JOHN V. C. O. LOURENÇO, JOÃO P. S. FONSECA
UNIVERSIDADE FEDERAL DE GOIAS
GOIÂNIA, GOIÁS
E-MAILS: JOHNVITORCONSTANT@GMAIL.COM, JPAULOSFONSECA@GMAIL.COM

CONTROLE E MONITORAMENTO DE SISTEMAS ELETROPNEUMÁTICOS VIA REDE DE PETRI DE ALTO NÍVEL

Abstract — This work aims to create a communication interface between high level graphic language of CPN Tools, Petri Nets, with a real electro-pneumatic system. For that, a client/server architecture was developed for the Arduino UNO R3, in order to translate the information received through the network and transform it into read or write commands on I/O ports. A standardization of the commands sent by CPN Tools (server) was necessary, as well a creating a software with graphical interface capable of executing macros that help in typing these commands. After client and server properly configured, several electronic components were used, in order to adapt the voltages and currents to those supported by the microcontroller. It was also verified the limitations that the CPN Tools program had in its simulation tool, and some methods of circumventing these limitations were proposed. At the end of the work, the execution time of each code cycle present in the microcontroller was also calculated.

Keywords— Arduino, Petri Nets, system design, automation, electro-pneumatic.

Resumo— Avanços tecnológicos nos campos da internet das coisas, ou IoT, e da Indústria 4.0 requerem maior integração entre níveis de controle de alto e baixo níveis. Uma técnica conhecida por representar modelos comportamentais de alto nível é a rede de Petri, porém, normalmente sua lógica é traduzida para linguagens textuais e a sua aplicação, como linguagem gráfica, diretamente no controle de dispositivos ainda é um desafio. No presente trabalho procurou-se criar uma interface capaz de integrar redes de Petri de alto nível com um sistema eletropneumático real. Para isso, desenvolveu-se uma arquitetura cliente/servidor TCP/IP para um Arduino UNO R3 e o software CPN Tools, com a finalidade de traduzir as informações recebidas através da rede e transformá-las em comandos de leitura ou escrita das portas I/O do controlador. Realizou-se também uma padronização dos comandos enviados pelo servidor CPN Tools, assim como a criação de uma interface gráfica capaz de executar macros que auxiliam na digitação destes comandos. Após cliente e servidor devidamente configurados, utilizaram-se componentes eletrônicos, com a finalidade de adequar as tensões e correntes operacionais. Verificaram-se também as limitações que o software CPN Tools possuía em sua ferramenta de simulação, e foram propostos alguns métodos de contornar tais limitações.

Palavras-chave— projeto de sistemas, automação, eletropneumática, arduino, redes de Petri.

1 Introdução

A indústria, nos últimos tempos, procura métodos cada vez mais eficientes para aumentar sua produtividade, flexibilidade e segurança. Entende-se por automação qualquer sistema que utiliza computadores para substituir o trabalho humano em favor da segurança do operador, qualidade dos produtos, aumento da produção e redução dos custos. (MORAES & CASTRUCCI).

Com a necessidade de facilitar a automatização de processos industriais, surgiram diversos dispositivos computacionais, como o CLP e microcontroladores, que utilizam linguagens computacionais de alto nível. Apesar da existência de linguagens gráficas, como Ladder e SFC (exclusivas para CLPs), a programação de microcontroladores e placas de desenvolvimento modernas se dá, em grande parte, através de linguagens textuais, tais como IL, ST, Python e C. Nestes casos, há uma necessidade de integração entre uma linguagem gráfica e dispositivos de campo baseados em IoT.

As rede de Petri é uma linguagem de modelagem gráfica concebida em 1939 pelo matemático e cientista da computação Carl Adam Petri, e posteriormente submetida à sua dissertação de doutoramento na Faculdade de Matemática e Física da Universidade Técnica de Darmstadt na Alemanha em 1962.

Definida graficamente através de lugares, transições e arcos, suas principais aplicações são: modelar, simular e analisar sistemas discretos paralelos, concorrentes e assíncronos.

Devido a essas características, as redes de Petri se tornou uma ferramenta poderosa para modelagem, simulação e monitoramento de sistemas distribuídos discretos (MURATA, 1989), pois é possível modelar sistemas complexos graficamente, inserir eventos não previsíveis que podem atuar de maneira aleatória e simular todo o sistema.

Com base nestas características, um grupo denominado “CPN group” da Universidade de Aarhus, no ano 2000, criou uma ferramenta de análise, modelagem e simulação em rede de Petri de alto nível denominado CPN Tools (CPN TOOLS, 2017).

Este trabalho propõe a integração de modelos comportamentais, criados no CPN Tools, com dispositivos de campo baseados em IoT, para automatizar um sistema eletropneumático montado em laboratório.

O CPN Tools possui funções que tornam possível a sua comunicação com outras aplicações, fazendo-se possível a utilização da mesma ferramenta tanto para a modelagem quanto para o comando de sistemas baseados em eventos discretos, através de uma estrutura cliente-servidor.

