Universidade Federal
do Rio de Janeiro
Escola Politécnica

DESENVOLVIMENTO DE UM SISTEMA DE DESMONTAGEM DE PEÇAS EM UMA PLANTA MECATRÔNICA UTILIZANDO UM BRAÇO ROBÓTICO

Gabriel Pelielo Amorim de Mattos

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro de Controle e Automação.

Orientadora: Lilian Kawakami Carvalho

Rio de Janeiro
Março de 2018

DESENVOLVIMENTO DE UM SISTEMA DE DESMONTAGEM DE PEÇAS EM UMA PLANTA MECATRÔNICA UTILIZANDO UM BRAÇO ROBÓTICO

Gabriel Pelielo Amorim de Mattos

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE CONTROLE E AUTOMAÇÃO.

Examinado por:

Prof. Lilian Kawakami Carvalho, D.Sc.

Prof. Jomar Gozzi, D.Sc.

Prof. Eduardo Vieira Leão Nunes, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2018

Pelielo Amorim de Mattos, Gabriel

Desenvolvimento de um sistema de desmontagem de peças em uma planta mecatrônica utilizando um braço robótico/Gabriel Pelielo Amorim de Mattos. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2018.

XIII, 78 p.: il.; 29, 7cm.

Orientadora: Lilian Kawakami Carvalho

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Controle e Automação, 2018.

Referências Bibliográficas: p. 71 – 72.

1. Automação industrial. 2. Redes de Petri. 3. Braço robótico. 4. Planta mecatrônica. I. Kawakami Carvalho, Lilian. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Controle e Automação. III. Desenvolvimento de um sistema de desmontagem de peças em uma planta mecatrônica utilizando um braço robótico.

Ao caráter não estacionário da
vida que desafia a nossa
capacidade de controle.

Agradecimentos

Agradeço primeiramente aos meus pais, Lidia e José Carlos, que sempre se preocuparam em me oferecer um ensino da melhor qualidade e nunca deixaram de me apoiar nas minhas decisões. Vocês são a minha inspiração para ser uma pessoa melhor. Agradeço também às minhas avós, que por toda a minha vida ofereceram o carinho que eu precisava, e aos meus avôs, que infelizmente não estão aqui para poder presenciar a minha graduação, mas que estarão sempre no meu coração. Obrigado à minha felina preferida, por fazer parte da minha infância e da minha vida.

Um obrigado a todos os meus amigos da escola, que me ensinaram a formar fortes laços de amizade, e um agradecimento especial a Luiz Rennó, Luan Cassano, Antonia Corrêa, Tainá Medina e Mia Tiziano, que estiveram sempre ao meu lado, nas horas boas e nas ruins. Nunca esquecerei de vocês.

Agradeço também aos amigos que encontrei na faculdade, que me ajudaram imensamente a percorrer o árduo caminho que foi a graduação. Vocês tornaram as matérias difíceis em um trabalho em equipe muito prazeroso. Um abraço especial à todos da turma T-17 de Controle e Automação e aos meus eternos companheiros de grupo, Rafael Accácio e Rodrigo Moysés.

Agradeço à todos os professores que tive na vida, aos bons e aos ruins, que me ensinaram como ser e como não ser. O ensino de vocês teve um papel importantíssimo em quem eu me tornei. Um obrigado à minha orientadora Lilian Carvalho, que sempre se mostrou dedicada e me ajudou a escrever este trabalho. Obrigado Carlos e Elenilton, engenheiros mecânicos que me auxiliaram no projeto do cesto.

Por fim, um obrigado do fundo do meu coração à minha noiva, Maria Elisa de Souza Tiziano, que me ajudou imensamente a conseguir terminar este trabalho, eu te amo.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Controle e Automação.

DESENVOLVIMENTO DE UM SISTEMA DE DESMONTAGEM DE PEÇAS EM UMA PLANTA MECATRÔNICA UTILIZANDO UM BRAÇO ROBÓTICO

Gabriel Pelielo Amorim de Mattos

Março/2018

Orientadora: Lilian Kawakami Carvalho

Curso: Engenharia de Controle e Automação

O objetivo deste trabalho é desenvolver um sistema de desmontagem de peças na planta mecatrônica Christiani Sharpline, que é composta de três módulos: seleção, montagem e estocagem de peças. Um novo módulo será criado para desmontar as peças e devolver suas partes para o sistema de seleção. Para tanto, será necessário criar um ambiente de trabalho do braço robótico DENSO VP-6242 com um sistema de cesto para auxiliar na desmontagem das peças e uma esteira que leva as peças para perto do braço. Com esse novo módulo, o sistema de manufatura poderá funcionar de forma cíclica, sem a necessidade de repor a matéria-prima manualmente. Além disso, é apresentado um tutorial de instalação e uso do braço robótico DENSO VP-6242.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Automation and Control Engineering.

DEVELOPMENT OF A CUBE DISASSEMBLY PROCESS IN A MECHATRONIC SYSTEM USING A ROBOTIC ARM

Gabriel Pelielo Amorim de Mattos

March/2018

Advisor: Lilian Kawakami Carvalho

Course: Automation and Control Engineering

In this work, we present the development of a cube disassembly process on a Christiani Sharpline mechatronic system, which consists on three modules: sorting, assembly and storage. A new module will be created to disassemble the cube and return its parts to the sorting module. To do so, a work enviroment for the robotic arm DENSO VP-6242 will be put together with a cradle to help with the disassembly and a conveyor belt to carry the cube pieces to a place near the robotic arm. With this new module, the manufacturing system will be able to work as a cicle, without the need to feed the material manually. Furthermore, a tutorial to install and operate the DENSO VP-6242 robotic arm is introduced.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A revolução industrial do século XVIII trouxe novas tecnologias, como a mecanização e máquinas a vapor, para o ambiente de manufatura das fábricas, com o principal objetivo de aumentar a produtividade. Futuramente, o desenvolvimento tecnológico possibilitou a produção em massa através de linhas de montagem e da eletricidade, dividindo os sistemas de manufatura em setores, cada um responsável pela produção de um bem. A terceira mudança no ambiente industrial foi quando os primeiros sistemas automáticos foram adicionados, utilizando circuitos eletrônicos e controlados principalmente por computadores [5].

A automatização de sistemas vem sendo cada vez mais frequente em ambientes industriais, seja para aumentar a eficiência de processos, melhorar a qualidade de produtos ou para reduzir custos. Atualmente acredita-se que o mundo está vivendo a próxima revolução industrial, a da indústria 4.0, e uma das suas vertentes é a utilização de sistemas ciber-físicos, em que componentes físicos e de software são profundamente interligados. Alguns exemplos desse tipo de sistema são *smart grids*, veículos autônomos e sistemas robóticos [6].

Uma método para descrever o funcionamento e modelar sistemas de automação é o de Redes de Petri. Ao modelar processos industriais, é necessário utilizar eventos discretos com temporizadores e também sincronizá-los com eventos externos, cujo mapeamento é feito por sensores. Redes de Petri interpretadas para Controle (RPIC) podem ser usadas para modelar tais processos, já que possuem estruturas necessárias para representar esses fatores. O modelo da RPIC desenvolvido pode ter sua lógica implementada em controladores lógicos programáveis, que podem ser programados em diversas linguagens, sendo o diagrama Ladder a mais utilizada [7].

A planta mecatrônica de manufatura disponível no Laboratório de Controle e Automação - UFRJ é composta de três módulos: seleção, montagem e estocagem de peças. A primeira modelagem utilizando redes de Petri da planta foi apresentada em [8], enquanto que em [9], uma modelagem dividida em subsistemas é desenvolvida e sincronizada utilizando eventos virtuais. Entretanto, nenhum trabalho aborda o

funcionamento da planta de forma cíclica, sem a necessidade de repor a matéria-prima manualmente.

O objetivo deste trabalho é produzir um tutorial didático da instalação e utilização de um braço robótico, mostrando como é a programação utilizando o software Wincaps III do mesmo fabricante do braço. Além disso, um novo módulo será criado para desmontar as peças e devolver suas partes para o sistema de seleção. Para tanto, será necessário criar um ambiente de trabalho do braço robótico DENSO VP-6242 com um sistema de cesto para auxiliar na desmontagem das peças e uma esteira que leva as peças para perto do braço. Todos esses elementos foram modelados por redes de Petri modulares e depois convertidos para linguagem Ladder.

Este trabalho está organizado em cinco capítulos. No capítulo 2 serão apresentados os fundamentos teóricos de redes de Petri, controladores lógicos programáveis, os principais componentes do diagrama Ladder e um método de conversão de redes de Petri interpretadas para controle em um diagrama Ladder. No capítulo 3 será apresentado o braço robótico, mostrando o controlador usado para comandá-lo, os seus acessórios, um tutorial de como montar o seu ambiente de trabalho e como criar *scripts* para programar a operação do braço robótico. O capítulo 4 descreve todo o sistema de desmontagem, que é composto pelo braço robótico, a esteira e o cesto; a modelagem em rede de Petri modular e os resultados obtidos. Por fim, no capítulo 5 serão apresentadas as conclusões deste trabalho e ideias para trabalhos futuros.

Capítulo 2

Fundamentos Teóricos

Neste capítulo serão abordados os fundamentos teóricos para a compreensão deste trabalho, a começar por redes de Petri, seguido por controlador lógico programável (CLP), linguagem Ladder e finalizando com um método para conversão de redes de Petri interpretadas para controle em linguagem Ladder.

2.1 Redes de Petri

Em geral, um sistema pode ser definido como "um grupo de itens interdependentes que formam um conjunto unificado" [10], ou seja, um sistema pode ser fragmentado em diversos componentes, tendo cada um a sua função específica dentro do grupo, mas nenhum deles consegue funcionar independentemente.

Sistemas podem ser interpretados, dependendo do nível de abstração, como sistemas a eventos discretos (SED), que são constituídos de estados e eventos. Um evento discreto pode ser representado como uma ação espontânea que, quando ocorre, leva à mudança de um estado. Esse tipo de sistema pode ser modelado com autômatos ou redes de Petri, e neste trabalho serão utilizadas redes de Petri.

2.1.1 Grafo de uma Rede de Petri

Redes de Petri podem ser descritas graficamente através de grafos, que mostram as informações estruturais de toda a rede. A definição do grafo de uma rede de Petri é mostrada a seguir.

Definição 2.1 *O grafo de uma rede de Petri é uma quádrupla (P, T, Pre, Post), sendo* $P = \{p_1, p_2, ..., p_n\}$ *o conjunto dos lugares,* $T = \{t_1, t_2, ..., t_n\}$ *o conjunto das transições, Pre:* $P \times T \rightarrow \mathbb{N}$ *a função de ponderação dos arcos que ligam lugares a transições e Post:* $T \times P \rightarrow \mathbb{N}$ *a função de ponderação dos arcos que ligam transições a lugares [10].*

Nos grafos da Rede de Petri, os lugares são representados por círculos e as transições por barras verticais. Os lugares podem ser ligados por arcos, representados por setas, indicando o sentido do grafo e podendo ser ponderados.

As funções de ponderação dos arcos indicam o peso de cada arco da rede, sendo a função *Pre* relativa ao peso de um arco que vai de um lugar a uma transição e a função *Post* relativa ao peso de um arco que vai de uma transição a um lugar.

A figura 2.1 mostra um exemplo de grafo com arco ponderado, uma transição e dois lugares.

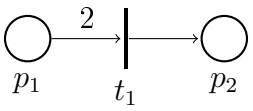


Figura 2.1: Exemplo de grafo de uma Rede de Petri.

O grafo pode ser descrito pela definição 2.1, sendo:

- $P = \{p_1, p_2\}$
- $T = \{t_1\}$
- $Pre(p_1, t_1) = 2$
- $Pre(p_2, t_1) = 0$
- $Post(t_1, p_1) = 0$
- $Post(t_1, p_2) = 1$

2.1.2 Rede de Petri Marcada

Uma rede de Petri marcada é um grafo de uma rede de Petri com a adição da função de marcação, utilizada para definir o estado da rede.

Definição 2.2 *Uma rede de Petri marcada é a quíntupla (P, T, Pre, Post, x) em que (P, T, Pre, Post) é o grafo da rede e* $x : P \to \mathbb{N}$ *é a função de marcação.*

É possível estabelecer a marcação dos lugares de uma rede de Petri para um dado estado [9] através de um vetor x, sendo x da seguinte forma:

$$x = \begin{bmatrix} x(p_1) \\ x(p_2) \\ \vdots \\ x(p_n) \end{bmatrix}, \tag{2.1}$$

em que $x(p_j)$, $j = \{1, 2, \ldots, n\}$, é a marcação do lugar p_j.

Exemplo 2.1 *Uma rede de Petri marcada é exemplificada pela figura 2.2(a), que mostra o modelo da reação química* $2H_2 + O_2 \rightarrow 2H_2O$ *[11].*

(a) Antes do disparo de t_1.

(b) Depois do disparo de t_1.

Figura 2.2: Exemplo de uma rede de Petri marcada.

No grafo da figura há duas fichas em p_1, outras duas em p_2 e nenhuma em p_3, ou seja, para esse estado, o vetor x que representa a função de marcação é $x = \begin{bmatrix} 2 & 2 & 0 \end{bmatrix}^T$.

2.1.3 Dinâmica da Rede de Petri

As ações de um sistema modelado por uma rede de Petri ditam a movimentação das fichas dentro do grafo, que define a dinâmica da rede. As seguintes notações ilustram as leis que regem essa dinâmica [12].

$I(t_j)$ é o conjunto de lugares de entrada de uma transição t_j:

$$I(t_j) = \{p \in P : Pre(p, t_j) > 0\} \tag{2.2}$$

$O(t_j)$ é o conjunto de lugares de saída de uma transição t_j:

$$O(t_j) = \{p \in P : Post(t_j, p) > 0\} \tag{2.3}$$

$I(p_i)$ é o conjunto de transições de entrada de um lugar p_i:

$$I(p_i) = \{t \in T : Post(t, p_i) > 0\} \tag{2.4}$$

$O(p_i)$ é o conjunto de transições de saída de um lugar p_i:

$$O(p_i) = \{t \in T : Pre(p_i, t) > 0\} \tag{2.5}$$

A movimentação de uma ficha muda o estado do sistema, e para que qualquer ficha saia de um lugar p_i para um lugar p_j, é necessário que haja uma transição que liga p_i e p_j e ela deve estar habilitada para ser disparada.

A figura 2.2 pode ser utilizada como exemplo para mostrar a dinâmica de uma rede de Petri. Em 2.2 (a), a transição t_1 está habilitada e depois de disparar, o resultado da rede será o mostrado por (b).

Definição 2.3 *Uma transição $t_j \in T$ é dita habilitada se*

$$x(p_i) \geq Pre(p_i, t_j), \forall p_i \in I(t_j). \tag{2.6}$$

Ou seja, a quantidade de fichas nos lugares p_i de entrada de uma transição t_j deve ser maior ou igual ao peso do arco que liga cada lugar $p_i \in I(t_j)$ a transição t_j.

Para disparar uma transição t_j, ela deve estar habilitada, e quando t_j dispara, o sistema passa do estado x para o estado x'. A seguinte equação modela a dinâmica do disparo de uma transição t_j:

$$x'(p_i) = x(p_i) + Post(t_j, p_i) - Pre(p_i, t_j), \text{ para } i = 1, 2, \ldots, n \tag{2.7}$$

Exemplo 2.2 *A figura 2.3 mostra uma rede de Petri com marcação inicial* $x_0 = \begin{bmatrix} 2 & 0 & 0 & 0 \end{bmatrix}^T$.

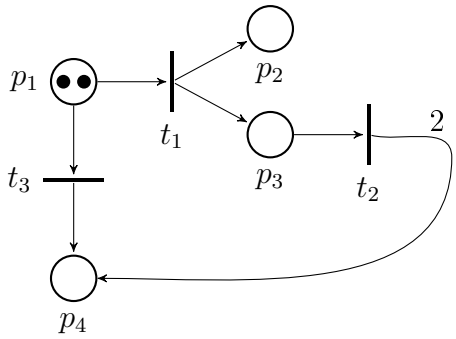


Figura 2.3: Grafo do estado inicial da rede de Petri.

A partir da inequação 2.6, sabemos que as transições t_1 e t_3 estão habilitadas, já que:

$$x(p_1) \geq Pre(p_1, t_1) \Rightarrow 2 > 1, \text{para } t_1 \text{ e}$$

$$x(p_1) \geq Pre(p_1, t_3) \Rightarrow 2 > 1, \text{para } t_3.$$

Se t_1 disparar, p_1 perderá uma das fichas e os lugares p_2 e p_3 ganharão uma ficha cada um, e o novo estado será $x_1 = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}^T$, como mostra a figura 2.4.

Utilizando novamente a inequação 2.6, as transições t_2 e t_3 estão habilitadas. Caso t_2 dispare e em seguida t_3, p_4 ganhará três fichas, uma provinda de p_1 e as outras duas de p_3, já que o arco de saída de t_2 tem peso 2. O estado final será $x_2 = \begin{bmatrix} 0 & 1 & 0 & 3 \end{bmatrix}^T$, como mostra a figura 2.5.

Esse exemplo ilustra a necessidade de incluir a definição de sequência de disparo, ou seja, é necessário ordenar os disparos de todas as transições de forma concatenada.

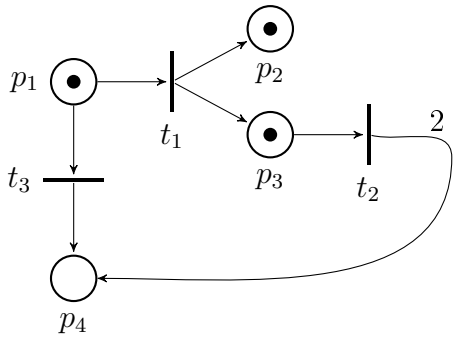


Figura 2.4: Grafo do estado x_1 da rede de Petri.

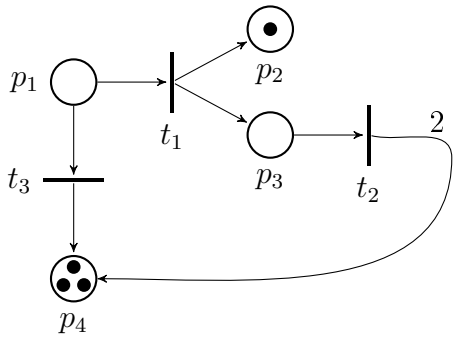


Figura 2.5: Grafo do estado x_2 da rede de Petri.

A sequência de disparo do exemplo anterior começando da marcação inicial x_0 até a x_2 foi $s = t_1t_2t_3$, significando que a primeira transição disparada foi t_1, seguida de t_2 e finalmente t_3.

Para ilustrar a diferença que a ordem de disparo faz no resultado do sistema, podemos novamente utilizar o exemplo anterior, mas dessa vez, se a ordem de disparo fosse $s' = t_1t_1t_2t_2$ a partir do estado inicial x_0, o resultado da marcação final seria $x' = \begin{bmatrix} 0 & 2 & 0 & 4 \end{bmatrix}^T$.

Equação de Estados

Se uma transição t_j dispara, de acordo com a equação 2.7, podemos abranger o modelo para a forma matricial da seguinte maneira:

$$\begin{aligned} x'(p_1) &= x(p_1) + Post(t_j, p_1) - Pre(p_1, t_j) \\ x'(p_2) &= x(p_2) + Post(t_j, p_2) - Pre(p_2, t_j) \\ &\vdots \\ x'(p_n) &= x(p_n) + Post(t_j, p_n) - Pre(p_n, t_j), \end{aligned}$$

que pode ser reescrito na forma matricial como:

$$\begin{bmatrix} x'(p_1) \\ x'(p_2) \\ \vdots \\ x'(p_n) \end{bmatrix} = \begin{bmatrix} x(p_1) \\ x(p_2) \\ \vdots \\ x(p_n) \end{bmatrix} + \begin{bmatrix} Post(t_j, p_1) \\ Post(t_j, p_2) \\ \vdots \\ Post(t_j, p_1) \end{bmatrix} - \begin{bmatrix} Pre(p_1, t_j) \\ Pre(p_2, t_j) \\ \vdots \\ Pre(p_n, t_j) \end{bmatrix}$$

ou como:

$$\underbrace{\begin{bmatrix} x'(p_1) \\ x'(p_2) \\ \vdots \\ x'(p_n) \end{bmatrix}}_{x'} = \underbrace{\begin{bmatrix} x(p_1) \\ x(p_2) \\ \vdots \\ x(p_n) \end{bmatrix}}_{x} +$$

$$+ \underbrace{\begin{bmatrix} Post(t_1, p_1) - Pre(p_1, t_1) & \dots & Post(t_m, p_1) - Pre(p_1, t_m) \\ Post(t_1, p_2) - Pre(p_2, t_1) & \dots & Post(t_m, p_2) - Pre(p_2, t_m) \\ \vdots & \ddots & \vdots \\ Post(t_1, p_n) - Pre(p_n, t_1) & \dots & Post(t_m, p_n) - Pre(p_n, t_m) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}}_{u}, \quad (2.8)$$

sendo A denominada matriz de incidência e u o vetor de disparo, que é um vetor nulo com apenas o valor 1 na posição j, indicando que a transição t_j foi disparada.

A Equação de Estados é:

$$\boxed{x' = x + Au} \quad (2.9)$$

A partir da equação de estados é possível definir o vetor contagem de disparo v. Para isso, seja x_0 o vetor de marcação inicial, e serão considerados por simplicidade apenas três vetores de disparo u_0, u_1 e u_2, tal que, seja obedecida a seguinte ordem de disparo:

$$x_1 = x_0 + Au_0$$
$$x_2 = x_1 + Au_1$$
$$x_3 = x_2 + Au_2,$$

logo:

$$x_3 = x_0 + A\underbrace{(u_0 + u_1 + u_2)}_{v} \tag{2.10}$$

isto é, $v = u_0 + u_1 + u_2$. Assim, a equação de estados passa a ser definida com o vetor de contagem de disparo v [11]:

$$\boxed{x' = x + Av} \tag{2.11}$$

Exemplo 2.3 *Considere a rede de Petri da figura 2.3. Caso a transição t_1 seja disparada, o vetor de disparo e a matriz de incidência serão:*

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \text{ e } A = \begin{bmatrix} -1 & 0 & -1 \\ 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 2 & 1 \end{bmatrix}.$$

A partir da equação 2.9, é possível encontrar a nova marcação x' da seguinte forma:

$$x' = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & -1 \\ 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Caso a sequência de disparo seja $s = t_1t_2t_3$, da mesma forma que foi escolhida no exemplo 2.2, o vetor de disparo u deve ser trocado pelo vetor de contagem de disparo v, que será:

$$v = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Utilizando dessa vez a equação de estados 2.11, a marcação final será:

$$x' = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & -1 \\ 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 3 \end{bmatrix},$$

como havia sido mostrado no exemplo 2.2 utilizando a equação 2.7.

2.1.4 Rede de Petri Estendida e Rotulada

Redes de Petri estendidas são redes de Petri com a adição de arcos inibidores, que ligam um lugar p_i a uma transição t_i e inibem o disparo de t_i se a quantidade de fichas em p_i for igual ou maior que o peso do arco que liga p_i a t_i.

Definição 2.4 *Uma rede de Petri estendida é uma sêxtupla (P, T, Pre, Post, x_o, I_n), sendo (P, T, Pre, Post) o grafo da rede de Petri, x_0 o vetor de estado inicial e $I_n : P \times T \to \mathbb{N}$ a função de ponderação do arco inibidor [13].*

A figura 2.6 mostra um exemplo de rede de Petri com arco inibidor. Para que t_1 esteja habilitada p_2 deve ter pelo menos uma ficha e p_1 deve ter menos de duas fichas, devido ao arco inibidor. Em (a), a transição t_1 não está habilitada e em (b), t_1 está habilitada.

(a) t_1 desabilitada. (b) t_1 habilitada.

Figura 2.6: Exemplo de grafo de uma Rede de Petri estendida.

Redes de Petri Rotuladas são redes de Petri com eventos associados à transições e marcação nos estados finais.

Definição 2.5 *Uma rede de Petri rotulada é uma óctupla (P, T, Pre, Post, E, l, x_0, X_m), sendo (P, T, Pre, Post) o grafo da rede de Petri, E o conjunto de eventos para rotulagem das transições, $l : T \to E$ a função de rotulagem, x_0 o vetor de marcação inicial e $X_m \subseteq \mathbb{N}^n$ o conjunto de estados marcados [10].*

A função de rotulagem das redes de Petri rotuladas é utilizada para associar um evento a uma transição. Ele indica uma condição necessária para o disparo de uma transição habilitada e é representado próximo à sua respectiva transição.

As marcações dos estados são utilizadas para dar a eles algum significado especial. Usualmente, o estado final é marcado, por ser aquele em que se quer chegar no fim da execução do sistema, e todos os estados marcados estão contidos no conjunto X_m.

