UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Gabriel Miranda Freitas

Implementação de Controle Supervisório em CLP de um Sistema Flexível de Manufatura com Aplicação em Planta Física e Virtual

Belo Horizonte

2024

Gabriel Miranda Freitas

Implementação de Controle Supervisório em CLP de um Sistema Flexível de Manufatura com Aplicação em Planta Física e Virtual

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito para obtenção do título de Mestre em Engenharia Elétrica.

Orientadora: Profª. Patrícia Nascimento Pena

Belo Horizonte
2024

Freitas, Gabriel Miranda.
F866i Implementação de controle supervisório em CLP de um sistema flexível de manufatura com aplicação em planta física e virtual [recurso eletrônico] / Gabriel Miranda Freitas. - 2024.
1 recurso online (79 f. : il., color.) : pdf.

Orientadora: Patrícia Nascimento Pena.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos e apêndices: f. 74-79.

Bibliografia: f. 71-73.

1. Engenharia elétrica - Teses. 2. Teoria do controle - Teses. 3. Controladores programáveis - Teses I. Pena, Patrícia Nascimento. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo bibliotecário Reginaldo César Vital dos Santos CRB/6 2165
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

ESCOLA DE ENGENHARIA

COLEGIADO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"Implementação de Controle Supervisório em Clp de Um Sistema Flexível de Manufatura Com Aplicação em Planta Física e Virtual"

Gabriel Miranda Freitas

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica. Aprovada em 21 de junho de 2024. Por:

Profa. Dr. Patrícia Nascimento Pena
DELT (UFMG) - Orientadora

Prof. Dr. Carlos Andrey Maia
DEE (UFMG)

Prof. Dr. Carmela Maria Polito Braga
DELT (Universidade Federal de Minas Gerais)

Documento assinado eletronicamente por **Patricia Nascimento Pena**, **Professora do Magistério Superior**, em 21/06/2024, às 18:44, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Carlos Andrey Maia**, **Professor do Magistério Superior**, em 25/06/2024, às 12:46, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Carmela Maria Polito Braga**, **Professora do Magistério Superior**, em 25/06/2024, às 13:45, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site

https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3305985** e o código CRC **42B62845**.

AGRADECIMENTOS

Agradeço a Deus, por abençoar toda a minha trajetória até aqui.

Aos meus pais Antônio e Mônica e aos meus irmãos Raquel e Daniel, pelo incentivo e confiança.

À minha companheira Jéssica, pela lealdade e apoio incondicional.

À prof.ª Patrícia Pena, pela paciência, inspiração e exemplo.

Aos colegas do LACSED, pela ótima convivência e companheirismo.

Agradeço à UFMG, ao PPGEE e, especialmente, ao IFMG, que, através do seu programa de afastamento docente, permitiu minha dedicação integral, viabilizando o desenvolvimento deste trabalho.

"E ainda que tivesse o dom de profecia, e conhecesse todos os mistérios e toda a ciência, e ainda que tivesse toda a fé, de maneira tal que transportasse os montes, e não tivesse amor, nada seria. (I Coríntios 13:2)"

RESUMO

Com o avanço tecnológico das últimas décadas, e o desenvolvimento de inúmeros sistemas dirigidos a eventos discretos, como sistemas flexíveis de manufatura, filas de serviços, sistemas de comunicação, de tráfego urbano, robótica, dentre outros, surgiram técnicas de controle formais para esta classe de sistemas, com regras lógicas, baseadas em eventos, que determinam sua evolução. A Teoria de Controle Supervisório é um método de controle formal para SEDs que sintetiza um supervisor não bloqueante e minimamente restritivo, que desabilita a ocorrência de eventos, modelando o comportamento do sistema de maneira a obedecer determinadas especificações. Os controladores lógicos programáveis constituem-se, ainda nos dias de hoje, como a principal solução tecnológica adotada no controle de sistemas automatizados. Diversas técnicas de implementação dos supervisores, obtidos por meio da TCS, em CLP, foram desenvolvidas e apresentadas nas últimas décadas, e lidam com limitações cíclicas dos circuitos eletrônicos. Além disso, com o surgimento de tecnologias como comissionamento virtual, sistemas ciber físicos e gêmeos digitais, novas soluções que permitem o desenvolvimento de modelos virtuais, capazes de simular o comportamento dinâmico de plantas industriais, têm se tornado cada vez mais presentes neste contexto. Este trabalho realiza o estudo, modelagem e implementação do controle supervisório em CLP, utilizando como estudo de caso um sistema flexível de manufatura. Diferentes metodologias são investigadas e problemas típicos de implementação são apresentados e possíveis soluções discutidas. São desenvolvidas três implementações com estruturas distintas, aplicadas em: um modelo virtual desenvolvido em software de simulação 3D; um protótipo didático do sistema; um modelo composto com sincronização entre planta física e virtual. Também é apresentada uma solução alternativa, quanto à política de geração de eventos controláveis, pela estrutura de controle. Os resultados obtidos demonstram o sucesso de cada proposta, bem como os desafios e limitações encontrados.

Palavras-chave: teoria de controle supervisório; controlador lógico programável; sistemas a eventos discretos; laboratório virtual.

ABSTRACT

With the technological advancement of recent decades and the development of numerous discrete event-driven systems, such as flexible manufacturing systems, service queues, communication systems, urban traffic systems, robotics, among others, formal control techniques for this class of systems have emerged, based on logical, event-driven rules that determine their evolution. Supervisory Control Theory is a formal control method for DES that synthesizes a non-blocking and minimally restrictive supervisor, which disables the occurrence of events, modeling the system's behavior to comply with certain specifications. Programmable logic controllers remain, even today, the main technological solution adopted in the control of automated systems. Various techniques for implementing supervisors, obtained through SCT, in PLC, have been developed and presented over the past decades, dealing with cyclical limitations of electronic circuits. Furthermore, with the emergence of technologies such as virtual commissioning, cyber-physical systems, and digital twins, new solutions that allow the development of virtual models capable of simulating the dynamic behavior of industrial plants have become increasingly prevalent in this context. This work conducts the study, modeling, and implementation of supervisory control in PLC, using a flexible manufacturing system as a case study. Different methodologies are investigated, and typical implementation problems are presented, with possible solutions discussed. Three implementations with distinct structures are developed, applied to: a virtual model developed in 3D simulation software; an educational prototype of the system; and a composite model with synchronization between physical and virtual plants. An alternative solution regarding the policy of generating controllable events by the control structure is also presented. The obtained results demonstrate the success of each proposal, as well as the challenges and limitations encountered.

Keywords: supervisory control theory; programmable logic controller; discrete event systems, virtual lab.

LISTA DE FIGURAS

Figura 1 – Autômato de estados finitos 23
Figura 2 – Composição síncrona entre autômatos 24
Figura 3 – Estrutura do controle monolítico 26
Figura 4 – Estrutura do controle modular local 26
Figura 5 – Implementação de um autômato em LD 29
Figura 6 – Geração de eventos em aplicações práticas 30
Figura 7 – Exemplo do efeito avalanche 30
Figura 8 – Arquitetura proposta por Queiroz e Cury (2002) 32
Figura 9 – Autômatos exemplificando: (a) o comportamento assíncrono de uma máquina genérica; (b) supervisor hipotético que implementa um comportamento desejado 33
Figura 10 – Implementação genérica do Sistema Produto em LD segundo (QUEIROZ; CURY, 2002) 34
Figura 11 – Implementação genérica do supervisor em LD segundo (QUEIROZ; CURY, 2002) 34
Figura 12 – Etapas da rotina principal ordenadas pela controlabilidade dos eventos (LEAL; CRUZ; HOUNSELL, 2012) 35
Figura 13 – Implementação em LD das etapas da rotina principal ordenadas pela controlabilidade dos eventos (LEAL; CRUZ; HOUNSELL, 2012) 36
Figura 14 – Sistema Flexível de Manufatura (SFM). Adaptado de (QUEIROZ; CURY; WONHAM, 2005) 37
Figura 15 – Planta didática LACSED 39
Figura 16 – (a) Encoder óptico de quadratura E2-Q2; (b) Sensores ópticos reflexivos. 39
Figura 17 – (a) Pick & Place 3D equivalente ao robô e à máquina de montagem; (b) Pick & Place 2D equivalente à máquina de pintura 41
Figura 18 – Machine center equivalente à fresa e ao torno 42
Figura 19 – Resultado final do Modelo I do SFM virtual 42
Figura 20 – Autômatos que modelam o comportamento de cada máquina 44
Figura 21 – Autômatos que modelam as especificações de controle 45
Figura 22 – Autômatos dos supervisores modulares locais reduzidos 46
Figura 23 – Arquitetura da Implementação I 47
Figura 24 – Fluxograma com as etapas de implementação do código 48
Figura 25 – Descrição em LD do autômato G_4 que modela o comportamento em malha aberta da fresa e legenda das TAGs 48
Figura 26 – Descrição em LD: (a) Supervisor S_{3red}; (b) Desabilitações 49
Figura 27 – Representação da comunicação via OPC UA 50

Figura 28 – Endereçamento dos elementos de campo com as variáveis globais 50
Figura 29 – *Set points* com variáveis contínuas do Pick & Place 3D 51
Figura 30 – Diferentes blocos brutos de entrada no sistema 52
Figura 31 – Arquitetura da Implementação II. Adaptado de (QUEIROZ; CURY, 2002) 53
Figura 32 – Fluxograma com as etapas da estrutura "DECON9" (LEAL; CRUZ; HOUNSELL, 2012) 54
Figura 33 – (a) Identificação nas transições dos sinais de entrada do CLP; (b) Rotina de espelhamento ou resgate dos eventos não controláveis 56
Figura 34 – Atualização do SP e dos supervisores com o resgate dos eventos não controláveis 56
Figura 35 – Tratamento da escolha: (a) etapa na rotina principal; (b) sub-rotina "Escolha51e53" 57
Figura 36 – Visão superior da planta com os pontos de posicionamento do eixo de giro do robô 58
Figura 37 – Solução com sub-rotinas a partir do último evento do robô 59
Figura 38 – Arquitetura da Implementação III 62
Figura 39 – Fluxograma com as etapas da Implementação III, adaptado de (LEAL; CRUZ; HOUNSELL, 2012) 63
Figura 40 – Sincronização utilizando blocos de entrada e saída de peças na planta virtual 64
Figura 41 – Fluxograma modificado com a rotina adicional 67
Figura 42 – Rotina de atualização do SP com base no evento controlável gerado . . 68

LISTA DE TABELAS

Tabela 1 – Classificação das linguagens de programação, conforme IEC 61131-3 . . 28
Tabela 2 – Funções básicas do LD (IEC 61131-3) 28
Tabela 3 – Lista de dispositivos 39
Tabela 4 – Eventos de cada máquina do SFM. 43
Tabela 5 – Lista das sub-rotinas para os eventos do Robô 60
Tabela 6 – Intervalo de tempo entre eventos (em unidades de tempo - u.t.) 61

LISTA DE ABREVIATURAS E SIGLAS

CLP	Controlador Lógico Programável
DELT	Departamento de Engenharia Eletrônica
E/S	Entradas/Saídas
LACSED	Laboratório de Análise e Controle de Sistemas a Eventos Discretos
LD	*Ladder*
MM	Máquina de Montagem
MP	Máquina de Pintura
OPC DA	*Open Platform Communications Data Access*
OPC UA	*Open Platform Communications Unified Architecture*
RPI	*Request Packet Interval*
SED	Sistemas a Eventos Discretos
SFC	*Sequential Function Charts*
SFM	Sistema Flexível de Manufatura
SO	Sequência Operacional
SP	Sistema Produto
ST	*Structured Text*
TCS	Teoria de Controle Supervisório

LISTA DE SÍMBOLOS

δ	função de transição
ϵ	cadeia vazia
G	autômato finito determinístico
Γ	função de eventos ativos
L	linguagem
$\overline{L}$	prefixo fechamento de uma linguagem
$\mathcal{L}(G)$	linguagem gerada de um autômato G
$\mathcal{L}_m(G)$	linguagem marcada de um autômato G
q_0	estado inicial
Q	conjunto de estados
Q_m	conjunto de estados marcados
σ	evento de um alfabeto
Σ	conjunto de eventos (alfabeto)
Σ^*	fechamento Kleene
Σ_c	conjunto de eventos controláveis
Σ_{uc}	conjunto de eventos não controláveis
$trim(G)$	autômato G aparado
$SupC(K, G)$	máxima sublinguagem controlável em relação a G contida em K
$\|\|$	composição síncrona de autômatos
$\cup$	união de conjuntos
$\cap$	interseção de conjuntos
E	especificações de controle
S	supervisor

SUMÁRIO

1 INTRODUÇÃO 16
1.1 Objetivo Geral 19
1.2 Objetivos Específicos 19
1.3 Organização do Texto 20
2 FUNDAMENTAÇÃO TEÓRICA 21
2.1 Sistemas a Eventos Discretos 21
2.2 Linguagens e Autômatos 22
2.3 Teoria de Controle Supervisório 24
2.4 Revisão Sobre a Implementação do Controle Supervisório em CLP 27
2.4.1 Controladores Lógicos Programáveis 27
2.4.2 Implementação da TCS em CLP 28
2.4.2.1 Causalidade 29
2.4.2.2 Sinais e Eventos 30
2.4.2.3 Sincronização Inexata 31
2.4.2.4 Escolha de Eventos 31
2.4.3 Metodologias de Implementação da TCS em CLP 31
2.4.3.1 Queiroz e Cury (2002) 32
2.4.3.2 Leal, Cruz e Hounsell (2012) 35
3 ESTUDO DE CASO: O SISTEMA FLEXÍVEL DE MANUFATURA 37
3.1 Detalhamento do Sistema 37
3.2 Planta Didática 38
3.2.1 Encoder e Sensores 39
3.3 Planta Virtual 40
3.3.1 Delineamento do SFM 3D 41
3.4 Controle Supervisório 42
3.4.1 Modelagem das Máquinas 42
3.4.2 Modelagem das Especificações 43
3.4.3 Síntese dos Supervisores 44
4 IMPLEMENTAÇÃO I 47
4.1 Metodologia de Implementação 47
4.2 Codesys e Tecnologia de Comunicação 49
4.3 Considerações sobre o Modelo Virtual 50
4.4 Resultados 52
5 IMPLEMENTAÇÃO II 53
5.1 Metodologia de Implementação 53

5.2 Represamento do Efeito Avalanche . . . 55
5.3 Tratamento da Escolha . . . 57
5.4 Detalhamento da Rotina do Robô . . . 58
5.5 Resultados . . . 59
6 IMPLEMENTAÇÃO III . . . 62
6.1 Metodologia de Implementação . . . 62
6.2 Sincronização entre Plantas Física e Virtual . . . 64
6.3 Tratamento da Escolha . . . 65
6.4 Resultados . . . 65
7 GERAÇÃO EXTERNA DE EVENTOS CONTROLÁVEIS . . . 66
8 CONCLUSÃO . . . 69
8.1 Perspectivas para Trabalhos Futuros . . . 70
REFERÊNCIAS . . . 71
Apêndice A . . . 74
Anexo A . . . 76

1 INTRODUÇÃO

Historicamente, o homem sempre buscou simplificar seu trabalho de forma a substituir o esforço braçal por outros meios e mecanismos, com o objetivo primário de otimizar o seu tempo. A primeira Revolução Industrial, também conhecida como a "revolução das máquinas", em meados do século XVIII, estabeleceu um marco transitório de uma sociedade que até então produzia bens de consumo de forma artesanal, para uma sociedade organizada dentro de um novo modelo, industrialmente mais produtiva (SILVEIRA; SANTOS, 2009).

Com o passar dos anos, houve avanços contínuos nas tecnologias de automação, onde a produção em massa começou a se consolidar, com a implementação de linhas de montagem e sistemas de transporte de materiais, sistemas de controle computadorizado, automação de processos complexos por meio de lógica programável, etc. Nas últimas décadas, com o desenvolvimento de inúmeros sistemas, tais como sistemas flexíveis de manufatura, filas de serviços, sistemas de comunicação, de tráfego urbano, robótica, dentre outros, baseados em eventos discretos, como o acionamento e desligamento de um atuador, como um cilindro pneumático ou mesmo um motor elétrico, o início e término de operação de uma determinada máquina, a mudança de estado de um sensor ou ainda o apertar de um botão realizado por um usuário, surgiram técnicas de controle de sistemas a eventos discretos com regras lógicas, de causa e efeito, para eventos, onde seus sinais são números naturais representando estados lógicos (*on-off*, sim-não) (MORAES; CASTRUCCI, 2010).

Um Sistema a Eventos Discretos (SED) é definido como um sistema dinâmico no qual eventos ocorridos em intervalos de tempos irregulares ou desconhecidos determinam sua evolução (CASSANDRAS; LAFORTUNE, 2009). Para esta categoria de sistemas, Autômatos e Linguagens formais são comumente utilizados (HOPCROFT; MOTWANI; ULLMAN, 2006), embora existam inúmeros outros modelos, com destaque para as Redes de Petri, Cadeias de Markov, Álgebra Max-Plus e Teoria das Filas.

A Teoria de Controle Supervisório (TCS) (RAMADGE; WONHAM, 1989) fornece um método de controle formal no domínio de SEDs, baseadas em propriedades como controlabilidade, observabilidade, segurança, vivacidade e diagnosticabilidade. O objetivo da TCS é sintetizar um supervisor que desabilita a ocorrência de um conjunto de eventos, fazendo com que o sistema se comporte de maneira a obedecer determinadas especificações. A abordagem Modular Local (QUEIROZ; CURY, 2000) divide o sistema em plantas locais e sintetiza uma série de supervisores que atuam sobre cada planta, explorando a modularidade do sistema.

Diversas abordagens e técnicas de implementação do controle supervisório são

apresentadas e discutidas nas últimas décadas que lidam com problemas de implementação física, devido à limitação cíclica de varredura dos controladores industriais.

Os Controladores Lógico Programáveis (CLPs), desenvolvidos no início da década de 1970, constituem, ainda nos dias de hoje, a principal solução tecnológica adotada no controle dos sistemas produtivos automatizados (HELLGREN; FABIAN; LENNARTSON, 2005). Tradicionalmente, a etapa de desenvolvimento de lógicas de controle aplicadas na programação dos CLPs baseia-se na experiência do desenvolvedor (programador). Algumas razões são apontadas por Frey e Litz (2000) para a necessidade de utilização de métodos formais na programação em CLPs, bem como na verificação e validação dos programas desenvolvidos:

a. elevada complexidade dos problemas de controle;

b. demanda pela redução no tempo de desenvolvimento da programação;

c. possibilidade de reutilização dos programas;

d. demanda por soluções de alta qualidade, com destaque para aplicações onde a segurança é um aspecto crítico a ser considerado.

Pena, Cury e Lafortune (2009) destacam que, em geral, a qualidade do controle obtido por técnicas formais deverá ser superior à qualidade do controle baseado apenas na experiência do projetista.

Segundo Fabian e Hellgren (1998), na implementação do controle supervisório em CLPs deve-se fazer com que o CLP se comporte como uma máquina de estados, onde eventos e estados são variáveis booleanas internas no programa. No entanto, alguns problemas podem surgir, devido à natureza cíclica dos controladores. Diversos trabalhos, como Dietrich et al. (2002), Malik (2002), Cruz et al. (2009) e Leal, Cruz e Hounsell (2012), abordam estes problemas, dividindo-os em classes: causalidade, sinais e eventos, escolha e sincronização inexata.

Queiroz e Cury (2002) definem uma arquitetura de implementação estruturada em três níveis, ou blocos, denominados “Supervisores Modulares”, “Sistema Produto” e “Sequências Operacionais”. Esta arquitetura impõe a ação de controle estabelecida pelo conjunto de supervisores obtidos com a aplicação da TCS e atua como uma interface entre o modelo teórico empregado na síntese dos supervisores e o sistema a ser controlado.

Cruz et al. (2009), Leal, Cruz e Hounsell (2012), James, Teixeira e Leal (2019) apresentam metodologias de implementação em CLP que utilizam a linguagem *Ladder* e propõem o tratamento de diversos eventos em um mesmo ciclo de varredura do CLP, por meio de uma estrutura de 9 etapas, ordenadas conforme a controlabilidade dos eventos. Vieira et al. (2016) apresentam um método sistemático de implementação da TCS em

CLP com um programa estruturado em blocos de função (*Function Blocks*) com descrição em *Sequential Function Charts* (SFC).

Aliado a isso, diversos estudos no campo de otimização lidam com planejamento e *scheduling* de tarefas na produção, utilizando a TCS de modo a estabelecer um espaço de busca e desenvolver heurísticas que determinam sequências ótimas, sob determinados parâmetros. Se o tempo necessário para realizar um processo de produção numa fábrica é reduzido, as instalações podem ser usadas para expandir a produção e aumentar o lucro. Assim, como o tempo é um recurso valioso, escolher boas sequências operacionais, como aquelas que minimizam o *makespan*, é de grande importância na fabricação (ALVES; PENA; TAKAHASHI, 2021).

Nos dias atuais, com o surgimento da tecnologia em nuvem e de tecnologias aliadas, como *machine learning*, análise avançada associada à Inteligência Artificial, Comissionamento Virtual, Gêmeos Digitais, dentre outras, a indústria moderna alcançou uma nova camada da automação, comumente chamada de quarta onda da revolução industrial ou Indústria 4.0. Neste contexto, surge a necessidade de ir além e buscar novas soluções, como o desenvolvimento de modelos virtuais capazes de simular o comportamento dinâmico de plantas industriais, através de computação gráfica, a fim de realizar testes diversos, com segurança, sem apresentar riscos ao sistema real e ao operador.

Kritzinger et al. (2018) define e classifica a virtualização de sistemas de automação em três níveis:

a) Modelo Digital (*Digital Model*) é a representação digital de um objeto físico existente ou planejado. Não há troca de dados automática entre os modelos físico e virtual;

b) Sombra Digital (*Digital Shadow*) caracteriza um sistema de virtualização onde há troca de dados unidirecional entre planta física e virtual, isto é, uma mudança de estado no objeto físico atualiza o objeto digital;

c) Gêmeo Digital (*Digital Twin*) caracteriza-se pela integração completa entre modelos físico e virtual, realizando a troca de dados automática de maneira bidirecional entre os objetos.

Riera et al. (2010) destacam os benefícios da utilização de plantas virtuais como elemento complementar no treinamento em programação de CLPs, principalmente no quesito de segurança, onde um mínimo erro no desenvolvimento do programa, pode acarretar danos ao operador e aos equipamentos. Riera e Vigario (2017) apresentam duas dinâmicas que utilizam simulação 3D aplicadas ao ensino de automação industrial, realizando o comissionamento virtual de uma planta de transporte de esferas metálicas. Martin e Bohuslava (2018) traz o conceito de realidade aumentada, utilizando módulos

de cenas virtuais de uma fábrica de treinamento multidisciplinar implementados em navegadores *web*.

Este trabalho propõe a implementação do controle supervisório em CLP, tendo como estudo de caso um Sistema Flexível de Manufatura, modelado como um SED, sob o controle da TCS. A aplicação é desenvolvida em uma planta didática presente no laboratório LACSED/DELT/UFMG, e também em uma planta virtual, desenvolvida através de software de simulação 3D.

São apresentadas três implementações que utilizam metodologias conhecidas na literatura. A Implementação I utiliza a abordagem de Queiroz e Cury (2002) desenvolvida com aplicação no modelo virtual da planta, utilizando um CLP em simulação virtual. As duas últimas implementações utilizam a metodologia proposta por (LEAL; CRUZ; HOUNSELL, 2012). A Implementação II é aplicada ao protótipo físico da planta, que possui um CLP modelo Compact Logix L32E da Rockwell Automation. A Implementação III realiza uma aplicação híbrida, utilizando objetos físico e virtual sincronizados, tendo os eventos gerados no plano físico e reproduzidos, em tempo real, no plano virtual. Problemas típicos de implementação em CLPs são discutidos, assim como possíveis soluções para tratá-los.

