FERNANDO HITOSHI WATANABE

Emulador de Sistemas de Controle

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para a conclusão de curso.

São Paulo

2021

FERNANDO HITOSHI WATANABE

Emulador de Sistemas de Controle

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para a conclusão de curso.

Curso de Graduação:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Fabrício Junqueira

São Paulo
2021

Catalogação-na-publicação

Watanabe, Fernando Hitoshi
Emulador de Sistemas de Controle F.H. Watanabe

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos. 39p.

1.Rede de Petri 2. MPS 3.FESTO
I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

AGRADECIMENTOS

Ao meu orientador Prof. Dr. Fabrício Junqueira pelo seu apoio e orientação para realizar esse projeto em um curto espaço de tempo.

Aos meus pais e meu irmão pelo apoio e auxílio durante toda a graduação.

À minha namorada Leticia Bonella pelo apoio durante grande parte da minha graduação.

Ao meu amigo Diego Hidek pelo auxílio quando tive dificuldades na parte de programação.

Ao meu mentor Henry Ishizaki por me incentivar a me formar e se preocupar com o balanço entre trabalho e faculdade.

RESUMO

Os simuladores didáticos permitem que os alunos aprendam melhor e se sintam mais motivados, por isso foi criado um emulador que permite simular as bancadas do *Modular Production System* (MPS) da FESTO utilizando Rede de Petri. A Rede de Petri é uma ferramenta gráfica e matemática que proporciona um ambiente uniforme para modelagem, análises formais e projeto de sistemas a eventos discretos, como é o caso de muitas linhas de processos sequenciais. Por isso a Rede de Petri consegue modelar muito bem como os sistemas produtivos feitos pelo homem funcionam. Dentre os sistemas produtivos se encontram as linhas de produção e mesmo equipamentos. Alguns desses equipamentos são as bancadas do MPS da FESTO. Os sistemas de controle destas bancadas foram modelados por meio de Rede de Petri, para sua verificação e validação, antes de serem codificados para a linguagem de controle das bancadas, de forma a garantir que serão codificados nas bancadas sem erros. Neste contexto, o objetivo desse trabalho é emular as bancadas MPS da FESTO, de forma que a Rede de Petri referente ao sistema de controle possa ser validada antes de ser convertida para outras linguagens como Ladder ou SFC (*Sequential Function Charts*).

ABSTRACT

Didactic simulators allow students to learn better and feel more motivated, so an emulator has been created that allows them to simulate FESTO's Modular Production System (MPS) benches using Petri Net. The Petri Net is a graphical and mathematical tool that provides an uniform environment for modeling, formal analysis and system design at discrete events, as is the case with many sequential process lines. That's why the Petri Net can model very well how man-made production systems work. Among the production systems are the production lines and even equipment. Some of these equipments sits on FESTO’s MPS benches. The control systems of these benches were modeled by Petri Net, for verification and validation, before being coded for the control language of the benches, in order to ensure that they will be coded in the benches without errors. In this context, the objective of this work is to emulate FESTO’s MPS benches, so that the Petri Net referring to the control system can be validated before being converted to other languages such as Ladder or SFC (Sequential Function Charts).

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

Sigla	**Descrição**
MPS	*Modular Production System*
PIPE	*Platform Independent Petri Net Editor*
XML	*eXtensive Markup Language*

SUMÁRIO

1. Introdução

A Rede de Petri é uma ferramenta gráfica e matemática muito útil para se modelar sistemas a eventos discretos, como é o caso de muitas linhas de processos sequenciais. Por sua relevância, ela é um dos tópicos ministrados e discutidos na disciplina de Sistemas a Eventos Discretos.

Nesta parte do curso os alunos são incentivados a aplicar a teoria aprendida por meio do *software* PIPE, que faz a verificação e validação de uma Rede de Petri construída. Existe, porém, uma carência de confrontar a evolução da lógica de controle frente ao objeto de controle, a fim de tornar o processo mais visual e intuitivo para o aluno da disciplina.

O curso conta também com uma parte prática, em que são utilizadas as bancadas *Modular Production System* (MPS) da FESTO para se entender o funcionamento de sistemas a eventos discretos. Estas mostraram-se bons equipamentos didáticos de metodologia de controle, devido à fácil compreensão do funcionamento de cada bancada e da possiblidade de gerar um sistema grande e complexo com a união delas.

Surge, porém, o contexto da pandemia de COVID 19, em que, segundo WEINTRAUB (2020) foi autorizada, em caráter excepcional, a substituição das disciplinas presenciais por aulas que utilizem meios e tecnologia de informação e comunicação, por instituição de educação superior integrante do sistema federal de ensino. Ainda segundo WEINTRAUB (2020), alternativamente à autorização da substituição das aulas presenciais, as instituições de educação superior poderão suspender as atividades acadêmicas presenciais pelo mesmo prazo. Para não precisar suspender as aulas, a adoção de aulas on-line permite que os alunos não percam o semestre letivo. Ainda mais em casos em que a suspensão de aulas presenciais dura mais de seis meses, já que as aulas da USP foram suspensas em 17/03/2020 e segundo KNOBEL (2020) e HERNANDES, et al. (2020), as aulas presenciais só devem voltar em 2021.

É a partir deste contexto, que nasce a ideia desta monografia. Com objetivo de trazer esta experiencia visual e virtual para os alunos do curso, é proposto o desenvolvimento de um emulador que irá juntar as instruções de controle, em forma de Rede de Petri, geradas no PIPE com as informações de entrada e saída do

equipamento a ser controlado (no caso, as bancadas MPS). Desta forma o emulador permite que o usuário controle o equipamento por meio de uma interface que mostra visualmente o que está acontecendo nele.

Nesse contexto um emulador permite que o aluno tenha a experiência didática de trabalhar com as bancadas MPS da FESTO sem precisar estar fisicamente na universidade e aproximar a teoria da Rede de Petri com a prática do funcionamento do equipamento a eventos discretos. Além disso permite que o aluno teste lógicas de controle em um sistema real que já é utilizado para fins didáticos.

Por fim, esta ideia vem embasada de uma bibliografia, que mostra os simuladores sendo estudados como ferramenta didática e sua eficácia frente a outros métodos, ajudando na absorção e fixação do conteúdo em um ambiente controlado, e até sua utilização em laboratórios virtuais como demonstram HERADIO, DE LA TORRE e DORMIDO (2016). O que apoia o desenvolvimento deste emulador.

1.1. Objetivos do trabalho

O objetivo desse trabalho é o projeto e implementação de um emulador para as bancadas didáticas *Modular Production System* (MPS) da FESTO de forma a se testar a lógica de controle previamente modelada em rede de Petri e validada na ferramenta PIPE.

1.2. Organização do texto

O capítulo 2 apresenta a revisão bibliográfica sobre as bases que apoiam esse trabalho. Nele são abordados simuladores didáticos, Rede de Petri e MPS da FESTO

No capítulo 3 é feita uma descrição do sistema proposto por meio de requisitos, Casos de Uso, Diagrama de Atividades e Diagrama de Classes.

No capítulo 4 é feita uma descrição do que foi implementado e comentários sobre os testes realizados.

Por fim o capítulo 5 mostra a conclusão obtida com o projeto.

2. Revisão Bibliográfica

Para alcançar os objetivos deste trabalho, foi realizada uma revisão bibliográfica sobre simuladores didáticos, que é o produto deste trabalho; sobre a Rede de Petri, que é a base teórica para a Linguagem que será utilizada para especificar a lógica de controle do simulador, e sobre o *Modular Production System* (MPS) da FESTO, que é a referência para o equipamento utilizado para testar o funcionamento.

2.1. Simuladores Didáticos

A escolha de uso de um simulador didático pode acontecer por vários motivos, seja para garantir uma experiencia prática de um fenômeno para os alunos, seja por fornecer um ambiente seguro e controlado em relação a eventuais erros ou até para engajar mais a participação e entendimento sobre o fenômeno ou teoria a ser apresentada. A fim de entender sobre os benefícios dos simuladores didáticos, foi feita uma pesquisa bibliográfica que traz exemplos e estudos de casos sobre o assunto.

Os artigos permeiam vários níveis de ensino. Pode-se citar, no ensino básico, o simulador como uma alternativa para auxiliar na visualização e representação dos fenômenos físicos. É o caso de SOUZA (2011), que aplica um *software* de visualização de fenômenos de ótica, como por exemplo, reflexão e refração, com exemplos reais e visuais. O resultado é que, quando ministrado junto com a aula teórica, os alunos demonstraram mais engajamento e melhor entendimento do fenômeno físico, obtendo uma nota maior no questionário teórico pós aula.

Além disso, SOUZA (2011) disserta sobre o momento ideal de se aplicar um simulador didático: antes ou depois da teoria. Em seus resultados, ele mostra que, os alunos que receberam a teoria, antes de utilizar o *software*, já entendem uma parte do fenômeno físico e fixam seu conhecimento através da simulação. Neste caso, o desempenho dos alunos no questionário aplicado foi melhor do que aqueles que receberam a teoria em segundo momento.

Já no uso de *softwares* de simulação que tem o objetivo de criar um ambiente de estudo prático controlado e passível de erros, são encontrados diversos estudos de caso no ramo da medicina, como SOLYMOS, et al. (2015) e SEMLER, et al. (2015).

No artigo de SOLYMOS, et al. (2015) analisa o uso de simuladores didáticos em uma matéria sobre socorro de pacientes em casos críticos. Como a própria natureza do curso já deixa opções limitadas de ensino, geralmente é dado por meio de aulas expositivas com estudos de caso. Os alunos, então, são divididos em grupos e participam, uma parte com aulas convencionais e a outra com aulas em um manequim simulador de condições críticas, com alto nível de fidelidade a um paciente em socorro. Nesse caso o simulador proporcionou notas maiores para os alunos que utilizaram o manequim simulador.

SEMLER, et al. (2015) realizaram um estudo parecido em que alunos de medicina participantes de um orientação sobre “trabalho em equipe durante emergências para médicos residentes”, foram divididos entre: aula teórica, expositiva de casos e teoria sobre trabalho em equipe, aula demonstrativa, em que profissionais qualificados demonstraram casos reais de trabalho em equipe em casos de emergência e aula com um manequim simulador de alta fidelidade, em que os alunos puderam ter uma experiencia de trabalho em equipe em socorro em um ambiente controlado.

Neste caso, os alunos com aulas demonstrativas tiveram as notas mais altas, seguidas das aulas com simulador e razoavelmente atrás as aulas teóricas. Apesar de ter tido uma média de notas um pouco abaixo das aulas demonstrativas, as aulas com simulador tiveram uma dispersão menor como pode ser visto na Figura 2.1.