Exemplo 2.4 *Considere um sistema que precisa indicar a presença de uma peça e se ela é de metal. Para isso, é necessário utilizar dois sensores. Um sensor fotoelétrico e um indutivo. Primeiro é checado se a peça está presente e depois se ela é de metal. A figura 2.7 ilustra a modelagem desse sistema em uma rede de Petri.*

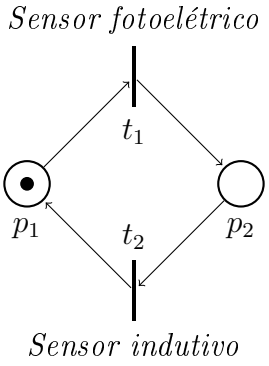


Figura 2.7: Exemplo de grafo de uma Rede de Petri rotulada.

Os eventos desse sistemas são: "Sensor fotoelétrico" e "Sensor indutivo". A transição t_1 está inicialmente habilitada. Se existir uma peça, o sensor fotoelétrico é acionado, disparando a transição t_1. Depois disso, a transição t_2 estará habilitada, já que o lugar p_2 terá uma ficha. Se a peça for de metal, então a transição t_2 será disparada, levando a ficha para o lugar p_1.

2.1.5 Rede de Petri Interpretada para Controle

Um sistema a eventos discretos pode ser modelado e controlado por uma rede de Petri Interpretada para Controle (RPIC), uma extensão da rede de Petri que é utilizada para lidar com sensores e atuadores do sistema. Além disso, podem também ser utilizados temporizadores em seu diagrama, ou seja, as transições podem estar associadas a um intervalo de tempo.

Os lugares da rede de Petri são então modelados como ações e relacionados com os comandos realizados pelo controlador e as transições como sinais dos sensores.

Dentro das ações existem duas categorias, ações impulsionais e ações contínuas. As ações impulsionais começam a ser executadas quando o lugar recebe uma ficha, e não param sua execução quando a ficha sai do lugar em que estava. Para interromper a atuação, é necessário especificar em outro lugar um evento de interrupção daquela ação. Para representar uma ação impulsional, um asterisco é colocado próximo à sua descrição, como ilustra a figura 2.8.

Figura 2.8: Exemplo de grafo de uma RPIC com ação impulsional.

Ao contrário das ações impulsionais, as ações contínuas param a sua execução assim que a ficha sai do lugar relacionado à ação. Utilizando a figura 2.9 como

exemplo de sistema de abertura de uma porta, ela é aberta assim que a ficha entra em p_1 e depois que a ficha sai, a porta fecha.

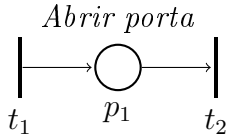


Figura 2.9: Exemplo de grafo de uma RPIC com ação contínua.

Outra característica das RPICs é a de temporizadores. Essas redes podem ter um tempo atrelado às suas transições e funcionam de acordo com duas regras:

- O tempo da transição começa a contar apenas depois que a essa se torna habilitada. Após terminar a contagem do tempo, a transição dispara e o contador zera.
- Caso uma transição se torne habilitada e desabilite antes de disparar, o seu temporizador é zerado.

As transições podem ser sincronizadas com eventos externos ao controlador ou temporizadas. As não temporizadas têm condições associadas, como ilustra a figura 2.10, em que c_j é uma condição e e_j é um evento associado à transição t_j.

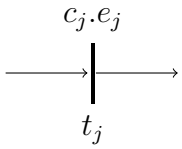


Figura 2.10: RPIC com condição associada à transição.

Definição 2.6 *Uma rede de Petri Interpretada para Controle é um conjunto* N_c = *(P, T, Pre, Post,* x_0, I_n, *C, E, l, D,* l_D, A_I, A_N, l_A*) em que [14]:*

- *(P, T, Pre, Post,* x_0, I_n*) é a rede de Petri estendida;*
- *C e E são os conjuntos de condições e eventos externos associados às transições não temporizadas pertencentes a* T_0*;*
- $l : T_0 \rightarrow (E \times C)$ *é a função que associa a cada transição não temporizada um evento e uma condição para o disparo da transição;*
- *D é o conjunto de atrasos de disparo associados às transições temporizadas* T_D*;*

- $l_D : T_D \rightarrow D$ *é a função que associa a cada transição de* T_D *um atraso do conjunto D;*
- A_I *denota o conjunto de ações impulsionais e de operações;*
- A_N *denota o conjunto de ações de nível;*
- $l_A : P \rightarrow (A_I \cup A_N)$ *é a função que associa aos lugares da rede uma ação pertencente a* $A_I \cup A_N$.

Exemplo 2.5 *Considere uma máquina de bebidas em que o procedimento para comprar uma bebida é o seguinte: Se a máquina estiver livre, primeiro é inserido o dinheiro nela e depois é selecionado o produto ao pressionar o respectivo botão na seleção de bebidas. A figura 2.11 mostra a rede de Petri modelada para esse exemplo.*

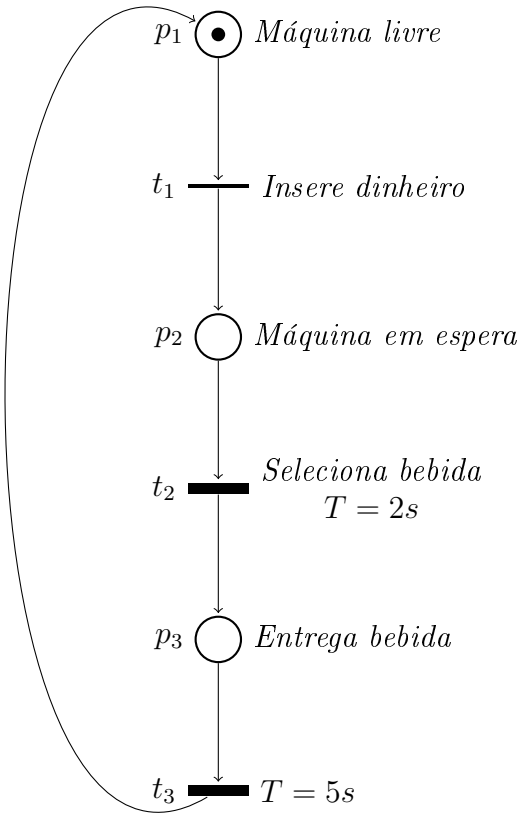


Figura 2.11: Modelo em rede de Petri para uma máquina automática de venda de bebidas.

A ficha no lugar p_1 indica que a máquina está livre e é possível utilizá-la. Ao inserir o dinheiro na máquina t_1 dispara e ela entra em um estado de espera para que o cliente selecione a bebida. Depois de apertar o botão desejado, a máquina leva 2 segundos para processar o pedido, que é representado pelo disparo da transição t_2, e depois entrega a bebida para o cliente quando a ficha estiver em p_3. Como forma de garantir que o produto foi entregue e retirado, a máquina só volta a estar livre para a próxima compra depois de 5 segundos, ou seja, só dispara t_3 depois desse tempo.

2.2 Controlador Lógico Programável

Um Controlador Lógico Programável (CLP) ou em inglês, *Programmable Logic Controller* (PLC) é um dos controladores mais populares da indústria e é utilizado para comandar e monitorar processos industriais [16].

O grande número de usuários desse tipo de controlador se deu principalmente pelas vantagens que ele oferece, tais como a facilidade de programação, de manutenção e a sua alta confiabilidade. Além do ambiente industrial, os CLPs também podem ser utilizados em residências nos projetos de automação residencial.

A interpretação do sinal de entrada é feita através da conversão das características físicas do sistema em sinais elétricos, que são lidos pelo CLP através das suas conexões de entrada, da mesma forma que os sinais elétricos de saída são emitidos pelo CLP e recebidos por atuadores, que os transformam em ações físicas.

Para guardar os valores dos sinais que passam pelo CLP são utilizadas suas regiões de memória, que são divididas em três regiões lógicas. O CLP utilizado neste trabalho foi um Siemens S7-300, e a identificação das regiões lógicas dessa fabricante é feita na primeira letra da variável, sendo I as que são de entrada, Q as de saída e M as de memória, utilizadas apenas para auxiliar na armazenagem de valores, não sendo diretamente convertidas para sinais elétricos. A segunda letra de uma posição de memória indica o número de bits utilizados para guardar informações, sendo X para 1 bit, B para 8 bits, W para 16 bits, D para 32 bits e Q para 64 bits. Os últimos números definem o lugar de memória do CLP.

Um CLP opera executando ciclos de varredura que consistem em três passos básicos:

1. Ler e guardar valores de variáveis de entrada.
2. Executar todo o código do programa do usuário.
3. Armazenar valores nas variáveis de saída.

Por isso, o CLP lê sinais gerados pelos sensores da planta, usa esses valores como variáveis de entrada para o programa, executa o código e, finalmente, gera os sinais

que serão aplicados à planta. Em geral, eventos de entrada são associados com a borda de subida ou com a borda de descida dos sinais que são gerados por sensores [15].

2.3 Linguagem Ladder

Um CLP pode ser programado em 5 linguagens, definidas pela norma IEC 61131-3, sendo o diagrama Ladder a mais utilizada. Ladder é uma linguagem simbólica que utiliza contatos, bobinas, temporizadores, contadores, instruções de comparação e simples operações matemáticas [15].

Escrever um programa em Ladder é equivalente a desenhar um circuito elétrico. O diagrama Ladder consiste de duas linhas verticais, representando os terminais elétricos e os circuitos são conectados como linhas horizontais, de forma que a criação de um programa se assemelha à de uma escada. O diagrama é lido da esquerda para a direita e de cima para baixo, e um ciclo é representado pela leitura da primeira a última linha do código uma vez [16].

A figura mostra todos os tipos de contatos e bobinas utilizados nesse trabalho, e a seguir serão feitas descrições de cada um desses elementos.

2.3.1 Contatos

Contatos representam condições para energizar uma linha do diagrama Ladder, sendo cada contato associado a uma variável de memória de entrada do CLP. Alguns dos contatos existentes serão mostrados a seguir.

Contato NA

Um contato NA (normalmente aberto) não fecha o circuito até que o valor da variável associada a ele seja igual a 1, e se esse valor mudar para 0, o contato abre.

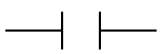


Figura 2.12: Contato NA.

Contato NF

Um contato NF (normalmente fechado) opera de forma oposta a um NA, porque só abre quando o valor da variável associada a ele é igual a 1 e fecha quando o valor for 0.

Figura 2.13: Contato NF.

Contato tipo P

Um contato tipo P é ativado durante apenas um instante, quando a borda de subida de um sinal é atingida, ou seja, no momento exato em que o valor de uma variável é alterado para um valor superior.

Figura 2.14: Contato tipo P.

Contato tipo N

Um contato tipo N funciona de forma análoga ao tipo P, porém é ativado na borda de descida do sinal, ou seja, quando o valor de uma variável atinge um valor inferior.

Figura 2.15: Contato tipo N.

2.3.2 Bobinas

Bobinas representam as saídas do CLP, e assim como os contatos, têm uma variável de memória associada a cada uma. Apenas uma bobina é permitida por linha do diagrama Ladder, e para energiza-la, é necessário que todos os contatos da sua linha também estejam energizados. Os principais tipos de bobinas serão mostrados a seguir.

Bobina simples

Uma bobina simples tem o seu valor lógico alterado para 1 quando é energizada e para 0 quando não está energizada.

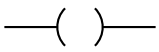


Figura 2.16: Bobina Simples.

Bobina SET

Uma bobina SET só pode ter o seu valor lógico alterado para 1, quando é energizada, e mesmo que pare de ser energizada, o seu valor lógico não muda.

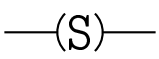


Figura 2.17: Bobina SET.

Bobina RESET

Para alterar o valor lógico de uma bobina SET para 0, é necessário utilizar uma bobina RESET, que tem o seu valor lógico alterado para 0 se ela for energizada, mantendo esse valor mesmo que deixe de ser energizada.

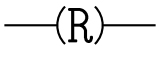


Figura 2.18: Bobina RESET.

2.3.3 Temporizadores

Blocos temporizadores são utilizados para contar tempo. Um exemplo é o bloco TON (*Timer on Delay*), que começa a contar um tempo determinado pela entrada PT assim que a sua entrada IN é energizada. Ao final da contagem, a sua saída Q é energizada, e só deixa de ser energizada quando IN também deixa, zerando a contagem de tempo. A imagem 2.19 ilustra um bloco TON.

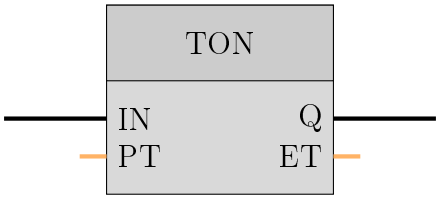


Figura 2.19: Bloco TON.

2.3.4 Contadores

Blocos contadores são utilizados para contar pulsos em um CLP. Um exemplo de bloco contador é o CTU (*Count Up*).

Contador CTU

Um bloco contador CTU conta quantos pulsos (bordas de subida) ocorreram no tempo maior que um ciclo de varredura do CLP. Quando a entrada CU é energizada, sua borda de subida é contabilizada e a quantidade armazenada na variável de saída CV. Quando o número em CV é igual ou maior a PV, a saída Q é energizada. Se a entrada R for energizada, a contagem é zerada. A imagem 2.20 ilustra um bloco CTU.

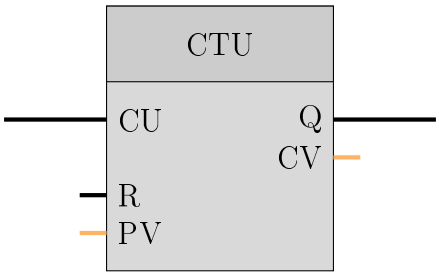


Figura 2.20: Bloco CTU.

2.4 Método de conversão entre RPICs e Ladder

Um método de conversão de um modelo de uma rede de Petri interpretada para controle em um diagrama Ladder foi proposto em [15]. O método consiste em separar o diagrama Ladder em 5 módulos diferentes: módulo dos eventos, módulo das condições de disparo de transições, módulo da dinâmica da rede de Petri, módulo de inicialização e módulo das ações.

Como exemplo, a seguir será feita a conversão para diagrama Ladder de todos os módulos para o caso da máquina de bebidas, apresentado inicialmente no exemplo 2.5.

1. Módulo dos Eventos

   O primeiro módulo é responsável por mapear as informações de algo que gere um evento e estiver na planta, podendo ser sensores, interruptores, botões ou chaves, seja para comandá-la ou monitorá-la. A cada um desses elementos é associado um contato NA, NF, tipo P ou tipo N, dependendo da sua função, e posteriormente a uma bobina, que carregará o estado do elemento durante a execução do diagrama Ladder.
   A figura 2.21 representa o diagrama Ladder do módulo de eventos da rede de Petri do exemplo da máquina de bebidas.

Figura 2.21: Lógica Ladder do módulo de eventos do exemplo da máquina de bebidas.

Na primeira linha do diagrama da figura 2.21, a variável de entrada *I1.0* está relacionada ao evento da transição t_1, de inserir o dinheiro na máquina. Na segunda linha, a variável *I1.1* está ligada à transição t_2, ao evento de apertar o botão de seleção de bebidas. As variáveis de memória *M2.1* e *M2.2* guardam o estado dos eventos no diagrama Ladder.

2. Módulo das Condições de Disparo de Transições

   O segundo módulo tem a finalidade de gerar uma linha no diagrama Ladder para cada transição da rede de Petri original, implementando assim as condições externas para que as transições sejam habilitadas. Cada transição é representada por uma bobina, os lugares de entrada da transição por contatos NA e os arcos inibidores por contatos NF. Com todos esses elementos colocados em série, é realizado o mapeamento da condição de disparo de cada transição.
   A figura 2.22 representa o o diagrama Ladder do módulo das condições de disparo de transições da rede de Petri do exemplo da máquina de bebidas.

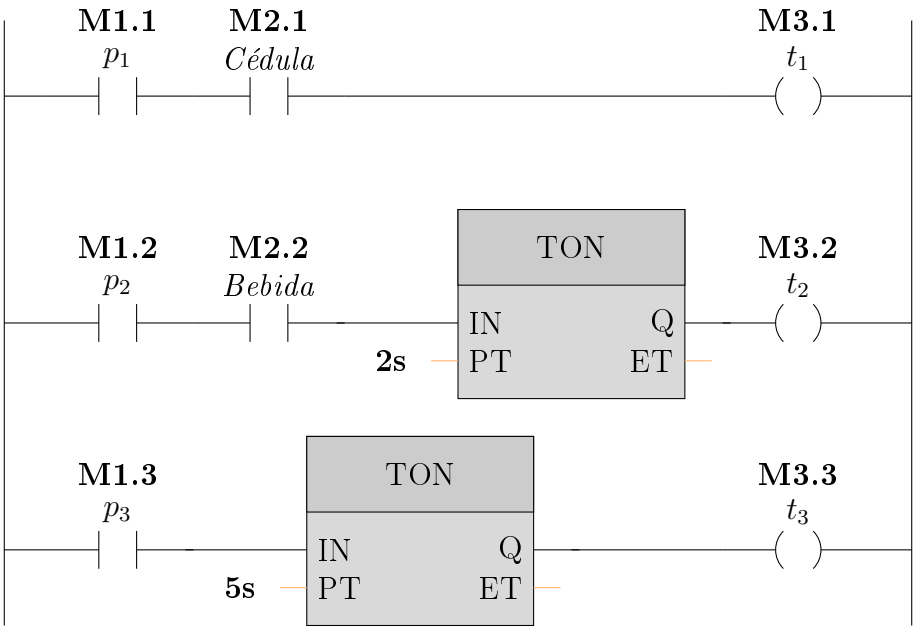


Figura 2.22: Lógica Ladder do módulo das condições de disparo de transições do exemplo da máquina de bebidas.

Na primeira linha do diagrama da figura 2.22, se existir uma ficha em p_1 (*M1.1*) e o cliente inserir dinheiro na máquina (*M2.1*), a transição t_1 será disparada (*M3.1*). De forma análoga, na segunda linha, se existir uma ficha em p_2 (*M1.2*) e o botão de seleção de bebidas for pressionado (*M2.2*), então o temporizador TON de 2 segundos será iniciado, e após esse tempo, a transição t_2 será disparada (*M3.2*). Na terceira linha, assim que houver uma ficha em p_3 (*M1.3*), o temporizador de 5 segundos será acionado, e ao seu término, a transição t_3 será disparada *M3.3*.

3. Módulo da Dinâmica da Rede de Petri

   O terceiro módulo representa a evolução do sistema, ou seja, o caminho que a ficha percorre com o disparo de cada transição. Da mesma forma que no módulo anterior, cada transição da rede tem a sua linha no diagrama Ladder, e somente as transições que são habilitadas durante um ciclo de varredura podem ser disparadas durante esse mesmo ciclo. Para construir esse módulo no diagrama, é necessário utilizar um contato NA para uma transição t_j, bobinas SET para os lugares de saída $O(t_j)$ e bobinas RESET para os lugares de entrada $I(t_j)$.
   A figura 2.23 representa o diagrama Ladder do módulo da dinâmica da rede de Petri do exemplo da máquina de bebidas.

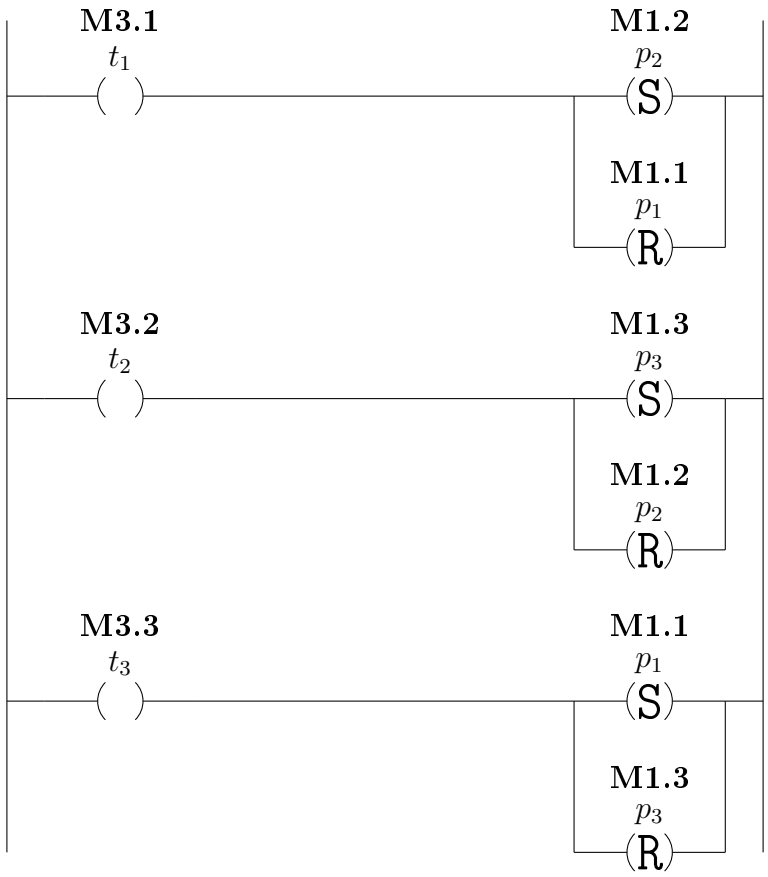


Figura 2.23: Lógica Ladder do módulo da dinâmica da rede de Petri do exemplo da máquina de bebidas.

Na figura 2.23, cada linha é associada ao disparo de uma transição, de forma que na primeira, quando houver o disparo da transição t_1 (*M3.1*), o lugar p_2 deve ganhar uma ficha (*M1.2*) e o lugar p_1 deve perder uma ficha (*M1.1*). As outras duas linhas operam de forma análoga.

4. Módulo de Inicialização

   O quarto módulo é utilizado para gerar a marcação inicial da rede de Petri no diagrama Ladder, ou seja, em uma linha deve haver apenas um contato NF e uma bobina SET para cada lugar que contêm uma ficha na rede de Petri. Além disso, esse módulo também fornece o valor inicial de variáveis da rede utilizadas para contabilizar algo e é executado apenas uma vez pelo CLP, quando for inicializado.
   A figura 2.24 representa o diagrama Ladder do módulo de marcação inicial da rede de Petri da figura 2.11, referente ao exemplo da máquina de bebidas.

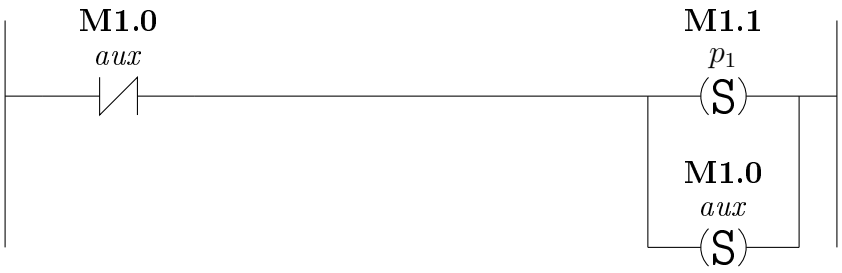


Figura 2.24: Lógica Ladder do módulo de inicialização do exemplo da máquina de bebidas.

No diagrama, a variável de memória *M1.0* é uma variável auxiliar para o módulo de inicialização, utilizada apenas para inserir o valor lógico 1 na variável *M1.1*, essa referente ao lugar p_1.

5. Módulo das Ações

   O quinto e último módulo se propõe a associar cada ação a uma linha do diagrama Ladder. Cada ação é associada a um lugar e é representada por bobinas simples quando forem ações contínuas e bobinas SET/RESET quando forem impulsionais, para iniciar e finalizar a ação, respectivamente.
   A figura 2.25 representa o diagrama Ladder do módulo de ações da rede de Petri do exemplo da máquina de bebidas.

Figura 2.25: Lógica Ladder do módulo de ações da rede de Petri do exemplo da máquina de bebidas.

Na figura 2.25, quando uma ficha chega no lugar p_3 (*M1.3*), a bobina *Q1.0* é acionada, inserindo o valor lógico 1 na variável de saída, que liga o motor da máquina, entregando a bebida selecionada para o cliente.

Para construir o diagrama Ladder inteiro, basta colocar todos os módulos em sequência, na ordem em que foram apresentados.

2.4.1 Sincronização de subsistemas

Para o caso de uma modelagem com dois ou mais subsistemas, é preciso realizar a composição paralela entre eles para encontrar o sistema final. Esse procedimento geralmente aumenta exponencialmente o número de transições, podendo inclusive criar

transições que nunca poderiam ser habilitadas, e consequentemente nunca seriam disparadas.