O objetivo principal deste trabalho é criar um sistema que proporcione a interface de comunicação entre o software de modelagem e simulação CPN Tools e uma aplicação real, com a finalidade de traduzir as informações recebidas através da rede e transformá-

las em comandos de leitura ou escrita das portas I/O do microcontrolador baseado na placa de desenvolvimento Arduino UNO R3.

A utilização de plataformas Arduino em aplicações eletropneumáticas a nível de pesquisa acadêmica é amplamente reportada na literatura. Nunez et al. (2013), por exemplo, utilizaram Arduino NANO para projetar um sistema robótico pneumático utilizado na inspeção de eletrodutos. Souza et al. (2017), por sua vez, utilizaram arduino UNO para automação de um mecanismo dosador para sistema de empacotamento de produtos em pó e granulados. Bueno et al. (2017) utilizaram Arduino UNO para controlar o nivelamento de uma peneira acionada pneumaticamente. Já Dambroz et al. (2017) utilizaram uma placa Arduino para automatizar uma bancada alimentadora de peças composta por atuadores pneumáticos.

A escolha da placa de desenvolvimento Arduino UNO R3 justifica-se pelo fato de a mesma apresentar em seu núcleo o ATmega328p, um microprocessador da família MEGA ATmel, desenvolvido pela Microchip capaz de trabalhar a 20Mhz com até 23 portas I/O recebendo e enviando informações simultaneamente (ATMEL, 2017), além de ser compatível com o *Arduino Integrated Development Environment*, que, de acordo com McRoberts (2011) permite a criação de programas para a placa sem a necessidade de um programador, e utiliza uma variação da linguagem C desde que o ATmega328p tenha o *Arduino bootloader* instalado em sua raiz, facilitando sua programação.

2 Metodologia

Para a realização deste trabalho, baseou-se nos três primeiros níveis da pirâmide da automação, representada na figura 1. Adotando-se uma arquitetura cliente-servidor entre o *software* CPN Tools, instalado em um computador pessoal, e o dispositivo de campo, Arduino UNO R3, estabeleceu-se a comunicação entre tais processos através do protocolo TCP/IP. A figura 2 representa a relação entre a pirâmide e o método adotado.

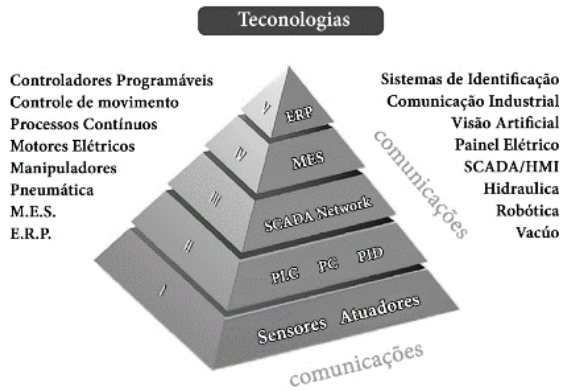

Figura 1. Pirâmide de automação. (Modificado de (Amazonaws 2017))

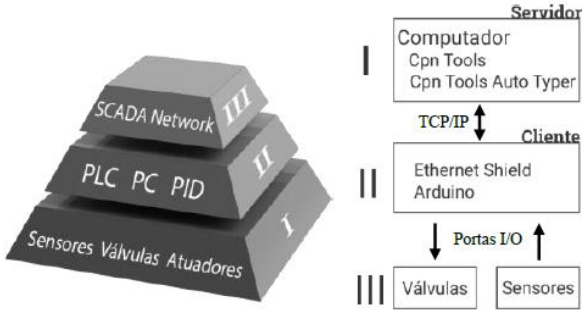

Figura 2. Relação entre a pirâmide da automação e a arquitetura proposta (PRIMEIRO AUTOR, 2017).

Admitindo-se um cenário composto por três atuadores de dupla ação, comandados por eletroválvulas mono ou biestáveis, e três sensores de fim de curso, definiu-se o diagrama eletropneumático apresentado na figura 3 como estudo de caso para este trabalho.

Para a comunicação entre o software e a bancada eletropneumática utilizou-se uma placa Arduino UNO R3 com um *shield* Ethernet W5100. O cliente executa uma lógica que associa os bytes recebidos do servidor com níveis de tensão correspondentes nas portas de saída do Arduino, assim como relaciona os níveis de tensão nas portas de entrada ao estado do sistema, enviando-os ao servidor CPN Tools.

Já o servidor é executado em uma máquina pessoal com os softwares “Cpn Tools Auto Typer” (PRIMEIRO AUTOR, 2017) e CPN Tools instalados.

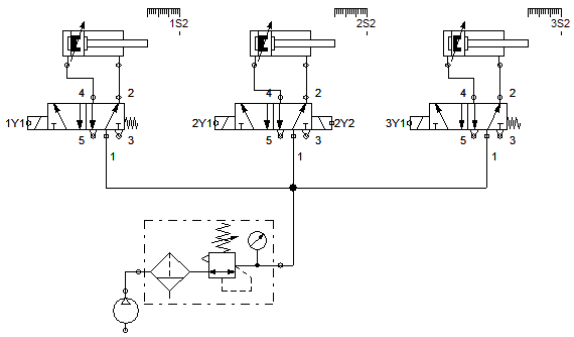

Figura 3. Diagrama eletropneumático simplificado para o estudo de caso admitido.

