{x}_{12} + \underline{u}_{13}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]

t_{10}: $\underline{u}_{14}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]

$\underline{x}_{14} = \underline{x}_{13} + \underline{u}_{14}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0]

t_{12}: $\underline{u}_{15}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

$\underline{x}_{15} = \underline{x}_{14} + \underline{u}_{15}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0]

t_{11}: $\underline{u}_{16}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

$\underline{x}_{16} = \underline{x}_{15} + \underline{u}_{16}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0]

t_{12}: $\underline{u}_{17}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

$\underline{x}_{17} = \underline{x}_{16} + \underline{u}_{17}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0]

t_{11}: $\underline{u}_{18}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

$\underline{x}_{18} = \underline{x}_{17} + \underline{u}_{18}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 2, 0]

t_{12}: $\underline{u}_{19}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

$\underline{x}_{19} = \underline{x}_{18} + \underline{u}_{19}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 2, 0]

t_{11}: $\underline{u}_{20}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

$\underline{x}_{20} = \underline{x}_{19} + \underline{u}_{20}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 3, 0]

t_{12}: $\underline{u}_{21}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

$\underline{x}_{21} = \underline{x}_{20} + \underline{u}_{21}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 3, 0]

t_{11}: $\underline{u}_{22}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

$\underline{x}_{22} = \underline{x}_{21} + \underline{u}_{22}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 4, 0]

t_{13}: $\underline{u}_{23}$ =[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

$\underline{x}_{23} = \underline{x}_{22} + \underline{u}_{23}.A$ = [0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1]

t_{8}: $\underline{u}_{24}$ =[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

$\underline{x}_{24} = \underline{x}_{23} + \underline{u}_{24}.A$ = [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]=$\underline{x}_0$

As dinâmicas da rede de Petri, decompondo o modelo dentro das dinâmicas de transições individuais, são:

Equações dos instantes de tempo que os lugares recebem o disco (1).

$$\pi_{1,k} = \begin{cases} 0, & k = 1 \\ \tau_{5,k\text{-}1}, & k = 2,3,4,5,7,8,9,10,12,13,... \\ \tau_{8,k\text{-}1}, & k = 6,11,16,21,... \end{cases}$$

$$\pi_{2,k} = \tau_{3,k}, \qquad k = 1,2,3,\ldots$$

$$\pi_{3,k} = \begin{cases} \tau_{1,k}, & k = 1 \\ \min\{\tau_{1,5(k\text{-}1)+1}, \tau_{4,k}\} & k = 2,3,4... \end{cases}$$

$$\pi_{4,k} = \begin{cases} \tau_{1,k}, & k = 1,6,11,16,21,... \\ \tau_{2,k}, & k = 2,3,4,5,7,8,9,10,12,... \end{cases}$$

$$\pi_{5,k} = \begin{cases} \tau_{1,k}, & k = 1,6,11,16,21,... \\ \tau_{2,k}, & k = 2,3,4,5,7,8,9,10,12,... \end{cases}$$

$$\pi_{6,k} = \tau_{2,k}, \qquad k = 1,2,3,\ldots$$

$\pi_{7,k} = \tau_{6,k}, \quad k = 1,2,3,\ldots$

$\pi_{8,k} = \tau_{7,k}, \quad k = 1,2,3,\ldots$

$\pi_{9,k} = \tau_{7,k}, \quad k = 1,2,3,\ldots$

$\pi_{10,k} = \tau_{13,k\text{-}1}, \quad k = 1,2,3,\ldots; \ \pi_{1,1} = 0$

$$\pi_{11,k} = \begin{cases} \tau_{9,k}, & k = 1,6,11,16,21,... \\ \tau_{12,k}, & k = 2,3,4,5,7,8,9,10,12,... \end{cases}$$

$$\pi_{12,k} = \begin{cases} \tau_{10,k}, & k = 1,6,11,16,21,... \\ \tau_{11,k}, & k = 2,3,4,5,7,8,9,10,12... \end{cases}$$

$$\pi_{13,k} = \begin{cases} \tau_{10,k}, & k = 1,6,11,16,21,... \\ \tau_{11,k}, & k = 2,3,4,5,7,8,9,10,12... \end{cases}$$

$\pi_{14,k} = \tau_{11,k}, \quad k = 1,2,3,\ldots$

$\pi_{15,k} = \tau_{13,k}, \quad k = 1,2,3,\ldots$

Equações dos instantes de tempos dos disparos das transições (2).

$\tau_{1,k} = \max\{\pi_{1,k}; \pi_{2,k}\} + v_{1,k}$

$\tau_{2,k} = \max\{\pi_{1,k}; \pi_{4,k}\} + v_{2,k}$

$\tau_{3,k} = v_{3,k}$

$\tau_{4,k} = \max\{\pi_{2,k}; \pi_{3,k}\}$

$\tau_{5,k} = \pi_{5,k} + v_{5,k}$

$\tau_{6,k} = \max\{\pi_{4,4k}; \pi_{5,4k}; \pi_{6,4k}\}$

$\tau_{7,k} = \pi_{7,k} + v_{7,k}$

$\tau_{8,k} = \max\{\pi_{3,k}; \pi_{8,k}; \pi_{15,k}\} + v_{8,k}$

$\tau_{9,k} = \max\{\pi_{9,k}; \pi_{10,k}\} + v_{9,k}$

$\tau_{10,k} = \pi_{11,k} + v_{10,k}$

$\tau_{11,k} = \max\{\pi_{11,k}; \pi_{12,k}\} + v_{11,k}$

$\tau_{12,k} = \pi_{13,k} + v_{12,k}$

$\tau_{13,k} = \max\{\pi_{12,4k}; \pi_{13,4k}; \pi_{14,4k}\}$

Substituindo as equações de 1 em 2 , em função de seqüência de disparos das transições, começando com a transição t_3, tem-se:

t_3: $\tau_{3,1} = v_{3,1}$; (disco: $\pi_{2,1}$)

t_1: $\tau_{1,1} = v_{3,1} + v_{1,1}$;(disco: $\pi_{3,1}, \pi_{4,1}, \pi_{5,1}$)

t_5: $\tau_{5,1} = \tau_{1,1} + v_{5,1} = v_{1,1} + v_{3,1} + v_{5,1}$; (disco: $\pi_{1,2}$)

t_2: $\tau_{2,1} = \tau_{5,1} + v_{2,1} = v_{1,1} + v_{2,1} + v_{3,1} + v_{5,1}$; (disco: $\pi_{4,2}, \pi_{5,2}, \pi_{6,1}$)

t_5: $\tau_{5,2} = \tau_{2,1} + v_{5,2} = v_{1,1} + v_{2,1} + v_{3,1} + v_{5,1} + v_{5,2}$; (disco: $\pi_{1,3}$)

t_2: $\tau_{2,2} = \tau_{5,2} + v_{2,2} = v_{1,1} + v_{2,1} + v_{2,2} + v_{3,1} + v_{5,1} + v_{5,2}$; (disco: $\pi_{4,3}$, $\pi_{5,3}$, $\pi_{6,2}$)

t_5: $\tau_{5,3} = \tau_{2,2} + v_{5,3} = v_{1,1} + v_{2,1} + v_{2,2} + v_{3,1} + v_{5,1} + v_{5,2} + v_{5,3}$; (disco: $\pi_{1,4}$)

t_2: $\tau_{2,3} = \tau_{5,3} + v_{2,3} = v_{1,1} + v_{2,1} + v_{2,2} + v_{2,3} + v_{3,1} + v_{5,1} + v_{5,2} + v_{5,3}$; (disco: $\pi_{4,4}$, $\pi_{5,4}$, $\pi_{6,3}$)

t_5: $\tau_{5,4} = \tau_{2,3} + v_{5,4} = v_{1,1} + v_{2,1} + v_{2,2} + v_{2,3} + v_{3,1} + v_{5,1} + v_{5,2} + v_{5,3} + v_{5,4}$; (disco: $\pi_{1,5}$)

t_2: $\tau_{2,4} = \tau_{5,4} + v_{2,4} = v_{1,1} + v_{2,1} + v_{2,2} + v_{2,3} + v_{2,4} + v_{3,1} + v_{5,1} + v_{5,2} + v_{5,3} + v_{5,4}$; (disco: $\pi_{4,5}$, $\pi_{5,5}$, $\pi_{6,4}$)

t_6: $\tau_{6,1} = \tau_{2,4} = v_{1,1} + v_{2,1} + v_{2,2} + v_{2,3} + v_{2,4} + v_{3,1} + v_{5,1} + v_{5,2} + v_{5,3} + v_{5,4}$; (disco: $\pi_{7,1}$)

t_7: $\tau_{7,1} = \tau_{6,1} + v_{7,1} = v_{1,1} + v_{2,1} + v_{2,2} + v_{2,3} + v_{2,4} + v_{3,1} + v_{5,1} + v_{5,2} + v_{5,3} + v_{5,4} + v_{7,1}$; (disco: $\pi_{8,1}$, $\pi_{9,1}$)

t_9: $\tau_{9,1} = \tau_{7,1} + v_{9,1} = v_{1,1} + v_{2,1} + v_{2,2} + v_{2,3} + v_{2,4} + v_{3,1} + v_{5,1} + v_{5,2} + v_{5,3} + v_{5,4} + v_{7,1} + v_{9,1}$ ($v_{2,1} = v_{2,2} = v_{2,3} = v_{2,4}$ e $v_{5,1} = v_{5,2} = v_{5,3} = v_{5,4}$); (disco: $\pi_{11,1}$)

t_{10}: $\tau_{10,1} = \tau_{9,1} + v_{10,1} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1}$; (disco: $\pi_{12,1}$, $\pi_{13,1}$)

t_{12}: $\tau_{12,1} = \tau_{10,1} + v_{12,1} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{12,1}$; (disco: $\pi_{11,2}$)

