SIMULADOR DE DISPOSITIVOS INDUSTRIAIS QUE UTILIZAM CÓDIGO PORTÁVEL

Guilherme Resende Ferreira, guilherme.ebio@hotmail.com[1]
Marcio José da Cunha, cunhamjg@gmail.com[1]
Gilson Fonseca Peres Filho, gilsonfonseca@yahoo.com.br[1]
Aniel Silva de Morais, aniel@eletrica.ufu.br[1]
Fábio Vincenzi Romualdo da Silva, fabiovince@gmail.com[1]

[1] Universidade Federal de Uberlândia – UFU, Faculdade de Engenharia Elétrica- FEELT, Núcleo de Pesquisa em Controle e Automação – NPCA, Av. João Naves de Ávila, 2121 Bloco 3N,Uberlândia, MG, Brasil

Resumo: *Este trabalho propõe a implementação de um simulador que fará uso de um conjunto de funções executadas por uma máquina virtual Lua. Este conjunto de funções está sendo desenvolvido em um trabalho em paralelo e faz parte de uma proposta para a redução da complexidade e do custo além de favorecer a implementação de controles personalizados em determinados equipamentos industriais. Assim, o simulador proposto permitirá a acadêmicos e fabricantes do ramo de automação e controle compreenderem e aperfeiçoarem novos equipamentos, testarem novos controles, desenvolverem e simularem novos protocolos, realizarem treinamentos, dentre outras aplicações.*

Palavras-chave: *simulador, conjunto de funções, máquina virtual Lua, controles personalizados, treinamento*

1. INTRODUÇÃO

Ao longo da evolução tecnológica na área de informática, os computadores adquiriram uma crescente capacidade de processamento. Consequentemente, os engenheiros passaram a ter em mãos uma gama cada vez maior de programas e ferramentas para analisar e simular problemas (Pereira et al., 2012). Assim, o uso de simuladores está presente nas mais diversas áreas do conhecimento (Pereira et al., 2012; Reis, 2002), sendo ferramentas importantes no treinamento de pessoas, tanto no meio acadêmico quanto industrial, pois ajudam a evitar acidentes em ambiente de risco e reduzem desperdícios diretos e indiretos (Dias, 2010).

São muito utilizados em casos de aprendizado e treinamento onde, muitas vezes, a obtenção do objeto a ser simulado é difícil ou inviável (Okada et al., 2010), como ocorre no setor aeroespacial, em cirurgias de risco, na área de energia atômica, dentre outras áreas (Almeida et al., 2004).

No ambiente industrial, auxiliam na redução dos custos, tempo e riscos envolvidos ao se alterar processos industriais. A simulação permite então a experimentação de novas situações de forma segura, econômica e rápida, permitindo uma grande proximidade com o sistema real (Monteiro, 1991). Nas entidades de ensino, como universidades e escolas de treinamento técnico especializado, onde o custo dos equipamentos é elevado e há uma dificuldade na aquisição dos mesmos, os simuladores são ferramentas fundamentais.

Especificamente dentro da área de redes industriais, alvo deste trabalho, a grande gama de equipamentos e padrões diferentes dificulta em muito as entidades a obterem laboratórios bem equipados para propósitos de ensino e pesquisa, e com o uso de simuladores não seria necessária à aquisição de todos os equipamentos, facilitando a capacitação de novos profissionais e o desenvolvimento de novas pesquisas.

Atualmente, há um grande número de padrões abertos de tecnologias para automação tais como Profibus, Fieldbus Foundation (FF), DeviceNet, EtherNet/IP, dentre outros (Gaj et al., 2013; Sauter, 2010; Shahraeini et al., 2011). Embora tais padrões apresentem interoperabilidade, onde há comunicações entre equipamento de fabricantes diferentes, intercâmbialidade, onde um dispositivo pode ser substituído por outro equivalente de outro fabricante, e unificação, onde pode haver comunicação entre diferentes protocolos de rede através de gateways (Panton, et al., 2007), existem particularidades na configuração e programação dos dispositivos industriais. Não existe, portanto, portabilidade de programação e configuração entre os diversos dispositivos industriais construídos por fabricantes diferentes. Esta situação dificulta a formação técnica especializada no sentido de realizar manutenção e expansão de processos industriais e contribui para que o usuário torne-se refém da tecnologia adquirida.

Embora o FF permita alguma customização através de alguns blocos padronizados, como AI, AO, DI, DO, MDI, MDO, etc (Foundation, 2008), eles são limitados em capacidade e velocidade, geralmente exigindo controladores mais poderosos como *linking devices* e PLCs.