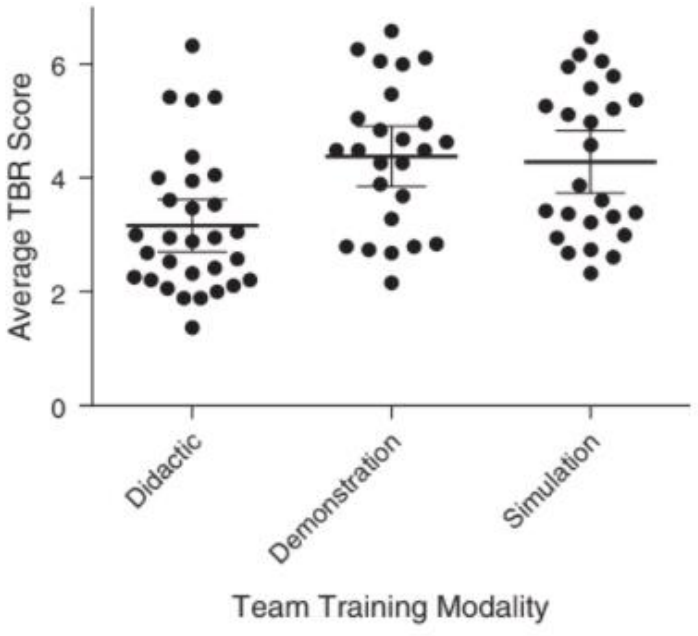

Figura 2.1– Distribuição das notas SEMLER, et al. (2015)

KLUG e HAUSBERGER (2009) dissertam sobre a implementação de um *software* de controle de tempo em processos de produção no curso de “Estruturas de Produção” na Alemanha. Os alunos deveriam criar hipóteses de otimização do processo e aplicar no simulador, até chegar no resultado esperado. Os *feedbacks*, além de serem positivos e terem impactado na nota média dos alunos, mostraram que vários deles se sentiram motivados a buscar mais cursos com aulas com simuladores.

Já GUOQIANG (2010) fez um estudo teórico sobre aplicações de simuladores didáticos em disciplinas de Engenharia e defende que eles permitem aos alunos entender mais profundamente e mais completamente, pois permite que o aluno tenha contato com experimentos que demandariam muito dinheiro ou tempo ou até por serem perigosos. Isso permite que o professor deixe a teoria complexa interessante.

Por fim, encontram-se casos em que simuladores são utilizados em cursos de eventos a sistemas discretos como é visto em KRESS, et al. (2010) e até focados em engenharia mecatrônica como é visto em AL-SHARIF, et al. (2014).

A aplicação de simuladores em um curso de eventos a sistemas discretos é analisada por KRESS, et al. (2010). Como o objetivo da disciplina era democratizar a teoria e prática de eventos discretos a engenheiros de diferentes áreas, os professores escolheram o *software* ExtendSim, que comporta vários modelos de simulações sobre diferentes temas, como por exemplo, *supply chain*, que, segundo autores, proporcionou uma versatilidade à teoria de eventos discretos e possibilitou a introdução de alguns conceitos sobre o tema. Como consequência, o oferecimento do curso foi continuado e os alunos se demonstraram engajados durante as aulas.

AL-SHARIF et al. (2014), dissertam sobre o uso de uma plataforma para estudantes de engenharia mecatrônica, construída a partir dos *softwares* Simulink/SimPowerSystems na Universidade de Philadelphia, que permite controlar e modelar um sistema real de elevador. Os autores defendem que este sistema é ideal para alunos deste curso, já que oferecem uma oportunidade de integração entre diversas disciplinas e habilidades que este profissional necessita: motores elétricos e atuadores, eletrônica de potência, sistemas de controle e dinâmica de sistemas.

Segundo os autores, a adição de um projeto prático e que ligue várias áreas do conhecimento do engenheiro, é de grande ganho para qualquer programa

universitário, principalmente em um curso de natureza tão teórica. Este argumento, em conjunto com as teses anteriormente apresentadas sobre a vantagem da implementação de simuladores didáticos no ambiente de sala de aula e seus benefícios, compõem a motivação por trás desta monografia.

2.2. Redes de Petri

A Rede de Petri possui esse nome em homenagem ao Carl Adam Petri que em 1962 criou uma ferramenta matemática, parecida com uma rede, para estudar a comunicação com automatos. De acordo com ZURAWSKI e ZHOU (1995) os desenvolvimentos futuros da Rede de Petri foram facilitados pelo fato delas poderem ser utilizadas para modelar propriedades como sincronização de processos, eventos assíncronos, operações concomitantes e conflitos ou recursos compartilhados. Além disso, como ferramenta gráfica e matemática, a Rede de Petri proporciona um ambiente uniforme para modelagem, análises formais e projeto de sistemas a eventos discretos.

2.2.1. Formalização

De acordo com DAVID e ALLA (2005) a Rede de Petri possui dois tipos de nós: lugares e transições. Um lugar é representado por um círculo e uma transição por uma barra (alguns autores representam a transição por uma caixa). Lugares e transições são conectados por arcos. O número de lugares é finito e diferente de zero. O número de transições também é finito e diferente de zero. Um arco possui direção e conecta um lugar a uma transição ou uma transição a um lugar.

É obrigatório que cada arco possua um nó em cada uma das suas pontas. (De um nó k, lugar ou transição, para um nó h, transição ou lugar, tem no máximo um arco).

A Figura 2.2 representa uma Rede de Petri com 7 lugares e 6 transições e 15 arcos direcionados. O conjunto de lugares de uma Rede de Petri será chamado de P e seu conjunto de transições será chamado de T. No exemplo em questão tem-se P = {P_1, P_2, P_3, P_4, P_5, P_6, P_7} e T = {T_1, T_2, T_3, T_4, T_5, T_6}.

O Lugar P_3 é dito *upstream* (ou entrada) da transição T_3 porque tem um arco direcionado de P_3 para T_3. P_5 é dito *downstream* (ou saída) da transição T_3 porque tem um arco direcionado de T_3 para P_5. De forma similar, uma transição é

considerada entrada (*upsteam*) ou saída (*downstream*) de um lugar. Uma transição sem lugar de entra é uma transição fonte. Uma transição sem lugar de saída é uma transição sorvedouro.

Cada lugar contém um número inteiro (positivo ou zero) de marcas. O número de marcas contidas em um Lugar P_i será chamado de $m(P_i)$ ou de m_i. No exemplo em questão tem-se $m_1 = m_3 = 1$, $m_6 = 2$ e $m_2 = m_4 = m_5 = m_7 = 0$. A marca da rede, m, é definida pelo vetor M = (1, 0, 1, 0, 0, 2, 0). A marca da rede define o estado da Rede de Petri, ou mais precisamente o estado do sistema descrito pela rede de Petri. A evolução do estado corresponde então a uma evolução no sistema, uma evolução que é causada ao se disparar uma transição.

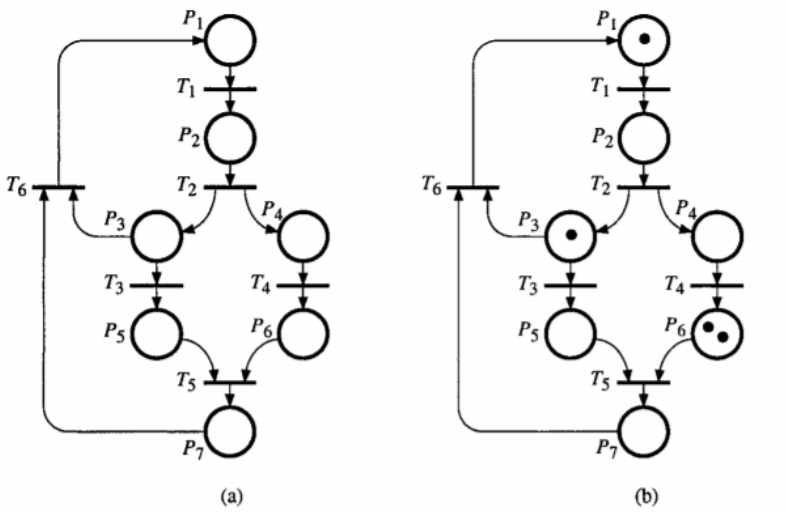

Figura 2.2 – Rede de Petri (a) Não marcada; e (b) Marcada (DAVID e ALLA 2005)

Uma transição só pode ser disparada se cada um dos lugares de entrada dessa transição contiver ao menos uma marca (considerando uma Rede de Petri ordinária, em que os arcos não possuem pesos - por padrão o peso de um arco é 1). A transição é então considerada disparável ou ativada. Uma transição fonte, portanto, está sempre ativada. As transições nas Figuras 2.3a, b e c (antes de disparar) estão ativadas porque em cada caso os lugares P_1 e P_2 contém pelo menos uma marca. Esse não é o caso da Figura 2.3d em que a transição T_1 não está ativada, pois P_1 não contém uma marca.

Disparar uma transição T_j consiste em retirar uma marca de cada um dos lugares de entrada da transição T_j e adicionar uma marca em cada um dos lugares de saída da transição T_j, considerando que os arcos possuem peso 1. Isso é ilustrado na Figura 2.3. Na Figura 2.3b, nota-se 2 marcas no lugar P_3 depois de

disparar a transição porque já havia um antes. Na Figura 2.3c, observa-se que uma marca permanece em P_1 depois de disparar.

Quando uma transição é ativada, não significa que ela vai ser disparada imediatamente. Isso apenas constitui uma possibilidade. Na Figura 2.2b, duas transições estão ativadas, T_1 e T_3. Apesar de não saber quando essas transições vão ser disparadas, sabe-se que a próxima evolução da marca vai corresponder ao disparo de T_1 ou o disparo de T_3 (pois nenhuma outra evolução é possível).