Como alternativa para evitar fazer a composição paralela dos subsistemas, uma extensão para o método anterior foi proposta em [14], que altera o módulo das condições de disparo de transições e insere um novo, denominado módulo de sincronização. Com esse novo módulo, é possível realizar a sincronização de dois sistemas independentes, desde que tenham eventos em comum, e transformá-los em um único diagrama Ladder.

Apesar do método proposto em [14] possibilitar a sincronização de subsistemas, na prática pode ser difícil sincronizar todos os eventos comuns aos subsistemas, porque a ativação de um mesmo sensor pode representar características distintas do sistema. Quando um processo é construído, os sensores são utilizados para fornecer alguma informação ao sistema ou ao usuário, e não para realizar sincronizações. Uma resolução desse problema é a criação dos chamados eventos virtuais, que representam um evento associado a uma transição específica que se deseja sincronizar.

Neste trabalho, quando for necessário sincronizar uma RPIC N_i com uma N_j, o evento virtual associado a um evento σ utilizado para fazer a sincronização será denotado por σ^{ij}. Para ilustrar o método da conversão, será apresentado um exemplo utilizando dois subsistemas, modelados pelas redes de Petri N_1 e N_2, ilustrados respectivamente pelas figuras 2.26 e 2.27.

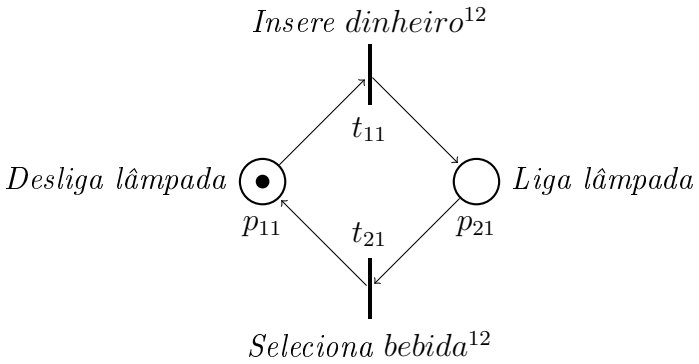


Figura 2.26: Grafo da RPIC N_1.

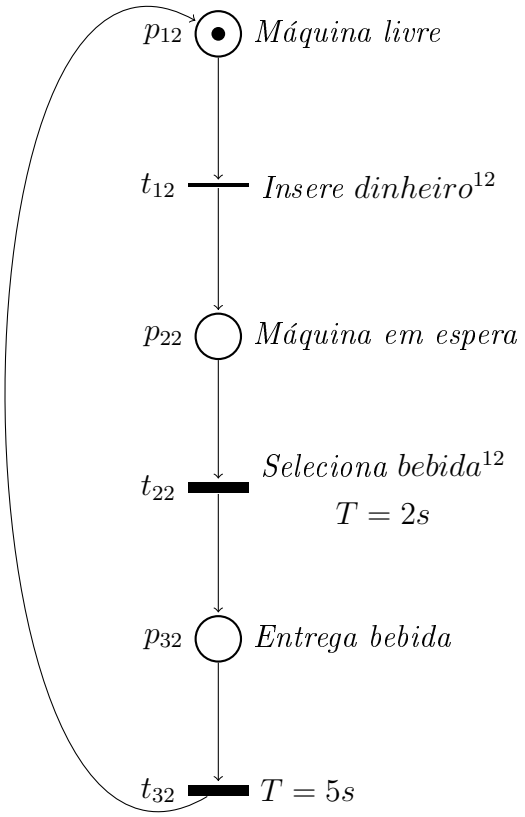


Figura 2.27: Grafo da RPIC N_2.

O sistema escolhido como exemplo é o da máquina de bebidas, conforme mostrado inicialmente na figura 2.11, porém com a adição de uma funcionalidade. Nela, o cliente insere dinheiro na máquina, a qual acende as lâmpadas em volta dos botões de seleção e, quando o cliente aperta o botão selecionando a sua bebida desejada, as lâmpadas se apagam. Para construir esse sistema, foi escolhido separá-lo em dois subsistemas, a RPIC N_1, da figura 2.26, referente ao processo de ligar e desligar as lâmpadas e a RPIC N_2, da figura 2.27, referente ao processo de seleção e entrega de bebidas.

Os subsistemas serão sincronizados duas vezes, quando acontecem os eventos *Insere dinheiro*^{12} e *Seleciona bebida*^{12}. A seguir serão explicados detalhadamente todos os módulos da conversão de RPICs para diagrama Ladder, utilizando o sistema da máquina de bebidas como exemplo.

1. Módulo de Eventos

A figura 2.28 representa o diagrama Ladder do módulo de eventos da rede de Petri do exemplo da máquina de bebidas com iluminação.

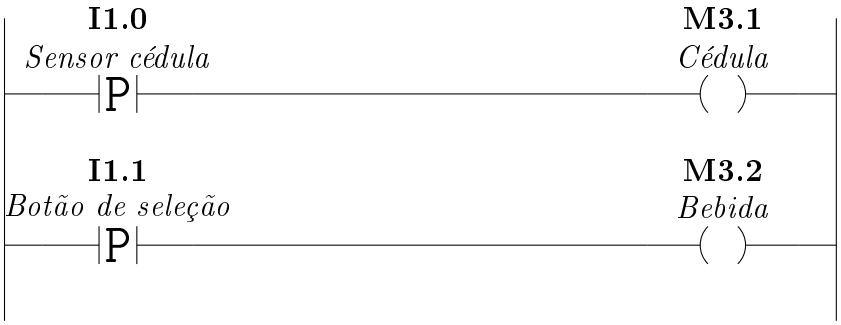


Figura 2.28: Lógica Ladder do módulo de eventos do exemplo da máquina de bebidas com iluminação.

Na primeira linha do diagrama da figura 2.28, a variável de entrada *I1.0* está relacionada ao evento *Insere dinheiro* e na segunda linha, a variável *I1.1* está ligada ao evento *Seleciona bebida*. As variáveis de memória *M3.1* e *M3.2* guardam o estado desses eventos no diagrama Ladder.

2. Módulo de Sincronização

   O módulo de sincronização é utilizado para garantir que duas ou mais redes de Petri distintas que possuem eventos em comum sejam sincronizadas. Dessa forma, as transições que possuem tais eventos só serão disparadas quando estiverem habilitadas em todos os subsistemas.
   A figura 2.29 representa o diagrama Ladder do módulo de sincronização da rede de Petri do exemplo da máquina de bebidas com iluminação.

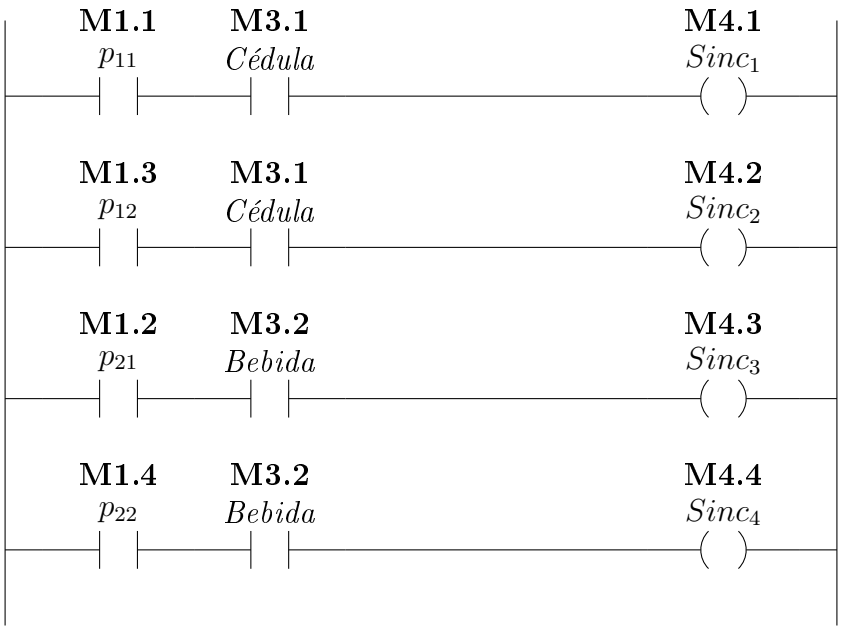


Figura 2.29: Lógica Ladder do módulo de sincronização do exemplo da máquina de bebidas com iluminação.

Na primeira linha da figura 2.29, se houver uma ficha em p_{11} (*M1.1*) e o evento *Insere dinheiro*[12] (*M3.1*) ocorrer, a variável de memória *M4.1* ficará com o valor lógico 1, podendo assim indicar quando a transição t_{11} está habilitada. A segunda linha funciona de forma análoga, pois a variável de memória *M4.2* indica quando a transição t_{12} está habilitada. As linhas 3 e 4 estão relacionadas à sincronização do evento *Seleciona bebida*[12] (*M3.2*), de forma que as variáveis de memória *M4.3* e *M4.4* indicam quando as transições t_{21} e t_{22} estão habilitadas, respectivamente.

3. Módulo das Condições de Disparo de Transições

   A alteração feita no módulo das condições de disparo de transições é a de substituir os eventos externos como condição para o disparo de transições pelos eventos de sincronização, criados no módulo anterior.
   A figura 2.30 representa o diagrama Ladder do módulo das condições de disparo de transições da rede de Petri do exemplo da máquina de bebidas com iluminação.

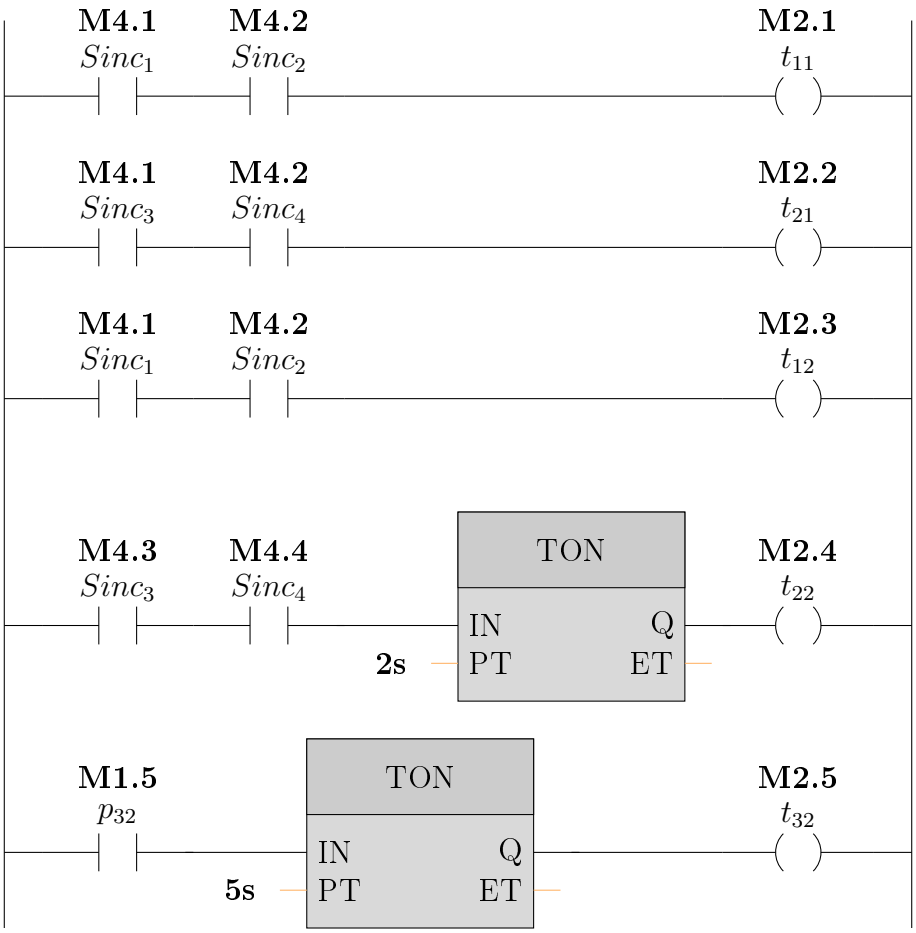


Figura 2.30: Lógica Ladder do módulo das condições de disparo de transições do exemplo da máquina de bebidas com iluminação.

A primeira e a terceira linha do diagrama da figura 2.30 mostra que apenas quando ambas as transições t_{11} (*M4.1*) e t_{12} (*M4.2*) estiverem habilitadas, as mesmas serão disparadas, colocando o valor lógico nas variáveis *M2.1* e *M2.2*. De forma semelhante, as transições t_{21} e t_{22} só serão disparadas quando ambas estiverem habilitadas e após o tempo de 2 segundos, para o caso da transição t_{22}. Como a transição t_{32} é a única que não é sincronizada, ela disparará 5 segundos depois que o lugar p_{32} receber uma ficha.

4. Módulo da Dinâmica da Rede de Petri

   A figura 2.31 representa o diagrama Ladder do módulo da dinâmica da rede de Petri do exemplo da máquina de bebidas com iluminação.

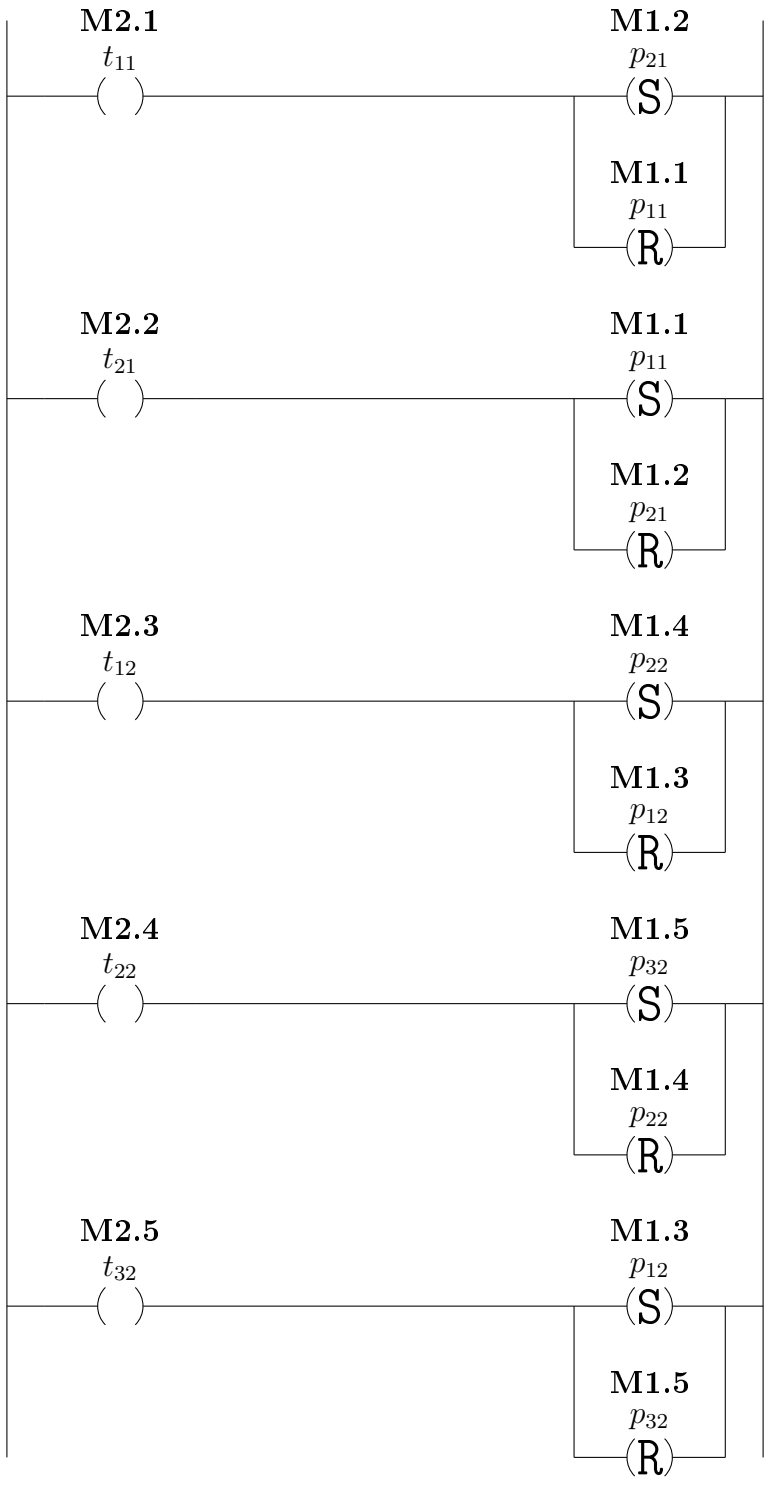


Figura 2.31: Lógica Ladder do módulo da dinâmica da rede de Petri do exemplo da máquina de bebidas com iluminação.

Na figura 2.31, cada linha é associada ao disparo de uma transição, tanto para a rede N_1 quanto para a N_2, de forma que na primeira linha, quando houver o disparo da transição t_{11} (*M2.1*), o lugar p_{21} deve ganhar uma ficha (*M1.2*) e o lugar p_{11} deve perder uma ficha (*M1.1*). As outras linhas operam de forma análoga.

5. Módulo de Inicialização

   A figura 2.32 representa o diagrama Ladder do módulo de marcação inicial da rede de Petri das figuras 2.26 e 2.27, referente ao exemplo da máquina de bebidas com iluminação.

Figura 2.32: Lógica Ladder do módulo de inicialização do exemplo da máquina de bebidas com iluminação.

   No diagrama, a variável de memória *M1.0* é uma variável auxiliar para o módulo de inicialização, utilizada apenas para inserir o valor lógico 1 nas variáveis *M1.1*, referente ao lugar p_{11} e *M1.3*, referente ao lugar p_{12}.

6. Módulo das Ações

   A figura 2.33 representa o diagrama Ladder do módulo de ações da rede de Petri do exemplo da máquina de bebidas com iluminação.

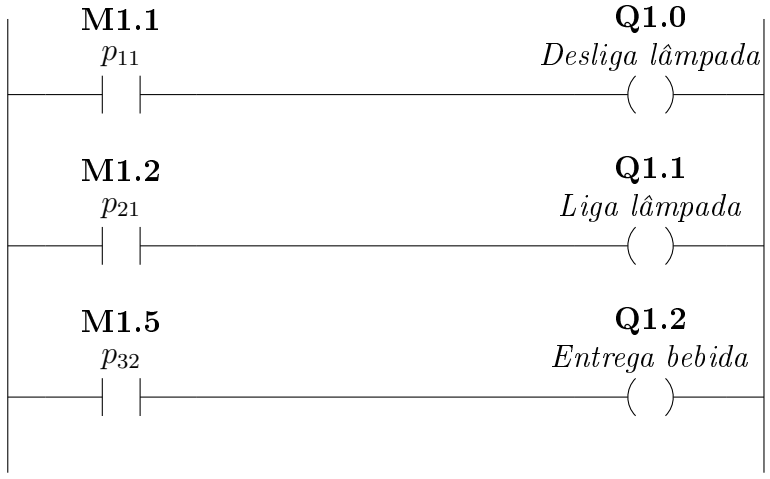


Figura 2.33: Lógica Ladder do módulo de ações da rede de Petri do exemplo da máquina de bebidas com iluminação.

   Na figura 2.33, quando uma ficha chega no lugar p_{11} (*M1.1*), a bobina *Q1.0* é acionada, inserindo o valor lógico 1 na variável de saída, desligando a lâmpada

da máquina. Da mesma forma, quando houver uma ficha no lugar p_{21}, a variável de saída *Q1.1* terá o valor lógico 1, ligando a lâmpada. Para o caso da rede de Petri N_2, se o lugar p_{32} tiver uma ficha, a bobina da variável de saída *Q1.2* é ativada, ligando o motor da máquina, que entrega a bebida selecionada para o cliente.

Capítulo 3

Braço Robótico

Neste capítulo, iremos descrever o braço robótico VP-6242M adquirido pelo Laboratório de Controle e Automação (LCA-COPPE-UFRJ) do fabricante DENSO, que tem 13kg e seis eixos de movimentação, ilustrado pela figura 3.1.

Figura 3.1: Braço robótico VP-6242M de 6 eixos do fabricante DENSO [1].

3.1 Ambiente do braço robótico

Dois referencias são utilizados para controlar o braço: o referencial inercial, representado pelo sistema de coordenadas xyz, e o referencial do pulso, representado pelo sistema de coordenadas XYZ, como pode ser visto na figura 3.2. O primeiro eixo do braço, J1, é utilizado para rotacionar no plano xy, o segundo e terceiro eixos, J2 e J3, respectivamente, são usados para rotacionar no plano xz, o quarto eixo, J4, para rotacionar no plano yz e os dois últimos eixos, J5 e J6, para rotacionar o pulso [2]. A extremidade do pulso oferece um encaixe onde podem ser instaladas

extensões, como uma câmera ou diversos tipos de garras pneumáticas.

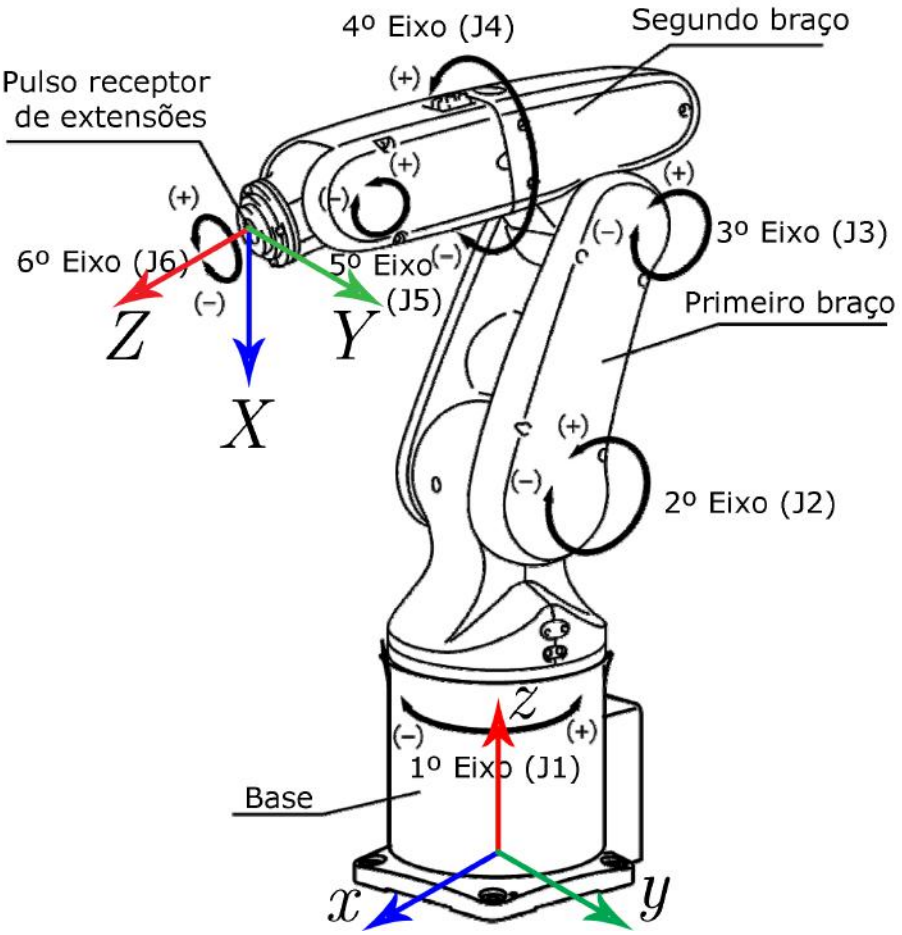


Figura 3.2: Ilustração da localização de todos os eixos de rotação do braço robótico e os dois referenciais [2].

As figuras 3.3 e 3.4 mostram precisamente o espaço de trabalho do braço: todas as regiões que são fisicamente possíveis de serem alcançadas por ele ao realizar as rotações de todos os eixos [1]. Em resumo, o braço tem um raio de atuação máximo de 432 mm, tanto no plano xy quanto no plano xz, podendo também rotacionar até 160° para cada lado e 120° tanto para frente quanto para trás.

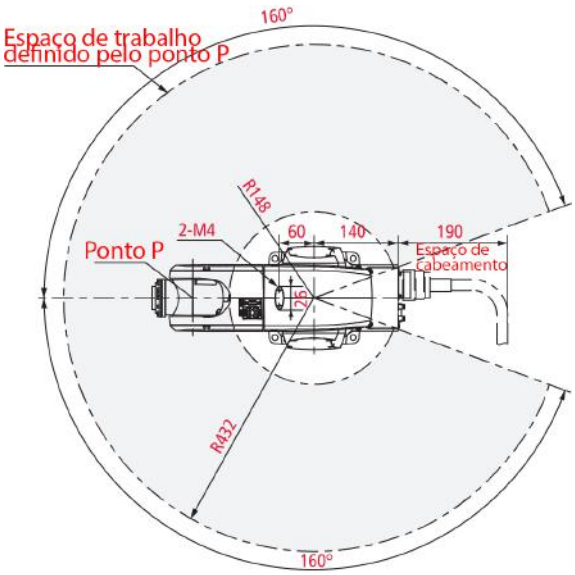


Figura 3.3: Vista superior do espaço de trabalho do braço [1].