A partir do exposto, este trabalho propõe o desenvolvimento de um simulador que contempla uma nova abordagem em redes industriais. Tal abordagem utiliza como componente principal um núcleo desenvolvido com um controlador de código comum (Núcleo 3C) que está sendo desenvolvido em paralelo com este trabalho. Este núcleo tem como objetivo proporcionar os seguintes benefícios:

1) Tornar a programação e configuração de dispositivos industriais portável;
2) Reduzir o esforço de interpretação de padrões e de testes por parte dos fabricantes;
3) Diminuir o custo do hardware do dispositivo;
4) Possibilitar a implementação de controles diversos.

Neste sentido, o objetivo deste trabalho é criar um simulador capaz de utilizar o Núcleo 3C de modo a proporcionar ao usuário um ambiente compatível com aquele que encontraria em um ambiente industrial se estivesse utilizando equipamentos com Núcleo 3C. O simulador proposto irá, deste modo, simular a nova abordagem de hardware e software provenientes do trabalho que está desenvolvendo o mencionado núcleo.

Para o desenvolvimento do simulador optou-se pelo uso da linguagem de programação C++ através do framework e IDE Qt. Esse framework foi escolhido por ser facilmente incorporado em multi plataformas, pela capacidade de criação de ricas interfaces em conjunto com códigos rápidos em C++.

Por fim, pretende-se, ao final do projeto, disponibilizar um código aberto, para que a ferramenta proposta possa possibilitar a pesquisadores e empresários realizarem pesquisa e desenvolvimento de dispositivos industriais e a educadores realizarem treinamento com investimento e risco reduzido.

2. METODOLOGIA

A metodologia adotada no desenvolvimento do software foi a XP (extreme programming), pertencente ao conjunto de metodologias ágeis de desenvolvimento, por oferecer elevada capacidade de resposta a mudanças e ser mais adequada para desenvolvimentos com requisitos vagos.

Figura 1. Modelo de desenvolvimento por testes (XP)

Conforme a Fig. 1, nessa metodologia é realizada uma fragmentação dos objetivos do software. Cada fragmento passa pelos processos de planejamento, projeto, codificação e testes, onde todas essas etapas podem e devem ser modificadas conforme a necessidade.

Após cada fragmento passar por uma série de testes, caso for necessário, os fragmentos são integrados e testados como subconjuntos de fragmentos. Em cada etapa de testes são gerados pequenos releases funcionais, que aos poucos vão contemplando todos os objetivos do software. Somente no final ocorre o teste de integração e validação do sistema por completo.

3. DIAGRAMA DE BLOCOS DO SOFTWARE PROPOSTO COM EXEMPLO DE APLICAÇÃO

A Fig. 2 mostra o diagrama de blocos geral do simulador proposto. Ele é composto pela Interface Gráfica do Usuário (GUI) ligada a um barramento virtual que por sua vez se liga ao comunicador. A interface permite ao usuário acrescentar ou excluir os elementos que irá simular.

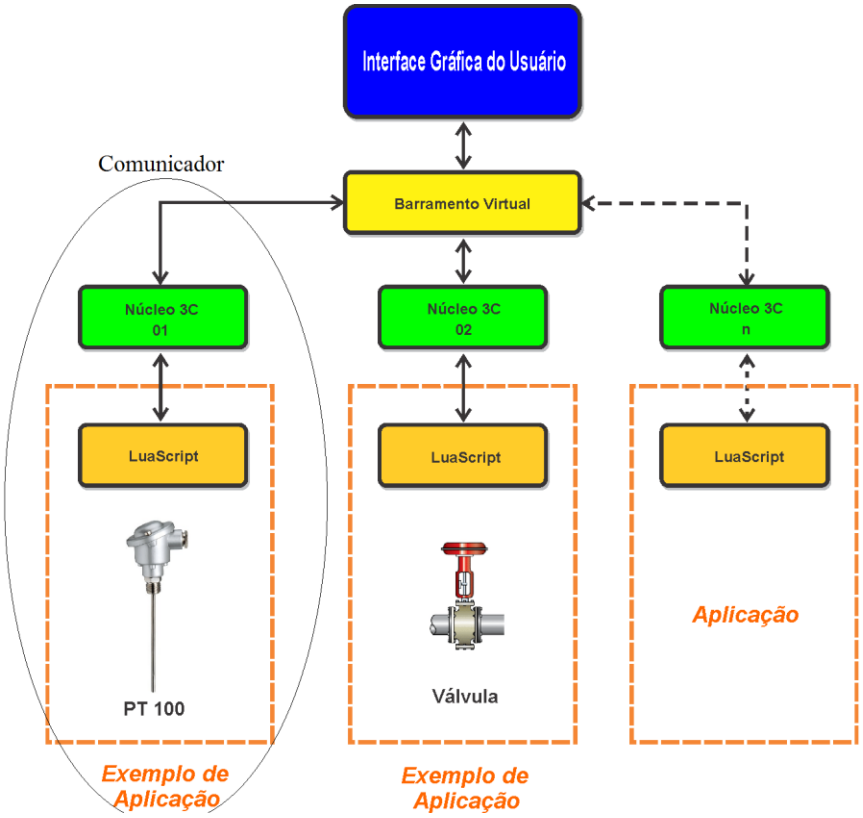

Figura 2. Diagrama de Blocos do Simulador Proposto.