Também é proposta uma alternativa à maneira em que os eventos controláveis são gerados, através da introdução de uma rotina intermediária na estrutura de implementação que recebe sequências de eventos geradas externamente, através, por exemplo, de algoritmos que lidam com planejamento e *scheduling* de tarefas na produção.

1.1 Objetivo Geral

O objetivo geral deste trabalho consiste em realizar o estudo, modelagem e implementação do controle supervisório em CLP tendo como estudo de caso o Sistema Flexível de Manufatura, explorando metodologias de implementação conhecidas da literatura. Realizar a aplicação em: um modelo virtual desenvolvido em software de simulação 3D; um protótipo didático montado no LACSED; um modelo híbrido que permita a sincronização entre planta física e virtual. Busca-se, também, viabilizar uma solução, como alternativa à geração espontânea de eventos controláveis pela estrutura de controle, que permita a execução de sequências de eventos geradas externamente.

1.2 Objetivos Específicos

- Realizar aprimoramentos no protótipo didático do LACSED;
- Desenvolver um modelo virtual da planta através de software de simulação 3D;

- Desenvolver a modelagem do comportamento assíncrono de cada máquina do sistema, das especificações de controle e realizar a síntese dos supervisores modulares locais reduzidos;
- Implementar o controle supervisório em CLP com aplicações nas plantas física e virtual;
- Obter os intervalos de tempo para realização das tarefas de cada máquina do protótipo da planta;
- Desenvolver uma rotina, dentro da estrutura implementada, que possibilite a execução de uma sequência de eventos controláveis obtida externamente.

1.3 Organização do Texto

O texto desta dissertação é estruturado em oito capítulos. O presente capítulo apresenta a introdução e os objetivos do trabalho. O Capítulo 2 é destinado à fundamentação teórica, com definições e formalismos, como Sistemas a Eventos Discretos, modelagem por Linguagens e Autômatos e a Teoria de Controle Supervisório. Também é apresentada uma breve revisão de literatura sobre a implementação do controle supervisório em CLP.

No Capítulo 3 apresenta-se o estudo de caso utilizado neste trabalho: o Sistema Flexível de Manufatura, onde são detalhadas suas características de funcionamento e as tarefas de operação do sistema. É apresentado um protótipo didático da planta, desenvolvido no LACSED, e também uma versão virtual, desenvolvida através de software de simulação 3D. Este capítulo também apresenta as modelagens por autômatos: do sistema em malha aberta, das especificações de controle e a síntese dos supervisores modulares locais reduzidos.

Os Capítulos 4, 5 e 6 são destinados à implementação do controle supervisório. No Capítulo 4 é apresentada uma implementação que utiliza a abordagem de Queiroz e Cury (2002), em CLP e planta virtuais, comunicando via OPC UA. No Capítulo 5 é implementado o controle supervisório utilizando a metodologia apresentada por Leal, Cruz e Hounsell (2012) em um CLP modelo Compact Logix L32E conectado ao protótipo físico da planta. No Capítulo 6 é desenvolvida uma implementação com aplicação híbrida, utilizando ambas as plantas, sincronizadas em tempo real.

No Capítulo 7 é apresentada uma solução como alternativa à geração espontânea de eventos controláveis do Sistema Produto, introduzindo uma rotina que recebe sequências de eventos geradas externamente.

Por fim, no Capítulo 8 são apresentadas as conclusões e contribuições deste trabalho, bem como perspectivas para desenvolvimentos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo é destinado à fundamentação teórica do trabalho, apresentando definições e formalismos sobre Sistemas a Eventos Discretos, Linguagens e Autômatos e a Teoria de Controle Supervisório. A Seção 2.4 apresenta uma revisão sobre a implementação do controle supervisório em CLP.

2.1 Sistemas a Eventos Discretos

Diferentemente dos sistemas contínuos clássicos, regidos por equações diferenciais, os sistemas a eventos discretos são sistemas cuja dinâmica é governada pela ocorrência de eventos que acontecem em intervalos de tempo irregulares ou desconhecidos (CASSANDRAS; LAFORTUNE, 2009). Por eventos, entende-se como uma ação específica no ambiente que está inserido, assíncrona no tempo, que pode afetar o comportamento desse sistema. Por exemplo, a abertura e o fechamento de uma válvula, o início e o término de operação de uma máquina, ou mesmo a mudança de estado de um sensor.

Diversos tipos de sistemas podem ser modelados como SEDs, como sistemas de comunicação, de transporte, fila de serviços, robótica, manufatura, dentre outros. Segundo Qamsane, Tajer e Philippot (2016), sistemas de manufatura automatizados são uma classe de sistemas a eventos discretos, usados para produzir produtos de qualidade com mais rapidez e eficiência ou para realizar serviços. Muler, Queiroz e Cury (2018) também destaca que a dinâmica dos problemas de sequenciamentos, intertravamentos e outras lógicas relativas à produção e segurança pode ser interpretada como sendo da classe de SEDs.

Um SED pode ser representado por diferentes modelos formais, com destaque para:

- Linguagens e Autômatos;
- Redes de Petri;
- Cadeias de Markov;
- Álgebra Max-Plus;
- Teoria das Filas.

2.2 Linguagens e Autômatos

Um alfabeto Σ é definido como um conjunto finito e não vazio de eventos. Uma sequência finita de eventos de um alfabeto é chamada de cadeia. O comportamento de um SED é modelado como cadeias sobre Σ.

O comprimento de uma cadeia é definido por $|\sigma_1\sigma_2...\sigma_i| = i$, onde $i > 0$. O comprimento da cadeia vazia é $|\epsilon| = 0$. O conjunto formado por todas as cadeias possíveis, com símbolos de um mesmo alfabeto, é chamado de Fechamento Kleene Σ^*, incluindo a cadeia vazia ϵ. A concatenação de duas cadeias é uma operação definida como uma cadeia seguida de outra cadeia, representada por su, onde $s, u, su \in \Sigma^*$. Uma cadeia s é chamada prefixo de $t \in \Sigma^*$, $s < t$, se $\exists u \in \Sigma^*$ tal que $su = t$.

Uma linguagem L é um subconjunto de cadeias de comprimento finito, composto por eventos do alfabeto Σ, $L \subseteq \Sigma^*$. O prefixo fechamento $\overline{L}$ de uma linguagem $L \subseteq \Sigma^*$ é o conjunto de todos os prefixos de cadeias em L, expresso por $\overline{L} = \{s \in \Sigma^* | s \leq t$ para algum $t \in L\}$.

Autômatos são uma forma gráfica de representar linguagens, representam-se através de grafos orientados cujos vértices (ou nós), são os estados e as arestas (ou arcos), as transições. Um autômato finito determinístico G é definido como uma quíntupla:

$$G = (Q, \Sigma, \delta, q_0, Q_m)$$

sendo:

- Q, o conjunto finito de estados;
- Σ, o conjunto finito de eventos (alfabeto);
- $\delta : Q \times \Sigma \to Q$, a função de transição de estados parcial;
- $q_0 \in Q$, o estado inicial;
- $Q_m \subseteq Q$, o conjunto de estados marcados.

Um autômato é determinístico se $\delta(q, \sigma) = q_1$ e $\delta(q, \sigma) = q_2$ implica em $q_1 = q_2$.

Como exemplo, a Figura 1 apresenta um autômato finito determinístico, onde tem-se: $Q = \{s0, s1\}$, $\Sigma = \{\alpha, \beta\}$, $\delta(s0, \alpha) = s1$, $\delta(s1, \beta) = s0$, $q_0 = \{s0\}$ e $Q_m = \{s0\}$.

O estado inicial é representado por uma seta que não possui origem em nenhum outro estado e aponta para si. Já os estados marcados são identificados por círculos duplos, que normalmente são estados que representam uma tarefa concluída de parte do sistema.

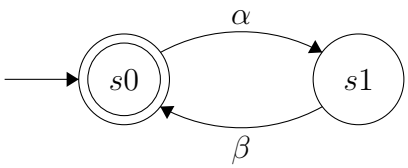


Figura 1 – Autômato de estados finitos

A função de transição pode ser estendida para cadeias na forma $\delta(q, \sigma_s) = q'$ se $\delta(q, \sigma) = x$ e $\delta(x, s) = q'$, em que $s \in \Sigma^*$ e Σ. A função de eventos ativos $\Gamma : Q \to 2^{\Sigma}$ é dada pelo conjunto de eventos $\sigma \in \Sigma$, em um dado estado q, onde $\delta(q, \sigma)$ é definido.

A linguagem gerada por um autômato é dada por $\mathcal{L}(G) = \{s \in \Sigma^* | \delta(q_0, s)$ é definido$\}$ e representa o conjunto de cadeias de eventos que, partindo do estado inicial, levam a algum estado do autômato G. A linguagem gerada é prefixo fechada por definição. A linguagem marcada por um autômato é dada por $\mathcal{L}_m(G) = \{s \in \mathcal{L}(G) | \delta(q_0, s) \in Q_m\}$ e representa o conjunto de cadeias de eventos que, do estado inicial, levam a algum estado marcado $q \in Q_m$ de G. A linguagem marcada não precisa ser prefixo fechada, uma vez que nem todos os estados são marcados. Assim, um SED pode ser modelado por um autômato G, onde $\mathcal{L}(G)$ é o comportamento gerado, e $\mathcal{L}_m(G)$ é o comportamento marcado, ou o conjunto de tarefas completas do sistema (CURY, 2001).

Um estado $q \in Q$ de um autômato é dito acessível se $s \in \Sigma^* | \delta(q_0, s) = q$, isto é, se existe uma cadeia de eventos que, partindo do estado inicial, leve ao referido estado. Um autômato é acessível se todo estado $q \in Q$ é acessível. Um estado $q \in Q$ é coacessível se $\exists s \in \Sigma^* | \delta(q, s) \in Q_m$, ou seja, caso exista uma cadeia de eventos que, partindo do estado q, leve a um estado marcado. Um autômato, portanto, é coacessível se todo estado $q \in Q$ é coacessível. Um autômato acessível que não atende o critério de coacessibilidade é dito bloqueante, pois existem estados, a partir dos quais, não é mais possível alcançar um estado marcado. Assim, um autômato é não bloqueante, se e somente se $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$. O $trim(G)$ é uma operação que consiste em eliminar todos os estados não acessíveis ou não coacessíveis de um autômato G, resultando em um autômato aparado.

A composição síncrona é uma operação destinada a representar o comportamento conjunto de dois ou mais autômatos. É uma operação comumente utilizada quando dois ou mais subsistemas de um SED apresentam eventos em comum em seus alfabetos, ou seja, a ocorrência desses eventos é simultânea nos subsistemas que os compartilham. Dados dois autômatos $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$, a composição síncrona é dada por:

$$G_1 || G_2 = G_{12} = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta_{12}, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$$

onde:

$$\delta_{12} = \delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), & \text{se } \sigma \in \Gamma_1(q_1) \cap \Gamma_2(q_2) \\ (\delta_1(q_1, \sigma), q_2), & \text{se } \sigma \in \Gamma_1(q_1) - \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)), & \text{se } \sigma \in \Gamma_2(q_2) - \Sigma_1 \\ \text{indefinido}, & \text{caso contrário} \end{cases}$$

E a função de eventos ativos é $\Gamma_{1||2}(q_1, q_2) = [\Gamma_1(q_1) \cap \Gamma_2(q_2)] \cup [\Gamma_1(q_1) - \Sigma_2] \cup [\Gamma_2(q_2) - \Sigma_1]$.

A composição síncrona (ou paralela) é o resultado da evolução dos autômatos G_1 e G_2 em paralelo, isto é, eventos exclusivos de cada autômato podem ser executados quando habilitados, e eventos compartilhados entre os dois autômatos somente podem ser executados quando habilitados simultaneamente por ambos. A Figura 2 ilustra a operação de composição síncrona entre dois autômatos.

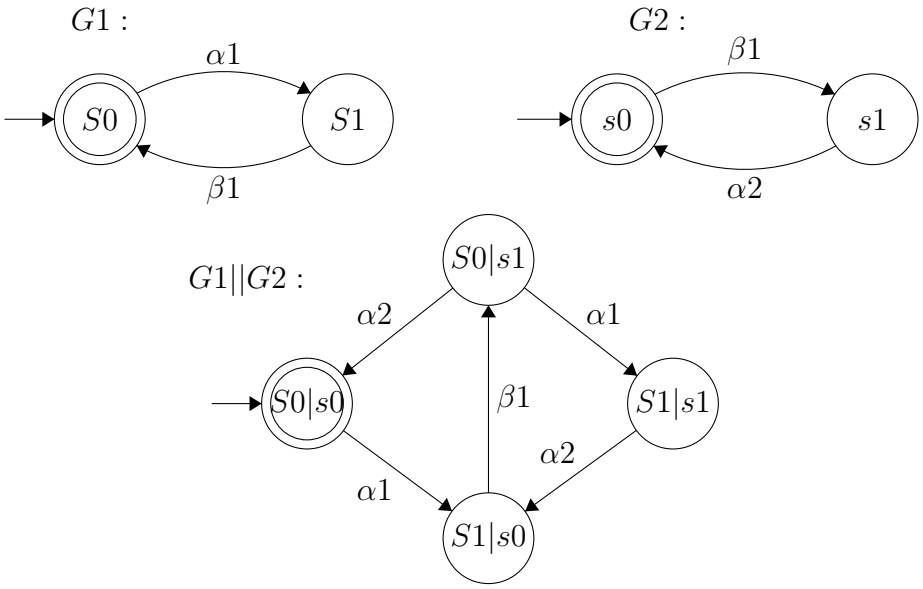


Figura 2 – Composição síncrona entre autômatos

2.3 Teoria de Controle Supervisório

A Teoria de Controle Supervisório (RAMADGE; WONHAM, 1989) fornece um método de controle formal no domínio de SEDs. De maneira geral, o objetivo da TCS é sintetizar um supervisor não bloqueante e minimamente restritivo, que desabilita a ocorrência de um conjunto de eventos, fazendo com que o sistema se comporte de maneira a obedecer determinadas especificações.

A planta é o conjunto de subsistemas (máquinas) modelados por autômatos que representam seu comportamento em malha aberta. Os eventos da planta são classificados

como controláveis e não controláveis, por meio do particionamento $\Sigma = \Sigma_c \cup \Sigma_{uc}$, onde Σ_c é o conjunto de eventos controláveis e Σ_{uc} o conjunto de eventos não controláveis. Os eventos controláveis estão associados aos comandos que geram sinais de saída provenientes do controlador, por exemplo, o acionamento do motor de uma esteira, o início de um processo ou tarefa. Já os eventos não controláveis estão associados às respostas provenientes da planta através de sinais de entrada, por exemplo, o fim de curso de uma esteira, sinalizada por um sensor. Essa divisão de eventos é importantíssima, pois o supervisor jamais poderá desabilitar um evento não controlável.

As especificações de controle estabelecem restrições no comportamento do sistema, isto é, condições de segurança fisicamente inadmissíveis, como o bloqueio de uma máquina ou sistema, a sobreposição (*overflow*) ou falta (*underflow*) de peças em um local ou máquina, por exemplo.

O comportamento desejado em malha fechada do sistema, ou seja, o comportamento da planta G sob as restrições de controle E, é definido por $K = G||E$, de forma que $\mathcal{L}_m(K) = \mathcal{L}_m(G||E) \subseteq \mathcal{L}_m(G)$. $K \subseteq \mathcal{L}(G)$ é uma sublinguagem controlável em relação a $\mathcal{L}(G) \subseteq \Sigma^*$, se

$$\overline{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$$

Esse critério estabelece que nenhum evento não controlável seja desabilitado, isto é, para qualquer cadeia $s \in \overline{K}$ concatenada com um evento $\sigma \in \Sigma_{uc}$ que pertença a $\mathcal{L}(G)$, $s\sigma$ deve ser um prefixo de $K, s\sigma \in \overline{K}$. Se K não atender esse critério, existe uma máxima sublinguagem controlável $SupC(K, G) \subseteq K$, ou seja, é possível projetar um supervisor $S = SupC(K, G)$, não bloqueante e minimamente restritivo que modela todo o comportamento em malha fechada da planta.

No controle supervisório Monolítico clássico, um único supervisor é projetado para coordenar o comportamento de toda a planta. Nesta abordagem, todos os subsistemas assíncronos que compõem a planta são sincronizados em um único modelo que representa o comportamento global G da planta. De forma análoga, todas as especificações de controle são compostas em uma especificação global E.

A Figura 3 apresenta diagramas de blocos que ilustram a forma como os subsistemas e especificações são relacionados na síntese monolítica: (a) retângulo em torno das especificações indicando a composição para obter somente uma planta e uma especificação de controle; (b) representação do supervisor monolítico que observa os eventos da planta e define desabilitações.

É importante observar que nessa abordagem, a medida que o sistema expande, o número de estados cresce exponencialmente, o que gera um grande esforço computacional na síntese do supervisor monolítico.

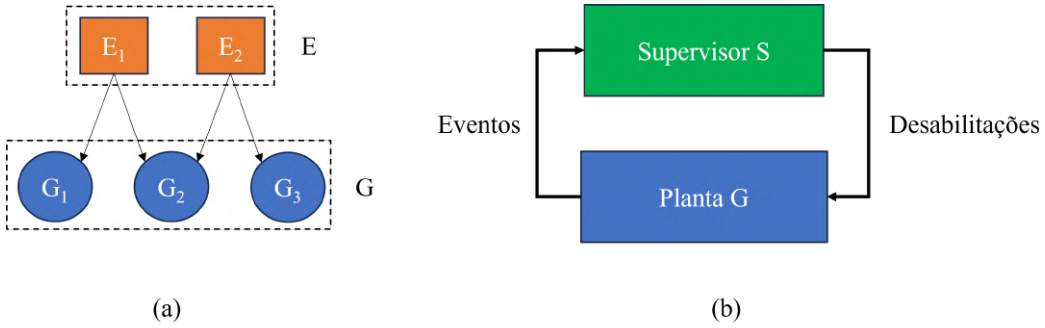


Figura 3 – Estrutura do controle monolítico

A abordagem Modular Local (QUEIROZ; CURY, 2000), compreende uma extensão da TCS clássica, onde é computado um supervisor para cada especificação, explorando uma modularidade do sistema. Cada supervisor engloba uma especificação e os subsistemas aos quais a especificação está relacionada, apontando os eventos que são compartilhados. O supervisor, assim como o monolítico, observa e desabilita eventos controláveis pertinentes.

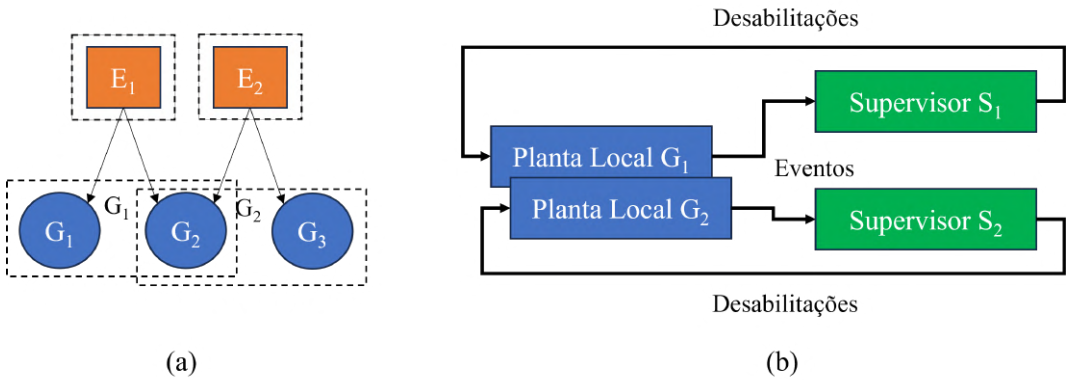


Figura 4 – Estrutura do controle modular local

A Figura 4 apresenta diagramas de blocos que ilustram a forma como os subsistemas e especificações são relacionados na síntese modular local: (a) retângulo em torno das especificações indicando que é obtido um supervisor por especificação com uma visão local da planta (representado pelo retângulo sobre um subconjunto de subsistemas); (b) representação do conjunto de supervisores que observam eventos das plantas locais e definem desabilitações.

Contudo, essa abordagem pode introduzir um problema de conflito entre supervisores, uma vez que, havendo mais de um supervisor, existe a possibilidade de um interferir na ação do outro. Portanto, torna-se necessária a execução de um teste de não conflito entre supervisores (CASSANDRAS; LAFORTUNE, 2009). No teste de não conflito (ou modularidade) realiza-se a composição síncrona de todos os supervisores locais e verifica-se se o gerador resultante é *trim*, isto é, se não apresenta estados não acessíveis e não co-acessíveis. Caso esta propriedade não seja verificada, significa que a ação conjunta dos

supervisores locais levará o sistema a um estado de bloqueio. Wong et al. (2000) abordam a resolução de conflito entre supervisores.

Assim, a síntese modular permite que sistemas complexos possam ser decompostos em módulos mais simples, podendo reduzir a complexidade e o custo computacional.

2.4 Revisão Sobre a Implementação do Controle Supervisório em CLP

O controle supervisório, na sua teoria, considera que todos os eventos da planta são assíncronos, discretos no tempo, e gerados pelo próprio sistema. Porém, na prática, os controladores, através de seus circuitos eletrônicos, possuem uma limitação, de natureza cíclica, para processamento do código implementado. Além disso, a geração de eventos controláveis é realizada pelo próprio controlador, promovendo o acionamento de atuadores. Essa diferença entre teoria e prática, devido à natureza de varredura cíclica dos controladores, em especial, e mais comumente presente na indústria, dos CLPs, pode acarretar alguns problemas típicos de implementação prática.

Nesta seção, estes problemas são apresentados e discutidos, revisando algumas metodologias de implementação da literatura com ênfase em CLPs. Também é apresentada uma breve definição sobre os CLPs, bem como suas características e recursos.

2.4.1 Controladores Lógicos Programáveis

Historicamente, a necessidade de sistemas de controle mais flexíveis, reconfiguráveis e fáceis de programar impulsionou o desenvolvimento de controladores que pudessem substituir os sistemas de controle convencionais, baseados em relés. Desenvolvidos no início da década de 1970, os controladores lógicos programáveis surgiram para suprir essa demanda. Projetados para receber informações de sensores e dispositivos de entrada, processar essas informações de acordo com um programa de controle pré-definido e acionar dispositivos de saída, como motores, válvulas, relés, entre outros, os CLPs são particularmente úteis em ambientes industriais devido à sua capacidade de lidar com ambientes adversos, com vibração, temperatura extrema e interferências eletromagnéticas. Também são altamente flexíveis e podem ser reprogramados para diferentes tarefas de controle sem a necessidade de alterações físicas no hardware. Desde então, os CLPs evoluíram consideravelmente em termos de capacidades de processamento, recursos de comunicação, robustez e flexibilidade de programação.

Em um ciclo de execução (ou varredura), o CLP realiza as seguintes etapas básicas:

1. atualização das entradas;
2. processamento das instruções do programa;
3. atualização das saídas.

O CLP lê a porta de entrada, gravando a informação dos estados na memória-imagem de entrada. Em seguida ocorre o processamento, e, por fim, a memória-imagem de saída é copiada na porta de saída, atualizando-a (MORAES; CASTRUCCI, 2010). Entender esse procedimento de operação do CLP é crucial para a etapa de construção da arquitetura lógica de implementação do controle supervisório que será apresentada nos capítulos seguintes.