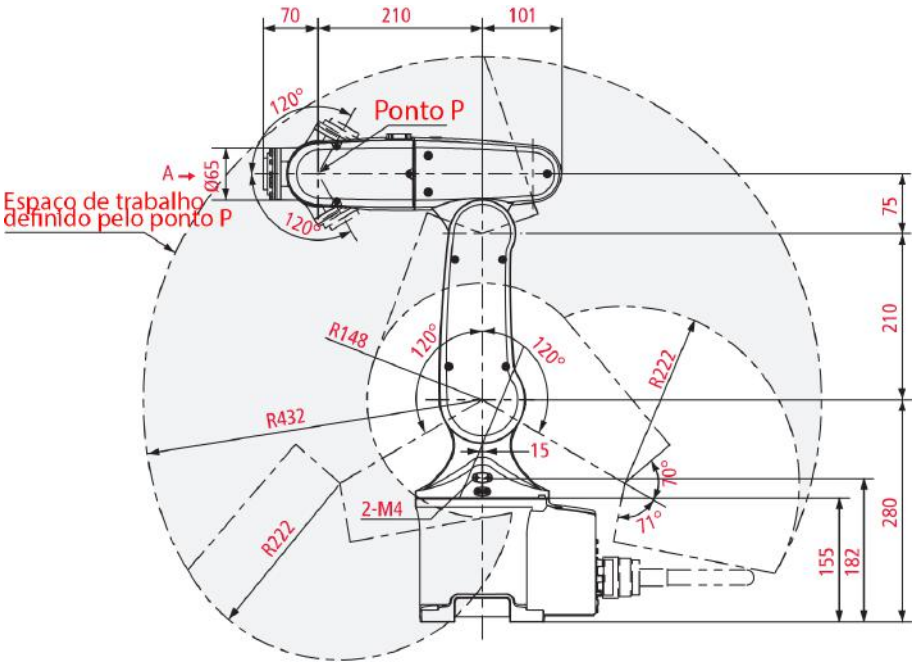


Figura 3.4: Vista lateral do espaço de trabalho do braço [1].

O robô é utilizado em áreas industriais e com a ajuda de uma extensão, pode sustentar cargas, com uma operação padrão de 2kg e um máximo de 2.5kg. Possui um sistema de detecção de colisões, que executa uma parada de emergência quando o braço encontra uma resistência ao seu movimento.

Uma extensão do tipo garra, do fabricante ASS modelo PGR 40, ilustrada pela figura 3.5 foi instalada na extremidade do robô e alimentada com ar comprimido, possibilitando pegar objetos ao abrir e fechá-la.

Para controlar o braço robótico, foi utilizado o controlador do modelo RC7M, do mesmo fabricante, ilustrado na figura 3.6, que tem as seguintes funções:

Figura 3.5: Extensão do tipo garra pneumática instalada no braço robótico.

- Controle de até oito eixos, incluindo a operação de esteiras de transporte e visão robótica.
- Suporte a diversos tipos de comunicação, como Ethernet, USB e RS-232C de fábrica e DeviceNet, Profibus, Parallel I/O e Ethernet/IP como opcionais.

Figura 3.6: Controlador RC7M do fabricante DENSO.

A figura 3.7 mostra um esquema do painel traseiro do controlador e todas as suas entradas. As funções utilizadas nesse trabalho foram:

1. **INPUT AC:** Cabo de força do controlador, ligado na tomada de 220 V.
2. **POWER:** Interruptor liga/desliga.
3. **PENDANT:** Conexão do *teach/mini pendant*
4. **USB/LAN/RS-232C:** Entradas USB, utilizadas para carregar programas através de um *pen drive*; Ethernet e serial padrão RS-232C, utilizadas para comunicar o controlador com o computador.
5. **Baia de extensões:** Três espaços para a instalação de placas de extensão.

6. **HAND I/O:** Entrada tanto da conexão do braço, realizando toda a troca de informações das rotações dos eixos, quanto da válvula pneumática, enviando o sinal de abertura ou fechamento da garra.

7. **SAFETY I/O:** Conexão ao botão de parada de emergência.

8. **MOTOR:** Entrada do cabo de força do braço, conectado no controlador.

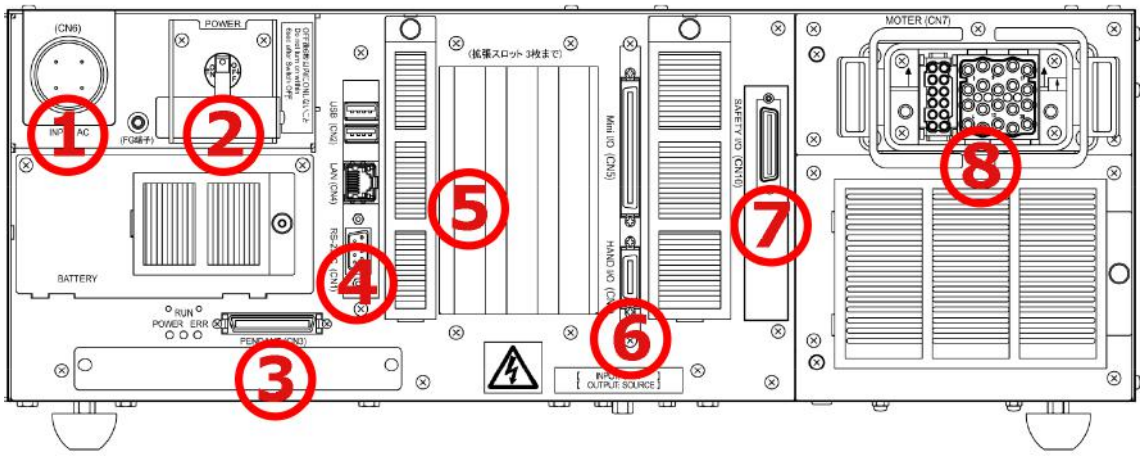


Figura 3.7: Painel traseiro do controlador RC7M [3].

O controlador adquirido pelo laboratório é do tipo PNP (INPUT: SINK, OUTPUT: SOURCE). No controlador foi instalada uma extensão para comunicação PROFIBUS do modelo CIF 50-PB do fabricante Hilscher, uma placa de conexão PCI com um conector tipo DSub fêmea para interface PROFIBUS, um conector tipo DSub macho para interface de diagnóstico e 4 LEDs para indicar o status da comunicação.

3.1.1 Equipamentos do braço

Para realizar a operação manual do braço robótico, é necessário conectar um acessório ao controlador, seja ele o *teach pendant* (figura 3.8) ou o *mini pendant* (figura 3.9). A diferença dos dois é que o *teach pendant* possui um painel de operação *touchscreen*, mostrando nele informações relevantes da função selecionada, e o *mini pendant* é um acessório menor, oferecendo funções semelhantes com uma performance reduzida.

Nesse trabalho, foi utilizado um *mini pendant*, instalado no controlador do laboratório, que funciona como um controle com fio utilizado para operar o braço, podendo ligar e desligar o motor e executar programas escritos no computador. A figura 3.10 mostra o aparelho e os seus botões. O acessório pode operar em três modos: *auto*, *manual* e *teach*, selecionados a partir de uma chave seletora. O modo

Figura 3.8: *Teach pendant* utilizado para operar o braço robótico.

Figura 3.9: *Mini pendant* utilizado para operar o braço robótico.

manual é utilizado para mover os eixos do braço individualmente utilizando comandos do próprio *mini pendant*, alterar a velocidade e executar programas escritos no computador. O modo *auto* é utilizado para indicar que todas as operações do braço devem ser feitas apenas pelo computador. O modo *teach* é utilizado para checar, durante a execução de um programa, se as posições de cada passo da movimentação do robô estão corretas e modificá-las se necessário.

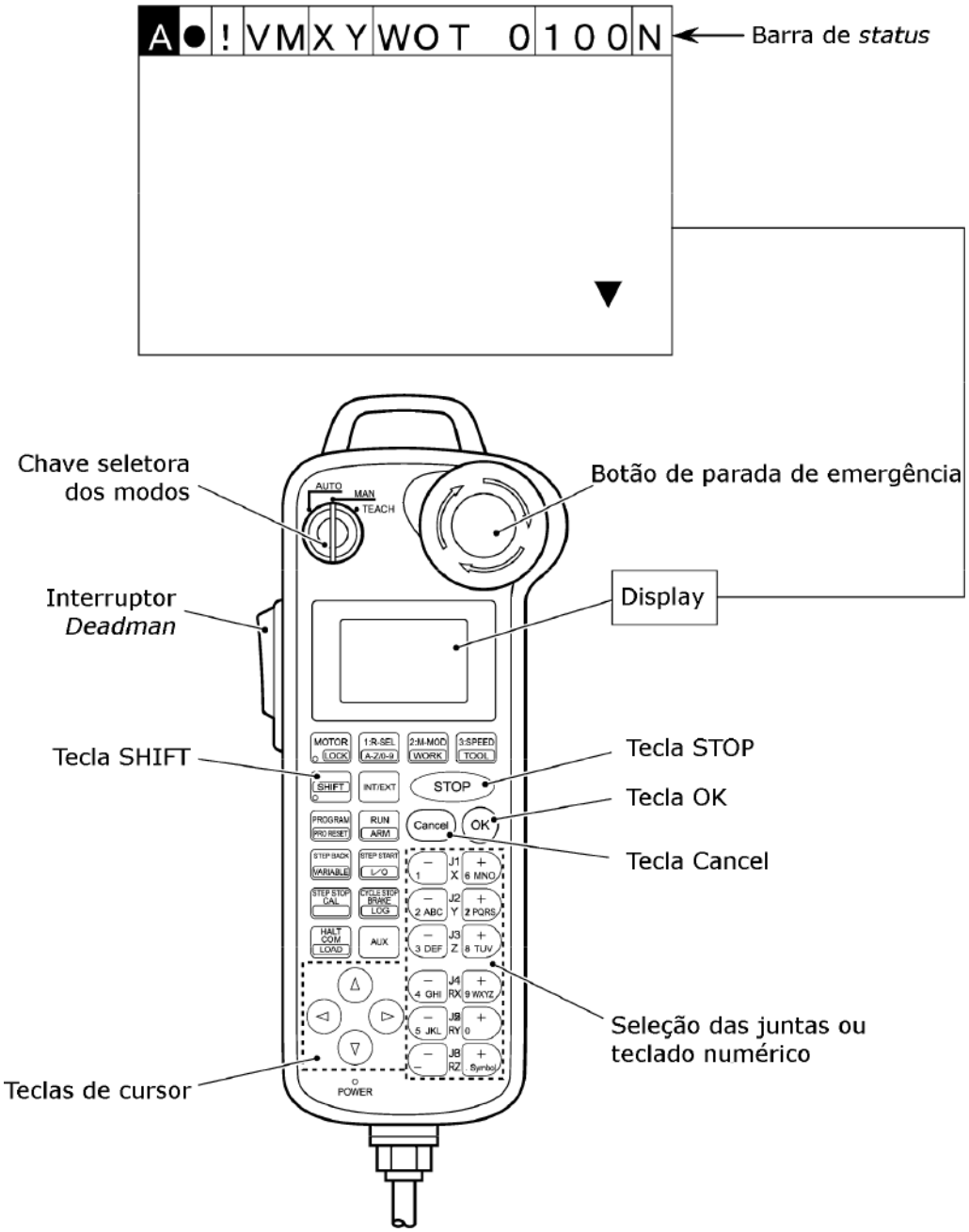


Figura 3.10: Ilustração do *mini pendant* e as suas funções [4].

3.1.2 Operação manual

Ao ligar o controlador, o *mini pendant* mostra o *display* da figura 3.10, onde se encontra permanentemente uma barra de *status*, que mostra sempre o modo de operação, o modelo do robô conectado, a velocidade e outras informações.

A seleção do modo manual habilita o operador a movimentação das juntas pelo *mini pendant*. Para realizar qualquer ação do braço robótico nesse modo, como acionar a garra pneumática ou uma das juntas, é necessário estar com o interruptor *deadman* sempre acionado. Esse interruptor funciona como uma proteção, porque

exige que o operador esteja sempre pressionando-o para que o motor continue ligado.

O próximo passo é ligar o motor acionando o botão [MOTOR], que indica seu estado através de um LED. O modo do movimento pode ser selecionado ao pressionar o botão [2:M-MOD], que retorna uma janela com as opções '*Joint*', '*X-Y*' e '*Tool*'. A primeira opção é usada para acionar cada junta individualmente e a segunda para mover o braço em alguma direção do plano de coordenadas xyz. Para alterar a velocidade de rotação das juntas, é necessário pressionar o botão [3:SPEED], e na janela resultante escolher um número de 0.1 a 100 para a variável '*Speed*', que pode ser alterado tanto com a ajuda dos cursores quanto da digitação do número desejado. Depois de configurar o modo do movimento, para acionar as juntas é necessário apertar o botão '[−]' ou '[+]' relativo a cada uma, que vai de J1 a J6 [4].

Por exemplo, para acionar a junta J2 na velocidade 30, é necessário pressionar o botão [2:M-MOD] e selecionar '*Joint*', depois o botão [3:SPEED] e os botões '3' e '0'. Finalmente, para girar o braço para cima, é necessário pressionar o botão '[−]' e para baixo, o botão '[+]'.

3.2 Computador

Para controlar o braço robótico no modo *auto*, uma das opções é operá-lo através de um computador com o software Wincaps III, disponibilizado pelo mesmo fabricante DENSO, e a comunicação pode ser feita através de uma entrada serial RS-232C ou de uma conexão Ethernet.

3.2.1 Comunicação

Por questões de praticidade, foi escolhido configurar a comunicação utilizando a conexão Ethernet, conectando o controlador ao computador usado para a operação.

No Windows 10, para configurar a conexão Ethernet, é necessário abrir o Painel de Controle, depois ir em 'Rede e Internet' → 'Central de Rede e Compartilhamento' → 'Alterar as configurações do adaptador'. Na janela que foi aberta, é necessário clicar com o botão direito em 'Ethernet' e depois em 'Propriedades'. Na lista que aparece, deve-se selecionar a opção 'Protocolo IP Versão 4 (TCP/IPv4)' e clicar em 'Propriedades'. Ao fazer isso, uma janela, ilustrada pela figura é aberta e na caixa de seleção 'Endereço IP:' deve ser inserido um endereço para o computador, do tipo 192.168.0.XXX, onde os últimos três dígitos são arbitrários e diferentes de 001, que será o endereço do controlador.

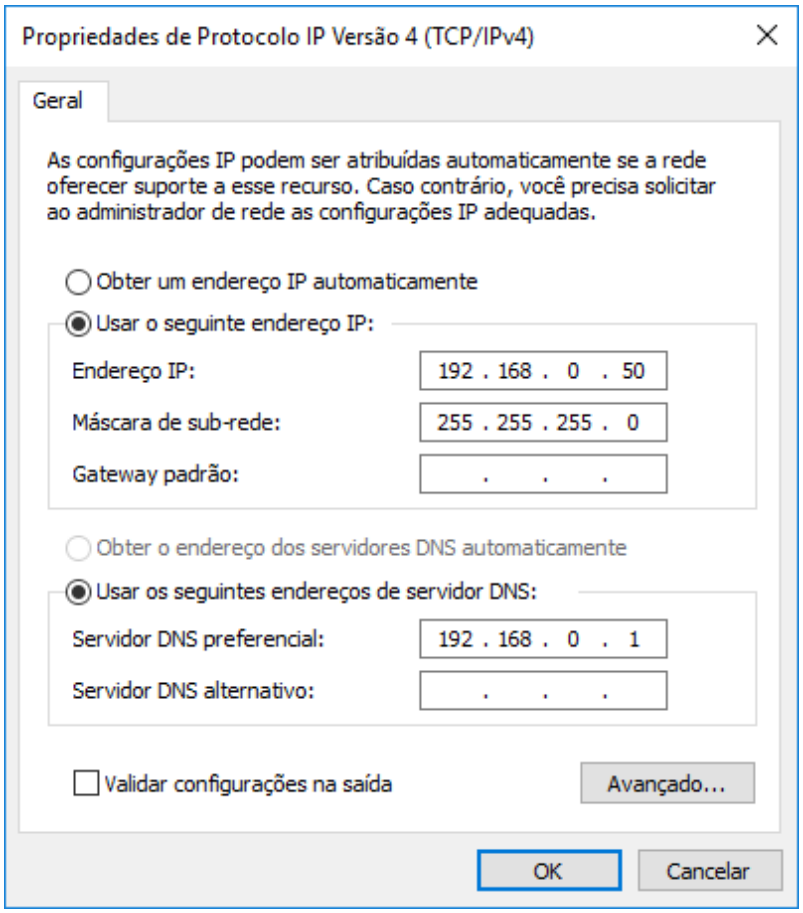


Figura 3.11: Janela de propriedades do adaptador Ethernet e as suas caixas de seleção.

Depois disso, é preciso configurar o endereço de rede do controlador. Isso é feito através da ajuda do *mini pendant* apertando o botão 'COM' em modo manual. Na janela 'COM Setting', deve ser selecionado 'Permit' → 'Ether' → 'IndivIP' para selecionar a opção de um endereço IP individual relacionado ao controlador, como ilustra a figura 3.12. Novamente na janela 'COM Setting', é necessário selecionar 'IPaddress' escolher um endereço IP para o controlador, que será 192.168.0.1. O último passo é selecionar o endereço do computador conectado. Através da janela 'COM Setting', deve-se selecionar 'Connect IP' e colocar o endereço IP do computador previamente escolhido [4].

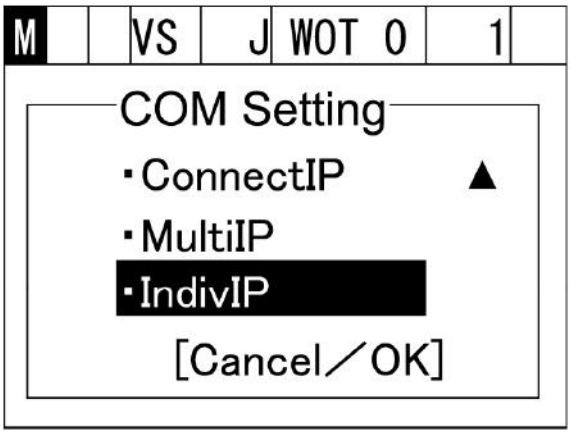


Figura 3.12: Janela de seleção das configurações de IP do *mini pendant* [3].

3.2.2 Programação

A programação de toda a operação do robô foi feita através do software Wincaps III, utilizado para realizar simulações e para executar os programas escritos, todos na linguagem PAC (*Programmable Automation Controller*) [17], voltada especificamente para controlar manipuladores DENSO. Para iniciar a criação de um projeto, é necessário clicar em 'File' → 'New Project'. Daí pode-se optar por duas alternativas: escolher manualmente o modelo do controlador conectado ou a busca automática através da conexão estabelecida anteriormente, como mostra a figura 3.13.

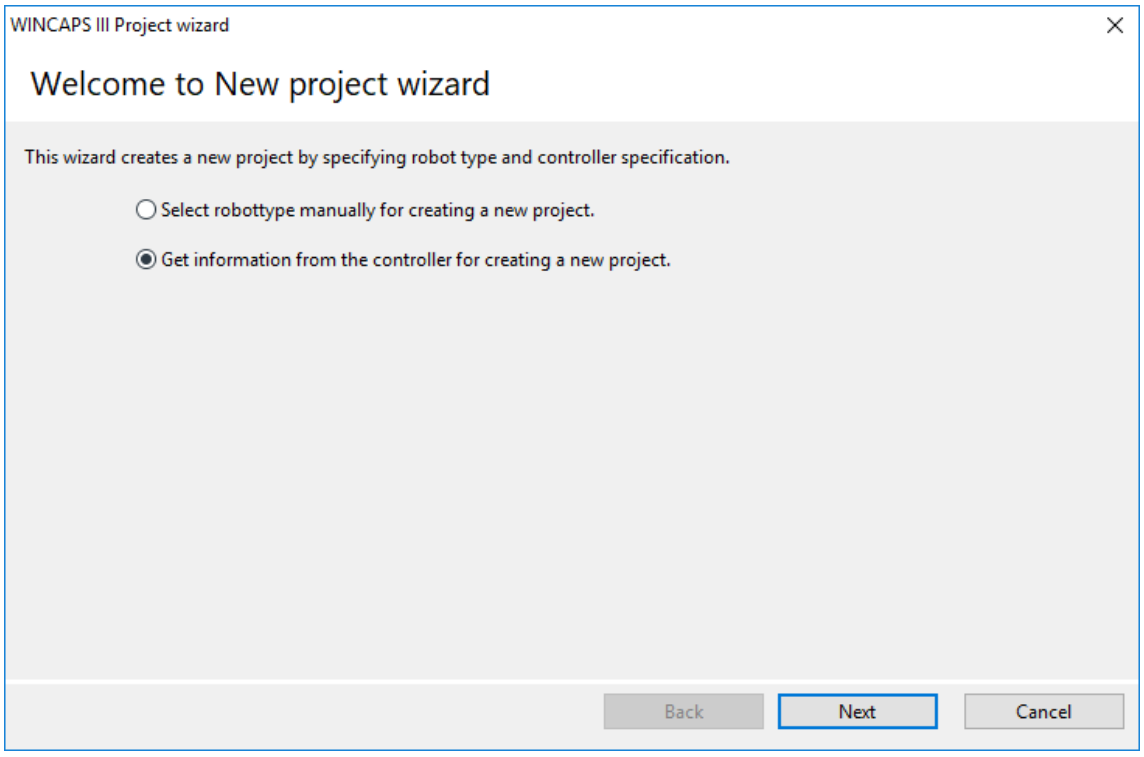


Figura 3.13: Janela do Wincaps ao iniciar a criação de um projeto novo.

Ao selecionar a busca automática de informações do controlador conectado, uma janela para a criação do nome do programa aparece, como mostra a figura 3.14.

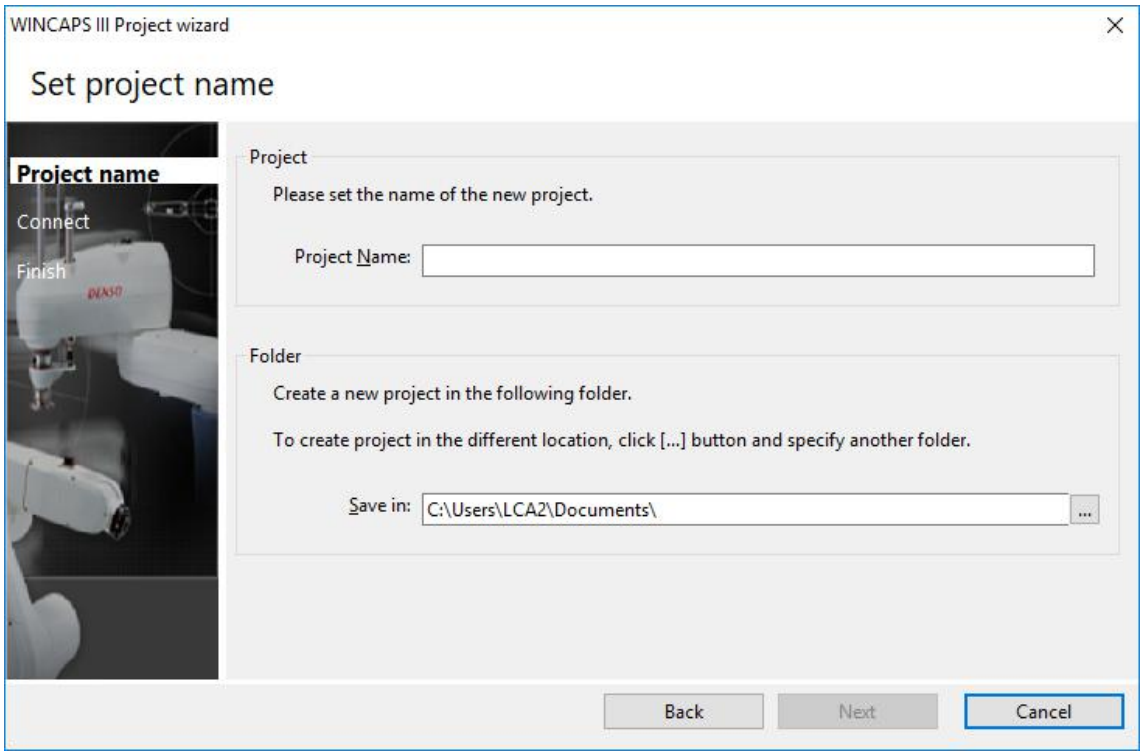


Figura 3.14: Janela do Wincaps de criação de nome do projeto ao selecionar a busca automática de informações do controlador.