À medida que um elemento de simulação é criado, é gerado automaticamente um novo comunicador (Composto pelo núcleo 3C e o LuaScript que será executado) que é ligado a uma porta TCP que utilizará para se comunicar com o barramento virtual. Deste modo, toda a troca de informação entre a interface de usuário e os demais Núcleos 3C ocorre por meio de sockets. A Fig. 1 demonstra um exemplo onde o comunicador executa o núcleo 3C 01 que, por sua vez, executa um LuaScript que simula um PT 100. Concomitantemente, outro comunicador executa o Núcleo 3C 02 que por sua vez executa um LuaScript que simula uma válvula. Assim, N comunicadores podem ser adicionados ao barramento virtual, cada um simulando um objeto diferente rodando seu próprio código.

A Fig. 3 apresenta o diagrama de Blocos do núcleo 3C proposto que é composto pelos blocos: LuaVM, Lua Manager, HAL (Hardware Abstraction Layer), Net Services, Net port e OS port. Estes blocos fazem parte do código comum que deve ser implementado para se obter os benefícios propostos.

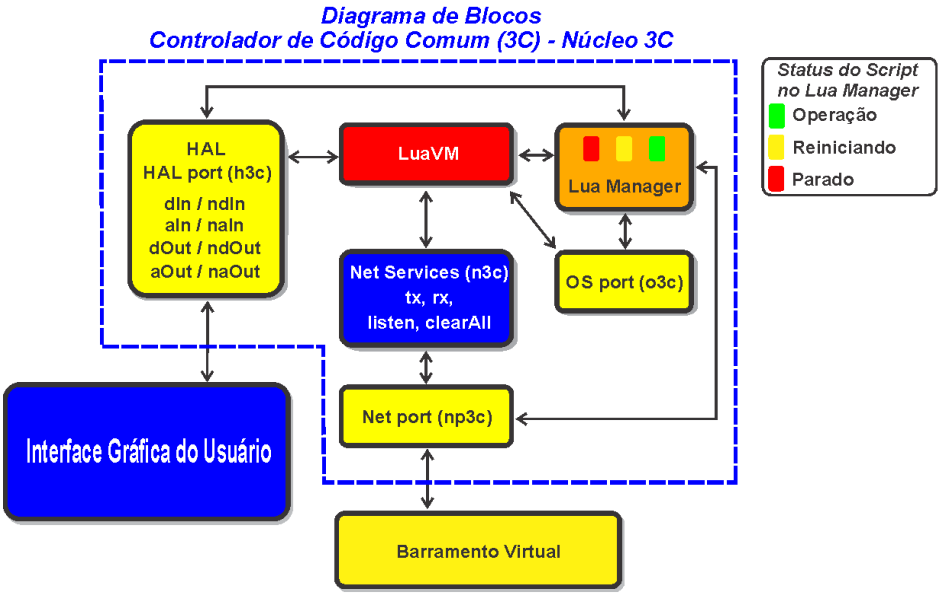

Figura 3. Diagrama do Núcleo 3C Proposto.

As funções de cada um desses blocos são:

LuaVM – Este bloco representa a máquina virtual Lua, que permite a execução de scripts Lua. Vale ressaltar que, Lua é uma linguagem de programação simples, eficiente, portável e leve, como apresentado em Ierusalimschy *et al*. (2001). Diversas características da linguagem Lua foram motivadas pela indústria e dicas de usuários. Por isso, atualmente ela é largamente utilizada na indústria de jogos no mundo todo. As principais características da Lua que motivaram a sua utilização nesta proposta foram:

- **Simplicidade:** Desde o início, esta linguagem foi projetada para ser simples. Isto implica uma sintaxe com um número reduzido de construtores e de modo que pudesse ser codificada em C;
- **Eficiência:** É uma linguagem que possui um compilador eficiente, rápido, de apenas uma passagem e que possui uma máquina virtual veloz.
- **Portabilidade:** Lua foi projetada para ser executada no maior número de plataformas possível. A intenção de seus desenvolvedores foi a de desenvolver o núcleo Lua para rodar em qualquer plataforma, sem a necessidade de sofrer modificações.
- **Leveza:** Até o momento, para o Linux, a versão completa do interpretador Lua, com todas as bibliotecas padrão, ocupa menos de 150 Kbytes; sendo que o núcleo ocupa menos de 100 Kbytes. Isto contribui para que Lua seja uma das linguagens mais rápidas no campo das linguagens de script, isto é, no ramo das linguagens interpretadas e dinamicamente tipadas.