2.1 Configuração do CPN Tools

O CPN Tools utiliza a biblioteca COMMS/CPN com funções de conexão com processos externos. Cada função de comunicação é associada a uma transição, de modo que a função só será executada no momento em que a transição associada for disparada.

- Função “conectar”

Para se conectar ao cliente, utiliza-se a função “acceptConnection” que disponibiliza uma porta pela qual o cliente interessado pode estabelecer sua conexão. A figura 4 ilustra um exemplo de utilização da função “acceptConnection”.

"Conn 1" consiste no nome atribuído à conexão enquanto o argumento "9000" refere-se à porta utilizada para conexão.

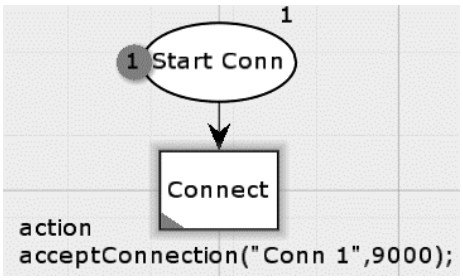

Figura 4. Função "acceptConecction" no CPN Tools.

- Função "enviar"

Para enviar determinada informação a algum processo externo, utiliza-se a função "send" em conjunto com o codificador "stringEncode". Um exemplo de como utilizar esta função está presente na figura 5, onde o argumento "Conn 1" refere-se ao identificador da conexão, a string "0 A+ C+" é a *string* que será enviada ao cliente, e o argumento "stringEncode" é o codificador utilizado.

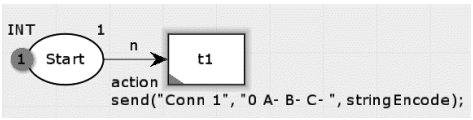

Figura 5. Função "send" no CPN Tools.

Para esta aplicação, na *string* "0 A+ C+", o termo "0" indica o início do pacote, o termo "A+" indica que a porta digital associada à eletroválvula A será energizada, e o termo "C-" indica que a porta digital relacionada à eletroválvula C será desenergizada.

Com base no diagrama da figura 3 e na estrutura da função "send", é possível criar a tabela 1, relacionando cada *substring* presente na *string* do CPN Tools com as eletroválvulas do sistema real.

Tabela 1 – Relação entre substring e estado dos pinos I/O.

Substring (CPN Tools)	Pino	Estado do pino/*ação*
A-	A0	ALTO
A+	A0	BAIXO
B-	A1	BAIXO
B+	A3	BAIXO
C-	A2	ALTO
C+	A2	BAIXO

- Função "receber"

Para receber determinada informação do cliente, é necessário definir a variável que irá receber a informação, a variável que irá sair da informação recebida e entrar no arco, o nome da conexão e o decodificador que será utilizado. Um exemplo de como utilizar a função receber está presente na figura 8, onde s é a variável de saída da transição, x é a variável onde será armazenado o valor recebido, "Conn 1" é o nome da conexão que está disponibilizando os dados e "stringDecode" é a função decodificação utilizada.

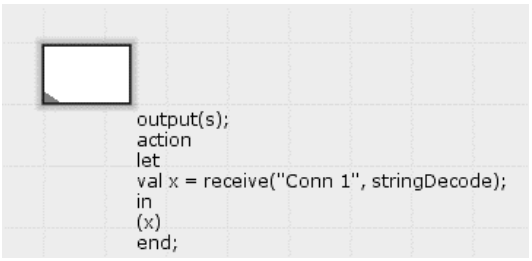

Figura 8. Função "receive" no CPN Tools.

É possível também estabelecer uma relação entre cada sensor do sistema real e cada *string* que o CPN Tools irá receber quando o cliente enviar informações pela conexão estabelecida. Esta relação está descrita na tabela 2.

Tabela 2. Relação entre o estado do sensor e a *string* s.

Sensor	Pino I/O	Leitura do Pino I/O	string s
1	3	0	1-
		1	1+
2	5	0	2-
		1	2+
3	9	0	3-
		1	3+

2.2 Cpn Tools auto Typer

Como o CPN Tools utiliza uma linguagem de modelagem pouco conhecida por operadores, optou-se pela implementação de um programa com uma interface gráfica mais amigável para facilitar sua utilização. Para isso, utilizou-se a ferramenta de criação de programas, macros e scripts denominada Auto Hotkey, para criar uma aplicação cuja a interface está apresentada na figura 9.

Esta aplicação facilita a inserção dos comandos nas transições, escrevendo as funções "conectar", "receber" ou "enviar", automaticamente na área de função da transição. Para utilizá-lo, com CPN Tools aberto, o usuário deve segurar as teclas ctrl e shift, e clicar com o botão esquerdo na transição desejada que o software se abrirá. Uma vez aberta, há três formas de se utilizar essa ferramenta: criar uma transição que irá comandar os atuadores, habilitar conexão ou ler informações dos sensores.