t_{11}: $\tau_{11,1} = \tau_{12,1} + v_{11,1} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{11,1} + v_{12,1}$; (disco: $\pi_{12,2}$, $\pi_{13,2}$, $\pi_{14,1}$)

t_{12}: $\tau_{12,2} = \tau_{11,1} + v_{12,2} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{11,1} + v_{12,1} + v_{12,2}$; (disco: $\pi_{11,3}$)

t_{11}: $\tau_{11,2} = \tau_{12,2} + v_{11,2} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{11,1} + v_{11,2} + v_{12,1} + v_{12,2}$; (disco: $\pi_{12,3}$, $\pi_{13,3}$, $\pi_{14,2}$)

t_{12}: $\tau_{12,3} = \tau_{11,2} + v_{12,3} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{11,1} + v_{11,2} + v_{12,1} + v_{12,2} + v_{12,3}$; (disco: $\pi_{11,4}$)

t_{11}: $\tau_{11,3} = \tau_{12,3} + v_{11,3} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{11,1} + v_{11,2} + v_{11,3} + v_{12,1} + v_{12,2} + v_{12,3}$; (disco: $\pi_{12,1}$, $\pi_{13,4}$, $\pi_{14,3}$)

t_{12}: $\tau_{12,4} = \tau_{11,3} + v_{12,4} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{11,1} + v_{11,2} + v_{11,3} + v_{12,1} + v_{12,2} + v_{12,3} + v_{12,4}$; (disco: $\pi_{11,5}$)

t_{11}: $\tau_{11,4} = \tau_{12,4} + v_{11,4} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{11,1} + v_{11,2} + v_{11,3} + v_{11,4} + v_{12,1} + v_{12,2} + v_{12,3} + v_{12,4}$; (disco: $\pi_{12,5}$, $\pi_{13,5}$, $\pi_{14,4}$)

t_{13}: $\tau_{13,1} = \tau_{11,4} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{9,1} + v_{10,1} + v_{11,1} + v_{11,2} + v_{11,3} + v_{11,4} + v_{12,1} + v_{12,2} + v_{12,3} + v_{12,4}$; (disco: $\pi_{10,2}$, $\pi_{15,1}$)

($v_{11,1} = v_{11,2} = v_{11,3} = v_{11,4}$ e $v_{12,1} = v_{12,2} = v_{12,3} = v_{12,4}$)

t_8: $\tau_{8,1} = \tau_{13,1} + v_{8,1} = v_{1,1} + 4.v_{2,1} + v_{3,1} + 4.v_{5,1} + v_{7,1} + v_{8,1} + v_{9,1} + v_{10,1} + 4.v_{11,1} + 4.v_{12,1;}$ (disco: $\pi_{1,6}$)

$\pi_{1,6}$ indica que foi executado um ciclo completo, desde que o botão para passagem de pedestre foi acionado.

O programa em linguagem ladder que será implementado no CLP que executa a seqüência de mudanças de estado no semáforo de pedestre é mostrado na figura 5.11. %I1.0 é a chave para ligar o sistema, %I1.1 é o botão para passagem de pedestre, %Q2.0 acende a lâmpada verde do semáforo de veículos, %Q2.1 acende a lâmpada amarela do semáforo de veículos, %Q2.2 acende a lâmpada vermelha do semáforo de veículos, %Q2.3 acende a lâmpada verde do semáforo de pedestres e %Q2.4 acende a lâmpada vermelha do semáforo de pedestre.

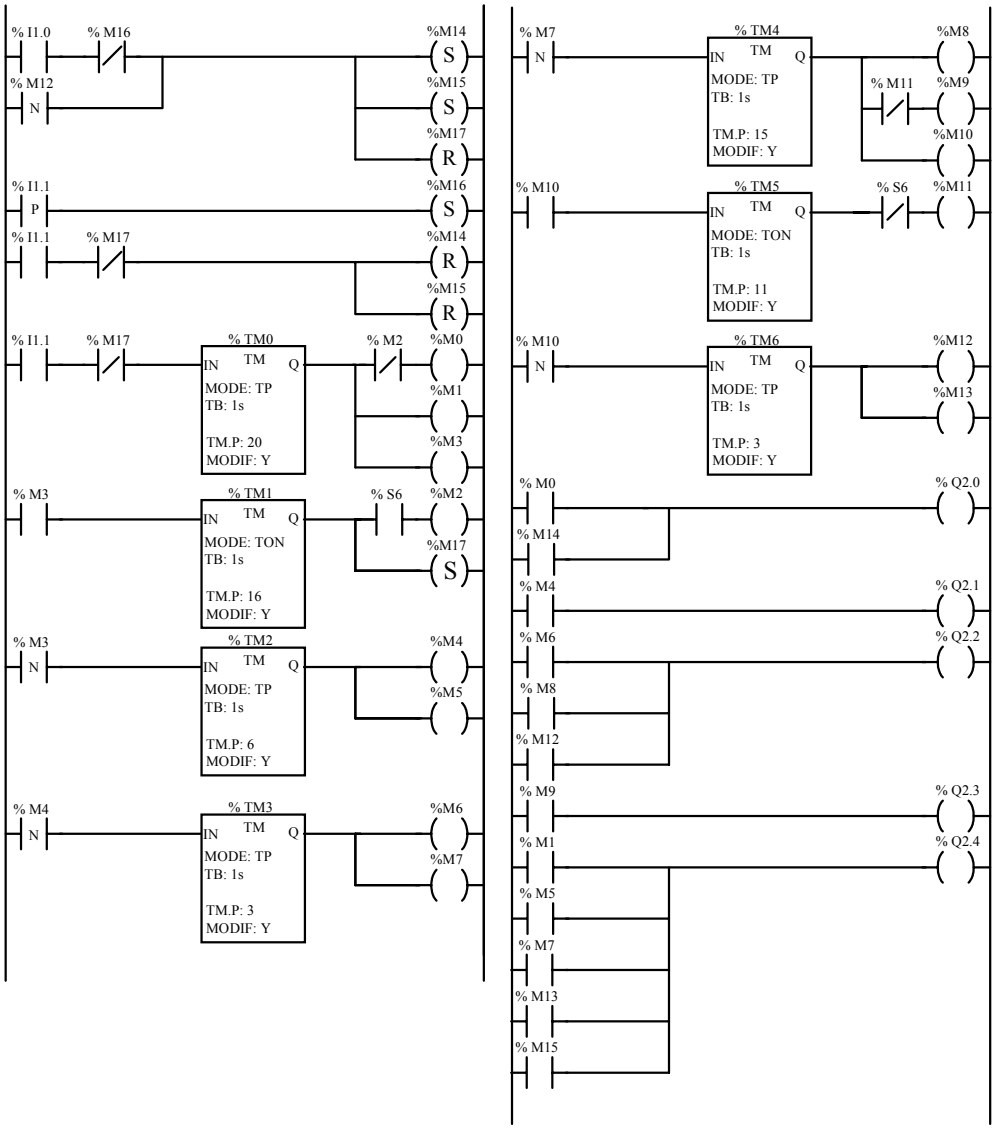


Figura 5.11: Linguagem ladder para passagem de pedestre.

5.3. Experimentos para simular uma linha de produção industrial

Nos experimentos a seguir (quinto, sexto e sétimo) a motivação principal é descrever como uma seqüência de comandos pode ser realizada em uma linha de produção industrial. Serão, para tanto, utilizadas duas esteiras e sensores para detectar os objetos para realização de controles como: parar por um determinado período de tempo, avançar e retroceder. Durante essas operações podem ser realizadas, em um ambiente industrial, operações de soldagem, pinturas, montagem de aparelhos eletrônicos, empacotamento etc e após a realização dessas atividades o processo continua normalmente.

(*i*) Quinto experimento

Utilizando-se apenas uma esteira, deve-se programa-la para que execute o seguinte movimento: quando ligada, a esteira deve avançar no sentido de A para B, quando um objeto é colocado sobre a esteira, em qualquer posição, esta passará a executar um movimento cíclico. Três são as possibilidades de o objeto ser colocado sobre a esteira: (i) o objeto é colocado na posição do sensor 1, sendo detectado por ele; (ii) colocado no meio da esteira, então ela deslocará o objeto até ser detectado pelo sensor 2 e (iii) o objeto é colocado na posição do sensor 2, sendo detectado por este outro sensor. Qualquer que seja a posição em que o objeto é colocado na esteira a seqüência de comandos deve ser a seguinte: (a) quando o sensor 1 detectar o objeto, a esteira deverá parar por 10s e em seguida avançar; (b) quando o objeto for detectado pelo sensor 2, a esteira deverá parar por 5s e em seguida retroceder. Repetindo um movimento periódico. A figura 5.12 ilustra a esteira e os sensores posicionados nos pontos A e B da esteira. A tabela 5.3 mostra os estados dos componentes da esteira. A figura 5.13 mostra uma modelagem por rede de Petri para o experimento proposto.

Serão desprezados em todos os experimentos com esteiras, os estados em que, no momento de acionamento das esteiras, tanto para avançar como também para retroceder, os sensores estejam detectando naquele instante; isto pelo fato da base de tempo ser desprezível. Contudo, este detalhe deve ser considerado na elaboração dos programas em linguagem ladder para que não ocorram problemas de conflito de comando.

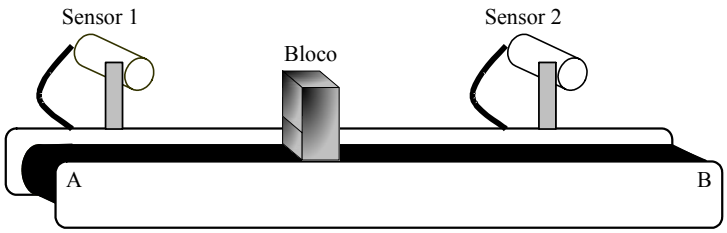


Figura 5.12: Esteira, posicionamento dos sensores.