Depois de escolher um nome, é necessário colocar o endereço IP do controlador previamente configurado, como mostra a figura 3.15.

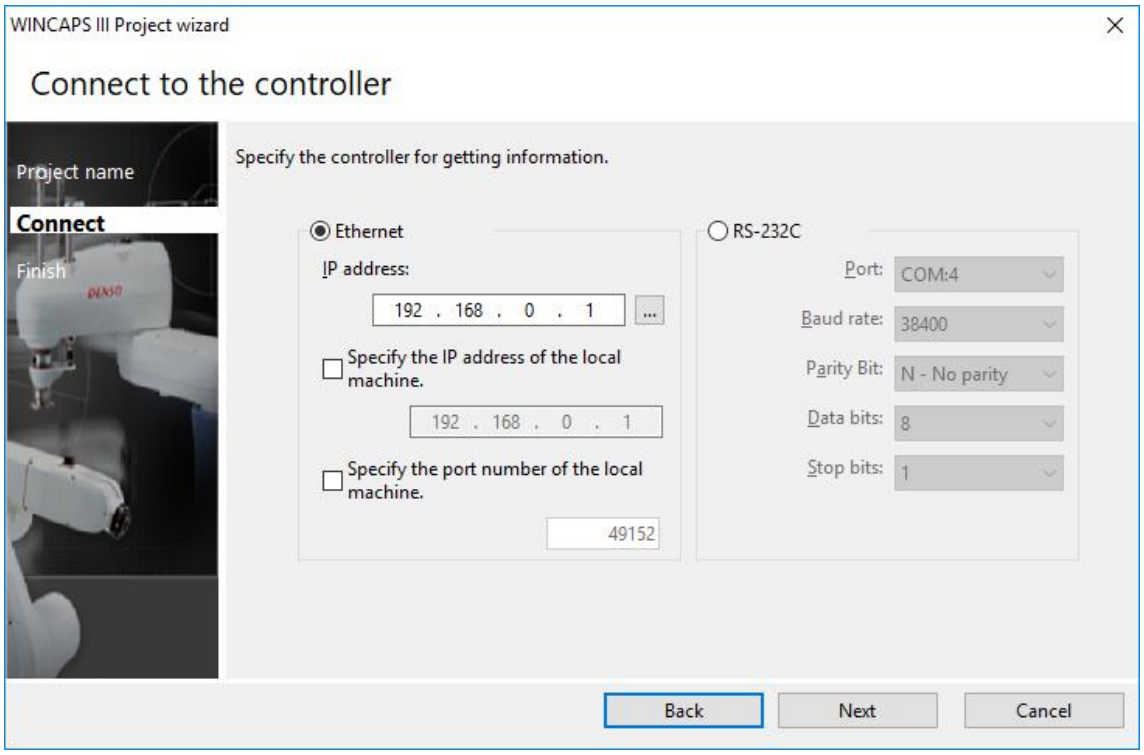


Figura 3.15: Janela do Wincaps de configuração de conexão do controlador.

Ao clicar em 'Next', uma janela com as informações resumidas do controlador é mostrada, como pode-se ver na figura 3.16. Para finalizar a criação do projeto, é necessário clicar em 'Finish'.

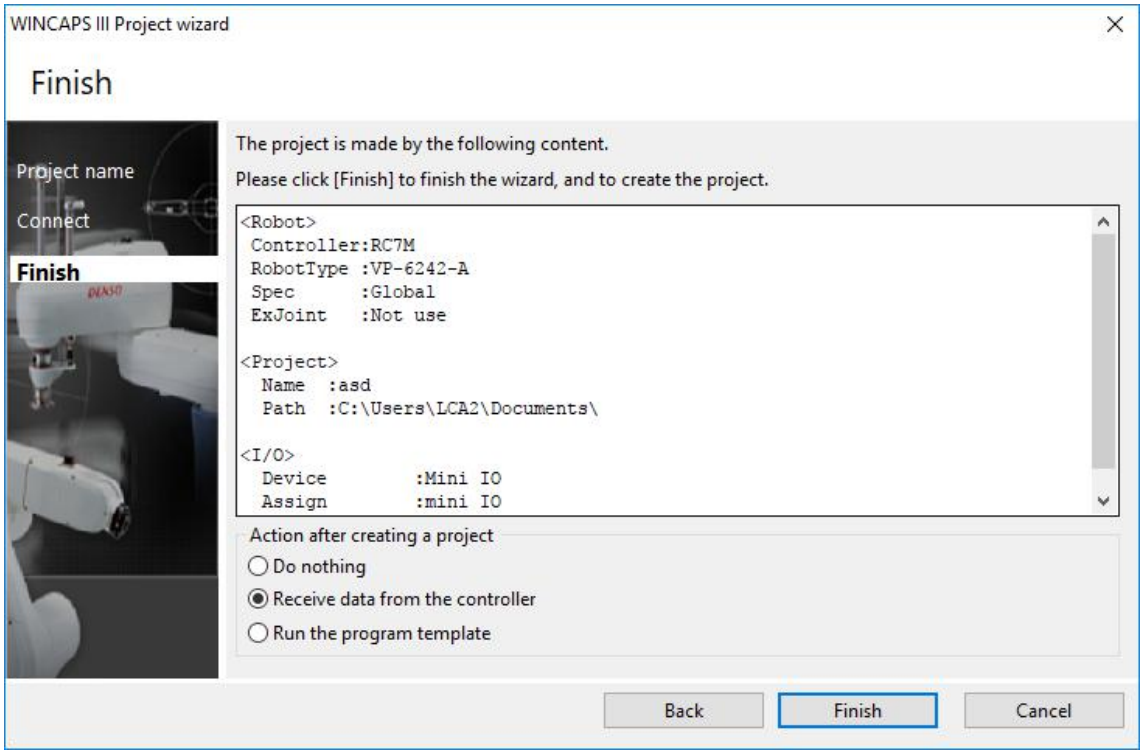


Figura 3.16: Janela do Wincaps com as informações sumarizadas da configuração.

Com o novo projeto aberto, para criar um programa é preciso clicar em 'Project' → 'Add Program'. Na caixa de criação do novo programa, deve ser selecionado o tipo 'Program (*.PAC)' e o seu nome, como mostra a figura 3.17. Depois de clicar em OK, uma nova janela se abrirá com uma base para o *script*.

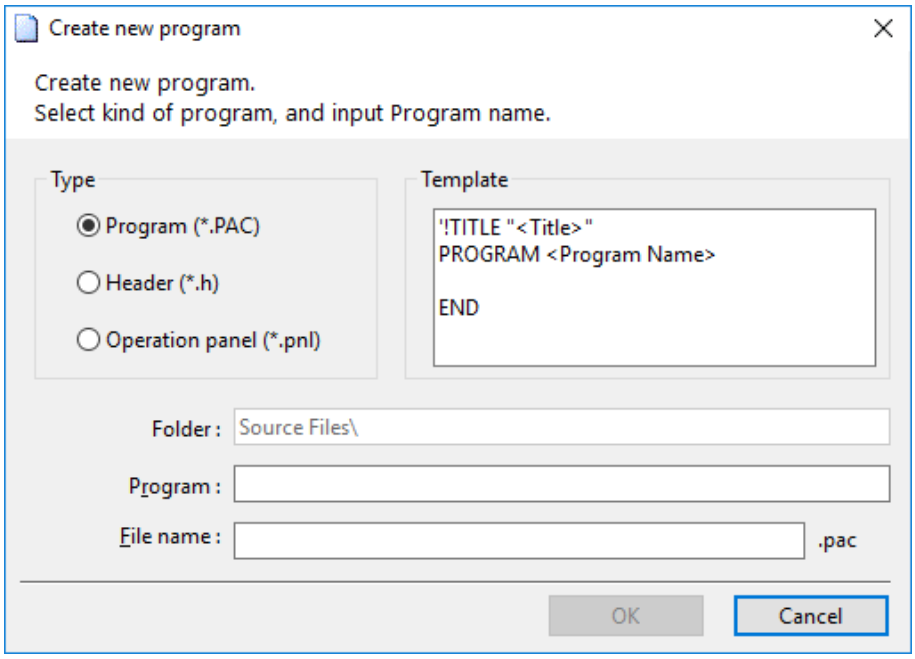


Figura 3.17: Janela do Wincaps ao iniciar a criação de um programa.

Algumas funções básicas da linguagem PAC utilizadas são [18]:

- PROGRAM: Declara o nome de um programa.
- TAKEARM: Seleciona e habilita um grupo de juntas.
- APPROACH: Executa um movimento absoluto até chegar em uma distância <Comprimento> em milímetros da posição <Posição base> na direção $\vec{Z}$.
  - Sintaxe: APPROACH <Método de interpolação>, <Posição base>, @<Deslocamento do caminho> <Comprimento>
- MOVE: Move o pulso do manipulador para as coordenadas especificadas pelo parâmetro <Posição de destino>.
  - Sintaxe: MOVE <Método de interpolação>, @<Deslocamento do caminho> <Posição de destino>
- DEPART: Executa um movimento relativo para se distanciar <Comprimento> milímetros para longe da posição atual na direção $\vec{Z}$.
  - Sintaxe: DEPART <Método de interpolação>, <Posição base>, @<Deslocamento do caminho> <Comprimento>

- SET: Estabelece o valor da porta no endereço I/O selecionado para ligado (nível lógico 1).
- RESET: Estabelece o valor da porta no endereço I/O selecionado para desligado (nível lógico 0).
- #DEFINE: Define uma função ou substitui o valor de uma constante.

Observação 1 *A garra pneumática está associada à porta de endereço I/O número 64. Para fechar a garra, o valor da porta deve ser igual ao nível lógico 1, e para abrir, igual ao nível lógico 0.*

Para realizar qualquer movimentação absoluta do braço, é necessário escolher as posições de origem e de destino como pontos no espaço de coordenadas xyz em relação a base, e para isso, é preciso salvar em variáveis os valores dessas posições. As coordenadas das posições podem ser simplesmente escolhidas e inseridas como variáveis, mas uma forma de facilitar a visualização desses pontos no espaço é criar um objeto no Wincaps III e salvar em uma variável a posição desse objeto, sendo possível avaliar a viabilidade do movimento e da tarefa projetada.

Para começar, é necessário criar um objeto no Wincaps, e para isso primeiro abrimos uma janela de visualização 3D do braço ao clicar em 'View' → 'Arm View'. O próximo passo é abrir o painel lateral esquerdo 'Model Tree' ao clicar em 'View' → 'Arm Modeling', mostrando informações sobre os objetos 3D criados. Para criar um objeto, é só clicar sobre o formato desejado e o modelo aparecerá na janela 'Arm 3D View', como mostra a figura 3.18.

No painel 'Model Tree', é possível selecionar o objeto criado, cujo nome é "Box", no exemplo da figura 3.18, e alterar a sua posição no espaço e a sua rotação na seção 'Relative Position' e as suas dimensões na seção 'Size (mm)'.

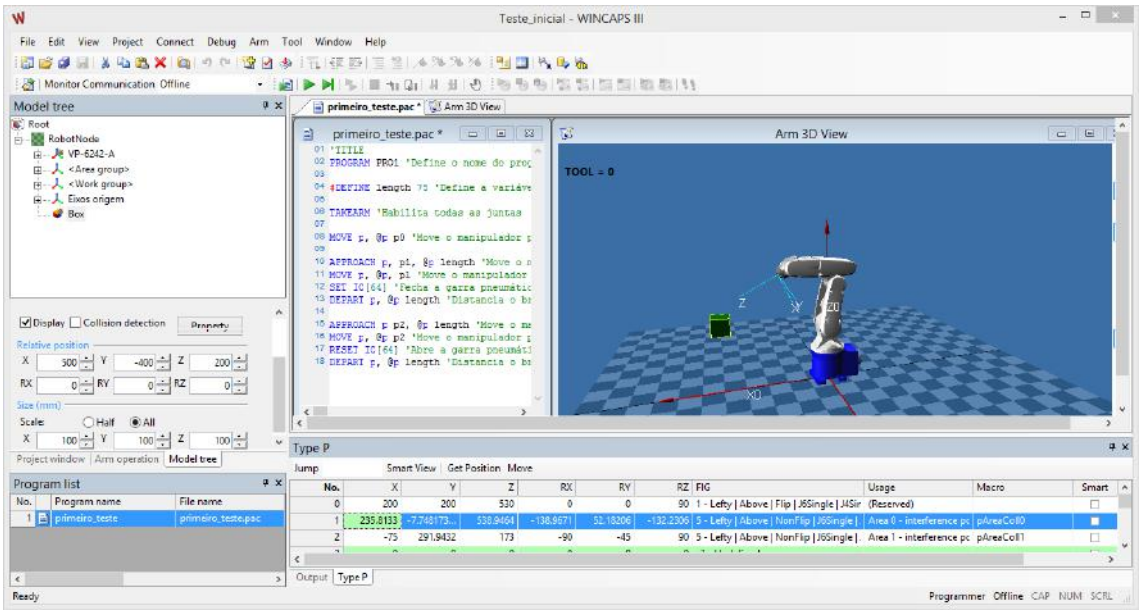


Figura 3.18: Janela principal do Wincaps mostrando o modelo 3D do braço, o programa escrito e outras informações adicionais.

Para abrir a tabela com as variáveis de posição, é necessário clicar em 'View' → 'Variable View' → 'Type P', que mostra todas as posições criadas, como pode ser visto pela figura 3.19. Essa tabela mostra a distância da origem até a extremidade do braço em relação aos três eixos (x, y e z), as rotações em relação a cada eixo (RX, RY e RZ), entre outras informações.

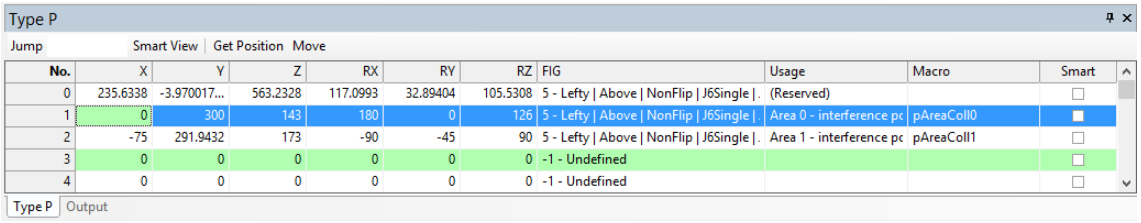


Figura 3.19: Janela 'Type P' do Wincaps mostrando a tabela das variáveis de posição criadas.

Com o objeto configurado para a posição desejada, o próximo passo é guiar o modelo do robô até ele, clicando no botão '3D view teach', que está destacado na figura 3.20. Com esse modo selecionado, ao clicar sobre um ponto do objeto, o robô tenta se deslocar até aquele ponto. Caso seja fisicamente impossível alcançar aquela posição, o software retorna a mensagem "*Could not move this point*" na janela 'Output'. Caso contrário, a mensagem retornada é a posição escolhida e o robô se move até aquele ponto.

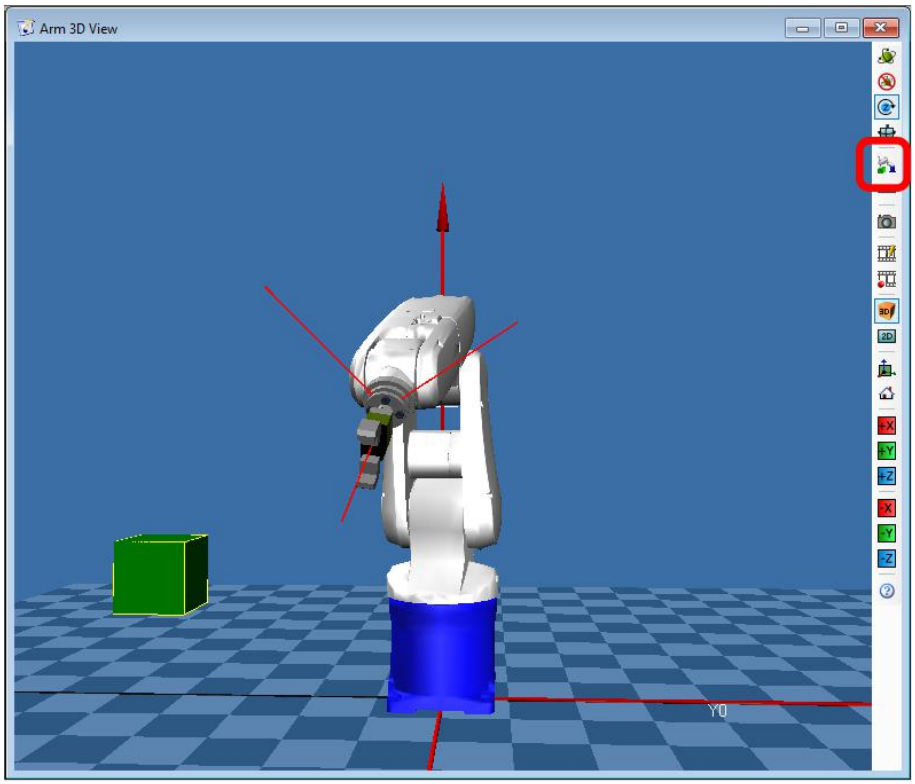


Figura 3.20: Janela 'Arm 3D View' do Wincaps mostrando o braço e os objetos criados nesse ambiente.

Para colocar as informações de posição em variáveis, é necessário clicar sobre o ponto da superfície do objeto com o modo '3D view teach' habilitado, selecionar a janela 'Type P', escolher um número de variável (0, 1, 2, 3, ...) e clicar sobre esse número na barra esquerda. A linha selecionada ficará destacada, como mostra a figura 3.19. Para pegar a posição atual do braço e inserir nessa variável, é necessário clicar no botão 'Get Position', e se a intenção for fazer o processo inverso, ou seja, mover o braço até uma posição pré-determinada, é necessário selecioná-la e clicar no botão 'Move'.

Como exemplo foram selecionadas três posições arbitrárias para serem as variáveis *p0*, *p1* e *p2*, que foram as seguintes:

Tabela 3.1: Tabela com os valores de três posições arbitrariamente escolhidas.

No.	x	y	z
p0	200	0	550
p1	305	-51	114
p2	-12	-319	379

Após a configuração de comunicação, inicialização do programa e da criação das variáveis de posição, é possível começar a escrever o programa desejado. Um

exemplo de *script* para a movimentação do braço de um ponto a outro é mostrado a seguir.

```
1   'TITLE
2   PROGRAM PRO1 'Define o nome do programa como PRO1.
3
4   #DEFINE length 75 'Define a variável "length" com o valor 75
        (milímetros).
5
6   TAKEARM 'Habilita todas as juntas.
7
8   MOVE p, @p p0 'Move o manipulador para a posição inicial p0 j
        á definida.
9
10  APPROACH p, p1, @p length 'Move o manipulador "length"
        unidades da posição p1 na direção z.
11  MOVE p, @p, p1 'Move o manipulador para a posição p1.
12  SET IO[64] 'Fecha a garra pneumática.
13  DEPART p, @p length 'Distancia o braço "length" unidades da
        posição atual na direção z.
14
15  APPROACH p p2, @p length 'Move o manipulador "length"
        unidades da posição p2 na direção z.
16  MOVE p, @p p2 'Move o manipulador para a posição p2.
17  RESET IO[64] 'Abre a garra pneumática.
18  DEPART p, @p length 'Distancia o braço "length" unidades da
        posição atual na direção z.
```

Caso alguma das posições definidas para a movimentação do robô seja inalcançável, o braço executará o programa normalmente até chegar na instrução dessa posição e parará antes de tentar executá-la, mandando uma mensagem de aviso para o computador que aquela posição é ilegal.

Capítulo 4

Sistema de desmontagem

Neste capítulo serão descritos os procedimentos práticos que culminaram no desenvolvimento de um sistema de desmontagem de blocos utilizando o braço robótico, que foi integrado a uma planta mecatrônica para completar o ciclo de manufatura de um produto.

4.1 Planta mecatrônica

Uma planta mecatrônica educacional, ilustrada pela figura 4.1, do fabricante Christiani Sharpline foi a utilizada para realizar a manufatura de cubos, que são montados utilizando uma peça metálica e outra plástica, essa podendo ser preta ou branca. A figura 4.2 mostra todas as peças separadamente.

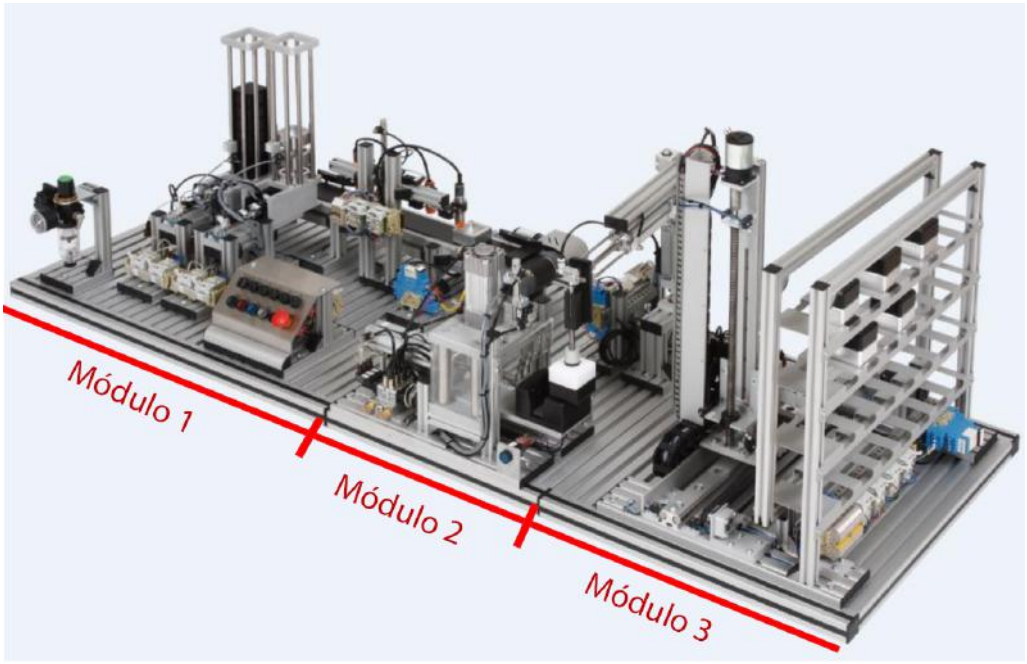


Figura 4.1: Planta mecatrônica educacional utilizada no laboratório.

Figura 4.2: Peças de plástico e de metal utilizadas na planta mecatrônica.

A planta é controlada por um CLP Siemens S7-300 e pode ser dividida em três módulos [8, 9]:

- Seleção: Cada peça do bloco é estocada em duas torres, a primeira contendo apenas peças plásticas e a segunda apenas peças metálicas. Ao retirar uma peça da torre através de um pistão, a peça segue por uma esteira e passa por diversos sensores utilizados para identificar a orientação da peça, o material e a cor.

- Montagem: Ao chegar no final da esteira, a peça é retirada por um braço pneumático giratório e colocada na base de uma prensa, onde espera até que outra peça seja colocada por cima e depois prensadas para realizar o encaixe do cubo.

- Armazenamento: Quando pronto, o bloco é retirado novamente pelo braço giratório e colocado na plataforma de um elevador, que se move tanto vertical quanto horizontalmente, posicionando o bloco no espaço que foi determinado como local para ser armazenado, finalizando o processo de manufatura do bloco.

O processo de montagem realizado em [8] foi filmado e está disponível em [19].