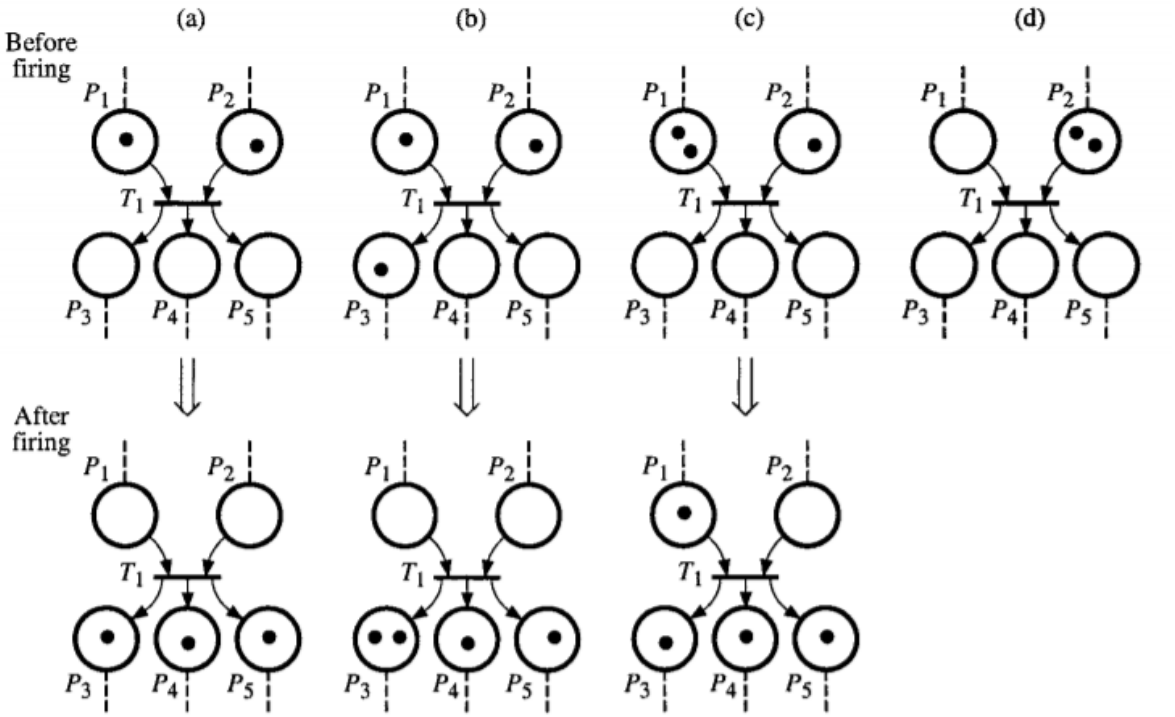

Figura 2.3 – Disparando uma transição (DAVID e ALLA 2005).

2.3. Modular Production System (MPS) da FESTO

De acordo com POLA (2012) a linha de produção em uma fábrica pode ser constituída por células de produção individuais. Cada célula tem uma função específica no processo (distribuição, análise, processamento, manuseio, montagem, armazenamento). Pode-se selecionar um processo que atenda às necessidades produtivas por meio de estações individuais ou combinações. Combinando de forma eficiente estações individuais, é possível montar sistemas de produção.

Por trás do MPS existe uma filosofia baseada em uma pirâmide com 4 níveis (Figura 2.4). Na base existem os básicos (matemática, física, eletricidade e eletrônica). Logo acima estão as tecnologias (pneumática, hidráulica, sensores,

PLC, CNC, Robótica, IT, etc.). Em seguida, tem-se os sistemas (comissionamento, detecção de falhas, planejamento, organização e trabalho em equipe). No topo estão os processos (otimização, TPM, kanban, etc.)

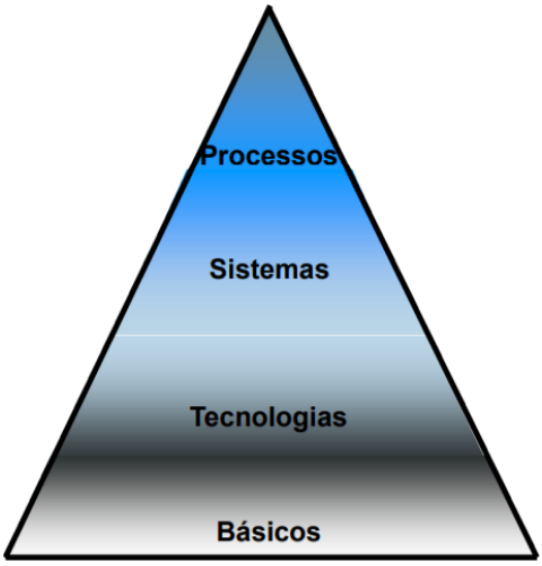

Figura 2.4 – Pirâmide da filosofia do MPS (POLA 2012)

2.3.1. Estações

Cada estação do MPS realiza uma tarefa. Todas possuem uma estrutura padrão como é mostrado na Figura 2.5. As que serão tratadas aqui são *Distributing*, *Testing*, *Processing*, *Sorting* e *Handling*.

2.3.1.1. Estação MPS®-Distributing

De acordo com POLA (2012) a Estação MPS®-Distributing (Figura 2.6) é responsável por fornecer peças de trabalho ao sistema. Podem ser armazenadas até nove peças no magazine, atuado por um cilindro pneumático de dupla ação. O módulo de transferência de peças é composto por um atuador giratório com uma ventosa para a sucção das peças. Os sinais de entrada e saída da Estação MPS®-Distributing estão documentados na Tabela 2.1.

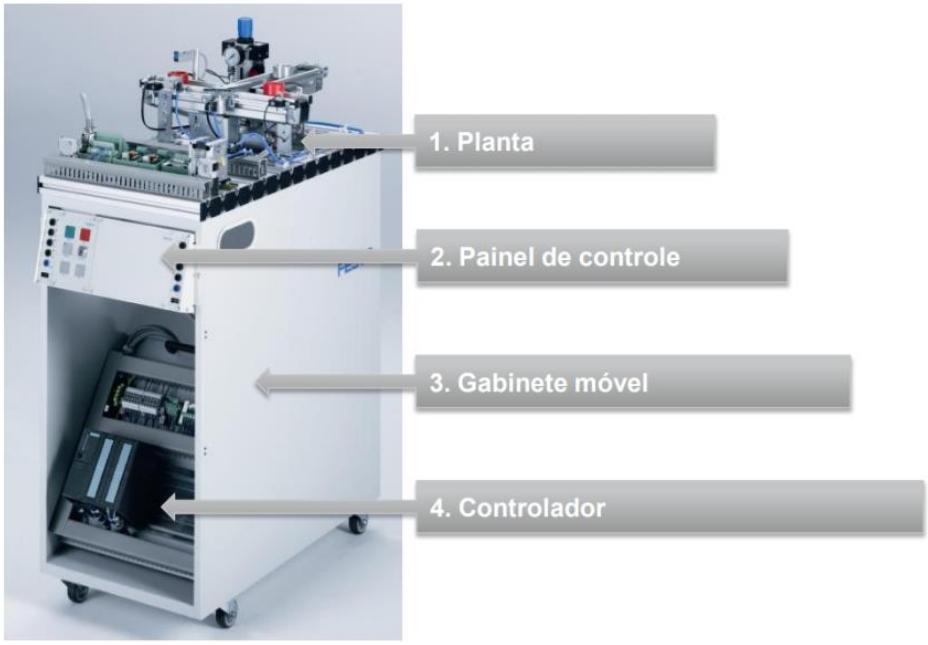

Figura 2.5 – Estrutura das estações (POLA 2012)

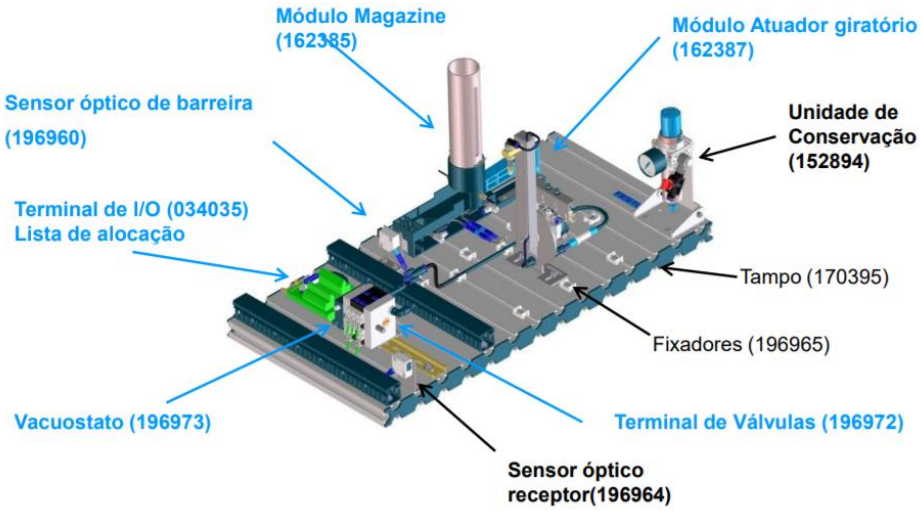

Figura 2.6 – Estação MPS®-Distributing (POLA 2012)

2.3.1.2. Estação MPS®-Testing

De acordo com POLA (2012) a estação MPS®-Testing (Figura 2.7) recebe peças providas da estação MPS® Distributing, realiza a verificação da peça e determina se a peça será aprovada ou rejeitada de acordo com os critérios estabelecidos. Por meio de sensores, é possível realizar a classificação da peça (verificação de cor), em sequência a verificação da altura da peça. Os sinais de entrada e saída da Estação MPS®-Testing estão documentados na Tabela 2.2.

Tabela 2.1 – Terminal de I/Os – Estação MPS® Distributing (POLA 2012).

Entradas digitais (IN)	Descrição	Saídas digitais (OUT)	Descrição
DI 0		DO 0	Avança atuador magazine
DI 1	Atuador magazine avançado	DO 1	Liga vácuo
DI 2	Atuador magazine recuado	DO 2	Liga Sopro
DI 3	Sensor de vácuo	DO 3	Move módulo giratório posição magazine
DI 4	Atuador giratório posição magazine	DO 4	Move módulo giratório posição próxima estação
DI 5	Atuador giratório posição próxima estação	DO 5	
DI 6	Presença de peças no magazine	DO 6	
DI 7	Próxima estação livre	DO 7	

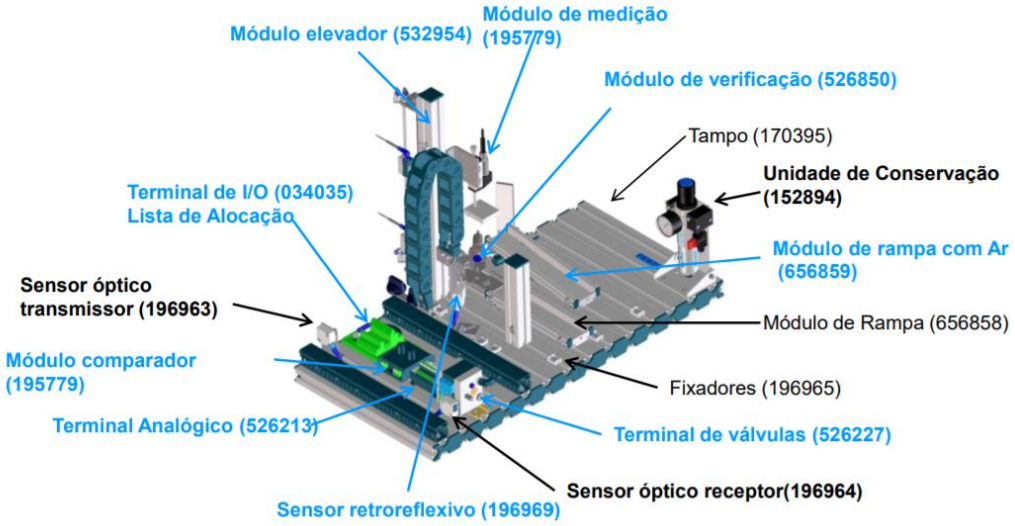

Figura 2.7 – Estação MPS®-Testing (POLA 2012)

2.3.1.3. Estação MPS®-Processing

De acordo com POLA (2012) na estação MPS®-Processing (Figura 2.8), as peças são transportadas por meio de uma mesa giratória, e processadas. É realizado a indexação da peça para verificação da posição do orifício. Em seguida é realizada uma simulação de usinagem no orifício da peça. Após esse processamento a peça é encaminhada a próxima estação. Essa estação não possui

nenhum atuador pneumático. Os sinais de entrada e saída da Estação MPS®-Processing estão documentados na Tabela 2.3.