Tabela 5.3: Evolução dos estados.

Estado dos componentes do experimento			
	Esteira	Sensor 1	Sensor 2
1	OFF	OFF	OFF
2	ON - Avançar	OFF	OFF
3	OFF	OFF	ON
4	ON - Retroceder	OFF	OFF
5	OFF	ON	OFF

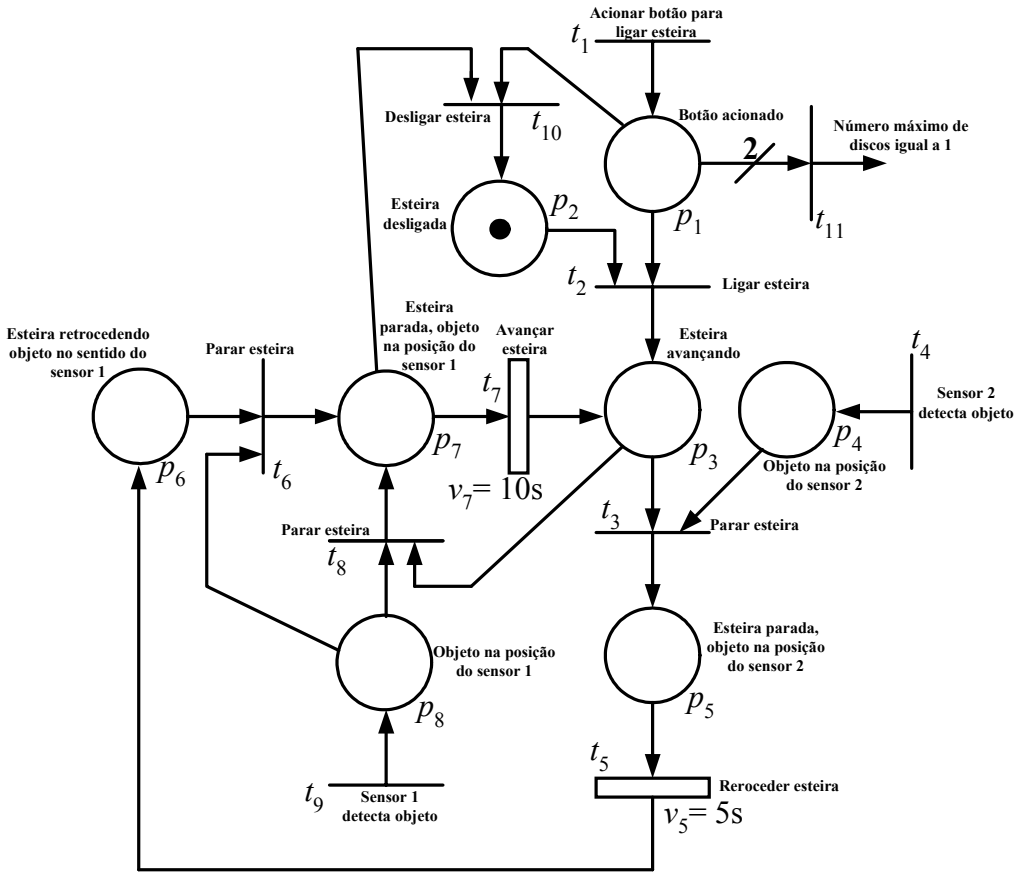


Figura 5.13: Rede de Petri para o quinto experimento.

Para se obter o modelo (P, T, A, w, x, **V**) da rede de Petri desse sistema, note que os eventos do sistema, considerando primeiro, que o objeto seja colocado no centro da esteira ou na posição do sensor 2, são: (i) ligar esteira; (ii); detecção do objeto pelo sensor 2; (iii) parada da esteira nas posições do sensor 2; (iv) retroceder esteira após 5s; (v) detecção do objeto pelo sensor 1; (vi) parar esteira na posição do sensor 1 e (vii) avançar esteira após 3s. Considerando, agora, que o objeto é colocado na posição do sensor 1, os eventos são: (i) ligar esteira; (ii); detecção do objeto pelo sensor 1; (iii) parada da esteira nas posições do sensor 1; (iv) avançar esteira após 10s; (v) detecção do objeto pelo sensor 2; (vi) parar esteira na posição do sensor 2 e (vii) retroceder esteira após 5s. Observa-se que a seqüência de eventos pode mudar, mas o movimento periódico permanece. Esses eventos dão origem ao conjunto $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}\}$. Os lugares, que representam as condições para que os eventos ocorram são: acionamento do botão (p_1), esteira desligada (p_2), esteira ligada e avançando (p_3), objeto na posição do sensor 2 (p_4), esteira parada na posição do sensor 2 (p_5), esteira retrocedendo (p_6), esteira parada na posição do sensor 1 (p_7), objeto na posição do sensor 1 (p_8), definindo, portanto, o conjunto $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$. Os elementos da rede de Petri são: $A=\{(p_1, t_2), (p_1, t_{10}), (p_1, t_{11}), (p_2, t_2), (p_3, t_3), (p_3, t_8), (p_4, t_3), (p_5, t_5), (p_6, t_6), (p_7, t_7), (p_7, t_{10}), (p_8, t_6), (p_8, t_8), (t_1, p_1), (t_2, p_3), (t_3, p_5), (t_4, p_4), (t_5, p_6), (t_6, p_7), (t_7, p_3), (t_8, p_7), (t_9, p_8), (t_{10}, p_2), (t_{11}, p_1)\}$, $w(p_1, t_2)=1$, $w(p_1, t_{10})=1$, $w(p_1, t_{11})=2$, $w(p_2, t_2)=1$, $w(p_3, t_3)=1$, $w(p_3, t_8)=1$, $w(p_4, t_3)=1$, $w(p_5, t_5)=1$, $w(p_6, t_6)=1$, $w(p_7, t_7)=1$, $w(p_7, t_{10})=1$, $w(p_8, t_6)=1$, $w(p_8, t_8)=1$, $w(t_1, p_1)=1$, $w(t_2, p_3)=1$, $w(t_3, p_5)=1$, $w(t_4, p_4)=1$, $w(t_5, p_6)=1$, $w(t_6, p_7)=1$, $w(t_7, p_3)=1$, $w(t_8, p_7)=1$, $w(t_9, p_8)=1$, $w(t_{10}, p_2)=1$, $w(t_{11}, p_1)=1$, $\underline{x}[\underline{x}(p_1), \underline{x}(p_2), \underline{x}(p_3), \underline{x}(p_4), \underline{x}(p_5), \underline{x}(p_6), \underline{x}(p_7), \underline{x}(p_8)]$, onde $\underline{x}_0=[0, 1, 0, 0, 0, 0, 0, 0]$ e $\underline{v}_5=\{v_{5,1}, v_{5,2}, v_{5,3}, ...\}= \{5, 5, 5, ...\}$, $\underline{v}_7=\{v_{7,1}, v_{7,2}, v_{7,3}, ...\}= \{10, 10, 10, ...\}$.

Uma vez caracterizados todos os elementos da rede de Petri, o próximo passo é obter a evolução dos estados. A partir da figura 5.13 pode-se obter a matriz de incidência $\mathbf{A}(j, i)_{11x8}$, onde $a_{ji}= w(t_j, p_i) – w(p_i, t_j)$. Portanto:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

definindo, portanto, $\underline{x}' = \underline{x} + \underline{u}A$. Assim sendo, para $\underline{x}_0$=[0, 1, 0, 0, 0, 0, 0, 0] e supondo que a esteira tenha sido ligada com o objeto sendo colocado no centro da esteira ou na posição do sensor 2, obtêm-se:

$\underline{x}_1 = \underline{x}_0 + [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [0, 0, 1, 0, 0, 0, 0, 0]$

$\underline{x}_2 = \underline{x}_1 + [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [0, 0, 1, 1, 0, 0, 0, 0]$

$\underline{x}_3 = \underline{x}_2 + [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [0, 0, 0, 0, 1, 0, 0, 0]$

$\underline{x}_4 = \underline{x}_3 + [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0].\mathbf{A} = [0, 0, 0, 0, 0, 1, 0, 0]$

$\underline{x}_5 = \underline{x}_4 + [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0].\mathbf{A} = [0, 0, 0, 0, 0, 1, 0, 1]$

$\underline{x}_6 = \underline{x}_5 + [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0].\mathbf{A} = [0, 0, 0, 0, 0, 0, 1, 0]$

$\underline{x}_7 = \underline{x}_6 + [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0].\mathbf{A} = [0, 0, 1, 0, 0, 0, 0, 0] = \underline{x}_1$, repetindo o ciclo.

seguindo esta seqüência, pode-se considerar um disparo da transição t_1, em qualquer instante após a esteira ser ligada, esse disparo solicita que a esteira seja desligada, contudo, a esteira só será desligada quando o objeto estiver na posição do sensor 1, conforme mostra a rede de Petri.

Observação 5.1: Para experimentos com esteiras, não serão mostradas as equações de estados para instantes de disparo das transições uma vez que irão depender da distância do ponto em que o objeto foi colocado na esteira. □

O programa em linguagem ladder para o experimento 5 é mostrado na figura 5.14, sendo %I1.0 é a chave para ligar e desligar a esteira, %I1.1 e %I1.2, respectivamente, os sensores 2 e 1, %Q2.0 e %Q2.1 são, rspectivamente, as saídas do CLP para ligar a esteira, Quando %Q2.0 está acionada a esteira avança e quando as duas saídas, %Q2.0 e %Q2.1, estão acionadas a esteira retrocede.