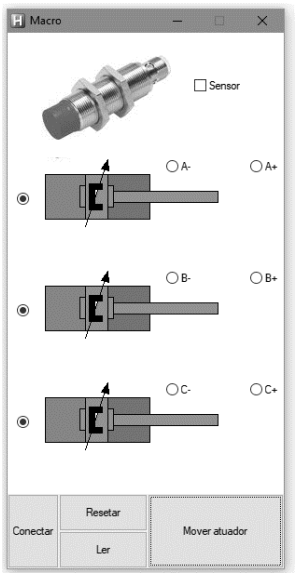

Figura 9. Interface do software Cpn Tools auto Typer (PRIMEIRO AUTOR, 2017).

Para criar a transição que comanda a ação dos atuadores, o usuário deve escolher se deseja ou não solicitar o monitoramento dos sensores, a posição de cada atuador após o disparo daquela transição, e logo em seguida clicar em "Mover atuador". Se o usuário desejar apenas retrair todos os atuadores quando a transição selecionada for disparada, basta utilizar o botão "reset"

Para criar uma transição que irá conectar o CPN Tools ao cliente, basta utilizar o botão "Conectar".

Para criar uma transição que irá coletar os dados dos sensores recebidos do cliente, basta utilizar o Botão "Ler".

2.4 Bancada eletropneumática

Uma vez configurados cliente e servidor, partiu-se para a montagem da bancada eletropneumática. O desenho esquemático do circuito pneumático utilizado pode ser observado na figura 10.

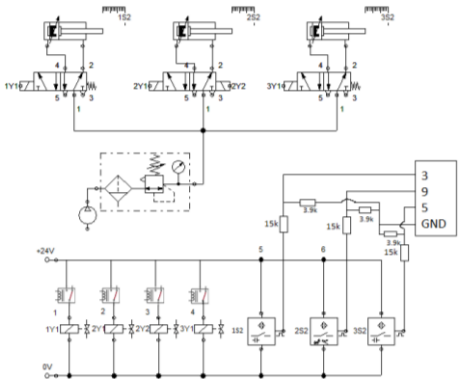

Figura 10. Esquemático da bancada

A figura 11 demonstra a bancada eletropneumática montada e a fotografia da figura 12 detalha as ligações presentes no Arduino.

.

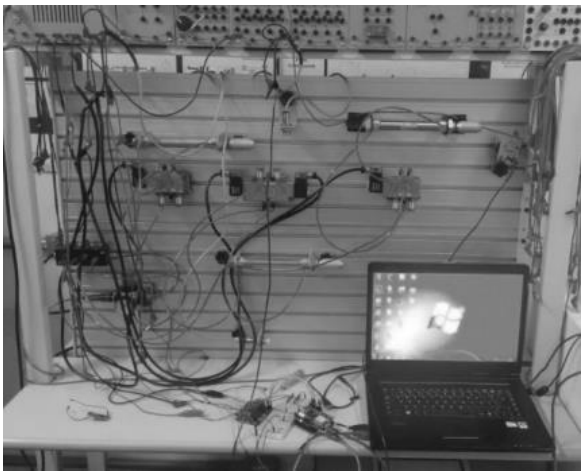
Figura 11. Bancada eletropneumática de testes montada

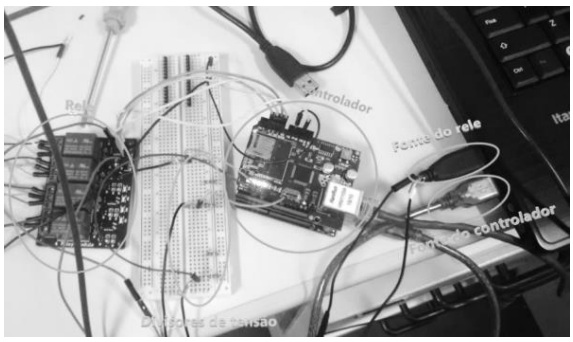

Figura 12 Componentes eletrônicos montados para o experimento

3 Limitações

Com o sistema montado, é necessário criar um modelo em redes de Petri utilizando o CPN Tools. Esta seção apresenta as principais limitações da utilização do software para controle em tempo real, bem como como o método utilizado para contorná-las.

3.1 Processos Paralelos

Uma limitação encontrada foi com relação à execução de sistemas paralelos, o CPN Tools executa 1 transição por vez, sendo impossível executar processos paralelos como demonstrado na figura 13.

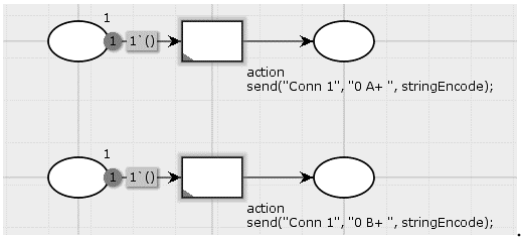

Figura 13. Exemplo de rede Concorrente Simultânea.