4.2 Processo de desmontagem

A desmontagem funciona como um processo inverso da manufatura. O sistema de desmontagem projetado é acoplado ao sistema de manufatura e consiste de uma nova esteira, um cesto e um braço robótico, como pode ser visto na figura 4.3. Como a

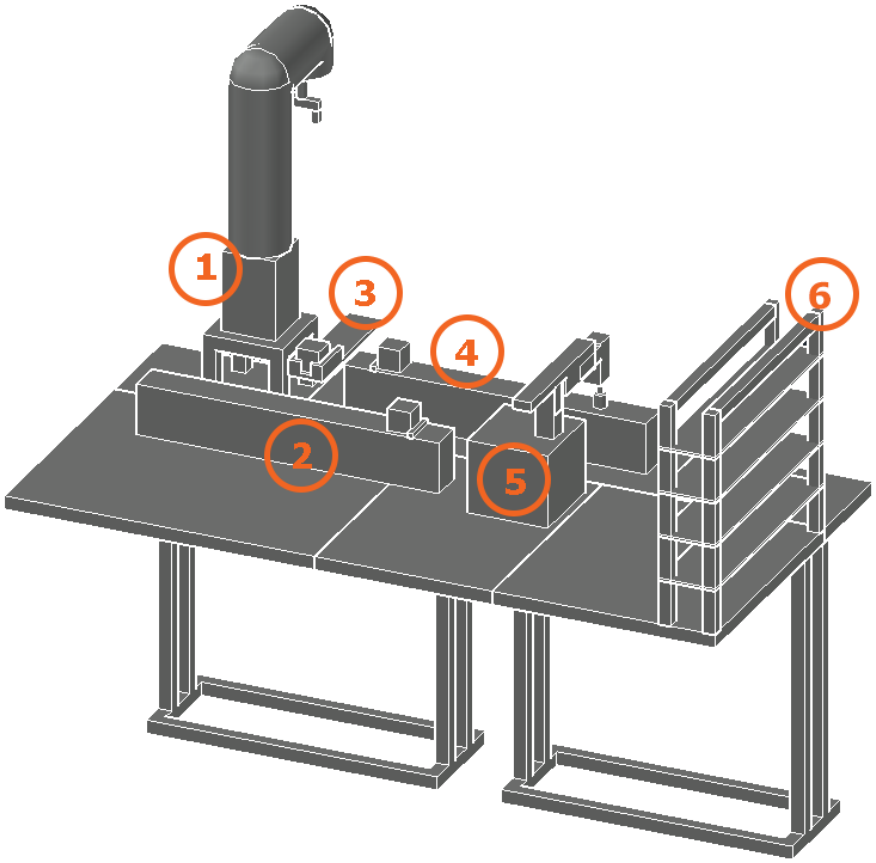


Figura 4.3: Sistema de desmontagem.

nova esteira é similar à do processo de desmontagem, será feita a diferenciação a partir de agora como esteira de montagem e esteira de desmontagem.

Os elementos da figura 4.3 são:

1. Braço robótico
2. Esteira de montagem
3. Cesto
4. Esteira de desmontagem
5. Braço pneumático
6. Estante

O braço robótico utilizado foi descrito no capítulo 3. A esteira de desmontagem e o cesto serão apresentados a seguir.

4.2.1 Esteira de desmontagem

Para iniciar o processo de desmontagem, o bloco montado deve ser levado pelo braço pneumático até a esteira de desmontagem.

A esteira de desmontagem foi projetada para levar o bloco montado até o braço robótico. Como foi mostrado no capítulo 3, o braço robótico tem um alcance de 432mm, e já que a distância entre o braço pneumático e a base do braço robótico é de aproximadamente 750mm, o mesmo não consegue alcançar o bloco sozinho, então a esteira é suficientemente longa a ponto de conseguir colocar o bloco em uma posição dentro do espaço de trabalho do braço robótico.

A esteira comprada é do modelo *Conveyor Unit long* 64325 do fabricante Christiani Sharpline, ilustrada pela figura 4.4. Essa esteira utilizada na desmontagem tem 680mm de extensão, conta com um motor DC de 24V, utilizado para girar a correia, dois relés e um sensor óptico, instalado no fim do curso da esteira, para detectar a presença de um bloco.

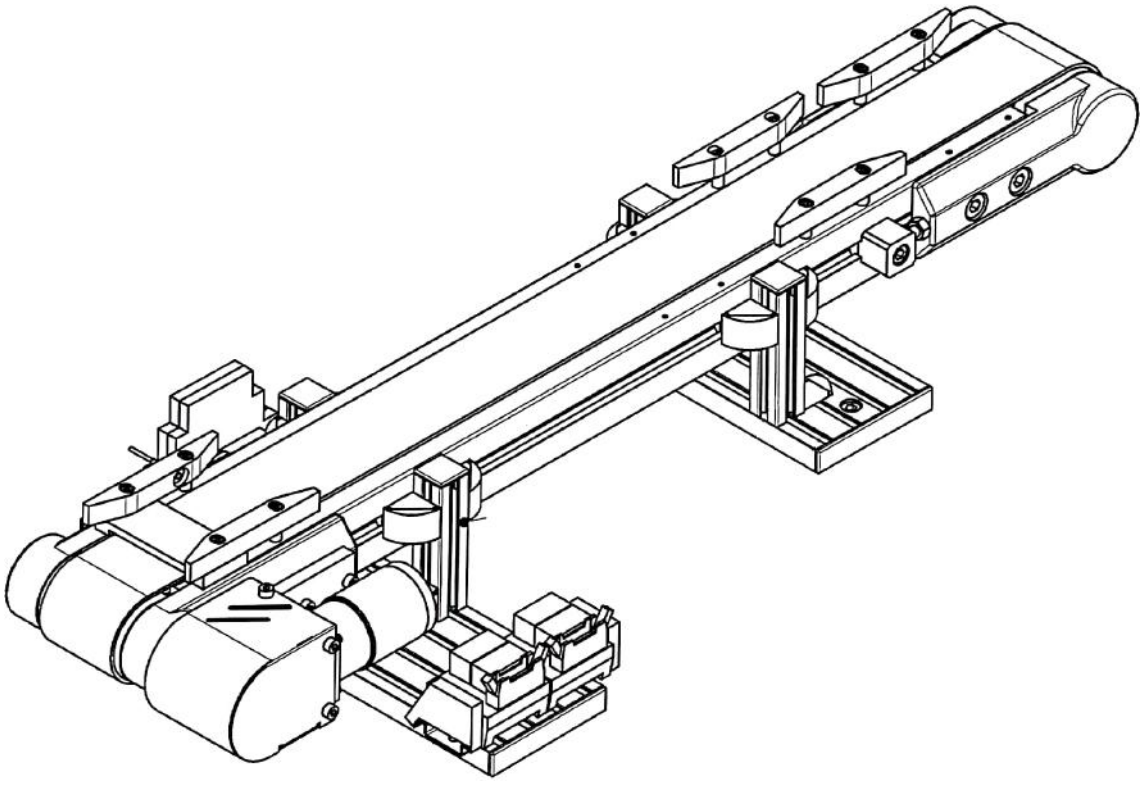


Figura 4.4: Esteira *Conveyor Unit long* 64325 do fabricante Christiani Sharpline.

Quando instalada, a esteira pode então levar o bloco montado do braço pneumático ao braço robótico.

4.2.2 Cesto

A solução encontrada para auxiliar o braço robótico a separar a peça metálica da peça plástica foi criar um cesto, que consegue prender a peça inferior do bloco, possibilitando que a garra pneumática do braço robótico possa retirar a peça superior como mostrado na figura 4.5. O cesto foi projetado como uma estrutura de alumínio para prender o bloco montado. Para tanto, foi necessário instalar três elementos no cesto:

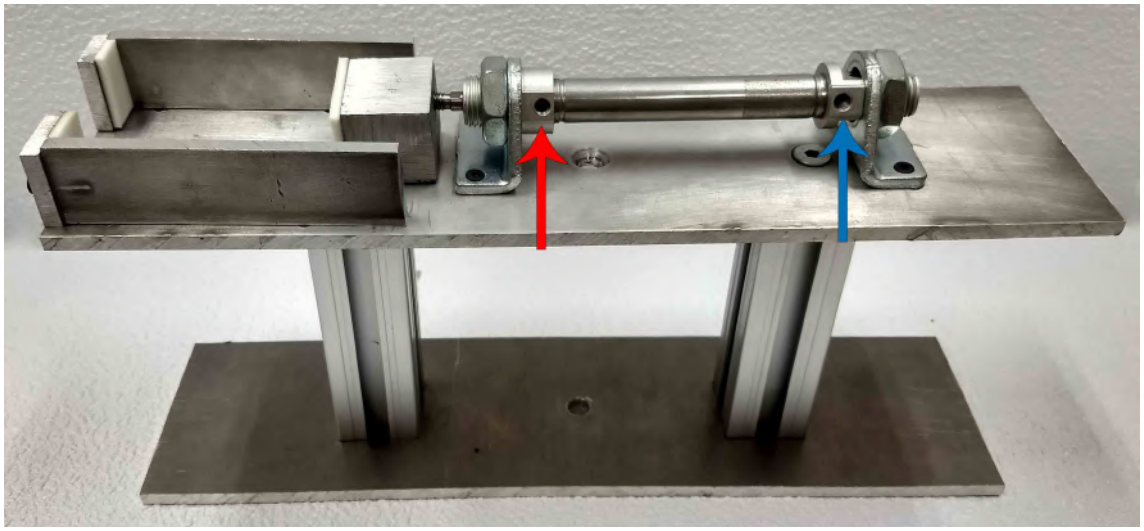


Figura 4.5: Sistema construído para prender o bloco montado.

- Um pistão, formado por um cilindro pneumático do fabricante FESTO tipo DSNU-12-50-P-A e uma cabeça em formato cúbico de alumínio, rosqueada na ponta do cilindro para criar uma área de contato entre o pistão e o cubo. O pistão cilíndrico foi escolhido por ter um curso de 50mm e um raio de 12mm, dimensões necessárias para conseguir prender a parte inferior do bloco quando a garra puxar a parte superior.

- Uma válvula direcional 24V simples solenoide 5/2 do fabricante AVENTICS, modelo 5777055302, usada para controlar o fluxo de ar comprimido no pistão. A figura 4.6 ilustra o modelo da válvula adquirida, que foi escolhida por ser similar à utilizada para a operação da garra pneumática do braço robótico e por operarem a uma pressão similar.

Figura 4.6: Válvula pneumática utilizada para alimentar o pistão.

Esse tipo de válvula funciona de acordo com o esquema da figura 4.7. São 5 conexões e 2 alternativas de saída. Quando a válvula não está energizada, uma mola empurra o cilindro interno, e a entrada de ar pressurizado pela conexão 1 sai pela conexão 4. Quando a válvula é energizada, a mola é comprimida, direcionando o fluxo de ar pressurizado da conexão 1 para a 2.

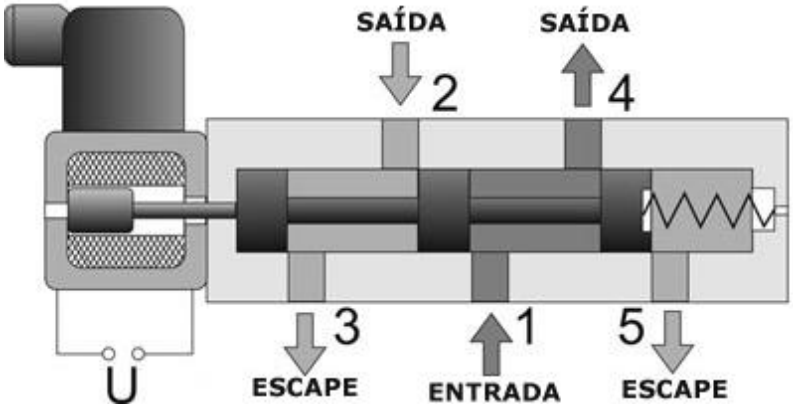


Figura 4.7: Diagrama de funcionamento de uma válvula pneumática 5/2 simples solenoide.

Para ligar a válvula ao pistão é necessário conectar a entrada da válvula no fluxo de ar comprimido, a saída 4 da válvula na entrada do pistão destacada pela seta vermelha na figura 4.5 e a saída 2 da válvula na entrada do pistão destacada pela seta azul. Dessa forma, é possível alternar entre a extensão (ao injetar ar comprimido pela entrada da seta azul) e a retração (ao injetar ar comprimido pela entrada da seta vermelha) do pistão através de um comando elétrico de 24V.

- Dois sensores magnéticos, do fabricante Bosch, tipo 0-830-100-476, usados para detectar se o pistão está estendido ou recuado. A figura 4.8 ilustra o tipo de

sensor utilizado, o qual foi escolhido por ser pequeno o suficiente para poder ser montado à estrutura do pistão e por ser similar aos já utilizados na planta mecatrônica.

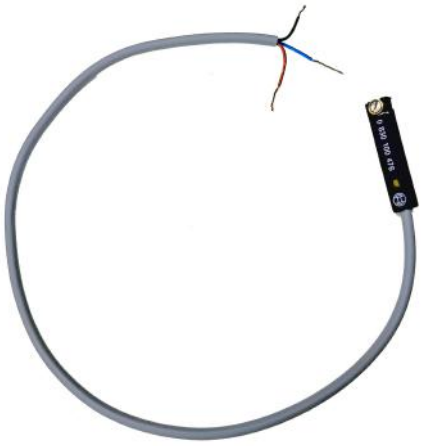


Figura 4.8: Sensor magnético Bosch 0-830-100-476.

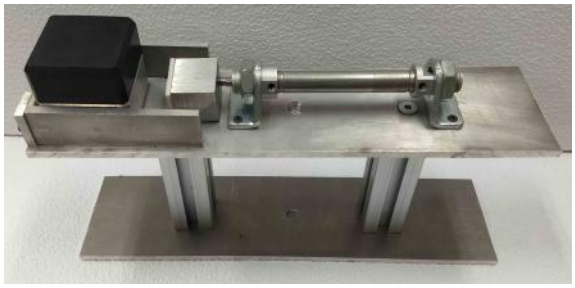
Os sensores comprados têm três fios, e para utilizá-los corretamente, é necessário ligar o fio marrom no positivo, com 10V a 30V e o fio azul no negativo. A tensão de saída entre o fio preto e o azul é que indica se o sensor está detectando o campo magnético ou não. Se o sensor não detectar nada, a tensão de saída será um pouco menor que a de entrada e se o sensor ativar, um LED na sua lateral se acenderá e a tensão de saída será igual a de entrada.

4.2.3 Funcionamento do sistema de desmontagem

Quando solicitada a desmontagem, os blocos são retirados da estante (elemento 6 da figura 4.3) pelo elevador e depois levados para a esteira pelo braço pneumático (elemento 5 da figura 4.3), realizando o processo inverso do armazenamento (módulo 3).

Como hipótese, considera-se que o bloco sempre chega na esteira com a peça metálica embaixo e a plástica em cima. A esteira transporta o bloco do braço pneumático para um ponto próximo do cesto e após a ativação do sensor de presença no final da esteira, ela é desligada. Assim que o bloco chegar no fim da esteira, o braço robótico inicia efetivamente o processo de desmontagem. Primeiramente, o bloco é retirado da esteira pelo braço e colocado no cesto (figura 4.9a). A partir daí, o pistão do cesto é acionado (figura 4.9b), travando apenas a peça inferior do cubo. O braço se movimenta até o bloco e a garra pneumática se prende à peça superior, garantindo que ao realizar um movimento de subida, o braço pegue apenas a peça plástica (figura 4.9c). O próximo passo é mover o braço com a peça até a torre de armazenamento (módulo 1). Esse transporte é dividido em duas etapas

para evitar a colisão entre o braço e as torres: (i) o braço é movido até um ponto intermediário ligeiramente acima e ao lado da torre de armazenamento de peças plásticas; (ii) depois, é necessário mover o braço até um ponto exatamente acima da torre e soltar a peça para que seja guardada corretamente. O pistão é retraído, liberando a peça metálica (figura 4.9d) e possibilitando que o braço possa pegá-la livremente e colocá-la na torre de peças metálicas da mesma forma que fez com a peça plástica.



(a) Cesto na posição 1: Bloco inteiro com pistão recuado.

(b) Cesto na posição 2: Bloco inteiro com pistão estendido.

(c) Cesto na posição 3: Apenas peça inferior com pistão estendido.

(d) Cesto na posição 4: Apenas peça inferior com pistão recuado.

(e) Cesto na posição 5: Sem peça com pistão recuado.

Figura 4.9: Posições possíveis do cesto durante o processo de desmontagem.

4.3 Modelagem em Rede de Petri do sistema de desmontagem

O processo de desmontagem descrito na seção anterior foi modelado em uma rede de Petri interpretada para controle dividida em 5 subsistemas:

- N_1: Braço pneumático da planta mecatrônica.
- N_2: Elevador da planta mecatrônica.
- N_3: Esteira de desmontagem.
- N_4: Braço robótico.
- N_5: Cesto.

Os subsistemas foram sincronizados utilizando eventos virtuais e a tabela 4.1 relaciona todas as transições que foram usadas para sincronismos.

Tabela 4.1: Relação de todas as transições utilizadas para sincronismo e os seus eventos virtuais.

Nº	Evento	RPIC	Transição
1	Fim de curso horizontal e vertical do elevador	Braço pneumático	t_p4
	Fim de curso horizontal e vertical do elevador	Elevador	t_e12
2	Contador do HSC = 1622	Braço pneumático	t_p6
	Contador do HSC = 1622	Elevador	t_e13
3	IEC_Counter1 = 7	Braço pneumático	t_p7
	IEC_Counter1 = 7	Elevador	t_e14
4	Sensor do atuador vertical do braço estendido	Braço pneumático	t_p10
	Sensor do atuador vertical do braço estendido	Elevador	t_e16
5	Fim de curso horizontal do elevador	Braço pneumático	t_p11
	Fim de curso horizontal do elevador	Elevador	t_e17
6	Sensor do atuador vertical do braço estendido	Braço pneumático	t_p16
	Sensor do atuador vertical do braço estendido	Esteira	t_s3
7	Sensor óptico da esteira ativado	Braço robótico	t_r1
	Sensor óptico da esteira ativado	Esteira	t_s4
8	Peças entregues	Braço robótico	t_r12
	Peças entregues	Esteira	t_s6
9	Braço robótico na posição inicial	Braço robótico	t_r4
	Braço robótico na posição inicial	Cesto	t_c1
10	Sensor magnético pistão estendido	Braço robótico	t_r5
	Sensor magnético pistão estendido	Cesto	t_c2
11	Braço robótico na torre de peças plásticas	Braço robótico	t_r8
	Braço robótico na torre de peças plásticas	Cesto	t_c3
12	Sensor magnético pistão recuado	Braço robótico	t_r9
	Sensor magnético pistão recuado	Cesto	t_c4

4.3.1 RPIC do braço pneumático

A RPIC do braço pneumático N_1 é representada na figura 4.10. Como hipótese, considera-se que o braço pneumático se encontra próximo a esteira de montagem ao iniciar o programa, como indica o lugar p_p1. Essa RPIC é utilizada para calibrar o braço robótico, pegar o bloco montado do elevador e entregar o bloco à esteira de desmontagem.

Primeiramente, é necessário calibrar a posição do braço pneumático, e isso é feito assim que o botão *Start* é pressionado. O braço pneumático gira no sentido horário até fazer 90º com a esteira de montagem. Nesse ponto há um sensor indutivo, que ativa, disparando a transição t_p2.

Para fazer a medição da rotação do braço pneumático, é necessário utilizar um bloco contador especial no diagrama Ladder, chamado de *High Speed Counter* (HSC), que funciona de forma similar a um contador normal, porém realiza a contagem em um tempo menor do que o ciclo de varredura do CLP, tornando possível a medição de rotações muito precisas, que é necessária para esse tipo de equipamento, e sempre que o HSC for utilizado, é necessário habilitá-lo.

Após disparar a transição t_p2, é necessário girar o braço 90º no sentido horário, com o objetivo de colocá-lo na posição da esteira de montagem. Uma rotação de 90º como a anterior equivale a aproximadamente 808 unidades de contagem no HSC, portanto, após esse número de contagens, a transição t_p3 pode disparar. Uma ficha no lugar p_p5 indica que o braço terminou o processo de calibração.

O próximo passo é pegar o bloco montado do elevador. Para tanto, é necessário esperar que o elevador esteja calibrado e tenha pegado a peça no armazém, e isso é feito ao sincronizar as transições t_p4 e t_e12. As transições utilizadas para sincronismo nas redes de Petri desse trabalho estão destacadas, para diferenciá-las das transições simples. Assim que o elevador estiver com o bloco e o seu fim de curso for ativado, indicando que o mesmo está na posição inicial, a transição t_p4 é disparada, levando uma ficha de p_p5 para p_p6, lugar esse que tem a ação de estender o atuador vertical do braço, levantando-o para evitar a colisão com outros elementos da planta mecatrônica. Após o braço ser estendido completamente, o sensor relacionado será ativado, habilitando a transição t_p5, que dispara depois de 1 segundo, por segurança. Os lugares p_p7 e p_p8 recebem uma ficha, fazendo o braço girar 180º e ficar próximo ao ponto de entrega do elevador. Ao atingir a contagem de 1622, a transição t_p6 é sincronizada com t_e13 e ambas disparam, levando uma ficha para p_p9. Como a válvula utilizada para estender o atuador vertical do braço pneumático é de simples solenoide, para elevar o braço é necessário ativá-la e para rebaixá-lo, basta deixar de acionar a válvula. Dessa forma, é necessário ter a ação em todos os lugares em que se deseja manter o braço elevado.

O lugar p_p9 apenas continua estendendo o atuador vertical do braço pneumático e espera o elevador andar para trás até se alinhar com o braço robótico. Ao chegar nessa posição, as transições t_p7 e t_e14 disparam juntas, e o braço pneumático pode então pegar o bloco do elevador. Uma ficha no lugar p_p10 indica que o braço continua com o atuador vertical estendido, começa a estender o atuador horizontal e liga o vácuo, que suga o ar através de uma ventosa, possibilitando grudar a ponta do braço pneumático em uma superfície lisa.

Assim que o atuador horizontal for completamente estendido, a transição t_p8 dispara, levando uma ficha para p_p11, que deixa de estender o atuador vertical, fazendo o braço pneumático descer até o bloco no elevador. Depois o braço volta a subir, dessa vez com a peça, utilizando a ventosa, como indicado no lugar p_p23. Para disparar a transição t_p10, é necessário esperar o elevador recuar completamente e sincronizar esse estado com o braço robótico, para evitar colisões.

No lugar p_p14 o braço deixa de estender o atuador horizontal, mas continua com o vertical estendido e o vácuo acionado. O próximo passo é gira o braço no sentido horário até encontrar com o sensor indutivo, onde o braço para estende novamente o atuador horizontal, deixando a peça logo acima da esteira de desmontagem. As ações do lugar p_p18 são de manter o atuador horizontal estendido e o vácuo ligado, recuando o atuador vertical até acionar o sensor, indicando que o braço pneumático estará com o bloco encostando na esteira de desmontagem. Com o bloco entregue, o braço pneumático pode então desligar o vácuo e estender o atuador vertical, liberando o bloco. O último passo é girar o braço pneumático até voltar para a posição inicial, que é acima da esteira de montagem, como indicam os lugares p_p20 e p_p21.

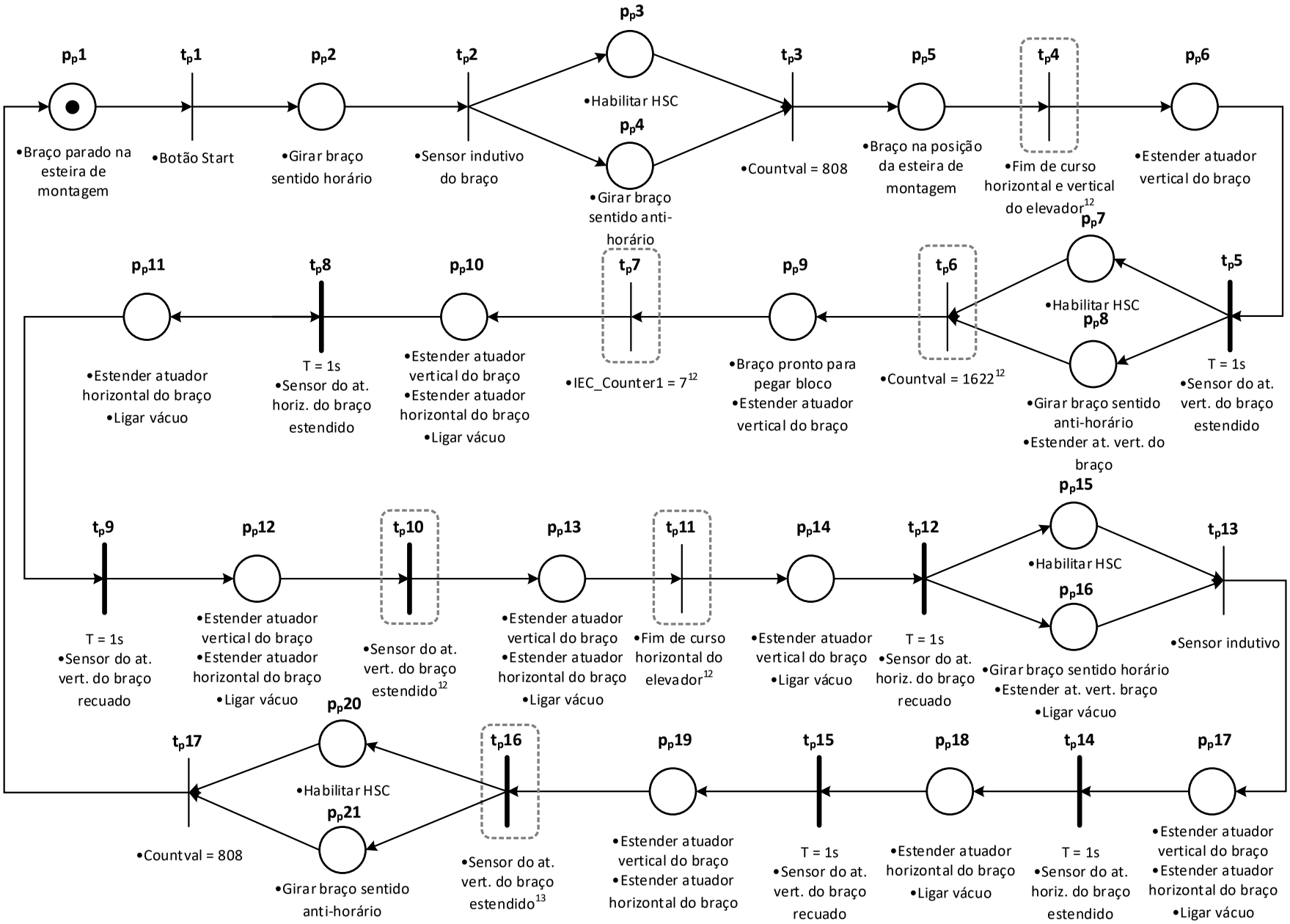


Figura 4.10: Rede de Petri N_1 do braço pneumático.