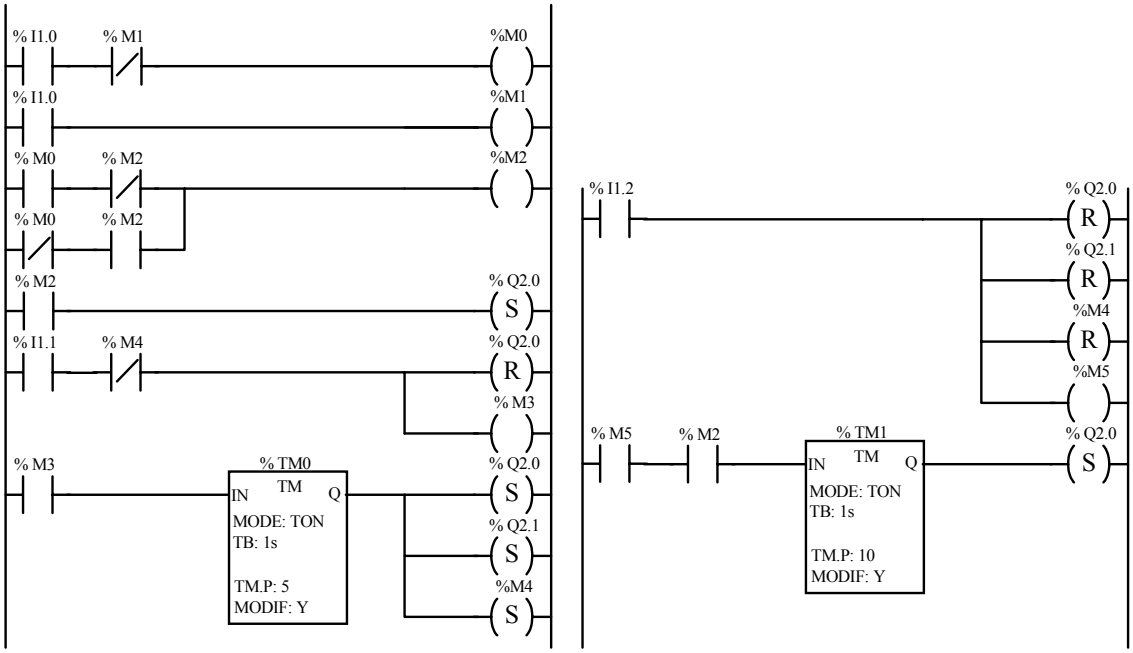


Figura 5.14: Linguagem ladder do quinto experimento.

(*ii*) Sexto experimento

Utilizando-se, agora, duas esteiras e três sensores, deve-se programar a esteira para executar o seguinte movimento: quando ligadas, as esteiras devem avançar no sentido de A para B. O objeto será sempre colocado no ponto A da esteira 1, e avançará no sentido da esteira 2 sendo detectado pelo sensor 1. Caso a esteira 2 esteja desocupada o objeto passará imediatamente para esta esteira e caso a esteira 2 esteja ocupada, o objeto deverá parar na posição do sensor 1 (desligando a esteira 1) e esperar até que a esteira 2 fique livre. Na esteira 2 o objeto quando detectado pelo sensor 2 deverá parar por 10s e, em seguida, a esteira avançará novamente levando o objeto até a posição do sensor 3, onde novamente a esteira 2 deverá parar por 5s. Após passar este intervalo de tempo o objeto é liberado da esteira 2 permitindo que a esteira 1 entre em funcionamento, colocando um novo objeto sobre a esteira 2. A figura 5.15 ilustra o posicionamento dos sensores deste experimento e a tabela 5.4 mostra a evolução dos estados. A figura 5.16 mostra a modelagem por rede de Petri para o experimento.

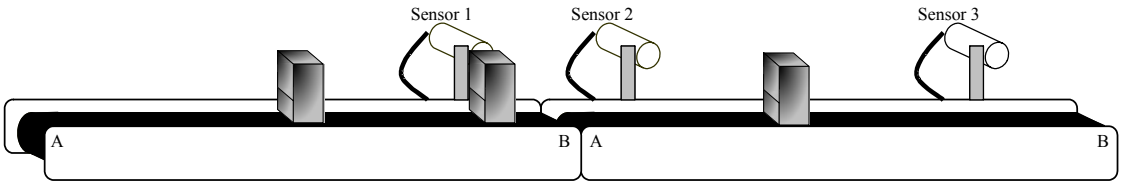


Figura 5.15: Montagem das esteiras para o experimento.

Tabela 5.4: Evolução dos estados.

Estado dos componentes do experimento						
	Esteira 1	Sensor 1	Esteira 2	Sensor 2	Sensor 3	Condição
1	OFF	OFF	OFF	OFF	OFF	Esteiras desligadas
2	ON - Avançar	OFF	ON - Avançar	OFF	OFF	Esteiras ligadas
3	ON - Avançar	ON	ON - Avançar	OFF	OFF	Primeiro passagem de bloco, esteira 2 liberada
4	ON - Avançar	OFF	OFF	ON	OFF	Esteira 2 parada com bloco na posição do sensor 2
5	OFF	ON	OFF	ON	OFF	Sensor 1 detecta objeto parando esteira 1
6	OFF	ON	ON - Avançar	OFF	OFF	Esteira 2 avançando
7	OFF	ON	OFF	OFF	ON	Esteira 2 parada com bloco na posição do sensor 3
8	ON - Avançar	OFF	ON - Avançar	OFF	OFF	Bloco sai da esteira 2 e esteira 1 envie outro bloco

Para se obter o modelo (P, T, A, w, x, V) da rede de Petri, note que os eventos do sistema são: (i) ligar esteiras; (ii); detecção do objeto pelo sensor 1; (iii) parar esteira 1 na detecção de um novo objeto pelo sensor 1; (iv) parar esteira 2 na detecção do objeto pelo sensor 2; (v) avançar esteira 2 após 5s; (vi) parar esteira 2 na detecção do objeto pelo sensor 3; (vii) avançar esteira 1 e 2 após 10s. Esses eventos dão origem ao conjunto $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$. Os lugares, que representam as condições para que os eventos ocorram são: esteira 1 avançando (p_1), objeto detectado pelo sensor 1 (p_2), esteira 2 com objeto (p_3), esteira 1 parada com objeto na posição do sensor 1 (p_4), esteira 2 livre (p_5), objeto detectado pelo sensor 2 (p_6), esteira 2 avançando (p_7), esteira 2 parada com objeto na posição do sensor 2 (p_8), objeto detectado pelo sensor 3 (p_9), esteira 2 parada com objeto na posição do sensor 3 (p_{10}), definindo, portanto, o conjunto $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}\}$. Os elementos da rede de Petri são: $A=\{(p_1, t_4), (p_2, t_4), (p_3, t_4), (p_4, t_5), (p_5, t_5), (p_6, t_6), (p_7, t_6), (p_7, t_9), (p_8, t_7), (p_9, t_9), (p_{10}, t_{10}), (t_1, p_1), (t_1, p_7), (t_2, p_2), (t_3, p_3), (t_3, p_6), (t_4, p_4), (t_5, p_1), (t_6, p_8), (t_7, p_7), (t_8, p_9), (t_9, p_{10})$, $(t_{10}, p_5), (t_{10}, p_7)\}$, $w(p_1, t_4)=1$, $w(p_2, t_4)=2$, $w(p_3, t_4)=1$, $w(p_4, t_5)=1$, $w(p_5, t_5)=1$, $w(p_6, t_6)=1$, $w(p_7, t_6)=1$, $w(p_7, t_9)=1$, $w(p_8, t_7)=1$, $w(p_9, t_9)=1$, $w(p_{10}, t_{10})=1$, $w(t_1, p_1)=1$, $w(t_1, p_7)=1$, $w(t_2, p_2)=1$, $w(t_3, p_3)=1$, $w(t_3, p_6)=1$, $w(t_4, p_4)=1$, $w(t_5, p_1)=1$, $w(t_6, p_8)=1$, $w(t_7, p_7)=1$, $w(t_8, p_9)=1$, $w(t_9, p_{10})=1$, $w(t_{10}, p_5)=1$, $w(t_{10}, p_7)=1$, $\underline{x}[\underline{x}(p_1), \underline{x}(p_2), \underline{x}(p_3), \underline{x}(p_4), \underline{x}(p_5), \underline{x}(p_6),$

$\underline{x}(p_7)$, $\underline{x}(p_8)$, $\underline{x}(p_9)$, $\underline{x}(p_{10})$], onde $\underline{x}_0$=[0, 0, 0, 0, 0, 0, 0, 0, 0, 0] e $\underline{v}_7$={$v_{7,1}$, $v_{7,2,}$ $v_{7,3,}$ ...}= {5, 5, 5, ...}, $\underline{v}_{10}$={$v_{10,1}$, $v_{10,2,}$ $v_{10,3,}$ ...}= {10, 10, 10, ...}.