Uma forma de "contornar" este problema é atribuir mais de um comando para a mesma transição, como mostrado na figura 14. Porém, vale ressalvar que este método pode não ser apropriado dependendo

do sistema a ser modelado, pois o operador está somente .

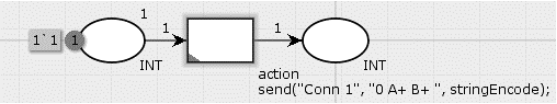

Figura 14. Movimentando mais de um atuador.

3.2 Tempo de Leitura dos Dados Recebidos do Client

Com relação ao tempo de leitura, para entrar na "subrede" de leitura é necessário 1 unidade de tempo, e cada sensor lido gasta 1 unidade de tempo para ser processado e sua marca de *string* ser posicionada em seu lugar. Portanto no sistema testado, onde haviam 3 sensores, eram gastas 4 unidades de tempo para efetuar a leitura do sensor. E nesse intervalo em que a o CPN Tools está efetuando a leitura do cliente, todas as demais funções ficam bloqueadas, o que forçou também a utilização do byte '1' no cabeçalho da função presente na transição "ativa A check" para chamar essa leitura. Uma "subrede" de leitura de sensores pode ser visualizada na figura 15.

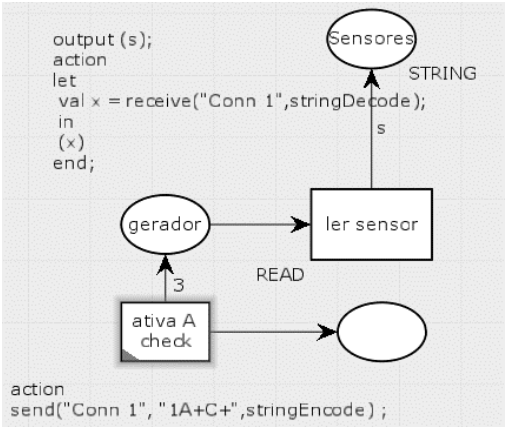

Figura 15. Rede para leitura de sensor.

3.3 Tempo

Uma das maiores limitações presente no sistema, é com relação ao tempo. Cada transição dispara obrigatoriamente após um tempo previamente estabelecido na simulação. Este tempo é o "*delay in milliseconds*" presente no botão "*Execute the specified number of transitions, showing the intermediate markings"* da barra de simulação, como pode ser visto na figura 16.

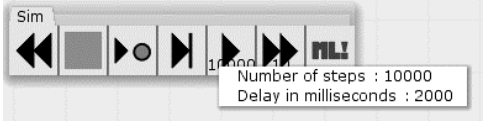

Figura 16. Barra de simulação do CPN Tools.

Com isso em mente, utilizou-se um bloco contador para conseguir contornar essa ausência de um tempo real no programa. A utilização do bloco contador pode ser observada na figura 17.

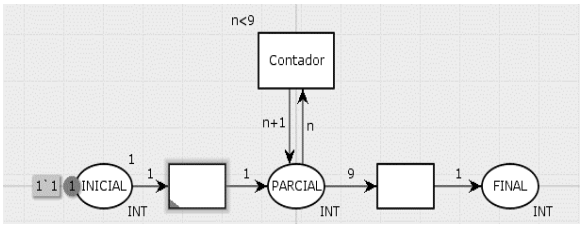

Figura 17. Bloco contador

No exemplo da figura 17, em t = 0 a marca estará em sua posição "inicial", em t = 1 unidade de tempo, ela entrará na posição "parcial", na qual permanecerá durante 10 unidades de tempo para então ir para a posição "final".

4 Execução

O modelo da figura 19 foi criado para o sistema indicado na figura 10, procurando obter o diagrama trajeto passo da figura 18, considerando-se as limitações abordadas na seção 3.

Modificações em tempo de execução demonstraram-se possíveis, mas instáveis, sendo indicada a interrupção da execução antes de proceder qualquer modificação da rede.