Devido às características apresentadas, todo equipamento que possuir uma máquina virtual Lua, poderá executar um script Lua. Desse modo, se os equipamentos industriais implementassem uma LuaVM, poderiam executar o script Lua de outro dispositivo que fosse fabricado para executar funções semelhantes. Por exemplo: Um script Lua elaborado para um CLP de um determinado fabricante poderia ser executado por um CLP de outro fabricante que possuísse os mesmos recursos de hardware.

Portanto, nesta proposta de trabalho, a LuaVM proporciona portabilidade na programação e configuração dos dispositivos que serão implementados com o Controlador de Código Comum (3C) proposto.

Lua Manager - gerencia a LuaVM permitindo diversos serviços através de um protocolo, como apresentado a seguir:

- Descarregar um script na LuaVM;
- Iniciar a LuaVM;
- Parar a máquina LuaVM;
- Monitorar o I/O;
- Colocar o I/O em modo automático (controlado pelo script Lua) ou em modo manual para forçar condições de segurança.

HAL - controle de acesso ao HAL. Provê recursos para que o Lua Manager monitore o I/O, assuma o controle das saídas ou permita o acesso pela LuaVM. A princípio, por simplicidade, quatro variáveis e quatro funções serão implementadas (visíveis ao script):

- **h3c.naIn** – variável que contém o número de entradas analógicas disponíveis no hardware;
- **h3c.naOut** – variável que contém o número de saídas analógicas disponíveis no hardware;
- **h3c.ndIn** – variável que contém o número de entradas digitais disponíveis no hardware;
- **h3c.ndOut** – variável que contém o número de saídas digitais disponíveis no hardware;
- **h3c.aIn(i)** – lê uma entrada analógica. Sendo **i** [0...**h3c.naIn**] a entrada;
- **h3c.aOut(i, v)** – escreve em uma saída analógica. Sendo **i** [0...**h3c.naOut**] a saída e **v** o valor;
- **h3c.dIn(i)** – lê uma entrada digital. Sendo **i** [0...**h3c.ndIn**] a entrada;
- **h3c.dOut(i, v)** – escreve em uma saída digital. Sendo **i** [0...**h3c.ndOut**] a saída e **v** o valor;

Net Services (n3c) – provê a LuaVM serviços de acesso a rede. A princípio, por simplicidade, quatro funções serão implementadas:

- **n3c.tx(id, data)** – enviar dados para a rede (id=identificador, data=dados a serem enviados). Os dados enviados são "bufferizados" e transmitidos quando possível, segundo o protocolo sobre o qual foi implementado;
- **n3c.listen(id)** – registra um id para ser recebido pelo Net Services. Uma vez registrado um id o sistema fará *buffering* de dados recebidos com aquele id;
- **n3c.clearAll()** – limpa todos os *id's* registrados;
- **r,id,data=n3c.rx()** – recebe qualquer dado com um dos *id's* registrados. Esta função retorna três parâmetros, sendo **r** *true* se um novo dado foi recebido, **id** o id da informação recebida, **data** os dados recebidos.

Os port (o3c) – provê recursos do sistema operacional como temporização. Alguns recursos importantes de temporização fundamentais foram implementados:

- **o3c.dms(i)** – "pausa" a execução por **i** milissegundos (equivalentes a funções típicas como *delay* e *sleep*);
- **o3c.dus(i)** – "pausa" a execução por **i** microssegundos (equivalentes a funções típicas como *delay* e *sleep*);
- **o3c.ems()** – obtém o número de milissegundos do *freeruning* de hardware ou sistema operacional (equivalente a funções típicas como *GetTickCount*, *getTicks*, etc).

Net port (np3c) – camada que permite ao Net *Services* e ao LuaManager acesso à rede. É por meio desta camada que é possível controlar remotamente o LuaManager para realizar a configuração e atualização do script que a LuaVM irá executar.

Esta abordagem permite que uma rede com distribuição de mensagens identificadas seja implementada sobre diversos protocolos e ao mesmo tempo propicia facilidade de uso por parte do script.

Com o objetivo de elucidar o trabalho proposto, a seguir são apresentados alguns exemplos experimentais de aplicação do simulador 3C.