Tabela 2.2 – Terminal de I/Os – Estação MPS® Testing (POLA 2012).

Entradas digitais (IN)	Descrição	Saídas digitais (OUT)	Descrição
DI 0	Peça disponível (sensor capacitivo)	DO 0	Desce módulo elevador
DI 1	Peça não preta (sensor óptico difuso)	DO 1	Sobe módulo elevador
DI 2	Sensor de segurança (cortina óptica)	DO 2	Avança cilindro de expulsão
DI 3	Sensor de segurança (cortina óptica)	DO 3	Aciona rampa de ar
DI 4	Módulo elevador posição superior	DO 4	
DI 5	Módulo elevador posição inferior	DO 5	
DI 6	Cilindro de expulsão recuado	DO 6	
DI 7	Próxima estação livre	DO 7	Estação ocupada

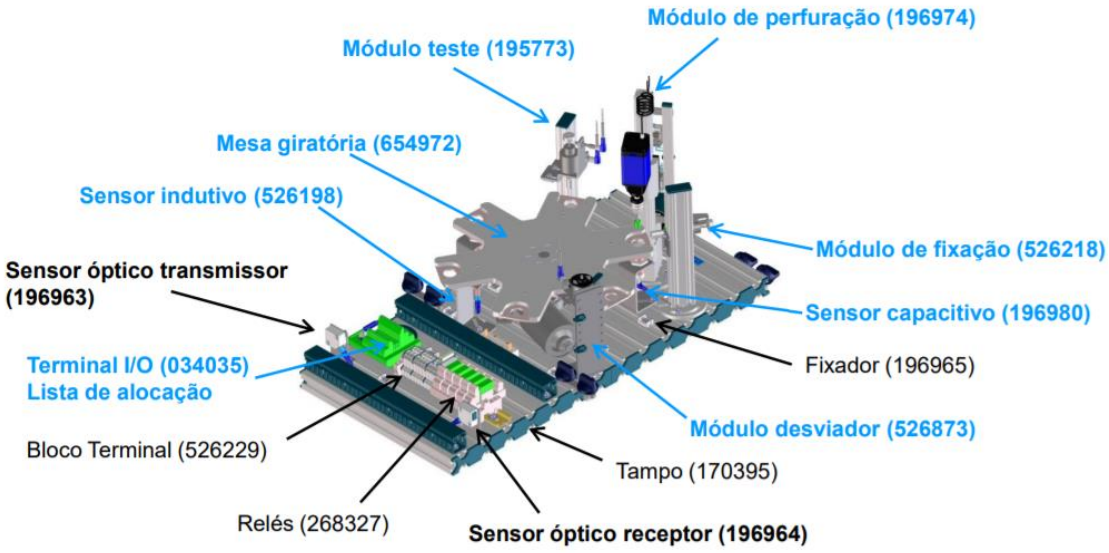

Figura 2.8 – Estação MPS®-Processing (POLA 2012)

2.3.1.1. Estação MPS®-Sorting

De acordo com POLA (2012) a estação MPS®-Sorting (Figura 2.9) realiza a classificação das peças, e as separa por tipo ou por cor. As peças são detectadas no início da esteira e levadas até o módulo de parada, pelo qual é classificada pelos sensores ópticos e indutivo. Após a classificação, as peças são separadas nas três

rampas, de acordo com os critérios estabelecidos na programação. Os sinais de entrada e saída da Estação MPS®-Sorting estão documentados na Tabela 2.4.

Tabela 2.3 – Terminal de I/Os – Estação MPS® Processing (POLA 2012).

Entradas digitais (IN)	Descrição	Saídas digitais (OUT)	Descrição
DI 0	Peça disponível (sensor capacitivo)	DO 0	Módulo de perfuração
DI 1	Peça na posição do módulo teste	DO 1	Aciona mesa giratória
DI 2	Peça na posição do módulo de perfuração	DO 2	Desce módulo de perfuração
DI 3	Módulo de perfuração recuado	DO 3	Sobe módulo de perfuração
DI 4	Módulo de perfuração avançado	DO 4	Aciona atuador de fixação
DI 5	Indexação da mesa	DO 5	Atuador de teste de profundidade
DI 6	Verificação do furo da peça	DO 6	Aciona atuador de expulsão de peças
DI 7	Próxima estação livre	DO 7	Estação ocupada

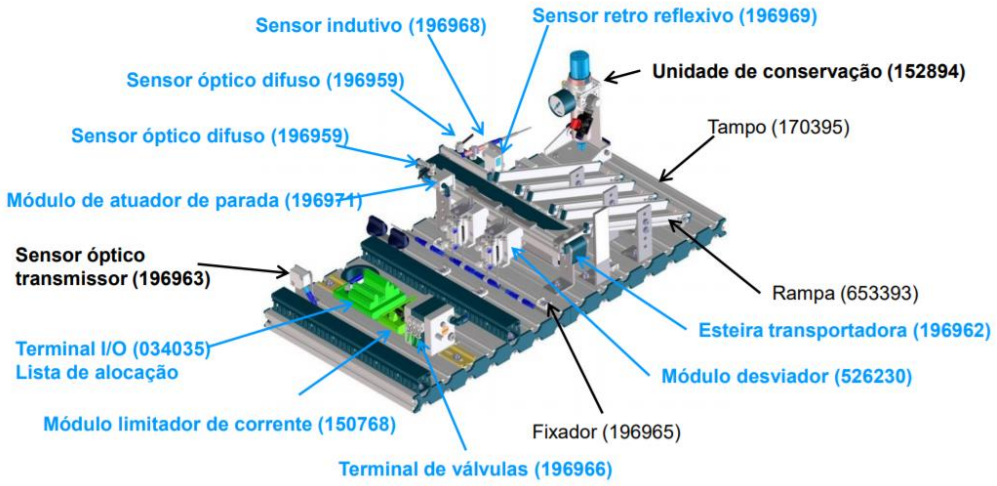

Figura 2.9 – Estação MPS®-Sorting (POLA 2012)

2.3.1.2. Estação MPS®-Handling Pneumática

De acordo com POLA (2012) a estação MPS®-Handling (Figura 2.10) é utilizada como estação de manipulação de peças. Com ela é possível realizar também a classificação, diferenciando-as pelas cores, e realizando a separação das peças. A manipulação é realizada por meio de um manipulador, com uma garra pneumática especifica para essa atividade. Os sinais de entrada e saída da Estação MPS®-Handling estão documentados na Tabela 2.5.

Tabela 2.4 – Terminal de I/Os – Estação MPS® Sorting (POLA 2012).

Entradas digitais (IN)	Descrição	Saídas digitais (OUT)	Descrição
DI 0	Peça disponível no início da esteira	DO 0	Liga esteira
DI 1	Peça metálica	DO 1	Avança atuador 1
DI 2	Peça não preta	DO 2	Avança atuador 2
DI 3	Rampa cheia	DO 3	Recua atuador de parada
DI 4	Atuador 1 recuado	DO 4	
DI 5	Atuador 1 avançado	DO 5	
DI 6	Atuador 2 recuado	DO 6	
DI 7	Atuador 2 avançado	DO 7	Estação ocupada

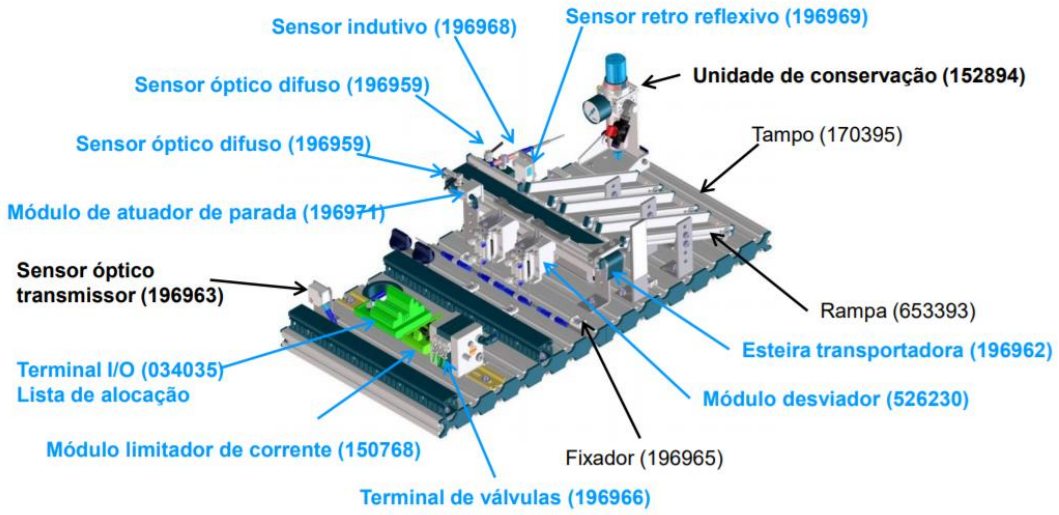

Figura 2.10 – Estação MPS®-Handling Pneumática (POLA 2012)

Tabela 2.5 – Terminal de I/Os – Estação MPS® Handling Pneumática (POLA 2012).