A norma internacional IEC 61131-3 estabelece uma padronização dos princípios de programação para CLPs, desde a especificação dos tipos de dados, declaração e representação de variáveis, passando pelo modelo de software até as linguagens de programação (VIEIRA; CURY; QUEIROZ, 2006). A Tabela 1 apresenta a classificação das linguagens de programação conforme IEC 61131-3.

Tabela 1 – Classificação das linguagens de programação, conforme IEC 61131-3

Classe	Linguagem
Textual	IL (Instruction List)
	ST (Structured Text)
Gráfica	LD (Ladder)
	FBD (Function Block Diagram)
	SFC (Sequential Flow Chart)

Fonte: Adaptado de Moraes e Castrucci (2010).

A linguagem *Ladder* (LD) é uma linguagem gráfica clássica, muito difundida no meio industrial, em forma de diagrama, que remete à lógica de relés, através da energização de bobinas por meio de contatos. A Tabela 2 apresenta as funções básicas da linguagem LD abordadas pela IEC 61131-3.

Tabela 2 – Funções básicas do LD (IEC 61131-3)

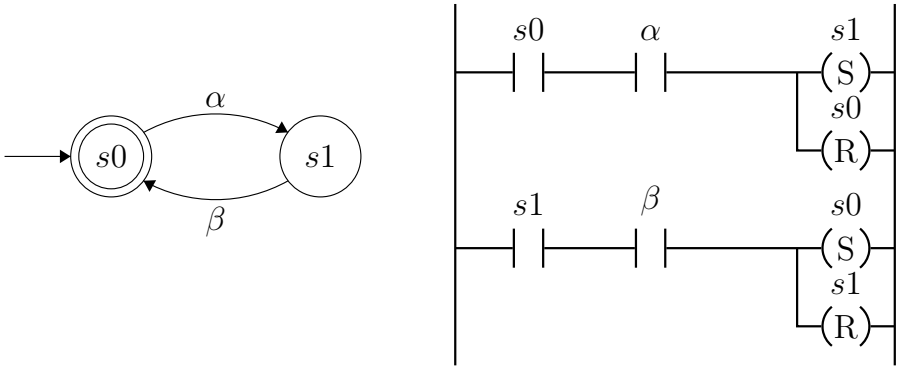
Símbolo	Nome	Descrição
-| |-	Contato NA	Contato normalmente aberto
-|/|-	Contato NF	Contato normalmente fechado
-( )	Bobina	A bobina é ligada quando sua linha é energizada, e desligada quando desenergizada
-(S)	Bobina SET	A bobina é ligada quando sua linha é energizada, e permanece ligada até que sua bobina RESET seja ativada
-(R)	Bobina RESET	A bobina é desligada quando sua linha é energizada

2.4.2 Implementação da TCS em CLP

Segundo Fabian e Hellgren (1998), na implementação da estrutura de controle supervisório em CLPs, deve-se fazer com que este se comporte como uma máquina de estados. Consiste, portanto, em representar eventos e estados de um autômato através

de variáveis booleanas internas, ou seja, bits de memórias do tipo 'BOOL' (0 ou 1), onde a memória correspondente ao estado atual assume o valor 1 e as memórias dos demais estados assumem o valor 0.

Na linguagem LD, as transições são modeladas por meio da lógica 'E' (*AND*) dos contatos referentes ao estado atual e ao evento, energizando, quando verdadeiros, as bobinas 'SET' e 'RESET', correspondentes ao próximo estado e ao estado atual, respectivamente. A Figura 5a representa um autômato com dois estados (s0 e s1) e dois eventos (α e β), na representação (b), é apresentado como esse autômato é implementado na linguagem LD.



(a) (b)

Figura 5 – Implementação de um autômato em LD

Entretanto, esta técnica de implementação não é trivial, e pode levar a alguns problemas, abordados por Fabian e Hellgren (1998), Dietrich et al. (2002), Malik (2002) e Leal, Cruz e Hounsell (2012) resumidamente apresentados nas seções seguintes.

2.4.2.1 Causalidade

Segundo Fabian e Hellgren (1998), causalidade está relacionada com a questão: "Quem gera o que?", ou seja, quem é responsável pela geração dos eventos: a planta ou o CLP? Isto ocorre devido ao fato de que, originalmente, a TCS (RAMADGE; WONHAM, 1989) assume que todos os eventos, controláveis e não controláveis, são gerados espontaneamente pela planta e que os supervisores têm o papel de desabilitar eventos controláveis. Entretanto, em aplicações práticas, os eventos controláveis são comandos provenientes do CLP (MALIK, 2002), isto é, a critério da estrutura de código implementada, o controlador gerará um evento controlável, se habilitado pelos supervisores, por meio de uma variável booleana que envia um comando através de um sinal de saída para a planta. Já os eventos não controláveis são gerados naturalmente pela planta por meio de sinais de entrada como respostas para o controlador (Figura 6).

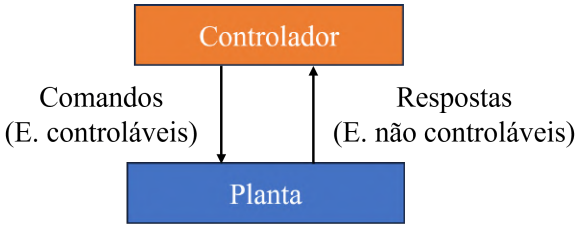


Figura 6 – Geração de eventos em aplicações práticas

2.4.2.2 Sinais e Eventos

A TCS considera que a evolução da planta é dirigida pela ocorrência de eventos discretos, de natureza instantânea e assíncrona. No entanto, a planta real e o CLP têm sinais que estão sempre presentes nas entradas e saídas, uma vez que a geração de eventos provenientes da planta se dá a partir da alteração destes sinais, e o processo de identificação destas mudanças é realizado pelo CLP, de forma cíclica, a planta pode gerar mais de um evento durante um ciclo de varredura e o sistema pode interpretar que os eventos ocorreram simultaneamente (VIEIRA; CURY; QUEIROZ, 2006).

Dependendo de como implementado, o registro da mudança do estado desse sinal, feito pelo CLP, pode provocar uma transição indevida de mais de um estado do supervisor num mesmo ciclo de varredura, este efeito é conhecido como *efeito avalanche*. Este efeito resulta em uma quebra de sincronização entre planta e supervisor, invalidando as próximas ações de controle.

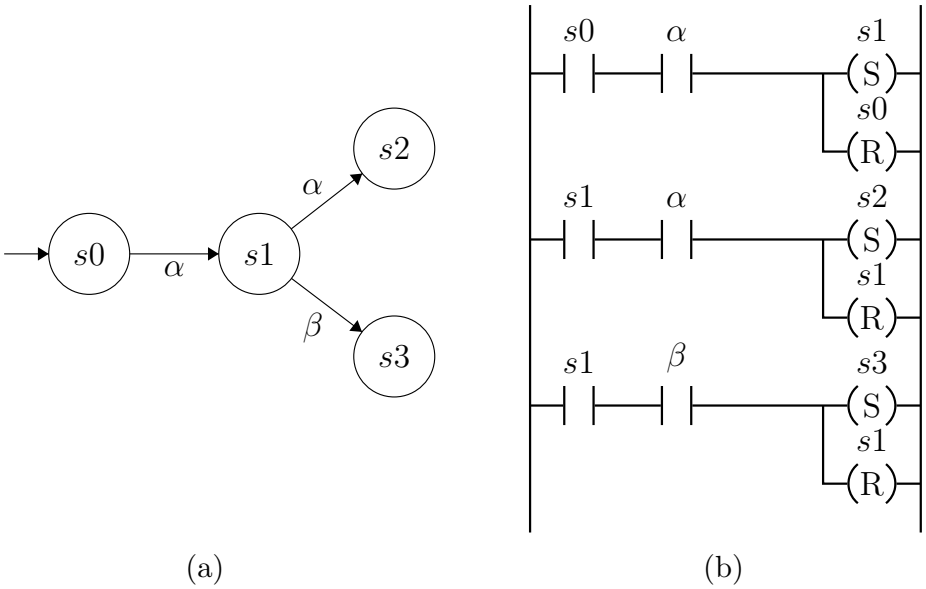


Figura 7 – Exemplo do efeito avalanche

A Figura 7 apresenta um exemplo de implementação que pode gerar o efeito avalanche: quando o autômato em (a) é implementado através do código (b), a ocorrência

do sinal correspondente ao evento α, registrado pelo CLP, causará duas transições, fazendo o sistema evoluir do estado 's0' para o estado 's2' no mesmo ciclo de varredura.

Além disso, dois ou mais eventos podem ocorrer de maneira não simultânea, mas serem registrados em um mesmo ciclo de varredura, quando isso ocorre, é impossível de o CLP determinar a ordem dos eventos registrados em um mesmo ciclo. Esse problema é conhecido como *incapacidade de reconhecer a ordem dos eventos* ou, também, como *simultaneidade*. Para garantir que essa limitação não interfira no sistema, é necessário que este possua uma propriedade definida por Fabian e Hellgren (1998) como *insensibilidade ao entrelaçamento*, onde um supervisor é insensível ao entrelaçamento se sua ação de controle não depende dos possíveis entrelaçamentos de eventos não controláveis que podem ocorrer após qualquer sequência s (VIEIRA; CURY; QUEIROZ, 2006).

2.4.2.3 Sincronização Inexata

Em aplicações práticas, a comunicação entre a planta física e o sistema de controle onde está implementado o supervisor, é sujeita a atrasos significativos. Quando uma mudança no sinal de entrada do CLP venha a invalidar uma ação de controle (geração de evento controlável), e essa alteração de sinal é registrada somente no próximo ciclo, o CLP pode vir a gerar essa ação de controle indevidamente, devido ao atraso. Fabian e Hellgren (1998) definem esse problema como *sincronização inexata*. Para garantir que este problema não ocorra, deve-se garantir que a linguagem gerada pelo autômato que modela o supervisor tenha a propriedade de ser *insensível ao atraso* (BALEMI; BRUNNER, 1992).

2.4.2.4 Escolha de Eventos

O problema da escolha está relacionado ao fato de que um sistema pode, em um determinado estado, ter dois ou mais eventos habilitados que gera múltiplas opções na evolução da planta. Dessa forma, sendo o SP responsável pela geração de uma parcela de eventos, a decisão sobre qual o próximo evento a ser gerado é compartilhada entre planta e sistema de controle (CRUZ et al., 2009).

A proposta de Fabian e Hellgren (1998) consiste em selecionar apenas um evento entre as opções existentes, porém, Malik (2002) alerta que, a priorização de um evento em detrimento de outro pode fazer com que uma parte do comportamento do sistema nunca seja alcançada, podendo até levar ao bloqueio desse sistema.

2.4.3 Metodologias de Implementação da TCS em CLP

Algumas abordagens e metodologias de implementação da TCS em CLP são resumidamente apresentadas nesta seção.

Fabian e Hellgren (1998) abordam a implementação da TCS em CLP, definindo premissas de descrição dos autômatos em LD, mas sem definir uma metodologia clara para o projeto.

Vieira, Cury e Queiroz (2006) apresentam uma metodologia de implementação, com um código estruturado em blocos de funções (*Function Blocks*), sendo uma solução de projeto de implementação mais completa, incluindo modos de operação automático, manual e falhas. Utiliza a linguagem *Sequential Funcion Chart* (SFC).

Vieira et al. (2016) apresentam um método sistemático de implementação da TCS em CLP com um detalhamento para obtenção de cada componente do programa, estruturado em *program organization units* (POUs), e suas variáveis. O código é estruturado em FBs com descrição em SFC.

As abordagens de Queiroz e Cury (2002) e Leal, Cruz e Hounsell (2012), utilizam a linguagem LD, e são apresentadas com destaque nas próximas seções, como principais referências deste trabalho.

2.4.3.1 Queiroz e Cury (2002)

Queiroz e Cury (2002) apresentam uma arquitetura de implementação com uma estrutura denominada Sistema Produto (SP), nesta arquitetura (Fig. 8), além dos autômatos dos supervisores, também são implementados os autômatos que correspondem aos subsistemas assíncronos da planta, e esta etapa intermediária é responsável por gerar os eventos controláveis quando habilitados pelos supervisores, solucionando o problema da causalidade.

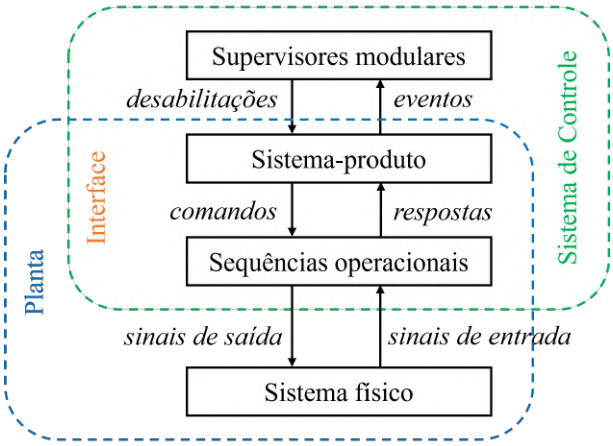


Figura 8 – Arquitetura proposta por Queiroz e Cury (2002)

Tomemos como exemplo os autômatos da Figura 9: (a) apresenta um autômato que representa o comportamento assíncrono de uma máquina genérica G, no estado '0' a

máquina encontra-se ociosa, no estado '1' a máquina está em processamento. O evento '*a1*' é um evento controlável que indica o início de funcionamento da máquina, transicionando o autômato do estado '0' para o estado '1'. Já o evento '*b1*' é um evento não controlável que indica o fim de processamento desta máquina, transicionando o autômato do estado '1' para o estado '0'. Os traços nos arcos de um autômato são comumente utilizados para identificar os eventos controláveis. Em (b) é apresentado o autômato de um supervisor hipotético S que modela um comportamento desejado desta planta. O tracejado em vermelho nos estados indicam os eventos controláveis desabilitados pelo supervisor. Considera-se que '*a2*' é um evento controlável que pertence à outra planta local que, para fins de exemplo, não foi citada.

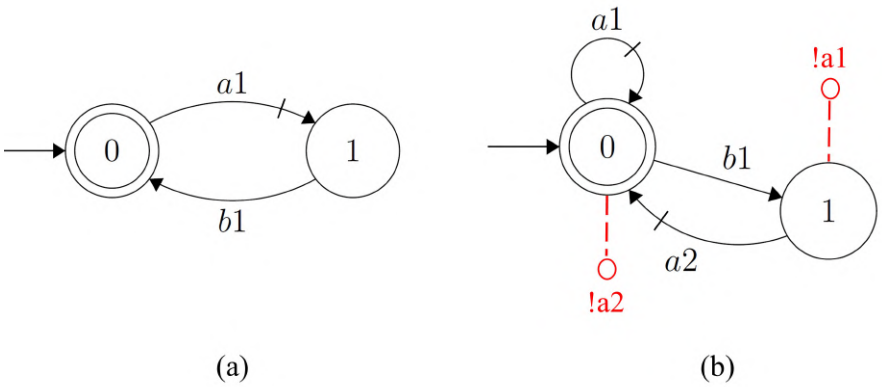


Figura 9 – Autômatos exemplificando: (a) o comportamento assíncrono de uma máquina genérica; (b) supervisor hipotético que implementa um comportamento desejado

Segundo a abordagem de Queiroz e Cury (2002), implementa-se a estrutura do Sistema Produto em LD, descrevendo primeiro o degrau (*rung*) com a transição que envolve o evento não controlável, e depois o degrau com a transição do evento controlável. Observa-se, no exemplo da Figura 10, que a variável booleana '*so_fim*' que indica o final da tarefa executada pela planta, aciona a bobina '*b1*', sinalizando a ocorrência do evento não controlável e, ao mesmo tempo, transiciona do estado '*g1*' para o estado '*g0*'. No degrau seguinte, a bobina '*a1*', que corresponde ao evento controlável, será acionada espontaneamente se habilitada pela variável '*d_a1*', simultaneamente, a bobina '*so_ini*' inicializará a sequência operacional que será executada pela planta e o autômato transicionará do estado '*g0*' para o estado '*g1*'. Ao fim de cada degrau, é utilizado um salto para o label *sup* do degrau de atualização dos supervisores, que é descrito na etapa seguinte.

Após a implementação do SP, implementam-se os supervisores modulares, seguindo a descrição em LD apresentada na Seção 2.4.2, e logo após, as desabilitações. A Figura 11 apresenta a implementação do supervisor S da Figura 9b, tomado como exemplo. Observa-se que o primeiro degrau possui o label *sup* para onde o código salta cada vez que um evento é gerado na etapa anterior. Na etapa das desabilitações, as bobinas '*d_a2*'

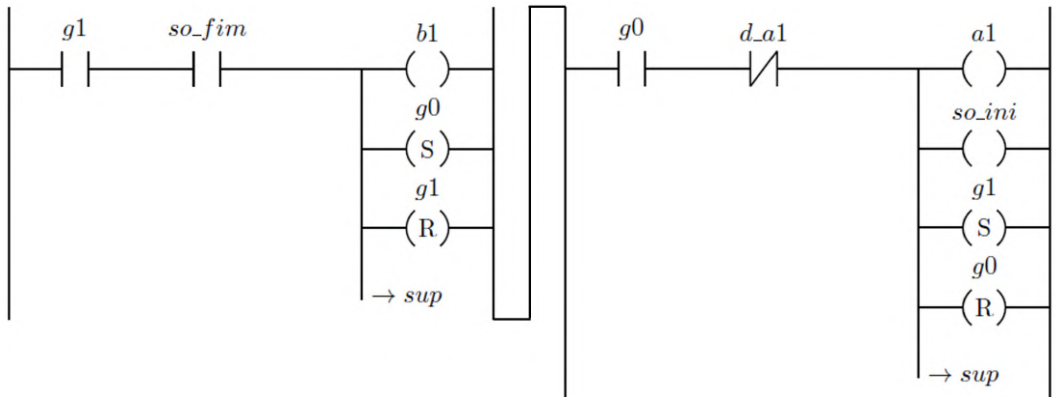


Figura 10 – Implementação genérica do Sistema Produto em LD segundo (QUEIROZ; CURY, 2002)

e '*d_a1*' desabilitam os respectivos eventos controláveis, por meio do contato NF presente no degrau de implementação do SP, dependendo do estado que o supervisor se encontra.

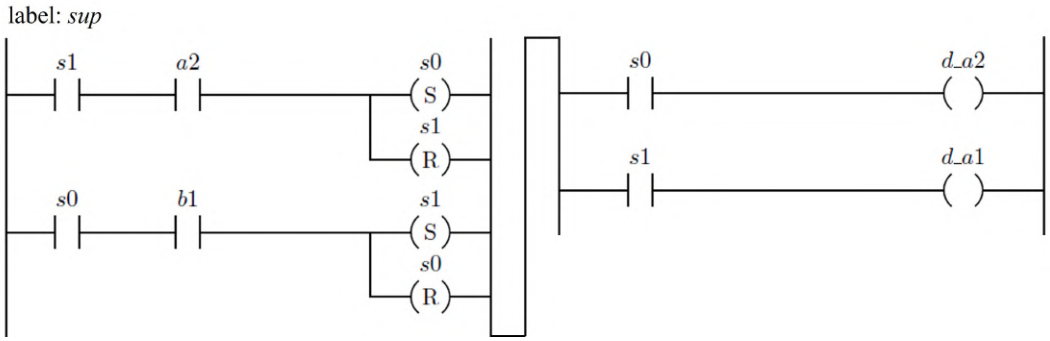


Figura 11 – Implementação genérica do supervisor em LD segundo (QUEIROZ; CURY, 2002)

Por fim, são implementadas as sequências operacionais que geram os sinais de saída do CLP para a planta, no exemplo, inicializada a partir da variável booleana '*so_ini*' gerada pelo SP, e finalizada pela variável '*so_fim*', sinalizando o término da tarefa. Dessa maneira, os eventos controláveis são gerados pelo SP, os eventos não controláveis são naturalmente gerados por sinais da planta e os supervisores mantêm a função original de desabilitar eventos controláveis. Esta abordagem é adotada na Implementação I do Capítulo 4 deste trabalho, onde cada etapa é detalhadamente apresentada.

A estrutura de Queiroz e Cury (2002) é implementada em um código com uma única rotina, apresenta uma solução para o efeito avalanche utilizando saltos no código que garantem que apenas um evento seja tratado a cada ciclo de varredura, porém isso limita o desempenho do CLP, pois utiliza-se pelo menos um ciclo para atualizar os supervisores, e outro ciclo para atualizar os sinais para a planta, tornando a atualização do supervisor mais lenta, pois quando há n mudanças nos sinais de entrada do CLP durante um ciclo de *scan*, o programa levará n ciclos para tratar a ocorrência destes eventos e atuar na

planta. Isto pode agravar o problema de sincronização inexata, apontado por Balemi e Brunner (1992), a medida que o sistema expande. Os autores não exploram soluções para o problema da escolha.

2.4.3.2 Leal, Cruz e Hounsell (2012)

Leal, Cruz e Hounsell (2012) apresentam uma metodologia denominada "Definição da Implementação Conforme a Controlabilidade do Evento". De maneira geral, a arquitetura de Queiroz e Cury (2002) é mantida, dividindo o sistema de controle em três níveis: Sistema Produto, supervisores modulares e sequências operacionais. Contudo, a ordem desses níveis é subdividida em etapas classificadas pela controlabilidade dos eventos. De acordo com a proposta, escreve-se o programa respeitando a ordem "eventos não controláveis - desabilitações - eventos controláveis". A Figura 12 ilustra as etapas da rotina principal ordenadas pela controlabilidade dos eventos, em grupos: A, B e C.

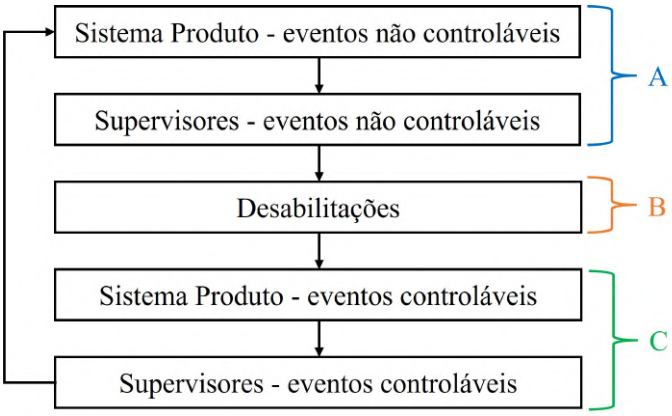


Figura 12 – Etapas da rotina principal ordenadas pela controlabilidade dos eventos (LEAL; CRUZ; HOUNSELL, 2012)

Para isso, é feito um desmembramento do Sistema Produto e dos supervisores em degraus que transicionam de estados pela ocorrência de eventos não controláveis, e degraus que transicionam de estados pela ocorrência de eventos controláveis. Ou seja, as transições dos autômatos são implementadas em pontos diferentes ao longo do código do CLP, dependendo apenas se a transição envolve um evento controlável ou não controlável (CRUZ et al., 2009). A Figura 13 ilustra a implementação em LD do exemplo tomado na seção anterior, obedecendo as etapas ordenadas conforme a metodologia de Leal, Cruz e Hounsell (2012).