4.3.2 RPIC do elevador

A RPIC do elevador N_2 é representada na figura 4.11. Como hipótese, assume-se que o bloco está montado e posicionado no primeiro espaço do armazém do módulo 3, ou seja, no primeiro andar e na posição mais a esquerda, utilizando como referência a figura 4.1. Essa RPIC é usada para pegar o bloco do armazém utilizando o elevador e entregá-lo para o braço pneumático.

Primeiramente, é necessário garantir que o elevador está na posição inicial, escolhida como a frente do elevador, que é a extremidade oposta ao braço pneumático. Desa forma, ao pressionar o botão *Start*, a transição t_e1 é disparada, e o elevador anda tanto para baixo quanto para frente até atingir os dois fins de curso respectivos. O elevador estará então pronto para pegar o bloco, como indicado nos lugares p_e3 e p_e6. Depois das transições t_e3 e t_e5 dispararem, duas ações são executadas independentemente, a de subir o elevador um andar até o valor do contador *IEC_ Counter3* ser igual a 1 e a de mover o elevador para trás uma posição, até o contador *IEC_ Counter2* ser igual a 1 também, disparando as transições t_e4 e t_e6, respectivamente. Sempre que forem utilizados, os contadores devem ter seus valores resetados, garantindo que o elevador mova até a posição correta na próxima vez que esses contadores forem usados.

Ao disparar a transição t_e7, o elevador estará alinhado com a posição do bloco, e o próximo passo é mandá-lo descer durante 0,25s, que equivale a uma pequena distância, suficiente para ficar mais baixo que o bloco, como mostra o lugar p_e9. Em p_e10 o atuador horizontal é acionado, colocando a bandeja do elevador logo abaixo do bloco, garantindo que quando o elevador subir em p_e11, a bandeja consiga mover o bloco para cima, retirando-o do armazém. Com o bloco já no elevador, é necessário então retrair a bandeja através do atuador horizontal em p_e12 e depois mover o elevador de volta para a posição inicial, como mostrado em p_e13.

Assim que os fins de curso do elevador forem ativados e o braço pneumático tiver feito a sua calibragem inicial da posição, a transição t_e12 é disparada, colocando uma ficha em p_e14, e o elevador espera nesse lugar até que o braço faça uma rotação de 180°, disparando simultaneamente as transições t_e13 e t_p6. O lugar p_e15 manda o elevador para trás até ficar alinhado com o braço pneumático, disparando a transição t_e14.

A ficha da rede de Petri fica parada no lugar p_e17 esperando o braço pneumático retirar o bloco do elevador, e quando essa ação é finalizada, as transições t_e16 e t_p10 são disparadas, colocando uma ficha em p_e18, que manda o elevador de volta para a posição inicial até ativar o sensor de fim e curso horizontal, possibilitando que o braço pneumático possa girar sem colidir com o elevador. A ficha é então devolvida para o lugar p_e1, onde fica até o botão *Start* ser pressionado novamente.

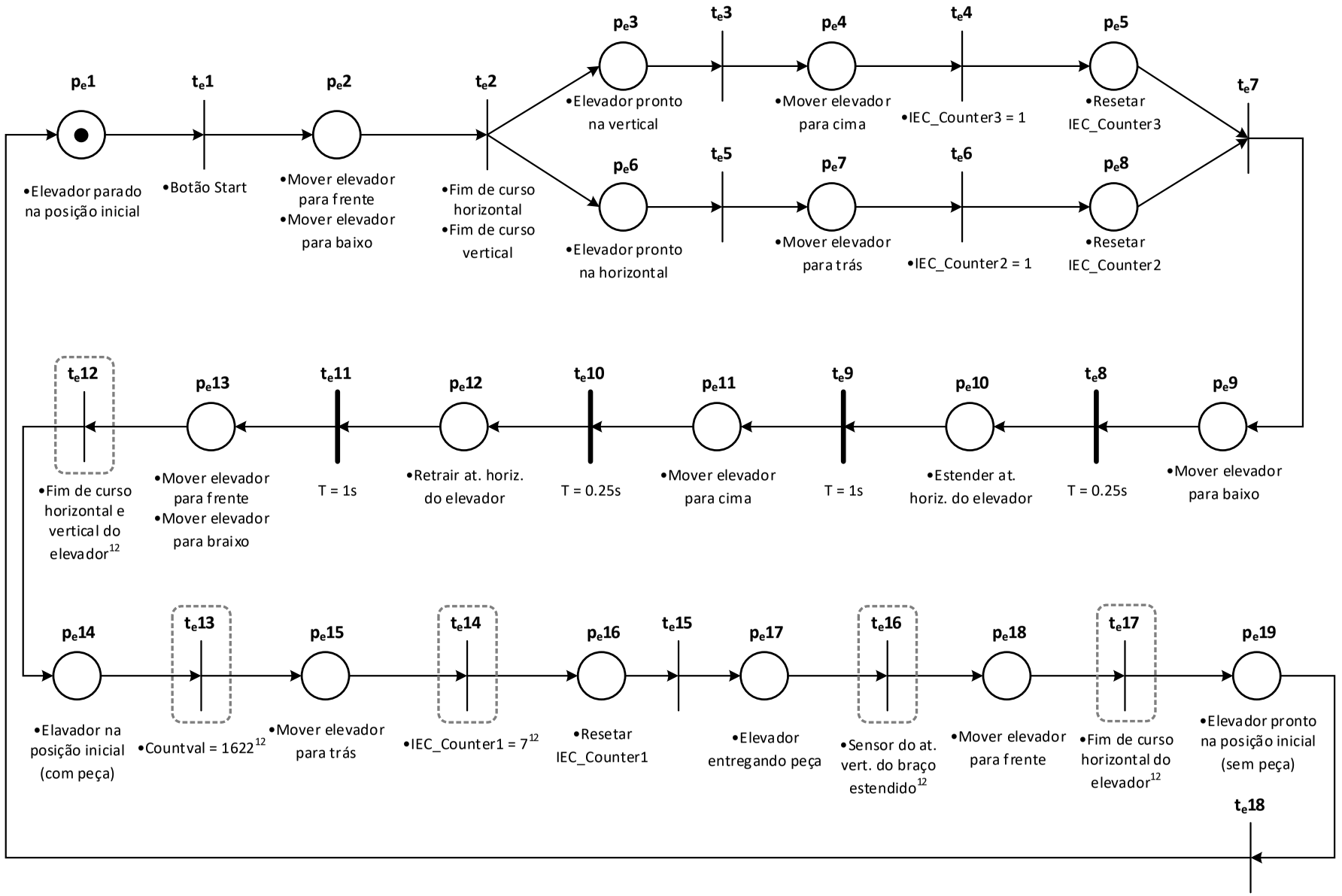


Figura 4.11: Rede de Petri N_2 do elevador.

4.3.3 RPIC da esteira de desmontagem

A RPIC da esteira de desmontagem N_3 é representada pela figura 4.12. Ela começa inicialmente parada, com uma ficha em p_s1. Essa RPIC é usada para levar o bloco deixado pelo braço pneumático de uma extremidade até a outra, onde poderá ser movido pelo braço robótico.

Ao apertar o botão *Start*, a esteira começa a andar para trás durante 12 segundos, tempo suficiente para retirar completamente qualquer peça ou bloco que tiver sido deixado por engano, descartando o objeto. Dessa forma, assim que passarem os 12 segundos, a transição t_s2 será disparada, colocando uma ficha em p_s3, onde espera até que o bloco seja colocado pelo braço pneumático no início da esteira, sincronizando e disparando as transições t_s3 e t_p16.

Com o bloco colocado na esteira, ela pode ligar e mandá-lo para frente até a outra extremidade, onde encontra uma barreira plástica para impedir que caia e um sensor de presença, que ativa assim que o bloco passar na frente, sincronizando e disparando as transições t_s4 e t_r1, alertando ao braço robótico que ele pode pegar a peça.

Quando a peça for retirada pelo braço robótico, a transição t_s5 será disparada, levando uma ficha para o lugar p_s6. A ficha só sai desse lugar quando as peças forem entregues às suas torres correspondentes pelo braço robótico, disparando a transição t_s6, que leva a ficha de volta para o p_s3, esperando o próximo bloco para realizar novamente o ciclo.

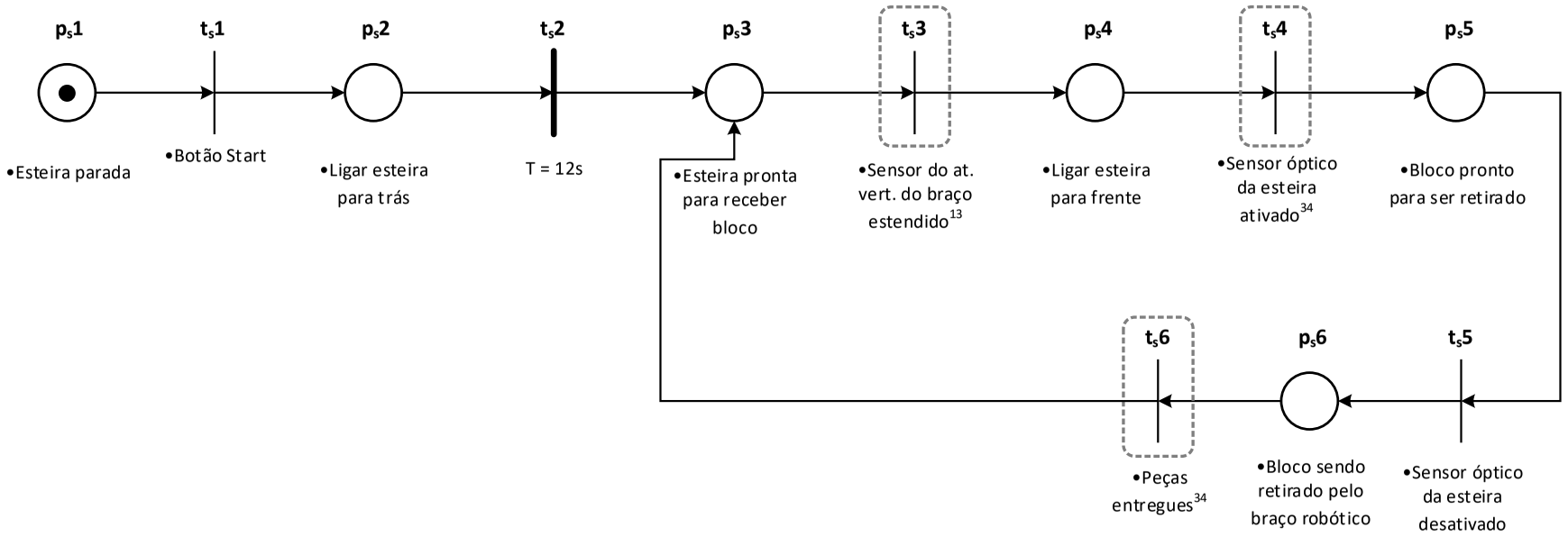


Figura 4.12: Rede de Petri N_3 da esteira.

4.3.4 RPIC do braço robótico

A RPIC N_4 inclui todas as ações do braço robótico, e é representada pela figura 4.13. Todas as suas ações do braço robótico foram separadas utilizando transições temporizadas por dois motivos: o primeiro é que não existem sensores que indiquem a posição atual do braço robótico e a segunda é que todas as instruções dadas ao braço robótico são encadeadas e executadas em apenas um programa, através do software Wincaps III, como será detalhado na seção A.

A rede de Petri N_4 começa com uma ficha no lugar p_r1, indicando que o braço robótico está na posição inicial. Assim que o bloco chegar no fim da esteira, o sensor de presença será ativado, disparando a transição t_r1, que coloca uma ficha em p_r2. A partir daí, o braço inicia o procedimento de guardar cada peça separadamente em sua respectiva torre. Primeiramente, o braço robótico pega o bloco da esteira de desmontagem, como mostra o lugar p_r2, e depois o coloca no cesto. O braço vai então para a posição inicial no lugar p_r4, apenas para garantir que o pistão do cesto seja ativado com segurança. Ao sincronizar as transições t_r4 e t_c1, a parte inferior do bloco é presa pelo pistão e quando o sensor magnético do pistão indicar que ele está completamente estendido, a transição t_r5 sincronizará com a t_c2 e ambas dispararão, colocando uma ficha em p_r6.

O braço robótico pode então se mover até o cesto e puxar a peça plástica para cima no lugar p_r7, desmontando o bloco. A peça plástica é então levada até a torre de armazenamento de peças plásticas, onde é guardada, como indica o lugar p_r8 e as transições t_r8 e t_c3 serão sincronizadas e o pistão será recuado, liberando a peça inferior. Após a primeira peça ser guardada o braço robótico vai para a posição inicial, o que acontece no lugar p_r9. Quando o pistão for recuado completamente, o sensor magnético indicará esse estado e as transições t_r9 e t_c4 serão sincronizadas e disparadas, colocando uma ficha no lugar p_r10. O braço robótico vai até o cesto e retira a peça metálica e da mesma forma que fez com a peça plástica, guarda a outra peça na torre armazenadora de peças metálicas, como indica o lugar p_r11.

Com as duas peças devidamente desmontadas e guardadas, o braço robótico termina o seu ciclo e vai de volta para a posição inicial, onde espera o próximo comando de desmontagem. Depois de passar o tempo necessário para o braço robótico executar todas as suas tarefas, a transição t_r12 é sincronizada com a t_s6 e ambas disparam, colocando uma ficha no lugar p_r1.

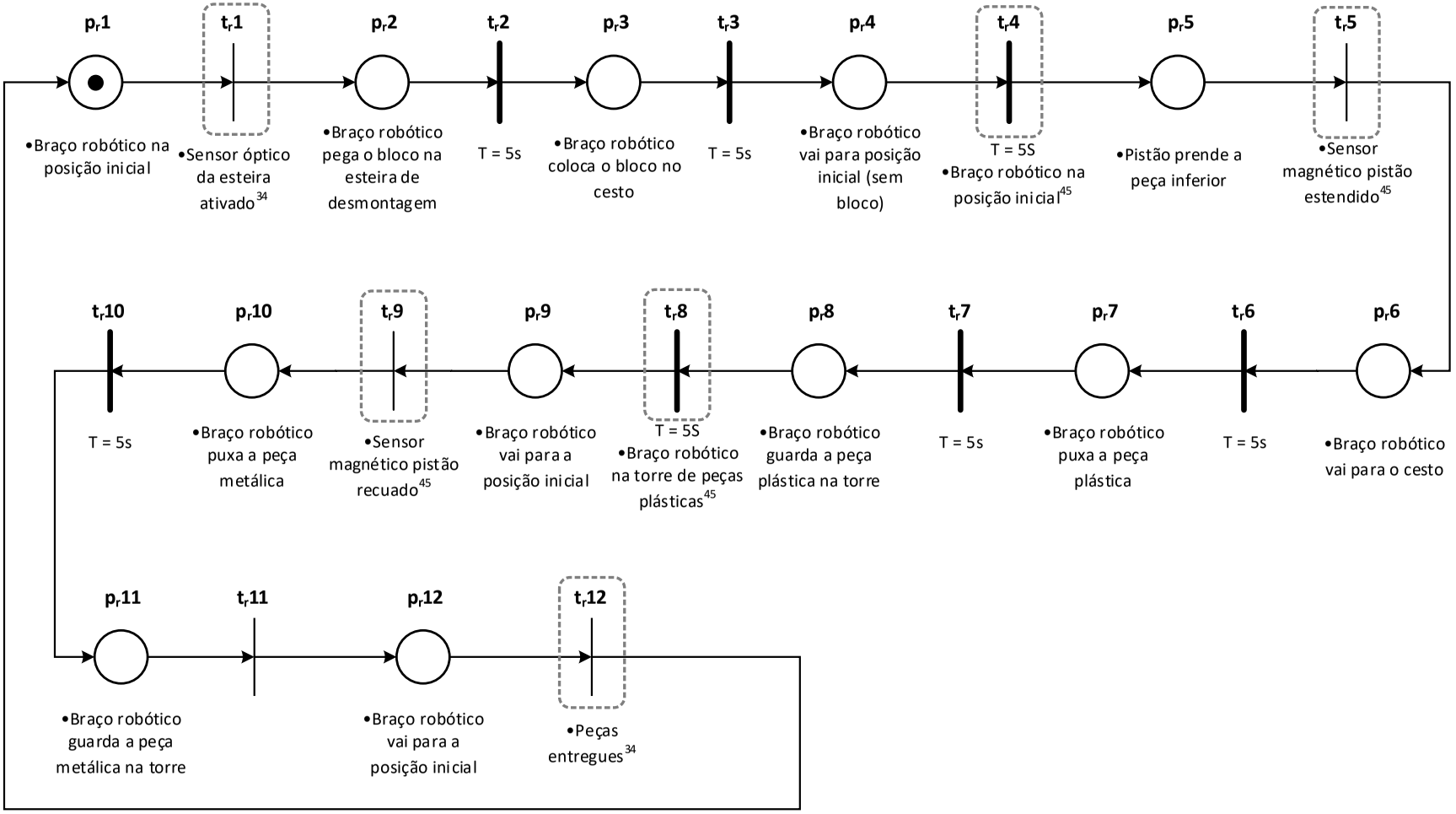


Figura 4.13: Rede de Petri N_4 do braço robótico.

4.3.5 RPIC do cesto

A RPIC N_5 é utilizada para estender e recuar o pistão do cesto, representada pela figura 4.14. O pistão começa recuado, com uma ficha no lugar p_c1. Depois que o braço robótico colocar o bloco no cesto e estiver na posição inicial, a transição t_c1 será disparada, acionando o pistão e travando a peça inferior do bloco. O braço robótico pega e guarda a peça, o que sincroniza as transições t_c3 e t_r8, colocando uma ficha no lugar p_c4 e faz o pistão recuar, liberando a peça inferior. O braço robótico pode então pegar e guardar a última peça.

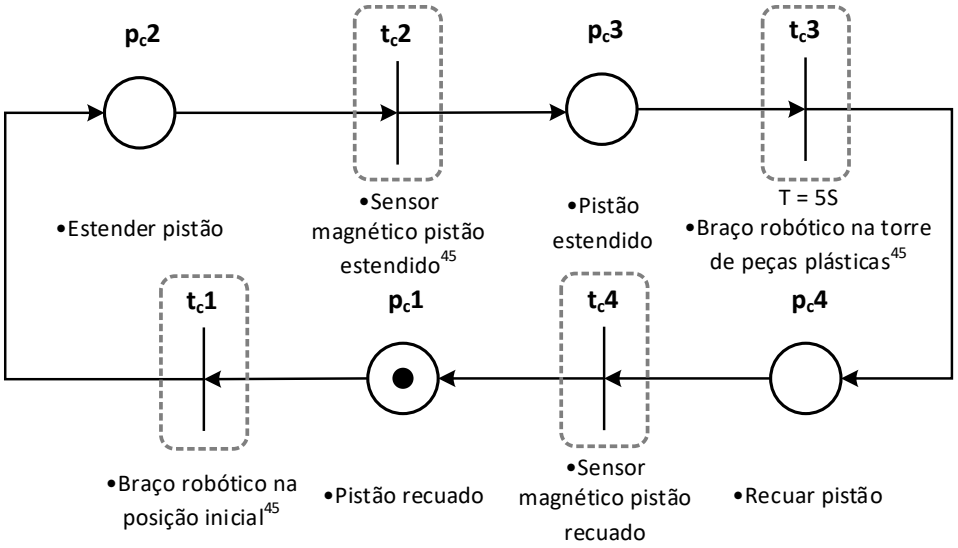


Figura 4.14: Rede de Petri N_5 do cesto.

4.4 Resultados obtidos

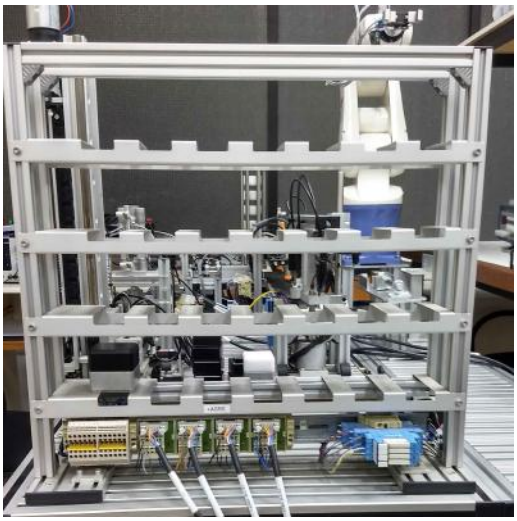
O modelo em redes de Petri desenvolvido na seção 4.3 foi utilizado como base para a produção de um diagrama Ladder utilizando o método de conversão da seção 2.4, porém apenas os subsistemas N_1 e N_2 foram convertidos. O subsistema N_3 não foi convertido porque no momento em que foi escrito este trabalho, a esteira de desmontagem ainda não tinha sido entregue ao laboratório. O subsistema N_4 do braço robótico é executado através do programa na linguagem PAC e o cesto ainda não foi integrado ao CLP, impossibilitando a conversão do subsistema N_5. Além disso, a sincronização entre o braço robótico e a planta mecatrônica pode ser feita com uma rede PROFIBUS que já foi adquirida pelo laboratório.

O programa de desmontagem na linguagem PAC executado pelo braço robótico através do software Wincaps III foi colocado no apêndice A, junto com uma imagem

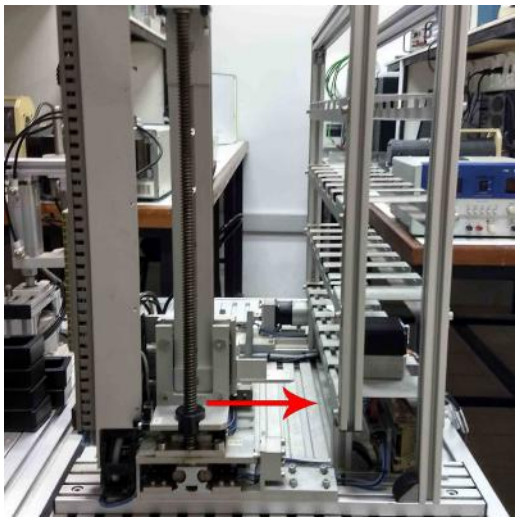
do ambiente de simulação e todas as coordenadas das posições utilizadas.

As etapas do processo de desmontagem foram divididas e as figuras 4.15 e 4.16 a seguir ilustram o processo passo a passo. Ao iniciar o processo de desmontagem, o bloco está montado e na primeira posição do armazenador, como ilustra a figura 4.15a. O atuador horizontal do elevador é acionado, pegando o bloco, como mostram as figuras 4.15b e 4.15c. O braço pneumático é girado para perto do elevador e o mesmo se move para trás até encontrar o braço, ilustrado pela figura 4.15d. O atuador horizontal do braço é estendido, o vertical é recuado e o vácuo é ligado, como mostra a figura 4.15e. O braço pode então retirar o o bloco do elevador e o mesmo pode ir de volta para a posição inicial, como ilustra a figura 4.15f.