Entradas digitais (IN)	Descrição	Saídas digitais (OUT)	Descrição
DI 0	Peça disponível no buffer	DO 0	Move módulo Handling para próxima estação
DI 1	Módulo Handling na posição buffer	DO 1	Recua módulo Handling para estação anterior
DI 2	Módulo Handling na posição próxima estação	DO 2	Avança módulo da garra
DI 3	Módulo Handling na posição da rampa	DO 3	Abre garra
DI 4	Garra avançada	DO 4	
DI 5	Garra recuada	DO 5	
DI 6	Sensor da garra	DO 6	
DI 7	Próxima estação livre	DO 7	Estação ocupada

3. Descrição do sistema proposto

O sistema deve ser capaz de simular as bancadas MPS da FESTO de modo que, utilizando uma Rede de Petri, seja possível emular o sistema de controle de qualquer uma das bancadas.

Para isso o sistema deve ser capaz de ler um arquivo que contenha a Rede de Petri referente ao sistema de controle da bancada, cujas transições estão associadas com os sinais dos sensores da bancada e os lugares estão associadas com sinais para os atuadores da bancada (ações a serem realizadas). O sistema também deve ler um arquivo com as descrições das entradas e saídas da bancada, para que a interface mostre para o usuário o que está acontecendo.

3.1. Requisitos

Como o objetivo é simular um MPS utilizando a Rede de Petri do sistema de controle, o projeto deve atender os seguintes requisitos:

- Apresentar um painel virtual que simule as entradas e saídas do MPS e que sirva para interação com o usuário;
- Carregar um modelo em Rede de Petri que corresponda à lógica de controle que se pretende testar para o MPS;
- Carregar um arquivo com as descrições das entradas e saídas do MPS;
- Associar os *inputs* às transições da Rede de Petri;
- Associar os *outputs* aos lugares da Rede de Petri;
- Atualizar corretamente a posição das marcas após o disparo de uma transição;
- Atualizar corretamente a interface após a iteração do usuário com algum dos botões.

3.2. Modelagem do sistema

O programa primeiramente deve abrir mostrando a interface com os botões de abrir Rede de Petri, abrir bancada e fechar programa, todos habilitados; os botões para simular o sinal dos sensores desativados, círculos vazios, associados aos sinais para os atuadores, e sem descrições das saídas do controlador. Nesse ponto deve ser possível abrir uma Rede de Petri que modele o sistema de controle ou abrir o arquivo com as descrições das entradas e saídas do MPS.

Após a leitura da Rede de Petri o programa, deve manter desativados os botões relativos às transições que estão desabilitadas e ativar os botões das transições que estão habilitadas. Isso deve ser feito analisando as pré e pós-condições relativas às transições. Com relação aos círculos, preencher os referentes aos lugares que possuem uma marca e manter vazios os círculos associados a lugares sem marca.

Após a leitura do arquivo com as descrições das entradas e saídas do MPS, as descrições das entradas (sinais de sensores) devem ser posicionadas ao lado dos botões e as descrições das saídas (sinais para os atuadores) devem ser alocadas ao lado dos círculos.

Após um botão referente a uma transição habilitada ser acionado, o programa deve atualizar a Rede de Petri de forma a atualizar a posição das marcas e quais transições estão habilitadas. Essas atualizações na Rede de Petri devem gerar uma atualização na interface para manter ativos apenas os botões das transições habilitadas e preenchidos apenas os círculos dos lugares com marcas.

3.2.1. Casos de Uso

Os casos de uso servem para descrever uma sequência de ações que devem ser realizadas por um sistema para produzir uma resposta para o ator. A Figura 3.1 mostra o diagrama de Casos de Uso do sistema.

3.2.1.1. Caso de Uso "Abrir Bancada"

a) Nome do Caso de Uso

   Abrir Bancada

b) Breve descrição

   Abre um arquivo XML com as descrições dos sinais de entrada e saída das bancadas

c) Atores

   Usuário

d) Fluxo de eventos

   i. Fluxo básico

      1) O Caso de Uso começa quando o ator abre o simulador

      2) O sistema exibe a tela sem as descrições das entradas e saídas

3) O ator toca em “Abrir Bancada”
4) O sistema abre uma aba para seleção de arquivo
5) O ator seleciona o arquivo a ser aberto
6) O ator toca em “abrir”
7) O sistema abre o arquivo **[FE 01]**
8) O sistema mostra na tela as descrições das entradas e saídas da bancada

ii. Fluxo de Exceção

FE01 – Arquivo diferente do padrão

1) O sistema não reconhece o padrão do arquivo
2) A operação é interrompida
3) O programa avisa o usuário que o arquivo não foi aberto
4) Volta para a tela do simulador

e) Pós-condições

O sistema mostra as descrições dos sinais de entrada e saída da bancada

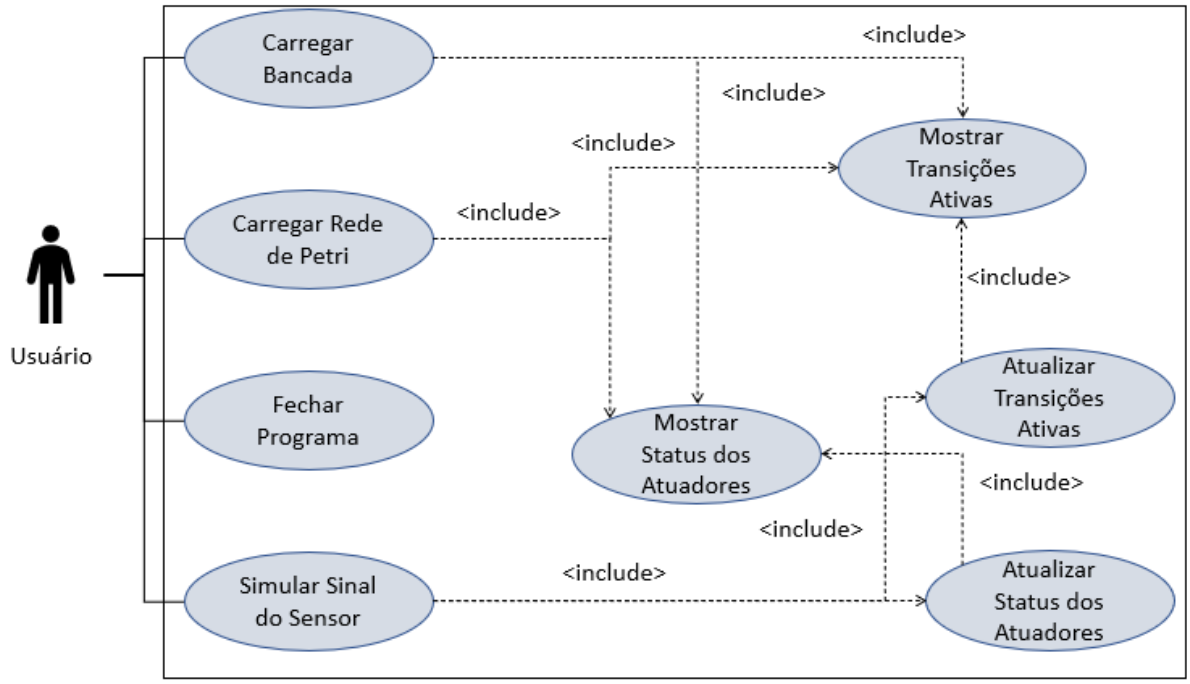

Figura 3.1 – Diagramas de Casos de Uso

3.2.1.2. Caso de Uso “Abrir RdP”

a) Nome do Caso de Uso

Abrir RdP

b) Breve descrição

Abre um arquivo XML com os parâmetros iniciais da Rede de Petri

c) Atores

Usuário

d) Fluxo de eventos

i. Fluxo básico

1) O Caso de Uso começa quando o ator abre o simulador
2) O sistema exibe a tela sem os parâmetros iniciais da RdP
3) O ator toca em “Abrir RdP”
4) O sistema abre uma aba para seleção de arquivo
5) O ator seleciona o arquivo a ser aberto
6) O ator toca em “abrir”
7) O sistema abre o arquivo **[FE 01]**
8) O sistema mostra na tela quais sinais de sensores podem ser usados e quais estão aptos a serem clicados

ii. Fluxo de Exceção

FE01 – Arquivo diferente do padrão

1) O sistema não reconhece o padrão do arquivo
2) A operação é interrompida
3) O programa avisa o usuário que o arquivo não foi aberto
4) Volta para a tela do simulador

e) Pós-condições

O sistema mostra as condições iniciais da bancada MPS

3.2.1.3. Caso de Uso “Fechar Programa”

a) Nome do Caso de Uso

Fechar Programa

b) Breve descrição

Fecha o simulador

c) Atores

Usuário

d) Fluxo de eventos
    i. Fluxo básico
        1) O Caso de Uso começa com o simulador aberto
        2) O ator toca em “Fechar”
        3) O sistema fecha o simulador
e) Pós-condições
    O simulador é fechado

3.2.1.4. Caso de Uso “Simular sinal do sensor”

a) Nome do Caso de Uso
    Simular sinal do sensor
b) Breve descrição
    Simula o sinal de um sensor que está associado à uma transição
c) Atores
    Usuário
d) Fluxo de eventos
    iii. Fluxo básico
        1) O Caso de Uso começa quando o ator toca um botão de sensor que está habilitado
        2) O sistema atualiza o *status* dos atuadores
        3) O sistema atualiza as os sinais de sensores que podem ser simulados
        4) O sistema atualiza os tanto os *status* dos atuadores como os sinais de sensores que podem ser simulados na tela
e) Pós-condições
    O sistema mostra a situação atualizada da bancada MPS

3.2.2. Diagrama de Atividades

A Figura 3.2 apresenta o diagrama de atividades que ilustra graficamente como será o funcionamento do *software*.

Nele é possível ver as atividades que o sistema executa e as decisões que precisam ser tomadas para se definir qual a próxima ação.