Dessa maneira, consegue-se tratar todos os eventos, gerados pela planta e pelo Sistema Produto, em um único ciclo de *scan* do CLP. Contudo, implementando-se mais de uma planta e mais de um supervisor nesta estrutura, pode surgir o problema do efeito avalanche, exemplificado na Seção 2.4.2.2. Como solução, os autores introduzem uma

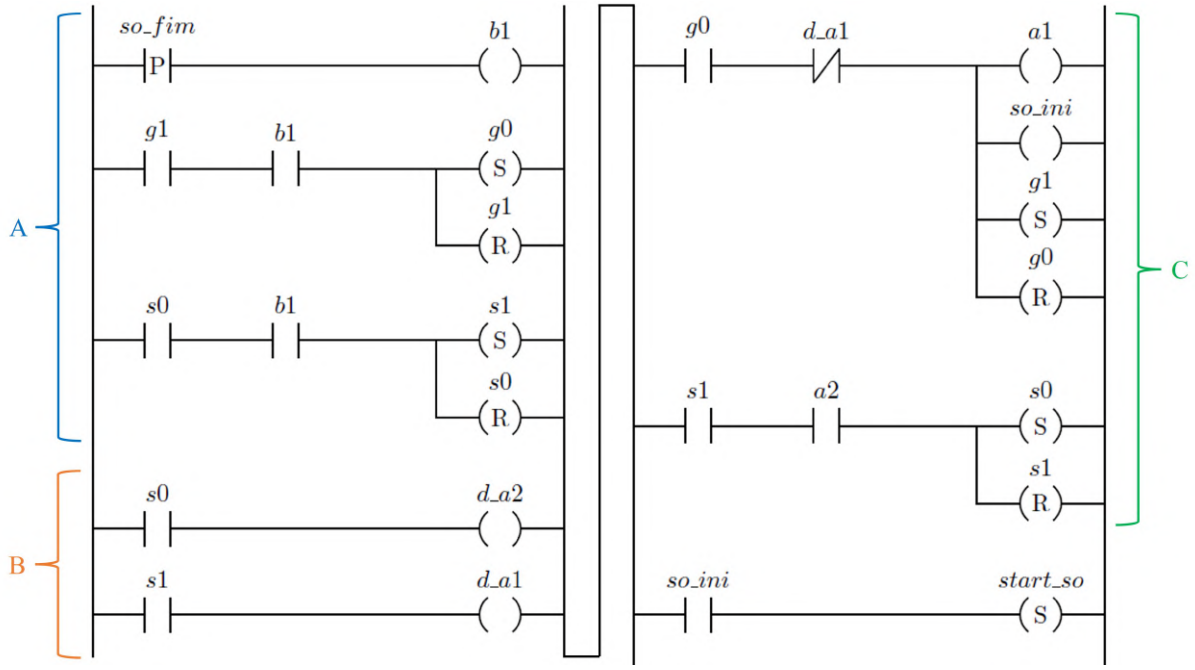


Figura 13 – Implementação em LD das etapas da rotina principal ordenadas pela controlabilidade dos eventos (LEAL; CRUZ; HOUNSELL, 2012)

rotina que faz uma espécie de "represamento" no código, onde, inicialmente, a informação da transição de um sinal de entrada é armazenada em uma variável booleana auxiliar, e, a cada degrau de atualização do SP e dos supervisores, essa informação é apagada. Entretanto, para não perder a informação original das entradas, utiliza-se uma rotina intermediária auxiliar, denominada "Resgate dos eventos não controláveis", que consiste em um espelhamento da variável booleana que guardou a transição inicial em uma segunda variável onde, esta sim, será apagada a cada degrau de atualização. Os autores também apresentam uma solução para o problema de escolha, que utiliza uma variável booleana interna que alterna a habilitação de um dos eventos de escolha a cada ciclo de varredura do CLP, promovendo uma aleatoriedade na escolha. A solução para o efeito avalanche e para o problema da escolha são detalhadas no Capítulo 5.

A metodologia de Leal, Cruz e Hounsell (2012) é adotada nas Implementações II e III apresentadas nos Capítulos 5 e 6 deste trabalho, incluindo o detalhamento de cada etapa do método.

No próximo capítulo, detalha-se o sistema flexível de manufatura utilizado como estudo de caso, a modelagem do comportamento de cada subsistema, das especificações e obtenção dos supervisores. É apresentado o delineamento do modelo virtual da planta e o *retrofit* do protótipo do LACSED.

3 ESTUDO DE CASO: O SISTEMA FLEXÍVEL DE MANUFATURA

Este capítulo apresenta, na Seção 3.1, o sistema utilizado como estudo de caso deste trabalho e suas características de operação. A Seção 3.2 apresenta a planta didática do sistema que utiliza CLP, sensores e atuadores reais, e como ela foi adaptada para funcionamento em laboratório. Também é apresentada, na Seção 3.3, sua versão virtual, como foi modelada e desenvolvida, além de recursos e softwares utilizados para as implementações apresentadas nos capítulos seguintes.

A Seção 3.4 é destinada à modelagem do comportamento do sistema por autômatos e apresenta os passos realizados para a síntese dos supervisores modulares locais do controle supervisório.

3.1 Detalhamento do Sistema

O sistema utilizado como estudo de caso neste trabalho, denominado Sistema Flexível de Manufatura (SFM) (Figura 14), é um modelo proposto por Queiroz, Cury e Wonham (2005) que produz dois tipos de produtos diferentes e possui máquinas que podem realizar tarefas simultaneamente, caracterizando-se como um modelo interessante para aplicação da TCS.

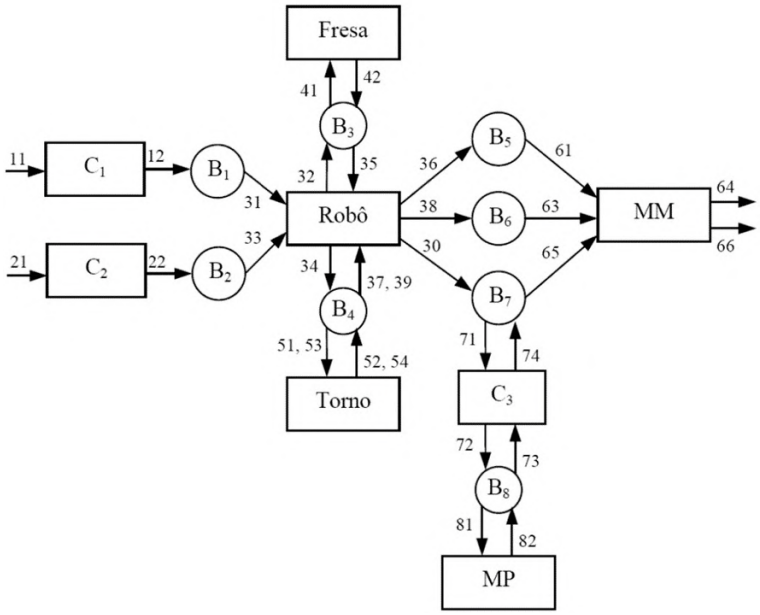


Figura 14 – Sistema Flexível de Manufatura (SFM). Adaptado de (QUEIROZ; CURY; WONHAM, 2005)

O sistema é alimentado por blocos brutos para confecção de bases e dois tipos de

pinos, cônicos e cilíndricos. Os produtos finais, produto A e produto B, são resultantes da composição de um pino montado sobre uma base, sendo um bloco de pino cônico (Produto A) e um bloco de pino cilíndrico pintado (Produto B). O SFM possui oito máquinas:

- C_1: esteira de entrada que transporta blocos brutos para confecção de bases;
- C_2: esteira de entrada que transporta blocos brutos para confecção de pinos;
- Robô: responsável por distribuir peças entre *buffers*;
- Fresa: realiza a confecção de bases;
- Torno: realiza a confecção de pinos cônicos e cilíndricos;
- C_3: esteira que transporta pinos cilíndricos para pintura;
- MP: máquina de pintura de pinos cilíndricos;
- MM: máquina de montagem de pinos sobre bases e saída dos produtos manufaturados.

Entre as máquinas há locais para armazenamento das peças, denominados *buffers* ($B_1, B_2, ..., B_8$). Estes buffers são unitários, isto é, só é possível armazenar uma peça por vez. Na Figura 14, os retângulos representam cada máquina, os círculos os buffers, a direção das setas indica o fluxo das peças no sistema e os rótulos destas setas são os eventos, sendo os ímpares eventos controláveis e pares eventos não controláveis. Um protótipo desse sistema foi desenvolvido no LACSED e é apresentado na Seção 3.2.

3.2 Planta Didática

O protótipo didático da planta montada no LACSED (Figura 15) foi construído originalmente utilizando peças do kit de robótica Fischertechnik que possui diversos sensores e atuadores para o controle da planta. O CLP utilizado é um Compact Logix L32E da Rockwell Automation montado em um kit didático desenvolvido no LVAS/DELT, denominado “MICA 05”, com tensão típica de E/S em 24V, uma interface de E/S adaptada para a planta, contendo 27 relés para acionamento das saídas com tensão típica de 9V e 6 transistores para chaveamento de entradas digitais em 24V. A Tabela 3 apresenta os dispositivos de E/S utilizados na construção do protótipo.

Na construção do protótipo, em geral, os buffers são uma parte das próprias máquinas, não separados como sugere o diagrama de blocos. Como por exemplo, os buffers B_1 e B_2 são o fim do curso das esteiras C_1 e C_2, respectivamente, B_3 e B_4 são as posições que o robô coloca e retira as peças na fresa e no torno.

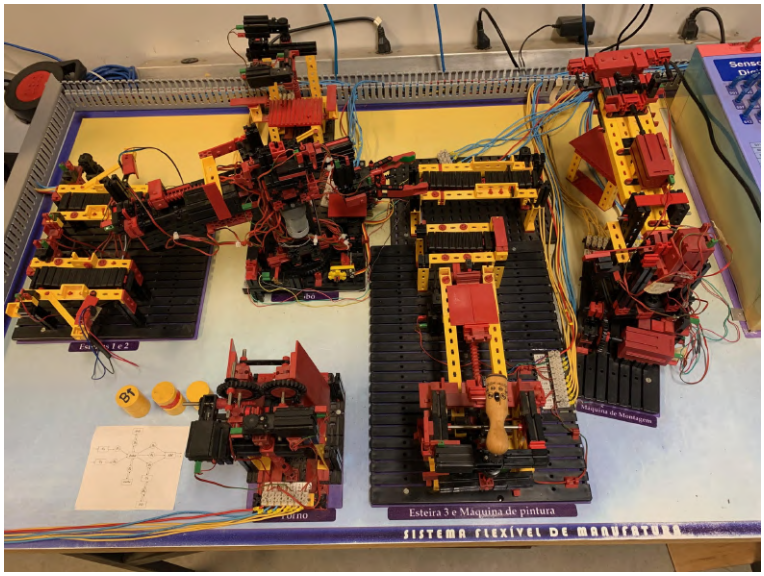


Figura 15 – Planta didática LACSED

Tabela 3 – Lista de dispositivos

Quantidade	Dispositivo	Tipo
17	Motores	DC
14	Chaves fim de curso	Microswitch
3	Contadores de giro	Microswitch
2	Sensores de posição	Fotoelétrico
2	Sensores de posição	Óptico reflexivo
1	Encoder	Óptico de quadratura

3.2.1 Encoder e Sensores

Na adaptação do protótipo para o desenvolvimento deste trabalho, houve a necessidade de substituição da tecnologia do contador de giro do robô, que utilizava microswitchs mecânicos, para um encoder óptico de quadratura com dois canais modelo E2-Q2 (Figura 16a). Este tipo de encoder realiza a leitura da reflexão de luz infravermelha emitida pelo sensor de refletância, que é retornada quando um dos 6 dentes da engrenagem do kit está sobre o componente.

(a) (b)

Figura 16 – (a) Encoder óptico de quadratura E2-Q2; (b) Sensores ópticos reflexivos.

Utilizando esse tipo de encoder, é possível aumentar a resolução do contador de pulsos e, portanto, obter maior precisão no posicionamento do eixo de giro do robô. Para isso, o intervalo de requisição de pacote de dados - RPI do canal de entrada digital do CLP, foi ajustado para 1 milissegundo, possibilitando a leitura de todos os pulsos em um giro completo do eixo do robô. Para o funcionamento do encoder na planta, foi desenvolvido um circuito de interface para alimentação em 5V para o encoder, utilizando regulador de tensão LM7805, e também um circuito de chaveamento para a entrada digital do CLP em 24V utilizando optoacoplador.

Para a detecção de peças nos buffers B_1 e B_2, foram utilizados sensores ópticos reflexivos (Figura 16b), para melhor precisão do posicionamento de chegada das peças, substituindo os sensores fotoelétricos anteriores.

O protótipo didático do SFM apresentado nesta seção é utilizado nas implementações II e III, apresentadas nos Capítulos 5 e 6.

3.3 Planta Virtual

O modelo virtual do SFM desenvolvido neste trabalho foi realizado através do software Factory IO[1]. Desenvolvido pela Real Games, o Factory IO é uma nova geração de simulação 3D da indústria, onde é possível construir uma indústria virtual através de um conjunto de elementos presentes na indústria real (RIERA; VIGARIO, 2017).

O software, que foi utilizado para virtualização da planta e simulação gráfica, possui uma vasta biblioteca de elementos de campo da indústria, como sensores diversos, atuadores, esteiras, máquinas de transporte e processamento, dentre outros. Permite utilização com tecnologias de comunicação modernas, como OPC DA, OPC UA e Modbus, comunicação direta com CLPs de diferentes fabricantes via protocolo TCP/IP, criação de cenários personalizados com foco em treinamento em programação de CLPs. É importante destacar que trata-se de um software pago, possui diferentes versões para aquisição e uma licença de teste gratuita por 30 dias com todos os recursos disponíveis.

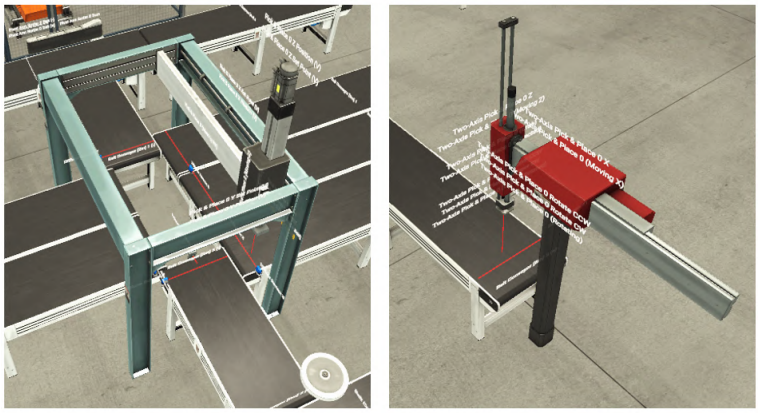
Outros softwares, como Gazebo, CoppeliaSim, FlexSim, simuladores de robótica, dentre outros, também são utilizados para virtualização e simulação na automação, contudo, o Factory IO foi selecionado devido à compatibilidade com os sistemas já existentes no laboratório, além de outros benefícios, como a fácil configuração, ambiente amigável e integração com softwares de diversos fabricantes.

[1] Factory IO - Disponível em: https://factoryio.com/. Em: 18 mai. 2024

3.3.1 Delineamento do SFM 3D

Com o objetivo de recriar o SFM no ambiente virtual, buscou-se ser o mais fiel ao sistema original quanto possível, no entanto, algumas abstrações foram necessárias devido às limitações de elementos específicos.

O robô central que distribui os blocos entre buffers, foi implementado através de um *Pick & Place 3D* (Figura 17a), com a função de distribuir as peças entre buffers. Os buffers, por sua vez, são modelados como esteiras, para evitar choque entre máquinas. A máquina de pintura é abstraída em um *Pick & Place 2D* (Figura 17b) que simula o processamento da peça nesta etapa.



(a) (b)

Figura 17 – (a) Pick & Place 3D equivalente ao robô e à máquina de montagem; (b) Pick & Place 2D equivalente à máquina de pintura

Além disso, *machine centers* (Figura 18) são utilizadas para confeccionar bases (fresa) e pinos, cônicos e cilíndricos (torno), representados por produtos (*lids*) verdes e azuis, respectivamente. Por fim, a máquina de montagem, também modelada por um Pick & Place 3D, faz a combinação das peças processadas (base e pino) e coloca nos buffers de saída do sistema. Esteiras rolantes unidirecionais e bidirecionais são utilizadas para o transporte de peças entre máquinas.

O resultado final deu origem a dois modelos da mesma planta, o primeiro, denominado Modelo I (Figura 19), é destinado à Implementação I apresentada no Capítulo 4, sendo responsável pela geração dos eventos não controláveis para o CLP virtual, e também receber os sinais dos eventos controláveis gerados pelo controlador e realizar tarefas através de seus atuadores. Para isso, o Modelo I utiliza sensores ópticos reflexivos para detectar a presença das peças em cada buffer.

O Modelo II é destinado à Implementação III e é apresentado no Capítulo 6, este já não possui sensores tampouco gera qualquer evento não controlável como resposta para

Figura 18 – Machine center equivalente à fresa e ao torno

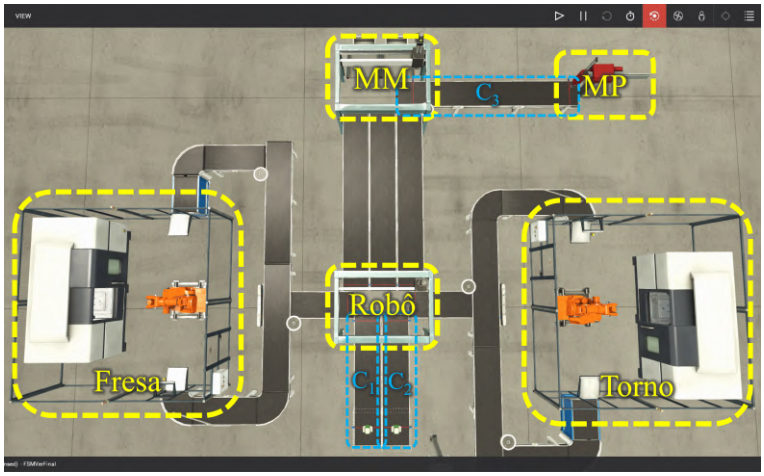


Figura 19 – Resultado final do Modelo I do SFM virtual

o CLP, sua função é a de representar, em tempo real, as operações das etapas da planta física, que gera os sinais que são recebidos via rede de comunicação pelo CLP real.

3.4 Controle Supervisório

Esta seção apresenta a modelagem dos comportamentos de cada máquina, das especificações de controle e a síntese dos supervisores modulares locais, conforme a descrição das tarefas de operação do SFM. Essa modelagem é descrita por Queiroz, Cury e Wonham (2005) e abordada em trabalhos como Rezende et al. (2012), Pena et al. (2016), Malik e Pena (2018) e Alves e Pena (2020).

3.4.1 Modelagem das Máquinas

A Tabela 4 apresenta a lista de todos os eventos do SFM, classificados por máquina, e as descrições de suas operações. O primeiro dígito de cada evento indica a qual máquina

Tabela 4 – Eventos de cada máquina do SFM.

Máquina	Evento	Descrição
C_1	11	Início de operação da esteira C_1
	12	Fim de operação da esteira C_1
C_2	21	Início de operação da esteira C_2
	22	Fim de operação da esteira C_2
Robô	31	Início de operação do robô ⎤ pega peça em B1 e deposita em B3
	32	Fim de operação do robô ⎦
	33	Início de operação do robô ⎤ pega peça em B2 e deposita em B4
	34	Fim de operação do robô ⎦
	35	Início de operação do robô ⎤ pega peça em B3 e deposita em B5
	36	Fim de operação do robô ⎦
	37	Início de operação do robô ⎤ pega peça em B4 e deposita em B6
	38	Fim de operação do robô ⎦
	39	Início de operação do robô ⎤ pega peça em B4 e deposita em B7
	30	Fim de operação do robô ⎦
Fresa	41	Início de operação da fresa
	42	Fim de operação da fresa
Torno	51	Início de operação do torno ⎤ confecção de pino cônico
	52	Fim de operação do torno ⎦
	53	Início de operação do torno ⎤ confecção de pino cilíndrico
	54	Fim de operação do torno ⎦
MM	61	Início de operação da MM pega base em B5
	63	Início de operação da MM ⎤ monta Produto A
	64	Fim de operação da MM ⎦
	65	Início de operação da MM ⎤ monta Produto B
	66	Fim de operação da MM ⎦
C_3	71	Início de operação da esteira C_3 ⎤ transporte de B7 até B8
	72	Fim de operação da esteira C_3 ⎦
	73	Início de operação da esteira C_3 ⎤ transporte de B8 até B7
	74	Fim de operação da esteira C_3 ⎦
MP	81	Início de operação da MP
	82	Fim de operação da MP

está associado e o segundo dígito indica a controlabilidade, sendo ímpar para eventos controláveis e par para eventos não controláveis. Observa-se que o evento 61 não possui um evento não controlável associado, sendo necessária a ocorrência de 64 ou 66 para finalizar a operação da máquina de montagem.

Os autômatos assíncronos que modelam o comportamento em malha aberta de cada máquina (ou subsistema) $G_i, i \in \{1, 2, ..., 8\}$ são apresentados na Figura 20.

3.4.2 Modelagem das Especificações

Para estabelecer o controle desejado para operação do SFM, busca-se desenvolver uma lógica que, basicamente, não permita sobreposição (*overflow*) nem falta (*underflow*) de peças nos buffers. Dessa forma, as especificações genéricas $E_j, j \in \{1, 2, ..., 8\}$, modelam o comportamento em relação à entrada e saída de peças nos buffers, de modo a obedecer estes critérios (Figura 21).