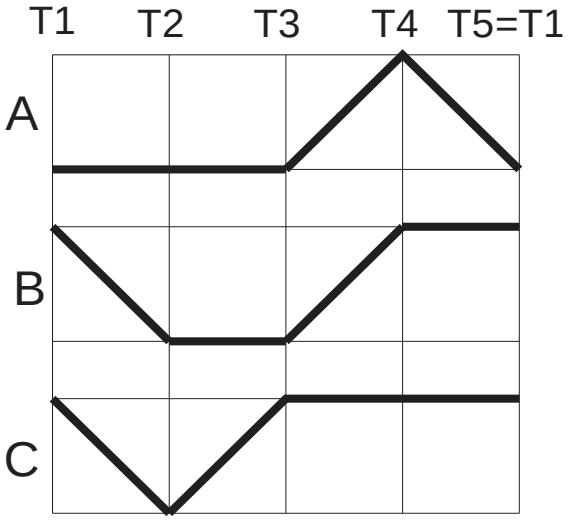

Figura 18. Trajeto passo I

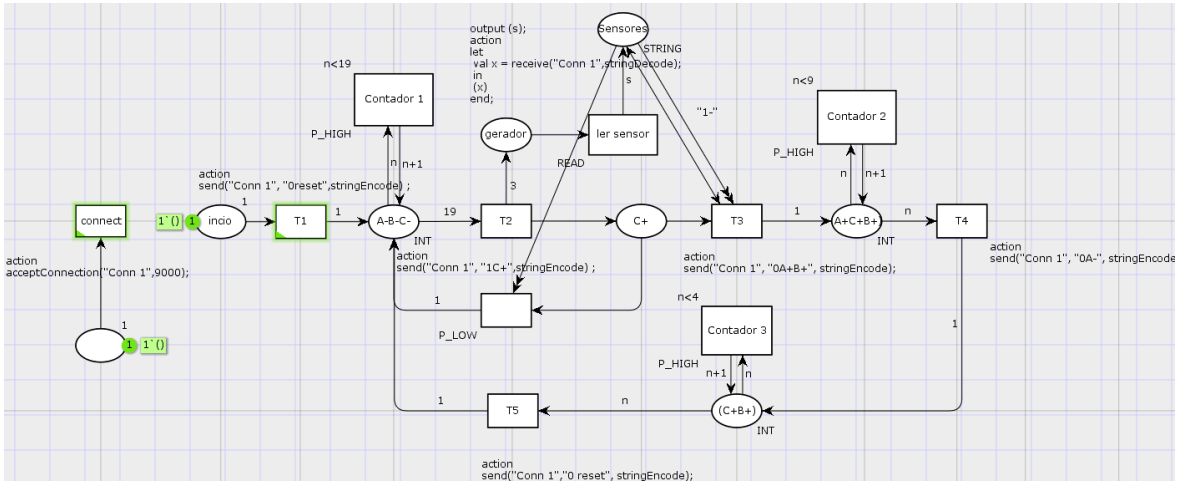

Figura 19. Rede de Petri I

Na figura 19, excluindo-se a subrede de conexão e o primeiro lugar/transição, há 4 lugares (*places*) e 5 transições na rede de Petri "principal", composta pelas transições T1, T2, T3, T4 e T5 e pelos lugares "A- B- C- ", "C+", "A+ C+ B+", "C+ B+ ". A primeira transição possui em sua *string* o comando "1 C+", representando que após o disparo daquela transição o atuador C deve ir para o final de seu curso, o caractere "1" desta mesma transição indica que os sensores deverão ser monitorados e seus dados disponibilizados pelo cliente. Uma vez executada essa transição, as marcas avançam para o próximo lugar, onde as mesmas deverão esperar a execução de toda "subrede" de leitura de sensor. Uma vez executada, 3 marcas contendo *strings* relacionadas ao estado dos sensores são inseridas no lugar denominado "Sensores". Caso essas *strings* estiverem corretas, ou seja, se o sistema atingir o estado esperado, então a rede avança um passo para a direita, acionando os atuadores A e B e removendo todas as marcas do lugar "Sensores". Caso algum sensor estiver com o estado inesperado, então a marca volta pela transição vazia e tenta executar a transição T2 novamente.

Em seguida, a marca com informação de posição entra no lugar (*place*) onde existe um contador, o qual conta 9 unidades de tempo e na 10ª unidade de tempo, avança novamente pela rede de Petri, para próxima transição, que é uma transição com a *string* "0A-", para então entrar em outro lugar com outro contador, este contador começa a contar e assim que completa 4 unidades de tempo a marca está pronta para avançar na rede de Petri, na 5ª unidade de tempo a transição com a *string* "0 reset" é acionada e todo o sistema volta para sua posição original, recomeçando o ciclo.

Após esse teste inicial do sistema, foram analisados novos cenários, com a finalidade de testar a funcionalidade do sistema criado. Então propôs-se alguns trajetos-passos e criou-se uma rede de Petri para cada, as figuras 20, 21, 22, 23, 24 e 25 demonstram cada trajeto passo assim como a rede de Petri necessária para se obter esse trajeto-passo.

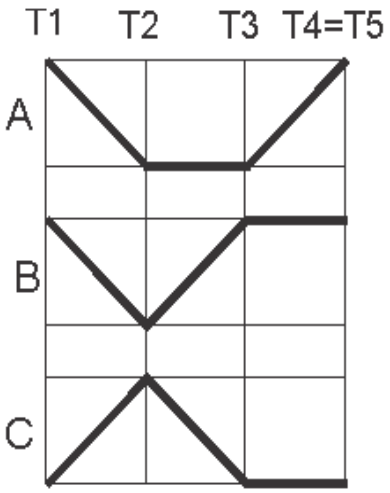

Figura 20. Diagrama trajeto passo da rede de Petri II

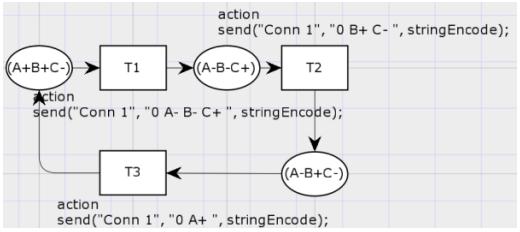

Figura 21. Rede de Petri II.