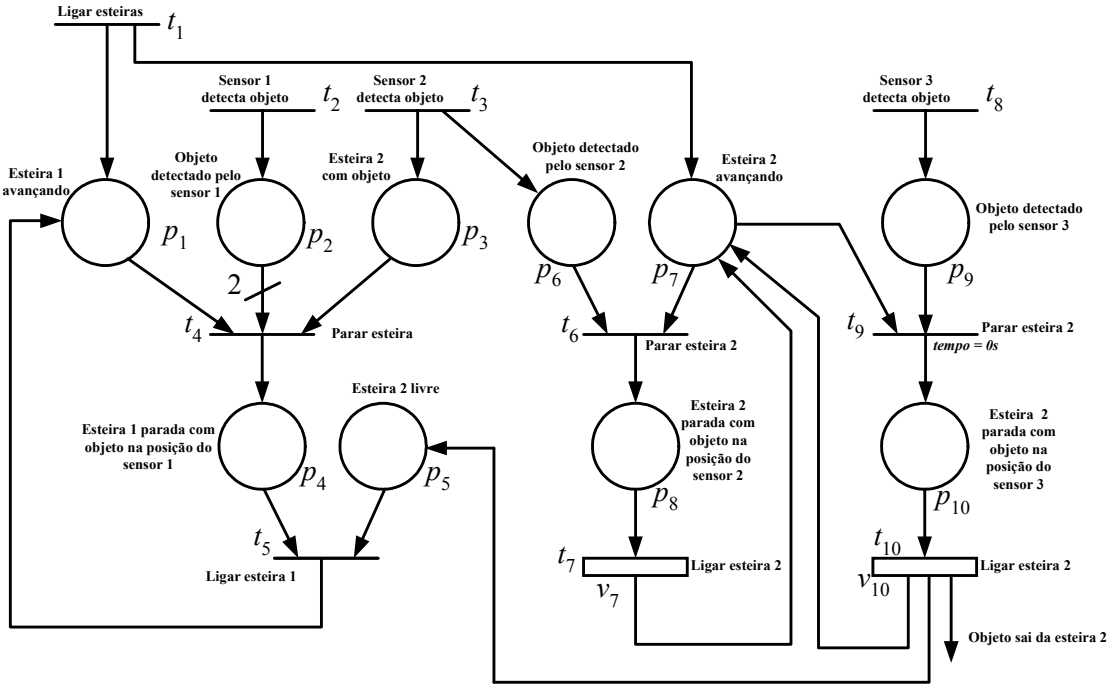


Figura 5.16: Rede de Petri para do sexto experimento.

O estado inicial desta rede de Petri é $\underline{x}_0$= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] . Por inspeção pode-se encontrar a matriz de incidência **A** da rede de Petri da figura 5.16, sendo dada por:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -2 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

Usando a matriz de incidência **A**, pode-se, a partir da equação de estados $\underline{x}' = \underline{x} + \underline{u}.A$, simular os resultados de todos os próximos estados que podem ser gerados a partir do disparo das transições.

A partir do disparo de t_1, que ocorre uma única vez, tendo em vista que se refere à operação de uma chave externa para ligar o sistema, as próximas transições que ocorrem, após o objeto ser colocado sobre a esteira 1, são: t_2, t_3, t_6, t_2, t_4, t_7, t_8, t_9, t_{10}, t_5, t_3, t_6, t_2, t_4,.... Quando o objeto passa para a esteira 2, imediatamente é colocado um novo objeto sobre a esteira 1.

Para este experimento, também, não será feita a decomposição do modelo dentro das dinâmicas de transições individuais, pois, as transições dos sensores dependem de fatores como: comprimento do objeto, arrasto da esteira, escorregamento da esteira devido ao peso do objeto, sensibilidade dos sensores na detecção dos objetos etc, tudo isso levaria a uma imprecisão na determinação dos tempos de disparos das transições.

Com o disparo de t_1 tem-se $\underline{u}_1$=[1, 0, 0, 0, 0, 0, 0, 0, 0, 0], então:

$\underline{x}_1 = \underline{x}_0 + [1, 0, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 0, 0, 0, 0, 0, 1, 0, 0, 0]$

$\underline{x}_2 = \underline{x}_1 + [0, 1, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 1, 0, 0, 0, 0, 1, 0, 0, 0]$

$\underline{x}_3 = \underline{x}_2 + [0, 0, 1, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 1, 1, 0, 0, 1, 1, 0, 0, 0]$

$\underline{x}_4 = \underline{x}_3 + [0, 0, 0, 0, 0, 1, 0, 0, 0, 0].\mathbf{A} = [1, 1, 1, 0, 0, 0, 0, 1, 0, 0]$

$\underline{x}_5 = \underline{x}_4 + [0, 1, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 2, 1, 0, 0, 0, 0, 1, 0, 0]$

$\underline{x}_6 = \underline{x}_5 + [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 0, 1, 0, 0]$

$\underline{x}_7 = \underline{x}_6 + [0, 0, 0, 0, 0, 0, 1, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 1, 0, 0, 0]$

$\underline{x}_8 = \underline{x}_7 + [0, 0, 0, 0, 0, 0, 0, 1, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 1, 0, 1, 0]$

$\underline{x}_9 = \underline{x}_8 + [0, 0, 0, 0, 0, 0, 0, 0, 1, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 0, 0, 0, 1]$

$\underline{x}_{10} = \underline{x}_9 + [0, 0, 0, 0, 0, 0, 0, 0, 0, 1].\mathbf{A} = [0, 1, 0, 1, 1, 0, 1, 0, 0, 0]$

$\underline{x}_{11} = \underline{x}_{10} + [0, 0, 0, 0, 1, 0, 0, 0, 0, 0].\mathbf{A} = [1, 1, 0, 0, 0, 0, 1, 0, 0, 0] = \underline{x}_2$

O programa a ser implementado no CLP para realização desse experimento está mostrado na figura 5.17, onde %I1.0 é a chave para ligar a esteira. %I1.1, %I1.2 %I1.3 são, respectivamente os sensores 1, 2 e 3, %Q2.0 e %Q2.1 são, respectivamente, as saídas do CLP para ligar as esteiras, Quando %Q2.0 está acionada a esteira 1 deve avançar e quando %Q2.1 está acionada a esteira 2 deve, também, avançar. Não existe movimento de retroceder neste experimento.

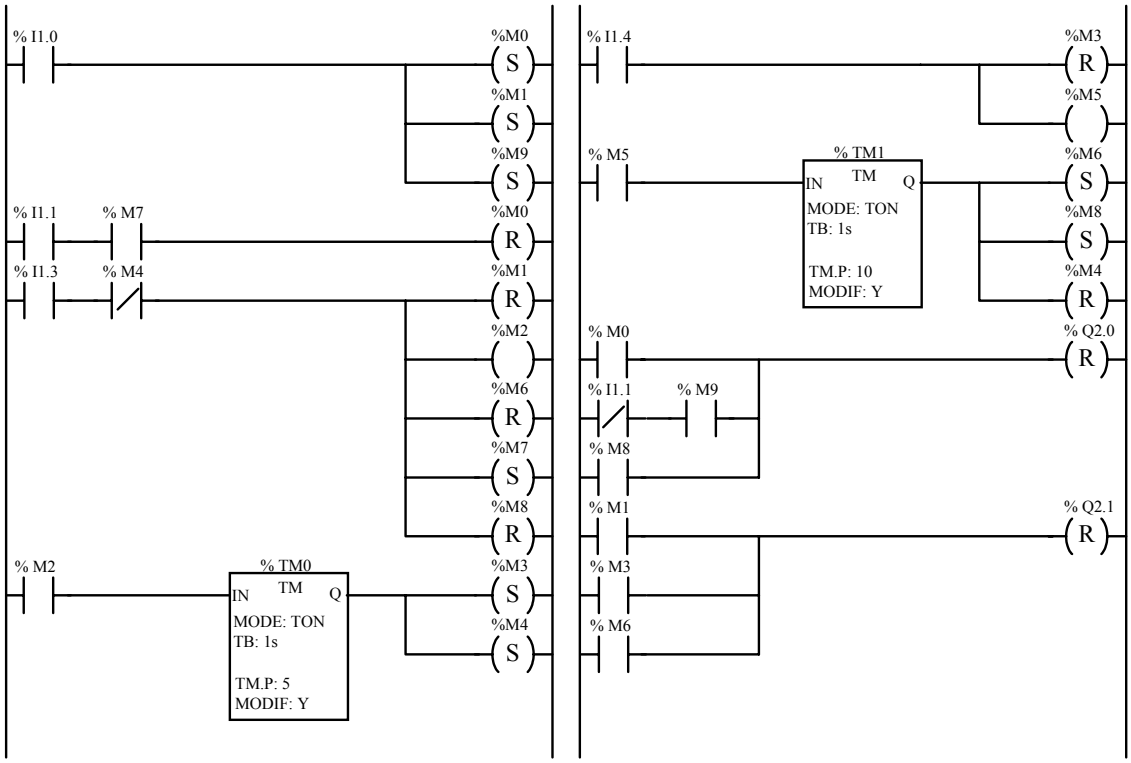


Figura 5.17: Linguagem ladder para o sexto experimento.

(*iii*) Sétimo experimento

Neste experimento serão utilizados duas esteiras e quatro sensores, devendo programa-las a fim de que executem os seguintes movimentos: quando ligadas, as esteiras devem avançar no sentido de A para B e quando um objeto é colocado no ponto A da esteira 1, este avançará no sentido da esteira 2 sendo detectado pelo sensor 1. Caso a esteira 2 esteja desocupada o objeto passará imediatamente para esta esteira e caso a esteira 2 esteja ocupada, o objeto deverá parar na posição do sensor 1 desligando a esteira 1 no momento em que o sensor 1 detectar um novo objeto e espera até que a esteira 2 fique livre. Na esteira 2 o objeto, quando detectado pelo sensor 3, deverá para por 5s e, em seguida, a esteira avançará novamente levando o objeto até a posição do sensor 4. Neste ponto novamente a esteira 2 deverá parar por 10s e, em seguida, a esteira deverá retroceder até a posição do sensor 3 e ali parar novamente por 5s. Logo após este intervalo de tempo, a esteira 2 avançará fazendo que o objeto seja detectado novamente pelo sensor 4 parando novamente por 10s. A quantidade de vezes que o movimento é repetido será determinada no CLP, em sua programação, através de um contador. Neste experimento esse número é 2, ou seja, após duas vezes detectado pelo sensor 4 a esteira 2 libera o objeto permitindo que a esteira 1 envie um novo objeto para

a esteira 2, para que um novo ciclo seja iniciado. Nesta nova entrada de objeto para a esteira 2 o sensor 2 tem a função de resetar o contador, para que o contador inicie uma nova contagem, sempre que um objeto passa por este. A figura 5.18 ilustra o posicionamento dos sensores deste experimento e a tabela 5.5 mostra a evolução dos estados. A figura 5.19 mostra a modelagem por rede de Petri para o experimento. A diferença deste experimento em relação ao experimento anterior é o fato que, agora, o objeto na esteira 2 ter que executar um ciclo repetitivo entre os dois pontos da esteira.