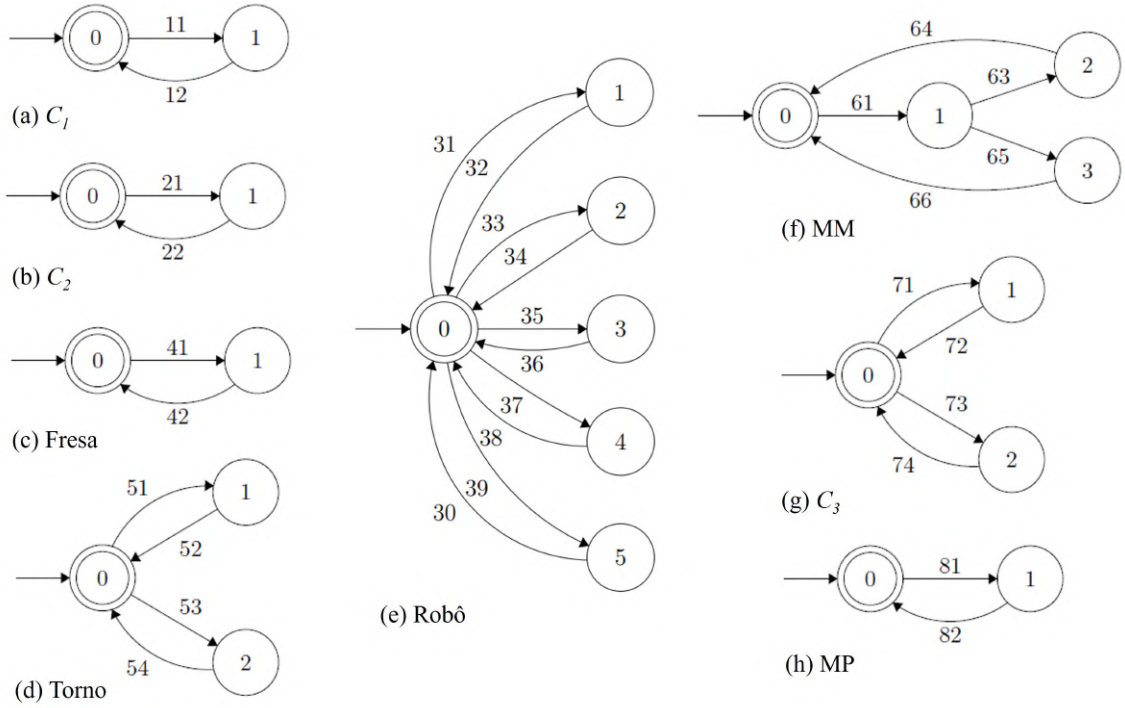


Figura 20 – Autômatos que modelam o comportamento de cada máquina

3.4.3 Síntese dos Supervisores

Modelados os comportamentos de cada subsistema e das especificações de controle, para a obtenção dos supervisores modulares locais, foram realizados os seguintes passos, conforme apresentado na Seção 2.3:

1. Composição síncrona dos subsistemas que possuem eventos em comum com a especificação em questão, obtendo as plantas locais;

$$G_1 = C_1 \;||\; \text{Robô}$$
$$G_2 = C_2 \;||\; \text{Robô}$$
$$G_3 = \text{Fresa} \;||\; \text{Robô}$$
$$G_4 = \text{Robô} \;||\; \text{Torno}$$
$$G_5 = G_6 = \text{Robô} \;||\; \text{MM}$$
$$G_7 = \text{Robô} \;||\; \text{MM} \;||\; C_3$$
$$G_8 = \text{MP} \;||\; C_3$$

2. Obtenção do comportamento desejado em malha fechada K_j, realizando a composição síncrona das especificações com as plantas locais;

$$K_j = E_j||\mathcal{L}_m(G_j), \forall j = 1, ..., 8$$

3. Obtenção dos supervisores modulares locais S_j que implementam as máximas sublinguagens controláveis em relação a G_j contidas em K_j;

$$S_j = SupC(K_j, \mathcal{L}(G_j)), \forall j = 1, ..., 8$$

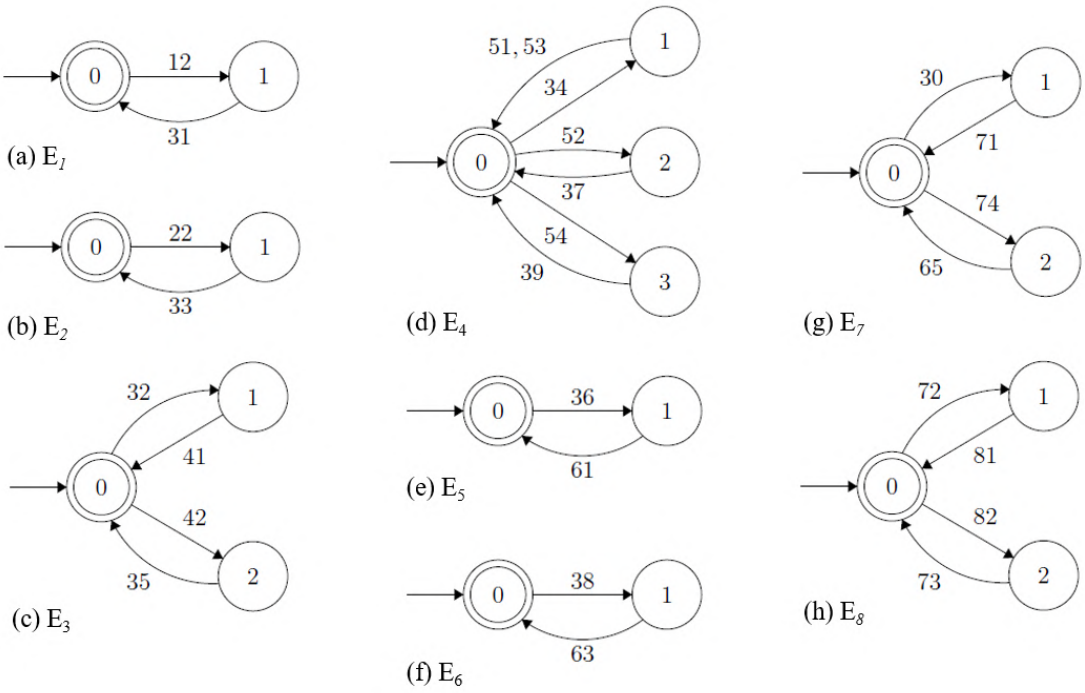


Figura 21 – Autômatos que modelam as especificações de controle

4. Realização do teste de não conflito: o resultado inicial foi conflitante entre supervisores modulares locais, mais especificamente entre os supervisores S_7 e S_8. Pena, Cury e Lafortune (2009) apresentam uma abordagem que torna a solução não bloqueante, realizando a composição síncrona entre as especificações E_7 e E_8 e das plantas locais G_7 e G_8, resultando em E_{78} e G_{78}, respectivamente. Assim, é obtido um novo comportamento desejado em malha fechada K_{78} e o supervisor S_{78}. Após realização de um novo teste de não conflito, o resultado foi não conflitante entre os sete supervisores.
5. Por fim, são obtidos os supervisores reduzidos. A redução de supervisores (SU; WONHAM, 2004), é uma técnica que visa a diminuição do número de estados do supervisor pela eliminação da redundância entre modelo e supervisor.

As etapas enumeradas acima foram implementadas computacionalmente através da biblioteca UltraDES[2], e os sete supervisores modulares locais reduzidos obtidos $S_{xred}, x \in \{1, 2, ..., 7\}$ são apresentados na Figura 22, as desabilitações em cada estado são representadas pelos traços em vermelho.

É importante destacar que no autômato do supervisor S_{4red}, observa-se que no estado '1' dois eventos controláveis estão habilitados: 51 e 53, levando à ocorrência dos eventos não controláveis associados 52 e 54, respectivamente. Isto é, existem dois eventos possíveis que levarão o sistema a diferentes estados, o que caracteriza um problema de

[2] UltraDES - Disponível em: https://lacsed.github.io/UltraDES/. Em: 18 mai. 2024.

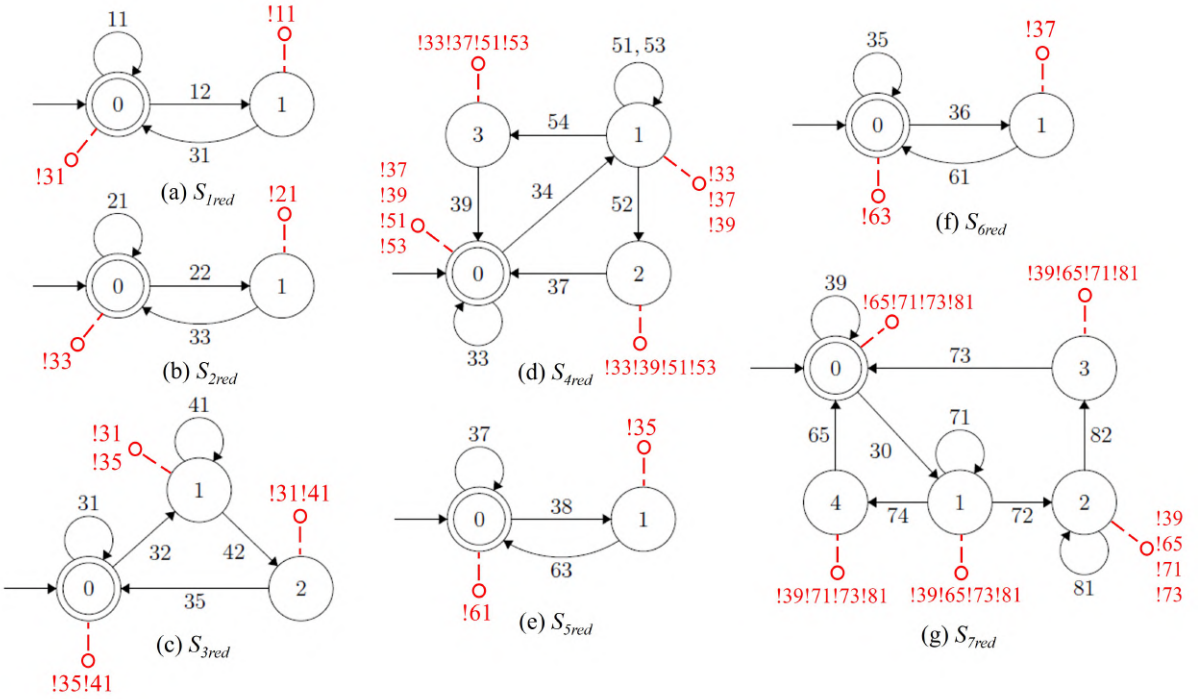


Figura 22 – Autômatos dos supervisores modulares locais reduzidos

escolha. Na prática, isto é traduzido na operação do torno, onde o evento 51 produz um pino cônico, e o evento 53 produz um pino cilíndrico.

Como não há uma política que determine qual tipo de produto ou qual quantidade de um lote de produtos serão produzidos, é necessário estabelecer um critério para tratar essa escolha. Diferentes soluções para lidar com o problema de escolha são discutidas e apresentadas nos próximos capítulos.

Nos Capítulos 4, 5 e 6 são realizadas três implementações do controle supervisório em CLP, explorando diferentes metodologias e aplicações.

4 IMPLEMENTAÇÃO I

Neste capítulo é realizada a implementação do controle supervisório em CLP virtual através do software CODESYS, como elemento de controle, e o Modelo I da planta virtual desenvolvida no Factory IO, que fornece os sinais para o sistema de controle e recebe os comandos para os atuadores. A abordagem de Queiroz e Cury (2002) é adotada nesta implementação. A apresentação desse modelo com a implementação adotada foi publicada em Freitas e Pena (2023).

4.1 Metodologia de Implementação

A arquitetura de controle supervisório proposta por Queiroz e Cury (2002) é utilizada nesta implementação, considerando um sistema de controle com um único CLP, utilizado em modo virtual através do CODESYS. A estrutura básica desta implementação com a hierarquia de três níveis, adaptada para o SFM, é ilustrada pela Figura 23.

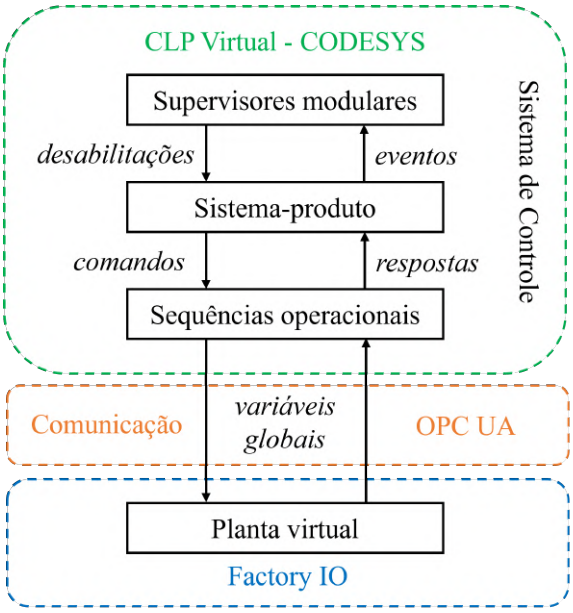


Figura 23 – Arquitetura da Implementação I

A implementação dos autômatos em LD foi realizada como descrito na Seção 2.4.3.1. Sendo assim, os eventos controláveis são gerados pelo Sistema Produto, os eventos não controláveis são naturalmente gerados por sinais da planta e os supervisores desabilitam eventos controláveis conforme o comportamento desejado. É importante destacar que quando realiza-se a redução dos supervisores, a informação dos eventos fisicamente possíveis de serem executados pela planta é suprimida. Implementando-se o Sistema Produto, isto é, o comportamento assíncrono de cada subplanta G_i, essa informação é preservada.

A Figura 24 apresenta o fluxograma com a organização do programa em etapas, obedecendo a hierarquia sugerida por Queiroz e Cury (2002). O código foi estruturado em uma rotina única.

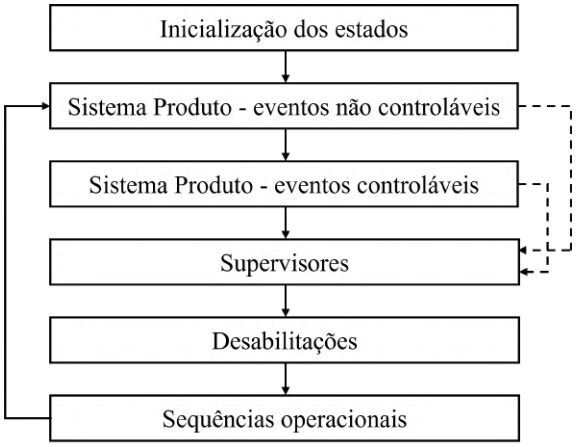
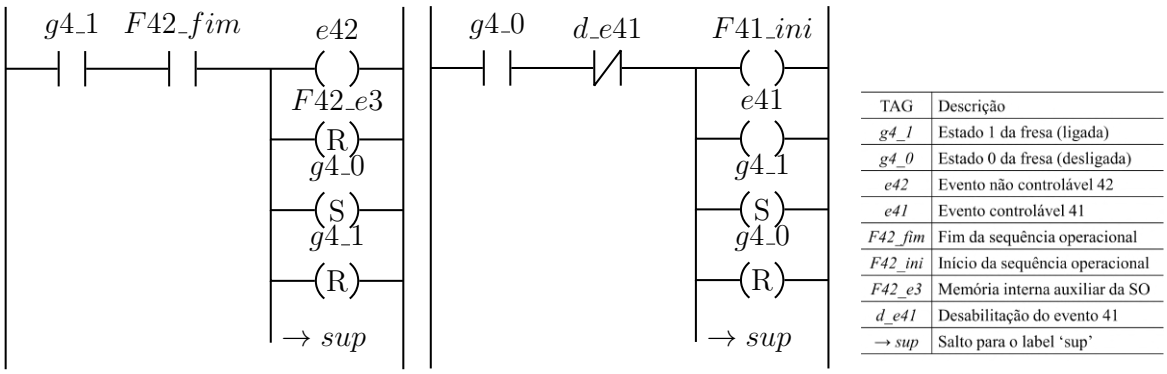


Figura 24 – Fluxograma com as etapas de implementação do código

A rotina de inicialização dos estados é executada uma única vez, no primeiro ciclo de *scan* do CLP, e tem a função de atribuir às variáveis booleanas, referentes aos estados iniciais das plantas e dos supervisores, nível lógico '1'.

No nível do SP, as subplantas $G_i, i \in \{1, 2, ..., 8\}$ são programadas como máquinas de estado assíncronas, onde os eventos não controláveis são disparados pelos sinais provenientes das sequências operacionais que sinalizam o fim de processamento da máquina (Figura 25a).



TAG	Descrição
g4_1	Estado 1 da fresa (ligada)
g4_0	Estado 0 da fresa (desligada)
e42	Evento não controlável 42
e41	Evento controlável 41
F42_fim	Fim da sequência operacional
F42_ini	Início da sequência operacional
F42_e3	Memória interna auxiliar da SO
d_e41	Desabilitação do evento 41
→ sup	Salto para o label 'sup'

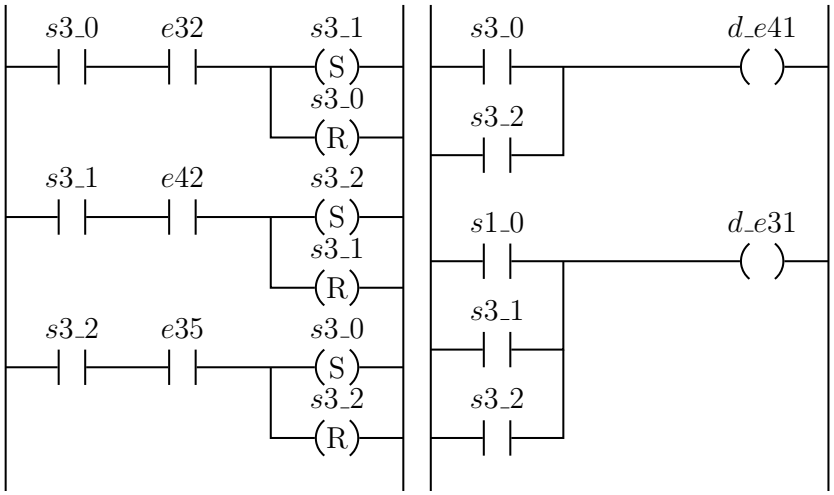
(a) (b)

Figura 25 – Descrição em LD do autômato G_4 que modela o comportamento em malha aberta da fresa e legenda das TAGs

Os eventos controláveis, por sua vez, são executados automaticamente caso não estejam desabilitados pelos supervisores, ativando as variáveis que iniciam os blocos com as sequências operacionais, e sinalizando para os supervisores a ocorrência do respectivo evento (Figura 25b).

Para garantir que não ocorram duas transições seguidas no SP sem que os supervisores tenham sido atualizados, isto é, o efeito avalanche, é utilizado um salto no final de cada transição para o primeiro degrau dos supervisores, sinalizado pelo *label* 'sup'. Esses saltos são indicados pelas setas tracejadas do fluxograma da Figura 24.

Na implementação dos supervisores modulares reduzidos $S_{xred}, x \in \{1, 2, ..., 7\}$, cada respectivo autômato é implementado como máquinas de estado concorrentes, da maneira descrita na Seção 2.4.3.1. A Figura 26a apresenta o trecho em LD que implementa o supervisor S_{3red} (Figura 22c). Na etapa de desabilitações, são agregadas as variáveis booleanas dos supervisores em paralelo que desabilitam um mesmo evento controlável, ativando a bobina que corresponde à desabilitação deste evento (Figura 26b).



TAG	Descrição
s3_0	Estado 0 do supervisor S_{3red}
s3_1	Estado 1 do supervisor S_{3red}
s3_2	Estado 2 do supervisor S_{3red}
e32	Evento não controlável 32
e42	Evento não controlável 42
e35	Evento controlável 35
d_e41	Desabilitação do evento 41
d_e31	Desabilitação do evento 31

(a) (b)

Figura 26 – Descrição em LD: (a) Supervisor S_{3red}; (b) Desabilitações

Por fim, as sequências operacionais de cada máquina são programadas também como máquinas de estados, cujos eventos iniciais são disparados pelos sinais provenientes do SP e os seguintes pelos sinais de entrada do CLP. Assim, cada máquina opera como determina a lógica de cada sequência e, ao fim do ciclo de processamento, é sinalizado ao SP ativando-se a variável booleana correspondente.

4.2 Codesys e Tecnologia de Comunicação

O software CODESYS[1] é a ferramenta utilizada para implementar a lógica programável em CLP virtual para controlar a planta virtualizada. Trata-se de uma plataforma de tecnologia de automação industrial utilizada como ferramenta de programação IEC-61131-3. Possui integração com diversos fabricantes de controladores, permite utilização de diferentes tecnologias de comunicação, além de diversos recursos necessários para configuração e interface com o simulador 3D.

[1] CODESYS - Disponível em: https://codesys.com. Em: 18 mai. 2024

Para comunicação com o software de simulação gráfica, é utilizada a tecnologia Open Platform Communications Unified Architecture - OPC UA[2] (ISO/IEC, 2020). Compatível com as atuais demandas da Indústria 4.0, essa tecnologia de comunicação *open-source* possui configuração simples de servidor e cliente e não é dependente de sistema operacional MS Windows, como em versões anteriores.

O servidor OPC UA é configurado através do CODESYS, que, uma vez inicializado, estabelece a comunicação com o cliente (Factory IO) recebendo uma lista de variáveis globais 'GIO' criada no programa do usuário que realizam a troca de dados entre servidor e cliente, ilustrado pelo diagrama da Figura 27.

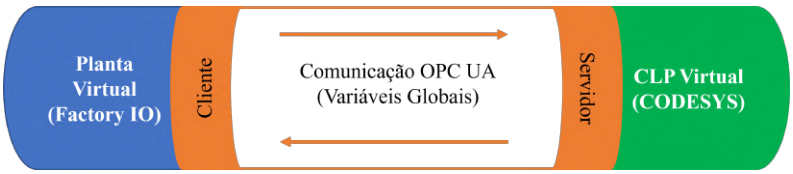


Figura 27 – Representação da comunicação via OPC UA

Uma vez realizada a implementação no CODESYS e configurada a comunicação OPC, é necessária a configuração do Factory IO para endereçar todos os elementos de campo utilizados com as variáveis globais provenientes do programa, para enviar os sinais dos sensores como elementos de entrada e receber os comandos para os atuadores realizarem tarefas. Essa etapa é configurada no próprio software, como apresenta a Figura 28.

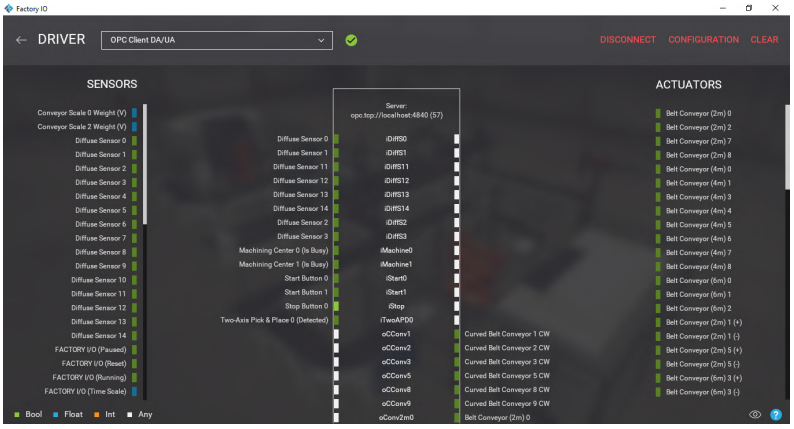


Figura 28 – Endereçamento dos elementos de campo com as variáveis globais

4.3 Considerações sobre o Modelo Virtual

No modelo virtual, os dois Pick & Place 3D que foram utilizados para modelar o robô e da máquina de montagem, operam com variáveis do tipo 'FLOAT' para posiciona-

[2] OPC UA - Disponível em: https://opcfoundation.org/about/opc-technologies/opc-ua/. Em: 18 mai. 2024

mento dos eixos x, y e z (Figura 29). Essa é uma abstração do sistema, que, apesar de modelado como SED, possui características de operação com variáveis contínuas, fato que é comumente presente em aplicações industriais.

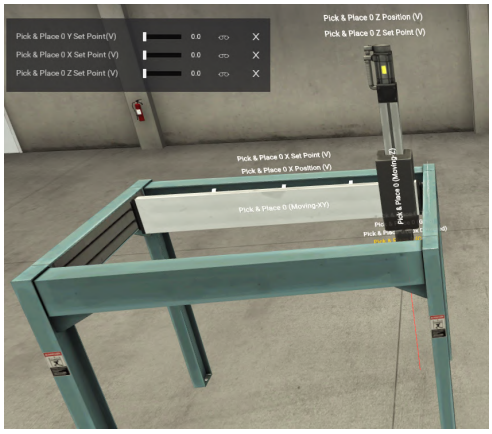


Figura 29 – *Set points* com variáveis contínuas do Pick & Place 3D

Dessa maneira, na construção das sequências operacionais desses dois dispositivos, são utilizados blocos de função do tipo 'MOVE' que aplicam valores pré-definidos às variáveis contínuas correspondentes para o sistema de controle. Os valores aplicados configuram o *set point* desejado, posicionando os eixos do Pick & Place de acordo com a sequência operacional executada. Variáveis do tipo 'FLOAT' também são utilizadas como entrada do sistema de controle para leitura da posição dos eixos atuais.

Apesar do sistema possuir elementos que operam com variáveis contínuas, o que realmente importa para o controle supervisório é o evento discreto que inicia uma operação e que indica o fim da mesma, independente da sequência intermediária que realiza essa operação. Essa é uma abstração comumente presente quando um sistema real é modelado por SED.

O problema da escolha existente entre os eventos 51 e 53 do SFM não é tratado nesta implementação, a solução adotada foi desenvolvida no próprio software de simulação, onde é possível atribuir uma aleatoriedade entre diferentes matérias-primas (blocos brutos) de entrada nas esteiras C_1 e C_2 (Figura 30). Dessa maneira, de acordo com o tipo de matéria-prima de entrada, determina-se qual evento será gerado, isto é, na esteira C_1 entram apenas blocos verdes para confecção de bases, já na esteira C_2 podem entrar blocos cinzas, que determinarão a escolha do evento 51 e, portanto, a confecção de pinos cônicos, ou blocos azuis que determinarão a escolha do evento 53 e a confecção de pinos cônicos. A probabilidade da entrada de blocos cinzas ou azuis adotada é de 50% e pode ser configurada no software de simulação.