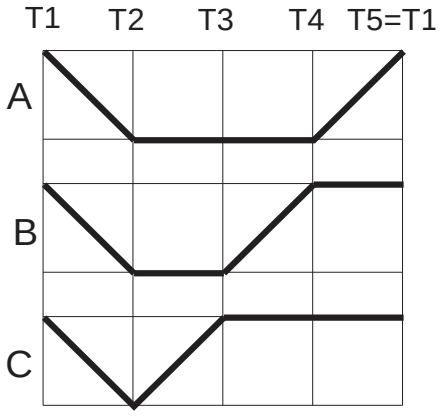

Figura 22. Diagrama trajeto-passo da rede de Petri III

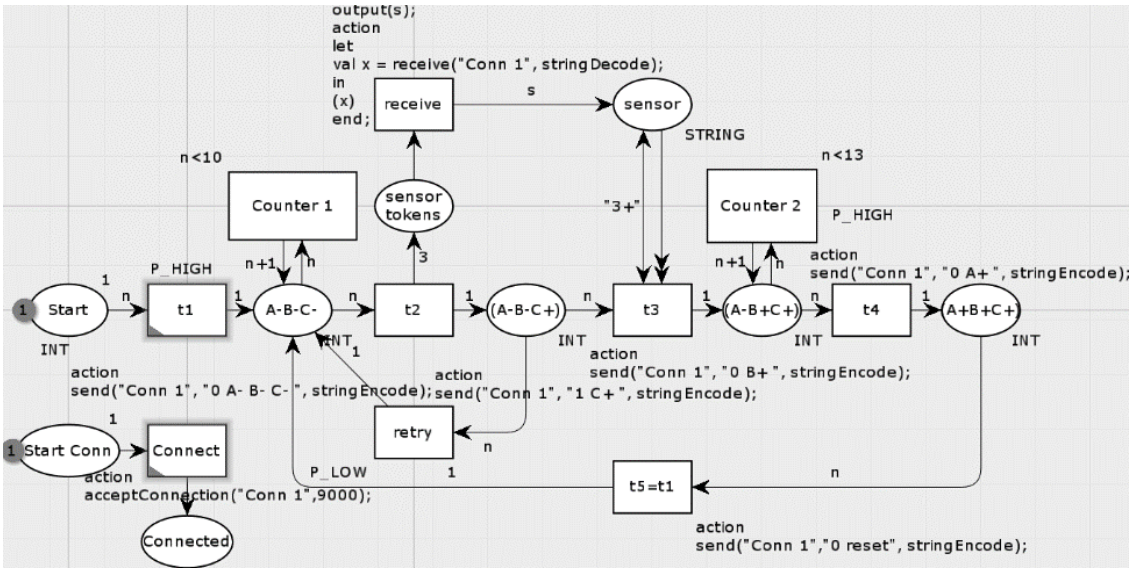

Figura 23. rede de Petri III

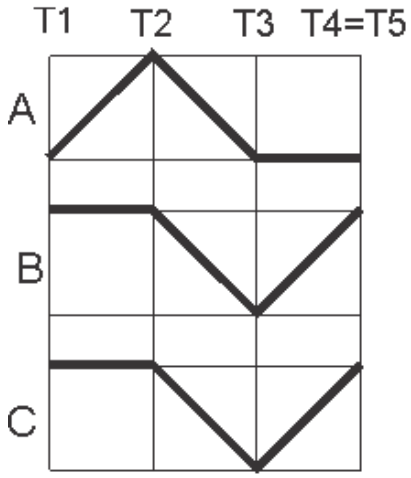

Figura 24. Diagrama trajeto-passo da rede de Petri IV

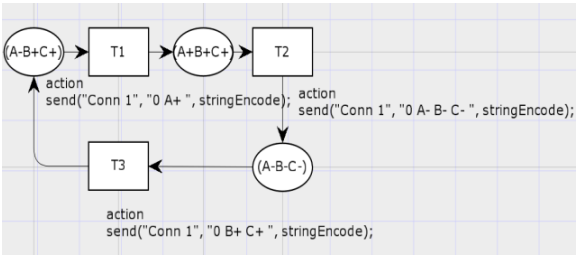

Figura 25. Rede de Petri IV

4.1 Tempo de processamento

Verificar o tempo de processamento também é muito importante neste trabalho, saber se a simulação está realmente sendo executada em tempo real ou se há um atraso crítico é importante na hora de modelar sistemas dinâmicos.

Para realizar esta análise, utilizou-se um Arduino Nano para contabilizar o tempo de operação . A figura 26 demonstra como as 2 placas foram conectadas.