4. RESULTADOS EXPERIMENTAIS

A seguir serão apresentados dois experimentos realizados para verificar o correto funcionamento das partes que constituem o Simulador 3C. Nos diagramas de blocos somente as ligações que estão ativas entre Núcleos 3C, Barramento Virtual e GUI são exibidas. Vale ressaltar que a troca de informações ocorre entre comunicadores e entre comunicadores e GUI através do Barramento Virtual.

O GUI possui elementos gráficos que podem plotar gráficos de dados que trafegam pelo barramento ou originários das saídas analógicas e digitais (HAL port) de cada núcleo.

Entretanto, no estado atual de desenvolvimento da GUI, estes recursos não são facilmente utilizados e acoplados com o recurso de arrastar e soltar, como ocorre com a maioria dos simuladores comerciais.

4.1 – Experimento 01 – Teste do Núcleo 3C

A Fig. 4 mostra o diagrama de blocos do Experimento 01. Neste diagrama, o Núcleo 3C-01 está executando o *Script* A, conforme apresentado na Tabela 1 e o Núcleo 3C-02 está executando o *Script* B, mostrado na Tabela 2.

O *Script* A publica o valor 1 no barramento virtual, espera 1ms, publica o valor 0, espera 1ms e em seguida repete o ciclo novamente. O Núcleo 3C-02 recebe o dado enviado, e aplica-o a saída analógica 1. Esta saída analógica está sendo monitorada pela GUI que plota o gráfico correspondente a esta saída. A Fig. 5 mostra a interface gráfica atual do sistema.

Este teste foi realizado para verificar o funcionamento da GUI, HAL, Net Service, Net port, LuaManager, LuaVM e Barramento Virtual.

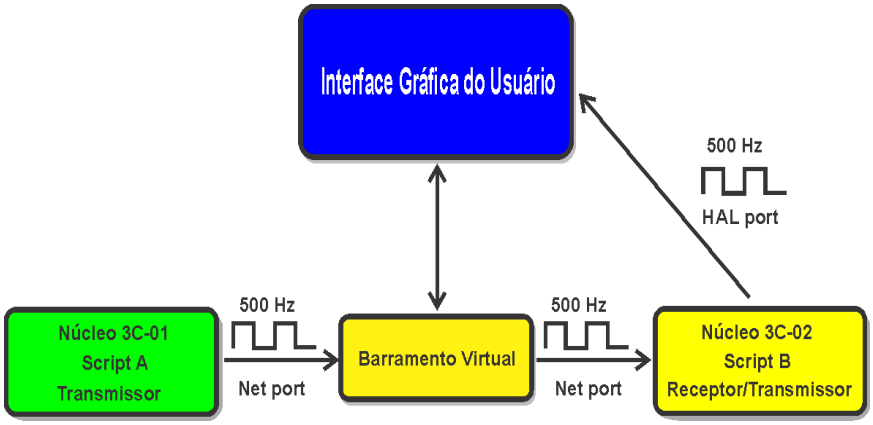

Figura 4. Diagrama de Blocos do Experimento 01.

Tabela 1. Script A - Gera onda quadrada com frequência de 500Hz na primeira saída analógica.

```
ID = 1908
 while true do
       nc3.tx(ID,1)   // Publica o valor 1 no Barramento Virtual com ID = 1908
       o3c.dms(1)     // Espera 1ms
       n3c.tx(ID,0)   // Publica o valor 0 no Barramento Virtual com ID = 1908
       o3c.dms(1)     // Espera 1ms
end
```

Tabela 2. Script B - Reproduz o sinal de uma entrada em uma saída.

```
ID = 98707;
n3c.listen(ID)  // Registra o ID=98707 para ser lido pelo stack de rede
while true do
      r,id,newValue=n3c.rx() // Lê o dado que estiver disponível (r informa se há algum)
      if r==true then
            h3c.aOut(1,newValue)  // Coloca na saída 1 o valor recebido
      end
end
```