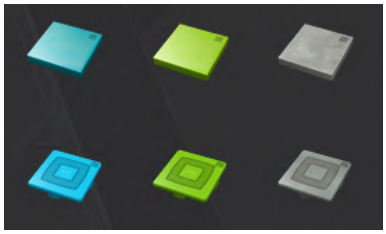


Figura 30 – Diferentes blocos brutos de entrada no sistema

4.4 Resultados

Com o desenvolvimento desta implementação, foi possível aplicar as teorias de controle supervisório em um ambiente de simulação que remete a aplicações reais. A descrição do código em LD utilizada, juntamente com a arquitetura adotada são as mesmas que podem ser utilizadas em CLPs reais, não se limitando à existência dos mesmos em laboratórios.

Os arquivos contendo o programa com a implementação em LD, e o modelo virtual da planta, apresentado na Seção 3.3, são disponibilizados no link: ⟨https://github.com/lacsed/Modelo-Virtual-SFM-arquivos⟩. Além disso, é disponibilizado um vídeo: ⟨https://lacsed.eng.ufmg.br/videos⟩ com uma simulação da planta virtual, onde são demonstradas as etapas do processo para confecção dos produtos.

A maior contribuição desse tipo de implementação verifica-se no ensino de automação. Simuladores gráficos 3D, como o Factory IO, aplicados à educação na engenharia, oferecem ao estudante uma experiência única e imersiva, com atividades que o levam a desenvolver o pensamento crítico, trazendo novas percepções sobre problemas práticos que muitas vezes não são encontrados nas teorias estudadas em sala de aula, mesmo que ainda em um ambiente de simulação. Abordagem tal que surge como alternativa aos elevados custos de aquisição e manutenção de laboratórios com plantas físicas.

Busca-se como principal contribuição no campo do ensino de automação industrial, a possibilidade de aplicação, tanto para escolas técnicas quanto universidades, de modelos como este apresentado, trazendo uma gama de novas possibilidades de aprendizado que não se limitam à disponibilidade de dispositivos físicos presentes em laboratório. Desta forma, espera-se contribuir para diminuição do distanciamento entre teoria e prática, utilizando o modelo como instrumento para aplicação de diversas teorias associadas à automação industrial estudadas em sala de aula, como: programação de CLPs, sistemas dinâmicos a eventos discretos, instrumentação e redes industriais, dentre outras (FREITAS; PENA, 2023).

5 IMPLEMENTAÇÃO II

Neste capítulo é realizada a implementação do Controle Supervisório em um CLP da Rockwell Automation, modelo Compact Logix L32E montado em um kit de automação didático que controla uma planta didática montada com diversos componentes do kit de robótica Fischertechnik, como apresentada na Seção 3.2.

Nesta implementação, é adotada a metodologia de Leal, Cruz e Hounsell (2012) apresentada na Seção 2.4.3.2, com uma estrutura onde é possível tratar mais de um evento por ciclo de *scan* do CLP, sem realizar saltos no programa.

5.1 Metodologia de Implementação

A estrutura da arquitetura utilizada nesta implementação é demonstrada através da Figura 31. De maneira geral, essa estrutura assemelha-se à Implementação I, dividindo o sistema de controle em três níveis (QUEIROZ; CURY, 2002), contudo, a ordem desses níveis é subdividida em etapas classificadas pela controlabilidade dos eventos.

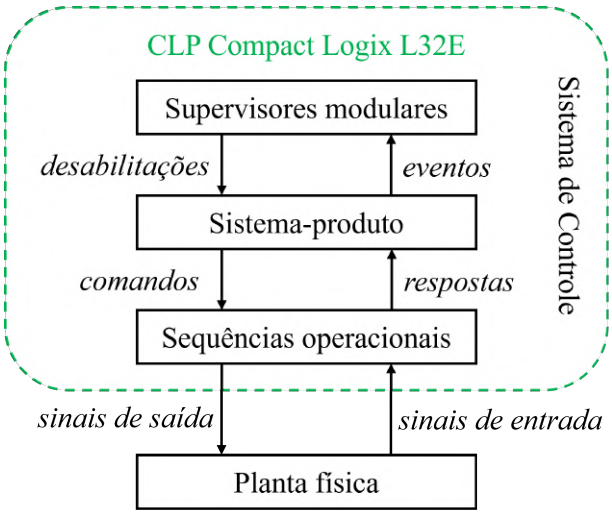


Figura 31 – Arquitetura da Implementação II. Adaptado de (QUEIROZ; CURY, 2002)

Leal, Cruz e Hounsell (2012) definem esta metodologia como "Definição da Implementação Conforme a Controlabilidade do Evento", em uma estrutura com 9 etapas (ou passos), denominada DECON9. De acordo com a metodologia, escreve-se o programa respeitando a ordem "eventos não controláveis - desabilitações - eventos controláveis", dessa maneira, consegue-se tratar todos os eventos não controláveis gerados pela planta em um ciclo de *scan*. A organização dessa estrutura, é apresentada pelo fluxograma da Figura 32.

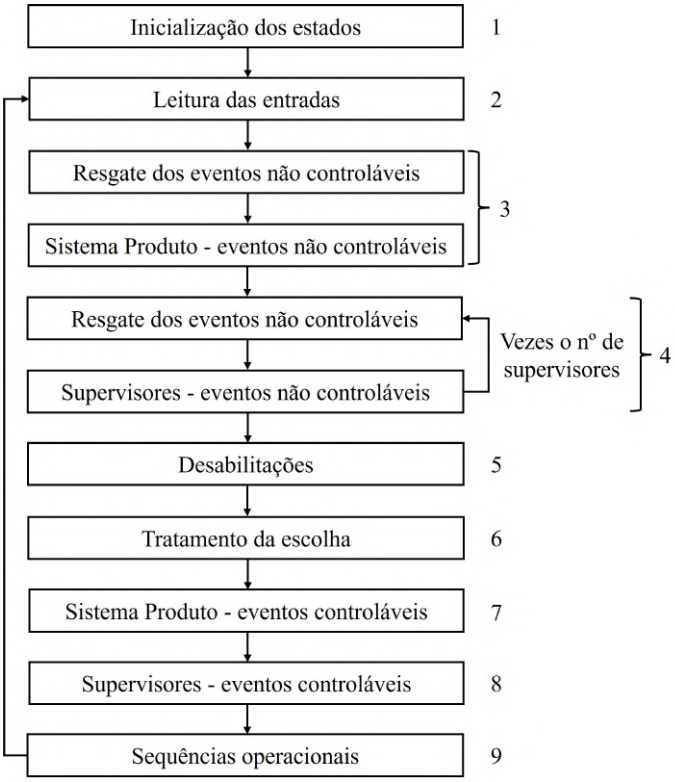


Figura 32 – Fluxograma com as etapas da estrutura "DECON9" (LEAL; CRUZ; HOUNSELL, 2012)

A organização do código que implementa essa estrutura foi desenvolvida por meio de uma rotina principal com diversas sub-rotinas no software RSLogix 5000, diferentemente da Implementação I, onde toda a estrutura era descrita em uma única rotina principal. Dessa forma, cada etapa descrita pelo fluxograma da Figura 32, é implementada em uma sub-rotina correspondente, tornando o programa mais limpo e organizado.

Seguindo essa ordem de etapas, a rotina principal é executada como descrito a seguir:

1. a rotina de inicialização atribui nível lógico '1' às variáveis booleanas referentes aos estados iniciais das plantas e dos supervisores (executada apenas no primeiro ciclo);
2. é realizada a leitura das entradas do CLP, registrando os eventos não controláveis gerados pela planta;
3. o Sistema Produto particionado com os eventos não controláveis gerados pela planta, realiza a mudança de estados;
4. os supervisores modulares particionados com os eventos não controláveis gerados pela planta, realizam a mudança de estados;
5. a rotina de desabilitações define a lista de eventos controláveis habilitados pelos supervisores;
6. a rotina de tratamento da escolha define o evento controlável escolhido (se houver);

7. o Sistema Produto particionado com as transições que envolvem eventos controláveis, realiza a mudança de estados, promovendo a geração dos eventos habilitados;
8. os supervisores modulares particionados com as transições que envolvem eventos controláveis gerados no passo anterior, realizam a mudança de estados;
9. finalmente os sinais para inicialização das sequências operacionais são realizados através da escrita nas saídas do CLP.

Dessa maneira, todas as mudanças de sinais na entrada do CLP, registradas em um único ciclo de *scan*, referentes aos eventos não controláveis gerados pela planta, são tratados, independentemente da ordem que foram gerados. De forma análoga, todos os eventos controláveis que não foram desabilitados pela rotina de desabilitações, são gerados neste mesmo ciclo. Portanto, a metodologia apresentada por Leal, Cruz e Hounsell (2012) aplicada neste capítulo, demonstra a vantagem de tratar um ou mais eventos em um mesmo ciclo de varredura, além de não utilizar saltos no código, como acontece na Implementação I, que utiliza a abordagem de Queiroz e Cury (2002).

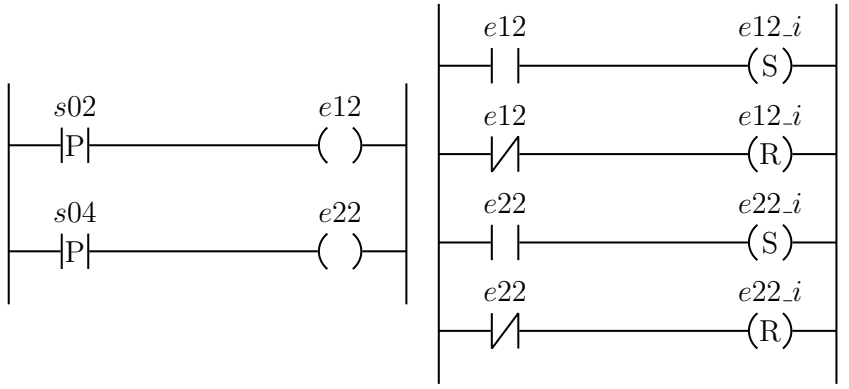
5.2 Represamento do Efeito Avalanche

Mas como o efeito avalanche é evitado? A solução apresentada por Leal, Cruz e Hounsell (2012) adotada nesta etapa, consiste em uma espécie de "represamento", onde, inicialmente, a informação da transição de um sinal de entrada é armazenada em uma variável booleana, e, a cada degrau de atualização do SP e dos supervisores, essa informação é apagada. Entretanto, para não perder a informação original das entradas, utiliza-se uma rotina intermediária auxiliar, denominada "Resgate dos eventos não controláveis" (indicada no passo 3 do fluxograma da Fig. 32), que consiste em um espelhamento da variável booleana que guardou a transição inicial em uma segunda variável onde, esta sim, será apagada a cada degrau de atualização.

Para exemplificar, o primeiro passo acontece na leitura das entradas do CLP, identificando os eventos gerados pela planta através da mudança dos sinais nas entradas. A Figura 33a apresenta o trecho onde as transições dos sinais 's02' e 's04', correspondentes aos sensores das respectivas esteiras C_1 e C_2, são identificadas pela borda positiva (ou de subida) e são armazenadas nas variáveis booleanas 'e12' e 'e22', respectivamente. Logo após, é criada a rotina de espelhamento, armazenando as informações das variáveis 'e12' e 'e22' nas variáveis "espelho" 'e12_i' e 'e22_i', respectivamente (Figura 33b).

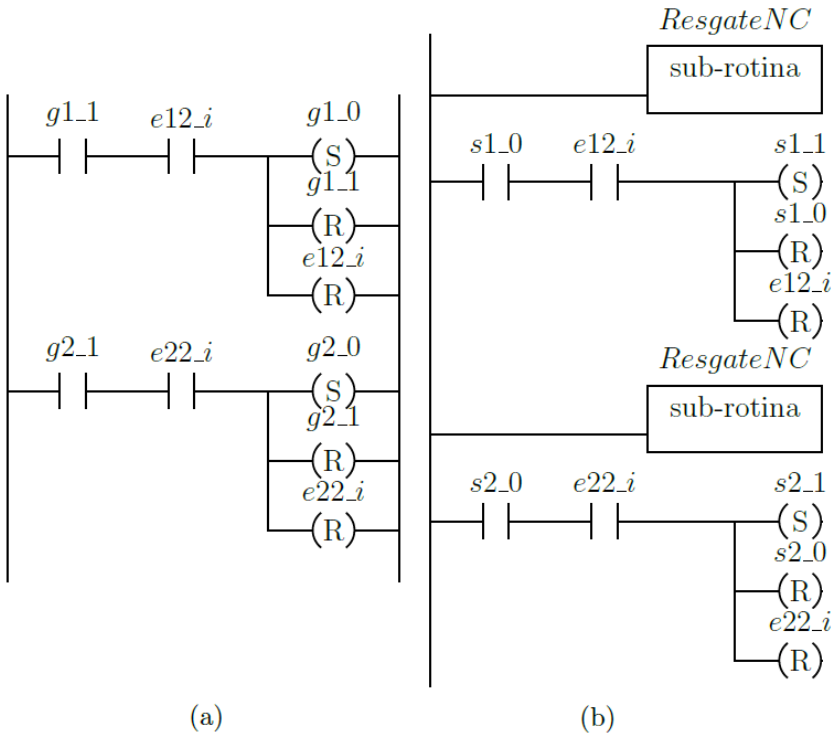
O próximo passo desta solução consiste na etapa de atualização de todos os estados do SP que envolvem eventos não controláveis, através das variáveis "espelho" que são apagadas na última linha de cada degrau, freando o efeito avalanche. A Figura 34a apresenta o trecho de atualização dos subsistemas G_1 e G_2.

Na próxima etapa do fluxograma da Figura 32, devem ser atualizados os super-



TAG	Descrição
s02	Sinal de entrada do sensor S02
s04	Sinal de entrada do sensor S04
e12	Evento não controlável 12
e22	Evento não controlável 22
e12_i	Evento não controlável 12 "espelho"
e22_i	Evento não controlável 22 "espelho"

Figura 33 – (a) Identificação nas transições dos sinais de entrada do CLP; (b) Rotina de espelhamento ou resgate dos eventos não controláveis



TAG	Descrição
g1_1	Estado 1 da esteira C_1 (ligada)
g2_1	Estado 1 da esteira C_2 (ligada)
s1_0	Estado 0 do supervisor S_{1red}
s2_0	Estado 0 do supervisor S_{2red}
e12_i	Evento não controlável 12 "espelho"
e22_i	Evento não controlável 22 "espelho"
Resgate NC	Rotina de resgate dos eventos não controláveis

Figura 34 – Atualização do SP e dos supervisores com o resgate dos eventos não controláveis

visores que envolvem eventos não controláveis gerados pela planta, através das variáveis "espelho". No entanto, como estas tiveram suas informações apagadas na etapa anterior, é necessário resgatar as informações de quais eventos foram gerados pela planta antes de atualizar cada supervisor. Para isso, basta chamar n vezes a rotina de espelhamento, intitulada "Resgate dos eventos não controláveis" antes de atualizar cada supervisor, onde n é igual ao número de supervisores utilizados nesta etapa (passo 4 do fluxograma da Fig. 32). É importante destacar que, também, na última linha de cada degrau de atualização dos supervisores as variáveis "espelho" devem ser apagadas novamente, represando o efeito avalanche (Figura 34b).

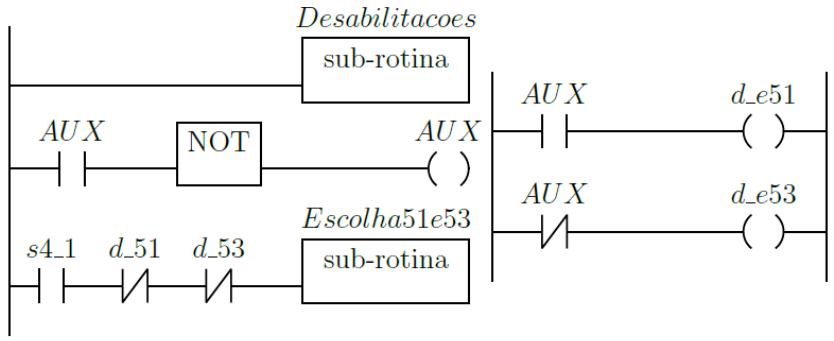
Desta forma se dá prioridade para o tratamento de eventos não controláveis,

mantendo o estado do SP e os estados dos supervisores em sincronia com o estado da planta física, evitando-se o efeito avalanche (CRUZ et al., 2009).

5.3 Tratamento da Escolha

O problema da escolha entre os eventos 51 e 53, descrito no Capítulo 3, evidencia a necessidade de uma solução, pois se realizada apenas a implementação natural, pela natureza de varredura do CLP, sempre será executado o evento que estiver no primeiro degrau LD, priorizando sempre um evento em detrimento do outro e limitando o sistema de alcançar todos os estados possíveis.

Nesta implementação foi adotada a *pseudo aleatoriedade* proposta por Leal, Cruz e Hounsell (2012). Esta solução consiste em adicionar uma rotina que é chamada após a rotina de desabilitações e somente será habilitada quando o supervisor S_{4red} estiver no estado '1', isto é, quando os eventos 51 e 53 estiverem habilitados. Nessa solução, é criada uma memória booleana interna, denominada 'AUX', que é utilizada na rotina principal como parte da etapa de escolha, e tem seu estado lógico invertido a cada ciclo de *scan* do CLP, através da lógica booleana NOT (Figura 35a). Esta mesma memória é utilizada dentro da sub-rotina de escolha, que, a partir do seu estado atual, desabilitará um dos dois eventos, determinando assim, qual evento estará habilitado e será gerado pelo SP (Figura 35b).



TAG	Descrição
Desabilitações	Rotina de desabilitações
AUX	Variável interna auxiliar
NOT	Função de inversão do estado da entrada
s4_1	Estado 1 do supervisor S_{4red}
d_51	Desabilitação do evento 51
d_53	Desabilitação do evento 53
Escolha 51e53	Rotina de escolha entre os eventos 51 e 53

Figura 35 – Tratamento da escolha: (a) etapa na rotina principal; (b) sub-rotina "Escolha51e53"

Adotando essa estratégia é possível promover uma aleatoriedade no tratamento da escolha entre os eventos do torno que resultam no tipo de pino que será produzido nesta etapa.

5.4 Detalhamento da Rotina do Robô

Devido às características físicas de operação, o eixo de giro do robô precisa se posicionar em sete pontos diferentes do seu curso, correspondentes aos buffers B_1 a B_7, indicados pela Figura 36. A cada sequência operacional que realiza, não é interessante ter uma posição de parada (*offset*) de fim de operação igual para todos os eventos, pois retardaria ainda mais as sequências operacionais, tornando o sistema menos eficiente sob o ponto de vista de tempo de execução.

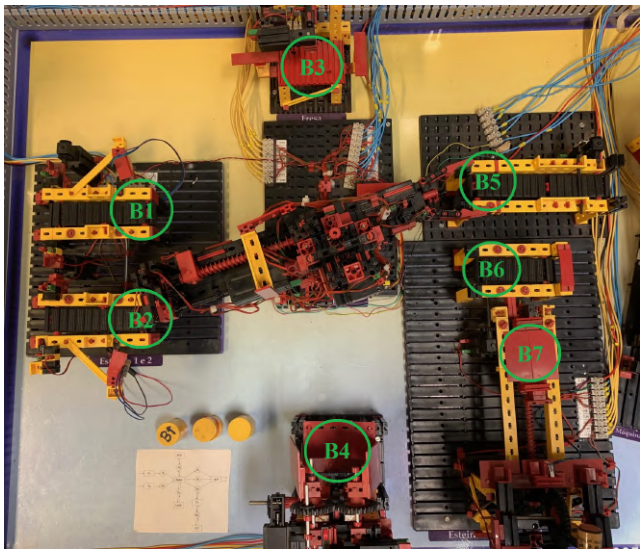


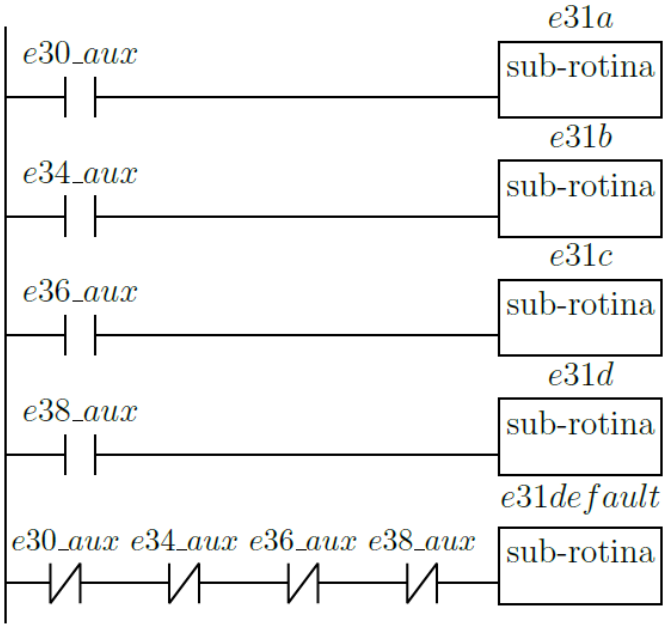
Figura 36 – Visão superior da planta com os pontos de posicionamento do eixo de giro do robô

Sendo assim, cada vez que o robô realiza uma operação, no caso os eventos 31/32, 33/34, 35/36, 37/38 e 39/30, ele para em um buffer diferente, correspondente ao último evento executado. Com isso, o robô pode partir de diferentes posições para um mesmo evento.

Sob o ponto de vista dos supervisores, da maneira que o sistema foi modelado, essa informação não é pertinente, bastando atualizar seus estados quando o SP gera um dos eventos controláveis que inicia uma tarefa, e a planta gera o evento não controlável que indica o fim desta tarefa.

Portanto, para solucionar esse problema sem alterar os supervisores, a rotina que executa as sequências operacionais do robô é extendida em uma série de sub-rotinas que são executadas com base no último evento executado, isto é, o último buffer que o eixo de giro do robô parou. Para isso, ao fim de cada sequência operacional, é ativada uma variável booleana auxiliar que fica em nível lógico '1' indicando qual o último estágio do robô. Quando uma nova sequência operacional é acionada, a partir de um dos eventos controláveis relacionados ao robô, uma nova rotina é executada com base nesta variável

booleana, onde seu respectivo degrau é “energizado” permitindo a execução da próxima sub-rotina. A Figura 37 representa um trecho da solução adotada para a rotina dos eventos ‘31/32’. Cada variável booleana ‘e3x_aux’, onde x é o número do último evento não controlável ocorrido, aciona o degrau que executa a próxima sub-rotina.



TAG	Descrição
e30_aux	Variável auxiliar que indica a conclusão da tarefa 39/30
e34_aux	Variável auxiliar que indica a conclusão da tarefa 33/34
e36_aux	Variável auxiliar que indica a conclusão da tarefa 35/36
e38_aux	Variável auxiliar que indica a conclusão da tarefa 37/38
e31a	Sub-rotina do evento 31 iniciando a partir do buffer B5
e31b	Sub-rotina do evento 31 iniciando a partir do buffer B4
e31c	Sub-rotina do evento 31 iniciando a partir do buffer B7
e31d	Sub-rotina do evento 31 iniciando a partir do buffer B6
e31default	Sub-rotina do evento 31 iniciando a partir do buffer B5

Figura 37 – Solução com sub-rotinas a partir do último evento do robô

No caso da primeira execução, como nenhuma das sub-rotinas foi executada anteriormente, é acionada a sub-rotina ‘default’ que parte da posição inicial do robô que é executada numa rotina de *setup*[1]. A Tabela 5 apresenta a lista dos eventos relacionados ao robô, com a descrição das suas operações, e todas as diferentes sub-rotinas que podem ser executadas para cada evento, iniciadas a partir dos buffers que indicam a última posição que o eixo de giro do robô parou.