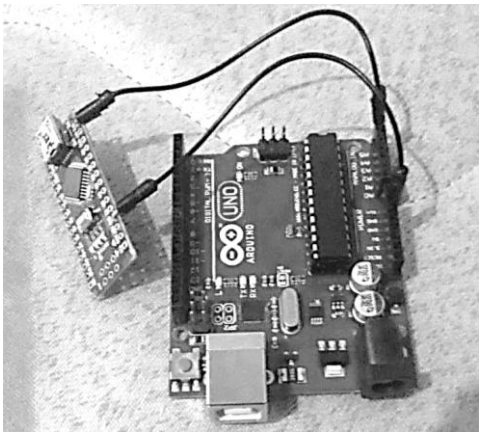

Figura 26. Arduino nano ligado ao arduino uno

Após 3 minutos rodando o programa com n = 1000 ciclos e depois com n = 10000 ciclos, encontrou-se um tempo médio de execução por ciclo de 0.49 ms. Cada caractere da *string* gasta 1 ciclo para ser processada, resultando em 3 ciclos para processar cada comando de válvula (3 caracteres). A figura 26 demonstra de maneira esquemática o tempo gasto pelo código para energizar ou desenergizar todos os 3 atuadores.

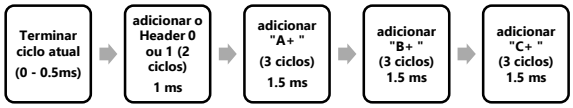

Figura 27. Tempo de execução

Observando a figura 27, verifica-se, que, na pior das hipóteses, o sistema gastará 6 milissegundos para alterar o nível de tensão de todas as portas I/O de saída. Somando esses 6 milissegundos com os 77 milissegundos de *delay* da combinação rele com válvula, resultam num total de 83 milissegundos de *delay* total no sistema. Estes resultados demonstram que o tempo de execução do código é apenas 7% do tempo total de execução do sistema.

5 Conclusão

Este trabalho demonstrou que é possível e perfeitamente viável modelar, simular, atuar e verificar todo o sistema eletropneumático utilizando uma única ferramenta para modelagem e execução, neste caso o CPN Tools. Uma vez montada, modificações na lógica de operação do sistema eletropneumático passa a depender apenas de modificações na rede de Petri para realizar tal alteração do sistema, sem necessidade de alterar a estrutura física dos componentes, como no método clássico que utiliza lógica de relés.

O custo de criação deste sistema também é outro diferencial, o "*Client"* (Arduino UNO R3, Ethernet Shield W5100 e rele 4 vias) possui um custo estimado de aproximadamente R$100,00, enquanto o servidor necessita apenas de um computador, que consiga executar o CPN Tools, ligado à mesma rede do *Client*.

Apesar das vantagens descritas anteriormente, não é possível afirmar a viabilidade da utilização deste trabalho em aplicações que requerem tempos precisos, uma vez que não foi feito um estudo relacionando o tempo de simulação do CPN Tools e o tempo real, portanto fica aqui proposto um estudo quantitativo e qualitativo, principalmente relacionado ao tempo, da utilização deste trabalho em outros estudos de caso.

Outro problema que pode acontecer relacionado ao tempo é o chamado "*millis overflow*". Este efeito acontece quando o número que contabiliza a quantidade de tempo em que o programa está em execução ultrapassa o limite de bits, este efeito acontece no Arduino quando o mesmo fica em execução ininterrupta por aproximadamente 49 dias. Ou seja, para utilizar este trabalho em um sistema que precisa funcionar por 49 dias ou mais, seria necessária uma modificação no código ou a utilização de um módulo RTC (*Real Time Clock*) para contornar o problema.

Fica proposto também a aplicação das redes de Petri na modelagem de outros sistemas discretos, assim como um refinamento na metodologia utilizada neste trabalho para contornar determinadas limitações ou controlar outros sistemas a eventos discretos.

6 Agradecimentos

Aos professores da Universidade Federal de Goiás, pelos conhecimentos transmitidos e por instigar o pensamento crítico, político, criativo e social.

Aos técnicos dos Laboratório da Universidade Federal de Goiás.

7 Referências bibliográficas

ATMEL (2017). ATmega 328/P Datasheet <http://www.atmel.com/Images/Atmel-42735-8-bit AVR-Microcontroller-ATmega328-328P_Datasheet.pdf> Acesso em 1 de dezembro de 2017.

BUENO (2017); Implementação de controle automático para o auto nivelamento de peneiras

CPN Tools (2017). <http://cpntools.org/> Acesso em 1 de dezembro de 2017

DAMBROZ et al (2017); Projeto assistido por computador de uma bancada automatizada para alimentação de peças

JENSEN, K. AND KRISTENSEN, L. M. (2001). Comms/CPN: A Communication Infrastructure for External Communication with Design/CPN

Constant (2017). Controle e Monitoramento de Sistemas Eletropneumáticos via Rede de Petri de Alto Nível

MCROBERTS; M. Arduino Básico, 1st ed São Paulo: Novatec, 2011

MURATA; T. (1989). Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4):541–580.