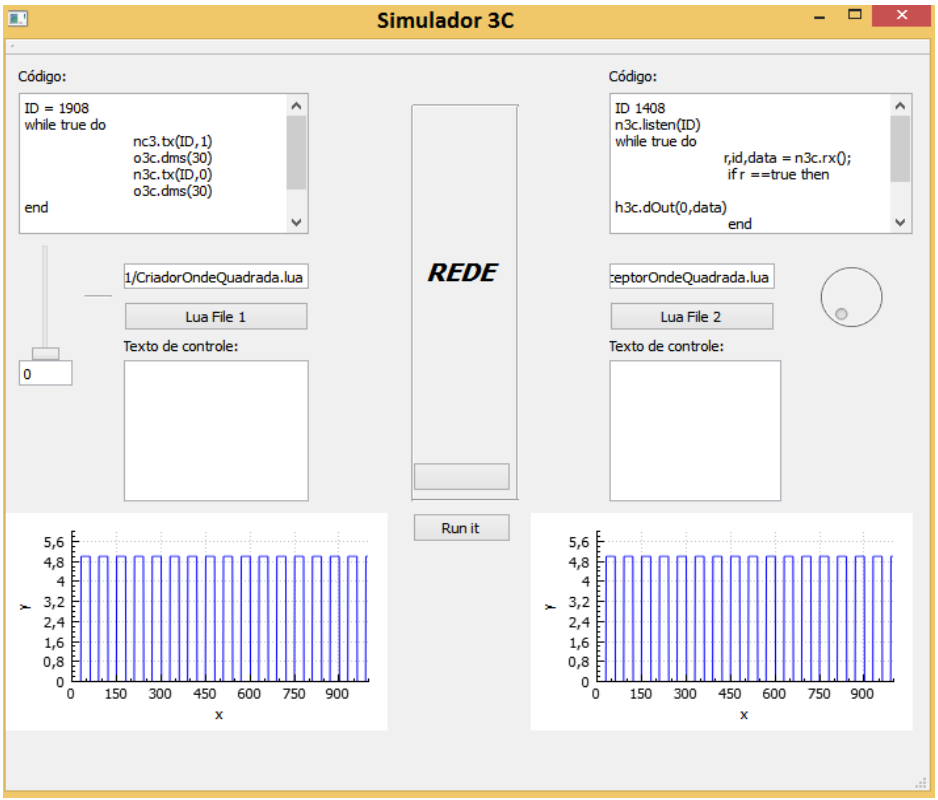

Figura 5. GUI utilizada no primeiro teste do Núcleo 3C.

4.2 – Experimento 02 – Teste do Núcleo 3C envolvendo cálculo

A Fig. 6 mostra o diagrama de blocos do Experimento 02. Neste diagrama, o Núcleo 3C-01 está executando o Script C, conforme apresentado na Tabela 3, e o Núcleo 3C-02 está executando o Script D, mostrado na Tabela 4. A Fig. 7 mostra a interface gráfica atual do sistema.

O *script* C lê o valor de uma variável T, aplica o valor de T na Equação 1 e, na sequência, publica o valor de Rt no Barramento Virtual. Quando o valor de Rt chega ao Núcleo 3C-02, ele é aplicado a Equação 2 e o valor resultante é atribuído a saída analógica 1.

As Equações 1 e 2 são semelhantes as equações de sensores de temperatura, porém, neste experimento, estas equações foram utilizadas simplesmente para verificar se os núcleos 3C estavam utilizando as bibliotecas matemáticas da LuaVM para realizar as operações de cálculo de forma correta.

Os gráficos obtidos no simulador foram comparados com gráficos gerados pelo software Matlab®, conforme a Fig. 8.

$$R_T(T) = R_0 * (1 + A * T + BT^2) \quad (1)$$

$$T = \frac{(R_T - 100)}{0.389} \quad (2)$$

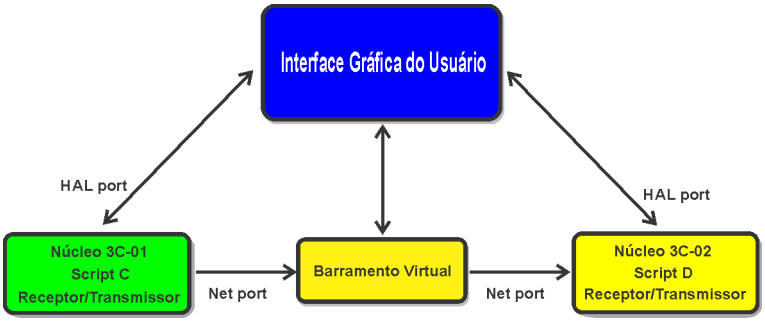

Figura 6. Diagrama de Blocos do Experimento 02.

Tabela 3. Script C – Calcula o valor de Rt e publica no Barramento.

```
ID = 94376
white true do
    T = h3c.aIn(1) // Lê o valor da temperatura na entrada analógica 1
    A = 0.0039083
    B = -0.0000005775
    Rt = 100*(1 + A*T + B*(T^2) )
    n3c.tx(ID,Rt); // Publica o valor de Rt no Barramento Virtual com ID = 94376
 end
```

.