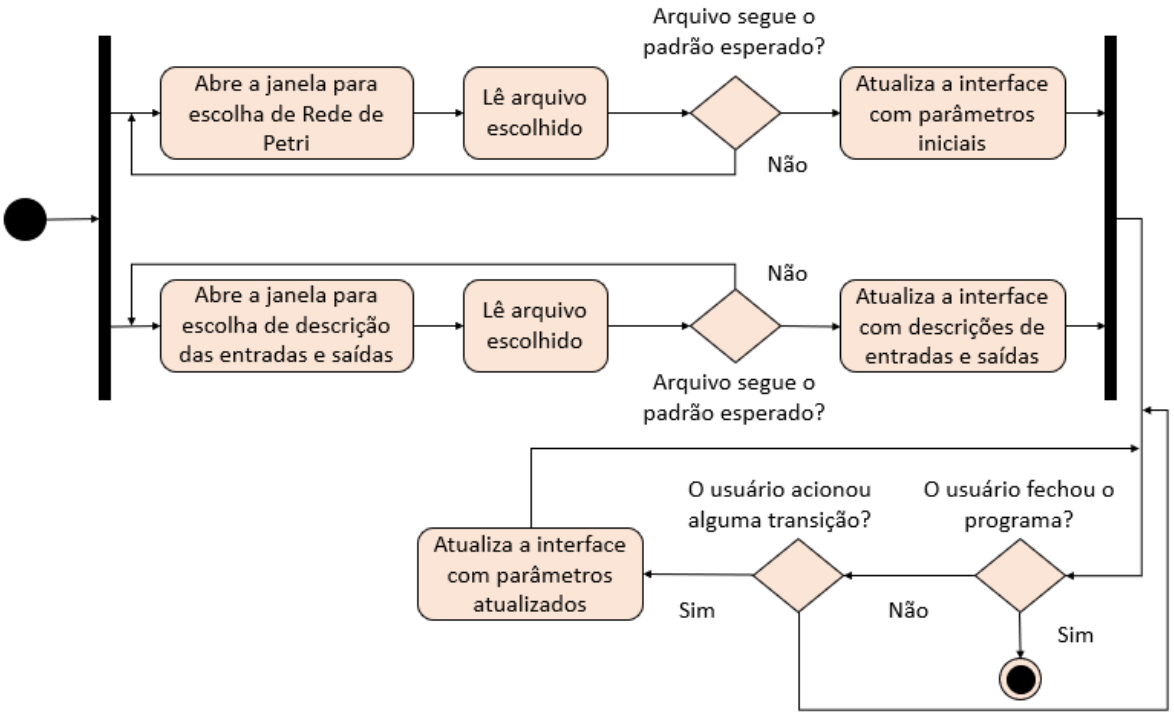

Figura 3.2 – Diagramas de Atividades

3.2.3. Diagrama de Classes

A Figura 3.3 apresenta o diagrama de classes que ilustra graficamente como será a estrutura do *software* e como cada um dos componentes da sua estrutura estarão interligados.

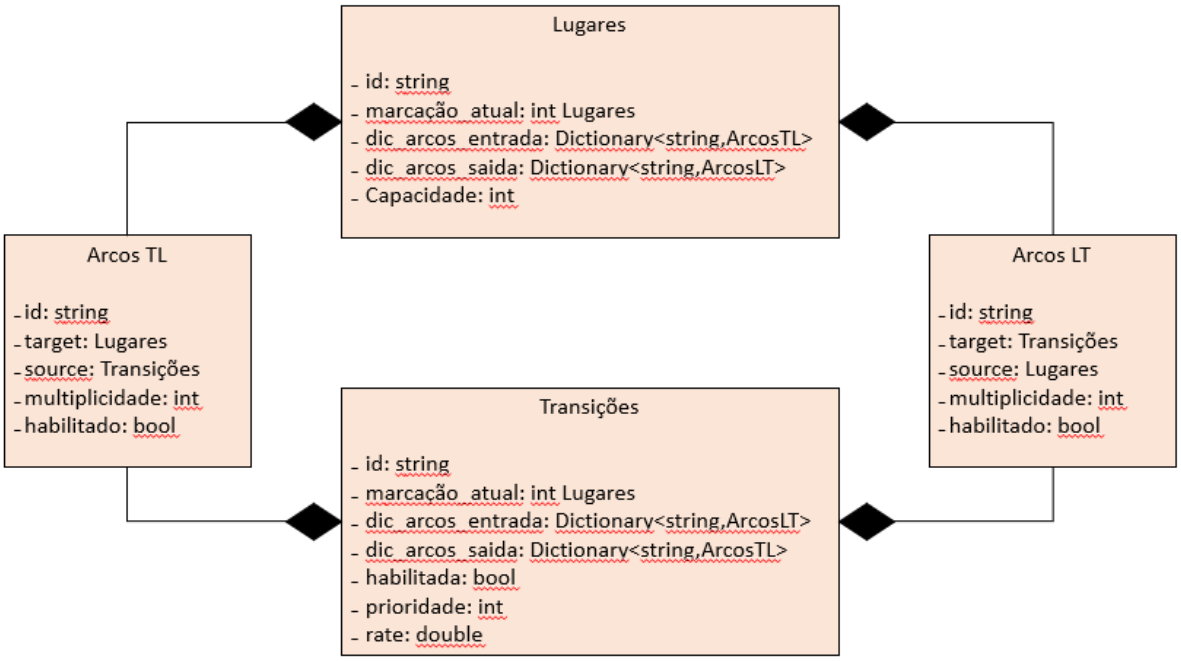

Figura 3.3 – Diagramas de Classes

4. Implementação do simulador

O ambiente de desenvolvimento utilizado para a implementação do simulador foi o *Microsoft Visual Studio 2019*. Nele foi possível criar funções que permitissem ler arquivos XML com o os parâmetros iniciais da Rede de Petri e com as descrições das entradas e saídas do MPS. Também foi possível criar fórmulas que permitissem simular o funcionamento de uma Rede de Petri, atualizando corretamente a posição de marcas e quais transições estão ativadas.

Além disso, no *Microsoft Visual Studio 2019* foi possível gerar a interface programa por meio de uma interface interativa própria para isso, o que facilitou a criação de objetos na interface e seu posicionamento.

4.1. Interpretação da Rede de Petri

Como o arquivo lido é gerado por meio do PIPE, ele possui um padrão que o permite ser lido e transformado em variáveis. Essas variáveis possuem os parâmetros iniciais da Rede de Petri em XML. Dessa forma foi possível replicar a situação inicial presente na Rede de Petri gerada no PIPE.

Tendo os parâmetros iniciais, é possível atualizá-los quando alguma transição é disparada. Para isso é preciso analisar as condições anteriores ao disparo e qual transição foi disparada, pois com essas informações é possível usar os arcos que ligam lugares e transições para remover marcas dos lugares imediatamente antes da transição disparada e adicionar marcas nos lugares imediatamente após a transição disparada. Dessa forma as marcas mudam de lugares. Seguindo também o modelo da Rede de Petri é possível distinguir quais transições estão habilitadas ou não por meio do posicionamento das marcas nos lugares.

As transições nomeadas com DI0, DI1, DI2, DI3, DI4, DI5 DI6 e DI7, serão associadas a botões e às entradas do sistema de controle. Da mesma forma, os lugares nomeados DO0, DO1, DO2, DO3, DO4, DO5, DO6 e DO7 serão associados a círculos, e às saídas do sistema de controle.

4.2. Implementação do I/O

Como o objetivo é simular as bancadas MPS da FESTO, as tabelas de I/O são padronizadas com 8 entradas e 8 saídas. Com esse padrão é possível criar um XML padrão para cada bancada. Ao ler esse XML é possível associar as entradas

com os botões e as saídas com os círculos. Esses botões e círculos são associados também às transições e lugares da Rede de Petri, respectivamente. A partir disso é possível simular a bancada utilizando as suas entradas e saídas.

4.3. Interface

A interface permite a aberturas do arquivo com a Rede de Petri e do arquivo com as entradas e saídas da bancada. Além disso, ela possui os botões que permitem o disparo das transições e círculos funcionam como sinalizadores/LEDs que permitem observar o *status* dos atuadores. Por fim ela também permite fechar o programa. Esses botões podem ser visualizados no canto esquerdo da Figura 4.1.

Como durante a leitura da Rede de Petri é possível identificar quais transições estão habilitadas, isso é repassado para os botões que simulam os sinais, só os habilitando se a respectiva transição estiver. A habilitação das transições ocorre com base nas pré e pós condições. Esses botões podem ser visualizados no meio da Figura 4.1.

O *status* dos atuadores também é atualizado assim que a base é lida e quando uma transição disparada. A variação é que quando o atuador estiver acionado, o círculo é preenchido; e se não estiver, o círculo ficará vazio. Esses círculos podem ser visualizados no lado direito da Figura 4.1.

4.4. Considerações a respeito dos testes realizados

Foram feitos alguns testes para verificar se o simulador está funcionando.

Primeiro foi testado se a leitura da Rede de Petri estava sendo feita corretamente. Desta forma, foi comparada a interface gerada com a Rede de Petri programada no PIPE. Para isso, deve-se observar os lugares na Figura 4.1 que possuem nome DO0, DO1, DO2, DO3, DO4, DO5, DO6 e DO7 e os comparar com os círculos na direita da Figura 4.2. O DO0 é representado pelo primeiro círculo, e assim sequencialmente até o DO7, que é representado pelo oitavo. Como pode ser observado no PIPE, existe marca em DO4 e o simulador mostra que o atuador está no quinto círculo.

Em seguida foi testado se o simulador conseguia identificar corretamente quais transições estavam habilitadas. Comparando as Figuras 4.2 e 4.3, é possível observar que a transição DI6 está habilitada e os botão 6 está habilitado.

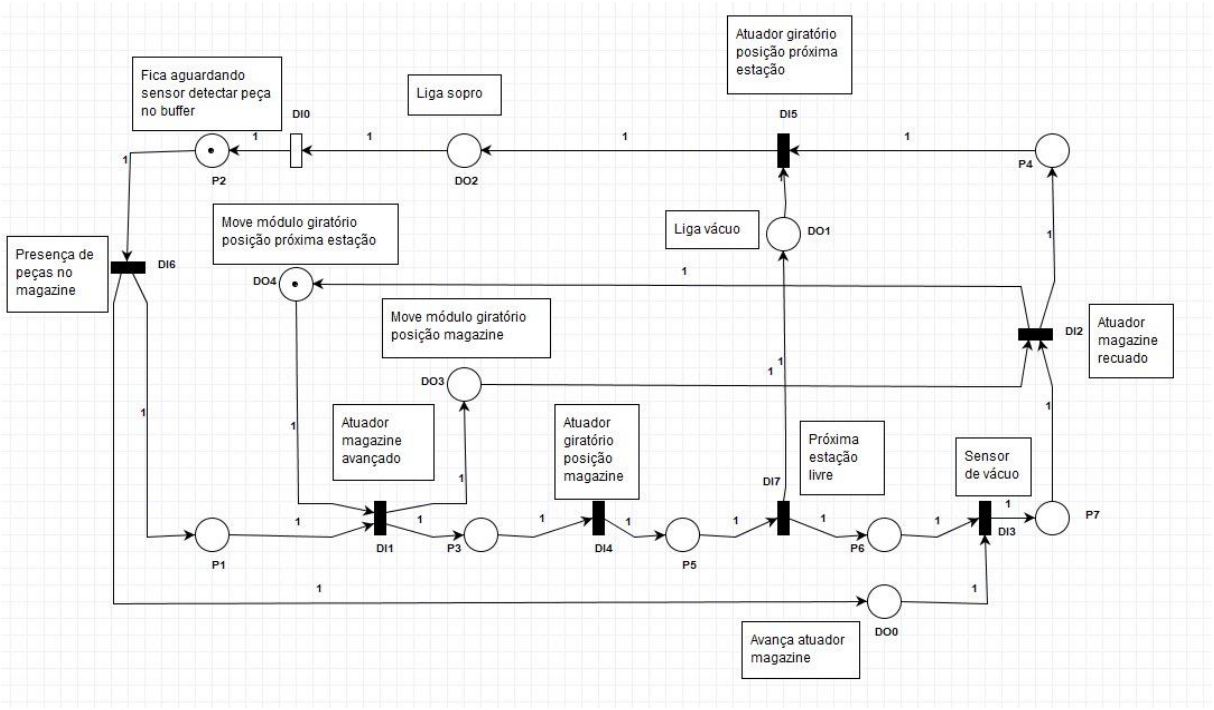

Figura 4.1 – Rede de Petri da lógica de controle da bancada *MPS®-Distributing* programada no PIPE