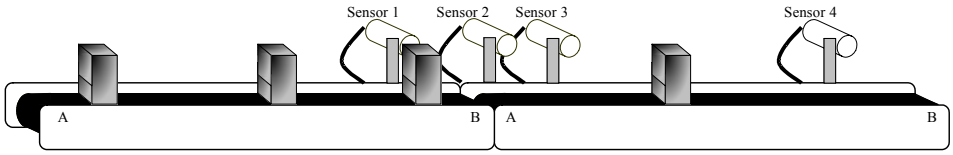


Figura 5.18: Esteiras do sétimo experimento.

Tabela 5.5: Evolução de estados do experimento

	Estado dos componentes do experimento						
	Esteira 1	Sensor 1	Esteira 2	Sensor 2	Sensor 3	Sensor 4	Condição
1	OFF	OFF	OFF	OFF	OFF	OFF	Esteiras deslígadas
2	ON - Avançar	OFF	ON - Avançar	OFF	OFF	OFF	Esteiras ligadas
3	ON - Avançar	ON	ON - Avançar	OFF	OFF	OFF	Sensor 1 detecta objeto, mas esteira 2 está livre
4	ON - Avançar	OFF	ON - Avançar	ON	OFF	OFF	Sensor 2 detecta objeto, esteira 2 ocupada
5	ON - Avançar	OFF	OFF	OFF	ON	OFF	Esteira 2 parada com objeto na posição do sensor 3
6	OFF	ON	OFF	OFF	ON	OFF	Esteira 1 parada com objeto na posição do sensor 1
7	OFF	ON	ON - Avançar	OFF	OFF	OFF	Esteira 2 avançando
8	OFF	ON	OFF	OFF	OFF	ON	Esteira 2 parada com objeto na posição do sensor 4
9	OFF	ON	ON - Retroceder	OFF	OFF	OFF	Esteira 2 retrocedendo
10	OFF	ON	OFF	OFF	ON	OFF	Esteira 2 parada com objeto na posição do sensor 3
11	OFF	ON	ON - Avançar	OFF	OFF	OFF	Esteira 2 avançando
12	OFF	ON	OFF	OFF	OFF	ON	Esteira 2 parada com objeto na posição do sensor 4
13	ON - Avançar	OFF	ON - Avançar	OFF	OFF	OFF	Esteira 2 libera objeto, esteira 1 envia outro objeto para esteira 2

Para se obter o modelo (*P, T, A, w, x*, V) da rede de Petri, note que os eventos do sistema são: (i) ligar esteiras; (ii); detecção do objeto pelo sensor 1; (iii) detecção do objeto pelo sensor 2; (iv) parar esteira 1 na detecção de um novo objeto pelo sensor 1; (v) parar esteira 2 na detecção do objeto pelo sensor 3; (vi) avançar esteira 2 após 5s; (vii) parar esteira 2 na detecção do objeto pelo sensor 4; (viii) retroceder esteira 2 após 10s; (ix) avançar esteira 2, após 10s, depois que o objeto for detectado pela segunda vez pelo sensor 4. Esses eventos dão origem ao conjunto $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}\}$. Os lugares, que representam as condições para que os eventos ocorram são: esteira 1 avançando (p_1), objeto detectado pelo sensor 1 (p_2), esteira 2 com objeto (p_3),

esteira 1 parada com objeto na posição do sensor 1 (p_4), esteira 2 livre (p_5), objeto detectado pelo sensor 3 (p_6), esteira 2 avançando (p_7), esteira 2 parada com objeto na posição do sensor 3 (p_8), objeto detectado pelo sensor 4 (p_9), esteira 2 parada com objeto na posição do sensor 4 (p_{10}), esteira 2 retrocedendo (p_{11}), contador do número de detecções do sensor 4 (p_{12}), indica o número de vezes que a esteira 2 deve retroceder após o disparo de t_2 (p_{13}), definindo, portanto, o conjunto $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}\}$. Os elementos da rede de Petri são: $A=\{(p_1, t_4), (p_2, t_4), (p_3, t_4), (p_4, t_5), (p_5, t_5), (p_6, t_7), (p_6, t_{12}), (p_7, t_7), (p_7, t_{10}), (p_8, t_8), (p_9, t_{10}), (p_{10}, t_{11}), (p_{10}, t_{13}), (p_{11}, t_{13}), (p_{12}, t_{13}), (p_{13}, t_{11}), (t_1, p_1), (t_1, p_7), (t_2, p_2), (t_3, p_3), (t_3, p_{13}), (t_4, p_4), (t_5, p_1), (t_6, p_6), (t_7, p_8), (t_8, p_7), (t_9, p_9), (t_9, p_{12}), (t_{10}, p_{10}), (t_{11}, p_{11}), (t_{12}, p_8), (t_{13}, p_5), (t_{13}, p_7)\}$, $w(p_1, t_4)=1$, $w(p_2, t_4)=2$, $w(p_3, t_4)=1$, $w(p_4, t_5)=1$, $w(p_5, t_5)=1$, $w(p_6, t_7)=1$, $w(p_6, t_{12})=1$, $w(p_7, t_7)=1$, $w(p_7, t_{10})=1$, $w(p_8, t_8)=1$, $w(p_9, t_{10})=1$, $w(p_{10}, t_{11})=1$, $w(p_{10}, t_{13})=1$, $w(p_{11}, t_{13})=1$, $w(p_{12}, t_{13})=2$, $w(p_{13}, t_{11})=1$, $w(t_1, p_1)=1$, $w(t_1, p_7)=1$, $w(t_2, p_2)=1$, $w(t_3, p_3)=1$, $w(t_3, p_{13})=1$, $w(t_4, p_4)=1$, $w(t_5, p_1)=1$, $w(t_6, p_6)=1$, $w(t_7, p_8)=1$, $w(t_8, p_7)=1$, $w(t_9, p_9)=1$, $w(t_9, p_{12})=1$, $w(t_{10}, p_{10})=1$, $w(t_{11}, p_{11})=1$, $w(t_{12}, p_8)=1$, $w(t_{13}, p_5)=1$, $w(t_{13}, p_7)=1$, $\underline{x}[\underline{x}(p_1), \underline{x}(p_2), \underline{x}(p_3), \underline{x}(p_4), \underline{x}(p_5), \underline{x}(p_6), \underline{x}(p_7), \underline{x}(p_8), \underline{x}(p_9), \underline{x}(p_{10}), \underline{x}(p_{11}), \underline{x}(p_{12}), \underline{x}(p_{13})]$, onde $\underline{x}_0=[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ e $\underline{v}_8=\{v_{8,1}, v_{8,2}, v_{8,3}, ...\}= \{5, 5, 5, ...\}$, $\underline{v}_{12}=\{v_{12,1}, v_{12,2}, v_{12,3}, ...\}= \{10, 10, 10, ...\}$, $\underline{v}_{13}=\{v_{13,1}, v_{13,2}, v_{13,3}, ...\}= \{10, 10, 10, ...\}$.

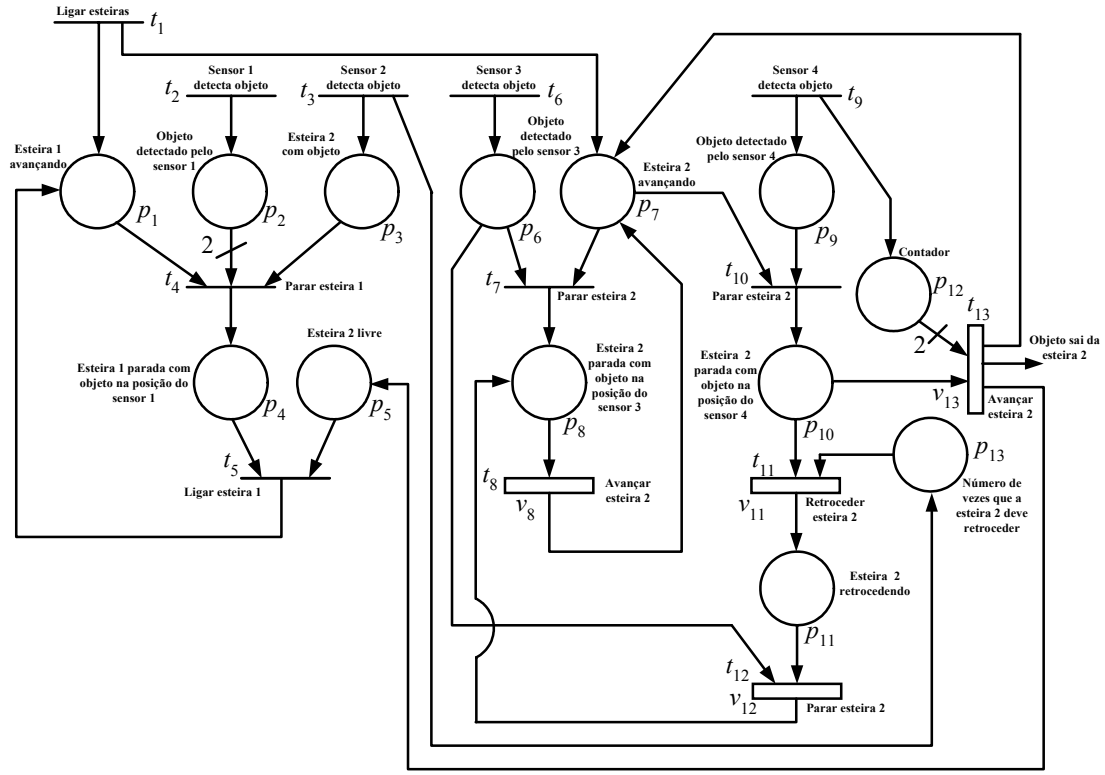


Figura 5.19: Rede de Petri do sétimo experimento.