5.5 Resultados

O SFM é objeto de estudo em diversos trabalhos acadêmicos já publicados, que geraram muitos frutos para a área de SEDs, como Pena et al. (2016), Alves, Pena e Takahashi (2021) e Pena et al. (2022). Ter novamente um protótipo do sistema em funcionamento no laboratório é um importante passo para o LACSED, vislumbrando a possibilidade de implementação de trabalhos futuros, tendo o protótipo como base de testes.

Além do funcionamento da planta, uma das contribuições deste trabalho foi obter os intervalos de tempo para realização de cada tarefa do sistema, isto é, foi possível

[1] A rotina de *setup* é uma rotina de inicialização, executada uma única vez no primeiro ciclo de *scan* do CLP, com a função de estabelecer posições iniciais (*offset*) para cada máquina do sistema.

Tabela 5 – Lista das sub-rotinas para os eventos do Robô

Eventos	Sub-rotina	Inicia a partir do buffer	Descrição
31/32	e31a	B5	Pega a peça em B1 e coloca em B3
	e31b	B4	
	e31c	B7	
	e31d	B6	
	e31default	B5 (primeira execução)	
33/34	e33a	B5	Pega a peça em B2 e coloca em B4
	e33b	B3	
	e33c	B7	
	e33d	B6	
	e33default	B5 (primeira execução)	
35/36	e35a	B3	Pega a peça em B3 e coloca em B5
	e35b	B4	
	e35c	B7	
	e35d	B6	
37/38	e37a	B3	Pega a peça em B4 e coloca em B6
	e37b	B4	
	e37c	B5	
39/30	e39a	B3	Pega a peça em B4 e coloca em B7
	e39b	B4	
	e39c	B5	

cronometrar os tempos de processamento entre eventos para todas as sub-plantas G_i, que, até então, como nos trabalhos anteriores citados, eram utilizados valores hipotéticos tabelados. A Tabela 6 apresenta os resultados obtidos, com os intervalos de tempo reais (em unidades de tempo - u.t.) entre eventos, classificados por planta.

O Apêndice A apresenta os dados com todos os resultados obtidos. O método para obtenção desses dados foi realizado cronometrando o intervalo de tempo entre os eventos de início e fim de cada tarefa. Por exemplo, o evento 41 inicia a operação da fresa, 28,02 segundos depois, o evento 42 ocorre no sistema, indicando o fim da operação. No caso do robô, como as tarefas podem iniciar a partir de locais diferentes, foram cronometrados os tempos de todas as sub-rotinas da Tabela 5. O resultado final foi a média aritmética de dez amostras de tempo para cada tarefa.

Tabela 6 – Intervalo de tempo entre eventos (em unidades de tempo - u.t.)

Plantas	Eventos controláveis	Eventos não controláveis	Tempo de operação [u.t]
C_1	11	12	1,41
C_2	21	22	1,50
Robô	31a	32a	18,20
	31b	32b	17,49
	31c	32c	18,60
	31d	32d	18,95
	31default	32default	18,20
	33a	34a	19,81
	33b	34b	18,62
	33c	34c	18,91
	33d	34d	19,48
	33default	34default	19,81
	35a	36a	13,80
	35b	36b	16,98
	35c	36c	18,17
	35d	36d	18,11
	37a	38a	17,54
	37b	38b	14,39
	37c	38c	18,69
	39a	30a	19,06
	39b	30b	15,39
	39c	30c	19,95
Fresa	41	42	28,02
Torno	51	52	26,29
	53	54	26,05
MM	61	-	6,54
	63	64	7,87
	65	66	8,78
C_3	71	72	6,81
	73	74	6,64
MP	81	82	12,86

6 IMPLEMENTAÇÃO III

Nos últimos anos, temas como comissionamento virtual, sistemas ciber físicos, gêmeos digitais, dentre outras abordagens, têm sido introduzidos, no contexto da Indústria 4.0, a fim de ampliar a utilização de ferramentas virtuais aliadas a sistemas de automação (ROLLE; MARTUCCI; GODOY, 2019).

Neste capítulo é idealizada e desenvolvida uma implementação com aplicação híbrida, utilizando a planta física do SFM em sincronização com o Modelo II da planta virtual desenvolvida no Factory IO. O objetivo é reproduzir, em tempo real, as ações do sistema real em ambiente virtual.

O controle supervisório é implementado no CLP Compact Logix L32E, presente na planta didática, juntamente com os mesmos elementos de campo da Implementação II. A comunicação entre RSLogix 5000 e Factory IO é feita através do protocolo TCP/IP, realizando a troca de dados entre os eventos gerados pela planta física e os eventos gerados pelo SP, através da estrutura de implementação em três níveis.

6.1 Metodologia de Implementação

A metodologia adotada nesta implementação tem sua estrutura semelhante à Implementação II, utiliza-se a estrutura hierárquica de três níveis, implementando os supervisores modulares, o SP, e as sequências operacionais (Figura 38).

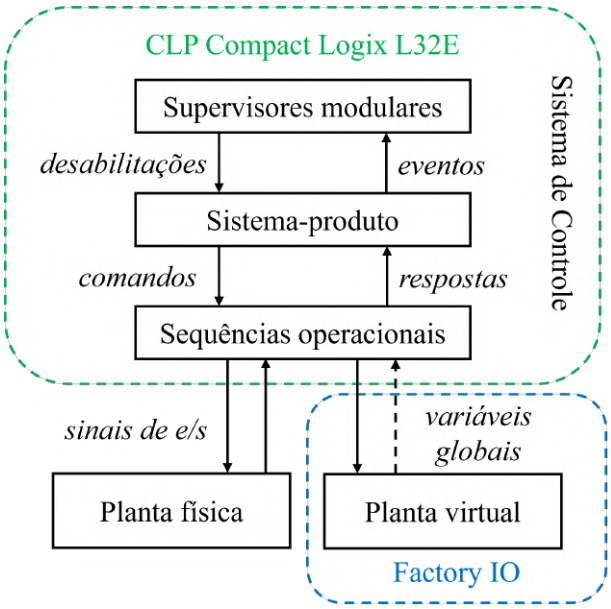


Figura 38 – Arquitetura da Implementação III

Observa-se, através da Figura 38, não apenas uma, mas agora duas plantas, física e virtual, que comunicam com o sistema de controle supervisório implementado no CLP. A estrutura DECON9 (LEAL; CRUZ; HOUNSELL, 2012) que ordena as etapas de acordo com a controlabilidade dos eventos, também é utilizada, possibilitando o tratamento de todos os eventos gerados pela planta e pelo SP em um único ciclo de *scan* do CLP, seguindo as mesmas diretrizes da Implementação II.

A principal diferença na estrutura desta implementação consiste na adição de uma nova etapa, isto é, uma rotina adicional para comunicação com a planta virtual. Esta nova etapa é criada após as sequências operacionais ilustrada através do fluxograma da Figura 39.

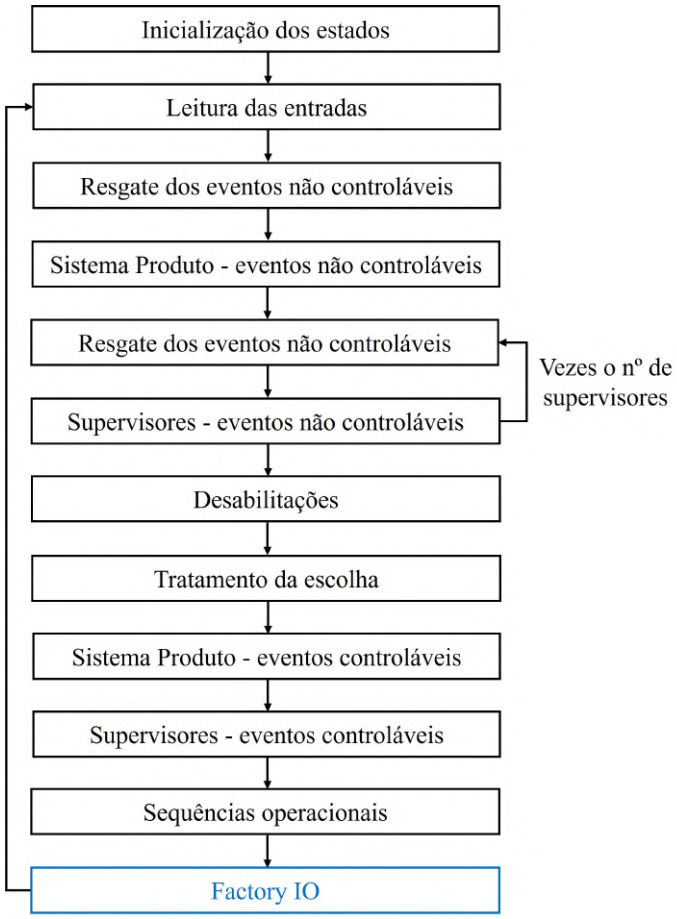


Figura 39 – Fluxograma com as etapas da Implementação III, adaptado de (LEAL; CRUZ; HOUNSELL, 2012)

Essa nova rotina, intitulada "Factory IO", funciona como uma interface entre objeto físico e virtual, com a função de tratar todas as variáveis globais, utilizadas na comunicação, para enviar e receber dados da planta virtual. Essas variáveis são criadas como variáveis do controlador (ou globais) com a função de transmitir os sinais, que correspondem aos eventos gerados pela planta e pelo SP, para o Factory IO, com o objetivo de realizar as mesmas operações que acontecem na planta física em tempo real.

Esta etapa não somente envia e recebe sinais booleanos, como também é responsável por tratar os eventos transmitidos, como por exemplo, a inicialização das tarefas do robô e da máquina de montagem, que utilizam Pick & Place 3D que operam através de variáveis contínuas para posicionamento dos eixos x, y e z, conforme apresentado na Seção 4.3. Em outras palavras, após gerado o evento controlável pelo SP e enviado o sinal para inicialização da sequência operacional pela planta física, é também inicializada uma sequência adaptada para a planta virtual, que habilita variáveis globais do tipo 'FLOAT' responsáveis por realizar as operações equivalentes ao respectivo evento controlável.

6.2 Sincronização entre Plantas Física e Virtual

Para não haver atrasos e perda de sincronia entre planta física e virtual, mesmo que uma determinada tarefa seja visualmente concluída em tempo menor pela planta virtual, o fim dessa tarefa só será concluído quando o evento não controlável gerado pela planta física for recebido pelo sistema de controle, respeitando a hierarquia de etapas. Por exemplo: quando gerado um evento referente ao robô, o mesmo será inicializado nas duas plantas, mesmo que o robô da planta virtual chegue ao local de destino primeiro, a peça só será solta quando o robô soltá-la na planta real, indicando o fim da operação.

Além disso, devido à velocidade constante das esteiras virtuais, foram adicionados blocos de entrada e saída de peças para ajuste na sincronização. Como por exemplo, quando uma peça é processada pelo torno, representado pela machine center da Figura 40, ela sai pelo bloco de saída presente na saída da máquina e é criada novamente já no ponto do buffer B_3 onde o robô irá transportar para o buffer B_5, não havendo a necessidade de continuidade entre esteiras e preservando a sincronia com os eventos gerados e executados pela planta física.

Figura 40 – Sincronização utilizando blocos de entrada e saída de peças na planta virtual

Dessa forma, planta física e virtual trabalham de maneira sincronizada, respeitando eventos controláveis para inicialização e não controláveis para término de tarefas.

A comunicação entre RSLogix e Factory IO foi feita de forma direta via protocolo TCP/IP, endereçando o CLP diretamente no Factory IO, que possui integração com os softwares dos principais fabricantes de CLPs, incluindo os modelos da Rockwell.

6.3 Tratamento da Escolha

O problema da escolha, existente entre os eventos 51 e 53 do SFM, foi solucionado, nesta implementação, utilizando uma técnica de *revezamento*. Nesta solução, um dos eventos é escolhido aleatoriamente na primeira execução, e, a partir daí, o próximo evento será sempre diferente do último executado, fazendo com que sempre alternem entre si, ou seja, produzindo sempre um pino cônico e um cilíndrico alternadamente.

Esta solução é implementada utilizando uma variável booleana que tem sua bobina ativada ao término de cada evento de escolha, ficando em nível lógico '1' até que o outro evento seja executado. Além disso, na rotina de desabilitações é utilizado um contato auxiliar normalmente aberto destas mesmas variáveis, em paralelo com os contatos dos supervisores que desabilitam o último evento, 'forçando' a desabilitação e permitindo que esse revezamento aconteça.

É importante destacar que nem sempre esse tipo de solução é viável quando se tratam de sistemas reais, o objetivo desta etapa foi apresentar uma solução alternativa ao problema de escolha para fins demonstrativos de operação da planta para visitantes.

6.4 Resultados

A implementação apresentada neste capítulo foi inspirada em temas atuais que permeiam a indústria moderna. Neste contexto, modelos digitais são usados para simular, analisar e prever o comportamento e o desempenho dos seus equivalentes físicos ao longo do tempo.

Com a conclusão desta etapa, foi possível observar a operação do SFM sob o controle supervisório, utilizando de metodologias de implementação conhecidas, com sua execução monitorada em tempo real pelo modelo digital desenvolvido. Isso viabiliza uma gama de novas possibilidades, dentro de uma variedade de aplicações, desde o treinamento e programação de CLPs até o nível de gestão ligado à redução de custos e aumento da produtividade. Modelos digitais estão se tornando uma parte essencial da transformação digital nas indústrias, impulsionando a eficiência, inovação e resiliência operacional.

A rotina principal do programa, implementado no RSLogix 5000, com todas as etapas descritas na Seção 6.1, programadas como sub-rotinas é apresentada no Anexo A.

7 GERAÇÃO EXTERNA DE EVENTOS CONTROLÁVEIS

Este capítulo é dedicado à apresentação de uma alternativa à maneira em que os eventos controláveis são gerados pelo sistema de controle. Essa alternativa consiste em uma modificação da estrutura da Implementação III deste trabalho, apresentada no Capítulo 6.

Até aqui, da maneira que as implementações foram delineadas, não existe uma política que defina quando e qual evento controlável deve ser gerado pelo SP. Considerando que os eventos não controláveis são gerados pela planta e, portanto, não podem ser desabilitados, tampouco determinar quando ocorrem. Portanto, de acordo com a lógica do SP que gera os eventos controláveis, se um evento não está desabilitado, ele será gerado, obedecendo o tratamento da escolha, quando existir.

Estudos que lidam com planejamento e *scheduling* de tarefas na produção, utilizam a TCS de modo a estabelecer um espaço de busca e desenvolver heurísticas que determinam sequências ótimas, sob determinados parâmetros. Uma heurística pode avaliar, por exemplo, o intervalo de tempo entre eventos para determinar uma sequência de eventos controláveis que devem ser gerados, com o objetivo de minimizar o tempo de produção (*makespan*) de um lote de produtos. Com destaque para os trabalhos de Pena et al. (2016), Malik e Pena (2018), Alves, Pena e Takahashi (2021) e Pena et al. (2022).

O objetivo aqui é elaborar uma solução onde seja possível receber informações externas para geração dos eventos controláveis, sem alterar as premissas hierárquicas de implementação já apresentadas. Para isso, foi desenvolvida uma rotina adicional, após as desabilitações, que recebe dados externos através de uma variável global e gera eventos controláveis de acordo com a informação recebida, se estes não estiverem desabilitados. A Figura 41 apresenta o fluxograma com a estrutura modificada.

Essa rotina foi programada na linguagem de Texto Estruturado (ST) e seu pseudocódigo é apresentado no Algoritmo 1. Basicamente, ele recebe qual evento deve ser gerado através da entrada '*Evento_C*' e, com base no dado recebido, entra na função '*Caso*' que verifica se este evento está habilitado ou não, utilizando as variáveis booleanas da rotina de desabilitações. Se o evento recebido estiver habilitado, ele faz com que a variável booleana correspondente ao evento receba o valor '1' e retorna, por meio da variável de saída 'Resposta' a informação de que o evento foi iniciado. No caso de o evento encontrar-se desabilitado, nenhum evento será gerado, retornando a informação de que o evento está desabilitado pelos supervisores. Na prática, devido à forma como tais sequências são obtidas, sabe-se que os eventos controláveis desabilitados estarão habilitados em algum momento, depois que eventos não controláveis ocorram na planta. Dessa maneira, eventos

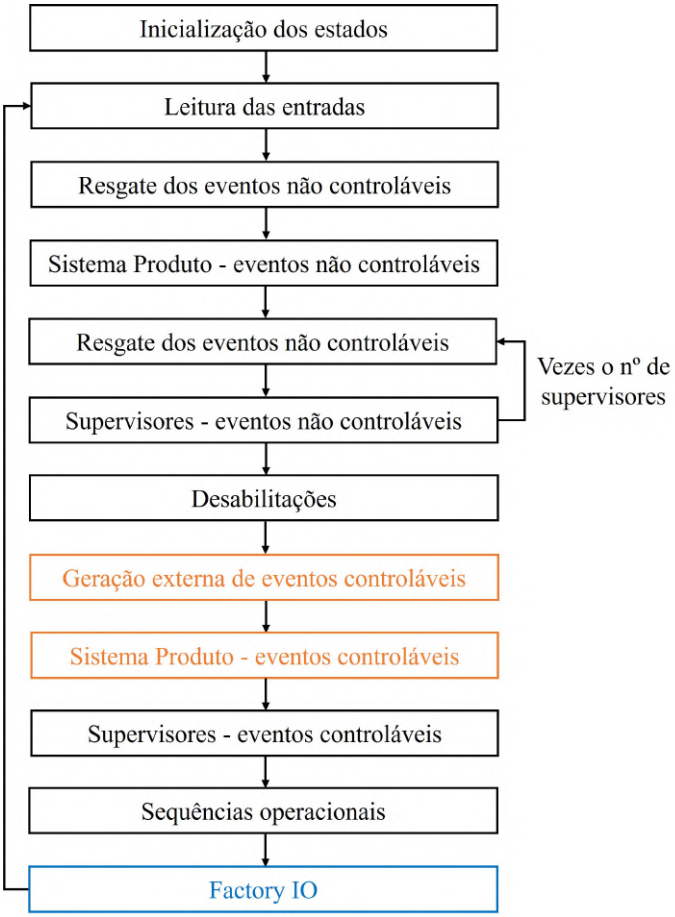


Figura 41 – Fluxograma modificado com a rotina adicional

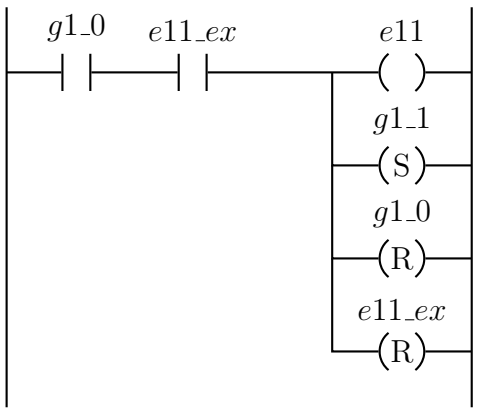
controláveis obtidos a partir de algoritmos de otimização externos serão gerados através dessa rotina, respeitando a rotina de desabilitações.

Além disso, é necessária uma modificação na rotina seguinte, que consiste na etapa de atualização do SP que abrange os eventos controláveis. Essa modificação consiste em tornar essa rotina 'passiva', com a função de apenas atualizar o SP se houver geração de quaisquer eventos controláveis pela rotina anterior. A Figura 42 apresenta o trecho em LD que implementa a atualização da planta G_1, que passa do estado 'g1_0' para o estado 'g1_1' quando a variável booleana 'e11_ex' é ativada pela rotina de geração, realizando a transição de estados da planta e inicializando a variável booleana do evento 11 de fato.

Essa rotina não mais gerará espontaneamente um evento controlável quando este não estiver desabilitado, como nas implementações anteriores, um evento controlável somente será gerado quando um dado externo for recebido. As demais rotinas permanecem inalteradas, respeitando a estrutura hierárquica de eventos não controláveis e atualização dos supervisores. Com base na proposta idealizada neste capítulo foi possível simular, com dados atribuídos manualmente à variável de entrada, e verificar o comportamento do sistema de controle perante cada informação recebida.

Algoritmo 1: Geração de eventos controláveis externos

```
   Entrada: Evento_C
   Saída: Resposta
 1 selecione Evento_C faça
 2     caso Evento_11 faça
 3         se Evento_11 está habilitado então
 4             e11_ex ⟵ 1
 5             Resposta ⟵ "Evento 11 iniciado"
 6         senão
 7             Resposta ⟵ "Evento 11 desabilitado"
 8         fim
 9     fim
10     ...
11     caso Evento_81 faça
12         se Evento_81 está habilitado então
13             e81_ex ⟵ 1
14             Resposta ⟵ "Evento 81 iniciado"
15         senão
16             Resposta ⟵ "Evento 81 desabilitado"
17         fim
18     fim
19 fim
```



TAG	Descrição
g1_0	Estado 0 da esteira C_1 (desligada)
g1_1	Estado 1 da esteira C_1 (ligada)
e11_ex	Variável que indica o evento 11 recebido externamente
e11	Evento controlável 11

Figura 42 – Rotina de atualização do SP com base no evento controlável gerado

8 CONCLUSÃO

Uma das principais motivações no desenvolvimento deste trabalho foi a possibilidade de associar teorias, como a modelagem de SEDs, linguagens e autômatos e a TCS, com aplicações práticas, o que é um desafio ainda muito grande e pouco explorado na área. Outro desafio também foi de trabalhar com o protótipo físico do SFM, que havia sido construído em projetos passados e estava parado, impossibilitando a exibição e demonstração de um dos resultados mais interessantes do LACSED.

Outra motivação foi o desenvolvimento de um modelo de realidade virtual como instrumento na área da docência, onde atuo, lecionando disciplinas de controle, automação e instrumentação industrial, dos cursos de nível técnico e superior do Instituto Federal de Minas Gerais.

O trabalho foi construído em etapas, inicialmente com o estudo, através de toda a fundamentação teórica que baseia a síntese do controle supervisório e suas formalidades, foi realizada uma revisão sobre a implementação em controladores industriais, onde problemas típicos foram apresentados e possíveis soluções discutidas. Logo após, foi apresentada a planta do SFM como estudo de caso, como foram desenvolvidos os aprimoramentos necessários para o funcionamento do protótipo físico e também como foi planejado e delineado um modelo digital do sistema. A primeira implementação foi desenvolvida explorando apenas o ambiente virtual, que trouxe contribuições importantes, no âmbito didático, publicadas em Freitas e Pena (2023). Posteriormente, os resultados foram reproduzidos no protótipo, com maior complexidade prática de dispositivos, interfaces de entradas e saídas, de comunicação, etc. possibilitando melhorias na estrutura e organização do programa e investigação de diferentes metodologias. Por fim, o modelo digital foi incorporado à implementação física, gerando um modelo composto (ou híbrido) com o objetivo de explorar temas atuais, como comissionamento virtual, gêmeos digitais, etc. difundidos na indústria moderna.

Com os resultados obtidos e o protótipo do SFM em funcionamento sob o controle supervisório, foi possível estabelecer intervalos de tempo reais entre eventos, que representam os tempos de processamento das tarefas do sistema, sendo este um passo importante, no desenvolvimento de estudos futuros, que lidam com algoritmos de otimização de sequências para produção, podendo utilizar a planta como estudo de caso e banco de testes.