Tabela 4. Script D – Calcula a temperatura em função da resistência, escrevendo-a na porta A1

```
ID = 98707;
n3c.listen(ID)  // Registra o ID=98707 para ser lido pelo stack de rede
while true do
    r,id,RT=n3c.rx()  // Lê o dado que estiver disponível (r informa se há algum)
    if r==true then
        T =(Rt-100) / 0.389
        h3c.aOut(1, T)  // Publica o valor da temperatura na saída analógica 1
    end
end
```

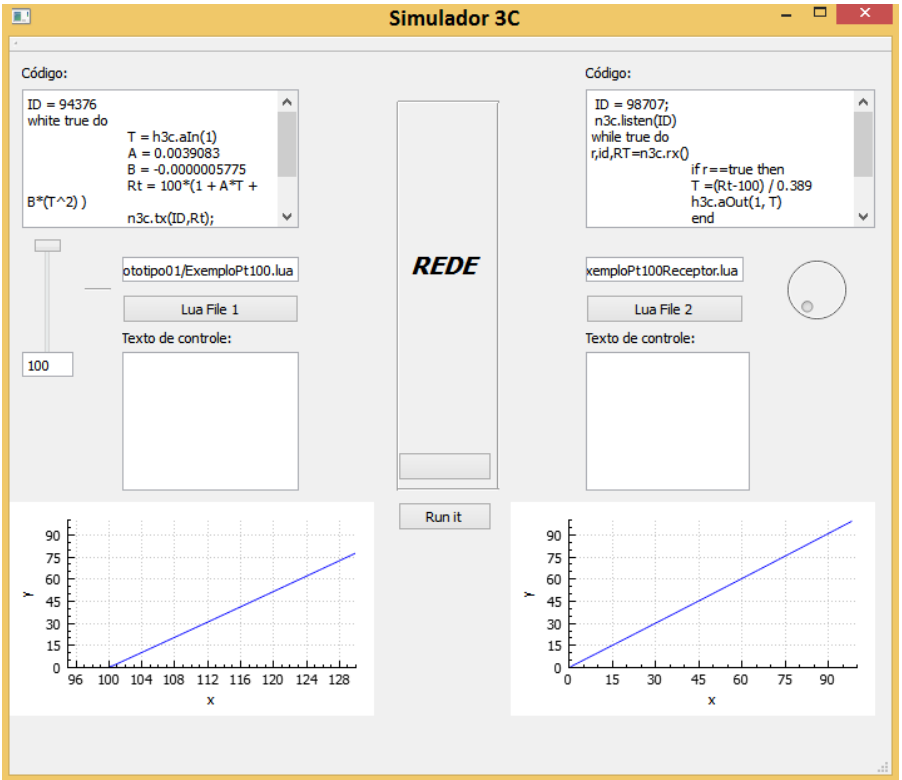

Figura 7. GUI utilizada no teste de Cálculo.

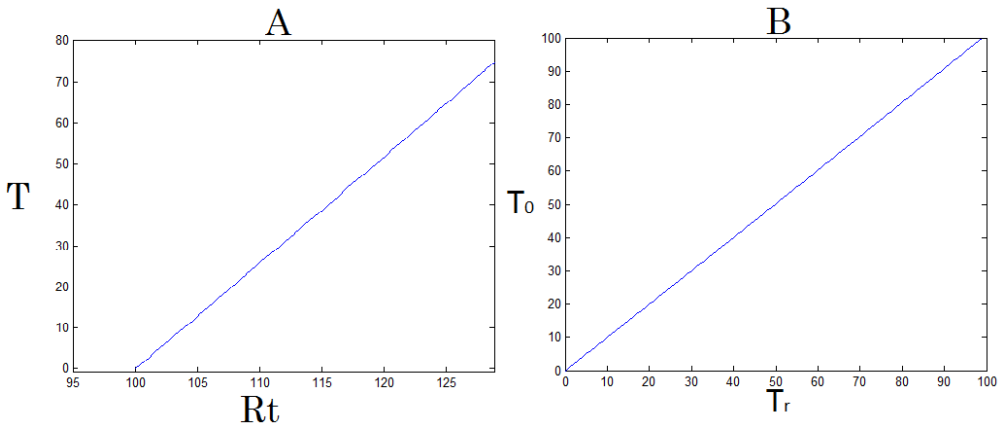

Figura 8: A) Gráfico MatLab Eq. 1, B) Gráfico MatLab Eq. 2

A Fig. 8A exibe o gráfico obtido com o software MatLab referente a Eq. (1) e a Fig. 8B o resultado correspondente a Eq. (2).

5. AGRADECIMENTOS

Os autores gostariam de agradecer ao CNPq, à CAPES e a FAPEMIG pelo apoio e suporte financeiro.

6. CONCLUSÃO

A proposta mostrou-se flexível no sentido de possibilitar a implementação de scripts Lua. Neste sentido, considerando-se que os scripts podem ser elaborados para realizarem tarefas diversas, o Núcleo 3C pode ser adotado como ferramenta flexível na confecção de equipamentos que podem ser utilizados na solução de problemas, em diversas áreas do conhecimento.