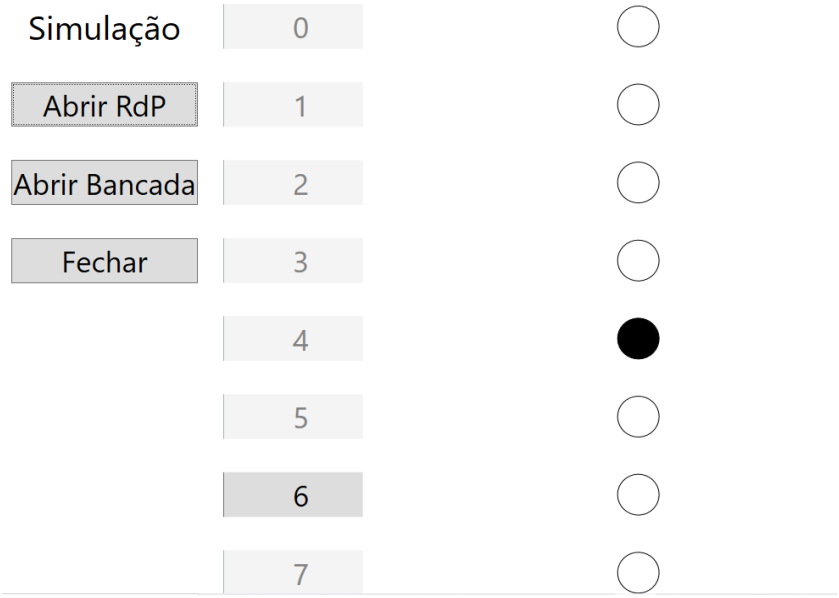

Figura 4.2 – Interface referente à Rede de Petri simulada

Como pode ser observado no PIPE existem marcas em DO0, DO2 e DO4 e o simulador mostra que os atuadores estão no primeiro, terceiro e quinto círculos.

Em seguida foi testado se o simulador conseguia identificar corretamente quais transições estavam habilitadas.

Comparando as Figura 4.3 e 4.4, é possível perceber que as transições DI0 e DI4 estão habilitadas e os botões 0 e 4 estão habilitados.

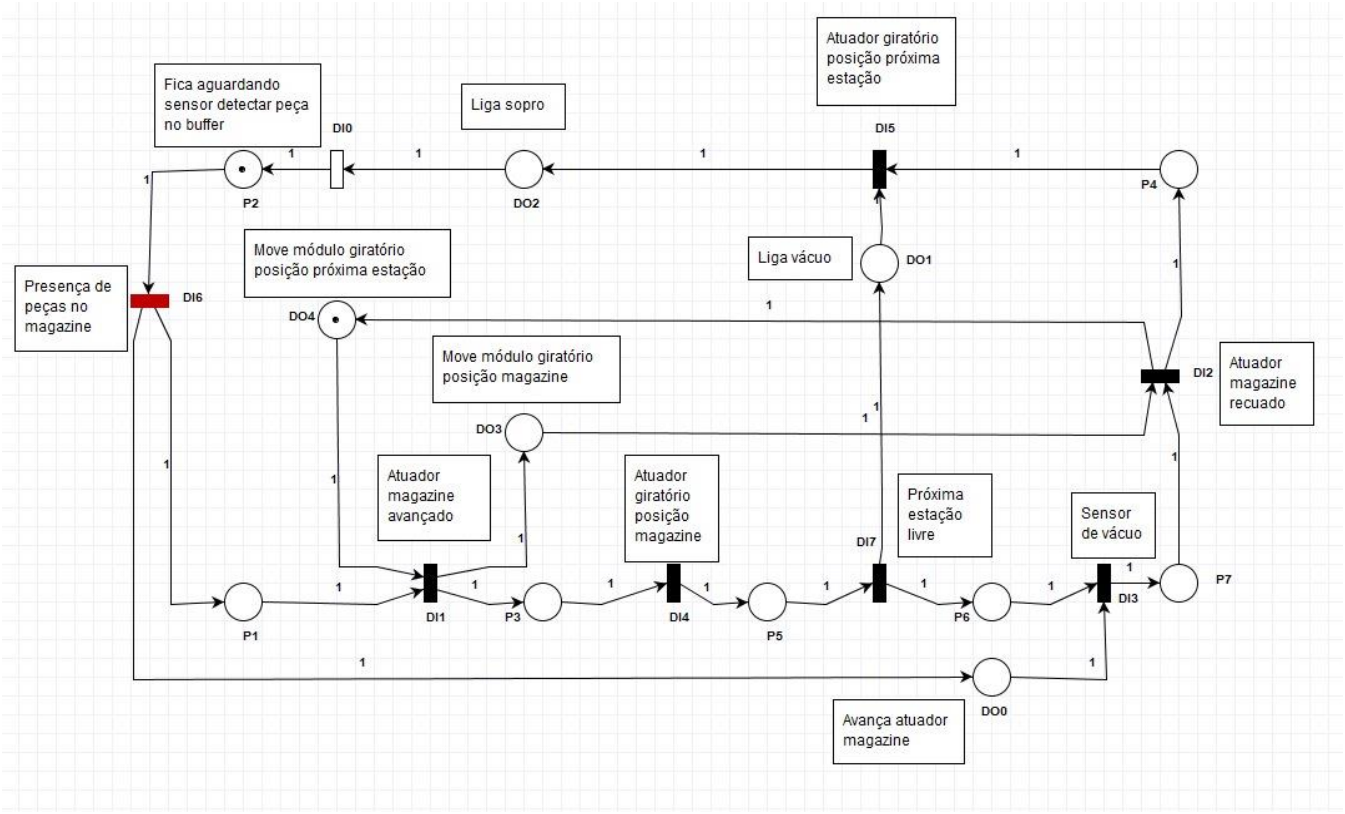

Figura 4.3 – Rede de Petri programada no PIPE com transições ativadas

Depois disso foi testado o disparo de uma transição para avaliar se o simulador estava atualizando corretamente a Rede de Petri. Comparando as Figura 4.4 e 4.5, observa-se que o disparo foi feito corretamente. As marcas passaram a estar em DO0 e DO4 na Rede de Petri, e no primeiro e quinto círculos no simulador. Da mesma forma as transições, DI1 está habilitada na Rede de Petri e os botão 1 está ativado no simulador.

Por fim foi testada a leitura do arquivo com as informações de entradas e saídas da bancada (Tabela 1).

Comparando a Tabela 1 e a Figura 4.6 observa-se que a tabela com as entradas e saídas foi lida corretamente de forma a atribuir os DIs às transições e os DOs aos lugares.

Esses testes mostram o simulador funcionando como esperado para uma bancada MPS da FESTO.