Foi investigada uma alternativa, quanto à política de geração de sequências de eventos controláveis, e elaborada uma solução com uma rotina baseada em dados gerados externamente. Não foram realizados testes da solução proposta com algoritmos de

scheduling de tarefas para o SFM, resumindo-se apenas ao teste com simulação de dados. Para isso, é necessária uma interface de comunicação para estabelecer como estas tarefas serão geradas externamente e recebidas pelo CLP.

8.1 Perspectivas para Trabalhos Futuros

Como propostas para desenvolvimentos futuros, destacam-se:

- Explorar linguagens de alto nível para reproduzir as implementações apresentadas, como SFC;
- Desenvolver uma interface de comunicação com algoritmos de otimização;
- Realizar a otimização online, que permita a produção de lotes de produtos por meio de sequências otimizadas, com a estrutura de implementação sob o controle supervisório;
- Desenvolver um gêmeo digital do SFM com uma integração completa entre modelos físico e virtual, com troca de dados bidirecionais;
- Utilizar o modelo virtual como instrumento de apoio ao ensino de automação em salas de aula e avaliar os resultados no ensino/aprendizagem.

REFERÊNCIAS

ALVES, L. V. R.; PENA, P. N. Secure recovery procedure for manufacturing systems using synchronizing automata and supervisory control theory. *IEEE Transactions on Automation Science and Engineering*, PP, p. 1–11, 12 2020.

ALVES, L. V. R.; PENA, P. N.; TAKAHASHI, R. H. C. Planning on discrete event systems using parallelism maximization. *Control Engineering Practice*, v. 112, p. 104813, 2021. ISSN 0967-0661.

BALEMI, S.; BRUNNER, U. Supervision of discrete event systems with communication delays. In: *1992 American Control Conference*. [S.l.: s.n.], 1992. p. 2794–2798.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to discrete event systems*. 2ª. ed. [S.l.]: Editora Springer, 2009.

CRUZ, D. L. L.; LEAL, A. B.; ROSSO-JR, R. S. U.; ROZARIO, J. G. Proposta de implementação de controle supervisório em controladores lógico programáveis. In: SBAI2009. *IX Simpósio Brasileiro de Automação Inteligente*. [S.l.], 2009.

CURY, J. E. R. Teoria de controle supervisório de sistemas a eventos discretos. In: SBAI2001. *V Simpósio Brasileiro de Automação Inteligente*. Canela, Rio Grande do Sul, 2001.

DIETRICH, P.; MALIK, R.; WONHAM, W. M.; BRANDIN, B. A. Implementation considerations in supervisory control. In: _____. *Synthesis and Control of Discrete Event Systems*. Boston, MA: Springer US, 2002. p. 185–201. ISBN 978-1-4757-6656-1.

FABIAN, M.; HELLGREN, A. PLC-based implementation of supervisory control for discrete event systems. In: *Proc. 37th IEEE Conf. Decision Control*. [S.l.: s.n.], 1998. p. 3305–3310.

FREITAS, G. M.; PENA, P. N. Modelo virtual de um sistema flexível de manufatura como instrumento no ensino de automação industrial. In: SBAI2023. *Anais do XVI Simpósio Brasileiro de Automação Inteligente*. [S.l.], 2023.

FREY, G.; LITZ, L. Formal methods in PLC programming. *IEEE Conference on systems Man and Cybernetics*, v. 4, p. 2431–2436, 2000.

HELLGREN, A.; FABIAN, M.; LENNARTSON, B. On the execution of sequential function charts. *Control Engineering Practice*, v. 13, n. 10, p. 1283–1293, 2005.

HOPCROFT, J.; MOTWANI, R.; ULLMAN, J. *Introduction to Automata Theory, Languages, and Computation*. 3^{rd}. [S.l.]: Pearson, 2006.

ISO/IEC. *International Standard IEC 62541:2020, OPC Unified Architecture*. 2020.

JAMES, L. D.; TEIXEIRA, C. A.; LEAL, A. B. Formal design and implementation of supervisory controller for a didactic manufacturing cell. In: *2019 IEEE International Conference on Industrial Technology (ICIT)*. [S.l.: s.n.], 2019. p. 935–940.

KRITZINGER, W.; KARNER, M.; TRAAR, G.; HENJES, J.; SIHN, W. Digital twin in manufacturing: A categorical literature review and classification. *16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018: Bergamo, Italy*, v. 51, p. 1016–1022, 2018.

LEAL, A. B.; CRUZ, D. L. L.; HOUNSELL, M. S. PLC-based implementation of local modular supervisory control for manufacturing systems. In: _____. [S.l.: s.n.], 2012. p. 159–182. ISBN 978-953-51-0530-5.

MALIK, P. Generating controllers from discrete-event models. *MOVEP*, p. 337–342, 2002.

MALIK, R.; PENA, P. N. Optimal task scheduling in a flexible manufacturing system using model checking. *IFAC-PapersOnLine*, v. 51, n. 7, p. 230–235, 2018. ISSN 2405-8963. 14th IFAC Workshop on Discrete Event Systems WODES 2018.

MARTIN, J.; BOHUSLAVA, J. Augmented reality as an instrument for teaching industrial automation. In: *2018 Cybernetics & Informatics (K&I)*. [S.l.: s.n.], 2018. p. 1–5.

MORAES, C. C.; CASTRUCCI, P. L. *Engenharia de Automação Industrial*. 2ª. ed. Rio de Janeiro: LTC, 2010.

MULER, O. P.; QUEIROZ, M. H.; CURY, J. E. R. Síntese e implementação de controle supervisório em rede foundation fieldbus. In: CBA2018. *XXII Congresso Brasileiro de Automática*. [S.l.], 2018.

PENA, P. N.; COSTA, T. A.; SILVA, R. S.; TAKAHASHI, R. H. C. Control of flexible manufacturing systems under model uncertainty using supervisory control theory and evolutionary computation schedule synthesis. *Information Sciences*, Elsevier, v. 329, p. 491–502, 2016.

PENA, P. N.; CURY, J. E. R.; LAFORTUNE, S. Verification of nonconflict of supervisors using abstractions. *IEEE Transactions on Automatic Control*, v. 54, n. 12, p. 2803–2815, 2009.

PENA, P. N.; VILELA, J. N.; ALVES, M. R. C.; RAFAEL, G. C. Abstraction of the supervisory control solution to deal with planning problems in manufacturing systems. *IEEE Transactions on Automatic Control*, v. 67, n. 1, p. 344–350, 2022.

QAMSANE, Y.; TAJER, A.; PHILIPPOT, A. Distributed supervisory control synthesis for discrete manufacturing systems. *IFAC-PapersOnLine*, Elsevier, v. 49, n. 12, p. 396–401, 2016.

QUEIROZ, M. H.; CURY, J. E. R. Modular supervisory control of large scale discrete event systems. In: *Proceedings of 5th WODES*. [S.l.: s.n.], 2000. p. 103–110.

QUEIROZ, M. H.; CURY, J. E. R. Synthesis and implementation of local modular supervisory control for a manufacturing cell. In: *Proceedings of 6th WODES*. [S.l.: s.n.], 2002. p. 377–382.

QUEIROZ, M. H.; CURY, J. E. R.; WONHAM, W. M. Multitasking supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, v. 15, p. 375–395, 2005.

RAMADGE, P. J. G.; WONHAM, W. M. The control of discrete event systems. *Proceedings of the IEEE*, IEEE, v. 77, n. 1, p. 81–98, 1989.

REZENDE, J. S.; NETO, M. M. A.; PENA, P. N.; MAIA, C. A. Estudo comparativo de quatro técnicas de implementação em clp de sistemas a eventos discretos. In: CBA2012. *Anais do XIX Congresso Brasileiro de Automática*. [S.l.], 2012. p. 2523–2530.

RIERA, B.; MARANGE, P.; GELLOT, F.; NOCENT, O.; MAGALHAES, A.; VIGARIO, B. Complementary usage of real and virtual manufacturing systems for safe plc training. *IFAC Proceedings Volumes*, v. 42, n. 24, p. 89–94, 2010. ISSN 1474-6670. 8th IFAC Symposium on Advances in Control Education.

RIERA, B.; VIGARIO, B. Home i/o and factory i/o: a virtual house and a virtual plant for control education. *20th IFAC World Congress*, v. 50, p. 9144–9149, 2017.

ROLLE, R. P.; MARTUCCI, V. O.; GODOY, E. P. Digitalization of manufacturing processes: Proposal and experimental results. In: IEEE. *2019 II Workshop on Metrology for Industry 4.0 and IoT*. [S.l.], 2019.

SILVEIRA, P. R.; SANTOS, W. E. *Automação e controle discreto*. 9ª. ed. São Paulo: Érica, 2009.

SU, R.; WONHAM, W. M. Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, v. 14, p. 31–53, 2004.

VIEIRA, A. D.; CURY, J. E. R.; QUEIROZ, M. H. A model for PLC implementation of supervisory control of discrete event systems. In: *2006 IEEE Conference on Emerging Technologies and Factory Automation*. [S.l.: s.n.], 2006. p. 225–232.

VIEIRA, A. D.; SANTOS, E.; QUEIROZ, M. H.; LEAL, A. B.; NETO, A.; CURY, J. E. R. A method for PLC implementation of supervisory control of discrete event systems. *IEEE Transactions on Control Systems Technology*, v. 25, p. 1–17, 04 2016.

WONG, K.; THISTLE, J.; MALHAME, R.; HOANG, H. Supervisory control of distributed systems: Conflict resolution. *Discrete Event Dynamic Systems*, v. 10, p. 131–186, 01 2000.

A

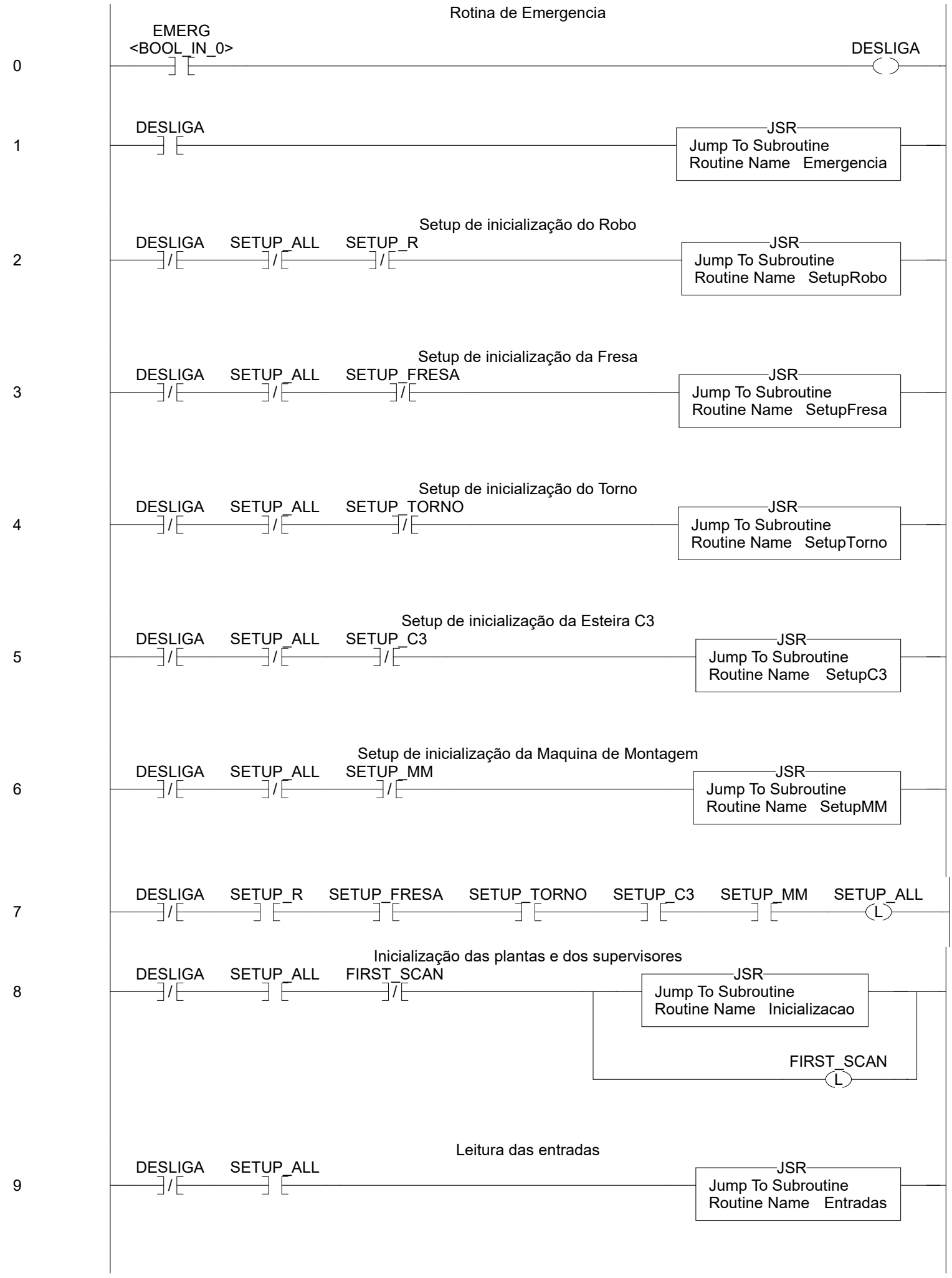
Apêndice A

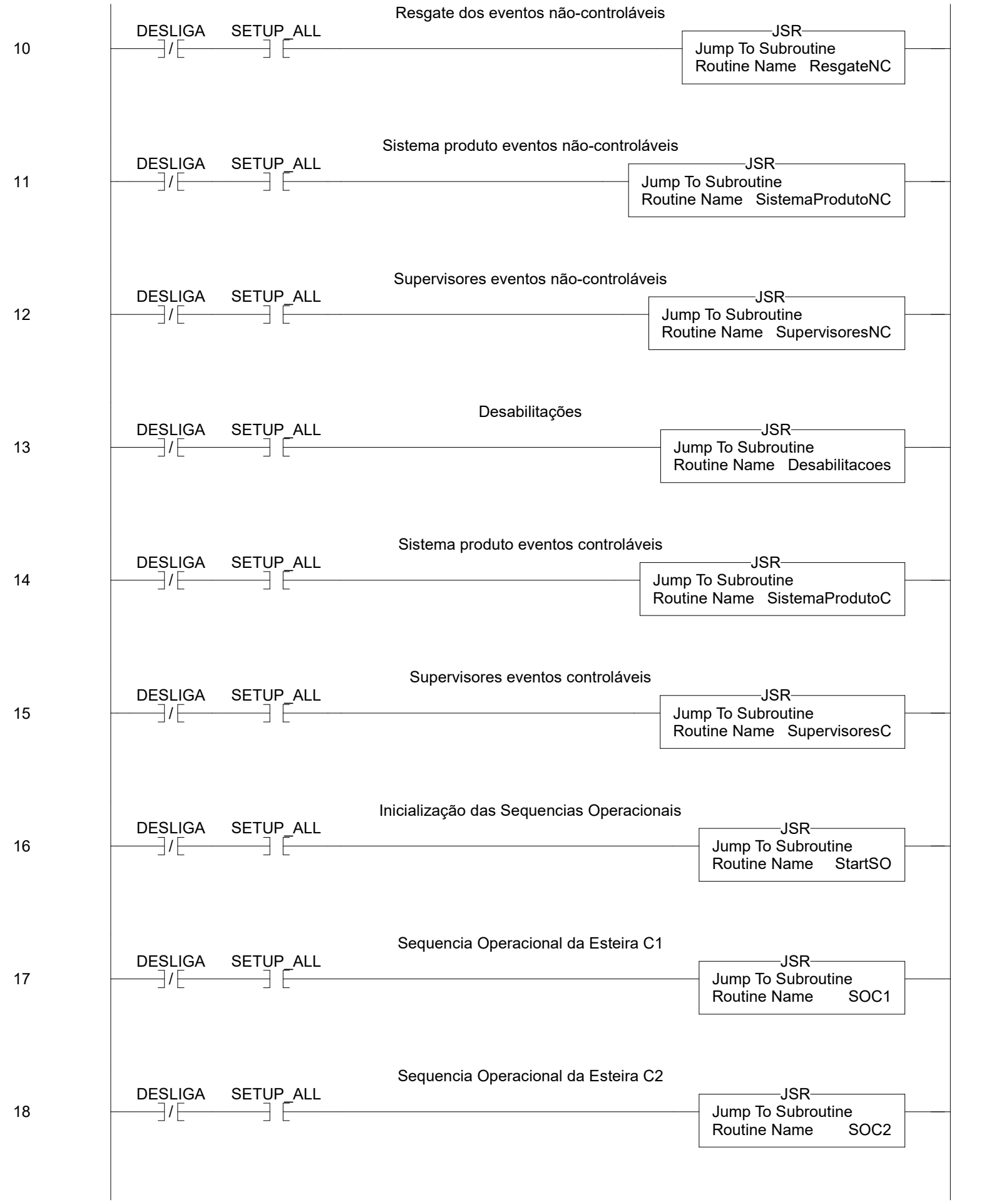
Intevalo de tempo entre eventos de início e fim de tarefas (em unidades de tempo - u.t.)

Planta	evento controlável	evento não controlável	medição 1	medição 2	medição 3	medição 4	medição 5	medição 6	medição 7	medição 8	medição 9	medição 10	média
C1	11	12	1,40	1,44	1,44	1,27	1,40	1,39	1,49	1,45	1,60	1,40	1,41
C2	21	22	1,42	1,52	1,54	1,54	1,55	1,45	1,38	1,50	1,54	1,49	1,50
Robô	31a	32a	18,50	18,16	18,05	18,25	18,38	18,05	18,01	18,28	18,14	18,06	18,20
	31b	32b	17,43	17,04	17,64	17,38	18,01	17,36	17,58	17,33	17,33	17,46	17,49
	31c	32c	18,69	18,59	18,58	18,30	19,71	18,59	18,68	18,53	18,37	18,26	18,60
	31d	32d	18,87	19,06	18,80	19,01	19,03	18,95	18,76	19,08	18,79	19,19	18,95
	31def	32def	18,50	18,16	18,05	18,25	18,38	18,05	18,01	18,28	18,14	18,06	18,20
	33a	34a	19,65	19,61	19,78	20,11	19,84	19,87	20,01	19,70	19,58	19,44	19,81
	33b	34b	18,59	18,36	18,43	18,45	18,68	18,79	18,98	18,77	18,49	18,59	18,62
	33c	34c	19,03	19,49	18,97	18,94	18,88	18,95	18,81	18,87	18,74	18,98	18,91
	33d	34d	21,30	19,48	19,44	19,28	19,26	19,16	19,12	19,09	19,45	19,38	19,48
	33def	34def	19,65	19,61	19,78	20,11	19,84	19,87	20,01	19,70	19,58	19,44	19,81
	35a	36a	14,21	13,63	13,91	13,44	14,06	13,63	13,83	13,79	13,93	13,74	13,80
	35b	36b	17,12	16,84	16,76	17,21	17,12	16,84	16,76	17,21	16,84	16,76	16,98
	35c	36c	18,12	17,72	18,40	18,30	17,99	18,19	18,14	17,88	18,34	18,05	18,17
	35d	36d	18,26	17,99	18,13	18,06	18,18	18,16	17,96	18,11	18,01	18,15	18,11
	37a	38a	17,33	17,74	17,76	17,76	17,43	16,99	17,70	17,66	17,56	17,43	17,54
	37b	38b	14,80	14,34	14,20	14,48	14,29	14,56	14,34	14,22	14,38	14,19	14,39
	37c	38c	18,60	18,81	18,78	18,65	18,89	18,48	18,62	18,68	18,75	18,79	18,69
	39a	30a	18,93	20,04	18,90	19,01	18,92	18,98	20,01	18,92	19,00	18,94	19,06
	39b	30b	15,49	15,66	15,06	15,27	15,65	15,48	15,78	15,09	15,17	15,61	15,39
	39c	30c	20,19	19,35	19,65	20,17	19,96	20,09	19,65	19,64	20,07	19,90	19,95
Fresa	41	42	27,93	27,91	28,00	28,01	27,94	27,94	28,30	27,97	28,04	28,03	28,02
Torno	51	52	26,19	26,24	26,26	26,19	26,41	26,26	26,34	26,31	26,51	26,23	26,29
	53	54	25,92	26,00	26,16	25,99	26,39	26,04	25,99	26,04	26,04	25,96	26,05
MM	61	-	7,00	6,34	6,35	6,38	6,68	6,40	6,54	6,56	6,61	6,46	6,54
	63	64	7,97	7,85	7,82	7,80	7,85	7,98	7,85	7,83	7,88	7,92	7,87
	65	66	9,00	8,64	8,89	8,70	8,77	8,86	8,57	8,83	8,79	8,73	8,78
C3	71	72	7,00	6,90	6,74	6,51	6,86	7,16	6,82	6,89	6,82	6,80	6,81
	73	74	6,74	6,51	6,63	6,45	6,66	6,62	6,99	6,51	6,61	6,69	6,64
MP	81	82	12,79	12,83	12,93	12,94	12,87	12,64	12,88	12,88	12,86	12,89	12,86

A

Anexo A


Rotina de Emergencia
EMERG
<BOOL_IN_0>
DESLIGA
0
DESLIGA
JSR
Jump To Subroutine
Routine Name Emergencia
1
Setup de inicialização do Robo
DESLIGA
SETUP_ALL
SETUP_R
JSR
Jump To Subroutine
Routine Name SetupRobo
2
Setup de inicialização da Fresa
DESLIGA
SETUP_ALL
SETUP_FRESA
JSR
Jump To Subroutine
Routine Name SetupFresa
3
Setup de inicialização do Torno
DESLIGA
SETUP_ALL
SETUP_TORNO
JSR
Jump To Subroutine
Routine Name SetupTorno
4
Setup de inicialização da Esteira C3
DESLIGA
SETUP_ALL
SETUP_C3
JSR
Jump To Subroutine
Routine Name SetupC3
5
Setup de inicialização da Maquina de Montagem
DESLIGA
SETUP_ALL
SETUP_MM
JSR
Jump To Subroutine
Routine Name SetupMM
6
DESLIGA
SETUP_R
SETUP_FRESA
SETUP_TORNO
SETUP_C3
SETUP_MM
SETUP_ALL
L
7
Inicialização das plantas e dos supervisores
DESLIGA
SETUP_ALL
FIRST_SCAN
JSR
Jump To Subroutine
Routine Name Inicializacao
FIRST_SCAN
L
8
Leitura das entradas
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name Entradas
9


Resgate dos eventos não-controláveis
10
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name ResgateNC
Sistema produto eventos não-controláveis
11
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SistemaProdutoNC
Supervisores eventos não-controláveis
12
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SupervisoresNC
Desabilitações
13
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name Desabilitacoes
Sistema produto eventos controláveis
14
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SistemaProdutoC
Supervisores eventos controláveis
15
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SupervisoresC
Inicialização das Sequencias Operacionais
16
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name StartSO
Sequencia Operacional da Esteira C1
17
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SOC1
Sequencia Operacional da Esteira C2
18
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SOC2

Sequencia Operacional do Robo
19
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SORobo
Sequencia Operacional da Fresa
20
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SOFresa
Sequencia Operacional do Torno
21
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SOTorno
Sequencia Operacional da Maquina de Montagem
22
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SOMM
Sequencia Operacional da Esteira C3
23
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SOC3
Sequencia Operacional da Maquina de Pintura
24
DESLIGA
SETUP_ALL
JSR
Jump To Subroutine
Routine Name SOMP
Variaveis Factory IO
25
SETUP_ALL
JSR
Jump To Subroutine
Routine Name FactoryIO
(End)