Por inspeção da da rede de Petri da figura 5.19 pode-se encontrar a matriz de incidência **A**. Então, tem-se:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -2 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & -2 & 0 \end{bmatrix}$$

Usando a matriz de incidência **A**, pode-se, a partir da equação de estados, $\underline{x}' = \underline{x} + \underline{u}.A$, simular os resultados de todos os próximos estados que podem ser gerados a partir do disparo das transições.

Com o disparo de t_1 tem-se $\underline{u}_1$=[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], então:

$\underline{x}_1 = \underline{x}_0 + [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$

$\underline{x}_2 = \underline{x}_1 + [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$

$\underline{x}_3 = \underline{x}_2 + [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1]$

$\underline{x}_4 = \underline{x}_3 + [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1]$

$\underline{x}_5 = \underline{x}_4 + [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1]$

$\underline{x}_6 = \underline{x}_5 + [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 2, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1]$

$\underline{x}_7 = \underline{x}_6 + [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1]$

$\underline{x}_8 = \underline{x}_7 + [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1]$

$\underline{x}_9 = \underline{x}_8 + [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1]$

$\underline{x}_{10} = \underline{x}_9 + [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1]$

$\underline{x}_{11} = \underline{x}_{10} + [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0]$

$\underline{x}_{12} = \underline{x}_{11} + [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0]$

$\underline{x}_{13} = \underline{x}_{12} + [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0]$

$\underline{x}_{14} = \underline{x}_{13} + [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0]$

$\underline{x}_{15} = \underline{x}_{14} + [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 2, 0]$

$\underline{x}_{16} = \underline{x}_{15} + [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0].\mathbf{A} = [0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 2, 0]$

$\underline{x}_{17} = \underline{x}_{16} + [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1].\mathbf{A} = [0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0]$

$\underline{x}_{17} = \underline{x}_{16} + [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0].\mathbf{A} = [1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0] = \underline{x}_2$

Para este experimento, também, não será feita a decomposição do modelo dentro das dinâmicas de transições individuais, uma vez que o os disparos dos sensores não podem ser precisamente determinados.

O programa em linguagem ladder que deverá ser implementado no CLP para realização do experimento 5 é mostrado na figura 5.20, onde %I1.0 é a chave para ligar a esteira, %I1.1, %I1.2, %I1.3 e %I1.4 são, respectivamente, os sensores 1, 2, 3 e 4. %Q2.0, %Q2.1 e Q2.2 são, respectivamente, as saídas do CLP para ligar as esteiras. Quando %Q2.0 está acionada a esteira 1 deve avançar. %Q2.1 e %Q2.2 controlam o movimento da esteira 2, quando %Q2.1 está acionada a esteira 2 deve avançar e quando %Q2.1 e %Q2.1 estiverem acionadas a esteira 2 retrocede.

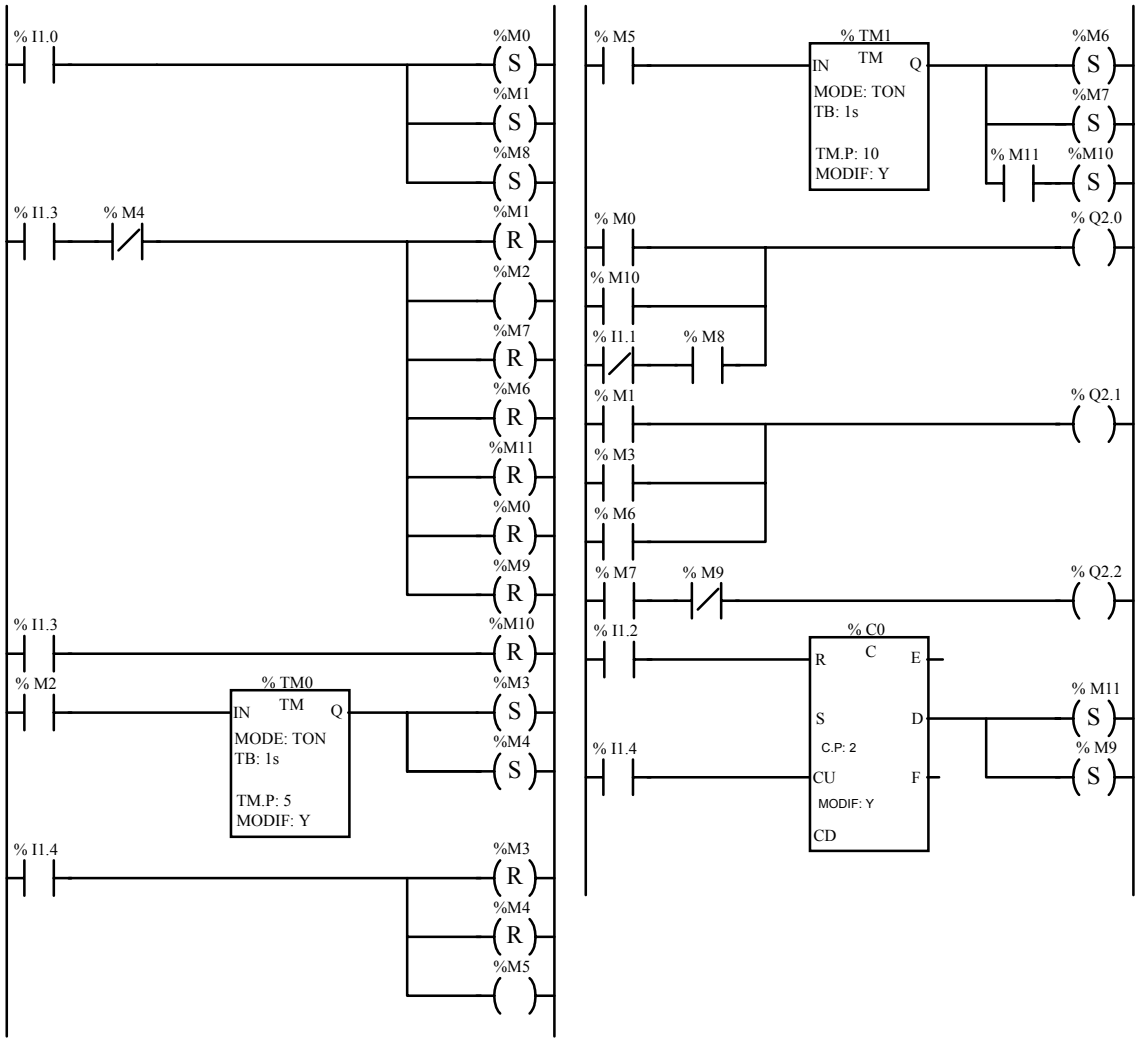


Figura 5.20: Linguagem ladder do sétimo experimento.

Capítulo 6. Conclusão e trabalhos futuros.

O grande aperfeiçoamento que há em equipamentos utilizados em automação, dentro dos seus mais variados setores da industria, comércio, médico-hospitalar etc, impedem que seja feito um modelo único de laboratório para a implantação de cursos na área de automação industrial. Dentro deste contexto, este trabalho teve por finalidade propor uma sistematização de vários experimentos que podem ser utilizados na implantação de um laboratório de automação industrial com base na teoria de sistemas a eventos discretos. Os experimentos aqui propostos, foram todos implantados na prática e podem ser utilizados como proposto, em qualquer outro laboratório, quer em cursos de automação, informática industrial ou controladores lógicos programáveis, podendo sofrer modificações e aperfeiçoamentos de acordo com o enfoque e características do curso ministrado. Em particular, o uso das esteiras com sensores e do conjunto de lâmpadas podem proporcionar uma variedade muito grande de experimentos.

Neste trabalho procurou-se mostrar os fundamentos principais de sistemas a eventos discretos, juntamente com suas formas de representação. A necessidade de ferramentas para modelagem de tais sistemas foi vista quando do estudo dos autômatos e das redes de Petri. Pôde-se observar a grande facilidade que essas ferramentas proporcionam para análise de sistemas a eventos discretos. Foi realizado um breve estudo entre conversões diretas entre redes de Petri e linguagem ladder, onde se pode notar as dificuldades em se trabalhar com estes tipos de conversões, as facilidades em documentar tais sistemas e a grande facilidade em se analisar e realizar manutenções em projetos elaborados em linguagem ladder, por este ser de difícil análise quando os programas são muito grandes e de difícil soluções de problemas quando estes acontecem.

Foram propostos sete experimentos para serem utilizados em um laboratório de automação industrial utilizando a teoria de sistemas a eventos discretos. Durante esses experimentos pôde-se perceber muitas dificuldades, podendo-se destacar: (i) a dificuldade de correção de programas em linguagens ladder à medida que os programas aumentam de tamanho; (ii) problemas de acionamentos contínuos de uma chave, quando se utiliza a modelagem por redes de Petri; (iii) dificuldade de transformação direta entre redes de Petri e linguagem ladder e (iv) representação clara da evolução dos

estados de um sistema, quando estes dependem de fatores externos a rede de Petri, como acionamento manual de botões, detecção por sensores etc.

A partir desse trabalho pôde-se observar uma grande vantagem do uso de redes de Petri dentro dos laboratórios de automação industrial, pela facilidade de compreensão da evolução dos estados do sistema (embora, em alguns casos de difícil representação) e por esta ferramenta proporcionar aos profissionais em formação uma ampla visão metodológica para o desenvolvimento, criação a análise de projetos envolvendo CLP. O grande crescimento da industria nacional depende e muito da capacidade de produção e geração de novos conceitos, estando as redes de Petri nesta linha de conhecimento. Assim, implantá-las nos cursos automação industrial pode representar um grande benefício para o estudo, desenvolvimento e expansão do ensino de sistemas a eventos discretos.