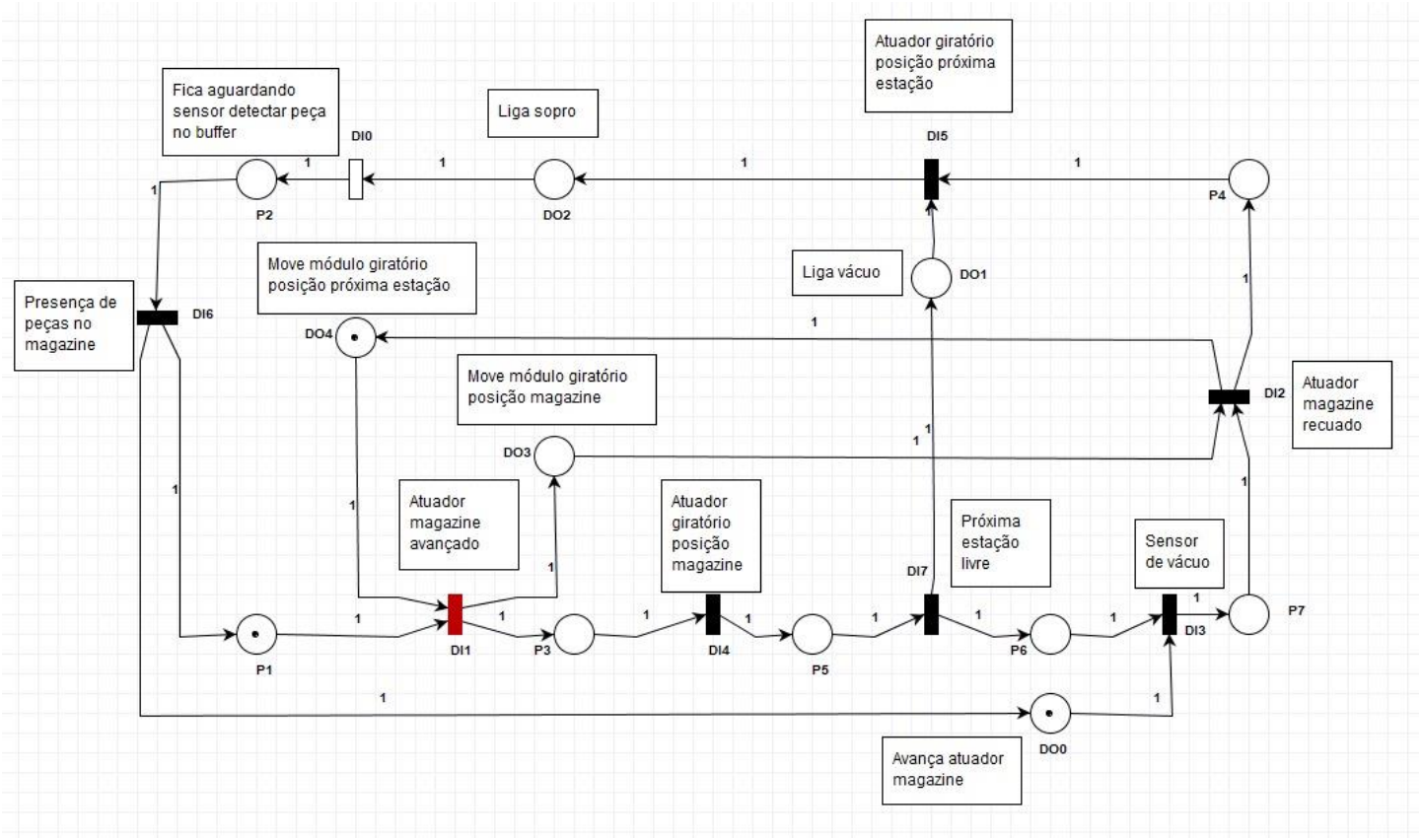

Figura 4.4 – Rede de Petri programada no PIPE após disparo de DI6

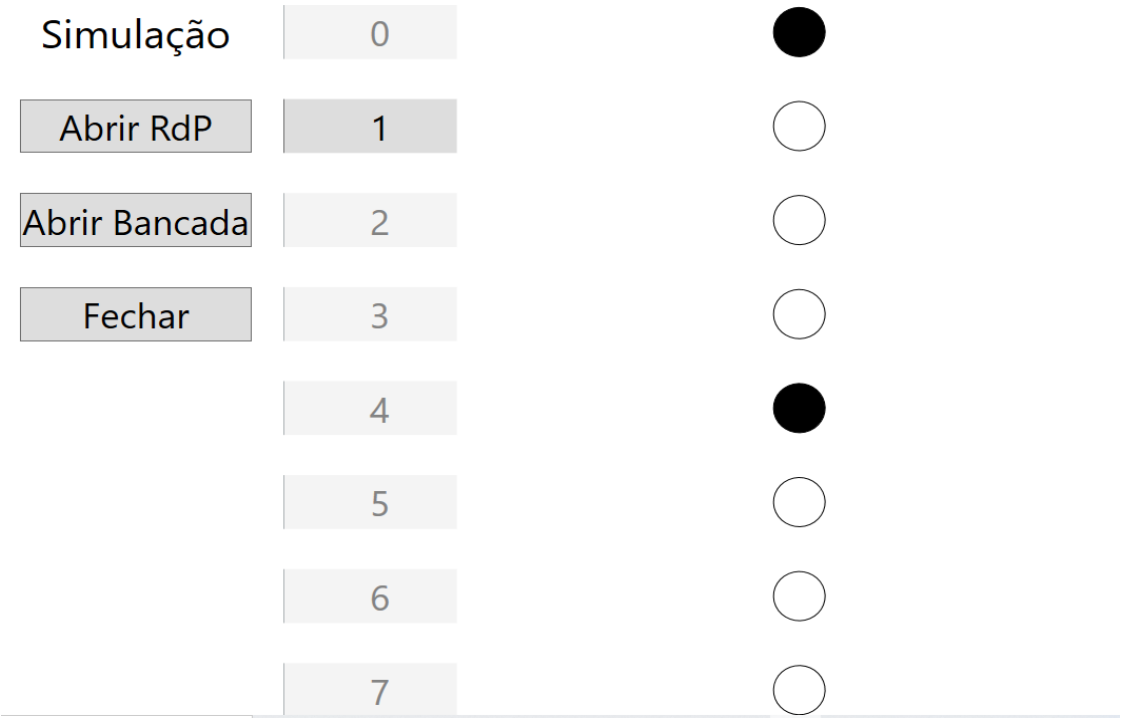

Figura 4.5 – Interface do simulador após disparo de DI6

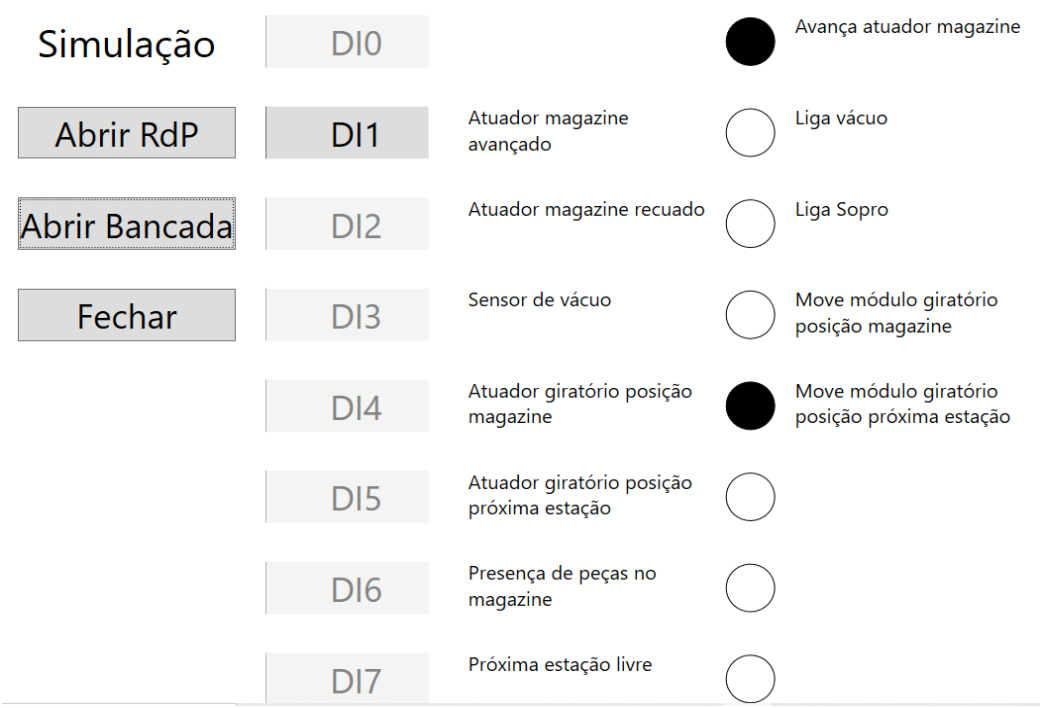

Figura 4.6 – Interface do simulador após ter sido carregado o arquivo com as informações de IOs

5. Conclusões

Seja por permitir uma melhor fixação de conteúdo entre os alunos, por meio da visualização ou por manter a classe mais motivada e participativa durante as aulas, os simuladores didáticos vêm sendo cada vez mais estudados e aprimorados, tomando seu espaço na comunidade científica de várias áreas.

São encontrados na literatura e citados na revisão bibliográfica deste trabalho, artigos comparando a eficácia de metodologias de simulação frente à didática clássica. É embasada nestas teorias que, em um período de pandemia em que as aulas presenciais se tornaram à distância, surge a ideia de produzir um emulador que permitiria simular uma bancada física, utilizada para fins didáticos, com o objetivo de diminuir as mudanças de conteúdo que esse novo modelo de aulas online exigiria.

Os testes realizados demonstram que o emulador consegue corretamente ler uma Rede de Petri e a associar a uma bancada MPS da FESTO. Além disso, o sistema consegue identificar corretamente uma simulação de sinal de sensor, atualizar os *status* dos atuadores e atualizar os sinais de sensores possíveis de serem simulados, permitindo que a bancada seja controlada por meio da interface e o aluno possa ter uma experiência mais próxima da que teria caso as aulas fossem presenciais.

Por permitir a leitura de diferentes Redes de Petri e diferentes arquivos de bancadas MPS, o sistema permite um espaço para testes de controle. Isto permite identificar se a Rede de Petri criada realmente é condizente com a bancada que se pretende controlar. Assim como permite experimentar com bancadas diferentes para poder treinar a modelagem de distintas lógicas de controle, baseadas em sistemas a eventos discretos distintos, para as bancadas.

6. Referências bibliográficas

AL-SHARIF, L.; TUTUNJI, T. A.; RAGAB, D. (2014). Using Elevator system modelling and simulation for integrated learning in mechatronics engineering. 15th International Workshop on Research and Education in Mechatronics (REM), pp. 1-8, IEEE.

DAVID, R.; ALIA, H. (2005) Discrete, Continuous, and Hybrid Petri Nets. Springer-Verlag Berlin Heidelberg.

GUOQIANG, C. (2010). Levels of application of computer simulation in engineering disciplines education. 2010 2nd International Conference on Education Technology and Computer, Vol. 2, pp. V2-117, IEEE.

HERADIO, R.; DE LA TORRE, L.; DORMIDO, S. (2016). Virtual and remote labs in control education: A survey. Annual Reviews in Control, Vol. 42, pp. 1-10.

HERNANDES, A. C.; TOMANARI, G. A. Y.; YASSUDA, M. S.; WENDLAND, Edson C.; FILHO, Tarcisio E. P. B.; COSTA, André L. (2020). Plano USP para o retorno gradual das atividades presenciais (Quarto Documento).

KLUG, M.; HAUSBERGER, P. (2009). Motivation of students for futher education in simulation by an applied example in a related other course in engineering education—A case study. Proceedings of the 2009 Winter Simulation Conference (WSC), pp. 248-255, IEEE.

KNOBEL, M. (2020). Comunicado CRUESP nº.03/2020.

KRESS, R.; CEMERLIC, A.; KRESS, J.; VARGHESE, J. (2010). Discrete event simulation class for engineering graduate students. Proceedings of the 2010 Winter Simulation Conference, pp. 344-352, IEEE.

LUCACHE, D. D.; GABOR, G.; DOSOFTEI, C. C. (2020). Integrated Learning in Electrical and Control Engineering Using an Educational Elevator System. 2020 International Conference and Exposition on Electrical And Power Engineering (EPE), pp. 744-748, IEEE.

POLA, D. F. S. R. (2012) Treinamento MPS_Rev 1.

SEMLER, M. W.; KERIWALA, R. D.; CLUNE, J. K.; RICE, T. W.; PUGH, M. E.; WHEELER, A. P.; MILLER, A. N.; BANERJEE, A.; TERHUNE, K.; BASTARACHE, J. A. (2015). A randomized trial comparing didactics, demonstration, and simulation for teaching teamwork to medical

residents. Annals of the American Thoracic Society, Vol. 12n N. 4, pp. 512-519.

SOLYMOS, O.; O'KELLY, P.; WALSHE, C. M. (2015). Pilot study comparing simulation-based and didactic lecture-based critical care teaching for final-year medical students. BMC anesthesiology, Vol. 15, N. 1, pp. 1-5.

SOUSA, M. L. (2011). Análise da eficácia do uso de simuladores computacionais no ensino de óptica. Universidade Federal de Uberlândia.

WEINTRAUB, A. (2020). PORTARIA Nº 343, de 17 de março de 2020.

ZURAW SKI, R.; ZHOU, M. (1995). Petri net and industrial application: A tutorial. IEEE Transactions on Industrial Electronics. Vol. 41, pp. 567 - 583. 10.1109/41.334574.