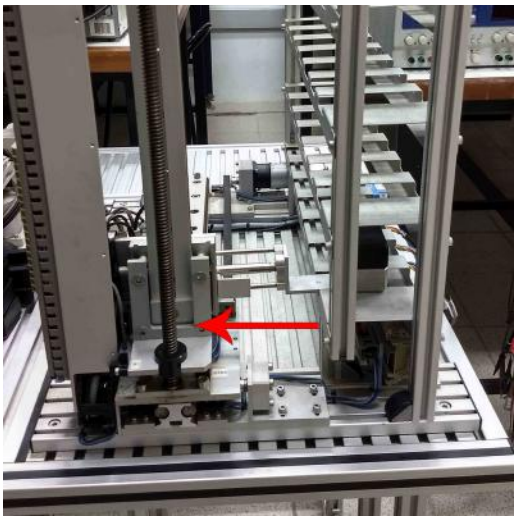
O braço pneumático gira no sentido horário, com o bloco, até encontrar o sensor indutivo, se colocando perto da esteira de desmontagem, como pode ser visto na figura 4.16a. Com o bloco entregue à esteira, o braço pode recuar e girar de volta para a posição inicial, como mostra a figura 4.16b. O bloco vai até o fim da esteira e o braço robótico o pega e o coloca no cesto. O pistão é então acionado, travando a peça inferior, e o braço robótico puxa a peça superior, desmontando o bloco, como mostra a figura 4.16c. Para finalizar, as peças agora desmontadas podem ser entregues de volta às suas respectivas torres pelo braço robótico, terminando o ciclo de manufatura, como pode ser visto na figura 4.16d.



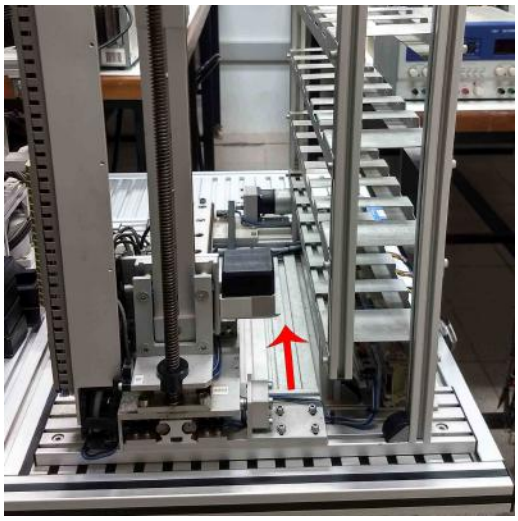
(a) Bloco no armazenador.



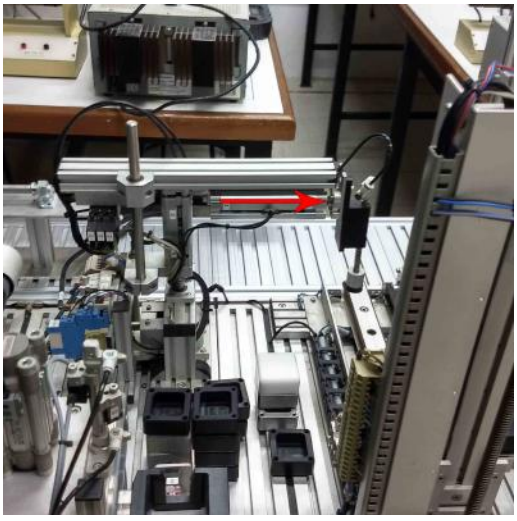
(b) Visão lateral do armazenador e elevador.



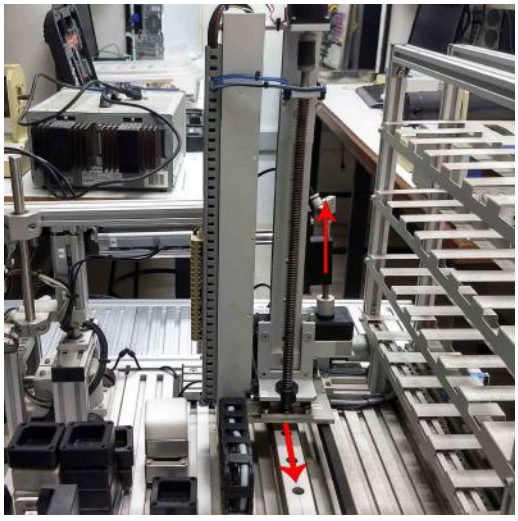
(c) Atuador horizontal do elevador é estendido.



(d) Atuador horizontal do elevador é retraído, pegando a peça.

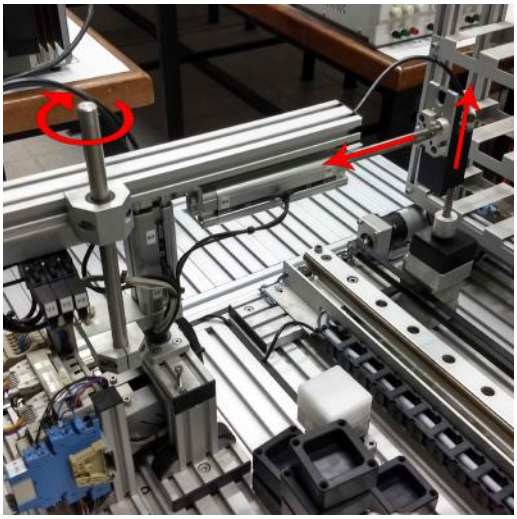


(e) Braço pneumático gira para o lado do elevador e estende o atuador horizontal.

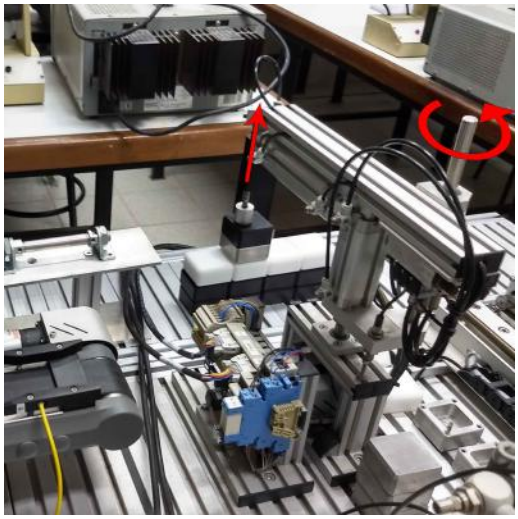


(f) Braço pneumático pega o bloco e elevador vai para posição inicial.

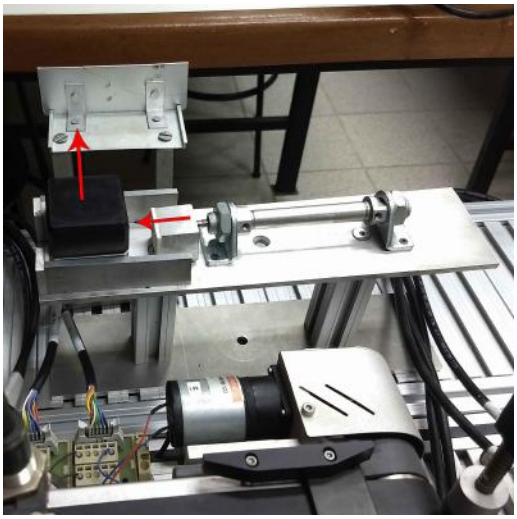
Figura 4.15: Etapas do processo de desmontagem, retirada do bloco montado.



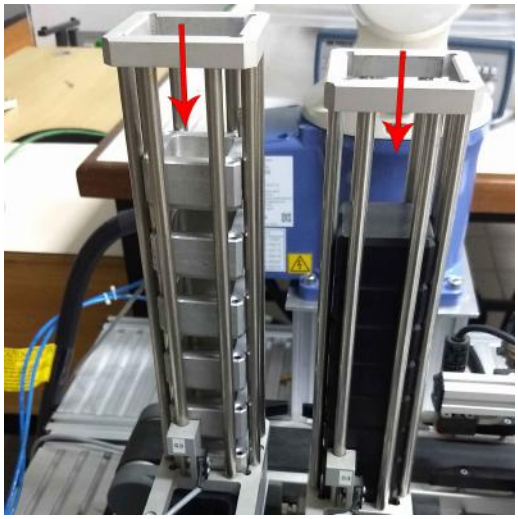
(a) Braço pneumático recua atuador horizontal, estende atuador vertical e gira no sentido horário até encontrar a esteira.



(b) Braço pneumático estende atuador vertical e gira no sentido anti-horário, voltando para a posição inicial.



(c) Pistão é estendido, travando a peça inferior, dando início ao processo de separação.



(d) Peças separadas são armazenadas nas respectivas torres pelo braço robótico.

Figura 4.16: Etapas do processo de desmontagem, separação do bloco e armazenagem das peças nas torres.

Capítulo 5

Conclusões

Este trabalho teve como objetivo apresentar um tutorial sobre a configuração e programação do braço robótico DENSO VP-6242 e o seu controlador RC7M e como esses dispositivos podem ser operados através do software Wincaps III do mesmo fabricante. Ao fim deste trabalho, foi feita a implementação de um sistema de desmontagem de blocos acoplado a um sistema de manufatura da planta mecatrônica disponível no Laboratório de Controle e Automação - UFRJ.

Foram apresentados os fundamentos teóricos de redes de Petri e de linguagem Ladder utilizados como base para a modelagem modular do sistema de desmontagem e as redes de Petri criadas foram convertidas para um diagrama Ladder, que foi implementado no CLP da planta mecatrônica.

Foi apresentado o braço robótico e as suas funções, incluindo as suas capacidades e os seus limites de mobilidade. Depois foi mostrado como montar o ambiente de operação por computador do braço robótico e como realizar a programação da movimentação do mesmo.

O sistema de desmontagem criado permitiu constatar o funcionamento favorável dos equipamentos: o braço robótico, a esteira e o cesto. Além disso, a modelagem da rede de Petri através de subsistemas atestou o correto funcionamento do sistema geral.

Como trabalhos futuros, é possível identificar os seguintes pontos:

- Instalação e integração da esteira de desmontagem.
- Conexão dos sensores e atuadores do cesto com o CLP da planta.
- Integração entre o controlador do braço e o CLP da planta, possivelmente utilizando redes PROFIBUS, para que seja possível acionar o *script* de funcionamento do braço robótico através de um comando do CLP no diagrama Ladder.

Referências Bibliográficas

[1] *DENSO Robotics Catalog*. DENSO WAVE INCORPORATED, 2013.

[2] *VP-G Series - General Information About Robot*, 11 ed. DENSO WAVE INCORPORATED, 2011.

[3] *RC7M Controller manual*. DENSO WAVE INCORPORATED, 2011.

[4] *Setting-up manual*, 4 ed. DENSO WAVE INCORPORATED, 2011.

[5] HERMANN, M., PENTEK, T., OTTO, B. "Design principles for industrie 4.0 scenarios". In: *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pp. 3928–3937. IEEE, 2016.

[6] GONZALEZ, A. G. C., ALVES, M. V. S., VIANA, G. S., etãl. "Supervisory Control-Based Navigation Architecture: A New Framework for Autonomous Robots in Industry 4.0 Environments", *IEEE Transactions on Industrial Informatics*, 2017.

[7] MOREIRA, M. V., BASILIO, J. C. "Bridging the gap between design and implementation of discrete-event controllers", *IEEE Transactions on Automation Science and Engineering*, v. 11, n. 1, pp. 48–65, 2014.

[8] JUNIOR, F. P. L. *Automação de uma Planta Mecatrônica Modelada por uma Rede de Petri Interpretada para Controle*. Projeto de graduação, Universidade Federal do Rio de Janeiro, Cidade Universitária, Rio de Janeiro, BRA, 2014.

[9] GAPANOWICZ, V. P. *Implementação de um Sistema de Automação Modular em uma Planta Mecatrônica Utilizando Sincronização Online*. Projeto de graduação, Universidade Federal do Rio de Janeiro, Cidade Universitária, Rio de Janeiro, BRA, 2016.

[10] CASSANDRAS, C., LAFORTUNE, S. *Introduction to Discrete Event Systems*. SpringerLink Engineering. Springer US, 2009. ISBN: 9780387333328.

[11] MURATA, T. "Petri nets: Properties, analysis and applications", *Proceedings of the IEEE*, v. 77, n. 4, pp. 541–580, 1989.

[12] PETERSON, J. L. "Petri nets", *ACM Computing Surveys (CSUR)*, v. 9, n. 3, pp. 223–252, 1977.

[13] DAVID, R., ALLA, H. *Discrete, Continuous, and Hybrid Petri Nets*. Springer Berlin Heidelberg, 2010. ISBN: 9783642106699.

[14] DA SILVA VIANA, G., MOREIRA, M. V., BASILIO, J. C. "Implementação de controladores a eventos discretos usando diagrama Ladder com módulo sincronizante", 2013.

[15] MOREIRA, M. V., BOTELHO, D. S., BASILIO, J. C. "Ladder diagram implementation of control interpreted Petri nets: A state equation approach", *IFAC Proceedings Volumes*, v. 42, n. 21, pp. 78–83, 2009.

[16] BOLTON, W. *Programmable logic controllers*. Newnes, 2006.

[17] *WINCAPS III Guide*, 7 ed. DENSO WAVE INCORPORATED, 2011.

[18] *Programmer's Manual I*, 4 ed. DENSO WAVE INCORPORATED, 2011.

[19] JUNIOR, F. P. L. "AUTOMAÇÃO INDUSTRIAL - PROJETO FINAL". `https://www.youtube.com/watch?v=0eCb8pFuhZk`. Acessado em Março/2018.

Apêndice A

Programa de desmontagem do braço robótico

Para controlar o braço robótico de forma a pegar os blocos na esteira, desmontá-los e guardar as peças nas torres armazenadoras, foi criado um *script* na linguagem PAC para ser executado através do Wincaps III, mostrado a seguir.

```
1   '<DESMONTAGEM.PAC>
2   'Script de desmontagem de blocos e armazenamento das peças em
        torres.
3   PROGRAM desmontagem
4
5   '##### INICIALIZAÇÃO #####
6   #DEFINE lengthDistant 80 'lengthDistant carrega o valor 80mm.
7   #DEFINE lengthClose 20 'lengthClose carrega o valor 20mm.
8
9   TAKEARM 'liga todas as juntas.
10
11  RESET IO[64] 'garante que a garra vai estar aberta antes de
        executar o programa.
12  MOVE p, @p p0 'coloca o braço na posição inicial.
13
14  '##### POSICIONAMENTO DO BLOCO #####
15  APPROACH p, p1, @p lengthDistant 'manda o robô "lengthDistant
        " milímetros acima de p1.
16  MOVE p, @p p1 'move a garra do robô para o fim de curso da
        esteira, onde o bloco está posicionado inicialmente.
17  SET IO[64] 'fecha a garra, pegando o bloco inteiro.
```

```
18        DEPART p, @p lengthDistant 'distancia o robô lengthDistant
              milímetros da posição p1.
19
20        APPROACH p, p2, @p lengthDistant 'manda o robo "lengthDistant
              " milímetros acima de p2.
21        MOVE p, @e p2 'move a garra do robô para o cesto.
22        RESET IO[64] 'abre a garra, soltando o bloco no cesto.
23        DEPART p, @p lengthDistant 'distancia o robô lengthDistant
              milímetros da posição p2.
24
25        '##### PEÇA PLÁSTICA #####
26        APPROACH p, p2, @p lengthDistant 'manda o robô "lengthDistant
              " unidades acima de p2.
27        MOVE p, @e p2 'move o robô para o cesto.
28        SET IO[64] 'fecha a garra, pegando apenas a peça de cima (plá
              stica).
29        DEPART p, @p lengthDistant 'distancia o robô lengthDistant
              milímetros da posição p2.
30
31        APPROACH p, p4, @p lengthClose 'manda o robô lengthClose milí
              metros acima de p4, ao lado da torre de peças plásticas
              para evitar a colisão com a torre.
32        APPROACH p, p3, @p lengthClose 'manda o robô lengthClose milí
              metros acima de p3, acima da torre de peças plásticas.
33        MOVE p, @e p3 'encaixa a peça na abertura da torre de peças
              plásticas.
34        RESET IO[64] 'solta a peça.
35        DEPART p, @p lengthClose 'sobe lengthClose milímetros.
36
37        MOVE p, @P p0 'volta com o robô para a posição inicial para n
              ão colidir com a torre de armazenamento.
38
39        '##### PEÇA METÁLICA #####
40        APPROACH p, p5, @p lengthClose 'manda o robo "lengthClose"
              unidades acima de p5.
41        MOVE p, @e p5 'move a garra do robô para o cesto (mais baixo)
              .
42        SET IO[64] 'fecha a garra, pegando a peça metálica.
```

```
43        DEPART p, @p lengthClose 'distancia o robô lengthClose milí
              metros da posição p5.
44
45        APPROACH p, p4, @p lengthClose 'manda o robô lengthClose
              milimetros acima de p4, ao lado da torre de peças plá
              sticas.
46        APPROACH p, p6, @p lengthClose 'manda o robô lengthClose
              milimetros acima de p6, acima da torre de peças metálicas.
47        MOVE p, @e p6 'encaixa a peça na abertura da torre de peças
              metálicas.
48        RESET IO[64] 'solta a peça.
49        DEPART p, @p lengthClose 'sobe lengthClose milímetros.
50
51        MOVE p, @P p0 'termina o programa colocando o robô na posição
               inicial.
52
53        END
```

A figura A.1 mostra a tela do Wincaps III com o ambiente virtual do sistema de desmontagem, incluindo 4 estruturas: a torre de peças metálicas, simbolizada pela cor amarela, a torre de peças plásticas, simbolizada pela cor rosa, o pistão, colorido em vermelho e a esteira, em verde. Além das estruturas, há também modelos do bloco montado e das peças individuais em posições importantes. As coordenadas de todas as posições utilizadas estão exibidas na figura A.2.

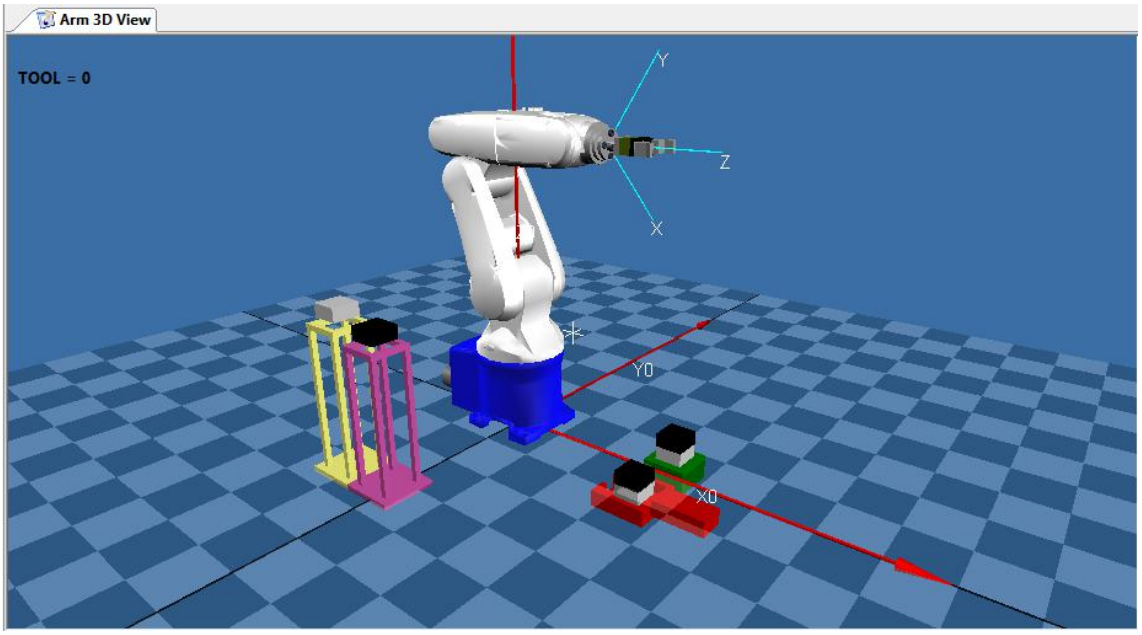


Figura A.1: Tela do Wincaps III mostrando o ambiente virtual do braço e o sistema de desmontagem.

Type P

Jump | Smart View | Get Position | Move

No.	X	Y	Z	RX	RY	RZ	FIG	Usage	Macro	Smart
0	200	0	550	90	45	90	5 - Lefty | Above | NonFlip | J6Single | .	(Reserved)		
1	324	50	149	180	0	45	5 - Lefty | Above | NonFlip | J6Single | .			
2	305	-51	114	180	0	45	5 - Lefty | Above | NonFlip | J6Single | .	Area 1 - interference pc	pAreaColl1	
3	-12	-319	379	180	0	45	5 - Lefty | Above | NonFlip | J6Single | .	Area 2 - interference pc	pAreaColl2	
4	110.6211	-318	390	180	0	45	5 - Lefty | Above | NonFlip | J6Single | .	Area 3 - interference pc	pAreaColl3	
5	305	-51	89	180	0	45	5 - Lefty | Above | NonFlip | J6Single | .			
6	-109	-319	397	180	0	45	5 - Lefty | Above | NonFlip | J6Single | .	Area 5 - interference pc	pAreaColl5	

Type P | Output

Figura A.2: Tela do Wincaps III mostrando as posições utilizadas no programa de desmontagem e as suas coordenadas.

Apêndice B

Modelo da esteira de desmontagem

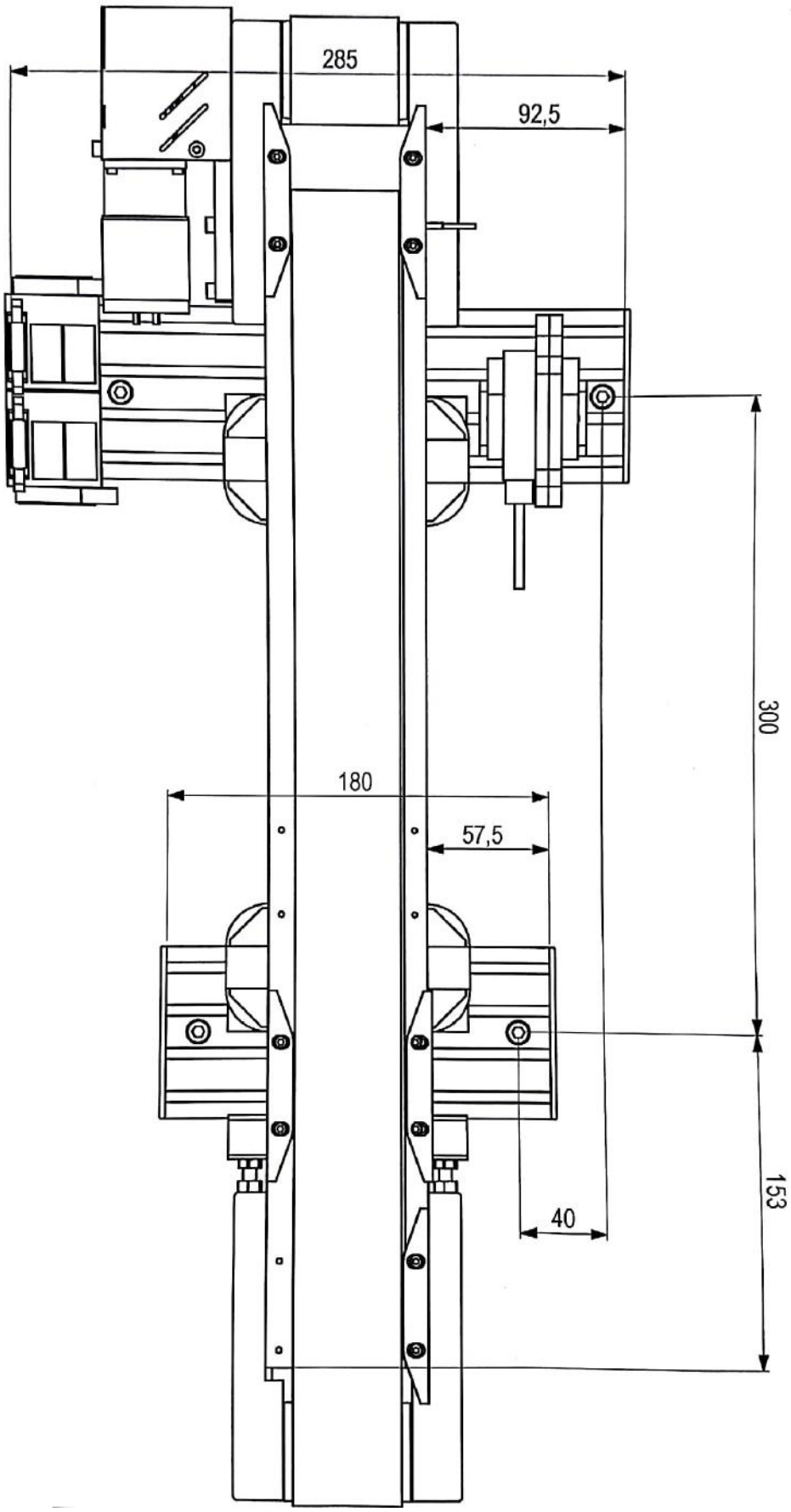


Figura B.1: Modelo e dimensões da esteira de desmontagem projetada.