Embora os experimentos aqui apresentados tenham focado em scripts mais simples, com o propósito de validar a ideia, funções matemáticas mais avançadas também estão disponíveis, permitindo a implementação de algoritmos usuais como PID, Timers, Contadores, etc.

Além disso, o barramento virtual permite a comunicação entre os Núcleos 3C e abre a possibilidade da implementação de protocolos de redes e os Núcleos 3C permitem a simulação de dispositivos que trocam informações entre si.

Por fim, os resultados obtidos corroboram no sentido de proporcionar uma ferramenta computacional capaz de implementar vários dispositivos industriais virtuais, por meio do correto modelamento dos mesmos, fomentando ainda a contribuição de empresas e entidades educacionais no desenvolvimento de novas estratégias de controle, por meio de algoritmos dedicados de fácil portabilidade.

7. REFERÊNCIAS

Almeida, Gevaldo L. de et al., 2004, “Development of a Simulator for Tomographic Images Generated by Radiation Transmission.”, Brazilian Journal Of Physics. São Paulo, pp. 767-772.

Dias, P. C. M., 2010 “Desenho de Simulador de Transformação Operacional na LeanKed Academy”, Dissertação de Mestrado - Curso de Mestrado Integrado em Engenharia Industrial e Gestão, Faculdade de Engenharia da Universidade do Porto.

Foundation, F., 2008, “FOUNDATION™ Specification Function Block Application Process”, Fieldbus Foundation - Part 4, Document: FF-893, Revision: FS 1.2.

Ierusalimschy, R., de Figueiredo, L. H., and Celes, W., 2001, “The evolution of an extension language: A history of Lua”, In Proceedings of V Brazilian Symposium on Programming Languages, pp. B 14-28.

Monteiro, P. O. R., 1991, “Concepção e implementação de um Sistema Simulador de Ambientes Industriais” Dissertação de Mestrado - Curso de Informática Industrial, Faculdade de Engenharia da Universidade do Porto, Porto.

Okada, D. M. *et al.*, 2010, “Surgical simulator for temporal bone dissection training: Simulador cirúrgico para treinamento de dissecção do osso temporal”, Brazilian Journal Of Otorhinolaryngology - Recife, pp. 575-578.

Panton, R. P., Torrisi, N. and Brandão, D., 2007, "An open and non-proprietary device description for fieldbus devices for public IP networks", 5th IEEE International Conference on Industrial Informatics", pp. 189-194.
Pereira, S. P. *et al*., 2012, "Mining simulation for room and pillar coal operation. The Journal Of The Southern African", Institute Of Mining And Metallurgy. South Africa, pp. 473-476.
Reis, L. P., 2002, "Um Agente para Utilizar o Simulador Ciber-Rato no Ensino da Inteligência Artificial e Robótica Inteligente", Revista do Detua Aveiro, Vol. 3, No. 7, pp. 1-5.
Shahraeini, M., Javidi, M. H. and Ghazizadeh, M. S., 2011, "Comparison Between Communication Infrastructures of Centralized and Decentralized Wide Area Measurement Systems", IEEE Transactions On Smart Grid, Vol. 2, No. 1, pp. 206-211.

RESPONSABILIDADE AUTORAL

Os autores são os únicos responsáveis pelo conteúdo do material impresso incluídos no seu trabalho.

INDUSTRIAL DEVICE SIMULATOR THAT USES PORTABLE CODE

Guilherme Resende Ferreira, guilherme.ebio@hotmail.com[1]
Marcio José da Cunha, cunhamjg@gmail.com[1]
Gilson Fonseca Peres Filho, gilsonfonseca@yahoo.com.br[1]
Aniel Silva de Morais, aniel@eletrica.ufu.br[1]
Fábio Vincenzi Romualdo da Silva, fabiovince@gmail.com[1]

[1] Federal University of Uberlândia – UFU, Electrical Engeneering Faculty- FEELT, Control and Automation Research Group – NPCA, Av. João Naves de Ávila, 2121 Bloco 3N,Uberlândia, MG, Brasil

Abstract*. This work proposes the implementation of an open code simulator that uses a set of functions executed by a Lua virtual machine. These set of functions are being developing by a parallel work that aims reduce complexity, cost and the implementation of custom controls in certain industrial equipment. Thus, the proposed simulator will allow academics and manufacturers from the branch of automation and control understand and improve new equipment, test new controls, develop and simulate new protocols, perform training, among other applications.*

Keywords*: open code simulator, set of functions, Lua virtual machine, custom controls, training*

RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.