Em relação a trabalhos futuros que podem ser desenvolvidos, a partir deste trabalho, pode-se destacar: (i) a implantação destas mesmas experiências em um laboratório de sistemas a eventos discretos com a utilização de controle supervisório; (ii) incorporação do uso de sistemas hidráulicos e pneumáticos com suas respectivas formas de modelagem em conjunto com a utilização de redes de Petri ou autômato; (iii) métodos de transformação direta entre redes de Petri e linguagem ladder, com vistas aos grandes aperfeiçoamentos de linguagem de programação ladder que são oferecidos por cada fabricante de CLP; (iv) a implantação de experiências utilizando métodos estocásticos com ajuda de softwares de simulação de redes de Petri e (vi) desenvolvimento de ferramentas para a atualização remota desse laboratório via internet.

Referências Bibliográficas

ASEE: American Society of Engineering Education, "Engineering Education for a Changing World", *Report of a Joint Project of the ASEE Engineering Deans Council and Corporate Roundtable*, *Annual Conference Proceedings*, Edmonton, Canada, 1994.

ATTIÉ, S. S., *Automação Hidráulica e Pneumática Empregando a Teoria de Sistemas a Eventos Discretos*. Tese de M. Sc, Universidade Federal de Santa Catarina, Florianópolis, Brasil, 1998.

AUGUSTINE, N. R., *Augustine's Views, National Academy of Engineering*, The Bridge, vol. 23, nº 3, 1994.

BASILIO, J. C., "A laboratory for a First Course in Control System", *International Journal of Electrical Engineering*, Education vol. 39, 54-70, 2002.

BASILIO, J. C., MOREIRA, M. V., "State-space Parameter Identification in a Second Control Laboratory", *IEEE Transactions on Education E*, vol. 47, nº 2, 204-210, 2004.

BAUM, E., DOWELL, E., McTAGUE, J., HOCKER, J., "Engineering Education for a Changing World", *Joint Project by the Engineering Deans Council and the Corporate Roundtable*, American Society of Engineering Education, vol. 4, nº 4, 1994.

BIGNELL, J. W., DONOVAN, R. L., *Eletrônica Digital*, Editora Makron Books, vol. 1 e 2, São Paulo, 1995.

BORDOGNA, J., FROMM, E., ERNST, E., "Engineering Education: Innovation Through Integration", *Journal of Engineering Education*, vol. 82, nº 1, 3-8, 1993.

BOUCHER, T. O., JAFARI, M. A., MEREDITH, G. A., "Petri Net Control of an Automated Manufacturing Cell", *Computers in Industrial Engineering,* vol. 17, Orlando, FL, USA 459-463, 1989.

BURNS, G. L., BINDANDA, B., "The Use of Hierarchical Petri Nets for the Automatic Generation of Ladder Logic Programs", *Proceedings ESD IPC-94 Conference & Exposition*, 169-179, Detroit, Michigan, 1994.

CASSANDRAS, C. G.,LAFORTUNE, S., *Introduction to Discrete Event Systems*. Kluwer Academic Publishers. 2º Edição. Boston, 2000.

CAZOLA, F., FERRARINI, L., PREZIOSA, M., "Interpretation rules of PETRI NET models for logic control", *Proc. IEEE Symp. Emerging Technologies and Factory Automation (ETFA'95),* vol. 2, 289-297, 1995.

CHIRN, J. L., McFARLANE, D. C., "Petri net based design of ladder logic diagrams", *Internal Report, Institute for Manufacturing,* Cambridge University, England, 2000.

CIBUZAR, G., STAEGE, J., NORDELL, J., RAU, A., BUSENBARRICK, D., "MTL Intranet: A University Microelectronics Laboratory WWW-based Management System", *IEEE Transactions on Industrial Electronics*, 92-95, 2001.

DAVID, R., "Grafcet: A Powerful Tool for Specification of Logic Controllers", *IEEE Transactions on Control Systems Technology*, vol. 3, nº 3, 253-268, 1995.

GEORGINI, M., *Automação Aplicada*. Editora Érica. 1º Edição. São Paulo, 2000.

GRAYSON, L. P. (Ed.), "Educating Engineers for World Competition" *Proc. 24th Frontiers in Education Conference*, *ASEE-IEEE*, 233-236, San Jose, CA, 1994.

IEC: International Electrotechnical Commission. International standard IEC 1131-3, Programmable Controllers, Part 3: Programming Languages. Geneva, 1992.

JAFARI, M., BOUCHER, T. O., "A rule-based system for generating a ladder logic control program from a high-level systems model", *Journal of Intelligent Manufacturing*, vol. 5, nº. 2, 103-120, 1994.

JIMÉNES, I., LOPÉS, E., RAMÍRES, A., "Synthesis of Ladder Diagrams from Petri Nets Controller Models", *Proceedings of the 2001 IEEE International Symposium on Intelligent Control,* 225-230, México City, México, 2001.

JONES, A. H., UZAM, M., AJLOUNI, N., "Design of discrete event control systems for programmable logic controllers using T-timed Petri nets", *Proc. 1996 IEEE Int. Symp. Computer-Aided Control System Design,* Dearborn, MI, 212-217, 1996.

KUMAR, R., GARG, V. K., *Modeling and Control of Logical Discrete Event Systems*, research monograph, Kluwer Academic Publishers, Norwell Massachusetts, 1995.

LAUZON, S. C., Ma, A. K. L., MILLS, J. K., BENHABIB, B., "Application of Discrete Event System Theory to flexible manufacturing", *IEEE Control Systems,* 41-48, 1996.

LEE, J. S., HSU, P. L., "A New Approach to Evaluate Ladder Logic Diagrams and Petri Nets via the IF-THEN Transformation", *IEEE Conference on Systems, Man and Cybernetics*, 2711-2716, Tucson, AZ, 2001.

LUCAS, M. R., TILBURY, D. M., "A study of current logic design practices in the automotive manufacturing industry", *International Journal of Human-Computer Studies*, vol. 59, 725-753, 2003.

MARTIN, T. W., BROWN, W. D., "A Downsized, Laboratory-Intensive Curriculum in Electrical Engineering", *Proceedings of the 1997 Frontiers in Education Conference*, 878-882, 1997.

MIDDLETON, N. T., GLASER, S., GOSINK, J. P., PARKER, T., "An Integrated Engineering Systems Laboratory", *Proceeding of the 1996 Frontiers in Education Conference* , 651-655, 1996.

MIYAGI, P. E., *Controle Programável*. Editora Edgard Blücher. São Paulo, 1996.

MORAES, C. C., CASTRUCCI, P. L., "Um Programa Didático em Automação Industrial", *Anais do XIV - Congresso Brasileiro de Automática*, pp. 1397-1402, Natal, Brasil, Set., 2002.

NATALE, F., *Automação Industrial*. Ed. Érica. 2º Edição. São Paulo, 1995.

OLIVEIRA, J. C. P., *Controlador Lógico Programável*. Editora Makron Books. São Paulo, 1993.

OMRON, *C200H Programmable Controllers (CPU01-E/03- E/11-E) Operation Manual*, Omron Co., 1994.

PETERSON, J. L., *Petri net theory and the modeling of systems*, Prentice Hall, 1981.

PENG, S. S., ZHOU, M. C., "Ladder Diagram and Petri Net Based Discrete Event Control Design Methods", *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 1-9, 2004.

PENG, S. S., ZHOU, M. C., "Conversion between Ladder Diagrams and Petri Net in Discrete Event Control Design – A Survey", In: *IEEE Conference on Systems, Man and Cybernetics*, Tucson, AZ, 2682-2687, 2001.

QUINN, R. G., "The E4 Introductory Engineering Test, Design and Simulation Laboratory", *Journal of Engineering Education*, vol. 82, nº 4, 1993.

SATO, T., NOSE, K., "Automatic Generation of Sequence Control Program via Petri Nets and Logic Tables for Industrial Applications", *Petri Nets in Flexible and Ag. Automation*, Kluwer Academic Publishers, 93-108, 1995.

SILVEIRA, P. R., SANTOS, W. E., , *Automação e Controle Discreto*. Editora Érica. 1º edição. São Paulo, 1999.

TAHOLAKIAN, A. HALES, W. M. M., “PN/PLC: A methodology for designing, simulating and coding PLC based control systems using Petri nets”, *International Journal of Product Research*, , vol. 35, nº 6, 1743-1762, 1997.

UZAM, M., JONES, A., AJLOUNI, N., “Conversion of Petri nets controllers for manufacturing systems into ladder logic diagrams", *IEEE Symposium on Emerging Technology and Factory Automation,* ETFA, vol. 2, 649-655, 1996.

VENKATESH, K., ZHOU, M.C., *Modeling, Simulation, and control of Flexible manufacturing Systems – A Petri Net Approach*, World Scientific Publishers*,* Singapore, 1998.

VENKATESH, K, ZHOU, M. C., CAUDILL, R. J., “Comparing Ladder Logic Diagrams and Petri Nets for Sequence Controller Design Through a Discrete Manufacturing System”, *IEEE Transactions on Industrial Electronics,* vol. 41, nº 6, 611-619, 1994*a*.

VENKATESH, K, ZHOU, M. C., CAUDILL, R. J., “Evaluating the Complexity of Petri Nets and Ladder Logic Diagrams and for Sequence Controllers Design in Flexible Automation” *Proceedings of the 1994 IEEE Symposium on Emerging Technologies e factory Automation,* 428-435, 1994*b*.

ZURAWSKI, R., ZHOU, M. C., “Petri Nets and Industrial Application: A Tutorial”, *IEEE Transactions on Industrial Electronics*, vol. 41, nº 6, 567-583, 1994.